.

Representation Learning @ Scale

Manzil Zaheer

July 2018
CMU-ML-18-108

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Barnabás Póczos, Co-Chair
Ruslan Salakhutdinov, Co-Chair
Alexander J Smola
Andrew McCallum
Geoffrey J Gordon

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy (PhD).

This research was supported by: the National Science Foundation award numbers CCF1316363 and IIS1409802; the Department of the Interior award number D17AP00001; the Air Force Research Laboratory award number FA87501720130; and a grant from the Intel Science & Technology Center for Cloud Computing.

To my parents

Abstract

Machine learning techniques are reaching or exceeding human level performances in tasks involving simple data like image classification, translation, and text-to-speech. The success of these machine learning algorithms is attributed to highly versatile representations learnt from data using deep networks or intricately designed Bayesian models. Representation learning has also provided hints in neuroscience, e.g. understanding how humans might categorize objects. Despite these instances of success, progress has been limited to simple data-types so far.

Most real-world data come in all shapes and sizes, not just as images or text, but also as point clouds, sets, graphs, compressed or even heterogeneous combinations thereof. In this thesis, we develop representation learning algorithms for such complex data types by leveraging structure and establishing new mathematical properties. Representations learned in this fashion were applied on diverse domains and found to be competitive with task-specific state-of-the-art methods.

Having representations is not enough in various applications - its interpretability is as crucial as its accuracy. Deep models often yield better accuracy but require a large number of parameters, often in contrast to the simplicity of the underlying data, rendering it uninterpretable. This is highly undesirable in tasks like user modeling. In this thesis, we show that by leveraging structure by incorporating domain knowledge in the form of Bayesian components on top of deep models, we learn sparser representations with discrete components that are more amenable to human interpretation. Our experimental evaluations show that the proposed techniques compare favorably with several state-of-the-art baselines.

Finally, inferring interpretable representations from large-scale data is desirable, but often hindered by a mismatch between computational resources and statistical models. In this thesis, we bridge this gap by again leveraging structure, albeit of a different kind. Our solutions are based on a combination of modern computational techniques/data structures on one side and modified statistical inference algorithms on the other which exploit topological properties of the training objective. This introduces new ways to parallelize, reduce look-ups, handle variable state space size, and escape saddle points. On latent variable models, like latent Dirichlet allocation (LDA), we observe significant gains in performance.

To summarize, in this thesis, we advance the three major aspects of representation learning — diversity: being able to handle different types of data, interpretability: being accessible to and understandable by humans, and scalability: being able to process massive datasets in a reasonable time and budget — all by leveraging some form of structure.

Acknowledgments

Five years of Ph.D. studies at Carnegie Mellon University (CMU) were a fantastic experience for me, to say the least. I was extremely fortunate to be advised by a number of professors at CMU, starting from Xin Li to Alexander Smola to Ruslan Salakhutdinov and Barnabás Póczos. I learned tremendously from each of them. Alex has been an exceptional mentor. His sheer brilliance and multifaceted expertise of machine learning and science (and life in general) have been an excellent source of inspiration for me. Barnabás is a brilliant researcher and an excellent mentor who taught me that patience and persistence is the key to successful research. Russ is a well-known researcher with an extremely friendly and calm demeanor which spurred me to ask many questions, even silly ones, without hesitation. Also, Russ is generously spreading my research (and slide template!) through his talks.

I am grateful that all of my advisors gave me a lot of freedom and leeway in choosing my research problems and how to pursue them. Although perhaps sometimes I was a bit too stubborn and should have listened more carefully to my advisors. My thesis committee gave me also excellent feedback after my thesis proposal. I am very proud of my committee.

My internships at Amazon, Google, Intel, and Oracle, introduced me to many amazing people, which had a significant impact on my research and in learning to become a researcher. Under Jean-Baptiste Tristan's mentorship in Oracle, I learned a lot about how to be a proper researcher. He asked the right questions and pushed me to concentrate on the essential problems.

I have been extremely fortunate to collaborate with a wonderful set of colleagues: Amr Ahmed, Francis Bach, Rajarshi Das, Shehzaad Dhuliawala, Bhuwan Dhingra, Avinava Dubey, Chris Dyer, Chenjie Gu, Abhishek Khetan, Satwik Kottur, Jay-Yoon Lee, Golan Levin, Chun-Liang Li, Andrew McCallum, Junier Oliva, Siamak Ravanbakhsh, Sashank Reddi, Shashank Singh, Suvrit Sra, Hsiao-Yu Fish Tung, Fa Wang, Michael Wick, Chao-Yuan Wu, Eric Xing, Yang Zhang, and Xun Zheng. Many chapters of this thesis are the culmination of these collaborations. Without their expertise in their respective field, I would have been totally lost.

I am very grateful and hugely indebted to all my teachers at CMU – Christos Faloutsos, Geoff Gordon, Venkat Guruswami, Jing Lei, Gary Miller, Jiashun Jin, Ryan Tibshirani, Larry Wasserman, – for sharing their valuable insights and research expertise with me. Also, thanks to the department staff members, especially Diane Stidle, Sandy Winkler, and Amy Protos, for their invaluable support throughout my time at CMU.

I was lucky enough to have a wonderful set of friends at CMU. I would like to thank my officemates Alnur Ali, Benjamin Cowley, Jing Xiang, and department colleagues Anthony Platanios, Leqi Liu, Yifei Ma, Willie Neiswanger, Ritesh Noothigattu, Mrinmaya Sachan, Otilia Stretcu for countless exciting discussions, helping me to take gradient steps for improvement. Also, I would like to thank Ameya, Anit, Avinava, Chun-Liang, Fish, Guru, Jay-Yoon, Jing, Jisu, Rajarshi, and Sashank for being such awesome friends. Without their company, my time in Pittsburgh would not have been so much fun.

Also, I would like to acknowledge the service of shuttle and escort drivers, especially Chuck and Al, who allowed me to work on my schedule – the times I am productive most without worrying about safety while commuting between university and home.

Finally and most importantly, I am deeply indebted to my loving mom Srinvanti Pal Zaheer and dad Hasan Zaheer. I am lucky to have a such a great family who always supported me.

Contents

List of Figures

List of Tables

1

Introduction

The digitally connected world of today involves potentially valuable data growing at a tremendous pace like text, images, and speech. The value generation lies is in transforming unstructured data into a usable information or representation, which is well organized, searchable and understandable. This has been a major enabler for personalized recommendation (of articles to read, videos to watch, locations to visit, restaurants to dine at), fraud detection, medical image analysis, face recognition, etc. Machine learning and intelligent systems are used to carry out the task listed above and have started to become an indispensable part of our modern society.

The current machine learning paradigms mostly focus on individual objects that have simple representations. For example in the tasks of classification, regression, and clustering, an object of interest is often described by a "feature vector", and abstracted as a point in certain metric space. The objective of learning is to estimate functions that map these points to target variables such as the class labels or cluster memberships, with the goal of achieving both empirical accuracies on the training data as well as the generalization power on unseen data. This "one point per object" abstraction has led to very concise representations, elegant mathematical theories, and very successful algorithms. The performance of machine learning on above mentioned well defined tasks with simple data have reached or exceeded human levels like image classification (He et al., 2016a) or speech recognition (W. Xiong et al., 2016). The success of the methods heavily depends on the choice of data representation (or features) used (Y. Bengio, Courville, and Vincent, 2013), for which deep learning has been a boon.

However, the data comes in all shapes and sizes, i.e. ranging from text to images & videos to point clouds to graphs to detailed user activity logs and shopping histories in sizes ranging from a few megabytes to hundreds of terabytes and combinations thereof. For instance, in the field of language modeling and text processing, an article can be considered as a group of paragraph or sections. In computer vision, the recently gaining 3D imaging consists of a group of coordinates. In recommendation systems, a user is mainly described by the heterogeneous data of products bought, videos watched, articles read. We call these kinds of data that are organized by groups or heterogeneous combinations as the complex data.

This thesis is centered around representation learning, i.e., learning representations of the above mentioned complex data that make it easier to extract useful information when building classifiers or other predictors for human and/or machine consumption. Traditionally, machine learning has focuses on individual objects, each of which is described by a feature vector and studied as a point in some metric space. When approaching more complex data, researchers often reduce the groups into vectors to which traditional methods can be applied. We, on the other hand, will try to develop machine learning methods that learn from them directly. In order to build such methods, one needs to leverage structure present in the data. Furthermore, by utilization of structure one can build faster inference techniques as well more interepretable representation. The exploitation of structure is indeed the crux of our thesis:

> **Thesis statement:** Leveraging structure can not only lead to better representation learning, but also results in speeding up inference and more interpretable models.

1.1 ASPECTS OF REPRESENTATION LEARNING

There are two main categories for learning representations. First is using Bayesian models with a parsimonious set of latent random variables that describe distribution over the observed data. The representation is the posterior distribution of the underlying explanatory factors for the observed input. A good representation is also one that is useful as input to a supervised predictor. Second way of learning representations is utilizing deep learning methods, those that are formed by the composition of multiple non-linear transformations, with the goal of yielding more abstract – and ultimately more useful representations. For either categories following are the open challenges:

diversity: Data come in all shapes and sizes: not just as images or text, but also as point clouds, sets, graphs, compressed, or even heterogeneous mixture of these data types. In this thesis, we want to develop representation learning algorithms for such unconventional data types. Our methodology relies on leveraging the structure present in the data which involve establishing new mathematical properties. Representations learned in this fashion were applied on diverse domains and found to be competitive with task specific state-of-the-art methods.

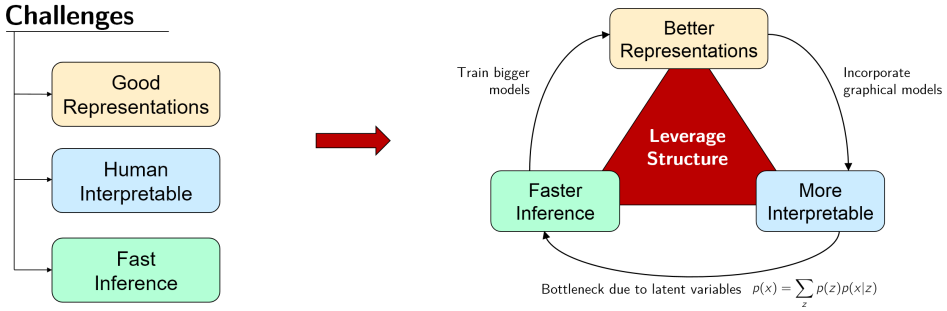
interpretability: Once we have the representations, in various applications its interpretability crucial. Deep models often yield better accuracy, require a large number of parameters, often notwithstanding the simplicity of the underlying data, rendering it uninterpretable which is highly undesirable in tasks like user modeling. On the other hand, Bayesian models produce sparse discrete representations, easily amenable to human interpretation. In this thesis, we want to explore that are capable of learning mixed representations retaining best of both the worlds. Our experimental evaluations show that the proposed techniques compare favorably with several state-of-the-art baselines.

scalability: Finally, one would want such interpretable representations to be inferred from large-scale data, however, often there is a mismatch between our computational resources and the statistical models. In this thesis, we want to bridge this gap by solutions based on a combination of modern computational techniques/data structures on one side and modified statistical inference algorithms on the other. We introduce new ways to parallelize, reduce look-ups, handle variable state space size, and escape saddle points. On latent variable models, like latent Dirichlet allocation (LDA), we find significant gains.

1.2 OUR APPROACH

The versatility of representation learning makes it highly desirable to apply them to variety of data, make the representations learned interpretable, and scale them to large datasets of many terabytes of observations and billions of objects. We would like to take a more holistic approach considering statistical and computational aspects simultaneously. This thesis is a step in the direction of addressing these new challenges in modern ML applications. We begin by tackling the issues case by case to begin with. Eventually we wish to extend these to generality.

- **Find mathematical properties to be leveraged:** Exploiting structure or finding mathematical properties present in data has found useful in improving performance of many machine learning tasks like optimization (Hsieh et al., 2015) or classification (Goel, Knoblock, and Lerman, 2012). We carry forward the idea to case of representation learning. In particular, we characterize function on sets to design an universal deep network architecture, or how to efficiently handle Stirling numbers which occur in many posterior distributions,



(a) Challenges (b) Contributions

Figure 1.1: Important challenges in machine learning and contributions of this thesis.

or how using the additivity of sufficient statistics for exponential family can lead to a lock free and parallel inference algorithm. We show methods leveraging these structure outperform simply throwing deep networks at the problem.

- **Find correct data structures for the task:** Data structures have been designed to solve computational problems efficiently by organizing the data and making it cheap to locate and retrieve data. In this thesis we want to identify inference algorithms having specific access patterns, invariances, where efficient data structures can be adopted. We also consider modifying the inference algorithm so as to enable us in leveraging some highly efficient data structures. For example, in case of clustering only a small part of the space needs to examined carefully, thus space partitioning data structures like cover trees can come handy but requires novel modifications to the inference algorithm. Similarly to reduce communication bandwidth, approximate counters can be employed in Gibbs sampling instead of keeping track of exact counts. But approximate counters do not support decrements, thus their adoption again calls for modifications in the inference strategies. We want to find such solutions.
- **Modify inference goal/strategy:** Often the inference goal can be much stronger than needed, *e.g.* one always does not need the full posterior but many a times just a point estimate would suffice. As we will see the altered goal allows much better utilization of modern computational resource, which are heavily distributed and each node itself being heavily multithreaded, *e.g.* servers can have in excess of 64 threads and GPUs over 2048 of them, but having finite memory bandwidth. One can exploit it for tasks like clustering and topic modeling. Furthermore, the alterations can open paths to use of efficient data structures as described above.

1.3 OVERVIEW OF THESIS & OUR RESULTS

This thesis presents a selection of my work on representation learning addressing the three important issues raised above. The content of the thesis is divided into three parts: Part I, consisting of Chapters 2 to 4, describes my work on handling complex data; Part II, organized as the next two chapters, describes my work on making the representation more interpretable while being expressive by mixing deep networks with graphical model incorporating domain knowledge; and finally Part III, composed of Chapters 7 to 11, describes my work on speeding up the inference procedure. We provide a brief overview of various chapters here and defer the exact details to the respective chapters.

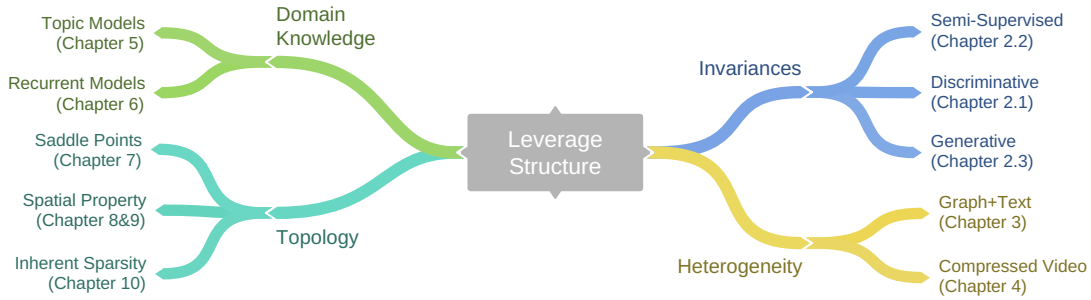


Figure 1.2: Caption

Chapter 2: Set Data

We study the problem of designing models for machine learning tasks defined on *sets*. In contrast to traditional approach of operating on fixed dimensional vectors, we consider objective functions defined on sets that are invariant to permutations. Such problems are widespread, ranging from estimation of population statistics (Poczos, Rinaldo, et al., 2013), to anomaly detection in piezometer data of embankment dams (Jung et al., 2015), to cosmology (Ntampaka et al., 2016; Ravanbakhsh, Junier Oliva, et al., 2016). Our main theorem characterizes the permutation invariant functions and provides a family of functions to which any permutation invariant objective function must belong. This family of functions has a special structure which enables us to design a deep network architecture that can operate on sets and which can be deployed on a variety of scenarios including both unsupervised and supervised learning tasks. We also derive the necessary and sufficient conditions for permutation equivariance in deep models. We demonstrate the applicability of our method on population statistic estimation, point cloud classification, set expansion, and outlier detection.

Chapter 3: Compressed Data

Training robust deep video representations has proven to be much more challenging than learning deep image representations and consequently hampered tasks like video action recognition. This is in part due to the enormous size of raw video streams, the associated amount of computation required, and the high temporal redundancy. The 'true' and interesting signal is often drowned in too much irrelevant data. Motivated by the fact that the superfluous information can be reduced by up to two orders of magnitude with video compression techniques (like H.264, HEVC, etc.), in this work, we propose to train a deep network directly on the compressed video, devoid of redundancy, rather than the traditional highly redundant RGB stream. This representation has a higher information density and we found the training to be easier. In addition, the signals in a compressed video provide free, albeit noisy, motion information. We propose novel techniques to use them effectively. Our approach is about 4.6 times faster than a state-of-the-art 3D-CNN model, 2.7 times faster than a ResNet-152, and very easy to implement. On the task of action recognition, our approach outperforms all the other methods on the UCF-101, HMDB-51, and Charades dataset.

Chapter 4: Heterogeneous Data

Existing question answering methods infer answers either from a knowledge base or from raw text. While knowledge base (KB) methods are good at answering compositional questions,

their performance is often affected by the incompleteness of the KB. Au contraire, web text contains millions of facts that are absent in the KB, however in an unstructured form. *Universal schema* can support reasoning on the union of both structured KBs and unstructured text by aligning them in a common embedded space. In this paper we extend universal schema to natural language question answering, employing *memory networks* to attend to the large body of facts in the combination of text and KB. Our models can be trained in an end-to-end fashion on question-answer pairs. Evaluation results on SPADES fill-in-the-blank question answering dataset show that exploiting universal schema for question answering is better than using either a KB or text alone. This model also outperforms the current state-of-the-art by 8.5 F_1 points. In this paper we look at a more practical setting, namely QA over a combination of a KB and entity-linked text, which is appropriate when an *incomplete* KB is available with a large text corpus. Building on recent advances in graph representation learning we propose a novel model GRAFT-Net for extracting answers from a question-specific subgraph consisting of both KB entities and text documents. We construct a suite of benchmark tasks for this problem, varying the difficulty of questions, the amount of training data, and KB completeness. We show that GRAFT-Net is competitive with the state-of-the-art when tested using either KBs or text only, and vastly outperforms existing methods in the combined setting.

Chapter 5: Domain Knowledge in Static Models

We study the problem of incorporating domain knowledge in data to enhance predictive power and interepretability. Deep learning methods have been shown to be extremely successful in a variety of well defined problems due their high expressive power across computer vision (CV), speech recognition as well as natural language processing (NLP). However, their internal representations are distributed codes and lack interpretability or manipulations. These dense unstructured representations are in direct conflict with many real-world problems which have inherent structures: like invariances, dependencies among random variables, etc. Graphical models have been typically employed to represent and exploit such invariances and dependencies. In this chapter, we will explore ways to incorporate domain knowledge so as to take advantage of high predictive power of deep learning but also be more interpretable and sample efficient by enforcing invariances and dependencies. We will also propose efficient inference strategies to tackle new difficulties encountering in these combined models. Here we look at diverse applications ranging from NLP and CV tasks to to even less relevant fields like physics or chemistry or user action modeling.

Chapter 6: Interpretable Recurrent Models

Recurrent neural networks, such as long-short term memory (LSTM) networks, are powerful tools for modeling sequential data like user browsing history (Korpusik, Sakaki, and F. C. Y.-Y. Chen, 2016; Tan, X. Xu, and Y. Liu, 2016) or natural language text (Mikolov, Karafiát, et al., 2010). However, to generalize across different user types, LSTMs require a large number of parameters, notwithstanding the simplicity of the underlying dynamics, rendering it uninterpretable, which is highly undesirable in user modeling. The increase in complexity and parameters arises due to a large action space in which many of the actions have similar intent or topic. In this paper, we introduce Latent LSTM Allocation (LLA) for user modeling combining hierarchical Bayesian models with LSTMs. In LLA, each user is modeled as a sequence of actions, and the model jointly groups actions into topics and learns the temporal dynamics over the topic sequence, instead of action space directly. This leads to a model that is highly interpretable, concise, and can capture intricate dynamics. We present an efficient Stochastic EM inference algorithm for our model that scales to millions of users/documents. Our experimen-

tal evaluations show that the proposed model compares favorably with several state-of-the-art baselines.

Chapter 7: Parallelization

We propose an embarrassingly parallel, memory efficient inference algorithm for latent variable models in which the complete data likelihood is in the exponential family. The algorithm is a stochastic cellular automaton and converges to a valid *maximum a posteriori* fixed point. Applied to latent Dirichlet allocation we find that our algorithm is over an order of magnitude faster than the fastest current approaches. A simple C++/MPI implementation on a 20-node Amazon EC2 cluster samples at more than **1 billion tokens per second**. We process 3 billion documents and achieve predictive power competitive with collapsed Gibbs sampling and variational inference.

Chapter 8: Reducing Lookups

Hierarchical Bayesian models often capture distributions over a very large number of distinct atoms. The need for these models arises when organizing huge amount of unsupervised data, for instance, features extracted using deep convnets that can be exploited to organize abundant unlabeled images. Inference for hierarchical Bayesian models in such cases can be rather non-trivial, leading to approximate approaches. In this work, we propose *Canopy*, a sampler based on Cover Trees that is exact, has guaranteed runtime logarithmic in the number of atoms, and is provably polynomial in the inherent dimensionality of the underlying parameter space. In other words, the algorithm is as fast as search over a hierarchical data structure. We provide theory for Canopy and demonstrate its effectiveness on both synthetic and real datasets, consisting of over 100 million images.

Chapter 9: Handling Variable State-Space Size

Latent variable models, such as topic models, have accumulated a considerable amount of interest from academia and industry for their versatility in a wide range of applications. Bayesian nonparametrics has used frequently in enabling latent variable models to adapt in accordance to the size of data. However, it is non-trivial to scale the inference procedures to deal with real life problems in industry consisting of massive datasets (1 TB+) in a reasonable time with reasonable resources. In this paper, we wish to suggest a scalable inference procedure for Bayesian Nonparametrics based on a combination of smart data structures for handling probability distributions on one side and modified statistical inference algorithms on the other. In particular, we take up the task of topic modelling with Hierarchical Dirichlet Process to demonstrate effectiveness of the proposed inference procedure. We provide experimental results for the proposed method on big datasets like PubMed and Wikipedia obtaining 10x speed-up.

Chapter 10: Escaping Saddle Points

A central challenge to using first-order methods for optimizing nonconvex problems is the presence of saddle points. First-order methods often get stuck at saddle points, greatly deteriorating their performance. Typically, to escape from saddles one has to use second- order methods. However, most works on second-order methods rely extensively on expensive Hessian-based computations, making them impractical in large-scale settings. To tackle this challenge, we introduce a generic framework that minimizes Hessian-based computations while at the same time provably converging to second- order critical points. Our framework carefully alternates between a first-order and a second-order subroutine, using the latter only close to saddle points,

and yields convergence results competitive to the state-of-the-art. Empirical results suggest that our strategy also enjoys a good practical performance.

1.4 BIBLIOGRAPHIC NOTES

The research presented in this thesis is based on joint work with many co-authors, as described below.

In Part I, Chapter 2 is based on joint work with Satwik Kottur, Siamak Ravanbhaksh, Chun-Liang Li, Yang Zhang, Barnabás Póczos, Ruslan Salakhutdinov, and Alex Smola (Zaheer, Kottur, Ravanbakhsh, et al., 2017; Zaheer, C.-L. Li, et al., 2018). Chapter 3 is joint work with Chao-Yuan Wu, Philipp Krahenbuhl, and Alex Smola (C.-Y. Wu et al., 2018). ´ Chapter 4 is joint work with Rajarshi Das, Bhuwan Dhingra, Haitian Sun, Siva Reddy, William Cohen, Ruslan Salakhutdinov, and Andrew McCallum (Haitian Sun et al., 2018; Zaheer, Das, Reddy, et al., 2017).

In Part II, Chapter 5 is based on joint work with Abhishek Khetan, Rajarshi Das, Guru Ganesh, Siamak Ravanbhaksh, Chris Dyer, Golan Levin, Venkatasubramanian Viswanathan, Barnabás Póczos, and Alex Smola (Khetan et al., 2018; Zaheer, Das, and Dyer, 2015; Zaheer, Guruganesh, et al., 2018). Chapter 6 is joint work with Xun Zheng, Amr Ahmed and Alex Smola (Zaheer, Amr Ahmed, and A. J. Smola, 2017; Zaheer, Amr Ahmed, Yuan Wang, et al., 2018; X. Zheng et al., 2017).

In Part III, Chapter 7 is based on joint work with Jean-Baptiste Tristan, Michael Wick, Alex Smola, and Guy Steele (Zaheer, Wick, et al., 2015). Chapter 8 is joint work with Satwik Kottur, Amr Ahmed, and Alex Smola (Zaheer, Kottur, Amr Ahmed, et al., 2017). Chapter 9 is joint work with Alex Smola (Zaheer and A. J. Smola, 2018). Chapter 10 is joint work with Sashank Reddi, Suvrit Sra, Francis Bach, Ruslan Salakhutdinov, Barnabás Póczos and Alex Smola (Zaheer, S. Reddi, et al., 2018).

1.4.1 *Excluded Research*

Besides representation learning, I have also worked on a number of other topics such as random graphs, small sample estimation, spectral methods, reinforcement learning, which are not part of my thesis. Here, I briefly describe my contributions in these areas.

Density Estimation (J. B. Oliva et al., 2018; Shashank Singh et al., 2018) The fundamental task of general density estimation $p(x)$ has been of keen interest to machine learning. In this line of work, we attempt to systematically characterize and categorize methods for density estimation. As a consequence of this categorization, we noticed that algorithmically, sampling, like MCMC and GAN, is a very distinct problem from density estimation, but we show that, given unlimited computational resources, the problems of density estimation and sampling are equivalent in a minimax statistical sense (Shashank Singh et al., 2018). Also, leveraging this characterization, we propose a novel density estimator called Transformative Autoregressive Networks (J. B. Oliva et al., 2018). Through a comprehensive study over both real world and synthetic data, we show that jointly leveraging transformations of variables and autoregressive conditional models, results in a considerable improvement in performance.

Understand Text Classifications (Sachan, Zaheer, and R. Salakhutdinov, 2018, 2019) In this line of work, we study deep networks for the task of text classification, which is one of the most widely studied tasks in natural language processing. Motivated by the principle of compositionality, large multilayer neural network models have been employed

for this task in an attempt to effectively utilize the constituent expressions. Several prior works have suggested that such complicated models with complex pretraining schemes using unsupervised methods such as language modeling are necessary to achieve a high classification accuracy. We explore whether these models actually learn to compose the meaning of the sentences or still just focus on some keywords or lexicons for classifying the document. To test our hypothesis, we carefully construct datasets and study various text classifiers to observe that there is a big performance drop, suggesting our hypothesis might be true. Finally, we develop a training strategy that allows even a simple BiLSTM model, when trained with cross-entropy loss, to achieve competitive results compared with more complex approaches. Furthermore, in addition to cross-entropy loss, by using a combination of entropy minimization, adversarial, and virtual adversarial losses for both labeled and unlabeled data, we report state-of-the-art results for text classification task on several benchmark datasets.

Small Sample Estimation (Gu, Zaheer, and X. Li, 2014; F. Wang et al., 2015; Zaheer, X. Li, and Gu, 2014; Zaheer, F. Wang, et al., 2015) Today's VLSI circuits are designed via a multi-stage flow. At each stage, simulation or measurement data are collected to validate the circuit design, before moving to the next stage. These measurements are extremely time consuming and expensive, hence only a few samples at each stage can be afforded. Traditional performance modeling techniques rely on the data collected at a single stage only and completely ignore the data that are generated at other stages. The key idea proposed was to reuse the early-stage data when fitting a late-stage performance model. As such, the performance modeling cost can be substantially reduced. Mathematically, the proposed method is casted as a hierarchical Bayesian model with transfer learning. This approach results in requiring very few late-stage samples points to produce reliable performance models. On many tasks, our approach demonstrates up to 9× reduction in validation time which can be translated to significant reduction of cost.

Random Graphs (Zaheer, J. Lee, et al., 2015) Preferential attachment models for random graphs are successful in capturing many characteristics of real networks such as power law behavior. At the same time they lack flexibility to take vertex to vertex affinities into account, a feature that is commonly used in many link recommendation algorithms. We propose a random graph model based on both node attributes and preferential attachment. This approach overcomes the limitation of existing models on expressing vertex affinity and on reflecting properties of different subgraphs. We analytically prove that our model preserves the power law behavior in the degree distribution as expressed by natural graphs and we show that it satisfies the small world property. Experiments show that our model provides an excellent fit of many natural graph statistics.

Reinforcement Learning for Question Answering (Das, Dhuliawala, et al., 2018) Knowledge bases (KB), both automatically and manually constructed, are often incomplete — many valid facts can be inferred from the KB by synthesizing existing information. A popular approach to KB completion is to infer new relations by combinatory reasoning over the information found along other paths connecting a pair of entities. Given the enormous size of KBs and the exponential number of paths, previous path-based models have considered only the problem of predicting a missing relation given two entities or evaluating the truth of a proposed triple. Additionally, these methods have traditionally used random paths between fixed entity pairs or more recently learned to pick paths between them. We propose a new algorithm MINERVA, which addresses the much more difficult and practical task of answering questions where the relation is known, but only one entity. Since random walks are impractical in a setting with combinatorially many desti-

nations from a start node, we present a neural reinforcement learning approach which learns how to navigate the graph conditioned on the input query to find predictive paths. Empirically, this approach obtains state-of-the-art results on several datasets, significantly outperforming prior methods

Spectral Methods (H.-Y. F. Tung et al., 2017) Nonparametric models are versatile, albeit computationally expensive, tool for modeling mixture models. We introduce spectral methods for the two most popular nonparametric models: the Indian Buffet Process (IBP) and the Hierarchical Dirichlet Process (HDP). We show proposed methods for the inference of nonparametric models are computationally and statistically efficient. In particular, we derive the lower-order moments, propose spectral algorithms for both models, and provide reconstruction guarantees for the algorithms. For the HDP, we further show that applying hierarchical models on dataset with hierarchical structure, which can be solved with the generalized spectral HDP, produces better solutions than flat models.

1.5 HOW TO READ THIS THESIS?

The introduction chapter provides a brief overview of all the chapters and relationship between results of various chapters, and is a prerequisite to rest of the thesis. Each chapter is self-contained and independent of other chapters so that it can be read without any reference to rest of this thesis. Furthermore, the reader is not assumed to have detailed knowledge of machine learning; we rather try to provide the necessary knowledge, allowing the thesis to be accessible to graduate students.

Part I

DIVERSITY BY HANDLING COMPLEX DATA

Data in real-world come in different shapes and sizes. We showcase ways to leverage structure in data, like invariances or heterogeneity in order to establish new mathematical properties. Incorporating these mathematical properties in the model and learning algorithms leads to versatile and sample efficient representations on diverse domains.

2

Set Data

We study the problem of designing models for machine learning tasks defined on *sets*. In contrast to traditional approach of operating on fixed dimensional vectors, we consider objective functions defined on sets that are invariant to permutations. Such problems are widespread, ranging from estimation of population statistics (Poczos, Rinaldo, et al., 2013), to anomaly detection in piezometer data of embankment dams (Jung et al., 2015), to cosmology (Ntampaka et al., 2016; Ravanbakhsh, Junier Oliva, et al., 2016). Our main theorem characterizes the permutation invariant functions and provides a family of functions to which any permutation invariant objective function must belong. This family of functions has a special *structure* which enables us to design a deep network architecture that can operate on sets and which can be deployed on all three types of learning scenarios: supervised, semi-supervised, and unsupervised learning tasks. We also derive the necessary and sufficient conditions for permutation equivariance in deep models. We demonstrate the applicability of our method on discriminative as well as generative tasks.

2.1 INTRODUCTION

A typical machine learning algorithm, like regression or classification, is designed for fixed dimensional data instances. Their extensions to handle the case when the inputs or outputs are permutation invariant sets rather than fixed dimensional vectors is not trivial and researchers have only recently started to investigate them (Muandet, Balduzzi, and Schoelkopf, 2013; Muandet, Fukumizu, et al., 2012; J. Oliva, Poczos, and J. Schneider, 2013; Szabo et al., 2016). In this paper, we present a generic framework to deal with the setting where input and possibly output instances in a machine learning task are sets.

Similar to fixed dimensional data instances, we can characterize two learning paradigms in case of sets. In **supervised learning**, we have an output label for a set that is invariant or equivariant to the permutation of set elements. Examples include tasks like estimation of population statistics (Poczos, Rinaldo, et al., 2013), where applications range from giga-scale cosmology (Ntampaka et al., 2016; Ravanbakhsh, Junier Oliva, et al., 2016) to nano-scale quantum chemistry (Faber, Lindmaa, et al., 2016).

Next, there can be the **semi-supervised setting**, where the "set" structure needs to be learned, *e.g.* by leveraging the homophily/heterophily tendencies within sets. An example is the task of set expansion (a.k.a. audience expansion), where given a set of objects that are similar to each other (*e.g.* set of words {*lion, tiger, leopard*}), our goal is to find new objects from a large pool of candidates such that the selected new objects are similar to the query set (*e.g.* find words like *jaguar* or *cheetah* among all English words). This is a standard problem in similarity search and metric learning, and a typical application is to find new image tags given a small set of possible tags. Likewise, in the field of computational advertisement, given a set of high-value customers, the goal would be to find similar people. This is an important problem in many scientific applications, *e.g.* given a small set of interesting celestial objects, astrophysicists might want to find similar ones in large sky surveys.

Finally, among **unsupervised tasks** generative modeling is perhaps the most fundamental and important where aslo "sets" can play a signigicant role. In this setting, one has to learn a generative model of the distribution of the provided training set and *capable of generating arbitrary many new sample points from the domain of this distribution*. We would like to have detailed generative models for set data, like point clouds. This would allow data augmentation or style transfer directly in point cloud domain which are gaining prominence in field of self-driving cars and virtual reality.

MAIN CONTRIBUTIONS. In this work, (i) we propose a fundamental architecture, *DeepSets*, to deal with sets as inputs and show that the properties of this architecture are both necessary and sufficient (Section 2.3). (ii) We extend this architecture to allow for conditioning on arbitrary objects, and (iii) based on this architecture we develop a *deep network* that can operate on sets with possibly different sizes (Section 2.4). We show that a simple parameter-sharing scheme enables a general treatment of sets within supervised and semi-supervised settings. (iv) We extend the standard GAN to a *hierarchical implicit model* by combining Bayesian hierarchical modeling to learn to generate set data like point clouds. (Section 2.8) (v) Finally, we demonstrate the wide applicability of our framework through experiments on diverse problems (Section 2.5, 2.7, 2.9).

2.2 WHAT ARE INVARIANCE AND EQUIVARIANCE?

Although multilayer perceptrons are universal approximators, however, often in practice rather than throwing large multilayer perceptrons at the problem, we often use specially designed layers and architectures. The specialized layers perform better and computationally more efficient as they encode prior knowledge about the domain. Typically, this domain specific knowledge is incorporated by specifying invariance and equivariance that needs to be obeyed by the network. So, we begin by defining a standard neural network layer and then when would it be said to satisfy given invariance or equivariance.

Definition 2.1 *A standard neural network is a function* $\mathbf{f} : \mathbb{R}^N \to \mathbb{R}^M$*, of the form:*

$$\mathbf{y} = \mathbf{f}(\mathbf{x}) = \sigma(\Theta\mathbf{x} + \mathbf{b}) \tag{2.1}$$

where θ *is the weight matrix and* σ *is a nonlinearity such as sigmoid function.* $\boldsymbol{\sigma}$ *is the point-wise application of* σ *to its input vector.*

In the above definition, the affine portion is often generalized in literature to arbitrary input and output vector spaces over the fields of reals $\mathbb{R}$. However, since elements of any vector space can be expressed in terms of basis (K. Hoffman and Kunze, 1971) and any linear transformation can be expressed as a matrix (K. Hoffman and Kunze, 1971), the definition above suffices.

Invariance is supposed to enforce notions of a feature to remain unaffected by some inconsequential alteration to the input. Formally,

Definition 2.2 *A function* $f : \mathcal{X} \to \mathcal{Y}$ *is said to be invariant with respect to a transformation* $T : \mathcal{X} \to \mathcal{X}$ *when* f*'s value is left unchanged upon application of* T *to the input,* i.e.

$$g(x) = g(T(x)) \tag{2.2}$$

On the other hand, equivariance does not require the values of f to be left unchanged by a transformation, but only that the input and output get transformed in a coupled fashion. To elaborate, consider the simple case when input space and output space are the same, *i.e.*

$\mathcal{X} = \mathcal{Y}$, then equivariance mean a transformation applied to the input of f gives the same result as applying it directly to the output y, *i.e.* $T(f(x)) = f(T(x))$. In general, the equivariance is defined formally as:

Definition 2.3 *A function $f : \mathcal{X} \to \mathcal{Y}$ is said to be equivariant with respect the coupled transformations $T : \mathcal{X} \to \mathcal{X}$ and $T' : \mathcal{Y} \to \mathcal{Y}$ when f's output is changed exactly according to T' upon application of T to the input*, i.e.

$$T'(g(x)) = g(T(x)) \tag{2.3}$$

An interesting property relating invariance and equivariance is that: any pooling of an equivariant function (*e.g.* max, sum, or requiring T' to be identity function) leads to an invariant function.

A common method of capitalizing on invariance and equivariance is through augmentation, *i.e.* adding extra training examples by applying appropriate transformations to account for changes to the data that should not affect changes to representation or corresponding label [7]. The invariance or equivariance is thus learned by seeing many examples from the same class (*i.e.* related to each other by an invariance transformation). Although sometimes effective, this method works by training over the extra augmented examples, which needs increased model capacity, more computation, extra training time, and there is still no guarantee that the resulting representation will be invariant or equivariant to the desired transformation.

We restrict the scope of this work to cases when the invariance and equivariance transformations are liner, *i.e.* it can be expressed by as matrix. Follow-up of this work, generalized to handling equivariance to any coupled group actions (Ravanbakhsh, Jeff Schneider, and Póczos, 2017). We will see next an example where such invariance/equivariance requirement is behind a highly successful specialized neural network architecture.

2.2.1 *Example: CNN as Translation Invariance*

Convolutional Neural Network (CNN) are popular in image processing, like objected detection. In this section, instead of using minimalistic description of a vector space, we stick to the popular convention that input is 2D (*e.g.* images) and output is 2D as well, *i.e.* $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{M \times N}$. Thus, input-output relation of neural network layer would be:

$$y_{ij} = \sigma\left(\sum_{k,l} \Theta_{ijkl} x_{kl} + b_{ij}\right) \tag{2.4}$$

which can be conveniently expressed in tensor-matrix form as:

$$\mathbf{y} = \boldsymbol{\sigma}(\Theta \mathbf{x} + \mathbf{b}) \tag{2.5}$$

For image processing a desirable property would be translation equivariant. For example, if the layer is designed to identify a ball in an image, then a translation in any direction should not affect the outcome, as it's still a rectangle after all. The outcome should also be translated, like the input. Formally, the translation transformation is a linear transformation and thus can be expressed as tensor T. Keeping this in mind the translation equivarance can be expressed as:

$$\begin{aligned} T\mathbf{y} &= \boldsymbol{\sigma}(\Theta T\mathbf{x} + \mathbf{b}) \\ T\boldsymbol{\sigma}(\Theta \mathbf{x} + \mathbf{b}) &= \boldsymbol{\sigma}(\Theta T\mathbf{x} + \mathbf{b}) \\ \boldsymbol{\sigma}(T\Theta \mathbf{x} + Tb) &= \boldsymbol{\sigma}(\Theta T\mathbf{x} + \mathbf{b}) \\ T\Theta \mathbf{x} + T\mathbf{x} &= \Theta T\mathbf{x} + \mathbf{b} \end{aligned} \tag{2.6}$$

Step two to three follows from the fact that $\boldsymbol{\sigma}$ is applied pointwise. Step three to four follows by assuming σ is one-to-one.

Since the polynomial of $\mathbf{x}$ in LHS and RHS are always equal, they must be the same and have equal coefficient. This implies:

$$T\Theta = \Theta T \quad \text{and} \quad T\mathbf{b} = \mathbf{b} \tag{2.7}$$

Thus for bias the condition reduces to mean that bias is same for all position (i, j) and can be succinctly denoted by a scalar c. As for the weight, we need to further simplify

$$T\Theta = \Theta T \Rightarrow \Theta = T^{-1}\Theta T \tag{2.8}$$

Notice that the translation transformation tensor for a translation of s units right and t units up can be explictly expressed as:

$$T_{ijkl} = \delta[(i-k)\%M = s, (j-l)\%N = t] \ , \ -M < s < M \text{ and } -N < t < N \tag{2.9}$$

Note that inverse of translation by (s, t) would be just translation in reverse direction, i.e. $(-s, -t)$. We plug these into (2.8) step by step. First let us evaluate:

$$\begin{aligned}(\Theta T)_{ijkl} &= \sum_{m,n} \theta_{ijmn} T_{mnkl} \\ &= \sum_{m,n} \theta_{ijmn} \delta[(m-k)\%M = s, (n-l)\%N = t] \\ &= \theta_{i,j,(k+s)\%M,(l+t)\%N}\end{aligned} \tag{2.10}$$

Now continuing with remaining T^{-1}, we obtain:

$$\begin{aligned}\Theta_{ijkl} &= (T^{-1}\Theta T)_{ijkl} \\ &= \sum_{m,n} T^{-1}_{ijmn} (\Theta T)_{mnkl} \\ &= \sum_{m,n} \delta[(i-m)\%M = -s, (j-n)\%N = -t] \Theta_{m,n,(k+s)\%M,(l+t)\%N} \\ &= \Theta_{(i+s)\%M,(j+t)\%N,(k+s)\%M,(l+t)\%N}\end{aligned} \tag{2.11}$$

As the above holds for all (s, t), there are many values in Θ are equal. To be specific, there are only $M \times N$ unique value, which can represented in a matrix $W \in \mathbb{R}^{M \times N}$ as:

$$W_{ij} = \Theta_{ij00} = \Theta_{(i+s)\%M,(j+t)\%N,s,t}, \quad \forall 0 \leqslant i, s < M, 0 \leqslant j, t < N. \tag{2.12}$$

or in other way round:

$$\Theta_{ijkl} = W_{(i-k)\%M,(j-l)\%N}. \tag{2.13}$$

Now notice the application of the translation invariant Θ on a 2D input, like an image:

$$\begin{aligned}y_{ij} &= \sigma\left(\sum_{m,n} \Theta_{ijmn} x_{mn} + b_{ij}\right) \\ &= \sigma\Big(\underbrace{\sum_{m,n} W_{(i-m)\%M,(j-n)\%N} x_{mn}}_{\text{convolution!}} + c\Big)\end{aligned} \tag{2.14}$$

This can be easily recognized as the popular Convolutional Neural Network (CNN) (LeCun et al., 1998). Thus the popular CNN automatically pops out when translation equivariance is imposed on a general neural network. It is interesting to note that original CNN in 1968 was not motivated on basis on invariance/equivariance (Hubel and Wiesel, 1968). It was not until in 1988, that CNN were understood to be consequence of shift invariance (W. Zhang, 1988). Motivated by this, in next section we will study effect of another type of invariance that has not be studied before, namely the permutation invariance.

2.3 PERMUTATION INVARIANCE AND EQUIVARIANCE

2.3.1 *Problem Definition*

A function f transforms its domain $\mathcal{X}$ into its range $\mathcal{Y}$. Usually, the input domain is a vector space $\mathbb{R}^d$ and the output response range is either a discrete space, e.g. $\{0, 1\}$ in case of classification, or a continuous space $\mathbb{R}$ in case of regression. Now, if the input is a set $X = \{x_1, \ldots, x_M\}, x_m \in \mathfrak{X}$, *i.e.*, the input domain is the power set $\mathcal{X} = 2^{\mathfrak{X}}$, then we would like the response of the function to be "indifferent" to the ordering of the elements. In other words,

Property 2.4 *A function $f : 2^{\mathfrak{X}} \to \mathcal{Y}$ acting on sets must be permutation **invariant** to the order of objects in the set*, i.e. *for any permutation π :*

$$f(\{x_1, ..., x_m\}) = f(\{x_{\pi(1)}, ..., x_{\pi(M)}\}) \tag{2.15}$$

In the supervised setting, given N examples of of $X^{(1)}, ..., X^{(N)}$ as well as their labels $y^{(1)}, ..., y^{(N)}$, the task would be to classify/regress (with variable number of predictors) while being permutation invariant w.r.t. predictors. Under unsupervised setting, the task would be to assign high scores to valid sets and low scores to improbable sets. These scores can then be used for set expansion tasks, such as image tagging or audience expansion in field of computational advertisement. In *transductive* setting, each instance $x_m^{(n)}$ has an associated labeled $y_m^{(n)}$. Then, the objective would be instead to learn a permutation **equivariant** function $\mathbf{f} : \mathfrak{X}^M \to \mathcal{Y}^M$ that upon permutation of the input instances permutes the output labels, *i.e.* for any permutation π:

$$\mathbf{f}([x_{\pi(1)}, \ldots, x_{\pi(M)}]) = [f_{\pi(1)}(\mathbf{x}), \ldots, f_{\pi(M)}(\mathbf{x})] \tag{2.16}$$

We want to study the structure of functions on sets. Their study in total generality is extremely difficult, so we analyze case-by-case. Roughly speaking, we claim that such functions must have a structure of the form $f(X) = \rho\left(\sum_{x \in X} \phi(x)\right)$ for some functions ρ and ϕ. Over the next two sections we try to formally prove this structure of the permutation invariant functions.

2.3.2 *Countable Case*

We begin by analyzing the **invariant** case when $\mathfrak{X}$ is a countable set and $\mathcal{Y} = \mathbb{R}$, where the next theorem characterizes its structure.

Theorem 2.5 *Assume the elements are countable, i.e. $|\mathfrak{X}| < \aleph_0$. A function $f : 2^{\mathfrak{X}} \to \mathbb{R}$ operating on a set X can be a valid set function, i.e. it is permutation invariant to the elements in X, if and only if it can be decomposed in the form $\rho\left(\sum_{x \in X} \phi(x)\right)$, for suitable transformations ϕ and ρ.*

Proof Permutation invariance follows from the fact that sets have no particular order, hence any function on a set must not exploit any particular order either. The sufficiency follows by observing that the function $\rho\left(\sum_{x\in X}\phi(x)\right)$ satisfies the permutation invariance condition.

To prove necessity, i.e. that all functions can be decomposed in this manner, we begin by noting that there must be a mapping from the elements to natural numbers functions, since the elements are countable. Let this mapping be denoted by $c : \mathfrak{X} \to \mathbb{N}$. Now if we let $\phi(x) = 4^{-c(x)}$ then $\sum_{x\in X}\phi(x)$ constitutes an unique representation for every set $X \in 2^{\mathfrak{X}}$. Now a function $\rho : \mathbb{R} \to \mathbb{R}$ can always be constructed such that $f(X) = \rho\left(\sum_{x\in X}\phi(x)\right)$. ■

2.3.3 *Uncountable Case*

The extension to case when $\mathfrak{X}$ is uncountable, *e.g.* $\mathfrak{X} = [0,1]$, is not so trivial. We could only prove in case of fixed set size, *e.g.* $\mathcal{X} = [0,1]^M$ instead of $\mathcal{X} = 2^{\mathfrak{X}} = 2^{[0,1]}$, that any permutation invariant continuous function can be expressed as $\rho\left(\sum_{x\in X}\phi(x)\right)$. Also, we show that there is a universal approximator of the same form. These results are discussed below.

To illustrate the uncountable case, we assume a fixed set size of M. Without loss of generality we can let $\mathfrak{X} = [0,1]$. Then the domain becomes $[0,1]^M$. Also, to handle ambiguity due to permutation, we often define the domain to be the set $\mathcal{X} = \{(x_1,...,x_M) \in [0,1]^M : x_1 \leqslant x_2 \leqslant \cdots \leqslant x_M\}$ for some ordering of the elements in $\mathfrak{X}$.

The proof builds on the famous Newton-Girard formulae which connect moments of a sample set (sum-of-power) to the elementary symmetric polynomials. But first we present some results needed for the proof. The first result establishes that sum-of-power mapping is injective.

Lemma 2.6 *Let* $\mathcal{X} = \{(x_1,...,x_M) \in [0,1]^M : x_1 \leqslant x_2 \leqslant \cdots \leqslant x_M\}$. *The sum-of-power mapping* $E : \mathcal{X} \to \mathbb{R}^{M+1}$ *defined by the coordinate functions*

$$Z_q := E_q(X) := \sum_{m=1}^{M}(x_m)^q, \qquad q = 0,...,M. \tag{2.17}$$

is injective.

Proof Suppose for some $u, v \in \mathcal{X}$, we have $E(u) = E(v)$. We will now show that it must be the case that $u = v$. Construct two polynomials as follows:

$$P_u(x) = \prod_{m=1}^{M}(x - u_m) \qquad P_v(x) = \prod_{m=1}^{M}(x - v_m) \tag{2.18}$$

If we expand the two polynomials we obtain:

$$\begin{aligned} P_u(x) &= x^M - a_1x^{M-1} + \cdots (-1)^{M-1}a_{M-1}x + (-1)^M a_M \\ P_v(x) &= x^M - b_1x^{M-1} + \cdots (-1)^{M-1}b_{M-1}x + (-1)^M b_M \end{aligned} \tag{2.19}$$

with coefficients being elementary symmetric polynomials in u and v respectively, *i.e.*

$$a_m = \sum_{1\leqslant j_1<j_2<\cdots<j_m\leqslant M} u_{j_1}u_{j_2}\cdots u_{j_m} \qquad b_m = \sum_{1\leqslant j_1<j_2<\cdots<j_m\leqslant M} v_{j_1}v_{j_2}\cdots v_{j_m} \tag{2.20}$$

These elementary symmetric polynomials can be uniquely expressed as a function of $E(u)$ and $E(v)$ respectively, by Newton-Girard formula. The m-th coefficient is given by the determinant of $m \times m$ matrix having terms from $E(u)$ and $E(v)$ respectively:

$$a_m = \frac{1}{m} \det \begin{pmatrix} E_1(u) & 1 & 0 & 0 & \cdots & 0 \\ E_2(u) & E_1(u) & 1 & 0 & \cdots & 0 \\ E_3(u) & E_2(u) & E_1(u) & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ E_{m-1}(u) & E_{m-2}(u) & E_{m-3}(u) & E_{m-4}(u) & \cdots & 1 \\ E_m(u) & E_{m-1}(u) & E_{m-2}(u) & E_{m-3}(u) & \cdots & E_1(u) \end{pmatrix} \tag{2.21}$$

$$b_m = \frac{1}{m} \det \begin{pmatrix} E_1(v) & 1 & 0 & 0 & \cdots & 0 \\ E_2(v) & E_1(v) & 1 & 0 & \cdots & 0 \\ E_3(v) & E_2(v) & E_1(v) & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ E_{m-1}(v) & E_{m-2}(v) & E_{m-3}(v) & E_{m-4}(v) & \cdots & 1 \\ E_m(v) & E_{m-1}(v) & E_{m-2}(v) & E_{m-3}(v) & \cdots & E_1(v) \end{pmatrix}$$

Since we assumed $E(u) = E(v)$ implying $[a_1, ..., a_M] = [b_1, ..., b_M]$, which in turn implies that the polynomials P_u and P_v are the same. Therefore, their roots must be the same, which shows that $u = v$. ■

The second result we borrow from Ćurgus and Mascioni (2006) which establishes a homeomorphism between coefficients and roots of a polynomial.

Theorem 2.7 *(Ćurgus and Mascioni, 2006) The function $f : \mathbb{C}^M \to \mathbb{C}^M$, which associates every $a \in \mathbb{C}^M$ to the multiset of roots, $f(a) \in \mathbb{C}^M$, of the monic polynomial formed using a as the coefficient* i.e. $x^M + a_1 x^{M-1} + \cdots (-1)^{M-1} a_{M-1} x + (-1)^M a_M$, *is a homeomorphism.*

Among other things, this implies that (complex) roots of a polynomial depends continuously on the coefficients. We will use this fact for our next lemma.

Finally, we establish a continuous inverse mapping for the sum-of-power function.

Lemma 2.8 *Let $\mathcal{X} = \{(x_1, ..., x_M) \in [0, 1]^M : x_1 \leqslant x_2 \leqslant \cdots \leqslant x_M\}$. We define the sum-of-power mapping $E : \mathcal{X} \to \mathcal{Z}$ by the coordinate functions*

$$Z_q := E_q(X) := \sum_{m=1}^{M} (x_m)^q, \qquad q = 0, ..., M. \tag{2.22}$$

where $\mathcal{Z}$ is the range of the function. The function E has a continuous inverse mapping.

Proof First of all note that $\mathcal{Z}$, the range of E, is a compact set. This follows from following observations:

- The domain of E is a bounded polytope (*i.e.* a compact set),

- E is a continuous function, and
- image of a compact set under a continuous function is a compact set.

To show the continuity of inverse mapping, we establish connection to the continuous dependence of roots of polynomials on its coefficients.

As in Lemma 4, for any $u \in \mathcal{X}$, let $z = E(u)$ and construct the polynomial:

$$P_u(x) = \prod_{m=1}^{M} (x - u_m) \tag{2.23}$$

If we expand the polynomial we obtain:

$$P_u(x) = x^M - a_1 x^{M-1} + \cdots (-1)^{M-1} a_{M-1} x + (-1)^M a_M \tag{2.24}$$

with coefficients being elementary symmetric polynomials in u, *i.e.*

$$a_m = \sum_{1 \leqslant j_1 < j_2 < \cdots < j_m \leqslant M} u_{j_1} u_{j_2} \cdots u_{j_m} \tag{2.25}$$

These elementary symmetric polynomials can be uniquely expressed as a function of z by Newton-Girard formula:

$$a_m = \frac{1}{m} \det \begin{pmatrix} z_1 & 1 & 0 & 0 & \cdots & 0 \\ z_2 & z_1 & 1 & 0 & \cdots & 0 \\ z_3 & z_2 & z_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ z_{m-1} & z_{m-2} & z_{m-3} & z_{m-4} & \cdots & 1 \\ z_m & z_{m-1} & z_{m-2} & z_{m-3} & \cdots & z_1 \end{pmatrix} \tag{2.26}$$

Since determinants are just polynomials, a is a continuous function of z. Thus to show continuity of inverse mapping of E, it remains to show continuity from a back to the roots u. In this regard, we invoke Theorem 5. Note that homeomorphism implies the mapping as well as its inverse is continuous. Thus, restricting to the compact set $\mathcal{Z}$ where the map from coefficients to roots only goes to the reals, the desired result follows. To explicitly check the continuity, note that limit of $E^{-1}(z)$, as z approaches z^* from inside $\mathcal{Z}$, always exists and is equal to $E^{-1}(z^*)$ since it does so in the complex plane. ■

With the lemma developed above we are in a position to tackle the main theorem.

Theorem 2.9 *Let $f : [0,1]^M \to \mathbb{R}$ be a permutation invariant continuous function iff it has the representation*

$$f(x_1, ..., x_M) = \rho \left(\sum_{m=1}^{M} \phi(x_m) \right) \tag{2.27}$$

for some continuous outer and inner function $\rho : \mathbb{R}^{M+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{M+1}$ respectively. The inner function ϕ is independent of the function f.

Proof The sufficiency follows by observing that the function $\rho\left(\sum_{m=1}^{M}\phi(x_m)\right)$ satisfies the permutation invariance condition.

To prove necessity, *i.e.* that all permutation invariant continuous functions over the compact set can be expressed in this manner, we divide the proof into two parts, with outline in Figure 2.1. We begin by looking at the continuous embedding formed by the inner function: $E(X) = \sum_{m=1}^{M}\phi(x_m)$. Consider $\phi : \mathbb{R} \to \mathbb{R}^{M+1}$ defined as $\phi(x) = [1, x, x^2, ..., x^M]$. Now as E is a polynomial, the image of $[0,1]^M$ in $\mathbb{R}^{M+1}$ under E is a compact set as well, denote it by $\mathcal{Z}$. Then by definition, the embedding $E : [0,1]^M \to \mathcal{Z}$ is surjective. Using Lemma 4 and 6, we know that upon restricting the permutations, *i.e.* replacing $[0,1]^M$ with $\mathcal{X} = \{(x_1, ..., x_M) \in [0,1]^M : x_1 \leqslant x_2 \leqslant \cdots \leqslant x_M\}$, the embedding $E : \mathcal{X} \to \mathcal{Z}$ is injective with a continuous inverse. Therefore, combining these observation we get that E is a homeomorphism between $\mathcal{X}$ and $\mathcal{Z}$. Now it remains to show that we can map the embedding to desired target value, *i.e.* to show the existence of the continuous map $\rho : \mathcal{Z} \to \mathbb{R}$ such that $\rho(E(X)) = f(X)$. In particular consider the map $\rho(z) = f(E^{-1}(z))$. The continuity of ρ follows directly from the fact that composition of continuous functions is continuous. Therefore we can always find continuous functions ϕ and ρ to express any permutation invariant function f as $\rho\left(\sum_{m=1}^{M}\phi(x_m)\right)$. ■

A very similar but more general results holds in case of any continuous function (not necessarily permutation invariant). The result is known as Kolmogorov-Arnold representation theorem (Khesin and Tabachnikov, 2014, Chap. 17) which we state below:

Theorem 2.10 ***(Kolmogorov–Arnold representation)*** *Let $f : [0,1]^M \to \mathbb{R}$ be an arbitrary multivariate continuous function iff it has the representation*

$$f(x_1, ..., x_M) = \rho\left(\sum_{m=1}^{M}\lambda_m\phi(x_m)\right) \tag{2.28}$$

with continuous outer and inner functions $\rho : \mathbb{R}^{2M+1} \to \mathbb{R}$ and $\phi : \mathbb{R} \to \mathbb{R}^{2M+1}$. The inner function ϕ is independent of the function f.

This theorem essentially states a representation theorem for any multivariate continuous function. Their representation is very similar to the one we are proved, except for the dependence of inner transformation on the co-ordinate through λ_m. Thus it is reassuring that behind all

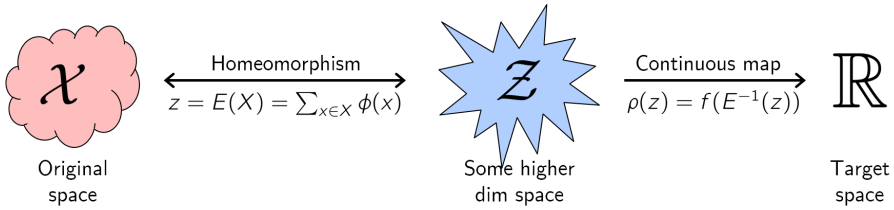


Figure 2.1: Outline of the proof strategy for Theorem 2.1. The proof consists of two parts. First, we desire to show that we can find unique embeddings for each possible input, *i.e.* we show that there exists a homeomorphism E of the form $E(X) = \sum_{x \in X}\phi(x)$ between original domain and some higher dimensional space $\mathcal{Z}$. The second part of the proof consists of showing we can map the embedding to desired target value, *i.e.* to show the existence of the continuous map ρ between $\mathcal{Z}$ and original target space such that $f(X) = \rho(\sum_{x \in X}\phi(x))$.

the beautiful mathematics something intuitive is happening. If the function is permutation invariant, this dependence on co-ordinate of the inner transformation gets dropped!

Further we can show that arbitrary approximator having the same form can be obtained for continuous permutation-invariant functions.

Theorem 2.11 *Assume the elements are from a compact set in* $\mathbb{R}^d$, i.e. *possibly uncountable, and the set size is fixed to* M. *Then any continuous function operating on a set* X, *i.e.* $f : \mathbb{R}^{d\times M} \to \mathbb{R}$ *which is permutation invariant to the elements in* X *can be approximated arbitrarily close in the form of* $\rho\left(\sum_{x\in X}\phi(x)\right)$, *for suitable transformations* ϕ *and* ρ.

Proof Permutation invariance follows from the fact that sets have no particular order, hence any function on a set must not exploit any particular order either. The sufficiency follows by observing that the function $\rho\left(\sum_{x\in X}\phi(x)\right)$ satisfies the permutation invariance condition.

To prove necessity, *i.e.* that all continuous functions over the compact set can be approximated arbitrarily close in this manner, we begin noting that polynomials are universal approximators by Stone–Weierstrass theorem (Marsden and M. J. Hoffman, 1993, sec. 5.7). In this case the Chevalley-Shephard-Todd (CST) theorem (Bourbaki, 1990, chap. V, theorem 4), or more precisely, its special case, the Fundamental Theorem of Symmetric Functions states that symmetric polynomials are given by a polynomial of homogeneous symmetric monomials. The latter are given by the sum over monomial terms, which is all that we need since it implies that all symmetric polynomials can be written in the form required by the theorem. ■

Finally, we still conjecture that even in case of sets of all sizes, *i.e.* when the domain is $2^{[0,1]}$, a representation of the form $f(X) = \rho\left(\sum_{x\in X}\phi(x)\right)$ should exist for all "continuous" permutation invariant functions for some suitable transformations ρ and ϕ. However, in this case even what a "continuous" function means is not clear as the space $2^{[0,1]}$ does not have any natural topology. As a future work, we want to study further by defining various topologies, like using Fréchet distance as used in Ćurgus and Mascioni (2006) or MMD distance. Our preliminary findings in this regards hints that using MMD distance if the representation is allowed to be in ℓ^2, instead of being finite dimensional, then the conjecture seems to be provable. Thus, clearly this direction needs further exploration. We end this section by providing some examples:

EXAMPLES:

- $x_1x_2(x_1+x_2+3)$, Consider $\phi(x) = [x, x^2, x^3]$ and $\rho([u, v, w]) = uv - w + 3(u^2 - v)/2$, then $\rho(\phi(x_1) + \phi(x_2))$ is the desired function.
- $x_1x_2x_3 + x_1 + x_2 + x_3$, Consider $\phi(x) = [x, x^2, x^3]$ and $\rho([u, v, w]) = (u^3 + 2w - 3uv)/6 + u$, then $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3))$ is the desired function.
- $1/n(x_1 + x_2 + x_3 + ... + x_m)$, Consider $\phi(x) = [1, x]$ and $\rho([u, v]) = v/u$, then $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3) + ... + \phi(x_m))$ is the desired function.
- $\max\{x_1, x_2, x_3, ..., x_m\}$, Consider $\phi(x) = [e^{\alpha x}, xe^{\alpha x}]$ and $\rho([u, v]) = v/u$, then as $\alpha \to \infty$, then we have $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3) + ... + \phi(x_m))$ approaching the desired function.
- Second largest among $\{x_1, x_2, x_3, ..., x_m\}$, Consider $\phi(x) = [e^{\alpha x}, xe^{\alpha x}]$ and $\rho([u, v]) = (v - (v/u)e^{\alpha v/u})/(u - e^{\alpha v/u})$, then as $\alpha \to \infty$, we have $\rho(\phi(x_1) + \phi(x_2) + \phi(x_3) + ... + \phi(x_m))$ approaching the desired function.

2.3.4 *Specialized Layer*

Next, we analyze the **equivariant** case when $\mathfrak{X} = \mathcal{Y} = \mathbb{R}$ and $\mathbf{f}$ is restricted to be a neural network layer. Our goal is to design neural network layers that are equivariant to permutations of elements in the input $\mathbf{x}$. The function $\mathbf{f} : \mathfrak{X}^M \to \mathcal{Y}^M$ is **equivariant** to the permutation of its inputs iff

$$\mathbf{f}(\pi\mathbf{x}) = \pi\mathbf{f}(\mathbf{x}) \quad \forall \pi \in \mathcal{S}_M$$

where the symmetric group $\mathcal{S}_M$ is the set of all permutation of indices $1, \ldots, M$.

Consider the standard neural network layer

$$\mathbf{f}_\Theta(\mathbf{x}) \doteq \boldsymbol{\sigma}(\Theta\mathbf{x}) \quad \Theta \in \mathbb{R}^{M\times M} \tag{2.29}$$

where Θ is the weight vector and $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinearity such as sigmoid function. The following lemma states the necessary and sufficient conditions for permutation-equivariance in this type of function.

Theorem 2.12 *The function $\mathbf{f}_\Theta : \mathbb{R}^M \to \mathbb{R}^M$ as defined in (2.29) is permutation equivariant if and only if all the off-diagonal elements of Θ are tied together and all the diagonal elements are equal as well. That is,*

$$\Theta = \lambda\mathbf{I} + \gamma\,(\mathbf{1}\mathbf{1}^\mathsf{T}) \qquad \lambda, \gamma \in \mathbb{R} \quad \mathbf{1} = [1, \ldots, 1]^\mathsf{T} \in \mathbb{R}^M$$

where $\mathbf{I} \in \mathbb{R}^{M\times M}$ is the identity matrix.

Proof From definition of permutation equivariance $\mathbf{f}_\Theta(\pi\mathbf{x}) = \pi\mathbf{f}_\Theta(\mathbf{x})$ and definition of $\mathbf{f}$ in (2.29), the condition becomes $\boldsymbol{\sigma}(\Theta\pi\mathbf{x}) = \pi\boldsymbol{\sigma}(\Theta\mathbf{x})$, which (assuming sigmoid is a bijection) is equivalent to $\Theta\pi = \pi\Theta$. Therefore we need to show that the necessary and sufficient conditions for the matrix $\Theta \in \mathbb{R}^{M\times M}$ to commute with all permutation matrices $\pi \in \mathcal{S}_M$ is given by this proposition. We prove this in both directions:

- To see why $\Theta = \lambda\mathbf{I} + \gamma\,(\mathbf{1}\mathbf{1}^\mathsf{T})$ commutes with any permutation matrix, first note that commutativity is linear – that is

  $$\Theta_1\pi = \pi\Theta_1 \wedge \Theta_2\pi = \pi\Theta_2 \quad \Rightarrow \quad (a\Theta_1 + b\Theta_2)\pi = \pi(a\Theta_1 + b\Theta_2).$$

  Since both Identity matrix $\mathbf{I}$, and constant matrix $\mathbf{1}\mathbf{1}^\mathsf{T}$, commute with any permutation matrix, so does their linear combination $\Theta = \lambda\mathbf{I} + \gamma\,(\mathbf{1}\mathbf{1}^\mathsf{T})$.

- We need to show that in a matrix Θ that commutes with "all" permutation matrices
  - *All diagonal elements are identical*: Let $\pi_{k,l}$ for $1 \leqslant k, l \leqslant M, k \neq l$, be a transposition (*i.e.* a permutation that only swaps two elements). The inverse permutation matrix of $\pi_{k,l}$ is the permutation matrix of $\pi_{l,k} = \pi_{k,l}^\mathsf{T}$. We see that commutativity of Θ with the transposition $\pi_{k,l}$ implies that $\Theta_{k,k} = \Theta_{l,l}$:

    $$\pi_{k,l}\Theta = \Theta\pi_{k,l} \;\Rightarrow\; \pi_{k,l}\Theta\pi_{l,k} = \Theta \;\Rightarrow\; (\pi_{k,l}\Theta\pi_{l,k})_{l,l} = \Theta_{l,l} \;\Rightarrow\; \Theta_{k,k} = \Theta_{l,l}$$

    Therefore, π and Θ commute for any permutation π, they also commute for any transposition $\pi_{k,l}$ and therefore $\Theta_{i,i} = \lambda\,\forall i$.
  - *All off-diagonal elements are identical*: We show that since Θ commutes with any product of transpositions, any choice two off-diagonal elements should be identical. Let

(i,j) and (i',j') be the index of two off-diagonal elements (*i.e.* $i \neq j$ and $i' \neq j'$). Moreover for now assume $i \neq i'$ and $j \neq j'$. Application of the transposition $\pi_{i,i'}\Theta$, swaps the rows i, i' in Θ. Similarly, $\Theta\pi_{j,j'}$ switches the j^{th} column with j'^{th} column. From commutativity property of Θ and $\pi \in \mathcal{S}_n$ we have

$$\begin{aligned}
&\pi_{j',j}\pi_{i,i'}\Theta = \Theta\pi_{j',j}\pi_{i,i'} \;\Rightarrow \pi_{j',j}\pi_{i,i'}\Theta(\pi_{j',j}\pi_{i,i'})^{-1} = \Theta &&\Rightarrow \\
&\pi_{j',j}\pi_{i,i'}\Theta\pi_{i',i}\pi_{j,j'} = \Theta \;\Rightarrow (\pi_{j',j}\pi_{i,i'}\Theta\pi_{i',i}\pi_{j,j'})_{i,j} = \Theta_{i,j} &&\Rightarrow\; \Theta_{i',j'} = \Theta_{i,j}
\end{aligned}$$

where in the last step we used our assumptions that $i \neq i'$, $j \neq j'$, $i \neq j$ and $i' \neq j'$. In the cases where either $i = i'$ or $j = j'$, we can use the above to show that $\Theta_{i,j} = \Theta_{i'',j''}$ and $\Theta_{i',j'} = \Theta_{i'',j''}$, for some $i'' \neq i, i'$ and $j'' \neq j, j'$, and conclude $\Theta_{i,j} = \Theta_{i',j'}$.

■

This result can be easily extended to higher dimensions, *i.e.*, $\mathfrak{X} = \mathbb{R}^d$ when λ, γ can be matrices.

2.3.5 *Related Results*

The general form of Theorem 2.5 is closely related with important results in different domains. Here, we quickly review some of these connections.

DE FINETTI THEOREM. A related concept is that of an exchangeable model in Bayesian statistics, It is backed by deFinetti's theorem which states that any exchangeable model can be factored as

$$p(X|\alpha, M_0) = \int d\theta \left[\prod_{m=1}^{M} p(x_m|\theta)\right] p(\theta|\alpha, M_0), \tag{2.30}$$

where θ is some latent feature and α, M_0 are the hyper-parameters of the prior. To see that this fits into our result, let us consider exponential families with conjugate priors, where we can analytically calculate the integral of (2.30). In this special case $p(x|\theta) = \exp(\langle\phi(x), \theta\rangle - g(\theta))$ and $p(\theta|\alpha, M_0) = \exp(\langle\theta, \alpha\rangle - M_0 g(\theta) - h(\alpha, M_0))$. Now if we marginalize out θ, we get a form which looks exactly like the one in Theorem 2.5

$$p(X|\alpha, M_0) = \exp\left(h\left(\alpha + \sum_m \phi(x_m), M_0 + M\right) - h(\alpha, M_0)\right). \tag{2.31}$$

REPRESENTER THEOREM AND KERNEL MACHINES. Support distribution machines use $f(p) = \sum_i \alpha_i y_i K(p_i, p) + b$ as the prediction function (Muandet, Fukumizu, et al., 2012; Poczos, L. Xiong, et al., 2012), where p_i, p are distributions and $\alpha_i, b \in \mathbb{R}$. In practice, the p_i, p distributions are never given to us explicitly, usually only i.i.d. sample sets are available from these distributions, and therefore we need to estimate kernel $K(p, q)$ using these samples. A popular approach is to use $\hat{K}(p, q) = \frac{1}{MM'}\sum_{i,j} k(x_i, y_j)$, where k is another kernel operating on the samples $\{x_i\}_{i=1}^{M} \sim p$ and $\{y_j\}_{j=1}^{M'} \sim q$. Now, these prediction functions can be seen fitting into the structure of our Theorem.

SPECTRAL METHODS. A consequence of the polynomial decomposition is that spectral methods (Anandkumar et al., 2014) can be viewed as a special case of the mapping $\rho \circ \phi(X)$: in that case one can compute polynomials, usually only up to a relatively low degree (such as $k = 3$), to perform inference about statistical properties of the distribution. The statistics are exchangeable in the data, hence they could be represented by the above map.

2.4 DEEP SETS

2.4.1 *Architecture*

2.4.1.1 *Invariant model*

The structure of permutation invariant functions in Theorem 2.5 hints at a general strategy for inference over sets of objects, which we call deep sets. Replacing ϕ and ρ by universal approximators leaves matters unchanged, since, in particular, ϕ and ρ can be used to approximate arbitrary polynomials. Then, it remains to learn these approximators. This yields in the following model:

- Each instance x_m $\forall 1 \leqslant m \leqslant M$ is transformed (possibly by several layers) into some representation $\phi(x_m)$.
- The addition $\sum_m \phi(x_m)$ of these representations processed using the ρ network very much in the same manner as in any deep network (*e.g.* fully connected layers, nonlinearities, *etc*).
- Optionally: If we have additional meta-information z, then the above mentioned networks could be conditioned to obtain the conditioning mapping $\phi(x_m|z)$.

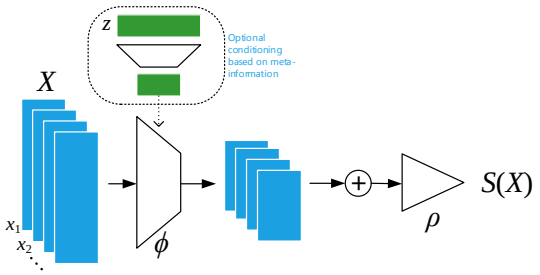


Figure 2.2: Architecture of DeepSets: Invariant

In other words, the key to deep sets is to add up all representations and then apply nonlinear transformations.

The overall model structure is illustrated in Figure 2.2.

This architecture has a number of desirable properties in terms of universality and correctness. We assume in the following that the networks we choose are, in principle, universal approximators. That is, we assume that they can represent any functional mapping. This is a well established property, *c.f.* Micchelli (1984) for details in the case of radial basis function networks.

What remains is to state the derivatives with regard to this novel type of layer. Assume parametrizations w_ρ and w_ϕ for ρ and ϕ respectively. Then we have

$$\partial_{w_\phi} \rho \left(\sum_{x' \in X} \phi(x') \right) = \rho' \left(\sum_{x' \in X} \phi(x) \right) \sum_{x' \in X} \partial_{w_\phi} \phi(x')$$

This result reinforces the common knowledge of parameter tying in deep networks when ordering is irrelevant. Our result backs this practice with theory and strengthens it by proving that it is the only way to do it.

2.4.1.2 *Equivariant model*

Our goal is to design neural network layers that are equivariant to the permutations of elements in the input $\mathbf{x}$. Based on Theorem 2.12, a neural network layer $\mathbf{f}_\Theta(\mathbf{x})$ is permutation equivariant if and only if all the off-diagonal elements of Θ are tied together and all the diagonal elements are equal as well, *i.e.*, $\Theta = \lambda \mathbf{I} + \gamma \, (\mathbf{1}\mathbf{1}^\top)$ for $\lambda, \gamma \in \mathbb{R}$.

This function is simply a non-linearity applied to a weighted combination of i) its input $\mathbf{Ix}$ and; ii) the sum of input values $(\mathbf{1}\mathbf{1}^\top)\mathbf{x}$. Since summation does not depend on the permutation, the layer is permutation-equivariant. Therefore we can manipulate the operations and parameters in this layer, for example to get another **variation***e.g.*:

$$\mathbf{f}(\mathbf{x}) \doteq \boldsymbol{\sigma}\left(\lambda\mathbf{I}\mathbf{x} + \gamma\ \text{maxpool}(\mathbf{x})\mathbf{1}\right). \tag{2.32}$$

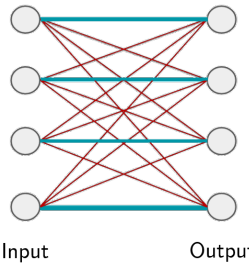


Figure 2.3: Illustration of permutation equivariant layer. Same color indicates weight sharing.

where the maxpooling operation over elements of the set (similar to sum) is commutative. In practice, this variation performs better in some applications. This may be due to the fact that for $\lambda = \gamma$, the input to the non-linearity is max-normalized. Since composition of permutation equivariant functions is also permutation equivariant, we can build DeepSets by stacking such layers as explained below.

So far we assumed that each instance $x_m \in \mathbb{R}$ – *i.e.* a single input and also output channel. For **multiple input-output channels**, we may speed up the operation of the layer using matrix multiplication. For D/D' input/output channels (*i.e.* $\mathbf{x} \in \mathbb{R}^{M\times D}$, $\mathbf{y} \in \mathbb{R}^{M\times D'}$, this layer becomes

$$f(\mathbf{x}) = \sigma\big(\mathbf{x}\Lambda - \mathbf{1}\mathbf{1}^\top\mathbf{x}\Gamma\big) \tag{2.33}$$

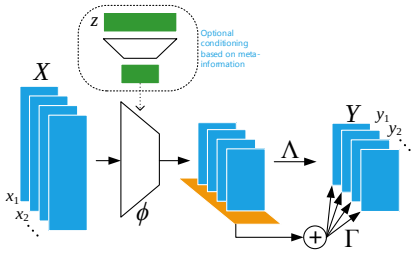


Figure 2.4: Architecture of DeepSets: Equivariant

where $\Lambda, \Gamma \in \mathbb{R}^{D\times D'}$ are model parameters. As before, we can have a maxpool version as: $f(\mathbf{x}) = \sigma\big(\mathbf{x}\Lambda - \mathbf{1}\text{maxpool}(\mathbf{x})\Gamma\big)$ where $\text{maxpool}(\mathbf{x}) = (\max_m \mathbf{x}) \in \mathbb{R}^{1\times D}$ is a row-vector of maximum value of $\mathbf{x}$ over the "set" dimension. We may further reduce the number of parameters in favor of better generalization by factoring Γ and Λ and keeping a single $\Lambda \in \mathbb{R}^{D,D'}$ and $\beta \in \mathbb{R}^{D'}$

$$f(\mathbf{x}) = \sigma\big(\beta + \big(\mathbf{x} - \mathbf{1}\text{maxpool}(\mathbf{x})\big)\Gamma\big) \tag{2.34}$$

STACKING: Since composition of permutation equivariant functions is also permutation equivariant, we can build deep models by stacking layers of (2.34). Moreover, application of any commutative pooling operation (*e.g.* max-pooling) over the set instances produces a permutation *invariant* function.

2.4.2 *Other Related Works*

Several recent works study equivariance and invariance in deep networks w.r.t. general group of transformations (T. S. Cohen and Welling, 2016; Gens and Domingos, 2014; Ravanbakhsh, Jeff Schneider, and Póczos, 2017). For example, X. Chen, Cheng, and Mallat (2014) construct deep permutation invariant features by pairwise coupling of features at the previous layer, where $f_{i,j}([x_i, x_j]) \doteq [|x_i - x_j|, x_i + x_j]$ is invariant to transposition of i and j. Pairwise interactions within sets have also been studied in M. B. Chang et al. (2016) and Guttenberg et al. (2016). Vinyals, S. Bengio, and Kudlur (2015) approach unordered instances by finding "good" orderings.

The idea of pooling a function across set-members is not new. In Lopez-Paz et al. (2016), pooling was used binary classification task for causality on a set of samples. B. Shi et al. (2015) use

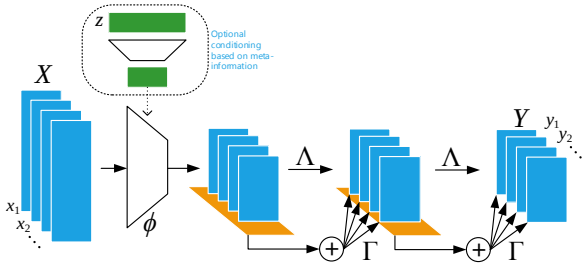


Figure 2.5: Using multiple permutation equivariant layers. Since permutation equivariance compose we can stack multiple such layers

pooling across a panoramic projection of 3D object for classification, while Hang Su et al. (2015) perform pooling across multiple views. Hartford, Wright, and Leyton-Brown (2016) observe the invariance of the payoff matrix in normal form games to the permutation of its rows and columns (*i.e.* player actions) and leverage pooling to predict the player action. The need of permutation equivariance also arise in deep learning over sensor networks and multi-agent setings, where a special case of Theorem 2.12 has been used as the architecture (Sukhbaatar, Fergus, et al., 2016).

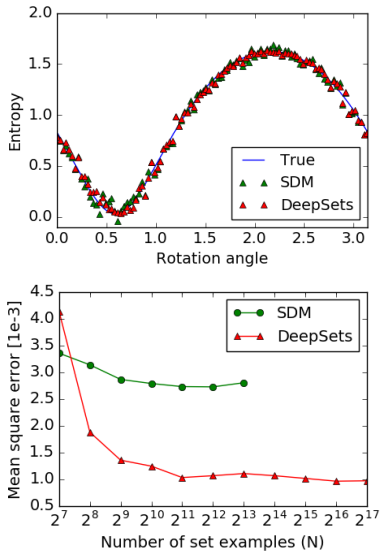
In light of these related works, we would like to emphasize our novel contributions: (i) the universality result of Theorem 2.5 for permutation invariance that also relates DeepSets to other machine learning techniques, see Section 2.3.5; (ii) the permutation equivariant layer of (2.32), which, according to Theorem 2.12 identifies necessary and sufficient form of parameter-sharing in a standard neural layer and; (iii) novel application settings that we study next.

2.5 EXPERIMENTS ON DISCRIMINATIVE TASKS

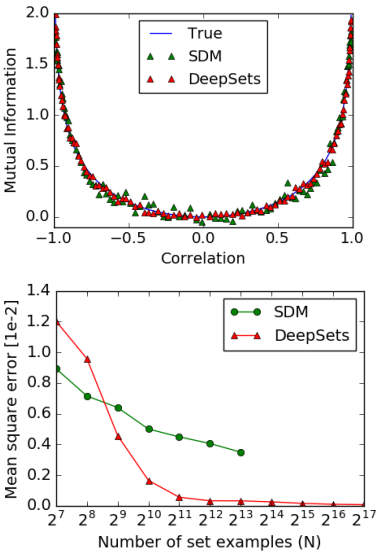
2.5.1 *Estimate Population Statistics*

In the first experiment, we learn entropy and mutual information of Gaussian distributions, without providing any information about Gaussianity to DeepSets. The Gaussians are generated as follows:

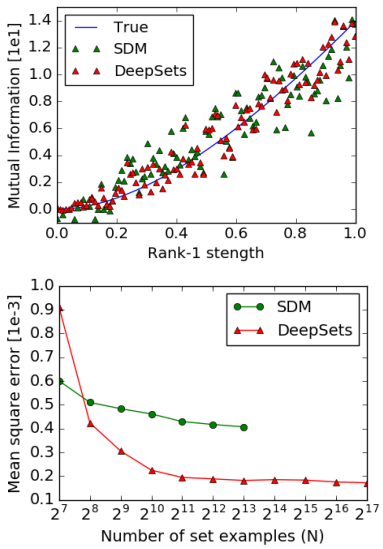
- Rotation: We randomly chose a 2×2 covariance matrix Σ, and then generated N sample sets from $\mathcal{N}(0, R(\alpha)\Sigma R(\alpha)^T)$ of size $M = [300 - 500]$ for N random values of $\alpha \in [0, \pi]$. Our goal was to learn the entropy of the marginal distribution of first dimension. $R(\alpha)$ is the rotation matrix.
- Correlation: We randomly chose a $d \times d$ covariance matrix Σ for $d = 16$, and then generated N sample sets from $\mathcal{N}(0, [\Sigma, \alpha\Sigma; \alpha\Sigma, \Sigma])$ of size $M = [300 - 500]$ for N random values of $\alpha \in (-1, 1)$. Goal was to learn the mutual information of among the first d and last d dimension.
- Rank 1: We randomly chose $v \in \mathbb{R}^{32}$ and then generated a sample sets from $\mathcal{N}(0, I + \lambda vv^T)$ of size $M = [300 - 500]$ for N random values of $\lambda \in (0, 1)$. Goal was to learn the mutual information.



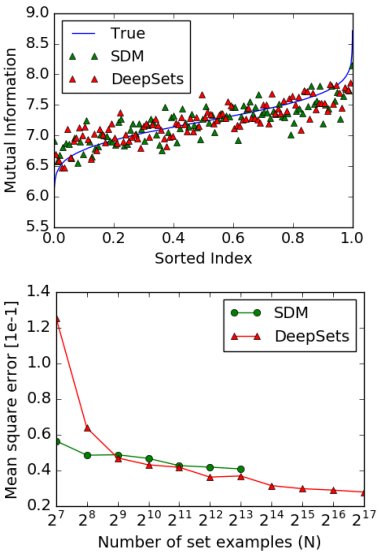
(a) Entropy estimation for first dimension of rotated of 2d Gaussian



(b) Mutual information estimation by varying correlation



(c) Mutual information estimation by varying rank-1 strength



(d) Mutual information on 32d random covariance matrices

Figure 2.6: Population statistic estimation: Top set of figures, show prediction of DeepSets vs SDM for $N = 2^{10}$ case. Bottom set of figures, depict the mean squared error behavior as number of sets is increased. SDM has lower error for small N and DeepSets requires more data to reach similar accuracy. But for high dimensional problems deep sets easily *scales* to large number of examples and produces much *lower* estimation error. Note that the $N \times N$ matrix inversion in SDM makes it prohibitively expensive for $N > 2^{14} = 16384$.

- Random: We chose N random $d \times d$ covariance matrices Σ for $d = 32$, and using each, generated a sample set from $\mathcal{N}(0, \Sigma)$ of size $M = [300 - 500]$. Goal was to learn the mutual information.

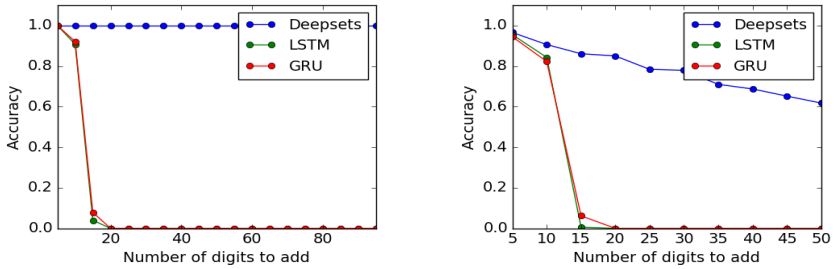


Figure 2.7: Accuracy of digit summation with text (*left*) and image (*right*) inputs. All approaches are trained on tasks of length 10 at most, tested on examples of length up to 100. We see that DeepSets generalizes better.

We train using L_2 loss with a DeepSets architecture having 3 fully connected layers with ReLU activation for both transformations ϕ and ρ. We compare against Support Distribution Machines (SDM) using a RBF kernel (Poczos, L. Xiong, et al., 2012), and analyze the results in Figure 2.6.

2.5.2 *Sum of Digits*

Next, we compare to what happens if our set data is treated as a sequence. We consider the task of finding sum of a given set of digits. We consider two variants of this experiment:

TEXT We randomly sample a subset of maximum $M = 10$ digits from this dataset to build 100k "sets" of training images, where the set-label is sum of digits in that set. We test against sums of M digits, for M starting from 5 all the way up to 100 over another 100k examples.

IMAGE MNIST8m (Loosli, Canu, and Léon Bottou, 2007) contains 8 million instances of 28×28 grey-scale stamps of digits in $\{0, \ldots, 9\}$. We randomly sample a subset of maximum $M = 10$ images from this dataset to build $N = 100$k "sets" of training and 100k sets of test images, where the set-label is the sum of digits in that set (*i.e.* individual labels per image is unavailable). We test against sums of M images of MNIST digits, for M starting from 5 all the way up to 50.

We compare against recurrent neural networks – LSTM and GRU. All models are defined to have similar number of layers and parameters. The output of all models is a scalar, predicting the sum of N digits. Training is done on tasks of length 10 at most, while at test time we use examples of length up to 100. The accuracy, *i.e.* exact equality after rounding, is shown in Figure 2.7. DeepSets generalize much better. Note for image case, the best classification error for single digit is around $p = 0.01$ for MNIST8m, so in a collection of N of images at least one image will be misclassified is $1 - (1 - p)^N$, which is 40% for $N = 50$. This matches closely with observed value in Figure 2.7(b).

2.5.3 *Point Cloud Classification*

A point-cloud is a set of low-dimensional vectors. This type of data is frequently encountered in various applications like robotics, vision, and cosmology. In these applications, existing methods often convert the point-cloud data to voxel or mesh representation as a preprocessing step, *e.g.* H.-W. Lin, Tai, and G.-J. Wang (2004), Maturana and Scherer (2015), and Ravanbakhsh, Junier Oliva, et al. (2016). Since the output of many range sensors, such as LiDAR, is in the form of point-cloud, direct application of deep learning methods to point-cloud is highly desirable.

Moreover, it is easy and cheaper to apply transformations, such as rotation and translation, when working with point-clouds than voxelized 3D objects.

As point-cloud data is just a set of points, we can use DeepSets to classify point-cloud representation of a subset of ShapeNet objects (A. X. Chang et al., 2015), called ModelNet40 (Z. Wu et al., 2015). This subset consists of 3D representation of 9,843 training and 2,468 test instances belonging to 40 classes of objects. We produce point-clouds with 100, 1000 and 5000 particles each $(x, y, z$-coordinates) from the mesh representation of objects using the point-cloud-library's sampling routine (Rusu and Cousins, 2011). Each set is normalized by the initial layer of the deep network to have zero mean (along individual axes) and unit (global) variance. Table 2.1 compares our method using three permutation equivariant layers against the competition. Note that we achieve our best accuracy using 5000×3 dimensional representation of each object, which is much smaller than most other methods. All other techniques use either voxelization or multiple view of the 3D object for classification.

TRAINING DETAILS We use a network comprising of 3 permutation-equivariant layers with 256 channels followed by max-pooling over the set structure. The resulting vector representation of the set is then fed to a fully connected layer with 256 units followed by a 40-way softmax unit. We use Tanh activation at all layers and dropout on the layers after set-max-pooling (*i.e.* two dropout operations) with 50% dropout rate. Applying dropout to permutation-equivariant layers for point-cloud data deteriorated the performance. We observed that using different types of permutation-equivariant layers (see Section 2.4) and as few as 64 channels for set layers changes the result by less than 5% in classification accuracy.

For the setting with 5000 particles, we increase the number of units to 512 in all layers and randomly rotate the input around the z-axis. We also randomly scale the point-cloud by $s \sim \mathcal{U}(.8, 1./.8)$. For this setting only, we use Adamax (D. P. Kingma and Ba, 2014) instead of Adam and reduce learning rate from .001 to .0005.

Model	Instance Size	Representation	Accuracy
3DShapeNets (Z. Wu et al., 2015)	30^3	voxels (using convolutional deep belief net)	77%
VoxNet (Maturana and Scherer, 2015)	32^3	voxels (voxels from point-cloud + 3D CNN)	83.10%
MVCNN (Hang Su et al., 2015)	$164 \times 164 \times 12$	multi-vew images (2D CNN + view-pooling)	90.1%
VRN Ensemble (Brock et al., 2016)	32^3	voxels (3D CNN, variational autoencoder)	95.54%
3D GAN (J. Wu et al., 2016)	64^3	voxels (3D CNN, generative adversarial training)	83.3%
DeepSets	$\mathbf{5000} \times 3$	point-cloud	$90 \pm .3\%$
DeepSets	$\mathbf{1000} \times 3$	point-cloud	$87 \pm .1\%$
DeepSets	$\mathbf{100} \times 3$	point-cloud	$82 \pm 2\%$

Table 2.1: Classification accuracy and the representation-size used by different methods on the ModelNet40.

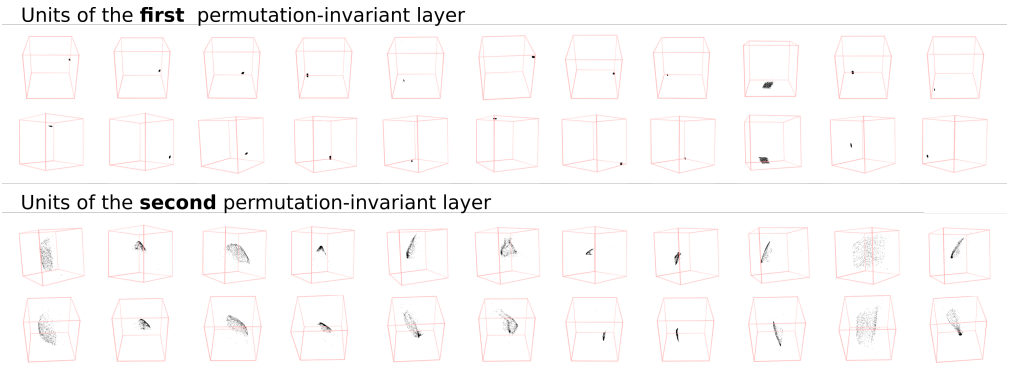


Figure 2.8: Each box is the particle-cloud maximizing the activation of a unit at the firs (**top**) and second (**bottom**) permutation-equivariant layers of our model. Two images of the same column are two different views of the same point-cloud.

Features. To visualize the features learned by the set layers, we used Adamax (D. P. Kingma and Ba, 2014) to locate 1000 particle coordinates maximizing the activation of each unit.[1] Activating the tanh units beyond the second layer proved to be difficult. Figure 2.8 shows the particle-cloud-features learned at the first and second layers of our deep network. We observed that the first layer learns simple localized (often cubic) point-clouds at different (x, y, z) locations, while the second layer learns more complex surfaces with different scales and orientations.

2.5.4 *Set Anomaly Detection*

The objective here is to find the anomalous face in each set, simply by observing examples and without any access to the attribute values. CelebA dataset (Ziwei Liu et al., 2015) contains 202,599 face images, each annotated with 40 boolean attributes. We build $N = 18,000$ sets of 64×64 stamps, using these attributes each containing $M = 16$ images (on the training set) as follows: randomly select 2 attributes, draw 15 images having those attributes, and a single target image where both attributes are absent. Using a similar procedure we build sets on the test images. No individual person's face appears in both train and test sets.

Our deep neural network consists of 9 convolution layers with 3×3 receptive fields. The model has convolution layers with $32, 32, 64$ feature-maps followed by max-pooling followed by 2D convolution layers with $64, 64, 128$ feature-maps followed by another max-pooling layer. The final set of convolution layers have $128, 128, 256$ feature-maps, followed by a max-pooling layer with pool-size of 5 that reduces the output dimension to batch-size.$M \times 256$, where the set-size $M = 16$. This is then forwarded to three permutation-equivariant layers with $256, 128$ and 1 output channels. The output of final layer is fed to the softmax, to identify the outlier. (Note that one could identify arbitrary number of outliers using a sigmoid activation at the output.) We use exponential linear units (Clevert, Unterthiner, and Hochreiter, 2015), drop out with 20% dropout rate at convolutional layers and 50% dropout rate at the first two set layers. When applied to set layers, the selected feature (channel) is simultaneously dropped in all the set members of that particular set. We use Adam (D. P. Kingma and Ba, 2014) for optimization and use batch-normalization only in the convolutional layers. We use mini-batches of 8 sets, for a

1 We started from uniformly distributed set of particles and used a learning rate of .01 for Adamax, with first and second order moment of .1 and .9 respectively. We optimized the input in 10^5 iterations. The results of Figure 2.8 are limited to instances where tanh units were successfully activated. Since the input at the first layer of our deep network is normalized to have a zero mean and unit standard deviation, we do not need to constrain the input while maximizing unit's activation.

Figure 2.9: Each row shows a set, constructed from CelebA dataset, such that all set members except for an outlier, share at least two attributes (on the right). The **outlier is identified with a red frame**. The model is trained by observing examples of sets and their anomalous members, **without access to the attributes**. The probability assigned to each member by the outlier detection network is visualized using a **red bar** at the bottom of each image. The probabilities in each row sum to one.

total of 128 images per batch. Our trained model successfully finds the anomalous face in **75% of test sets**. Visually inspecting these instances suggests that the task is non-trivial even for humans; see Figure 2.9.

As a *baseline*, we repeat the same experiment by using a set-pooling layer after convolution layers, and replacing the permutation-equivariant layers with fully connected layers of same size, where the final layer is a 16-way softmax. The resulting network shares the convolution filters for all instances within all sets, however the input to the softmax is not equivariant to the permutation of input images. Permutation equivariance seems to be crucial here as the baseline model achieves a training and **test accuracy of** $\sim 6.3\%$; the same as random selection.

2.6 SET EXPANSION

In the set expansion task, we are given a set of objects that are similar to each other and our goal is to find new objects from a large pool of candidates such that the selected new objects are similar to the query set. To achieve this one needs to reason out the concept connecting the given set and then retrieve words based on their relevance to the inferred concept. It is an important task due to wide range of potential applications including personalized information retrieval, computational advertisement, tagging large amounts of unlabeled or weakly labeled datasets.

Going back to de Finetti's theorem in Section 2.3.5, where we consider the marginal probability of a set of observations, the marginal probability allows for very simple metric for scoring additional elements to be added to X. In other words, this allows one to perform set expansion via the following score

$$s(x|X) = \log p(X \cup \{x\}|\alpha) - \log p(X|\alpha)p(\{x\}|\alpha) \tag{2.35}$$

Note that $s(x|X)$ is the point-wise mutual information between x and X. Moreover, due to exchangeability, it follows that regardless of the order of elements we have

$$S(X) = \sum_m s\left(x_m|\{x_{m-1}, \dots x_1\}\right) = \log p(X|\alpha) - \sum_{m=1}^{M} \log p(\{x_m\}|\alpha) \tag{2.36}$$

Figure 2.10: Each row shows a set, constructed from CelebA dataset, such that all set members except for an outlier, share at least two attributes (on the right). The **outlier is identified with a red frame**. The model is trained by observing examples of sets and their anomalous members, **without access to the attributes**. The probability assigned to each member by the outlier detection network is visualized using a **red bar** at the bottom of each image.

When inferring sets, our goal is to find set completions $\{x_{m+1}, \ldots x_M\}$ for an initial set of query terms $\{x_1, \ldots, x_m\}$, such that the aggregate set is coherent. This is the key idea of the Bayesian Set algorithm (Zoubin Ghahramani and Heller, 2005).

2.6.1 *Bayes Set*

Bayes Set (Zoubin Ghahramani and Heller, 2005) assumes parametric formulation in the above approach. If we consider exponential families, the above approach assumes a particularly nice form whenever we have conjugate priors. Here we have

$$p(x|\theta) = \exp\left(\langle \phi(x), \theta \rangle - g(\theta)\right) \text{ and } p(\theta|\alpha, M_0) = \exp\left(\langle \theta, \alpha \rangle - M_0 g(\theta) - h(\alpha, M_0)\right). \tag{2.37}$$

The mapping $\phi : x \to \mathcal{F}$ is usually referred as sufficient statistic of x which maps x into a feature space $\mathcal{F}$. Moreover, $g(\theta)$ is the log-partition (or cumulant-generating) function. Finally, $p(\theta|\alpha, M_0)$ denotes the conjugate distribution which is in itself a member of the exponential family. It has the normalization $h(\alpha, M_0) = \int d\theta \exp\left(\langle \theta, \alpha \rangle - M_0 g(\theta)\right)$. The advantage of this

is that $s(x|X)$ and $S(X)$ can be computed in closed form (Zoubin Ghahramani and Heller, 2005) via

$$s(X) = h\left(\alpha + \phi(X), M_0 + M\right) + (M-1)h(\alpha, M_0) - \sum_{m=1}^{M} h(\alpha + \phi(x_m), M+1) \tag{2.38}$$

$$\begin{aligned} s(x|X) = {} & h\left(\alpha + \phi(\{x\} \cup X), M_0 + M + 1\right) + h(\alpha, M_0) \\ & - h\left(\alpha + \phi(X), M_0 + M\right) - h(\alpha + \phi(x), M+1) \end{aligned} \tag{2.39}$$

For convenience we defined the sufficient statistic of a set to be the sum over its constituents, i.e. $\phi(X) = \sum_m \phi(x_m)$. It allows for very simple computation and maximization over additional elements to be added to X, since $\phi(X)$ can be precomputed.

2.6.1.1 *Beta-Binomial Model*

The model is particularly simple when dealing with the Binomial distribution and its conjugate Beta prior, since the ratio of Gamma functions allows for simple expressions. In particular, we have

$$h(\beta) = \log\Gamma(\beta^+) + \log\Gamma(\beta^-) - \Gamma(\beta). \tag{2.40}$$

With some slight abuse of notation we let $\alpha = (\beta^+, \beta^-)$ and $M_0 = \beta^+ + \beta^-$. Setting $\phi(1) = (1, 0)$ and $\phi(0) = (0, 1)$ allows us to obtain $\phi(X) = (M^+, M^-)$, i.e. $\phi(X)$ contains the counts of occurrences of $x_m = 1$ and $x_m = 0$ respectively. This leads to the following score functions

$$\begin{aligned} s(X) = {} & \log\Gamma(\beta^+ + M^+) + \log\Gamma(\beta^- + M^-) - \log\Gamma(\beta + M) \\ & - \log\Gamma(\beta^+) - \log\Gamma(\beta^-) + \log\Gamma(\beta) - M^+ \log\frac{\beta^+}{\beta} - M^- \log\frac{\beta^-}{\beta} \end{aligned} \tag{2.41}$$

$$s(x|X) = \begin{cases} \log\frac{\beta^+ + M^+}{\beta + M} - \log\frac{\beta^+}{\beta} & \text{if } x = 1 \\ \log\frac{\beta^- + M^-}{\beta + M} - \log\frac{\beta^-}{\beta} & \text{otherwise} \end{cases} \tag{2.42}$$

This is the model used by Zoubin Ghahramani and Heller (2005) when estimating Bayesian Sets for objects. In particular, they assume that for any given object x the vector $\phi(x) \in \{0; 1\}^d$ is a d-dimensional binary vector, where each coordinate is drawn independently from some Beta-Binomial model. The advantage of the approach is that it can be computed very efficiently while only maintaining minimal statistics of X.

In a nutshell, the *algorithmic* operations performed in the Beta-Binomial model are as follows:

$$s(x|X) = 1^\top \left[\sigma\left(\sum_{m=1}^{M} \phi(x_m) + \phi(x) + \beta\right) - \sigma\left(\phi(x) + \beta\right)\right] \tag{2.43}$$

In other words, we sum over statistics of the candidates x_m, add a bias term β, perform a *coordinate-wise* nonlinear transform over the aggregate statistic (in our case a logarithm), and finally we aggregate over the so-obtained scores, weighing each contribution equally. $s(X)$ is expressed analogously.

2.6.1.2 *Gauss Inverse Wishart Model*

Before abstracting away the probabilistic properties of the model, it is worth paying some attention to the case where we assume that $x_i \sim \mathcal{N}(\mu, \Sigma)$ and $(\mu, \Sigma) \sim \text{NIW}(\mu_0, \lambda, \Psi, \nu)$, for a suitable set of conjugate parameters. While the details are (arguably) tedious, the overall structure of the model is instructive.

First note that the sufficient statistic of the data $x \in \mathbb{R}^d$ is now given by $\phi(x) = (x, xx^\top)$. Secondly, note that the conjugate log-partition function h amounts to computing *determinants* of terms involving $\sum_m x_m x_m^\top$ and moreover, nonlinear combinations of the latter with $\sum_m x_m$.

The *algorithmic* operations performed in the Gauss Inverse Wishart model are as follows:

$$s(x|X) = \sigma\left(\sum_{m=1}^{M} \phi(x_m) + \phi(x) + \beta\right) - \sigma\left(\phi(x) + \beta\right) \tag{2.44}$$

Here σ is a nontrivial convex function acting on a (matrix, vector) pair and $\phi(x)$ is no longer a trivial map but performs a nonlinear dimension altering transformation on x. We will use this general template to fashion the DeepSets algorithm.

2.6.2 *Using DeepSets*

Using DeepSets, we can solve this problem in more generality as we can drop the assumption of data belonging to certain exponential family.

For learning the score $s(x|X)$, we take recourse to large-margin classification with structured loss functions (Taskar, Guestrin, and Koller, 2004) to obtain the relative loss objective

$$l(x, x'|X) = \max(0, s(x'|X) - s(x|X) + \Delta(x, x')). \tag{2.45}$$

In other words, we want to ensure that $s(x|X) \geqslant s(x'|X) + \Delta(x, x')$ whenever x should be added and x' should not be added to X.

CONDITIONING Often machine learning problems do not exist in isolation. For example, task like tag completion from a given set of tags is usually related to an object z, for example an image, that needs to be tagged. Such meta-data are usually abundant, *e.g.* author information in case of text, contextual data such as the user click history, or extra information collected with LiDAR point cloud.

Conditioning graphical models with meta-data is often complicated. For instance, in the Beta-Binomial model we need to ensure that the counts are always nonnegative, regardless of z. Fortunately, DeepSets does not suffer from such complications and the fusion of multiple sources of data can be done in a relatively straightforward manner. Any of the existing methods in deep learning, including feature concatenation by averaging, or by max-pooling, can be employed. Incorporating these meta-data often leads to significantly improved performance as will be shown in experiments; Section 2.7.2.

2.7 EXPERIMENTS ON SET EXPANSION

2.7.1 *Text Concept Set Retrieval*

We consider the task of text concept set retrieval, where the objective is to retrieve words belonging to a 'concept' or 'cluster', given few words from that particular concept. For example, given the set of words {*tiger*, *lion*, *cheetah*}, we would need to retrieve other related words like *jaguar*, *puma*, *etc*, which belong to the same concept of big cats. The model implicitly needs to reason out the concept connecting the given set and then retrieve words based on their relevance to the inferred concept. Concept set retrieval is an important due to wide range of potential applications including personalized information retrieval, tagging large amounts of unlabeled or weakly labeled datasets, *etc*. This task of concept set retrieval can be seen as a set completion task conditioned on the latent semantic concept, and therefore our DeepSets form a desirable approach.

Method	LDA-1k (Vocab = 17k)					LDA-3k (Vocab = 38k)					LDA-5k (Vocab = 61k)				
	Recall (%)			MRR	Med.	Recall (%)			MRR	Med.	Recall (%)			MRR	Med.
	@10	@100	@1k			@10	@100	@1k			@10	@100	@1k		
Random	0.06	0.6	5.9	0.001	8520	0.02	0.2	2.6	0.000	28635	0.01	0.2	1.6	0.000	30600
Bayes Set	1.69	11.9	37.2	0.007	2848	2.01	14.5	36.5	0.008	3234	1.75	12.5	34.5	0.007	3590
w2v Near	**6.00**	**28.1**	**54.7**	0.021	**641**	4.80	21.2	43.2	0.016	2054	4.03	16.7	35.2	0.013	6900
NN-max	4.78	22.5	53.1	0.023	779	5.30	24.9	54.8	0.025	672	4.72	21.4	47.0	0.022	1320
NN-sum-con	4.58	19.8	48.5	0.021	1110	5.81	27.2	60.0	**0.027**	453	4.87	23.5	53.9	0.022	731
NN-max-con	3.36	16.9	46.6	0.018	1250	5.61	25.7	57.5	0.026	570	4.72	22.0	51.8	0.022	877
DeepSets	5.53	24.2	54.3	**0.025**	696	**6.04**	**28.5**	**60.7**	**0.027**	**426**	**5.54**	**26.1**	**55.5**	**0.026**	**616**

Table 2.2: Results on Text Concept Set Retrieval on LDA-1k, LDA-3k, and LDA-5k. Our DeepSets model outperforms other methods on LDA-3k and LDA-5k. However, all neural network based methods have inferior performance to w2v-Near baseline on LDA-1k, possibly due to small data size. Higher the better for recall@k and mean reciprocal rank (MRR). Lower the better for median rank (Med.)

DATASET To construct a large dataset containing sets of related words, we make use of Wikipedia text due to its huge vocabulary and concept coverage. First, we run topic modeling on publicly available wikipedia text with K number of topics. Specifically, we use the famous latent Dirichlet allocation (D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003; Pritchard, Stephens, and Donnelly, 2000), taken out-of-the-box[2]. Next, we choose top $N_T = 50$ words for each latent topic as a set giving a total of K sets of size N_T. To compare across scales, we consider three values of $k = \{1k, 3k, 5k\}$ giving us three datasets LDA-1k, LDA-3k, and LDA-5k, with corresponding vocabulary sizes of 17k, 38k, and 61k. Few of the topics from LDA-1k are visualized in Table 2.11.

METHODS Our DeepSets model uses a feedforward neural network (NN) to represent a query and each element of a set, *i.e.*, $\phi(x)$ for an element x is encoded as a NN. Specifically, $\phi(x)$ represents each word via 50-dimensional embeddings that are we learn jointly, followed by two fully connected layers of size 150, with ReLU activations. We then construct a set representation or feature, by sum pooling all the individual representations of its elements, along with that of the query. Note that this sum pooling achieves permutation invariance, a crucial property of our DeepSets (Theorem 2.5). Next, use input this set feature into another NN to assign a single score to the set, shown as $\rho(.)$. We instantiate $\rho(.)$ as three fully connected layers of sizes $\{150, 75, 1\}$ with ReLU activations. In summary, our DeepSets consists of two neural networks – (a) to extract representations for each element, and (b) to score a set after pooling representations of its elements.

BASELINES We compare to several baselines: (a) **Random** picks a word from the vocabulary uniformly at random. (b) **Bayes Set** (Zoubin Ghahramani and Heller, 2005), and (c) **w2v-Near** that computes the nearest neighbors in the word2vec (Mikolov, Sutskever, et al., 2013) space. Note that both Bayes Set and w2v NN are strong baselines. The former runs Bayesian inference using Beta-Binomial conjugate pair, while the latter uses the powerful 300 dimensional word2vec trained on the billion word GoogleNews corpus[3]. (d) **NN-max** uses a similar architecture as our DeepSets with an important difference. It uses max pooling to compute the set

2 github.com/dmlc/experimental-lda
3 code.google.com/archive/p/word2vec/

feature, as opposed to DeepSets which uses sum pooling. (e) **NN-max-con** uses max pooling on set elements but concatenates this pooled representation with that of query for a final set feature. (f) **NN-sum-con** is similar to NN-max-con but uses sum pooling followed by concatenation with query representation.

EVALUATION To quantitatively evaluate, we consider the standard retrieval metrics – recall@K, median rank and mean reciprocal rank. To elaborate, recall@K measures the number of true labels that were recovered in the top K retrieved words. We use three values of $K = \{10, 100, 1k\}$. The other two metrics, as the names suggest, are the median and mean of reciprocals of the true label ranks, respectively. Each dataset is split into TRAIN (80%), VAL (10%) and TEST (10%). We learn models using TRAIN and evaluate on TEST, while VAL is used for hyperparameter selection and early stopping.

RESULTS AND OBSERVATIONS Table 2.2 contains the results for the text concept set retrieval on LDA-1k, LDA-3k, and LDA-5k datasets. We summarize our findings below: (a) Our /deepsets model outperforms all other approaches on LDA-3k and LDA-5k by any metric, highlighting the significance of permutation invariance property. For instance, /deepsets is better than the w2v-Near baseline by 1.5% in Recall@10 on LDA-5k. (b) On LDA-1k, neural network based models do not perform well when compared to w2v-Near. We hypothesize that this is due to small size of the dataset insufficient to train a high capacity neural network, while w2v-Near has been trained on a billion word corpus. Nevertheless, our approach comes the closest to w2v-Near amongst other approaches, and is only 0.5% lower by Recall@10.

Topic 1	**Topic 2**	**Topic 3**	**Topic 4**	**Topic 5**	**Topic 6**
legend	president	plan	newspaper	round	point
airy	vice	proposed	daily	teams	angle
tale	served	plans	paper	final	axis
witch	office	proposal	news	played	plane
devil	elected	planning	press	redirect	direction
giant	secretary	approved	published	won	distance
story	presidency	planned	newspapers	competition	surface
folklore	presidential	development	editor	tournament	curve

Figure 2.11: Examples from our LDA-1k datasets. Notice that each of these are latent topics of LDA and hence are semantically similar.

2.7.2 *Image Tagging*

We next experiment with image tagging, where the task is to retrieve all relevant tags corresponding to an image. Images usually have only a subset of relevant tags, therefore predicting other tags can help enrich information that can further be leveraged in a downstream supervised task. In our setup, we learn to predict tags by conditioning /deepsets on the image. Specifically, we train by learning to predict a partial set of tags from the image and remaining tags. At test time, we the test image is used to predict relevant tags.

DATASETS We report results on the following three datasets:
(a) *ESPgame* (Von Ahn and Dabbish, 2004): Contains around 20k images spanning logos, drawings, and personal photos, collected interactively as part of a game. There are a total of 268 unique tags, with each image having 4.6 tags on average and a maximum of 15 tags.

