Learning Generative Models using Transformations

Chun-Liang Li

August 2019
CMU-ML-19-115

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

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

This research was supported by the Department of Energy award DESC0011114, Air Force Research Laboratory award FA87501420244, National Science Foundation award IIS1563887, a grant from Foxconn Technology Group and a fellowship from IBM.

Keywords: Generative Models, Sampling, Deep Learning, Adversarial Networks, Adversarial Learning, Kernel Learning, Two-Sample Test

For my family

Abstract

One of the fundamental problems in machine learning and statistics is learning generative models of data. Explicit generative models, which model probability densities of data, have been intensively studied in numerous applications. However, it is usually difficult to model complex data, such as natural images, by using combinations of simple parametric distributions. *Implicit generative models* (IGMs), which model transformations between known source distributions and target distributions to simulate the sampling process without specifying densities explicitly, regain its attention with explosion of interests. With recent success of deep learning, IGMs have yielded impressive empirical performance in different applications.

While there are new algorithms for learning IGMs, its theoretical properties are weakly justified and their relationships with existing methods are underexplored. The first thrust of this thesis is to understand statistical guarantees of learning IGMs. By connecting IGMs with two-sample test, we propose a new generic algorithm that can be built on the top of many existing approaches and bring performance improvement over the state-of-the-art. On the other hand, from the perspective of statistical analysis, IGMs, which model transformations, is fundamentally different from traditional explicit models, which makes the existing results not directly applicable. We then study error bounds and sample complexities of learning IGMs taking a step forward in building its rigorous foundations.

In the second part, we shift our focus to different types of data that we are interested in. We develop algorithms for learning IGMs on various data ranging from images, text, to point clouds, by exploiting their underlying structures. Instead of modeling IGM transformations blindly via powerful functions only, such as deep neural networks, we propose to leverage human priors into algorithm designs to reduce model sizes, save computational overhead, and achieve interpretable results. In this thesis, we show an example of incorporating a simple yet fairly representative renderer developed in computer graphics into IGM transformations for generating realistic and highly structured body data, which paves a new path of learning IGMs.

Finally, we study how IGMs can improve existing machine learning algorithms. From its nature of modeling sampling processes, we propose learning powerful kernels via Fourier analysis and IGM sampling. By thinking IGMs as learning transformations, we extend IGMs to broader applications in different domains. In the second example, we present how to learn proximal operators as IGM transformations to solve important linear inverse problems in computer vision. Lastly, we introduce a new way of using IGMs by treating them as auxiliary components to benefit non-generative tasks while the final output of the interest is not the generative models. We present an application of optimizing test power in anomaly detection by constructing a lower bound of test power via auxiliary IGMs.

Acknowledgments

The work presented in this thesis could not have been possible without the help of my advisors and mentors along this journey. First and foremost, I want to thank my advisor Barnabás Póczos for his support and guidance throughout the past 4.5 years. Barnabás is the kindest advisor a PhD student could have asked for. He fully supports every decision I made and gives me lots of freedom to grow as an independent researcher; he encourages me at every difficult moment of PhD and takes good care of me both academically and personally. I learned how to be both theoreticians and practitioners, and remember to enjoy the fun of doing research from him. Before Barnabás, I am fortunate enough to have Jeff Schneider as my advisor at the beginning of my PhD and as my thesis committee at the end of this journey. Jeff showed his patience to advice a fresh PhD and provided useful feedback to complete this thesis. I also want to thank my other thesis committee members, including Ruslan Salakhutdinov and Phillip Isola. I am benefited greatly from many discussions with Russ to gain excellent insights and creative ideas. The invaluable comments and suggestions from Phillip inspire many on-going research projects.

In addition to advisors, I am also grateful to have many mentors, which have significant impact on my research, shape my thinking and contribute to several chapters of this thesis. Firstly, I would like to thank Siamak Ravanbakhsh who brought me into the world of generative models. Secondly, I am indebted to my great mentors, collaborators and friends, Yu Cheng and Youssef Mroueh at IBM Watson. Yu Cheng inspired me the foundations of this thesis, while Youssef paved the way from theories to wider applications with me for this thesis. I deeply appreciate all of their unreserved help during my PhD even after my internship ended. I also want to thank them for encouraging and helping me in applying IBM fellowship. Under Tomas Simon and Jason Saragih's mentorship in Facebook Reality Labs, I learned a lot about how to do practical research. They pushed me to concentrate on important problems and introduced me to the research in geometry and graphics.

During my PhD, I had great pleasure to collaborate with brilliant researchers on a large spectrum of topics. My respectful list of those amazing folks are (in random order): Wei-Cheng Chang, Manzil Zaheer, Jen-Hao Chang, Kirthevasan Kandasamy, Po-Wei Wang, En-Hsu Yen, Hsueh-Ti Liu, Derek Nowrouzezahrai, Alec Jacobson, Michael Tao, Austin Dill, Songwei Ge, Eunsu Kang, François Lanusse, Yusha Liu, Rachel Mandelbaum, Anant Raj, Tom Sercu, Yaser Sheikh, Shashank Singh, Ananya Uppal, Chenghui Zhou, Lingyao Zhang, George Cai, and Yang Zhang. Many of the published works with them contribute to this thesis and I learned a lot from each of them in not only doing research, but also being a mentee, being a collaborator, and being a mentor.

Looking back to the very beginning of my research journey, I cannot forget to express my sincerest gratitude to my advisor at National Taiwan University, Hsuen-Tien Lin, who believed in my potential. I would not be at CMU today without him. His enthusiasm and ethics for research, teaching and life have immense impact to my life and make him a long-lasting role model for me. I am also fortunate to work with the other advisor before coming to CMU, Chi-Jen Lu, who taught me how to do theory research and be a humble person.

Special thanks to my officemates and Machine Learning Department for creating a supportive and comfortable working environment. I had several interesting discussions and conversations with my officemates, including Christoph Dann (for 5 years in two offices), Mrinmaya Sachan, Eric Lei, Ritesh Noothigattu, Eric Wong, Jacob Tyo, and Qizhe Xie. I also want to thank all the staff members, especially Diane Stidle, for taking good care of whole MLD, Sandy Winkler and Sharon Cavlovich for reimbursements.

I wish I could thank all of my friends and roommates over the years for all the help and friendship they gave to me from the moment I came to Pittsburgh to the moment I am leaving. Their companionship makes Pittsburgh a home away from home. I give my gratitude to my roommates, Tai-Ying Chen, Jui-Pin Wang, Po-Wei Wang and Hsueh-Ti Liu. My PhD journey would not have such a smooth start without those sport games, XBox games and movies with Tai-Ying and Jui-Pin. My second and third years were also wonderful with Po-Wei and Hsueh-Ti by having fun chats on research, NBA, beers and life in numerous nights. I must have to thank my gang in the second half of the PhD, including Olive Ho, Carol Cheng, Ting-Yao Hu, Hui-Chun Chen, Shih-En Wei, Shao-Wen Chiu, Yen-Chia Hsu and Rick Chang for our countless and unforgettable dinners, trips, birthday celebrations and driving skill practices. I also want to thank Yen-Chi Chen, Manzil Zaheer, Yu-Hsin Lin, Sz-Rung Shiang, Peter Chang, Chia-Yin Tsai, Yen-Chia Hsu and Rick Chang for having several meals and coffees together. I am specially thankful for being a part of Café Philo at Pittsburgh with Favonia, Wei Jeng, Hsu-Chieh Hu, Yun-Hao Hsiao and Li Chen to pursue my non-academic passions on societies and politics.

Finally and most importantly, I owe my deepest gratitude to my family. I thank my loving mom Hsiu-Feng Hou and dad Yu-Lin Li for their unconditional love. I particularly thank the best sister over the world, Ying-Chiu Lee for always being with me at every moment when I ever needed her in my life and her unconditional support in whatever I pursue, especially this PhD. Last but not the least, I dedicate this thesis to my best friend, partner in crime, life-long partner and my beloved wife, Olive Ho. Every chapter of this thesis happened after meeting her, with memories of us at different corners of the world. I could not have done this without her constant encouragement, unswerving support and endless love to go through every difficulty and challenge.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In machine learning and statistics, the heart of unsupervised learning lies in learning generative models that describes probabilistic distributions of data. Learning generative models has demonstrated success in various applications, including but not limited to image processing (Isola et al., 2017; Ledig et al., 2017), science simulation (Louppe and Cranmer, 2019), chemical design (Kusner et al., 2017), anomaly detection (Chang et al., 2019) and art creation (Elgammal et al., 2017). Learning good generative models also benefits other major tasks in machine learning, for example, data augmentation in supervised and semi-supervised learning (Salimans et al., 2016), data imputation in unsupervised learning (Yoon et al., 2018), planning (Finn and Levine, 2017) in reinforcement learning.

There are two categories of generative models, prescribed (explicit) and implicit models (Diggle and Gratton, 1984). *Explicit generative models* explicitly model densities of underlying distributions, which are widely studied in last decades. Take a coin-toss as an example. The outcome could be modeled by a Bernoulli distribution. In Bayesian statistics, graphical models (Bishop, 2006; Koller et al., 2009) are widely studied with different mixture and hierarchical structures, where each building component is modeled by some parametric distributions. However, using combinations of simple parametric distributions to model densities, does not have much success in modeling complex, structured and high dimensional data, such as natural images. In non-parametric statistics, kernel density estimation (Tsybakov, 2009; Wasserman, 2013) only assumes limited smoothness for modeling data density, but it usually suffers from the curse of dimensionality in practice.

In many applications, we are actually interested in sampling from the distribution instead of measuring the density. The other category of generative model is *implicit generative models* (IGMs), which models the *sampling process* of underlying distributions. In the coin-toss example, from physics, the outcome is determined by every initial conditions (e.g. environments, impulse, angles). The randomness does not originate from the Bernoulli distribution but from the randomness of those initializations (Diaconis et al., 2007; Keller, 1986). The physics system *transforms* these randomness into coin-toss outcomes by following the laws of physics. Inspired by the idea, IGMs simulate the sampling by using a **transformation** to map a prior randomness into the data of interest. Those generative models also have deep connections with other fields, including molecular biology (Pritchard et al., 1999), statistical physics (Anelli et al., 2008), and ecology (Beaumont, 2010). Using neural networks to model those transformations for (implicit) generative models can be dated back to Bishop et al. (1998); Diggle and Gratton (1984); MacKay (1995). It regains attentions because of recent breakthroughs in deep learning (Goodfellow et al., 2014; Kingma and Welling, 2013; Mohamed and Lakshminarayanan, 2016) and the successful applications aforementioned.

1.1 Implicit Generative Model: Generative Models using Transformations

In the aforementioned coin-toss example, we can treat the sampling process as a *transformation* process which transforms the randomness of environments into the outcome. Implicit Generative Models (IGMs) (MacKay, 1995; Mohamed and Lakshminarayanan, 2016) follow this idea to simulate the sampling process by modeling a deterministic function to transform an initial randomness such that the transformed samples follow the target distribution.