Method	ESP game				IAPRTC-12.5			
	P	R	F1	N+	P	R	F1	N+
Least Sq.	35	19	25	215	40	19	26	198
MBRM	18	19	18	209	24	23	23	223
JEC	24	19	21	222	29	19	23	211
FastTag	46	22	30	**247**	**47**	26	34	**280**
Least Sq.(D)	**44**	32	**37**	232	46	30	36	218
FastTag(D)	**44**	32	**37**	229	46	**33**	**38**	254
DeepSets	39	**34**	36	246	42	31	36	247

Table 2.3: Results of image tagging on ESPgame and IAPRTC-12.5 datasets. Performance of our DeepSets approach is roughly similar to the best competing approaches, except for precision. Refer text for more details. Higher the better for all metrics – precision (P), recall (R), f1 score (F1), and number of non-zero recall tags (N+).

(b) *IAPRTC-12.5* (Grubinger, 2007): Comprises of around 20k images including pictures of different sports and actions, photographs of people, animals, cities, landscapes, and many other aspects of contemporary life. A total of 291 unique tags have been extracted from captions for the images. For the above two datasets, train/test splits are similar to those used in previous works (M. Chen, A. Zheng, and Weinberger, 2013; Guillaumin et al., 2009).
(c) *COCO-Tag:* We also construct a dataset in-house, based on MSCOCO dataset (T.-Y. Lin et al., 2014). COCO is a large image dataset containing around 80k train and 40k test images, along with five caption annotations. We extract tags by first running a standard spell checker[4] and lemmatizing these captions. Stopwords and numbers are removed from the set of extracted tags. Each image has 15.9 tags on an average and a maximum of 46 tags. We show examples of image tags from COCO-Tag in Figure 2.12. The advantages of using COCO-Tag are three fold–richer concepts, larger vocabulary and more tags per image, making this an ideal dataset to learn image tagging using /deepsets.

IMAGE AND WORD EMBEDDINGS Our models use features extracted from Resnet, which is the state-of-the-art convolutional neural network (CNN) on ImageNet 1000 categories dataset using the publicly available 152-layer pretrained model[5]. To represent words, we jointly learn embeddings with the rest of /deepsets neural network for ESPgame and IAPRTC-12.5 datasets. But for COCO-Tag, we bootstrap from 300 dimensional word2vec embeddings[6] as the vocabulary for COCO-Tag is significantly larger than both ESPgame and IAPRTC-12.5 (13k vs 0.3k).

METHODS The setup for DeepSets to tag images is similar to that described in Section 2.7.1. The only difference being the conditioning on the image features, which is concatenated with the set feature obtained from pooling individual element representations. In particular, $\phi(x)$ represents each word via 300-dimensional word2vec embeddings, followed by two fully connected layers of size 300, with ReLU activations, to construct the set representation or features. As mentioned earlier, we concatenate the image features and pass this set features into another NN to assign a single score to the set, shown as $\rho(.)$. We instantiate $\rho(.)$ as three fully connected layers of sizes $\{300, 150, 1\}$ with ReLU activations. The resulting feature forms the new input to a neural network used to score the set, in this case, score the relevance of a tag to the image.

4 http://hunspell.github.io/
5 github.com/facebook/fb.resnet.torch
6 https://code.google.com/p/word2vec/

Method	Recall @10	Recall @100	Recall @1k	MRR	Med.
w2v NN (blind)	5.6	20.0	54.2	0.021	823
DeepSets (blind)	9.0	39.2	71.3	0.044	310
DeepSets	**31.4**	**73.4**	**95.3**	**0.131**	**28**

Table 2.4: Results on COCO-Tag dataset. Clearly, DeepSets outperforms other baselines significantly. Higher the better for recall@K and mean reciprocal rank (MRR). Lower the better for median rank (Med).

BASELINES We perform comparisons against several baselines, previously reported from M. Chen, A. Zheng, and Weinberger (2013). Specifically, we have Least Sq., a ridge regression model, MBRM (S. L. Feng, Manmatha, and Lavrenko, 2004), JEC (Makadia, Pavlovic, and S. Kumar, 2008) and FastTag (M. Chen, A. Zheng, and Weinberger, 2013). Note that these methods do not use deep features for images, which could lead to an unfair comparison. As there is no publicly available code for MBRM and JEC, we cannot get performances of these models with Resnet extracted features. However, we report results with deep features for FastTag and Least Sq., using code made available by the authors [7].

EVALUATION For ESPgame and IAPRTC-12.5, we follow the evaluation metrics as in Guillaumin et al. (2009) – precision (P), recall (R), F1 score (F1) and number of tags with non-zero recall (N+). Note that these metrics are evaluate for each tag and the mean is reported. We refer to Guillaumin et al. (2009) for further details. For COCO-Tag, however, we use recall@K for three values of $K = \{10, 100, 1000\}$, along with median rank and mean reciprocal rank (see evaluation in Section 2.7.1 for metric details).

RESULTS AND OBSERVATIONS Table 2.3 contains the results of image tagging on ESPgame and IAPRTC-12.5, and Table 2.4 on COCO-Tag. Here are the key observations from Table 2.3: (a) The performance of DeepSets is comparable to the best of other approaches on all metrics but precision. (b) Our recall beats the best approach by 2% in ESPgame. On further investigation, we found that DeepSets retrieves more relevant tags, which are not present in list of ground truth tags due to a limited 5 tag annotation. Thus, this takes a toll on precision while gaining on recall, yet yielding improvement in F1. On the larger and richer COCO-Tag, we see that DeepSets approach outperforms other methods comprehensively, as expected. We show qualitative examples in Figure 2.12.

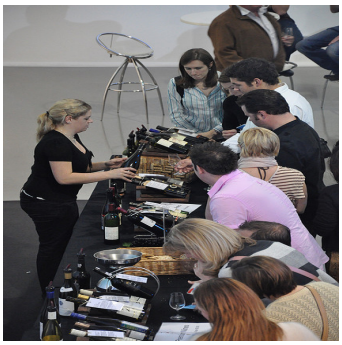
We present examples of our in-house tagging datasets, COCO-Tag in Figure 2.12.

2.8 SET GENERATION

A fundamental problem in machine learning is to develop efficient methods that can learn a generative model of the distribution of the provided training set and *capable of generating arbitrary many new sample points from the domain of this distribution* (Christopher, 2016). Deep generative models use deep neural networks as a tool for learning complex distributions. These approaches include variational autoencoders (D. Kingma and Welling, 2014), autoregressive models (Oord, Kalchbrenner, and Kavukcuoglu, 2016) and generative adversarial networks (GAN) (Goodfellow et al., 2014). The compelling results have been demonstrated on simple

7 http://www.cse.wustl.edu/~mchen/

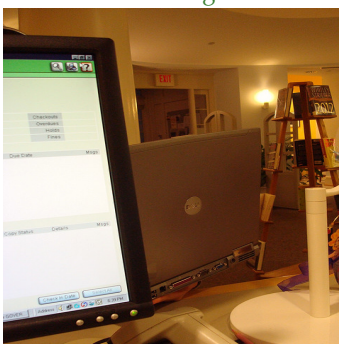
GT	Pred
building	building
sign	street
brick	city
picture	brick
empty	sidewalk
white	side
black	pole
street	white
image	stone



GT	Pred
standing	person
surround	group
woman	man
crowd	table
wine	sit
person	room
group	woman
table	couple
bottle	gather

GT	Pred
traffic	clock
city	tower
building	sky
tall	building
large	tall
tower	large
European	cloudy
front	front
clock	city

GT	Pred
photograph	ski
snowboarder	snow
snow	slope
glide	person
hill	snowy
show	hill
person	man
slope	skiing
young	skier



GT	Pred
laptop	refrigerator
person	fridge
screen	room
room	magnet
desk	cabinet
living	kitchen
counter	shelf
computer	wall
monitor	counter

GT	Pred
beach	jet
shoreline	airplane
stand	propeller
walk	ocean
sand	plane
lifeguard	water
white	body
person	person
surfboard	sky

Figure 2.12: Qualitative examples of image tagging using DeepSets. *Top row*: Positive examples where most of the retrieved tags are present in the ground truth (brown) or are relevant but not present in the ground truth (green). *Bottom row*: Few failure cases with irrelevant/wrong tags (red). From left to right, (i) Confusion between snowboarding and skiing, (ii) Confusion between back of laptop and refrigerator due to which other tags are kitchen-related, (iii) Hallucination of airplane due to similar shape of surfboard.

data types like images, speech, and text (Denton, Chintala, Fergus, et al., 2015; Finn, Goodfellow, and Levine, 2016; Karras et al., 2017; Lamb et al., 2016; Radford, Metz, and Chintala, 2015; Van Den Oord et al., 2016). Their wide range of applicability was also demonstrated in many important problems, including data augmentation (Salimans et al., 2016), image style transformation (J.-Y. Zhu et al., 2017), image captioning (Kelvin Xu et al., 2015) and art creations (Kang, 2017).

Hoowever, there are many unexplored, yet useful complex data type, like sets which has been theme of this chapter. In particular set of 3D points, such as CAD designs, 3D meshes, and point clouds, have recently garnered a lot of attention because various easily accessible sensors like LiDAR on self-driving cars, Kinect for xbox, and face identification sensor on phones. Similar to useful applications of generative models previous works on simple data type, we want to extend to set data.

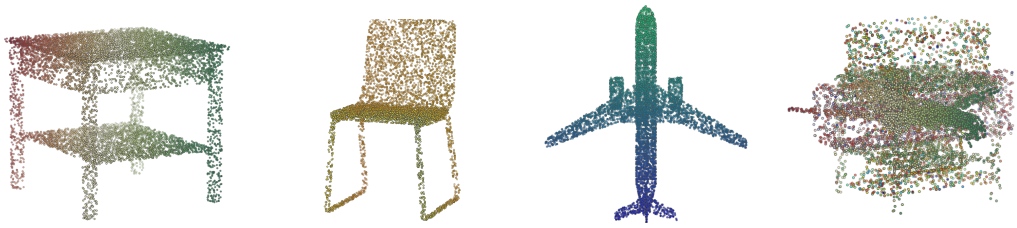
We propose a deep generative adversarial network for set data, in particular for point clouds (PC-GAN). The proposed architecture learns a stochastic procedure which can draw samples for point clouds without explicitly modeling the underlying density function.

2.8.1 *Difficulty of the Task*

We begin by defining the problem and ideal generative process for point cloud over objects. Formally, a point cloud for an object is nothing but a *set* of n low dimensional vectors $X = \{x_1, ..., x_n\}$, where n can be infinite and typically each $x_i \in \mathbb{R}^3$ or $x_i \in \mathbb{R}^2$. Point cloud for M different objects is a collection of M sets $X^{(1)}, ..., X^{(M)}$, where each $X^{(m)}$ is as defined above. Thus, we basically need a generative model $p(X)$ for sets which should be able to:

- Sample entirely new sets X, as well as
- Sample more points for a given set, *i.e.* $x \sim p(x|X)$.

De-Finetti theorem allows us to express the set probability in a factored format as $p(X) = \int_\theta \prod_{i=1}^{n} p(x_i|\theta)p(\theta)d\theta$ for some suitably defined θ. In case of point clouds, the latent variable θ can be interpreted as an object representation. In this view, the factoring can be understood as follows: Given an object, θ, the points x_i in the point cloud can be considered as i.i.d. samples



(a) Table: $p(x_i|\theta_1)$ (b) Chair: $p(x_i|\theta_2)$ (c) Plane: $p(x_i|\theta_3)$ (d) Marginalized: $p(x_i)$

Figure 2.13: Unlike in case of images, the marginal distribution $p(x_i) = \int p(x_i|\theta)p(\theta)d\theta$ is not useful. We have to learn the joint distribution $p(X, \theta)$. An illustrative parallel can be drawn between pixels of an image and point cloud — marginal distribution of pixels is quite uninformative, one has to consider the joint distribution of all pixels in one image (not across different images).

from $p(x|\theta)$, an unknown latent distribution representing the object θ. The joint likelihood can be expressed as:

$$p(X, \theta) = \underbrace{p(\theta)}_{\text{object}} \underbrace{\prod_{i=1}^{n} p(x_i|\theta)}_{\text{points for object}} \tag{2.46}$$

Thus, the task boils down to modeling these two probabilities.

Attempts have been made to characterize these conditional probabilities with parametric models like Gaussian Mixture Models or parametric hierarchical models (Eckart et al., 2015; Jian and Vemuri, 2005; Strom, A. Richardson, and Olson, 2010). However, such approaches have limited success as the point cloud conditional density $p(x|\theta)$ is highly non-linear and complicated (example of point clouds can be seen in Figure 2.15, 2.18).

With advent of implicit generative models, like GANs (Goodfellow et al., 2014), it is possible to model complicated distributions, but it is not clear how to extend the existing GAN frameworks to handle the hierarchical model of point clouds as developed above. The aim of most GAN framework(Arjovsky, Chintala, and Léon Bottou, 2017; Goodfellow et al., 2014; C.-L. Li et al., 2017) is to learn to generate new samples from a distribution $p(x)$, for fixed dimensional $x \in \mathbb{R}^d$, given a training dataset. We, in contrast, aim to develop a new method where the training data is a set of sets, and learning from this training data would allow us to generate points from this hierarchical distribution (e.g. point clouds of 3D objects). To re-emphasize the incompatibility, note that the training data of traditional GANs is a *set* of fixed dimensional instances, in our case, however, it is a *set of sets*. Using existing GANs to just learning marginal distribution $p(x)$, in case of point clouds x, is not of much use because the marginal distribution is quite uninformative as can be seen from Figure 2.13. GANs have been extended to sample from conditional distribution (Isola et al., 2017; Mirza and Osindero, 2014), but they require conditioning variable to be observed, such as the label. While in case of point clouds we only have partial knowledge of the conditional, *i.e.* we only have groupings of point coming from the same object but we have no representation of the conditional or the object other than the points themselves.

Another approach can be to model the distribution of the point cloud set together, i.e., $\{\{x_i^{(1)}\}_{i=1}^n, \ldots, \{x_i^{(m)}\}_{i=1}^n\}$. In this setting, a naíve application of traditional GAN is possible through treating the point cloud as finite dimensional vector by fixing number and order of the points, i.e. reducing the problem to instances in $\mathbb{R}^{n\times 3}$. Again, in this setting another natural extension of traditional GAN framework is to use a DeepSets classifier as the discriminator to distinguish real sets vs fake sets. However, it would not work because the IPM guarantees behind the traditional GAN do not hold any longer (*e.g.* in case of Arjovsky, Chintala, and Léon Bottou (2017), even Lipschitz-1 function over sets is not well defined) and counter examples for breaking IPM guarantees can be easily found. We study a counter example next.

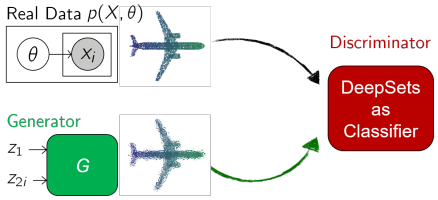


Figure 2.14: Natural extension of GAN to handle set data

COUNTER EXAMPLE Consider a simple GAN (Goodfellow et al., 2014) with a DeepSets classifier as the discriminator. In order to generate coherent sets of variable size, we consider a generator G having two noise sources: z_1 and z_{2i}. To generate a set, z_1 is sampled once and z_{2i} is sampled for $i = 1, 2, ..., n$ to produce n points in the generated set. Intuitively, fixing the first

noise source z_1 selects a set and ensures the points generated by repeated sampling of z_{2i} are coherent and belong to same set. The setup is depicted in Figure 2.14. In this setup, the GAN minimax problem would be:

$$\min_G \max_D \mathop{\mathbb{E}}_{\substack{\theta\sim p(\theta)\\ x_i\sim p(x_i|\theta)}} [\log D(\{x_i\})] + \mathop{\mathbb{E}}_{\substack{z_1\sim q_1(z_1)\\ z_{2i}\sim q_2(z_{2i})}} [\log(1 - D(\{G(z_1, z_{2i})\}))] \tag{2.47}$$

Now consider the case, when there exist an 'oracle' mapping T which maps the set deterministically to the object identified, i.e. $\exists T : T(\{x_i\}) = \theta$. An example is when different θ leads to conditional distribution $p(x|\theta)$ with non-overlapping support. Let $D = D' \circ T$ and G ignores z_2 then optimization becomes

$$\begin{aligned}
&\min_G \max_{D'} \mathop{\mathbb{E}}_{\substack{\theta\sim p(\theta)\\ x_i\sim p(x_i|\theta)}} \left[\log D'(T(\{x_i\}))\right] + \mathop{\mathbb{E}}_{\substack{z_1\sim q_1(z_1)\\ z_{2i}\sim q_2(z_{2i})}} \left[\log\left(1 - D'(T(\{G(z_1, z_{2i})\}))\right)\right] \\
\Rightarrow &\min_G \max_{D'} \mathop{\mathbb{E}}_{\substack{\theta\sim p(\theta)\\ x_i\sim p(x_i|\theta)}} \left[\log D'(\theta)\right] + \mathop{\mathbb{E}}_{\substack{z_1\sim q_1(z_1)\\ z_{2i}\sim q_2(z_{2i})}} \left[\log\left(1 - D'(T(\{G(z_1)\}))\right)\right] \\
\Rightarrow &\min_G \max_{D'} \mathop{\mathbb{E}}_{\theta\sim p(\theta)} \left[\log D'(\theta)\right] + \mathop{\mathbb{E}}_{z_1\sim q_1(z_1)} \left[\log\left(1 - D'(T(\{G(z_1)\}))\right)\right]
\end{aligned} \tag{2.48}$$

Thus, we can achieve the lower bound $-\log(4)$ by only matching the $p(\theta)$ component, while the conditional $p(x|\theta)$ is allowed to remain arbitrary. So simply using DeepSets classifier in simple GAN in order to handle sets does not lead to a valid generative model.

2.8.2 *Proposed Method*

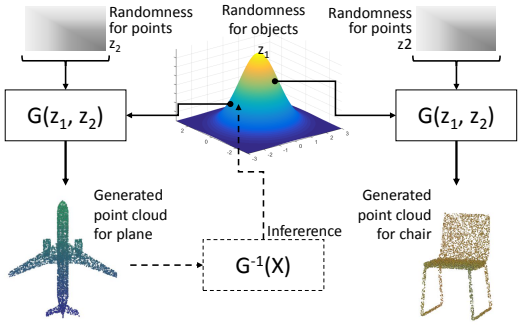


Figure 2.15: Overview

We present a solution to this problem by learning a two-level implicit generative models for modeling the sampling process of $p(x|\theta)$ and $p(\theta)$. In Isola et al. (2017) and Mirza and Osindero (2014), the θ is a simple one-hot vector of class labels or a given image for transformations. However, naively modeling θ to be a one-hot vector to indicate which object can not be generalized to unseen testing data. Instead, different from Isola et al. (2017) and Mirza and Osindero (2014), our θ is an unobserved latent variable for modeling different objects with semantic meaning.

Our solution comprises of a generator $G_x(z, \psi)$ which takes in a noise source $z \in \mathbb{R}^{d_1}$ and a descriptor $\psi \in \mathbb{R}^{d_2}$ encoding information about distribution of θ. For a given θ_0, the descriptor ψ would encode information about the distribution $\delta(\theta - \theta_0)$ and samples generated as $x = G_x(z, \psi)$ would follow the distribution $p(x|\theta_0)$. More generally, ψ can used to encode more complicated distributions regarding θ as well. In particular, it could be used to encode the posterior $p(\theta|X)$ for a given sample set X in order to produce samples that follow the posterior predictive distribution:

$$p(x|X) = \int p(x|\theta)p(\theta|X)d\theta. \tag{2.49}$$

We can train such a generator G_x using a GAN-like objective, for which we would need two more components. First, we would need a discriminator $D_\theta(\cdot)$ which distinguishes between the generated samples and true samples conditioned on θ. Second, as ψ is unobserved, we

would need an inference network $Q(X)$ which would try to predict the descriptor ψ about the distribution $p(\theta|X)$ given the sample set X. Under this formulation, if we use an IPM-based GAN (Arjovsky, Chintala, and Léon Bottou, 2017; Mroueh, C.-L. Li, et al., 2017; Mroueh and Sercu, 2017), the objective can be written as

$$\mathbb{E}_{\theta\sim p(\theta)}\left[\min_{D_\theta\in\Omega_{D_\theta}}\max_{G_x,Q}\mathbb{E}_{X\sim p(X|\theta)}\left[D_\theta(X)\right]-\mathbb{E}_{z\sim p(z),X\sim p(X|\theta)}\left[D_\theta(G_x(z,Q(X)))\right]\right] \tag{2.50}$$

where Ω_D is the constraint for different probabilistic distances, such as 1-Lipschitz (Arjovsky, Chintala, and Léon Bottou, 2017), L^2 ball (Mroueh and Sercu, 2017) or Sobolev ball (Mroueh, C.-L. Li, et al., 2017).

A major hurdle in taking this path is that X is a set of points, which can vary in size and permutation of elements. Thus, making design of Q complicated as traditional neural network can not handle this and possibly is the reason for absence of such framework in the literature despite being a natural solution for the important problem of generative modeling of point clouds. However, we can overcome this challenge and We propose to construct the inference network by utilizing the permutation equivariant layers from DeepSets as presented in Section 2.4.1.2. This allows it handle variable number of inputs points in arbitrary order, yet yielding a consistent descriptor ψ.

After training $G_x(\cdot,\theta)$ and the inference network Q, we use trained Q to collect inferred $\psi = Q(x)$ and train another generator $G_\psi(z)$ with a different noise source $z \in \mathbb{R}^{d_3}$ such that samples correspond to $p(\theta)$ to enable hierarchical sampling. Moreover, the two generators can be combined and viewed as $G(z_1,z_2) = G_x(z_2,G_\psi(z_1))$ for carrying out the hierarchical sampling at one go. In addition to the layer-wise training, a joint training may further boost the performance.

2.8.3 *Tighter Solutions via Sandwiching*

In our setting each point x_i in the point cloud can be considered to correspond to single images in case of training GANs over images. When multiple images are sampled from a GAN on images, many great images are generated, but still some not so good images can be generated. An example is illustrated in Figure 2.16 where samples from MMD-GAN (C.-L. Li et al., 2017) trained is on celebA faces consists of both good and bad faces. In case of images, when quality is evaluated, it primarily focuses on coherence individual images and the few bad ones can easily be left out. Whereas in case of point cloud, to get representation of an object we need many points together and presence of outlier points degrades the quality of the object. Thus, when training a generative model for point cloud, we need to ensure a much lower distance $D(P||P_G)$ between true distribution P and generator distribution P_G than would be needed in case of images.

We begin by noting that the popular Wasserstein GAN (Arjovsky, Chintala, and Léon Bottou, 2017), tries to optimize G for reducing the Wasserstein distance $w(P,G)$ between the truth P and generated distribution G, i.e. one aims to solve

$$\min_G w(P,G) \tag{2.51}$$

Direct evaluation of wasserstein distance is computationally very expensive in general (Arjovsky, Chintala, and Léon Bottou,

Figure 2.16: Faces generated from a GAN

2017). To practically solve this problem, the dual form of Wasserstein distance, is invoked, which is

$$w(P, G) = \sup_{\|f\|_L \leqslant 1} \mathbb{E}_{x\sim P} f(x) - \mathbb{E}_{x\sim G} f(x), \tag{2.52}$$

where $\|f\|_L \leqslant 1$ is the set of functions whose Lipschitz constant is no larger than 1. Then in practice, deep neural networks with parameters ϕ and representing function f_ϕ with weight clipping (Arjovsky, Chintala, and Léon Bottou, 2017) or gradient penalty (Gulrajani et al., 2017) have been used to approximate $\|f\|_L \leqslant 1$, i.e. solve

$$\hat{w}(P, G) = \max_{\phi} \mathbb{E}_{x\sim P} f_\phi(x) - \mathbb{E}_{x\sim G} f_\phi(x) \tag{2.53}$$

But this yields a lower bound on the Wasserstein distance, as $\hat{w}(P, G) \leqslant w(P, G)$. When solving a minimization problem, it is known that optimizing the lower bound is not an ideal idea always.

On the other hand, there have been developed $1 + \epsilon$ approximations $w_\epsilon(P, G)$ to Wasserstein distance between to sample sets X and Y, where

$$w(P, G) \leqslant w_\epsilon(P, G) \leqslant (1 + \epsilon) w(P, G),$$

which is an upper bound to Wassertein distance (Bertsekas, 1985; H. Fan, Hao Su, and Guibas, n.d.). We propose to combine both the lower bound and upper bound, in order to sandwich the solution between the two, i.e. we solve the following minimization problem:

$$\begin{aligned} \min_G \quad & w_\epsilon(P, G) \\ \text{s.t.} \quad & w_\epsilon(P, G) - \hat{w}(P, G) < \lambda \end{aligned} \tag{2.54}$$

This optimization problem be solved using Lagrange multipliers. In our experiments we use this formulation and see significantly better recovery of true distribution.

2.8.4 *Simplified Theoretical Justification*

We want to understand what happens to GANs under weak discriminators (lower bounds) from statistical perspective. Ignoring the implementation of these models, from the perspective of statistical analysis, generative model using GANs are different from classical density estimators as loss is measured not with $\mathcal{L}^p$ distances (as is conventional in nonparametric statistics (Tsybakov, 2009; Wasserman, 2006)), but rather with weaker losses, such as

$$d_{\mathcal{F}_D}(P, Q) = \sup_{f\in\mathcal{F}_D} \left| \mathbb{E}_{X\sim P}[f(X)] - \mathbb{E}_{X\sim Q}[f(X)] \right|, \tag{2.55}$$

where $\mathcal{F}_D$ is a *discriminator class* of bounded, Lebesgue-measurable functions, and P and Q lie in a *generator class* $\mathcal{F}_G$ of Borel probability measures on a sample space $\mathcal{X}$. To allow probability measures P without densities (i.e., $P \not\ll \mu$), we assume each basis element $\phi_z : \mathcal{X} \to \mathbb{R}$ is a bounded function, so that $\tilde{P}_z := \mathbb{E}_{X\sim P}[\phi_z(X)]$ is well-defined for a fixed orthonormal basis $\mathcal{B} = \{\phi_z\}_{z\in\mathcal{Z}}$ of $\mathcal{L}^2_\mu$ indexed by a countable family $\mathcal{Z}$.

We want to upper bound the adversarial risk of the following density estimator. Define, for constants $L > 0$ and $p \geqslant 1$ and real-valued net $\{a_z\}_{z\in\mathcal{Z}}$, generalized ellipses of the form

$$\mathcal{H}_{p,a}(L) = \left\{ f \in \mathcal{L}^1(\mathcal{X}) : \left(\sum_{z\in\mathcal{Z}} a_z^p |\tilde{f}_z|^p \right)^{1/p} \leqslant L \right\}. \tag{2.56}$$

(where $\tilde{f}_z := \int_{\mathcal{X}} f\phi_z \, d\mu$ is the z^{th} coefficient of f in the basis $\mathcal{B}$). For any finite set $Z \subseteq \mathcal{Z}$, let $\hat{P}_Z$ be the truncated series estimate

$$\hat{P}_Z := \sum_{z \in Z} \hat{P}_z \phi_z, \quad \text{where, for any } z \in \mathcal{Z}, \quad \hat{P}_z := \frac{1}{n} \sum_{i=1}^{n} \phi_z(X_i). \tag{2.57}$$

Z is a tuning parameter that typically corresponds to a smoothing parameter; for example, when $\mathcal{B}$ is the Fourier basis and $Z = \{z \in \mathbb{Z}^d : \|z\|_\infty \leqslant \zeta\}$ for some $\zeta > 0$, $\hat{P}_Z$ is equivalent to a kernel density estimator using a sinc product kernel $K_h(x) = \prod_{j=1}^{d} \frac{2}{h} \frac{\sin(2\pi x/h)}{2\pi x/h}$ with bandwidth $h = 1/\zeta$ (Owen, 2007).

We now present our main upper bound on the minimax rate of density estimation under adversarial losses. The upper bound is given for the orthogonal series estimator given in Equation (2.57), but we expect kernel and other standard linear density estimators to converge at the same rate.

Theorem 2.13 *Fix constants* $L_D, L_G > 0$, $p, q \leqslant 2$ *and real-valued families* $\{a_z\}_{z \in \mathcal{Z}}$ *and* $\{b_z\}_{z \in \mathcal{Z}}$. *Suppose* $\mathcal{F}_D = \mathcal{H}_{p,a}(L_D)$ *and* $\mathcal{F}_G = \mathcal{H}_{q,b}(L_G)$. *Then, for all* $P \in \mathcal{F}_G$ *and* $Z \subseteq \mathcal{Z}$,

$$\mathbb{E}_{X_1,\ldots,X_n \overset{\text{IID}}{\sim} P} \left[d_{\mathcal{F}_D}\left(P, \hat{P}\right) \right] \leqslant \frac{L_D}{\sqrt{n}} \sqrt{\sum_{z \in Z} \frac{\|\phi_z\|_{\mathcal{L}_P^\infty}^2}{a_z^2}} + L_D L_G \left(\sup_{z' \in \mathcal{Z} \setminus Z} a_z^{-1} \right) \left(\sup_{z' \in \mathcal{Z} \setminus Z} b_z^{-1} \right). \tag{2.58}$$

Proof One can check, using the fact that $\mathcal{F}_D$ is ortho-symmetric (*i.e.* for every $f \in \mathcal{F}_D$ and $Z \subseteq \mathcal{Z}$, $f - 2\sum_{z \in Z} \tilde{f}_z \phi_z \in \mathcal{F}_D$, so that $\mathcal{F}_D$ is symmetric across every axis defined by the basis $\mathcal{B}$), Lebesgue's monotone convergence theorem, and the assumption that each $f \in \mathcal{F}_D$ is bounded, that, for all probability measures P and Q,

$$\mathbb{E}_{X_1,\ldots,X_n} \left[d_{\mathcal{F}_D}\left(P, Q\right) \right] \leqslant \mathbb{E}_{X_1,\ldots,X_n} \left[\sup_{f \in \mathcal{F}_D} \sum_{z \in \mathcal{Z}} |\tilde{f}_z \left(\tilde{P}_z - \tilde{Q}_z \right)| \right].$$

Therefore,

$$\begin{aligned}
\mathbb{E}_{X_1,\ldots,X_n} \left[d_{\mathcal{F}_D}\left(P, \hat{P}\right) \right] &\leqslant \mathbb{E}_{X_1,\ldots,X_n} \left[\sup_{f \in \mathcal{F}_D} \sum_{z \in \mathcal{Z}} |\tilde{f}_z \left(\tilde{P}_z - \hat{P}_z \right)| \right] \\
&= \mathbb{E}_{X_1,\ldots,X_n} \left[\sup_{f \in \mathcal{F}_D} \sum_{z \in Z} |\tilde{f}_z \left(\tilde{P}_z - \hat{P}_z \right)| + \sum_{z \in \mathcal{Z} \setminus Z} |\tilde{f}_z \left(\tilde{P}_z - \hat{P}_z \right)| \right] \\
&= \mathbb{E}_{X_1,\ldots,X_n} \left[\sup_{f \in \mathcal{F}_D} \sum_{z \in Z} |\tilde{f}_z \left(\tilde{P}_z - \hat{P}_z \right)| + \sum_{z \in \mathcal{Z} \setminus Z} |\tilde{f}_z \tilde{P}_z| \right] \\
&\leqslant \mathbb{E}_{X_1,\ldots,X_n} \left[\sup_{f \in \mathcal{F}_D} \sum_{z \in Z} |\tilde{f}_z \left(\tilde{P}_z - \hat{P}_z \right)| \right] + \sup_{f \in \mathcal{F}_D} \sum_{z \in \mathcal{Z} \setminus Z} |\tilde{f}_z \tilde{P}_z|.
\end{aligned}$$

Note that we have decomposed the risk into two terms, the first comprising estimation error (variance) and the second comprising approximation error (bias). Indeed, in the case that $\mathcal{F}_D = \mathcal{L}^2(\mathcal{X})$, the above becomes precisely the usual bias-variance decomposition of mean squared error.

To bound the first term, applying the Cauchy-Schwarz inequality, the fact that $f \in \mathcal{F}_D$, and Jensen's inequality (in that order), we have

$$\begin{aligned}
\mathbb{E}_{X_1,\dots,X_n}\left[\sup_{f\in\mathcal{F}_D}\sum_{z\in Z}|\tilde{f}_z\left(\tilde{P}_z-\hat{P}_z\right)|\right] &= \mathbb{E}_{X_1,\dots,X_n}\left[\sup_{f\in\mathcal{F}_D}\sum_{z\in Z}a_z|\tilde{f}_z|\frac{|\tilde{P}_z-\hat{P}_z|}{a_z}\right] \\
&\leqslant \mathbb{E}_{X_1,\dots,X_n}\left[\sup_{f\in\mathcal{F}_D}\left(\sum_{z\in Z}a_z^2|\tilde{f}_z|^2\right)^{1/2}\left(\sum_{z\in Z}\frac{\left(\tilde{P}_z-\hat{P}_z\right)^2}{a_z^2}\right)^{1/2}\right] \\
&\leqslant L_D\mathbb{E}_{X_1,\dots,X_n}\left[\left(\sum_{z\in Z}\frac{\left(\tilde{P}_z-\hat{P}_z\right)^2}{a_z^2}\right)^{1/2}\right] \\
&\leqslant L_D\left(\sum_{z\in Z}\frac{\mathbb{E}_{X_1,\dots,X_n}\left[\left(\tilde{P}_z-\hat{P}_z\right)^2\right]}{a_z^2}\right)^{1/2} \\
&\leqslant \frac{L_D}{\sqrt{n}}\left(\sum_{z\in Z}\frac{\|\phi_z\|^2_{\mathcal{L}^\infty_P}}{a_z^2}\right)^{1/2},
\end{aligned}$$

since

$$\mathbb{E}_{X_1,\dots,X_n}\left[\left(\tilde{P}_z-\hat{P}_z\right)^2\right] = \mathrm{Var}_{X_1,\dots,X_n}\left[\frac{1}{n}\sum_{i=1}^n\phi_z(X_i)\right] \leqslant \frac{\|\phi_z\|^2_{\mathcal{L}^\infty_P}}{n}.$$

For the second term, by the Cauchy-Schwarz inequality,

$$\begin{aligned}
\sup_{f\in\mathcal{F}_D}\sum_{z\in\mathcal{Z}\setminus Z}|\tilde{f}_z\tilde{P}_z| &\leqslant \sup_{f\in\mathcal{F}_D}\left(\sum_{z\in\mathcal{Z}\setminus Z}|\tilde{f}_z|^2\right)^{1/2}\left(\sum_{z\in\mathcal{Z}\setminus Z}\tilde{P}_z\right)^{1/2} \\
&\leqslant \sup_{f\in\mathcal{F}_D}\left(\sum_{z\in\mathcal{Z}\setminus Z}\frac{a_z^2|\tilde{f}_z|^2}{\inf_{z'\in\mathcal{Z}\setminus Z}a_{z'}^2}\right)^{1/2}\left(\sum_{z\in\mathcal{Z}\setminus Z}\frac{b_z^2|\tilde{P}_z|^2}{\inf_{z'\in\mathcal{Z}\setminus Z}b_{z'}^2}\right)^{1/2} \\
&\leqslant L_DL_G\left(\sup_{z'\in\mathcal{Z}\setminus Z}a_z^{-1}\right)\left(\sup_{z'\in\mathcal{Z}\setminus Z}b_z^{-1}\right)
\end{aligned}$$

■

The two terms in the bound (2.58) demonstrate a bias-variance tradeoff, in which the first term (*variance*) increases with the truncation set Z and is typically independent of the class $\mathcal{F}_G$ of distributions, while the second term (*bias*) decreases with Z at a rate depending on the complexity of $\mathcal{F}_G$. Thus, if discriminator is weak, we might obtain $d_{\mathcal{F}_D}(P,Q)$ to be 0, while in reality it is not zero. Thus, combining GAN objective with an upper bound as well as presented in Section 2.8.3 would help in strengthening the discriminating and producing better samples.

2.9 EXPERIMENTS ON SET GENERATION

In this section we demonstrate the point cloud generation capabilities of PC-GAN and compare it with some of the existing methods. We train the proposed PC-GAN on synthetic 3D point

cloud and ModelNet40 benchmark datasets. The method of Achlioptas et al. (2017) could be treated as an AAE extension to point clouds, therefore we call it AAE in the discussion below. We use the implementation provided by Achlioptas et al. (2017) [8] to train AAE with the approximated EMD objective (Bertsekas, 1985). For AAE, the decoder is a MLP, where the output is *#points* $\times$ *point dimensions*.

For PC-GAN, we combine the objective of Fisher GAN and Approximated Wasserstein distance (Bertsekas, 1985) with mixture $1:20$ without tuning. We use Fisher GAN (Mroueh and Sercu, 2017) to replace WGAN-GP (Gulrajani et al., 2017). We found that this way the performance is similar to WGAN-GP, but the training is more stable and faster.

2.9.1 *Synthetic Datasets*

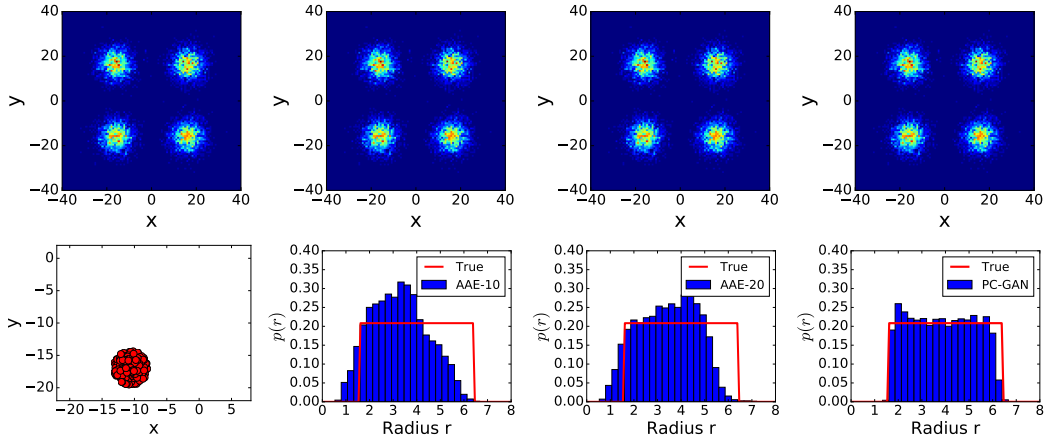
We created a simple 2D synthetic point cloud datasets from parametric distributions on which we can carry out thorough evaluations of our model and draw comparisons with Achlioptas et al. (2017) (AAE). We generate 2D point clouds for circles, where the center of circles is followed a mixture of four Gaussians with means equal to $\{\pm 16\} \times \{\pm 16\}$. The variances were set to be $16I$ and we used equal mixture weights. The radius of the circles was drawn from a uniform distribution $\mathrm{Unif}(1.6, 6.4)$ as shown in Figure 2.17a. We sampled $10,000$ circles for the training and testing data, respectively.

For the proposed PC-GAN, the architecture is a stack of 3 Permutation Equivariance Layers with size 30. The size of the latent variables θ in (2.46) was set to be 15. In this experiment the total number of parameters for PC-GAN was 12K. For AAE encoder, we follow the same network architecture used in Achlioptas et al. (2017). Here the decoder is a 4-layer MLP where the output is 500×2 dimensions for 500 points. We consider two model configurations AAE-10 and AAE-20, which use 10 and 20 units for the hidden layers of the decoder, respectively. The total number of parameters (encoder+decoder) are 14K and 24K for AAE-10 and AAE-20, respectively. Detailed model configurations are provided in the supplementary material.

We evaluated the conditional distributions on the $10,000$ testing circles. For the proposed PC-GAN, we pass the same points into the inference network Q, then sample 500 points with the conditional generator G_x to match the output number of AAE. We measured the empirical distributions of the centers and the radius of the generated circles conditioning on the testing data for PC-GAN. Similarly, we measured the reconstructed circles of the testing data for AAE. The results are shown in Figure 2.17.

As we can see from the results, the models can successfully recover the center distribution, but AAE does not learn the radius distribution well. Even if we increase number the hidden layer unit to be 20 (AAE-20), which almost doubles the number of parameters, the performance is still not satisfactory. Compared with AAE, the proposed PC-GAN recovers the both center and radius distributions well with less parameters. The gap of memory usage could be larger if we configure AAE to generate more points, while the model size required for PC-GAN is independent of the number of points. The reason is MLP decoder adopted by Achlioptas et al. (2017) wasted parameters for nearby points. Using the more advanced point cloud autoencoder (Y. Yang et al., 2018) may resolve the issue. However, it is still restricted to generate a fixed number of points for each object.

8 https://github.com/optas/latent_3d_points



(a) True Data (b) AAE-10 (c) AAE-20 (d) PC-GAN

Figure 2.17: The reconstructed center and radius distributions. (a) (top) the true center distribution and (bottom) one example of the 2D circle point cloud. (b-d) are the reconstructed center and radius distributions of different algorithms.

2.9.2 *ModelNet40*

We consider the ModelNet40 (Z. Wu et al., 2015) benchmark, which contains 40 classes of objects. There are 9,843 training and 2,468 testing instances. We follow Zaheer, Kottur, Ravanbakhsh, et al. (2017) to do pre-processing. For each object, we sampled 10,000 points from the mesh representation and normalize it to have zero mean (for each axis) and unit (global) variance. During the training, we augment the data by uniformly rotating $0, \pi/8, \ldots, 7\pi/8$ rad on the x-y plane. For PC-GAN, the random noise z_2 is fixed to be 10 dimensional for all experiments. For other settings, we follow Achlioptas et al. (2017).

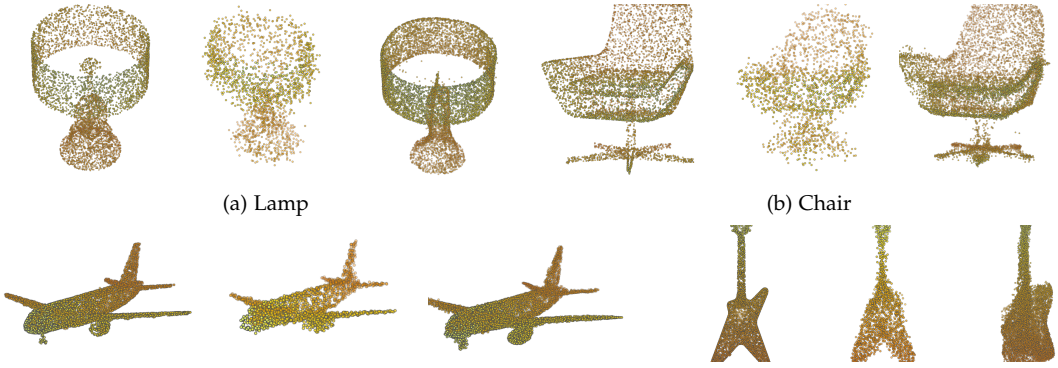
TRAINING ON SINGLE CLASS We start from a smaller model which is only trained on single class of objects. For AAE, the latent code size for its encoder is 128 and the decoder outputs 2,048 points for each object. The number of parameters for encoder and decoder are 15M in total. Therefore, we set the size of PC-GAN latent variable to be 128 dimensional. The number of parameters for G and G-inv is less than 1M in total.

TRAINING ON ALL CLASSES. We also train the proposed model on all 9,843 objects in the training set. The size of AAE latent code of is increased to be 256. The number of parameters of its encoder and decoder is 15.2M. we set the size of PC-GAN latent variable to be 256 dimensional. The number of parameters for G and G-inv is around 3M in total.

2.9.2.1 *Conditional Generation Results*

We first evaluate the performance of trained conditional generator and the inference network. We are interested in whether the learned model can model the distribution of the unseen testing data. Therefore, for each testing point cloud, we use the inference network to infer the latent variable xx, then use the conditional generator to generate points. We then measure the distribution between the input point cloud and the generated point cloud.

There are many criteria based on finite sample estimation for evaluation, such f-divergence and IPM. However, the estimator with finite samples are either biased or with high variance (Peyré, Cuturi, et al., 2017; Póczos, Liang Xiong, and Jeff Schneider, 2012; Q. Wang, Kulkarni, and



(a) Lamp (b) Chair

(c) Plane (d) Guitar

Figure 2.18: The reconstruction on test objects from seen categories. For each object, from left to right is training data, AAE, and PC-GAN. PC-GAN is better in capturing fine details like wheels of aeroplane or proper chair legs.

Table 2.5: The quantitative results of different models trained on different subsets of ModelNet40. MultiNet10 is a subset containing 10 classes of objects, while MultiNet40 is a full training set.

Data	Distance				Coverage			
	AAE	Approx	GAN	PC-GAN	AAE	Approx	GAN	PC-GAN
Aeroplanes	1.99E+01	1.53E+01	2.49E+01	1.89E+01	2.99E-02	1.73E-01	1.88E-01	1.95E-01
Benches	1.41E+01	1.05E+01	2.46E+01	1.09E+01	2.35E-01	2.58E-01	3.83E-01	4.44E-01
Cars	6.23E+01	4.25E+01	6.68E+01	4.39E+01	4.98E-02	1.78E-01	2.35E-01	2.35E-01
Chairs	1.08E+01	1.06E+01	1.08E+01	1.01E+01	1.82E-01	3.57E-01	3.95E-01	3.90E-01
Cups	1.79E+03	1.28E+03	3.01E+03	1.44E+03	3.31E-01	4.32E-01	5.68E-01	6.31E-01
Guitars	1.93E+02	1.97E+02	1.81E+02	2.16E+02	7.98E-02	2.11E-01	2.27E-01	2.25E-01
Lamps	1.60E+03	1.64E+03	2.77E+03	1.47E+03	2.33E-01	3.79E-01	3.66E-01	3.89E-01
Laptops	3.73E+00	2.65E+00	2.58E+00	2.43E+00	2.56E-01	3.93E-01	4.55E-01	4.31E-01
Sofa	1.64E+01	1.45E+01	2.76E+01	1.71E+01	1.62E-01	2.94E-01	3.47E-01	3.65E-01
Tables	2.96E+00	2.44E+00	3.69E+00	2.79E+00	2.59E-01	3.20E-01	3.53E-01	1.82E-01
ModelNet10	6.89E+00	6.03E+00	9.19E+00	5.77E+00	1.90E-01	3.36E-01	3.67E-01	3.47E-01
ModelNet40	5.86E+01	5.24E+01	7.96E+01	4.84E+01	1.85E-01	3.65E-01	3.71E-01	3.80E-01

Verdú, 2009). Also, it is impossible to use these estimators with infinitely many samples if they are accessible.

For ModelNet40, the meshes of each object are available. In many statistically guaranteed distance estimates, the adopted statistics are commonly based on distance between nearest neighbors (Póczos, Liang Xiong, and Jeff Schneider, 2012; Q. Wang, Kulkarni, and Verdú, 2009). Therefore, we propose to measure the performance with the following criteria. Given a point cloud $\{x_i\}_{i=1}^n$ and a mesh, which is a collection of faces $\{F_j\}_{j=1}^m$, we measure

$$D\left(\{x_i\}_{i=1}^n, \{F_j\}_{j=1}^m\right) = \frac{1}{n} \min_j d(x_i, F_j),$$

where $d(x_i, F_j)$ is the Euclidean distance from x_i to the face F_j. This distance is similar to Chamfer distance, which is commonly used for measuring images and point clouds (Achlioptas et al., 2017; H. Fan, Hao Su, and Guibas, n.d.), with infinitely samples from true distributions (meshes).

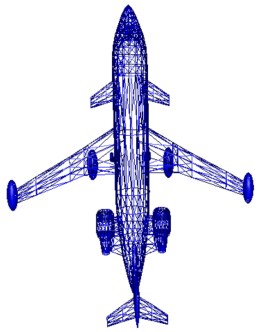


Figure 2.19: Sample mesh from ModelNet40

On the other hand, mode collapse is commonly happened in GAN algorithm. We are interested in whether the generated points recover enough supports of the distribution. We compute the *Coverage* ratio as follows. For each points, we find the its nearest face, we then treat this face is covered[9]. We then compute the ratio of number of faces of a mesh is covered. A sampled mesh is showed in Figure 2.19, where we can observe the distribution of mesh is not uniform. Therefore, it is difficult to get high coverage for AAE or PC-GAN, since we only sample limited number of points for training. On the other hand, we can also observe the details of the objects have more meshes. Therefore, the coverage measured based on faces could also serve an indicator for how much details of the objects covered by the generated point cloud.

The results are reported in Table 2.5. We compare four different algorithm, AAE and use three objective for training conditional generators, Approx (approximated Wasserstein distance), GAN (Fisher GAN), and the primal-dual loss by combining Approx and GAN as mentioned in Section 2.8. From Table 2.5, we observe that Approx usually results in smaller distance measure, which suggests the validity of training GAN with primal formulation. However, it loses more details with worse coverage than GAN. However, GAN results in worse distance measure. From the empirical observation, by balancing these two objectives for training a generative model, PC-GAN results in an desirable middle ground. The simple balancing strategy is also theoretically supported from "Sandwiching" objective discussed in Section 2.8.

2.9.2.2 *Hierarchical Sampling*

In Section 2.8, we propose a hierarchical sampling process for sampling point clouds. In the first hierarchy, the generator XX samples a object, while the second generator sample points to form the point cloud. The randomly sampled results without given any data as input are shown in Figure 2.20. The point clouds are all smooth, structured and almost symmetric. It shows PC-GAN captures inherent symmetries and patterns in all the randomly sampled objects, even if overall object is not perfectly formed. This highlights that learning point-wise generation scheme encourages learning basic building blocks of objects.

9 We should do thresholding to ignore outlier points. In our experiments, we observe that without excluding outliers does not change conclusion for comparison.

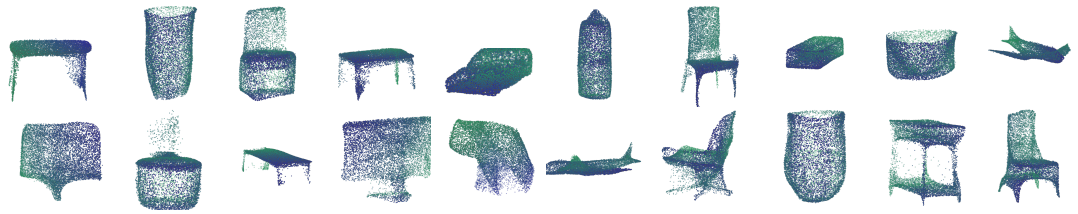


Figure 2.20: Randomly sampled objects and corresponding point cloud from the hierarchical sampling. Even if there are some defects, the objects are smooth, symmetric and structured. It suggests PC-GAN captures inherent patterns and learns basic building blocks of objects.

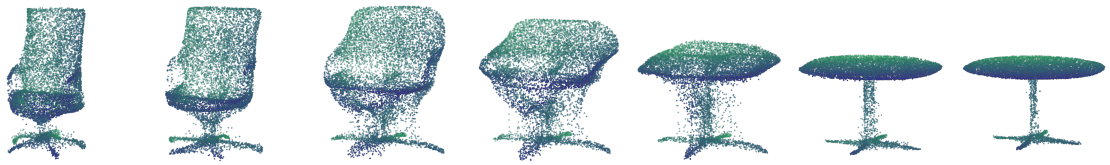


Figure 2.21: Interpolating between a table and a chair point clouds, using our latent space representation.

2.9.2.3 *Understand the Learned Manifold*

CLASSIFICATION We evaluate the quality of the representation acquired from the learned inference network. We train the inference network and the generator on the training split of ModelNet40 with data augmentation as mentioned above. We then extract the latent representation and train linear SVM on the that. We apply the same setting to a linear classifier on the latent code of Achlioptas et al. (2017).

The encoder of Achlioptas et al. (2017) took 2048 points for each obj as input while we only sample 1000 as input for our inference network. Benefited by the Deep Sets architecture for the inference network, which is invariant to number of points. Therefore, we are still allowed to sample more points as input to the trained inference network for evaluation. Because of the randomness of sampling points for extracting latent representation, we repeat the experiments 20 times and report the average accuracy and standard deviation on the testing split in Table 2.6. By using 1000 points, we are already better than Achlioptas et al. (2017) with 2048 points, and competitive with the supervised learning algorithm Deep Sets.

INTERPOLATION A commonly used method to demonstrate the quality of the learned latent space is showing whether the interpolation between two objects on the latent space results in

Method	# points	Accuracy
PC-GAN	1000	$87.5 \pm .6\%$
PC-GAN	2048	$87.8 \pm .2\%$
AAE (Achlioptas et al., 2017)	2048	$85.5 \pm .3\%$
Supervised DeepSets	1000	$87 \pm 1\%$
Supervised DeepSets	5000	$90 \pm .3\%$

Table 2.6: Classification accuracy results.

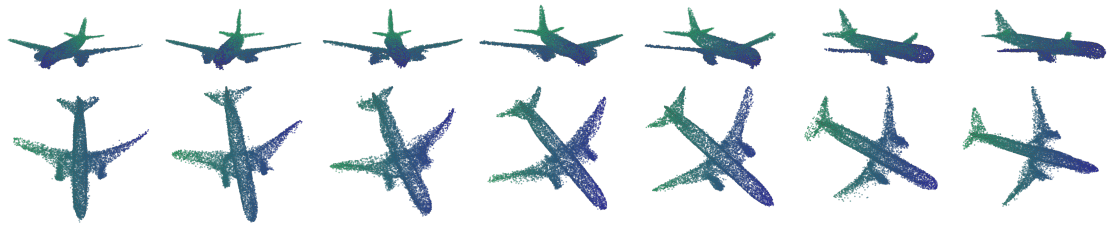


Figure 2.22: Interpolating between rotation of an aeroplane, using our latent space representation.

smooth change. We interpolate the inferred representations from two objects by the inference network, and use the generator to sample points. The inter-class result is shown in Figure 2.21.

It is also popular to show intra-class interpolation. In addition show simple intra-class interpolations, where the objects are almost aligned, we present an interesting study on interpolations between rotations. During the training, we only rotate data with 8 possible angles for augmentation, here we show it generalizes to other unseen rotations as shown in Figure 2.22.

However, if we linearly interpolate the code, the resulted change is scattered and not smooth as shown in Figure 2.22. Instead of using linear interpolation, We train a 2-layer MLP with limited hidden layer size to be 16, where the input is the angle, output is the corresponding latent representation of rotated object. We then generate the code for rotated planes with this trained MLP. It suggests although the transformation path of rotation on the latent space is not linear, it follows a smooth trajectory[10]. It may also suggest the *geodesic* path of the learned manifold may not be nearly linear between rotations. Finding the geodesic path with a principal method (Shao, A. Kumar, and Fletcher, 2017) and Understanding the geometry of the manifold for point cloud worth more deeper study as future work.

GENERALIZATION ON UNSEEN CATEGORIES In above, we studied the reconstruction of unseen testing objects, while PC-GAN saw the training objective from the same category. Here we study the more challenging task. We train PC-GAN on first 30 (Alphabetic order of category names) categories only, and test on the unseen 10 classes. The mean distance and coverage are 57.4 and 0.36. The reconstruted (conditionally generated) point clouds is shown in Figure 2.23. The results are generalized surprisingly well. For the object from the unseen categories, the conditionally generated point clouds still recovers main shape and reasonable gemotry structure, which confirms the advantage of the proposed PC-GAN: by enforcing the point-wise transformation, the model is forced to learn the underlying goemetry structure and the shared building blocks, instead of naively copying the input from the conditioning.

2.10 POINTS TO PONDER

In this chapter, we developed DeepSets, a model for sets *leveraging the permutation invariance and equivariance structure*. We demonstrated the generalization ability of DeepSets across several domains by extensive experiments, and show both qualitative and quantitative results. In particular, we explicitly show that DeepSets outperforms other intuitive deep networks on discrimintative tasks, which are not backed by theory (Section 2.5). We also proposed a new generative model for sets using DeepSets with application to point clouds. We can achieve very competitive results using smaller network sizes than existing methods (Section 2.9) because we

10 By the capability of 1-layer MLP.

(a) Sofa (b) Stool (c) Table

(d) Toilet (e) TV Stand (f) Vase

Figure 2.23: The reconstructed objects from unseen categories. In each plot, LHS is true data while RHS is PC-GAN. PC-GAN generalizes well as it can match patterns and symmetries from categories seen in the past to new unseen categories.

leverage the iid structure of points in a set and learn 'point-wise' transformations which can enforce the model to learn the building components of the objects, instead of just naively copying the whole object. This allows PC-GAN can capture delicate details of 3D point clouds and can generalize well even on unseen data. Last but not least, it is worth noting that the state-of-the-art we compare to is a specialized technique for each task, whereas our one model, *i.e.* DeepSets, is competitive across the board.

Despite the instances of success, there are still many open questions left. First of all the proof for expressing permutation invariance over uncountable universe is existential in nature, not constructive. Thus, using DeepSets might not be be the best/easiest way to learn as for certain problems the representation can be excessively difficult to represent the desired function in form of DeepSets. For example, consider we want to express maximum of a set S of real number. (For countable sets things are easy.) However, there cannot exist a algebraic expression for maximum of a set of real number of size 5 or greater, and so cannot be expressed exactly by any multilayer perceptron (i.e. without any max pooling operator). The reason for non-existence follows from Abel's impossibility theorem. Consider the polynomial $p(x) = \prod_{a \in S}(x - a)$. If there was an algebraic expression for finding the maximum of the set, it would implify there exists an algebraic expression for one of the roots of polynomial of degree 5 or greater. But this violates the "casus irreducibilis" of Abel's impossibility, thus such an expression cannot exist. Having said the technical difficulty of DeepSets in finding exact maximum, however, all hope is not lost as DeepSets architecture can still find arbitrary close approximation (with increasing architecture size) as mentioned in Section 2.3.

3

Compressed Data

We study the problem of learning representations for *compressed data*, which have become ubiquitous with advent of internet and smartphones, where cheap communication and computation are of utmost importance. We raise the question is it possible to apply successful machine learning techniques like deep networks directly on compressed data without decompressing it first, thereby reducing memory, computation, and latency. A prevalent example of compressed data are videos and try to answer the raised question in this context. Training robust deep video representations has proven to be much more challenging than learning deep image representations and consequently hampered tasks like video action recognition. This is in part due to the enormous size of raw video streams, the associated amount of computation required, and the high temporal redundancy. The 'true' and interesting signal is often drowned in too much irrelevant data. video compression techniques (like H.264, HEVC, etc.) reduce such superfluous information by up to two orders of magnitude by exploiting the *structure* in video stream. Motivated by the aforementioned fact, in this chapter, we propose to train a deep network directly on the compressed video, devoid of redundancy, rather than the traditional highly redundant RGB stream. This compressed representation has a higher information density and we found that leveraging the *structure* from compressed representation lead to easier training. In addition, the signals in a compressed video provide free, albeit noisy, motion information. We propose novel techniques to use them effectively. Our approach is about 4.6 times faster than a state-of-the-art 3D-CNN model, 2.7 times faster than a ResNet-152, and very easy to implement. On the task of action recognition, our approach outperforms all the other methods on the UCF-101, HMDB-51, and Charades dataset.

3.1 Introduction

Video nowadays commands the lion's share of internet traffic at 70% and rising (Networking Index, 2016). Most cell phone cameras now capture high resolution video streams in addition to images. Many real world data sources for computer vision are video based, ranging from inventory systems at warehouses to self-driving cars or autonomous drones. Video is also arguably the next frontier in computer vision, as it captures a wealth of information still images cannot convey. They carry more emotion (Gunnar A Sigurdsson et al., 2016), allow us to predict the future to a certain extent (Mathieu, Couprie, and LeCun, 2016), provide temporal context and give us better spatial awareness (Pollefeys et al., 2008). Unfortunately, very little of this information is currently exploited.

State-of-the-art deep learning models for video analysis are quite basic. Most of them naïvely try to exploit the deep learning based methods that have been highly successful in the image domain by feeding the video into image-based convolutional neural networks (CNNs) *frame by frame*. They often demonstrate results no better than hand-crafted techniques (Karpathy, Toderici, et al., 2014; H. Wang and Schmid, 2013). So, why did deep learning not yet make as transformative of an impact on video tasks, such as action recognition, as it did on images?

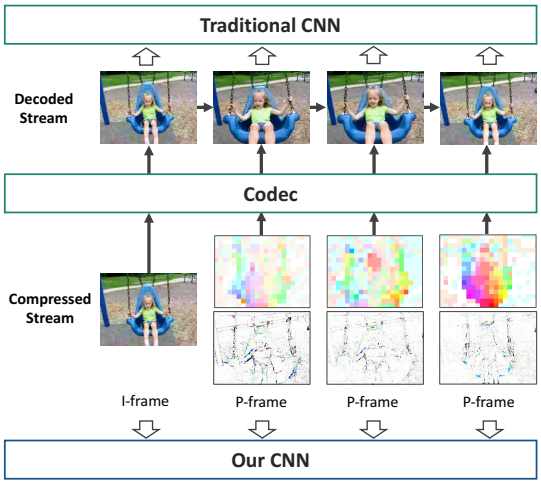


Figure 3.1: Traditional architectures first decode the video and then feed it into a network. Instead, we propose to use the compressed video directly.

We argue that the reason is two-fold. First, videos have a very low information density, as 1h of 720p video can be compressed from 222GB raw to 1GB. In other words, videos are filled with boring and repeating patterns, drowning the 'true' and interesting signal. It is not just harder for convolutional neural networks to extract the meaningful information, but also the unnecessary replication of data makes training much slower. Second, with only RGB images, learning temporal structure is difficult. While a single still image does not convey motion, one could imagine stacking multiple RGB frames could. A vast body of literature attempts to exploit this natural idea by processing videos as RGB image sequences, either with 2D CNNs, 3D CNNs, or recurrent neural networks (RNNs), but has yielded limited success (Karpathy, Toderici, et al., 2014; Tran, Ray, et al., 2017). Using precomputed optical flow almost always boosts the performance (Carreira and Zisserman, 2017).

To address these issues, we exploit the compressed representation developed for storage and transmission of videos rather than operating on the RGB frames (Figure 3.1). These compression techniques (like MPEG, H.264 etc.) leverage the fact that successive frames are usually very similar. They retain only a few frames completely and reconstruct other frames based on offsets from the complete images, called motion vectors and residual error (Section 3.2). Our model consists of multiple CNNs, each of which learns to model one kind of representation in a compressed video: images, motion vectors, and residuals (Section 3.3). To simplify the dependency in the compressed stream, we propose a novel preprocessing technique that allows training a simple decoupled model which enables parallelism during test time. In particular, we show that even though compressed representation, *e.g.* MPEG, does not seem amenable to CNNs directly, our method makes it effective in practice.

Why is this better? First, video compression removes up to two orders of magnitude of superfluous information, making interesting signals prominent. Second, the motion vectors in video compression provide us the *motion* information that lone RGB images do not have. Furthermore, the motion signals already exclude spatial variations, *e.g.* two people performing the same action in different clothings or in different lighting conditions exhibit the same *motion* signals. This improves generalization, and the lowered variance further simplifies training. Third, with compressed video, we account for correlation in video frames, *i.e.* spatial view plus some small changes over time, instead of i.i.d. images. Constraining data in this structure helps us tackling

the curse of dimensionality. Last but not least, our method is also much more efficient as we only look at the true signals instead of repeatedly processing near-duplicates. Efficiency is also gained by avoiding to decompress the video, because video is usually stored or transmitted in the compressed version, and access to the motion vectors and residuals are free.

We demonstrate that our approach on the task of action recognition significantly outperforms all other methods that train on traditional RGB images without using any recurrent neural networks, complicated fusion or 3D convolutions on the UCF-101 (Soomro, Zamir, and M. Shah, 2012), HMDB-51 (Kuehne et al., 2011), and Charades (Gunnar A Sigurdsson et al., 2016) dataset. In addition, our model is 4.6 times faster than state-of-the-art 3D CNN model Res3D (Tran, Ray, et al., 2017), and 2.7 times faster than ResNet-152 (He et al., 2016a). When combined with scores from a standard temporal stream network, our model outperforms state-of-the-art methods on all datasets (Section 3.4).

3.2 BACKGROUND

We provide a brief overview about video action recognition and video compression.

3.2.1 *Action Recognition*

Traditionally, for the task of action recognition on video, the community utilized hand-crafted features, such as Histogram of Oriented Gradients (HOG; Dalal and Triggs 2005) or Histogram of Optical Flow (HOF; Laptev et al. 2008), both sparsely (Laptev et al., 2008) and densely (H. Wang, Ullah, et al., 2009) sampled. While early methods consider independent interest points across frames, smarter aggregation based on dense trajectories have been used (X. Peng et al., 2014; H. Wang, Kläser, et al., 2013; H. Wang and Schmid, 2013). Some of these traditional methods are competitive even today, like iDT which corrects for camera motion (H. Wang and Schmid, 2013).

In the past few years, deep learning has brought significant improvements to video understanding (Donahue et al., 2015; Karpathy, Toderici, et al., 2014). However, the improvements mainly stem from improvements in deep image representations. Temporal structure is still relatively simple — most algorithms subsample a few frames and perform average pooling to make final predictions (Simonyan and Zisserman, 2014; Limin Wang et al., 2016). Recurrent neural networks (RNNs) (Donahue et al., 2015; Yue-Hei Ng et al., 2015), temporal CNNs (Ma et al., 2017), or other feature aggregation techniques (Girdhar, Ramanan, et al., 2017; Limin Wang et al., 2016) on top of CNN features have also been explored. However, while introducing new computation overhead, these methods do not necessarily outperform simple average pooling (Limin Wang et al., 2016). Some works explore 3D CNNs to model temporal structure (Tran, Bourdev, et al., 2015; Tran, Ray, et al., 2017). Nonetheless, it results in an explosion of parameters and computation time and only marginally improves the performance (Tran, Ray, et al., 2017).

More importantly, evidence suggests that these methods are not sufficient to capture all temporal structures — using of pre-computed optical flow almost always boosts the performance (Carreira and Zisserman, 2017; Feichtenhofer, Pinz, and Zisserman, 2016; Simonyan and Zisserman, 2014; Limin Wang et al., 2016). This emphasizes the importance of using the right input representation and inadequacy of RGB frames. Finally, note all these methods need raw video frame-by-frame and cannot exploit the fact that video is stored in some compressed format.

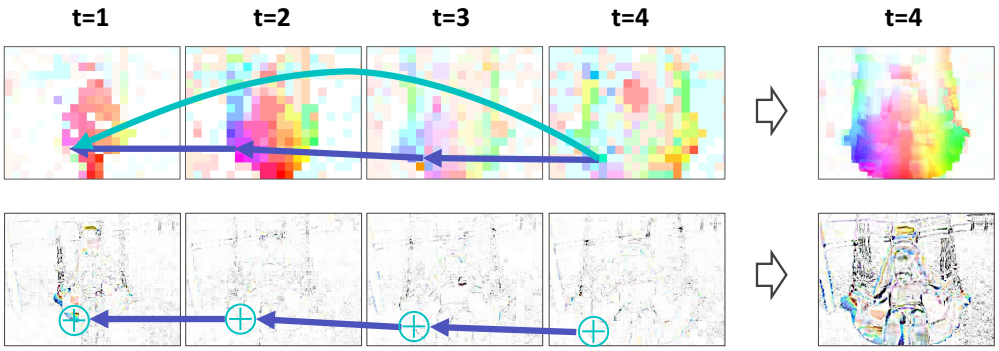


Figure 3.2: We trace all motion vectors back to the reference I-frame and accumulate the residual. Now each P-frame depends only on the I-frame but not other P-frames.

3.2.2 *Video Compression*

The need for efficient video storage and transmission has led to highly efficient video compression algorithms, such as MPEG, H.264, and HEVC, some of which date back to 1990s (Le Gall, 1991). Most video compression algorithms leverage the fact that successive frames are usually very similar. We can efficiently store one frame by reusing contents from another frame and only store the difference.

Most modern codecs split a video into *I-frames* (intra-coded frames), *P-frames* (predictive frames) and zero or more *B-frames* (bi-directional frames). I-frames are regular images and compressed as such. P-frames reference the previous frames and encode only the 'change'. A part of the change – termed motion vectors – is represented as the movements of block of pixels from the source frame to the target frame at time t, which we denote by $\mathcal{T}^{(t)}$. Given a pixel position, its motion vector describes the displacement from the current position to the referred position in the reference picture. Formally, we denote by $\mathbf{d}_{t,p} = (d_x, d_y)_{t,p}$ the motion vector at position p in frame t. d_x denotes the horizontal displacement and d_y denotes the vertical displacement. We represent the residual in RGB space, and denote it by $\mathbf{r}_{t,p} \in \mathbb{R}^3$. The source frame can be an I-frame or a P-frame. In most cases, the source frame is the previous frame. Even after this compensation for block movement, there can be difference between the original image and the predicted image at time t, we denote this residual difference by Δ_t. Putting it together, a P-frame at time t only comprises of motion vectors $\mathcal{T}^{(t)}$ and a residual $\Delta^{(t)}$. This gives the recurrence relation for reconstructing P-frames as

$$I_i^{(t)} = I_{i-\mathcal{T}_i^{(t)}}^{(t-1)} + \Delta_i^{(t)}, \tag{3.1}$$

where $I^{(t)}$ denotes the RGB image at time t and the subscripts denote the index of spatial location. The motion vectors and the residuals are then passed through a discrete cosine transform (DCT) space, and entropy-encoded.

A B-frame may be viewed as a special P-frame, where motion vectors are computed bi-directionally and may reference a future frame as long as there are no circles in referencing. A frame residual needs to be added back similar to a P-frame. Both B- and P- frames capture only what changes in the video, and are easier to compress owing to smaller dynamic range (I. E. Richardson, 2002). See Figure 3.3 for a visualization of the motion estimates and the residuals. Modeling arbitrary decoding order is beyond the scope of this paper. We focus on videos encoded using only backward references, namely I- and P- frames.

FEATURES FROM COMPRESSED DATA Some prior works have utilized signals from compressed video for detection or recognition, but only as a non-deep feature (Kantorov and Laptev, 2014; Sukmarg and Rao, 2000; Töreyin et al., 2005; Yeo and B. Liu, 1995). To the best of our knowledge, this is the first work that considers training deep networks on compressed videos. MV-CNN apply distillation to transfer knowledge from an optical flow network to a motion vector based network (B. Zhang et al., 2016). However, unlike our approach, it does not consider the general setting of representation learning on a compressed video, it still needs the entire decompressed video as RGB stream, and it requires optical flow as an additional supervision.

Equipped with this background, next we will explore how to utilize the compressed representation, devoid of redundant information, directly for action recognition.

3.3 MODELING COMPRESSED REPRESENTATIONS

Our goal is to design a computer vision system for action recognition that operates directly on the stored compressed video. The compression is solely designed to optimize the size of the encoding, thus the resulting representation has very different statistical and structural properties than the images in a raw video. It is not clear if the successful deep learning techniques can be adapted to compressed representations in a straightforward manner. So we ask how to feed a compressed video into a computer vision system, specifically a deep network?

Feeding I-frames into a deep network is straightforward since they are just images. How about P-frames? From Figure 3.3 we can see that motion vectors roughly resemble optical flows, though noisy. As modeling optical flows with CNNs has been proven quite effective, it is tempting to do the same for motion vectors. The third row of Figure 3.3 visualizes the residuals. We can see that they roughly give us a motion boundary in addition to a change of appearance, such as the change of lighting conditions. Again, CNNs are well-suited for such patterns. The outputs of corresponding CNNs from the image, motion vectors, and residual will have different properties. To combine them, we tried various fusion strategies, including mean pooling, maximum pooling, concatenation, convolution pooling, and bilinear pooling, on both middle layers and the final layer, but with limited success.

Digging deeper, one can argue that the motion vectors and residuals alone do not contain the full information of a P-frame — a P-frame depends on the reference frame, which again might be a P-frame. This chain continues all the way back to the preceding I-frame. Treating each P-frame as an independent observation clearly violates this dependency. A simple strategy to address this would be to reuse features from the reference frame, and only *update* the features given the new information. This recurrent definition screams for recurrent neural networks (RNNs) to aggregate features along the chain. However, preliminary experiments suggests the elaborate modeling effort in vain (see Section 3.5.1 for details). The difficulty arises due to the long chain of dependencies of the P-frames. To mitigate this issue, we devise a novel yet simple back-tracing technique that decouples individual P-frames.

DECOUPLED MODEL To break the dependency between consecutive P-frames, we trace all motion vectors back to the reference I-frame and accumulate the residual on the way. In this way, each P-frame depends only on the I-frame but not other P-frames.

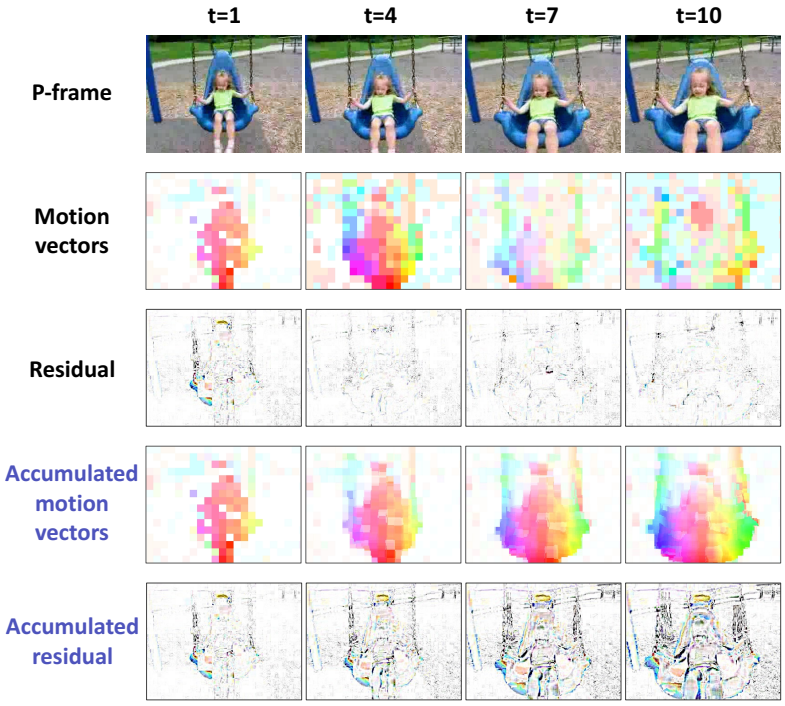


Figure 3.3: Original motion vectors and residuals describe only the change between two frames. Usually the signal to noise ratio is very low and hard to model. The accumulated motion vectors and residuals consider longer term difference and show clearer patterns. Assume I-frame is at $t = 0$. Motion vectors are plotted in HSV space, where the H channel encodes the direction of motion, and the S channel shows the amplitude. For residuals we plot the absolute values in RGB space. Best viewed in color.

Figure 3.2 illustrates the back-tracing technique. Given a pixel at location i in frame t, let $\mu_{\mathcal{T}^{(t)}}(i) := i - \mathcal{T}_i^{(t)}$ be the referenced location in the previous frame. The location traced back to frame $k < t$ is given by

$$\mathcal{J}_i^{(t,k)} := \mu_{\mathcal{T}^{(k+1)}} \circ \cdots \circ \mu_{\mathcal{T}^{(t)}}(i). \tag{3.2}$$

Then the accumulated motion vectors $\mathcal{D}^{(t)} \in \mathbb{R}^{H \times W \times 2}$ and the accumulated residuals $\mathcal{R}^{(t)} \in \mathbb{R}^{H \times W \times 3}$ at frame t are

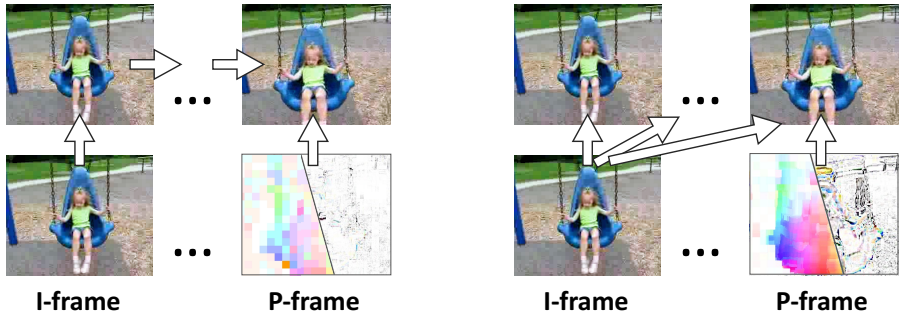
$$\mathcal{D}_i^{(t)} := i - \mathcal{J}_i^{(t,k)}, \text{ and}$$
$$\mathcal{R}_i^{(t)} := \Delta_{\mathcal{J}_i^{(t,k+1)}}^{(k+1)} + \cdots + \Delta_{\mathcal{J}_i^{(t,t-1)}}^{(t-1)} + \Delta_i^{(t)},$$

respectively. This can be efficiently calculated in linear time through a simple feed forward algorithm, accumulating motion and residuals as we decode the video. As shown in Figure 3.4b, each P-frame now has a different dependency

$$I_i^{(t)} = I_{i-\mathcal{D}_i^{(t)}}^{(0)} + \mathcal{R}_i^{(t)}, \quad t = 1, 2, \ldots, \tag{3.3}$$

Here P-frames depend only on the I-frame and can be processed in parallel.

A nice side effect of the back-tracing is robustness. The accumulated signals contain longer-term information, which is more robust to noise or camera motion. Figure 3.3 shows the accu-



(a) Original dependency. (b) New dependency.

Figure 3.4: We decouple the dependencies between P-frames so that they can be processed in parallel.

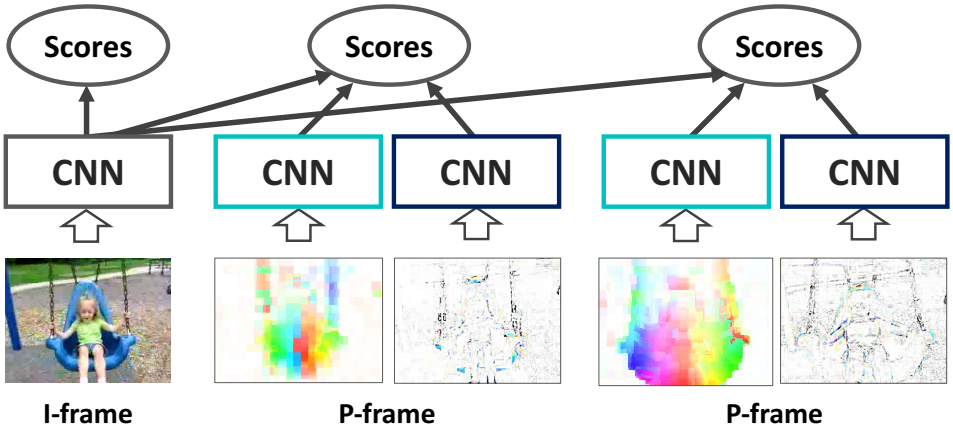


Figure 3.5: Decoupled model. All networks can be trained independently. Models are shared across P-frames.

mulated motion vectors and residuals respectively. They exhibit clearer and smoother patterns than the original ones.

PROPOSED NETWORK Figure 3.5 shows the graphical illustration of the proposed model. The input of our model is an I-frame, followed by T P-frames, *i.e.* $\left(I^0, \mathcal{D}^{(1)}, \mathcal{R}^{(1)}, \ldots, \mathcal{D}^{(T)}, \mathcal{R}^{(T)}\right)$. For notation simplicity we set $t = 0$ for I-frame. Each input source is modeled by a CNN, *i.e.*

$$\begin{aligned} x^{(0)}_{\text{RGB}} &:= \phi_{\text{RGB}}\left(I^{(0)}\right) \\ x^{(t)}_{\text{motion}} &:= \phi_{\text{motion}}\left(\mathcal{D}^{(t)}\right) \\ x^{(t)}_{\text{residual}} &:= \phi_{\text{residual}}\left(\mathcal{R}^{(t)}\right) \end{aligned} \tag{3.4}$$

While I-frame features $x^{(0)}_{\text{RGB}}$ are used as is, P-frame features $x^{(t)}_{\text{motion}}$ and $x^{(t)}_{\text{residual}}$ need to incorporate the information from $x^{(0)}_{\text{RGB}}$. There are several reasonable candidates for such a fusion, *e.g.* maximum, multiplicative or convolutional pooling. We also experiment with transforming RGB features according to the motion vector. Interestingly, we found a simple summing of scores to work best (see Section 3.5.2 for details). This gives us a model that is easy to train and flexible for inference.

IMPLEMENTATION Note that most of the information is stored in I-frames, and we only need to learn the *update* for P-frames. We thus focus most of the computation on I-frames, and use a much smaller and simpler model to capture the updates in P-frames. This yields significant saving in terms of computation, since in modern codecs most frames are P-frames.

Specifically, we use ResNet-152 to model I-frames, and ResNet-18 to model the motion vectors and residuals (He et al., 2016b). This offers a good trade-off between speed and accuracy. For video-level tasks, we use Temporal Segments (Limin Wang et al., 2016) to capture long term dependency, *i.e.* feature at each step is average of features across $k = 3$ segments during training.

3.4 EXPERIMENTS

We now validate for task of action recognition that (i) compressed video is a better representation (Section 3.4.1), leading to (ii) good accuracy (Section 3.4.3) and (iii) high speed (Section 3.4.2). To illustrate these claims, we evaluate on task of action recognition. However, note that the principle of the proposed method can be applied effortlessly to other tasks like video classification (Abu-El-Haija et al., 2016), object detection (Russakovsky et al., 2015), or action localization (Gunnar A Sigurdsson et al., 2016). We pick action recognition due to its wide range of applications and strong baselines.

DATASETS AND PROTOCOL We evaluate our method Compressed Video Action Recognition (CoViAR) on three action recognition datasets, UCF-101 (Soomro, Zamir, and M. Shah, 2012), HMDB-51 (Kuehne et al., 2011), and Charades (Gunnar A Sigurdsson et al., 2016). UCF-101 and HMDB-51 contain short ($<$ 10-second) trimmed videos, each of which is annotated with one action label. Charades contains longer ($\sim$ 30-second) untrimmed videos. Each video is annotated with one or more action labels and their intervals (start time, end time). UCF-101 contains 13,320 videos from 101 action categories. HMDB-51 contains 6,766 videos from 51 action categories. Each dataset has 3 (training, testing)-splits. We report the average performance of the 3 testing splits unless otherwise stated. The Charades dataset contains 9,848 videos split into 7,985 training and 1,863 test videos. It contains 157 action classes.

During testing we uniformly sample 25 frames, each with flips plus 5 crops, and then average the scores for final prediction. On UCF-101 and HMDB-51 we use temporal segments, and perform the averaging before softmax following Limin Wang et al. (2016). On Charades we use mean average precision (mAP) and weighted average precision (wAP) to evaluate the performance, following previous work Gunnar A. Sigurdsson et al. (2017).

TRAINING DETAILS Following TSN (Limin Wang et al., 2016), we resize UCF-101 and HMDB-51 videos to 340×256. As Charades contains both portrait and landscape videos, we resize to 256×256. Our models are pre-trained on the ILSVRC 2012-CLS dataset (Deng et al., 2009), and fine-tuned using Adam (D. P. Kingma and Ba, 2014) with a batch size of 40. Learning rate starts from 0.001 for UCF-101/HMDB-51 and 0.03 for Charades. It is divided by 10 when the accuracy plateaus. Pre-trained layers use a learning rate that is 100 times smaller than the base learning rate. We apply color jittering and random cropping to 224×224 for data augmentation following Limin Wang et al. (2016). Where available, we tune the hyper-parameters on splits other than the tested one. We use MPEG-4 encoded videos, which have on average 11 P-frames for every I-frame. When optical flows are used, we use TV-L1 flows (Zach, Pock, and Bischof, 2007).

Dataset	I	M	R	I+M	I+R	I+M+R (gain)
UCF-101						
Split 1	88.4	63.9	79.9	90.4	90.0	**90.8** (+2.4)
Split 2	87.4	64.6	80.8	89.9	89.6	**90.5** (+3.1)
Split 3	87.3	66.6	82.1	89.6	89.4	**90.0** (+2.7)
Average	87.7	65.0	80.9	89.9	89.7	**90.4** (+2.7)
HMDB-51						
Split 1	54.1	37.8	44.6	60.3	55.9	**60.4** (+6.3)
Split 2	51.9	38.7	43.1	57.9	54.2	**58.2** (+6.3)
Split 3	54.1	39.7	44.4	58.5	55.6	**58.7** (+4.6)
Average	53.3	38.8	44.1	58.9	55.2	**59.1** (+5.8)

Table 3.1: Action recognition accuracy on UFC-101 (Soomro, Zamir, and M. Shah, 2012) and HMDB-51 (Kuehne et al., 2011). Here we compare training with different sources of information. "+" denotes score fusion of models. I: I-frame RGB image. M: motion vectors. R: residuals. The bold numbers indicate the best and the underlined numbers indicate the second best performance.

Model	M	R	I+M	I+R	I+M+R
Original	58.3	79.0	90.0	89.8	90.4
Accumulated	**63.9**	**79.9**	**90.4**	**90.0**	**90.8**

Table 3.2: Action recognition accuracy on UFC-101 (Soomro, Zamir, and M. Shah, 2012) (Split 1). The two rows show the performance of the models trained using the original motion vectors/residuals and the models using the accumulated ones respectively. I: I-frame RGB image. M: motion vectors. R: residuals.

3.4.1 *Ablation Study*

Here we study the benefits of using compressed representations over RGB images. We focus on UCF-101 and HMDB-51, as they are two of the most well-studied action recognition datasets. Table 3.1 presents a detailed analysis. On both datasets, training on compressed videos significantly outperforms training on RGB frames. In particular, it provides 5.8% and 2.7% absolute improvement on HMDB-51 and UCF-101 respectively.

Quite surprisingly, while residuals contribute to a very small amount of data, it alone achieves good accuracy. Motion vectors alone perform not as well, as they do not contain spatial details. However, they offer information orthogonal to what still images provide. When added to other streams, it significantly boosts the performance. Note that we use only I-frames as full images, which is a small subset of all frames, yet CoViAR achieves good performance.

ACCUMULATED MOTION VECTORS AND RESIDUALS Accumulating them across multiple frames results in clearer patterns to model. This improves the performance, as shown in Table 3.2. On the first split of UCF-101, our accumulation technique provides 5.6% improvement on the motion vector stream network and on the full model, 0.4% improvement (4.2% error reduction). Performance of the residual stream also improves by 0.9% (4.3% error reduction).

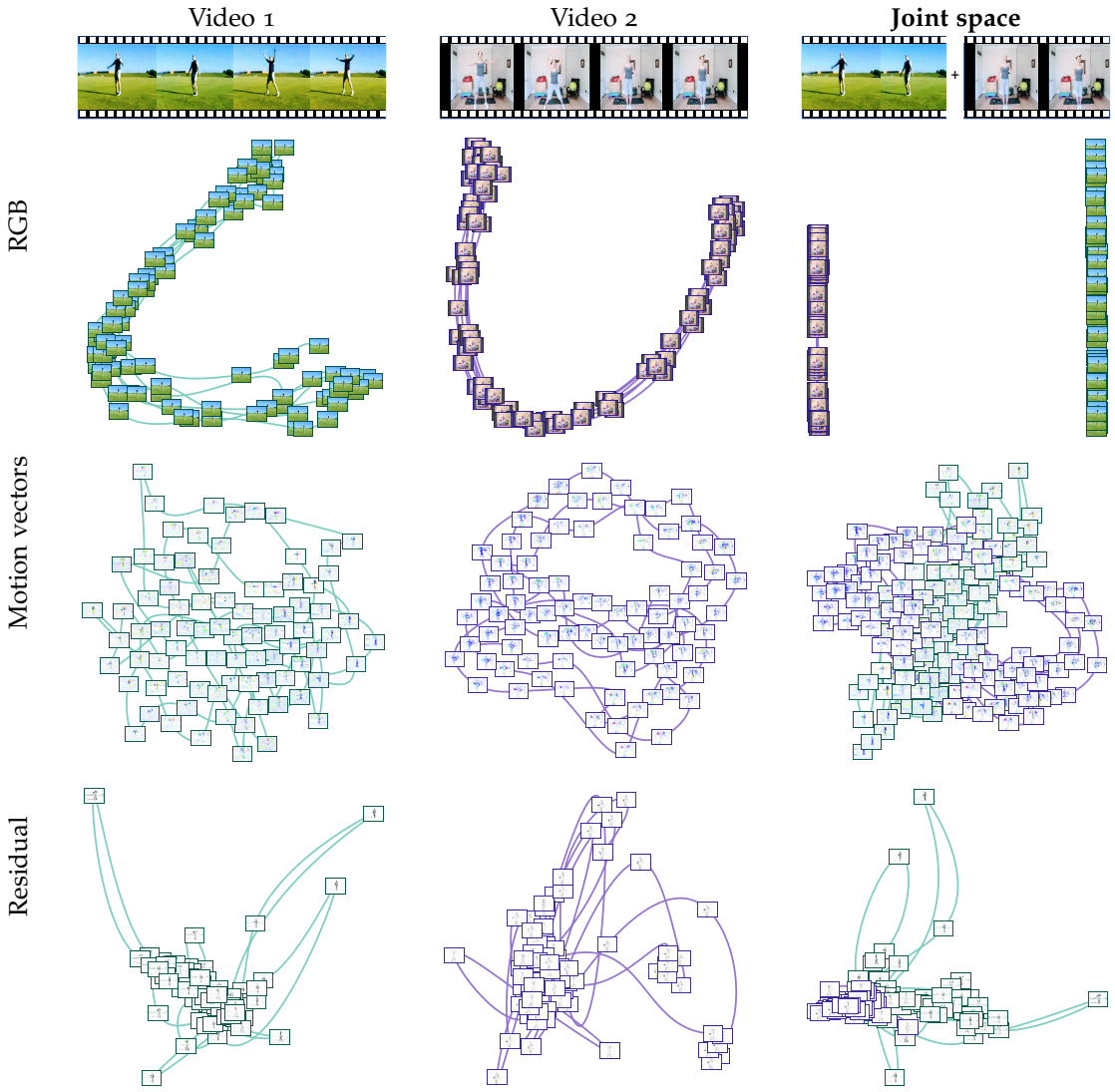


Figure 3.6: Two videos of "Jumping Jack" from UCF-101 in their RGB, motion vector, and residual representations plotted in t-SNE (Maaten and G. Hinton, 2008) space. The curves show video trajectories. While in the RGB space the two videos are clearly separated, in the motion vector and residual space they overlap. This suggests that with compressed signals, videos of the same action can share statistical strength better. Also note that the RGB images contain no motion information, and thus the two ways of the trajectories overlap. This is in contrast to the *circular* patterns in the trajectories of motion vectors. Best viewed on screen.

Method	Accuracy (%)		
	GFLOPs	UCF-101	HMDB-51
ResNet-50	**3.8**	82.3	48.9
ResNet-152	11.3	83.4	46.7
C3D	38.5	82.3	51.6
Res3D	19.3	85.8	54.9
CoViAR	4.2	**90.4**	**59.1**

Table 3.3: Network computation complexity and accuracy of each method. Our method is 4.6x more efficient than state-of-the-art 3D CNN, while being much more accurate.

VISUALIZATIONS In Figure 3.6, we qualitatively study the RGB and compressed representations of two videos of the same action in t-SNE (Maaten and G. Hinton, 2008) space. We can see that in RGB space the two videos are clearly separated, and in motion vector and residual space they overlap. This suggests that a RGB-image based model needs to learn the two patterns separately, while a compressed-video based model sees a shared representation for videos of the same action, making training and generalization easier.

In addition, note that the two ways of the RGB trajectories overlap, showing that RGB images cannot distinguish between the up-moving and down-moving motion. On the other hand, compressed signals preserve motion information. The trajectories thus form circles instead of going back and forth on the same path.

3.4.2 *Speed and Efficiency*

Our method is efficient because the computation on I-frame is shared across multiple frames, and the computation on P-frames is cheaper. Table 3.3 compares the CNN computational cost of our method with state-of-the-art 2D and 3D CNN architectures. Since for our model the P- and I-frame computational costs are different, we report the average GFLOPs over all frames. As shown in the table, CoViAR is 2.7 times faster than ResNet-152 (He et al., 2016a) and is 4.6 times more than Res3D (Tran, Ray, et al., 2017), while being significantly more accurate.

A more detailed speed analysis is presented in Table 3.4. The preprocessing time of two-stream methods, *i.e.* optical flow computation, is measured on a Tesla P100 GPU with an implementation of the TV-L1 flow algorithm from OpenCV. Our preprocessing, *i.e.* the calculation of the accumulated motion vectors and residuals, is measured on Intel E5-2698 v4 CPUs. CNN time is measured on the same P100 GPU. We can see that the optical flow computation is the bottleneck for two-stream networks, even with low-resolution 256×340 videos. Our preprocessing is much faster despite our CPU-only implementation.

For CNN time, we consider both settings where (i) we can forward multiple CNNs at the same time, and (ii) we do it sequentially. For both settings, our method is significantly faster than traditional methods. Overall, our method can be up to 100 times faster than traditional methods with multi-thread preprocessing, when preprocessing time is considered, processing more than 1,300 frames per second. Figure 3.7 summarizes the results. CoViAR achieves the best efficiency and good accuracy, while requiring a far lesser amount of data.

Method	Preprocess	CNN	
		sequential	concurrent
Two-stream			
BN-Inception	75.0	1.6	0.9
ResNet-152	75.0	7.5	4.0
CoViAR	**2.87/0.46**	**1.3**	**0.3**

Table 3.4: Speed (ms) per frame of each method. Our method is fast in both preprocessing and CNN computation. The preprocessing speed of our method is presented for both single-thread / multi-thread settings.

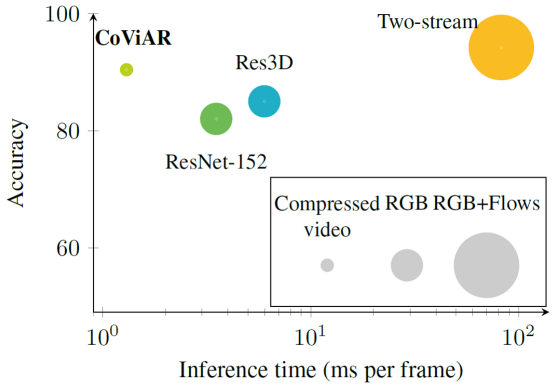


Figure 3.7: Speed and accuracy on UCF-101 (Soomro, Zamir, and M. Shah, 2012), compared to Two-stream Network (Simonyan and Zisserman, 2014), Res3D (Tran, Ray, et al., 2017), and ResNet-152 (He et al., 2016a) trained using RGB frames. Node size denotes the input data size. Training on compressed videos is both accurate and efficient, while requiring a minimal amount of data.

3.4.3 *Accuracy*

We now compare accuracy of our method CoViAR with state-of-the-art models in Table 3.6. For fair comparison, here we focus on models using the same pre-training dataset, ILSVRC 2012-CLS (Deng et al., 2009). While pre-training using Kinetics yields better performance (Carreira and Zisserman, 2017), since it is larger and more similar to the datasets used in this paper, those results are not directly comparable. From the upper part of the table, we can see that our model significantly outperforms traditional RGB-image based methods. C3D (Tran, Bourdev, et al., 2015), Res3D (Tran, Ray, et al., 2017) and I3D (Carreira and Zisserman, 2017) consider 3D convolution to learn temporal structures. Karpathy, Toderici, et al. (2014) and Diba, Sharma, and Van Gool (2017) consider more complicated fusions and pooling. MV-CNN (B. Zhang et al., 2016) applies distillation to transfer knowledge from a optical-flow-based model. Our method uses much faster 2D CNNs, simple late fusion without additional supervision, and still significantly outperforms these methods.

2 Despite our best efforts, we were not able to reproduce the performance reported in the original paper. Here we report the performance based on our implementation. For fair comparison, we use the same data augmentation and architecture as ours. Training follows the 2-stage procedure described in the original paper. We reached out to the authors, but they were unable to share their implementation.

Dataset	UCF-101			HMDB-51		
	CoViAR	Flow	CoViAR +flow	CoViAR	Flow	CoViAR +flow
Split 1	90.8	87.7	**94.0**	60.4	61.8	**71.5**
Split 2	90.5	90.2	**95.4**	58.2	63.7	**69.4**
Split 3	90.0	89.1	**95.2**	58.7	64.2	**69.7**
Average	90.4	89.0	**94.9**	59.1	63.2	**70.2**

Table 3.5: Action recognition accuracy on UFC-101 (Soomro, Zamir, and M. Shah, 2012) and HMDB-51 (Kuehne et al., 2011). Combining our model with a temporal-stream network achieves a new state-of-the-art.

TWO-STREAM NETWORK Most state-of-the-art models use the two-stream framework, *i.e.* one stream trained on RGB frames and the other on optical flows. It is natural to ask: What if we replace the RGB stream with our compressed stream? Here we train a temporal-stream network using 7 segments with BN-Inception (Ioffe and Szegedy, 2015), and combine it with our model by late fusion. Despite its simplicity, this achieves very good performance as shown in Table 3.5.

The lower part of Table 3.6 compares our method with state-of-the-art models using optical flow. CoViAR outperforms all of them. In particular, LRCN (Donahue et al., 2015), Composite LSTM Model (Srivastava, Mansimov, and Salakhudinov, 2015), and L^2STM (L. Sun et al., 2017) use RNNs to model temporal dynamics. ActionVLAD (Girdhar, Ramanan, et al., 2017) and TLE (Diba, Sharma, and Van Gool, 2017) apply more complicated feature aggregation. iDT+FT (H. Wang and Schmid, 2013) is based on hand-engineered features. Again, our method simply trains 2D CNNs separately without any complicated fusion or RNN and still outperforms these models.

Finally we evaluate our method on the Charades dataset. As the Charades dataset consists of longer videos with labels annotated at frame-level, we train our network to predict the labels of each frame. At test time we average the scores of the sampled frames as the final prediction. Table 3.7 shows the results. We can see that our method again outperforms other models trained on RGB images. Note that Gunnar A. Sigurdsson et al. (2017) use additional annotations including objects, scenes, and intentions to train a conditional random field (CRF) model. Our model requires only action labels. When using optical flow, we beat all other state-of-the-art methods. The effectiveness on Charades demonstrates that our method is not just effective for video-level predictions, but also for frame-level predictions.

3.5 FURTHER ANALYSIS

3.5.1 *RNN-Based Models*

Given the recurrent definition of frames in a compressed video, it is natural to use a recurrent neural network (RNN) to model the dependency. Figure 3.8 presents a graphical illustration. In preliminary experiments, we tried Conv-LSTM (Xingjian et al., 2015) to model the evolution of feature maps. Let $x^{(t)}_{\text{fusion}} := \max\left(x^{(t)}_{\text{motion}}, x^{(t)}_{\text{residual}}\right)$, where $x^{(t)}_{\text{motion}}$ and $x^{(t)}_{\text{residual}}$ are the output

Method	UCF-101	HMDB-51
Without optical flow		
Karpathy, Toderici, et al. (2014)	65.4	-
ResNet-50 (He et al., 2016a)	82.3	48.9
(from ST-Mult (Feichtenhofer, Pinz, and R. P. Wildes, 2017))		
ResNet-152 (He et al., 2016a)	83.4	46.7
(from ST-Mult (Feichtenhofer, Pinz, and R. P. Wildes, 2017))		
C3D (Tran, Bourdev, et al., 2015)	82.3	51.6
Res3D (Tran, Ray, et al., 2017)	85.8	54.9
TSN (RGB-only) (Limin Wang et al., 2016)*	85.7	-
TLE (RGB-only) (Diba, Sharma, and Van Gool, 2017)[2]	87.9	54.2
I3D (RGB-only) (Carreira and Zisserman, 2017)*	84.5	49.8
MV-CNN (B. Zhang et al., 2016)	86.4	-
Attentional Pooling (Girdhar and Ramanan, 2017)	-	52.2
CoViAR	**90.4**	**59.1**
With optical flow		
iDT+FV (H. Wang and Schmid, 2013)	-	57.2
Two-Stream (Simonyan and Zisserman, 2014)	88.0	59.4
Two-Stream fusion (Feichtenhofer, Pinz, and Zisserman, 2016)	92.5	65.4
LRCN (Donahue et al., 2015)	82.7	
Composite LSTM Model (Srivastava, Mansimov, and Salakhudinov, 2015)	84.3	44.0
ActionVLAD (Girdhar, Ramanan, et al., 2017)	92.7	66.9
ST-ResNet (Feichtenhofer, Pinz, and R. Wildes, 2016)	93.4	66.4
ST-Mult (Feichtenhofer, Pinz, and R. P. Wildes, 2017)	94.2	68.9
I3D (Carreira and Zisserman, 2017)*	93.4	66.4
TLE (Diba, Sharma, and Van Gool, 2017)[2]	93.8	68.8
L^2STM (L. Sun et al., 2017)	93.6	66.2
ShuttleNet (Y. Shi et al., 2017)	94.4	66.6
STPN (Yunbo Wang et al., 2017)	**94.6**	68.9
TSN (Limin Wang et al., 2016)	94.2	69.4
CoViAR + optical flow	**94.9**	**70.2**

Table 3.6: Action recognition accuracy on UCF-101 (Soomro, Zamir, and M. Shah, 2012) and HMDB-51 (Kuehne et al., 2011). The upper part of the table lists real-time methods that do not require computing optical flow. The lower part of the table lists methods that requires computing optical flow. Our method outperforms all baselines in both settings. Asterisk indicates results evaluated only on split 1 of the datasets. They are listed purely for reference.

feature maps of the motion vector network and the residual-stream network respectively. The input sequence is

$$\left[g\left(x_{\text{RGB}}^{(0)}\right), x_{\text{fusion}}^{(0)}, x_{\text{fusion}}^{(1)}, \dots\right],$$

Method	mAP (%)	wAP (%)
Without optical flow		
ActionVLAD (Girdhar, Ramanan, et al., 2017) (RGB only)	17.6	25.1
Gunnar A. Sigurdsson et al. (2017) (RGB only)	18.3	-
CoViAR	**21.9**	**29.4**
With optical flow		
Two-stream (Simonyan and Zisserman, 2014)	14.3	-
(from Gunnar A Sigurdsson et al. (2016))		
Two-stream (Simonyan and Zisserman, 2014) + iDT (H. Wang and Schmid, 2013)	18.6	-
(from Gunnar A Sigurdsson et al. (2016))		
ActionVLAD (Girdhar, Ramanan, et al., 2017) (RGB only) + iDT	21.0	29.9
Gunnar A. Sigurdsson et al. (2017)	22.4	-
CoViAR + optical flow	**24.1**	**32.3**

Table 3.7: Action recognition accuracy on Charades (Gunnar A Sigurdsson et al., 2016). Without using additional annotations as Gunnar A. Sigurdsson et al. (2017) or complicated feature aggregation, our method achieves the best performance.

where g performs 1×1 convolution so that the dimensionality of the I-frame feature map matches those of the P-frames. We use 512 dimensional hidden states and 3×3 kernels in the Conv-LSTM cell. Due to memory constraint, we subsample one every two P-frames to reduce the length of sequences. Since here we are modeling the original dependency of a compressed video, we use the original compressed signals instead of the accumulated ones as CNN inputs.

RGB-only	Conv-LSTM	Conv-LSTM-Skip	CoViAR
88.4	<u>89.1</u>	87.8	**90.8**

Table 3.8: Accuracy on UCF-101 (split 1). CoViAR decouples the long chain of dependency and outperforms RNN-based models.

Table 3.8 presents the results. While the Conv-LSTM model outperforms traditional methods based on RGB images, the decoupled CoViAR achieves the best accuracy. We also try adding the input of Conv-LSTM to its output as skip connections (as, *e.g.* used in Pixel RNN (Oord, Kalchbrenner, and Kavukcuoglu, 2016)), but it leads to inferior performance ("Conv-LSTM-Skip" in Table 3.8).

3.5.2 *Feature Fusion*

We experiment with different ways of combining P-frame features, $x^{(t)}_{\text{motion}}$ and $x^{(t)}_{\text{residual}}$, with I-frame features $x^{(0)}_{\text{RGB}}$. In particular, we evaluate maximum, mean, multiplicative fusion, concatenation of feature maps, and late fusion (summing softmax scores). Since the I-frame feature maps (from ResNet-152) and P-frame feature maps (from ResNet-18) have different number of channels, Similar to before, for maximum, mean and multiplicative fusion, when needed, we perform 1×1 convolution on I-frame feature maps before fusion, so that the dimensionality matches those of P-frames.

Table 3.9 shows that simply summing scores (late fusion) works the best for CoViAR. Note late fusion allows training of a decoupled model, while the rest requires training multiple CNNs jointly. The ease of training of late fusion may also contribute to its superior performance.

Max	Mean	Mult	Concat	Late
87.9	88.1	87.8	89.7	**90.8**

Table 3.9: Accuracy on UCF-101 (split 1). **Max**: element-wise maximum of feature maps. **Mean**: element-wise average of feature maps. **Mult**: element-wise multiplication of feature maps. **Concat**: concatenating channels of feature maps. **Late** (used in CoViAR): summing of softmax scores (late fusion).

3.5.3 *Confusion Matrix*

Figure 3.9 and Figure 3.10 show the confusion matrices of CoViAR and the model using only RGB images respectively, on UCF-101. Figure 3.11 shows the difference between their predictions. We can see that CoViAR, by using the motion information, corrects many mistakes made by the RGB-based model (off-diagonal purple blocks in Figure 3.11). For example, while the RGB-based model gets confused about the similar actions of *Cricket Bowling* and *Cricket Shot*, our model can distinguish between them.

3.6 POINTS TO PONDER

In this paper, we proposed to train deep networks directly on compressed videos. This is motivated by the *practical* observation that either video compression is essentially free on all modern cameras, due to hardware-accelerated video codecs or that the video is directly *available* in its compressed form. In other words, decompressing the video is actually an inconvenience.

We demonstrate that, quite surprisingly, this is not a drawback but rather a virtue. In particular, video compression reduces irrelevant information from the data, thus rendering it more robust. After all, compression is not meant to affect the content that humans consider pertinent. Secondly, the increased relevance and reduced dimensionality makes computation much more effective (we are able to use much simpler networks for motion vectors and residuals). Finally,

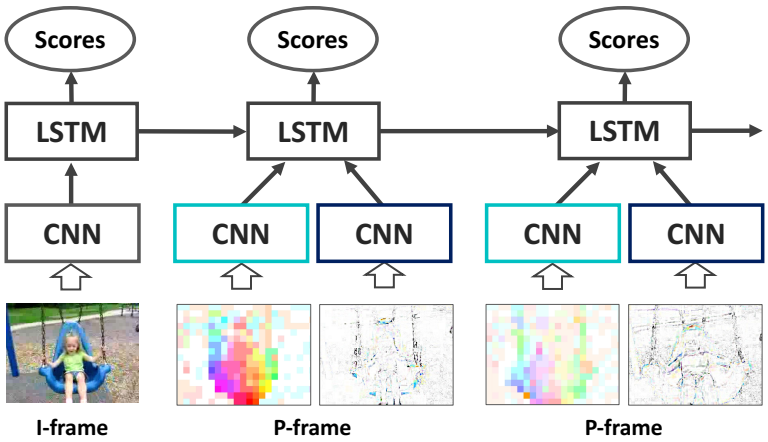


Figure 3.8: An alternative way of modeling compressed videos is to use recurrent neural networks. However, in experiments, we find that the long chain of dependency leads to difficulties in training. A decoupled model works better.

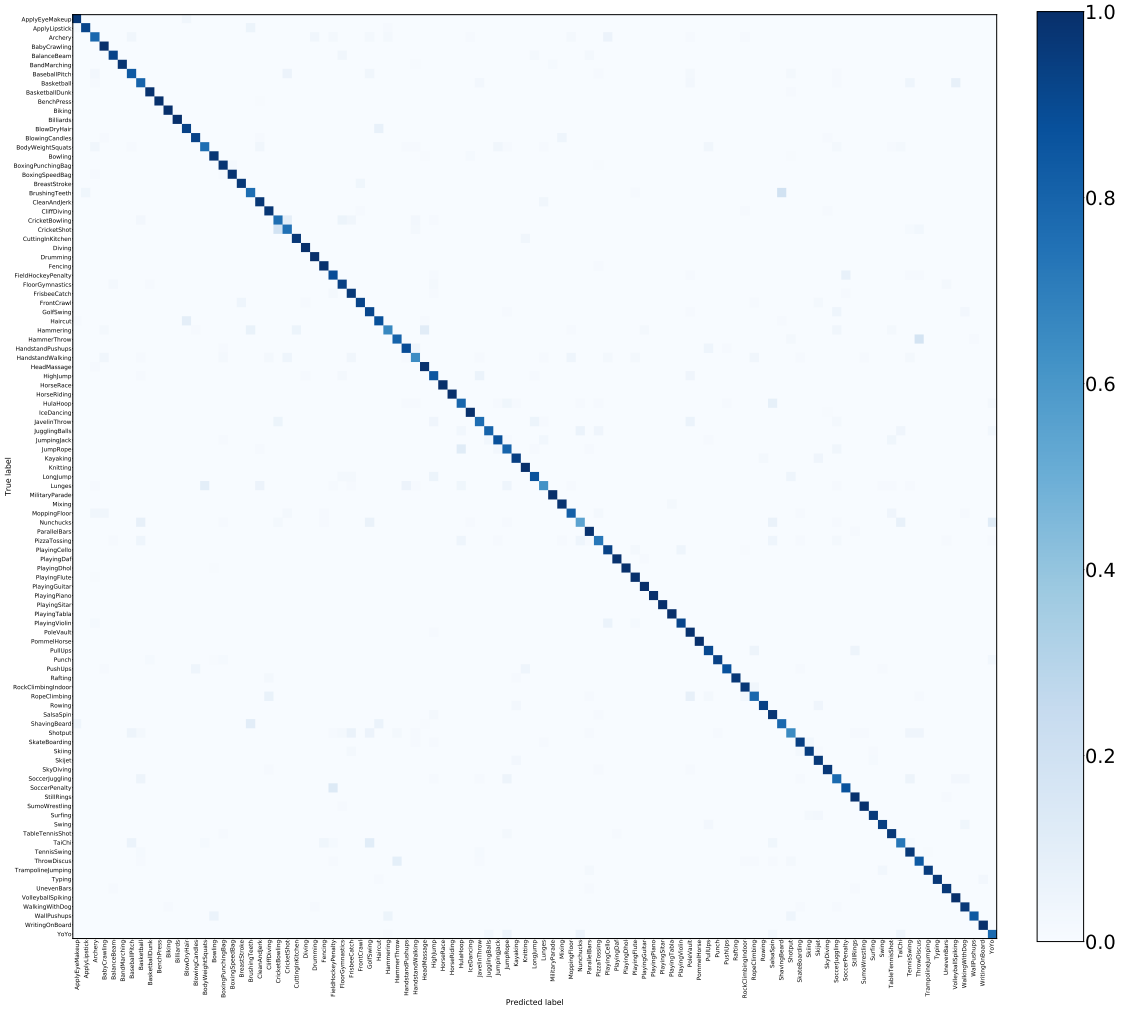


Figure 3.9: Confusion matrix of CoViAR on UCF-101.

the accuracy of the model actually *improves* when using compressed data, yielding new state of the art.

Inspite of instances of success, the proposed method is not a general framework, or a recepie, or automation to handle compressed data. The proposed CoViAR was a carefully handcrafted architecture for compressed videos encoded in MPEG4 compression technique, which has only I and P frames. Whereas many codecs have B-frames and CoViAR cannot out-of-the-box handle them. While with little thinking it is possible to extend CoViAR to handle B-frames, we do not have a fixed recepie to exploit the structure in compressed video yet. Another MpEG4 specific design choice was the final late fusion. It might not work as well with other compression techniques that have very long group of pictures (GOP). Finally, it is still an open question if there can be general machine learning technique to automatically handle compressed data thereby enabling applicability to compressed data in other domains, like audio, MRI, *etc*.

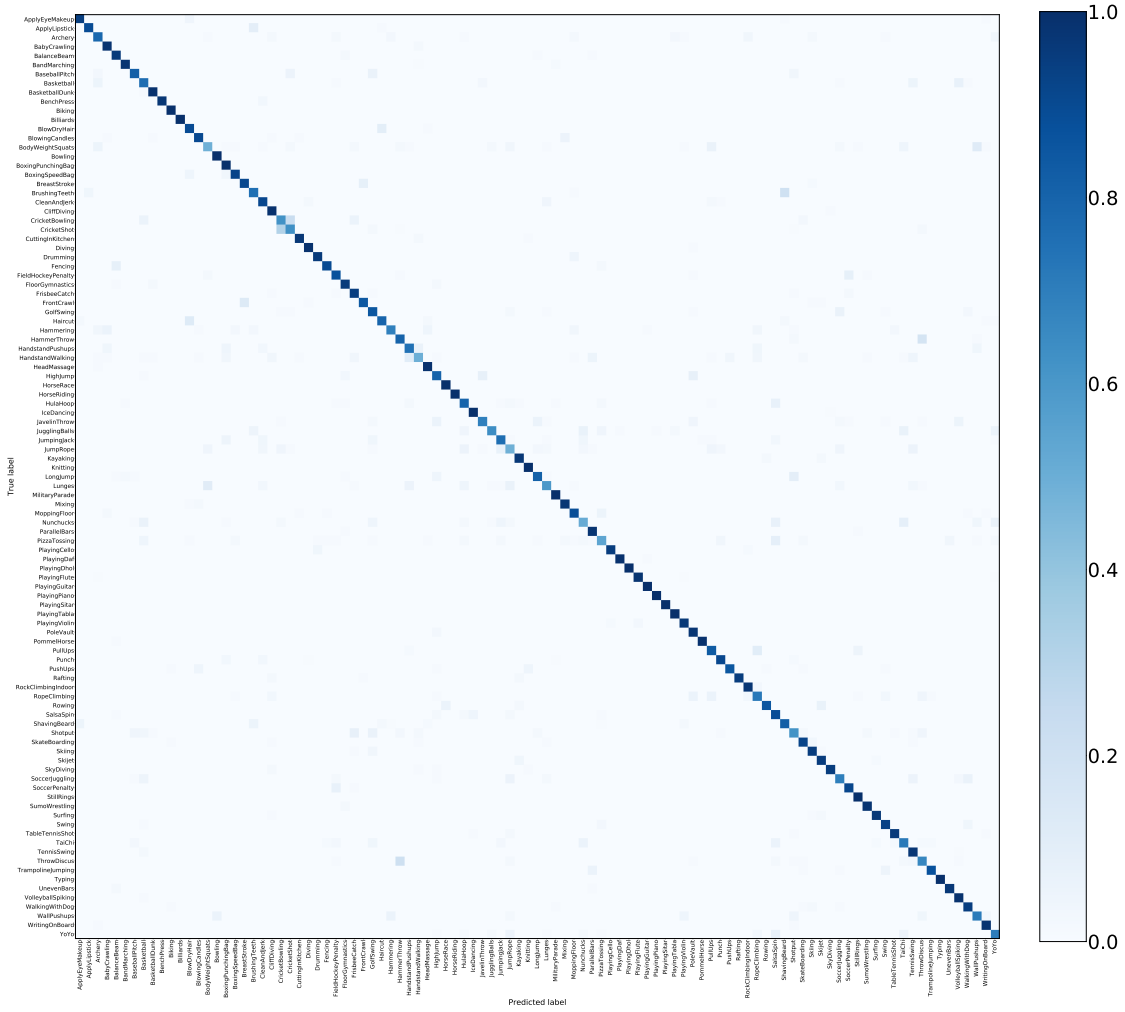


Figure 3.10: Confusion matrix of the model using RGB images on UCF-101.

Despite above mentioned weaknesses, for the specific case of MPEG4 compressed videos, our method is both faster and more accurate, while being simpler to implement than previous works.

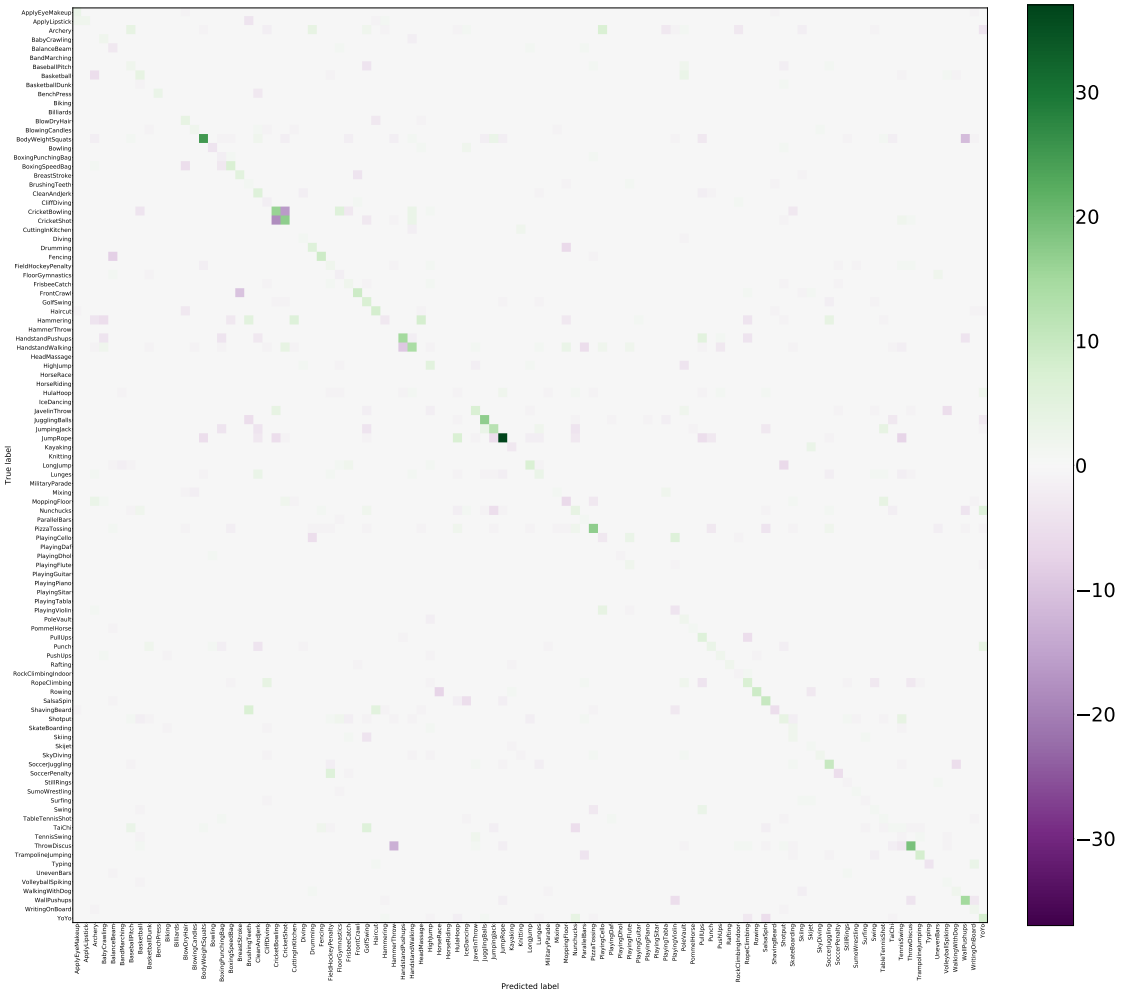


Figure 3.11: Difference between CoViAR's predictions and the RGB-based model's predictions. For diagonal entries, positive values (in green) is better (increase of correct predictions). For off-diagonal entries, negative values (purple) is better (reduction of wrong predictions).

4

Heterogeneous Data

We study the problem of learning representations for multimodal data, that have become ubiquitous. Common examples include video with audio and images with text. Most work on multimodal data focus aim to translate, align, or fuse data from different modalities (Baltrušaitis, Ahuja, and Morency, 2018). There has been, however, limited work explicitly leveraging the *heterogeneous structure* of multimodal data to capture complementary information and still operate in case of missing information in one of the modalities. A beautiful scenario is of question answering (QA) where knowledge source are heterogeneous – raw text or knowledge base (KB). Existing QA methods infer answers either from knowledge base alone or from raw text alone. While KB methods are good at answering compositional questions, their performance is often affected by the incompleteness of the KB. Au contraire, web text contains millions of facts that are absent in the KB, however in an unstructured form. In this chapter, we look at this more practical setting, namely QA over the *heterogeneous* combination of a KB and entity-linked text. We extend universal schema to natural language QA, employing memory networks to attend to the large body of facts in the combination of text and KB. Natural language facts are directly embedded using RNNs, but for embedding KB facts, we build upon recent advances in representation learning leveraging graph structure. We propose novel models which naïvely retrieves KB and text facts by direct matching with question as well as scalable approach for extracting answers from only a question-specific subgraph consisting of both KB facts and text facts. We construct a suite of benchmark tasks for this problem, varying the difficulty of questions, the amount of training data, and KB completeness. We show that proposed method is competitive with the state-of-the-art when tested using either KBs or text only, and vastly outperforms existing methods in the combined setting.

4.1 Introduction

Question Answering (QA) has been a long-standing goal of natural language processing. Two main paradigms evolved in solving this problem: 1) answering questions on a knowledge base (KB); and 2) answering questions using text, such as encyclopedia. Models in either paradigm, however, answer questions using a *single* information source. Intuitively, the suitability of an information source for QA depends on both the *coverage* of the information source and the *difficulty* of extracting the relevant information.

Knowledge bases (KB) contains facts expressed in a fixed schema, facilitating information extraction and even compositional reasoning. These attracted research ever since the early days of computer science, *e.g.* BASEBALL (Green Jr et al., 1961). This problem has matured into learning semantic parsers from parallel question and logical form pairs (Zelle and Raymond J Mooney, 1996; L. S. Zettlemoyer and Collins, 2005), to recent scaling of methods to work on very large KBs like Freebase using question and answer pairs (Jonathan Berant et al., 2013). However, a major drawback of this paradigm is that KBs are highly incomplete (Dong et al., 2014). It is also an open question whether KB relational structure is expressive enough to rep-

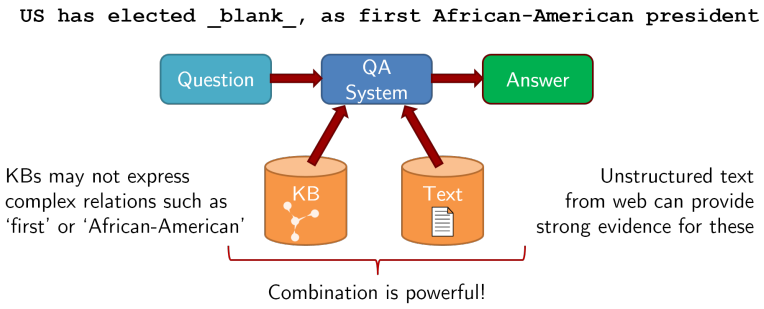


Figure 4.1: A typical QA framework. Existing approaches utilizes only one type of information source. However combination of the heterogeneous information source is powerful as can be seen from the example illustrated. We want to develop a question answering system that can leverage both the information source.

resent world knowledge (Gardner and Krishnamurthy, 2017; Stanovsky, O. Levy, and Dagan, 2014).

A large text corpus has high coverage, but the information is expressed using many different text patterns. The paradigm of exploiting text for questions started in the early 1990s (Kupiec, 1993). With the advent of web, access to text resources became abundant and cheap. Initiatives like TREC QA competitions helped popularizing this paradigm (Voorhees et al., 1999). With the recent advances in deep learning and availability of large public datasets, there has been an explosion of research in a very short time (Choi, Hewlett, et al., 2016; K. Lee et al., 2016; Nguyen et al., 2016; Rajpurkar et al., 2016; Seo, Min, et al., 2016; Trischler et al., 2016; S. Wang and Jing Jiang, 2016; C. Xiong, Zhong, and Socher, 2016). Still, text representation is unstructured and does not allow the compositional reasoning which structured KB supports. As a result, methods which operate on these unstructured patterns (Seo, Kembhavi, et al., 2017) do not generalize beyond their training domains (Dhingra, Pruthi, and Rajagopal, 2018; Wiese, Weissenborn, and Neves, 2017) and have difficulty with compositional reasoning (Talmor and J. Berant, 2018; Welbl, Stenetorp, and Riedel, 2018).

An important but under-explored QA paradigm is where KB and text are exploited together (Ferrucci et al., 2010). Such combination is attractive because text contains millions of facts not present in KB, and a KB's generative capacity represents infinite number of facts that are never seen in text (Figure 4.1). As an illustrative example for power of the combination, consider the question *USA has elected _blank_, our first african-american president* with its answer *Barack Obama*. While Freebase has a predicate for representing presidents of USA, it does not have one for 'african-american' presidents. Whereas in text, we find many sentences describing the presidency of Barack Obama and his ethnicity at the same time. Exploiting both KB and text makes it relatively easy to answer this question than relying on only one of these sources. However QA inference on this combination is challenging due to the structural non-uniformity of KB and text. In this chapter, we focus on a scenario in which a large-scale KB as well as a text corpus is available, but neither is sufficient for answering all questions.

We begin by formally defining the problem of natural language QA using KB+text as the information source and cover relevant work/background (Section 4.2). To use utilize both KB and text facts, we present our solution in Section 4.3, which use universal schema that jointly embeds relevant KB facts and text into a uniform structured representation, allowing interleaved propagation of information. Using the unified representation of all KB+text facts, we employ

memory networks (Weston, Chopra, and Bordes, 2015) to answer the natural language query. In this framework, a simple approach of independently embedding KB and textual facts on the SPADES dataset (Bisk et al., 2016), containing real world fill-in-the-blank questions, outperforms the state-of-the-art semantic parsing baseline, with 8.5 F_1 points. Our analysis shows how individual data sources help fill the weakness of the other, thereby improving overall performance (Section 4.4). This approach of independent embedding, however, ignores the rich relational structure present between the KB and textual facts, thanks to linking (Figure 4.2). One obstacle in utilizing the relational structure is scalability as we operate jointly on all facts present in KB+text, which can be very large. To make the approach more scalable, we propose a method, called GRAFT-Net (Graphs of Relations Among Facts and Text Networks), to extract a 'heterogeneous' question subgraph first and operate on that leveraging the relational structure between KB and text facts (Section 4.5). On WikiMovies (A. H. Miller et al., 2016) and WebQuestionsSP (Yih, M. Richardson, et al., 2016) dataset, we show that our proposed scalable approach have superior performance over varying amount of training supervision, KB completeness, and question complexity (Section 4.6). Finally, we also compare against naïve approaches in KB+text setting that take state-of-the-art QA systems developed for each source, and aggregate their predictions using some heuristic (Baudiš, 2015; Ferrucci et al., 2010). We call such ad-hoc approach *late fusion*, and show that it can be sub-optimal, as models have limited ability to aggregate evidence across the different sources compared to our *early fusion* strategy, which allows more flexibility in combining information from multiple sources. (Section 4.6.4).

4.2 PROBLEM DEFINITION & BACKGROUND

4.2.1 *Task Setup*

A knowledge base is denoted as $\mathcal{K} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, where $\mathcal{V}$ is the set of entities in the KB, and the edges $\mathcal{E}$ are triplets (s, r, o), which states the fact that relation $r \in \mathcal{R}$ holds between the subject $s \in \mathcal{V}$ and object $o \in \mathcal{V}$. A text corpus $\mathcal{D}$ is a set of textual facts $\{d_1, \ldots, d_{|\mathcal{D}|}\}$ where each textual fact is a sequence of words $d_i = (w_1, \ldots, w_{|d_i|})$. We further assume that an (imperfect) entity linking system has been run on the text corpus $\mathcal{D}$ whose output is a set $\mathcal{L}$ of links (v, d_p) connecting an entity $v \in \mathcal{V}$ with a word at position p in the textual fact d. We denote with $\mathcal{L}_d$ the set of entity links for any position in textual fact d.

The task is, given a natural language question $q = (w_1, \ldots, w_{|q|})$, extract its answers $\{a\}_q$ from $\mathcal{G} = (\mathcal{K}, \mathcal{D}, \mathcal{L})$. There may be multiple correct answers for a question (Figure 4.2). In this chapter, we assume that the answers are entities $\mathcal{V}$. Hence the task reduces to classify each member of the set $\mathcal{V}$ as being an answer or not. We also consider a more structured variation of the question, namely cloze style. Here a question q is again words $w_1, w_2, \ldots, w_n$, but one of these words contain a _blank_ and at least an entity. Our goal is to fill in this _blank_ with an answer entity q_a using a knowledge base $\mathcal{K}$ and text $\mathcal{D}$ and the task reduces to classify each member of the set $\mathcal{V}$ as being an answer or not. A few example question answer pairs are shown in Table 4.2. We are interested in a wide range of settings, where the KB $\mathcal{K}$ varies from highly incomplete to complete for answering the questions, and we will introduce datasets for testing our models under these settings.

4.2.2 *Background*

Memory networks (MemNN; Weston, Chopra, and Bordes 2015), which are a class of neural models which have an external memory component for encoding short and long term context, have been successful in both type of individual information source QA. In this approach, we

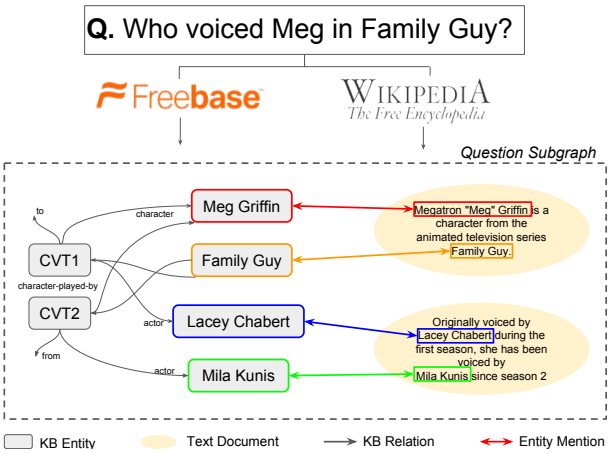


Figure 4.2: To answer a question posed in natural language, GRAFT-Net considers a heterogeneous graph constructed from text and KB facts and can leverage the rich relational structure between the two information sources.

define the memory components as observed cells of an universal schema matrix, and train an end-to-end QA model on question-answer pairs. The cells of universal schema might be as simple as independent embedding of individual facts or more involved ones *leveraging rich relational structure* present between KB and textual facts due to entity linking. We explore both the approaches, but before that briefly describe the pre-requisite and point to references in the literature.

UNIVERSAL SCHEMA Traditionally universal schema is used for relation extraction in the context of knowledge base population. Rows in the schema are formed by entity pairs (e.g. USA, NYC), and columns represent the relation between them. A relation can either be a KB relation, or it could be a pattern of text that exist between these two entities in a large corpus. The embeddings of entities and relation types are learned by low-rank matrix factorization techniques. Riedel et al. (2013) treat textual patterns as static symbols, whereas recent work by Verga et al. (2016) replaces them with distributed representation of sentences obtained by a RNN. Using distributed representation allows reasoning on sentences that are similar in meaning but different on the surface form. We too use this variant to encode our textual relations and shows its applicability to QA.

MEMORY NETWORKS MemNNs are neural attention models with external and differentiable memory. MemNNs decouple the memory component from the network thereby allowing it store external information. Previously, these have been successfully applied to question answering on KB where the memory is filled with distributed representation of KB triples (Bordes et al., 2015), or for reading comprehension (Hill et al., 2016; Sukhbaatar, Szlam, et al., 2015), where the memory consists of distributed representation of sentences in the comprehension. Recently, key-value MemNN are introduced (A. H. Miller et al., 2016) where each memory slot consists of a key and value. The attention weight is computed only by comparing the question with the key memory, whereas the value is used to compute the contextual representation to predict the answer. We use this variant of MemNN for our model. A. H. Miller et al. (2016), in their experiments, store either KB triples or sentences as memories but they do not explicitly model multiple memories containing distinct data sources like we do.

PERSONALIZED PAGERANK PageRank (Page et al., 1999) is a way of measuring 'democratic' importance of nodes in a graph that has no preference for any category of nodes. Random Walk with Restart algorithm (Tong, Faloutsos, and Pan, 2008) and Personalized PageRank (Haveliwala, 2002) were independent works that extended PageRank to handle prior preference for certain nodes. The idea can be explained clearly through an example of document retrival for a given user. Simple PageRank ignores the 'personal preference' that a user may have to a particular set of documents (such as the documents she considers most interesting), but accounting for such preference would improve importance score of the document for that user. Aforementioned works accounted for preferred documents in the random walk by introducing a teleportation probability γ, *i.e.* at each step, the random walk restarts from a 'preferred' node with a probability γ, and with probability $1-\gamma$ continues the random walk along an edge of the current node. Then, it can be shown that in the limit distribution of this random walk would favour the 'preferred' nodes, the pages linked-to by the 'preferred' nodes, pages linked-to in turn, etc. Thus, formally the Personalized PageRank score for a graph with adjacency matrix A is defined as the fixed point R of:

$$R = \gamma R_0 + (1-\gamma)AR \tag{4.1}$$

where R_0 represents the distribution over the 'preferred' nodes. In case of QA, the Personalized PageRank can allow us to retrieve not only the documents that explicitly contain entities present in the question, but also other highly relevant documents.

GRAPH CONVOLUTION NETWORK The popular versions of Graph Convolution Networks (GCN) are neural networks operating on graphs with weights tied in a particular fashion (Duvenaud et al., 2015). Despite its name, the relation to convolution is limited (Huszar, 2016). However, GCN has been successful in learning node representations encoding features from local, structured neighborhoods, and has led to significant improvements in graph classification (Duvenaud et al., 2015; Schlichtkrull et al., 2017) and graph-based semi-supervised learning (Kipf and Welling, 2016). (Schlichtkrull et al., 2017). Such models follow the standard gather-apply-scatter paradigm to learn the node representation with homogeneous updates, *i.e.* treating all neighbors equally. The basic recipe for these models is as follows:

1. Initialize node representations $h_v^{(0)}$.
2. For $l = 1, \ldots, L$ update node representations

$$h_v^{(l)} = \phi\left(h_v^{(l-1)}, \sum_{v' \in N_r(v)} h_{v'}^{(l-1)}\right),$$

   where $N_r(v)$ denotes the neighbours of $v \in \mathcal{V}$ along incoming edges of type $r \in \mathcal{R}$, and ϕ is a neural network layer.

Here L is the number of *layers* in the model and corresponds to the maximum length of the paths along which information should be propagated in the graph. Once the propagation is complete the final layer representations $h_v^{(L)}$ are used to perform the desired task, for example link prediction in knowledge bases (Schlichtkrull et al., 2017). A newly designed variation of GCN would be helpful for learning representation in the *heterogeneous* KB+text graph.

4.2.3 *Related Works*

A majority of the QA literature that focused on exploiting KB and text either improves the inference on the KB using text based features (Choi, Kwiatkowski, and L. Zettlemoyer, 2015; Guu,

J. Miller, and P. Liang, 2015; Joshi, Sawant, and Chakrabarti, 2014; Krishnamurthy and Mitchell, 2012; Neelakantan, Roth, and Andrew McCallum, 2015; Reddy, Lapata, and Steedman, 2014; Savenkov and Agichtein, 2016; Kun Xu, Reddy, et al., 2016; X. Yao and Van Durme, 2014; Yih, M.-W. Chang, et al., 2015) or improves the inference on text using KB (Huan Sun et al., 2015).

Limited work exists on exploiting text and KB jointly for question answering. Gardner and Krishnamurthy (2017) is the closest to ours who generate a open-vocabulary logical form and rank candidate answers by how likely they occur with this logical form both in Freebase and text. Our models are trained on a weaker supervision signal without requiring the annotation of the logical forms. Ryu, Jang, and H.-K. Kim (2014) use a pipelined system aggregating evidence from both unstructured and semi-structured sources for open-domain QA. This approach falls into 'late-fusion' category and thus is sub-optimal as discussed earlier.

A few QA methods infer on curated databases combined with OpenIE triples (Fader, L. Zettlemoyer, and Etzioni, 2014; Kun Xu, Y. Feng, et al., 2016; Yahya et al., 2016). Our work differs from them in two ways: 1) we do not need an explicit database query to retrieve the answers (Andreas et al., 2016; Neelakantan, Q. V. Le, and Sutskever, 2015); and 2) our text-based facts retain complete sentential context unlike the OpenIE triples (Banko et al., 2007; Carlson et al., 2010).

Another line of work has looked at learning combined representations of KBs and text for relation extraction and Knowledge Base Completion (KBC) (Das, Neelakantan, et al., 2017; X. Han, Zhiyuan Liu, and M. Sun, 2016; Lao et al., 2012; Riedel et al., 2013; Toutanova, D. Chen, et al., 2015; Verga et al., 2016). The key difference in QA compared to KBC is that in QA the inference process on the knowledge source has to be conditioned on the question, so different questions induce different representations of the KB and warrant a different inference process. Furthermore, KBC operates under the fixed schema defined by the KB before-hand, whereas natural language questions might not adhere to this schema.

The GRAFT-Net model itself is motivated from the large body of work on graph representation learning (Atwood and Towsley, 2016; Kipf and Welling, 2016; Y. Li et al., 2016; Scarselli et al., 2009; Schlichtkrull et al., 2017). Like most other graph-based models, GRAFT-Nets can also be viewed as an instantiation of the Message Passing Neural Network (MPNN) framework of Gilmer et al. (2017). GRAFT-Nets are also *inductive* representation learners like GraphSAGE (Hamilton, Ying, and Leskovec, 2017), but operate on a heterogeneous mixture of nodes and use retrieval for getting a subgraph instead of random sampling. The recently proposed Walk-Steered Convolution model uses random walks for learning graph representations (Jiatao Jiang et al., 2018). Our personalization technique also borrows from such random walk literature, but uses it to localize propagation of embeddings.

Tremendous progress on QA over KB has been made with deep learning based approaches of memory networks (Bordes et al., 2015; Jain, 2016) and reinforcement learning (Das, Dhuliawala, et al., 2018; C. Liang et al., 2017). But extending them with text is non-trivial, which is our main focus. In the other direction, there is also work on producing parsimonious graphical representations of textual data (S. Krause et al., 2016; Lu et al., 2017), but in this paper we use a simple sequential representation augmented with entity links to the KB to work well.

For QA over text only, a major focus has been on the task of reading comprehension (Gong and Bowman, 2017; M. Hu, Y. Peng, and Qiu, 2017; Seo, Kembhavi, et al., 2017; Shen et al., 2017; A. W. Yu et al., 2018), since the introduction of the SQuAD (Rajpurkar et al., 2016). These systems assume answer-containing passage is known apriori, but there has been progress when this assumption is relaxed (D. Chen et al., 2017; Dhingra, Mazaitis, and W. W. Cohen, 2017; Raison et al., 2018; S. Wang, M. Yu, X. Guo, et al., 2018; S. Wang, M. Yu, Jing Jiang, et al., 2017;

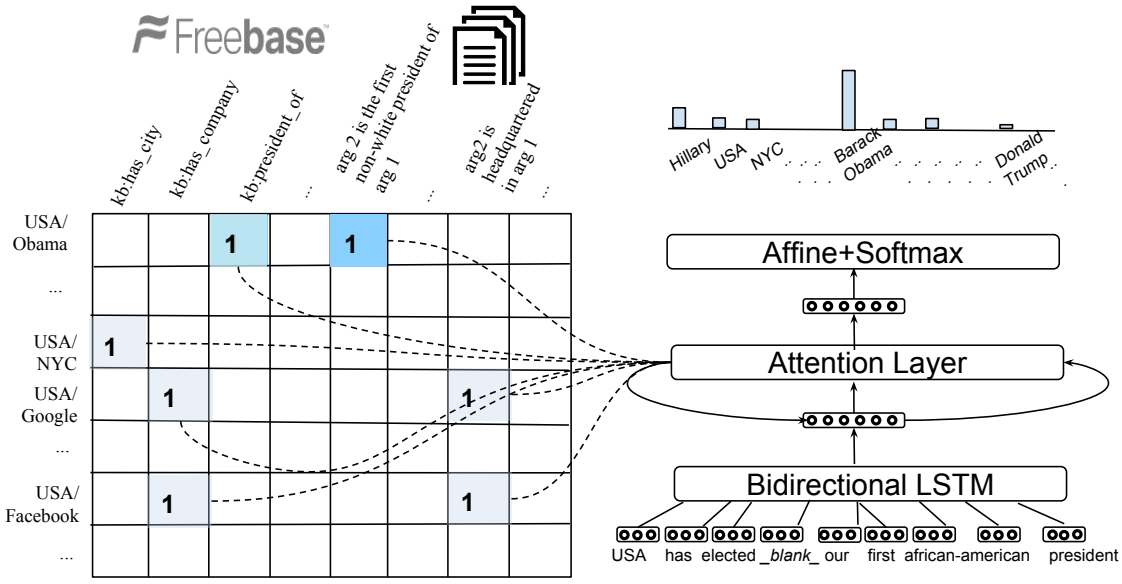


Figure 4.3: Memory network attending the facts in the universal schema (matrix on the left). The color gradients denote the attention weight on each fact.

Watanabe, Dhingra, and R. Salakhutdinov, 2017). We also work in the latter setting, where relevant information must be first retrieved from large information sources, but we go beyond and incorporate KBs into this process as well.

4.3 SIMPLE MODEL

Our model is a MemNN with universal schema as its memory with independent embedding of KB and textual facts. Figure 4.3 shows the model architecture.

4.3.1 *Fact Retrieval*

We retrieve the relevant facts using two parallel pipelines – one over the KB $\mathcal{K}$, and the other over corpus $\mathcal{D}$ which returns a set of documents. We first perform entity linking on the question q. This set of entities is denoted by S_q.

KB We retrieve all KB facts of the form (s, r, o) where either $s, r \in S_q$.

TEXT We retrieve all textual facts which contain any entity in S_q.

4.3.2 *Heterogeneous Embedding*

MEMORY: Our memory $\mathcal{M}$ comprise of both KB and textual triples from universal schema. Each memory cell is in the form of key-value pair. Let $(s, r, o) \in \mathcal{K}$ represent a KB triple. We represent this fact with distributed key $\mathbf{k} \in \mathbb{R}^{2d}$ formed by concatenating the embeddings $\mathbf{s} \in \mathbb{R}^d$ and $\mathbf{r} \in \mathbb{R}^d$ of subject entity s and relation r respectively. The embedding $\mathbf{o} \in \mathbb{R}^d$ of object entity o is treated as its value $\mathbf{v}$.

Let $(s, [w_1, \ldots, arg_1, \ldots, arg_2, w_n], o) \in \mathcal{T}$ represent a textual fact, where arg_1 and arg_2 correspond to the positions of the entities 's' and 'o'. We represent the key as the sequence formed by replacing arg_1 with 's' and arg_2 with a special '_blank_' token, i.e., k =

$[w_1, \ldots, s, \ldots, \text{\_blank\_}, w_n]$ and value as just the entity 'o'. We convert k to a distributed representation using a bidirectional LSTM (Graves and Jürgen Schmidhuber, 2005; Hochreiter and Jürgen Schmidhuber, 1997), where $\mathbf{k} \in \mathbb{R}^{2d}$ is formed by concatenating the last states of forward and backward LSTM, i.e., $\mathbf{k} = \left[\overrightarrow{\text{LSTM}}(k); \overleftarrow{\text{LSTM}}(k)\right]$. The value $\mathbf{v}$ is the embedding of the object entity o. Projecting both KB and textual facts to $\mathbb{R}^{2d}$ offers a unified view of the knowledge to reason upon. In Figure 4.3, each cell in the matrix represents a memory containing the distributed representation of its key and value.

QUESTION ENCODER: A bidirectional LSTM is also used to encode the input question q to a distributed representation $\mathbf{q} \in \mathbb{R}^{2d}$ similar to the key encoding step above.

ATTENTION OVER CELLS: We compute attention weight of a memory cell by taking the dot product of its key $\mathbf{k}$ with a contextual vector $\mathbf{c}$ which encodes most important context in the current iteration. In the first iteration, the contextual vector is the question itself. We only consider the memory cells that contain at least one entity in the question. For example, for the input question in Figure 4.3, we only consider memory cells containing USA. Using the attention weights and values of memory cells, we compute the context vector $\mathbf{c_t}$ for the next iteration t as follows:

$$\mathbf{c_t} = \mathbf{W_t}\left(\mathbf{c_{t-1}} + \mathbf{W_p}\sum_{(k,v)\in\mathcal{M}}(\mathbf{c_{t-1}}\cdot\mathbf{k})\mathbf{v}\right)$$

where $\mathbf{c_0}$ is initialized with question embedding $\mathbf{q}$, $\mathbf{W_p}$ is a projection matrix, and $\mathbf{W_t}$ represents the weight matrix which considers the context in previous hop and the values in the current iteration based on their importance (attention weight). This multi-iterative context selection allows multi-hop reasoning without explicitly requiring a symbolic query representation.

4.3.3 *Cross-Normalization*

To make the representations from both information source have similar distribution, we apply a trick similar to *batch normalization* (Ioffe and Szegedy, 2015). We normalize the mean and variance of the textual fact embeddings and then scale and shift to match the mean and variance of the KB fact embeddings. We term this transformation 'cross-entropy'. Empirically, this stabilized the training and gave a boost in the final performance.

Furthermore, as neural networks are lazy, they latch on to strongest signals, even though it may lead to poorer performance. When jointly training on both KB and textual facts, the signal from KB would be clearer (atleast for a sizable portion of the dataset), and thus network would focus only on KB facts and ignore text facts. If we pre-train the model using text only, then during training with KB facts, the network sees strong signal from KB and undergoes a catastrophic forgetting of using textual facts. To train the UNISCHEMA model successfully circumventing the aforementioned obstacle, we pre-train the network on KB facts only and subsequently train for textual fact. We found that this is crucial in making the UNISCHEMA to work.

4.3.4 *Answer Selection*

The final contextual vector c_t is used to select the answer entity q_a (among all 1.8M entities in the dataset) which has the highest inner product with it.

$$\mathbf{q_a} = \underset{e}{\text{argmax}}\ \mathbf{e}\cdot\mathbf{c_t}$$

Finally, we minimize the cross-entropy loss to train the complete MemNN in an end-to-end fashion.

4.4 EXPERIMENTS ON SIMPLE MODEL

We describe our evaluation datasets, implementation details, baseline models and results.

4.4.1 *Dataset*

We use Freebase (Bollacker et al., 2008) as our KB, and ClueWeb (Gabrilovich, Ringgaard, and Subramanya, 2013) as our text source to build universal schema. For evaluation, literature offers two options: 1) datasets for text-based question answering tasks such as answer sentence selection and reading comprehension; and 2) datasets for KB question answering.

Although the text-based question answering datasets are large in size, e.g., SQuAD (Rajpurkar et al., 2016) has over 100k questions, answers to these are often not entities but rather sentences which are not the focus of our work. Moreover these texts may not contain Freebase entities at all, making these skewed heavily towards text. Coming to the alternative option, WebQuestions Jonathan Berant et al. (2013) is widely used for QA on Freebase. This dataset is curated such that all questions can be answered on Freebase alone. But since our goal is to explore the impact of universal schema, testing on a dataset completely answerable on a KB is not ideal. WikiMovies dataset (A. H. Miller et al., 2016) also has similar properties. Gardner and Krishnamurthy (2017) created a dataset with motivations similar to ours, however this is not publicly released during the submission time.

Instead, we use SPADES (Bisk et al., 2016) as our evaluation data which contains fill-in-the-blank cloze-styled questions created from ClueWeb. This dataset is ideal to test our hypothesis for following reasons: 1) it is large with 93K sentences and 1.8M entities; and 2) since these are collected from Web, most sentences are natural. A limitation of this dataset is that it contains only the sentences that have entities connected by at least one relation in Freebase, making it skewed towards Freebase as we will see (Section 4.4.3). We use the standard train, dev and test splits for our experiments. For text part of universal schema, we use the sentences present in the training set.

4.4.2 *Compared Models*

We evaluate following models to measure impact of different knowledge sources for QA.

ONLYKB: In this model, MemNN memory contains only the facts from KB. For each KB triple (e_1, r, e_2), we have two memory slots, one for (e_1, r, e_2) and the other for its inverse (e_2, r^i, e_1).

ONLYTEXT: SPADES contains sentences with blanks. We replace the blank tokens with the answer entities to create textual facts from the training set. Using every pair of entities, we create a memory cell similar to as in universal schema.

Model	Dev. F_1	Test F_1
Bisk et al. (2016)	32.7	31.4
OnlyKB	39.1	38.5
OnlyText	25.3	26.6
Ensemble.	39.4	38.6
UniSchema	**41.1**	**39.9**

Table 4.1: QA results on SPADES.

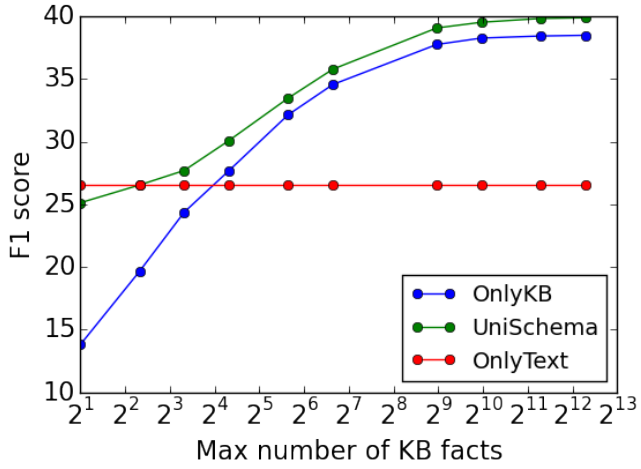


Figure 4.4: Performance on varying the number of available KB facts during test time. UniSchema model consistently outperforms OnlyKB

ENSEMBLE This is an ensemble of the above two models. We use a linear model that combines the scores from, and use an ensemble to combine the evidences from individual models. An instantiation of so-called 'late-fussion' models.

UNISCHEMA This is our main model with universal schema as its memory, i.e., it contains memory slots corresponding to both KB and textual facts.

The dimensions of word, entity and relation embeddings, and LSTM states were set to d =50. The word and entity embeddings were initialized with word2vec (Mikolov, Sutskever, et al., 2013) trained on 7.5 million ClueWeb sentences containing entities in Freebase subset of SPADES. The network weights were initialized using Xavier initialization (Glorot and Y. Bengio, 2010). We considered up to a maximum of 5k KB facts and 2.5k textual facts for a question. We used Adam (D. P. Kingma and Ba, 2014) with the default hyperparameters (learning rate=1e-3, β_1=0.9, β_2=0.999, ϵ=1e-8) for optimization. To overcome exploding gradients, we restricted the magnitude of the ℓ_2 norm of the gradient to 5. The batch size during training was set to 32.

4.4.3 *Main Results*

Table 4.1 shows the main results on SPADES. UniSchema outperforms all our models validating our hypothesis that exploiting universal schema for QA is better than using either KB or

Question	Answer
1. USA have elected _blank_, our first african-american president.	Obama
2. Angelina has reportedly been threatening to leave _blank_.	Brad_Pitt
3. Spanish is more often a second and weaker language among many _blank_.	Latinos
4. _blank_ is the third largest city in the United_States.	Chicago
5. _blank_ was Belshazzar 's father.	Nabonidus

Table 4.2: A few questions on which OnlyKB fails to answer but UniSchema succeeds.

Model	Dev. F_1
OnlyKB correct	39.1
OnlyText correct	25.3
UniSchema correct	41.1
OnlyKB or OnlyText got it correct	45.9
Both OnlyKB and OnlyText got it correct	18.5
OnlyKB got it correct and OnlyText did not	20.6
OnlyText got it correct and OnlyKB did not	6.80
Both UniSchema and OnlyKB got it correct	34.6
UniSchema got it correct and OnlyKB did not	6.42
OnlyKB got it correct and UniSchema did not	4.47
Both UniSchema and OnlyText got it correct	19.2
UniSchema got it correct and OnlyText did not	21.9
OnlyText got it correct and UniSchema did not	6.09

Table 4.3: Detailed results on Spades.

text alone. Despite Spades creation process being friendly to Freebase, exploiting text still provides a significant improvement. Table 4.2 shows some of the questions which UniSchema answered but OnlyKB failed. These can be broadly classified into (a) relations that are not expressed in Freebase (e.g., african-american presidents in sentence 1); (b) intentional facts since curated databases only represent concrete facts rather than intentions (*e.g.* threating to leave in sentence 2); (c) comparative predicates like first, second, largest, smallest (e.g., sentences 3 and 4); and (d) providing additional type constraints (e.g., in sentence 5, Freebase does not have a special relation for *father*. It can be expressed using the relation *parent* along with the type constraint that the answer is of gender *male*).

We have also analyzed the nature of UniSchema attention. In 58.7% of the cases the attention tends to prefer KB facts over text. This is as expected since KBs facts are concrete and accurate than text. In 34.8% of cases, the memory prefers to attend text even if the fact is already present in the KB. For the rest (6.5%), the memory distributes attention weight evenly, indicating for some questions, part of the evidence comes from text and part of it from KB. Table 4.3 gives a more detailed quantitative analysis of the three models in comparison with each other.

To see how reliable is UniSchema, we gradually increased the coverage of KB by allowing only a fixed number of randomly chosen KB facts for each entity. As Figure 4.4 shows, when the KB coverage is less than 16 facts per entity, UniSchema outperforms OnlyKB by a wide-margin indicating UniSchema is robust even in resource-scarce scenario, whereas OnlyKB is very sensitive to the coverage. UniSchema also outperforms Ensemble showing joint modeling is superior to ensemble on the individual models. We also achieve the state-of-the-art with 8.5 F_1 points difference. Bisk et al. (2016) use graph matching to convert natural language to Freebase queries whereas even without an explicit query representation, we outperform them.

4.5 GRAFT-NETS

The simple model ignores the rich relational *structure* present in the data induced by the entity linking. To leverage this structure we propose a novel graph convolution based neural network, called GRAFT-Net (Graphs of Relations Among Facts and Text Networks), specifically designed to operate over heterogeneous graphs of KB facts and text sentences. We build upon recent work on graph representation learning (Kipf and Welling, 2016; Schlichtkrull et al., 2017), but propose two key modifications to adopt them for the task of QA. First, we propose *heterogeneous update rules* that handle KB nodes differently from the text nodes: for instance, LSTM-based updates are used to propagate information into and out of text nodes. Second, we introduce a *directed propagation method*, inspired by personalized Pagerank in IR (Haveliwala, 2002), which constrains the propagation of embeddings in the graph to follow paths starting from seed nodes linked to the question. Empirically, we show that both these extensions are crucial for the task of QA.

We begin by recognizing that the question q and its answers $\{a\}_q$ induce a labeling of the nodes in $\mathcal{V}_q$: we let $y_v = 1$ if $v \in \{a\}_q$ and $y_v = 0$ otherwise for all $v \in \mathcal{V}_q$. The task of QA then reduces to performing binary classification over the nodes of the graph $\mathcal{G}_q$. Several graph-propagation based models have been proposed in the literature which learn node representations and then perform classification of the nodes (Kipf and Welling, 2016; Schlichtkrull et al., 2017).

However, there are three differences in our setting from previously studied graph-based classification tasks. The first difference is that, in our case, the graph $\mathcal{G}_q$ consists of *heterogeneous* nodes. Some nodes in the graph correspond to KB entities which represent symbolic objects, whereas other nodes represent textual documents which are variable length sequences of words.

The second difference is that we want to condition the representation of nodes in the graph on the natural language question q. In Section 4.5.3 we introduce heterogeneous updates to address the first difference, and in Section 4.5.4 we introduce mechanisms for conditioning on the question (and its entities) for the second.

Finally, operating on entire text+KB graph is not scalable and only retrieving KB and textual facts directly containing entities mentioned in the question might miss out many relevant and necessary facts to answer the question or contain many spurious facts, especially for high degree nodes. As the relation queried for by the question may exist between many pairs of entities: we also want to only focus on relations that *connect the question to a potential answer*. Hence, we also introduce a personalization technique to retrieve facts (Section 4.5.1) and also use it to constrains the updates to only happen along paths starting from a seed nodes in the question (Section 4.5.4).

4.5.1 *Question Subgraph Retrieval*

We retrieve the subgraph $\mathcal{G}_q$ using two parallel pipelines – one over the KB $\mathcal{K}$ which returns a set of entities, and the other over the corpus $\mathcal{D}$ which returns a set of documents. The retrieved entities and documents are then combined with entity links to produce a fully-connected graph.

KB RETRIEVAL. To retrieve relevant entities from the KB we first perform entity linking on the question q. This set of *seed entities* is denoted S_q. Next we run the Personalized PageRank (PPR) method (Haveliwala, 2002) around these seeds to identify other entities which might be an answer to the question. The edge-weights around S_q are distributed equally among all edges of the same type, and they are weighted such that edges relevant to the question receive a higher weight than those which are not. Specifically, we average word vectors to compute

a relation vector $v(r)$ from the surface form of the relation, and a question vector $v(q)$ from the words in the question, and use cosine similarity between these as the edge weights. After running PPR we retain the top E entities $v_1, \ldots, v_E$ by PPR score, along with any edges between them, and add them to $\mathcal{G}_q$.

TEXT RETRIEVAL. We use Wikipedia as the corpus and retrieve textual facts at the sentence level, *i.e.* facts in $\mathcal{D}$ are defined along sentences boundaries. We perform text retrieval in two steps: first we retrieve the top 5 most relevant Wikipedia articles, using the weighted bag-of-words model from DrQA (D. Chen et al., 2017); then we populate a Lucene[1] index with sentences from these articles, and retrieve the top ranking ones $d_1, \ldots, d_D$, based on the words in the question. For the sentence-retrieval step, we found it beneficial to include the title of the article as an additional field in the Lucene index. As most sentences in an article talk about the title entity, this helps in retrieving relevant sentences that do not explicitly mention the entity in the question. We add the retrieved sentence, along with any entities linked to them, to the subgraph $\mathcal{G}_q$.

The final question subgraph is $\mathcal{G}_q = (\mathcal{V}_q, \mathcal{E}_q, \mathcal{R}^+)$, where the vertices $\mathcal{V}_q$ consist of all the retrieved entities and sentences, i.e. $\mathcal{V}_q = \{v_1, \ldots, v_E\} \cup \{d_1, \ldots, d_D\}$. The edges are all relations from $\mathcal{K}$ among these entities, plus the entity-links between sentences and entities, i.e.

$$\begin{aligned}\mathcal{E}_q =& \{(s, o, r) \in \mathcal{E} : s, o \in \mathcal{V}_q, r \in \mathcal{R}\} \\ & \cup \{(v, d_p, r_L) : (v, d_p) \in \mathcal{L}_d, d \in \mathcal{V}_q\},\end{aligned}$$

where r_L denotes a special "linking" relation. $\mathcal{R}^+ = \mathcal{R} \cup \{r_L\}$ is the set of all edge types in the subgraph.

4.5.2 *Node Initialization*

Nodes corresponding to entities are initialized using fixed-size vectors $h_v^{(0)} = x_v \in \mathbb{R}^n$, where x_v can be pre-trained KB embeddings or random, and n is the embedding size. Textual nodes in the graph describe a variable length sequence of text. Since multiple entities might link to different positions in the document, we maintain a variable length representation of document in each layer. This is denoted by $H_d^{(l)} \in \mathbb{R}^{|d| \times n}$. Given the words in the textual fact $(w_1, \ldots, w_{|d|})$, we initialize its hidden representation as:

$$H_d^{(0)} = \text{LSTM}(w_1, w_2, \ldots),$$

where LSTM refers to a long short-term memory unit. With a slight abuse of notation, we refer to the embedding of p-th word in the textual fact d at layer l as $H_{d,p}^{(l)}$.

4.5.3 *Heterogeneous Updates*

ENTITIES Let $M(v) = \{(d, p)\}$ be the set of positions p in textual fact d which correspond to a mention of entity v. The update for entity nodes involves a single-layer feed-forward network (FFN) over the concatenation of four states:

$$h_v^{(l)} = \text{FFN}\left(\begin{bmatrix} h_v^{(l-1)} \\ h_q^{(l-1)} \\ \sum_r \sum_{v' \in N_r(v)} \alpha_r^{v'} \psi_r(h_{v'}^{(l-1)}) \\ \sum_{(d,p) \in M(v)} H_{d,p}^{(l-1)} \end{bmatrix}\right). \tag{4.2}$$

1 https://lucene.apache.org/

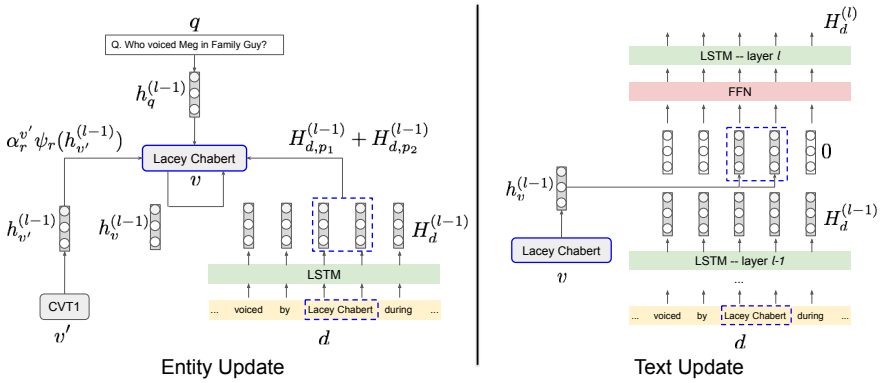


Figure 4.5: Illustration of the heterogeneous update rules for entities (**left**) and text documents (**right**)

The first two terms correspond to the entity representation and question representation (details below), respectively, from the previous layer.

The third term aggregates the states from the entity neighbours of the current node $N_r(v)$ after scaling with an attention weight $\alpha_r^{v'}$ (described in the next section), and applying relation specific transformations ψ_r. Previous work on Relational-Graph Convolution Networks (Schlichtkrull et al., 2017) used a linear projection for ψ_r. For a batched implementation, this results in matrices of size $O(B|\mathcal{R}_q||\mathcal{E}_q|n)$, where B is the batch size, which can be prohibitively large for large subgraphs.[2] Hence in this work we use *relation vectors* x_r for $r \in \mathcal{R}_q$ instead of matrices, and compute the update along an edge as:

$$\psi_r(h_{v'}^{(l-1)}) = p_{v'}^{(l-1)} \text{FFN}\left(x_r, h_{v'}^{(l-1)}\right). \tag{4.3}$$

The memory complexity of the above is $O(B(|\mathcal{F}_q| + |\mathcal{E}_q|)n)$, where $|\mathcal{F}_q|$ is the number of facts in the subgraph $\mathcal{G}_q$. Here $p_{v'}^{(l-1)}$ is a PageRank score used to control the propagation of embeddings along paths starting from the seed nodes, which we describe in detail in the next section.

The last term aggregates the states of all tokens that correspond to mentions of the entity v among the documents in the subgraph. Note that the update depends on the positions of entity linking—this is to disambiguate between multiple entities mentioned in the same document.

SENTENCES Let $L(d, p)$ be the set of all entities linked to the word at position p in textual fact d. Then, the textual node update proceeds in two steps. First we aggregate over the entity states coming in at each position separately:

$$\tilde{H}_{d,p}^{(l)} = \text{FFN}\left(H_{d,p}^{(l-1)}, \sum_{v \in L(d,p)} h_v^{(l-1)}\right). \tag{4.4a}$$

Here $h_v^{(l-1)}$ are normalized by the number of outgoing edges at v. Next we aggregate states within the document using an LSTM:

$$H_d^{(l)} = \text{LSTM}(\tilde{H}_d^{(l)}). \tag{4.4b}$$

2 This is because we have to use adjacency matrices of size $|\mathcal{R}_q| \times |\mathcal{E}_q| \times |\mathcal{E}_q|$ to aggregate embeddings from neighbours of all nodes simultaneously.

4.5.4 *Conditioning on Question*

For the parts described thus far, the graph learner is largely agnostic of the question. We introduce dependence on question in two ways: by attention over relations, and by personalized propagation.

To represent q, let $w_1^q, \ldots, w_{|q|}^q$ be the words in the question. Initial representation is computed as:

$$h_q^{(0)} = \text{LSTM}(w_1^q, \ldots, w_{|q|}^q)_{|q|} \in \mathbb{R}^n, \tag{4.5}$$

where we extract the final state from the output of the LSTM. In subsequent layers the question representation is updated as $h_q^{(l)} = \text{FFN}\left(\sum_{v \in S_q} h_v^{(l)}\right)$, where S_q denotes the seed entities mentioned in the question.

ATTENTION OVER RELATIONS. The attention weight in the third term of (4.2) is computed using the question and relation embeddings:

$$\alpha_r^{v'} \propto \exp(x_r^T h_q^{(l-1)}),$$

where the normalization is over all outgoing edges from v'. This ensures that embeddings are propagated more along edges relevant to the question.

DIRECTED PROPAGATION. Many questions require multi-hop reasoning, which follows a path from a seed node mentioned in the question to the target answer node. To encourage such a behaviour when propagating embeddings, we develop a technique inspired from personalized PageRank in IR (Haveliwala, 2002). The propagation starts at the seed entities S_q mentioned in the question. In addition to the vector embeddings $h_v^{(l)}$ at the nodes, we also maintain scalar "PageRank" scores $p_v^{(l)}$ which measure the total weight of paths from a seed entity to the current node, as follows:

$$p_v^{(0)} = \begin{cases} \frac{1}{|S_q|} & \text{if} \quad v \in S_q \\ 0 & \text{o.w.} \end{cases},$$

$$p_v^{(l)} = (1-\lambda) p_v^{(l-1)} + \lambda \sum_r \sum_{v' \in N_r(v)} \alpha_r^{v'} p_{v'}^{(l-1)}.$$

Notice that we reuse the attention weights $\alpha_r^{v'}$ when propagating PageRank, to ensure that nodes along paths relevant to the question receive a high weight. The PageRank score is used as a scaling factor when propagating embeddings along the edges in (4.3). For $l = 1$, the PageRank score will be 0 for all entities except the seed entities, and hence propagation will only happen outward from these nodes. For $l = 2$, it will non-zero for the seed entities and their 1-hop neighbors, and propagation will only happen along these edges.

4.5.5 *Answer Selection*

The final representations $h_v^{(L)} \in \mathbb{R}^n$, is used for binary classification to select the answers:

$$\Pr(v \in \{a\}_q | \mathcal{G}_q, q) = \sigma(w^T h_v^{(L)} + b), \tag{4.6}$$

where σ is the sigmoid function. Training uses binary cross-entropy loss over these probabilities.

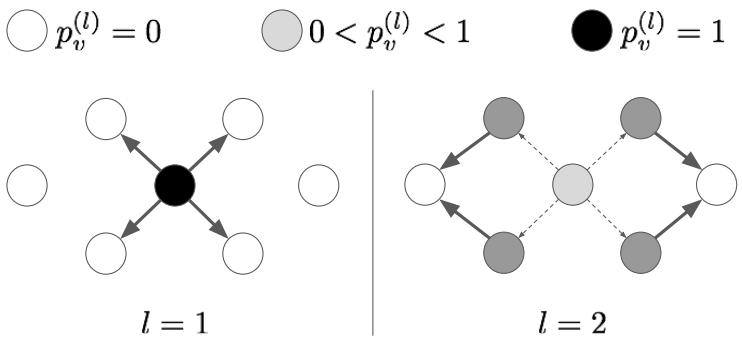


Figure 4.6: Directed propagation of embeddings in GRAFT-Net. A scalar *pagerank* score $p_v^{(l)}$ is maintained for each node v across layers, which distributes out from the seed node. Embeddings are only propagated from nodes with pagerank > 0.

4.5.6 *Regularization via Fact Dropout*

To encourage the model to learn a robust classifier, which exploits all available sources of information, we randomly drop edges from the graph during training with p_0. We call this *fact-dropout*. It is usually much easier to learn to reason over the KB than over textual documents, so the model tends to overly rely on the KB, especially when the KB is complete. This method shares similarities with DropConnect (Wan, Zeiler, et al., 2013).

4.6 EXPERIMENTS ON GRAFT-NETS

4.6.1 *Datasets*

WIKIMOVIES-10K It consists of 10K randomly sampled training questions from the WikiMovies dataset (A. H. Miller et al., 2016), along with the original test and validation sets. We sample the training questions to create a more difficult setting, since the original dataset has 100K questions over only 8 different relation types, which is unrealistic in our opinion. In Section 4.6.4 we also compare to existing state-of-the-art using the full training set.

We use the KB and text corpus (constructed from Wikipedia) released by A. H. Miller et al. (2016). For entity linking we use simple surface level matches, and retrieve the top 50 entities around these to create the question subgraph. We further add the top 50 sentences (along with article titles) to add to the subgraph using Lucene search over the text corpus. The overall answer recall in our constructed subgraphs is 99.6%.

WEBQUESTIONSSP (Yih, M. Richardson, et al., 2016) It consists of 4737 natural language questions posed over Freebase entities, split up into 3098 training and 1639 test questions. We

Dataset	# train / dev / test	# entity nodes	# edges	# document nodes	# question vocab
WikiMovies-10K	10K / 10K / 10K	43,233	9	79,728	1759
WebQuestionsSP	2848 / 250 / 1639	528,617	513	235,567	3781

Table 4.4: Statistics of all the retrieved subgraphs $\cup_q \mathcal{G}_q$ for WikiMovies-10K and WebQuestionsSP.

Model	Text Only	KB + Text			
		10 %	30%	50%	100%
WikiMovies-10K					
KV-KB	–	15.8 / 9.8	44.7 / 30.4	63.8 / 46.4	94.3 / 76.1
KV-EF	50.4 / 40.9	53.6 / 44.0	60.6 / 48.1	75.3 / 59.1	93.8 / 81.4
GN-KB	–	19.7 / 17.3	48.4 / 37.1	67.7 / 58.1	**97.0 / 97.6**
GN-LF	**73.2 / 64.0**	74.5 / 65.4	78.7 / 68.5	83.3 / 74.2	96.5 / 92.0
GN-EF		75.4 / 66.3	82.6 / 71.3	87.6 / 76.2	96.9 / 94.1
GN-EF+LF		**79.0 / 66.7**	**84.6 / 74.2**	**88.4 / 78.6**	**96.8 / 97.3**
WebQuestionsSP					
KV-KB	–	12.5 / 4.3	25.8 / 13.8	33.3 / 21.3	46.7 / 38.6
KV-EF	23.2 / 13.0	24.6 / 14.4	27.0 / 17.7	32.5 / 23.6	40.5 / 30.9
GN-KB	–	15.5 / 6.5	34.9 / 20.4	47.7 / 34.3	66.7 / 62.4
GN-LF	**25.3 / 15.3**	29.8 / 17.0	39.1 / 25.9	46.2 / 35.6	65.4 / 56.8
GN-EF		31.5 / 17.7	40.7 / 25.2	49.9 / 34.7	67.8 / 60.4
GN-EF+LF		**33.3 / 19.3**	**42.5 / 26.7**	**52.3 / 37.4**	**68.7 / 62.3**

Table 4.5: Hits@1 / F1 scores of GRAFT-Nets (GN) compared to KV-MemNN (KV) in KB only (-KB), early fusion (-EF), and late fusion (-LF) settings.

reserve 250 training questions for model development and early stopping. We use the entity linking outputs from the S-MART system[3] and retrieve 500 entities from the neighbourhood around them in Freebase to populate the question subgraphs.[4] We further retrieve the top 50 sentences from Wikipedia with the two-stage process described in Section 4.5.1. The overall recall of answers among the subgraphs is 94.0%.

Table 4.4 shows the combined statistics of all the retreived subgraphs for the questions in each dataset. These two datasets present varying levels of difficulty. While all questions in WikiMovies correspond to a single KB relation, for WebQuestionsSP the model needs to aggregate over two KB facts for ~30% of the questions, and also requires reasoning over constraints for ~7% of the questions (C. Liang et al., 2017). For maximum portability, QA systems need to be robust across several degrees of KB availability since different domains might contain different amounts of structured data; and as KBs vary across time, their completeness also varies. Hence, we construct an additional 3 datasets each from the above two, with the number of KB facts downsampled to 10%, 30% and 50% of the original to simulate settings where the KB is incomplete. We repeat the retrieval process for each sampled KB.

4.6.2 *Compared Models*

KV-KB is the Key Value Memory Networks model from A. H. Miller et al. (2016) but using only KB and ignoring the text. **KV-EF** is the simple model from Section 4.3 with access to both KB and text as memories. For text we use a BiLSTM over the entire sentence as keys, and entity mentions as values. This re-implementation shows better performance on the text-only

3 https://github.com/scottyih/STAGG

4 A total of 13 questions had no detected entities. These were ignored during training and considered as incorrect during evaluation.

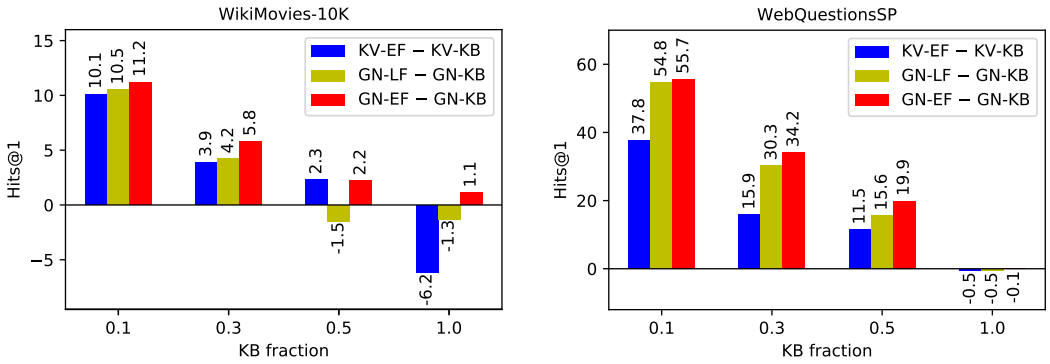


Figure 4.7: Improvement of early fusion (-EF) and late fusion (-LF) over KB only (-KB) settings.

and KB-only WikiMovies tasks than the results reported previously[5] (see Table 4.6). **GN-KB** is the GRAFT-Net model ignoring the text. **GN-LF** is a late fusion version of the GRAFT-Net model: we train two separate models, one using text only and the other using KB only, and then ensemble the two[6]. **GN-EF** is our main GRAFT-Net model with early fusion. **GN-EF+LF** is an ensemble over the GN-EF and GN-LF models, with a similar ensembling to GN-LF alone. We report Hits@1, which is the accuracy of the top-predicted answer from the model, and the F1 score. To compute the F1 score we tune a threshold on the development set, to select answers based on binary probabilities for each node in the subgraph.

4.6.3 *Main Results*

Table 4.5 presents a comparison of the above models across all datasets. GRAFT-Nets (GN) show consistent improvement over KV-Memory Nets on both datasets in all settings, including KB only (-KB), text only (-EF, Text Only column), and early fusion (-EF). Interestingly, we observe a larger relative gap between the Hits and F1 scores for the KV models than we do for our GN models. We believe this is because the attention for KV is normalized over the memories, which are KB facts (or text sentences): hence the model is unable to assign high probabilities to multiple facts at the same time. On the other hand, in GN, we normalize the attention over *types of relations* outgoing from a node, and hence can assign high weights to all the correct answers.

We also see a consistent improvement of early fusion compared to late fusion (-LF), and by ensembling them together we see the best performance across all the models. In Figure 4.7, we further show the improvement for KV-EF over KV-KB, and, GN-LF and GN-EF over GN-KB, as the amount of KB is increased, to measure how effective these approaches are in utilizing text plus a KB. For KV-EF we see improvements when the KB is highly incomplete, but in the full KB setting, the performance of the fused approach is worse. A similar trend holds for GN-LF. On the other hand, GN-EF effectively utilizes text over the KB-only approach across all the settings. As we would expect, though, the benefit of adding text decreases as the KB becomes more and more complete.

5 For all KV models we tuned the number of layers $\{1, 2, 3\}$, batch size $\{10, 30, 50\}$, model dimension $\{50, 80\}$. We also use fact dropout regularization in the KB+Text setting tuned between $\{0, 0.2, 0.4\}$.

6 For ensembles we take a weighted combination of the answer probabilities produced by the models, with the weights tuned on the dev set. For answers only in text or only in KB, we use the probability as is.

Method	WikiMovies (full)		WebQuestionsSP	
	kb	doc	kb	doc
MINERVA	**97.0 / –**	–	–	–
R2-AsV	–	85.8 / –	–	–
NSM	–	–	**– / 69.0**	–
DrQA*	–	–	–	21.5 / –
R-GCN#	96.5 / 97.4	–	37.2 / 30.5	–
KV	93.9 / –	76.2 / –	– / –	– / –
KV#	95.6 / 88.0	80.3 / 72.1	46.7 / 38.6	23.2 / 13.0
GN	**96.8 / 97.2**	**86.6 / 80.8**	67.8 / 62.8	**25.3 / 15.3**

Table 4.6: Hits@1 / F1 scores compared to SoA models using only KB or text: MINERVA Das, Dhuliawala, et al., 2018, R2-AsV Watanabe, Dhingra, and R. Salakhutdinov, 2017, Neural Symbolic Machines (NSM) C. Liang et al., 2017, DrQA D. Chen et al., 2017, R-GCN Schlichtkrull et al., 2017 and KV-MemNN. *DrQA is pretrained on SQuAD. #We re-implement KV-MemNN and R-GCN.

4.6.4 *Comparison to Specialized Methods*

In Table 4.6 we compare GRAFT-Nets to state-of-the-art models that are specifically designed and tuned for QA using either only KB or only text. For this experiment we use the full WikiMovies dataset to enable direct comparison to previously reported numbers. For DrQA (D. Chen et al., 2017), following the original paper, we restrict answer spans for WebQuestionsSP to match an entity in Freebase. In each case, for a fair comparison, we also train GRAFT-Nets using only KB facts or only text sentences. In three out of the four cases, we find that GRAFT-Nets either match or outperform the existing state-of-the-art models. We emphasize that the latter have no mechanism for dealing with the fused setting.

The one exception is the KB-only case for WebQuestionsSP where GRAFT-Net does 6.2% F1 points worse than Neural Symbolic Machines (C. Liang et al., 2017). On analyzing we found that this is because: (1) In the KB-only setting, the recall of subgraph retrieval is only 90.2%, which limits overall performance. In an oracle setting where we ensure the answers are part of the subgraph, the F1 score increases by 4.8%. (2) We use the same probability threshold for all questions, even though the number of answers may vary significantly. Models which parse the query into a symbolic form do not suffer from this problem since answers are retrieved in a deterministic fashion. If we tune separate thresholds for each question the F1 score improves by 7.6%. (3) GRAFT-Nets perform poorly in the few cases where there is a constraint involved in picking out the answer (for example, "who *first* voiced Meg in Family Guy"). If we ignore such constraints, and consider all entities with the same sequence of relations to the seed as correct, the performance improves to 66.6% F1. Heuristics such as those used by M. Yu et al. (2017) can be used to improve these cases. Figure 4.7 shows examples where GRAFT-Net fails to predict the correct answer set exactly.

4.6.5 *Effect of Model Components*

HETEROGENEOUS UPDATES. We test a non-heterogeneous version of our model where instead of using fine-grained entity linking information for updating the node representations ($M(v)$ and $L(d, p)$ in (4.2) and (4.4a)), we aggregate the document states across all its positions

Question	Correct Answers	Predicted Answers
what language do most people speak in afghanistan	Pashto language, Farsi (Eastern Language)	Pashto language
what college did john stockton go to	Gonzaga University	Gonzaga University, Gonzaga Preparatory School

Table 4.7: Examples from WebQuestionsSP dataset. **Top:** The model misses a correct answer. **Bottom:** The model predicts an extra incorrect answer.

as $\sum_p H^{(l)}_{d,p}$ and use this combined state for all updates. Without the heterogeneous update, all entities $v \in L(d, \cdot)$ will receive the same update from document d. Therefore, the model cannot disambiguate different entities mentioned in the same document. The result in Table 4.8 shows that this version is consistently worse than the heterogeneous model.

	0 KB	0.1 KB	0.3 KB	0.5 KB	1.0 KB
NH	22.7 / 13.6	28.7 / 15.8	35.6 / 23.2	47.2 / 33.3	66.5 / 59.8
H	25.3 / 15.3	31.5 / 17.7	40.7 / 25.2	49.9 / 34.7	67.8 / 60.4

Table 4.8: Non-Heterogeneous (NH) vs. Heterogeneous (H) updates on WebQuestionsSP

CONDITIONING ON QUESTION. We also performed an ablation test on the directed propagation method and attention over relations. We observe that both components lead to better performance. Such effects are observed in both complete and incomplete KB scenarios, *e.g.* on WebQuestionsSP dataset, as shown in Figure 4.8.

FACT DROPOUT. Figure 4.8 (Right) compares the performance of the early fusion model as we vary the rate of fact-dropout. Moderate levels of fact-dropout improve performance on both datasets. The performance increases as the fact-dropout rate increases until the model is unable to learn the inference chain from KB.

4.7 POINTS TO PONDER

In this work, we showed universal schema is a promising knowledge source for QA than using KB or text alone. Our results conclude though KB is preferred over text when the KB contains the fact of interest, a large portion of queries still attend to text indicating the amalgam of both text and KB is superior than KB alone.

In this paper we look at QA using text combined with an incomplete KB, a task which has received limited attention in the past. We introduce several benchmark problems for this task by modifying existing question-answering datasets, and discuss two broad approaches to solving this problem—"late fusion" and "early fusion". We show that early fusion approaches perform better.

We also introduce a novel early-fusion model, called GRAFT-Net, for classifying nodes in subgraph consisting of both KB entities and text documents. GRAFT-Net builds on recent advances in graph representation but includes several innovations which improve performance on this task. GRAFT-Nets are a single model which achieve performance competitive to state-of-the-

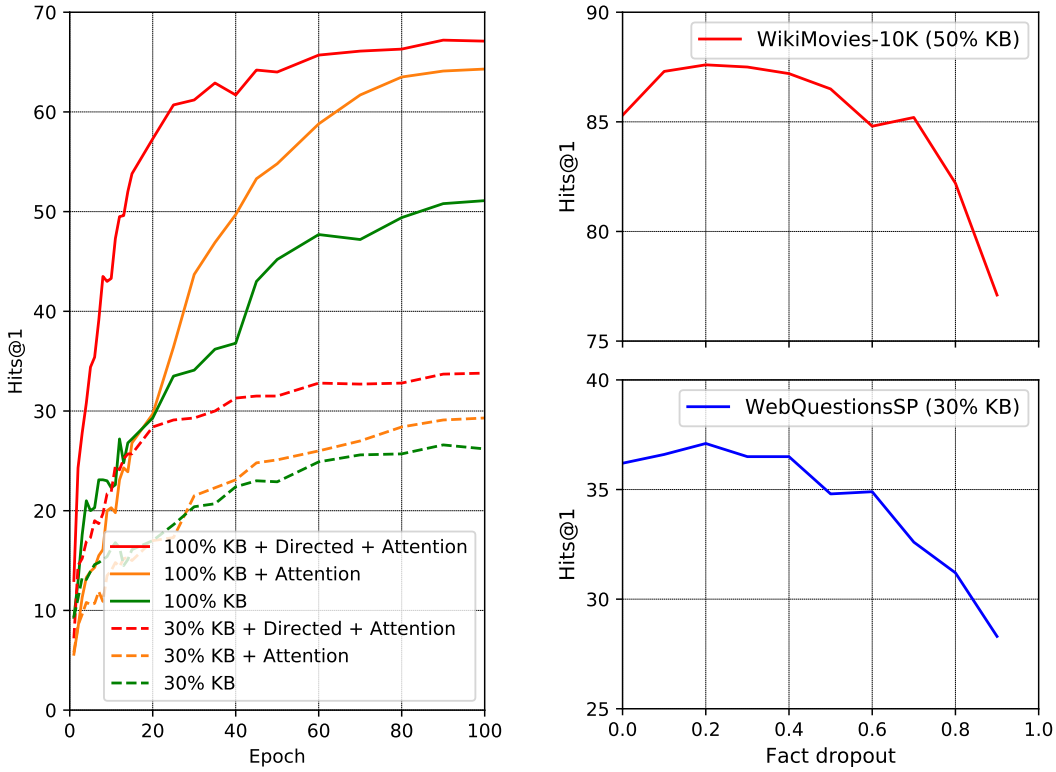


Figure 4.8: **Left:** Effect of directed propagation and query-based attention over relations for the WebQuestionsSP dataset with 30% KB and 100% KB. **Right:** Hits@1 with different rates of fact-dropout on and WikiMovies and WebQuestionsSP.

art methods in either a text-only setting or a KB-only setting, and outperform baseline models when using text combined with an incomplete KB. Current limitations of the model include – (1) GRAFT-Nets are currently restricted to picking entities as answers, whereas in general open-domain questions are better answered by picking spans of text; and (2) the subgraph retrieval process currently limits the maximum possible performance of the model, especially in the text only case. These are promising directions for future work.

Part II

INTERPRETABLITY BY INCORPORATING DOMAIN KNOWLEDGE

Internal representations in most deep learning methods are distributed codes with a large number of parameters and lack interpretability. These dense unstructured representations are in direct conflict with many real-world problems which have inherent structures: like invariances, dependencies among random variables, etc. We will utilize methods developed in Part I to handle structures like invariances along with leveraging structure in form of domain knowledge by using Bayesian components on top of deep models. Such modifications will yield interpretable and sample efficient model. We look at diverse applications ranging from static NLP and CV tasks to to even less relevant fields like basic sciences to temporal user action modeling.

5 Static Models

We study the problem of incorporating domain knowledge in completely observed data (as compared to sequentially observed data, which we will look at in next chapter) to enhance predictive power and interepretability. Deep learning methods have been shown to be extremely successful in a variety of well defined problems due their high expressive power across computer vision (CV), speech recognition as well as natural language processing (NLP). However, their internal representations are distributed codes and lack interpretability or manipulations. These dense unstructured representations are in direct conflict with many real-world problems which have inherent structures: like invariances, dependencies among random variables, etc. Graphical models have been typically employed to represent and exploit such invariances and dependencies. In this chapter, we will explore ways to incorporate domain knowledge so as to take advantage of high predictive power of deep learning but also be more interpretable and sample efficient by enforcing invariances and dependencies. We will also propose efficient inference strategies to tackle new difficulties encountering in these combined models. Here we look at diverse applications ranging from NLP and CV tasks to to even less relevant fields like physics or chemistry.

5.1 Introduction

In this section, we look at methods to incorporate domain knowledge into predictive tasks like classification and regression and exploratory tasks like clustering, organization, and search. In particular for predictive data analyis, we look at applications from basic science: molecular properties estimation in quantum chemistry and red-shift estimation in cosmology. In these applications, we design neural networks leveraging structure from the domain knowledge and show reduction in preprocessing cost while improving the accuracy and interpretability. For exploratory data analysis, we look at organizing satellite imagery and improving document topic modeling with deep word embeddings using external knowledge.

Molecular properties Graph convolutions have recently become popular for learning molecular properties, as they move beyond commonly employed molecular fingerprints and excel at encoding representation of the molecule as a part of the learning process itself. These models, however, require extensive featurization of the input data, which often relies on support from existing chemical fingerprinting schemes and phenomenological classifications like chirality, hybridization, aromaticity, bond ordering and partial atomic charges. In this work, we make use of a novel idea of learning molecular representation to predict DFT computed properties using just the set of 3D coordinates of atoms in the molecules. Based on the condition that any regression model used for predicting quantum mechanical observables of the electronic density should be invariant to the permutation of atoms in the molecular set, we utilize a recently proposed theorem on *DeepSets* that characterizes such permutation invariant functions and provides a convenient architecture to map such functions. The proposed method is shown to be

competitive to graph convolutions and generality of the learnt molecular representations is demonstrated without loss in prediction accuracy.

COSMOLOGICAL PARAMETERS An important regression problem in cosmology is to estimate the red-shift of galaxies, corresponding to their age as well as their distance from us (Binney and Merrifield, 1998) based on photometric observations. One way to estimate the red-shift from photometric observations is using a regression model (Connolly et al., 1995) on the galaxy clusters. The prediction for each galaxy does not change by permuting the members of the galaxy cluster. Therefore, we can treat each galaxy cluster as a "set". There is an opportunity to and use DeepSets to estimate the individual galaxy red-shifts.

SATELLITE IMAGERY Creative inquiries through similarity search have historically offered novel perspectives in many domains (Broder, 1997; Kiefer and Laub, 2013; Ngai et al., 2011; Shin and Y. H. Kim, 2014; Saransh Singh, F. Ram, and De Graef, 2017). A real-time online nearest neighbor (NN) search allows artists to expressively use the power of computation to explore vast datasets. We develop, Terrapattern, a prototype for visual query-by-example in satellite imagery. Terrapattern is an interface for finding "more like this, please" images in satellite photos: when a user selects an interesting tile on Terrapattern's map, the system finds other locations that look similar. This tool has been used by artists, citizen scientists, and hobbyists alike to spot and explore emerging trends, and has been well received by the media and public at large. To facilitate a fast online NN search, we develop a tree data structure–Stable Greedy Tree(SG-Tree)–which allows for almost linear time construction and logarithmic query time under real world assumptions. SG-Tree is amenable to parallelization and distribution which greatly enhances its appeal over existing approaches such as NNet trees and Cover Trees and is essential for the scale required on large datasets. Moreover, the data structure adapts to the intrinsic dimension of the data, which is particularly beneficial when the feature vectors come from neural networks that have high euclidean dimensions but often lie on a (significantly) smaller dimensional manifold. The proposed method significantly outperforms existing NN search tools in construction time on many real world data-sets and is well suited for interactive queries.

DOCUMENT MODELLING Continuous space word embeddings learned from large, unstructured corpora have been shown to be effective at capturing semantic regularities in language. In this paper we replace LDA's parameterization of "topics" as categorical distributions over opaque word types with multivariate Gaussian distributions on the embedding space. This encourages the model to group words that are *a priori* known to be semantically related into topics. To perform inference, we introduce a fast collapsed Gibbs sampling algorithm based on Cholesky decompositions of covariance matrices of the posterior predictive distributions. We further derive a scalable algorithm that draws samples from stale posterior predictive distributions and corrects them with a Metropolis–Hastings step. Using vectors learned from a domain-general corpus (English Wikipedia), we report results on two document collections (20-newsgroups and NIPS). Qualitatively, Gaussian LDA infers different (but still very sensible) topics relative to standard LDA. Quantitatively, our technique outperforms existing models at dealing with OOV words in held-out documents.

5.2 PREDICTIVE DATA ANALYSIS: MOLECULAR PROPERTIES

5.2.1 *Learning Density Functional from Molecular Geometry*

Machine learning has recently made significant advances in prediction of properties and descriptors for material discovery and building *quantitative structure-activity relationships* (QSAR) for material design. This approach relies on learning the fundamental relationship between the electronic structure of a material and its thermal, chemical and physical properties. In its most general usage, the structural or chemical features of a molecule X are often extracted using molecular fingerprinting software (Φ) to compute fixed-dimensional representation vectors $R = \Phi(X)$. Any desired property P is then predicted using those features as inputs to a regression model(ρ) $P = \rho(R)$ by means of generalization over several examples.

Encoding the representation of molecules as fingerprints is a difficult task because of their diverse chemical compositions, arbitrary sizes and shapes. Standard topological or circular fingerprints record the occurrence of certain known functional groups in a given molecule within a certain distance or along a certain connected path (Cereto-Massagué et al., 2015). The representations are built by applying a fixed hash function to concatenate the features of the neighborhood of an atom, wherein the information about the 3D geometry and electronic structure of the molecule is lost, while a lot of representation bits are used up for rarely occurring features. Besides, the same style of fingerprint is not necessarily optimum for training and predicting different classes of properties.

A recent strategy to move beyond these limitations is to learn the representation of the molecule as a part of the regression model itself. For instance, in Coley et al. (2017), Duvenaud et al. (2015), and Kearnes et al. (2016) to name a few, the molecular fingerprint vectors are obtained as differentiable convolutional graphs, in which node and edge features of the graph are analogous to atoms and bonds in the molecule. The features of the convolutional graph are iteratively updated based on those calculated at the previous depth. The resulting features from the convolutional neighborhood of each atom are then pooled together to yield a representation vector, which is finally passed through additional hidden layers in a feed forward network to regress on a desired property. This approach is a significant improvement over conventional off-the-shelf fingerprints, as has been demonstrated conclusively by Faber, Hutchison, et al. (2017), in which they compared the performance of several combinations of representation schemes and regression models for predicting density functional theory (DFT) calculated properties.

These models, however, are heavily dependent on the featurization of the input representations. Kearnes et al. (2016) used atom and bond descriptors such as *formal charge*, *Crippen contribution to logP and molar refractivity*, *total polar surface area contribution*, *Labute approximate surface area contribution*, *estate index*, *Gasteiger partial charges*, *aromaticity*, *bond order* and *conjugation* for featurizing their initial input representation for prediction of properties such as *solubility*, *drug efficacy* and *photovoltaic efficiency*. In the same spirit, (Coley et al., 2017) used *bond length*, *chirality*, *ring size*, *hybridization* and *hydrogen bond donor/acceptor* as features to their input, in addition to several from those mentioned above, for prediction of *aqueous solubility*, *octanol solubility*, *melting point*, and *toxicity*. Several of these attributes were calculated using beforehand or had to be estimated from wet lab. Moreover, the graph itself used for graph convolution in all these methods is a featurization of the input data. This is because the each bond in the graph has to be also obtained painstakingly, i.e. it is not inherently known from geometry which atoms have a bond between them.

Featurization of input data, however, is not necessarily generalizable for all molecules even if their 3D geometry is known. Many of the attributes mentioned above are prone to uncertainties in their estimation and are often not readily available for molecules of interest (McDonagh et al., 2014). Besides, they rely on support from external chemical fingerprinting software and phenomenological classification schemes derived from the electronic structure, which are not trivial to determine consistently. Although quantum mechanics lets us compute the electronic structure of materials from first principles using the Schrödinger equation, it is far too difficult to solve exactly for many-electron systems. Density functional theory (DFT) is the most widely used many-body approach for such calculations but it scales poorly (as $O(N^3)$, where N is the number of electrons) when applied to large systems.

In this work, we develop an efficient deep learning approach that enables prediction of quantum-mechanical observables of many-body electronic systems using only the molecular geometry in its crudest form, i.e. just the set of 3D coordinates of atoms and their types in the molecules. The reason for existence of such a mapping between the molecular geometry and quantum-mechanical observables can be argued from the first of the two Hohenberg-Kohn theorems (Koch and Holthausen, 2001), which are the fundamental basis for DFT. Under the Born-Oppenheimer approximation of fixed atomic nuclei in a many-body electron system, the first Hohenberg-Kohn theorem describes the relation between the electronic charge density and the total external potential $v_{ext}(\mathbf{r})$. Stated simply, the ground state energy of a system of electrons $E[n(\mathbf{r})]$ is a unique functional of the ground state density $n(\mathbf{r})$ as

$$E[n(\mathbf{r})] = \int n(\mathbf{r}) v_{ext}(\mathbf{r}) d\mathbf{r} + F[n(\mathbf{r})] \tag{5.1}$$

where $F[n(\mathbf{r})]$ is an unknown, but otherwise universal *functional* of the electron density only. In fact all properties of the system, including excited state properties, are, in principle, exact functionals of the ground state density. In absence of additional electromagnetic fields, the external potential for electrons is decided uniquely by the three spatial coordinates of atoms and their types in any given molecule. The implication of this is that all the quantum mechanical observables of the electronic structure of a molecule will therefore depend only on its 3D geometry. In this work, we present a method to learn this mapping.

The representation of a molecule as a set of 3D coordinates is not expected to possess any inherent ordering, and the dimension of such sets can vary from one molecule to other, thus posing challenges in applying common machine learning tools that require ordered and fixed dimensional inputs. In addition to this, the function that maps molecular geometry to properties that are quantum-mechanical observables of the system should, in principle, possesses permutation invariance with respect to the order in which spatial information of atoms is being provided. In this regard, we employ a recently proposed theorem (Zaheer, Kottur, Ravanbakhsh, et al., 2017) that characterizes such permutation invariant functions on sets and provides a family of functions that can express any such relation. This family of functions has an iteratively decoupled structure which enables the design of a deep neural network that can operate directly on molecular geometries, while simultaneously learning their representations and predicting their properties.

The proposed method is demonstrated to be competitive to graph convolutions for several properties that are directly dependent on the electronic structure, without any need for feature engineering. We demonstrate robust generalization of the learnt representations for predicting similar properties simultaneously, without the need for extending the network or loss in prediction accuracy Additionally, we show handling output data of different dimensions from the *representation* half of the network to predict Mulliken charges, which which, to the best of our

knowledge, has not yet been reported. Finally, we discuss the systematic origin of prediction errors depending in the number of atoms in the molecules.

5.2.2 *Model*

Formally, our problem can be described as follows: Given N molecular geometries of $X^{(1)}, ..., X^{(N)}$, where each geometry can be represented as a set of co-ordinates of M atoms in the molecule, i.e. $X^{(i)} = \{x_1^{(i)}, ..., x_M^{(i)}\}$. The learning tasks can be classified into two categories depending on the target: (a) to regress (with variable M per molecule) on target molecular property values $Y^{(1)}, ..., Y^{(N)}$ while obeying invariance w.r.t. specification of molecular geometry, and (b) to regress on a property $y_m^{(n)}$ corresponding to every atom m with coordinates $x_m^{(n)}$ in any molecule n, like partial charge.

For a function $f : \mathfrak{X}_M^{(i)} \rightarrow y_M^{(i)}$ that takes only 3D coordinates as input, an arbitrary choice of input data expressed in specified units (Å in this case) must be enabled by the following:

1. Translation invariance: The output of the model should be invariant to a constant shift in the coordinates of the molecule $X = \{x_1, ..., x_M\}$, *i.e.* for any translation τ:

$$f(\{x_j + \tau\}) = f(\{x_j\}) \tag{5.2}$$

2. Rotation invariance: The output of the model should be invariant to all possible rotations of the molecule X, *i.e.* for any rotation $\mathcal{R}$

$$f(\mathcal{R}X) = f(X) \tag{5.3}$$

3a. Permutation invariance: The predictions of the model for molecular target properties Y corresponding to a molecule X should always be invariant to the order in which the set of atoms belonging to any molecule are fed into the network, *i.e.* for any permutation π of the input atoms:

$$f([x_{\pi(1)}, \ldots, x_{\pi(M)}]) = Y \tag{5.4}$$

3b. Permutation equivariance: In order to regress on a property y_m corresponding to every atom m with coordinates x_m in any molecule X, the function should posses an *equivariance* such that the application of the function on a different order of the input atoms transforms the output labels correspondingly, *i.e.* for any permutation π of the input atoms:

$$f([x_{\pi(1)}, \ldots, x_{\pi(M)}]) = [y_{\pi(1)}, \ldots, y_{\pi(M)}] \tag{5.5}$$

Translation invariance is easily handled by always centering the data. However, note that rotational invariance is not inherent to the network architecture but is enforced by training over several transformations of the same dataset in a self-consistent manner, until there is no change in the training set errors upon performing rotation.

5.2.3 *Experiment*

5.2.3.1 *Dataset*

The publicly available QM9 dataset (Ramakrishnan et al., 2017) was chosen as a suitable reference as it has been used for such prediction tasks previously. It contains B3LYP/6-31G(2df,p)

level computed geometric, energetic, electronic, and thermodynamic properties for 133,885 stable small organic molecules composed from the elements H, C, N, O and F.

The dataset contains properties that relate to the frontier orbital energy states of the molecule such as the highest occupied molecular orbital eigenvalue ε_{HOMO} (eV), the lowest molecular orbital eigenvalue ε_{LUMO} (eV) and the HOMO-LUMO gap $\Delta\varepsilon$ (eV). In addition to these, some target types characterize the spatial distribution of electronic charge in the molecule in terms of quantities like the norm of dipole moment μ (Debye), static isotropic polarizability α (Bohr^3), and electronic spatial extent $\langle R^2 \rangle$ (Bohr^2). The zero point vibrational energy ZPVE (eV), which relates to the fundamental vibrations of the molecule at 0 K, is also included. Lastly, the heat capacity at room temperature C_v (cal/molK) is also included as a target type. We predict the Mulliken partial charge for any given atom in a molecule ‹q (e^-). In total, 9 types of data were used as targets for training and prediction. Only 2 molecules of size 3 and 4 molecules of size 4 were excluded from the study as their atoms could not be assigned at least 4 nearest neighbors (Section 5.2.3.2).

5.2.3.2 *Model training and evaluation*

Only the DFT computed 3D geometry and atom types were provided as input data. In order to accelerate the learning of topological features such as bonds, angles and dihedrals, the 3D coordinates and atom types of the 4 next nearest neighbors were concatenated to the input data of any given atom in the molecule, without any explicit specification of the values of bond distances or angles. Inclusion of the 5th and 6th next nearest neighbors did not yield any significant improvement in prediction accuracy. In order to utilize the equivariance inherent to the *representation* half of the network architecture, the learning process was implemented in a *curriculum* based on the dimension of the molecule, in which, only molecules of the same dimension, i.e., same number of atoms, were passed together in batches of size 100. The ADAM optimizer was chosen with an initial learning rate of 1e-03, which was reduced by a factor of 0.9 after every 50 epochs, except when the input 3D coordinates of all molecules in the training set were rotated randomly after every 200 epochs in order to enforce rotational invariance until prediction errors on the training set are self-consistently converged.

In the same spirit as previous machine learning efforts on the QM9 dataset by Faber, Hutchison, et al. (2017) the entire dataset set was divided into a 9:1 ratio for training and testing, respectively. Several randomly generated instances of training/test sets were created and used in order to ensure reproducibility of errors on the unknown test set.

5.2.3.3 *Performance*

In Table 5.1, we compile the mean absolute errors (MAE) of our prediction model on the test set and compare it with the best case scenarios of several combinations of *regressors* and *representations* on the same dataset taken from the work of Faber, Hutchison, et al. (2017) for all the targets at 0 K (except C_v at 298.15 K). In addition, we list the mean, mean absolute deviation (MAD)[1] and DFT chemical accuracy estimates for each of the targets (Ramakrishnan et al., 2017). Faber, Hutchison, et al. (2017) tested a host of regression models such as Elastic Net (EN), Bayesian Ridge Regression (BR), Random Forest (RF), Kernel Ridge Regression (KRR), Graph Convolutions (GC) and Gated Graph Neural Networks (GG). From Table 5.1, it can be observed that irrespective of the representation format, graph convolutions outperform other regression methods with a general ordering of GC ~ GG > KRR > RF ~ BR ~ EN.

1 *Mean and MAD values of the targets shown here are slightly different from the work of (Faber, Hutchison, et al., 2017) as they exclude a larger set of molecules from the total set for which the SMILES consistency tests failed.

These regression models were tested on many schemes of molecular representations, amongst which, *BAML* (Bonds, Angles, Machine Learning) and *HDAD* (Histogram of distances, Angles and Dihedral angles) are worth mentioning as they consistently resulted in comparably low prediction errors (Faber, Hutchison, et al., 2017). The BAML representation format requires Morse and Lennard-Jones potentials for bonded and non-bonded atoms, respectively. Three and four-body interactions between covalently bonded atoms are included using angular and torsional terms, respectively. These parameters and their functional forms are based on the universal force field (UFF) (Rappe et al., 1992), generating which requires very controlled and systematic series of quantum chemical calculations that are difficult to execute for any unknown system. HDAD, on the other hand, is a significant improvement in generalization over BAML because it accounts for pairwise distances, triple-wise angles, and quad-wise dihedral angles through manually generated bins in histograms. The collection of features obtained from the analysis of histograms is subsequently rendered into fixed-size representations. With regards to featurizing the input, HDAD is even more general than featurized convolutional graphs, as it depends only on the knowledge of the 3D geometry of the molecule.

Overall, our model produces errors which were within or quite close to the DFT level errors in estimation of targets at 0 K. Targets that characterize the spatial distribution of electronic charge in the molecule, such as dipole moment (μ), static isotropic polarizability (α) and electronic spatial extent ($\langle R^2 \rangle$) are predicted with accuracy of the same order as graph convolutions by the present model. $\langle R^2 \rangle$ measures the extent to which a molecule can spontaneously incur a dipole moment in response to an external field and is relatively difficult to predict because in this case van der Waals interactions play a role in the dynamics of the medium. It must be noted that DFT in its standard flavor, as applied in the QM9 dataset, is known to fail patently when accounting for van der Waals interactions.

Prediction accuracies for frontier orbital ($\varepsilon_{\text{HOMO}}$-$\varepsilon_{\text{LUMO}}$) energy levels is achieved well within the DFT error limits by the current model, which performs better than or equal to all baselines except graph convolutions. DFT uncertainties in estimation of $\varepsilon_{\text{HOMO}}$-$\varepsilon_{\text{LUMO}}$ is generally quite high owing to the dependence on basis sets/functionals used for performing computations, and accurate estimation needs higher order perturbation and wavefunction based methods. It is not surprising, therefore, that convolutional graphs featurized with quantum mechanical meta-information, such as hybridization of orbitals and aromaticity, perform better in predicting these values.

The zero-point vibrational energy (ZPVE) at 0 K is calculated from bond force constants under a harmonic oscillator approximation. It is predicted equally well by the current model in comparison to graph convolutions, and is equivalent to DFT error limits. The heat capacity is predicted poorly by the current model as compared to the standard methods, which are better even than graph convolutions. A reason for this could be the fact that at 0 K, the contributions of translation, rotation, and vibration degrees of freedom to the heat capacity of gas phase molecules vanish and the current model fails to encode this information. Heat capacities in the present data set have not been estimated as a direct quantum mechanical observable of the electronic structure of the molecule, but rather from principles of thermochemistry, wherein the contributions of the translational, rotational and vibrational degrees of freedom are only obtained as corrections under the rigid rotor approximation.

5.2.3.4 *Prediction of Mulliken charges*

Mulliken charges are not integer charges, rather fractions of an electron corresponding to the percentage of time an electron is near each nucleus. They are best obtained with smaller, generalized basis sets using linear combination of molecular orbitals, and are routinely used as

Regressor (Fingerprint)	μ Debye	α Bohr^3	$\langle R^2 \rangle$ Bohr^2	ε_{HOMO} eV	ε_{LUMO} eV	$\Delta\varepsilon$ eV	ZPVE eV	C_v cal/molK
EN(HDAD)	0.563	0.437	6.19	0.139	0.238	0.278	0.0064	0.088
BR(HDAD)	0.565	0.430	5.94	0.140	0.238	0.278	0.0032	0.079
RF(BAML)	0.434	0.638	51.1	0.107	0.118	0.141	0.0132	0.451
KRR(HDAD)	0.334	0.175	1.62	0.066	0.084	0.107	0.0019	0.044
GG(MG)	0.247	0.161	6.30	0.057	0.063	0.088	0.0043	0.084
GC(MG)	0.101	0.232	4.71	0.055	0.062	0.087	0.0097	0.097
This work	0.283	0.472	6.45	0.135	0.155	0.183	0.0096	0.200
DFT errors	0.1	0.4	-	2.0	2.6	1.2	0.0097	0.34
Mean	2.706	75.19	1189.53	-6.530	0.303	6.833	4.042	31.600
MAD	1.189	6.300	202.04	0.443	1.052	1.079	0.720	3.204

Table 5.1: MAE on unknown test data from the current model and its comparison with predictions from combinations of regressors and representations taken from work of Faber, Hutchison, et al. (2017) for all DFT calculated properties from the QM9 dataset.

variables in linear regression (QSAR) procedures. For any given molecule, every atom is assigned a Mulliken charge using population analysis. From a machine learning perspective, this is a difficult task to implement and has not been performed yet in this area, as molecules of different dimensions will have to be trained against outputs of different dimensions. However, the inherent equivariance in the current model as well as the curriculum based learning implementation scheme allows for the handle outputs of the same dimension as the molecule, which is the case for Mulliken charges.

Without needing to extend the representation half of the network, Mulliken charges were predicted for the unknown test with an error of $0.013e^-$, where e^- represents an electron's charge. The error cannot be compared to a DFT benchmark as it is obtained entirely from mathematical constructions, rather than as a fundamental quantum mechanical observable. Its usefulness and accuracy can be gauged from the fact that classical molecular dynamics simulations employ such values as point charges specified up to $\sim 0.01e^-$.

5.3 PREDICTIVE DATA ANALYSIS: COSMOLOGICAL PARAMETERS

5.3.1 *Red Shift Estimation in Galaxy Clusters*

An important regression problem in cosmology is to estimate the red-shift of galaxies, corresponding to their age as well as their distance from us (Binney and Merrifield, 1998). Two common types of observation for distant galaxies include a) photometric and b) spectroscopic observations, where the latter can produce more accurate red-shift estimates.

One way to estimate the red-shift from photometric observations is using a regression model (Connolly et al., 1995). We use a multi-layer Perceptron for this purpose and use the more accurate spectroscopic red-shift estimates as the ground-truth. As another baseline, we

have a photometric redshift estimate that is provided by the catalogue and uses various observations (including clustering information) to estimate individual galaxy-red-shift. Our objective is to use clustering information of the galaxies to improve our red-shift prediction.

5.3.2 *Model*

Formally, our problem can be described as follows: Given N galaxy clusters of $X^{(1)}, ..., X^{(N)}$, where each galaxy cluster can be represented as a set of features of M galaxies in the galaxy cluster, *i.e.* $X^{(i)} = \{x_1^{(i)}, ..., x_M^{(i)}\}$. The learning task is to regress on a property $y_m^{(n)}$ corresponding to every galaxy m with features $x_m^{(n)}$ in any galaxy cluster n, like red shift. That is we want to learn a function, $f : \mathfrak{X}_M^{(i)} \to y_M^{(i)}$ that takes only features as input and output are the per galaxy cosmological parameters.

In order to regress on a property y_m corresponding to every galaxy m with features x_m in any galaxy cluster X, the function should posses an *equivariance* such that the application of the function on a different order of the input galaxy transforms the output labels correspondingly, *i.e.* for any permutation π of the input galaxy:

$$f([x_{\pi(1)}, \ldots, x_{\pi(M)}]) = [y_{\pi(1)}, \ldots, y_{\pi(M)}] \tag{5.6}$$

Note that the prediction for each galaxy does not change by permuting the members of the galaxy cluster. Therefore, we can treat each galaxy cluster as a "set" and use permutation-equivariant layer to estimate the individual galaxy red-shifts.

5.3.3 *Experiment*

5.3.3.1 *Dataset*

For each galaxy, we have 17 photometric features [2] from the redMaPPer galaxy cluster catalog (Rozo and Rykoff, 2014), which contains photometric readings for 26,111 red galaxy clusters. In this task in contrast to the previous ones, sets have different cardinalities; each galaxy-cluster in this catalog has between $\sim 20 - 300$ galaxies – *i.e.* $\mathbf{x} \in \mathbb{R}^{N(c)\times 17}$, where $N(c)$ is the cluster-size. See Figure 5.1(a) for distribution of cluster sizes. The catalog also provides accurate spectroscopic red-shift estimates for a *subset* of these galaxies as well as photometric estimates that uses clustering information. Figure 5.1(b) reports the distribution of available spectroscopic red-shift estimates per cluster.

5.3.3.2 *Method*

We randomly split the data into 90% training and 10% test clusters, and use the following simple architecture for semi-supervised learning. We use four permutation-equivariant layers with 128, 128, 128 and 1 output channels respectively, where the output of the last layer is used as red-shift estimate. The squared loss of the prediction for available spectroscopic red-shifts is minimized.[3] Figure 5.1(c) shows the agreement of our estimates with spectroscopic readings on the galaxies in the test-set with spectroscopic readings. The figure also compares the photometric estimates provided by the catalogue (Rozo and Rykoff, 2014), to the ground-

2 We have a single measurement for each u,g,r, i and z band as well as measurement error bars, location of the galaxy in the sky, as well as the probability of each galaxy being the cluster center. We do not include the information regarding the richness estimates of the clusters from the catalog, for any of the methods, so that baseline multi-layer Preceptron is blind to the clusters.

3 We use mini-batches of size 128, Adam (D. P. Kingma and Ba, 2014), with learning rate of .001, $\beta_1 = .9$ and $\beta_2 = .999$. All layers except for the last layer use Tanh units and simultaneous dropout with 50% dropout rate.

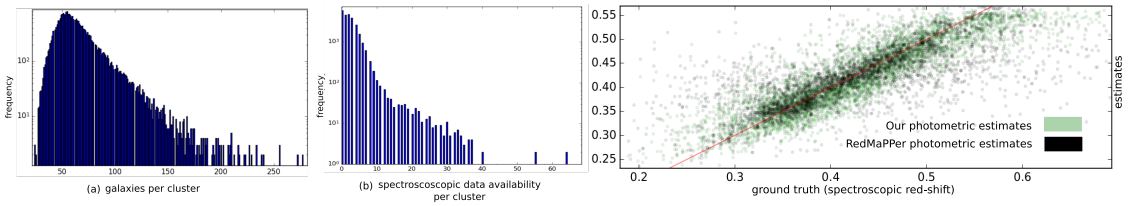


Figure 5.1: Application of permutation-equivariant layer to semi-supervised red-shift prediction using clustering information: (a) distribution of cluster (set) size; (b) distribution of reliable red-shift estimates per cluster; (c) prediction of red-shift on test-set (versus ground-truth) using clustering information as well as RedMaPPer photometric estimates (also using clustering information).

truth. As it is customary in cosmology literature, we report the average **scatter** $\frac{|z_{\text{spec}}-z|}{1+z_{\text{spec}}}$, where z_{spec} is the accurate spectroscopic measurement and z is a photometric estimate.

5.3.3.3 *Performance*

Our result is summarized in Table 5.2. The average scatter using **our model** is .023 compared to the scatter of .025 in the **original photometric estimates** for the redMaPPer catalog. Both of these values are averaged over all the galaxies with spectroscopic measurements in the test-set. We repeat this experiment, replacing the permutation-equivariant layers with fully connected layers (with the same number of parameters) and only use the individual galaxies with available spectroscopic estimate for training. The resulting average scatter for **multi-layer Perceptron** is .026, demonstrating that using clustering information indeed improves photometric red-shift estimates.

Method	Avg Scatter
MLP	0.026
redMaPPer	0.025
DeepSets	0.023

Table 5.2: Red-shift experiment. Lower scatter is better.

5.4 EXPLORATORY DATA ANALYSIS: SATELLITE IMAGERY

There has never been a more exciting time to observe humanity and understand it's impact on the world. Recognition of such patterns through similarity search has helped us in various domains: wildlife tracking and identifying conflicts (Suju and Jose, 2017), discovering survivors in battlefield dynamics (Y. Han et al., 2015), detecting fraudulent online behavior (Ngai et al., 2011), finding diagnosis using medical histories (Korn et al., 1996), etc. We aim to help people discover such patterns and repeat this success story using satellite data.

In light of this, we present Terrapattern, a prototype to demonstrate a workflow by which lay users - such as journalists, citizen scientists, humanitarian agencies, and others - can easily search for visually consistent "patterns of interest". Our goal is to provide a geospatial software tool that makes it easy for people, who may lack expertise in machine vision,

1. to specify an image that they are interested in and automatically find similar examples
2. to provide the locations of those instances in a common data format that easily allows for further examination

After its launch, Terrapattern was well received by the media and the public at large (see (Coldeway, 2016; Robinson, 2016; Ryan, 2016)).

Terrapattern consists of two components: a feature extraction service based on Deep Nets (see Section 5.4.3) and the main contribution of this chapter, a nearest neighbor (NN) search. To be popular in the aforementioned task, such a NN search service would have to satisfy the following requirements:

- **Online**: Be responsive for interactive queries by artists, *i.e.* to be able to handle online (not batch mode) queries over large high dimensional satellite imagery datasets in real-time. This rules out many efficient batch-mode NN search service running on GPUs (Jeff Johnson, Douze, and Jégou, 2017).
- **High Recall**: Must have high recall as humans are the direct consumers of the results of the NN search. Furthermore, the number of neighbors needed would be small. The super-fast approximate NN search methods work best when they must output a high number of neighbors at moderate recall (*e.g.* as an intermediary step of an automated ML pipeline), and thus are not applicable.
- **Fast Construction**: Allows for fast construction/indexing. For example, artists can load various datasets and start playing with it immediately.
- **Distributed**: For extremely large datasets that do not fit on a single machine, the NN search service should be distributable over multiple machines, without high cost of synchronization in case of updates.
- **Malleable**: Possible to modify (insert/delete points) the search index. This is useful, for example, when a higher resolution satellite imagery becomes available for certain region.

Since none of the existing NN search service meets all of our requirements, we develop a tree data structure Stable Greedy Tree (SG-Tree). The proposed tree data structure allows for almost linear time construction and fast query time for exact NN search under real world assumptions (Section 5.4.1.3). Perhaps the most surprising features, are that SG-Tree is extremely simple and as a result, our construction time is orders of magnitude faster than any comparable algorithm. While these complexities are quite standard, the major distinguishing factor of SG-Tree is its simplicity (Section 5.4.1.5).

Historically, NN Search has been used to classify unlabeled data based on a existing labeled data-set. Underlying this task is an assumption that the exists data-set has some inherent clustering. Existing methods such as Locally Sensitive Hashing, often implicitly use this assumption, whereas we explicitly use this assumption. In some sense, our data structure is designed to perform well on data that has some clustering.

At a high level, SG-Tree tries to create a hierarchical tree where each node performs a "coarse" clustering. The centers of these "clusters" become the children and subsequent insertions are recursively performed on these children. When performing the NN query, we prune out solutions based on a subset of the dimensions that are being queried. This is particularly useful when trying to find the nearest neighbor in highly clustered subset of the data, *e.g.* when the data comes from a Gaussian. The effect of these two optimizations is that our data structure

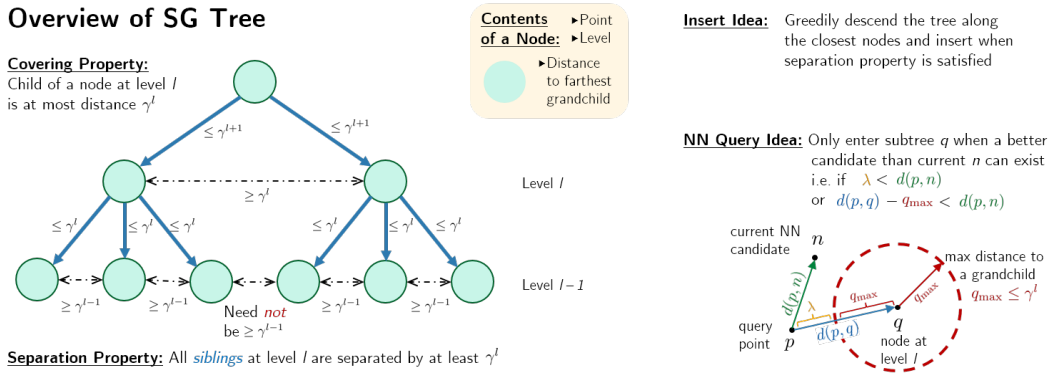


Figure 5.2: Overview of proposed hierarchical data structure that allows fast retrieval in logarithmic time.

is extremely simple, highly parallelizable and is comparable in performance to existing NN implementations on many data-sets.

Another appealing aspect of our data structure is its simplicity. As a result, it is amenable to theoretical guarantees. Existing data structures with theoretical guarantees such as Navigating Net (Krauthgamer and J. R. Lee, 2004), Cover Trees (Beygelzimer, Kakade, and Langford, 2006), HNSW (Malkov and Yashunin, 2016), P-DCI, (K. Li and Malik, 2017), RP-tree (Dasgupta and Sinha, 2013), are either sequential in nature or hide large constants. We are able to show that SG-Tree inherits some of the nice properties of Cover Trees and adapts to the intrinsic dimension of the data. This is particularly beneficial when the feature vectors come from neural networks that have high euclidean dimensions but often lie on a (significantly) smaller dimensional manifold. In a similar vein, (G. H. Chen, D. Shah, et al., 2018) have used the clusterability of the data to give non-asymptotic guarantees on the NN search. We take a similar step and show that if the data comes from a mixture of Gaussians, then the resulting tree structure has some nice properties. Our work, like that of (G. H. Chen, D. Shah, et al., 2018), relies on the clusters being sufficiently far away so that noise is unlikely too many points to appear in the wrong cluster. Although, this work only takes the first steps, we believe it should be possible to give data-structures that have strong theoretical guarantees and are quite fast in practice.

5.4.1 *Stable Greedy Trees*

5.4.1.1 *Definition*

Stable Greedy Tree (SG-Tree)is defined with respect to a hyper-parameter $\gamma > 1$ over a dataset S in a metric space with distance $d(\cdot,\cdot)$. SG-Tree is an example of a hierarchical trees where the input points are leaves and each input point is associated with a unique internal point. Nodes at a certain height are said to be in the same level. When we compress this tree (*i.e.* if a node has only itself as a child, then we simply delete this child), it has the property that each node is associated to a unique level.

The nodes of SG-Tree contains following attributes:

- `point`: The data point vector
- `level`: The level of the node in the tree as integer
- `max`: The upper bound to distance of furthest node in the current subtree.

Insertion: To insert a node at a given subtree rooted at r at level l , we find the closest child p. If $d(p, q) \leqslant \gamma^{l-1}$, then we recursively insert q into the subtree defined at p. On the other hand, if $d(p, q) > \gamma^{l-1}$ then we insert it a sibling in the current level and connect it to the parent at the last level. This is outlined in Algorithm 5.1.

Query: To find the closest neighbor to a query point, we traverse down the levels of the SG-Tree starting from the root and maintain the current best node seen. At each level, we only maintain nodes that could still contain the nearest neighbor to a given query point p. To help prune out bad nodes, we first prune nodes using the first half of the coordinates before running the entire distance. We make this formal in Algorithm 5.2.

5.4.1.2 *Connections to Net Trees and Cover Trees*

SG-Tree is inspired by and closely related to Cover Trees introduced by Beygelzimer, Kakade, and Langford (2006). Cover trees form a hierarchical data structure that allows fast retrieval in logarithmic time when the metric has a small expansion constant (defined below). In particular, it allows for $O(n \log n)$ construction time, $O(\log n)$ retrieval, and it only depends polynomialy on the expansion rate. Moreover, the degree of all internal nodes is well controlled.

Cover trees are defined as an infinite succession of levels S_l with $l \in \mathbb{Z}$. Each level l contains (a nested subset of) the data with the following properties:

- Nesting property: $S_{l-1} \subseteq S_l$.
- Separation property: All $p, q \in S_l$ satisfy $d(p, q) \geqslant \gamma^l$.
- All $q \in S_{l-1}$ have a parent in $p \in S_l$, possibly with $p = q$, with $d(p, q) \leqslant \gamma^l$.

We compress the data structure so each p is represented only once: it is only stored in the largest level l for which $p \in S_l$. This data structure has a number of highly desirable properties, as proved in Beygelzimer, Kakade, and Langford (2006). SG-Tree does not satisfy the separation property globally for all children in S_l but only for siblings under a common parent. In that sense, SG-Tree changes this property from being a global one to a *local* one. This slight modification will have huge repercussions in performance, as it will allow us to insert in parallel and distribute query.

Another very related data structure is Net Trees introduced by Krauthgamer and J. R. Lee (2004). Their data structure is also a hierarchical tree that has stronger properties. In fact, Jahanseir and Sheehy (2016) showed that Net Trees capture Cover Trees for an appropriate choice of parameters. They get stronger guarantees in part by maintaining epsilon nets at each level which are constructed greedily, and maintain lists of relatives at each point that maintain close neighbors. SG-Tree also builds an epsilon net, however only on the points that are in a ball around some node. Once again, we substitute a global property for a local one. These changes lose many of the theoretical properties but we gain by having a simple implementation that can be optimized for modern architectures.

5.4.1.3 *Structural Properties*

In this section, we show that the SG-Tree shares some of the properties of Cover Trees in terms of tree structure and construction time.

Definition 5.1 *Let* $\delta_{\max} := \max_{p,q \in S} \|p - q\|$ *denote the maximum distance between any two points in* S *and* $\delta_{\min} := \min_{p \neq q \in S} \|p - q\|$ *denote the minimum distance between any two points. We define the* ***aspect ratio*** *of the metric to be* $\Delta := \delta_{\max}/\delta_{\min}$.