Formally, assume we are given data $\{x_i\}_{i=1}^n$, where $x_i \in \mathcal{X}$ and $x_i \sim \mathbb{P}_\mathcal{X}$, we are interested in sampling from $\mathbb{P}_\mathcal{X}$. Instead of estimating the density of $\mathbb{P}_\mathcal{X}$, IGMs train a generator g_θ parameterized by θ to *transform* samples $z \sim \mathbb{P}_\mathcal{Z}$, where $z \in \mathcal{Z}$ is a base distribution, into $g_\theta(z) \sim \mathbb{Q}_\theta$ such that $\mathbb{Q}_\theta \approx \mathbb{P}_\mathcal{X}$. The only requirement of $\mathbb{P}_\mathcal{Z}$ is we can easily draw samples from. We can roughly categorize $\mathbb{P}_\mathcal{Z}$ into three types.

- The first choice of $\mathbb{P}_\mathcal{Z}$ is a known parametric distribution, such as Gaussian and uniform distributions. In this case, we usually call $\mathbb{P}_\mathcal{Z}$ as *noise* or initial randomness. The transformation g_θ is mapping *noise into data*.
- In addition to using Gaussian distributions as $\mathbb{P}_\mathcal{Z}$, we can use domain-specific priors as base distributions to ease the difficulty of learning desirable transformations, which can be treated as transforming *priors into data*.
- To draw samples from $\mathbb{P}_\mathcal{Z}$, the density of $\mathbb{P}_\mathcal{Z}$ is not necessary required if we have enough many samples from $\mathbb{P}_\mathcal{Z}$. For example, in image translations, it only requires the access to the source images instead of knowing the image distributions. In this case, we learn a *data to data* transformation, which learns to adapt from one domain to the other.

In this thesis, we study generative models using different transformations for different source $\mathbb{P}_\mathcal{Z}$ and target $\mathbb{P}_\mathcal{X}$ pairs for various applications, by leveraging structures and priors of data with mathematical analysis.

1.1.1 Example: Probability Integral Transformation

Assume X is a continuous univariate random variable following a distribution $\mathbb{P}_\mathcal{X}$, where its CDF F_X is strictly increasing, we have $F_X^{-1}(Z) \sim \mathbb{P}_X$, where $Z \sim \mathbb{P}_\mathcal{Z}$ and $\mathbb{P}_\mathcal{Z}$ is an uniform distribution over $[0, 1]$ (Casella and Berger, 2002). With continuous and invertible assumptions on F_X, F_X^{-1} exists. Therefore, a general algorithm for sampling from a distribution $\mathbb{P}_\mathcal{X}$ can be done by

1. sampling from a uniform distribution Z and
2. applying the transformation F_X^{-1} on the samples.

The above example can be treated as an IGM, where $\mathbb{P}_{\mathcal{Z}}$ is the base distribution and F_X^{-1} is the desired transformation function. Although the analytical form of F_X^{-1} may not exist (e.g. Gaussian distribution), we can use powerful functions to approximate it numerically, such as neural networks (Zaheer et al., 2017b). We note that there are extensions of the probability integral transformation on none-strictly increasing CDF and multivariate random variables. We refer interested readers to Casella and Berger (2002); Genest and Rivest (2001).

1.1.2 Connections with Explicit Generative Models

We remark that although the idea of IGMs is focusing on learning transformation functions for sampling, IGMs are not conflict with modeling densities of distributions. In variational auto-encoder (VAE), the probability is assigned via a smoothing function on each sample (Kingma and Welling, 2013; Rezende et al., 2014). In flow-based models (Dinh et al., 2014, 2016; Kingma and Dhariwal, 2018), with an invertible transformation function, the probability can be derived via the change of variable theorem.

1.2 Our Goals

Although generative models have been extensively studied in the era of explicit models, by using powerful functions (e.g. deep neural networks) to model underlying transformations, IGMs lead to new research questions:

- With different designs and mechanisms, what are the algorithms for IGMs and what are the theoretical guarantees of learning sampling processes compared with traditional explicit models?
- With recent developments of neural networks, which provide powerful and expressive transformations for IGMs, what are the new applications?

This thesis studies learning implicit generative models in a broad spectrum as a first step toward solving new challenges. We focuses on advancing learning implicit generative models in the following directions:

- **Design via Mathematical Analysis:** The first goal of this thesis focuses on advancing the understanding of learning sampling via an IGM transformation from a statistical perspective to improve existing algorithms. In particular, we study the connection between modern deep IGMs, two-sample test, and kernel learning. It leads to a new generic IGM algorithm without much assumptions of data, MMD GAN, with theoretical guarantees and opens a new research direction of IGMs. We then dive deeper to study IGMs by conducting statistical analysis. We bound the sample complexities and errors under different smoothness assumptions, which provide a rigorous foundation of learning IGMs.

- **Leverage Underlying Structures and Priors of Data:** We study how to learn IGMs from different types of data and how to learn transformations between different modalities by leveraging our understanding of the *structures* of data, including the Markov assumption of languages and permutation invariance of 3D point clouds. On the other hand, in addition to taking structures in to account for algorithm designs, we explore how to encode human prior knowledges into IGMs with an explicit way for not only efficient and effective learning algorithms but also interpretable generative models. We present a successful example of incorporating differentiable renders into IGM transformations for generating highly-structured human body point clouds. The proposed approach has rich connections with existing problems in computer graphics, including pose estimation, corrective modeling and correspondence finding.
- **Find the Properties of Applications:** There are important yet straightforward applications of generative models, from language models to image generations. We explore new applications of using IGMs. In the first example, we revisit the random Fourier features of kernels. We learn the sampling process of an implicit spectral density, which leads to powerful kernels in different applications. In the second example, we learn proximal operators as a transformation via IGMs for solving linear inverse problems. In the last example, we propose to learn an auxiliary IGM to construct a lower bound for optimizing test power of hypothesis test.

1.3 Thesis Overview

This thesis contains selections of my works on addressing aforementioned issues of learning IGMs, which consists of three parts. Next, we give an overview of this thesis.

Part I: Learning Implicit Generative Models. The fundamental problem of learning IGMs is to define proper *distance* between distributions. In the first part, we present the study of probability distances for learning implicit generative models with limited assumptions on the data. We propose practical data-agnostic algorithms and conduct statistical analysis.

- In Chapter 2, we revisit classical two-sample test in statistics and introduce its connection with learning IGMs. From this perspective, we propose a generic algorithm MMD GAN (Li et al., 2017) without much assumptions of data, which can be built on the top of most existing GAN algorithms with improvements brought by the tools developed for two-sample test, maximum mean discrepancy (Gretton et al., 2012a).
- In Chapter 3, we generalize many widely used probability distances in learning IGMs as *adversarial loss* (Singh et al., 2018). We study the sample complexities and error bounds of nonparametric density estimation under a large class of adversarial loss functions with different assumed smoothness of the underlying density. In the end, we discuss how we can bring the analysis for adversarial density estimations to learning IGMs.

Part II: Implicit Generative Models by Leveraging Structures and Priors of Data. In the second part, we study how to learn IGMs on different data types. We answer three questions (1) how to design IGMs algorithms by leveraging our assumptions into objective formulations (2) how to design IGM algorithms which satisfies the structures of the data and (3) how to encode our knowledge of data as human priors into IGMs in an explicit way for better quality of generations.

- In Chapter 4, we propose Sobolev IPM (Mroueh et al., 2018), an objective for learning IGMs. We show that Sobolev IPM encourages leave-one-out (LOO) conditional distribution matching, which meets the standard assumption in language structures and has advantage in text generation. We further draw the connection between the proposed Sobolev IPM, transportation plan and Fokker-Planck Diffusion.
- In Chapter 5, we propose a framework by combining the hierarchical Bayesian modeling and IGMs to learn a hierarchical and interpretable sampling process for point clouds (Li et al., 2018). We further propose a sandwiching objective, which results in a tighter Wasserstein distance estimator than the commonly used estimator based on the dual of Wasserstein distance.
- In Chapter 6, we take a closer look at a special category of point clouds, body point clouds, which is known to be highly structured because of complex human poses and various body types. We demonstrate how to incorporate a linear-blend skinning (LBS) into the transformation pipeline of IGMs as a powerful prior (Li et al., 2019b). LBS encodes our knowledges of bodies via a body template describing the geometry and the hierarchy of skeletons which defines rigged transformations. With this expressive and interpretable prior, leaning IGMs has strong connections with model registrations and shape corrective compensations in computer graphics. The learned IGM can simultaneously solve important problems, including pose estimations and correspondence findings without external labeling. Thereby, the proposed IGM also bridges generative models and deep geometry learning.

Part III: Improvements from Implicit Generative Models In the third part, in addition to typical generation tasks for different types of data, we present different examples of how IGM can boost existing algorithms in different applications. The first example presents how to *reduce* problems into finite-sample estimations, which can be learned via IGMs. The second example shows how to *replace* core component of existing algorithms with IGMs to better adapt to the data. The third example, we demonstrate how to *create* auxiliary components via IGMs to improve performance.

- In addition to using kernel methods to learn IGMs, IGMs can be used to improve kernel learning (Li et al., 2019a) as well. Via standard random Fourier features, kernel evaluations can estimated via finite samples from a certain spectral distribution. In Chapter 7, we introduce powerful IGM kernels by modeling the sampling process of the spectral distribution via IGMs to better accommodate to the data.
- The second example presented in Chapter 8 is using iterative algorithms (e.g. ADMM) to solve linear inverse problems in computer vision. The heart of those algorithms is

the proximal operation in every iteration. Instead of hand-crafting projection steps, we learn the proximal operations via the transformation in IGMs (Chang et al., 2017a). The data-driven transformation outperforms traditional human-designed proximal operators.

- In Chapter 9, we study time-series change point detection. One commonly used algorithm is hypothesis test with test power maximization. We show a perturbed distribution learned via IGMs can be used to construct a power lower bound for optimization, which leads to more effective hypothesis test (Chang et al., 2019). Different from most of the existing algorithms, where the target of interest is the generative model, we show a different use of IGM which learns auxiliary generative models to benefit *non-generative* tasks.

Part IV: Conclusion Finally, we summarize this thesis in Chapter 10, and discuss the open problems for future research.

Part I

Learning Implicit Generative Models

Chapter 2

MMD GAN: Implicit Generative Models and Two-Sample Test

Modeling arbitrary density is a statistically challenging task (Wasserman, 2013). In many applications, however, accurate density estimation is not necessary since we are only interested in *sampling* from the approximated distribution. Rather than estimating the density of $\mathbb{P}_\mathcal{X}$, IGMs start from a base distribution $\mathbb{P}_\mathcal{Z}$ over $\mathcal{Z}$ (e.g. Gaussian distribution), then trains a transformation function g_θ such that $\mathbb{Q}_\theta \approx \mathbb{P}_\mathcal{X}$, where $\mathbb{Q}_\theta$ is the underlying distribution of $g_\theta(z)$ and $z \sim \mathbb{P}_\mathcal{Z}$.