Theorem 5.2 *The height of the SG-Tree is at most* $O(\log(\Delta))$ *where* Δ *is the aspect ratio of the metric.*

Algorithm 5.1 Insert a new point into SG-Tree

```
1: function INSERT(Node n, Point p)
2:     # Start with n ← root
3:     Find child c of n closest in distance to p
4:     if d(c, p) < γ^c.level then
5:         if c.max < d(c, p) then
6:             c.max ← d(c, p)
7:         end if
8:         Insert(c, p)
9:     else
10:        Assign p as child of n
11:    end if
12:    return
13: end function
```

Proof Observe that the deepest level is at least $\log_\gamma(\delta_{\min}/2)$ as all balls of radius $\delta_{\min}/2$ are disjoint. The top level is at most $\log_\gamma(2 \cdot \delta_{\max})$ as all points are at most $\delta_{\max}$ apart. The total height of the tree is at most the difference in these levels: $O(\log(\frac{\delta_{\max}}{\delta_{\min}}))$. ■

We use the definition of expansion constant α and the following lemmas from Beygelzimer, Kakade, and Langford (2006).

Definition 5.3 *The expansion constant α is defined as smallest $\alpha \geqslant 2$ such that $|\mathcal{B}(p, 2r)| \leqslant \alpha \cdot |\mathcal{B}(p, r)|$ for all $p \in S$ where $\mathcal{B}(p, r)$ denotes a ball of radius r around the point p.*

Lemma 5.4 *The maximum degree of any node in this tree is at most α^3.*

Proof Let r be a node at level l and let $c_1, \ldots, c_t$ be its children. Each c_i creates a disjoint ball of radius $\gamma^{l-1}/2$ as we know that $d(c_i, c_j) \geqslant \gamma^{l-1}$. WLOG let c_1 be the ball that contains the least number of points in a ball of radius $\mathcal{B}(c_1, \gamma^{l-1}/2)$. Observe that $\mathcal{B}(c_1, \gamma^{l-1}/2) \subseteq \mathcal{B}(r, \gamma^l) \subseteq \mathcal{B}(p, 2 \cdot \gamma^l)$. Since the larger ball is at most $4 \cdot \gamma$ times bigger than the smaller ball, and we choose $\gamma < 2$, we can bound the number of total points by the expansion constant. In particular, $|\mathcal{B}(p, 2 \cdot \gamma^l)| \leqslant \alpha^3 \cdot |\mathcal{B}(p, \gamma^{l-1}/2)|$. Since we chose p_1 so that $B(p_1, \gamma^{l-1}/2)$ to have the least number of points, we know that there can be at most α^3 such nodes. ■

Corollary 5.5 *The time to insert a new point p is $\alpha^3 \log(\Delta)$.*

Proof This is an easy consequence of Lemma 5.4 and 5.2. Upon insertion, we query all the children of a node which is at most α^3 and proceed to the next level. Since there are at most $\log(\Delta)$ levels and each node has at most α^3 children, this gives us the required bound. ■

5.4.1.4 *Average Case Analysis*

Since we do not have a global seperation property, this severely limits our ability to analyze the tree under a *worst case* model. It is therefore necessary to go beyond worst-case and analyze the model in an average case model.

Algorithm 5.2 Find nearest neighbor in SG-Tree

```
1: function NN(Node r, Query point p, Candidate NN n)
2:     # Start with r ← root and n ← root
3:     if d(p, r) < d(p, n) then
4:         n ← r
5:     end if
6:     for each child q of r do
7:         #high dim data: prune in two steps
8:         if d_{1/2}(p, q) − q.max < d(p, n) then
9:             if d(p, q) − q.max < d(p, n) then
10:                n ← NN(q, p, n)
11:            end if
12:        end if
13:    end for
14:    return n
15: end function
```

In this section, we assume that the points are drawn from the uniform distribution. We do this to elucidate the main ideas clearly, however, these ideas can be extended to the case where the points are drawn from any sufficiently symmetric distribution (this includes the Gaussian distribution).

To be more specific, we will assume that the points are drawn uniformly independently at random from a ball of radius R in some ambient dimension d. Our proofs would also work if these points are subsequently embedded in a higher dimensional space via a linear transformation. In this sense, it is a simple linear generative model that we study. We will assume that $d = O(1)$ or some small growing function of n. (Otherwise, most points are very far away from each other and are not very interesting for NN search.) This implies that the expansion coefficient is $\alpha = O(2^d)$.

The key to our understanding in this regime is the following lemma.

Lemma 5.6 *Suppose S is drawn from the uniform distribution as described above. Then with high probability each node in the first k levels has at least $\Omega(\log n)$ nodes in its subtree when $k = O(\log \log n)$.*

Proof We will do this by induction on each level and by the method of deffered decisions. In particular, Conditioning on the first k levels of SG-Tree, we will argue that with high probability, each node will contain a "voronoi" region that has a significant measure. Hence, the probability that the remaining nodes do not fall in that cell is exponentially small. Thus we can guarantee with probabity that at least a $1/\text{poly}(n)$ at least $\Omega(\log n)$ points will fall in a particular node. Taking a union bound finishes the proof. ■

Using the above lemma we can prove two important facts about the SG-Tree when S and q is drawn from the uniform distribution.

Lemma 5.7 *Find neighbor takes on at most $O(\alpha^4 \log \Delta)$ time per query when the input points S are drawn from the uniform distribution and the query q in expectation.*

Proof Assume we insert the query point p into the tree and let us consider the root to leaf path $p_1 = r, \ldots, p_t$. Observe the Algorithm 5.2 will also search along this path first.

By Lemma 5.6, we know that with high probability that there are at least $\Omega(\log n)$ nodes in the node p_k where $k \in o(\log \log n)$. Thus the distance $\tilde{d} = d(p_{\log \log n}, q)$ is a good upper-bound on the optimal distance d^*. Using this information, it remains to show two parts. The set of nodes Q_l queried at each level l is at most $O(\alpha)$ when $l = o(\log \log n)$. For the remaining nodes, since we have a good estimate, we will argue that with high probability there are only $O(\alpha)$ nodes that will be better than this estimate. Since we might query each node and its children, we can bound this total by $O(\alpha^4)$. ■

Lemma 5.8 *Given a SG-Tree with n points coming from a uniform distribution on a ball of radius R in dimension d. the probability of a new point being inserted in top k levels is at most $n^{(-\Theta_d(1))}$ when k and d are constants and n is large.*

Proof Let k be a constant and we will bound the probability that a new point p is inserted at level k as a child of node r. A point p gets inserted as a child of r at level k only if there is no other existing child q violating the separation constraint, i.e. $d(p, q) \geqslant \gamma^{l-1}$. This requires that of all the points inserted as children of r, none of them fall in $\mathcal{B}(r, \gamma^l) \cap \mathcal{B}(p, \gamma^{l-1}$ given $p \in \mathcal{B}(r, \gamma^l)$. Observe that this is a function of d and γ but not n.

Since k is a constant, we can use Lemma 5.6 to show that the number of points expected to fall in under node r is going to be $t := \Omega(\log n)/V_d(R) \cdot V_d(\gamma^l)$ with high probability. It remains to calculate the probability conditioned on t points arriving inside $\mathcal{B}(r, \gamma^l)$ we can assume by symmetry that they are uniformly chosen in this ball. Let $\rho = |\mathcal{B}(r, \gamma^l) \cap \mathcal{B}(p, \gamma^{l-1})|/|\mathcal{B}(r, \gamma^l)|$ denote an uppper bound on the probability that a point falls in this region. The value ρ depends only on d and not on n (In fact, it is roughly $O(1/\gamma^d)$). The probability that all t points ill be away from this is at least $(1 - \rho)^t$. This event occurs with probability roughly $e^{-\rho t} \sim e^{\Theta_d(\log n)}$. Taking a union bound over all n points, we see that with high probability, a new insertion will not create a new node in the top k levels. . ■

5.4.1.5 *Practical Deployment*

SG-Tree has simple construction and query procedures as outlined in Algorithm 5.1 and 5.2. The main vantage point of SG-Tree over the theoretically guaranteed fast exact search method cover trees, come from absence of a global separation constraint, making parallelization hard. On a modern heavily multi-cored CPU, we have a threadpool and want to insert a batch of points in parallel using the threadpool. In the case of a cover tree, if any worker thread inserts a point, every other worker thread has to ensure that the inserted point does not violate the global separation property for their point. This results in a lot of contention and deadlock, and significantly poor performance. In Beygelzimer, Kakade, and Langford (2006), a batch insert method is proposed for cover tree which is much faster than sequential insert on a single threaded environment. Nonetheless, the algorithm is inherently sequential and it is not clear how to modify it to make use of modern multi-threaded compute resources. In SG-Tree, a worker node inserting a point does not need to worry about modifications to the tree happening at places other than current node because separation property needs to be only locally maintained among siblings. Even in case the of current node being modified, the worker thread does not need to discard any work, but only has to account for the newly inserted point. Thus,

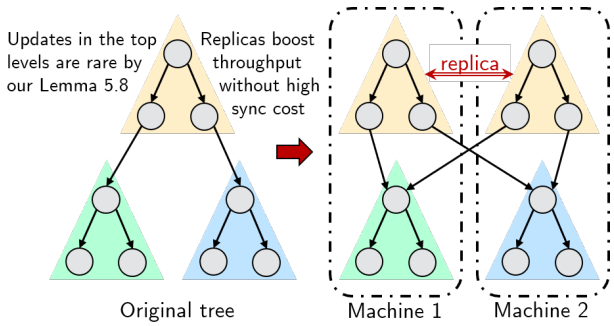


Figure 5.3: A natural distributed implementation of SG-Tree. Multiple replicas can be maintained at cheaply, i.e. with low synchronization cost as modifications to top layers are guaranteed to be rare by Lemma 6.

a very efficient construction for SG-Tree can be implemented in a work-stealing Fork/Join framework.

In other methods, like HNSW, RP-Trees, etc. the bulk of the work can be easily parallelized, but the total work needed is often much higher. For example, building the small world graphs in worst case is $O(n^2)$ (Fu et al., 2017), and unknown for others.

HEURISTICS FOR IMPROVED PERFORMANCE We discovered a practical trick that makes SG-Tree (and cover trees as well) more balanced and thus increasing performance: Start with sizable chuck of data points representative of the data distribution. In this chunk, find the point closest to the mean. Use this point as the root of the tree. Next, insert remaining points from the chunk in descending order of the distance. Despite slight additional work at the beginning, this trick reduces both the overall construction time and query search time. We conjecture that this trick allows top layers of the tree to quickly spread out, identifying different regions in the data and allowing more effective pruning during search.

EXTENSION TO SOME IMPROPER DISTANCES Apart from finding NN according to L2 distance in Euclidean space, there are other popular naturally occurring criteria that are not proper metrics such as: angular distance, maximum cosine similarity or maximum inner product search. These criteria do not obey triangle inequality, so SG-Tree cannot be directly used Following P. Ram and Gray (2012) and Bachrach et al. (2014), we can show that SG-Tree can be used for even such improper distances by performing suitable and cheap transformations of the data points.

- Maximum Cosine Similarity Search (MCSS): Instead of inserting data points x_i in SG-Tree, insert normalized version $\tilde{x}_i = x_i/\|x_i\|$. Performing NN search in ℓ_2-distance is equivalent to MCSS as:

$$\begin{aligned} \mathrm{NN}(q|\tilde{x}) &= \operatorname*{argmin}_i \|\tilde{x}_i - q\| = \operatorname*{argmin}_i \|\tilde{x}_i - q\|^2 \\ &= \operatorname*{argmin}_i \|\tilde{x}_i\|^2 + \|q\|^2 - 2\langle \tilde{x}_i, q\rangle \\ &= \operatorname*{argmax}_i \langle \tilde{x}_i, q\rangle = \operatorname*{argmax}_i \frac{\langle x_i, q\rangle}{\|x_i\|\|q\|} \\ &= \mathrm{MCSS}(q|x) \end{aligned} \tag{5.7}$$

- Maximum Inner Product Search (MIPS): As before, we will insert transformed points in SG-Tree. Assume, we can have an upper bound u for the L2 norm, i.e. $u \geqslant \|x\|, \forall x$. Instead of inserting original data points x_i, we insert the augmented data points $\tilde{x}_i = [x_i; \sqrt{u^2 - \|x_i\|^2}]$ in SG-Tree. The query vector q is also modified as $\tilde{q} = [q; 0]$. Now performing NN search in L2 distance with modified query is equivalent to MIPS for original query as:

$$\begin{aligned} \text{NN}(\tilde{q}|\tilde{x}) &= \operatorname*{argmin}_i \|\tilde{x}_i - \tilde{q}\| = \operatorname*{argmin}_i \|\tilde{x}_i - \tilde{q}\|^2 \\ &= \operatorname*{argmin}_i u^2 + \|q\|^2 - 2\langle x_i, q\rangle \\ &= \operatorname*{argmax}_i \langle x_i, q\rangle = \text{MIPS}(q|x) \end{aligned} \tag{5.8}$$

DISTRIBUTED HIGH THROUGHPUT IMPLEMENTATION For large scale data, such as satellite imagery data of whole earth (> 8TB), that do not fit in memory of a single computer, one resorts to distribution across multiple machines. In a natural setup for a distributed tree search, as first proposed by Patwary et al. (2016) using kd-trees, first the data would be spatially partitioned into roughly equal size portions and distributed among the nodes. Then a global kd-tree containing representation data points of these spatial partitions would be constructed and would be used to direct queries/inserts to appropriate nodes. Results from the polled nodes would be aggregated and returned. However, with kd-tree such spatial partitioning of can be very expensive to construct. Moreover with new points coming in the spliting points/directions in the global kd-tree might have to be updated often to maintain fast look-ups.

SG-Tree has favorable properties for distribution. First, no two-step construction is needed. The tree can be grown normally till memory is exhausted, at which point sub-trees starting from level k can be distributed among other nodes. The data can be continued to be added after distribution. Unlike kd-trees, in SG-Tree there is no axis aligned splits that have to be recomputed with new points coming in. Second, the top-k levels of SG-Tree can be replicated across all nodes, without incurring much synchronization penalty. This is because by Lemma 6, modifications in top-k levels of SG-Tree are rare, thus replica would hardly needed to be synchronized. Thus, every node can process incoming queries and direct it to appropriate nodes, thereby increasing the throughput as illustrated in Figure 5.3.

5.4.2 *Experiments*

In this section, we present empirical studies comparing SG-Tree with other data structures. These show that SG-Tree has fast construction speed and competitive query time meeting the needs of TerraPattern.

DATASETS To test the above claims, we evaluate on a number of datasets from UCI repository[4] and from the popular ANN-Benchmark[5], as listed in Figure 5.4 along with their size (N) and dimensionality (D). These are multivariate datasets from a varied set of sources meant to provide a broad picture of performance across different domains. They also include a mix of Euclidean and Angular distance (1-cosine similarity) as the natural metric for NN query among the selected datasets. The datasets come with a train and test split; we use the train set to build the NN index and test set for query.

METHOD As our metric for comparison, we use the "experience time", i.e. the total time taken for constructing the index and performing 1k queries to find the 10-NN in the index for each

4 https://archive.ics.uci.edu/ml
5 https://github.com/erikbern/ann-benchmarks

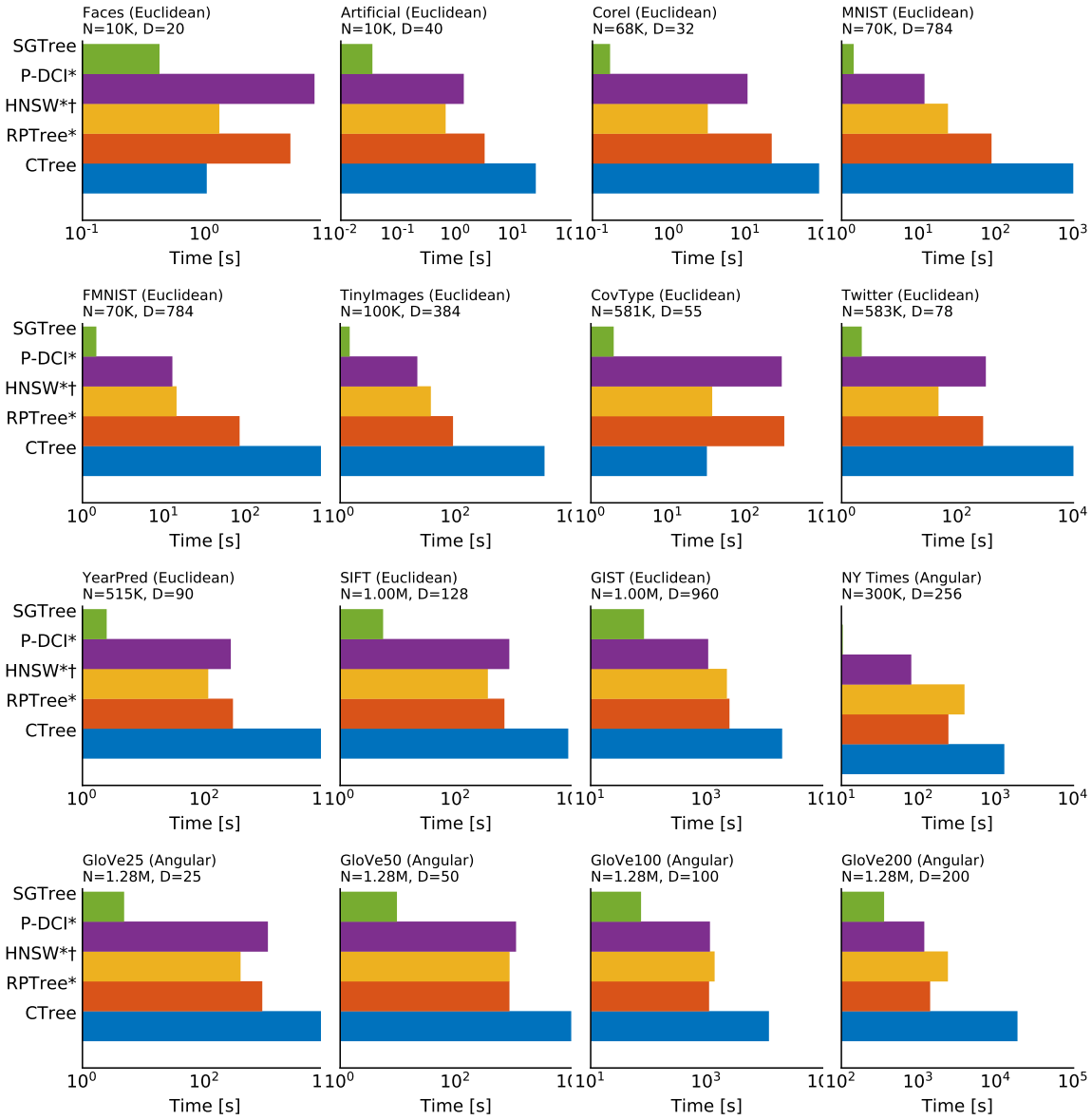


Figure 5.4: Comparison of SG-Tree on numerous benchmark datasets in terms of "experience time", i.e. total time taken to build the index and perform 1K queries. Approximate NN methods are marked with * and for such methods the hyper-parameters are chosen to produce Recall@10 $\geqslant 0.99$ on all the datasets. Non-modifiable data structures are marked with †.

query vector. This metric better reflects the experience of an user who interactively wants to perform some NN search. The traditional metrics of construction time and time taken per query is provided in Table 5.3 and 5.4. We compare our multi-threaded implementation of SG-Tree in C++14 against following methods, each covering a broad class of NN search strategy:

- Cover Tree: We use an in-house C++14 implementation of Cover Tree as no open source multi-threaded code was available that exactly implemented cover tree. [6] Although con-

6 MLPack claims to implement Cover Tree, but they ignore the global separation property https://github.com/mlpack/mlpack/blob/master/src/mlpack/core/tree/cover_tree/cover_tree.hpp#L39

	Dataset	N	D	Construction Time [s]				
				CoverTree	RPTree*	HNSW*†	P-DCI*	SG-Tree
Euclidean	Artificial	10K	40	2.38E+01	2.47E+00	5.82E-01	5.34E-01	2.01E-02
	Faces	10K	20	1.01E+00	4.27E+00	1.25E+00	7.00E+00	4.14E-01
	Corel	68k	32	8.84E+01	2.07E+01	3.08E+00	8.67E+00	1.65E-01
	MNIST	70K	784	9.71E+02	8.41E+01	2.28E+01	9.56E+00	8.41E-01
	FMNIST	70K	784	9.01E+02	8.12E+01	1.36E+01	1.03E+01	8.75E-01
	TinyImages	100K	384	3.28E+03	8.55E+01	3.52E+01	1.75E+01	9.78E-01
	CovType	581K	55	3.10E+01	3.10E+02	3.63E+01	2.84E+02	1.90E+00
	Twitter	583K	78	9.74E+03	2.68E+02	4.55E+01	2.95E+02	2.09E+00
	YearPred	515K	90	9.75E+03	2.77E+02	1.10E+02	2.51E+02	1.69E+00
	SIFT	1.00M	128	8.41E+03	6.64E+02	3.44E+02	8.02E+02	2.56E+00
	GIST	1.00M	960	1.94E+04	2.39E+03	2.16E+03	1.01E+03	1.96E+01
Angular	NYTimes	300K	256	1.25E+03	2.35E+02	3.80E+02	7.54E+01	6.29E+00
	GloVe25	1.28M	25	8.47E+03	8.35E+02	3.69E+02	1.03E+03	2.44E+00
	GloVe50	1.28M	50	9.46E+03	8.11E+02	8.16E+02	1.05E+03	3.60E+00
	GloVe100	1.28M	100	1.15E+04	1.06E+03	1.33E+03	1.10E+03	6.15E+01
	GloVe200	1.28M	200	1.85E+04	1.37E+03	2.33E+03	1.14E+03	3.32E+02

Table 5.3: Comparison of SG-Tree on numerous benchmark datasets in terms of construction time on the task of building NN graph. Approximate NN methods have been marked with * and for such methods the hyper-parameters are chosen so as to produce Recall@10 $\geqslant 0.99$ on all the datasets. Non-modifiable data structures are marked with †.

struction is not parallelizable, we tried to parallelize distance computations where possible. It is an exact search method.

- Random Projection (RP) Tree: We use Annoy[7], an highly optimized implementation of forest of RP trees in C++11. It is an approximate search method based on space partitioning.
- Hierarchical Navigation Small World (HNSW) Graphs: We use the excellent implementation of HNSW from NMSLIB[8] in C++11. The method builds a proximity graph at multiple resolutions for the points in index. It is also an approximate search method in the category of neighborhood based methods.
- Prioritized Dynamic Continuous Indexing (P-DCI): We use the multi-threaded C code[9] released by the authors. The method is a clever application of Johnson-Lindenstrauss lemma to query in lower dimension at a much lower cost. Theoretically, it is an exact search method with high probability, but in practice can be considered to be an approximate search method as well.

7 https://github.com/spotify/annoy
8 https://github.com/nmslib/nmslib/
9 https://github.com/UOMXiaoShuaiShuai/PDCI

	Dataset	N	D	Avg Time per Query [s]				
				CoverTree	RPTree*	HNSW*†	P-DCI*	SG-Tree
Euclidean	Artificial	10K	40	3.75E-05	6.00E-04	6.30E-05	8.00E-04	1.43E-05
	Faces	10K	20	7.05E-06	6.15E-04	3.80E-05	6.80E-04	5.19E-06
	Corel	68k	32	1.07E-05	6.10E-04	3.83E-05	1.63E-03	1.34E-06
	MNIST	70K	784	1.52E-03	9.38E-04	3.98E-04	1.98E-03	5.56E-04
	FMNIST	70K	784	8.02E-04	6.17E-04	2.20E-04	2.03E-03	5.72E-04
	TinyImages	100K	384	1.58E-03	7.56E-04	5.49E-04	3.36E-03	4.37E-04
	CovType	581K	55	9.54E-06	5.46E-04	5.10E-05	3.89E-03	3.11E-06
	Twitter	583K	78	2.29E-04	5.81E-04	7.97E-05	4.45E-03	7.05E-05
	YearPred	515K	90	1.06E-03	7.01E-04	2.94E-04	5.21E-03	7.30E-04
	SIFT	1.00M	128	3.79E-03	1.22E-03	5.03E-04	6.87E-03	2.80E-03
	GIST	1.00M	960	6.66E-02	6.04E-03	3.07E-03	1.60E-02	6.10E-02
Angular	NYTimes	300K	256	3.92E-03	1.74E-03	2.62E-03	2.92E-03	1.89E-03
	GloVe25	1.28M	25	2.24E-03	3.79E-04	3.79E-04	6.36E-03	2.22E-03
	GloVe50	1.28M	50	5.79E-03	1.56E-03	8.57E-04	7.21E-03	5.69E-03
	GloVe100	1.28M	100	1.05E-02	2.37E-03	1.35E-03	8.66E-03	1.02E-02
	GloVe200	1.28M	200	1.77E-02	3.63E-03	2.44E-03	1.08E-02	1.69E-02

Table 5.4: Comparison of SG-Tree on numerous benchmark datasets in terms of average time per query of 10-NN search. Approximate NN methods have been marked with * and for such methods the hyper-parameters are chosen so as to produce Recall@10 $\geqslant$ 0.99 on all the datasets. Non-modifiable data structures are marked with †.

For the approximate search methods, we selected the following hyper-parameters such that Recall@10 $\geqslant$ 0.99, *i.e.* we required the recall to be high as it is needed by TerraPattern.

- Cover Tree: It has only one hyper-parameters. Scale was chosen to 1.3 as suggest by Beygelzimer, Kakade, and Langford (2006). We use transformation presented in (5.7) to support angular criterion.
- RPTree/Annoy: It has only two hyper-parameters. We had to select `n_tree`=400 and `search_k`=400k. It inherently supports angular criterion.
- HNSW/NMSLIB: It has multiple hyper-parameters. Following the guide provided by developers, we had to select `M`=100, `efCons`=2000, `efSearch`=2000, and all others were default value. It inherently supports angular criterion.
- P-DCI: It has multiple hyper-parameters as well. Following the hints provided by authors, we had to select `num_comp_indices`=10, `num_simp_indices`=20, `num_levels`=3, `construction_field_of_view`=200, `construction_prop_to_retrieve`=1, `query_field_of_view`=400, and `query_prop_to_retrieve`=1. We use transformation presented in (5.7) to support angular criterion.
- SG-Tree: It has only one hyper-parameter: scale. Similar to cover tree, we chose it to be 1.3.

Also note that HNSW does not allow modification to the index, which is undesirable for our application, but we still include them in the comparison.

SETUP & HARDWARE We follow the experimental setup of the popular ANN-Benchmark . Similar to their protocol, the queries are not batched as we are interested in online queries. Instead each query is processed individually, but in parallel using a thread pool that saturates all the CPU cores. We run our experiments on an Amazon EC2 c5.18xlarge nodes having 72 virtual threads per node and 144GB of memory. For purpose of experiments, all data and calculations are carried out at single floating-point precision.

OBSERVATIONS Overall, Figure 5.4 is in line with our claim that SG-Tree can enhance user experience for interactive NN search compared to other methods because it takes lowest time for build plus search across multiple datasets of varying sizes, dimensionality, and distance. From Table 5.3, we can see that SG-Tree is way faster than all other methods over most of the datasets. Even in terms of time taken per query (or equivalently queries per second), SG-Tree is quite competitive with other state of the art methods like HNSW which are not modifiable as can be seen from Table 5.4. Another salient observation is regarding the GloVe datasets, which are word embeddings of various ambient dimensions 25, 50, 100, 200. Ideally all of these vector collections should contain similar semantic information and thus posses similar intrinsic dimension. As predicted by theory, the time scales linearly with ambient dimension.

5.4.3 *Terrapattern Service*

Terrapattern provides an open-ended interface for visual query-by-example, a test bed for our SG-Tree nearest neighbor search algorithm. Simply click an interesting spot on Terrapattern's map, and it will find other locations that look similar. Our tool is ideal for locating specialized 'nonbuilding structures' and other forms of soft infrastructure that aren't usually indicated on maps. Terrapattern is a "panoptic perceptron" that allows a user to perform arbitrary queries-by-example in satellite imagery. A guest clicks on a "feature of interest" in a satellite image; the Terrapattern system presents a batch of the most similar-looking places nearby; and the guest can then download a list of these locations in GeoJSON format. An example query is shown in Figure 5.5 for school bus parking.

For our purposes, "interesting" features are anthropogenic or natural phenomena that are not only socially or scientifically meaningful, but also visually distinctive–thus lending themselves ideally to machine recognition. Examples could include things like animal herds, methane blowholes, factories, destroyed homes, or logging roads. Many other patterns await discovery.

The Terrapattern search tool features three visualizations: a slippy map, for specifying visual queries; an "Geographical Plot" (or minmap), which shows the locations of search responses in the surrounding metro region; and a "Similarity Plot", which organizes the returned results within an abstract 2D space using Principal Component Analysis, or PCA. The Terrapattern website is built using Ruby and JavaScript, with satellite imagery from Google Maps, while the Geographical PLot and Similarity Plot were created in JavaScript using p5.js. The user-interface (UI) is shown in Figure 5.5.

The Terrapattern project seeks to democratize geospatial intelligence. By providing a means by which researchers, journalists, citizen scientists, and artists can quickly and easily scan extremely large geographical areas for specific visual features, our hope is that the public at large will gain the ability to research and answer questions about our world that are otherwise unknowable.

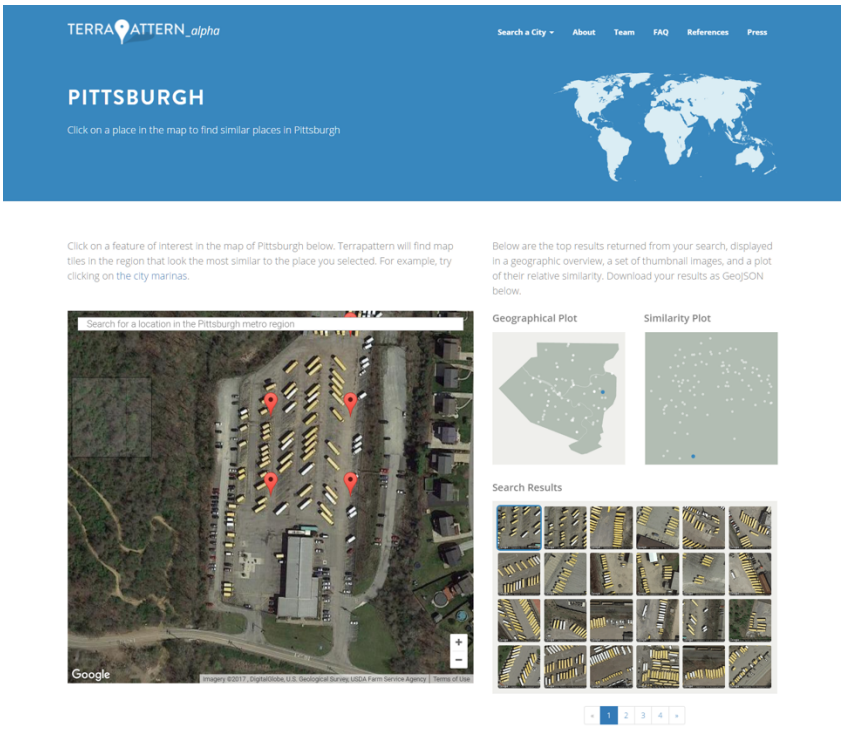
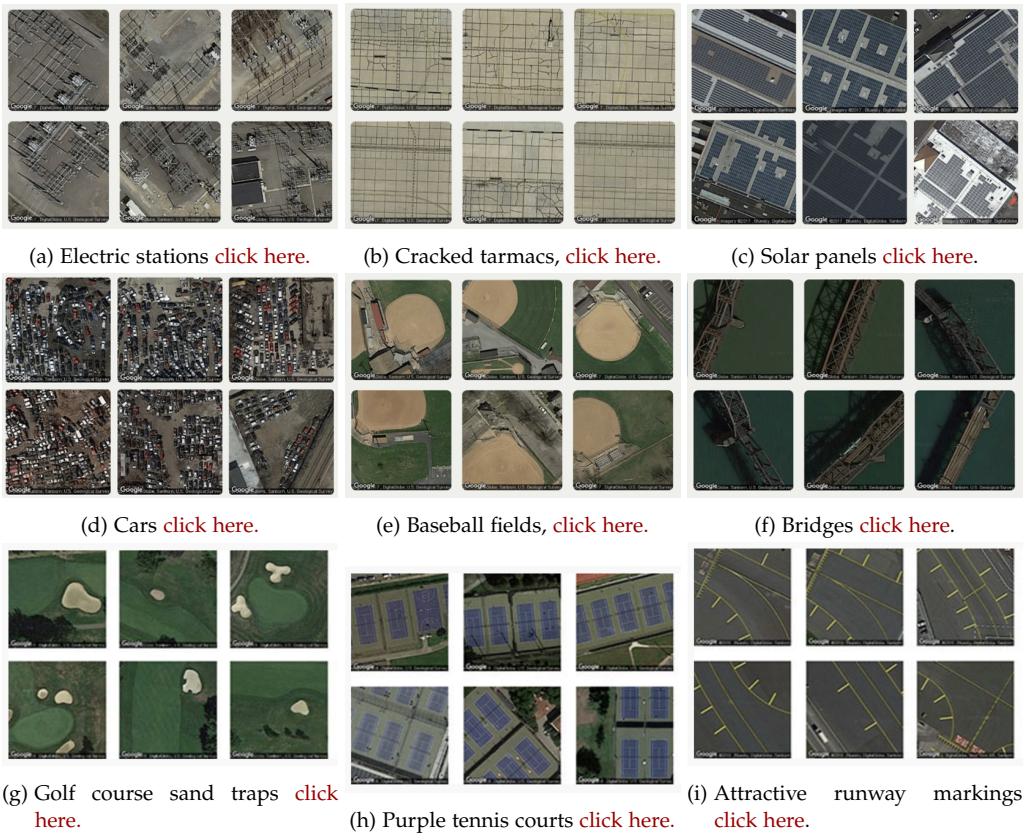


Figure 5.5: Using Terrapattern we can identify some of Pittsburgh's finest school bus depots. We click on one bus depot in the map, then Terrapattern uses SG-Tree nearest neighbor search to find other visually similar locations. The retrieved results location, similarity, and images are shown in the right. Any of the results can be further explored in the main window on left, where we can possibly issue another query based on a newly discovered feature of interest.

We emphasize that Terrapattern is a limited prototype. As of May 2018, it allows users to search in the greater metropolitan regions of a few major cities: New York City, San Francisco, Pittsburgh, Berlin etc. Altogether more than 5,200 square miles (13500 km^2) are fully searchable. Allowing high-resolution searches in size of the United States (e.g. 3.8M mi^2 or 9.9M km^2) is financially beyond the scope of project. Website: `http://www.terrapattern.com/`.

Terrapattern uses a deep convolutional neural network (DCNN), based on the ResNet ("Residual Network") architecture developed by He et al. (2016a). We trained a 34-layer DCNN using hundreds of thousands of satellite images labeled in OpenStreetMap, teaching the neural network to predict the category of a place from a satellite photo. Terrapattern was only possible due to the astonishing crowdsourced mapping effort of the OpenStreetMap project, which has generously categorized large parts of the world with its Nominatim taxonomy. We trained our DCNN using 466 of the Nominatim categories (such as "airport", "marsh", "gas station", "prison", "monument", "church", etc.), with approximately 1000 satellite images per category. Our resulting model, which took 5 days to compute on an nVidia 980 GPU, has a top-5 error rate of 25.4%. In the process, our network learned which high-level visual features (and combinations of those features) are important for the classification of satellite imagery.

After training the model, we removed the final classification layer of the network and extracted the next-to-last layer of the DCNN. Using this layer of proto-features (a technique called "transfer learning"), we computed descriptions for millions more satellite photos that cover a few metropolitan regions. When we want to discover places that look similar to your query, we just



(a) Electric stations click here. (b) Cracked tarmacs, click here. (c) Solar panels click here.

(d) Cars click here. (e) Baseball fields, click here. (f) Bridges click here.

(g) Golf course sand traps click here. (h) Purple tennis courts click here. (i) Attractive runway markings click here.

Figure 5.6: Some examples of interesting patterns found by our method.

have to find places whose descriptions are similar to those of the tile you selected. To perform this search in near real time, we use the proposed SG-Tree algorithm for kNN.

Some interesting example searches patterns are shown in Figure 5.6. It is important to point out that the Terrapattern system was not trained on any of the categories shown below, but instead recognizes them because of their common visual features.

5.5 EXPLORATORY DATA ANALYSIS: DOCUMENT MODELING

Latent Dirichlet Allocation (LDA) is a Bayesian technique that is widely used for inferring the topic structure in corpora of documents. It conceives of a document as a mixture of a small number of topics, and topics as a (relatively sparse) distribution over word types (D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003). These priors are remarkably effective at producing useful results. However, our intuitions tell us that while documents may indeed be conceived of as a mixture of topics, we should further expect topics to be *semantically coherent*. Indeed, standard human evaluations of topic modeling performance are designed to elicit assessment of semantic coherence (J. Chang et al., 2009; Newman, Karimi, and Cavedon, 2009). However, this prior preference for semantic coherence is not encoded in the model, and any such observation

of semantic coherence found in the inferred topic distributions is, in some sense, accidental. In this paper, we develop a variant of LDA that operates on continuous space embeddings of words—rather than word types—to impose a prior expectation for semantic coherence. Our approach replaces the opaque word types usually modeled in LDA with continuous space embeddings of these words, which are generated as draws from a multivariate Gaussian.

How does this capture our preference for semantic coherence? Word embeddings have been shown to capture lexico-semantic regularities in language: words with similar syntactic and semantic properties are found to be close to each other in the embedding space (Agirre et al., 2009; Mikolov, Yih, and Zweig, 2013). Since Gaussian distributions capture a notion of centrality in space, and semantically related words are localized in space, our Gaussian LDA model encodes a prior preference for semantically coherent topics. Our model further has several advantages. Traditional LDA assumes a fixed vocabulary of word types. This modeling assumption drawback as it cannot handle *out of vocabulary* (OOV) words in "held out" documents. Zhai and J. L. Boyd-Graber (2013) proposed an approach to address this problem by drawing topics from a Dirichlet Process with a base distribution over all possible character strings (*i.e.* words). While this model can in principle handle unseen words, the only bias toward being included in a particular topic comes from the topic assignments in the rest of the document. Our model can exploit the contiguity of semantically similar words in the embedding space and can assign high topic probability to a word which is similar to an existing topical word even if it has never been seen before.

Our main contributions are as follows: We propose a new technique for topic modeling by treating the document as a collection of word embeddings and topics itself as multivariate Gaussian distributions in the embedding space (Section 5.5.1). We explore several strategies for collapsed Gibbs sampling and derive scalable algorithms, achieving asymptotic speed-up over the naïve implementation (Section 5.5.2). We qualitatively show that our topics make intuitive sense and quantitatively demonstrate that our model captures a better representation of a document in the topic space by outperforming other models in a classification task (Section 5.5.3).

5.5.1 *Gaussian LDA*

Before going to the details of our model we provide some background on two topics relevant to our work: vector space word embeddings and LDA.

5.5.1.1 *Vector Space Semantics*

According to the **distributional hypothesis** (Harris, 1954), words occurring in similar contexts tend to have similar meaning. This has given rise to data-driven learning of word vectors that capture lexical and semantic properties, which is now a technique of central importance in natural language processing. These word vectors can be used for identifying semantically related word pairs (Agirre et al., 2009; Turney, 2006) or as features in downstream text processing applications (J. Guo et al., 2014; Turian, Ratinov, and Y. Bengio, 2010). Word vectors can either be constructed using low rank approximations of cooccurrence statistics (Deerwester et al., 1990) or using internal representations from neural network models of word sequences (Collobert and Weston, 2008). We use a recently popular and fast tool called `word2vec`[10], to generate skip-gram word embeddings from unlabeled corpus. In this model, a word is used as an input to a log-linear classifier with continuous projection layer and words within a certain window before and after the words are predicted.

10 https://code.google.com/p/word2vec/

5.5.1.2 *Latent Dirichlet Allocation (LDA)*

LDA (D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003) is a probabilistic topic model of corpora of documents which seeks to represent the underlying thematic structure of the document collection. They have emerged as a powerful new technique of finding useful structure in an unstructured collection as it learns distributions over words. The high probability words in each distribution gives us a way of understanding the contents of the corpus at a very high level. In LDA, each document of the corpus is assumed to have a distribution over K topics, where the discrete topic distributions are drawn from a symmetric Dirichlet distribution. The generative process is as follows.

1. for $k = 1$ to K
    a) Choose topic $\boldsymbol{\beta}_k \sim \mathsf{Dir}(\boldsymbol{\eta})$
2. for each document d in corpus D
    a) Choose a topic distribution $\boldsymbol{\theta}_d \sim \mathsf{Dir}(\boldsymbol{\alpha})$
    b) for each word index n from 1 to N_d
        i. Choose a topic $z_n \sim \mathsf{Categorical}(\boldsymbol{\theta}_d)$
        ii. Choose word $w_n \sim \mathsf{Categorical}(\boldsymbol{\beta}_{z_n})$

As it follows from the definition above, a topic is a discrete distribution over a fixed vocabulary of word types. This modeling assumption precludes new words to be added to topics. However modeling topics as a continuous distribution over word embeddings gives us a way to address this problem. In the next section we describe Gaussian LDA, a straightforward extension of LDA that replaces categorical distributions over word types with multivariate Gaussian distributions over the word embedding space.

5.5.1.3 *Our Model*

As with multinomial LDA, we are interested in modeling a collection of documents. However, we assume that rather than consisting of sequences of word types, documents consist of sequences of word embeddings. We write $\mathbf{v}(w) \in \mathbb{R}^M$ as the embedding of word of type w or $\mathbf{v}_{d,i}$ when we are indexing a vector in a document d at position i.

Since our observations are no longer discrete values but continuous vectors in an M-dimensional space, we characterize each topic k as a multivariate Gaussian distribution with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. The choice of a Gaussian parameterization is justified by both analytic convenience and observations that Euclidean distances between embeddings correlate with semantic similarity (Collobert and Weston, 2008; Hermann and Blunsom, 2014; Turney and Pantel, 2010). We place conjugate priors on these values: a Gaussian centered at zero for the mean and an inverse Wishart distribution for the covariance. As before, each document is seen as a mixture of topics whose proportions are drawn from a symmetric Dirichlet prior. The generative process can thus be summarized as follows:

1. for $k = 1$ to K
    a) Draw topic covariance $\boldsymbol{\Sigma}_k \sim \mathcal{W}^{-1}(\boldsymbol{\Psi}, \nu)$
    b) Draw topic mean $\boldsymbol{\mu}_k \sim \mathcal{N}(\boldsymbol{\mu}, \frac{1}{\kappa}\boldsymbol{\Sigma}_k)$
2. for each document d in corpus D
    a) Draw topic distribution $\boldsymbol{\theta}_d \sim \mathsf{Dir}(\boldsymbol{\alpha})$

b) for each word index n from 1 to N_d

i. Draw a topic $z_n \sim \text{Categorical}(\theta_d)$

ii. Draw $\mathbf{v}_{d,n} \sim \mathcal{N}(\boldsymbol{\mu}_{z_n}, \boldsymbol{\Sigma}_{z_n})$

Similar model has previously been proposed for obtaining indexing representations for audio retrieval (P. Hu et al., 2012). They use variational/EM method for posterior inference. Although we don't do any experiment to compare the running time of both approaches, the per-iteration computational complexity is same for both inference methods. We propose a faster inference technique using Cholesky decomposition of covariance matrices which can be applied to both the Gibbs and variational/EM method. However we are not aware of any straightforward way of applying the aliasing trick proposed by (A. Q. Li et al., 2014) on the variational/EM method which gave us huge improvement on running time (see Figure 5.7). Another work which combines embedding with topic models is by Wan, L. Zhu, and Fergus (2012) where they jointly learn the parameters of a neural network and a topic model to capture the topic distribution of low dimensional representation of images.

5.5.2 *Posterior Inference*

In our application, we observe documents consisting of word vectors and wish to infer the posterior distribution over the topic parameters, proportions, and the topic assignments of individual words. Since there is no analytic form of the posterior, approximations are required. Because of our choice of conjugate priors for topic parameters and proportions, these variables can be analytically integrated out, and we can derive a collapsed Gibbs sampler that resamples topic assignments to individual word vectors, similar to the collapsed sampling scheme proposed by (T.L. Griffiths and Steyvers, 2004).

The conditional distribution we need for sampling is:

$$p(z_{d,i} = k \mid \mathbf{z}_{-(d,i)}, \mathbf{V}_d, \zeta, \boldsymbol{\alpha}) \propto (n_{k,d} + \alpha_k) \times t_{\nu_k - M+1}\left(\mathbf{v}_{d,i} \,\middle|\, \mu_k, \frac{\kappa_k + 1}{\kappa_k}\boldsymbol{\Sigma}_k\right) \tag{5.9}$$

Here, $\mathbf{z}_{-(d,i)}$ represents the topic assignments of all word embeddings, excluding the one at i^{th} position of document d; $\mathbf{V}_d$ is the sequence of vectors for document d; $t_{\nu'}(\mathbf{x} \mid \boldsymbol{\mu}', \boldsymbol{\Sigma}')$ is the multivariate t - distribution with ν' degrees of freedom and parameters $\boldsymbol{\mu}'$ and $\boldsymbol{\Sigma}'$. The tuple $\zeta = (\boldsymbol{\mu}, \kappa, \boldsymbol{\Sigma}, \nu)$ represents the parameters of the prior distribution.

It should be noted that the first part of the equation which expresses the probability of topic k in document d is the same as that of LDA. This is because the portion of the model which generates a topic for each word (vector) from its document topic distribution is still the same. The second part of the equation which expresses the probability of assignment of topic k to the word vector $\mathbf{v}_{d,i}$ given the current topic assignments (aka posterior predictive) is given by a multivariate t distribution with parameters $(\boldsymbol{\mu}_k, \kappa_k, \boldsymbol{\Sigma}_k, \nu_k)$. The parameters of the posterior predictive distribution are given as (Murphy, 2012):

$$\begin{aligned}
\kappa_k &= \kappa + N_k \qquad \boldsymbol{\mu}_k = \frac{\kappa\boldsymbol{\mu} + N_k\bar{\mathbf{v}}_k}{\kappa_k} \\
\nu_k &= \nu + N_k \qquad \boldsymbol{\Sigma}_k = \frac{\boldsymbol{\Psi}_k}{(\nu_k - M + 1)} \\
\boldsymbol{\Psi}_k &= \boldsymbol{\Psi} + \mathbf{C}_k + \frac{\kappa N_k}{\kappa_k}(\bar{\mathbf{v}}_k - \boldsymbol{\mu})(\bar{\mathbf{v}}_k - \boldsymbol{\mu})^\top
\end{aligned} \tag{5.10}$$

where $\bar{\mathbf{v}}_k$ and $\mathbf{C}_k$ are given by,

$$\bar{\mathbf{v}}_k = \frac{\sum_d \sum_{i:z_{d,i}=k} (\mathbf{v}_{d,i})}{N_k}$$
$$\mathbf{C}_k = \sum_d \sum_{i:z_{d,i}=k} (\mathbf{v}_{d,i} - \bar{\mathbf{v}}_k)(\mathbf{v}_{d,i} - \bar{\mathbf{v}}_k)^\top$$

Here $\bar{\mathbf{v}}_k$ is the sample mean and $\mathbf{C}_k$ is the scaled form of sample covariance of the vectors with topic assignment k. N_k represents the count of words assigned to topic k across all documents. Intuitively the parameters $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ represents the posterior mean and covariance of the topic distribution and κ_k, ν_k represents the strength of the prior for mean and covariance respectively.

5.5.2.1 *Analysis of running time complexity*

As can be seen from (5.9), for computation of the posterior predictive we need to evaluate the determinant and inverse of the posterior covariance matrix. Direct naïve computation of these terms require $O(M^3)$ operations. Moreover, during sampling as words get assigned to different topics, the parameters $(\boldsymbol{\mu}_k, \kappa_k, \boldsymbol{\Psi}_k, \nu_k)$ associated with a topic changes and hence we have to recompute the determinant and inverse matrix. Since these step has to be recomputed several times (as many times as number of words times number of topics in one Gibbs sweep, in the worst case), it is critical to make the process as efficient as possible. We speed up this process by employing a combination of modern computational techniques and mathematical (linear algebra) tricks, as described in the following subsections.

5.5.2.2 *Faster sampling using Cholesky decomposition of covariance matrix*

Having another look at the posterior equation for $\boldsymbol{\Psi}_k$, we can re-write the equation as:

$$\begin{aligned}\boldsymbol{\Psi}_k &= \boldsymbol{\Psi} + \mathbf{C}_k + \frac{\kappa N_k}{\kappa_k}(\bar{\mathbf{v}}_k - \boldsymbol{\mu})(\bar{\mathbf{v}}_k - \boldsymbol{\mu})^\top \\ &= \boldsymbol{\Psi} + \sum_d \sum_{i:z_{d,i}=k} \mathbf{v}_{d,i}\mathbf{v}_{d,i}^\top - \kappa_k \boldsymbol{\mu}_k \boldsymbol{\mu}_k^\top \\ &\quad + \kappa \boldsymbol{\mu}\boldsymbol{\mu}^\top.\end{aligned} \tag{5.11}$$

During sampling when we are computing the assignment probability of topic k to $\mathbf{v}_{d,i}$, we need to calculate the updated parameters of the topic. Using (5.11) it can be shown that $\boldsymbol{\Psi}_k$ can be updated from current value of $\boldsymbol{\Psi}_k$, after updating $\kappa_k.\nu_k$ and μ_k, as follows:

$$\boldsymbol{\Psi}_k \leftarrow \boldsymbol{\Psi}_k + \frac{\kappa_k}{\kappa_k - 1}\left(\boldsymbol{\mu}_k - \mathbf{v}_{d,i}\right)\left(\boldsymbol{\mu}_k - \mathbf{v}_{d,i}\right)^\top. \tag{5.12}$$

This equation has the form of a rank 1 update, hinting towards use of Cholesky decomposition. If we have the Cholesky decomposition of $\boldsymbol{\Psi}_k$ computed, then we have tools to update $\boldsymbol{\Psi}_k$ cheaply. Since $\boldsymbol{\Psi}_k$ and $\boldsymbol{\Sigma}_k$ are off by only a scalar factor, we can equivalently talk about $\boldsymbol{\Sigma}_k$. Equation (5.12) can also be understood in the following way. During sampling, when a word embedding $\mathbf{v}_{d,i}$ gets a new assignment to a topic, say k, then the new value of the topic covariance can be computed from the current one using just a rank 1 update.[11] We next describe how to exploit the Cholesky decomposition representation to speed up computations.

For sake of completeness, any symmetric $M \times M$ real matrix $\boldsymbol{\Sigma}_k$ is said to be positive definite if $\forall \mathbf{z} \in \mathbb{R}^M : \mathbf{z}^\top \boldsymbol{\Sigma}_k \mathbf{z} > 0$. The Cholesky decomposition of such a symmetric positive definite

11 Similarly the covariance of the old topic assignment of the word w can be computed using a rank 1 downdate

matrix $\mathbf{\Sigma}_k$ is nothing but its decomposition into the product of some lower triangular matrix $\mathbf{L}$ and its transpose, i.e.

$$\mathbf{\Sigma}_k = \mathbf{L}\mathbf{L}^\top. \tag{5.13}$$

Finding this factorization also take cubic operation. However given Cholesky decomposition of $\mathbf{\Sigma}_k$, after a rank 1 update (or downdate), i.e. the operation:

$$\mathbf{\Sigma}_k \leftarrow \mathbf{\Sigma}_k + \mathbf{z}\mathbf{z}^\top \tag{5.14}$$

we can find the factorization of new $\mathbf{\Sigma}_k$ in just quadratic time (Stewart, 1998). We will use this trick to speed up the computations[12]. Basically, instead of computing determinant and inverse again in cubic time, we will use such rank 1 update (downdate) to find new determinant and inverse in an efficient manner as explained in details below.

To compute the density of the posterior predictive t−distibution, we need to compute the determinant $|\mathbf{\Sigma}_k|$ and the term of the form $(\mathbf{v}_{d,i} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}_k^{-1}(\mathbf{v}_{d,i} - \boldsymbol{\mu}_k)$. The Cholesky decomposition of the covariance matrix can be used for efficient computation of these expression as shown below.

COMPUTATION OF DETERMINANT The determinant of $\mathbf{\Sigma}_k$ can be computed from from its Cholesky decomposition $\mathbf{L}$ as:

$$\log(|\mathbf{\Sigma}_k|) = 2 \times \sum_{i=1}^{M} \log\left(\mathbf{L}_{i,i}\right). \tag{5.15}$$

This takes linear time in the order of dimension and is clearly a significant gain from cubic time complexity.

COMPUTATION OF $(\mathbf{v}_{d,i} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}_k^{-1}(\mathbf{v}_{d,i} - \boldsymbol{\mu})$ Let $\mathbf{b} = (\mathbf{v}_{d,i} - \boldsymbol{\mu}_k)$ and cholesky decomposition of $\Sigma = LL^T$ as before. Then $\mathbf{b}^\top \mathbf{\Sigma}^{-1}\mathbf{b}$ can be written as

$$\begin{aligned}\mathbf{b}^\top \mathbf{\Sigma}^{-1}\mathbf{b} &= \mathbf{b}^\top (\mathbf{L}\mathbf{L}^\top)^{-1}\mathbf{b} \\ &= \mathbf{b}^T (\mathbf{L}^{-1})^\top \mathbf{L}^{-1}\mathbf{b} \\ &= (\mathbf{L}^{-1}\mathbf{b})^\top (\mathbf{L}^{-1}\mathbf{b})\end{aligned} \tag{5.16}$$

Now $(\mathbf{L}^{-1}\mathbf{b})$ is the solution of the equation $\mathbf{L}\mathbf{x} = \mathbf{b}$. Also since $\mathbf{L}$ is a lower triangular matrix, this equation can be solved easily using forward substitution. Lastly we will have to take an inner product of $\mathbf{x}$ and $\mathbf{x}^\top$ to get the value of $(\mathbf{v}_{d,i} - \boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{v}_{d,i} - \boldsymbol{\mu}_k)$. This step again takes quadratic time and is again a savings from the cubic time complexity.

5.5.2.3 *Derivation of Rank 1 update of the covariance matrix*

Due to the conjugate properties of Normal and Normal Inverse Wishart distribution, the parameters of the posterior predictive distribution are given as:

$$\begin{aligned}\kappa_k &= \kappa + N_k \qquad \boldsymbol{\mu}_k = \frac{\kappa\boldsymbol{\mu} + N_k\bar{\mathbf{v}}_k}{\kappa_k} \\ \nu_k &= \nu + N_k \qquad \mathbf{\Sigma}_k = \frac{\mathbf{\Psi}_k}{(\nu_k - M + 1)} \\ \mathbf{\Psi}_k &= \mathbf{\Psi} + \mathbf{C}_k + \frac{\kappa N_k}{\kappa_k}(\bar{\mathbf{v}}_k - \boldsymbol{\mu})(\bar{\mathbf{v}}_k - \boldsymbol{\mu})^\top\end{aligned} \tag{5.17}$$

12 For our experiments, we set the prior covariance to be 3*I, which is a positive definite matrix.

where $\bar{\mathbf{v}}_k$ and $\mathbf{C}_k$ are given by,

$$\begin{aligned}\bar{\mathbf{v}}_k &= \frac{\sum_d \sum_{i:z_{d,i}=k}(\mathbf{v}_{d,i})}{N_k} \\ \mathbf{C}_k &= \sum_d \sum_{i:z_{d,i}=k}(\mathbf{v}_{d,i}-\bar{\mathbf{v}}_k)(\mathbf{v}_{d,i}-\bar{\mathbf{v}}_k)^\top \\ &= \sum_{i:z_{*,i}=k}(\mathbf{v}_i-\bar{\mathbf{v}}_k)(\mathbf{v}_i-\bar{\mathbf{v}}_k)^\top\end{aligned}$$

In the last step we replace two summations by one.

Intuitively $\bar{\mathbf{v}}_k$ is the sample mean and $\mathbf{C}_k$ is the scaled form of sample covariance of the vectors with topic assignment k. N_k represents the count of words assigned to topic k across all documents.

First lets look at the update for the mean

$$\boldsymbol{\mu}_k = \frac{\kappa\boldsymbol{\mu}+N_k\bar{\mathbf{v}}_k}{\kappa_k} \tag{5.18}$$

One way of looking at equation (5.18): $\boldsymbol{\mu}_k$ is the mean of topic k which has presently N_k words assigned to it. (Equivalently table k has N_k customers (words) sitting on it). When a new word is assigned to (or an existing word is removed from) a topic, we have to update the mean $\boldsymbol{\mu}_k$ of the table based on the update equation (5.18). Let us denote the mean of the topic k which presently has N words assigned to it as $\boldsymbol{\mu}_k^N$. Then from equation (5.18), it can be seen that $\boldsymbol{\mu}_k^{N_k}$ can be recursively written interms of $\boldsymbol{\mu}_k^{N_k-1}$ as

$$\boldsymbol{\mu}_k^{N_k} = \frac{(\kappa+N_k-1)\boldsymbol{\mu}_k^{N_k-1}+\mathbf{v}_{d,i}}{\kappa+N_k} \tag{5.19}$$

Here $\mathbf{v}_{d,i}$ denotes the word embedding at position i of document d which was assigned topic k during sampling and which made the mean of the topic to change from $\boldsymbol{\mu}_k^{N_k-1}$ to $\boldsymbol{\mu}_k^{N_k}$. The reader should note that this could be any given word during sampling.

Now lets look at the update for $\boldsymbol{\Psi}_k$.

$$\begin{aligned}\boldsymbol{\Psi}_k &= \boldsymbol{\Psi}+\mathbf{C}_k+\frac{\kappa N_k}{\kappa_k}(\bar{\mathbf{v}}_k-\boldsymbol{\mu})(\bar{\mathbf{v}}_k-\boldsymbol{\mu})^\top \\ &= \boldsymbol{\Psi}+\sum_{i:z_{*,i}=k}(\mathbf{v}_i-\bar{\mathbf{v}}_k)(\mathbf{v}_i-\bar{\mathbf{v}}_k)^\top+\frac{\kappa N_k}{\kappa_k}(\bar{\mathbf{v}}_k-\boldsymbol{\mu})(\bar{\mathbf{v}}_k-\boldsymbol{\mu})^\top \\ &= \boldsymbol{\Psi}+\sum_{i:z_{*,i}=k}\mathbf{v}_i\mathbf{v}_i^\top-\sum_{i:z_{*,i}=k}\mathbf{v}_i\bar{\mathbf{v}}_k^\top-\bar{\mathbf{v}}_k\sum_{i:z_{*,i}=k}\mathbf{v}_i^\top+N_k\bar{\mathbf{v}}_k\bar{\mathbf{v}}_k^\top+\frac{\kappa N_k}{\kappa_k}(\bar{\mathbf{v}}_k-\boldsymbol{\mu})(\bar{\mathbf{v}}_k-\boldsymbol{\mu})^\top \\ &= \boldsymbol{\Psi}+\sum_{i:z_{*,i}=k}\mathbf{v}_i\mathbf{v}_i^\top-N_k\bar{\mathbf{v}}_k\bar{\mathbf{v}}_k^\top+\frac{\kappa N_k}{\kappa_k}(\bar{\mathbf{v}}_k\bar{\mathbf{v}}_k^\top-\boldsymbol{\mu}\bar{\mathbf{v}}_k^\top-\bar{\mathbf{v}}_k\boldsymbol{\mu}^\top+\boldsymbol{\mu}\boldsymbol{\mu}^\top) \\ &= \boldsymbol{\Psi}+\sum_{i:z_{*,i}=k}\mathbf{v}_i\mathbf{v}_i^\top-\frac{(\kappa\boldsymbol{\mu}+N_k\bar{\mathbf{v}}_k)(\kappa\boldsymbol{\mu}+N_k\bar{\mathbf{v}}_k)^\top}{\kappa+N_k}+\kappa\boldsymbol{\mu}\boldsymbol{\mu}^\top\end{aligned}$$

(5.20)

$$= \Psi + \sum_{i:z_{*,i}=k} \mathbf{v}_i \mathbf{v}_i^\top - (\kappa + N_k)\boldsymbol{\mu}_k \boldsymbol{\mu}_k{}^\top + \kappa \boldsymbol{\mu}\boldsymbol{\mu}^\top \tag{5.21}$$

We get the last step by using equation (5.18). Also following similar nomenclature we will be denoting the scaled covariance of a topic with N words as $\boldsymbol{\Psi}_k^N$.

If we look closely, equation (5.21) also can be recursively written as follows:

$$\boldsymbol{\Psi}_k^{N_k} = \boldsymbol{\Psi}_k^{(N_k-1)} + \mathbf{v}_{d,i}\mathbf{v}_{d,i}^\top - (\kappa + N_k)\boldsymbol{\mu}_k^{N_k}\boldsymbol{\mu}_k^{N_k}{}^\top + (\kappa + N_k - 1)\boldsymbol{\mu}_k^{(N_k-1)}\boldsymbol{\mu}_k^{(N_k-1)\top}$$

Now writing $\boldsymbol{\mu}_k^{(N_k-1)}$ in terms of $\boldsymbol{\mu}_k^{(N_k)}$ using equation (5.19),

$$\boldsymbol{\Psi}_k^{N_k} = \boldsymbol{\Psi}_k^{(N_k-1)} + \frac{\kappa + N_k}{(\kappa + N_k - 1)}(\boldsymbol{\mu}_k^{(N_k)} - \mathbf{v}_{d,i})(\boldsymbol{\mu}_k^{(N_k)} - \mathbf{v}_{d,i})^\top$$

This is the same as equation (5.11) and is in the form of a Rank 1 update. Therefore, if we have the Cholesky decomposition for $\boldsymbol{\Psi}_k^{(N_k-1)}$, then we can compute the Cholesky decomposition of $\boldsymbol{\Psi}_k^{(N_k)}$ in quadratic time complexity in the number of dimensions (which is an order of magnitude improvement from the cubic time complexity)

5.5.2.4 *Further reduction of sampling complexity using Alias Sampling*

Although Cholesky trick helps us to reduce the sampling complexity of a embedding to $O(KM^2)$, it can still be impractical.In Gaussian LDA, the Gibbs sampling equation (5.9) can be split into two terms. The first term $n_{k,d} \times t_{\nu_k - M+1}\left(\mathbf{v}_{d,i} \middle| \boldsymbol{\mu}_k, \frac{\kappa_k+1}{\kappa_k}\boldsymbol{\Sigma}_k\right)$ denotes the document contribution and the second term $\alpha_k \times t_{\nu_k - M+1}\left(\mathbf{v}_{d,i} \middle| \boldsymbol{\mu}_k, \frac{\kappa_k+1}{\kappa_k}\boldsymbol{\Sigma}_k\right)$ denotes the language model contribution. Empirically one can make two observations about these terms. First, $n_{k,d}$ is often a sparse vector, as a document most likely contains only a few of the topics. Secondly, topic parameters (μ_k, Σ_k) captures global phenomenon, and rather change relatively slowly over the iterations. We can exploit these findings to avoid the naive approach to draw a sample from:

$$\begin{aligned} p(z_{d,i} = k \mid \mathbf{z}_{-(d,i)}, \mathbf{V}_d, \boldsymbol{\zeta}, \boldsymbol{\alpha}) \propto & \underbrace{(n_{k,d} \times t_{\nu_k - M+1}\left(\mathbf{v}_{d,i} \middle| \boldsymbol{\mu}_k, \frac{\kappa_k+1}{\kappa_k}\boldsymbol{\Sigma}_k\right)}_{\text{doc-specific}} \\ & \underbrace{+ \alpha \times t_{\nu_k - M+1}\left(\mathbf{v}_{d,i} \middle| \boldsymbol{\mu}_k, \frac{\kappa_k+1}{\kappa_k}\boldsymbol{\Sigma}_k\right)}_{\text{language-model}} \end{aligned} \tag{5.22}$$

In particular, we compute the document-specific sparse term exactly and for the remainder language model term we borrow idea from A. Q. Li et al. (2014). We use a slightly stale distribution for the language model. Then Metropolis Hastings (MH) algorithm allows us to convert the stale sample into a fresh one, provided that we compute ratios between successive states correctly. It is sufficient to run MH for a few number of steps because the stale distribution

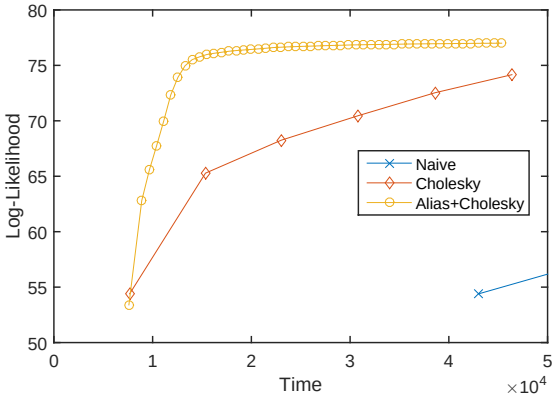


Figure 5.7: Plot comparing average log-likelihood vs time (in sec) achieved after applying each trick on the NIPS dataset. The shapes on each curve denote end of each iteration.

acting as the proposal is very similar to the target. This is because, as pointed out earlier, the language model term does not change too drastically whenever we resample a single word. The number of words is huge, hence the amount of change per word is concomitantly small. (Only if one could convert stale bread into fresh one, it would solve world's food problem!)

The exercise of using stale distribution and MH steps is advantageous because sampling from it can be carried out in $O(1)$ amortized time, thanks to alias sampling technique (Vose, 1991). Moreover, the task of building the alias tables can be outsourced to other cores.

With the combination of both Cholesky and Alias tricks, the sampling complexity can thus be brought down to $O(K_d M^2)$ where K_d represents the number of actually instantiated topics in the document and $K_d \ll K$. In particular, we plot the sampling rate achieved naively, with Cholesky (CH) trick and with Cholesky+Alias (A+CH) trick in Figure 5.7 demonstrating better likelihood at much less time. Also after initial few iterations, the time per iteration of A+CH trick is 9.93 times less than CH and 53.1 times less than naive method. This is because initially we start with random initialization of words to topics, but after few iterations the $n_{k,d}$ vector starts to become sparse.

5.5.3 *Experiments*

In this section we evaluate our Word Vector Topic Model on various experimental tasks. Specifically we wish to determine:

- Is our model is able to find coherent and meaningful topics?
- Is our model able to infer the topic distribution of a held-out document even when the document contains words which were previously unseen?

We run our experiments on two datasets 20-NEWSGROUP[13] and NIPS[14]. All the datasets were tokenized and lowercased with `cdec` (Dyer et al., 2010).

13 A collection of newsgroup documents partitioned into 20 news groups. After pre-processing we had 18768 documents. We randomly selected 2000 documents as our test set. This dataset is publicly available at http://qwone.com/~jason/20Newsgroups/

14 A collection of 1740 papers from the proceedings of Neural Information Processing System. The dataset is available at http://www.cs.nyu.edu/~roweis/data.html

Gaussian LDA topics								
hostile	play	government	people	university	hardware	scott	market	gun
murder	round	state	god	program	interface	stevens	buying	rocket
violence	win	group	jews	public	mode	graham	sector	military
victim	players	initiative	israel	law	devices	walker	purchases	force
testifying	games	board	christians	institute	rendering	tom	payments	machine
provoking	goal	legal	christian	high	renderer	russell	purchase	attack
legal	challenge	bill	great	research	user	baker	company	operation
citizens	final	general	jesus	college	computers	barry	owners	enemy
conflict	playing	policy	muslims	center	monitor	adams	paying	fire
victims	hitting	favor	religion	study	static	jones	corporate	flying
rape	match	office	armenian	reading	encryption	joe	limited	defense
laws	ball	political	armenians	technology	emulation	palmer	loans	warning
violent	advance	commission	church	programs	reverse	cooper	credit	soldiers
trial	participants	private	muslim	level	device	robinson	financing	guns
intervention	scores	federal	bible	press	target	smith	fees	operations
0.8302	0.9302	0.4943	2.0306	0.5216	2.3615	2.7660	1.4999	1.1847
Multinomial LDA topics								
turkish	year	people	god	university	window	space	ken	gun
armenian	writes	president	jesus	information	image	nasa	stuff	people
people	game	mr	people	national	color	gov	serve	law
armenians	good	don	bible	research	file	earth	line	guns
armenia	team	money	christian	center	windows	launch	attempt	don
turks	article	government	church	april	program	writes	den	state
turkey	baseball	stephanopoulos	christ	san	display	orbit	due	crime
don	don	time	christians	number	jpeg	moon	peaceful	weapons
greek	games	make	life	year	problem	satellite	article	firearms
soviet	season	clinton	time	conference	screen	article	served	police
time	runs	work	don	washington	bit	shuttle	warrant	control
genocide	players	tax	faith	california	files	lunar	lotsa	writes
government	hit	years	good	page	graphics	henry	occurred	rights
told	time	ll	man	state	gif	data	writes	article
killed	apr	ve	law	states	writes	flight	process	laws
0.3394	0.2036	0.1578	0.7561	0.0039	1.3767	1.5747	-0.0721	0.2443

Table 5.5: Top words of some topics from Gaussian-LDA and multinomial LDA on 20-newsgroups for $K = 50$. Words in Gaussian LDA are ranked based on density assigned to them by the posterior predictive distribution. The last row for each method indicates the PMI score (w.r.t. Wikipedia co-occurence) of the topic's fifteen highest ranked words.

5.5.3.1 *Topic Coherence*

QUANTITATIVE ANALYSIS Typically topic models are evaluated based on the likelihood of held-out documents. But in this case, it is not correct to compare perplexities with models which do topic modeling on words. Since our topics are continuous distributions, the probability of a word vector is given by its density w.r.t. the normal distribution based on its topic assignment, instead of a probability mass from a discrete topic distribution. Moreover, J. Chang et al. (2009) showed that higher likelihood of held-out documents doesn't necessarily correspond to human perception of topic coherence. Instead to measure topic coherence we follow Newman, Karimi, and Cavedon (2009) to compute the Pointwise Mutual Information (PMI) of topic words w.r.t. wikipedia articles. We extract the document co-occurrence statistics of topic

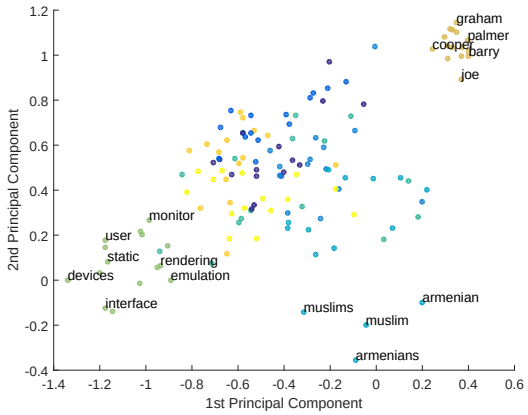


Figure 5.8: The first two principal components for the word embeddings of the top words of topics shown in Table 5.5 have been visualized. Each blob represents a word color coded according to its topic in the Table 5.5.

words from Wikipedia and compute the score of a topic by averaging the score of the top 15 words of the topic. A higher PMI score implies a more coherent topic as it means the topic words usually co-occur in the same document. In the last line of Table 5.5, we present the PMI score for some of the topics for both Gaussian LDA and traditional multinomial LDA. It can be seen that Gaussian LDA is a clear winner, achieving an average 275% higher score on average.

However, we are using embeddings trained on Wikipedia corpus itself, and the PMI measure is computed from co-occurrence in the Wikipedia corpus. As a result, our model is definitely biased towards producing higher PMI. Nevertheless Wikipedia PMI is a believed to be a good measure of semantic coherence.

QUALITATIVE ANALYSIS Table 5.5 shows some top words from topics from Gaussian-LDA and LDA on the 20-news dataset for $K = 50$. The words in Gaussian-LDA are ranked based on their density assigned to them by the posterior predictive distribution in the final sample. As shown, Gaussian LDA is able to capture several intuitive topics in the corpus such as sports, government, 'religion', 'universities', 'tech', 'finance' etc. One interesting topic discovered by our model (on both 20-news and NIPS dataset) is the collection of human names, which was not captured by classic LDA. While one might imagine that names associated with particular topics might be preferable to a 'names-in-general' topic, this ultimately is a matter of user preference. More substantively, classic LDA failed to identify the 'finance' topics. We also noticed that there were certain words ('don', 'writes', etc) which often came as a top word in many topics in classic LDA. However our model was not able to capture the 'space' topics which LDA was able to identify.

Also we visualize a part of the continuous space where the word embedding is performed. For this task we performed the Principal Component Analysis (PCA) over all the word vectors and plot the first two components as shown in Figure 5.8. We can see clear separations between some of the clusters of topics as depicted. The other topics would be separated in other dimensions.

Model	Accuracy	
	PPDB	WordNet
infvoc	28.00%	19.30%
G-LDA (*fix*)	44.51%	43.53%
G-LDA (*1*)	44.66%	43.47%
G-LDA (*100*)	43.63%	43.11%
G-LDA (*1932*)	44.72%	42.90%

Table 5.6: Accuracy of our model and *infvoc* on the synthetic datasets. In Gaussian LDA *fix*, the topic distributions learnt during training were fixed; G-LDA(*1, 100, 1932*) is the online implementation of our model where the documents comes in minibatches. The number in parenthesis denote the size of the batch. The full size of the test corpus is 1932.

5.5.3.2 *Performance on document containing new words*

In this experiment we evaluate the performance of our model on documents which contains previously unseen words. It should be noted that traditional topic modeling algorithms will typically ignore such words while inferring the topic distribution and hence might miss out important words. The continuous topic distributions of the Word Vector Topic Model on the other hand, will be able to assign topics to an unseen word, if we have the vector representation of the word. Given the recent development of fast and scalable methods of estimating word embeddings, it is possible to train them on huge text corpora and hence it makes our model a viable alternative for topic inference on documents with new words.

EXPERIMENTAL SETUP Since we want to capture the strength of our model on documents containing unseen words, we select a subset of documents and replace words of those documents by its synonyms if they haven't occurred in the corpus before. We obtain the synonym of a word using two existing resources and hence we create two such datasets. For the first set, we use the Paraphrase Database (Ganitkevitch, Van Durme, and Callison-Burch, 2013) to get the lexical paraphrase of a word. The paraphrase database[15] is a semantic lexicon containing around 169 million paraphrase pairs of which 7.6 million are lexical (one word to one word) paraphrases. The dataset comes in varying size ranges starting from S to XXXL in increasing order of size and decreasing order of paraphrasing confidence. For our experiments we selected the L size of the paraphrase database.

The second set was obtained using WordNet (G. A. Miller, 1995), a large human annotated lexicon for English that groups words into sets of synonyms called synsets. To obtain the synonym of a word, we first label the words with their part-of-speech using the Stanford POS tagger (Toutanova, Klein, et al., 2003). Then we use the WordNet database to get the synonym from its sysnset.[16] We select the first synonym from the synset which hasn't occurred in the corpus before. On the 20-news dataset (vocab size = 18,179 words, test corpus size = 188,694 words), a total of 21,919 words (2,741 distinct words) were replaced by synonyms from PPDB and 38,687 words (2,037 distinct words) were replaced by synonyms from WordNet.

EVALUATION BENCHMARK As mentioned before traditional topic model algorithms cannot handle OOV words. So comparing the performance of our document with those models would be unfair. Recently (Zhai and J. L. Boyd-Graber, 2013) proposed an extension of LDA (*infvoc*)

15 http://www.cis.upenn.edu/~ccb/ppdb/
16 We use the `JWI` toolkit (Finlayson, 2014)

Model	PPDB (Mean Deviation)		
	L_1	L_2	L_∞
infvoc	94.95	7.98	1.72
G-LDA (*fix*)	15.13	1.81	0.66
G-LDA (*1*)	15.71	1.90	0.66
G-LDA (*10*)	15.76	1.97	0.66
G-LDA (*174*)	14.58	1.66	0.66

Table 5.7: This table shows the Average L_1 Deviation, Average L_2 Deviation, Average L_∞ Deviation for the difference of the topic distribution of the actual document and the synthetic document on the NIPS corpus. Compared to *infvoc*, G-LDA achieves a lower deviation of topic distribution inferred on the synthetic documents with respect to actual document. The full size of the test corpus is 174.

which can incorporate new words. They have shown better performances in a document classification task which uses the topic distribution of a document as features on the 20-news group dataset as compared to other fixed vocabulary algorithms. Even though, the *infvoc* model can handle OOV words, it will most likely not assign high probability to a new topical word when it encounters it for the first time since it is directly proportional to the number of times the word has been observed On the other hand, our model could assign high probability to the word if its corresponding embedding gets a high probability from one of the topic Gaussians. With the experimental setup mentioned before, we want to evaluate performance of this property of our model. Using the topic distribution of a document as features, we try to classify the document into one of the 20 news groups it belongs to. If the document topic distribution is modeled well, then our model should be able to do a better job in the classification task.

To infer the topic distribution of a document we follow the usual strategy of fixing the learnt topics during the training phase and then running Gibbs sampling on the test set (G-LDA (*fix*) in Table 5.6). However *infvoc* is an online algorithm, so it would be unfair to compare our model which observes the entire set of documents during test time. Therefore we implement the online version of our algorithm using Gibbs sampling following L. Yao, Mimno, and Andrew McCallum (2009). We input the test documents in batches and do inference on those batches independently also sampling for the topic parameter, along the lines of *infvoc*. The batch size for our experiments are mentioned in parentheses in Table 5.6. We classify using the multi class logistic regression classifier available in `Weka` (M. Hall et al., 2009).

It is clear from Table 5.6 that we outperform *infvoc* in all settings of our experiments. This implies that even if new documents have significant amount of new words, our model would still do a better job in modeling it. We also conduct an experiment to check the actual difference between the topic distribution of the original and synthetic documents. Let h and h' denote the topic vectors of the original and synthetic documents. Table 5.7 shows the average L_1, L_2 and L_∞ norm of $(h - h')$ of the test documents in the NIPS dataset. A low value of these metrics indicates higher similarity. As shown in the table, Gaussian LDA performs better here too.

5.6 POINTS TO PONDER

We looked at specific ways to alter neural network design so as to enforce invariances and dependences. However, many questions still remain. We discuss them application-wise, but a common question among the different applications is how can we automatically design networks obeying the invariances or dependencies.

PREDICTING MOLECULAR PROPERTIES A method based on DeepSets to predict DFT computed properties of molecules only from their raw 3D geometrical coordinates was developed. The model architecture was able to handle molecules containing different number of atoms, and in contrast to available models in literature, it did not require any featurization of the input data with additional meta-information. Prediction errors from the proposed model on several target properties were found to be competitive to the state-of-the-art convolutional graphs, which are heavily featurized. As expected, the properties with explicit spatial dependence were found to have better prediction accuracies, as spatial invariances were incorporated in the network. However, robustness to rotation is not clear. Instead of Cartesian coordinates, one might use internal coordinates which are rotation invariant, but then feeding them to neural networks is not clear.

SEARCHING SATELLITE IMAGERY There are a number of burgeoning, visually consistent, and in many cases worrisome phenomena which future versions of Terrapattern could be useful in tracking. These include Concentrated Animal Feeding Operations, or CAFOs; uranium mill tailings deposits; "Gamma Gardens" (also called radiation farms) for mutation breeding; Siberian methane blowholes, which are arising due to global warming; and megafauna poaching. The Terrapattern project is only a prototype—especially in its scale—and we feel we have only scratched the surface of what is possible. The major bottleneck lies how well we can say two image portions are similar. Our approach using resnet trained on labeled satellite data can measure similarity only upto some extent.

TOPIC MODELLING WITH EMBEDDINGS While word embeddings have been incorporated to produce state-of-the-art results in numerous supervised natural language processing tasks from the word level to document level (Q. Le and Mikolov, 2014) and (Turian, Ratinov, and Y. Bengio, 2010, *inter alia*); however, they have played a more minor role in unsupervised learning problems. This work shows some of the promise that they hold in this domain. Our model can be extended in a number of potentially useful, but straightforward ways. First, DPMM models of word emissions would better model the fact that identical vectors will be generated multiple times, and perhaps add flexibility to the topic distributions that can be captured, without sacrificing our preference for topical coherence. More broadly still, running LDA on documents consisting of different modalities than just text is facilitated by using the *lingua franca* of vector space representations, so we expect numerous interesting applications in this area. An interesting extension to our work would be the ability to handle polysemous words based on multi-prototype vector space models (Neelakantan, Shankar, et al., 2014; Reisinger and Raymond J. Mooney, 2010) and we keep this as an avenue for future research.

6 Recurrent Data Analysis

We study the problem of incorporating domain knowledge in sequential data to enhance interepretability. Recurrent neural networks, such as long-short term memory (LSTM) networks, are powerful tools for modeling sequential data like user browsing history (Korpusik, Sakaki, and F. C. Y.-Y. Chen, 2016; Tan, X. Xu, and Y. Liu, 2016) or natural language text (Mikolov, Karafiát, et al., 2010). Despite the strong performance, it has not gained popularity for user-facing applications, mainly owing to a large number of parameters and lack of interpretability. In this chapter, to further understand aforementioned shortcoming of LSTMs and provide solutions we focus on user modelling as user data is representative of challenges in sequential data, namely large symbol space and non-linear dynamics with long range interactions. In context of user modelling, LSTMs require a large number of parameters to generalize across different user types, notwithstanding the simplicity of the underlying dynamics/*structure*, rendering it uninterpretable, which is highly undesirable in user modeling. The increase in complexity and parameters arises due to naïve handling of a large action space, ignoring the *structure* that many of the actions can be clustered to have similar intent or topic. In this chapter, we introduce Latent LSTM Allocation (LLA) for user modeling leveraging the above mentioned *structure* by combining hierarchical Bayesian models with LSTMs. In LLA, each user is modeled as a sequence of actions, and the model jointly groups actions into topics and learns the temporal dynamics over the topic sequence, instead of action space directly. This leads to a model that is highly interpretable, concise, and can capture intricate dynamics. Furthermore, user data sequence data often exhibit multi-modes of behaviors with abrupt dynamic changes, for which we use a threaded LLA (thLLA) to break each sequence into a set of segments and then model the dynamic in each segment using an LSTM mixture. We present an efficient Stochastic EM inference algorithm for our model that scales to millions of users/documents. Our experimental evaluations show that the proposed model compares favorably with several state-of-the-art baselines.

6.1 Introduction

Sequential data prediction is an important problem in machine learning spanning over a diverse set of applications ranging from text (Mikolov, Karafiát, et al., 2010) to finance (Pai and C.-S. Lin, 2005) to user behavior (Köck and Paramythis, 2011). For example, when applied to statistical language modeling, the goal is to predict the next word in textual data given context, very similar to that in user activity modeling where the aim is to predict the next activity of the user given the history. Accurate user activity modeling is very important for serving relevant, personalized, and useful contents to the user. A good model of sequential data should be accurate, sparse, and interpretable. Unfortunately, none of the existing techniques for user or language modeling satisfy all of these requirements.

The state-of-the-art for modeling sequential data is to employ recurrent neural networks (RNN; Lipton, Berkowitz, and Elkan 2015), such as LSTMs (Long-Short Term Memory; Hochreiter and Jürgen Schmidhuber 1997). Such RNNs have been shown to be effective at capturing long and

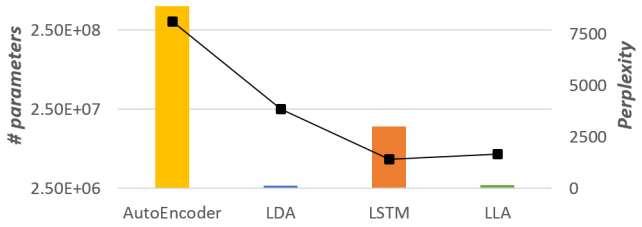


Figure 6.1: LLA has better perplexity (lower is better) than LDA but much fewer parameters than LSTMs, as shown in a language modeling task on Wikipedia.

short patterns in data, e.g. token-level semantic as well as syntactic regularities in language (Jozefowicz et al., 2016). However, the neural network representations are generally uninterpretable and inaccessible to humans (Strobelt et al., 2016). Moreover, the number of parameters of the model is proportional to the number of observed word types or action types, which can grow to tens or hundreds of millions. Note that for user modeling task, character level RNN is not feasible because user actions are often not words but hash indices or URLs.

On the other hand of the spectrum lies latent variable models with multi-task learning, such as LDA (D. Blei, A. Ng, and M. Jordan, 2002) and other topic model variants. These topic models are not strictly sequence models, nevertheless have proved to be powerful tools for uncovering latent structure in both text and user data (Aly et al., 2012) with good commercial success (D. J. Hu, R. Hall, and Attenberg, 2014). Topic models are popular for their ability to organize the data into a smaller set of prominent themes or topics through statistical strength sharing across users or documents. These topic representations are generally accessible to humans and easily lend themselves to being interpreted.

In this chapter, we propose Latent LSTM Allocation (LLA), a model that bridges the gap between the sequential RNN's and the non-sequential LDA. LLA borrows graphical model techniques to infer topics (groups of related word or user activities) by sharing statistical strength across users/documents and recurrent deep networks to model the dynamics of topic evolution inside each sequence (document or user activities) rather than at user action/word level (Section 6.3.1). LLA inherits sparsity and interpretability from LDA, while borrowing accuracy from LSTM. We provide various variants of LLA that trade model size vs. accuracy without sacrificing interpretability (Section 6.3.5). As shown in Figure 6.1, for the task of language modeling on the Wikipedia dataset, LLA achieves comparable accuracy to LSTM while being as sparse as LDA in terms of models size. For exact inference, we develop a particle MCMC method sampler that can draw from the joint posterior directly (Section 6.3.3). However, as the exact inference is computationally very intensive, we provide an approximate but efficient inference algorithm for parameter inference in LLA, with only slight degration in performance (Section 6.3.4). Furthermore, LLA, while expressive and interpretable, can still capture spurious dynamics in sequential data with mixed themes. This is because, while theoretically LSTM has the expressive power to capture long-range dependency structures, in practice vanishing gradients and the introduction of latent variables (topics) exacerbate the complexity of the surface of objective function and precludes the optimization algorithm from linking each item in the sequence to the most relevant item in the the same theme from the user history. To circumvent this issue in data with abrupt change in themes, we combine LLA with switching state space models to break each sequence into a set of segments and then model the dynamic in each segment using an LSTM mixture (Section 6.5). We show quantitatively as well as qualitatively efficacy and *interpretability* of LLA and its variants over several datasets (Section 6.4 and 6.6).

6.2 BACKGROUND

In this section, we provide a brief review of user/language modeling, LSTMs, and sequential Monte Carlo.

6.2.1 *User/Language Modeling*

User activity modeling and language modeling amounts to learning a function that computes the log probability, $\log p(\boldsymbol{w}|\text{model})$, of a user activity or sentence $\boldsymbol{w} = (w_1, \ldots, w_n)$. Subsequently, this function can be used to predict the next set of actions or words. This function can be decomposed and learned in different ways under various assumptions. Imposing a bag of words assumption - as used in LDA - leads to ignoring the sequence information and yields $\sum_{i=1}^{n} \log p(w_i|\text{model})$. Alternatively, one could decompose according to the chain rule into sum of the conditional log probabilities $\sum_{i=1}^{n} \log p(w_i \mid w_1, \ldots, w_{i-1}, \text{model})$, thereby preserving temporal information and use some RNN to model $\log p(w_i \mid w_1, \ldots, w_{i-1}, \text{model})$ (Mikolov, Karafiát, et al., 2010; Sundermeyer, Schlüter, and Ney, 2012).

6.2.2 *Long Short-Term Memories*

Temporal aspect is very important for user activity modeling. LSTM, a type of RNN, is well suited for the task as it can learn from experience to classify, process, and predict time series when there are very long time lags of unknown size between important events. LSTM is designed to cope with the vanishing gradient problem inherent in simple RNNs (Hochreiter and Jürgen Schmidhuber, 1997). This is one of the main reasons why LSTMs outperform simple RNNs, hidden Markov models, and other sequence learning methods in numerous application.

In general, a RNN is a triplet (Σ, S, δ):

- $\Sigma \subseteq \mathbb{R}^D$ is the input alphabet
- $S \subseteq \mathbb{R}^H$ is the set of states
- $\delta : S \times \Sigma \to S$ is the state transition function made up of a neural network.

RNN maintains an internal state and at each time step take an input $\mathbf{x}_t$ and updates its state $\mathbf{s}_t$ by applying the transition function made up of neural network δ the previous time step's state $\mathbf{s}_{t-1}$ and the input.

Often the input is not available directly as elements of Σ and the output desired is not the state of the RNN. In such cases, input is appropriately transformed and desired output is produced at each time step from the state $\mathbf{s}_t$:

$$\mathbf{y}_t = g(\mathbf{s}_t),$$

where g is an arbitrary differentiable function.

For example, in a regular recurrent language model (RRLM; Mikolov, Karafiát, et al. 2010) a document is treated as a sequence and an LSTM is trained to predict directly the next word conditioned on the sequence of words before it, *i.e.* maximize $p(w_t|w_{t-1|}, w_{t-2}, ..., w_0; \text{model})$. In this case, the input transformation is done by using a word lookup table from word to a vector in Σ. This word representation is used to update the state of the LSTM. The output transformation is the projection of $\mathbf{s}_t$ into a vector of size of the vocabulary V followed by a softmax. However, this method will require two large matrices of dimension $V \times H$ and cannot handle out of vocabulary words.

Algorithm 6.1 Sequential Monte Carlo

Let $z_0^p = z_0$ and weights $\alpha_0^p = 1/P$ for $p = 1, \ldots, P$.
for $t = 1$ **to** T **do**
  for $p = 1$ **to** P **do**
    Sample ancestor $a_{t-1}^p \sim \text{Categorical}\left(\alpha_{t-1}^1, \alpha_{t-1}^2, ..., \alpha_{t-1}^P\right)$
    Sample particle $z_t^p \sim f(z_t | z_{1:t-1}^{a_{t-1}^p})$
    Set $z_{1:t}^p = (z_{1:t-1}^{a_{t-1}^p}, z_t^p)$
    Compute the weight $\alpha_t^p \leftarrow \frac{\nu_t(z_{1:t}^p)}{\nu_{t-1}(z_{1:t-1}^{a_{t-1}^p}) f(z_t^p | z_{1:t-1}^{a_{t-1}^p})}$
  end for
  Normalize the weights $\alpha_t^p \leftarrow \frac{\alpha_t^p}{\sum_{l=1}^P \alpha_t^l}$
end for

To overcome above mentioned shortcomings, another input transformation has been proposed in context of natural languages, which looks at characters directly instead of words (Ling et al., 2015). On the spelling of the words another LSTM is run having characters as the input and the final state is used as the word representation. In this case, the memory requirement is reduced to half and the model can handle out-of-vocabulary words like typographical errors or new words made from composing existing ones. To use character level embedding, we can simply replace the word lookup table with the representation obtained from character-level LSTM. In case of natural languages, a similar trick of using a character-level LSTM to emit words can be applied as the output transformation as well. However, user modeling outputs (hash indices and URLs) lack morphological structure, and hence cannot leverage the technique.

6.2.3 *Sequential Monte Carlo*

Finally, we need a tool to sample from complex posteriors which would arise unavoidably in models having combination of LSTMs and Bayesian components. Sequential Monte Carlo (SMC; Doucet, Freitas, and Gordon 2001) is an algorithm that samples from a series of potentially unnormalized densities $\nu_1(z_1), \ldots, \nu_T(z_{1:T})$. At each step t, SMC approximates the target density ν_t with P weighted particles using importance distribution $f(z_t | z_{1:t-1})$:

$$\nu_t(z_{1:t}) \approx \hat{\nu}_t(z_{1:t}) = \sum_p \alpha_t^p \delta_{z_{1:t}^p}(z_{1:t}), \tag{6.1}$$

where α_t^p is the importance weight of the p-th particle and δ is the Dirac delta function. Repeating this approximation for every t leads to the SMC method, outlined in Algorithm 6.1.

The key to this method lies in the resampling, which is implemented by repeatedly drawing the ancestors of particles at each step. Intuitively, it encourages particles with a higher likelihood to survive longer, since the weight reflects the likelihood of the particle path. Final Monte Carlo estimate (6.1) consists only of survived particle paths, and sampling from this point masses is equivalent to choosing a particle path according to the last weights α_T. We refer to Andrieu, Doucet, and Holenstein (2010) and Doucet, Freitas, and Gordon (2001) for detailed proof.

6.2.4 *Related Works*

Our work is related to various dynamic topic models, however, previous works like ddLDA (ddCRP in general) or hidden Markov topic models provide only limited modeling capabilities of the temporal dynamics. Specifically, they impose smoothness constraints, *i.e.* topic of a word

is biased toward nearby topics. This constraint cannot learn a predictive action model, *e.g.* after "booking a flight" topic, the "booking a hotel" topic is likely to follow. Moreover, internet users often wander between related activities, *e.g.* "booking a hotel" topic will happen shortly after "booking a flight", but not necessarily immediately as the user might have been interrupted by something else. Thus we need a much richer temporal model such as an LSTM.

Another similar work would be lda2vec (Moody, 2016), where LDA is combined with local context in a fixed window size. This provides a sparse, interpretable model but lacks modeling the temporal aspect. The model cannot be used in a predictive-action setting. Also the sparsity is only in terms of per document parameters, whereas it suffers from huge dense of size vocabulary times embedding size.

Another relevant recent work is Sentence Level Recurrent Topic Model (SLRTM; Tian, B. Gao, and T.-Y. Liu 2016). However, this model cannot capture long-range temporal dependencies, as the topic for each sentence is still decided using an exchangeable (non-sequential) Dirichlet-multinomial scheme similar to the LDA. It might be able to preserve local sentence structure or grammar, but that is particularly not very useful for the task of user modeling. Furthermore, as it contains RNN that operates on the entire vocabulary (not topic as in our case), the SLTRM has lots of parameters.

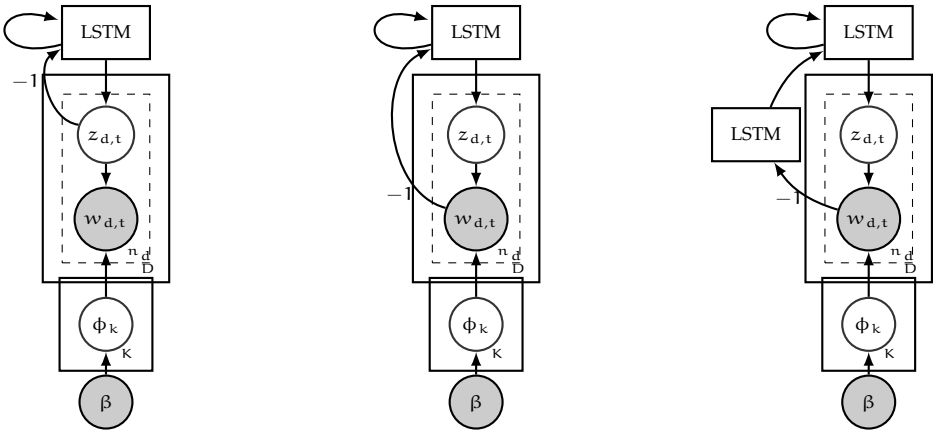
Finally, a solution to abrupt changes in themes could be use switching state-space (SSS) models such as Cacciatore and Nowlan (1994) and Zoubin Ghahramani and Geoffrey E. Hinton (1996). Most SSS models use linear (or simple non-linear) update or assume Markovianity, but it would not be capable to intricate dynamics of user data. Enhancing SSS models, like any other state space models, using neural networks to boost expressive power is not a new idea. Traditional approaches can be traced back to nonlinear extensions of linear dynamical systems, such as extended or unscented Kalman filters (Julier and Uhlmann, 1997), where both state transition and emission are generalized to nonlinear functions. The idea of parameterizing them with neural networks can be found in Zoubin Ghahramani and S. T. Roweis (1999), as well as many recent works (Archer et al., 2015; M. J. Johnson et al., 2016; Karl et al., 2017; Krishnan, Shalit, and Sontag, 2015, 2017) thanks to the development of recognition networks (D. Kingma and Welling, 2014; Rezende, Mohamed, and Wierstra, 2014). Similarly one could extend SSS, by employing LSTMs with latent variables. However, as in standard SSS models all mixtures evolve at each time step and a switch variable at time t selects a them to generate the input, they do not mitigate the vanishing gradient problem directly (Hochreiter and Jürgen Schmidhuber, 1997). This is the primary reason for developing a tailored model for the problem of sudden theme swings instead of just using LLA or LSTM+SSS.

6.3 LATENT LSTM ALLOCATION

In this section, we provide a detailed description of the proposed LLA model and its variant for performing joint clustering and modeling non-linear dynamics. An ideal model for such a task should have three characteristics. First, it should have a low number of parameters. Second, it should be interpretable, allowing human analysis of the components. Lastly and most importantly, the model should be accurate in terms of predicting future events. We show how LLA satisfies all of these requirements.

6.3.1 *Generative model*

In this model, we try to marry LDA with its strong interpretability capabilities with LSTM having excellent track record in capturing temporal information. In this regard, we propose



(a) Topic LLA (b) Word LLA (c) Char LLA

Figure 6.2: Graphical models for LDA and variants of proposed latent LSTM Allocation (LLA). In a slight abuse of plate notation, we do not imply exchangibility by the dashed plate diagram and -1 on an edge means the dependence is from one time step back.

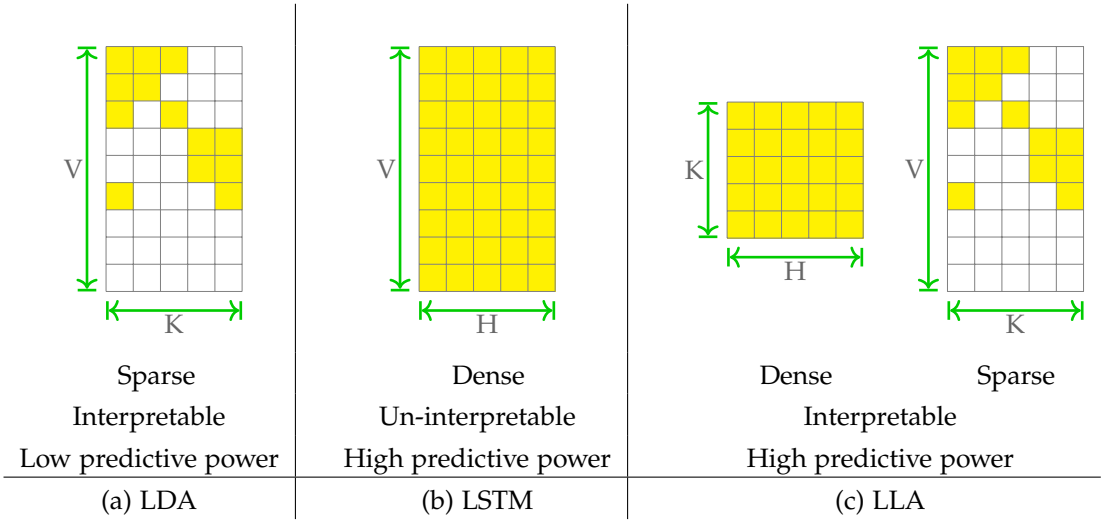
a factored model, *i.e.* we use the LSTM to model sequence of topics $p(z_t|z_{t-1}, z_{t-2}, ..., z_1)$ and multinomial-Dirichlet to model word emissions $p(w_i|z_i)$, similar to LDA. Suppose we have K topics, vocabulary size V, and a document collection $\mathcal{D}$ where each document $d \in \mathcal{D}$ is composed of N_d words. With these notations, the complete generative process can be illustrated as in Figure 6.2a and can be described as:

1. **for** $k = 1$ **to** K
   a) Choose topic $\boldsymbol{\phi}_k \sim \text{Dir}(\beta)$
2. **for each** document d in corpus $\mathcal{D}$
   a) Initialize LSTM with $\mathbf{s}_0 = \emptyset$
   b) **for each** word index t from 1 to N_d
      i. Update $\mathbf{s}_t = \text{LSTM}(z_{d,t-1}, \mathbf{s}_{t-1})$
      ii. Get topic proportions at time t from the LSTM state, $\theta = \text{softmax}_K(\mathbf{W}_p \mathbf{s}_t + \mathbf{b}_p)$
      iii. Choose a topic $z_{d,t} \sim \text{Categorical}(\theta)$
      iv. Choose word $w_{d,t} \sim \text{Categorical}(\boldsymbol{\phi}_{z_{d,t}})$

Under this model, the marginal probability of observing a given document d can be written as:

$$\begin{aligned} p(\boldsymbol{w}_d|\text{LSTM}, \boldsymbol{\phi}) &= \sum_{\boldsymbol{z}_d} p(\boldsymbol{w}_d, \boldsymbol{z}_d|\text{LSTM}, \boldsymbol{\phi}) \\ &= \sum_{\boldsymbol{z}_d} \prod_t p(w_{d,t}|z_{d,t}; \boldsymbol{\phi}) p(z_{d,t}|\boldsymbol{z}_{d,1:t-1}; \text{LSTM}). \end{aligned} \tag{6.2}$$

Here $p(z_{d,t}|\boldsymbol{z}_{d,1:t-1}; \text{LSTM})$ is the probability of generating topic for the next word in the document given topics of previous words and $p(w_{d,t}|z_{d,t}; \phi)$ is the probability of generating word given the topic, illustrating the simple modification of LSTM and LDA based language models.



(a) LDA (b) LSTM (c) LLA

Figure 6.3: Different parameters employed by LDA, LSTM and LLA. K is number of topics, V, is vocabulary size, and H is the dimension of LSTM state. (a) Topics of LDA, (b) word embedding of LSTM (c) factored topic embedding of LLA.

The advantage of this modification is two fold. Firstly, we obtain a factored model as shown in Figure 6.3, thereby the number of parameters is reduced significantly compared to RRLM. This is because, unlike RRLM we operate at topic level instead of words directly and the number of topics is much smaller than the vocabulary size, *i.e.* $K << V$. This allows us to get rid of large $V \times H$ word embedding look-up table and $V \times H$ matrix used in softmax over the entire vocabulary. Instead we map from hidden to state to topics first using a $K \times H$ matrix, which will be dense and then from topics to words using a $V \times K$ matrix under the Dirichlet-multinomial model similar to LDA which will be very sparse. Secondly, this model is highly interpretable. We recover global themes present in the documents as ϕ. The LSTM output represents topic proportions for document/user at time t. The LSTM input over topics can capture semantic notion of topics.

6.3.2 *Inference Overview*

As the computation of the true posterior of LLA as described above is intractable, we have to resort to an inference technique like mean field variational inference (Wainwright and Michael I Jordan, 2008) or stochastic EM (SEM) as will be described in Chapter 7. Below we describe the generic aspects of the inference algorithm for LLA.

Given a document collection, the inference task is to find the LSTM weights and word|topic probability matrix ϕ. This can be carried out using SEM. We begin by writing out the likelihood and lower bounding it as,

$$\sum_d \log p(\boldsymbol{w}_d|\text{LSTM}, \boldsymbol{\phi}) \geqslant \sum_d \sum_{z_d} q(z_d) \log \frac{p(z_d; \text{LSTM}) \prod_t p(w_{d,t}|z_{d,t}; \boldsymbol{\phi})}{q(z_d)}, \tag{6.3}$$

for any distribution q. Then the goal is to ensure an increase in this evidence lower bound (ELBO) with each iteration in expectation. Following the suit of most EM based inference methods, each iteration involves three phases, each of which is described below:

E-STEP We find the optimal variational distribution, which is nothing but the posterior:

$$q^\star(z_d) \propto p(z_d; \text{LSTM}) \prod_t p(w_{d,t}|z_{d,t}; \phi). \tag{6.4}$$

In the case of linear dynamical systems or HMM, efficient smoothing algorithms such as the RTS smoother (Rauch, F. Tung, and Striebel, 1965) or the forward-backward algorithm (Stratonovich, 1960) are available for computing the posterior expectations of sufficient statistics. However, without Markovian state transition, although the forward messages can still be computed, the backward recursion can no longer evaluated or efficiently approximated.

S-STEP Due to the difficulties in evaluating expectations with respect to the variational difference, we take an alternative approach to collect samples. For fixed LSTM parameters and ϕ, we sample the topic assignments z.

$$z_d \sim q^\star(\cdot), \tag{6.5}$$

given only the filtering equations. This acts as an unbiased sample estimate of the expectation with respect to the conditional distribution of z required in traditional EM. This sampled estimate not only enjoys similar statistical convergence guarantees (Nielsen, 2000) but also provides many computational benefits like speed-up and reduced memory bandwidth requirements a will be explained in Chapter 7.We will discuss an exact and approximate but fast posterior sampling algorithm for the S-Step in detail over the next two subsections.

M-STEP For fixed topic assignment z, update the ϕ and LSTM so as to improve the likelihood in expectation. As the dependence of likelihood on LSTM parameters and ϕ are independent given z, we can update them independently and in parallel. For the former, we use the closed form expression for the maximizer,

$$\phi_{wk} = \frac{\#\{w_{d,t} = w \text{ and } z_{d,t} = k\} + \beta}{\#\{z_{d,t} = k\} + V\beta} = \frac{n_{wk} + \beta}{n_k + V\beta}, \tag{6.6}$$

and for the latter we resort to stochastic gradient ascent which will increase the likelihood in expectation.

6.3.3 *Exact Inference with SMC*

In this subsection, we discuss how to draw samples from the posterior (6.4), corresponding to the S-step of the stochastic EM algorithm. For sake of simplicity of notation, we consider a document $w_1, ..., w_T$ of length T. Thus, the posterior from (6.4) is:

$$z_{1:T} \sim q^\star(z_{1:T}) = p(z_{1:T}|w_{1:T}) \propto \prod_t p(z_t|z_{1:t-1}; \text{LSTM}) p(w_t|z_t; \phi). \tag{6.7}$$

As the integration and normalization can be performed efficiently, the following quantities can be evaluated in the forward pass without Markov state transition:

$$\begin{aligned} \alpha_t &\equiv p(w_t|z_{1:t-1}) \propto \sum_{z_t} p(z_t|z_{1:t-1}; \text{LSTM}) p(w_t|z_t; \phi) = \sum_k \theta_{t,k} \phi_{w_t,k} \\ \gamma_t &\equiv p(z_t|z_{1:t-1}, w_t) \propto p(z_t|z_{1:t-1}; \text{LSTM}) p(w_t|z_t; \phi) = \theta_{t,z_t} \phi_{w_t,z_t}. \end{aligned} \tag{6.8}$$

The task is to draw from the joint posterior of $z_{1:T}$ only given access to these *forward messages*.

Algorithm 6.2 Inference with Particle Gibbs for LLA

Let $z_0^p = z_0$ and $\alpha_0^p = 1/P$ for $p = 1, \ldots, P$.
for $t = 1$ **to** T **do**
  Fix reference path: set $a_{t-1}^1 = 1$ and $z_{1:t}^1 = z_{1:t}^\star$ from previous iteration.
  for $p = 2$ **to** P **do**
    Sample ancestor $a_{t-1}^p \sim \text{Categorical}\left(\alpha_{t-1}^1, \alpha_{t-1}^2, ..., \alpha_{t-1}^P\right)$
    Sample particle $z_t^p \sim \gamma_t^p \propto p(z_t | z_{1:t-1}^{a_{t-1}^p}; \text{LSTM}) p(w_t | z_t; \boldsymbol{\phi})$ in (6.9)
    Set $z_{1:t}^p = (z_{1:t-1}^{a_{t-1}^p}, z_t^p)$
    Compute the weight $\alpha_t^p \leftarrow \sum_k p(z_t = k | z_{1:t-1}^{a_{t-1}^p}; \text{LSTM}) p(w_t | z_t = k; \boldsymbol{\phi})$ in (6.10)
  end for
  Normalize the weights $\alpha_t^p \leftarrow \frac{\alpha_t^p}{\sum_{l=1}^P \alpha_t^l}$
end for
Sample $r \sim \alpha_T$, return the particle path $z_{1:T}^{a_T^r}$.

We propose to use a method based on SMC, which is a principled way of sampling from a sequence of distributions. More importantly, it does not require the model to be Markovian (Frigola et al., 2013; Lindsten, Michael I. Jordan, and Schön, 2014). As described earlier in Section 6.2.3, the idea is to approximate the posterior (6.7) with point masses, *i.e.*, weighted particles. Let $f(z_t | z_{1:t-1}, w_t)$ be the importance density, and P be the number of particles. We then can run Algorithm 6.1 with $\nu_t(z_{1:t}) = p(w_{1:t}, z_{1:t})$ being the unnormalized target distribution at time t. As for the choice of the proposal distribution $f(\cdot)$, we use the predictive distribution $p(z_t | z_{1:t-1}, w_t)$, which is a locally optimal proposal in terms of variance (Andrieu, Doucet, and Holenstein, 2010), *i.e.* $f^\star(z_t | z_{1:t-1}, w_t) = p(z_t | z_{1:t-1}, w_t)$. Note that resulting proposal distribution is precisely one of the available forward messages:

$$f^\star(z_t | z_{1:t-1}^{a_{t-1}^p}, w_t) = \gamma_t^p \propto p(z_t | z_{1:t-1}^{a_{t-1}^p}; \text{LSTM}) p(w_t | z_t; \boldsymbol{\phi}). \tag{6.9}$$

With the choice of predictive distribution as the proposal density $p(z_t | z_{1:t-1}, w_t)$, the importance weight simplifies to

$$\alpha_t^p \propto \tilde{\alpha}_t^p = \sum_k p(z_t = k | z_{1:t-1}^{a_{t-1}^p}; \text{LSTM}) p(w_t | z_t = k; \boldsymbol{\phi}) = \sum_k \theta_{t,k}^p \phi_{w_t,k},, \tag{6.10}$$

which is not a coincidence that the name collides with the message α_t. Interestingly, this quantity no longer depends on the current particle z_t^p. Instead, it marginalizes over all possible particle assignments of the current time step.

After a full pass over the sequence, the algorithm produces Monte Carlo approximation of the posterior and the marginal likelihood:

$$\begin{aligned} \hat{p}(z_{1:T} | w_{1:T}) &= \sum_p \alpha_T^p \delta_{z_{1:T}^p}(z_{1:T}), \\ \hat{p}(w_{1:T}) &= \prod_t \frac{1}{P} \sum_p \tilde{\alpha}_t^p. \end{aligned} \tag{6.11}$$

The inference is completed by a final draw from the approximate posterior,

$$z_{1:T}^\star \sim \hat{p}(z_{1:T} | w_{1:T}), \tag{6.12}$$

which is essentially sampling a particle path indexed by the last particle. Specifically, the last particle z_T^p is chosen according to the final weights α_T, and then earlier particles can be obtained by tracing backwards to the beginning according to the ancestry indicators a_t^p at each position.

Since SMC produces a Monte Carlo estimate, as the number of particles $P \to \infty$ the approximate posterior (6.11) is guaranteed to converge to the true posterior for a fixed sequence. However, as the length of the sequence T increases, the number of particles needed to provide a good approximation grows exponentially. This is the well-known depletion problem of SMC (Andrieu, Doucet, and Holenstein, 2010).

One elegant way to avoid simulating enormous number of particles is to marry the idea of MCMC with SMC (Andrieu, Doucet, and Holenstein, 2010). The idea of such Particle MCMC (PMCMC) methods is to treat the particle estimate $\hat{p}(\cdot)$ as a proposal, and design a Markov kernel that leaves the target distribution invariant. Since the invariance is ensured by the MCMC, it does not demand SMC to provide an accurate approximation to the true distribution, but only to give samples that are approximately distributed according to the target. As a result, for any fixed $P > 0$ the PMCMC methods ensure the target distribution is invariant.

We choose the Gibbs kernel that requires minimal modification from the basic SMC. The resulting algorithm is known as Particle Gibbs (PG), which is a conditional SMC update in a sense that a reference path $z_{1:T}^{ref}$ with its ancestral lineage is fixed throughout the particle propagation of SMC. It can be shown that this simple modification to SMC produces a transition kernel that is not only invariant, but also ergodic under mild assumptions. In practice, we use the assignments from previous step as the reference path. The final algorithm is summarized in Algorithm 6.2. Combined with the stochastic EM outer iteration, the final algorithm is an instance of the particle SAEM (Lindsten, 2013; Schön et al., 2015), under non-Markovian state transition.