To learn a powerful transformation for sampling, the core problem is to define a *distance* $D(\mathbb{P}_\mathcal{X} \| \mathbb{Q}_\theta)$ between $\mathbb{Q}_\theta$ and $\mathbb{P}_\mathcal{X}$ if we can only access samples from these two distributions. We then train g_θ by optimizing the defined distance. Generative Adversarial Network (GAN) (Goodfellow et al., 2014) trains an auxiliary network f_ϕ to estimate the distance between $\mathbb{P}_\mathcal{X}$ and $\mathbb{Q}_\theta$. In GAN, f_ϕ is a simple binary classifier. Therefore, f_ϕ is usually called as *discriminator* or *critic* in literature. The defined distance via a binary classifier is the dual form of Jensen-Shannon divergence. Using auxiliary nerual networks to estimate different distances for training GAN have been widely studied. The extension of Goodfellow et al. (2014) is using f-divergence (Mao et al., 2017; Nowozin et al., 2016). The counterpart of f-divergence, Integral probability metrics (IPM) (Müller, 1997) based GANs, are also extensively studied, including total variation (Zhao et al., 2017) Wasserstain distance (Arjovsky and Bottou, 2017; Arjovsky et al., 2017; Gulrajani et al., 2017), Cramer distance (Bellemare et al., 2017), and Fisher IPM (Mroueh and Sercu, 2017). The aforementioned works are all based on the *dual* of either f-divergence or IPM. Since learning with dual loss usually results in a minmax objective, it is also called adversarial loss. In addition the *dual* loss, learning IGMs in *primal* has also been proposed (Genevay et al., 2018; Kingma and Welling, 2013; Makhzani et al., 2015; Tolstikhin et al., 2017a).

In this chapter, we focus on training IGMs by optimizing kernel maximum mean discrepancy (MMD), which is also an IPM-based distance (Gretton et al., 2012a). The preliminary works have been done by Dziugaite et al. (2015); Li et al. (2015b). In this thesis, we extend Dziugaite et al. (2015); Li et al. (2015b) and draw a deeper connection between IGMs, two-sample test, kernel learning and adversarial training. The presented results are based on Li et al. (2017).

2.1 IGM and Two-Sample Test

Assume we are given data $\{x_i\}_{i=1}^n$, where $x_i \in \mathcal{X}$ and $x_i \sim \mathbb{P}_{\mathcal{X}}$. Generative Adversarial Network (GAN) (Goodfellow et al., 2014) learns a generator g_θ parameterized by θ to transform samples $z \sim \mathbb{P}_{\mathcal{Z}}$, where $z \in \mathcal{Z}$, into $g_\theta(z) \sim \mathbb{Q}_\theta$ such that $\mathbb{Q}_\theta \approx \mathbb{P}_{\mathcal{X}}$. To measure the distance $D(\mathbb{P}_{\mathcal{X}} \| \mathbb{Q}_\theta)$ between $\mathbb{P}_{\mathcal{X}}$ and $\mathbb{Q}_\theta$ via their samples $\{x\}_{i=1}^n$ and $\{g_\theta(z_j)\}_{j=1}^n$ during the training, GAN trains a discriminator f_ϕ parameterized by ϕ, which is a binary classifier, for help. During the training, f_ϕ tries to distinguish x_i and $g_\theta(z_j)$ as a proxy for measuring the distance. If $\{x_i\}$ and $\{g_\theta(z_j)\}$ are easily distinguished by the classifier, the distance $D(\mathbb{P}_{\mathcal{X}} \| \mathbb{Q}_\theta)$ should be high, and vice versa. Therefore, the problem can be formulated by a minmax optimization:

$$\min_\theta \max_\phi \mathbb{E}_{\mathbb{P}_{\mathcal{X}}}[\log f_\phi(x)] + \mathbb{E}_{\mathbb{P}_{\mathcal{Z}}}[\log(1 - f_\phi(g_\theta(z)))], \tag{2.1}$$

where the inner maximization is the maximum likelihood for binary classification and serves as $D(\mathbb{P}_{\mathcal{X}} \| \mathbb{Q}_\theta)$.

On the other hand, distinguishing two distributions by finite samples is known as *Two-Sample Test* in statistics. One way to conduct two-sample test is via kernel maximum mean discrepancy (MMD) (Gretton et al., 2012a). Given two distributions $\mathbb{P}$ and $\mathbb{Q}$, and a kernel k, the square of MMD distance is defined as

$$M_k(\mathbb{P}, \mathbb{Q}) = \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|_{\mathcal{H}}^2 = \mathbb{E}_{\mathbb{P},\mathbb{P}}[k(x, x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x, y)] + \mathbb{E}_{\mathbb{Q},\mathbb{Q}}[k(y, y')].$$

Theorem 1. *(Gretton et al., 2012a) Given a kernel k, if k is a characteristic kernel, then $M_k(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$.*

One example of characteristic kernel is Gaussian kernel $k(x, x') = \exp(\|x-x'\|^2)$. Based on Theorem 1, Dziugaite et al. (2015); Li et al. (2015b) propose the Generative Moment-Matching Network (GMMN), which uses $M_k(\mathbb{P}_{\mathcal{X}}, \mathbb{Q}_\theta)$ as $D(\mathbb{P}_{\mathcal{X}} \| \mathbb{Q}_\theta)$ and trains g_θ by

$$\min_\theta M_k(\mathbb{P}_{\mathcal{X}}, \mathbb{Q}_\theta), \tag{2.2}$$

with a fixed Gaussian kernel k rather than training an additional discriminator f as GAN. Although GMMN is easy to train without solving a minmax optimization as Eq. (2.1), the empirical performance is not satisfactory as we show in Figure 2.1.

2.1.1 MMD with Kernel Learning

In practice we use finite samples from distributions to estimate MMD distance. Given $X = \{x_1, \cdots, x_n\} \sim \mathbb{P}$ and $Y = \{y_1, \cdots, y_n\} \sim \mathbb{Q}$, one estimator of $M_k(\mathbb{P}, \mathbb{Q})$ is

$$\hat{M}_k(X, Y) = \frac{1}{\binom{n}{2}} \sum_{i \neq i'} k(x_i, x_i') - \frac{2}{\binom{n}{2}} \sum_{i \neq j} k(x_i, y_j) + \frac{1}{\binom{n}{2}} \sum_{j \neq j'} k(y_j, y_j').$$

Because of the sampling variance, $\hat{M}(X, Y)$ may not be zero even when $\mathbb{P} = \mathbb{Q}$. We then conduct hypothesis test with null hypothesis $H_0 : \mathbb{P} = \mathbb{Q}$. For a given allowable probability of false rejection α, we can only reject H_0, which implies $\mathbb{P} \neq \mathbb{Q}$, if $\hat{M}(X, Y) > c_\alpha$ for some chosen threshold $c_\alpha > 0$. Otherwise, $\mathbb{Q}$ passes the test and $\mathbb{Q}$ is indistinguishable from $\mathbb{P}$ under this test. Please refer to Gretton et al. (2012a) for more details.

Intuitively, if kernel k cannot result in high MMD distance $M_k(\mathbb{P}, \mathbb{Q})$ when $\mathbb{P} \neq \mathbb{Q}$, $\hat{M}_k(\mathbb{P}, \mathbb{Q})$ has more chance to be smaller than c_α. Then we are unlikely to reject the null hypothesis H_0 with finite samples, which implies $\mathbb{Q}$ is not distinguishable from $\mathbb{P}$. Therefore, instead of training g_θ via Eq. (2.2) with a pre-specified kernel k as GMMN, we consider training g_θ via

$$\min_\theta \max_{k \in \mathcal{K}} M_k(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta), \tag{2.3}$$

which takes different possible characteristic kernels $k \in \mathcal{K}$ into account. On the other hand, we could also view Eq. (2.3) as replacing the fixed kernel k in Eq. (2.2) with the *adversarially learned kernel* via kernel learning $\arg\max_{k\in\mathcal{K}} M_k(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ to have stronger signals where $\mathbb{P}_\mathcal{X} \neq \mathbb{Q}_\theta$ to train g_θ. We refer interested readers to Fukumizu et al. (2009) for more rigorous discussions about test power and increasing MMD distances.

However, it is difficult to optimize over all characteristic kernels when we solve Eq. (2.3). By Gretton et al. (2012a,c) if f is a injective function and k is characteristic, then the resulted kernel $k \circ f = k(f(x), f(x'))$ is still characteristic. If we have a family of injective functions parameterized by ϕ, which is denoted as f_ϕ, we are able to change the objective to be

$$\min_\theta \max_\phi M_{k\circ f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta). \tag{2.4}$$

An example parameterization is combining Gaussian kernels with injective functions f_ϕ, where $k_\phi(x, x') = \exp(-\|f_\phi(x) - f_\phi(x)'\|^2)$. If the function class of f is $\{f_\phi | f_\phi(x) = \phi x, \phi > 0\}$, which is equivalent to the kernel bandwidth tuning. A more advanced realization will be discussed in Section 2.2. Next, we abuse the notation $M_\phi(\mathbb{P}, \mathbb{Q})$ to be the MMD distance given the composition kernel of Gaussian kernel and f_ϕ in the following. Note that Gretton et al. (2012b) consider a linear combination of characteristic kernels, which can also be incorporated into the discussed composition kernels. More general kernels are studied by Wilson et al. (2016), and we leave the detailed discussion in Chapter 7.

2.1.2 Properties of MMD with Kernel Learning

Arjovsky et al. (2017) discuss different distances between distributions adopted by existing deep learning algorithms, and show many of them are discontinuous, such as Jensen-Shannon divergence (Goodfellow et al., 2014) and Total variation (Zhao et al., 2017), except for Wasserstein distance. The discontinuity makes the gradient descent infeasible for training. From Eq. (2.4), we train g_θ via minimizing $\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$, which can be treated as a distance in a variational form. Next, we show $\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ also enjoys several advantages under mild assumptions.

Assumption 2. *$g : \mathcal{Z} \times \mathbb{R}^m \to \mathcal{X}$ is locally Lipschitz, where $\mathcal{Z} \subseteq \mathbb{R}^d$. We will denote $g_\theta(z)$ the evaluation on (z, θ) for convenience. Given Lipschitz f_ϕ and a probability distribution $\mathbb{P}_z$ over $\mathcal{Z}$, g satisfies Assumption 2 if there are local Lipschitz constants $L(\theta, z)$ for $f_\phi \circ g$, which is independent of ϕ, such that $\mathbb{E}_{z\sim\mathbb{P}_z}[L(\theta, z)] < +\infty$.*

Theorem 3. *The generator function g_θ parameterized by θ is under Assumption 2. Let $\mathbb{P}_\mathcal{X}$ be a fixed distribution over $\mathcal{X}$ and Z be a random variable over the space $\mathcal{Z}$. We denote $\mathbb{Q}_\theta$ the distribution of $g_\theta(Z)$, then $\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ is continuous everywhere and differentiable almost everywhere in θ.*

If g_θ is parameterized by a feed-forward neural network, it satisfies Assumption 2 and can be trained via gradient descent as well as propagation, since the objective is continuous and differentiable followed by Theorem 3.

From integral probability metrics (IPM), the probability distance can be defined as

$$D(\mathbb{P}\|\mathbb{Q}) = \sup_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mathbb{P}}[f(x)] - \mathbb{E}_{y\sim\mathbb{Q}}[f(y)]. \tag{2.5}$$

By changing the function class $\mathcal{F}$, we can recover several distances, such as total variation, Wasserstein distance and MMD distance. From Arjovsky et al. (2017), the discriminator f_ϕ in different existing works of GAN can be explained to be used to solve different probabilistic metrics based on Eq. (2.5). For MMD, the function class $\mathcal{F}$ is $\{\|f\|_{\mathcal{H}_k} \leq 1\}$, where $\mathcal{H}$ is RKHS associated with kernel k. Different form many distances, such as total variation and Wasserstein distance, there is an analytical representation (Gretton et al., 2012a) as we show in Section 2.1, which is

$$\begin{aligned} M_k(\mathbb{P},\mathbb{Q}) &= \sup_{f\in\mathcal{H}_k} \mathbb{E}_{x\sim\mathbb{P}}[f(x)] - \mathbb{E}_{y\sim\mathbb{Q}}[f(y)] \\ &= \sqrt{\mathbb{E}_{\mathbb{P},\mathbb{P}}[k(x,x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x,y)] + \mathbb{E}_{\mathbb{Q},\mathbb{Q}}[k(y,y')]}. \end{aligned} \tag{2.6}$$

Because of the analytical representation of Eq. (2.6), GMMN does not need an additional network f_ϕ for estimating the distance. Here we provide an interpretation of the proposed MMD with kernel learning under the IPM framework. The MMD distance with adversarially learned kernels is represented as

$$\max_{k\in\mathcal{K}} M_k(\mathbb{P},\mathbb{Q}),$$

The corresponding IPM formulation is

$$\max_{k\in\mathcal{K}} M_k(\mathbb{P},\mathbb{Q}) = \sup_{f\in\mathcal{H}_{k_1}\cup\cdots\cup\mathcal{H}_{k_n}} \mathbb{E}_{x\sim\mathbb{P}}[f(x)] - \mathbb{E}_{y\sim\mathbb{Q}}[f(y)],$$

where $k_i \in \mathcal{K}, \forall i$. From this perspective, the proposed MMD distance with kernel learning is still defined by IPM but with a larger function class. Next, we prove $\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ is not only a valid probability distance, but also *weak* under Assumption 2.

Theorem 4. *(weak* topology) Let $\{\mathbb{P}_n\}$ be a sequence of distributions. Considering $n \to \infty$, under mild Assumption 2, $\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{P}_n) \to 0 \iff \mathbb{P}_n \xrightarrow{D} \mathbb{P}_\mathcal{X}$, where $\xrightarrow{D}$ means* converging in distribution *(Wasserman, 2013).*

Theorem 4 shows that $\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{P}_n)$ is a sensible cost function to the distance between $\mathbb{P}_\mathcal{X}$ and $\mathbb{P}_n$. The distance is decreasing when $\mathbb{P}_n$ is getting closer to $\mathbb{P}_\mathcal{X}$, which benefits the supervision of the improvement during the training. All proofs are omitted to Section 2.5. In the next section, we introduce a practical realization of training g_θ via optimizing $\min_\theta \max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$.

Algorithm 1 MMD GAN

```
1: Input: $\alpha$ the learning rate, $c$ the clipping parameter, $B$ the batch size, $n_c$ the number of iterations of discriminator per generator update.
2: while $\theta$ has not converged do
3:     for $t = 1, \ldots, n_c$ do
4:         Sample a minibatches $\{x_i\}_{i=1}^B \sim \mathbb{P}_\mathcal{X}$ and $\{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z})$
5:         $g_\phi \leftarrow \nabla_\phi M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$
6:         $\phi \leftarrow \phi + \alpha \cdot \text{RMSProp}(\phi, g_\phi)$
7:         $\phi \leftarrow \text{clip}(\phi, -c, c)$
8:     end for
9:     Sample a minibatch $\{x_i\}_{i=1}^B \sim \mathbb{P}_\mathcal{X}$ and $\{z_j\}_{j=1}^B \sim \mathbb{P}_\mathcal{Z}$
10:    $g_\theta \leftarrow \nabla_\theta M_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$
11:    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
12: end while
```

2.2 MMD GAN

To approximate Eq. (2.4), we use neural networks to parameterized g_θ and f_ϕ with expressive power. For g_θ, the assumption is locally Lipschitz, where commonly used feed-forward neural networks satisfy this constraint. The bounded gradient $\bigtriangledown_\theta (\max_\phi f_\phi \circ g_\theta)$ have been studied by Arjovsky et al. (2017); Gulrajani et al. (2017).

The non-trivial part is f_ϕ has to be injective. For an injective function f, there exists an function f^{-1} such that $f^{-1}(f(x)) = x, \forall x \in \mathcal{X}$ and $f^{-1}(f(g(z))) = g(z), \forall z \in \mathcal{Z}$[1], which can be approximated by an autoencoder. Bińkowski et al. (2018) extend the analysis of Theorem 4 to get rid of the injective constraint. Our empirical study suggests autoencoder objective is not necessary to successful GAN training as we will show in Section 2.3, which is consistent with Bińkowski et al. (2018).

The proposed algorithm is similar to GAN (Goodfellow et al., 2014), which aims to optimize two neural networks g_θ and f_ϕ in a minmax formulation, while the meaning of the objective is different. In GAN, f_ϕ is a discriminator (binary) classifier to distinguish two distributions. In the proposed algorithm, distinguishing two distribution is still done by two-sample test via MMD, but with an adversarially learned kernel parametrized by f_ϕ. g_θ is then trained to pass the hypothesis test. Because of the similarity of GAN, we call the proposed algorithm *MMD GAN*. We present an implementation with weight clipping (Arjovsky et al., 2017) in Algorithm 1. One can extend it to use other Lipschitz approximations, such as Gulrajani et al. (2017).

2.2.1 Feasible Set Reduction

Theorem 5. *For any f_ϕ, there exists f'_ϕ such that $M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta) = M_{f'_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ and $\mathbb{E}_x[f_\phi(x)] \succeq \mathbb{E}_z[f_{\phi'}(g_\theta(z))]$.*

[1]Note that injective is not necessary invertible.

With Theorem 5, we could reduce the feasible set of ϕ during the optimization by solving

$$\min_\theta \max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta) \quad s.t.\ \ \mathbb{E}[f_\phi(x)] \succeq \mathbb{E}[f_\phi(g_\theta(z))]$$

which the optimal solution is still *equivalent* to solving Eq. (2.3).

However, it is hard to solve the constrained optimization problem with backpropagation. We relax the constraint by ordinal regression (Herbrich et al., 1999) to be

$$\min_\theta \max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta) + \lambda \min\big(\mathbb{E}[f_\phi(x)] - \mathbb{E}[f_\phi(g_\theta(z))], 0\big),$$

which only penalizes the objective when the constraint is violated. In practice, we observe that reducing the feasible set makes the training converge faster and stabler.

2.2.2 Encoding Perspectives and Connections with WGAN

Besides from using kernel learning to explain MMD GAN, the other way to interpret the proposed MMD GAN is viewing f_ϕ as a feature transformation function, and the kernel two-sample test is performed on this transformed feature space or the learnt embeddings (i.e., the code space of the autoencoder). The optimization is finding a manifold with stronger signals for MMD two-sample test. From this perspective, Li et al. (2015b) is the special case of MMD GAN if f_ϕ is the identity mapping function. In such circumstance, the kernel two-sample test is conducted in the original data space.

If we composite f_ϕ with linear kernel instead of Gaussian kernel, and restricting the output dimension h to be 1, we then have the objective

$$\min_\theta \max_\phi \|\mathbb{E}[f_\phi(x)] - \mathbb{E}[f_\phi(g_\theta(z))]\|^2. \tag{2.7}$$

Parameterizing f_ϕ and g_θ with neural networks and assuming $\exists \phi' \in \Phi$ such $f'_\phi = -f_\phi, \forall \Phi$, recovers Wasserstein GAN (WGAN) (Arjovsky et al., 2017) If we treat $f_\phi(x)$ as the data transform function, WGAN can be interpreted as first-order moment matching (linear kernel) while MMD GAN aims to match infinite order of moments with Gaussian kernel form Taylor expansion (Li et al., 2015b). Theoretically, Wasserstein distance has similar theoretically guarantee as Theorem 1, 3 and 4. In practice, Arora et al. (2017) show neural networks does not have enough capacity to approximate Wasserstein distance. In Section 2.3, we demonstrate matching high-order moments benefits the results. Mroueh et al. (2017) also propose McGAN that matches second order moment from the primal-dual norm perspective, which can be treated as a middle-ground between MMD GAN (infinitely many order moments matching) and WGAN (first-order moment matching).

2.3 Experiment

We train MMD GAN for image generation on the MNIST (LeCun et al., 1998), CIFAR-10 (Krizhevsky and Hinton, 2009), CelebA (Liu et al., 2015), and LSUN bedrooms (Yu et al., 2015) datasets, where the size of training instances are 50K, 50K, 160K, 3M respectively.

Network Architecture. In our experiments, we follow the architecture of DCGAN (Radford et al., 2016) to design g_θ by its generator and f_ϕ by its discriminator except for expanding the output layer of f_ϕ to be h dimensions.

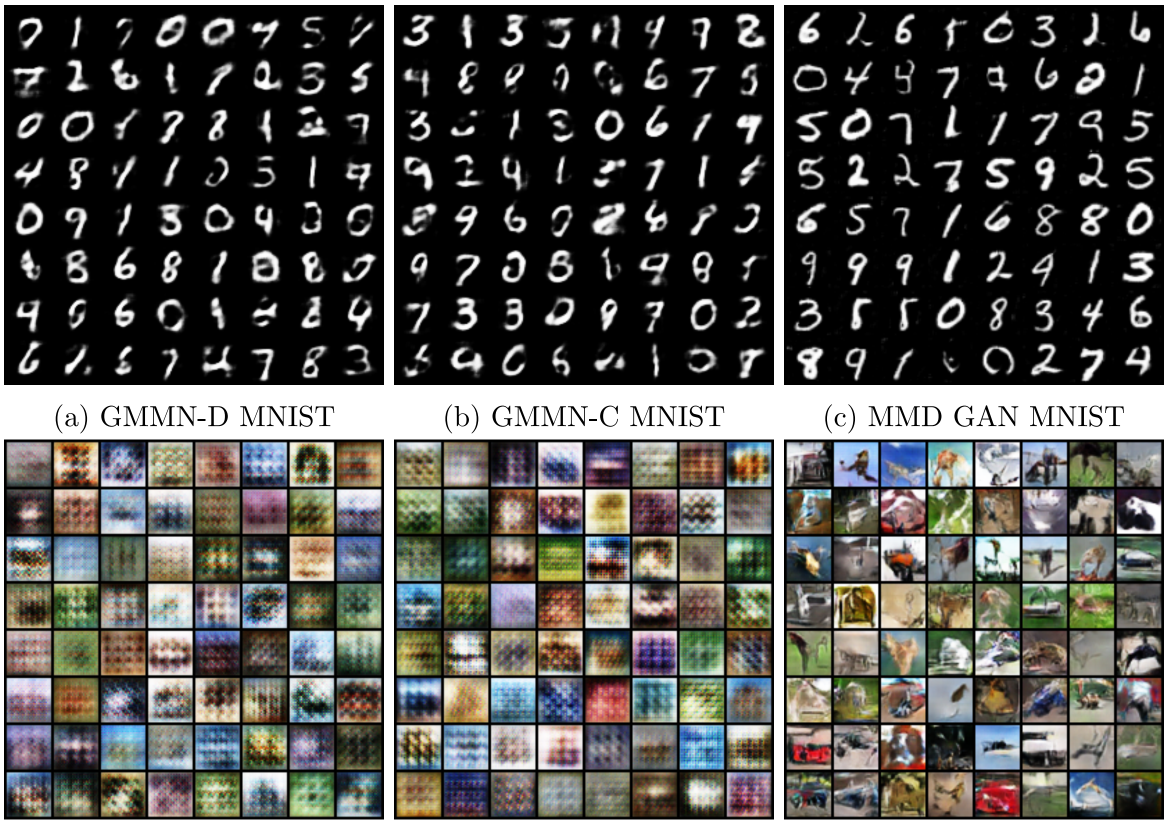
Kernel Designs. The loss function of MMD GAN is implicitly associated with a family of characteristic kernels. Similar to the prior works (Dziugaite et al., 2015; Li et al., 2015b; Sutherland et al., 2017), we consider a mixture of K RBF kernels $k(x, x') = \sum_{q=1}^{K} k_{\sigma_q}(x, x')$ where k_{σ_q} is a Gaussian kernel with bandwidth parameter σ_q. Tuning kernel bandwidth σ_q optimally still remains an open problem. In this works, we fixed $K = 5$ and σ_q to be $\{1, 2, 4, 8, 16\}$ and left the f_ϕ to learn the kernel (feature representation) under these σ_q.