6.3.4 *Fast Approximate Inference*

While particle Gibbs based S-Step a shown in Algorithm 6.2 provides exact samples, it is computationally very experiment. In this subsection, we provide approximate techniques for S-Step to speed up inference.

We begin by revisiting the conditional probability in S-Step for topic at time step t given word at time t and past history of topics under LLA,

$$p(z_{d,t} = k|w_{d,t}, z_{d,1:t-1}; \text{LSTM}, \boldsymbol{\phi}) \propto p(z_{d,t} = k|z_{d,1:t-1}; \text{LSTM}) p(w_{d,t}|z_{d.t} = k; \boldsymbol{\phi}) \quad (6.13)$$

The difficulty arises in sampling due to complicated dependencies among the latent variables $z_{d,t}$, unlike that in a Markov model. One way to circumvent the tight dependencies in $z_{d,t}$ is to make a factorization assumption:

$$p(z_{d,t} = k|w_{d,t}, z_{d,1:t-1}; \text{LSTM}, \boldsymbol{\phi}) \propto p(z_{d,t} = k|z_{d,1:t-1}^{prev}; \text{LSTM}) p(w_{d,t}|z_{d.t} = k; \boldsymbol{\phi}) \quad (6.14)$$

where $z_{d,1:t-1}^{prev}$ is the assignments from the previous iteration of S-Step. With this approximation, samples can be easily generated, whose computational cost we will discuss next.

The first term represents probability of various topics predicted by LSTM dynamics after final softmax and second term comes from multinomial-Dirichlet word emission model given the topic. Naïvely sampling from this distribution would cost $O(KH + H^2)$ per word.

Optionally, one could speed up this sampling. Note that the second term in (6.13) has a sparse structure. For sampling from (6.13) the computation of the normalizer is not essential. To elab-

orate, let us define n_{wk} as the number of times word w currently assigned to topic k and n_k as the number of tokens assigned to topic k. Explicitly writing down first term:

$$p(w_{d,t} = w|z_{d,t} = k, \boldsymbol{\phi}) = \phi_{wk} = \frac{n_{wk} + \beta}{n_k + V\beta} = \underbrace{\frac{n_{wk}}{n_k + V\beta}}_{\text{sparse}} + \underbrace{\frac{\beta}{n_k + V\beta}}_{\text{dense}} \tag{6.15}$$

There is an inherent sparsity present in n_{wk}, as a given word would be typically about only a handful of topics, $K_w \ll K$. The second term represents the global count of tokens in the topic and will be dense, regardless of the number of documents, words or topics.

Following A. Q. Li et al. (2014), we devise an *optional* fast sampling strategy for exploiting this sparsity and construct a Metropolis-Hastings sampler. The idea is to replace the low weight dense term by a cheap approximation, while keeping the high weight sparse term exact. This leads to a proposal distribution that is close to (6.13), while at the same time allowing us to draw from it efficiently:

- First draw a biased coin to decide whether to draw from the sparse n_{wk} term or from the dense term. The bias is

$$b = \frac{p}{p+q}, \text{ where}$$
$$p = \sum_{k:n_{wk}\neq 0} \frac{n_{wk}}{n_k + V\beta} p(z_{d,t} = k|\boldsymbol{z}_{d,1:t-1}; \text{LSTM})$$
$$q = \sum_k \frac{\beta}{n_k + V\beta} \qquad \text{(pre-computed)}.$$

- If we draw from the sparse term, the cost is $O(K_w H + H^2)$, else the cost is $O(H^2)$ using the alias trick.
- Finally, perform a MH accept/reject move by comparing the approximation with the true distribution.

The execution of the whole inference and learning process includes several iterations involving the above mentioned two phases as outlined in Algorithm 6.3. This inference procedure is not an ad-hoc method, but an instantiation of the well studied SEM method. We ensure that in each iteration we increase the ELBO in expectation. Also, it is just a coincidence that for all such latent variable models the equations for SE-step looks like Gibbs sampling. (Exploiting this coincidence, in fact, one can prove that parallel Gibbs update will work for such models, as we will see in Chapter 7 or in Tristan, Tassarotti, and Steele Jr. (2015), whereas in general parallel Gibbs sampling fails miserably *e.g.* for Ising models.) Moreover, we found empirically that augmenting the SEM inference with a beam search improved the performance.

Automatic differentiation software packages such as TensorFlow (Martin Abadi et al., 2015) significantly speed up development time, but have limited control structures and indexing. Whereas random sampling requires varied control statements, thus was implemented separately on the CPU. Furthermore, the SEM saved us precious GPU-CPU bandwidth by transmitting only an integer per token instead of a K-dimensional variational parameter.

Algorithm 6.3 Stochastic EM for LLA

Input: Document corpus $\mathcal{D}$.

1: Initialize ϕ and LSTM with a few iterations of LDA
2: **repeat**
SE-Step:
3: **for** all document $d \in \mathcal{D}$ **in parallel do**
4: **for** $t \leftarrow 1$ to N_d (possibly with padding) **do**
5: $\forall k \in \{1, \cdots, K\}$, *i.e.* for every topic index obtain by LSTM forward pass:

$$\pi_k = \phi_{w_{d,t}k} p(z_{d,t} = k | \boldsymbol{z}_{d,1:t-1}; \text{LSTM}).$$

6: Sample $z_{d,t} \sim \text{Categorical}(\boldsymbol{\pi})$
7: **end for**
8: **end for**
M-Step:
9: Collect sufficient statistics to obtain:

$$\phi_{wk} = \frac{n_{wk} + \beta}{n_k + V\beta}, \quad \forall w, k$$

10: **for** mini-batch of documents $\mathcal{B} \subset \mathcal{D}$ **do**
11: Compute the gradient by LSTM backward pass

$$\frac{\partial \mathcal{L}}{\partial \text{LSTM}} = \sum_{d \in \mathcal{B}} \sum_{t=1}^{N_d} \frac{\partial \log p(z_{d,t} | z_{d,1:t-1}; \text{LSTM})}{\partial \text{LSTM}}$$

12: Update LSTM parameters by stochastic gradient descent methods like Adam (D. P. Kingma and Ba, 2014).
13: **end for**
14: **until** Convergence

6.3.5 *Adding Context*

Utilizing the word or user action, $w_{d,t}$, directly would be more informative to model the dynamics and predict the next topic. For example, rather than only knowing the previous action belonged to "electronics purchase" topic, knowing exactly that a camera was bought, makes it easier to predict user's next interest, *e.g.* camera lenses.

In order to incorporate the exact context, we construct a variant of LLA, called **word LLA**, where the LSTM is provided with an embedding of previous word $w_{d,t-1}$ directly instead of $z_{d,t-1}$. The corresponding graphical model is shown in Figure 6.2b and the joint likelihood is:

$$\sum_d \log p(\boldsymbol{w}_d | \text{LSTM}, \phi) = \sum_d \sum_t \log \sum_{z_{d,t}} p(w_{d,t} | z_{d,t}; \phi) p(z_{d,t} | \boldsymbol{w}_{d,1:t-1}; \text{LSTM}) \tag{6.16}$$

A SEM inference strategy can be devised similar to topic LLA as presented in Section 6.3.4.

However, this modification brings back the model to the dense and high number of parameter regime as a large trainable $V \times H$ lookup table for the word embeddings is needed for the input transformation. This problem can be mitigated by using char-level LSTM (**char LSTM** for short) to construct the input transformation. Such character-level LSTM have recently shown promising results in generating structured text (Karpathy, Justin Johnson, and Fei-Fei, 2015)

Model	Input Tranformation	Output Transformation
word LSTM	Word Embedding	Softmax over vocabulary
Char LSTM	Char Embedding	Softmax over vocabulary
Topic LLA	Topic Embedding	Topic based
Word LLA	Word Embedding	Topic based
Char LLA	Char Embedding	Topic based

Table 6.1: Enumerating different input and output representation, leading to variants of LLA models. Note we exclude char level output transformation due to its inapplicability to user modeling.

as well as in producing semantically valid word embeddings with a model we will refer to as char-LSTM) (Ling et al., 2015). The character-level models do not need the huge lookup table for words, as they operate directly on the characters and the number of distinct characters is extremely small. We chose the latter as our input transformation and briefly describe it below.

The input word w is decomposed into a sequence of characters $c_1, \ldots, c_m$, where m is the length of w. Each character c_i is transformed using a lookup table and fed into a bidirectional LSTM The forward LSTM evolves on the character sequence and produces a final state $\mathbf{h}^f_m$. While the backward LSTM receives as input the reverse sequence, and yields its own final state $\mathbf{h}^b_0$. Both LSTMs use a different set of parameters. The representation $\mathbf{e}^c_w$ of word w is obtained by combining forward and backward final states:

$$\mathbf{e}^C_w = \mathbf{D}^f \mathbf{s}^f_m + \mathbf{D}^b \mathbf{s}^b_0 + \mathbf{b}_d, \tag{6.17}$$

where $\mathbf{D}^f$, $\mathbf{D}^b$ and $\mathbf{b}_d$ are parameters that determine how the states are combined. This e^c_w is provided to the topic level LSTM as the input providing information about the word. Thus we are able to incorporate more context information by providing some information about the previous word without large number of parameters. We call this model, **char LLA**.

The different variants of LLA and LSTM as language model can be unified and thought of having different input and out transformations over LSTM for capturing the dynamics. The possible combinations are listed in Table 6.1. We will study each of them empirically next.

6.4 EXPERIMENTS ON LLA

We perform a comprehensive evaluation of our model against several deep models, dynamic models, and topic models. *For reproducibility we focus on the task of language modeling over the publicly available Wikipedia dataset*, and for generality, we show additional experiments on the less-structured domain of user modeling.

6.4.1 *Setup*

For all experiments we follow the standard setup for evaluating temporal models, *i.e.* divide each document (user history) into 60% for training and 40% for testing. The task is to predict the remaining 40% of the document (user data) based on the representation inferred from the first 60% of the document. We use perplexity as our metric (lower values are better) and compare our models (topic-LLA, word-LLA, and char-LLA) against the following baselines:

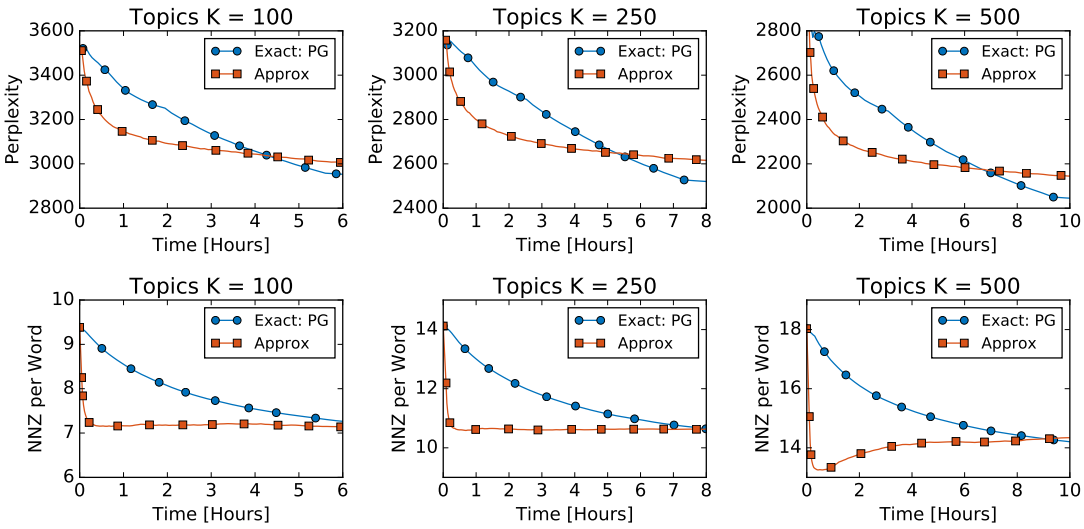


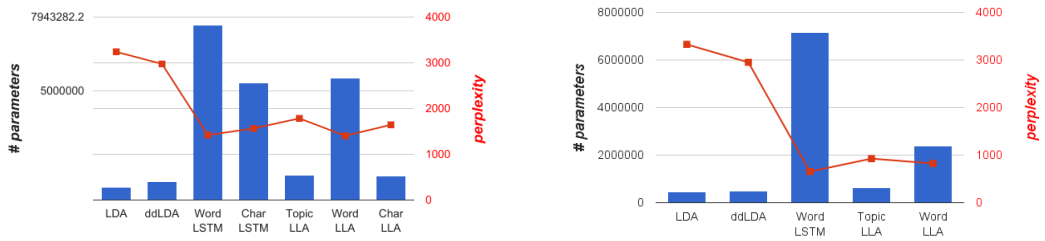
Figure 6.4: Comparison of the exact inference based on SMC to the approximate one assuming factored model. The top row represents perplexity on the held out set and the lower row represents the non zero entries in the word-topic count matrix. Lower perplexity indicates a better fit to the data and lower NNZ results in a sparser model and usually having better generalization.

- **Autoencoder:** A feed forward neural network that comprises of encoder and decoder. The encoder projects the input data into a low-dimensional representation and the decoder reconstructs the input from this representation. The model is trained to minimize reconstruction error.
- **LSTMs:** We consider both word-level LSTM language model (word-LSTM) and character-level hierarchical LSTM (char-LSTM) language model.
- **LDA:** The vanilla version trained using collapsed Gibbs sampling over the bag of word representation.
- **ddLDA:** Distance-dependent LDA is a variant of LDA incorporating the word sequence. It is based on a fixed-dimensional approximaiton of the distance-dependent Dirichlet process (David Blei and Frazier, 2011). In this model a word is assigned to a topic based on the popularity of the topics assigned to nearby words. Note that this model subsumes other LDA-based temporal models such as hidden Markov topic Model (Andrews and Vigliocco, 2010) and RCRP based models (Amr Ahmed and Xing, 2008).

We trained all deep models using stochastic gradient decent with Adam (D. P. Kingma and Ba, 2014). All LSTM and LLA based models used the sampled softmax method for efficiency (Cho, Memisevic, and Y. Bengio, 2015). Finally, all hyper-parameters of the models were tuned over a development set.

6.4.2 *Language modeling*

We used the publicly available Wikipedia dataset to evaluate the models on the task of sequence prediction. Extremely short documents, *i.e.* documents having length less than 500 words were excluded. The dataset contains ~0.7 billion tokens and ~1 million documents. The most frequent 50k word-types were retained as our vocabulary. Unless otherwise stated, we used 1000



(a) Wikipedia

(b) User search click history

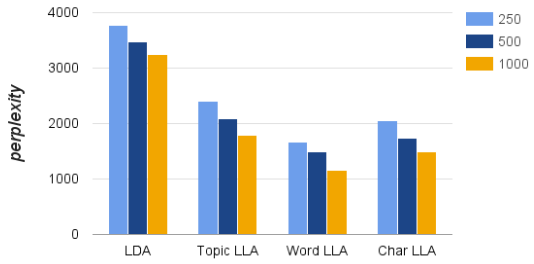
Figure 6.5: Test perplexity and number of parameters of various models. Bars represent model sizes and the solid curve represents perplexity over testset (lower is better).

topics for LLA and LDA variants. For LSTM and LLA variants, we selected the dimensions of the input embedding (word or topic) and evolving latent state (over words or topics) in the range of {50, 150, 250}. In case of character-based models, we tuned the dimensions of the character embedding and latent state (over characters) in the range of {50, 100, 150}.

EXACT VS APPROXIMATE INFERENCE We want to compare the inference with the exact posterior sampling using SMC and the faster approximate posterior sampling. Figure 6.4 shows the test perplexity (lower is better) and number of nonzeros in the learned word topic count matrix (lower is better). For varying number of topics, the SMC algorithm eventually gives better perplexity, but is significantly slower than the approximate solution and the gap between the two approaches is small. Similarly, the nonzeros in the word|topic decreases rapidly for the approximate inference, which is highly desirable. The rapid sparsification further speeds-up the subsequent iterations.

ACCURACY VS MODEL SIZE Figure 6.5(a) compares model size in terms of number of parameters with model accuracy in terms of test perplexity. As shown in the figure, LDA yields the smallest model size due to the sparsity bias implicit in the Dirichlet prior over the topic|word distributions. On the other hand, word-LSTM gives the best accuracy due to its ability to utilize word level statistics; however, the model size is order of magnitudes larger than LDA. Char-LSTM achieves almost 50% reduction in model size over word-LSTM at the expense of a slightly worse perplexity. The figure also shows that LLA variants (topic-LLA, word-LLA, and char-LLA) can trade accuracy vs model size while still maintaining interpretability since the output of the LSTM component is always at the topic level. Note that the figure depicts the smallest word-LSTM at this perplexity level, as our goal is not to beat LSTM in terms of accuracy, but rather to provide a tuning mechanism that can trade-off perplexity vs model size while still maintaining interpretability. Finally, compared to LDA, which is a widely used tool, LLA variants achieve a significant perplexity reduction at a modest increase in model size while still maintaining topic sparsity. As shown in Figure 6.1, the autoencoder model performs poorly in terms of model size and perplexity thus we eliminated it from Figure 6.5(a) and the following figures to avoid cluttering the display.

CONVERGENCE OVER TIME At first glance, LLA seems to be more involved than both LDA and LSTM. So, we raise the question whether the added complexity leads to a slower training. Figure 6.8 shows that compared to LSTM based models, LLA variants achieve comparable convergence speed. Moreover, compared to fast LDA samplers, our LLA variants introduce only a modest increase in training time. The figure also shows that character based models (char-LSTM and char-LLA) are slightly slower to train compared to word level variants due to their nested nature and need to propagate gradients over both word and character level LSTMs.



Wikipedia

Figure 6.6: The effect of the number of topics over the performance of LDA and LLA.

ABLATION STUDY Since both LDA and LLA result in interpretable models, we want to explore if LDA can achieve a perplexity similar to a given LLA model by just increasing the number of topics in LDA. Figure 6.17 shows the performance of LDA and variants of LLA for different number of topics. As can be seen from the figure, even with 250 topics, all LLA based models achieve much lower perplexity than LDA with 1000 topics. In other words, LLA improves over LDA not because it uses a slightly larger model, but rather because it models the sequential order in the data.

INTERPRETABILITY Last but not least, we demonstrate the interpretability of our models. Similar to LDA, the proposed LLA also uncovers global themes, a.k.a topics prevalent in the dataset. We found qualitatively the topics produced by LLA to be cleaner. For example, in Table 6.2b we show a topic about funding, charity, and campaigns recovered by both. LDA includes spurious words like Iowa in the topic just because it co-occurs in the same documents. Whereas modeling the temporal aspect allows LLA to correctly switch topics at different parts of the sentence producing cleaner topic regarding the same subject.

Modeling based on only co-occurrences in LDA leads to further issues. For example, strikes among mine workers are common, so the two words will co-occur heavily but it does not imply that strikes and mining should be in the same topic. LDA assign both the words in the same topic as shown in Table 6.2a. Modeling sentences as an ordered sequence allows distinction between the subjects and objects in the sentence. In the previous example, this leads to factoring into two separate topics of strikes and mine workers.

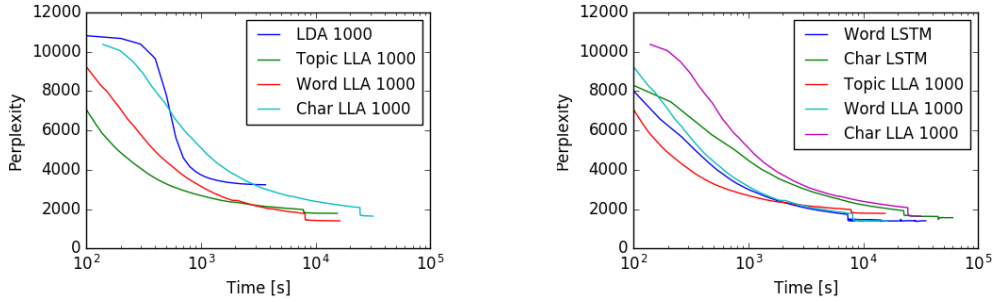
LDA	strike, strikes, striking, miners, strikers, workers, day, began, general, called, pinkerton, action, hour, hunger, keefe
LLA	union, unions, strike, workers, labor, federation, trade, afl, bargaining, cio, organization, relations, strikes, national, industrial
	mining, coal, mine, mines, gold, ore, miners, copper, iron, rush, silver, mineral, deposits, minerals, mined

(a) Factored topics

LDA	foundation, **iowa**, charity, fund, money, campaign, raised, donated, funds, donations, raise, support, charitable, **million**, donation
LLA	fund, foundation, money, funds, support, charity, funding, donations, campaign, raised, donated, **trust**, raising, **contributions**, **awareness**

(b) Cleaner topics

Table 6.2: LLA vs LDA



(a) LLA-variant against vanilla LDA

(b) LLA-variants against LSTM-variants

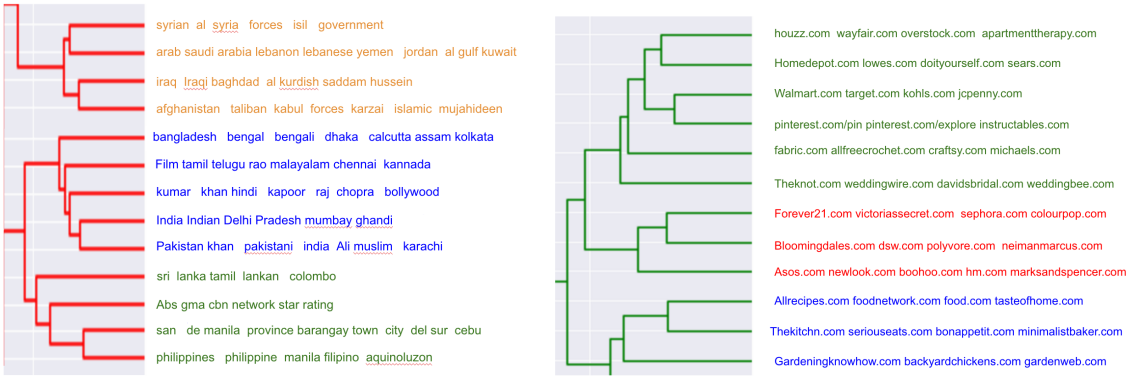
Figure 6.8: Convergence speed for various models. The x-axis represents time in second while the Y-axis gives the perplexity over the test dataset.

The topic LLA provides embedding of the topics in a Euclidean metric space. This embedding allows us to understand the temporal structure learned by the model: topics close in this space are expected to appear in close sequential proximity in documents. To understand this, we built a topic hierarchy using the embeddings which uncovers interesting facts about the data. For example in Figure 6.9a, we show a portion of the topic hierarchy discovered by topic-LLA with 1000 topics. For each topic we list the top few words. The theme of the figure is countries in Asia. It groups topics relating to one country together and puts topics related to neighboring countries close by. Three prominent clusters are shown form top to bottom which corresponds to moving from west to east on the map of Asia. Top cluster is about Arab world, second one represent the Indian sub-continent, and the third one starts with south-east Asia like Philippines. (The topic abs gma cbn represents TV station in south-east Asia.) The complete hierarchy of topic is very meaningful and can be viewed at `http://manzil.ml/lla.html`.

6.4.3 *User modeling*

We use an anonymized sample of user search click history to measure the accuracy of different models on predicting users' future clicks. An accurate model would enable better user experience by presenting the user with relevant content. The dataset is anonymized by removing all items appearing less than a given threshold, this results in a dataset of ~50 million tokens, and 40K vocabulary size. This domain is less structured than the language modeling task since users' click patterns are less predictable than the sequence of words which follow definite syntactic rules. We used a setup similar to the one used in the experiments over the Wikipedia dataset for parameters ranges and selections.

Figure 6.5(b) gives the same conclusion as Figure Figure 6.5(a): LLA variants achieve significantly lower perplexity compared to LDA and at the same time they are considerably smaller models compared to LSTM. Furthermore, even compared to ddLDA, an improved variant of LDA, our proposed LLA achieves better perplexity. ddLDA models time by introducing smoothness constraints that bias future actions to be generated from recent topics; however, it does not possess a *predictive* action model. As a result, it can neither model the fact that "booking a flight" topic should not be repeated over a short course of time nor that "booking a hotel" topic would likely follow shortly, but not necessarily immediately, after "booking a flight" topic. LLA, on the other hand, is capable of capturing this intricate dynamics by modeling the evolution of user topics via an LSTM – an extremely powerful dynamic model. This



(a) Wikipedia (b) User search click history

Figure 6.9: Topic embedding discovered using LLA

point is more evident if we consider the following *hand-crafted*[1] user click trace in context of the topics depicted in Figure 6.9b:

theknot.com zola.com weddingwire.com www.bridalguide.com pinterest.com food.com doityourself.com .....

There are four topics represented and color-coded (best viewed in color): wedding (sixth topic from top), social media (fourth from top), food (third form bottom) and home improvement (second form top) – all in Figure 6.9b. We asked each model to predict what would be the three most likely topics that would appear next in current user's session. LDA predicted wedding as the top topic followed by a tie for the remaining topics. ddLDA, whose output is based on exponential decay, yields wedding as most likely, followed by doityourself topic, and then the food topic as expected. In contrast, LLA ranks the most likely three topics as: doityourself topic, houzz topic (top topic in Figure 6.9b), and the wedding topic. This is indicative of LLA learning that once a user starts in the doitourself topic, the user will stay longer in this part of the tree for the task and wander in the nearby topics for a while. Moreover, LLA still remembers the wedding topic, and unlike other models, LLA did not erroneously pick neighbors of the wedding topic among the three most likely topics to follow. More generally, one can extract the topic transitions learned by LLA, as illustrated in Figure 6.10.

Finally, to demonstrate the interpretability of our model, in Figure 6.9b, we show a portion of the topic hierarchy discovered by topic-LLA with 250 topics. For each topic we list the top few clicked items. The theme of the figure is wedding and various house chores. Three prominent clusters are shown from top to bottom. The top cluster is about house renovations and wedding. Second cluster captures makeup, designer shoes, and fashion. The bottom cluster capture kitchen, cooking and gardening. It is clear form the hierarchy that LLA groups topics into the embedding space based on *sequential proximity* in the search click history.

6.4.4 *Effect of Joint Training*

One might wonder what would happen if we first train LDA and then simply train LSTM over the topic assignments from the LDA. We refer to this baseline as "independent learner" since LDA and LSTM are trained independently. In Table 6.3, we compare its performance against LLA, which jointly learns the topics and the evolution dynamics. As we can see, joint

1 For privacy concerns, we construct an artificial example manually for illustration purposes.

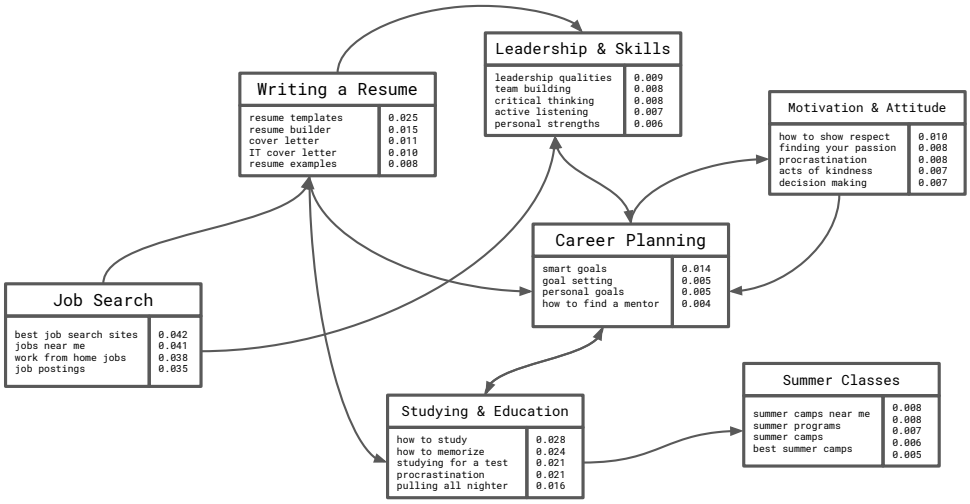


Figure 6.10: State transition graph over topics from user data.

training is significantly better, since the LSTM will fit the random errors introduced by the topic assignments inferred form the LDA model, and in fact LSTM will learn to be as good as the sequence produced by an LDA model (which is time-oblivious to start with). The effect is more pronounced in the user history data due to the unstructured nature of this domain.

Dataset	Independent learner	Joint learner
Wikipedia	2119	1785
User Click	1572	927

Table 6.3: Advantage in terms of perplexity for joint learning.

6.5 THREADED LLA

LLA, while expressive and interpretable, can still capture spurious dynamics in sequential data with mixed themes. From now onwards, we focus on user modelling task only. In Figure 6.11 we visualize the learned transition graph over topics as extracted from LLA when fitted over a user search history dataset. As evident, while the model captured useful transitions between camping gear and hiking topics, it also captured a couple of spurious transitions between ice-cream and painting topics. This can be attributed to mixed nature of user data, for instance, the user might search for hiking, home remodeling, and baking desserts with abrupt transitions between these totally different themes. While LSTM has the expressive power to capture long-range dependency structures, the introduction of latent variables (topics) exacerbate the complexity of the surface of objective function and precludes the optimization algorithm from linking each item in the sequence to the most relevant item in the the same theme from the user history. In addition LLA can not break the transition graph into themes such as camping and baking deserts that would give another layer of structure and interpretability.

To address these shortcomings of LLA, we introduce thLLA, threading LLA as a more faithful model for sequential data with mixed themes and abrupt transitions between the themes.

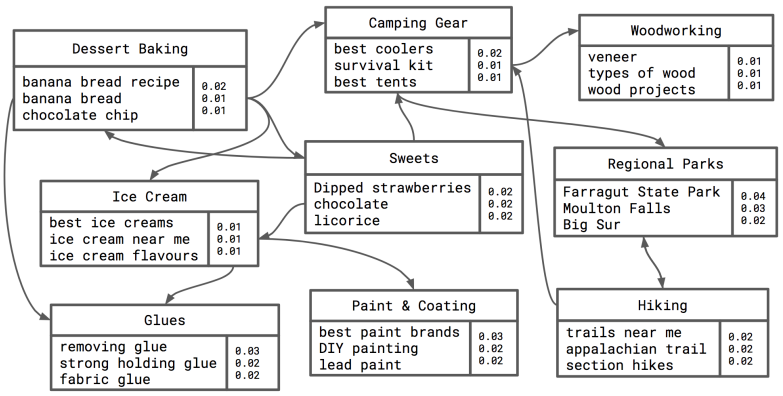


Figure 6.11: Transition graph from LLA illustrating its shortcoming. The LLA model is fit over user search history data, where each topic is a distribution of search queries and each edge denotes a temporal transition. As evident, LLA captures both useful and spurious dynamics.

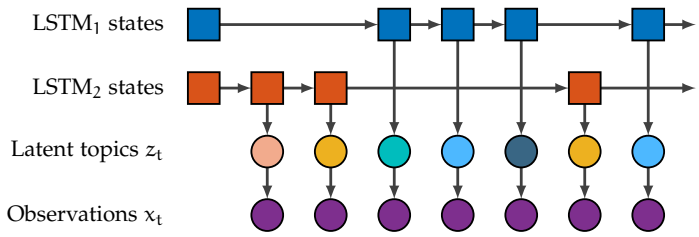
thLLA generates a sequence from a mixture of LLA models, topics are shared among all LLA models in the mixture, however, each LLA model in the mixture has its own LSTM component that captures dynamic evolution over a subset of the topics in the shared space. The input data is endowed with a switch variable for each item that selects a given mixture to generate this item. In that regard each item in the sequence is generated NOT conditioned on all the previous items as in LLA but rather conditioned on items that are associated with the same mixture in the history. In that regard our model can be seen as a threading or segmentation model that breaks the user sequence into coherent themes and models dynamic evolution within each theme separately. thLLA still enjoys all nice properties of LLA, namely, interpret-ability and small model size while being more accurate. Unfortunately, this comes with an added complex dependency structure for which inference would be extremely non-trivial. To remedy this, we present an efficient sampler based on particle MCMC method for inference that can draw from the joint posterior over both segments and topic assignments. We show quantitatively as well as qualitatively the efficacy and *interpretability* of thLLA over several datasets.

6.5.1 *Generative process*

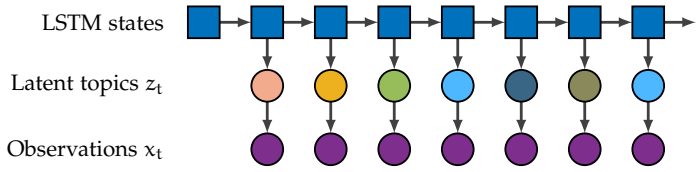
In thLLA, we employ a mixture of different LSTM components to model different segments of the dynamics in each sequence. Switching between the mixture components is modeled in the topic space like LLA and we use a multinomial-Dirichlet distribution to model emissions $p(x_t|z_t)$ similar to LDA, where x_t is the observed sequence and z_t is the latent topic index.

Suppose we have K topics and M mixtures in the dynamics. For each of the M mixtures, we have different LSTMs, denote them by $\text{LSTM}_1, ..., \text{LSTM}_M$. All the LSTMs share a common embedding look-up table E. Assume that we have a dataset of user activity histories $\mathcal{D}$ where each user history $u \in \mathcal{D}$ is composed of N_u actions. With these notations in place, the complete generative process can be described as follows:

1. **for** $k = 1$ **to** K:
   a) Draw topic distribution $\boldsymbol{\phi}_k \sim \text{Dir}(\beta)$
2. **for each** user u **in** dataset $\mathcal{D}$:
   a) Initialize all LSTMs with $\mathbf{s}_m = 0$ and set $z_m^{\text{prev}} = 0$ for $1 \leqslant m \leqslant M$
   b) Draw user $\boldsymbol{\pi}_u \sim \text{Dir}(\alpha)$



(a) Example of thLLA: Can easily switch context



(b) Corresponding LLA struggles during abrupt switch

Figure 6.12: Illustration of generative process of thLLA vs LLA. In this example, thLLA breaks the sequence into two segments: one about blue topics (camping) and one about red topics (food and desert). It models each segment with separate LSTM that can capture clean, intricate dynamics over these subset of topics, whereas LLA struggles in practice. Note that, in LLA, topic at time t is linked to that at time $t-1$ but in thLLA, each time step is linked to the most recent time step in its theme (not necessarily the most recent one in time).

c) **for each** action index $t = 1$ **to** N_u:
   i. Select behavior mode at time t as $y_t \sim \text{Categorical}(\boldsymbol{\pi}_u)$
   ii. Update corresponding dynamics $\mathbf{s}_{y_t} = \text{LSTM}_{y_t}\left(E\left[z_{y_t}^{\text{prev}}\right], \mathbf{s}_{y_t}\right)$
   iii. Get topic proportions at time t from the active LSTM, $\theta = \text{softmax}_K(\mathbf{W}_p \mathbf{s}_{y_t} + \mathbf{b}_p)$
   iv. Choose a topic $z_{u,t} \sim \text{Categorical}(\theta)$
   v. Save $z_{y_t}^{\text{prev}} = z_{u,t}$
   vi. Choose user action $x_{u,t} \sim \text{Categorical}(\boldsymbol{\phi}_{z_{u,t}})$

For a single user history sequence u, the above mentioned generative process specifies the following marginal probability distribution of observing the sequence:

$$\begin{aligned} p(x_u|\omega, \boldsymbol{\phi}, \boldsymbol{\pi}_u) &= \sum_{y_u, z_u} p(x_u, y_u, z_u|\omega, \boldsymbol{\phi}_u, \boldsymbol{\pi}_u) \\ &= \sum_{y_u, z_u} \prod_t p(z_{u,t}|z_{u,0:t-1}, y_{u,1:t}, \omega) p(y_{u,t}|\boldsymbol{\pi}_u) p(x_{u,t}|\boldsymbol{\phi}_{z_{u,t}}) \end{aligned} \tag{6.18}$$

Here ω is the union of parameters of all $\text{LSTM}_1, ..., \text{LSTM}_M$, the embedding matrix E, the projection matrix W_p and bias b_p. The structure of the generative process/probability function and importance of the switch variable y_t is better illustrated in Figure 6.12 through an example. Here each element of the observed sequence x_t is associated with a topic z_t and a switch variable y_t where y_t links (x_t, z_t) to the elements in the history with the same mode of behavior (theme). For instance in Figure 6.12, we have two themes in this sequence, one is given by topics with yellow shades (*e.g.* camping) and the other is defined by topics with blue shades

(*e.g.* baking deserts). The dynamic evolution of each theme is modeled using a different LSTM mixture. In that regard, the switch variable introduces large jumps in the sequence to connect relevant items so that data point (x_t, z_t) is generated conditioned only on elements from the same theme in the history $(x_{1,t-1}, z_{1,t-1})$, whereas LLA generates this element conditioned on all elements in the history. While in theory LSTM should be able to figure out these jumps, due to the diminishing gradient problem, and the fact that these jumps can occur weeks apart, the optimization algorithm in practice misses them as depicted in Figure 6.12 via the missing yellow topic in the second-to-last position.

Note that LLA is a special case of thLLA when the number of mixtures $M = 1$. thLLA still enjoys all good properties of LLA, namely sparsity, fewer parameters, and interpretability. In addition, the introduction of different dynamic mixtures adds another layer of interpretablility by breaking the data into a set of themes.

6.5.2 *Parameter estimation*

Similar to LLA, exact inference for thLLA is intractable due to complex dependences structure. As a result we have to resort to an approximate technique like mean field variation inference or stochastic EM. We begin by writing down, the variational lower bound to the marginal data likelihood is given by

$$\sum_u \log p(\mathbf{x}_u|\omega, \boldsymbol{\phi}, \boldsymbol{\pi}_u) \geqslant \sum_u \mathbb{E}_q \left[\log \frac{p(z_u|y_u, \omega)p(y_u|\boldsymbol{\pi}_u)p(x_u|z_u, \boldsymbol{\phi})}{q(\mathbf{y}_u, \mathbf{z}_u)}\right], \tag{6.19}$$

where q is the variational distribution. Following the (stochastic) EM approach, we iteratively maximize the lower bound with respect to q and the model parameters $(\omega, \pi, \boldsymbol{\phi})$, which leads to the following updates:

E-STEP : As before, the optimal variational distribution for each user u is given by the posterior:

$$q^\star(\mathbf{y}_u, \mathbf{z}_u) \propto p(\mathbf{z}_u|\mathbf{y}_u, \omega) \prod_t p(y_{u,t}|\boldsymbol{\pi}_u)p(x_t|\boldsymbol{\phi}_{z_{u,t}}). \tag{6.20}$$

Unlike Markov models for which efficient smoothing algorithms such as the forward-backward algorithm are available for computing the posterior expectations of sufficient statistics, in case of thLLA, although the forward messages can still be computed, the backward recursion can no longer evaluated or efficiently approximated.

S-STEP : As taking expectations is difficult, we take an alternative approach to collect posterior samples for every user u instead:

$$(\mathbf{y}_u, \mathbf{z}_u) \sim q^\star(\cdot, \cdot), \tag{6.21}$$

given only the filtering equations and SMC. Similar to LLA, we define the following quantities as *forward messages*, which can be computed in the standard forward pass:

$$\begin{aligned} \alpha_t &\equiv p(x_t|y_{1:t-1}, z_{1:t-1}) \propto \sum_{y_t, z_t} p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi}) \\ \gamma_t &\equiv p(y_t, z_t|z_{1:t-1}, y_{1:t-1}, x_t) \propto p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi}). \end{aligned} \tag{6.22}$$

As before, we use an SMC algorithm to approximate the posterior (6.20) with P-particles. We then can use SMC as in (6.1) with $\nu_t(z_{1:t}) = p(x_{1:t}, y_{1:t}, z_{1:t})$ (*i.e.* $z \leftarrow (y, z)$) being the unnormalized target distribution at time t.

Algorithm 6.4 Inference with Particle Gibbs for thLLA

Let $y_0^p = y_0, z_0^p = z_0$ and $\alpha_0^p = 1/P$ for $p = 1, \ldots, P$.
for $t = 1$ **to** T **do**
  Fix reference path: set $a_{t-1}^1 = 1$ and $y_{1:t}^1 = y_{1:t}^\star, z_{1:t}^1 = z_{1:t}^\star$ from previous iteration.
  for $p = 2$ **to** P **do**
    Sample ancestor $a_{t-1}^p \sim \text{Categorical}\left(\alpha_{t-1}^1, \alpha_{t-1}^2, ..., \alpha_{t-1}^P\right)$
    Sample particle $y_t^p, z_t^p \sim \gamma_t^p \propto p(z_t|z_{1:t-1}, y_{1:t}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi})$
    Set $y_{1:t}^p = (y_{1:t-1}^{a_{t-1}^p}, y_t^p)$, and $z_{1:t}^p = (z_{1:t-1}^{a_{t-1}^p}, z_t^p)$
    Compute the weight $\alpha_t^p \leftarrow \sum_{y_t, z_t} p(z_t|z_{1:t-1}^{a_{t-1}^p}, y_{1:t}^{a_{t-1}^p}, \omega)p(y_t|\pi)p(x_t|z_t, \boldsymbol{\phi})$
  end for
  Normalize the weights $\alpha_t^p \leftarrow \frac{\alpha_t^p}{\sum_{l=1}^P \alpha_t^l}$
end for
Sample $r \sim \alpha_T$, return the particle path $(y_{1:T}^{a_T^r}, z_{1:T}^{a_T^r})$.

For the choice of the proposal distribution $f(\cdot)$, as before, we chose the predictive distribution. This choice makes precisely one of the available forward messages in (6.22):

$$f^\star(y_t, z_t|z_{1:t-1}^{a_{t-1}^p}, y_{1:t-1}^{a_{t-1}^p}, x_t) = \gamma_t^p. \tag{6.23}$$

The computational cost for sampling from above at each time step is $O(M(K + H^2))$ where M is the number of mixtures, K is the number of topics (needed to compute $P(x_t|z_t)$) and H is the size of the LSTM hidden state (required to compute $P(z_t|z_{1:t-1})$). Finally M is required since we need to normalize the distribution across all possible mixture assignments for y_t. Indeed, with the choice of predictive distribution as the proposal density, the importance weight simplifies to the other forward message in (6.22), as before. Furthermore, to circumvent the issue of path-depletion, we employ Particle Gibbs, on the lines of LLA. The final algorithm is summarized in Algorithm 6.4.

M-STEP : For fixed switching and topic assignments (y, z), update the parameters ω, $\boldsymbol{\pi}$, and $\boldsymbol{\phi}$ so as to improve the likelihood in expectation. As the dependence of likelihood on LSTM parameters ω and $\boldsymbol{\pi}, \boldsymbol{\phi}$ are independent given y, z, we can update them independently and in parallel as follows:

$$\begin{aligned}
\omega &= \operatorname*{argmax}_\omega \sum_u \log p(z_u|y_u, \omega), && \Rightarrow \text{ solve approximately by SGD} \\
\boldsymbol{\pi}_u &= \operatorname*{argmax}_{\boldsymbol{\pi}_u} \sum_t \log p(y_{u,t}|\boldsymbol{\pi}_u) \ \ \forall u && \Rightarrow \ \pi_{u,j} = \frac{\#\{y_{u,t} = j\} + \alpha}{N_u + M\alpha} \ \ \forall u \\
\boldsymbol{\phi} &= \operatorname*{argmax}_{\boldsymbol{\phi}} \sum_u \sum_t \log p(x_{u,t}|\boldsymbol{\phi}_{z_{u,t}}) && \Rightarrow \ \phi_{x,k} = \frac{\#\{x_{u,t} = x \text{ and } z_{u,t} = k\} + \beta}{\#\{z_{u,t} = k\} + V\beta}.
\end{aligned} \tag{6.24}$$

The first equation basically implies training each LSTM with sequences z corresponding to the time when its active as specified by y. The other two equations are simply the MLE of switch variable, and the MLE of the given emission model, which is same as LDA or LLA. The full algorithm is summarized in Figure 6.13

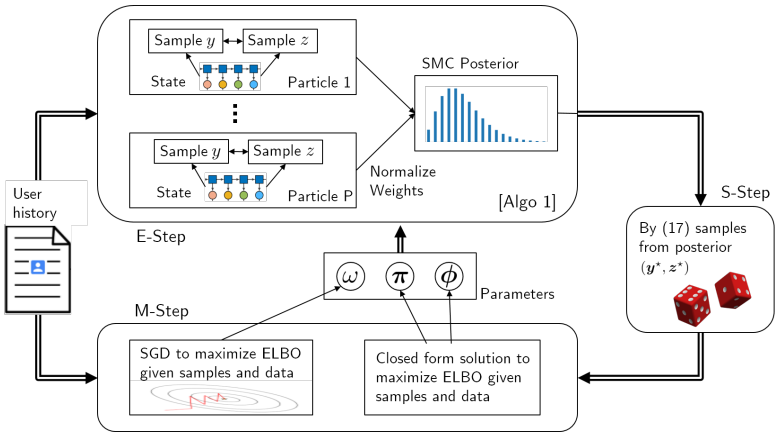


Figure 6.13: Overview of the Stochastic EM inference algorithm.

6.6 EXPERIMENTS ON THREADED LLA

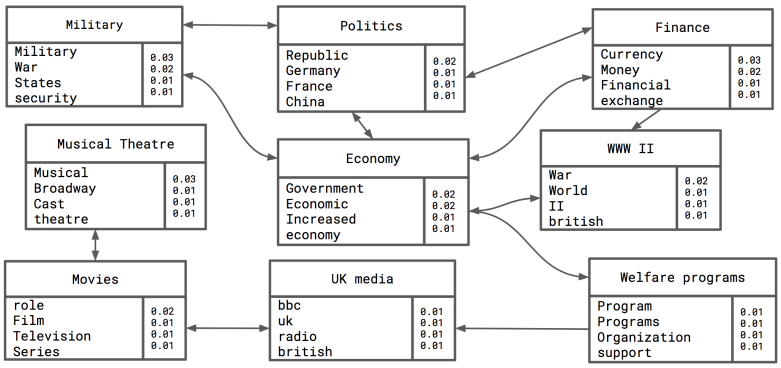
n this section we study empirically the properties of thLLA to establish that (1) it is flexible in capturing underlying nonlinear dynamics in sequences with mixed themes and abrupt jumps, (2) the inference algorithm is efficient (3) it produces cleaner and more interpretable topic transitions compared to LLA especially with the add layer of structure due to the dynamic mixture components.

6.6.1 *Setup and Datasets*

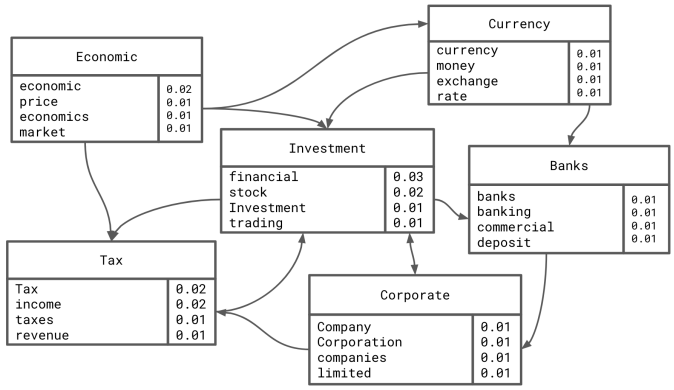
For all experiments we follow the standard setup for evaluating temporal models, *i.e.* we use the first 60% portion of the each sequence for training and then predict the remaining 40% of the sequence based on the representation inferred from the first 60%. We use perplexity as our metric (lower values are better) and compare against the baseline LLA. Since superiority of LLA against other temporal variants of topic models such as Andrews and Vigliocco (2010) and David Blei and Frazier (2011) has been established in Section 6.4, we **exclude** those comparison. Similarly we exclude comparisons against base LSTM as the same trade-off holds as in LLA. We focus here on the effect of the added layer of structure. Unless otherwise stated we run our experiments with number of topics $K = 1000$ and number of mixtures $M = 20$ (for thLLA). For both, we chose the dimensions of the input embedding to LSTM in the range $\{50, 100, 150\}$ and the size of the evolving hidden LSTM state in the range of $\{50, ..., 500\}$. For thLLA, we share the embeddings of topics among all LSTM in the mixtures, leaving each mixture to only learn the evolving hidden LSTM state. For SMC inference, we gradually increase the number of particles P from 1 to K during training, where K is the number of topics. Finally, all the algorithms are implemented on TensorFlow and run on machines with 4 GPUs.

To establish our claims we use data of different characteristics:

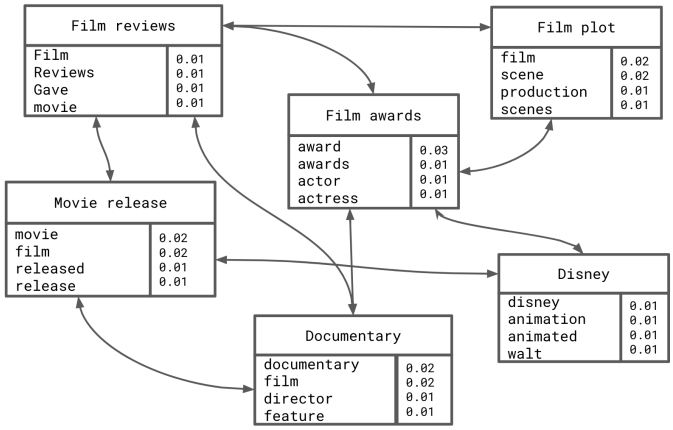
- **Wikipedia**: The publicly available Wikipedia dataset comprised of short documents containing 0.7 billion tokens and 1 million documents (Each document is a sequence). We chose the most frequent 50k words as a vocabulary. This domain is structured with each sequence having a few themes with no sharp transitions.
- **User Search History**: An anonymized sample of user search query history from a web search engine. For privacy reasons, we remove all queries appearing less than a given



(a) Base LLA transition graph

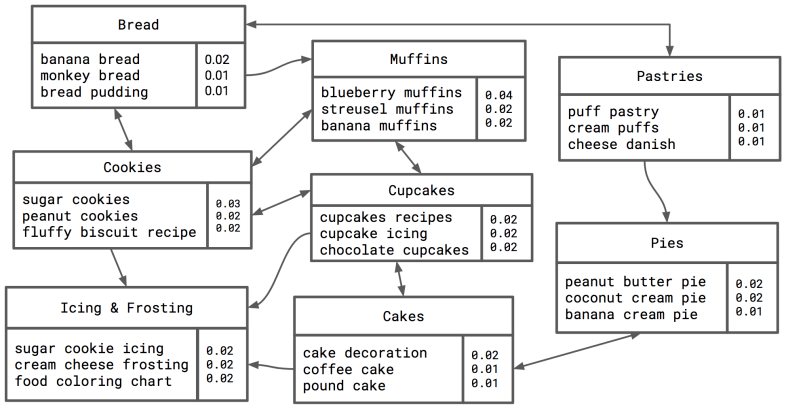


(b) thLLA transition graph for finance-related topics

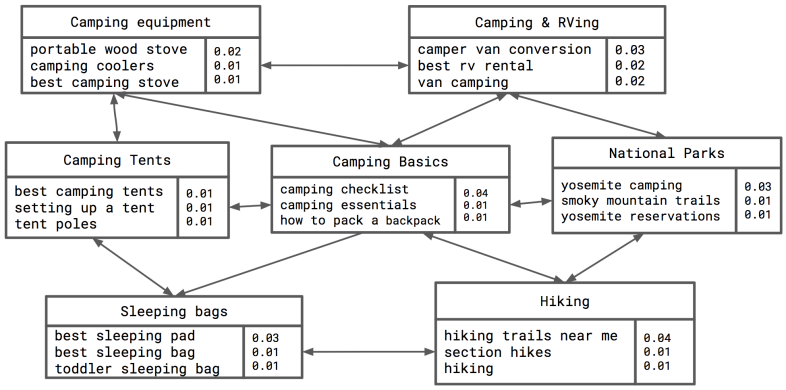


(c) thLLA transition graph for film-related topics

Figure 6.14: Transition graphs on Wikipedia dataset. Top row shows base LLA and bottom row depicts transition from two different thLLA mixtures about Movies and Finance. As evident thLLA gives cleaner representation. Probabilities are trimmed to the first two decimal digits.



(a) Transition graph for dessert-related topics



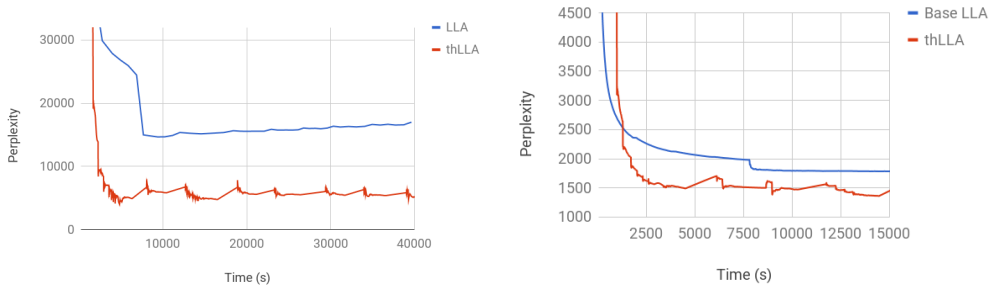
(b) Transition graph for outdoors-related topics

Figure 6.15: Transition graphs on user history dataset. Figure depicts transition from two different thLLA mixtures about Camping and Desert. As evident thLLA gives cleaner representation compared to LLA as show in Figure 6.11. Probabilities are trimmed to the first two decimal digits.

threshold, and use only head queries. This results in a dataset of 10M users and 1M vocabulary. This domain is less structured than the Wikipedia dataset with longer sequences (history of a given user), and abrupt jumps back and forth between themes in each sequence.

6.6.2 *Qualitative Evaluation: Topic Graph*

First we test the claim that thLLA gives cleaner and more interpretable representation compared to LLA. In Figure 6.14 and 6.15, we show the topic transition graph from both LLA and thLLA, each with 1000 topics and a comparable size of the LSTM(s) hidden state. For user search history dataset the LLA transition graph is shown in Figure 6.11. There are various ways to illustrate the dynamic behavior learned by the LSTM component(s) in each model. For instance in LLA, hierarchical clustering over the input topic embedding from the LSTM showed that topics that appears closer in time also appears closer in the hierarchy. Here we make this more explicit and visualize the dynamics learnt by each model as a topic transition graph using the one step transition from the LSTM components, *i.e.* for LLA we use $P_{LLA}(\text{topic}_t = k|\text{topic}_{t-1} = k', \text{LSTM parameters})$ and for mixture m from thLLA, we use $P_{thLLA}(\text{topic}_t = k|\text{topic}_{t-1} = k', \text{LSTM parameters of mixture } m)$. In the above figures, we



(a) User Search History (b) Wikipedia

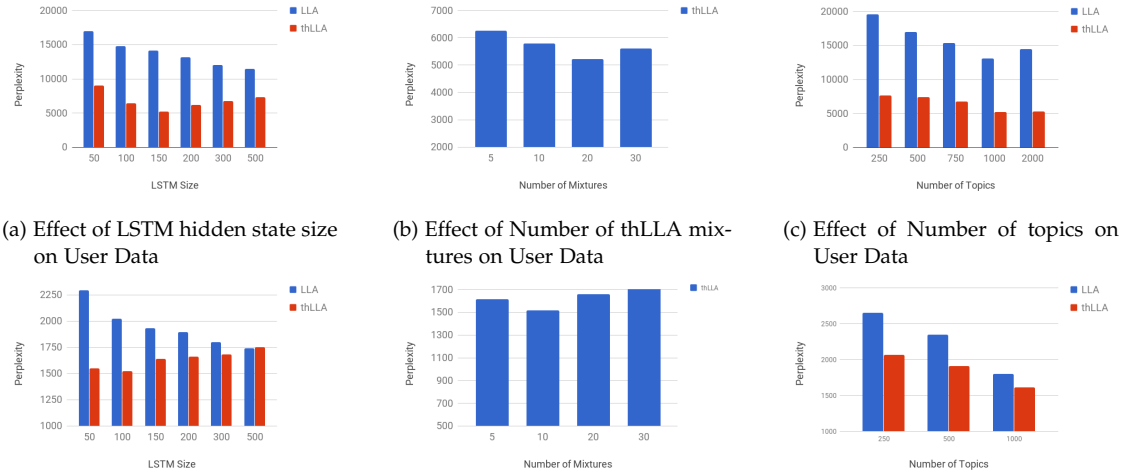
Figure 6.16: Convergence curves for LLA and thLLA on a) User history and b) Wikpedia (lower better)

show top words (queries) for each topic as well as top transitions. As evident, while LLA captures useful transition, the added layer of structure (themes) in thLLA results in a much cleaner models that uncovers fine-grained transitions such as between camping basics, equipments and parks, as can be seen in Figure 6.15(b), due to having a separate dynamic model for each mixture.

6.6.3 *Quantitative Evaluation*

ACCURACY AND PERPLEXITY Figure 6.16 compares LLA and thLLA quantitatively using held-out perplexity on both datasets. We use number of topics = 1000, for thLLA number of mixtures = 10 and we fix the size of the hidden state of each LSTM mixture to be 50 while for LLA we use a value of 500. We use the same size for topic embedding in all models (100) and we share topic embedding across all mixtures in thLLA making the overall size of the two models comparable. First, overall thLLA give better results in both domain, however, the improvement is more pronounced over the user search history dataset as expected due to its mixed-theme nature and abrupt transitions. Second, thLLA while being more complex still converges at the same rate compared to LLA. To understand this, note that the time complexity of LLA training is dominated by the M-Step due to the LSTM component. Since thLLA utilizes smaller-sized LSTMs in its mixtures, this compensates for the extra complexity introduced in the E-step to sample both the topic and mixture assignment for each item in the sequence.

EFFECT OF LSTM SIZE In this subsection we preform an ablation study to show the effect of the LSTM size on the thLLA and LLA. One might argue that if we increase the size of the evolving LSTM hidden state in LLA we can capture the same effect introduced by thLLA, *i.e.* the LSTM will figure it out due to its ability to model long-range dependencies. To test this hypothesis, we fix the number of topics to be 1000, the number of mixtures in thLLA to be 10, and LSTM topic embedding to be 100. We vary the size of the evolving hidden LSTM state in both LLA and each mixture in thLLA from 50 to 500. As shown in Figure 6.17(a,d), this is not the case, and in fact thLLA outperforms LLA across the entire range especially in the user history dataset. In the Wikipedia dataset, thLLA peaks at around hidden sizes = 50-100 while LLA peaks around 500, given that we use 10 mixtures for thLLA we see that at comparable LSTM sizes thLLA still outperform LLA. One should note here that while the LSTM in LLA can theoretically model very long-range dependencies, in practice this is hard due to the vanishing gradient problem, thus especially for user sequences, the fact that users go back and forth between themes over weeks poses a real challenge to LLA. In contrast, in thLLA, each mixture considers only a small portion of the input sequence that shares the same theme modeled by this mixture. Thus, two items might be adjacent in the input to the LSTM's mixture (thanks



(a) Effect of LSTM hidden state size on User Data

(b) Effect of Number of thLLA mixtures on User Data

(c) Effect of Number of topics on User Data

(d) Effect of LSTM hidden state size on Wikipedia Data

(e) Effect of Number of thLLA mixtures on Wikipedia Data

(f) Effect of Number of topics on Wikipedia Data

Figure 6.17: Ablation study comparing. From Left to right: effect of LSTM size, number of mixtures in thLLA, and number of topics. Top shows results on User Search History dataset and bottom shows results on the Wikipedia dataset.

to the inference in the E-Step) while in fact they are separated by many steps in the observed sequence. This ability of thLLA to model long-range dependencies via jumps, as explained in Section 6.5, is the key to its performance.

EFFECT OF NUMBER OF TOPICS In Figure 6.17(c,f) we show how varying the number of topics affect the quality of both models. We fix number of mixtures in thLLA to be 10 and use the best performing setting for LSTM configuration in each model. As shown in the figure thLLA outperform LLA across the entire topic range. For Wikipedia, we vary number of topics in the range {250, 500, 1000} which does not change the overall trend.

EFFECT OF NUMBER OF MIXTURES In Figure 6.17(b,e) we show how varying the number of mixtures in thLLA affects the model quality while fixing the number of topics to be 1000. As expected from the figure, this is a model selection problem where the performance peaks at 20 mixtures for the user search history dataset. However, it should be noted that overall thLLA is not very sensitive to slight variation around the optimal value. In practice, cross validation over a development dataset can be used to find the best value of the number of mixtures.

6.6.4 *A User Prediction Task*

In this subsection we provide an additional quantitative evaluation besides the standard perplexity metric using the user search history dataset. The task here is to predict users' future query. We divide queries in each test user into 80% for topic inference and 20% for evaluation. Each model produces a ranked list of queries based on the first 80% portion of the user history.

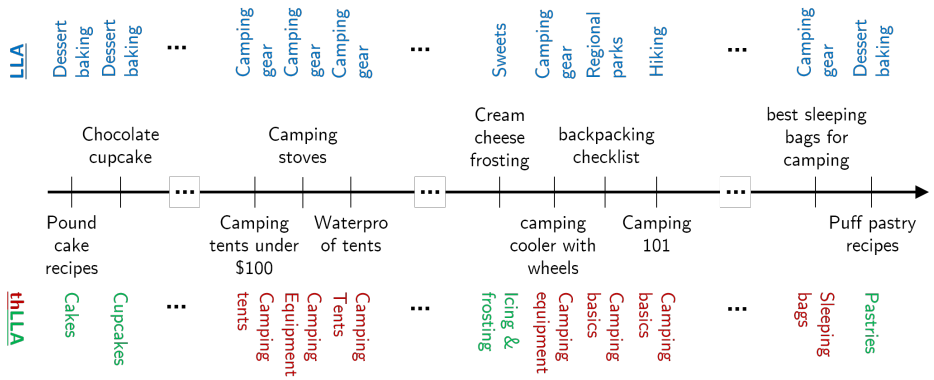


Figure 6.18: Illustrating how LLA and thLLA breaks a user history into topics. As evident, thLLA first breaks the history into themes: outdoor (red) and deserts (blue), then gives a more fine-grained annotation in each theme resulting in better prediction as shown in Table 6.4 (best viewed in color).

We use precision@K to evaluate the accuracy of each model in predicting future queries where $P(\text{Query} = q|\text{user history})$ is computed as follows:

$$\begin{aligned} P^{LLA}(q|u) &= \sum_{k=1}^{K} P^{LLA}(\text{topic} = k|u) P(q|k) \\ P^{thLLA}(q|u) &= \sum_{m=1}^{M} \pi_{u,m} \sum_{k=1}^{K} P^{thLLA,m}(\text{topic} = k|u) P(q|k) \end{aligned} \tag{6.25}$$

In other words, in LLA we first run inference and use the last state of the LSTM to produce a future topic distribution that we combine with the topic-query distribution to produce the predicted ranked list of queries. In thLLA we perform the same computation on a per-mixture basis where the contribution of each mixture is weighted by its prevalence in the user history given by $\pi_{u,m}$. The results are shown in Table 6.4.

As the table shows thLLA produced better results than LLA as it learns much more tighter topics that enables a more accurate prediction. To further explain this result, we refer the reader to Figure 6.18 where we show how LLA and thLLA segments a user history into topics. As evident from the figure, thLLA produces a fine-grained annotation of the user activities in both outdoor and desert-making themes. For instance, while LLA groups many events into "camping gear", thLLA breaks them down into camping tents and camping equipments. This breakage allows for a better future prediction as the scope of future queries is reduced from all possible queries in camping gears in addition to their immediate neighboring topics (as in LLA, Figure 6.11) to only those queries in tents and camping equipments in addition to their immediate neighboring topics (as in thLLA, Figure 6.15). This happens as thLLA devotes a

Model	P@5	P@10	P@15
LLA	0.174	0.151	0.139
thLLA	**0.219**	**0.192**	**0.183**

Table 6.4: Query prediction on the user search history dataset. Improvements are statistically significant.

whole mixture to modeling camping activities and this enables learning fine-grained topics with accurate, theme-focused transitions.

6.7 POINTS TO PONDER

In this chapter we present LLA: Latent LSTM allocation. LLA leverages the powerful dynamic modeling capability of LSTM without blowing up the number of parameters while adding interpretability to the model. We achieve this by shifting from modeling the temporal dynamics at the observed word level to modeling the dynamics at a higher level of abstraction: topics. As the number of topics K is much smaller than the number of words V, it can act as a knob that can trade-off accuracy vs model size. In the extreme case, if we allow $K \to V$ in LLA then we recover LSTM. Furthermore, the topics provide an informative embedding and transitions that can reveal interesting temporal relationship as shown in Figure 6.9a-6.9b and Figure 6.14-6.15, – which is a novel contribution to the best of our knowledge.

Inspite of nice interpretablity, it might seem from Figure 6.5,6.16 that LLA (and its variants) produce high perplexity compared to best known language models (Jozefowicz et al., 2016; Melis, Dyer, and Blunsom, 2017). Upon careful investigation, we found that the apparent gap arises due to different evaluation scheme. When perplexity is reported in literature for language models, entire test set is concatenated into a long text sequence. Over this long text sequence, LSTM or other language model is run to evaluate the perplexity. Whereas, in our task, concatenating different user history does not make sense. If we evaluate the best available language model[2] with 1 billion parameters in our setting for Wikipedia we get a perplexity of 983, which is in similar range to our small LLA models. Even in case of text, we raise the question is it meaningful to evaluate perplexity by concatenating all the documents? Or one should evaluate perplexity for each document independently then report the average perplexity. This because it is not clear what state should the LSTM carry over from one document to another.

LLA (and its variants) are more elaborate models, thus come with more knobs tune. This makes the training little unstable and needs careful hyper-parameter tuning to achieve good performance. We found the hack of initializing the word|topic matrix by running a few iterations of vanilla LDA helps a lot in generalization. The benefit of such initialization maybe because when LLA model is initialized randomly, the latent variables do not have any meaningful information. So the LSTM on the sequence of latent variable has hard time to start learning useful patterns and the overall model gets stuck. It would be worthwhile to come up with a general training procedure that can train a recurrent neural network on a sequence of latent variables, starting from scratch.

Finally, based on preliminary experimentation, we find the proposed LLA does not work very well on continuous input space such as images and speech data. We conjecture that the poor performance might be due to simple emission model of elementary parametric distribution like Gaussian or uniform. Other recent work in recurrent latent variable models where a recurrent model is endowed with a more flexible family of distributions such as restricted Boltzmann machines (RBM) (Boulanger-Lewandowski, Y. Bengio, and Vincent, 2012) or variational auto-encoders (VAE) (Chung et al., 2015; Gregor et al., 2015), perform better but are not as interpretable as LLA. However for our application of user modelling, interpretability is important and we need to focus on discrete data only, for which LLA meets the bill.

2 https://github.com/tensorflow/models/tree/master/research/lm_1b

Part III

SCALING UP INFERENCE BY EXPLOITING TOPOLOGY

Inferring interpretable representations from large-scale data is desirable, but often hindered by a mismatch between computational resources and statistical models. We aim to bridge this gap by again leveraging structure, albeit of a different kind. We will use a combination of modern computational techniques/data structures on one side and modified statistical inference algorithms on the other which exploit topological properties of the training objective. This introduces new ways to parallelize, reduce look-ups, handle variable state space size, and escape saddle points.

7

PARALLELIZATION

We study the problem of scaling up inference for latent variable models (LVM) by parallelization. The popularity of LVM rises from the ability to discover hidden thematic structures of the data in a human-understandable format. The rich statistical dependencies and intractable gradients of LVM, leave only variational methods or Markov chain Monte-Carlo (MCMC) for inference, which are inherently sequential. Moreover, these methods attempt to infer the complete posterior distribution whereas we are usually interested in some statistics of the posterior like maximum or expected value of the parameter under the posterior. In this chapter, instead of attempting to parallelize an inherently sequential algorithm, we start with a parallel computational paradigm and design update rules that give rise to correct inference algorithms for quantities that matter. We propose a model based on stochastic cellular automata (SCA), perhaps the most embarrassingly parallel computational model and good match for the *structure* of modern computational resources. This approach results in an embarrassingly parallel, memory efficient inference algorithm for LVM in which the complete data likelihood is in the exponential family. The algorithm converges to a valid *maximum a posteriori* fixed point and achieves better computational efficiency by exploiting the *structure* of the sufficient statistics of LVM with complete data likelihood is in the exponential family. Applied to latent Dirichlet allocation we find that our algorithm is over an order of magnitude faster than the fastest current approaches. A simple C++/MPI implementation on a 20-node Amazon EC2 cluster samples at more than 1 billion tokens per second. We process 3 billion documents and achieve predictive power competitive with collapsed Gibbs sampling and variational inference.

7.1 INTRODUCTION

In the past decade, frameworks such as stochastic gradient descent (SGD; Robbins and Monro 1951) and map-reduce (Dean and Ghemawat, 2008) have enabled machine learning algorithms to scale to larger and large datasets. However, these frameworks are not always applicable to Bayesian latent variable models with rich statistical dependencies and intractable gradients. Variational methods (Michael I. Jordan et al., 1999) and Markov chain Monte-Carlo (MCMC; Gilks, S. Richardson, and Spiegelhalter 1995) have thus become the *sine qua non* for inferring the posterior in these models.

Sometimes—due to the concentration of measure phenomenon associated with large sample sizes—computing the full posterior is unnecessary and *maximum a posteriori* (MAP) estimates suffice. It is hence tempting to employ gradient descent. However, for latent variable models such as latent Dirichlet allocation (LDA), calculating gradients involves expensive expectations over rich sets of variables (Patterson and Y. W. Teh, 2013).

MCMC is an appealing alternative, but traditional algorithms such as the Gibbs sampler are inherently sequential and the extent to which they can be parallelized depends heavily upon how the structure of the statistical model interacts with the data. For instance, chromatic sampling (Gonzalez et al., 2011) is infeasible for LDA, due to its dependence structure. We propose an

Emission/Mixture	Categorical	Dirichlet Mixture
Bernoulli	Latent Class Model (Formann and Kohlmann, 1996)	Grade of Membership Model (Woodbury, Clive, and Garson, 1978)
Multinomial	Unigram Document Clustering	Latent Dirichlet Allocation (D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003)
Gaussian	Mixture of Gaussians (R. Neal, 1998)	Gaussian-LDA (Section 5.5)
Poisson	-	GaP Model (Canny, 2004)

Table 7.1: Examples of some popular models to which ESCA is applicable.

alternate approach based on stochastic cellular automata (SCA). The automaton is massively parallel like conventional cellular automata, but employs stochastic updates.

Our proposed algorithm, exponential SCA (ESCA), is a specific way of mapping inference in latent variable models with complete data likelihood in the exponential family into an instance of SCA (Section 7.2). ESCA has a minimal memory footprint because it stores only the data and the minimal sufficient statistics (by the very definition of minimal sufficient statistics, the footprint cannot be further reduced). In contrast, variational approaches such as stochastic variational inference (SVI; M. D. Hoffman et al. 2013) require storing the variational parameters, while MCMC-based methods, such as YahooLDA (Alexander Smola and Narayanamurthy, 2010) require storing the latent variables. Thus, ESCA substantially reduces memory costs, enabling larger datasets to fit in memory, and significantly reducing communication costs in distributed environments.

Furthermore, the sufficient statistics dramatically improves efficiency. Typically, in the general case of SCA, updating a cell requires first assembling the values of all the neighboring cells before aggregating them into a local stochastic update. However, in ESCA, the sufficient statistics adequately summarize the states of the neighbors; the computational load is therefore small and perfectly balanced across the cells. As a result, ESCA addresses a key problem pervasive to other graph-based models in which the computation factorizes over vertices, such as traditional cellular automta and models employed by Pregel and GraphLab (Low et al., 2010; Malewicz et al., 2010). Such vertex-centric models have difficulty with natural graphs because low-degree vertices idle while high-degree vertices perform expensive computations.

We demonstrate how ESCA is a flexible framework for exponential latent variable models such as LDA (Section 7.3). In our experiments, we process over 3 billion documents at a rate of 1 billion tokens per second on a cluster of 20 commodity servers (Section 7.5). That said, ESCA is much more general. Table 7.1 explicitly lists some of the more common modeling choices for which ESCA can be easily employed. Our algorithm implicitly simulates stochastic expectation maximization (SEM), and is thus provably correct in the sense that it converges in distribution to a stationary point of the *posterior*.

7.2 EXPONENTIAL SCA

Stochastic cellular automata (SCA), also known as probabilistic cellular automata, or locally-interacting Markov chains, are a stochastic version of a discrete-time, discrete-space dynamical system in which a noisy local update rule is homogeneously and synchronously applied to every site of a discrete space. They have been studied in statistical physics, mathematics, and computer science, and some progress has been made toward understanding their ergodicity and equilibrium properties. A recent survey by Mairesse and Marcovici (2014) is an excellent introduction to the subject, and the dissertation by Louis (2002) contains a comprehensive and precise presentation of SCA.

The automaton, as a (stochastic) discrete-time, discrete-space dynamical system, is given by an evolution function $\Phi : \mathcal{S} \longrightarrow \mathcal{S}$ over the state space $\mathcal{S} = \mathcal{Z} \longrightarrow C$ which is a mapping from the space of cell identifiers Z to cell values C. The global evolution function applies a local function $\phi_z(c_1, c_2, \cdots, c_r) \mapsto c$ such that $c_i = s(z_i)$ to every cell $z \in \mathcal{Z}$. That is, ϕ examines the values of each of the neighbors of cell z and then stochastically computes a new value c. The dynamics begin with a state $s_0 \in S$ that can be configured using the data or a heuristic.

Exponential SCA (ESCA) is based on SCA but achieves better computational efficiency by exploiting the structure of the sufficient statistics for latent variable models in which the complete data likelihood is in the exponential family. Most importantly, the local update function ϕ for each cell depends only upon the sufficient statistics and thus does *not* scale linearly with the number of neighbors.

7.2.1 *Latent Variable Exponential Family*

Latent variable models are useful when reasoning about partially observed data such as collections of text or images in which each *i.i.d.* data point is a document or image. Since the same local model is applied to each data point, they have the following form

$$p(\mathbf{z}, \mathbf{x}, \eta) = p(\eta) \prod_i p\left(z_i, x_i | \eta\right). \tag{7.1}$$

Our goal is to obtain a MAP estimate for the parameters η that explain the data $\mathbf{x}$ through the latent variables $\mathbf{z}$. To expose maximum parallelism, we want each cell in the automaton to correspond to a data point and its latent variable. However, in general all latent variables depend on each other via the global parameters η, and thus the local evolution function ϕ would have to examine the values of every cell in the automaton.

Fortunately, if we further suppose that complete data likelihood is in exponential family, *i.e.*

$$p(z_i, x_i | \eta) = \exp\left(\left\langle T\left(z_i, x_i\right), \eta \right\rangle - g(\eta)\right) \tag{7.2}$$

then the complete and sufficient statistics are given by

$$T(\mathbf{z}, \mathbf{x}) = \sum_i T(z_i, x_i) \tag{7.3}$$

and we can thus express any estimator of interest as a function of just $T(\mathbf{z}, \mathbf{x})$. Further, when employing expectation maximization (EM), the M-step is possible in closed form for many members of the exponential family. This allows us to reformulate the cell level updates to depend only upon the sufficient statistics instead of the neighboring cells. The idea is that, unlike SCA (or MCMC in general) which produces a sequence of states that correspond to complete variable assignments $s^0, s^1, \ldots$ via a transition kernel $q(s^{t+1}|s^t)$, ESCA produces a sequence of sufficient statistics $T^0, T^1, \ldots$ directly via an evolution function $\Phi(T^t) \mapsto T^{t+1}$.

7.2.2 *Stochastic EM*

Before we present ESCA, we must first describe stochastic EM (SEM). Suppose we want the MAP estimate for η and employ a traditional expectation maximization (EM) approach:

$$\max_{\eta} p(\mathbf{x}, \eta) = \max_{\eta} \int p(\mathbf{z}, \mathbf{x}, \eta) \mu(d\mathbf{z}) \tag{7.4}$$

EM finds a mode of $p(\mathbf{x}, \eta)$ by iterating two steps:

E-STEP Compute in parallel $p(z_i|x_i, \eta^{(t)})$.

M-STEP Find $\eta^{(t+1)}$ that maximizes the expected value of the log-likelihood with respect to the conditional probability, i.e.

$$\begin{aligned} \eta^{(t+1)} &= \operatorname*{argmax}_{\eta} \mathbb{E}_{\mathbf{z}|\mathbf{x},\eta^{(t)}}[\log p(\mathbf{z}, \mathbf{x}, \eta)] \\ &= \xi^{-1}\left(\frac{1}{n + n_0} \sum_i \mathbb{E}_{\mathbf{z}|\mathbf{x},\eta^{(t)}}[T(z_i, x_i)] + T_0\right) \end{aligned} \tag{7.5}$$

where $\xi(\eta) = \nabla g(\eta)$ is invertible as $\nabla^2 g(\theta) \succ 0$ and n_0, T_0 parametrize conjugate prior. [1]

Although EM exposes substantial parallelism, it is difficult to scale, since the dense structure $p(z_i|x_i, \eta^{(t)})$ defines values for all possible outcomes for z and thus puts tremendous pressure on memory bandwidth.

To overcome this we introduce sparsity by employing stochastic EM (SEM; Celeux and Diebolt 1985). SEM substitutes the E-step for an S-step that replaces the full distribution with a single sample:

S-STEP Sample $z_i^{(t)} \sim p(z_i|x_i; \eta^{(t)})$ in parallel.

Subsequently, we perform the M-step using the imputed data instead of the expectation. This simple modification overcomes the computational drawbacks of EM for cases in which sampling from $p(z_i|x_i; \eta^{(t)})$ is feasible. We can now employ fast samplers, such as the alias method, exploit sparsity, reduce CPU-RAM bandwidth while still maintaining massive parallelism.

The S-step has other important consequences. Notice that the M-step is now a simple function of current sufficient statistics. This implies that the conditional distribution for the next S-step is expressible in terms of the complete sufficient statistics

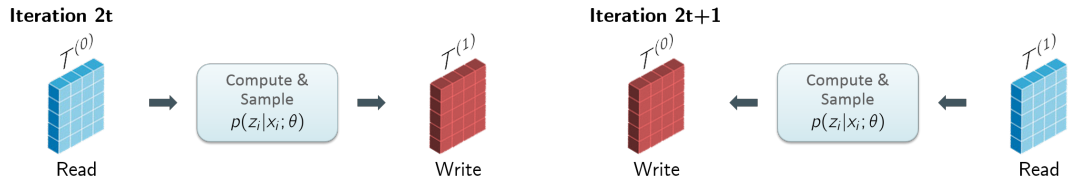
$$p(z_i|x_i; \eta^{(t)}) = f(z_i, T(\mathbf{x}, \mathbf{z}^{(t)})). \tag{7.6}$$

Thus each S-step depends only upon the sufficient statistics generated by the previous step. Therefore, we can operate directly on sufficient statistics without the need to assign or store latent variables/states. Moreover it opens up avenues for distributed and parallel implementations that execute on an SCA.

7.2.3 *ESCA for Latent Variable Models*

SEM produces an alternating sequence of S and M steps in which the M step produces the parameters necessary for the next S step. Since we can compute these parameters on the fly there is no need for an explicit M step. Instead, ESCA produces a sequence consisting only of S steps. We require the exponential family to ensure that these steps are both efficient and compact. We now present ESCA more formally.

1 The inversion may not be always available in closed form, in which case we resort to numerical techniques.



(a) Phase 1 (b) Phase 2

Figure 7.1: Efficient (re)use of buffers

Define an SCA over the state space $\mathcal{S}$ of the form:

$$\mathcal{S} = \mathcal{Z} \longrightarrow \mathcal{K} \times \mathcal{X} \tag{7.7}$$

where $\mathcal{Z}$ is the set of cell identifiers (*e.g.* one per token in a text corpus), $\mathcal{K}$ is the domain of latent variables, and $\mathcal{X}$ is the domain of the observed data.

The initial state s_0 is the map defined as follows: for every data point, we associate a cell z to the pair (k_z, x) where k_z is chosen at random from $\mathcal{K}$ and independently from $k_{z'}$ for all $z' \neq z$. This gives us the initial state s_0.

$$s_0 = z \mapsto (k_z, x) \tag{7.8}$$

We now need to describe the evolution function Φ. First, assuming that we have a state s and a cell z, we define the following distribution:

$$p_z(k|s) = f(z, T(s)) \tag{7.9}$$

Assuming that $s(z) = (k, x)$ and that k' is a sample from p_z (hence the name "stochastic" cellular automaton) we define the local update function as:

$$\phi(s, z) = (k', x) \quad \text{where} \quad s(z) = (k, x) \quad \text{and} \quad k' \sim p_z(\,\cdot\,|s) \tag{7.10}$$

That is, the observed data remain unchanged, but we choose a new latent variable according to the distribution p_z induced by the state. We obtain the evolution function of the stochastic cellular automaton by applying the function ϕ uniformly on every cell.

$$\Phi(s) = z \mapsto \phi(s, z) \tag{7.11}$$

Finally, the SCA algorithm simulates the evolution function Φ starting with s_0.

As explained earlier, due to our assumption of complete data likelihood belonging to the exponential family, we never have to represent the states explicitly, and instead employ the sufficient statistics.

An implementation can, for example, have two copies of the data structure containing sufficient statistics $T^{(0)}$ and $T^{(1)}$. We do not compute the values $T(\mathbf{z}, \mathbf{x})$ but keep track of the sum as we impute values to the cells/latent variables. During iteration $2t$ of the evolution function, we apply Φ by reading from $T^{(0)}$ and incrementing $T^{(1)}$ as we sample the latent variables (See Figure 7.1). Then in the next iteration $2t + 1$ we reverse the roles of the data structures, *i.e.* read from $T^{(1)}$ and increment $T^{(0)}$. We summarize in Algorithm 7.1.

Use of such read/write buffers offer a virtually lock-free (assuming atomic increments) implementation scheme for ESCA and is analogous to double-buffering in computer graphics.

Algorithm 7.1 ESCA

```
Randomly initialize each cell
for t = 0 → num iterations do
    for cell z independently in parallel do do
        Read sufficient statistics from T^(t mod 2)
        Compute stochastic updates using p_z(k|s)
        Write sufficient statistics to T^(t+1 mod 2)
    end for
end for
```

Although there is a synchronization barrier after each round, its effect is mitigated because each cell's work depends only upon the sufficient statistics and thus does the same amount of work. Therefore, evenly balancing the work load across computation nodes is trivial, even for a heterogeneous cluster.

Furthermore, in the case of discrete latent variable, updating sufficient statics only requires increments to the data structure $T^{(r)}$ allowing the use of approximate counters (Csűrös, 2010; Morris, 1978). Approximate counters greatly reduce memory costs for the counts: *e.g.* only 4 or 8 bits per counter. Recent empirical evidence demonstrates that approximate counters preserve statistical performance without compromising runtime performance (Tristan, Tassarotti, and Steele Jr., 2015). In fact, speed often increases because not every increment to the counter results in a write to memory. Note, due to the compression, maintaining two buffers requires less memory than one. Finally, if the latent variables are discrete valued then we can leverage the fast Vose' alias method (Vose, 1991) to sample. The $O(|\mathcal{K}|)$ construction cost for the alias method can be easily amortized because the rule is homogeneous and thus alias tables can be shared. Details about alias sampling method is provided in Section 7.4.4.

7.2.4 *Wide Applicability of ESCA*

As stated previously, ESCA is technically applicable to any model in which the complete data likelihood is in the exponential family. Designing an ESCA algorithm for a model of interest requires simply deriving the S-step for the local update function in the automaton. The S-step is the full conditional (7.9) which is easy to derive for many models; *e.g.* mixture models in which (1) the data and parameter components are conjugate and (2) the latent variables and priors are conjugate. We list a few examples of such models in Table 7.1 and provide additional details in Section 7.4.5. Of course, the extent to which the models enable exploitation of sparsity varies.

7.2.5 *Convergence*

We now address the critical question of how the invariant measure of ESCA for the model presented in Section 7.2.1 is related to the true MAP estimates. First, note that SCA is ergodic (Louis, 2002), a result that immediately applies if we ignore the deterministic components of our automata (corresponding to the observations). Now that we have established ergodicity, we next study the properties of the stationary distribution and find that the modes correspond to MAP estimates.

We make a few mild assumptions about the model:

- The observed data Fisher information is non-singular, i.e. $I(\eta) \succ 0$.

- For the Fisher information for $\mathbf{z}|\mathbf{x}$, we need it to be non-singular and central limit theorem, law of large number to hold, i.e. $\mathbb{E}_{\eta_0}[I_X(\eta_0)] \succ 0$ and

$$\sup_{\eta} \left| \frac{1}{n} \sum_{i=1}^{n} I_{x_i}(\eta) - \mathbb{E}_{\eta_0}[I_X(\eta)] \right| \to 0 \text{ as } n \to \infty$$

- We assume that $\frac{1}{n} \sum_{i=1}^{n} \nabla_\eta \log p(x_i; \eta) = 0$ has at least one solution, let $\hat{\eta}$ be a solution.

These assumptions are reasonable. For example in case of mixture models (or topic models), it just means all component must be exhibited at least once and all components are unique. The details of this case are worked out in Section 7.4.3. Also when the number of parameters grow with the data, *e.g.* for topic models, the second assumption still holds. In this case, we resort to corresponding result from high dimensional statistics by replacing the law of large numbers with Donsker's theorem and everything else falls into place.

Consequently, we show ESCA converges weakly to a distribution with mean equal to some root of the score function ($\nabla_\eta \log p(x_i; \eta)$) and thus a MAP fixed point by borrowing the results known for SEM (Nielsen, 2000). In particular, we have:

Theorem 7.1 *Let the assumptions stated above hold and $\tilde{\eta}$ be the estimate from ESCA. Then as the number of independent and identically distributed data point goes to infinity,* i.e. $n \to \infty$, *we have*

$$\sqrt{n}(\tilde{\eta} - \hat{\eta}) \xrightarrow{\mathcal{D}} \mathcal{N}\left(0, I(\eta_0)^{-1}[I - F(\eta_0)^{-1}\right) \tag{7.12}$$

where $F(\eta_0) = I + \mathbb{E}_{\eta_0}[I_X(\eta_0)](I(\eta_0) + \mathbb{E}_{\eta_0}[I_X(\eta_0)])$.

This result implies that SEM flocks around a stationary point under very reasonable assumptions and tremendous computational benefits. Also, for such complicated models, reaching a stationary point is the best that most methods achieve anyway. Now we switch gears to adopt ESCA for LDA and perform some simple experimental evaluations.

7.3 ESCA FOR LDA

Topic modeling, and latent Dirichlet allocation (LDA) (D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003) in particular, have become a must-have of analytics platforms and consequently needs to scale to larger and larger datasets. In LDA, we model each document m of a corpus of M documents as a distribution θ_m that represents a mixture of topics. There are K such topics, and we model each topic k as a distribution ϕ_k over the vocabulary of words that appear in our corpus. Each document m contains N_m words w_{mn} from a vocabulary of size V, and we associate a latent variable z_{mn} to each of the words. The latent variables can take one of K values indicating the topic for the word. Both distributions θ_m and ϕ_k have a Dirichlet prior, parameterized respectively with a constant α and β. See Section 7.4.1 for more details.

7.3.1 *Existing systems*

Many of the scalable systems for topic modeling are based on one of two core inference methods: the collapsed Gibbs sampler (CGS; T.L. Griffiths and Steyvers 2004), and variational inference (VI; D. M. Blei, A. Y. Ng, and Michael I. Jordan 2003) and approximations thereof (Asuncion et al., 2009). To scale LDA to large datasets, or for efficiency reasons, we may need to distribute and parallelize them. Both algorithms can be further approximated to meet such implementation requirements.

COLLAPSED GIBBS SAMPLING In collapsed Gibbs sampling the full conditional distribution of the latent topic indicators given all the others is

$$p(z_{mn} = k|\boldsymbol{z}^{\neg mn}, \boldsymbol{w}) \propto \boxed{(D_{mk} + \alpha)\frac{W_{kw_{mn}} + \beta}{T_k + \beta V}} \tag{7.13}$$

where D_{mk} is the number of latent variables in document m that equal k, W_{kv} is the number of latent variables equal to k and whose corresponding word equals v, and T_k is the number of latent variables that equal k, all excluding current z_{mn}.

CGS is a sequential algorithm in which we draw latent variables in turn, and repeat the process for several iterations. The algorithm performs well statistically, and has further benefited from breakthroughs that lead to a reduction of the sampling complexity (A. Q. Li et al., 2014; L. Yao, Mimno, and Andrew McCallum, 2009). This algorithm can be approximated to enable distribution and parallelism, primarily in two ways. One is to partition the data, perform one sampling pass and then assimilate the sampler states, thus yielding an approximate distributed version of CGS (AD-LDA; Newman, Asuncion, et al. 2009). Another way is to partition the data and allow each sampler to communicate with a distributed central storage continuously. Here, each sampler sends the differential to the global state-keeper and receives from it the latest global value. A scalable system built on this principle and leveraging inherent sparsity of LDA is YahooLDA (Alexander Smola and Narayanamurthy, 2010). Further improvement and sampling using alias table was incorporated in lightLDA (Yuan et al., 2015). Contemporaneously, a nomadic distribution scheme and sampling using Fenwick tree was proposed in F+LDA (H.-F. Yu et al., 2015).

VARIATIONAL INFERENCE In variational inference (VI), we seek to optimize the parameters of an approximate distribution that assumes independence of the latent variables to find a member of the family that is close to the true posterior. Typically, for LDA, document-topic proportions and topic indicators are latent variables and topics are parameter. Then, coordinate ascent alternates between them.

One way to scale VI is stochastic variational inference (SVI) which employs SGD by repeatedly updating the topics via randomly chosen document subsets (M. D. Hoffman et al., 2013). Adding a Gibbs step to SVI introduces sparsity for additional efficiency (Mimno, M. Hoffman, and David Blei, 2012). In some ways this is analogous to our S-step, but in the context of variational inference, the conditional is much more expensive to compute, requiring several rounds of sampling.

Another approach, CVBo, achieves scalability by approximating the collapsed posterior (W. Y. Teh, Newman, and Welling, 2007). Here, they minimize the free energy of the approximate distribution for a given parameter γ_{mnk} and then use the zero-order Taylor expansion (Asuncion et al., 2009).

$$\gamma_{mnk} \propto \boxed{(D_{mk} + \alpha) \times \frac{W_{kw_{mn}} + \beta}{T_k + \beta\ V}} \tag{7.14}$$

where D_{mk} is the fractional contribution of latent variables in document m for topic k, W_{kv} is the contribution of latent variables for topic k and whose corresponding word equals v, and T_k is the the contribution of latent variables for topic k. Inference updates the variational parameters until convergence. It is possible to distribute and parallelize CVBo over tokens (Asuncion et al., 2009). VI and CVBo are the core algorithms behind several scalable topic modeling systems including Mr.LDA (Zhai, J. Boyd-Graber, et al., 2012) and the Apache Spark machine-learning suite.

REMARK It is worth noticing that Gibbs sampling and variational inference, despite being justified very differently, have at their core the very same formulas (shown in a box in formula (7.13) and (7.14)). Each of which are literally deciding how important is some topic k to the word v appearing in document m by asking the questions: "How many times does topic k occur in document m?", "How many times is word v associated with topic k?", and "How prominent is topic k overall?". It is reassuring that behind all the beautiful mathematics, something simple and intuitive is happening. As we see next, ESCA addresses the same questions via analogous formulas for SEM.

7.3.2 *An ESCA Algorithm for LDA*

To re-iterate, the point of using such a method for LDA is that the parallel update dynamics of the ESCA gives us an algorithm that is simple to parallelize, distribute and scale. In the next section, we will evaluate how it works in practice. For now, let us explain how we design our SCA to analyze data.

We begin by writing the stochastic EM steps for LDA (derivation is in Section 7.4.1):

E-STEP: independently in parallel compute the conditional distribution locally:

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} \tag{7.15}$$

S-STEP: independently in parallel draw z_{ij} from the categorical distribution:

$$z_{mn} \sim \mathsf{Categorical}(q_{mn1}, \ldots, q_{mnK}) \tag{7.16}$$

M-STEP: independently in parallel compute the new parameter estimates:

$$\begin{aligned}\theta_{mk} &= \frac{D_{mk} + \alpha - 1}{N_m + K\alpha - K}\\ \phi_{kv} &= \frac{W_{kv} + \beta - 1}{T_k + V\beta - V}\end{aligned} \tag{7.17}$$

We simulate these inference steps in ESCA, which is a dynamical system with evolution function $\Phi : \mathcal{S} \longrightarrow \mathcal{S}$ over the state space $\mathcal{S}$. For LDA, the state space $\mathcal{S}$ is

$$\mathcal{S} = \mathcal{Z} \longrightarrow \mathcal{K} \times \mathcal{M} \times \mathcal{V} \tag{7.18}$$

where $\mathcal{Z}$ is the set of cell identifiers (one per token in our corpus), $\mathcal{K}$ is a set of K topics, $\mathcal{M}$ is a set of M document identifiers, and $\mathcal{V}$ is a set of V identifiers for the vocabulary words.

The initial state s_0 is the map defined as follows: for every occurrence of the word v in document m, we associate a cell z to the triple (k_z, m, v) where k_z is chosen uniformly at random from $\mathcal{K}$ and independently from $k_{z'}$ for all $z' \neq z$. This gives us

$$s_0 = z \mapsto (k_z, m, v) \tag{7.19}$$

We now need to describe the evolution function Φ. First, assuming that we have a state s and a cell z, we define the following distribution:

$$p_z(k|s) \propto \boxed{(D_{mk} + \alpha) \times \frac{W_{kv} + \beta}{T_k + \beta V}} \tag{7.20}$$

where $D_{mk} = \left|\left\{ z \mid \exists v.\ s(z) = (k, m, v) \right\}\right|$,
$W_{kv} = \left|\left\{ z \mid \exists m.\ s(z) = (k, m, v) \right\}\right|$, and
$T_k = \left|\left\{ z \mid \exists m.\ \exists v.\ s(z) = (k, m, v) \right\}\right|$. Note that we have chosen our local update rule slightly different without an offset of -1 for the counts corresponding to the mode of the Dirichlet distributions and requiring $\alpha, \beta > 1$. Instead, our local update rule allows us to have the relaxed requirement $\alpha, \beta > 0$ which is more common for LDA inference algorithms.

Assuming that $s(z) = (k, m, v)$ and that k' is a sample from p_z (hence the name "stochastic" cellular automaton) we define the local update function as:

$$\begin{aligned} \phi(s, z) &= (k', m, v) \\ \text{where} \quad s(z) &= (k, m, v) \quad \text{and} \quad k' \sim p_z(\cdot \mid s) \end{aligned} \tag{7.21}$$

That is, the document and word of the cell remain unchanged, but we choose a new topic according to the distribution p_z induced by the state. We obtain the evolution function of the stochastic cellular automaton by applying the function ϕ uniformly on every cell.

$$\Phi(s) = z \mapsto \phi(s, z) \tag{7.22}$$

Finally, the SCA algorithm simulates the evolution function Φ starting with s_0. Of course, since LDA's complete data likelihood is in the exponential family, we never have to represent the states explicitly, and instead employ the sufficient statistics.

Our implementation has two copies of the count matrices D^i, W^i, and T^i for $i = 0$ or 1 (as in CGS or CVB0, we do not compute the values D_{ik}, W_{kv}, and T_k but keep track of the counts as we assign topics to the cells/latent variables). During iteration i of the evolution function, we apply Φ by reading $D^{i \bmod 2}$, $W^{i \bmod 2}$, and $T^{i \bmod 2}$ and incrementing $D^{i+1 \bmod 2}$, $W^{i+1 \bmod 2}$, and $T^{i+1 \bmod 2)}$ as we assign topics.

7.3.3 *Advantages of ESCA for LDA*

The positive consequences of ESCA as a choice for inference on LDA are many:

- Our memory footprint is minimal since we only store the data and sufficient statistics. In contrast to MCMC methods, we do not store the assignments to latent variables $\mathbf{z}$. In contrast to variational methods, we do not store the variational parameters γ. Further, variational methods require K memory accesses (one for each topic) per word. In contrast, the S-step ensures we only have a single access (for the sampled topic) per word. Such reduced pressure on the memory bandwidth can improve performance significantly for highly parallel applications.
- We can further reduce the memory footprint by compressing the sufficient statistics with approximate counters (Csűrös, 2010; Morris, 1978). This is possible because updating the sufficient statistics only requires increments as in Mean-for-Mode (Tristan, Tassarotti, and Steele Jr., 2015). In contrast, CGS decrements counts, preventing the use of approximate counters.
- Our implementation is lock-free (in that it does not use locks, but assumes atomic increments) because the double buffering ensures we never read or write to the same data structures. There is less synchronization, which at scale is significant.

- Finally, our algorithm is able to fully benefit from Vose's alias method (Vose, 1991) because homogeneous update rule for SCA ensures that the cost for constructing the alias table is amortized across the cells. To elaborate, the SCA update (7.20) decomposes as

$$p_z(k|s) \propto \left[D_{mk}\frac{W_{kv}+\beta}{T_k+\beta V}\right] + \left[\alpha\frac{W_{kv}+\beta}{T_k+\beta V}\right] \tag{7.23}$$

  allowing us to treat it as a discrete mixture and divide the sampling procedure into a two steps. First, we toss a biased coin to decide which term of the equation to sample, and second, we employ a specialized sampler depending on the chosen term. The first term is extremely sparse (documents comprise only a small handful of topics) and a basic sampling procedure suffices. The second term is not sparse, but is independent of the current document m and depends only on the W and T matrices. Moreover, as mentioned earlier, during iteration i, we will be only reading values from non-changing $W^{i \mod 2}$, and $T^{i \mod 2}$ matrices. As a result, at the start of each iteration we can precompute, from the W and T matrices, tables for use with Vose's alias method, which enables sampling from the second term in a mere 3 CPU operations. Thus, the evolution for ESCA is extremely efficient.

7.3.3.1 *Connection to SGD*

We can view ESCA as implicit SGD on MAP for LDA. This connection alludes to the convergence rate of ESCA. To illustrate, we consider θ only. As pointed out in R. Salakhutdinov, S. Roweis, and Zoubin Ghahramani (n.d.) and L. Xu and Michael I Jordan (1996), one EM step is:

$$\theta_m^+ = \theta_m + M\frac{\partial \log p}{\partial \theta_{mk}}$$

which is gradient descent with a Frank-Wolfe update and line search. Similarly, for ESCA using stochastic EM, one step is

$$\theta_{mk}^+ = \frac{Dn_{mk}}{N_m} = \frac{1}{N_m}\sum_{n=1}^{N_m}\delta(z_{mn}=k)$$

Again vectorizing and re-writing as earlier:

$$\theta_m^+ = \theta_m + Mg$$

where $M = \frac{1}{N_m}\left[\mathrm{diag}(\theta_m) - \theta_m\theta_m^T\right]$ and $g = \frac{1}{\theta_{mk}}\sum_{n=1}^{N_m}\delta(z_{mn}=k)$. The vector g can be shown to be an unbiased noisy estimate of the gradient, *i.e.*

$$\mathbb{E}[g] = \frac{1}{\theta_{mk}}\sum_{n=1}^{N_i}\mathbb{E}[\delta(z_{ij}=k)] = \frac{\partial \log p}{\partial \theta_{mk}}$$

Thus, a single step of SEM on our SCA is equivalent to a single step of SGD. Consequently, we could further embrace the connection to SGD and use a subset of the data for the S and M steps, similar to incremental EM (R. M. Neal and Geoffrey E Hinton, 1998). Further details are provided in Section 7.4.2. Note that in the limit in which batches comprise just a single token, the algorithm emulates a collapsed Gibbs sampler. This interpretation strengthens the theoretical justification for many existing approximate Gibbs sampling approaches.

7.4 FURTHER DETAILS AND DERIVATIONS

7.4.1 *(S)EM Derivation for LDA*

We derive an EM procedure for LDA.

7.4.1.1 *LDA Model*

In LDA, we model each document m of a corpus of M documents as a distribution θ_m that represents a mixture of topics. There are K such topics, and we model each topic k as a distribution ϕ_k over the vocabulary of words that appear in our corpus. Each document m contains N_m words w_{mn} from a vocabulary of size V, and we associate a latent variable z_{mn} to each of the words. The latent variables can take one of K values that indicate which topic the word belongs to. We give each of the distributions θ_m and ϕ_k a Dirichlet prior, parameterized respectively with a constant α and β. More concisely, LDA has the following mixed density.

$$p(\boldsymbol{w}, \boldsymbol{z}, \theta, \boldsymbol{\phi}) = \left[\prod_{m=1}^{M}\prod_{n=1}^{N_m} \mathsf{Categorical}(w_{mn} \mid \phi_{z_{mn}})\ \mathsf{Categorical}(z_{mn} \mid \theta_m)\right]\left[\prod_{m=1}^{M} \mathsf{Dir}(\theta_m \mid \alpha)\right]\left[\prod_{k=1}^{K} \mathsf{Dir}(\phi_k \mid \beta)\right] \tag{7.24}$$

The choice of a Dirichlet prior is not a coincidence: we can integrate all of the variables θ_m and ϕ_k and obtain the following closed form solution.

$$p(\boldsymbol{w}, \boldsymbol{z}) = \left[\prod_{m=1}^{M} \mathsf{Pol}\big(\{z_{m'n} \mid m' = m\}, K, \alpha\big)\right]\left[\prod_{k=1}^{K} \mathsf{Pol}\big(\{w_{mn} \mid z_{mn} = k\}, V, \beta\big)\right] \tag{7.25}$$

where Pol is the Polya distribution

$$\mathsf{Pol}(S, X, \eta) = \frac{\Gamma(\eta K)}{\Gamma(|S| + \eta X)} \prod_{x=1}^{X} \frac{\Gamma\big(\big|\{z \mid z \in S, z = x\}\big| + \eta\big)}{\Gamma(\eta)} \tag{7.26}$$

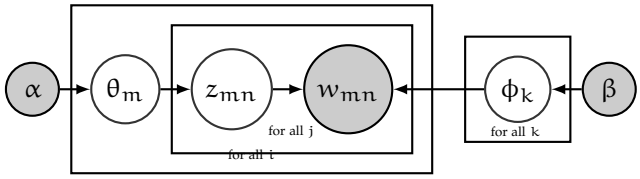


Figure 7.2: LDA Graphical Model

The joint probability density can be expressed as:

$$\begin{aligned} p(W, Z, \theta, \phi|\alpha, \beta) &= \left[\prod_{k=1}^{K} p(\phi_k|\beta)\right]\left[\prod_{m=1}^{M} p(\theta_m|\alpha) \prod_{n=1}^{N_m} p(z_{mn}|\theta_m)p(w_{mn}|\phi_{z_{mn}})\right] \\ &\propto \left[\prod_{k=1}^{K}\prod_{v=1}^{V} \phi_{kv}^{\beta-1}\right]\left[\prod_{m=1}^{M}\left(\prod_{k=1}^{K} \theta_{mk}^{\alpha-1}\right)\prod_{n=1}^{N_m} \theta_{mz_{mn}}\phi_{z_{mn}w_{mn}}\right] \end{aligned} \tag{7.27}$$

Algorithm 7.2 LDA Generative Model

input: α, β

for $k = 1 \to K$ **do**
  Choose topic $\phi_k \sim \text{Dir}(\beta)$
end for
for all document m in corpus D **do**
  Choose a topic distribution $\theta_m \sim \text{Dir}(\alpha)$
  for all word index n from 1 to N_m **do**
    Choose a topic $z_{mn} \sim \text{Categorical}(\theta_m)$
    Choose word $w_{mn} \sim \text{Categorical}(\phi_{z_{mn}})$
  end for
end for

7.4.1.2 *Expectation Maximization*

We begin by marginalizing the latent variable Z and finding the lower bound for the likelihood/posterior:

$$\begin{aligned}
\log p(W, \theta, \phi|\alpha, \beta) &= \log \sum_Z p(W, Z, \theta, \phi|\alpha, \beta) \\
&= \sum_{m=1}^{M} \sum_{n=1}^{N_m} \log \sum_{k=1}^{K} p(z_{mn} = k|\theta_m) p(w_{mn}|\phi_k) \\
&\qquad + \sum_{k=1}^{K} \log p(\phi_k|\beta) + \sum_{m=1}^{M} \log p(\theta_m|\alpha) \\
&= \sum_{m=1}^{M} \sum_{n=1}^{N_m} \log \sum_{k=1}^{K} q(z_{mn} = k|w_{mn}) \frac{p(z_{mn} = k|\theta_m) p(w_{mn}|\phi_k)}{q(z_{mn} = k|w_{mn})} \\
&\qquad + \sum_{k=1}^{K} \log p(\phi_k|\beta) + \sum_{m=1}^{M} \log p(\theta_m|\alpha) \\
\text{(Jensen Inequality)} \qquad &\geqslant \sum_{m=1}^{M} \sum_{n=1}^{N_m} \sum_{k=1}^{K} q(z_{mn} = k|w_{mn}) \log \frac{p(z_{mn} = k|\theta_m) p(w_{mn}|\phi_k)}{q(z_{mn} = k|w_{mn})} \\
&\qquad + \sum_{k=1}^{K} \log p(\phi_k|\beta) + \sum_{m=1}^{M} \log p(\theta_m|\alpha)
\end{aligned} \tag{7.28}$$

Let us define the following functional:

$$\begin{aligned}
F(q, \theta, \phi) := &- \sum_{m=1}^{M} \sum_{n=1}^{N_m} D_{KL}(q(z_{mn}|w_{mn}) || p(z_{mn}|w_{mn}, \theta_m, \phi)) \\
&+ \sum_{m=1}^{M} \sum_{n=1}^{N_m} p(w_{mn}|\theta_m, \phi) + \sum_{k=1}^{K} \log p(\phi_k|\beta) + \sum_{m=1}^{M} \log p(\theta_m|\alpha)
\end{aligned} \tag{7.29}$$

7.4.1.3 *E-Step*

In the E-step, we fix θ, ϕ and maximize F for q. As q appears only in the KL-divergence term, it is equivalent to minimizing the KL-divergence between $q(z_{mn}|w_{mn})$ and $p(z_{mn}|w_{mn}, \theta_m, \phi)$. We know that for any distributions f and g the KL-divergence is minimized when $f = g$ and is equal to 0. Thus, we have

$$\begin{aligned} q(z_{mn} = k|w_{mn}) &= p(z_{mn} = k|w_{mn}, \theta_m, \phi) \\ &= \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K} \theta_{mk'}\phi_{k'w_{mn}}} \end{aligned} \tag{7.30}$$

For simplicity of notation, let us define

$$\boxed{q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K} \theta_{mk'}\phi_{k'w_{mn}}}} \tag{7.31}$$

7.4.1.4 *M-Step*

In the E-step, we fix q and maximize F for θ, ϕ. As this will be a constrained optimization (θ and ϕ must lie on simplex), we use standard constrained optimization procedure of Lagrange multipliers. The Lagrangian can be expressed as:

$$\begin{aligned} \mathcal{L}(\theta, \phi, \lambda, \mu) = &\sum_{m=1}^{M}\sum_{m=1}^{N_m}\sum_{k=1}^{K} q(z_{mn} = k|w_{mn}) \log \frac{p(z_{mn} = k|\theta_m)p(w_{mn}|\phi_k)}{q(z_{mn} = k|w_{mn})} + \sum_{k=1}^{K} \log p(\phi_k|\beta) \\ &+ \sum_{m=1}^{M} \log p(\theta_m|\alpha) + \sum_{k=1}^{K} \lambda_k \left(1 - \sum_{v=1}^{V} \phi_{kv}\right) + \sum_{m=1}^{M} \mu_i \left(1 - \sum_{k=1}^{K} \theta_{mk}\right) \\ = &\sum_{m=1}^{M}\sum_{n=1}^{N_m}\sum_{k=1}^{K} q_{mnk} \log \theta_{mk}\phi_{kw_{mn}} + \sum_{k=1}^{K}\sum_{v=1}^{V} (\beta_v - 1) \log \phi_{kv} + \sum_{m=1}^{M}\sum_{k=1}^{K} (\alpha_k - 1) \log \theta_{mk} \\ &+ \sum_{k=1}^{K} \lambda_k \left(1 - \sum_{v=1}^{V} \phi_{kv}\right) + \sum_{m=1}^{M} \mu_m \left(1 - \sum_{k=1}^{K} \theta_{mk}\right) + \text{const.} \end{aligned} \tag{7.32}$$

MAXIMISING θ Taking derivative with respect to θ_{mk} and setting it to 0, we obtain

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta_{mk}} = 0 &= \sum_{j=1}^{N_m} \frac{q_{mnk} + \alpha_k - 1}{\theta_{mk}} - \mu_m \\ \mu_m \theta_{mk} &= \sum_{j=1}^{N_i} q_{mnk} + \alpha_k - 1 \end{aligned} \tag{7.33}$$

After solving for μ_m, we finally obtain

$$\theta_{mk} = \frac{\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1}{\sum_{k'=1}^{K}\sum_{j=1}^{N_m} q_{mnk'} + \alpha_{k'} - 1} \tag{7.34}$$

Note that $\sum_{k'=1}^{K} q_{mnk'} = 1$, we reach at the optimizer:

$$\boxed{\theta_{mk} = \frac{1}{N_m + \sum(\alpha_{k'} - 1)} \left(\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1\right)} \tag{7.35}$$

MAXIMISING ϕ Taking derivative with respect to ϕ_{kv} and setting it to 0, we obtain

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \phi_{kv}} = 0 = \sum_{m=1}^{M}\sum_{n=1}^{N_m} \frac{q_{mnk}\delta(v - w_{mn}) + \beta_v - 1}{\phi_{kv}} - \lambda_k \\ \lambda_k \phi_{kv} = \sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}\delta(v - w_{mn}) + \beta_v - 1\end{aligned} \tag{7.36}$$

After solving for λ_k, we finally obtain

$$\phi_{kv} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}\delta(v - w_{mn}) + \beta_v - 1}{\sum_{v'=1}^{V}\sum_{m=1}^{M}\sum_{n=1}^{N_m} \delta(v' - w_{mn}) + \beta_{v'} - 1} \tag{7.37}$$

Note that $\sum_{v'=1}^{V} \delta(v' - w_{mn}) = 1$, we reach at the optimizer:

$$\boxed{\phi_{kv} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}\delta(v - w_{mn}) + \beta_v - 1}{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk} + \sum(\beta_{v'} - 1)}} \tag{7.38}$$

7.4.1.5 *Introducing Stochasticity*

After performing the E-step, we add an extra simulation step, i.e. we draw and impute the values for the latent variables from its distribution conditioned on data and current estimate of the parameters. This means basically q_mnk gets transformed into $\delta(z_{mn} - \tilde{k})$ where $\tilde{k}$ is value drawn from the conditional distribution. Then we proceed to perform the M-step, which is even simpler now. To summarize SEM for LDA will have following steps:

E-STEP : in parallel compute the conditional distribution locally:

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K} \theta_{mk'}\phi_{k'w_{ij}}} \tag{7.39}$$

S-STEP : in parallel draw z_{mn} from the categorical distribution:

$$z_{mn} \sim \mathsf{Categorical}(q_{mn1}, ..., q_{mnK}) \tag{7.40}$$

M-STEP : in parallel compute the new parameter estimates:

$$\begin{aligned}\theta_{mk} &= \frac{D_{mk} + \alpha_k - 1}{N_m + \sum(\alpha_{k'} - 1)} \\ \phi_{kv} &= \frac{W_{kv} + \beta_v - 1}{T_k + \sum(\beta_{v'} - 1)}\end{aligned} \tag{7.41}$$

where $D_{mk} = \Big|\{\, z_{mn} \mid z_{mn} = k \,\}\Big|$,
$W_{kv} = \Big|\{\, z_{mn} \mid w_{mn} = v,\, z_{mn} = k \,\}\Big|$, and
$T_k = \Big|\{\, z_{mn} \mid z_{mn} = k \,\}\Big| = \sum_{v=1}^{V} W_{kv}$.