Hyper-parameters. We use RMSProp (Tieleman and Hinton, 2012) with learning rate of 0.00005 for a fair comparison with WGAN as suggested in Arjovsky et al. (2017). We ensure the boundedness of model parameters of discriminator by clipping the weights point-wisely to the range $[-0.01, 0.01]$ as required by Assumption 2. The dimensionality h of the latent space is manually set according to the complexity of the dataset. We thus use $h = 16$ for MNIST, $h = 64$ for CelebA, and $h = 128$ for CIFAR-10 and LSUN bedrooms. The batch size is set to be $B = 64$ for all datasets.

2.3.1 Qualitative Analysis

We start with comparing MMD GAN with GMMN on two standard benchmarks, MNIST and CIFAR-10. We consider two variants for GMMN. The first one is original GMMN, which trains generators by minimizing the MMD distances on the original data space. We call it as *GMMN-D*. To compare with MMD GAN, we also pretrain an autoencoder for projecting data to a manifold, then fix the autoencoder as a feature transformation, and train generators by minimizing the MMD distances in the code space. We call it as *GMMN-C*. The results are shown in Figure 2.1. Both GMMN-D and GMMN-C are able to generate meaningful digits on MNIST because of its simple structures. By a closer look, nonetheless, the edges and silhouettes of the digits in Figure 2.1a and 2.1b are often irregular and non-smooth. In contrast, the sample digits in Figure 2.1c are more natural with better silhouettes and sharper edges. For CIFAR-10 dataset, both GMMN variants fail to generate meaningful images, but results in low-level visual features. We observe similar cases in other complex large-scale datasets such as CelebA and LSUN bedrooms, thus those results are omitted. On the other hand, the proposed MMD GAN successfully outputs natural images with sharp boundaries and high diversities. The results in Figure 2.1 confirm the success of the proposed kernel learning to enrich statistical test power for a more discriminative distance, which is the key difference between GMMN and MMD GAN.

If we increase the batch size of GMMN to be 1024, the image quality shown in Figure 2.2 is improved. However, it is still not competitive to MMD GAN with $B = 64$. This demonstrates that the proposed MMD GAN can be trained more efficiently than GMMN with smaller batch sizes.



(a) GMMN-D MNIST (b) GMMN-C MNIST (c) MMD GAN MNIST

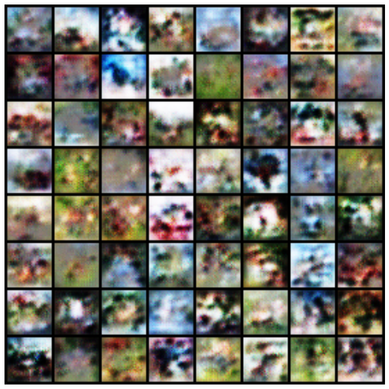
(d) GMMN-D CIFAR-10 (e) GMMN-C CIFAR-10 (f) MMD GAN CIFAR-10

Figure 2.1: Generated samples from GMMN-D (Dataspace), GMMN-C (Codespace) and our MMD GAN with batch size $B = 64$.

Comparisons with GANs. There are several representative extensions of GANs. We consider recent state-of-art WGAN (Arjovsky et al., 2017) based on DCGAN structure (Radford et al., 2016). The results are shown in Figure 2.3. For MNIST, the digits generated from WGAN in Figure 2.3a are more unnatural with peculiar strikes. In contrary, the digits from MMD GAN in Figure 2.3d enjoy smoother contour. Furthermore, both WGAN and MMD GAN generate diversified digits, avoiding the mode-collapse problems appeared in the literature of training GANs. For CelebA, we can see the difference of generated samples from WGAN and MMD GAN. Specifically, we observe varied poses, expressions, genders, skin colors and light exposure in Figure 2.3b and 2.3e. By a closer look (view on-screen with zooming in), we observe that faces from WGAN have higher chances to be blurry and twisted while faces from MMD GAN are more spontaneous with sharp and acute outline of faces. For LSUN dataset, we could not distinguish salient differences between the samples generated from MMD GAN and WGAN.

(a) GMMN-D on MNIST

(b) GMMN-C on MNIST


(c) GMMN-D on CIFAR-10

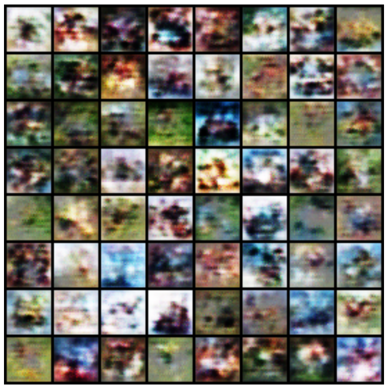

(d) GMMN-C CIFAR-10

Figure 2.2: Generated samples from GMMN-D and GMMN-C with large training batch size $B = 1024$.

2.3.2 Quantitative Analysis

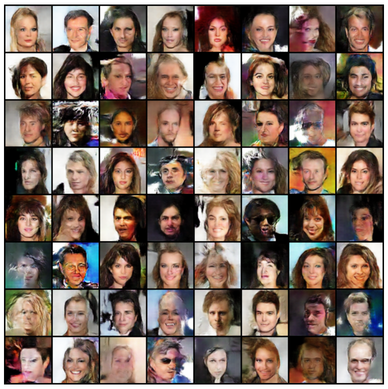
To quantitatively measure the quality and diversity of generated samples, we compute the inception score (Salimans et al., 2016) on CIFAR-10 images. The inception score is used for GANs to measure samples quality and diversity on the pretrained inception model (Salimans et al., 2016). Models that generate collapsed samples have a relatively low score. Table 2.1 lists the results for $50K$ samples generated by various unsupervised generative models trained on CIFAR-10 dataset. The inception scores of Dumoulin et al. (2017); Salimans et al. (2016) are directly derived from the corresponding references.

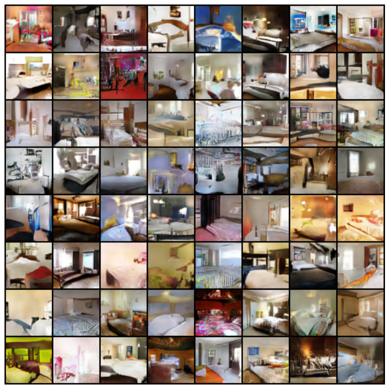
Although both WGAN and MMD GAN can generate sharp images as we show in Section 2.3.1, our score is better than other GAN techniques. This seems to confirm empirically that higher order of moment matching between the real data and fake sample distribution benefits generating more diversified sample images.

2.3.3 Stability of MMD GAN

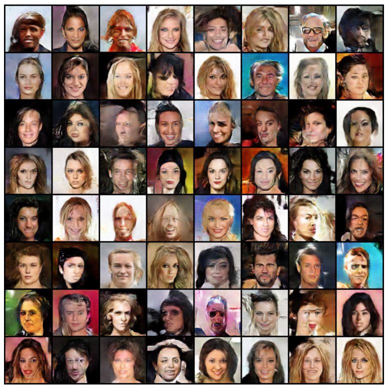
We further illustrate how the MMD distance with kernel learning correlates well with the quality of the generated samples. Figure 2.5 plots the evolution of the distance during the

(a) WGAN MNIST


(b) WGAN CelebA


(c) WGAN LSUN

(d) MMD GAN MNIST


(e) MMD GAN CelebA

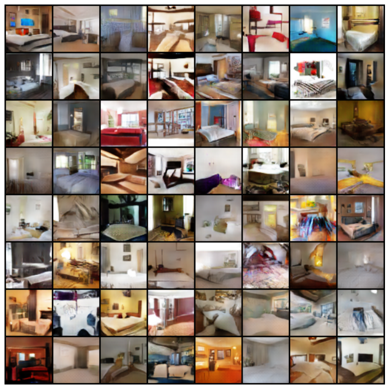

(f) MMD GAN LSUN

Figure 2.3: Generated samples from WGAN and MMD GAN on MNIST, CelebA, and LSUN bedroom datasets.

Method	Scores ± std.
Real data	11.95 ± .20
ALI (Dumoulin et al., 2017)	5.34
Improved GANs (Salimans et al., 2016)	4.36
MMD GAN	**6.17** ± .07
WGAN (Arjovsky et al., 2017)	5.88 ± .07
GMMN-C	3.94 ± .04
GMMN-D	3.47 ± .03

Table 2.1: Inception scores of different GAN algorithms.

MMD GAN training for MNIST, CelebA and LSUN datasets. We report the average of the $\hat{M}_{f_\phi}(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$ with moving average to smooth the graph to reduce the variance caused by mini-batch stochastic training. We observe during the whole training, samples generated from the same noise vector across iterations, remain similar in nature. (e.g., face identities and bedroom styles are alike while details and backgrounds are evolving.) This qualitative

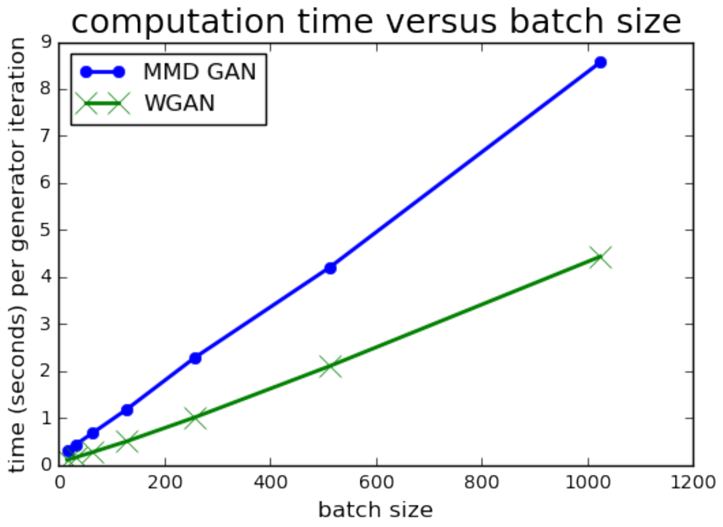
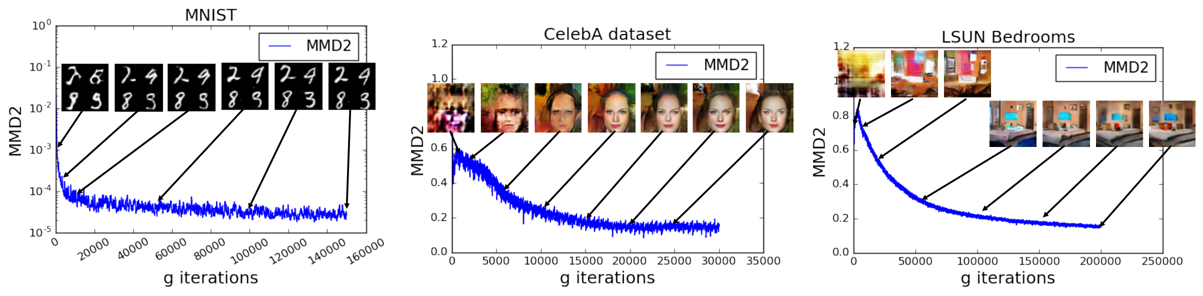


Figure 2.4: Computational time for one generator update iteration with different batch sizes.



(a) MNIST (b) CelebA (c) LSUN Bedrooms

Figure 2.5: Training curves and generative samples at different stages of training. We can see a clear correlation between lower distance and better sample quality.

observation indicates valuable stabilities of the training process. The decreasing curve with the improving quality of images supports the weak* topology in Theorem 4. Also, from the plot, the model converges very quickly. In Figure 2.5b, for example, it converges shortly after tens of thousands of generator iterations on CelebA dataset.

2.3.4 Computational Issue

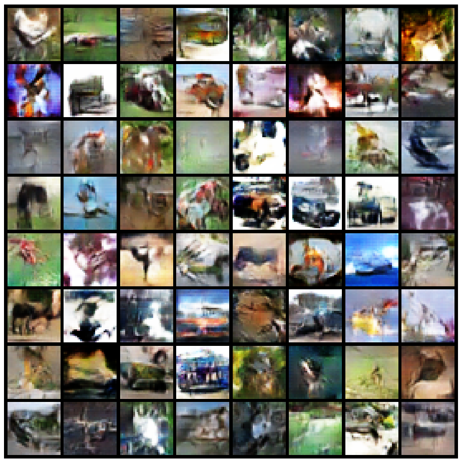
We conduct time complexity analysis with respect to the batch size B. The time complexity of each iteration is $O(B)$ for WGAN and $O(KB^2)$ for our proposed MMD GAN with a mixture of K RBF kernels. The quadratic complexity $O(B^2)$ of MMD GAN is introduced by computing kernel matrix, which is sometimes criticized for being inapplicable with large batch size in practice. However, we point out that several recent works, such as Zhao et al. (2017), also match pairwise relations between samples in a batch, leading to $O(B^2)$

complexity as well.

Empirically, we find that under GPU environments, their highly parallelized matrix operations tremendously alleviate the quadratic time to be almost the same as being linear with modest B. Figure 2.4 compares the computational time per generator iterations versus different B on Titan X. When $B = 64$, which is adapted for training MMD GAN in our experiments setting, the time per iteration of WGAN and MMD GAN is 0.268 and 0.676 seconds, respectively. When $B = 1024$, which is used for training GMMN in its references (Li et al., 2015b), the time per iteration becomes 4.431 and 8.565 seconds, respectively. This result coheres our argument that the empirical computational time for MMD GAN is not quadratically expensive compared to WGAN with powerful GPU parallel computation.

2.3.5 Better Lipschitz Approximation and Necessity of Auto-Encoder

We used weight-clipping for Lipschitz constraint in Assumption 2. Another approach for obtaining a discriminator with the similar constraint that approximates a Wasserstein distance is Gulrajani et al. (2017), where the gradient of the discriminator is constrained to be 1 between the generated and data points. Inspired by Gulrajani et al. (2017), an alternative approach is to apply the gradient constraint as a regularization of f_ϕ. This idea was firstly proposed by Bellemare et al. (2017) for the energy distance. Here we undertake an investigation of this approach. We also drop the requirement in Algorithm 1 that f_ϕ has to be injective by following Bińkowski et al. (2018), which we observe that it is not necessary in practice. We show results of training MMD GAN with the gradient penalty and without autoencoders in Figure 2.6. The study indicates that MMD GAN can generate satisfactory results with other Lipschitz constraint approximation and without autoencoders.


(a) Cifar10, $Giter = 300K$

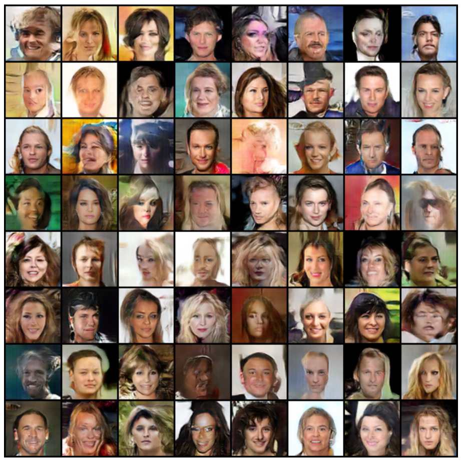

(b) CelebA, $Giter = 300K$

Figure 2.6: MMD GAN results using the gradient penalty (Gulrajani et al., 2017) and without autoencoders during training.

2.4 Discussion

We introduce a new generic algorithm for implicit generative models, MMD GAN, trained via MMD with adversarially learned kernels. We further study its theoretical properties and propose a practical realization MMD GAN, which can be trained with much smaller batch size than using simple Gaussian kernel and has competitive performances with state-of-the-art GANs. From the connection with state-of-the-art WGAN, which is using first-order moment matching, MMD GAN can always bring performance gain over WGAN by leveraging higher-order moment matching of learnt embeddings via MMD.

2.5 Technical Proof

2.5.1 Proof of Theorem 3

The proof is following the Lemma from Borisenko and Minchenko (1992).

Lemma 6. *(Borisenko and Minchenko (1992)) Define $\tau(x) = \max\{f(x,u) | u \in U\}$. If f is locally Lipschitz in x, U is compact and $\bigtriangledown f(x, u^*(x))$ exists, where $u^*(x) = \arg\max_u f(x,u)$, then $\tau(x)$ is differentiable almost everywhere.*

For simplifying notations, we define $k_\phi := k \circ f_\phi$. We are going to show

$$\max_\phi M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta) = \mathbb{E}_{x,x'}[k_\phi(x,x')] - 2\mathbb{E}_{x,z}[k_\phi(x, g_\theta(z))] + \mathbb{E}_{z,z'}[k_\phi(g_\theta(z'), g_\theta(z))] \tag{2.8}$$

is differentiable with respect to ϕ almost everywhere by using the auxiliary Lemma 6. We first show $\mathbb{E}_{z,z'}[k_\phi(g_\theta(z'), g_\theta(z))]$ in (2.8) is locally Lipschitz in θ. By definition, $k_\phi(x,x') = k(f_\phi(x) - f_\phi(x'))$, therefore,

$$\begin{aligned}
&\mathbb{E}_{x,x'}\left[k_\phi\Big(g_\theta(z), g_\theta(z')\Big) - k_\phi\Big(g_{\theta'}(z), g_{\theta'}(z')\Big)\right] \\
=&\mathbb{E}_{z,z'}\left[k\Big(f_\phi\big(g_\theta(z)\big) - f_\phi\big(g_\theta(z')\big)\Big)\right] - \mathbb{E}_{z,z'}\left[k\Big(f_\phi\big(g_{\theta'}(z)\big) - f_\phi\big(g_{\theta'}(z')\big)\Big)\right] \\
\leq&\mathbb{E}_{z,z'}\left[L_k\Big\|f_\phi\big(g_\theta(z)\big) - f_\phi\big(g_\theta(z')\big) - f_\phi\big(g_{\theta'}(z)\big) + f_\phi\big(g_{\theta'}(z')\big)\Big\|\right] \\
\leq&\mathbb{E}_{z,z'}\Big[L_k L(\theta,z)\|\theta-\theta'\| + L_k L(\theta,z')\|\theta-\theta'\|\Big] \\
=&2L_k\mathbb{E}_z\big[L(\theta,z)\big]\|\theta-\theta'\|.
\end{aligned}$$

The first inequality is because Gaussian kernel k is Lipschitz (locally Lipschitz) in (x,x'), with a upper bound L_k for Lipschitz constants. By Assumption 2, $\mathbb{E}_z\big[L(\theta,z)\big] < \infty$, we prove $\mathbb{E}_{z,z'}\Big[k\big(f_\phi(g_\theta(z)) - f_\phi(g_\theta(z'))\big)\Big]$ is locally Lipschitz. The similar argument is applicable to other terms in (2.8); therefore, (2.8) is locally Lipschitz in θ.

Last, with the compactness assumption on Φ, and the differentiable assumption on $M_\phi(\mathbb{P}_\mathcal{X}, \mathbb{Q}_\theta)$, applying Lemma 6 proves Theorem 3.

2.5.2 Proof of Theorem 4

Proof. We first show $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$ then $\max_\phi M_\phi(\mathbb{P}, \mathbb{P}_n) \to 0$. The results is based on Corollary 11.3.4 of Dudley (2018). Similar proof is shown in Arbel et al. (2018). With Dudley (2018), the only thing we remain to show is proving $\|k(f_\phi(x), \cdot) - k(f_\phi(y), \cdot)\|_{\mathcal{H}_K}$ is Lipschitz. By definition, we know that $\|k(f_\phi(x), \cdot) - k(f_\phi(y), \cdot)\|_{\mathcal{H}_K} = 2(1 - k(f_\phi(x), f_\phi(y)))$. Also, since Gaussian kernel k is Lipschitz, we have $k(0) - k(x, x') \leq L_k \|0 - (x - x')\|$. With $k(0) = 1$ for Gaussian kernel,

$$\|k(f_\phi(x), \cdot) - k(f_\phi(y), \cdot)\|_{\mathcal{H}_K} \leq 2L_k \|f_\phi(x) - f_\phi(y)\| \leq 2L_k L \|x - y\|,$$

where the last inequality is since f_ϕ is also a Lipschitz function with a Lipschitz constant L.

The other direction, $\max_\phi M_\phi(\mathbb{P}, \mathbb{P}_n) \to 0$ then $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$, is relatively simple. Without loss of generality, we assume there exists ϕ' such that f_ϕ is an identity function (up to scaling), which recover the Gaussian kernel k. Therefore, $\max_\phi M_\phi(\mathbb{P}, \mathbb{P}_n) \to 0$ implies $M_\phi(\mathbb{P}, \mathbb{P}_n) \to 0$, which completes the proof because MMD with any Gaussian kernel is weak (Gretton et al., 2012a). $\square$

2.5.3 Proof of Theorem 5

Proof. The proof assumes $f_\phi(x)$ is a scalar, but the vector case can be proved with the same sketch. First, if $\mathbb{E}[f_\phi(x)] > \mathbb{E}[f_\phi(g_\theta(z))]$, then $\phi = \phi'$. If $\mathbb{E}[f_\phi(x)] < \mathbb{E}[f_\phi(g_\theta(z))]$, we let $f = -f_\phi$, then $\mathbb{E}[f(x)] > \mathbb{E}[f(g_\theta(z))]$ and flipping sign does not change the MMD distance. If we parameterized f_ϕ by a neural network, which has a linear output layer, ϕ' can realized by flipping the sign of the weights of the last layer. $\square$

Chapter 3

Learning with Adversarial Losses

Learning generative models, which models the distribution from which data are drawn, is a central task in machine learning and statistics. Often, prior information is insufficient to guess the form of the data distribution. In statistics, generative modeling in these settings is usually studied from the perspective of nonparametric density estimation, in which histogram, kernel, orthogonal series, and nearest-neighbor methods are popular approaches with well-understood statistical properties (Biau and Devroye, 2015; Efromovich, 2010; Tsybakov, 2009; Wasserman, 2006). Recently, implicit generative models have made significant empirical progress in generative modeling. Computationally, IGMs are quite distinct from classical density estimators; it relies on powerful transformation functions, such as neural networks, fit by black-box optimization, rather than a mathematically prescribed smoothing operator, such as convolution with a kernel or projection onto a finite-dimensional subspace.