7.4.2 *Equivalency between (S)EM and (S)GD for LDA*

We study the equivalency between (S)EM and (S)GD for LDA.

7.4.2.1 *EM for LDA*

EM for LDA can be summarized by follows:

E-STEP

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} \tag{7.42}$$

M-STEP

$$\begin{aligned}\theta_{mk} &= \frac{1}{N_m + \sum(\alpha_{k'} - 1)}\left(\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1\right) \\ \phi_{kv} &= \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}\delta(v - w_{mn}) + \beta_v - 1}{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk} + \sum(\beta_{v'} - 1)}\end{aligned} \tag{7.43}$$

7.4.2.2 *GD for LDA*

The joint probability density can be expressed as:

$$\begin{aligned}p(W, Z, \theta, \phi|\alpha, \beta) &= \left[\prod_{k=1}^{K} p(\phi_k|\beta)\right]\left[\prod_{m=1}^{M} p(\theta_m|\alpha)\prod_{n=1}^{N_m} p(z_{mn}|\theta_m)p(w_{mn}|\phi_{z_{mn}})\right] \\ &\propto \left[\prod_{k=1}^{K}\prod_{v=1}^{V}\phi_{kv}^{\beta-1}\right]\left[\prod_{m=1}^{M}\left(\prod_{k=1}^{K}\theta_{mk}^{\alpha-1}\right)\prod_{n=1}^{N_m}\theta_{mz_{mn}}\phi_{z_{mn}w_{mn}}\right]\end{aligned} \tag{7.44}$$

The log-probability of joint model with Z marginalized can be written as:

$$\begin{aligned}\log p(W, \theta, \phi|\alpha, \beta) &= \log\sum_{Z} p(W, Z, \theta, \phi|\alpha, \beta) \\ &= \sum_{m=1}^{M}\sum_{n=1}^{N_m}\log\sum_{k=1}^{K} p(z_{mn} = k|\theta_m)p(w_{mn}|\phi_k) \\ &\qquad + \sum_{k=1}^{K}\log p(\phi_k|\beta) + \sum_{m=1}^{M}\log p(\theta_m|\alpha) \\ &= \sum_{m=1}^{M}\sum_{n=1}^{N_m}\log\sum_{k=1}^{K}\theta_{mk}\phi_{kw_{mn}} \\ &\qquad + \sum_{m=1}^{M}\sum_{k=1}^{K}(\alpha_k - 1)\log\theta_{mk} + \sum_{k=1}^{K}\sum_{v=1}^{V}(\beta_v - 1)\log\phi_{kv}\end{aligned} \tag{7.45}$$

GRADIENT FOR TOPIC PER DOCUMENT Now take derivative with respect to θ_{mk}:

$$\begin{aligned}\frac{\partial\log p}{\partial\theta_{mk}} &= \sum_{j=1}^{N_m}\frac{\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} + \frac{\alpha_k - 1}{\theta_{mk}} \\ &= \frac{1}{\theta_{mk}}\left(\sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1\right)\end{aligned} \tag{7.46}$$

GRADIENT FOR WORD PER TOPIC Now take derivative with respect to ϕ_{kv}:

$$\begin{aligned}\frac{\partial \log p}{\partial \phi_{kv}} &= \sum_{m=1}^{M}\sum_{n=1}^{N_m}\frac{\theta_{mk}\delta(v-w_{mn})}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}}+\frac{\beta_v-1}{\phi_{kv}} \\ &= \frac{1}{\phi_{kv}}\left(\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}\delta(v-w_{mn})+\beta_v-1\right)\end{aligned} \tag{7.47}$$

7.4.2.3 *Equivalency*

If we look at one step of EM:

FOR TOPIC PER DOCUMENT

$$\begin{aligned}\theta^{+}_{mk} &= \frac{1}{N_m+\sum(\alpha_{k'}-1)}\left(\sum_{n=1}^{N_m} q_{mnk}+\alpha_k-1\right) \\ &= \frac{\theta_{mk}}{N_m+\sum(\alpha_{k'}-1)}\frac{\partial \log p}{\partial \theta_{mk}}\end{aligned}$$

Vectorize and can be re-written as:

$$\theta^{+}_{m} = \theta_m + \frac{1}{N_m+\sum(\alpha_{k'}-1)}\left[\operatorname{diag}(\theta_m)-\theta_m\theta_m^{T}\right]\frac{\partial \log p}{\partial \theta_m} \tag{7.48}$$

FOR WORD PER TOPIC

$$\begin{aligned}\phi^{+}_{kv} &= \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}\delta(v-w_{mn})+\beta_v-1}{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}+\sum(\beta_{v'}-1)} \\ &= \frac{\phi_{kv}}{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}+\sum(\beta_{v'}-1)}\frac{\partial \log p}{\partial \phi_{kv}}\end{aligned}$$

Vectorize and can be re-written as:

$$\phi^{+}_{k} = \phi_k + \frac{1}{\sum_{m=1}^{M}\sum_{n=1}^{N_m} q_{mnk}+\sum(\beta_{v'}-1)}\left[\operatorname{diag}(\phi_k)-\phi_k\phi_k^{T}\right]\frac{\partial \log p}{\partial \phi_k} \tag{7.49}$$

7.4.2.4 *SEM for LDA*

We summarize our SEM derivation for LDA as follows:

E-STEP

$$q_{mnk} = \frac{\theta_{mk}\phi_{kw_{mn}}}{\sum_{k'=1}^{K}\theta_{mk'}\phi_{k'w_{mn}}} \tag{7.50}$$

S-STEP

$$z_{mn} \sim \mathsf{Categorical}(q_{mn1},\ldots,q_{mnK}) \tag{7.51}$$

M-STEP

$$\begin{aligned}\theta_{mk} &= \frac{D_{mk}+\alpha_k-1}{N_m+\sum(\alpha_{k'}-1)} \\ \phi_{kv} &= \frac{W_{kv}+\beta_v-1}{T_k+\sum(\beta_{v'}-1)}\end{aligned} \tag{7.52}$$

Here D_{mk} is the total number of tokens that belong to topic k in document m, W_{kv} is the number of times a word v belongs to topic k, i.e.,

$$D_{mk} = \sum_{n=1}^{N_m} z_{mnk} \tag{7.53}$$

$$W_{kv} = \sum_{n=1}^{N_m} \sum_{m=1}^{N_d} z_{mnk} \delta(w_m = v) \tag{7.54}$$

However, observe that all our z_{mn} are one-hot categorical random variables and hence, the above sums can be easily computed without going through the entire dataset. This is where the stochastic nature of SEM helps in reducing the training time. We next show the equivalency of SEM to SGD.

7.4.2.5 *Equivalency*

In case of LDA, let us begin with θ for which the update over one step stochastic EM is:

$$\theta^+_{mk} = \frac{D_{mk} + \alpha_k - 1}{N_m + \sum_{k'=1}^{K}(\alpha_{k'} - 1)} = \frac{1}{N_m + \sum_{k'=1}^{K}(\alpha_{k'} - 1)} \sum_{n=1}^{N_m} \delta(z_{mnk} = 1) + \alpha_k - 1$$

Again vectorizing and re-writing as earlier:

$$\theta^+_i = \theta_i + Mg$$

where $M = \frac{1}{N_m + \sum_{k'=1}^{K}(\alpha_{k'} - 1)} \left[\text{diag}(\theta_m) - \theta_m \theta_m^T\right]$ and $g = \frac{1}{\theta_{mk}} \sum_{n=1}^{N_m} \delta(z_{mnk} = 1) + \alpha_k - 1$. The vector g can be shown to be an unbiased noisy estimate of the gradient, i.e.

$$\begin{aligned} \mathbb{E}[g] &= \frac{1}{\theta_{mk}} \sum_{n=1}^{N_m} \mathbb{E}[\delta(z_{mnk} = 1)] + \alpha_k - 1 \\ &= \frac{1}{\theta_{mk}} \sum_{n=1}^{N_m} q_{mnk} + \alpha_k - 1 \qquad\qquad = \frac{\partial \log p}{\partial \theta_{mk}} \end{aligned}$$

Thus, it is SGD with constraints. We have a similar result for ϕ_{kv}, where we can see that an unbiased, noisy estimator of the gradient has been used instead of the pure gradient, in the SEM update of parameters. However, note that stochasticity does not arise from sub-sampling data as usually in SGD, rather from the randomness introduced in the S-step. But this immediately hints for developing an online/incremental version where we can subsample data also. This can remove the barrier in current implementation and we can have a revolver like structure, which would be loved by the hardware.

7.4.3 *Non-singularity of Fisher Information for Mixture Models*

Let us consider a general mixture model:

$$p(x|\theta, \phi) = \sum_{k=1}^{K} \theta_k f(x|\phi_k) \tag{7.55}$$

Then the log-likelihood can be written as:

$$\log p(x|\theta, \phi) = \log \left(\sum_{k=1}^{K} \theta_k f(x|\phi_k) \right) \tag{7.56}$$

The Fisher Information is given by:

$$\begin{aligned} I(\theta,\phi) &= \mathbb{E}\left[(\nabla \log p(x|\theta,\phi))(\nabla \log p(x|\theta,\phi))^T\right] \\ &= \begin{bmatrix} \frac{\partial}{\partial\theta}\log p(x|\theta,\phi) \\ \frac{\partial}{\partial\phi}\log p(x|\theta,\phi) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial\theta}\log p(x|\theta,\phi) \\ \frac{\partial}{\partial\phi}\log p(x|\theta,\phi) \end{bmatrix}^T \end{aligned}$$

These derivatives can be computed as follows:

$$\begin{aligned} \frac{\partial}{\partial\theta_k}\log p(x|\theta,\phi) &= \frac{\partial}{\partial\theta_k}\log\left((\sum_{k=1}^{K}\theta_k f(x|\phi_k)\right) \\ &= \frac{f(x|\phi_k)}{\sum_{k'=1}^{K}\theta_{k'}f(x|\phi_{k'})} \\ \frac{\partial}{\partial\phi_k}\log p(x|\theta,\phi) &= \frac{\partial}{\partial\phi_k}\log\left((\sum_{k=1}^{K}\theta_k f(x|\phi_k)\right) \\ &= \frac{\theta_k\frac{\partial}{\partial\phi_k}f(x|\phi_k)}{\sum_{k'=1}^{K}\theta_{k'}f(x|\phi_{k'})} \end{aligned} \tag{7.57}$$

For any $u, v \in \mathbb{R}^K$ (with at least one nonzero), then the Fisher Information is positive definite as:

$$\begin{aligned} (u^T\ v^T)I\begin{pmatrix} u \\ v \end{pmatrix} &= (u^T\ v^T)\mathbb{E}\left[\begin{bmatrix} \frac{\partial}{\partial\theta}\log\left(\sum_{k=1}^{K}\theta_k f(X|\phi_k)\right) \\ \frac{\partial}{\partial\phi}\log\left(\sum_{k=1}^{K}\theta k f(X|\phi_k)\right) \end{bmatrix}\begin{bmatrix} \frac{\partial}{\partial\theta}\log\left(\sum_{k=1}^{K}\theta_k f(X|\phi_k)\right) \\ \frac{\partial}{\partial\phi}\log\left(\sum_{i=1}^{K}\theta_k f(X|\phi_k)\right) \end{bmatrix}^T\right]\begin{pmatrix} u \\ v \end{pmatrix} \\ &= \mathbb{E}\left[\left(u^T\frac{\partial}{\partial\theta}\log\left(\sum_{k=1}^{K}\theta_k f(X|\phi_k)\right) + v^T\frac{\partial}{\partial\theta}\log\left(\sum_{i=1}^{K}\theta_k f(X|\phi_i)\right)\right)^2\right] \\ &= \mathbb{E}\left[\left(\frac{\sum_{k=1}^{K}u_k f(X|\phi_k) + v_k\theta_k\frac{\partial}{\partial\phi_k}f(X|\phi_k)}{\sum_{k=1}^{K}\theta_k f(X|\phi_k)}\right)^2\right] \end{aligned}$$

This can be 0 if and only if

$$\sum_{k=1}^{K} u_k f(x|\phi_i) + v_k\theta_k\frac{\partial}{\partial\phi_k}f(x;\phi_k) = 0 \quad \forall x. \tag{7.58}$$

In case of exponential family emission models this cannot hold if all components are unique and all $\theta_k > 0$. Thus, if we assume all components are unique and every component has been observed at least once, the Fisher information matrix becomes non-singular.

7.4.4 *Alias Sampling Method*

The alias sampling method is an efficient method for drawing samples from a K outcome discrete distribution in $O(1)$ amortized time and we describe it here for completeness. Denote by p_i for $i \in \{1 \dots K\}$ the probabilities of a distribution over K outcomes from which we would like to sample. If p were the uniform distribution, i.e. $p_i = K^{-1}$, then sampling would be trivial. For the general case, we must pre-process the distribution p into a table of K triples of the form (i, j, π_i) as follows:

- Partition the indices $\{1 \dots K\}$ into sets U and L where $p_i > K^{-1}$ for $i \in U$ and $p_i \leqslant K^{-1}$ for $i \in L$.
- Remove any i from L and j from U and add (i, j, p_i) to the table.
- Update $p_j = p_i + p_j - K^{-1}$ and if $p_j > K^{-1}$ then add j to U, else to L.

By construction the algorithm terminates after K steps; moreover, all probability mass is preserved either in the form of π_i associated with i or in the form of $K^{-1} - \pi_i$ associated with j. Hence, sampling from p can now be accomplished in constant time:

- Draw (i, j, π_i) uniformly from the set of k triples in K.
- With probability $K\pi_i$ emit i, else emit j.

Hence, if we need to draw from p at least K times, sampling can be accomplished in amortized $O(1)$ time.

7.4.5 *Applying ESCA*

We begin with a simple Gaussian mixture model (GMM) with K components. Let $x_1, ..., x_n$ be independent and identically distributed observations, $z_1, ..., z_n$ be hidden component assignment variable and $\eta = \eta(\theta_1, ..., \theta_K, \mu_1, \Sigma_1, \mu_2, \Sigma_2, ..., \mu_K, \Sigma_K)$ be the parameters. Then the GMM fits into ESCA with sufficient statistics given by:

$$\begin{aligned} T(x_i, z_i) = [&\mathbb{1}\{z_i = 1\}, ..., \mathbb{1}\{z_i = K\}, \\ &x_i \mathbb{1}\{z_i = 1\}, ..., x_i \mathbb{1}\{z_i = K\}, \\ &x_i x_i^T \mathbb{1}\{z_i = 1\}, ..., x_i x_i^T \mathbb{1}\{z_i = K\}]. \end{aligned} \tag{7.59}$$

The conditional distribution for the E-step is:

$$p(z_i = k | x_i; \eta) \propto \theta_k \mathcal{N}(x_i | \mu_k, \Sigma_k) \tag{7.60}$$

In the S-step we draw from this conditional distribution and the M-step, through inversion of link function, is:

$$\begin{aligned} \tilde{\theta}_k &= \frac{1}{n + K\alpha - K} \sum_{i=1}^{n} (\mathbb{1}\{z_i = k\} + \alpha - 1) \\ \tilde{\mu}_k &= \frac{\kappa_0 \mu_0 + \sum_{i=1}^{n} x_i \mathbb{1}\{z_i = k\}}{\kappa_0 + \sum_{i=1}^{n} \mathbb{1}\{z_i = k\}} \\ \tilde{\Sigma}_k &= \frac{\Psi_0 + \kappa_0 \mu_0 \mu_0^T + \sum_{i=1}^{n} x_i x_i^T \mathbb{1}\{z_i = k\} - (\kappa_0 + \sum_{i=1}^{n} \mathbb{1}\{z_i = k\}) \tilde{\mu}_k \tilde{\mu}_k^T}{\nu_0 + d + 2 + \sum_{i=1}^{n} \mathbb{1}\{z_i = k\}} \end{aligned} \tag{7.61}$$

and is only function of the sufficient statistics.

Next, we provide more details on how to employ ESCA for any conditional exponential family mixture model; i.e., in which n random variables x_i, $i = 1, \dots, n$ correspond to observations, each distributed according to a mixture of K components, with each component belonging to the same exponential family of distributions (e.g., all normal, all multinomial, etc.), but with different parameters:

$$p(x_i | \phi) = \exp(\langle \psi(x_i), \phi \rangle - g(\phi)). \tag{7.62}$$

The model also has n latent variables z_i that specify the identity of the mixture component of each observation x_i, each distributed according to a K-dimensional categorical distribution. A set of K mixture weights θ_k, $k = 1, \ldots, K$, each of which is a probability (a real number between 0 and 1 inclusive) and collectively sum to one. A Dirichlet prior on the mixture weights with hyper-parameters α. A set of K parameters ϕ_k, $k = 1, \ldots, K$, each specifying the parameter of the corresponding mixture component. For example, observations distributed according to a mixture of one-dimensional Gaussian distributions will have a mean and variance for each component. Observations distributed according to a mixture of V-dimensional categorical distributions (e.g., when each observation is a word from a vocabulary of size V) will have a vector of V probabilities, collectively summing to 1. Moreover, we put a shared conjugate prior on these parameters:

$$p(\phi; n_0, \psi_0) = \exp\left(\langle \psi_0, \phi \rangle - n_0 g(\phi) - h(m_0, \psi_0)\right). \tag{7.63}$$

Then joint sufficient statistics would be given by:

$$\begin{aligned} T(z_i, x_i) = [&\mathbb{1}\{z_i = 1\}, ..., \mathbb{1}\{z_i = K\}, \\ &\psi(x_i)\mathbb{1}\{z_i = 1\}, ..., \psi(x_i)\mathbb{1}\{z_i = K\}] \end{aligned} \tag{7.64}$$

In the E-step of t^{th} iteration, we derive the conditional distribution $p(z_i|x_i, \eta)$, namely

$$\begin{aligned} p(z_i = k|x_i, \eta) &\propto p(x_i|\phi_k^{t-1}, z_i = k) p(z_i = k|\theta^{t-1}) \\ &= \frac{\theta_k^{t-1} p(x_i|\phi_k^{t-1})}{\sum_{k'} \theta_{k'}^{t-1} p(x_i|\phi_{k'}^{t-1})} \end{aligned} \tag{7.65}$$

In the S-step we draw z_i^t from this conditional distribution and the M-step through inversion of the link function yields:

$$\begin{aligned} \nabla g(\tilde{\phi}_k) &= \frac{\phi_0 + \sum_i \psi(x_i)\mathbb{1}\{z_i = k\})}{n_0 + \sum_i \mathbb{1}\{z_i = k\}} \\ \text{or} \qquad \tilde{\phi}_k &= \xi^{-1}\left(\frac{\psi_0 + \sum_i \psi(x_i)\mathbb{1}\{z_i = k\}}{n_0 + \sum_i \mathbb{1}\{z_i = k\}}\right) \\ \tilde{\theta}_k &= \frac{\sum_i \mathbb{1}\{z_i = k\} + \alpha_k - 1}{n + \sum_k \alpha_k - k}. \end{aligned} \tag{7.66}$$

This encompasses most of the popular mixture models (and with slight more work all the mixed membership or admixture models) with Binomial, multinomial, or Gaussian emission model, e.g. beta-binomials for identification, Dirichlet-multinomial for text or Gauss-Wishart for images as listed in Table 7.1.

7.5 EXPERIMENTS

To evaluate the strength and weaknesses of our algorithm, we compare against parallel and distributed implementations of CGS and CVB0. We also compare our results to performance numbers reported in the literature including those of F+LDA and lightLDA.

Dataset	V	M	Tokens
PubMed	141,043	8,200,000	737,869,085
Wikipedia	210,233	6,631,176	1,133,050,514
Large	~140,000	~3 billion	~171 billion

Table 7.2: Statistics of the dataset used.

7.5.1 *Setup*

SOFTWARE & HARDWARE All three algorithms – CGS, CVB0, and ESCA – are implemented in simple C++11. We implement multithreaded parallelization within a node using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the process binding to a socket over MPI. We also implemented a version of ESCA with a sparse representation for the array D of counts of topics per documents and Vose's alias method to draw from discrete distributions. We run our experiments on a small cluster of 8 Amazon EC2 c4.8xlarge nodes connected through 10Gb/s Ethernet. Each node has a 36 virtual threads per node. For random number generation we employ Intel®Digital Random Number Generators through instruction RDRAND, which uses thermal noise within the silicon to output a random stream of bits at 3 Gbit/s, producing true random numbers.

DATASETS We experiment on two public datasets, both of which are cleaned by removing stop words and rare words: PubMed abstracts and English Wikipedia. To demonstrate scalability, run on 100 copies of English Wikipedia and a third proprietary dataset. Statistics of the three dataset are reported in Table 7.2

7.5.2 *Evaluation*

QUALITATIVE Table 7.3 shows three global topics inferred from 20- Newsgroups by MGCTM. Each topic is represented by the ten most probable words for that topic. It can be seen that these global topics capture the common semantics in the whole corpus and is not specifically associated with a certain news group. Global topic 9 is about news archive organization. Topic 10 is about time. Topic 19 is about article writing. These topics can be used to generate documents in all groups.

QUANTITATIVE How to quantitatively evaluate topic models is a open problem in general. To evaluate the proposed method we use predicting power as a metric by calculating the per-word log-likelihood (equivalent to negative log of perplexity) on 10,000 held-out documents conditioned on the trained model. In other words, we use:

$$\text{LogLikelihood}(D_{\text{test}}) = \sum_{m \in D_{\text{test}}} \text{frac1}N_m \sum_{i=1}^{N_m} \log p(w_{mn}).$$

Here w_{mn} is the n-th word in document m, N_m denotes the number of words in document m. We set $K = 1000$ to demonstrate performance for a large number of topics. The hyper parameters are set as $\alpha = 50/K$ and $\beta = 0.1$ as suggested in T.L. Griffiths and Steyvers, 2004; other systems such as YahooLDA and Mallet also use this as the default parameter setting. The results are presented in Figure 7.3.

Finally, for the larger datasets, our implementation of ESCA (only 300 lines of C++) processes more than **1 billion tokens per second** (tps) by using a 20-node cluster. In comparison, some

PubMed				
seizures	data	local	gene	state
epilepsy	information	block	transcript	change
seizure	available	lidocaine	exon	transition
epileptic	provide	anethesia	genes	states
temporal_lobe	regarding	anethetic	expression	occur
anticonvulsant	sources	acupuncture	region	process
convulsion	literature	bupivacaine	mrna	shift
kindling	concerning	anaesthesia	mouse	condition
partial	limited	under	expressed	changed
generalized	provided	anaesthetic	human	dynamic
Wikipedia				
hockey	medical	von	boy	music
ice	medicine	german	youth	music
league	hospital	karl	boys	pop
played	physician	carl	camp	music
junior	doctor	friedrich	girl	artists
nhl	clinical	wilhelm	scout	electronic
professional	md	johann	girls	duo
games	physicians	ludwig	guide	genre
playing	doctors	prussian	scouts	genres
national	surgeon	heinrich	scouting	musicians

Table 7.3: The first five topics inferred via ESCA on LDA from both PubMed and Wikipedia datasets.

of the best existing systems achieve 112 million tps (F+LDA, personal communication) and 60 million tps (lightLDA) Yuan et al., 2015 using more hardware.

COMPARISON TO LARGE SCALE SYSTEMS We choose not to evaluate directly against highly tuned and optimized systems

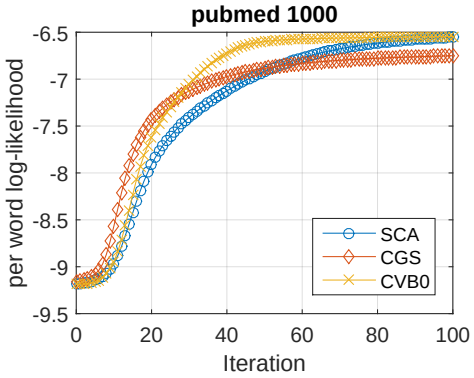
7.6 POINTS TO PONDER

We have described a novel inference method for latent variable models that simulates a stochastic cellular automaton. The equilibrium of the dynamics are MAP fixed points and the algorithm has many desirable computational properties: it is embarrassingly parallel, memory efficient, and like HOGWILD! (Recht et al., 2011), is virtually lock-free. Further, for many models, it enables the use of approximate counters and the alias method. Thus, we were able to achieve an order of magnitude speed-up over the current state-of-the-art inference algorithms for LDA with accuracy comparable to collapsed Gibbs sampling.

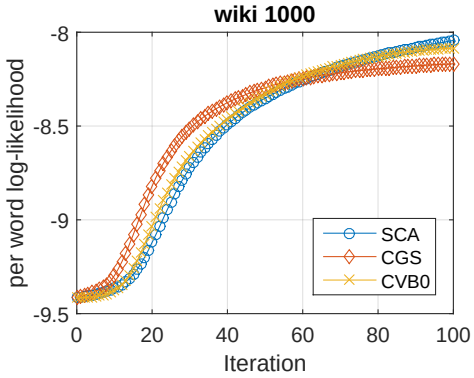
In general, we cannot always guarantee the correct invariant measure (Dawson, 1974), and found that parallelizing improperly causes convergence to incorrect MAP fixed points. Even so, SCA is used for simulating Ising models in statistical physics (Vichniac, 1984). Interestingly, in previous work by Lebowitz, Maes, and Speer (1990), it has been shown that stochastic cellular automata are closely related to equilibrium statistical models and the stationary distribution is known for a large class of finite stochastic cellular automata.

Method	Dataset	Number of Topics	Size of Vocabulary	Number of Documents	Number of Tokens	Infrastructure	Year	Processing Speed
YahooLDA (Alexander Smola and Narayanamurthy, 2010)	PubMed	1K	140K	8.2M	797M	10 machines on hadoop	2010	12.87M tokens/s
lightLDA (Yuan et al., 2015)	Bing "web chunk"	1000K	50K	1.2B	200B	24 machines (480 cores)	2014	60M tokens/s
F+LDA (H.-F. Yu et al., 2015)	Amazon reviews	1K	1680K	29M	1.5B	32 machines (640 cores)	2014	110M tokens/s
ESCA	**100 copies of Wikipedia**	**1K**	**210K**	**667M**	**128B**	**8 Amazon c4.8x large (288 virtual cores)**	2015	**503M tokens/s**
ESCA	**100 copies of Wikipedia**	**1K**	**210K**	**667M**	**128B**	**20 Amazon c4.8x large (720 virtual cores)**	2015	**1200M tokens/s**

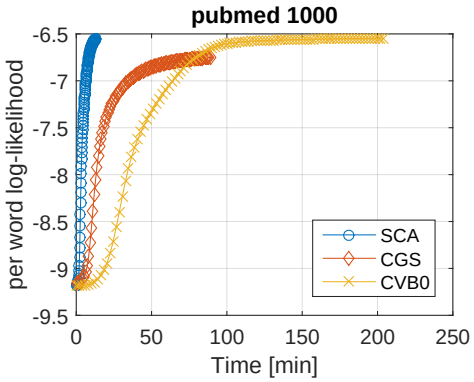
Table 7.4: Comparison with existing scalable LDA frameworks.



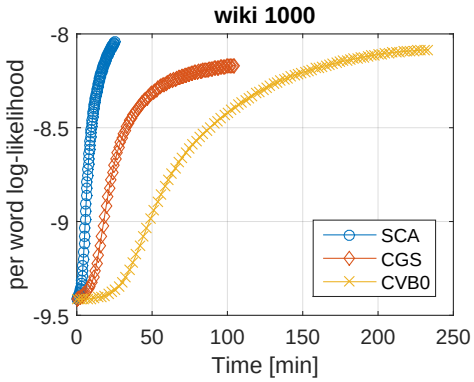
(a) PubMed, K = 1000, $\alpha = 0.05$, $\beta = 0.1$



(b) Wikipedia, K = 1000, $\alpha = 0.05$, $\beta = 0.1$



(c) Pubmed, K = 1000, $\alpha = 0.05$, $\beta = 0.1$



(d) Wikipedia, K = 1000, $\alpha = 0.05$, $\beta = 0.1$

Figure 7.3: Evolution of log likelihood on Wikipedia and Pubmed over number of iterations and time.

7.6.1 *Understanding the limitations for Full Posterior*

While ESCA has tremendous potential as a computational model for machine learning, in some cases, using it to obtain MAP estimates is not clear.

Consider an Ising model on an d-dimensional torus $\mathbb{H}$

$$p(x) \propto \prod_{\langle i,j\rangle \in \mathbb{H}} \exp(w_{ij} x_i x_j) \tag{7.67}$$

in which x_i takes on values in $\{-1, 1\}$. The equilibrium distribution of SCA with a Gibbs update is then (Neumann and Derrida, 1988)

$$q(x) \propto \prod_{\langle i,j\rangle \in \mathbb{H}} \cosh(w_{ij} x_i x_j). \tag{7.68}$$

Note that the hyperbolic cosine function (cosh) is symmetric in the sense that $\cosh(r) = \cosh(-r)$. For values $r \geqslant 0$ cosh is a good approximation and has a maximum that corresponds to the exponential function; however, for values $r < 0$, the cosh is a poor approximation for the exponential function.

Let x_1, x_2 be two random variables taking on values in $\{-1, 1\}$. We define a simple two-variable Ising model on a trivial one-dimensional torus:

$$p(x_1, x_2) \propto \exp(x_1 x_2) \tag{7.69}$$

We can enumerate and quantify the state space under both SCA $q(x_1, x_2)$ and the true distribution $p(x_1, x_2)$:

state	x_1	x_2	$x_1 * x_2$	$q(x_1, x_2) \propto$	$p(x_1, x_2) \propto$
0	-1	-1	1	cosh(1)	exp(1)
1	-1	1	-1	cosh(-1)	exp(-1)
2	1	-1	-1	cosh(-1)	exp(-1)
3	1	1	1	cosh(1)	exp(1)

Since cosh is symmetric, all states are equally probable for SCA and states 1 and 2 are MAP states. Yet, under the true distribution, they are not. Consequently, SCA with a Gibbs rule for the local evolution function can yield incorrect MAP estimates.

Fortunately, in most cases we are interested in a model over a dataset in which the data is independent and identically distributed That is, we can fix our example as follows. Rather than parallelizing a single Ising model at the granularity of pixels (over a single torus or grid), we instead parallelize the Ising model at the granularity of the data (over multiple tori, one for each image). Then, we could employ Gibbs sampling on each image for the S-step.

7.6.2 *Understanding the limitations for Non-Factored Models*

Till now we assumed certain factorization of LVM to enable ESCA. There are models in the exponential family such as restricted Boltzmann machines (RBMs) which do not factorize. Direct application of ESCA to such models will not work, but there is a potential work and ESCA can be still applicable to such models. For example, if the data were a collection of images, each cell could independently compute the S-step for its respective image. For RBMs the cell would flip a biased coin for each latent variable, and for deep Boltzmann machines, the cells could perform Gibbs sampling.

To elabortate, consider 2-layer RBM (1 observed, 1 latent), then ESCA should work as it is. That is, we sample latent variables conditioned on data and weights. Then optimize weights, given latent variables and observed data. Now if we have deep RBM, i.e. one with many hidden layers. Then ESCA will have similar problem as Ising model. But there is a quick fix borrowing ideas from chromatic samplers.

for each iteration

1. Sample all odd layers of the RBM
2. Optimize for weights
3. Sample all even layers of the RBM
4. Optimize for weights

end for

We save a precise derivation and empirical evaluation for future work.

8

Reducing Lookups

We study the problem of scaling up inference for latent variable models (LVM) over a large space by reducing look-ups. Hierarchical Bayesian models often capture distributions over a very large number of distinct atoms. The need for these models arises when organizing huge amount of unsupervised data, for instance, features extracted using deep convnets that can be exploited to organize abundant unlabeled images. Inference for hierarchical Bayesian models in such cases can be rather nontrivial, leading to approximate approaches. In this chapter, we propose *Canopy*, a sampler based on Cover Trees that is exact, has guaranteed runtime logarithmic in the number of atoms, and is provably polynomial in the inherent dimensionality of the underlying parameter space. In other words, the algorithm is as fast as search over a hierarchical data structure. We provide theory for Canopy and demonstrate its effectiveness on both synthetic and real datasets, consisting of over 100 million images.

8.1 Introduction

Fast nearest-neighbor algorithms have become a mainstay of information retrieval (Beygelzimer, Kakade, and Langford, 2006; Indyk and Motwani, 1998; T. Liu, Rosenberg, and Rowley, 2007). Search engines are able to perform virtually instantaneous lookup among sets containing billions of objects. In contrast, inference procedures for latent variable models (Gibbs sampling, EM, or variational methods) are often problematic even when dealing with thousands of distinct objects. This is largely because, for any inference methods, we potentially need to evaluate *all* probabilities whereas search only needs the *best* instance.

While the above is admittedly an oversimplification of matters (after all, we can use Markov-Chain Monte Carlo methods for inference), it is nonetheless nontrivial to perform exact sampling for large state spaces. In the current work, we propose *Canopy*, an inference technique to address this issue by marrying a fast lookup structure with an adaptive rejection sampler. This leads to a surprisingly simple design for a plethora of sampling-based inference algorithms. Moreover, we provide runtime guarantees for Canopy that depend only on the inherent dimen-

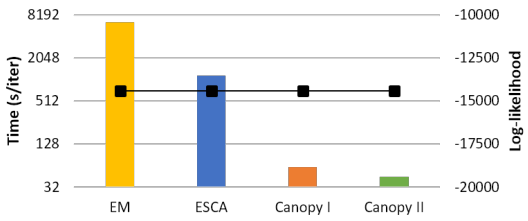


Figure 8.1: Canopy is much faster yet as accurate as other methods like EM or ESCA (Zaheer, Wick, et al., 2015). The bar graph shows time per iteration while line plots the likelihood on held-out test set. Results shown are for inference of a Gaussian mixture model with 32 million points having 4096 clusters at 1024 dimensions.

sionality of both parameter and data distributions. The expected depth for lookups is never worse than logarithmic in the number of atoms and the characteristic length scale at which models can be sufficiently well distinguished. Furthermore, we can parallelize Canopy for hierarchical Bayesian models using stochastic cellular automata (ESCA; Zaheer, Wick, et al. 2015), thus leading to an extremely scalable and efficient system design.

Most latent variable models, *e.g.*, Gaussian mixture models (GMM), latent Dirichlet allocation (D. M. Blei, A. Y. Ng, and Michael I. Jordan, 2003), hidden Markov models, Dirichlet process clustering (R. Neal, 1998), or hierarchical generative models (Adams, Z. Ghahramani, and M. Jordan, 2010), have the structure of the form:

$$p(x) = \sum_z p(z)p(x|\theta_z) \tag{8.1}$$

where x denotes observed variables, z latent variables, and θ_z parameters of the conditional. Often the conditional distribution $p(x|\theta_z)$ belongs to the exponential family, which we assume to be the case as well. The inference procedure on these models using either Gibbs sampling, stochastic variation methods, or ESCA would require to draw $z \sim p(z|x)$ repeatedly. Naïvely producing these draws would be expensive, especially when the number of latent classes is huge. We aim to bring the per-iteration cost down from $O(mn)$ to $\tilde{O}(m+n)$, where m, n are the number of latent classes and data points, respectively. For example, on GMM, the proposed method Canopy is much faster than EM or ESCA, while achieving the same accuracy as shown in Figure 8.1.

Our approach is as follows: we use cover trees (Beygelzimer, Kakade, and Langford, 2006) to design an efficient lookup structure for $p(x|\theta_z)$ and approximate the values of $p(x|\theta_z)$ for a large number of θ_z. In combination with an efficient node summary for $p(z)$, this allows us to design a rejection sampler that has an increasingly low rejection rate as we descend the tree. Moreover, for large numbers of observations x, we use another cover tree to aggregate points into groups of similar points, perform expensive pre-computation of assignment probabilities $p(z|x)$ only once, and amortize them over multiple draws. In particular, the alias method (Vose, 1991; Walker, 1977) allows us to perform sampling in $O(1)$ time once the probabilities have been computed.

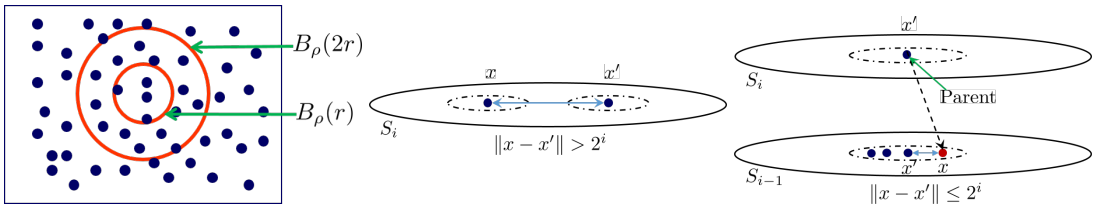
In summary, Canopy has three parts: construction of cover trees for both parameters and data (Section 8.3.1, Section 8.3.2), an adaptive rejection sampler at the top-level of the cover tree until the data representation is sufficiently high to exploit it for sampling (Section 8.3.2.1), and a rejection sampler in the leaves (Section 8.3.2.2), whenever the number of clusters is large. Most importantly, the algorithm becomes more efficient as we obtain larger amounts of data since they lead to greater utilization of the alias table in Walker (1977) as shown by theoretical analysis in Section 8.4. This makes it particularly well-suited to big data problems as demonstrated through experiments in Section 8.5.

8.2 BACKGROUND

We briefly discuss latent variable models, cover trees, and the alias method needed to explain this work.

8.2.1 *Latent Variable Models*

The key motivation for this work is to make inference in latent variable models more efficient. As expressed in (8.1), we consider latent models which have mixtures of exponential family.



(a) Expansion rate (b) Separation property (c) Covering property

Figure 8.2: Illustration of various properties of covering tree.

The reasons for limiting to exponential families are two fold. First, most of the mixture models used in practice belong to this class. Second, assumptions on model structure, for instance exponential family, allows for efficient design of fast inference. In particular, we first assume that updates to $p(z)$ can be carried out by modifying $O(1)$ values at any given time. For instance, for Dirichlet process mixtures, the collapsed sampler uses $p(z_i = j|Z\backslash\{z_i\}) = n_j^{-i}/(n + \alpha - 1)$. Here, n is the total number of observations, n_j^{-i} denotes the number of occurrences of $z_l = j$ when ignoring z_i, and α is the concentration parameter. Second, the conditional $p(x|\theta)$ in (8.1) is assumed to be a member of the exponential family, *i.e.*,

$$p(x|\theta) = \exp(\langle \phi(x), \theta \rangle - g(\theta)). \tag{8.2}$$

Here $\phi(x)$ represents the sufficient statistics and $g(\theta_z)$ is the (normalizing) log-partition function.

Trying to find a metric data structure for fast retrieval is not necessarily trivial for the exponential family. K. Jiang, Kulis, and M. Jordan (2012) and Cayton (2008) design Bregman divergence based methods for this problem. Unfortunately, such methods are costlier to maintain and have less efficient lookup properties than those using Euclidean distance, as computing and optimizing over Bregman divergences is less straightforward. For example, whenever we end up on the boundary of the marginal polytope, as is common with natural parameters associated with single observations, optimization becomes intractable. Fortunately, this problem can be avoided entirely by rewriting the exponential family model as

$$p(x|\theta) = e^{\langle (\phi(x), -1), \theta, g(\theta) \rangle} = e^{\left\langle \tilde{\phi}(x), \tilde{\theta} \right\rangle} \tag{8.3}$$
$$\text{where } \tilde{\phi}(x) := (\phi(x), -1) \text{ and } \tilde{\theta} := (\theta, g(\theta)).$$

In this case, being able to group similar $\tilde{\theta}$ together allows us to assess their contributions efficiently without having to inspect individual terms. Finally, we assume that $\left\|\tilde{\phi}(x_i)\right\| \leqslant R$ and $\left\|\tilde{\theta}_z\right\| \leqslant T$ for all i and for all $z \in \mathcal{Z}$ respectively.

8.2.2 *Alias Sampler*

A key component of Canopy is the alias sampler (Vose, 1991; Walker, 1977). Given an arbitrary discrete probability distribution on n outcomes, it allows for $O(1)$ sampling once an $O(n)$ preprocessing step has been performed. Hence, drawing n observations from a distribution over n outcomes costs an amortized $O(1)$ per sample. Section 7.4.4 has more details.

8.2.3 *Cover Trees*

Cover Trees (Beygelzimer, Kakade, and Langford, 2006) are a hierarchical data structure that allow fast retrieval in logarithmic time. The key properties are: $O(n \log n)$ construction time,

$O(\log n)$ retrieval, and polynomial dependence on the expansion constant (Karger and Ruhl, 2002) of the underlying space, which we refer to as c. Moreover, the degree of all internal nodes is well controlled, thus giving guarantees for retrieval (as exploited by (Beygelzimer, Kakade, and Langford, 2006)), and for sampling (as we will be using in this paper).

Cover trees are defined as an infinite succession of levels S_i with $i \in \mathbb{Z}$ as illustrated in Figure 8.2. Each level i contains (a nested subset of) the data with the following properties:

- Nesting property: $S_i \subseteq S_{i-1}$.
- All $x, x' \in S_i$ satisfy $\|x - x'\| \geqslant 2^i$.
- All $x \in S_i$ have children $x' \in S_{i-1}$, possibly with $x = x'$, with $\|x - x'\| \leqslant 2^i$.
- As a consequence, the subtree for any $x \in S_i$ has distance at most 2^{i+1} from x.

To analyze the performance of cover tree, we need the concept of expansion rate of a set, defined by Karger and Ruhl (2002), which captures several key properties.

Definition 8.1 (Expansion Rate) *Denote by $B_\rho(r)$ a ball of radius of r at ρ. Then a set S has a (l, c) expansion rate iff all $r > 0$ and $\rho \in S$ satisfy*

$$|B_\rho(r) \cap S| \geqslant l \implies |B_\rho(2r) \cap S| \leqslant c\,|B_\rho(r) \cap S|\,. \tag{8.4}$$

In the following we set $l = O(\log |S|)$, thus referring to c simply as the expansion rate of S.

With this definition, this data structure has a number of highly desirable properties, as proved in Beygelzimer, Kakade, and Langford (2006). We list the most relevant ones below:

- The depth of the tree in terms of its explicit representation is at most $O(c^2 \log n)$.
- The maximum degree of any node is $O(c^4)$.
- Insertion & removal take at most $O(c^6 \log n)$ time.
- Retrieval of the nearest neighbor takes at most $O(c^{12} \log n)$ time.
- The time to construct the tree is $O(c^6 n \log n)$.

8.3 OUR APPROACH

Now we introduce notation and explain details of our approach when the number of clusters is (a) moderate (Section 8.3.1) and (b) large (Section 8.3.2). In what follows, the number of data points and clusters are denoted with n and m respectively. The function $\mathrm{ch}(x)$ returns children of a node x of any tree.

DATA TREE ($\mathbb{T}_D$): Cover tree built with levels S_j on all available data using the sufficient statistic $\phi(x)$, constructed *once* for our setup. We record ancestors at level $\bar{\jmath}$ as prototypes $\bar{x}$ for each data point x. In fact, we only need to construct the tree up to a fixed degree of accuracy $\bar{\jmath}$ in case of moderate number of clusters. A key observation is that multiple points can have a same prototype $\bar{x}$, making it a many-to-one map. This helps us amortize costs over points by re-using proposal computed with $\bar{x}$ (Section 8.3.1).

CLUSTER TREE ($\mathbb{T}_C$): Similarly, $\mathbb{T}_C$ is the cover tree generated with cluster parameters $\tilde{\theta}_z$. For simplicity, we assume that the expansion rates of clusters and data are both c.

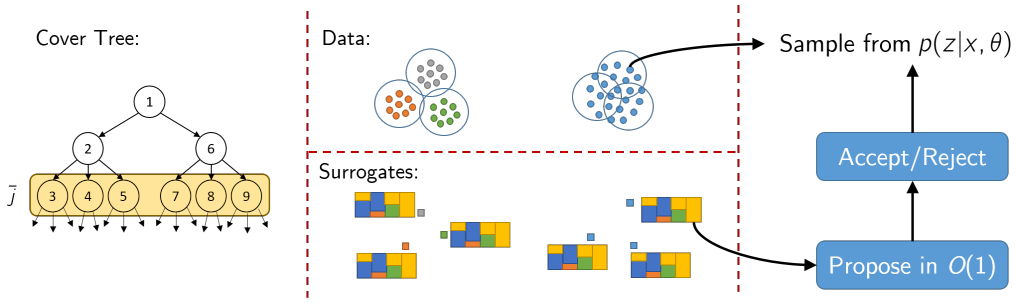


Figure 8.3: Overview of Canopy sampler, when the number of clusters is relatively small compared to the total number of observations: (1) Build a cover tree on data points with cost $O(n \log n)$. (2) Truncate tree at an accuracy level $\bar{j}$ such that nodes have $O(m)$ descendants on average. (3) Build alias tables for these surrogate points $\bar{x}$ points with total cost $O(n)$. (4) For each observation x perform rejection sampling using the alias table as the proposal in $O(1)$ time.

8.3.1 *Canopy I: Moderate number of clusters*

We introduce our sampler, Canopy I, when the number of clusters is relatively small compared to the total number of observations. This addresses many cases where we want to obtain a flat clustering on large datasets. For instance, it is conceivable that one might not want to infer more than a thousand clusters for one million observations. In a nutshell, our approach is illustrated in Figure 8.3 and described below:

1. Construct $\mathbb{T}_D$ and pick a level $\bar{j} \in \mathbb{Z}$ with accuracy $2^{\bar{j}}$ such that the average number of elements per node in $S_{\bar{j}}$ is $O(m)$.
2. For each of the prototypes $\bar{x}$, which are members of $S_{\bar{j}}$, compute $p(z|\bar{x})$ using the alias method to draw from m components θ_z. By construction, this cost amortizes $O(1)$ per observation, *i.e.*, a total cost of $O(n)$.
3. For each observation x with prototype $\bar{x}$, perform rejection sampling using the draws from $p(z|\bar{x}) =: q(z)$ as proposal. Hence we accept a sample z with probability

$$\mathcal{A}(z) := \frac{p(z|x)}{M p(z|\bar{x})}, \text{ where } M = \max_z \frac{p(z|x)}{p(z|\bar{x})}. \tag{8.5}$$

The key reason why this algorithm has a useful acceptance probability is that the normalizations for $p(z|x)$ and $p(z|\bar{x})$, and mixing proportions $p(z)$ cancel out respectively. Only terms remaining in (8.5) for M, the expected number of rejections is

$$M = \max_z \frac{p(z|x)}{p(z|\bar{x})} = \max_z \exp\left(\langle \phi(x) - \phi(\bar{x}), \tilde{\theta}_z \rangle\right) = \max_z \exp\left(\|\phi(x) - \phi(\bar{x})\| \left\|\tilde{\theta}_z\right\|\right) \leqslant e^{2^{\bar{j}+1} L}$$

for $\left\|\tilde{\theta}_z\right\| \leqslant L$. This follows from the Cauchy Schwartz inequality and the covering property of cover trees, which ensures all descendants of $\bar{x}$ are no more than $2^{\bar{j}+1}$ apart from $\bar{x}$, *i.e.*, $\|\phi(x) - \phi(\bar{x})\| \leqslant 2^{\bar{j}+1}$. Thus, we can bound the expected number of rejections by selecting the accuracy level $\bar{j}$, thereby controlling the efficiency of the sampler.

8.3.2 *Canopy II: Large number of clusters*

The key difficulty in dealing with many clusters is that it forces us to truncate $\mathbb{T}_D$ at a granularity in x that is less precise than desirable in order to benefit from the alias sampler naively. In other words, for a given sampling complexity, a larger m reduces the affordable granularity in

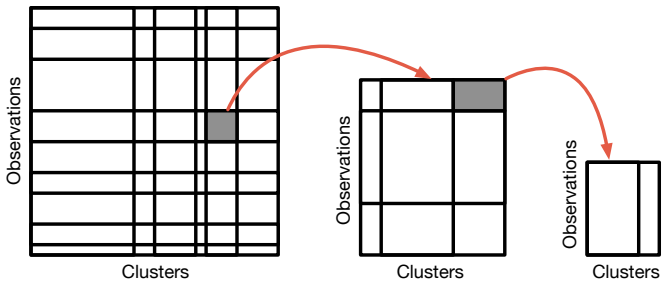


Figure 8.4: Hierarchical partitioning over both data observations and clusters. Once we sample clusters at a coarser level, we descend the hierarchy and sample at a finer level, until we have few number of points per cluster. We then use Section 8.3.2.1 for rejection sampler.

x. The problem arises because we are trying to distinguish clusters at a level of resolution that is too coarse. A solution is to apply cover trees not only to observations but also to the clusters themselves, *i.e.*, use both $\mathbb{T}_D$ and $\mathbb{T}_C$. This allows us to decrease the minimum observation-group size at the expense of having to deal with an *aggregate* of possible clusters.

Our method for large number of clusters operates in two phases: (a) Descend the hierarchy in cover trees while sampling (Section 8.3.2.1) (b) Sample for a single observation x from a subset of clusters arranged in $\mathbb{T}_C$ (Section 8.3.2.2), when appropriate conditions are met in (a). We begin with initialization and then elaborate each of these phases in detail.

INITIALIZE 1: Construct $\mathbb{T}_C$ and for each node θ_z, assign $\alpha(i,z) = p(z)$, where i is the highest level S_i such that $z \in S_i$, else 0. Then perform bottom-up aggregation via

$$\beta(i,z) = \alpha(i,z) + \sum_{z' \in \mathsf{ch}(z)} \beta(i+1,z') \tag{8.6}$$

This creates at most $O(m)$ distinct entries $\beta(i,z)$. Notice that aggregated value $\beta(i,z)$ captures the mixing probability of the node and its children in $\mathbb{T}_C$.

INITIALIZE 2: Partition both the observations and the clusters at a resolution that allows for efficient sampling and precomputation. More specifically, we choose accuracy levels $\hat{\jmath}$ and $\hat{\imath}$ to truncate $\mathbb{T}_D$ and $\mathbb{T}_C$, so that there are n' and m' nodes respectively after truncation. These serve as partitions for data points and clusters such that $n' \cdot m' = O(m)$ is satisfied. The aggregate approximation error

$$\delta := 2^{\hat{\jmath}+1} L + 2^{\hat{\imath}+1} R + 2^{\hat{\imath}+\hat{\jmath}+2} \tag{8.7}$$

due to quantizing observations and clusters is minimized over the split, searching over the levels.

8.3.2.1 *Descending* $\mathbb{T}_D$ *and* $\mathbb{T}_C$

Given $\mathbb{T}_D$ and $\mathbb{T}_C$ with accuracy levels $\hat{\jmath}$ and $\hat{\imath}$, we now iterate over the generated hierarchy, as shown in Figure 8.4. We recursively descend simultaneously in both the trees until the number of observations for a given cluster is too small. In that case, we simply default to the sampling algorithm described in Section 8.3.2.2 for each observation in a given cluster.

The reasoning works as follows: Once we have the partitioning into levels $\hat{\jmath}, \hat{\imath}$ for data and clusters respectively with $n' \cdot m' = O(m)$, we draw from the proposal distribution

$$q(\bar{z}|x) \propto \beta(\hat{\imath}, \bar{z}) \exp\left(\langle \phi(\bar{x}), \theta_{\bar{z}} \rangle - g(\theta_{\bar{z}})\right) \tag{8.8}$$

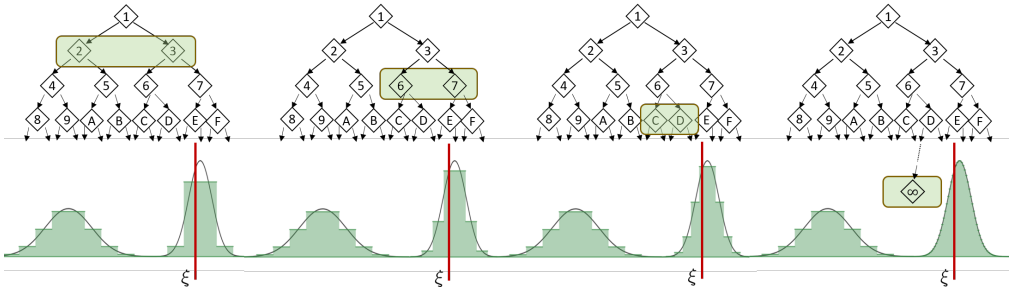


Figure 8.5: Caption

for all the observations and clusters above the partitioned levels $\hat{\jmath}$ and $\hat{\imath}$, respectively. That is, we draw from a distribution where both observations and clusters are grouped. We draw from the proposal for each x in $\mathbb{T}_D$ truncated at level $\hat{\jmath}$. Here, $\beta(\hat{\imath}, \bar{z})$ collects the prior cluster likelihood from $\bar{z}$ and all its children. As described earlier, we can use the alias method for sampling efficiently from (8.8).

Within each group of observations, drawing from (8.8) leads to a distribution over a (possibly smaller) subset of cluster groups. Whenever the number of observations per cluster group is small, we default to the algorithm described in Section 8.3.2.2 for each observation. On the other hand, if we have a sizable number of observations for a given cluster, which should happen whenever the clusters are highly discriminative for observations (a desirable property for a good statistical model), we repeat the strategy on the subset to reduce the aggregate approximation error (8.7). In other words, we descend the hierarchy to yield a new pair (i', j') on the subset of clusters/observations with $i' < \hat{\imath}$ and $j' < \hat{\jmath}$ and repeat the procedure.

The process works in a depth-first fashion in order to avoid using up too much memory. The sampling probabilities according to (8.8) are multiplied out for the path over the various hierarchy levels and used in a rejection sampling procedure. Each level of the hierarchy can be processed in $O(1)$ operations per instance, without access to the instance itself. Moreover, we are guaranteed to descend by at least one step in the hierarchy of observations and clusters, hence the cost is at most $O(c^2 \min(\log n, \log m))$.

To summarize, we employ a rejection sampling scheme as before, with the aim of using a highly accurate, yet cheap proposal. To overcome the loss in precision of Canopy I proposal due to large number of clusters, we devise a proposal wherein we look at aggregates of both data and clusters at comparable granularity using both $\mathbb{T}_D$ and $\mathbb{T}_C$. Note that the acceptance probabilities are always at least as high as the bounds derived in Section 8.3.1 as the errors on the paths are log-additive. Instead of rejection sampler, a MH procedure can also be devised. Details are omitted for the sake of brevity, since they mirror the single-observation argument of the following section.

8.3.2.2 *Sampling for a single observation* x

Let x be the single observation for which we want to sample from possibly subset of clusters z that are arranged in $\mathbb{T}_C$. In this case, we hierarchically descend $\mathbb{T}_C$ using each aggregate as a proposal for the clusters below. As before, we can either use MH sampling or a rejection sampler. To illustrate the effects of the latter, we describe one below, whose theoretical analysis is provided in Section 8.4. Before we delve into details, let us consider a simple case without $\mathbb{T}_C$. If we are able to approximate $p(x|\theta_z)$ by some q_z such that

$$e^{-\epsilon} p(x|\theta_z) \leqslant q_z \leqslant e^{\epsilon} p(x|\theta_z) \tag{8.9}$$

for all z, then it follows that a sampler drawing z from

$$z \sim \frac{q_z p(z)}{\sum_{z'} q_{z'} p(z')} \tag{8.10}$$

and then accepting with probability $e^{-\epsilon} q_z^{-1} p(x|\theta_z)$ will draw from $p(z|x)$ (see Section 8.4.1 for details). Moreover, the acceptance probability is at least $e^{-2\epsilon}$. However, finding such q_z with a small ϵ is not easy in general. Thus, we propose to cleverly utilize structure of the cover tree $\mathbb{T}_C$ to begin with a very course approximation and successively improving the approximation only for a subset of θ_z which are of interest. The resultant sampler is described below:

1. Choose approximation level $\hat{\imath}$ and compute normalization at accuracy level $\hat{\imath}$:

$$\gamma_0 := \sum_{z \in S_{\hat{\imath}}} \beta(\hat{\imath}, z) \exp \left\langle \tilde{\theta}_z, \tilde{\phi}(x) \right\rangle . \tag{8.11}$$

2. Set $e^{-\epsilon} := e^{-2^{\hat{\imath}} \|\tilde{\phi}(x)\|}$ as multiplier for the acceptance threshold of the sampler and $\gamma := e^{\epsilon} \gamma_0$.
3. Draw a node $z \in S_{\hat{\imath}}$ with probability $\delta_z := \gamma^{-1} e^{\epsilon} \beta(\hat{\imath}, z) \exp \left\langle \tilde{\theta}_z, \tilde{\phi}(x) \right\rangle$.
4. Accept $z_{\hat{\imath}}$ at the current level with probability $\pi := \gamma^{-1} \delta_{z_{\hat{\imath}}}^{-1} p(z_{\hat{\imath}}) \exp \left\langle \tilde{\theta}_{z_{\hat{\imath}}}, \tilde{\phi}(x) \right\rangle$.
5. For $i := \hat{\imath} - 1$ down to $-\infty$ do
   a) Set $e^{-\epsilon} := e^{-2^i \|\tilde{\phi}(x)\|}$ as the new multiplier and $\gamma := \delta_{z_{i+1}}(1 - \pi)$ as the new normalizer.
   b) Draw one of the children z of z_{i+1} with probability $\delta_z := \gamma^{-1} e^{\epsilon} \beta(i, z) \exp \left\langle \tilde{\theta}_z, \tilde{\phi}(x) \right\rangle$. Exit if we do not draw any of them (since $\sum_{z \in \mathsf{ch}(z_{i+1})} \delta_z \leqslant 1$) and restart from step 2, else denote this child by z_i.
   c) Accept z_i at the current level with probability $\pi := \gamma^{-1} \delta_{z_i}^{-1} p(z_i) \exp \left\langle \tilde{\theta}_{z_i}, \tilde{\phi}(x) \right\rangle$. Do *not include* z_{i+1} in this setting. as we consider z only the first time we encounter it.

The above describes a rejection sampler that keeps on upper-bounding the probability of accepting a particular cluster or any of its children. It is as aggressive as possible at retaining tight lower bounds on the *acceptance* probability such that not too much effort is wasted in traversing the cover tree to the bottom, *i.e.*, we attempt to reject as quickly as possible.

8.4 THEORETICAL ANALYSIS

8.4.1 *Correctness of Sampler*

For ease of explanation, we first consider how to use a simple proposal distribution that meets some approximation bounds. Next, we show how using cover tree we can construct such a proposal meeting the approximation bounds and its use in inference.

8.4.1.1 *Flat Proposal*

The proof for the proposed rejection sampler in case of sampling a cluster for a single observation x is as follows. If we approximate $p(x|\theta_z)$ by some q_z such that

$$e^{-\epsilon} p(x|\theta_z) \leqslant q_z \leqslant e^{\epsilon} p(x|\theta_z) \tag{8.12}$$

then it follows that a sampler drawing z from

$$z \sim \frac{q_z p(z)}{\sum_{z'} q_{z'} p(z')} \tag{8.13}$$

and then accepting with probability $e^{-\epsilon} q_z^{-1} p(x|\theta_z)$ will draw from $p(z|x)$. To prove this, we begin by computing the probability of this sampler $r(z)$ to return a particular value z. The sampler returns z when it (a) samples and accepts z, or (b) samples any value, rejects it to proceed to next iteration of sampling. Using $\gamma = \sum_{z'} q_{z'} p(z')$ and $\gamma_T = \sum_{z'} p(x|\theta_{z'}) p(z')$ to denote normalization for proposal and true posterior respectively, we have:

$$\begin{aligned} r(z) &= \frac{q_z p(z)}{\gamma} e^{-\epsilon} q_z^{-1} p(x|\theta_z) + \sum_{z'} (1 - e^{-\epsilon} q_{z'}^{-1} p(x|\theta_{z'})) \frac{q_{z'} p(z')}{\gamma} r(z) \\ &= \frac{e^{-\epsilon}}{\gamma} p(z) p(x|\theta_z) + \frac{r(z)}{\gamma} \sum_{z'} q_{z'} p(z') - r(z) \frac{e^{-\epsilon}}{\gamma} \sum_{z'} p(x|\theta_{z'}) p(z') \\ &= \frac{e^{-\epsilon}}{\gamma} p(z) p(x|\theta_z) + r(z) - r(z) \frac{e^{-\epsilon}}{\gamma} \gamma_T \\ r(z) &= \frac{p(z) p(x|\theta_z)}{\gamma_T} \end{aligned} \tag{8.14}$$

Hence the procedure will draw from the true posterior $p(z|x)$.

8.4.1.2 *Proposal in Cover Tree*

We now extend the above proof strategy when the clusters are arranged in a cover tree, thereby proving the correctness of our rejection sampler in Section 8.3.2.2.

Similar to previous case, we approximate $p(x|\theta_z)$ for z in level i by some q_z such that

$$e^{-\epsilon_i} p(x|\theta_z) \leqslant q_z \leqslant e^{\epsilon_i} p(x|\theta_z). \tag{8.15}$$

Note that approximation error ϵ_i now depends on the location of the cluster in the cover tree. To be specific, if the cluster z is located at level i, then $\epsilon_i = 2^i \|\bar{\phi}(x)\|$. Also, we assume the path to reach the node z starting from its (grand) parent at level $\hat{\imath}$ is given by $\mathcal{T} = [\mathcal{T}(\hat{\imath}), \mathcal{T}(\hat{\imath} - 1), ..., \mathcal{T}(i)]$, with $\mathcal{T}(i) = z$.

To prove the correctness of our rejection sampler in Section 8.3.2.1, we simply show that probability of this sampler to return a particular value z is equal to the true posterior. The sampler returns z when it (a) reaches the corresponding node in the cover tree and accepts it, or (b) rejects or exits to proceed to next iteration of sampling. So, the probability of this sampler to return z is given by:

$$r(z) = \underbrace{\mathcal{A}(z)}_{\substack{\text{Probability of the} \\ \text{sampler accepting } z}} + r(z) \underbrace{\mathcal{E}}_{\substack{\text{Probability of the} \\ \text{sampler rejecting or exiting}}} \tag{8.16}$$

We calculate these individual terms beginning with the probability of sampler accepting z. Using γ as defined in (8.11) and $\gamma_T = \sum_{z'} p(x|\theta_{z'})p(z')$, we have:

$$\begin{aligned}
\mathcal{A}(z) &= \underbrace{\frac{p(z)p(x|\theta_z)}{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})}}_{\text{Accepting node } z} \times \underbrace{\frac{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})}{\gamma}}_{\text{Selecting the first parent (Step 4)}} \\
&\times \left[\underbrace{\prod_{i'=i+1}^{\hat{i}}}_{\substack{\text{The loop} \\ \text{(Step 5)}}} \underbrace{\left(1 - \frac{p(\mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})}{e^{\epsilon_{i'}}\beta(i',\mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})}\right)}_{\text{Rejecting the nodes (Step 5iii)}} \underbrace{\frac{e^{\epsilon_{i'-1}}\beta(i'-1,\mathcal{T}(i'-1))p(x|\theta_{\mathcal{T}(i'-1)})}{e^{\epsilon_{i'}}\beta(i',\mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')}) - p(\mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})}}_{\text{Selecting the next node (Step 5ii)}} \right] \\
&= \frac{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})}{\gamma} \left[\prod_{i'=i+1}^{\hat{i}} \frac{e^{\epsilon_{i'-1}}\beta(i'-1,\mathcal{T}(i'-1))p(x|\theta_{\mathcal{T}(i'-1)})}{e^{\epsilon_{i'}}\beta(i',\mathcal{T}(i'))p(x|\theta_{\mathcal{T}(i')})} \right] \frac{p(z)p(x|\theta_z)}{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})} \\
&= \frac{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})}{\gamma} \left[\frac{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})}{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})} \right] \frac{p(z)p(x|\theta_z)}{e^{\epsilon_{\hat{i}}}\beta(\hat{i},\mathcal{T}(\hat{i}))p(x|\theta_{\mathcal{T}(\hat{i})})} \\
&= \frac{p(z)p(x|\theta_z)}{\gamma} \qquad \text{(by telescoping)}
\end{aligned} \tag{8.17}$$

Next, the probability of rejecting or exiting from the sampler is one minus probability of accepting any node z, *i.e.*

$$\begin{aligned}
\mathcal{E} &= 1 - \sum_{z' \in \mathcal{Z}} \mathcal{A}(z') \\
&= 1 - \sum_{z' \in \mathcal{Z}} \frac{p(z')p(x|\theta_{z'})}{\gamma} \\
&= 1 - \frac{\gamma_T}{\gamma}
\end{aligned} \tag{8.18}$$

Plugging back the acceptance and exit probabilities into (8.16):

$$\begin{aligned}
r(z) &= \mathcal{A}(z) + r(z)\mathcal{E} \\
&= \frac{p(z)p(x|\theta_z)}{\gamma} + r(z)\left(1 - \frac{\gamma_T}{\gamma}\right) \\
&= \frac{p(z)p(x|\theta_z)}{\gamma} + r(z) - r(z)\frac{\gamma_T}{\gamma} \\
r(z) &= \frac{p(z)p(x|\theta_z)}{\gamma_T}
\end{aligned} \tag{8.19}$$

Hence the procedure will draw from the true posterior $p(z|x)$.

The above describes a rejection sampler that keeps on upper-bounding the probability of accepting a particular parameter or any of its children. It is as aggressive as possible at retaining tight lower bounds on the *acceptance* probability such that not too much effort is wasted in traversing the cover tree to he bottom. In other words, we attempt to reject as quickly as possible. Some computational considerations are in order:

1. The computationally most expensive part is to compute the inner products $\langle \tilde{\phi}(x), \tilde{\theta}_z \rangle$.
2. As soon as we compute this value for a particular $\tilde{\theta}_z$ we cache it at the corresponding vertex of the cover tree.
3. To avoid expensive bookkeeping we attach to each vertex two variables: the value of the last compute inner product and the observation ID of x that it is associated with. +

8.4.2 *Runtime of Sampler*

The main concern is to derive a useful bound regarding the runtime required for drawing a sample. Secondary concerns are those of generating the data structure. We address each of these components, reporting all costs per data point.

CONSTRUCTION The data structure $\mathbb{T}_D$ costs $O(c^6 \log n)$ (per data-point) to construct and $\mathbb{T}_C$ costs $O(c^6 \log m)$ (per data-point, as $m < n$) — all additional annotations cost negligible time and space. This includes computing α and β, as discussed above.

STARTUP The first step is to draw from S_i. This costs $O(|S_i|)$ for the first time to compute all probabilities and to construct an alias table. Subsequent samples only cost 3 CPU cycles to draw from the associated alias table. The acceptance probability at this step is $e^{-2\epsilon}$. Hence the aggregate cost for the top level is bounded by $O\left(|S_i| + e^{2^i \|\tilde{\Phi}(x)\|}\right)$.

TERMINATION To terminate the sampler successfully, we need to traverse $\mathbb{T}_C$ at least once to its leaf in the worst case. This costs $O(c^2 \log m)$ if the leaf is at maximum depth.

REJECTIONS The main effort of the analysis is to obtain useful guarantees for the amount of effort wasted in drawing from the cover tree. A brute-force bound immediately would yield $O\left(e^{2^i \|\tilde{\Phi}(x)\|} c^6 \log m\right)$. Here the first term is due to the upper bound on the acceptance probability, a term of c^4 arises from the maximum number of children per node and lastly the $c^2 \log m$ term quantifies the maximum depth. It is quite clear that this term would dominate all others. We now derive a more refined (and tighter) bound.

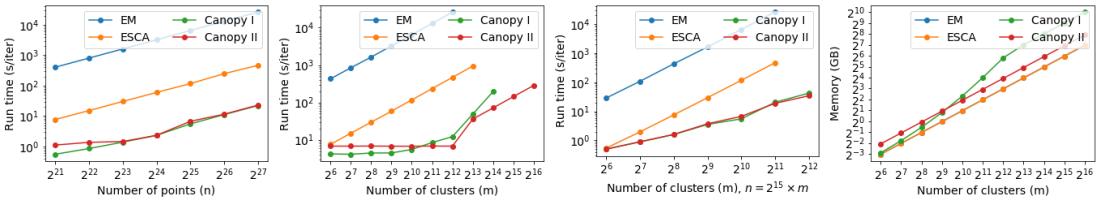
Essentially we will exploit the fact that the deeper we descend into the tree, the less likely we will have wasted computation later in the process. We use the following relations

$$e^x - 1 \leqslant e^a \frac{x}{a} \text{ for } x \in [0, a] \text{ and } \sum_{i=1}^{\infty} 2^{-i} = 1. \tag{8.20}$$

In expectation, the first step of the sampler requires $e^\epsilon = e^{2^i \|\tilde{\Phi}(x)\|}$ steps in expectation until a sample is accepted. Thus, $e^\epsilon - 1$ effort is wasted. At the next level below we waste at most $e^{2^{i-1} \|\tilde{\Phi}(x)\|} - 1$ effort. Note that we are less likely to visit this level commensurate with the acceptance probability. These bounds are conservative since any time we terminate above the very leaf levels of the tree we are done. Moreover, not all vertices have children at all levels, and we only need to revisit them whenever they do. In summary, the wasted effort can be bounded from above by

$$c^4 \sum_{i=1}^{\infty} \left[e^{2^{i-i} \|\tilde{\Phi}(x)\|} - 1 \right] \leqslant c^4 e^{2^i \|\Phi(x)\|} \sum_{i=1}^{\infty} 2^{-i} = c^4 e^{2^i \|\Phi(x)\|}.$$

Here c^4 was a consequence of the upper bound on the number of children of a vertex. Moreover, note that the exponential upper bound is rather crude, since the inequality (8.20) is very loose for large a. Nonetheless we see that the rejection sampler over the tree has computational



(a) Varying n with fixed $(m, d) = (2^{10}, 2^{10})$

(b) Varying m with fixed $(n, d) = (2^{25}, 2^{10})$

(c) Varying m with $n = 2^{15}m$ and $d = 2^{10}$

(d) Varying m with fixed $(n, d) = (2^{25}, 2^{10})$

Figure 8.6: Showing scalability of per-iteration runtime of different algorithms with increasing dataset size. From Figure 8.6a, Figure 8.6b, and Figure 8.6c we see that our approaches take orders of magnitude less time compared to the traditional EM and ESCA methods, while varying the number of points and clusters respectively. Note that we trade off memory for speed as seen from Figure 8.6d. For instance, with $(n, m, d) = (32\text{mil}, 4096, 1024)$, we see that there is a speed-up of $150\times$ for a mere $2\times$ memory overhead.

overhead independent of the tree size! This result is less surprising than it may seem. Effectively we pay for lookup plus a modicum for the inherent top-level geometry of the set of parameters.

Theorem 8.2 *The cover tree sampler incurs worst-case computational complexity per sample of*

$$O\left(|S_{\hat{\imath}}| + c^6 \log n + c^6 \log m + c^4 e^{2\hat{\imath}} \left\|\tilde{\phi}(x)\right\|\right) \tag{8.21}$$

Note that the only data-dependent terms are $c, S_{\hat{\imath}}, \hat{\imath}$ and $\left\|\tilde{\phi}(x)\right\|$ and that nowhere the particular structure of $p(z)$ entered the analysis. This means that our method will work equally well regardless of the type of latent variable model we apply. For example, we can even apply the model to more complicated latent variable models like latent Dirichlet allocation (LDA). The aforementioned constants are all natural quantities inherent to the problems we analyze. The constant c quantifies the inherent dimensionality of the parameter space, $\left\|\tilde{\phi}(x)\right\|$ measures the dynamic range of the distribution, and $S_{\hat{\imath}}, \hat{\imath}$ measure the "packing number" of the parameter space at a minimum level of granularity.

8.5 EXPERIMENTS

We now present empirical studies for our fast sampling techniques in order to establish that (i) Canopy is fast (Section 8.5.1), (ii) Canopy is accurate (Section 8.5.2), and (iii) it opens new avenues for data exploration and unsupervised learning (Section 8.5.3), previously unthinkable. To illustrate these claims, we evaluate on finite mixture models, more specifically, Gaussian Mixture models (GMM), a widely used probabilistic models. However, the proposed method can be applied effortlessly to any latent variable model like topic modeling through Gaussian latent Dirichlet allocation (Gaussian LDA; Zaheer, Das, and Dyer 2015). We pick GMMs due to their wide-spread application in various fields spanning computer vision, natural language processing, neurobiology, *etc*.

METHODS For each experiment, we compare our two samplers (Canopy I, Section 8.3.1 and Canopy II, Section 8.3.2) with both the traditional Expectation Maximization (EM; Dempster, Laird, and Rubin 1977) and the faster Stochastic EM through ESCA (ESCA; Zaheer, Wick, et al. 2015) using execution time, cluster purity, and likelihood on a held out TEST set.

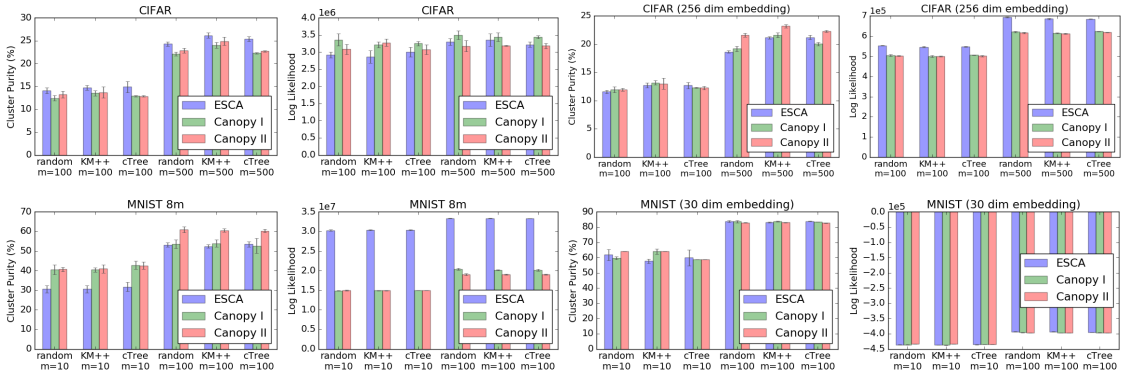


Figure 8.7: Plots of cluster purity and loglikehood of ESCA, Canopy I, and Canopy II on benchmark real datasets –MNIST8m and CIFAR-100. All three methods have roughly same performance on cluster purity. See Section 8.5.2 for more details.

SOFTWARE & HARDWARE All the algorithms are implemented multithreaded in simple C++11 using a distributed setup. Within a node, parallelization is implemented using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the process binding to a socket over MPI. We run our experiments on a cluster of 16 Amazon EC2 c4.8xlarge nodes connected through 10Gb/s Ethernet. There are 36 virtual threads per node and 60GB of memory. For purpose of experiments, all data and calculations are carried out at double floating-point precision.

INITIALIZATION Recall that speed and quality of inference algorithms depend on initialization of the random variables and parameters. Random initializations often lead to poor results, and so many specific initialization schemes have been proposed, like KMeans++ (Arthur and Vassilvitskii, 2007), K-MC2 (Bachem et al., 2016). However, these initializations can be costly, roughly $O(mn)$.

Our approach provides a good initialization using cover trees free of cost, as the construction of cover tree is at the heart of our sampling approach. The proposed initialization scheme relies on the observation that cover trees partition the space of points while preserving important invariants based on its structure. They thus help in selecting initializations that span the entirety of space occupied by the points, which is desired to avoid local minima. The crux of the approach is to descend to a level l in $\mathbb{T}_D$ such that there are no more than m points at level l. These points from level l are included in set of initial points I. We then randomly pick a point from I such that it belongs to level l and replace it with its children from level $l+1$ in I. This is repeated until we finally have m elements in I. The chosen m elements are mapped to parameter space through the inverse link function $g^{-1}(\cdot)$ and used as initialization. All our experiments use *cover tree based* initializations. We also make comparisons against *random* and *KMeans++* in Section 8.5.2.

8.5.1 *Speed*

To gauge the speed of Canopy, we begin with inference on GMMs using *synthetic data*. Working with synthetic data is advantageous as we can easily vary parameters like number of clusters, data points, or dimensionality to study its effect on the proposed method. Note that, from a computational perspective, data being real or synthetic does not matter as all the required computations are data independent, once the cover tree has been constructed.

SYNTHETIC DATASET GENERATION Data points are assumed to be i.i.d. samples generated from m Gaussian probability distributions parameterized by (μ_i^*, Σ_i^*) for $i = 1, 2, \cdots, m$, which mix with proportions given by π_i^*. Our experiments operate on three free parameters: (n, m, d) where n is the total number of points, m is the number of distributions, and d is the dimensionality. For a fixed (n, m, d), we randomly generate a TRAIN set of n points as follows: (1) Randomly pick parameters (μ_i^*, Σ_i^*) along with mixing proportions π_i^*, for $i = 1, 2, \cdots, m$, uniformly random at some scale. (2) To generate each point, select a distribution based on $\{\pi_i^*\}$ and sample from the corresponding d-dimensional Gaussian pdf. Additionally, we also generate another set of points as TEST set using the same procedure. For all the four models (Canopy I, Canopy II, EM, ESCA), parameters are learnt using TRAIN and log-likelihood on the TEST set is used as evaluation.

OBSERVATIONS We run all algorithms for a *fixed number of iterations* and vary n, m, d individually to investigate the respective dependence on performance of our approach as shown in Figure 8.6. We make the following observations: (1) Overall, Figure 8.6 is in line with our claim that the proposed method reduced the per iteration complexity from $O(nm)$ of EM/ESCA to $\tilde{O}(n + m)$. (2) To illustrate this further, we consider $n = O(m)$ and vary m (shown in Figure 8.6c). While EM and ESCA have per-iteration time of $O(mn)$, *i.e.*, $O(m^2)$ in this case, our Canopy I and Canopy II show $\tilde{O}(m + n)$, *i.e.*, $\tilde{O}(m)$. (3) However, there is no free lunch. The huge speed-up comes at the cost of increased memory usage (for storing the data-structures). For example, in the case of $n = 32$ mil, $m = 4096$, and $d = 1024$ (Figure 8.1), a mere $2\times$ increase in memory gets us a speed up of $150\times$.

8.5.2 *Correctness*

Next, we demonstrate correctness of Canopy using medium sized real world datasets with labels, *i.e.*, ground truth grouping of the points are known. We setup an unsupervised classification task on these datasets and perform evaluation on both cluster purity and log-likelihood.

DATASETS We use two benchmark image datasets–MNIST8m (Loosli, Canu, and Léon Bottou, 2007) and CIFAR-100 (Krizhevsky and G. Hinton, 2009). The former contains *8 million* annotated handwritten digits of size 28×28, giving us data points of dimension 784. CIFAR-100, on the other hand, contains 50k images annotated with one of 100 object categories. Each image has 3 channels (RGB) and of size 32×32, resulting in a vector of dimension 3072.

UNSUPERVISED CLASSIFICATION. We run unsupervised classification on the above two datasets and evaluate using cluster purity and log-likelihood. Here, cluster purity is defined as the mean of accuracy across all clusters, where each cluster is assigned the class of majority of its members. In addition to using data points as is, we also experiment with unsupervised features learnt from a denoising autoencoder (Geoffrey E Hinton and R. R. Salakhutdinov, 2006). We extract 30 and 256 dimensional features for MNIST8m and CIFAR-100 respectively. Details of our unsupervised feature extraction are provided below. Further, we evaluate in multiple scenarios that differ in (a) number of clusters: $m = 10, 100$ for MNIST8m and $m = 100, 500$ for CIFAR-100, and (b) parameter initializations (Random, Kmeans++ and CTree).

DENOISING AUTOENCODER FOR MNIST The autoencoder consists of an encoder with fully connected layers of size (28x28)-1000-500-250-30 and a symmetric decoder. The thirty units in the code layer were linear and all the other units were logistic. The network was trained on the 8 million images using mean square error loss.

DENOISING AUTOENCODER FOR CIFAR100 The autoencoder consists of an encoder with convolutional layers of size (3x32x32)-(64, 5, 5)-(32, 5, 5)-(16, 4, 4) and having a 2x2 max pool-

Figure 8.8: Illustration of concepts captured by clustering images in the ResNet (He et al., 2016a,b) feature space. We randomly pick three clusters and show four closest images (one in each row), possibly denoting the semantic concepts of 'crowd', 'ocean rock scenery' and 'horse mounted police'. Our clustering discovers new concepts beyond the Resnet supervised categories (does not include 'crowd').

ing after each convolutional layer. The decoder is symmetric with max pooling replaced by upsampling. The 256 units in the code layer were linear and all the other internal units were RelU while the final layer was sigmoid. The network was trained on the 50 thousand images using mean square error loss.

OBSERVATIONS Figure 8.7 shows our results on MNIST8m ($m = 10,100$) and CIFAR-100 ($m = 100,500$), with error bars computed over 5 runs. Here are the salient observations: (1) All the methods (SEM, Canopy I, Canopy II) have roughly the same cluster purity with Canopy II outperforming in CIFAR-100 (256 dim) and MNIST8m by around 10% and 3% respectively. In CIFAR-100, SEM does slightly better than other methods by 2-3%. (2) Similar results are obtained for log-likelihood except for MNIST8m, where SEM heavily outperforms Canopy. However, note that log-likelihood results in an unsupervised task can be misleading (J. Chang et al., 2009), as evidenced here by superior performance of Canopy in terms of cluster purity.

8.5.3 *Scalability - A New Hope*

Finally, we demonstrate the scalability of our algorithm by clustering a crawled dataset having more than 100 million images that belong to more than 80,000 classes. We query Flickr[1] with the key words from WordNet (Fellbaum, 1998) and downloaded the returned images for each key word, those images roughly belong to the same category. We extracted the image features of dimension 2048 with ResNet (He et al., 2016a,b) – the state-of-the-art convolutional neural network (CNN) on ImageNet 1000 classes data set–using publicly available pre-trained model of 200 layers[2]. It takes 5 days with 20 GPUs to extract these features for all the images. We then use Canopy II to cluster these images with $m = 64000$, taking around 27 hours.

1 http://www.flickr.com/
2 github.com/facebook/fb.resnet.torch

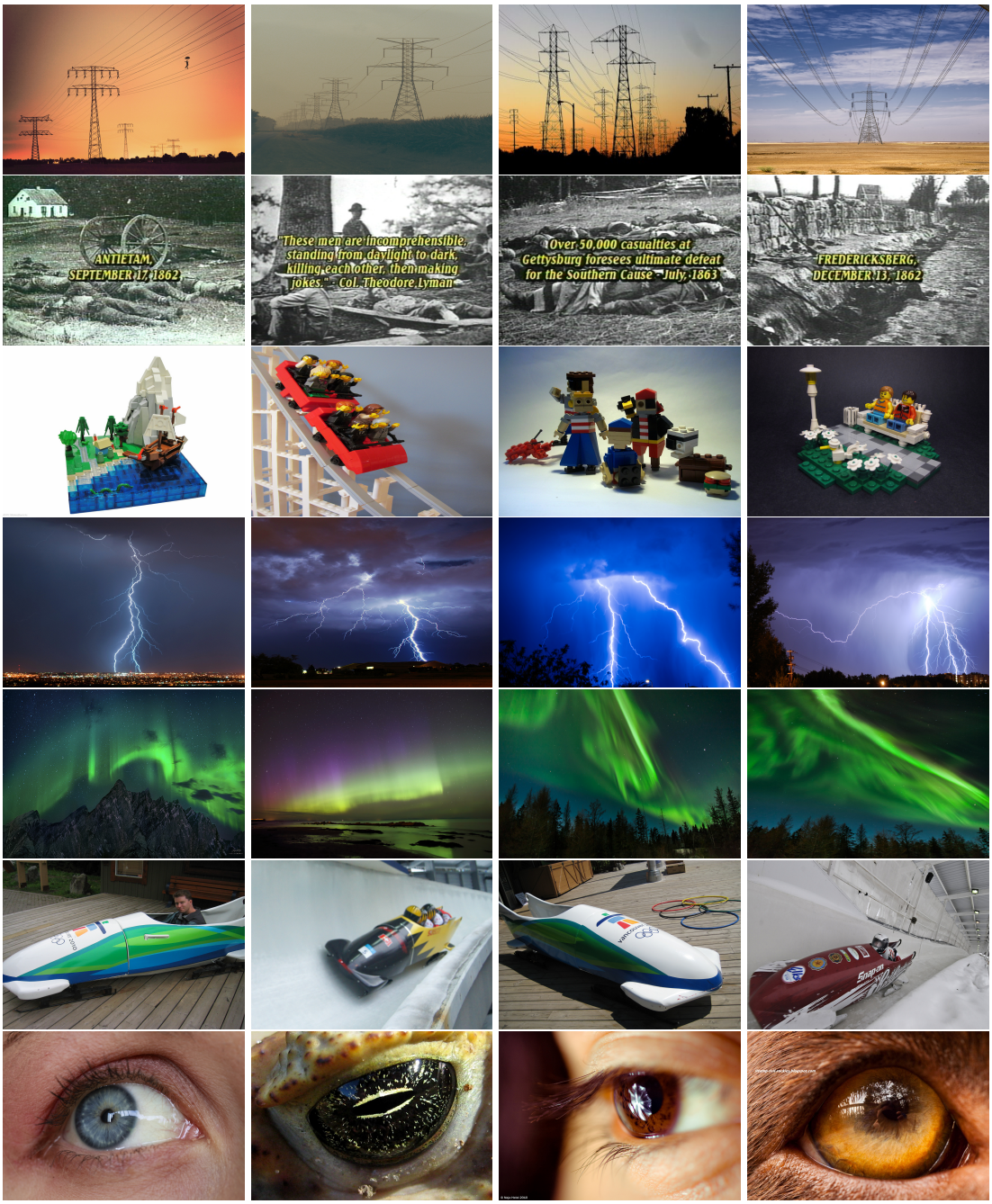


Figure 8.9: More Illustration of concepts captured by clustering images in the feature space extracted by ResNet (He et al., 2016a,b). Figure shows four closest images of seven more randomly selected clusters (one in each row) possibly denoting the semantic concepts of 'electrical transmission lines', 'image with text', 'lego toys', 'lightening', 'Aurora', 'buggy' and 'eyes'. Few of the concepts are discovered by clustering as Resnet received supervision only for 1000 categories (for example does not include label 'lightening', 'thunder', or 'storm'). Full set of 1000 imagenet label can be seen at `http://image-net.org/challenges/LSVRC/2014/browse-synsets`.

RESNET FOR IMAGENET We use the state of the art deep convolutional neural network (DCNN), based on the ResNet ("Residual Network") architecture (He et al., 2016a,b). ResNet consists of small building blocks of layers which learn the residual functions with reference to the input. It is demonstrated that ResNet is able to train networks that are substantially deeper without the problem of noisy backpropagation gradient. For feature extraction We use a 200 layer ResNet that is trained on a task of classification on ImageNet. In the process, the network learned which high-level visual features (and combinations of those features) are important. After training the model, we remove the final classification layer of the network and extract from the next-to-last layer of the DCNN, as the representation of the input image which is of dimension 2048.

OBSERVATIONS For a qualitative assessment, we randomly pick four clusters and show four images (more in Figure 8.9) closest to the means in Figure 8.8 (each cluster in a row). We highlight two important observations: (a) Though the underlying visual feature extractor, ResNet, is trained on 1000 semantic classes, our clustering is able to discover semantic concepts that go beyond. To illustrate, images from the first row indicate a semantic class of `crowd` even though ResNet never received any supervision for such a concept. (b) The keywords associated with these images do not necessarily collate with the semantic concepts in the image. For example, images in first row are associated with key words 'heave' ,'makeshift', 'bloodbath', and 'fullfillment', respectively. It is not too surprising as the relatedness of retrieved images for a query key word generally decreases for lower ranked images. This suggests that pre-processing images to obtain more meaningful semantic classes could potentially improve the quality of labels used to learn models. Such a cleanup would definitely prove beneficial in learning deep image classification models from weakly supervised data.

8.6 POINTS TO PONDER

We present an efficient sampler, *Canopy*, for mixture models over exponential families using cover trees that brings the per-iteration cost down from $O(mn)$ to $\tilde{O}(m+n)$. The use of cover trees over both data and clusters combined with alias sampling can significantly improve sampling time with no effect on the quality of the final clustering. We demonstrate speed, correctness, and scalability of Canopy on both synthetic and large real world datasets. To the best of our knowledge, our clustering experiment on a hundred million images is the largest to be reported. We conclude with some related works and future extensions.

RELATED WORKS There has been work using nearest-neighbor search for guiding graphical model inference like kd-trees (Gray and Moore, 2000; Moore, 1999). But use of kd-trees is not scalable with respect to dimensionality of the data points. Moreover, kd-trees could be deeper (especially for small c) and do not have special properties like covering, which can be exploited for speeding up sampling. We observe this empirically when training kd-tree based methods using publicly available code[3]. The models fail to train for dimensions greater than 8, or number of points greater than few thousands. In contrast, our method handles millions of points with thousands of dimensions.

FURTHER APPROXIMATIONS From our experiments, we observe that using a simplified single observation sampling in Canopy II works well in practice. Instead of descending on the hierarchy of clusters, we perform exact proposal computation for k closest clusters obtained through fast lookup from $\mathbb{T}_C$. All other clusters are equally assigned the least out of these k exact posteriors.

3 http://www.cs.cmu.edu/~psand/

In the future, we plan to integrate Canopy with:

CORESETS Another line of work to speed up mixture models and clustering involves finding a weighted subset of the data, called coreset (Feldman, Melanie Schmidt, and Sohler, 2013; Lucic, Bachem, and A. Krause, 2016). Models trained on the coreset are provably competitive with those trained on the original data set. Such approaches reduce the number of samples n, but perform traditional inference on the coreset. Thus, our approach can be combined with coreset for additional speedup.

INNER PRODUCT ACCELERATION In an orthogonal direction to Canopy, several works like A. Ahmed, Ravi, et al. (2012) and Mussmann and Ermon (2016) have used maximum inner product search to speed up inference and vice versa in Auvolat et al. (2015). We want to incorporate these ideas into Canopy as well, since the inner product is evaluated m times each iteration, it becomes the bottleneck for large m and d. A solution to overcome this problem would be to use binary hashing (A. Ahmed, Ravi, et al., 2012) as a good approximation and therefore a proposal distribution with high acceptance rate.

Combining these ideas, one could build an extremely scalable and efficient system, which potentially could bring down the per-iteration sampling cost from $O(mnd)$ to $\tilde{O}(m+n+d)$ or less!

9

Handling Variable State-Space Size

We study the problem of scaling up inference for latent variable models (LVM) over a variable size state space. Such variable size state space occur in non-parametric models, enabling LVMs to adapt in accordance to the size of data. This adaptability data sizes make non-parametric models, including Bayesian ones, a highly versatile tool for inferring the structures and hierarchies from unstructured data and they have become ubiquitous in statistical data analysis spanning a diverse set of applications from text to images to user behaviour. However, it is non-trivial to scale the inference procedures of LVM to deal with real life problems consisting of massive datasets in a reasonable time with reasonable resources and the variable size state space only aggravates the issue. In this chapter, we suggest a scalable inference procedure for Bayesian Nonparametrics based on a combination of smart data structures for handling probability distributions on one side and modified statistical inference algorithms on the other. In particular, we take up the task of topic modelling with Hierarchical Dirichlet Process (HDP) to demonstrate effectiveness of the proposed inference procedure. We propose a log structured alias sampling data structure for fast sampling from variable sized distribution and derive an alternative parametrization of HDP for decoupled posterior allowing parallelism. Experimental results for the proposed method on big datasets like PubMed and Wikipedia show 10x speed-up.

9.1 Introduction

Bayesian nonparametrics such as the Dirichlet Process (DP; Ferguson 1973), the Chinese Restaurant Process (Pitman, 1995), the Indian Buffet Process (IBP; T. Griffiths and Z. Ghahramani 2011) and the Hierarchical Dirichlet Process (HDP; Y. W. Teh et al. 2006) or the Chinese Restaurant Franchise (A. Ahmed, Hong, and A.J. Smola, 2013) pose to be successful alternatives to model selection and promising results have been reported for describing documents and networks, for modelling user interests and behaviours, for collaborative filtering as well as for priors in multi-task learning. However, inference procedures reported in the academic literature are limited and focused on providing proofs of concept. The scaling of these procedures to be able to deal with real life problems in industry consisting of massive datasets (1 TB+) in a reasonable time with reasonable resources is non-trivial.

One type of very celebrated LVM, the Topic models, are useful in modelling problems involving groups of data. In these problems each observation within a group is drawn from a mixture model and it is desirable to share mixture components across all the groups. They are very useful tools in discovering the hidden thematic structure in large archives of documents, and hence the name. Such categorization of data can help us perform tasks such as information retrieval, document browsing, content recommendation (e.g. articles to read, movies to watch, advertisements to display) etc.

Topic models achieve this goal by introducing a discrete latent factor, z, for each observation $x \in \mathcal{D}$. This latent factor corresponds to the hidden causes motivating this behaviour, e.g. the underlying theme of the document (finance, sports, etc.) or user tastes. Specifically, we model x as being generated from $p(x|z)$ and the latent factor has its own distribution $p(z)$. For instance, z may be the cluster indicator of a document, in which case it leads to the traditional mixture models. When z is a vector of fixed dimension (i.e. number of topics), it leads to Latent Dirichlet Allocation (LDA). However, we do not know the number of topics prevalent in a given corpus apriori and model selection for LDA is prohibitively expensive. Hence, Bayesian nonparametrics, like a DP has emerged as an attractive choice for $p(z)$, leading to the HDP topic model. Such a model is both a boon and a bane. On one hand it provides the much needed flexibility of adopting model complexity with the amount of data available (i.e. inherently performing model selection) and on the other hand such flexibility poses a bigger challenge for inference, which is already pretty complicated for LVM.

As mentioned before, given a large set of data $\mathcal{D}$ it is expensive to use traditional inference procedures for such fancy LVMs using Markov Chain Monte Carlo methods or stochastic variational algorithms. One of the key obstacles in performing scalable inference in topic models is to draw from $p(z|\mathcal{D})$ over the discrete state space associated with the latent factor given the data. The efficiency of this step is critical as it will be repeated for every token in the dataset (which run into billions) multiple times. For example, if we have a distribution over K outcomes and we draw only a single sample before the distribution changes, then each step will require $O(K)$ computation.

Recently, a big improvement for parametric topic models was proposed in A. Q. Li et al. (2014) by making the key observation that in many inference problems the model parameters only change relatively slowly during sampling (e.g., the location of cluster centres, the definition of topics) and appear as decoupled terms in the posterior. Leveraging this observation, they propose generating an alias table (A. Q. Li et al., 2014) for the slow varying parts of the distribution enabling $O(1)$ sampling from it. Next, to account for the slight drift in the distribution, they apply Metropolis-Hastings (MH) steps with samples from the stale tables acting as the proposal, thus regaining its freshness, all in $O(1)$. Moreover, this sample generation method can be outsourced to another processor, enabling parallelism.

The problem of scalable inference is further exacerbated in case of nonparametric models and aforementioned speed-up techniques cannot be used directly, mainly for two reasons. First, in standard expressions of posterior for Bayesian nonparametrics the slowly varying part is not decoupled from the rest. Second, the size of the discrete state space is no longer pre-determined and keeps changing during the inference procedure. Now, if the size of the state space, K, increases since the generation of the alias table, we cannot use the stale alias tables for MH steps because now the domain of proposal does not contain the domain of target distribution.

In order to circumvent this issue, first of all we derive an alternative parametrization of HDP, wherein the slowly varying terms are decoupled in the posterior. Next we propose a log structured alias sampling data structure for storing the variable size distribution and allowing almost a constant time generation and cheap updates. To extend the idea of A. Q. Li et al. (2014) for HDP, we take advantage of this data structure by handling a mismatch in the state-space size by updating the proposal cheaply on the fly. Also we describe a multithreaded implementation, leveraging the fact that modern computers have multiple processors. With this new strategy we speed up the inference procedure on HDP significantly, e.g. gaining a 10x speed-up on PubMed and Wikipedia datasets.

The key contribution is to study sampling algorithms in the context of training LVM. This includes both improvements to sampling algorithms (for example, using Walker's alias method to train HDP) and more efficient ways of computing Stirling numbers. Hence it is worth providing a comprehensive and self-contained overview of nonparametric probabilistic topic models. Readers intimately familiar with the subject matter will likely only want to consult the discussion below as notation reference.

9.2 BACKGROUND

9.2.1 *Nonparametric Models*

With large numbers of objects (documents, images, users, etc.) parametric models are less useful since they fail to capture the amount of detail inherent in the data. For instance, it is only marginally useful to strive for 10,000 or more topics. The result of such a strategy would be nothing more than an extreme over-segmentation of the data. While this might be useful as feature generator, it fails to capture the inherent structure of the data. Suggestions to the contrary by Yi Wang et al. (2014) are likely due to the fact that their discriminative model suffers from an unsophisticated parametrization, hence over-segmentation of the space helps.

Instead, nonparametric models allow for both structured and fine-grained information representation and extraction. However, this often renders sampling inference painfully slow due to the large degree of complexity. Even worse, the size and structure of the discrete state space is no longer predetermined and keeps changing during the inference procedure. In short, the cost of scaling latent variable models to large datasets is often superlinear in practice due to inefficient data structures, unless great care is taken.

Hierarchical Bayesian Nonparametrics is an extraordinarily fertile field, leading to a vast range of models such as the Hierarchical Dirichlet Process (Y. W. Teh et al., 2006), the Nested Chinese Restaurant Process (D. Blei, T. Griffiths, and M. Jordan, 2010), Pachinko Allocation (W. Li, D. Blei, and A. McCallum, 2007), the Mondrian Process (Roy and Y. W. Teh, 2009), the Indian Buffet Process (T. Griffiths and Z. Ghahramani, 2011) , or the Nested Chinese Restaurant Franchise (A. Ahmed, Hong, and A.J. Smola, 2013; Paisley et al., 2013), just to name a few of them. It is thus virtually impossible to cover all sampling modifications of the 'flat' model in this paper. Instead, we focus on the HDP, as it exhibits all relevant properties and poses associated challenges: structured and time-varying datastructures, dependence between states, and computation of nontrivial combinatorial coefficients.

The HDP uses a Dirichlet Process (DP) G_j for each document j to handle uncertainty in number of mixture components. At the same time, in order to share mixture components and clusters across groups, each of these DPs is drawn from a global DP G_0.

$$G_j \sim \mathrm{DP}(\alpha_0, G_0) \qquad G_0 \sim \mathrm{DP}(\alpha_0, H) \tag{9.1}$$

We proceed to draw for each word i in document j a topic distribution ψ_{ji} from G_j and a word from it. As before the conjugate pair (F, H) form the language model.

$$\psi_{ji} \sim G_j \qquad x_{ji} \sim F(\psi_{ji}) \tag{9.2}$$

Exploiting conjugacy, the HDP is best explained via the Chinese Restaurant Franchise metaphor. We have a franchise of restaunts with same menu across all of its restaurants. Every document amounts to its own restaurant belonging to this franchise, wherein the customers (tokens) enter sequentially.

Symbol	Description
J	Number of documents
K	Number of topics active
N_j	Number of tokens in document j
x_{ji}	Token i in document j
$\mathcal{X}$	Entire collection of documents, i.e. $\mathcal{X} = \{\{x_{ji}\}_{i=1}^{N_j}\}_{j=1}^{J}$
θ_j	Topic-proportions for document j
ϕ_k	Word distribution for topic k
$n(j,k)$	Number of customers/tokens assigned to dish/topic k in restaurant/-document j.
$c(x,k)$	Number of times word x assigned to dish/topic k
$m(j,k)$	Number of tables serving dish/topic k in restaurant/document j
M	Total number of tables globally, i.e. $M = \sum_{j,k} m(j,k)$
z_{ji}	Dish/topic assignment of customer/token i in restaurant/document j.
u_{ji}	Auxiliary variable to denote if customer/token i in restaurant/ document j was responsible for creation of a new table
α_0	Scaling parameter of document specific DPs
γ	Scaling parameter of root DP
S_m^n	Stirling numbers of the first kind
$(a)_b$	Falling factorial, i.e. $(a)_b = a(a+1)\cdots(a+b-1)$

Table 9.1: A quick reference for notations used in describing nonparametric models.

- Whenever a new customer walks into a particular restaurant, he will sit at an existing table with probability proportional to the number of people sitting at the table and share the dish (topic) being served on that table
- Alternatively the new customer will sit at a new table with probability proportional to α_0 and order a new dish (topic) k from G_0. In this case a phantom customer is sent to the restaurant responsible for G_0.
- There, with probability proportional to γ, a new table is generated with a new dish drawn from H. Otherwise, an existing dish is drawn with probability proportional to the number of phantom customers associated with the dish.

G_0 acts as a parent restaurant which controls the menu, i.e. the creation or deletion of dishes (topic). This allows documents (or groups) to share *statistical strength*. By recognizing that draws from a Dirichlet process are discrete almost surely we can express the HDP in a form very similar to LDA as shown in Figure 9.1. In particular $G_0 \sim \text{DP}(\gamma, H)$ can be represented as (Sethuraman, 1994):

$$G_0 = \sum_{k=1}^{\infty} \theta_{0k} \delta_{\phi_k} \qquad \text{where} \qquad \phi_k \sim H(\cdot) \qquad \text{and} \qquad \theta_0 = \{\theta_{0k}\}_{k=1}^{\infty} \sim \text{GEM}(\gamma) \tag{9.3}$$

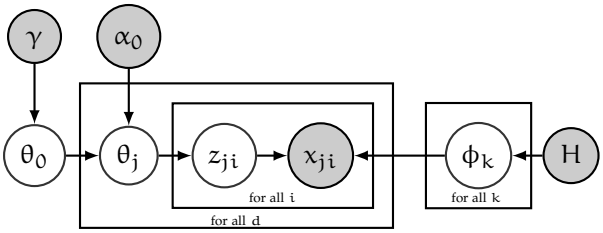


Figure 9.1: HDP Graphical Model.

Here GEM(γ) denotes the Griffiths, Engen and McCloskey stick breaking process (Pitman, 2006). Since G_0 has support only at $\{\phi_k\}_{k=1}^{\infty}$, each G_j necessarily has support there as well, and can thus be written as:

$$G_j = \sum_{k=1}^{\infty} \theta_{jk} \delta_{\phi_k} \tag{9.4}$$

It can be shown that each $\theta_j = \{\theta_{jk}\}_{k=1}^{\infty}$ is independently distributed according to DP(α_0, θ_0), i.e. $\theta_j \sim \text{DP}(\alpha_0, \theta_0)$ (Y. W. Teh et al., 2006). Here we interpret θ_0 and θ_j as probability measures on natural numbers. As in any mixture model, since each factor ψ_{ji} is distributed according to G_j, it takes on the value ϕ_k with probability θ_{jk}. To simplify, let z_{ji} be an indicator variable such that $\psi_{ji} = \phi_{z_{ji}}$. Given z_{ji}, we have $x_{ji} \sim F(\phi_{z_{ji}})$. Thus we obtain an equivalent representation of the HDP, resembling LDA via the following conditional distributions:

$$\begin{aligned} \theta_0 &\sim \text{GEM}(\gamma) \\ \theta_j &\sim \text{DP}(\alpha_0, \theta_0) & \phi_k &\sim H \\ z_{ji} &\sim \theta_j & x_{ji} &\sim F(\phi_{z_{ji}}) \end{aligned} \tag{9.5}$$

If we let $F(\phi)$ have density $f(\cdot|\phi)$ and H have density $h(\cdot)$, then under this view one can write the joint density of the model as:

$$p(\mathcal{X}, z, \theta, \phi | \alpha, H) = \prod_{k=1}^{\infty} h(\phi_k) \prod_{j=1}^{J} p(\theta_j|\alpha) \prod_{i=1}^{N_j} p(z_{ji}|\theta_j) f(x_{ji}|z_{ji}, \phi_{z_{di}}) \tag{9.6}$$

These representations will be helpful in deriving the inference using Gibbs sampling.

Inference in hierarchical models is decidedly more complex. A brief history of inference in LVMs is illustrated in Figure 9.2. We discuss two main strategies below. This is done for self-consistency and also to derive a computationally more efficient expansion for the associated combinatorial coefficients. We discuss two samplers: sampling by direct assignment (Y. W. Teh et al., 2006) and the more recently proposed sampling by compact representation (C. Chen, Du, and W.L. Buntine, 2011). The latter has the property of mixing considerably faster. In a nutshell, it decouples the issue of provenance, i.e. which level in the hierarchy gives rise to a specific topic from the issue of which instance takes credit for its introduction at that level (which is uniformly distributed over the instances satisfying the former condition). We begin with the more conventional model.

9.2.1.1 *Sampling by direct assignment*

Similar to the case of LDA to speed up convergence, we would want to collapse out as many variables as possible. However, due to the hierarchy collapsing out all of the variables leads to a complex dependence structure and booking-keeping. As a solution Y. W. Teh et al. (2006) proposed to collapse out the document specific DPs G_j, but retain the global DP G_0. The

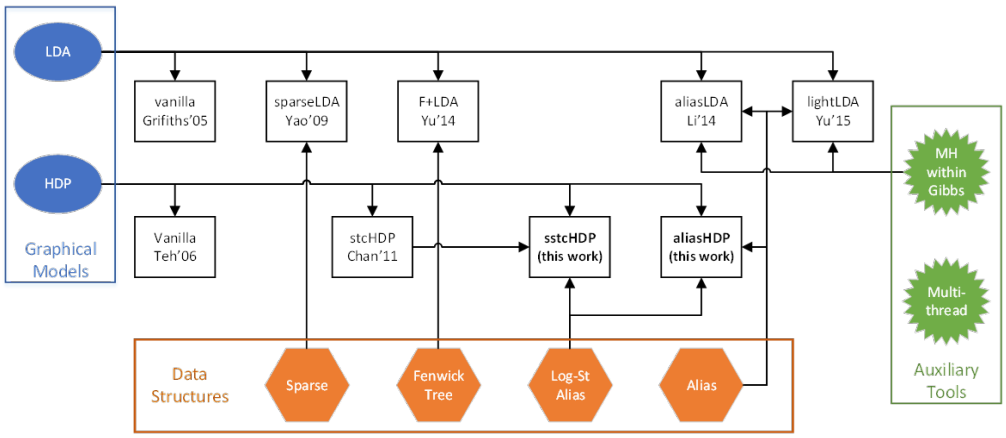


Figure 9.2: Overview of various inference algorithms for parametric LDA and non-parametric HDP model. Most models use certain data structures and auxiliary tools.

instantiation of G_0 (equivalently θ_0) makes the posterior conditioned on G_0 to factorize across documents, i.e. sampling for a given document will only depend on counts from the same document given G_0. This leads to a much simpler book-keeping as we will see.

Let us begin by deriving the posterior of z_{ji} given other topic assignments z_{-ji}, the global DP G_0 (equivalently θ_0), and document specific DP G_j (equivalently θ_j) integrated out. Using the exchangeability property of DPs (Antoniak, 1974) for integrating out θ_j in (9.5), one can find that posterior of z_{ji} depends only on local counts and is given by:

$$z_{ji}|z_{-ji},\alpha_0,\theta_0 \propto \sum_{k=1}^{K} \frac{n^{-ji}(j,k)}{N_j+\alpha_0}\delta_k + \frac{\alpha_0}{N_j+\alpha}\theta_0 \tag{9.7}$$

Similarly at root level, as mentioned earlier, phantom customers from document level restaurants are sent each time a new table is created. Let $m(j,k)$ denote the number of tables in restaurant j, serving dish k. This means $\bar{m}(k) = \sum_j m(j,k)$ customers are sitting in a table eating dish k at the root level and in total there are $M = \sum_k \bar{m}(k)$ customers at the root level. Then posterior of G_0, given table counts in each of the restaurants, is another Dirichlet process:

$$G_0|m,\gamma,H \sim DP\left(M+\gamma, \sum_{k=1}^{K}\frac{\bar{m}(k)}{M+\gamma}\delta_{\phi_k} + \frac{\gamma}{M+\gamma}H\right) \tag{9.8}$$

Now, to construct a finite instantiation of the posterior G_0, we use the definition of DP that any measurable partitions of parameter space under the DP is Dirichlet distributed. In particular, if we consider partition of space into $K+1$ components where first K corresponds to topics seen till now and the last component is rest of the space, we have:

$$\theta_0 = (\underbrace{\theta_{01},\theta_{02},...,\theta_{0K}}_{\text{Representing existing topics}}, \underbrace{\theta_{0u}}_{\text{Representing unseen topics}}) \sim \text{Dirichlet}(\underbrace{\bar{m}(1),...,\bar{m}(K)}_{\text{Measure of existing topic partitions}}, \underbrace{\gamma}_{\text{Measure for rest of the space}}), \tag{9.9}$$

Next, to incorporate the emission model, note that:

$$\Pr(z_{ji}=k|z_{-ji},\mathcal{X},\{\phi\},\alpha_0,\theta_0) \propto \Pr(z_{ji}=k|z_{-ji}\alpha_0,\theta_0)f(x_{ji}|\phi_k) \tag{9.10}$$

Finally, plugging (9.7) into (9.10), we obtain

$$\Pr(z_{ji} = k|\text{rest}) \propto \begin{cases} (n^{-ji}(j,k) + \alpha_0\theta_{0k})f(x_{ji}|\phi_k) & \text{if } k \text{ in use} \\ \alpha_0\theta_{0u}f(x_{ji}|\phi_{k^{new}}) & \text{if } k \text{ is new} \end{cases} \tag{9.11}$$

We can further collapse out ϕ_k, as in the case of LDA.

In the above discussion, we skipped how we obtained table counts, $\bar{m}(k)$ in the first place. As we do not keep track of table creations at document levels (but only keep track which customer is consuming which dish for efficiency), to obtain $\bar{m}(k)$ we have to sample $m(j,k)$ for all j. Note that $m(j,k)$ is just the number of tables for the DP G_j, for which the desired conditional distribution of has been shown by Antoniak (1974) to be:

$$\Pr(m(j,k) = m) \propto S_m^{n(j,k)}(\alpha_0\theta_{0k})^m. \tag{9.12}$$

Here the terms S_m^n are Stirling numbers of the first kind.

In summary, this method requires three kinds of sampling, which are very similar to LDA:

- **Sampling z:**

$$\Pr(z_{ji} = k|\text{rest}) \propto \begin{cases} (n^{-ji}(j,k) + \alpha_0\theta_{0k})f_k^{-ji}(x_{ji}) & \text{if } k \text{ in use} \\ \alpha_0\theta_{0u}f_{k^{new}}^{-ji}(x_{ji}) & \text{if } k \text{ is new} \end{cases} \tag{9.13}$$

- **Sampling m:**

$$\Pr(m(j,k) = m) \propto S_m^{n(j,k)}(\alpha_0\theta_{0k})^m. \tag{9.14}$$

- **Sampling θ_0:**

$$\theta_0 \sim \text{Dirichlet}(\bar{m}(1), \bar{m}(2), ..., \bar{m}(K), \gamma). \tag{9.15}$$

9.2.1.2 *Sampling by table configuration*

The key difference of C. Chen, Du, and W.L. Buntine (2011) relative to Y. W. Teh et al. (2006) is that rather than keeping track of relative assignments of tables to each other it simply keeps track of the level (global topic restaurant, current document restaurant, or no new table), denoted by u_{ji}, within the hierarchy of restaurants at which an individual customer opens a new table, if any. The advantage is that this allows us to factor out the relative assignment of customers to specific tables but rather only keep track of the dishes that they consume, very similar to LDA.

Using the table configuration model, sampling in the HDP occurs through the following process: in addition to the topic selector variable z_{ji}, we use the auxiliary variable u_{ji} to determine whether the dish existed before ($u_{ji} = 2$), whether a new table was allocated ($u_{ji} = 1$), or whether a new dish and a new table are allocated ($u_{ji} = 0$). The reason for this decomposition is that for any topic we only need one token $u_{ji} = 0$ to instantiate the topic and only one per restaurant to instantiate a table $u_{ji} = 1$. Thus, the counts, $m(j,k)$ for table serving dish k in restaurant j can be expressed in terms of u_{ji}'s as $m(j,k) = \sum_i \mathbb{I}\{z_{ji} = k, u_{ji} < 2\}$. Note that C. Chen, Du, and W.L. Buntine (2011) discuss this in the context of arbitrary hierarchies, where u_{ji} could be as large as the depth of the hierarchical model. The benefit of this auxiliary

variable decomposition is that we need not keep track of u explicitly but rather just resample one variable at a time as needed, as pointed out by C. Chen, Du, and W.L. Buntine (2011),

The joint posterior of z and u for the HDP is

$$\Pr(\{z,u\}) = \frac{\gamma^K}{(\gamma)_K}\left[\prod_{k=1}^{K}\frac{(\bar{m}(k)-1)!}{\bar{m}(k)}\right]\left[\prod_{j=1}^{J}\frac{\alpha_0^{\bar{m}(j)}}{(\alpha_0)_{N_j}}\prod_{k=1}^{K}S^{n(j,k)}_{m(j,k)}\frac{m(j,k)!(n(j,k)-m(j,k))!}{n(j,k)!}\right] \tag{9.16}$$

where $(x)_n$ is the falling factorial $x(x+1)\cdots(x+n-1)$. We can introduce the emission model as before in (9.10) as well as marginalize out ϕ as before. To obtain conditional posteriors for a particular z_{ji}, u_{ji}, we just divide $\Pr(\{z,u\})/\Pr(\{z,u\}^{-ji})$ and simplify the resulting expression to obtain $\Pr(z_{ji}, u_{ji}|\{z,u\}^{-ji})$.