Ignoring the implementation of these models, from the perspective of statistical analysis, these recent methods have at least two main differences from classical density estimators. First, they are *implicit*, rather than *explicit* generative models (Diggle and Gratton, 1984; Mohamed and Lakshminarayanan, 2016); that is, they model the sampling process instead of the probability of a set or the density at a point. Second, in many recent models, loss (distance) 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

$$d_{\mathcal{F}_D}(\mathbb{P}, \mathbb{Q}) = \sup_{f \in \mathcal{F}_D} \left| \mathop{\mathbb{E}}_{X \sim \mathbb{P}} [f(X)] - \mathop{\mathbb{E}}_{X \sim \mathbb{Q}} [f(X)] \right|, \tag{3.1}$$

where $\mathcal{F}_D$ is a *discriminator class* of bounded, Borel-measurable functions, and $\mathbb{P}$ and $\mathbb{Q}$ lie in a *generator class* $\mathcal{F}_G$ of Borel probability measures on a sample space $\mathcal{X}$. Specifically, IGMs (or GANs) often use a neural network, which is called discriminator, to approximate (3.1).

This chapter is based on the results in Singh et al. (2018). We attempt to bridge the gap between traditional nonparametric statistics and recent advances of IGMs by studying these two differences from a statistical minimax perspective. Specifically, under traditional statistical smoothness assumptions, Singh et al. (2018) identify minimax convergence rates for density estimation under several losses of the form (3.1) that have been used in IGMs, including 1-Wasserstein distance (Arjovsky et al., 2017; Gulrajani et al., 2017), MMD (Li et al., 2017), Sobolev distances (Mroueh et al., 2018), and the Dudley metric (Abbasnejad et al., 2018). Given the generality of Singh et al. (2018), it relates to many prior works on distribution estimation, including classical work in nonparametric statistics and empirical process theory, as well as more recent work studying Wasserstein distances and MMD (Gretton et al., 2012a; Lei, 2018; Liang, 2017; Singh and Póczos, 2018; Tolstikhin et al., 2017b; Tsybakov, 2009; Wasserman, 2006; Weed and Bach, 2017). In this chapter, we discuss some consequences from minimax analysis for particular neural network implementations of IGMs based on these losses. Finally, we study connections between minimax rates for explicit and implicit generative models, under a plausible notion of risk for implicit generative models.

3.1 Adversarial Losses

The quantity (3.1) has been extensively studied, in the case that $\mathcal{F}_D$ is a reproducing kernel Hilbert space under the name *maximum mean discrepancy* (Gretton et al., 2012a; Tolstikhin et al., 2017b), and, in a wider context under the name *integral probability metric* (IPM) (Bottou et al., 2017; Müller, 1997; Sriperumbudur et al., 2010, 2012). Arora et al. (2017) also called Eq. (3.1) the *$\mathcal{F}_D$-distance*, or, when $\mathcal{F}_D$ is a family of functions that can be implemented by a neural network, the *neural network distance*. We settled on the name "adversarial loss" because, without assuming any structure on $\mathcal{F}_D$, this matches the intuition of Eq. (3.1), namely that of an adversary selecting the most distinguishing linear projection $f \in \mathcal{F}_D$ between densities $\mathbb{P}$ and $\mathbb{Q}$ (e.g., by the discriminator network in a GAN).

One can check that $d_{\mathcal{F}_D} : \mathcal{F}_G \times \mathcal{F}_G \to [0, \infty]$ is a pseudometric (i.e., it is non-negative and satisfies the triangle inequality, and $d_{\mathcal{F}_D}(\mathbb{P}, \mathbb{Q}) > 0 \Rightarrow \mathbb{P} \neq \mathbb{Q}$, although $d_{\mathcal{F}_D}(\mathbb{P}, \mathbb{Q}) = 0 \not\Rightarrow \mathbb{P} = \mathbb{Q}$ unless $\mathcal{F}_D$ is sufficiently rich). Many popular (pseudo) metrics between probability distributions, including $\mathcal{L}^p$ (Tsybakov, 2009; Wasserman, 2006), Sobolev (Leoni, 2017; Mroueh et al., 2018), MMD (Tolstikhin et al., 2017b)), energy distance (Ramdas et al., 2017; Székely et al., 2007), total variation (Villani, 2008), (1-)Wasserstein/Kantorovich-Rubinstein (Kantorovich and Rubinstein, 1958; Villani, 2008), Kolmogorov-Smirnov (Kolmogorov, 1933; Smirnov, 1948), and Dudley (Abbasnejad et al., 2018; Dudley, 1972) metrics can be written in this form, for appropriate choices of $\mathcal{F}_D$.

Formal Problem Statement: Let $\mathbb{P} \in \mathcal{F}_G$ be an unknown probability measure on a sample space $\mathcal{X}$, from which we observe n IID samples $X_{1:n} = X_1, ..., X_n \overset{IID}{\sim} \mathbb{P}$. In this chapter, we are interested in using the samples $X_{1:n}$ to estimate the measure $\mathbb{P}$, with error measured using the adversarial loss $d_{\mathcal{F}_D}$. Specifically, for various choices of spaces $\mathcal{F}_D$ and $\mathcal{F}_G$, we seek to bound the minimax rate

$$M(\mathcal{F}_D, \mathcal{F}_G) := \inf_{\widehat{\mathbb{P}}} \sup_{\mathbb{P} \in \mathcal{F}_G} \mathop{\mathbb{E}}_{X_{1:n}} \left[d_{\mathcal{F}_D} \left(\mathbb{P}, \widehat{\mathbb{P}}(X_{1:n}) \right) \right]$$

of estimating distributions assumed to lie in a class $\mathcal{F}_G$, where the infimum is taken over all estimators $\widehat{\mathbb{P}}$ (i.e., all (potentially randomized) functions $\widehat{\mathbb{P}} : \mathcal{X}^n \to \mathcal{F}_G$).

3.1.1 Notation

For a non-negative integer n, we use $[n] := \{1, 2, ..., n\}$ to denote the set of positive integers at most n. For sequences $\{a_n\}_{n \in \mathbb{N}}$ and $\{b_n\}_{n \in \mathbb{N}}$ of non-negative reals, $a_n \lesssim b_n$ and, similarly $b_n \gtrsim a_n$, indicate the existence of a constant $C > 0$ such that $\limsup_{n \to \infty} \frac{a_n}{b_n} \leq C$. $a_n \asymp b_n$ indicates $a_n \lesssim b_n \lesssim a_n$. For functions $f : \mathbb{R}^d \to \mathbb{R}$, we write

$$\lim_{\|z\| \to \infty} f(z) := \sup_{\{z_n\}_{n \in \mathbb{N}} : \|z_n\| \to \infty} \lim_{n \to \infty} f(z_n),$$

where the supremum is taken over all diverging $\mathbb{R}^d$-valued sequences. Note that, by equivalence of finite-dimensional norms, the exact choice of the norm $\| \cdot \|$ does not matter here. We will also require summations of the form $\sum_{z \in \mathcal{Z}} f(z)$ in cases where $\mathcal{Z}$ is a (potentially infinite) countable index set and $\{f(z)\}_{z \in \mathcal{Z}}$ is summable but not necessarily absolutely summable. Therefore, to ensure that the summation is well-defined, the order of summation will need to be specified, depending on the application (as in, e.g., Section 3.1.3).

Fix the sample space $\mathcal{X} = [0, 1]^d$ to be the d-dimensional unit cube, over which λ denotes the usual Lebesgue measure. Given a measurable function $f : \mathcal{X} \to \mathbb{R}$, let, for any Borel measure μ on $\mathcal{X}$, $p \in [1, \infty]$, and $L > 0$,

$$\|f\|_{\mathcal{L}^p_\mu} := \left(\int_{\mathcal{X}} |f|^p \, d\mu \right)^{1/p} \quad \text{and} \quad \mathcal{L}^p_\mu(L) := \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; \|f\|_{\mathcal{L}^p_\mu} < L \right\}$$

(taking the appropriate limit if $p = \infty$) denote the Lebesgue norm and ball of radius L, respectively.

Fix an orthonormal basis $\mathcal{B} = \{\phi_z\}_{z \in \mathcal{Z}}$ of $\mathcal{L}^2_\lambda$ indexed by a countable family $\mathcal{Z}$. To allow probability measures $\mathbb{P}$ without densities (i.e., $\mathbb{P} \not\ll \mu$), we assume each basis element $\phi_z : \mathcal{X} \to \mathbb{R}$ is a bounded function, so that $\widetilde{\mathbb{P}}_z := \mathbb{E}_{X \sim \mathbb{P}} [\phi_z(X)]$ is well-defined. For constants $L > 0$ and $p \geq 1$ and real-valued net $\{a_z\}_{z \in \mathcal{Z}}$, our results pertain to generalized ellipses of the form

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

(where $\widetilde{f}_z := \int_{\mathcal{X}} f \phi_z \, d\mu$ is the z^{th} coefficient of f in the basis $\mathcal{B}$). We sometimes omit dependence on L (e.g., $\mathcal{H}_{p,a} = \mathcal{H}_{p,a}(L)$) when its value does not matter (e.g., when discussing *rates* of convergence).

A particular case of interest is the scale of the Sobolev spaces defined for $s, L \geq 0$ and $p \geq 1$ by

$$\mathcal{W}^{s,p}(L) = \left\{ f \in \mathcal{L}^1(\mathcal{X}) : \left(\sum_{z \in \mathcal{Z}} |z|^{sp} |\widetilde{f}_z|^p \right)^{1/p} \leq L \right\}.$$

For example, when $\mathcal{B}$ is the standard Fourier basis and s is an integer, for a constant factor c depending only on s and the dimension d,

$$\mathcal{W}^{s,p}(cL) := \left\{ f \in \mathcal{L}^p_\lambda \middle| \|f^{(s)}\|_{\mathcal{L}^p_\lambda} < L \right\}$$

corresponds to the natural standard smoothness class of $\mathcal{L}^p_\lambda$ functions having s^{th}-order (weak) derivatives $f^{(s)}$ in $\mathcal{L}^p_\lambda(L)$ (Leoni, 2017).