In summary, we have the following (unnormalized) probabilities for document j for a word at position i:

No change — $u_{ki} = 2$: Whenever we do not change the number of tables in a restaurant, we have

$$\Pr(z_{ji}=k, u_{ji}=2|\text{rest}) \propto \frac{S^{n(j,k)+1}_{m(j,k)}}{S^{n(j,k)}_{m(j,k)}}\frac{n(j,k)+1-m(j,k)}{n(j,k)+1}f_k^{-ji}(x_{ji}) \tag{9.17}$$

New table — $u_{ji} = 1$: Whenever the customer starts a new table by picking one of the existing dishes in the global menu we have

$$\Pr(z_{ji}=k, u_{ji}=1|\text{rest}) \propto \frac{\alpha_0\bar{m}^2(k)}{(\bar{m}(k)+1)(M+\gamma)}\frac{S^{n(j,k)+1}_{m(j,k)+1}}{S^{n(j,k)}_{m(j,k)}}\frac{m(j,k)+1}{n(j,k)+1}f_k^{-ji}(x_{ji}) \tag{9.18}$$

New table, new global dish — $u_{ji} = 0$: In this case we get

$$\Pr(z_{ji}=k^{\text{new}}, u_{ji}=0|\text{rest}) \propto \frac{\alpha_0\gamma}{M+\gamma}f_{k^{\text{new}}}^{-ji}(x_{ji}) \tag{9.19}$$

Note that while the probabilities in (9.17)-(9.19) are unnormalized, they share the same normalization constant. Hence joint normalization is immediately available. Note that the original paper and its errata had some bugs, hence we re-derived the expressions.

9.2.2 *Composing MCMC Algorithms*

Over the past decade most of the fast samplers proposed for parametric LVMs like topic models have been based on different Metropolis-Hastings (MH) schemes. To understand their relative merits we need to review some background on the mixing and composition properties of Markov Chain Monte Carlo (MCMC) algorithms.

We denote by $\mathcal{K}(x,y)$ the *transition kernel* between subsequent states x and y of a Markov Chain. In MH sampling we have the following:

$$\mathcal{K}(x,y) = \mathcal{A}(x,y)q(y|x) + (1-r(x))\delta(y-x) \text{ where } r(x) = \int \mathcal{A}(x,y)q(y|x)dy. \tag{9.20}$$

Under mild conditions, it is possible to show that such chains are ergodic with equilibrium being the target distribution (Tierney, 1994). However, MCMC is efficient only when the chain converges *quickly*.

In MH this is governed by an effective choice of p. A common misconception (A. Q. Li et al., 2014; Yuan et al., 2015) is that getting higher acceptance probability in MH implies faster convergence. As counterexample consider using $\mathcal{N}(5,1)$ as proposal to sample from $\frac{1}{2}[\mathcal{N}(5,1)+\mathcal{N}(-5,1)]$ as target distribution. If we begin near 5 the acceptance ratio would almost always be close to 1, yet the chain would be extremely slow in exploring the true distribution. Below we list a number of proposals whose mixing behaviour has been extensively studied.

9.2.2.1 *Random walks*

In general the selection of the next state depends on the current state. Common choices include choosing a neighbor uniformly at random in the case of graphs or a normal distribution centered around current position in the case of continuous space. Ergodicity (irreducible and aperiodic) can be proved *e.g.* when the proposal is non-zero on the entire state space. Likewise, if the state space is open and connected, then proposal being non-zero just on a neighbourhood of current position suffices. Rosenthal (1995) shows that the mixing rate of the Markov chain is determined by the second largest eigenvalue of $\mathcal{K}$. The smaller it is the faster the chain converges.

9.2.2.2 *Independence chain*

A degenerate case (that we will use) is when the selection of the next state is independent of the current state, i.e. $q(j|i) = q_j$ for all i, j and that moreover the support of q contains that of p. Convergence of this proposal was analyzed by Mengersen, Tweedie, et al. (1996).

Theorem 9.1 *The Markov chain under the independence proposal is uniformly ergodic if there exists* $C \geqslant 1$ *such that*

$$p_k \leqslant C q_k \text{ for all } k \tag{9.21}$$

In this case the total variation distance between p *and probability distribution of the chain at time* t*, i.e.* P^t *is bounded by*

$$\left\| P^t - p \right\|_1 \leqslant (1 - C^{-1})^t. \tag{9.22}$$

Clearly, the closer C is to 1, the faster the rate of convergence. Hence, $\max_k p_k/q_k$ can be used to indicate the goodness of the proposal. Note that it is same as for rejection sampling, albeit without the requirement to know the C explicitly.

9.2.2.3 *Hybrid methods*

A great benefit in designing MCMC proposal distributions is that we can combine them into a hybrid proposal. Suppose $\mathcal{K}_1, ..., \mathcal{K}_r$ are transition kernels with the same invariant distribution p, then we can combine them *e.g.* sequentially or in parallel.

PARALLEL COMPOSITION. We define the mixture kernel via

$$\mathcal{K} = \sum_{i=1}^{r} \alpha_i \mathcal{K}_i \text{ where } \alpha_i \geqslant 0 \text{ and } \|\alpha\|_1 = 1. \tag{9.23}$$

Here α defines a probability distribution over $\mathcal{K}_i$. At each step one of the kernels is selected according to α.

SEQUENTIAL COMPOSITION. We may carry out the kernels in sequence, *i.e.*

$$\mathcal{K} = \mathcal{K}_1 \odot \mathcal{K}_2 \odot \cdots \odot \mathcal{K}_r. \tag{9.24}$$

Each kernel is used in turn, and when the last one is the cycle is restarted. This is desirable, e.g. whenever the eigenfunctions of the $\mathcal{K}_i$ differ considerably. In this poor convergence with regard to some mode in one kernel can be mitigated by improved convergence in another kernel and vice versa.

While individual kernels might not be ergodic, their composition can be. Unfortunately, exact convergence analysis is only known for very limited cases (Tierney, 1994).

9.2.2.4 *Gibbs sampling and variants*

Perhaps the most famous example of the composition principle is the Gibbs sampler. In it, each kernel $\mathcal{K}_i$ samples only one component (or a subset of variables) at a time. That is,

$$\mathcal{K}_i(x^t, x^{t+1}) = \begin{cases} 0 & \text{if } x_j^{t+1} \neq x_j^t \text{ for any } j \neq i, \\ p(x_i^{t+1} | x_{\neg i}^t) & \text{otherwise} \end{cases} \tag{9.25}$$

Clearly $\mathcal{K}_i$ has p as its stationary distribution. However, it is not irreducible. Neither of the kernels $\mathcal{K}_j$ on their own possess ergodicity, but after parallel or sequential composition the chain becomes ergodic. See Gonzalez et al. (2011) for a discussion and analysis.

Whenever sampling from the conditional distribution $p(x_i|x_{\neg i})$ is too costly, we may replace $\mathcal{K}_i$ by an MH-step, *i.e.* by a sub-kernel that has $p(x_i|x_{\neg i})$ as its invariant distribution. This preserves mathematical validity, *i.e.* the chain remains ergodic and the stationary distribution is still the true posterior.

In the following we will combine the above sampling techniques for efficient inference in Bayesian latent variable methods. This will be accomplished by *smart data structures*, *Metropolis-within-Gibbs*, and by using the composition principle, such as to speed up topic model inference on large models and data considerably.

9.3 SPEEDING UP INFERENCE IN NONPARAMETRIC MODELS

In this section we show how techniques developed for speeding up parametrics models, like LDA, can be extended to non-parametric models. However, scalable inference is rather more complex in this case than in LDA, reasons for which are two fold. First, the model does not decompose into local sparse terms and global, slowly varying terms, since local choices may affect the composition in terms of topics all the way throughout the HDP. All the fast samplers for LDA are based on such decompositions, thus invalidating their direct application. Second, the size of the discrete state space is no longer pre-determined and keeps changing during the inference procedure. This makes it difficult to design fast samplers that precompute parts of the distribution.

9.3.1 *Exploiting Sparsity*

In order to speed-up the sampler by leveraging sparsity, one needs to identify a partition of posterior having sparse terms, as in case of fast LDA samplers like SparseLDA (L. Yao, Mimno, and Andrew McCallum, 2009) and AliasLDA (A. Q. Li et al., 2014). In case of inference on HDP using Sampling by Direct Assignment (Section 9.2.1.1), due to resemblance of posterior to LDA, one can directly use same partition of the posterior into a sparse local document-only term and a slowly varying global word-topic count term as in LDA. Thus same techniques as

presented for AliasLDA (A. Q. Li et al., 2014) directly applies. Whereas in the other case of inference for HDP utilizing the faster mixing Sampling by Table Configuration (Section 9.2.1.2) no straightforward decoupling can be seen. The probability of sitting at an existing table in the restaurant, (9.17), is sparse and depends only on counts within restaurants. However, the probability of starting a new table, (9.18), depends both on local and global counts and must be evaluated for all active topics, which would not be sparse.

We now develop a reformulation of the posterior under Sampling by Table Configuration by separating out the sparse terms. We begin by observing that for topics which have no representation in the current restaurant, i.e. when $n(j,k) = 0$ and $m(j,k) = 0$, the probability in (9.18) just becomes:

$$\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \frac{\alpha_0 \bar{m}^2(k)}{(\bar{m}(k)+1)(M+\gamma)} f_k(x_{ji}) := \pi \tag{9.26}$$

Now we show that for topics which are already represented in the current restaurant, i.e. $n(j,k) > 0, m(j,k) > 0$ we can collect π separately as:

$$\begin{aligned}\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) &\propto \pi \frac{S^{n(j,k)+1}_{m(j,k)+1}}{S^{n(j,k)}_{m(j,k)}} \frac{m(j,k)+1}{n(j,k)+1} \\ &= \pi \left(\frac{S^{n(j,k)+1}_{m(j,k)+1}}{S^{n(j,k)}_{m(j,k)}} \frac{m(j,k)+1}{n(j,k)+1} - 1 \right) + \pi\end{aligned} \tag{9.27}$$

If term in the bracket is non-zero, then it can become a local sparse distribution on its own and the leftover π can be combined with distribution of (9.26). But for this split to yield valid distributions, we need to prove that the term in the bracket is non-negative. To prove this, we will first need a couple of lemma.

Lemma 9.2 *For all $m \leqslant n \in \mathbb{N}$, we have*

$$\frac{m+1}{n+1}\left(\frac{2n}{m} - 1\right) \geqslant 1 \tag{9.28}$$

Proof It is equivalent to show that:

$$\frac{m+1}{n+1}\left(\frac{2n}{m} - 1\right) - 1 \geqslant 0 \tag{9.29}$$

Simplifying the LHS we obtain:

$$\begin{aligned}\text{LHS} &= \frac{m+1}{n+1}\left(\frac{2n}{m} - 1\right) - 1 \\ &= \frac{(m+1)(2n-m) - m(n+1)}{n+1} \\ &= \frac{-(m+2)(m-n)}{(n+1)}\end{aligned} \tag{9.30}$$

As a concave quadratic is always non-negative between its two roots, the LHS is always non-negative for $-2 \leqslant m \leqslant n$ for all n. This completes our proof. ■

Lemma 9.3 *For all $n \in \mathbb{N}$, the following sequence is non-increasing in m:*

$$a_{n,m} = \frac{m}{n-m}\frac{S^n_{m+1}}{S^n_m}, \qquad m = 1, 2, .., n-1 \tag{9.31}$$

In other words $a_{n,1} \geqslant a_{n,2} \geqslant \cdots \geqslant a_{n,n-1}$.

Proof We prove the non-increasing sequence of $a_{n,m}$ in m by induction. The induction hypothesis is equivalent to

$$(m-1)(n-m)\left(S^n_m\right)^2 - m(n-m+1)S^n_{m+1}S^n_{m-1} \geqslant 0 \tag{9.32}$$

The base case for induction using $n = 2, m = 1$ holds trivially as $S^2_0 = 0$. Assume the statement holds till n, then for $n+1$ we need to prove:

$$(m-1)(n-m+1)\left(S^{n+1}_m\right)^2 - m(n-m+2)S^{n+1}_{m+1}S^{n+1}_{m-1} \tag{9.33}$$

We begin by simplifying the LHS, which would reveal that each term in the LHS is indeed positive or non-negative by the induction hypothesis.

$$\begin{aligned}
\text{LHS} &= (m-1)(n-m+1)\left(S^{n+1}_m\right)^2 - m(n-m+2)S^{n+1}_{m+1}S^{n+1}_{m-1} \\
&= (m-1)(n-m+1)\left(S^n_{m-1} + nS^n_m\right)^2 \\
&\quad - m(n-m+2)\left(S^n_m + nS^n_{m+1}\right)\left(S^n_{m-2} + nS^n_{m-1}\right) \\
&= \underbrace{(m-1)(n-m+1)\left(S^n_{m-1}\right)^2 - m(n-m+2)S^n_m S^n_{m-2}}_{=a} \\
&\quad + \underbrace{n^2(m-1)(n-m+1)\left(S^n_m\right)^2 - n^2 m(n-m+2)S^n_{m+1}S^n_{m-1}}_{=b} \\
&\quad + \underbrace{[n(m-2)(n-m) - 2n]\,S^n_m S^n_{m-1} - nm(n-m+2)S^n_{m+1}S^n_{m-2}}_{=c} \\
&= a + b + c
\end{aligned} \tag{9.34}$$

Upon simplifying a further, we obtain:

$$\begin{aligned}
a &= (m-1)(n-m+1)\left(S^n_{m-1}\right)^2 - m(n-m+2)S^n_m S^n_{m-2} \\
&= \frac{m}{m-1}\left[(m-2)(n-m+1)\left(S^n_{m-1}\right)^2 - (m-1)(n-m+2)S^n_m S^n_{m-2}\right] \\
&\quad + \frac{n-m+1}{m-1}\left(S^n_{m-1}\right)^2 \\
&\geqslant \frac{n-m+1}{m-1}\left(S^n_{m-1}\right)^2
\end{aligned} \tag{9.35}$$

where the last inequality follows by applying induction hypothesis. Similarly, proceeding for b one obtains:

$$\begin{aligned}
b &= n^2(m-1)(n-m+1)\left(S^n_m\right)^2 - n^2 m(n-m+2)S^n_{m+1}S^n_{m-1} \\
&= n^2\frac{n-m+2}{n-m+1}\left[(m-1)(n-m)\left(S^n_m\right)^2 - m(n-m+1)S^n_{m+1}S^n_{m-1}\right] \\
&\quad + n^2\frac{m-1}{n-m+1}\left(S^n_m\right)^2 \\
&\geqslant n^2\frac{m-1}{n-m+1}\left(S^n_m\right)^2
\end{aligned} \tag{9.36}$$

again where the last inequality follows by applying induction hypothesis. Finally, we simplify c,

$$\begin{aligned} c &= [n(m-2)(n-m) - 2n]\, S^n_m S^n_{m-1} - nm(n-m+2) S^n_{m+1} S^n_{m-2} \\ &\geqslant -2n S^n_m S^n_{m-1} \end{aligned} \tag{9.37}$$

To see the last inequality, note again from induction hypothesis that $a_{n,m-2} \geqslant a_{n,m}$, i.e.

$$(m-2)(n-m) S^n_m S^n_{m-1} - m(n-m+2) S^n_{m+1} S^n_{m-2} \geqslant 0 \tag{9.38}$$

Substituting the simplified expressions for a, b, c back into (9.34):

$$\begin{aligned} \text{LHS} &= a + b + c \\ &\geqslant \frac{n-m+1}{m-1} \left(S^n_{m-1}\right)^2 + \frac{m-1}{n-m+1} \left(S^n_m\right)^2 - 2n S^n_m S^n_{m-1} \\ &= \left(\sqrt{\frac{n-m+1}{m-1}} S^n_{m-1} - n \sqrt{\frac{m-1}{n-m+1}} S^n_m \right)^2 \\ &\geqslant 0 \end{aligned} \tag{9.39}$$

This completes the proof. ■

Theorem 9.4 *The Stirling number of first kind satisfies the following inequality for all* $0 \leqslant m \leqslant n$*:*

$$\frac{m+1}{n+1} \frac{S^{n+1}_{m+1}}{S^n_m} \geqslant 1 \tag{9.40}$$

Proof From Lemma 9.3, we have that $a_{n,m}$ is an non-increasing sequence in m for all n. Thus every element $a_{n,m}$ in the sequence is lower bounded by the last element $a_{n,n-1} = 2/n$, i.e. $\forall n,\, 1 \leqslant m \leqslant n-1 : a_m \geqslant a_{n-1}$. As a result

$$\frac{S^n_{m+1}}{S^n_m} \geqslant \frac{2(n-m)}{nm} \tag{9.41}$$

Finally to show the original claim (9.40), we use Lemma 9.2 and the recurrence relation on S^{n+1}_{m+1} to obtain the desired result:

$$\frac{m+1}{n+1} \frac{S^{n+1}_{m+1}}{S^n_m} \geqslant \frac{m+1}{n+1} \left(1 + \frac{n S^n_{m+1}}{S^n_m}\right) \geqslant \frac{m+1}{n+1} \left(\frac{2n}{m} - 1\right) \geqslant 1. \tag{9.42}$$

■

This result allows us to partition the posterior for sampling table configurations into sparse local terms and slowly varying global terms, as follows:

No change — $u_{ki} = 2$: As before, whenever we do not change the number of tables in a restaurant, we have

$$\Pr(z_{ji} = k, u_{ji} = 2 | \text{rest}) \propto \frac{S^{n(j,k)+1}_{m(j,k)}}{S^{n(j,k)}_{m(j,k)}} \frac{n(j,k) + 1 - m(j,k)}{n(j,k)+1} f_k(x_{ji}) \tag{9.43}$$

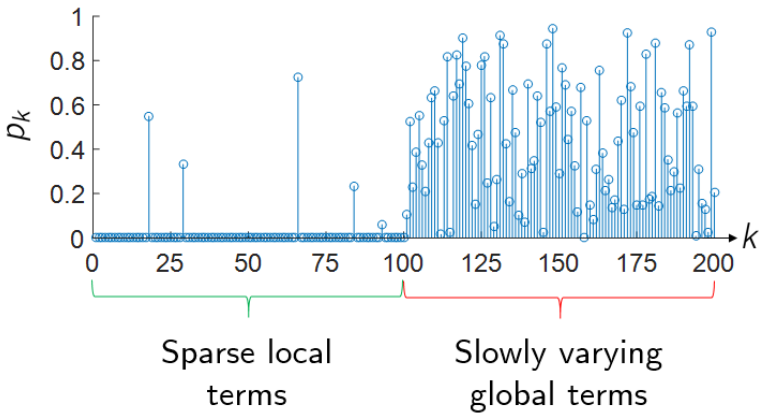


Figure 9.3: Typical posterior distribution of the HDP using a sampling table configuration representation. Without loss of generality we order local and global terms into a joint list. What is displayed are unnormalized probabilities.

New table among existing local dishes — $u_{ji} = 1$: Whenever the customer starts a new table by picking one of the existing dishes in the restaurant we have

$$\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \frac{\alpha_0 \bar{m}^2(k)}{(\bar{m}(k)+1)(M+\gamma)} \left(\frac{S^{n(j,k)+1}_{m(j,k)+1}}{S^{n(j,k)}_{m(j,k)}} \frac{m(j,k)+1}{n(j,k)+1} - 1 \right) f_k(x_{ji}) \tag{9.44}$$

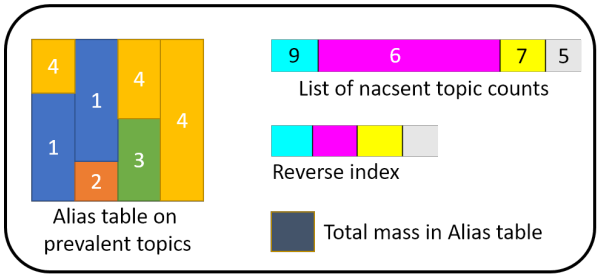
New table — $u_{ji} = 1$: Whenever the customer starts a new table picking one of the existing dishes from the global menu (existing dish among the restaurant or new local dishes from the global menu) we have

$$\Pr(z_{ji} = k, u_{ji} = 1|\text{rest}) \propto \frac{\alpha_0 \bar{m}^2(k)}{(\bar{m}(k)+1)(M+\gamma)} f_k(x_{ji}) \tag{9.45}$$

New table, new global dish — $u_{ji} = 0$: In this case we get

$$\Pr(z_{ji} = k^{\text{new}}, u_{ji} = 0|\text{rest}) \propto \frac{\alpha_0 \gamma}{M+\gamma} f_{k^{\text{new}}}(x_{ji}) \tag{9.46}$$

Careful inspection of the posterior illustrates that (9.43) and (9.44) are the sparse local document terms (after all, no new dish is added to the local document), whereas (9.45) is the slowly varying global term. Thus, in this formulation of HDP the sparse and slowly varying terms are decoupled as desired. Figure 9.3 shows an instance of unnormalized posterior after the decoupling which exhibits the division of sparse local quantities and dense global ones explicitly. This allows us to exploit sparsity in a manner similar to LDA.



1. Traverse the list to sum of weights of nacsent topics s_n
2. Use the cached sum of weights in alias table s_a to find total mass s_T
3. Draw $u \sim U[0, s_T]$
4. If $u < s_a$: Sample from the alias table
5. Else: Sample from the weights in the list

Figure 9.4: Log structured alias table

9.3.2 *Log Structured Alias Sampling*

The resemblance of the sampling by direct assignment (Section 9.2.1.1) tempts one to immediately borrow the alias sampling trick. One might consider to partition the posterior of z_{ji} as before into a sparse component and a slowly varying one.

$$\Pr(z_{ji} = k|\text{rest}) \propto \begin{cases} \underbrace{n_{kj}^{-ji} f_k^{-ji}(x_{ji})}_{\text{sparse local terms}} + \underbrace{\alpha_0 \theta_{0k} f_k^{-ji}(x_{ji})}_{\text{slowly varying global terms}} & \text{if } k \text{ in use} \\ \underbrace{\alpha_0 \theta_{0u} f_{k^{new}}^{-ji}(x_{ji})}_{\text{singleton}} & \text{if } k \text{ is new} \end{cases} \tag{9.47}$$

Unlike the case of LDA, now the size of the discrete state space is no longer pre-determined and keeps changing during the inference procedure. This makes it difficult to precompute parts of the distribution and store it efficiently in some data structure. This is because it is costly recompute or update the data structure every time a new topic is created/deleted for all the words.

To circumvent this issue, we propose a Log Structured Alias Sampler. It is inspired by log-structured merge data-structures such as BigTable (F. Chang et al., 2006). We process and store the counts for prevalent topics in alias tables and the counts for nascent topics are lazily stored in a list as a log as shown in Figure 9.4. Then once in a while we merge growing topics from the nascent list into the main alias table. To sample from this data structure costs $O(K_{\text{nascent}})$, as we can efficient draw from prevalent topics in $O(1)$ time, but we traverse the list of nascent topics. Typically, the number of nascent topics and the probability mass associated with it is very small. As a result it does not cause a huge performance penalty as compared to direct sampling from alias table.

Also we incorporate a reverse index for the log nascent topics to enable $O(1)$ lookup of their counts which is required in accept/reject step of Metropolis-Hastings.

9.3.3 *Dealing with Stirling Numbers*

In case of Sampling by Direct Assignment, we face Stirling's number in posterior of the root DP in form of Antoniak's distribution. Using the following trick we can sample all of m from Antoniak's distribution with a cost equal to that of flipping a coin per token in the corpus.

- Draw auxiliary variables $s_{jk}^{l} \sim \text{Bernoulli}\left(\frac{\alpha_0 \theta_{0k}}{\alpha_0 \theta_{0k} + l - 1}\right)$ for $l \in \{1, \ldots n_{kj}\}$.

- A sample m from the Antoniak distribution can be computed by $m = \sum_l s_{jk}^l$.

Lemma 9.5 *Denote by $\mathcal{A}(n, \theta)$ the Antoniak distribution. That is, it denotes the distribution where $p(x = m) \propto S_m^n \theta^m$ for $m \in \{1, \dots m\}$ and where S_m^n is the Stirling number of the first kind. Moreover, assume that we have m Bernoulli random variables s_l, drawn from $s_l \sim \text{Bin}\left(\frac{\theta}{\theta+l-1}\right)$ for $l \in \{1, \dots m\}$. Then $s := \sum_{l=1}^n s_l$ satisfies $s \sim \mathcal{A}(n, \theta)$. Hence sampling from $\mathcal{A}(n, \theta)$ can be accomplished in $O(n)$ time and $O(1)$ space.*

Proof We begin with the defining property of the Stirling numbers, namely that they are the coefficients of a polynomial of degree n given by $\Gamma(\theta + n)/\Gamma(\theta)$, i.e.

$$\theta^{\bar{n}} := \frac{\Gamma(\theta + n)}{\Gamma(\theta)} = \theta(\theta + 1) \cdots (\theta + n - 1) =: \sum_{m=1}^{n} S_m^n \theta^m \tag{9.48}$$

By construction $S_0^n = 0$, hence it is omitted from the expansion. We obtain

$$\begin{aligned} \Pr\left[\sum_{l=1}^{n} s_l = m\right] &= \sum_{s_1, \dots s_n | s = m} \Pr[s_1, \dots x_n] && (9.49) \\ &= \frac{\theta^m}{\prod_{l=1}^n (\theta + l - 1)} \sum_{\{s_1, \dots s_m\} | s = m} \prod_{s_i = 0} (i - 1) && (9.50) \\ &= \frac{\theta^m}{\prod_{l=1}^n (\theta + l - 1)} S_m^n = \frac{\theta^m S_m^n}{\sum_{l=1}^n \theta^l S_l^n} && (9.51) \end{aligned}$$

Here the equality (9.50) follows from the fact that we need to multiply over all terms $1 - \frac{\theta}{\theta+l-1} = \frac{l-1}{\theta+l-1}$ for which $s_l = 0$. Moreover, (9.51) follows from the definition of the Stirling number of the first kind by decomposing $(\theta + i - 1) = \theta + (i - 1)$ and the last equality follows from the property of $\theta^{\bar{n}}$. ■

In case of Sampling by Table Configurations, many terms of the conditional posterior involves Stirling numbers. However, Stirling numbers rapidly become very large and even computation of Stirling numbers S_m^n for moderately large n and m, which will frequently occur for large text corpus, can cause numerical instability. Hence one would typically resort to store and compute them in log domain. The recursion for Stirling number yields:

$$\log S_{m+1}^{n+1} = \log S_{m+1}^n + \log\left(\exp\left(\log S_m^n - \log S_{m+1}^n\right) + n\right) \tag{9.52}$$

But such implementation can be slow as the $\log()$ and $\exp()$ functions can make the evaluation slow if implemented naively.

Fortunately, for sampling of hierarchical models we only need the Stirling numbers in the following forms:

$$U_t^n = \frac{S_m^{n+1}}{S_m^n} \qquad V_t^n = \frac{S_{m+1}^{n+1}}{S_{m+1}^{n+1}} \tag{9.53}$$

since only the ratios are required, we can directly compute them. W. Buntine (2002) showed that just by manipulating the recursion obeyed by the Stirling numbers, we can obtain the following recursion directly for the ratio:

$$U_m^n = U_{m-1}^{n-1} + n - (n - 1)\frac{U_{m-1}^{n-1}}{U_m^{n-1}} \tag{9.54}$$

with $U_0^0 = 1, U_0^1 = \infty, U_1^1 = 1$. Also in backwards direction as:

$$U_m^n = U_{m+1}^{n+1} - (n+1) + n\frac{U_{m+1}^{n+1} - (n+1)}{U_{m+1}^n - n} \tag{9.55}$$

At every sampling instance of HDP, n or m can change by at most ± 1. So we can cache a few rows of Stirling numbers around the presently active values. If during sampling we move towards the boundary we can re-compute some more rows and discard values at the other end. Furthermore, numerically using the recursion (9.54) has been shown to be much more stable, even when using float instead of double (W. Buntine, 2002). We do not need to cache the other ratio V_m^n separately, as it can be readily expressed in terms of U_m^n as follow:

$$V_m^n = \frac{U_{m+1}^n}{U_{m+1}^n - n} \tag{9.56}$$

To summarize, in this section we explored ways to handle Stirling numbers which are hard compute in numerically stable way. In case of sampling by direct assignment, we found an alternate way to sample from Antoniak's distribution which totally avoids use of Stirling's number. The new sampling strategy is much cheaper, costs $O(n)$ per token and needs no caching memory. In case of sampling table configuration, we only need ratios of certain Stirling number and we found a numerically more stable and faster recursion to directly compute the ratios.

When all of speed-up approaches, i.e. leveraging sparsity, log structured alias sampling, and tackling Stirling numbers, are combined a significant gain in speed and scalability is achieved for inference on HDP as will be seen from experiments presented in next section. Furthermore, parallelization scheme same as that for flat models (Section 9.3.4) can be applied to HDP as well.

9.3.4 *Parallelization*

All of the method we discussed can be easily parallelized, if we tolerate a small approximation. Modern computational resources are heavily multi-threaded and an efficient inference procedure should utilize this structure. All the inference strategies discussed keeps track and operates of two kinds of data:

- **Global statistics** which represents the word given topic counts, and
- **Local statistics** consisting of the latent topic indicators and topic given document counts.

There is a common framework behind each strategy, i.e. reading the current statistics, computing the conditional distribution and sampling from it and finally updating the statistics.

Reading and writing to the local statistics present no obstacles to parallelizing and can be handled independently by individual threads. Now the key idea for parallelizing the sampler is to observe that the global topic distribution and the topic given word counts (which we will refer to as state of the system) change only little given the changes in a single document (we may have millions of documents). Hence, we can assume that $\bar{c}(k)$ and $c(x,k)$ are essentially constant while sampling topics for a document. This means that instantaneously updating $c(k)$ and $c(x,k)$ or not during the sampling process would hardly have any manifestations and we can defer this action to a separate thread without much repercussions. This setup allows us to execute a large number of sampling threads simultaneously, speeding up the inference considerably as we will in the see in the experiments (Section 9.4).

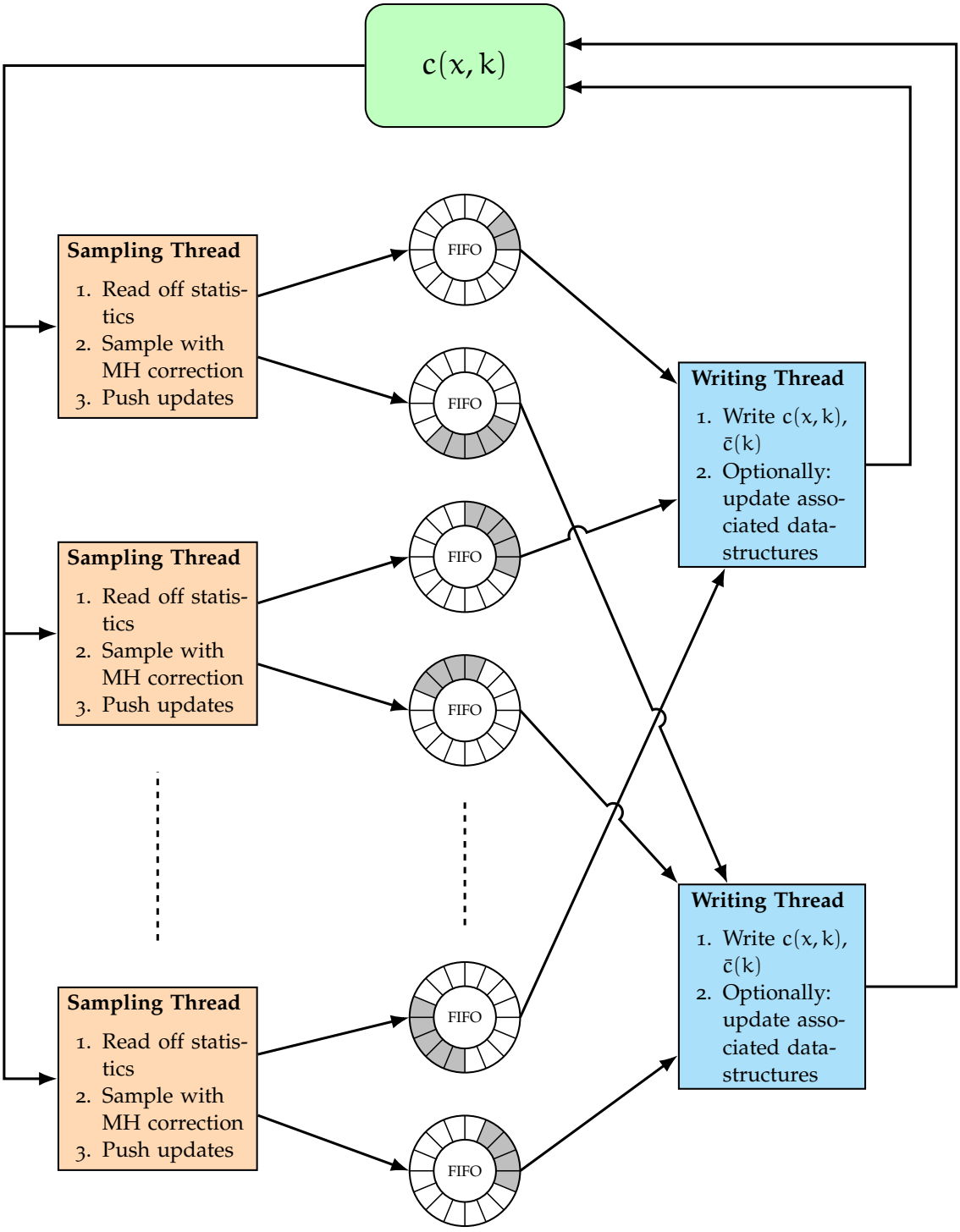


Figure 9.5: Framework for parallel implementation of fast inference of nonparametric models. We have two thread pools: **Sampling threads** are read-only. They process the documents in parallel by sampling new topic assignments based on current statistic values and pushing the updates to a FIFO queue. **Writing threads** receives updates from the FIFO queue and operate on their own non-overlapping shards of the statistics, making them lock-free.

The details of the parallelization framework are as follows: We propose to use create two thread pools. One is tasked with maintaining the data structures and the other pool consists of sampling threads which sample the documents. After drawing a new sample the sampling thread pushes into a FIFO queue (often implemented as circular buffer) and a thread from the other pool pops from this queue the sample to update the global $c(x, k)$ table and any associated data-structure. This way one set of threads only do writes and another only reads, thereby minimizing locks required (can be eliminated completely by using atomics). Further, the writing threads can be sharded according to words, i.e. each writing thread is responsible for a subset of words thereby eliminating any contention among the writing threads. In practice, we only create 1 or few threads for the maintaining the data structure, but use many more threads, at least the number of available CPU cores, for the sampling pool. The framework is illustrated in Figure 9.5.

9.4 EXPERIMENTAL STUDY

We now present empirical studies for the various fast sampling techniques in order to study (i) computation speed (Section 9.4.1), and (ii) convergence speed (Section 9.4.2). For these studies, we evaluate on hierarchical Bayesian nonparametric models (Section 9.2.1), with Dirichlet-Categorical emission model, i.e.

$$f_k(x) = \frac{c(x, k) + \beta}{\bar{c}(k) + \bar{\beta}} \qquad \text{(Dirichlet-Categorical)} \tag{9.57}$$

Using this emission model results in the widely used probabilistic model: HDP-LDA. However, the proposed method can be applied effortlessly to any emission model like Normal-Inverse Wishart in case of topic modeling through Gaussian latent Dirichlet allocation (Gaussian LDA; Zaheer, Das, and Dyer 2015). We pick Dirichlet-Categorical emission model due to their widespread application in various fields spanning natural language processing, computational biology, user activity modeling etc.

METHODS We compare both the single-threaded and multi-threaded version (Section 9.3.4) of the following samplers:

- SDA HDP: Gibbs sampler through sampling by direct assignment (without any speed-up tricks) as proposed by Y. W. Teh et al. (2006) and discussed in Section 9.2.1.1
- Alias HDP: Improved version of SDA HDP by leveraging sparsity with log structured alias sampling (Section 9.3.2) and fast Antoniak sampling using Lemma 9.5.
- STC HDP: Gibbs sampler through sampling table configuration (without any speed-up tricks) as proposed by C. Chen, Du, and W.L. Buntine (2011) and discussed in Section 9.2.1.2
- SSTC HDP: Improved version of STC HDP by utilizing modified posterior (Section 9.3.1) with log structured alias sampling (Section 9.3.2) and direct evaluation of ratio of Stirling numbers (Section 9.3.3).

We follow the tradition in setting $\alpha = 50/K$ and $\beta = 0.01$ where K is the number of topics for all the methods (T.L. Griffiths and Steyvers, 2004; L. Yao, Mimno, and Andrew McCallum, 2009). In order to compare speed we use the execution time as the metric. To measure the accuracy

Dataset	V	L	D	L/V	L/D
20 News	18,127	1,191,840	11,266	65.75	105.79
HEP	37,729	1,548,935	27,770	41.05	55.78
NIPS[1]	12,305	2,301,375	1740	187.03	1322.63
Reuters	69,973	2,624,373	14,377	37.51	182.54
Enron[1]	28,102	6,412,174	40,861	228.18	160.86
ACM	133,325	12,258,310	132,032	41.83	93.06
NY Times	101,330	99,542,127	299,753	982.36	332.08
PubMed[1]	141,043	737,869,085	8,200,000	5,231.52	89.98
Wikipedia	189,583	1,418,129,813	3,825,305	7,480.26	370.72

Table 9.2: Experimental datasets and their statistics. V denotes vocabulary size, L denotes the number of training tokens, D denotes the number of documents, L/V indicates the average number of occurrences of a word, L/D indicates the average length of a document.

and convergence, the log-likelihood on a held out test set $\mathcal{X}_{\text{test}}$ is computed as follows on every dataset:

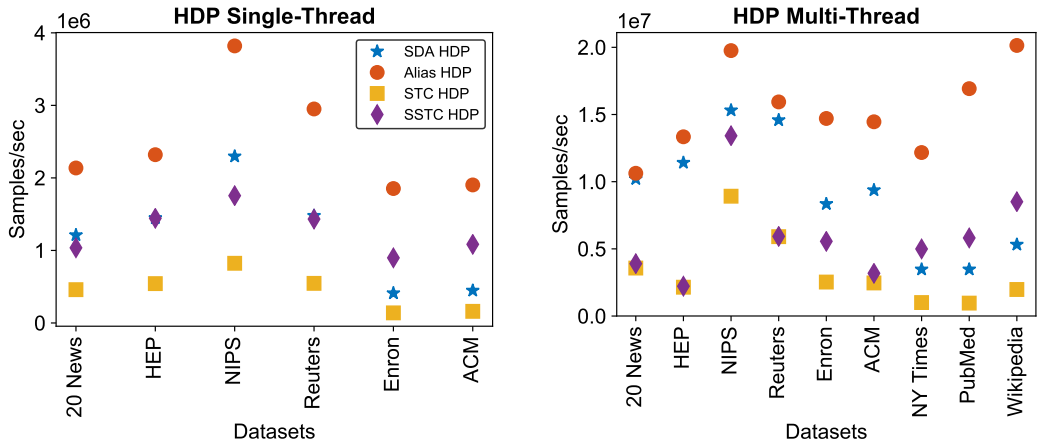
$$\text{LogLikelihood}(\mathcal{X}_{\text{test}}) = \frac{1}{|\mathcal{X}_{\text{test}}|} \sum_{i \in \mathcal{X}_{\text{test}}} \frac{1}{N_j} \sum_{i=1}^{N_j} \log p(x_{ji} | \text{Learnt model}) \tag{9.58}$$

SOFTWARE & HARDWARE All the algorithms are implemented multithreaded in simple C++11 using a distributed setup. Within a node, parallelization is implemented using the work-stealing Fork/Join framework, and the distribution across multiple nodes using the process binding to a socket over MPI. We run our experiments on a cluster of 16 Amazon EC2 c4.8xlarge nodes. There are 36 virtual threads per node and 60GB of memory. For purpose of experiments, all data and calculations are carried out at double floating-point precision. For random number generation we employ Intel® Digital Random Number Generators through instruction RDRAND, which uses thermal noise within the silicon to output a random stream of bits at 3 Gbit/s, producing true random numbers.

DATASETS The performance of all methods are evaluated against nine real-world datasets (Table 9.2). These datasets cover a various range in corpus size, which helps to test the accuracy and efficiency of each method at different data scale. The statistics for each of the dataset is given over Table 9.2 after cleaning by removing stop words and rare words.

9.4.1 *Computation Speed*

We run all algorithms for a *fixed number of iterations*. In Figure 9.6, the samples produced per second by different algorithms over datasets of varying size is reported. We make the following observations: (1) In case of HDP, Alias HDP seems to be clear winner across all datasets. (2) However, there is no free lunch. The huge speed-up comes at the cost of increased memory usage (for storing the data-structures). However, looking at only computational speed paints an incomplete picture. The computationally fastest method may not converge as fast overall.



(a) Sampling rate for single threaded HDP

(b) Sampling rate for multi threaded LDA

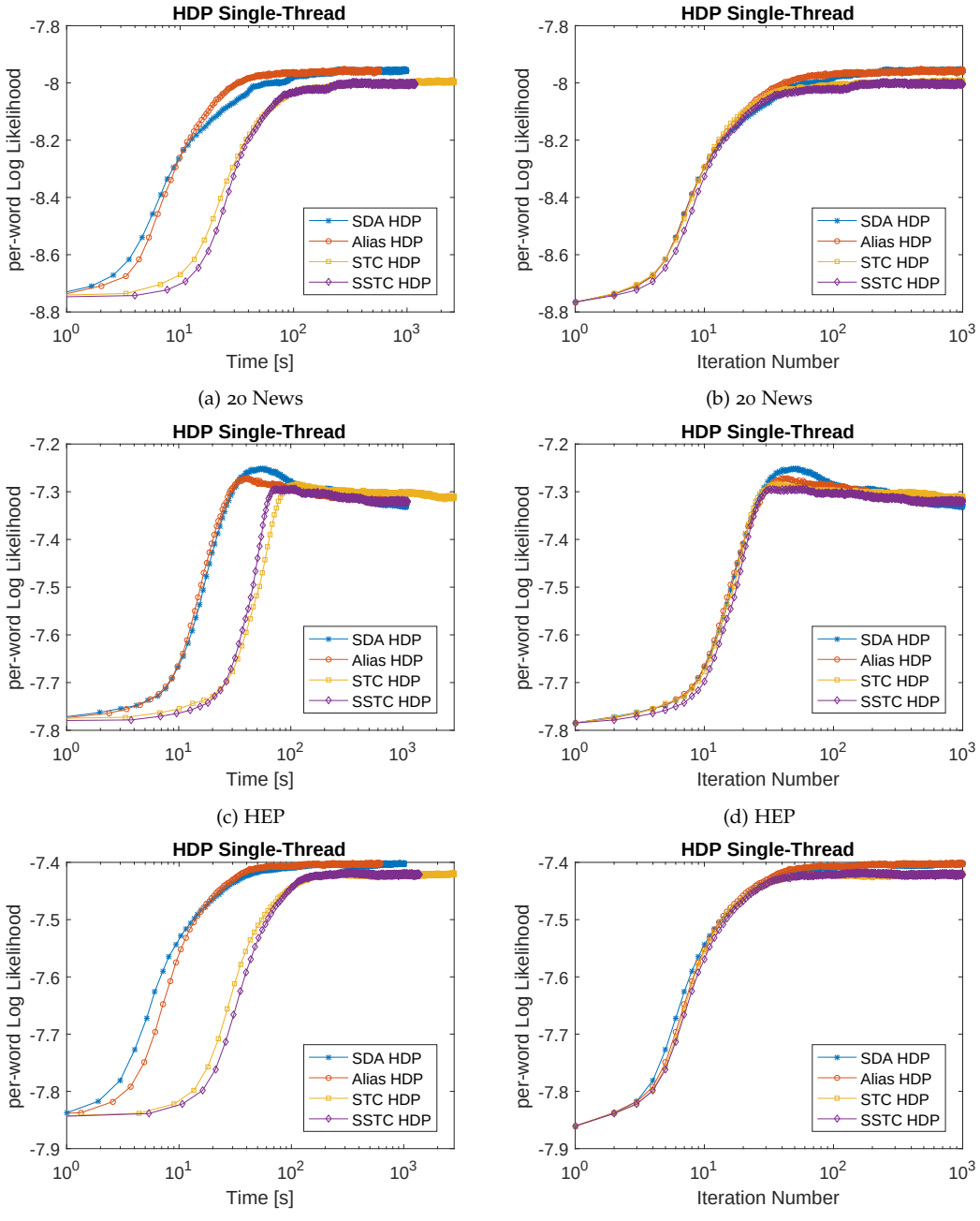
Figure 9.6: Samples produced per second by different algorithms across datasets of varying size. In case of LDA over big datasets (NY Times, PubMed, Wikipedia), Light LDA is the fastest computationally. Similarly for HDPs, Alias HDP is the fastest computationally for all datasets. However, looking at only computational speed paints an incomplete picture. The computationally fastest may not converge as fast overall.

9.4.2 *Convergence Speed*

To compare overall convergence of samplers, we measure how log-likelihood changes over time and over iterations. We run all algorithms for a *fixed number of iterations*. The comparison between various implementation of HDP are given over Figure 9.7, 9.8, 9.9, 9.10, and 9.11. Figure 9.7 and 9.8 consists of relative smaller datasets 10k vocabulary size such as 20 News, HEP, NIPS, and Reuters. Figure 9.9 and 9.10 consists of mid-size datasets with 100k vocabulary size such as Enron, and ACM. Figure 9.11 consists of the largest datasets which are NY Times, PubMed, and Wikipedia consisting over millions of documents and close to billions of trainig tokens. Single threaded performance is reported in Figure 9.7 and 9.9. Multi threaded performance is reported in Figure 9.8, 9.10, and 9.11.

We make the following key observations: (1) At the end, almost all methods, except Light LDA, produce similar per-word log likelihoods, indicating good convergence of all methods. (2) In terms of convergence and per-word log likelihoods, there does not seem to be a difference between single threaded and multi threaded implementations. This implies that our mild assumptions made in Section 9.3.4 regarding negligible difference in global word-topic count $c(x, k)$ between sampling a few tokens holds true. (3) Overall F+Nomad LDA seems to be converging fastest among all methods. (4) The computationally fastest sampler Light LDA, exhibits poor convergence as dataset grows larger and the discrepancy between its proposal and target becomes wider. Thus, Light LDA is not the fastest method overall. (5) Alias HDP consistently converges fastest and produce highest per-word Log likelihood among all methods.

Finally, we sweep LDA for different number of topics and compare against HDP. For this experiment, since many LDA needs to be trained, we reduced the datasize to 10k most frequently words for all datasets. HDP achieves a perplexity equivalent to the best LDA model as shown in Figure 9.12a, 9.12b, and 9.12c.

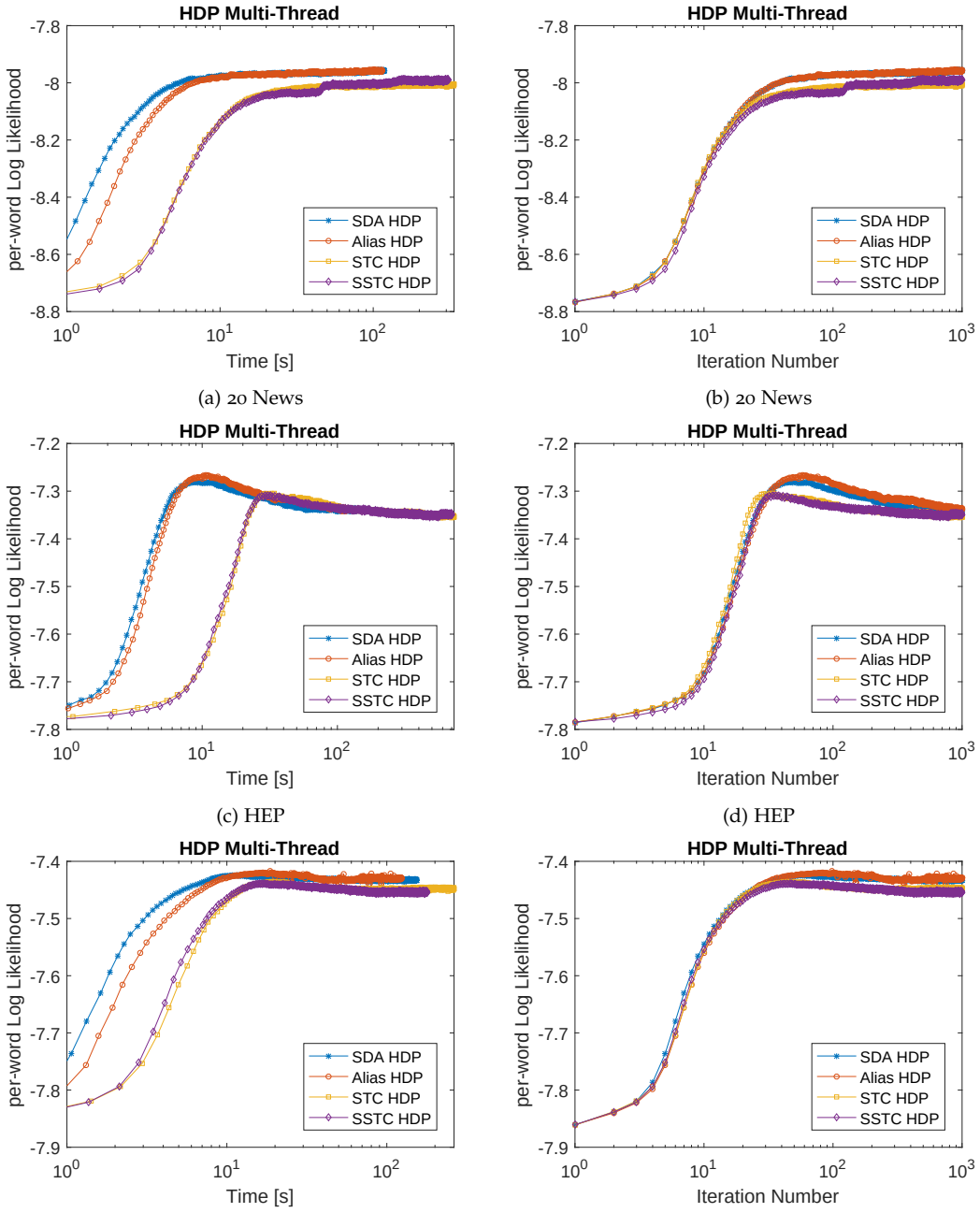


(a) 20 News (b) 20 News

(c) HEP (d) HEP

(e) NIPS (f) NIPS

Figure 9.7: Single Threaded HDP on small datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number.

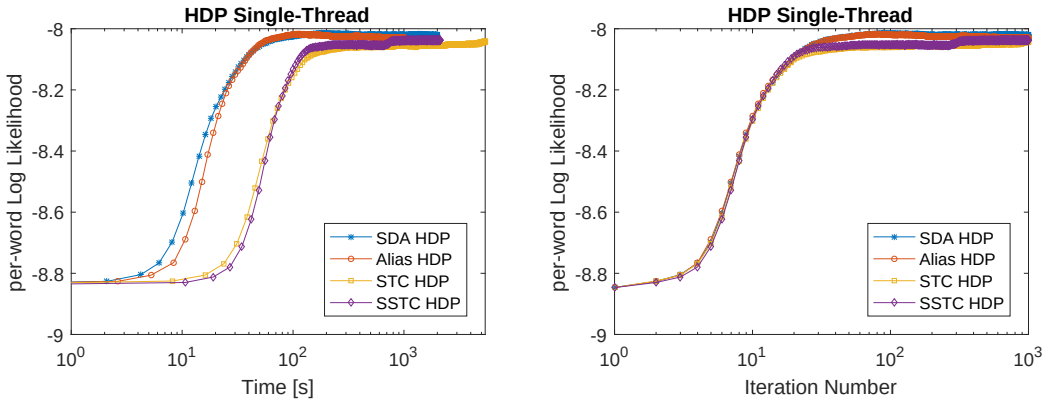


(a) 20 News (b) 20 News

(c) HEP (d) HEP

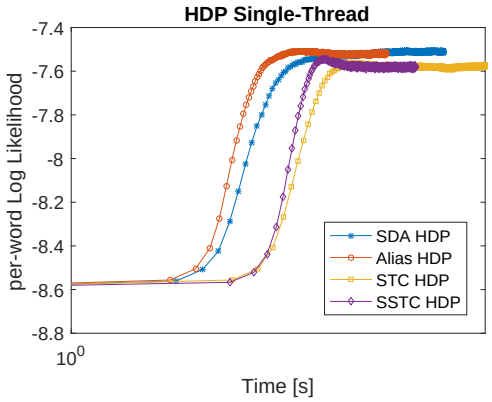
(e) NIPS (f) NIPS

Figure 9.8: Multi Threaded HDP on small datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number.

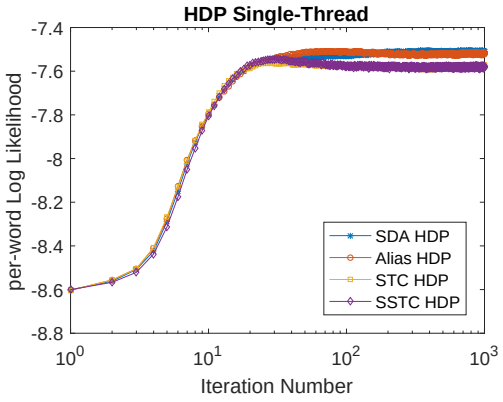


(a) Reuters

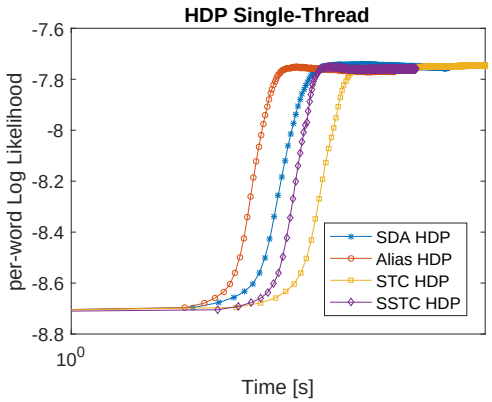
(b) Reuters



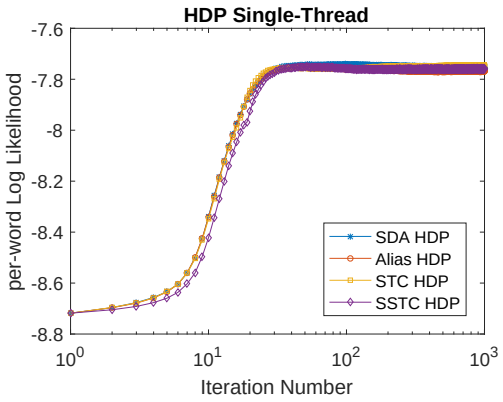
(c) Enron



(d) Enron

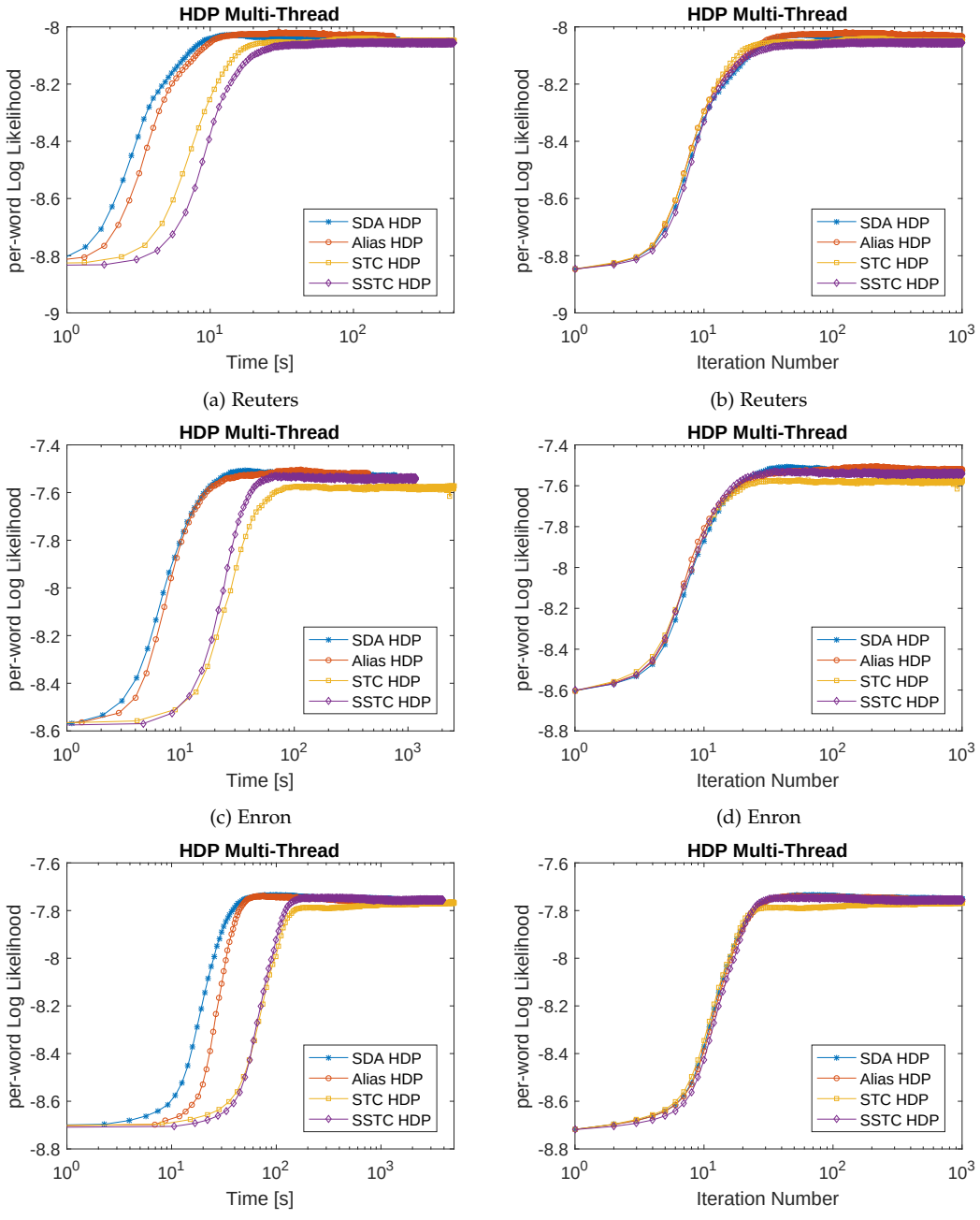


(e) ACM



(f) ACM

Figure 9.9: Single Threaded HDP on medium datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, Alias HDP was the fastest.

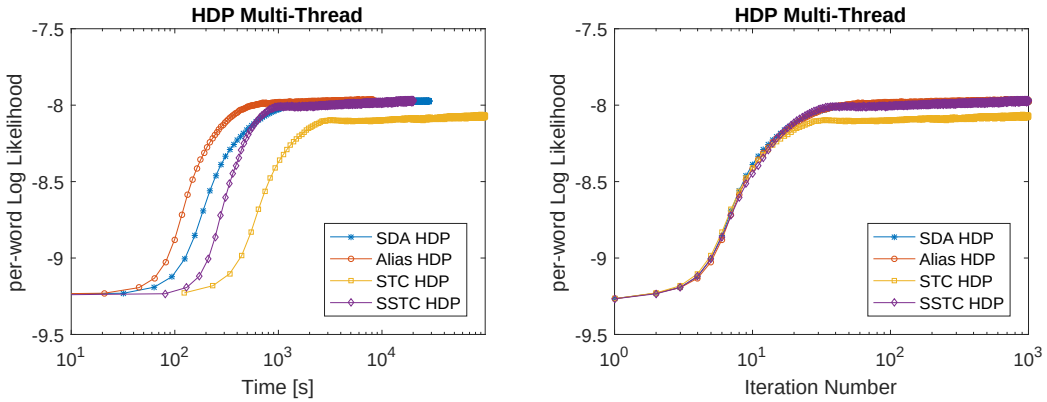


(a) Reuters (b) Reuters

(c) Enron (d) Enron

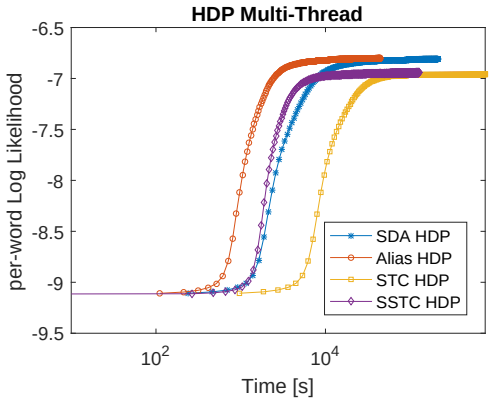
(e) ACM (f) ACM

Figure 9.10: Multi Threaded HDP on medium datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, SDA HDP was the fastest.

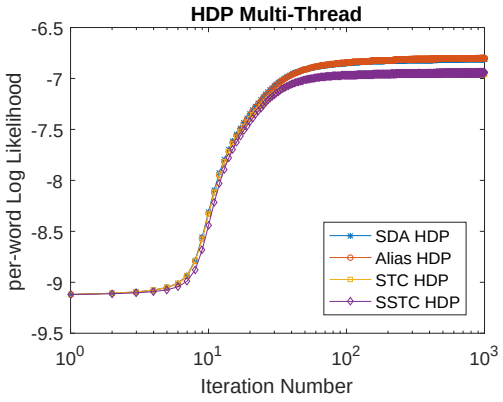


(a) NY Times

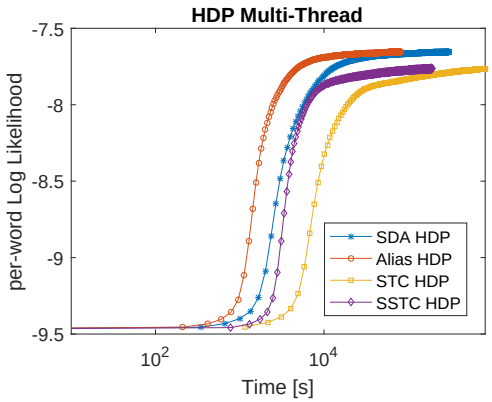
(b) NY Times



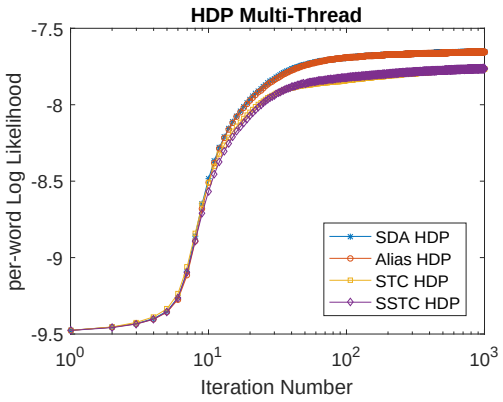
(c) PubMed



(d) PubMed

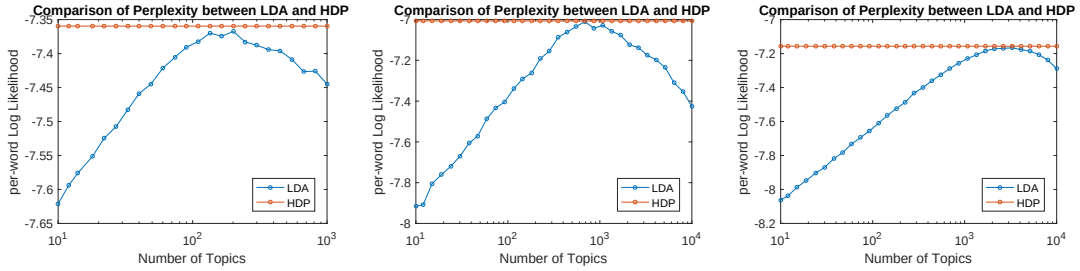


(e) Wikipedia



(f) Wikipedia

Figure 9.11: Multi Threaded HDP on large datasets. The left column plots log-likelihood vs. Time and the right column plots log-likelihood vs. iteration number. While it was not observable previously due to small datasets, the difference between each method is clearer. In the comparison between time, Alias HDP was the fastest.



(a) NIPS (b) Enron (c) NY Times

Figure 9.12: Comparison of perplexity between LDA and HDP on a small, medium, and large dataset.

9.5 POINTS TO PONDER

HDP mixture models are a useful nonparametric Bayesian tool. Unfortunately the computational complexity of the inference algorithms is high. In this chapter, we described a high performance inference procedure for latent variable models with Bayesian nonparametrics, and empirically showed its efficiency by using it to analyse large scale data with billions of tokens and thousands of topics at an unprecedented speed.

Despite the empirical success of the proposed approach, analyzing consistency and other statistical properties of the multi-threaded implementation would be useful. Even analyzing the single threaded implementation of proposed sampler is not trivial. On one hand the sampler is proved to produce the correct equilibrium distribution, as we have shown it satisfies the detailed balance equation and is ergodic. On the other hand, however, tricks used for reducing computational complexity such as MH-within-Gibbs, can have repercussions on the convergence rates, which needs to be understood. Finally, automatic development and use of smart data structure or modification to posterior for any model would be desirable than carefully hand-crafted ones for a given family of models.

10

Escaping Saddle Points

We study the problem of improving solution quality of nonconvex optimization for training deep network portion of a model. Till now we had focused on scaling up the inference for latent variable portion of the model involving terms of the form $p(x) = \sum_z p(x|z)p(z)$, which is usually the main bottleneck. For learning a versatile representation the deep neural network is often needed to achieve high accuracy. Using stochastic gradient descent, the training for the deep network can be parallelized and carried out on heavily multi-cored modern compute infrastructures like GPUs. However, a central challenge to using first-order methods like stochastic gradient descent for optimizing nonconvex problems, which arises in case of deep networks, is the presence of saddle points. First-order methods often get stuck at saddle points, greatly deteriorating their performance. Typically, to escape from saddles one has to use second-order methods. However, most works on second-order methods rely extensively on expensive Hessian-based computations, making them impractical in large-scale settings. To tackle this challenge, we introduce a generic framework that minimizes Hessian-based computations while at the same time provably converging to second- order critical points. Leveraging the *structure* of objective function, our framework carefully alternates between a first-order and a second-order subroutine, using the latter only close to saddle points, and yields convergence results competitive to the state-of-the-art. Empirical results suggest that our strategy also enjoys a good practical performance.

10.1 INTRODUCTION

We study nonconvex *finite-sum* problems of the form

$$\min_{x \in \mathbb{R}^d} \; f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x), \tag{10.1}$$

where neither $f : \mathbb{R}^d \to \mathbb{R}$ nor the individual functions $f_i : \mathbb{R}^d \to \mathbb{R}$ ($i \in [n]$) are necessarily convex. We operate in a general nonconvex setting except for few smoothness assumptions like Lipschitz continuity of the gradient and Hessian. Optimization problems of this form arise naturally in machine learning and statistics as empirical risk minimization (ERM) and M-estimation respectively.

In the large-scale settings, algorithms based on first-order information of functions f_i are typically favored as they are relatively inexpensive and scale seamlessly. An algorithm widely used in practice is stochastic gradient descent (SGD), which has the iterative update:

$$x_{t+1} = x_t - \eta_t \nabla f_{i_t}(x_t), \tag{10.2}$$

where $i_t \in [n]$ is a randomly chosen index and η_t is a learning rate. Under suitable selection of the learning rate, we can show that SGD converges to a point x that, in expectation, satisfies the stationarity condition $\|\nabla f(x)\| \leqslant \epsilon$ in $O(1/\epsilon^4)$ iterations (Ghadimi and Lan, 2013). This result has two critical weaknesses: (i) It does not ensure convergence to local optima or second-order critical points; (ii) The rate of convergence of the SGD algorithm is slow.

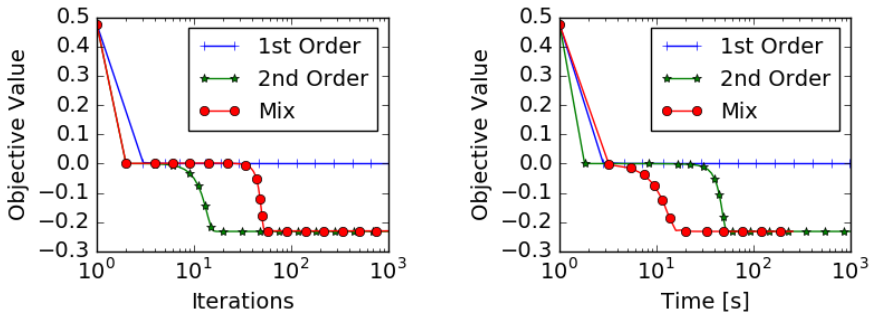


Figure 10.1: First order methods like GD can potentially get stuck at saddle points. Second-order methods can escape it in very few iterations (as observed in the left plot) but at the cost of expensive Hessian based iterations (see time plot to the right). The proposed framework, which is a novel mix of the two strategies, can escape saddle points *faster* in time by carefully trading off computation and iteration complexity.

For general nonconvex problems, one has to settle for a more modest goal than sub-optimality, as finding the global minimizer of finite-sum nonconvex problem will be in general intractably hard. Unfortunately, SGD does not even ensure second-order critical conditions such as local optimality since it can get stuck at saddle points. This issue has recently received considerable attention in the ML community, especially in the context of deep learning (Choromanska et al., 2015; Y. N. Dauphin et al., 2014; Y. Dauphin, Vries, and Y. Bengio, 2015). These works argue that saddle points are highly prevalent in most optimization paths, and are the primary obstacle for training large deep networks. To tackle this issue and achieve a second-order critical point for which $\|\nabla f\| \leqslant \epsilon$ and $\nabla^2 f \succeq -\sqrt{\epsilon}\mathbb{I}$, we need algorithms that either use the Hessian explicitly or exploit its structure.

A key work that explicitly uses Hessians to obtain faster convergence rates is the cubic regularization (CR) method (Nesterov and Polyak, 2006). In particular, Nesterov and Polyak (2006) showed that CR requires $O(1/\epsilon^{3/2})$ iterations to achieve the second-order critical conditions. However, each iteration of CR is expensive as it requires computing the Hessian and solving multiple linear systems, each of which has complexity $O(d^\omega)$ (ω is the matrix multiplication constant), thus, undermining the benefit of its faster convergence. Recently, N. Agarwal, Allen Zhu, et al. (2016) designed an algorithm to solve the CR more efficiently, however, it still exhibits slower convergence in practice compared to first-order methods. Both of these approaches use Hessian based optimization in each iteration, which make them slow in practice.

A second line of work focuses on using Hessian information (or its structure) whenever the method gets stuck at stationary points that are not second-order critical. To our knowledge, the first work in this line is Ge et al. (2015), which shows that for a class of functions that satisfy a special property called "strict-saddle" property, a noisy variant of SGD can converge to a point close to a local minimum. For this class of functions, points close to saddle points have a Hessian with a large negative eigenvalue, which proves instrumental in escaping saddle points using an isotropic noise. While such a noise-based method is appealing as it only uses first-order information, it has a very bad dependence on the dimension d, and furthermore, the result only holds when the strict-saddle property is satisfied (Ge et al., 2015). More recently, Carmon et al. (2018) presented a new faster algorithm that alternates between first-order and second-order subroutines. However, their algorithm is designed for the simple case of $n = 1$ in (10.1) and hence, can be expensive in practice.

Inspired by this line of work, we develop a general framework for finding second-order critical points. The key idea of our framework is to use first-order information for the most part of the optimization process and invoke Hessian information only when stuck at stationary points that are not second-order critical. We summarize the key idea and main contributions of this paper below.

MAIN CONTRIBUTIONS We develop an algorithmic framework for converging to second-order critical points and provide convergence analysis for it. Our framework carefully alternates between two subroutines that use gradient and Hessian information, respectively, and ensures second-order criticality. Furthermore, we present two instantiations of our framework and provide convergence rates for them. In particular, we show that a simple instance of our framework, based on SVRG, achieves convergence rates competitive with the current state-of-the-art methods; thus highlighting the simplicity and applicability of our framework. Finally, we demonstrate the empirical performance of a few algorithms encapsulated by our framework and show their superior performance.

RELATED WORK There is a vast literature on algorithms for solving optimization problems of the form (10.1). A classical approach for solving such optimization problems is SGD, which dates back at least to the seminal work of Robbins and Monro (1951). Since then, SGD has been a subject of extensive research, especially in the convex setting (Léon Bottou, 1991; Kushner and Clark, 2012; Ljung, 1977; Poljak and Tsypkin, 1973). Recently, new faster methods, called variance reduced (VR) methods, have been proposed for convex finite-sum problems. VR methods attain faster convergence by reducing the variance in the stochastic updates of SGD, *c.f.* A. J. Defazio, Caetano, and Domke (2014), A. Defazio, Bach, and Lacoste-Julien (2014), R. Johnson and T. Zhang (2013), Konečny et al. (2016), Mark Schmidt, Le Roux, and Bach (2017), and Shalev-Shwartz and T. Zhang (2013). Accelerated variants of these methods achieve the lower bounds proved in A. Agarwal and Leon Bottou (2015) and Lan and Zhou (2017), thereby settling the question of their optimality. Furthermore, S. Reddi et al. (2015) developed an asynchronous framework for VR methods and demonstrated their benefits in parallel environments.

Most of the aforementioned prior works study stochastic methods in convex or very specialized nonconvex settings that admit theoretical guarantees on sub-optimality. For the general nonconvex setting, it is only recently that non-asymptotic convergence rate analysis for SGD and its variants was obtained by Ghadimi and Lan (2013), who showed that SGD ensures $\|\nabla f\| \leqslant \epsilon$ (in expectation) in $O(1/\epsilon^4)$ iterations. A similar rate for parallel and distributed SGD was shown in Lian et al. (2015). For these problems, Sashank J. Reddi, Hefny, et al. (2016), Sashank J. Reddi, Sra, et al. (2016), and Sashank J Reddi et al. (2016) proved faster convergence rates that ensure the same optimality criteria in $O(n + n^{2/3}/\epsilon^2)$, which is an order $n^{1/3}$ faster than GD. While these methods ensure convergence to *stationary* points at a faster rate, the question of convergence to local minima (or in general to second-order critical points) has not been addressed. To our knowledge, convergence rates to second-order critical points (defined in Definition 10.1) for general nonconvex functions was first studied by Nesterov and Polyak (2006). However, each iteration of the algorithm in Nesterov and Polyak (2006) is prohibitively expensive since it requires eigenvalue decompositions, and hence, is unsuitable for large-scale high-dimensional problems. More recently, N. Agarwal, Allen Zhu, et al. (2016) and Carmon et al. (2018) presented algorithms for finding second-order critical points by tackling some practical issues that arise in Nesterov and Polyak (2006). However, these algorithms are either only applicable to a restricted setting or heavily use Hessian based computations, making them unappealing from a practical standpoint. Also, recently Carmon et al. (2017) proposed a fast accelerated first-order method for nonconvex optimization; however, similar to Carmon et al. (2018), it requires computation of the full gradient at each iteration and only provides convergence to stationary

points. *Noisy* variants of first-order methods have also been shown to escape saddle points (Ge et al., 2015; C. Jin et al., 2017; K. Y. Levy, 2016), however, they have strong dependence on either n or d, both of which are undesirable.

10.2 BACKGROUND & PROBLEM SETUP

We assume that each of the functions f_i in (10.1) is L-smooth, *i.e.*, $\|\nabla f_i(x) - \nabla f_i(y)\| \leqslant L\|x - y\|$ for all $i \in [n]$. Furthermore, we assume that the Hessian of f in (10.1) is Lipschitz, *i.e.*, we have

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leqslant M\|x - y\|, \tag{10.3}$$

for all $x, y \in \mathbb{R}^d$. Such a condition is typically necessary to ensure convergence of algorithms to the second-order critical points (Nesterov and Polyak, 2006). In addition to the above smoothness conditions, we also assume that the function f is bounded below, *i.e.*, $f(x) \geqslant B$ for all $x \in \mathbb{R}^d$.

In order to measure stationarity of an iterate x, similar to Ghadimi and Lan (2013), Nesterov (2004), and Nesterov and Polyak (2006), we use the condition $\|\nabla f(x)\| \leqslant \epsilon$. In this paper, we are interested in convergence to second-order critical points. Thus, in addition to stationarity, we also require the solution to satisfy the Hessian condition $\nabla^2 f(x) \succeq -\gamma \mathbb{I}$ (Nesterov and Polyak, 2006). For iterative algorithms, we require both $\epsilon, \gamma \to 0$ as the number of iterations $T \to \infty$. When all saddle points are non-degenerate, such a condition implies convergence to a local optimum.

Definition 10.1 *An algorithm $\mathcal{A}$ is said to obtain a point x that is a (ϵ, γ)-second order critical point if $\mathbb{E}[\|\nabla f(x)\|] \leqslant \epsilon$ and $\nabla^2 f(x) \succeq -\gamma \mathbb{I}$, where the expectation is over any randomness in $\mathcal{A}$.*

We must exercise caution while interpreting results pertaining to (ϵ, γ)-second order critical points. Such points need not be close to any local minima either in objective function value, or in the domain of (10.1). For our algorithms, we use only an Incremental First-order Oracle (IFO) (A. Agarwal and Leon Bottou, 2015) and an Incremental Second-order Oracle (ISO), defined below.

Definition 10.2 *An IFO takes an index $i \in [n]$ and a point $x \in \mathbb{R}^d$, and returns the pair $(f_i(x), \nabla f_i(x))$. An ISO takes an index $i \in [n]$, point $x \in \mathbb{R}^d$ and vector $v \in \mathbb{R}^d$ and returns the vector $\nabla^2 f_i(x)v$.*

IFO and ISO calls are typically cheap, with ISO call being relatively more expensive. In many practical settings that arise in machine learning, the time complexity of these oracle calls is linear in d (N. Agarwal, Bullins, and Hazan, 2016; Pearlmutter, 1994). For clarity and clean comparison, the dependence of time complexity on Lipschitz constant L, M, initial point and any polylog factors in the results is hidden.

10.3 GENERIC FRAMEWORK

In this section, we propose a generic framework for escaping saddle points while solving non-convex problems of form (10.1). One of the primary difficulties in reaching a second-order critical point is the presence of saddle points. To evade such points, one needs to use properties of both gradients and Hessians. To this end, our framework is based on two core subroutines: Gradient-Focused-Optimizer and Hessian-Focused-Optimizer.

Algorithm 10.1 Generic Framework

Input - Initial point: x^0, total iterations T, error threshold parameters ϵ, γ and probability p
for $t = 1$ **to** T **do**
  $(y^t, z^t) =$ GRADIENT-FOCUSED-OPTIMIZER(x^{t-1}, ϵ) (refer to **G.1** and **G.2**)
  Choose u^t as y^t with probability p and z^t with probability $1-p$
  $(x^{t+1}, \tau^{t+1}) =$ HESSIAN-FOCUSED-OPTIMIZER(u^t, ϵ, γ) (refer to **H.1** and **H.2**)
  if $\tau^{t+1} = \emptyset$ **then**
    Output set $\{x^{t+1}\}$
  end if
end for
Output set $\{y^1, ..., y^T\}$

The idea is to use these two subroutines, each focused on different aspects of the optimization procedure. GRADIENT-FOCUSED-OPTIMIZER focuses on using gradient information for decreasing the function. On its own, the GRADIENT-FOCUSED-OPTIMIZER might not converge to a local minimizer since it can get stuck at a saddle point. Hence, we require the subroutine HESSIAN-FOCUSED-OPTIMIZER to help avoid saddle points. A natural idea is to interleave these subroutines to obtain a second-order critical point. But it is not even clear if such a procedure even converges. We propose a carefully designed procedure that effectively balances these two subroutines, which not only provides meaningful theoretical guarantees, but also translates into strong empirical gains in practice.

Algorithm 10.1 provides pseudocode of our framework. Observe that the presented algorithm is still abstract, since it does not specify the subroutines GRADIENT-FOCUSED-OPTIMIZER and HESSIAN-FOCUSED-OPTIMIZER. These subroutines determine the crucial update mechanism of the algorithm. We will present specific instance of these subroutines in the next section, but we assume the following properties to hold for these subroutines.

- GRADIENT-FOCUSED-OPTIMIZER: Suppose $(y, z) =$ GRADIENT-FOCUSED-OPTIMIZER(x, n, ϵ), then there exists positive function $g : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{R}^+$, such that

  G.1 $\mathbb{E}[f(y)] \leqslant f(x)$,

  G.2 $\mathbb{E}[\|\nabla f(y)\|^2] \leqslant \frac{1}{g(n,\epsilon)} \mathbb{E}[f(x) - f(z)]$.

  Here the outputs $y, z \in \mathbb{R}^d$. The expectation in the conditions above is over any randomness that is a part of the subroutine. The function g will be critical for the overall rate of Algorithm 10.1. Typically, GRADIENT-FOCUSED-OPTIMIZER is a first-order method, since the primary aim of this subroutine is to focus on gradient based optimization.

- HESSIAN-FOCUSED-OPTIMIZER: Suppose $(y, \tau) =$ HESSIAN-FOCUSED-OPTIMIZER(x, n, ϵ, γ) where $y \in \mathbb{R}^d$ and $\tau \in \{\emptyset, \diamond\}$. If $\tau = \emptyset$, then y is a (ϵ, γ)-second order critical point with probability at least $1 - q$. Otherwise if $\tau = \diamond$, then y satisfies the following condition:

  H.1 $\mathbb{E}[f(y)] \leqslant f(x)$,

  H.2 $\mathbb{E}[f(y)] \leqslant f(x) - h(n, \epsilon, \gamma)$ when $\lambda_{\min}(\nabla^2 f(x)) \leqslant -\gamma$ for some function $h : \mathbb{N} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$.

  Here the expectation is over any randomness in subroutine HESSIAN-FOCUSED-OPTIMIZER. The two conditions ensure that the objective function value, in expectation, never increases and furthermore, decreases with a certain rate when $\lambda_{\min}(\nabla^2 f(x)) \leqslant -\gamma$. In general, this subroutine utilizes the Hessian or its properties for minimizing the objec-

tive function. Typically, this is the most expensive part of the Algorithm 10.1 and hence, needs to be invoked judiciously.

The key aspect of these subroutines is that they, in expectation, never increase the objective function value. The functions g and h will determine the convergence rate of Algorithm 10.1. In order to provide a concrete implementation, we need to specify the aforementioned subroutines. Before we delve into those details, we will provide a generic convergence analysis for Algorithm 10.1.

10.3.1 *Convergence Analysis*

Theorem 10.3 *Let* $\Delta = f(x^0) - B$, *and* $\theta = \min\{(1-p)\epsilon^2 g(n,\epsilon), ph(n,\epsilon,\gamma)\}$. *Also, let* Γ *be the set output by Algorithm 10.1 with* Gradient-Focused-Optimizer *satisfying* **G.1** *and* **G.2** *and* Hessian-Focused-Optimizer *satisfying* **H.1** *and* **H.2**. *Further, let* T *be such that* $T > \Delta/\theta$.

Suppose the multiset $S = \{i_1, \ldots i_k\}$ *contains* k *indices selected independently and uniformly from* $\{1, \ldots, |\Gamma|\}$. *Then the following holds for the indices in* S*:*

1. y^t *(where* $t \in S$*) is an* (ϵ,γ)*-critical point with probability at least* $1 - \max(\Delta/(T\theta), q)$.
2. *If* $k = O(\log(1/\zeta)/\min(\log(\Delta/(T\theta)), \log(1/q)))$, *with at least probability* $1-\zeta$, *at least one iterate* y^t *where* $t \in S$ *is an* (ϵ,γ)*-critical point.*

Proof The case of $\tau = \emptyset$ can be handled in a straightforward manner, so let us focus on the case where $\tau = \diamond$. We split our analysis into cases, each analyzing the change in objective function value depending on second-order criticality of y^t.

We start with the case where the gradient condition of second-order critical point is violated and then proceed to the case where the Hessian condition is violated.

CASE 1 : $\mathbb{E}[\|\nabla f(y^t)\|] \geqslant \epsilon$ for some $t > 0$

We first observe the following: $\mathbb{E}[\|\nabla f(y^t)\|^2] \geqslant (\mathbb{E}\|\nabla f(y^t)\|)^2 \geqslant \epsilon^2$. This follows from a straightforward application of Jensen's inequality. From this inequality, we have the following:

$$\epsilon^2 \leqslant \mathbb{E}[\|\nabla f(y^t)\|^2] \leqslant \frac{1}{g(n,\epsilon)}\mathbb{E}[f(x^{t-1}) - f(z^t)]. \tag{10.4}$$

This follows from the fact that y^t is the output of Gradient-Focused-Optimizer subroutine, which satisfies the condition that for (y, z) = Gradient-Focused-Optimizer(x, n, ϵ), we have

$$\mathbb{E}[\|\nabla f(y)\|^2] \leqslant \frac{1}{g(n,\epsilon)}\mathbb{E}[f(x) - f(z)].$$

From Equation (10.4), we have

$$\mathbb{E}[f(z^t)] \leqslant \mathbb{E}[f(x^{t-1})] - \epsilon^2 g(n,\epsilon).$$

Furthermore, due to the property of non-increasing nature of Gradient-Focused-Optimizer, we also have $\mathbb{E}[y^t] \leqslant \mathbb{E}[f(x^{t-1})]$.

We now focus on the HESSIAN-FOCUSED-OPTIMIZER subroutine. From the property of HESSIAN-FOCUSED-OPTIMIZER that the objective function value is non-increasing, we have $\mathbb{E}[f(x^t)] \leqslant \mathbb{E}[f(u^t)]$. Therefore, combining with the above inequality, we have

$$\begin{aligned}\mathbb{E}[f(x^t)] &\leqslant \mathbb{E}[f(u^t)] \\ &= p\mathbb{E}[f(y^t)] + (1-p)\mathbb{E}[f(z^t)] \\ &\leqslant p\mathbb{E}[f(x^{t-1})] + (1-p)(\mathbb{E}[f(x^{t-1})] - \epsilon^2 g(n,\epsilon)) \\ &= \mathbb{E}[f(x^{t-1})] - (1-p)\epsilon^2 g(n,\epsilon). \end{aligned} \tag{10.5}$$

The first equality is due to the definition of u^t in Algorithm 10.1. Therefore, when the gradient condition is violated, irrespective of whether $\lambda_{\min}(\nabla^2 f(x)) \leqslant -\gamma$ or $\nabla^2 f(y^t) \succeq -\gamma \mathbb{I}$, the objective function value always decreases by at least $\epsilon^2 g(n,\epsilon)$.

CASE II : $\mathbb{E}[\|\nabla f(y^t)\|] < \epsilon$ and $\lambda_{\min}(\nabla^2 f(x)) \leqslant -\gamma$ for some $t > 0$

In this case, we first note that for y = HESSIAN-FOCUSED-OPTIMIZER(x, n, ϵ, γ) and $\lambda_{\min}(\nabla^2 f(x)) \leqslant -\gamma$, we have $\mathbb{E}[f(y)] \leqslant f(x) - h(n,\epsilon,\gamma)$. Observe that x^t = HESSIAN-FOCUSED-OPTIMIZER$(u^t, n, \epsilon, \gamma)$. Therefore, if $u^t = y^t$ and $\lambda_{\min}(\nabla^2 f(x)) \leqslant -\gamma$, then we have

$$\mathbb{E}[f(x^t)|u^t = y^t] \leqslant f(y^t) - h(n,\epsilon,\gamma) \leqslant f(x^{t-1}) - h(n,\epsilon,\gamma).$$

The second inequality is due to the non-increasing property of GRADIENT-FOCUSED-OPTIMIZER. On the other hand, if $u^t = z^t$, we have hand, if we have $\mathbb{E}[f(x^t)|u^t = z^t] \leqslant f(z^t)$. This is due to the non-increasing property of HESSIAN-FOCUSED-OPTIMIZER. Combining the above two inequalities and using the law of total expectation, we get

$$\begin{aligned}\mathbb{E}[f(x^t)] &= p\mathbb{E}[f(x^t)|u^t = y^t] + (1-p)\mathbb{E}[f(x^t)|u^t = z^t] \\ &\leqslant p\left(\mathbb{E}[f(y^t)] - h(n,\epsilon,\gamma)\right) + (1-p)\mathbb{E}[f(z^t)] \\ &\leqslant p\left(\mathbb{E}[f(x^{t-1})] - h(n,\epsilon,\gamma)\right) + (1-p)\mathbb{E}[f(x^{t-1})] \\ &= \mathbb{E}[f(x^{t-1})] - ph(n,\epsilon,\gamma). \end{aligned} \tag{10.6}$$

The second inequality is due to he non-increasing property of GRADIENT-FOCUSED-OPTIMIZER. Therefore, when the hessian condition is violated, the objective function value always decreases by at least $ph(n,\epsilon,\gamma)$.

CASE III : $\mathbb{E}[\|\nabla f(y^t)\|] < \epsilon$ and $\nabla^2 f(y^t) \succeq -\gamma \mathbb{I}$ for some $t > 0$

This is the favorable case for the algorithm. The only condition to note is that the objective function value will be non-increasing in this case too. This is, again, due to the non-increasing properties of subroutines GRADIENT-FOCUSED-OPTIMIZER and HESSIAN-FOCUSED-OPTIMIZER. In general, greater the occurrence of this case during the course of the algorithm, higher will the probability that the output of our algorithm satisfies the desired property.

The key observation is that Case I & II cannot occur large number of times since each of these cases strictly decreases the objective function value. In particular, from Equation (10.5) and (10.6), it is easy to see that each occurrence of Case I & II the following holds:

$$\mathbb{E}[f(x^t)] \leqslant \mathbb{E}[f(x^{t-1})] - \theta,$$

where $\theta = \min((1-p)\epsilon^2 g(n,\epsilon), ph(n,\epsilon,\gamma))$. Furthermore, the function f is lower bounded by B, thus, Case I & II cannot occur more than $(f(x^0) - B)/\theta$ times. Therefore, the probability of

Algorithm 10.2 SVRG(x^0, ϵ)

Input: $x_m^0 = x^0 \in \mathbb{R}^d$, epoch length m, step sizes $\{\eta_i > 0\}_{i=0}^{m-1}$, iterations T_g, $S = \lceil T_g/m \rceil$
for $s = 0$ **to** $S - 1$ **do**
  $\tilde{x}^s = x_0^{s+1} = x_m^s$
  $g^{s+1} = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\tilde{x}^s)$
  for $t = 0$ **to** $m - 1$ **do**
    Uniformly randomly pick i_t from $\{1, \ldots, n\}$
    $v_t^{s+1} = \nabla f_{i_t}(x_t^{s+1}) - \nabla f_{i_t}(\tilde{x}^s) + g^{s+1}$
    $x_{t+1}^{s+1} = x_t^{s+1} - \eta_t v_t^{s+1}$
  end for
end for
Output: (y, z) where y is Iterate x_a chosen uniformly random from $\{\{x_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$ and $z = x_m^S$.

occurrence of Case III is at least $1 - (f(x^0) - B)/(T\theta)$, which completes the first part of the proof.

The second part of the proof simply follows from first part. As seen above, the probability of Case I & II is at most $(f(x^0) - B)/T\theta$. Therefore, probability that an element of the set S falls in Case III is at least $1 - ((f(x^0) - B)/T\theta)^k$, which gives us the required result for the second part. ■

The key point regarding the above result is that the overall convergence rate depends on the magnitude of both functions g and h. Theorem 10.3 shows that the slowest amongst the subroutines Gradient-Focused-Optimizer and Hessian-Focused-Optimizer governs the overall rate of Algorithm 10.1. Thus, it is important to ensure that both these procedures have good convergence. Also, note that the optimal setting for p based on the result above satisfies $1/p = 1/\epsilon^2 g(n, \epsilon) + 1/h(n, \epsilon, \gamma)$. We defer further discussion of convergence to next section, where we present more specific convergence and rate analysis.

10.4 CONCRETE INSTANTIATIONS

We now present specific instantiations of our framework in this section. Before we state our key results, we discuss an important subroutine that is used as Gradient-Focused-Optimizer for rest of this paper: Svrg. We give a brief description of the algorithm in this section and show that it meets the conditions required for a Gradient-Focused-Optimizer. Svrg (R. Johnson and T. Zhang, 2013; Sashank J. Reddi, Hefny, et al., 2016) is a stochastic algorithm recently shown to be very effective for reducing variance in finite-sum problems. We seek to understand its benefits for nonconvex optimization, with a particular focus on the issue of escaping saddle points. Algorithm 10.2 presents Svrg's pseudocode.

Observe that Algorithm 10.2 is an epoch-based algorithm. At the start of each epoch s, a full gradient is calculated at the point $\tilde{x}^s$, requiring n calls to the IFO. Within its inner loop Svrg performs m stochastic updates. Suppose m is chosen to be $O(n)$ (typically used in practice), then the total IFO calls per epoch is $\Theta(n)$. Strong convergence rates have been proved Algorithm 10.2 in the context of convex and nonconvex optimization (R. Johnson and T. Zhang, 2013; Sashank J. Reddi, Hefny, et al., 2016). The following result shows that Svrg meets the requirements of a Gradient-Focused-Optimizer.

Lemma 10.4 *Suppose* $\eta_t = \eta = 1/4Ln^{2/3}$, $m = n$ *and* $T_g = T_\epsilon$, *which depends on* ϵ, *then Algorithm 10.2 is a* GRADIENT-FOCUSED-OPTIMIZER *with* $g(n, \epsilon) = T_\epsilon/40Ln^{2/3}$.

Proof The proof follows from the analysis in Sashank J. Reddi, Hefny, et al. (2016) with some additional reasoning. We need to show two properties: **G.1** and **G.2**, both of which are based on objective function value. To this end, we start with an update in the s^{th} epoch. We have the following:

$$\begin{aligned}\mathbb{E}[f(x_{t+1}^{s+1})] &\leqslant \mathbb{E}[f(x_t^{s+1}) + \langle \nabla f(x_t^{s+1}), x_{t+1}^{s+1} - x_t^{s+1}\rangle + \tfrac{L}{2}\|x_{t+1}^{s+1} - x_t^{s+1}\|^2] \\ &\leqslant \mathbb{E}[f(x_t^{s+1}) - \eta_t\|\nabla f(x_t^{s+1})\|^2 + \tfrac{L\eta_t^2}{2}\|v_t^{s+1}\|^2]. \end{aligned} \tag{10.7}$$

The first inequality is due to L-smoothness of the function f . The second inequality simply follows from the unbiasedness of SVRG update in Algorithm 10.2. For the analysis of the algorithm, we need the following Lyapunov function:

$$A_t^{s+1} := \mathbb{E}[f(x_t^{s+1}) + \mu_t\|x_t^{s+1} - \tilde{x}^s\|^2].$$

This function is a combination of objective function and the distance of the current iterate from the latest snapshot $\tilde{x}_s$. Note that the term μ_t is introduced only for the analysis and is not part of the algorithm (see Algorithm 10.2). Here $\{\mu_t\}_{t=0}^m$ is chosen such the following holds:

$$\mu_t = \mu_{t+1}(1 + \eta_t\beta_t + 2\eta_t^2L^2) + \eta_t^2L^3,$$

for all $t \in \{0, \cdots, m-1\}$ and $\mu_m = 0$. For bounding the Lypunov function A, we need the following bound on the distance of the current iterate from the latest snapshot:

$$\begin{aligned}&\mathbb{E}[\|x_{t+1}^{s+1} - \tilde{x}^s\|^2] = \mathbb{E}[\|x_{t+1}^{s+1} - x_t^{s+1} + x_t^{s+1} - \tilde{x}^s\|^2] \\ &= \mathbb{E}[\|x_{t+1}^{s+1} - x_t^{s+1}\|^2 + \|x_t^{s+1} - \tilde{x}^s\|^2 + 2\langle x_{t+1}^{s+1} - x_t^{s+1}, x_t^{s+1} - \tilde{x}^s\rangle] \\ &= \mathbb{E}[\eta_t^2\|v_t^{s+1}\|^2 + \|x_t^{s+1} - \tilde{x}^s\|^2] - 2\eta_t\mathbb{E}[\langle \nabla f(x_t^{s+1}), x_t^{s+1} - \tilde{x}^s\rangle] \\ &\leqslant \mathbb{E}[\eta_t^2\|v_t^{s+1}\|^2 + \|x_t^{s+1} - \tilde{x}^s\|^2] + 2\eta_t\mathbb{E}\left[\tfrac{1}{2\beta_t}\|\nabla f(x_t^{s+1})\|^2 + \tfrac{1}{2}\beta_t\|x_t^{s+1} - \tilde{x}^s\|^2\right]. \end{aligned} \tag{10.8}$$

The second equality is due to the unbiasedness of the update of SVRG. The last inequality follows from a simple application of Cauchy-Schwarz and Young's inequality. Substituting Equation (10.7) and Equation (10.8) into the Lypunov function A_{t+1}^{s+1}, we obtain the following:

$$\begin{aligned}A_{t+1}^{s+1} &\leqslant \mathbb{E}[f(x_t^{s+1}) - \eta_t\|\nabla f(x_t^{s+1})\|^2 + \tfrac{L\eta_t^2}{2}\|v_t^{s+1}\|^2] \\ &\quad + \mathbb{E}[\mu_{t+1}\eta_t^2\|v_t^{s+1}\|^2 + \mu_{t+1}\|x_t^{s+1} - \tilde{x}^s\|^2] \\ &\quad + 2\mu_{t+1}\eta_t\mathbb{E}\left[\tfrac{1}{2\beta_t}\|\nabla f(x_t^{s+1})\|^2 + \tfrac{1}{2}\beta_t\|x_t^{s+1} - \tilde{x}^s\|^2\right] \\ &\leqslant \mathbb{E}[f(x_t^{s+1}) - \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t}\right)\|\nabla f(x_t^{s+1})\|^2 \\ &\quad + \left(\tfrac{L\eta_t^2}{2} + \mu_{t+1}\eta_t^2\right)\mathbb{E}[\|v_t^{s+1}\|^2] + (\mu_{t+1} + \mu_{t+1}\eta_t\beta_t)\,\mathbb{E}\left[\|x_t^{s+1} - \tilde{x}^s\|^2\right]. \end{aligned} \tag{10.9}$$

To further bound this quantity, we need the bound on the variance of SVRG:

Proposition 10.5 *(Sashank J. Reddi, Hefny, et al., 2016) Let* v_t^{s+1} *be computed by Algorithm 10.2. Then,*

$$\mathbb{E}[\|v_t^{s+1}\|^2] \leqslant 2\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] + 2L^2\mathbb{E}[\|x_t^{s+1} - \tilde{x}^s\|^2].$$

We can use the above Proposition 10.5 to bound $\mathbb{E}[\|v_t^{s+1}\|^2]$, so that upon substituting it in Equation (10.9), we see that

$$\begin{aligned}A_{t+1}^{s+1} &\leqslant \mathbb{E}[f(x_t^{s+1})] - \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t} - \eta_t^2 L - 2\mu_{t+1}\eta_t^2\right)\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \\ &\quad + \left[\mu_{t+1}\left(1+\eta_t\beta_t+2\eta_t^2L^2\right)+\eta_t^2L^3\right]\mathbb{E}\left[\|x_t^{s+1}-\tilde{x}^s\|^2\right] \\ &\leqslant A_t^{s+1} - \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t} - \eta_t^2 L - 2\mu_{t+1}\eta_t^2\right)\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2].\end{aligned}$$

The second inequality follows from the definition of μ_t and A_t^{s+1}. Since $\eta_t = \eta = 1/(4Ln^{2/3})$ for $j > 0$ and $t \in \{0, \ldots, j-1\}$,

$$A_j^{s+1} \leqslant A_0^{s+1} - \upsilon_n \sum_{t=0}^{j-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2], \tag{10.10}$$

where

$$\upsilon_n = \left(\eta_t - \tfrac{\mu_{t+1}\eta_t}{\beta_t} - \eta_t^2 L - 2\mu_{t+1}\eta_t^2\right).$$

We will prove that for the given parameter setting $\upsilon_n > 0$ (see the proof below). With $\upsilon_n > 0$, it is easy to see that $A_j^{s+1} \leqslant A_0^{s+1}$. Furthermore, note that $A_0^{s+1} = \mathbb{E}[f(x_0^{s+1}) + \mu_0\|x_0^{s+1} - \tilde{x}^s\|^2] = \mathbb{E}[f(x_0^{s+1})]$ since $x_0^{s+1} = \tilde{x}^s$ (see Algorithm 10.2). Also, we have

$$\mathbb{E}[f(x_j^{s+1}) + \mu_j\|x_j^{s+1} - \tilde{x}^s\|^2] \leqslant \mathbb{E}[f(x_0^{s+1})]$$

and thus, we obtain $\mathbb{E}[f(x_j^{s+1})] \leqslant \mathbb{E}[f(x_0^{s+1})]$ for all $j \in \{0, \ldots, m\}$. Furthermore, using simple induction and the fact that $x_0^{s+1} = x_m^s$ for all epoch $s \in \{0, \ldots, S-1\}$, it easy to see that $\mathbb{E}[f(x_j^{s+1})] \leqslant f(x^0)$. Therefore, with the definition of y specified in the output of Algorithm 10.2, we see that the condition **G.1** of Gradient-Focused-Optimizer is satisfied for Svrg algorithm.

We now prove that $\upsilon_n > 0$ and also **G.2** of Gradient-Focused-Optimizer is satisifed for Svrg algorithm. By using telescoping the sum with $j = m$ in Equation (10.10), we obtain

$$\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leqslant \frac{A_0^{s+1} - A_m^{s+1}}{\upsilon_n}.$$

This inequality in turn implies that

$$\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leqslant \frac{\mathbb{E}[f(\tilde{x}^s) - f(\tilde{x}^{s+1})]}{\upsilon_n}, \tag{10.11}$$

where we used that $A_m^{s+1} = \mathbb{E}[f(x_m^{s+1})] = \mathbb{E}[f(\tilde{x}^{s+1})]$ (since $\mu_m = 0$), and that $A_0^{s+1} = \mathbb{E}[f(\tilde{x}^s)]$ (since $x_0^{s+1} = \tilde{x}^s$). Now sum over all epochs to obtain

$$\frac{1}{T_g}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1} \mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leqslant \frac{\mathbb{E}[f(x^0) - f(x_m^S)]}{T_g\upsilon_n}. \tag{10.12}$$

Here we used the the fact that $\tilde{x}^0 = x^0$. To obtain a handle on υ_n and complete our analysis, we will require an upper bound on μ_0. We observe that $\mu_0 = \frac{L}{16n^{4/3}}\frac{(1+\theta)^m - 1}{\theta}$ where $\theta = 2\eta^2L^2 + \eta\beta$. This is obtained using the relation $\mu_t = \mu_{t+1}(1 + \eta\beta + 2\eta^2L^2) + \eta^2L^3$ and the fact that $\mu_m = 0$. Using the specified values of β and η we have

$$\theta = 2\eta^2L^2 + \eta\beta = \frac{1}{8n^{4/3}} + \frac{1}{4n} \leqslant \frac{3}{4n}.$$

Using the above bound on θ, we get

$$\mu_0 = \frac{L}{16n^{4/3}}\frac{(1+\theta)^m - 1}{\theta} = \frac{L((1+\theta)^m - 1)}{2(1+2n^{1/3})}$$
$$\leqslant \frac{L((1+\frac{3}{4n})^{\lfloor 4n/3 \rfloor} - 1)}{2(1+2n^{1/3})} \leqslant n^{-1/3}(L(e-1)/4), \tag{10.13}$$

wherein the second inequality follows upon noting that $(1+\frac{1}{l})^l$ is increasing for $l > 0$ and $\lim_{l\to\infty}(1+\frac{1}{l})^l = e$ (here e is the Euler's number). Now we can lower bound v_n, as

$$v_n = \min_t\left(\eta - \tfrac{\mu_{t+1}\eta}{\beta} - \eta^2 L - 2\mu_{t+1}\eta^2\right) \geqslant \left(\eta - \tfrac{\mu_0\eta}{\beta} - \eta^2 L - 2\mu_0\eta^2\right) \geqslant \frac{1}{40Ln^{2/3}}.$$

The first inequality holds since μ_t decreases with t. The second inequality holds since (a) μ_0/β can be upper bounded by $(e-1)/4$ (follows from Equation (10.13)), (b) $\eta^2 L \leqslant \eta/4$ and (c) $2\mu_0\eta^2 \leqslant (e-1)\eta/8$ (follows from Equation (10.13)). Substituting the above lower bound in Equation (10.12), we obtain the following:

$$\frac{1}{T_g}\sum_{s=0}^{S-1}\sum_{t=0}^{m-1}\mathbb{E}[\|\nabla f(x_t^{s+1})\|^2] \leqslant \frac{40Ln^{2/3}\mathbb{E}[f(x^0) - f(x_m^S)]}{T_g}. \tag{10.14}$$

From the definition of (y, z) in output of Algorithm 10.2 *i.e.*, y is Iterate x_a chosen uniformly random from $\{\{x_t^{s+1}\}_{t=0}^{m-1}\}_{s=0}^{S-1}$ and $z = x_m^S$, it is clear that Algorithm 10.2 satisfies the **G.2** requirement of Gradient-Focused-Optimizer with $g(n,\epsilon) = T_\epsilon/40Ln^{2/3}$. Since both **G.1** and **G.2** are satisfied for Algorithm 10.2, we conclude that Svrg is a Gradient-Focused-Optimizer. ■

In rest of this section, we discuss approaches using Svrg as a Gradient-Focused-Optimizer. In particular, we propose and provide convergence analysis for two different methods with different Hessian-Focused-Optimizer but which use Svrg as a Gradient-Focused-Optimizer.

10.4.1 *Hessian descent*

The first approach is based on directly using the eigenvector corresponding to the smallest eigenvalue as a Hessian-Focused-Optimizer. More specifically, when the smallest eigenvalue of the Hessian is negative and reasonably large in magnitude, the Hessian information can be used to ensure descent in the objective function value. The pseudo-code for the algorithm is given in Algorithm 10.3.

The key idea is to utilize the minimum eigenvalue information in order to make a descent step. If $\lambda_{\min}(\nabla^2 f(x)) \leqslant -\gamma$ then the idea is to use this information to take a descent step. Note the subroutine is designed in a fashion such that the objective function value never increases. Thus, it naturally satisfies the requirement **H.1** of Hessian-Focused-Optimizer. The following result shows that HessianDescent is a Hessian-Focused-Optimizer.

Lemma 10.6 HessianDescent *is a* Hessian-Focused-Optimizer *with* $h(n,\epsilon,\gamma) = \frac{\rho}{24M^2}\gamma^3$.

Proof The first important observation is that the function value never increases because $y = \arg\min_{z\in\{u,x\}} f(z)$ *i.e.*, $f(y) \leqslant f(x)$, thus satisfying **H.1** of Hessian-Focused-Optimizer. We now analyze the scenario where $\lambda_{min}(\nabla^2 f(x)) \leqslant -\gamma$. Consider the event where we obtain v such that

$$\langle v, \nabla^2 f(x) v\rangle \leqslant \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2}.$$

This event (denoted by $\mathcal{E}$) happens with at least probability ρ. Note that, since $\lambda_{min}(\nabla^2 f(x)) \leqslant -\gamma$, we have $\langle v, \nabla^2 f(x) v\rangle \leqslant -\frac{\gamma}{2}$. In this case, we have the following relationship:

$$\begin{aligned} f(y) &\leqslant f(x) + \langle \nabla f(x), y - x\rangle + \frac{1}{2}(y-x)^T \nabla^2 f(x)(y-x) + \frac{M}{6}\|y-x\|^3 \\ &= f(x) - \alpha|\langle \nabla f(x), v\rangle| + \frac{\alpha^2}{2} v^T \nabla^2 f(x) v + \frac{M\alpha^3}{6}\|v\|^3 \\ &\leqslant f(x) + \frac{\alpha^2}{2} v^T \nabla^2 f(x) v + \frac{M\alpha^3}{6} \\ &\leqslant f(x) - \frac{1}{2M^2}|v^T \nabla^2 f(x) v|^3 + \frac{1}{6M^2}|v^T \nabla^2 f(x) v|^3 \\ &= f(x) - \frac{1}{3M^2}|v^T \nabla^2 f(x) v|^3 \leqslant f(x) - \frac{1}{24M^2}\gamma^3. \end{aligned} \tag{10.15}$$

The first inequality follows from the M-lipschitz continuity of the Hessain $\nabla^2 f(x)$. The first equality follows from the update rule of HessianDescent. The second inequality is obtained by dropping the negative term and using the fact that $\|v\| = 1$. The second equality is obtained by substituting $\alpha = \frac{|v^T \nabla^2 f(x) v|}{M}$. The last inequality is due to the fact that$\langle v, \nabla^2 f(x) v\rangle \leqslant -\frac{\gamma}{2}$. In the other scenario where

$$\langle v, \nabla^2 f(x) v\rangle \leqslant \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2},$$

we can at least ensure that $f(y) \leqslant f(x)$ since $y = \arg\min_{z \in \{u,x\}} f(z)$. Therefore, we have

$$\begin{aligned} \mathbb{E}[f(y)] &= \rho \mathbb{E}[f(y)|\mathcal{E}] + (1-\rho)\mathbb{E}[f(y)|\bar{\mathcal{E}}] \\ &\leqslant \rho \mathbb{E}[f(y)|\mathcal{E}] + (1-\rho) f(x) \\ &\leqslant \rho \left[f(x) - \tfrac{\rho}{24M^2}\gamma^3\right] + (1-\rho) f(x) \\ &= f(x) - \tfrac{\rho}{24M^2}\gamma^3. \end{aligned} \tag{10.16}$$

The last inequality is due to Equation (10.15). Hence, Hessian-Focused-Optimizer satisfies **H.2** of Hessian-Focused-Optimizer with $h(n, \epsilon, \gamma) = \frac{\rho}{24M^2}\gamma^3$, thus concluding the proof. ■

With Svrg as Gradient-Focused-Optimizer and HessianDescent as Hessian-Focused-Optimizer, we show the following key result:

Theorem 10.7 *Suppose* Svrg *with* $m = n$, $\eta_t = \eta = 1/4Ln^{2/3}$ *for all* $t \in \{1, ..., m\}$ *and* $T_g = 40Ln^{2/3}/\epsilon^{1/2}$ *is used as* Gradient-Focused-Optimizer *and* HessianDescent *is used as* Hessian-Focused-Optimizer *with* $q = 0$*, then Algorithm 10.1 finds a* $(\epsilon, \sqrt{\epsilon})$*-second order critical point in* $T = O(\Delta/\min(p, 1-p)\epsilon^{3/2})$ *with probability at least* 0.9.

The result directly follows from using Lemma 10.4 and Lemma 10.6 in Theorem 10.3. The result shows that the iteration complexity of Algorithm 10.1 in this case is $O(\Delta/\epsilon^{3/2}\min(p, 1-p))$. Thus, the overall IFO complexity of Svrg algorithm is $(n + T_g) \times T = O(n/\epsilon^{3/2} + n^{2/3}/\epsilon^2)$. Since each IFO call takes $O(d)$ time, the overall time complexity of all Gradient-Focused-Optimizer steps is $O(nd/\epsilon^{3/2} + n^{2/3}d/\epsilon^2)$. To understand the time complexity of HessianDescent, we need the following result due to N. Agarwal, Allen Zhu, et al. (2016).

Proposition 10.8 *(N. Agarwal, Allen Zhu, et al., 2016) The time complexity of finding* $v \in \mathbb{R}^d$ *that* $\|v\| = 1$*, and with probability at least* ρ *the following inequality holds:* $\langle v, \nabla^2 f(x) v\rangle \leqslant \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2}$ *is* $O(nd + n^{3/4}d/\gamma^{1/2})$.

Algorithm 10.3 HessianDescent (x, ϵ, γ)

1: Find v such that $\|v\| = 1$, and with probability at least ρ the following inequality holds: $\langle v, \nabla^2 f(x)v\rangle \leqslant \lambda_{min}(\nabla^2 f(x)) + \frac{\gamma}{2}$.
2: Set $\alpha = |\langle v, \nabla^2 f(x)v\rangle|/M$.
3: $u = x - \alpha \operatorname{sign}(\langle v, \nabla f(x)\rangle)v$.
4: $y = \arg\min_{z\in\{u,x\}} f(z)$
5: **Output:** $(y, \diamond)$.

Note that each iteration of Algorithm 10.1 in this case has just linear dependence on d. Since the total number of HessianDescent iterations is $O(\Delta/\min(p, 1-p)\epsilon^{3/2})$ and each iteration has the complexity of $O(nd + n^{3/4}d/\epsilon^{1/4})$, using the above remark, we obtain an overall time complexity of HessianDescent is $O(nd/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4})$. Combining this with the time complexity of Svrg, we get the following result.

Corollary 10.9 *The overall running time of Algorithm 10.1 to find a $(\epsilon, \sqrt{\epsilon})$-second order critical point, with parameter settings used in Algorithm 10.7, is* $O(nd/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4} + n^{2/3}d/\epsilon^2)$.

Note that the dependence on ϵ is much better in comparison to that of Noisy SGD used in Ge et al. (2015). Furthermore, our results are competitive with N. Agarwal, Allen Zhu, et al. (2016) and Carmon et al. (2018) in their respective settings, but with a much more practical algorithm and simpler analysis. More specifically, N. Agarwal, Allen Zhu, et al. (2016) has running time of $O(nd/\epsilon^{3/2} + n^{3/4}d/\epsilon^{7/4})$. When $\epsilon > c/n^{2/3}$ where c is some constant (which is very reasonable for most machine learning settings), our results would be better. However, our rates can be slightly worse when this is not the case. Next, Carmon et al. (2018) does not consider the finite-sum setting specifically, and if applied as it is for our finite-sum setup, it will have a running time of $O(nd/\epsilon^{7/4})$. In comparison, our rates are better in a larger regime where $\epsilon > c/n^{4/3}$. Note that in machine learning applications the goal is to obtain better generalization which roughly translates to such regimes in practice. Thus, in settings of interest to machine learning, our results are competitive (or better) in comparison to N. Agarwal, Allen Zhu, et al. (2016) and Carmon et al. (2018). Finally, we also note that our algorithm is faster than the one proposed in C. Jin et al. (2017), which has a time complexity of $O(nd/\epsilon^2)$.

10.4.2 *Cubic Descent*

In this section, we show that the cubic regularization method in Nesterov and Polyak (2006) can be used as Hessian-Focused-Optimizer. More specifically, here Hessian-Focused-Optimizer approximately solves the following optimization problem:

$$y = \arg\min_z \langle \nabla f(x), z - x\rangle + \frac{1}{2}\left\langle z - x, \nabla^2 f(x)(z-x)\right\rangle + \frac{M}{6}\|z-x\|^3, \qquad \text{(CubicDescent)}$$

and returns $(y, \diamond)$ as output. The following result can be proved for this approach.

Theorem 10.10 *Suppose* Svrg *(same as Theorem 10.7) is used as* Gradient-Focused-Optimizer *and* CubicDescent *is used as* Hessian-Focused-Optimizer *with $q = 0$, then Algorithm 10.1 finds a $(\epsilon, \sqrt{\epsilon})$-second order critical point in $T = O(\Delta/\min(p, 1-p)\epsilon^{3/2})$ with probability at least 0.9.*

Proof First note that cubic method is a descent method (refer to Theorem 1 of Nesterov and Polyak (2006)); thus, **H.1** is trivially satisfied. Furthermore, cubic descent is a HESSIAN-FOCUSED-OPTIMIZER with $h(n, \epsilon, \gamma) = \frac{2\gamma^3}{81M^3}\gamma^3$. This, again, follows from Theorem 1 of Nesterov and Polyak (2006). The result easily follows from the aforementioned observations. ■

In principle, Algorithm 10.1 with CUBICDESCENT as HESSIAN-FOCUSED-OPTIMIZER can converge without the use of GRADIENT-FOCUSED-OPTIMIZER subroutine at each iteration since it essentially reduces to the cubic regularization method of Nesterov and Polyak (2006). However, in practice, we would expect GRADIENT-FOCUSED-OPTIMIZER to perform most of the optimization and HESSIAN-FOCUSED-OPTIMIZER to be used for far fewer iterations. Using the method developed in Nesterov and Polyak (2006) for solving CUBICDESCENT, we obtain the following corollary.

Corollary 10.11 *The overall running time of Algorithm 10.1 to find a $(\epsilon, \sqrt{\epsilon})$-second order critical point, with parameter settings used in Theorem 10.10, is $O(nd^\omega/\epsilon^{3/2} + n^{2/3}d/\epsilon^2)$.*

Here ω is the matrix multiplication constant. The dependence on ϵ is weaker in comparison to Corollary 10.9. However, each iteration of CUBICDESCENT is expensive (as seen from the factor d^ω in the corollary above) and thus, in high dimensional settings typically encountered in machine learning, this approach can be expensive in comparison to HESSIANDESCENT.

10.4.3 *Practical Considerations*

The focus of this section was to demonstrate the wide applicability of our framework; wherein using a simple instantiation of this framework, we could achieve algorithms with fast convergence rates. To further achieve good empirical performance, we had to slightly modify these procedures. For HESSIAN-FOCUSED-OPTIMIZER, we found stochastic, adaptive and inexact approaches for solving HESSIANDESCENT and CUBICDESCENT work well in practice. We begin by describing an inexact approach fo CUBICDESCENT which we will use for our experiments as well.

CUBICDESCENT method of Nesterov and Polyak (2006), which is designed to operate on full batch, *i.e.*, it does not exploit the finite-sum structure of the problem and requires the computation of the gradient and the Hessian on the entire dataset to make an update. However, such full-batch methods do not scale gracefully with the size of data and become prohibitively expensive on large datasets. To overcome this challenge, we devised an approximate cubic regularization method described below:

1. Pick a mini-batch $\mathcal{B}$ and obtain the gradient and the hessian based on $\mathcal{B}$, *i.e.*,

$$g = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla f_i(x) \qquad H = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \nabla^2 f_i(x) \tag{10.17}$$

2. Solve the sub-problem

$$v^* = \arg\min_v \langle g, v\rangle + \frac{1}{2}\langle v, Hv\rangle + \frac{M}{6}\|v\|^3 \tag{10.18}$$

3. Update: $x \leftarrow x + v^*$

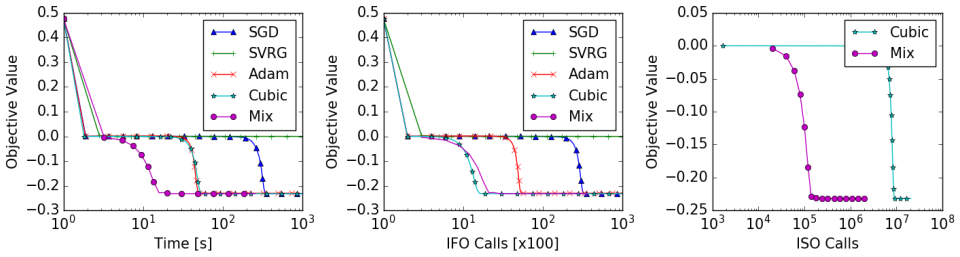


Figure 10.2: Comparison of various methods on a synthetic problem. Our mix framework successfully escapes saddle point and uses relatively few ISO calls in comparison to CubicDescent.

We found that this mini-batch training strategy, which requires the computation of the gradient and the Hessian on a small subset of the dataset, to work well on a few datasets (CURVES, MNIST, CIFAR10). A similar method has been analysed in Cartis and Scheinberg (2017).

Furthermore, in many deep-networks, adaptive per-parameter learning rate helps immensely (D. P. Kingma and Ba, 2014). One possible explanation for this is that the scale of the gradients in each layer of the network often differ by several orders of magnitude. A well-suited optimization method should take this into account. This is the reason for popularity of methods like Adam or RMSprop in the deep learning community. On similar lines, to account for different per-parameter behaviour in cubic regularization, we modify the sub-problem by adding a diagonal matrix M_d in addition to the scalar regularization coefficient M, *i.e.*,

$$\min_v \langle g, v\rangle + \frac{1}{2}\langle v, Hv\rangle + \frac{1}{6}M\|M_d v\|^3. \tag{10.19}$$

Also we devised an adaptive rule to obtain the diagonal matrix as $M_d = \mathrm{diag}((s+10^{-12})^{1/9})$, where s is maintained as a moving average of third order polynomial of the mini-batch gradient g, in a fashion similar to RMSprop and Adam:

$$s \leftarrow \beta s + (1-\beta)(|g|^3 + 2g^2), \tag{10.20}$$

where $|g|^3$ and g^2 are vectors such that $[|g|^3]_i = |g_i|^3$ and $[g^2]_i = g_i^2$ respectively for all $i \in [n]$. The experiments reported on CURVES and MNIST in this paper utilizes both the above modifications to the cubic regularization, with β set to 0.9. We refer to this modified procedure as ACubic in our results.

Finally, in the context of deep learning, empirical evidence suggests that first-order methods like Adam (D. P. Kingma and Ba, 2014) exhibit behavior that is in congruence with properties **G.1** and **G.2**. While theoretical analysis for a setting where Adam is used as Gradient-Focused-Optimizer is still unresolved, we nevertheless demonstrate its performance through empirical results in the following section.

10.5 EXPERIMENTS

We now present empirical results for our saddle point avoidance technique with an aim to highlight three aspects of our framework: (i) it successfully escapes non-degenerate saddle points, (ii) it is fast, and (iii) it is practical on large-scale problems. All the algorithms are implemented on TensorFlow (Martin Abadi et al., 2015). In case of deep networks, the Hessian-vector product is evaluated using the trick presented in Pearlmutter (1994). We run our experiments on a commodity machine with Intel® Xeon® CPU E5-2630 v4 CPU, 256GB RAM, and NVidia® Titan X (Pascal) GPU.

10.5.1 *Synthetic Problem*

To demonstrate the fast escape from a saddle point of our method, we consider the following simple nonconvex finite-sum problem:

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} x^T A_i x + b_i^T x + \|x\|_{10}^{10} \tag{10.21}$$

Here the parameters are designed such that $\sum_i b_i = 0$ and $\sum_i A_i$ matrix has exactly one negative eigenvalue of -0.001 and other eigenvalues randomly chosen in the interval $[1, 2]$. The total number of examples n is set to be 100,000 and d is 1000. It is not hard to see that this problem has a non-degenerate saddle point at the origin. This allows us to explore the behaviour of different optimization algorithms in the vicinity of the saddle point. In this experiment, we compare a mix of SVRG and HESSIANDESCENT (as in Theorem 10.7) with SGD (with constant step size), ADAM, SVRG and CUBICDESCENT. The parameter of these algorithms is chosen by grid search so that it gives the best performance. The subproblem of CUBICDESCENT was solved with gradient descent (Carmon et al., 2018) until the gradient norm of the subproblem is reduced below 10^{-3}. We study the progress of optimization, *i.e.*, decrease in function value with wall clock time, IFO calls, and ISO calls. All algorithms were initialized with the same starting point very close to origin.

The parameter selection for all the methods were carried as follows:

1. SGD: The scalar step-size was determined by a grid search.
2. ADAM: We performed a grid search over α and ε parameters of ADAM tied together, *i.e.*, $\alpha = \varepsilon$.
3. SVRG: The scalar step-size was determined by a grid search.
4. CUBICDESCENT: The regularization parameter M was chosen by grid search. The subproblem was solved with gradient descent (Carmon et al., 2018) with the step-size of solver to be 10^{-2} and run till the gradient norm of the sub-problem is reduced below 10^{-3}.

The results are presented in Figure 10.2, which shows that our proposed mix framework was the *fastest* to escape the saddle point in terms of wall clock time. We observe that performance of the first order methods suffered severely due to the saddle point. Note that SGD eventually escaped the saddle point due to inherent noise in the mini-batch gradient. The other first order methods like ADAM with higher noise could escape relatively faster whereas SVRG with reduced noise stayed stuck at the saddle point. CUBICDESCENT, a second-order method, escaped the saddle point faster in terms of iterations using the Hessian information. But operating on Hessian information is expensive as a result this method was slow in terms of wall clock time. The proposed framework, which is a mix of the two strategies, inherits the best of both worlds by using cheap gradient information most of the time and reducing the use of relatively expensive Hessian information (ISO calls) by 100x. This resulted in *faster* escape from saddle point in terms of wall clock time.

10.5.2 *Deep Networks*

To investigate the practical performance of the framework for deep learning problems, we applied it to two deep autoencoder optimization problems from Geoffrey E Hinton and R. R. Salakhutdinov (2006) called "CURVES" and "MNIST". Due to their high difficulty, performance on these problems has become a standard benchmark for neural network optimization methods, *c.f.* Martens (2010), Martens and Grosse (2015), Sutskever et al. (2013), and Vinyals and Povey (2012). The "CURVES" autoencoder consists of an encoder with layers of size (28×28)-400-200-100- 50-25-6 and a symmetric decoder totaling in 0.85M parameters. The six units in the code

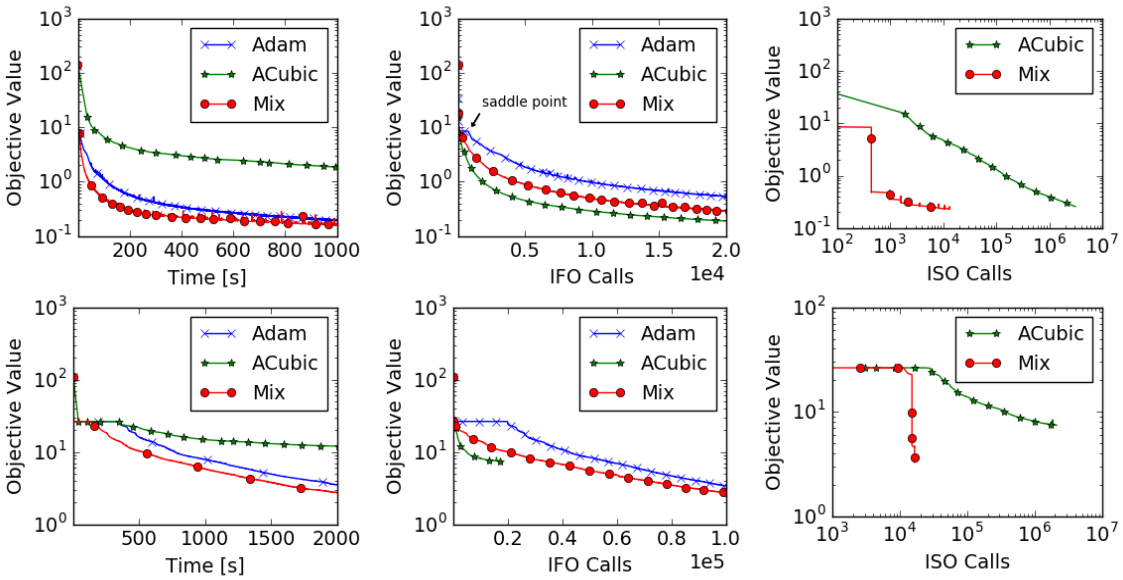


Figure 10.3: Comparison of various methods on a Deep Autoencoder on CURVES (top) and MNIST (bottom). Our mix approach converges faster than the baseline methods and uses relatively few ISO calls in comparison to ApproxCubicDescent

layer were linear and all the other units were logistic. The network was trained on 20,000 images and tested on 10,000 new images. The data set contains images of curves that were generated from three randomly chosen points in two dimensions. [1] The "MNIST" autoencoder consists of an encoder with layers of size (28×28)-1000-500-250-30 and a symmetric decoder, totaling in 2.8M parameters. The thirty units in the code layer were linear and all the other units were logistic. The network was trained on 60,000 images and tested on 10,000 new images. The data set contains images of handwritten digits 0–9. The pixel intensities were normalized to lie between 0 and 1.[2]

As an instantiation of our framework, we use a mix of Adam, which is popular in deep learning community, and approximate cubic regularization method described in Section 10.4.3. For comparison the algorithm in Carmon et al. (2018) is impractical for our setting because it requires computing the full gradient at each iteration and is, thus, intractable for problems of our interest. Furthermore, we tried to compare our algorithm with the method in N. Agarwal, Allen Zhu, et al. (2016), however, the algorithm as stated in the paper is not practical[3]. To this end, we compare our algorithm with a practical version of cubic method, ApproxCubicDescent, described in Section 10.4.3. We also compared our algorithm with Adam, which is typically the de-facto method for training large deep networks. The parameters of these algorithms were chosen to produce the best generalization on a held out test set. The regularization parameter M was chosen as the smallest value such that the function value does not fluctuate in the first 10 epochs. We use the initialization suggested in Martens (2010) and a mini-batch size of 1000 for all the algorithms. We report objective function value against wall clock time and ISO calls.

The parameter selection for all the methods were carried as follows:

1 Data available at: www.cs.toronto.edu/~jmartens/digs3pts_1.mat
2 Data available at: www.cs.toronto.edu/~jmartens/mnist_all.mat
3 We would like to emphasize that N. Agarwal, Bullins, and Hazan (2016) does not contain any empirical results. In order to be fair, we also reached out to the authors of the paper, who via personal communication admitted that the algorithm as implemented verbatim in the paper is not practical and comparison to it is not useful.

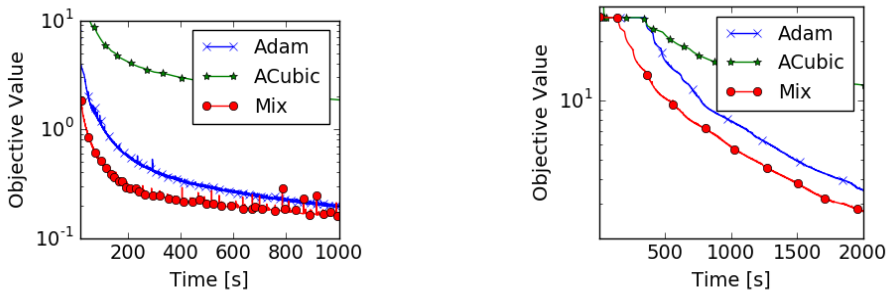


Figure 10.4: Zoomed in version of plots with respect to time. Here we show progress from time=10s onwards. This better exhibits the relative differences between the methods, and is illustrative of the advantage of our method.

1. ADAM: We performed a grid search over α and ε parameters of ADAM so as to produce the best generalization on a held out test set. We found it to be $\alpha = 10^{-3}, \varepsilon = 10^{-3}$ for CURVES and $\alpha = 10^{-2}, \varepsilon = 10^{-1}$ for MNIST.
2. APPROXCUBICDESCENT: The regularization parameter M was chosen as the largest value such function value does not jump in first 10 epochs. We found it to be $M = 10^3$ for both CURVES and MNIST. The sub-problem was solved with gradient descent (Carmon et al., 2018) with the step-size of solver to be 10^{-3} and run till the gradient norm of the sub-problem is reduced below 0.1.

The results presented in Figure 10.3 show that our proposed mix framework was the *fastest* to escape the saddle point in terms of wall clock time. ADAM took considerably more time to escape the saddle point, especially in the case of MNIST. While APPROXCUBICDESCENT escaped the saddle point in relatively fewer iterations, each iteration required considerably large number of ISO calls; as a result, the method was extremely slow in terms of wall clock time, despite our efforts to improve it via approximations and code optimizations. On the other hand, our proposed framework seamlessly balances these two methods, resulting in fast decrease of training loss.

10.6 POINTS TO PONDER

In this chapter, we examined a generic strategy to escape saddle points in nonconvex finite-sum problems and presented its convergence analysis. The key intuition is to alternate between a first-order and second-order based optimizers; the latter is mainly intended to escape points that are only stationary but are not second-order critical points. We presented two different instantiations of our framework and provided their detailed convergence analysis. In this paper, we primarily used SVRG as GRADIENT-FOCUSED-OPTIMIZER, however, investigating the use of other first-order methods, like Carmon et al. (2017), is an interesting research direction. Also, while both our methods explicity use the Hessian information, one can also use noisy first-order methods as HESSIAN-FOCUSED-OPTIMIZER (*c.f.* noisy SGD in Ge et al. (2015)). In such a scenario, we exploit the negative eigenvalues of the Hessian to escape saddle points by using isotropic noise, and do not explicitly use ISO. For these methods, under strict-saddle point property (Ge et al., 2015), we can show convergence to local optima within our framework.

Our primary goal in this paper was to develop a well-founded, yet practical, framework for finding local minima in nonconvex optimization. While convergence rates in N. Agarwal, Allen Zhu, et al. (2016) may seem slightly better at first glance, we would like to point out that these

running times are worst-case time complexity and might involve large constants. For example, N. Agarwal, Allen Zhu, et al. (2016) solve a subroutine based on both gradient and Hessian information at *each* iteration. Such methods do not scale well in practice because they rely on expensive computations based on the Hessian at every iteration.

We primarily focused on obtaining second-order critical points for nonconvex finite-sums (10.1). This does not necessarily imply low test error or good generalization capabilities. Thus, we should be careful when interpreting the results presented in this paper. A detailed discussion or analysis of these issues is out of scope of this paper. While a few prior works argue for convergence to local optima, the exact connection between generalization and local optima is not well understood, and is an interesting open problem. Nevertheless, we believe the techniques presented in this paper can be used alongside other optimization tools for faster and better nonconvex optimization.

11

Conclusion

Machine learning techniques are reaching or exceeding human level performances in tasks involving simple data like image classification, translation, and text-to-speech. The success of these machine learning algorithms is primarily attributed to highly versatile representations learnt from data using deep networks or intricately designed Bayesian models. Despite these instances of success, progress has been limited to simple data-types so far.

Most real-world data come in different forms, encapsulating different kinds of information, not limited to images or text, but also as point clouds, sets, graphs, compressed or even heterogeneous combinations thereof. In this thesis, we focus on different aspect of representation learning, namely, enhancing versatility, interpretability and bridging the gap between the statistical and computational perspective.

We begin by showcasing ways to leverage structure in data, like invariances or heterogeneity in order to establish new mathematical properties. Incorporating these mathematical properties in the model and learning algorithms leads to versatile representations on diverse domains.

Having representations is not enough in various applications - its interpretability is as crucial as its accuracy. Deep models often yield better accuracy but require a large number of parameters, often in contrast to the simplicity of the underlying data, rendering it uninterpretable. This is highly undesirable in tasks like user modeling. In this thesis, we show that by leveraging structure by incorporating domain knowledge in the form of Bayesian components on top of deep models, we learn sparser representations with discrete components that are more amenable to human interpretation.

Finally, inferring interpretable representations from large-scale data is desirable, but often hindered by a mismatch between computational resources and statistical models. In this thesis, we bridge this gap by again leveraging structure, albeit of a different kind. Our solutions are based on a combination of modern computational techniques/data structures on one side and modified statistical inference algorithms on the other which exploit topological properties of the training objective. This introduces new ways to parallelize, reduce look-ups, handle variable state space size, and escape saddle points.

Publications

Das, Rajarshi, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum (2018). "Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning." In: *Proceedings of the International Conference on Representation Learning*.

Gu, Chenjie, Manzil Zaheer, and Xin Li (2014). "Multiple-Population Moment Estimation: Exploiting Interpopulation Correlation for Efficient Moment Estimation in Analog/Mixed-Signal Validation." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.7, pp. 961–974.

Khetan, Abhishek, Manzil Zaheer, Venkatasubramanian Viswanathan, and Barnabás Póczos (2018). "Learning Density Functional Theory Solely From Molecular Geometry." In: *arXiv preprint*.

Oliva, Junier B, Avinava Dubey, Barnabás Póczos, Jeff Schneider, and Eric P Xing (2018). "Transformation Autoregressive Networks." In: *Proceedings of the International Conference on Machine Learning*.

Sachan, Devendra Singh, Manzil Zaheer, and Ruslan Salakhutdinov (2018). "Investigating the Working of Text Classifiers." In: *Proceedings of the 28th International Conference on Computational Linguistics: Technical Papers*.

Sachan, Devendra Singh, Manzil Zaheer, and Ruslan Salakhutdinov (2019). "Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function." In:

Singh, Shashank, Ananya Uppal, Boyue Li, Chun-Liang Li, Manzil Zaheer, and Barnabás Póczos (2018). "Nonparametric Density Estimation under Adversarial Losses." In: *arXiv preprint arXiv:1805.08836*.

Sun, Haitian, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen (2018). "Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4231–4242.

Tung, Hsiao-Yu Fish, Chao-Yuan Wu, Manzil Zaheer, and Alexander J Smola (2017). "Spectral Methods for Nonparametric Models." In: *arXiv preprint arXiv:1704.00003*.

Wang, Fa, Manzil Zaheer, Xin Li, Jean-Olivier Plouchart, and Alberto Valdes-Garcia (2015). "Co-learning Bayesian model fusion: efficient performance modeling of analog and mixed-signal circuits using side information." In: *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, pp. 575–582.

Wu, Chao-Yuan, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl (2018). "Compressed video action recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6026–6035.

Zaheer, Manzil, Amr Ahmed, and Alexander J Smola (2017). "Latent LSTM Allocation: Joint Clustering and Non-Linear Dynamic Modeling of Sequence Data." In: *Proceedings of the International Conference on Machine Learning*, pp. 3967–3976.

Zaheer, Manzil, Amr Ahmed, Yuan Wang, Daniel Silva, and Yuchen Wu (2018). "Uncovering Hidden Structure in Sequence Data via Threading Recurrent Models." In: *arXiv preprint*.

Zaheer, Manzil, Rajarshi Das, and Chris Dyer (2015). "Gaussian lda for topic models with word embeddings." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational*

Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 795–804.

Zaheer, Manzil, Rajarshi Das, Siva Reddy, and Andrew McCallum (2017). "Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 358–365.

Zaheer, Manzil, Guru Guruganesh, Golan Levin, and Alexander Smola (2018). "TerraPattern: A Nearest Neighbor Service." In: *arXiv preprint*.

Zaheer, Manzil, Satwik Kottur, Amr Ahmed, José Moura, and Alex Smola (2017). "Canopy Fast Sampling with Cover Trees." In: *Proceedings of the International Conference on Machine Learning*, pp. 3977–3986.

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola (2017). "Deep Sets." In: *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3391–3401.

Zaheer, Manzil, Jay Lee, Stephan Günnemann, and Alex Smola (2015). "Preferential Attachment in Graphs with Affinities." In: *Proceedings of the Artificial Intelligence and Statistics*, pp. 571–580.

Zaheer, Manzil, Chun-Liang Li, Yang Zhang, Ruslan Salakhutdinov, and Barnabás Póczos (2018). "Point Cloud GAN." In: *arXiv preprint*.

Zaheer, Manzil, Xin Li, and Chenjie Gu (2014). "MPME-DP: Multi-population moment estimation via dirichlet process for efficient validation of analog/mixed-signal circuits." In: *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, pp. 316–323.

Zaheer, Manzil, Sashank Reddi, Suvrit Sra, Barnabas Poczos, Francis Bach, Ruslan Salakhutdinov, and Alex Smola (2018). "A Generic Approach for Escaping Saddle points." In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1233–1242.

Zaheer, Manzil and Alexander J Smola (2018). "Fast Sampling Algorithms for Sparse Latent Variable Models." In: *arXiv preprint*.

Zaheer, Manzil, Jean-Baptiste Tristan, Michael L Wick, and Guy L Steele Jr (2016). "Learning a Static Analyzer: A Case Study on a Toy Language." In:

Zaheer, Manzil, Fa Wang, Chenjie Gu, and Xin Li (2015). "mTunes: Efficient post-silicon tuning of mixed-signal/RF integrated circuits based on Markov decision process." In: *Proceedings of the 52nd Annual Design Automation Conference*. ACM, pp. 170–176.

Zaheer, Manzil, Michael Wick, Jean-Baptiste Tristan, Alex Smola, and Guy L Steele Jr (2015). "Exponential stochastic cellular automata for massively parallel inference." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics 2016*, pp. 966–975.

Zheng, Xun, Manzil Zaheer, Amr Ahmed, Yuan Wang, Eric P Xing, and Alexander J Smola (2017). "State space LSTM models with particle MCMC inference." In: *arxiv preprint arxiv:1711.11179*.

Bibliography

Abu-El-Haija, Sami, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan (2016). "YouTube-8M: A large-scale video classification benchmark." In: *arXiv preprint arXiv:1609.08675*.

Achlioptas, Panos, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas (2017). "Learning Representations and Generative Models for 3D Point Clouds." In: *arXiv preprint arXiv:1707.02392*.

Adams, R., Z. Ghahramani, and M. Jordan (2010). "Tree-Structured Stick Breaking for Hierarchical Data." In: *Neural Information Processing Systems*, pp. 19–27.

Agarwal, Alekh and Leon Bottou (2015). "A Lower Bound for the Optimization of Finite Sums." In: *International Conference on Machine Learning*, pp. 78–86.

Agarwal, Naman, Zeyuan Allen Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma (2016). "Finding Approximate Local Minima for Nonconvex Optimization in Linear Time." In: *Arxiv* abs/1611.01146.

Agarwal, Naman, Brian Bullins, and Elad Hazan (2016). "Second Order Stochastic Optimization in Linear Time." In: *Arxiv* abs/1602.03943.

Agirre, Eneko, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa (2009). "A study on similarity and relatedness using distributional and WordNet-based approaches." In: *Proceedings of NAACL*.

Ahmed, A., L. Hong, and A.J. Smola (2013). "Nested Chinese Restaurant Franchise Processes: Applications to User Tracking and Document Modeling." In: *ICML*. Atlanta, GA.

Ahmed, A., S. Ravi, S. Narayanamurthy, and A.J. Smola (2012). "FastEx: Hash Clustering with Exponential Families." In: *Neural Information Processing Systems 25*, pp. 2807–2815.

Ahmed, Amr and Eric Xing (2008). "Dynamic non-parametric mixture models and the recurrent chinese restaurant process: with applications to evolutionary clustering." In: *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, pp. 219–230.

Aly, M., A. Hatch, V. Josifovski, and V.K. Narayanan (2012). "Web-scale user modeling for targeting." In: *Conference on World Wide Web*. ACM, pp. 3–12.

Anandkumar, Animashree, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky (2014). "Tensor decompositions for learning latent variable models." In: *The Journal of Machine Learning Research* 15.1, pp. 2773–2832.

Andreas, Jacob, Marcus Rohrbach, Trevor Darrell, and Dan Klein (2016). "Learning to Compose Neural Networks for Question Answering." In: *NAACL*.

Andrews, Mark and Gabriella Vigliocco (2010). "The hidden Markov topic model: A probabilistic model of semantic representation." In: *Topics in Cognitive Science* 2.1, pp. 101–113.

Andrieu, Christophe, Arnaud Doucet, and Roman Holenstein (2010). "Particle Markov chain Monte Carlo methods." In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

Antoniak, C. (1974). "Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems." In: *Annals of Statistics* 2, pp. 1152–1174.

Archer, Evan, Il Memming Park, Lars Buesing, John P. Cunningham, and Liam Paninski (2015). "Black box variational inference for state space models." In: *arXiv preprint arXiv:1511.07367*.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017). "Wasserstein GAN." In: *ICML*.

Arthur, David and Sergei Vassilvitskii (2007). "k-means++: the advantages of careful seeding." In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pp. 1027–1035. URL: http://dl.acm.org/citation.cfm?id=1283383.1283494.

Asuncion, Arthur, Max Welling, Padhraic Smyth, and Yee Whye Teh (2009). "On Smoothing and Inference for Topic Models." In: *Proc. Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. UAI '09. Montreal, Quebec, Canada: AUAI Press, pp. 27–34. ISBN: 978-0-9749039-5-8.

Atwood, James and Don Towsley (2016). "Diffusion-convolutional neural networks." In: *Advances in Neural Information Processing Systems*, pp. 1993–2001.

Auvolat, Alex, Sarath Chandar, Pascal Vincent, Hugo Larochelle, and Yoshua Bengio (2015). "Clustering is efficient for approximate maximum inner product search." In: *arXiv preprint arXiv:1507.05910*.

Bachem, Olivier, Mario Lucic, S. Hamed Hassani, and Andreas Krause (2016). "Approximate K-Means++ in Sublinear Time." In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Pp. 1459–1467. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12147.

Bachrach, Yoram, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet (2014). "Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces." In: *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, pp. 257–264.

Baltrušaitis, Tadas, Chaitanya Ahuja, and Louis-Philippe Morency (2018). "Multimodal machine learning: A survey and taxonomy." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Banko, Michele, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni (2007). "Open Information Extraction from the Web." In: *IJCAI*.

Baudiš, Petr (2015). "YodaQA: a modular question answering system pipeline." In: *POSTER 2015-19th International Student Conference on Electrical Engineering*, pp. 1156–1165.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). "Representation learning: A review and new perspectives." In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1798–1828.

Berant, Jonathan, Andrew Chou, Roy Frostig, and Percy Liang (2013). "Semantic Parsing on Freebase from Question-Answer Pairs." In: *EMNLP*.

Bertsekas, Dimitri P (1985). "A distributed asynchronous relaxation algorithm for the assignment problem." In: *Decision and Control, 1985 24th IEEE Conference on*. IEEE, pp. 1703–1704.

Beygelzimer, Alina, Sham Kakade, and John Langford (2006). "Cover trees for nearest neighbor." In: *Proceedings of the 23rd international conference on Machine learning*. ACM, pp. 97–104.

Binney, James and Michael Merrifield (1998). *Galactic astronomy*. Princeton University Press.

Bisk, Yonatan, Siva Reddy, John Blitzer, Julia Hockenmaier, and Mark Steedman (2016). "Evaluating Induced CCG Parsers on Grounded Semantic Parsing." In: *EMNLP*.

Blei, D., T. Griffiths, and M. Jordan (2010). "The Nested Chinese Restaurant Process and Bayesian Nonparametric Inference of Topic Hierarchies." In: *Journal of the ACM* 57.2, pp. 1–30. URL: http://doi.acm.org/10.1145/1667053.1667056.

Blei, D., A. Ng, and M. Jordan (2002). "Latent Dirichlet Allocation." In: *Advances in Neural Information Processing Systems 14*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. Cambridge, MA: MIT Press.

Blei, David M., Andrew Y. Ng, and Michael I. Jordan (2003). "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3, pp. 993–1022. ISSN: 1532-4435.

Blei, David and Peter I. Frazier (2011). *Distance Dependent Chinese Restaurant Processes*. JMLR.

Bollacker, Kurt, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor (2008). "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge." In: *ICDM*. Vancouver, Canada.

Bordes, Antoine, Nicolas Usunier, Sumit Chopra, and Jason Weston (2015). "Large-scale simple question answering with memory networks." In: *arXiv preprint arXiv:1506.02075*.

Bottou, Léon (1991). "Stochastic gradient learning in neural networks." In: *Proceedings of Neuro-Nimes* 91.8.

Boulanger-Lewandowski, Nicolas, Yoshua Bengio, and Pascal Vincent (2012). "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription." In: *ICML*.

Bourbaki, Nicolas (1990). *Eléments de mathématiques: théorie des ensembles, chapitres 1 à 4*. Vol. 1. Masson.

Brock, Andrew, Theodore Lim, JM Ritchie, and Nick Weston (2016). "Generative and Discriminative Voxel Modeling with Convolutional Neural Networks." In: *arXiv preprint arXiv:1608.04236*.

Broder, Andrei Z (1997). "On the resemblance and containment of documents." In: *Compression and complexity of sequences 1997. proceedings*. IEEE, pp. 21–29.

Buntine, W. (2002). "Variational extensions to EM and multinomial PCA." In: *Proc. European Conf. Machine Learning*. Springer, pp. 23–34.

Cacciatore, Timothy W and Steven J Nowlan (1994). "Mixtures of controllers for jump linear and non-linear plants." In: *Advances in neural information processing systems*, pp. 719–726.

Canny, J. (2004). "GaP: a factor model for discrete data." In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, pp. 122–129.

Carlson, Andrew, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell (2010). "Toward an Architecture for Never-ending Language Learning." In: *AAAI*.

Carmon, Yair, John C. Duchi, Oliver Hinder, and Aaron Sidford (2017). ""Convex Until Proven Guilty": Dimension-Free Acceleration of Gradient Descent on Non-Convex Functions." In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 654–663.

Carmon, Yair, John C. Duchi, Oliver Hinder, and Aaron Sidford (2018). "Accelerated Methods for Non-Convex Optimization." In: *SIAM Journal on Optimization* 28.2, pp. 1751–1772.

Carreira, Joao and Andrew Zisserman (2017). "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset." In: *CVPR*.

Cartis, C. and K. Scheinberg (2017). "Global convergence rate analysis of unconstrained optimization methods based on probabilistic models." In: *Mathematical Programming*, pp. 1–39. ISSN: 1436-4646. DOI: 10.1007/s10107-017-1137-4.

Cayton, L. (2008). "Fast nearest neighbor retrieval for bregman divergences." In: *International Conference on Machine Learning ICML*. ACM, pp. 112–119.

Celeux, Gilles and Jean Diebolt (1985). "The SEM algorithm: a probabilistic teacher algorithm derived from the EM algorithm for the mixture problem." In: *Computational statistics quarterly* 2.1, pp. 73–82.

Cereto-Massagué, Adrià, María José Ojeda, Cristina Valls, Miquel Mulero, Santiago Garcia-Vallvé, and Gerard Pujadas (2015). "Molecular fingerprint similarity search in virtual screening." In: *Methods* 71, pp. 58–63.

Chang, Angel X, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. (2015). "Shapenet: An information-rich 3d model repository." In: *arXiv preprint arXiv:1512.03012*.

Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber (2006). "Bigtable: a distributed storage system for structured data." In: *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*. Seattle, Washington: USENIX Association, pp. 205–218.

Chang, Jonathan, Sean Gerrish, Chong Wang, Jordan L. Boyd-graber, and David M. Blei (2009). "Reading Tea Leaves: How Humans Interpret Topic Models." In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta. Curran Associates, Inc., pp. 288–296.

Chang, Michael B, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum (2016). "A compositional object-based approach to learning physical dynamics." In: *arXiv preprint arXiv:1612.00341*.

Chen, C., L. Du, and W.L. Buntine (2011). "Sampling Table Configurations for the Hierarchical Poisson-Dirichlet Process." In: *European Conference on Machine Learning*. Ed. by D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis. Springer, pp. 296–311. URL: http://dx.doi.org/10.1007/978-3-642-23780-5.

Chen, Danqi, Adam Fisch, Jason Weston, and Antoine Bordes (2017). "Reading Wikipedia to Answer Open-Domain Questions." In: *Association for Computational Linguistics (ACL)*.

Chen, George H, Devavrat Shah, et al. (2018). "Explaining the Success of Nearest Neighbor Methods in Prediction." In: *Foundations and Trends® in Machine Learning* 10.5-6, pp. 337–588.

Chen, Minmin, Alice Zheng, and Kilian Weinberger (2013). "Fast Image Tagging." In: *Proceedings of The 30th International Conference on Machine Learning*, pp. 1274–1282.

Chen, Xu, Xiuyuan Cheng, and Stéphane Mallat (2014). "Unsupervised deep haar scattering on graphs." In: *Advances in Neural Information Processing Systems*, pp. 1709–1717.

Cho, Sébastien Jean Kyunghyun, Roland Memisevic, and Yoshua Bengio (2015). "On Using Very Large Target Vocabulary for Neural Machine Translation." In:

Choi, Eunsol, Daniel Hewlett, Alexandre Lacoste, Illia Polosukhin, Jakob Uszkoreit, and Jonathan Berant (2016). "Hierarchical Question Answering for Long Documents." In: *arXiv preprint arXiv:1611.01839*.

Choi, Eunsol, Tom Kwiatkowski, and Luke Zettlemoyer (2015). "Scalable Semantic Parsing with Partial Ontologies." In: *ACL*.

Choromanska, Anna, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun (2015). "The Loss Surface of Multilayer Networks." In: *Artificial Intelligence and Statistics*, pp. 192–204.

Christopher, M Bishop (2016). *Pattern Recognition and Machine Learning*. Springer-Verlag New York.

Chung, Junyoung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio (2015). "A recurrent latent variable model for sequential data." In: *Advances in Neural Information Processing Systems*, pp. 2980–2988.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)." In: *arXiv preprint arXiv:1511.07289*.

Cohen, Taco S and Max Welling (2016). "Group Equivariant Convolutional Networks." In: *arXiv preprint arXiv:1602.07576*.

Coldeway, Devin (May 2016). "Terrapattern is reverse image search for maps, powered by a neural network"." In: *Techcrunch*. URL: https://techcrunch.com/2016/05/25/terrapattern-is-a-neural-net-powered-reverse-image-search-for-maps/.

Coley, Connor W., Regina Barzilay, William H. Green, Tommi S. Jaakkola, and Klavs F. Jensen (2017). "Convolutional Embedding of Attributed Molecular Graphs for Physical Property Prediction." In: *Journal of Chemical Information and Modeling* 57.8, pp. 1757–1772.

Collobert, Ronan and Jason Weston (2008). "A unified architecture for natural language processing: deep neural networks with multitask learning." In: *Proceedings of ICML*.

Connolly, AJ, I Csabai, AS Szalay, DC Koo, RG Kron, and JA Munn (1995). "Slicing through multicolor space: Galaxy redshifts from broadband photometry." In: *arXiv preprint astro-ph/9508100*.

Csűrös, Miklós (2010). "Approximate Counting with a Floating-Point Counter." In: *Computing and Combinatorics (COCOON 2010)*. Ed. by M. T. Thai and Sartaj Sahni. Lecture Notes in Computer Science 6196. See also http://arxiv.org/pdf/0904.3062.pdf. Springer Berlin Heidelberg, pp. 358–367. ISBN: 978-3-642-14030-3. DOI: 10.1007/978-3-642-14031-0_39.

Ćurgus, Branko and Vania Mascioni (2006). "Roots and polynomials as homeomorphic spaces." In: *Expositiones Mathematicae* 24.1, pp. 81–95.

Dalal, Navneet and Bill Triggs (2005). "Histograms of oriented gradients for human detection." In: *CVPR*.

Das, Rajarshi, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum (2018). "Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning." In: *Proceedings of the International Conference on Representation Learning*.

Das, Rajarshi, Arvind Neelakantan, David Belanger, and Andrew McCallum (2017). "Chains of reasoning over entities, relations, and text using recurrent neural networks." In: *EACL*.

Dasgupta, Sanjoy and Kaushik Sinha (2013). "Randomized partition trees for exact nearest neighbor search." In: *Conference on Learning Theory*, pp. 317–337.

Dauphin, Yann N., Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio (2014). "Identifying and Attacking the Saddle Point Problem in High-dimensional Non-convex Optimization." In: *Proceedings of the 27th International Conference on Neural Information Processing Systems*. NIPS'14. Montreal, Canada, pp. 2933–2941.

Dauphin, Yann, Harm de Vries, and Yoshua Bengio (2015). "Equilibrated adaptive learning rates for non-convex optimization." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., pp. 1504–1512.

Dawson, Donald A. (1974). "Synchronous and asynchronous reversible Markov systems." In: *Canadian mathematical bulletin* 17, pp. 633–649. ISSN: 0008-4395. DOI: 10.4153/CMB-1974-117-4.

Dean, Jeffrey and Sanjay Ghemawat (Jan. 2008). "MapReduce: Simplified Data Processing on Large Clusters." In: *Commun. ACM* 51.1, pp. 107–113. ISSN: 0001-0782. DOI: 10.1145/1327452.1327492.

Deerwester, S. C., S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman (1990). "Indexing by latent semantic analysis." In: *Journal of the American Society for Information Science*.

Defazio, Aaron J, Tibério S Caetano, and Justin Domke (2014). "Finito: A faster, permutable incremental gradient method for big data problems." In: *International Conference on Machine Learning*, pp. 1125–1133.

Defazio, Aaron, Francis Bach, and Simon Lacoste-Julien (2014). "SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives." In: *NIPS 27*, pp. 1646–1654.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." In: *Journal of the Royal Statistical Society B* 39.1, pp. 1–22.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). "Imagenet: A large-scale hierarchical image database." In: *CVPR*.

Denton, Emily L, Soumith Chintala, Rob Fergus, et al. (2015). "Deep generative image models using a laplacian pyramid of adversarial networks." In: *Advances in neural information processing systems*, pp. 1486–1494.

Dhingra, Bhuwan, Kathryn Mazaitis, and William W Cohen (2017). "Quasar: Datasets for Question Answering by Search and Reading." In: *arXiv preprint arXiv:1707.03904*.

Dhingra, Bhuwan, Danish Pruthi, and Dheeraj Rajagopal (2018). "Simple and Effective Semi-Supervised Question Answering." In: *NAACL*.

Diba, Ali, Vivek Sharma, and Luc Van Gool (2017). "Deep temporal linear encoding networks." In: *CVPR*.

Donahue, Jeffrey, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell (2015). "Long-term recurrent convolutional networks for visual recognition and description." In: *CVPR*.

Dong, Xin, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang (2014). "Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion." In: KDD '14. New York, NY, USA.

Doucet, Arnaud, Nando de Freitas, and Neil Gordon (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Duvenaud, David K, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams (2015). "Convolutional networks on graphs for learning molecular fingerprints." In: *Advances in neural information processing systems*, pp. 2224–2232.

Dyer, Chris, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik (2010). "cdec: A Decoder, Alignment, and Learning framework for finite-state and context-free translation models." In: *Proceedings of ACL*.

Eckart, Ben, Kihwan Kim, Alejandro Troccoli, Alonzo Kelly, and Jan Kautz (2015). "Mlmd: Maximum likelihood mixture decoupling for fast and accurate point cloud registration." In: *3D Vision (3DV), 2015 International Conference on*. IEEE, pp. 241–249.

Faber, Felix A., Luke Hutchison, Bing Huang, Justin Gilmer, Samuel S. Schoenholz, George E. Dahl, Oriol Vinyals, Steven Kearnes, Patrick F. Riley, and O. Anatole von Lilienfeld (2017). "Prediction Errors of Molecular Machine Learning Models Lower than Hybrid DFT Error." In: *Journal of Chemical Theory and Computation* 13.11, pp. 5255–5264.

Faber, Felix A., Alexander Lindmaa, O. Anatole von Lilienfeld, and Rickard Armiento (2016). "Machine Learning Energies of 2 Million Elpasolite (ABC_2D_6) Crystals." In: *Phys. Rev. Lett.* 117 (13), p. 135502. DOI: 10.1103/PhysRevLett.117.135502. URL: http://link.aps.org/doi/10.1103/PhysRevLett.117.135502.

Fader, Anthony, Luke Zettlemoyer, and Oren Etzioni (2014). "Open question answering over curated and extracted knowledge bases." In: *KDD*. ACM, pp. 1156–1165.

Fan, Haoqiang, Hao Su, and Leonidas Guibas (n.d.). "A point set generation network for 3d object reconstruction from a single image." In:

Feichtenhofer, Christoph, Axel Pinz, and Richard Wildes (2016). "Spatiotemporal residual networks for video action recognition." In: *NIPS*.

Feichtenhofer, Christoph, Axel Pinz, and Richard P Wildes (2017). "Spatiotemporal multiplier networks for video action recognition." In: *CVPR*.

Feichtenhofer, Christoph, Axel Pinz, and Andrew Zisserman (2016). "Convolutional two-stream network fusion for video action recognition." In: *CVPR*.

Feldman, Dan, Melanie Schmidt, and Christian Sohler (2013). "Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering." In: *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, pp. 1434–1453.

Fellbaum, C. (1998). *WordNet: An electronic lexical database*. The MIT press.

Feng, S. L., R. Manmatha, and V. Lavrenko (2004). "Multiple Bernoulli Relevance Models for Image and Video Annotation." In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. CVPR'04. Washington, D.C., USA: IEEE Computer Society, pp. 1002–1009.

Ferguson, Thomas S (1973). "A Bayesian analysis of some nonparametric problems." In: *The annals of statistics*, pp. 209–230.

Ferrucci, David, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. (2010). "Building Watson: An overview of the DeepQA project." In: *AI magazine*.

Finlayson, Mark (2014). "Proceedings of the Seventh Global Wordnet Conference." In: Tartu, Estonia. Chap. Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation, pp. 78–85. URL: http://aclweb.org/anthology/W14-0111.

Finn, Chelsea, Ian Goodfellow, and Sergey Levine (2016). "Unsupervised learning for physical interaction through video prediction." In: *Advances in neural information processing systems*, pp. 64–72.

Formann, Anton K and Thomas Kohlmann (1996). "Latent class analysis in medical research." In: *Statistical methods in medical research* 5.2, pp. 179–211.

Frigola, Roger, Fredrik Lindsten, Thomas B. Schön, and Carl E. Rasmussen (2013). "Bayesian Inference and Learning in Gaussian Process State-Space Models with Particle MCMC." In: *NIPS*.

Fu, Cong, Chao Xiang, Changxu Wang, and Deng Cai (2017). "Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph." In: *arXiv preprint arXiv:1707.00143*.

Gabrilovich, Evgeniy, Michael Ringgaard, and Amarnag Subramanya (2013). *FACC1: Freebase annotation of ClueWeb corpora*. (http://lemurproject.org/clueweb09/.

Ganitkevitch, Juri, Benjamin Van Durme, and Chris Callison-Burch (June 2013). "PPDB: The Paraphrase Database." In: *Proceedings of NAACL-HLT*. Atlanta, Georgia: Association for Computational Linguistics, pp. 758–764. URL: http://cs.jhu.edu/~ccb/publications/ppdb.pdf.

Gardner, Matt and Jayant Krishnamurthy (2017). "Open-Vocabulary Semantic Parsing with both Distributional Statistics and Formal Knowledge." In: *AAAI*.

Ge, Rong, Furong Huang, Chi Jin, and Yang Yuan (2015). "Escaping From Saddle Points - Online Stochastic Gradient for Tensor Decomposition." In: *Proceedings of The 28th Conference on Learning Theory, COLT 2015*, pp. 797–842.

Gens, Robert and Pedro M Domingos (2014). "Deep symmetry networks." In: *Advances in neural information processing systems*, pp. 2537–2545.

Ghadimi, Saeed and Guanghui Lan (2013). "Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming." In: *SIAM Journal on Optimization* 23.4, pp. 2341–2368. DOI: 10.1137/120880811.

Ghahramani, Zoubin and Katherine A Heller (2005). "Bayesian sets." In: *NIPS*. Vol. 2, pp. 22–23.

Ghahramani, Zoubin and Geoffrey E. Hinton (1996). *Parameter estimation for linear dynamical systems*. Tech. rep.

Ghahramani, Zoubin and Sam T. Roweis (1999). "Learning Nonlinear Dynamical Systems using an EM Algorithm." In: *NIPS*.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter (1995). *Markov Chain Monte Carlo in Practice*. Chapman & Hall.

Gilmer, Justin, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl (2017). "Neural message passing for quantum chemistry." In: *ICML*.

Girdhar, Rohit and Deva Ramanan (2017). "Attentional Pooling for Action Recognition." In: *NIPS*.

Girdhar, Rohit, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan Russell (2017). "ActionVLAD: Learning spatio-temporal aggregation for action classification." In: *CVPR*.

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks." In: *AISTATS*.

Goel, Aman, Craig A Knoblock, and Kristina Lerman (2012). "Exploiting structure within data for accurate labeling using conditional random fields." In: *Proceedings on the International Conference on Artificial Intelligence (ICAI)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), p. 1.

Gong, Yichen and Samuel R Bowman (2017). "Ruminating reader: Reasoning with gated multi-hop attention." In: *arXiv preprint arXiv:1704.07415*.

Gonzalez, Joseph, Yucheng Low, Arthur Gretton, and Carlos Guestrin (2011). "Parallel gibbs sampling: from colored fields to thin junction trees." In: *International Conference on Artificial Intelligence and Statistics*, pp. 324–332.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014). "Generative adversarial nets." In: *NIPS*.

Graves, Alex and Jürgen Schmidhuber (2005). "Framewise phoneme classification with bidirectional lstm and other neural network architectures." In: *Neural Networks*.

Gray, Alexander G and Andrew W Moore (2000). "N-Body'problems in statistical learning." In: *NIPS*. Vol. 4. Citeseer, pp. 521–527.

Green Jr, Bert F, Alice K Wolf, Carol Chomsky, and Kenneth Laughery (1961). "Baseball: an automatic question-answerer." In: *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*. ACM, pp. 219–224.

Gregor, Karol, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra (2015). "DRAW: A Recurrent Neural Network For Image Generation." In: *ICML*.

Griffiths, T. and Z. Ghahramani (2011). "The Indian Buffet Process: An Introduction and Review." In: *Journal of Machine Learning Research* 12, pp. 1185–1224.

Griffiths, T.L. and M. Steyvers (2004). "Finding scientific topics." In: *Proceedings of the National Academy of Sciences* 101, pp. 5228–5235.

Grubinger, Michael (2007). *Analysis and evaluation of visual information systems performance*. English. Thesis. Thesis (Ph. D.)–Victoria University (Melbourne, Vic.), 2007. URL: http://eprints.vu.edu.au/1435.

Gu, Chenjie, Manzil Zaheer, and Xin Li (2014). "Multiple-Population Moment Estimation: Exploiting Interpopulation Correlation for Efficient Moment Estimation in Analog/Mixed-Signal Validation." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.7, pp. 961–974.

Guillaumin, Matthieu, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid (2009). "Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation." In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, pp. 309–316.

Gulrajani, Ishaan, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville (2017). "Improved Training of Wasserstein GANs." In: *arXiv:1704.00028*.

Guo, Jiang, Wanxiang Che, Haifeng Wang, and Ting Liu (2014). "Revisiting embedding features for simple semi-supervised learning." In: *Proceedings of EMNLP*.

Guttenberg, Nicholas, Nathaniel Virgo, Olaf Witkowski, Hidetoshi Aoki, and Ryota Kanai (2016). "Permutation-equivariant neural networks applied to dynamics prediction." In: *arXiv preprint arXiv:1612.04530*.

Guu, K., J. Miller, and P. Liang (2015). "Traversing Knowledge Graphs in Vector Space." In: *EMNLP*.

Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (Nov. 2009). "The WEKA Data Mining Software: An Update." In: *SIGKDD Explor. Newsl.* 11.1, pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278. URL: http://doi.acm.org/10.1145/1656274.1656278.

Hamilton, William L., Rex Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs." In: *CoRR* abs/1706.02216. arXiv: 1706.02216. URL: http://arxiv.org/abs/1706.02216.

Han, Xu, Zhiyuan Liu, and Maosong Sun (2016). "Joint representation learning of text and knowledge for knowledge graph completion." In: *arXiv preprint arXiv:1611.04125*.

Han, Yongkoo, Kisung Park, Jihye Hong, Noor Ulamin, and Young-Koo Lee (2015). "Distance-constraint k-nearest neighbor searching in mobile sensor networks." In: *Sensors* 15.8, pp. 18209–18228.

Harris, Zellig (1954). "Distributional structure." In: *Word* 10.23, pp. 146–162.

Hartford, Jason S, James R Wright, and Kevin Leyton-Brown (2016). "Deep learning for predicting human strategic behavior." In: *Advances in Neural Information Processing Systems*, pp. 2424–2432.

Haveliwala, Taher H (2002). "Topic-sensitive pagerank." In: *Proceedings of the 11th international conference on World Wide Web*. ACM, pp. 517–526.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016a). "Deep residual learning for image recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016b). "Identity mappings in deep residual networks." In: *ECCV*.

Hermann, Karl Moritz and Phil Blunsom (2014). "Multilingual Models for Compositional Distributed Semantics." In: *arXiv preprint arXiv:1404.4641*.

Hill, Felix, Antoine Bordes, Sumit Chopra, and Jason Weston (2016). "The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations." In: *ICLR*.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks." In: *Science* 313.5786, pp. 504–507.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation* 9.8, pp. 1735–1780.

Hoffman, Kenneth and Ray Kunze (1971). "Linear Algebra." In: *Englewood Cliffs, New Jersey*.

Hoffman, Matthew D., David M. Blei, Chong Wang, and John Paisley (May 2013). "Stochastic Variational Inference." In: *Journal of Machine Learning Research* 14, pp. 1303–1347. ISSN: 1533-7928.

Hsieh, Cho-Jui et al. (2015). "Exploiting structure in large-scale optimization for machine learning." PhD thesis.

Hu, Diane J, Rob Hall, and Josh Attenberg (2014). "Style in the long tail: Discovering unique interests with latent variable models in large scale social e-commerce." In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 1640–1649.

Hu, Minghao, Yuxing Peng, and Xipeng Qiu (2017). "Mnemonic reader for machine comprehension." In: *arXiv preprint arXiv:1705.02798*.

Hu, Pengfei, Wenju Liu, Wei Jiang, and Zhanlei Yang (2012). "Latent Topic Model Based on Gaussian-LDA for Audio Retrieval." In: *Pattern Recognition*. Vol. 321. CCIS. Springer, pp. 556–563.

Hubel, David H and Torsten N Wiesel (1968). "Receptive fields and functional architecture of monkey striate cortex." In: *The Journal of physiology* 195.1, pp. 215–243.

Huszar, Ferenc (2016). *How powerful are Graph Convolutions?* URL: https://www.inference.vc/how-powerful-are-graph-convolutions-review-of-kipf-welling-2016-2/ (visited on 07/24/2018).

Indyk, Piotr and Rajeev Motwani (1998). "Approximate nearest neighbors: towards removing the curse of dimensionality." In: *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, pp. 604–613.

Ioffe, Sergey and Christian Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *ICML*.

Isola, Phillip, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros (2017). "Image-to-image translation with conditional adversarial networks." In: *arXiv preprint*.

Jahanseir, Mahmoodreza and Don Sheehy (2016). "Transforming Hierarchical Trees on Metric Spaces." In: *CCCG*, pp. 107–113.

Jain, Sarthak (2016). "Question answering over knowledge base using factual memory networks." In: *Proceedings of the NAACL Student Research Workshop*, pp. 109–115.

Jian, Bing and Baba C Vemuri (2005). "A robust algorithm for point set registration using mixture of Gaussians." In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 2. IEEE, pp. 1246–1251.

Jiang, Jiatao, Zhen Cui, Chunyan Xu, Chengzheng Li, and Jian Yang (2018). "Walk-Steered Convolution for Graph Classification." In: *arXiv preprint arXiv:1804.05837*.

Jiang, K., B. Kulis, and M. Jordan (2012). "Small-variance asymptotics for exponential family Dirichlet process mixture models." In: *Neural Information Processing Systems NIPS*, pp. 3167–3175.

Jin, Chi, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan (2017). "How to Escape Saddle Points Efficiently." In: *International Conference on Machine Learning*, pp. 1724–1732.

Johnson, Jeff, Matthijs Douze, and Hervé Jégou (2017). "Billion-scale similarity search with gpus." In: *arXiv preprint arXiv:1702.08734*.

Johnson, Matthew J., David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, and Ryan P. Adams (2016). "Composing graphical models with neural networks for structured representations and fast inference." In: *NIPS*.

Johnson, Rie and Tong Zhang (2013). "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction." In: *NIPS 26*, pp. 315–323.

Jordan, Michael I., Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul (Nov. 1999). "An Introduction to Variational Methods for Graphical Models." In: *Mach. Learn.* 37.2, pp. 183–233. ISSN: 0885-6125. DOI: 10.1023/A:1007665907178.

Joshi, Mandar, Uma Sawant, and Soumen Chakrabarti (Oct. 2014). "Knowledge Graph and Corpus Driven Segmentation and Answer Inference for Telegraphic Entity-seeking Queries." In: *EMNLP*.

Jozefowicz, Rafal, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu (2016). "Exploring the limits of language modeling." In: *arXiv preprint arXiv:1602.02410*.

Julier, Simon J. and Jeffrey K. Uhlmann (1997). "A new extension of the Kalman filter to nonlinear systems." In: *International Symposium on Aerospace/Defense Sensing, Simulation and Controls*.

Jung, I., M. Berges, J. Garrett, and B. Poczos (2015). "Exploration and Evaluation of AR, MPCA and KL Anomaly Detection Techniques to Embankment Dam Piezometer Data." In: *Advanced Engineering Informatics*.

Kang, Eunsu (2017). *FACE Exhibition*. Judith Rae Solomon Gallery, Youngstown, OH. http://art.ysu.edu/2017/09/06/face-by-eunsu-kang-and-collaborators/.

Kantorov, Vadim and Ivan Laptev (2014). "Efficient feature extraction, encoding and classification for action recognition." In: *CVPR*.

Karger, D. R. and M. Ruhl (2002). "Finding nearest neighbors in growth-restricted metrics." In: *Symposium on Theory of Computing STOC*. ACM, pp. 741–750.

Karl, Maximilian, Maximilian Soelch, Justin Bayer, and Patrick van der Smagt (2017). "Deep variational Bayes filters: Unsupervised learning of state space models from raw data." In: *ICLR*.

Karpathy, Andrej, Justin Johnson, and Li Fei-Fei (2015). "Visualizing and understanding recurrent networks." In: *arXiv preprint arXiv:1506.02078*.

Karpathy, Andrej, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei (2014). "Large-scale video classification with convolutional neural networks." In: *CVPR*.

Karras, Tero, Timo Aila, Samuli Laine, and Jaakko Lehtinen (2017). "Progressive growing of gans for improved quality, stability, and variation." In: *arXiv preprint arXiv:1710.10196*.

Kearnes, Steven, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley (Aug. 2016). "Molecular graph convolutions: moving beyond fingerprints." In: *Journal of Computer-Aided Molecular Design* 30.8, pp. 595–608.

Khesin, Boris A and Serge L Tabachnikov (2014). *Arnold: Swimming Against the Tide*. Vol. 86. American Mathematical Society.

Khetan, Abhishek, Manzil Zaheer, Venkatasubramanian Viswanathan, and Barnabás Póczos (2018). "Learning Density Functional Theory Solely From Molecular Geometry." In: *arXiv preprint*.

Kiefer, Cedric and Julia Laub (2013). *Google Faces*. https://onformative.com/work/google-faces. Accessed: 2018-08-15.

Kingma, Diederik P. and Jimmy Ba (2014). "Adam: A Method for Stochastic Optimization." In: *Arxiv* abs/1412.6980.

Kingma, Diederik and Max Welling (2014). "Auto-encoding variational bayes." In: *ICLR*.

Kipf, Thomas N and Max Welling (2016). "Semi-supervised classification with graph convolutional networks." In: *arXiv preprint arXiv:1609.02907*.

Koch, W. and M. C. Holthausen (2001). "The Hohenberg-Kohn Theorems." In: *A Chemist's Guide to Density Functional Theory*. Chap. 4, pp. 33–40.

Köck, Mirjam and Alexandros Paramythis (2011). "Activity sequence modelling and dynamic clustering for personalized e-learning." In: *User Modeling and User-Adapted Interaction* 21.1-2, pp. 51–97.

Konečny, Jakub, Jie Liu, Peter Richtárik, and Martin Takáč (2016). "Mini-batch semi-stochastic gradient descent in the proximal setting." In: *IEEE Journal of Selected Topics in Signal Processing* 10.2, pp. 242–255.

Korn, Flip, Nikolaos Sidiropoulos, Christos Faloutsos, Eliot Siegel, and Zenon Protopapas (1996). "Fast Nearest Neighbor Search in Medical Image Databases." In: *Proceedings of the*

22th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc., pp. 215–226.

Korpusik, Mandy, Shigeyuki Sakaki, and Francine Chen Yan-Ying Chen (2016). "Recurrent Neural Networks for Customer Purchase Prediction on Twitter." In: *CBRecSys 2016*, p. 47.

Krause, Sebastian, Leonhard Hennig, Andrea Moro, Dirk Weissenborn, Feiyu Xu, Hans Uszkoreit, and Roberto Navigli (2016). "Sar-graphs: A language resource connecting linguistic knowledge with semantic relations from knowledge graphs." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 37, pp. 112–131.

Krauthgamer, Robert and James R Lee (2004). "Navigating nets: simple algorithms for proximity search." In: *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, pp. 798–807.

Krishnamurthy, Jayant and Tom Mitchell (2012). "Weakly Supervised Training of Semantic Parsers." In: *EMNLP*.

Krishnan, Rahul G., Uri Shalit, and David Sontag (2015). "Deep Kalman Filters." In: *arXiv preprint arXiv:1511.05121*.

Krishnan, Rahul G., Uri Shalit, and David Sontag (2017). "Structured Inference Networks for Nonlinear State Space Models." In: *AAAI*.

Krizhevsky, Alex and Geoffrey Hinton (2009). "Learning multiple layers of features from tiny images." In:

Kuehne, Hildegard, Hueihan Jhuang, Estibaliz Garrote, Tomaso Poggio, and Thomas Serre (2011). "HMDB: a large video database for human motion recognition." In: *ICCV*.

Kupiec, Julian (1993). "MURAX: A robust linguistic approach for question answering using an on-line encyclopedia." In: *SIGIR*. ACM.

Kushner, Harold Joseph and Dean S Clark (2012). *Stochastic approximation methods for constrained and unconstrained systems*. Vol. 26. Springer Science & Business Media.

Lamb, Alex M, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio (2016). "Professor forcing: A new algorithm for training recurrent networks." In: *Advances In Neural Information Processing Systems*, pp. 4601–4609.

Lan, Guanghui and Yi Zhou (2017). "An optimal randomized incremental gradient method." In: *Mathematical programming*, pp. 1–49.

Lao, Ni, Amarnag Subramanya, Fernando Pereira, and William W Cohen (2012). "Reading the web with learned syntactic-semantic inference rules." In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, pp. 1017–1026.

Laptev, Ivan, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld (2008). "Learning realistic human actions from movies." In: *CVPR*.

Le Gall, Didier (1991). "MPEG: A video compression standard for multimedia applications." In: *Communications of the ACM* 34.4, pp. 46–58.

Le, Quoc and Tomas Mikolov (2014). "Distributed Representations of Sentences and Documents." In: *Proc. ICML*.

Lebowitz, Joel L., Christian Maes, and Eugene R. Speer (Apr. 1990). "Statistical mechanics of probabilistic cellular automata." In: *Journal of statistical physics* 59, pp. 117–170. ISSN: 0022-4715. DOI: 10.1007/BF01015566.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lee, Kenton, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das (2016). "Learning Recurrent Span Representations for Extractive Question Answering." In: *arXiv preprint arXiv:1611.01436*.

Levy, Kfir Y. (2016). "The Power of Normalization: Faster Evasion of Saddle Points." In: *Arxiv* abs/1611.04831.

Li, Aaron Q, Amr Ahmed, Sujith Ravi, and Alexander J Smola (2014). "Reducing the sampling complexity of topic models." In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. ACM, pp. 891–900.

Li, Chun-Liang, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos (2017). "MMD GAN: Towards deeper understanding of moment matching network." In: *Advances in Neural Information Processing Systems*, pp. 2200–2210.

Li, Ke and Jitendra Malik (2017). "Fast k-Nearest Neighbour Search via Prioritized DCI." In: *International Conference on Machine Learning*, pp. 2081–2090.

Li, W., D. Blei, and A. McCallum (2007). "Nonparametric Bayes Pachinko Allocation." In: *Uncertainty in Artificial Intelligence*.

Li, Yujia, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel (2016). "Gated graph sequence neural networks." In: *ICLR*.

Lian, Xiangru, Yijun Huang, Yuncheng Li, and Ji Liu (2015). "Asynchronous parallel stochastic gradient for nonconvex optimization." In: *Advances in Neural Information Processing Systems*, pp. 2737–2745.

Liang, Chen, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao (2017). "Neural symbolic machines: Learning semantic parsers on freebase with weak supervision." In: *ACL*.

Lin, Hong-Wei, Chiew-Lan Tai, and Guo-Jin Wang (2004). "A mesh reconstruction algorithm driven by an intrinsic property of a point cloud." In: *Computer-Aided Design* 36.1, pp. 1–9.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick (2014). "Microsoft coco: Common objects in context." In: *European Conference on Computer Vision*. Springer, pp. 740–755.

Lindsten, Fredrik (2013). "An efficient stochastic approximation EM algorithm using conditional particle filters." In: *ICASSP*.

Lindsten, Fredrik, Michael I. Jordan, and Thomas B. Schön (2014). "Particle Gibbs with Ancestor Sampling." In: *Journal of Machine Learning Research*.

Ling, Wang, Tiago Luis, Luis Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso (2015). "Finding function in form: Compositional character models for open vocabulary word representation." In: *arXiv preprint arXiv:1508.02096*.

Lipton, Zachary C, John Berkowitz, and Charles Elkan (2015). "A critical review of recurrent neural networks for sequence learning." In: *arXiv preprint arXiv:1506.00019*.

Liu, Ting, Charles Rosenberg, and Henry A Rowley (2007). "Clustering billions of images with large scale nearest neighbor search." In: *Applications of Computer Vision, 2007. WACV'07. IEEE Workshop on*. IEEE, pp. 28–28.

Liu, Ziwei, Ping Luo, Xiaogang Wang, and Xiaoou Tang (2015). "Deep Learning Face Attributes in the Wild." In: *Proceedings of International Conference on Computer Vision (ICCV)*.

Ljung, Lennart (1977). "Analysis of recursive stochastic algorithms." In: *Automatic Control, IEEE Transactions on* 22.4, pp. 551–575.

Loosli, Gaëlle, Stéphane Canu, and Léon Bottou (2007). "Training Invariant Support Vector Machines using Selective Sampling." In: *Large Scale Kernel Machines*. Ed. by Léon Bottou, Olivier Chapelle, Dennis DeCoste, and Jason Weston. Cambridge, MA.: MIT Press, pp. 301–320.

Lopez-Paz, David, Robert Nishihara, Soumith Chintala, Bernhard Schölkopf, and Léon Bottou (2016). "Discovering Causal Signals in Images." In: *arXiv preprint arXiv:1605.08179*.

Louis, Pierre-Yves (Sept. 2002). "Automates Cellulaires Probabilistes : mesures stationnaires, mesures de Gibbs associées et ergodicité." PhD thesis. Université des Sciences et Technologies de Lille and il Politecnico di Milano.

Low, Yucheng, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein (2010). "GraphLab: A New Framework for Parallel Machine Learning." In: *CoRR* abs/1006.4990. URL: http://arxiv.org/abs/1006.4990.

Lu, Zhengdong, Haotian Cui, Xianggen Liu, Yukun Yan, and Daqi Zheng (2017). "Object-oriented Neural Programming (OONP) for Document Understanding." In: *arXiv preprint arXiv:1709.08853*.

Lucic, Mario, Olivier Bachem, and Andreas Krause (2016). "Strong coresets for hard and soft Bregman clustering with applications to exponential family mixtures." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 1–9.

Ma, Chih-Yao, Min-Hung Chen, Zsolt Kira, and Ghassan AlRegib (2017). "TS-LSTM and Temporal-Inception: Exploiting Spatiotemporal Dynamics for Activity Recognition." In: *arXiv preprint arXiv:1703.10667*.

Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE." In: *Journal of Machine Learning Research* 9.Nov, pp. 2579–2605.

Mairesse, Jean and Irène Marcovici (Nov. 2014). "Around probabilistic cellular automata." In: *Theoretical Computer Science* 559, pp. 42–72. ISSN: 0304-3975. DOI: doi:10.1016/j.tcs.2014.09.009.

Makadia, Ameesh, Vladimir Pavlovic, and Sanjiv Kumar (2008). "A New Baseline for Image Annotation." In: *Proceedings of the 10th European Conference on Computer Vision: Part III*. ECCV '08. Marseille, France: Springer-Verlag, pp. 316–329.

Malewicz, Grzegorz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski (2010). "Pregel: A System for Large-scale Graph Processing." In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*. SIGMOD '10. Indianapolis, Indiana, USA: ACM, pp. 135–146. ISBN: 978-1-4503-0032-2. DOI: 10.1145/1807167.1807184.

Malkov, Yu A and DA Yashunin (2016). "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs." In: *arXiv preprint arXiv:1603.09320*.

Marsden, Jerrold E and Michael J Hoffman (1993). *Elementary classical analysis*. Macmillan.

Martens, James (2010). "Deep learning via Hessian-free optimization." In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 735–742.

Martens, James and Roger Grosse (2015). "Optimizing neural networks with Kronecker-factored approximate curvature." In: *International Conference on Machine Learning*, pp. 2408–2417.

Martin Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: http://tensorflow.org/.

Mathieu, Michael, Camille Couprie, and Yann LeCun (2016). "Deep multi-scale video prediction beyond mean square error." In: *ICLR*.

Maturana, Daniel and Sebastian Scherer (2015). "Voxnet: A 3d convolutional neural network for real-time object recognition." In: *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, pp. 922–928.

McDonagh, James L., Neetika Nath, Luna De Ferrari, Tanja van Mourik, and John B. O. Mitchell (2014). "Uniting Cheminformatics and Chemical Theory To Predict the Intrinsic Aqueous Solubility of Crystalline Druglike Molecules." In: *Journal of Chemical Information and Modeling* 54.3, pp. 844–856.

Melis, Gábor, Chris Dyer, and Phil Blunsom (2017). "On the state of the art of evaluation in neural language models." In: *arXiv preprint arXiv:1707.05589*.

Mengersen, Kerrie L, Richard L Tweedie, et al. (1996). "Rates of convergence of the Hastings and Metropolis algorithms." In: *The Annals of Statistics* 24.1, pp. 101–121.

Micchelli, Charles A (1984). "Interpolation of scattered data: distance matrices and conditionally positive definite functions." In: *Approximation theory and spline functions*. Springer, pp. 143–145.

Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur (2010). "Recurrent neural network based language model." In: *Interspeech*. Vol. 2, p. 3.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013). "Distributed representations of words and phrases and their compositionality." In: *Advances in neural information processing systems*, pp. 3111–3119.

Mikolov, Tomas, Wen-tau Yih, and Geoffrey Zweig (June 2013). "Linguistic Regularities in Continuous Space Word Representations." In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, pp. 746–751. URL: http://www.aclweb.org/anthology/N13-1090.

Miller, Alexander H., Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston (2016). "Key-Value Memory Networks for Directly Reading Documents." In: *In EMNLP*.

Miller, George A. (Nov. 1995). "WordNet: A Lexical Database for English." In: *Commun. ACM* 38.11, pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: http://doi.acm.org/10.1145/219717.219748.

Mimno, David, Matt Hoffman, and David Blei (July 2012). "Sparse stochastic inference for latent Dirichlet allocation." In: *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*. Ed. by John Langford and Joelle Pineau. ICML '12. Edinburgh, Scotland, GB: Omnipress, pp. 1599–1606. ISBN: 978-1-4503-1285-1.

Mirza, Mehdi and Simon Osindero (2014). "Conditional generative adversarial nets." In: *arXiv preprint arXiv:1411.1784*.

Moody, Christopher E (2016). "Mixing Dirichlet Topic Models and Word Embeddings to Make lda2vec." In: *arXiv preprint arXiv:1605.02019*.

Moore, Andrew W (1999). "Very fast EM-based mixture model clustering using multiresolution kd-trees." In: *Advances in Neural information processing systems*, pp. 543–549.

Morris, Robert (Oct. 1978). "Counting Large Numbers of Events in Small Registers." In: *Commun. ACM* 21.10, pp. 840–842. ISSN: 0001-0782. DOI: 10.1145/359619.359627.

Mroueh, Youssef, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng (2017). "Sobolev GAN." In: *arXiv preprint arXiv:1711.04894*.

Mroueh, Youssef and Tom Sercu (2017). "Fisher GAN." In: *arXiv:1705.09675 NIPS*.

Muandet, K., D. Balduzzi, and B. Schoelkopf (2013). "Domain Generalization via Invariant Feature Representation." In: *In Proceeding of the 30th International Conference on Machine Learning (ICML 2013)*.

Muandet, K., K. Fukumizu, F. Dinuzzo, and B. Schoelkopf (2012). "Learning from Distributions via Support Measure Machines." In: *In Proceeding of the 26th Annual Conference on Neural Information Processing Systems (NIPS 2012)*.

Murphy, Kevin P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Mussmann, Stephen and Stefano Ermon (2016). "Learning and Inference via Maximum Inner Product Search." In: *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2587–2596.

Neal, R. (1998). *Markov chain sampling methods for Dirichlet process mixture models*. Tech. rep. 9815. University of Toronto.

Neal, Radford M and Geoffrey E Hinton (1998). "A view of the EM algorithm that justifies incremental, sparse, and other variants." In: *Learning in graphical models*. Springer, pp. 355–368.

Neelakantan, Arvind, Quoc V Le, and Ilya Sutskever (2015). "Neural programmer: Inducing latent programs with gradient descent." In: *arXiv preprint arXiv:1511.04834*.

Neelakantan, Arvind, Benjamin Roth, and Andrew McCallum (2015). "Compositional Vector Space Models for Knowledge Base Completion." In: *ACL*.

Neelakantan, Arvind, Jeevan Shankar, Alexandre Passos, and Andrew McCallum (2014). "Efficient Non-parametric Estimation of Multiple Embeddings per Word in Vector Space." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*.

Nesterov, Yurii (2004). *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media.

Nesterov, Yurii and Boris T Polyak (2006). "Cubic regularization of Newton method and its global performance." In: *Mathematical Programming* 108.1, pp. 177–205.

Networking Index, Cisco Visual (2016). "Forecast and methodology, 2016-2021, white paper." In: *San Jose, CA, USA*.

Neumann, A. U. and B. Derrida (Aug. 1988). "Finite size scaling study of dynamical phase transitions in two dimensional models: Ferromagnet, symmetric and non symmetric spin glasses." In: *J. Phys. France* 49, pp. 1647–1656.

Newman, David, Arthur Asuncion, Padhraic Smyth, and Max Welling (Dec. 2009). "Distributed Algorithms for Topic Models." In: *J. Machine Learning Research* 10. http://dl.acm.org/citation.cfm?id=1577069.1755845, pp. 1801–1828. ISSN: 1532-4435.

Newman, David, Sarvnaz Karimi, and Lawrence Cavedon (Dec. 2009). "External Evaluation of Topic Models." In: pp. 11–18.

Ngai, Eric WT, Yong Hu, YH Wong, Yijun Chen, and Xin Sun (2011). "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature." In: *Decision Support Systems* 50.3, pp. 559–569.

Nguyen, Tri, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng (2016). "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset." In: *CoRR* abs/1611.09268.

Nielsen, Søren Feodor (2000). "The stochastic EM algorithm: estimation and asymptotic results." In: *Bernoulli*, pp. 457–489.

Ntampaka, M., H. Trac, D. Sutherland, S. Fromenteau, B. Poczos, and J. Schneider (2016). "Dynamical Mass Measurements of Contaminated Galaxy Clusters Using Machine Learning." In: *The Astrophysical Journal*. URL: http://arxiv.org/abs/1509.05409.

Oliva, J., B. Poczos, and J. Schneider (2013). "Distribution to Distribution Regression." In: *International Conference on Machine Learning (ICML)*.

Oliva, Junier B, Avinava Dubey, Barnabás Póczos, Jeff Schneider, and Eric P Xing (2018). "Transformation Autoregressive Networks." In: *Proceedings of the International Conference on Machine Learning*.

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). "Pixel recurrent neural networks." In: *arXiv preprint arXiv:1601.06759*.

Owen, Mark (2007). *Practical signal processing*. Cambridge university press.

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (1999). *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab.

Pai, Ping-Feng and Chih-Sheng Lin (2005). "A hybrid ARIMA and support vector machines model in stock price forecasting." In: *Omega* 33.6, pp. 497–505.

Paisley, John, Chong Wang, David Blei, and Michael I Jordan (2013). "A Nested HDP for Hierarchical Topic Models." In: *arXiv preprint arXiv:1301.3570*.

Patterson, Sam and Yee Whye Teh (2013). "Stochastic gradient Riemannian Langevin dynamics on the probability simplex." In: *Advances in Neural Information Processing Systems*, pp. 3102–3110.

Patwary, Md Mostofa Ali, Nadathur Rajagopalan Satish, Narayanan Sundaram, Jialin Liu, Peter Sadowski, Evan Racah, Suren Byna, Craig Tull, Wahid Bhimji, Pradeep Dubey, et al. (2016). "PANDA: Extreme Scale Parallel K-Nearest Neighbor on Distributed Architectures." In: *Parallel and Distributed Processing Symposium, 2016 IEEE International*. IEEE, pp. 494–503.

Pearlmutter, Barak A. (Jan. 1994). "Fast Exact Multiplication by the Hessian." In: *Neural Computation* 6.1, pp. 147–160. ISSN: 0899-7667.

Peng, Xiaojiang, Changqing Zou, Yu Qiao, and Qiang Peng (2014). "Action recognition with stacked fisher vectors." In: *ECCV*.

Peyré, Gabriel, Marco Cuturi, et al. (2017). *Computational Optimal Transport*. Tech. rep.

Pitman, Jim (1995). "Exchangeable and partially exchangeable random partitions." In: *Probability theory and related fields* 102.2, pp. 145–158.

Pitman, Jim (2006). *Combinatorial Stochastic Processes: Ecole d'Eté de Probabilités de Saint-Flour XXXII-2002*. Springer.

Poczos, B., A. Rinaldo, A. Singh, and L. Wasserman (2013). "Distribution-free Distribution Regression." In: *International Conference on AI and Statistics (AISTATS)*. JMLR Workshop and Conference Proceedings.

Poczos, B., L. Xiong, D. Sutherland, and J. Schneider (2012). *Support Distribution Machines*. URL: http://arxiv.org/abs/1202.0302.

Póczos, Barnabás, Liang Xiong, and Jeff Schneider (2012). "Nonparametric divergence estimation with applications to machine learning on distributions." In: *arXiv preprint arXiv:1202.3758*.

Poljak, BT and Ya Z Tsypkin (1973). "Pseudogradient adaptation and training algorithms." In: *Automation and Remote Control* 34, pp. 45–67.

Pollefeys, Marc, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. (2008). "Detailed real-time urban 3d reconstruction from video." In: *International Journal of Computer Vision* 78.2, pp. 143–167.

Pritchard, Jonathan K., Matthew Stephens, and Peter Donnelly (2000). "Inference of Population Structure Using Multilocus Genotype Data." In: *Genetics* 155.2, pp. 945–959. ISSN: 0016-6731. eprint: http://www.genetics.org/content/155/2/945.pdf. URL: http://www.genetics.org/content/155/2/945.

Radford, Alec, Luke Metz, and Soumith Chintala (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks." In: *arXiv preprint arXiv:1511.06434*.

Raison, Martin, Pierre-Emmanuel Mazaré, Rajarshi Das, and Antoine Bordes (2018). "Weaver: Deep Co-Encoding of Questions and Documents for Machine Reading." In: *arXiv preprint arXiv:1804.10490*.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang (2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Text." In: *EMNLP*. Austin, Texas.

Ram, Parikshit and Alexander G Gray (2012). "Maximum inner-product search using cone trees." In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 931–939.

Ramakrishnan, Raghunathan, Pavlo Dral, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld (June 2017). *Quantum chemistry structures and properties of 134 kilo molecules*. URL: https://figshare.com/collections/Quantum_chemistry_structures_and_properties_of_134_kilo_molecules/978904/4.

Rappe, A. K., C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff (1992). "UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations." In: *Journal of the American Chemical Society* 114.25, pp. 10024–10035.

Rauch, Herbert E., F. Tung, and Charlotte T. Striebel (1965). "Maximum likelihood estimates of linear dynamic systems." In: *AIAA Journal*. ISSN: 0001-1452.

Ravanbakhsh, Siamak, Junier Oliva, Sebastien Fromenteau, Layne C Price, Shirley Ho, Jeff Schneider, and Barnabás Póczos (2016). "Estimating Cosmological Parameters from the Dark Matter Distribution." In: *Proceedings of The 33rd International Conference on Machine Learning*.

Ravanbakhsh, Siamak, Jeff Schneider, and Barnabás Póczos (2017). "Equivariance Through Parameter-Sharing." In: *International Conference on Machine Learning*, pp. 2892–2901.

Recht, B., C. Re, S.J. Wright, and F. Niu (2011). "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent." In: *Advances in Neural Information Processing Systems 24*, pp. 693–701. URL: http://books.nips.cc/nips24.html.

Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alexander J. Smola (2016). "Stochastic Variance Reduction for Nonconvex Optimization." In: *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 314–323.

Reddi, Sashank J., Suvrit Sra, Barnabás Póczos, and Alexander J. Smola (2016). "Fast Stochastic Methods for Nonsmooth Nonconvex Optimization." In: *CoRR*.

Reddi, Sashank J, Suvrit Sra, Barnabás Póczos, and Alex Smola (2016). "Fast incremental method for smooth nonconvex optimization." In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE, pp. 1971–1977.

Reddi, Sashank, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex J Smola (2015). "On Variance Reduction in Stochastic Gradient Descent and its Asynchronous Variants." In: *NIPS 28*, pp. 2629–2637.

Reddy, Siva, Mirella Lapata, and Mark Steedman (2014). "Large-scale semantic parsing without question-answer pairs." In: *TACL* 2. (Visited on 09/22/2015).

Reisinger, Joseph and Raymond J. Mooney (2010). "Multi-prototype Vector-space Models of Word Meaning." In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. HLT '10.

Rezende, Danilo Jimenez, Shakir Mohamed, and Daan Wierstra (2014). "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In: *ICML*.

Richardson, Iain E (2002). *Video codec design: developing image and video compression systems*. John Wiley & Sons.

Riedel, Sebastian, Limin Yao, Andrew McCallum, and Benjamin M Marlin (2013). "Relation extraction with matrix factorization and universal schemas." In: *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 74–84.

Robbins, Herbert and Sutton Monro (Sept. 1951). "A Stochastic Approximation Method." In: *Ann. Math. Statist.* 22.3, pp. 400–407. DOI: 10.1214/aoms/1177729586.

Robinson (May 2016). "The Thrill of Terrapattern, a New Way to Search Satellite Imagery." In: *The Atlantic*. URL: https://www.theatlantic.com/technology/archive/2016/05/the-promise-of-terrapattern-the-visual-search-engine-for-satellite-imagery/484610/.

Rosenthal, Jeffrey S (1995). "Convergence rates for Markov chains." In: *Siam Review* 37.3, pp. 387–405.

Roy, Daniel M. and Yee Whye Teh (2009). "The Mondrian Process." In: *Advances in Neural Information Processing Systems 23*.

Rozo, Eduardo and Eli S Rykoff (2014). "redMaPPer II: X-ray and SZ Performance Benchmarks for the SDSS Catalog." In: *The Astrophysical Journal* 783.2, p. 80.

Russakovsky, Olga et al. (2015). "ImageNet Large Scale Visual Recognition Challenge." In: *International Journal of Computer Vision (IJCV)* 115.3, pp. 211–252.

Rusu, Radu Bogdan and Steve Cousins (2011). "3D is here: Point Cloud Library (PCL)." In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

Ryan (May 2016). "This Website Lets You Find The Hidden Similarities In Big Cities." In: *Popular Science*. URL: https://www.popsci.com/you-can-now-search-google-maps-by-matching-similar-landscapes.

Ryu, Pum-Mo, Myung-Gil Jang, and Hyun-Ki Kim (2014). "Open domain question answering using Wikipedia-based knowledge model." In: *Information Processing and Management* 50.5, pp. 683–692. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2014.04.007. URL: http://www.sciencedirect.com/science/article/pii/S0306457314000351.

Sachan, Devendra Singh, Manzil Zaheer, and Ruslan Salakhutdinov (2018). "Investigating the Working of Text Classifiers." In: *Proceedings of the 28th International Conference on Computational Linguistics: Technical Papers*.

Sachan, Devendra Singh, Manzil Zaheer, and Ruslan Salakhutdinov (2019). "Revisiting LSTM Networks for Semi-Supervised Text Classification via Mixed Objective Function." In:

Salakhutdinov, Ruslan, Sam Roweis, and Zoubin Ghahramani (n.d.). *Relationship between gradient and EM steps in latent variable models*.

Salimans, Tim, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen (2016). "Improved techniques for training gans." In: *Advances in Neural Information Processing Systems*, pp. 2234–2242.

Savenkov, Denis and Eugene Agichtein (2016). "When a knowledge base is not enough: Question answering over knowledge bases with external text data." In: *SIGIR*. ACM.

Scarselli, Franco, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini (2009). "The graph neural network model." In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80.

Schlichtkrull, Michael, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling (2017). "Modeling Relational Data with Graph Convolutional Networks." In: *arXiv preprint arXiv:1703.06103*.

Schmidt, Mark, Nicolas Le Roux, and Francis Bach (2017). "Minimizing Finite Sums with the Stochastic Average Gradient." In: *Mathematical Programming* 162.1-2, pp. 83–112.

Schön, Thomas B., Fredrik Lindsten, Johan Dahlin, Johan Wågberg, Christian A. Naesseth, Andreas Svensson, and Liang Dai (2015). "Sequential Monte Carlo Methods for System Identification." In: *IFAC Symposium on System Identification (SYSID)*.

Seo, Minjoon, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi (2017). "Bidirectional attention flow for machine comprehension." In: *ICLR*.

Seo, Minjoon, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi (2016). "Query-Reduction Networks for Question Answering." In: *arXiv preprint arXiv:1606.04582*.

Sethuraman, Jayaram (1994). "A constructive definition of Dirichlet priors." In: *Statistica Sinica* 4.2, pp. 639–650. ISSN: 10170405, 19968507. URL: http://www.jstor.org/stable/24305538.

Shalev-Shwartz, Shai and Tong Zhang (2013). "Stochastic dual coordinate ascent methods for regularized loss." In: *The Journal of Machine Learning Research* 14.1, pp. 567–599.

Shao, Hang, Abhishek Kumar, and P Thomas Fletcher (2017). "The Riemannian Geometry of Deep Generative Models." In: *arXiv preprint arXiv:1711.08014*.

Shen, Yelong, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen (2017). "Reasonet: Learning to stop reading in machine comprehension." In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, pp. 1047–1055.

Shi, Baoguang, Song Bai, Zhichao Zhou, and Xiang Bai (2015). "Deeppano: Deep panoramic representation for 3-d shape recognition." In: *IEEE Signal Processing Letters* 22.12, pp. 2339–2343.

Shi, Yemin, Yonghong Tian, Yaowei Wang, Wei Zeng, and Tiejun Huang (2017). "Learning Long-Term Dependencies for Action Recognition With a Biologically-Inspired Deep Network." In: *ICCV*.

Shin, Seung Back and Yong Hun Kim (2014). *Cloud Face*. http://ssbkyh.com/works/cloud_face. Accessed: 2018-08-15.

Sigurdsson, Gunnar A., Santosh Divvala, Ali Farhadi, and Abhinav Gupta (2017). "Asynchronous Temporal Fields for Action Recognition." In: *CVPR*.

Sigurdsson, Gunnar A, Gül Varol, Xiaolong Wang, Ali Farhadi, Ivan Laptev, and Abhinav Gupta (2016). "Hollywood in homes: Crowdsourcing data collection for activity understanding." In: *ECCV*.

Simonyan, Karen and Andrew Zisserman (2014). "Two-stream convolutional networks for action recognition in videos." In: *NIPS*.

Singh, Saransh, Farangis Ram, and Marc De Graef (2017). "Application of forward models to crystal orientation refinement." In: *Journal of Applied Crystallography* 50.6, pp. 1664–1676.

Singh, Shashank, Ananya Uppal, Boyue Li, Chun-Liang Li, Manzil Zaheer, and Barnabás Póczos (2018). "Nonparametric Density Estimation under Adversarial Losses." In: *arXiv preprint arXiv:1805.08836*.

Smola, Alexander and Shravan Narayanamurthy (Sept. 2010). "An Architecture for Parallel Topic Models." In: *Proc. VLDB Endowment* 3.1-2, pp. 703–710. ISSN: 2150-8097. DOI: 10.14778/1920841.1920931.

Soomro, Khurram, Amir Roshan Zamir, and Mubarak Shah (2012). "UCF101: A dataset of 101 human actions classes from videos in the wild." In: *CRCV-TR-12-01*.

Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov (2015). "Unsupervised learning of video representations using lstms." In: *ICML*.

Stanovsky, Gabriel, Omer Levy, and Ido Dagan (2014). "Proposition Knowledge Graphs." In: *COLING 2014*.

Stewart, G. (1998). *Matrix Algorithms*. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9781611971408. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9781611971408. URL: http://epubs.siam.org/doi/abs/10.1137/1.9781611971408.

Stratonovich, Ruslan L. (1960). "Conditional markov processes." In: *Theory of Probability and Its Applications*.

Strobelt, Hendrik, Sebastian Gehrmann, Bernd Huber, Hanspeter Pfister, and Alexander M Rush (2016). "Visual analysis of hidden state dynamics in recurrent neural networks." In: *arXiv preprint arXiv:1606.07461*.

Strom, Johannes, Andrew Richardson, and Edwin Olson (2010). "Graph-based segmentation for colored 3D laser point clouds." In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, pp. 2131–2136.

Su, Hang, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller (2015). "Multi-view convolutional neural networks for 3d shape recognition." In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 945–953.

Suju, D Arul and Hancy Jose (2017). "FLANN: Fast approximate nearest neighbour search algorithm for elucidating human-wildlife conflicts in forest areas." In: *Signal Processing, Communication and Networking (ICSCN), 2017 Fourth International Conference on*. IEEE, pp. 1–6.

Sukhbaatar, Sainbayar, Rob Fergus, et al. (2016). "Learning multiagent communication with backpropagation." In: *Neural Information Processing Systems*, pp. 2244–2252.

Sukhbaatar, Sainbayar, Arthur Szlam, Jason Weston, and Rob Fergus (2015). "End-To-End Memory Networks." In: *NIPS*.

Sukmarg, Orachat and Kamisetty R Rao (2000). "Fast object detection and segmentation in MPEG compressed domain." In: *TENCON 2000. Proceedings*. Vol. 3. IEEE, pp. 364–368.

Sun, Haitian, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen (2018). "Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text." In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4231–4242.

Sun, Huan, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang (2015). "Open domain question answering via semantic enrichment." In: *WWW*. ACM.

Sun, Lin, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E. Shi, and Silvio Savarese (2017). "Lattice Long Short-Term Memory for Human Action Recognition." In: *ICCV*.

Sundermeyer, Martin, Ralf Schlüter, and Hermann Ney (2012). "LSTM Neural Networks for Language Modeling." In: *Interspeech*, pp. 194–197.

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton (2013). "On the importance of initialization and momentum in deep learning." In: *International conference on machine learning*, pp. 1139–1147.

Szabo, Z., B. Sriperumbudur, B. Poczos, and A. Gretton (2016). "Learning Theory for Distribution Regression." In: *Journal of Machine Learning Research*.

Talmor, A. and J. Berant (2018). "The Web as a Knowledge-base for Answering Complex Questions." In: *North American Association for Computational Linguistics (NAACL)*.

Tan, Yong Kiam, Xinxing Xu, and Yong Liu (2016). "Improved recurrent neural networks for session-based recommendations." In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, pp. 17–22.

Taskar, Ben, Carlos Guestrin, and Daphne Koller (2004). "Max-margin Markov networks." In: *Advances in neural information processing systems*, pp. 25–32.

Teh, Whye Yee, David Newman, and Max Welling (2007). "A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation." In: *Advances in Neural Information Processing Systems 19*. NIPS 2006. Vancouver, Canada: MIT Press, pp. 1353–1360. ISBN: 9780262195683.

Teh, Yee Whye, Michael I Jordan, Matthew J Beal, and David M Blei (2006). "Hierarchical Dirichlet processes." In: *Journal of the American Statistical Association* 101.476, pp. 1566–1581.

Tian, Fei, Bin Gao, and Tie-Yan Liu (2016). "Sentence Level Recurrent Topic Model: Letting Topics Speak for Themselves." In: *arXiv preprint arXiv:1604.02038*.

Tierney, Luke (1994). "Markov chains for exploring posterior distributions." In: *the Annals of Statistics*, pp. 1701–1728.

Tong, Hanghang, Christos Faloutsos, and Jia-Yu Pan (2008). "Random walk with restart: fast solutions and applications." In: *Knowledge and Information Systems* 14.3, pp. 327–346.

Töreyin, B Ugur, A Enis Cetin, Anil Aksay, and M Bilgay Akhan (2005). "Moving object detection in wavelet compressed video." In: *Signal Processing: Image Communication* 20.3, pp. 255–264.

Toutanova, Kristina, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon (2015). "Representing text for joint embedding of text and knowledge bases." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1499–1509.

Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer (2003). "Feature-rich Part-of-speech Tagging with a Cyclic Dependency Network." In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. NAACL '03. Edmonton, Canada: Association

for Computational Linguistics, pp. 173–180. DOI: 10.3115/1073445.1073478. URL: http://dx.doi.org/10.3115/1073445.1073478.

Tran, Du, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri (2015). "Learning spatiotemporal features with 3d convolutional networks." In: *ICCV*.

Tran, Du, Jamie Ray, Zheng Shou, Shih-Fu Chang, and Manohar Paluri (2017). "ConvNet Architecture Search for Spatiotemporal Feature Learning." In: *arXiv preprint arXiv:1708.05038*.

Trischler, Adam, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman (2016). "NewsQA: A Machine Comprehension Dataset." In: *CoRR* abs/1611.09830.

Tristan, Jean-Baptiste, Joseph Tassarotti, and Guy L. Steele Jr. (2015). "Efficient Training of LDA on a GPU by Mean-For-Mode Gibbs Sampling." In: *32nd International Conference on Machine Learning*. Vol. 37. ICML 2015. Volume 37 of the Journal in Machine Learning Research: Workshop and Conference Proceedings.

Tsybakov, Alexandre B (2009). *Introduction to nonparametric estimation*. Springer Series in Statistics. Springer, New York.

Tung, Hsiao-Yu Fish, Chao-Yuan Wu, Manzil Zaheer, and Alexander J Smola (2017). "Spectral Methods for Nonparametric Models." In: *arXiv preprint arXiv:1704.00003*.

Turian, Joseph, Lev Ratinov, and Yoshua Bengio (2010). "Word representations: a simple and general method for semi-supervised learning." In: *Proc. of ACL*.

Turney, Peter D. (Sept. 2006). "Similarity of Semantic Relations." In: *Comput. Linguist.* 32.3, pp. 379–416. ISSN: 0891-2017. DOI: 10.1162/coli.2006.32.3.379. URL: http://dx.doi.org/10.1162/coli.2006.32.3.379.

Turney, Peter D. and Patrick Pantel (2010). "From frequency to meaning : Vector space models of semantics." In: *JAIR*, pp. 141–188.

Van Den Oord, Aaron, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu (2016). "Wavenet: A generative model for raw audio." In: *arXiv preprint arXiv:1609.03499*.

Verga, Patrick, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum (2016). "Multilingual relation extraction using compositional universal schema." In: *NAACL*.

Vichniac, Gérard Y. (Jan. 1984). "Simulating Physics with Cellular Automata." In: *Physica D: Nonlinear Phenomena* 10.1-2, pp. 96–116. ISSN: 0167-2789. DOI: 10.1016/0167-2789(84)90253-7.

Vinyals, Oriol, Samy Bengio, and Manjunath Kudlur (2015). "Order matters: Sequence to sequence for sets." In: *arXiv preprint arXiv:1511.06391*.

Vinyals, Oriol and Daniel Povey (2012). "Krylov Subspace Descent for Deep Learning." In: *AISTATS*, pp. 1261–1268.

Von Ahn, Luis and Laura Dabbish (2004). "Labeling images with a computer game." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 319–326.

Voorhees, Ellen M et al. (1999). "The TREC-8 Question Answering Track Report." In: *Trec*. Vol. 99, pp. 77–82.

Vose, Michael D (1991). "A linear algorithm for generating random numbers with a given distribution." In: *Software Engineering, IEEE Transactions on* 17.9, pp. 972–975.

Wainwright, Martin J and Michael I Jordan (2008). "Graphical models, exponential families, and variational inference." In: *Foundations and Trends® in Machine Learning* 1.1-2, pp. 1–305.

Walker, Alastair J (1977). "An efficient method for generating discrete random variables with general distributions." In: *ACM Transactions on Mathematical Software (TOMS)* 3.3, pp. 253–256.

Wan, Li, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus (2013). "Regularization of neural networks using dropconnect." In: *International Conference on Machine Learning*, pp. 1058–1066.

Wan, Li, Leo Zhu, and Rob Fergus (2012). "A Hybrid Neural Network-Latent Topic Model." In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*. Ed. by Neil D. Lawrence and Mark A. Girolami. Vol. 22, pp. 1287–1294. URL: http://jmlr.csail.mit.edu/proceedings/papers/v22/wan12/wan12.pdf.

Wang, Fa, Manzil Zaheer, Xin Li, Jean-Olivier Plouchart, and Alberto Valdes-Garcia (2015). "Co-learning Bayesian model fusion: efficient performance modeling of analog and mixed-signal circuits using side information." In: *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. IEEE Press, pp. 575–582.

Wang, Heng, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu (2013). "Dense trajectories and motion boundary descriptors for action recognition." In: *IJCV* 103.1, pp. 60–79.

Wang, Heng and Cordelia Schmid (2013). "Action recognition with improved trajectories." In: *ICCV*.

Wang, Heng, Muhammad Muneeb Ullah, Alexander Klaser, Ivan Laptev, and Cordelia Schmid (2009). "Evaluation of local spatio-temporal features for action recognition." In: *BMVC*.

Wang, Limin, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool (2016). "Temporal segment networks: Towards good practices for deep action recognition." In: *ECCV*.

Wang, Qing, Sanjeev R Kulkarni, and Sergio Verdú (2009). "Divergence estimation for multidimensional densities via k-nearest-neighbor distances." In: *IEEE Transactions on Information Theory* 55.5, pp. 2392–2405.

Wang, Shuohang and Jing Jiang (2016). "Machine comprehension using match-lstm and answer pointer." In: *arXiv preprint arXiv:1608.07905*.

Wang, Shuohang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang (2018). "R^3: Reinforced Reader-Ranker for Open-Domain Question Answering." In: *AAAI*.

Wang, Shuohang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell (2017). "Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering." In: *arXiv preprint arXiv:1711.05116*.

Wang, Yi, Xuemin Zhao, Zhenlong Sun, Hao Yan, Lifeng Wang, Zhihui Jin, Liubin Wang, Yang Gao, Jia Zeng, Qiang Yang, et al. (2014). "Peacock: Learning Long-Tail Topic Features for Industrial Applications." In: *arXiv preprint arXiv:1405.4402*.

Wang, Yunbo, Mingsheng Long, Jianmin Wang, and Philip S Yu (2017). "Spatiotemporal Pyramid Network for Video Action Recognition." In: *CVPR*.

Wasserman, Larry (2006). *All of Nonparametric Statistics*. Springer Science & Business Media.

Watanabe, Yusuke, Bhuwan Dhingra, and Ruslan Salakhutdinov (2017). "Question Answering from Unstructured Text by Retrieval and Comprehension." In: *arXiv preprint arXiv:1703.08885*.

Welbl, Johannes, Pontus Stenetorp, and Sebastian Riedel (2018). "Constructing Datasets for Multi-hop Reading Comprehension Across Documents." In: *TACL*.

Weston, Jason, Sumit Chopra, and Antoine Bordes (2015). "Memory Networks." In: *In ICLR*.

Wiese, Georg, Dirk Weissenborn, and Mariana Neves (Aug. 2017). "Neural Domain Adaptation for Biomedical Question Answering." In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, pp. 281–289.

Woodbury, Max A, Jonathan Clive, and Arthur Garson (1978). "Mathematical typology: a grade of membership technique for obtaining disease definition." In: *Computers and biomedical research* 11.3, pp. 277–298.

Wu, Chao-Yuan, Manzil Zaheer, Hexiang Hu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl (2018). "Compressed video action recognition." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6026–6035.

Wu, Jiajun, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum (2016). "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling." In: *arXiv preprint arXiv:1610.07584*.

Wu, Zhirong, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao (2015). "3d shapenets: A deep representation for volumetric shapes." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1912–1920.

Xingjian, Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo (2015). "Convolutional LSTM network: A machine learning approach for precipitation nowcasting." In: *NIPS*.

Xiong, Caiming, Victor Zhong, and Richard Socher (2016). "Dynamic Coattention Networks For Question Answering." In: *arXiv preprint arXiv:1611.01604*.

Xiong, Wayne, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig (2016). "Achieving human parity in conversational speech recognition." In: *arXiv preprint arXiv:1610.05256*.

Xu, Kelvin, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio (2015). "Show, attend and tell: Neural image caption generation with visual attention." In: *International Conference on Machine Learning*, pp. 2048–2057.

Xu, Kun, Yansong Feng, Songfang Huang, and Dongyan Zhao (2016). "Hybrid Question Answering over Knowledge Base and Free Text." In: *COLING*.

Xu, Kun, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao (2016). "Question Answering on Freebase via Relation Extraction and Textual Evidence." In: *ACL*.

Xu, Lei and Michael I Jordan (1996). "On convergence properties of the EM algorithm for Gaussian mixtures." In: *Neural computation* 8.1, pp. 129–151.

Yahya, Mohamed, Denilson Barbosa, Klaus Berberich, Qiuyue Wang, and Gerhard Weikum (2016). "Relationship queries on extended knowledge graphs." In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, pp. 605–614.

Yang, Yaoqing, Chen Feng, Yiru Shen, and Dong Tian (2018). "FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation." In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Vol. 3.

Yao, Limin, David Mimno, and Andrew McCallum (2009). "Efficient Methods for Topic Model Inference on Streaming Document Collections." In: *Proc. 15th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*. KDD '09. Paris, France: ACM, pp. 937–946. ISBN: 978-1-60558-495-9. DOI: 10.1145/1557019.1557121.

Yao, Xuchen and Benjamin Van Durme (June 2014). "Information Extraction over Structured Data: Question Answering with Freebase." In: *ACL*.

Yeo, Boon-Lock and Bede Liu (1995). "Rapid scene analysis on compressed video." In: *IEEE Transactions on circuits and systems for video technology* 5.6, pp. 533–544.

Yih, Wen-tau, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao (July 2015). "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base." In: *ACL*.

Yih, Wen-tau, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh (2016). "The value of semantic parse labeling for knowledge base question answering." In: *Proceedings*

of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). Vol. 2, pp. 201–206.

Yu, Adams Wei, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le (2018). "QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension." In: *ICLR*.

Yu, Hsiang-Fu, Cho-Jui Hsieh, Hyokun Yun, SVN Vishwanathan, and Inderjit S Dhillon (2015). "A Scalable Asynchronous Distributed Algorithm for Topic Modeling." In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 1340–1350.

Yu, Mo, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou (2017). "Improved neural relation detection for knowledge base question answering." In: *ACL*.

Yuan, Jinhui, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma (2015). "LightLDA: Big Topic Models on Modest Computer Clusters." In: *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, pp. 1351–1361.

Yue-Hei Ng, Joe, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici (2015). "Beyond short snippets: Deep networks for video classification." In: *CVPR*.

Zach, Christopher, Thomas Pock, and Horst Bischof (2007). "A duality based approach for realtime TV-L 1 optical flow." In: *Pattern Recognition*, pp. 214–223.

Zaheer, Manzil, Amr Ahmed, and Alexander J Smola (2017). "Latent LSTM Allocation: Joint Clustering and Non-Linear Dynamic Modeling of Sequence Data." In: *Proceedings of the International Conference on Machine Learning*, pp. 3967–3976.

Zaheer, Manzil, Amr Ahmed, Yuan Wang, Daniel Silva, and Yuchen Wu (2018). "Uncovering Hidden Structure in Sequence Data via Threading Recurrent Models." In: *arXiv preprint*.

Zaheer, Manzil, Rajarshi Das, and Chris Dyer (2015). "Gaussian lda for topic models with word embeddings." In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pp. 795–804.

Zaheer, Manzil, Rajarshi Das, Siva Reddy, and Andrew McCallum (2017). "Question Answering on Knowledge Bases and Text using Universal Schema and Memory Networks." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pp. 358–365.

Zaheer, Manzil, Guru Guruganesh, Golan Levin, and Alexander Smola (2018). "TerraPattern: A Nearest Neighbor Service." In: *arXiv preprint*.

Zaheer, Manzil, Satwik Kottur, Amr Ahmed, José Moura, and Alex Smola (2017). "Canopy Fast Sampling with Cover Trees." In: *Proceedings of the International Conference on Machine Learning*, pp. 3977–3986.

Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola (2017). "Deep Sets." In: *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3391–3401.

Zaheer, Manzil, Jay Lee, Stephan Günnemann, and Alex Smola (2015). "Preferential Attachment in Graphs with Affinities." In: *Proceedings of the Artificial Intelligence and Statistics*, pp. 571–580.

Zaheer, Manzil, Chun-Liang Li, Yang Zhang, Ruslan Salakhutdinov, and Barnabás Póczos (2018). "Point Cloud GAN." In: *arXiv preprint*.

Zaheer, Manzil, Xin Li, and Chenjie Gu (2014). "MPME-DP: Multi-population moment estimation via dirichlet process for efficient validation of analog/mixed-signal circuits." In:

Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design. IEEE Press, pp. 316–323.

Zaheer, Manzil, Sashank Reddi, Suvrit Sra, Barnabas Poczos, Francis Bach, Ruslan Salakhutdinov, and Alex Smola (2018). "A Generic Approach for Escaping Saddle points." In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1233–1242.

Zaheer, Manzil and Alexander J Smola (2018). "Fast Sampling Algorithms for Sparse Latent Variable Models." In: *arXiv preprint*.

Zaheer, Manzil, Fa Wang, Chenjie Gu, and Xin Li (2015). "mTunes: Efficient post-silicon tuning of mixed-signal/RF integrated circuits based on Markov decision process." In: *Proceedings of the 52nd Annual Design Automation Conference*. ACM, pp. 170–176.

Zaheer, Manzil, Michael Wick, Jean-Baptiste Tristan, Alex Smola, and Guy L Steele Jr (2015). "Exponential stochastic cellular automata for massively parallel inference." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics 2016*, pp. 966–975.

Zelle, John M and Raymond J Mooney (1996). "Learning to parse database queries using inductive logic programming." In: *AAAI*. Portland, Oregon.

Zettlemoyer, Luke S. and Michael Collins (2005). "Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars." In: *UAI*. Edinburgh, Scotland.

Zhai, Ke and Jordan L. Boyd-Graber (2013). "Online Latent Dirichlet Allocation with Infinite Vocabulary." In: *ICML (1)*. Vol. 28. JMLR Proceedings. JMLR.org, pp. 561–569. URL: http://dblp.uni-trier.de/db/conf/icml/icml2013.html#ZhaiB13.

Zhai, Ke, Jordan Boyd-Graber, Nima Asadi, and Mohamad L Alkhouja (2012). "Mr. LDA: A flexible large scale topic modeling package using variational inference in mapreduce." In: *Proceedings of the 21st international conference on World Wide Web*. ACM, pp. 879–888.

Zhang, Bowen, Limin Wang, Zhe Wang, Yu Qiao, and Hanli Wang (2016). "Real-time action recognition with enhanced motion vector CNNs." In: *CVPR*.

Zhang, Wei (1988). "Shift-invariant pattern recognition neural network and its optical architecture." In: *Proceedings of annual conference of the Japan Society of Applied Physics*.

Zheng, Xun, Manzil Zaheer, Amr Ahmed, Yuan Wang, Eric P Xing, and Alexander J Smola (2017). "State space LSTM models with particle MCMC inference." In: *arxiv preprint arxiv:1711.11179*.

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A Efros (2017). "Unpaired image-to-image translation using cycle-consistent adversarial networks." In: *arXiv preprint arXiv:1703.10593*.