3.1.2 Upper Bound and Minimax Lower Bound

For any finite set $Z \subseteq \mathcal{Z}$, let $\widehat{\mathbb{P}}_Z$ be the truncated series estimate

$$\widehat{\mathbb{P}}_Z := \sum_{z \in Z} \widehat{\mathbb{P}}_z \phi_z, \qquad \text{where, for any } z \in \mathcal{Z}, \quad \widehat{\mathbb{P}}_z := \frac{1}{n} \sum_{i=1}^{n} \phi_z(X_i). \tag{3.2}$$

Z is a tuning parameter that typically corresponds to a smoothing parameter; for example, when $\mathcal{B}$ is the Fourier basis and $Z = \{z \in \mathbb{Z}^d : \|z\|_\infty \leq \zeta\}$ for some $\zeta > 0$, $\widehat{\mathbb{P}}_Z$ 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 present the upper bound on the minimax rate of density estimation under adversarial losses proven by Singh et al. (2018). The upper bound is given by the orthogonal series estimator given in Eq. (3.2), but we expect kernels and other standard linear density estimators to converge at the same rate.

Theorem 7 (Upper Bound (Singh et al., 2018)). *Suppose that $\mu(\mathcal{X}) < \infty$ and there exist constants $L_D, L_G > 0$, real-valued nets $\{a_z\}_{z\in\mathcal{Z}}$, $\{b_z\}_{z\in\mathcal{Z}}$ such that $\mathcal{F}_D = \mathcal{H}_{p,a}(\mathcal{X}, L_D)$ and $\mathcal{F}_G = \mathcal{H}_{q,b}(\mathcal{X}, L_G)$, where $p, q \geq 1$. Let $p' = \frac{p}{p-1}$ denote the Hölder conjugate of p. Then, for any $\mathbb{P} \in \mathcal{F}_G$,*

$$\mathop{\mathbb{E}}_{X_{1:n}} \left[d_{\mathcal{F}_D}\left(\mathbb{P}, \widehat{\mathbb{P}}\right) \right] \leq L_D \frac{c_{p'}}{\sqrt{n}} \left\| \left\{ \frac{\|\phi_z\|_{\mathcal{L}^\infty_{\mathbb{P}}}}{a_z} \right\}_{z \in Z} \right\|_{p'} + L_D L_G \left\| \left\{ \frac{1}{a_z b_z} \right\}_{z \in \mathcal{Z} \setminus Z} \right\|_{\frac{1}{1-1/p-1/q}} \tag{3.3}$$

The two terms in the bound Eq. (3.3) 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 $\mathcal{F}_G$.

Corollary 8 (Sufficient Conditions for Parametric Rate). *Consider the setting of Theorem 7. If*

$$A := \sum_{z \in \mathcal{Z}} \frac{\|\phi_z\|^2_{\mathcal{L}^\infty_{\mathbb{P}}}}{a_z^2} < \infty \qquad \textit{and} \qquad \max\{a_z, b_z\} \to \infty.$$

whenever $\|z\| \to \infty$, then, the minimax rate is parametric; specifically, $M(\mathcal{F}_D, \mathcal{F}_G) \leq L_D\sqrt{A/n}$. In particular, letting $c_z := \sup_{x\in\mathcal{X}} |\phi_z(x)|$ for each $z \in \mathcal{Z}$, this occurs whenever $\sum_{z\in\mathcal{Z}} \frac{c_z^2}{a_z^2} < \infty$.

In many contexts (e.g., if $\mathbb{P} \ll \lambda$ and $\lambda \ll \mathbb{P}$), the simpler condition $\sum_{z\in\mathcal{Z}} \frac{c_z^2}{a_z^2} < \infty$ suffices. The first, and slightly weaker condition in terms of $\|\phi_z\|^2_{\mathcal{L}^\infty_P}$ is useful when we restrict $\mathcal{F}_G$; e.g., if $\mathcal{B}$ is the wavelet basis and $\mathcal{F}_G$ contains only discrete distributions supported on at most k points, then $\|\phi_{i,j}\|^2_{\mathcal{L}^\infty_P} = 0$ for all but k values of $j \in [2^i]$, at each resolution $i \in \mathbb{N}$. The assumption $\max\left\{\lim_{\|z\|\to\infty} a_z, \lim_{\|z\|\to\infty} b_z\right\} = \infty$ is quite mild; for example, the Riemann-Lebesgue lemma and the assumption that $\mathcal{F}_D$ is bounded in $\mathcal{L}_\lambda^\infty \subseteq \mathcal{L}_\lambda^1$ together imply that this condition always holds if $\mathcal{B}$ is the Fourier basis.

Singh et al. (2018) also lower bound the minimax risk $M(\mathcal{F}_D, \mathcal{F}_G)$ of distribution estimation under $d_{\mathcal{F}_D}$ loss over $\mathcal{F}_G$, for the case when $\mathcal{F}_D = \mathcal{H}_{p,a}$ and $\mathcal{F}_G := \mathcal{H}_{q,b}$ are generalized ellipses.

Theorem 9 (Minimax Lower Bound (Singh et al., 2018))**.** *Fix $\mathcal{X} = [0,1]^d$, and let p_0 denote the uniform density (with respect to Lebesgue measure) on $\mathcal{X}$. Suppose $\{p_0\} \cup \{\phi_z\}_{z\in\mathcal{Z}}$ is an orthonormal basis in $\mathcal{L}_\mu^2$, and $\{a_z\}_{z\in\mathcal{Z}}$ and $\{b_z\}_{z\in\mathcal{Z}}$ are two real-valued nets. Let $L_D, L_G \geq 0$ and $p, q \geq 2$. For any $Z \subseteq \mathcal{Z}$, let*

$$A_Z := |Z|^{1/2} \sup_{z\in Z} a_z \quad \text{and} \quad B_Z := |Z|^{1/2} \sup_{z\in Z} b_z.$$

Then, for $\mathcal{F}_D = \mathcal{H}_{p,a}(L_D)$ and $\mathcal{F}_G := \mathcal{H}_{q,b}(L_G)$, for any $Z \subseteq \mathcal{Z}$ satisfying

$$B_Z \geq 16 L_G \sqrt{\frac{n}{\log 2}} \quad \text{and} \quad 2\frac{L_G}{B_Z} \sum_{z\in Z} \|\phi_z\|_{\mathcal{L}_\mu^\infty} \leq 1, \tag{3.4}$$

we have $M(\mathcal{F}_D, \mathcal{F}_G) \geq \dfrac{L_G L_D |Z|}{64 A_Z B_Z} = \dfrac{L_G L_D}{64\left(\sup_{z\in Z} a_z\right)\left(\sup_{z\in Z} b_z\right)}$.

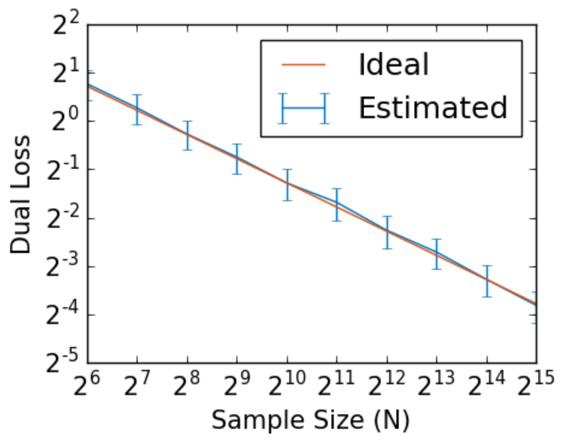
3.1.3 Examples on Different Loss Functions

In this section, we apply the above upper and lower bounds to compute concrete minimax convergence rates for different examples choices of $\mathcal{F}_D$ and $\mathcal{F}_G$, We show reproducing kernel Hilbert spaces to connect discussions in Chapter 2 and Sobolev spaces for later discussions in Chapter 4. We suppose that $\mathcal{X} = [0, 2\pi]^d$, $\mathcal{Z} = \mathbb{Z}^d$, and, for each $z \in \mathcal{Z}$, ϕ_z is the z^{th} standard Fourier basis element given by $\phi_z(x) = e^{i\langle z,x\rangle}$ for all $x \in \mathcal{X}$. In this case, we will always choose the truncation set Z to be of the form $Z := \{z \in \mathcal{Z} : \|z\|_\infty \leq \zeta\}$, for some $\zeta > 0$, so that $|Z| \leq \zeta^d$. Moreover, for every $z \in Z$, $\|\phi_z\|_{\mathcal{L}_\mu^\infty} = 1$, and hence $C_Z \leq 1$.

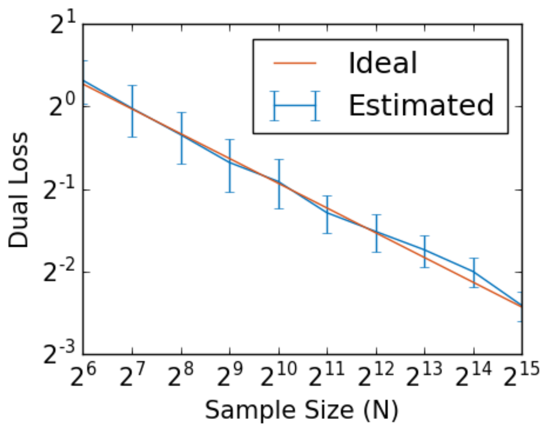
Example 1 (Reproducing Kernel Hilbert Space/MMD Loss)**.** *Suppose $\mathcal{H}_k$ is a reproducing kernel Hilbert space (RKHS) with reproducing kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (Aronszajn, 1950; Berlinet and Thomas-Agnan, 2011). If k is translation invariant (i.e., there exists $\kappa \in \mathcal{L}_\mu^2$ such that, for all $x, y \in \mathcal{X}$, $k(x,y) = \kappa(x-y)$), then Bochner's theorem (see, e.g., Theorem 6.6 of (Wendland, 2004)) implies that, up to constant factors,*

$$\mathcal{H}_k(L) := \{f \in \mathcal{H}_k : \|f\|_{\mathcal{H}_k} \leq L\} = \left\{ f \in \mathcal{H}_k : \sum_{z\in\mathcal{Z}} |\widetilde{\kappa}_z|^2 |\widetilde{f}_z|^2 < L^2 \right\}.$$

Thus, in the setting of Theorem 7, we have $\mathcal{H}_k = \mathcal{H}_{2,a}$, where $a_z = |\widetilde{\kappa}_z|$ satisfies $\sum_{z\in\mathcal{Z}} a_z^{-2} = \|\kappa\|_{\mathcal{L}_\mu^2}^2 < \infty$. Corollary 8 then gives $M(\mathcal{H}_k(L_D), \mathcal{F}_G) \leq L_D \|\kappa\|_{\mathcal{L}_\mu^2} n^{-1/2}$ for any class $\mathcal{F}_G$. It is well-known known that MMD can always be estimated at the parametric rate $n^{-1/2}$ (Gretton et al., 2012a); however, to the best of our knowledge, only recently has it



(a) Parametric Regime



(b) Nonparametric Regime

Figure 3.1: Simple synthetic experiment to showcase the tightness of the bound.

been shown that any probability distribution can be estimated at the rate $n^{-1/2}$ under MMD loss(Sriperumbudur et al., 2016), emphasizing the fact that MMD is a very weak metric. This has important implications for applications such as two-sample testing (Ramdas et al., 2015).

Example 2 (Sobolev Spaces)**.** *Suppose that, for some $s, t \geq 0$, $a_z = \|z\|_\infty^s$ and $b_z = \|z\|_\infty^t$. Then, setting $\zeta = n^{\frac{1}{2t+d}}$ in Theorems 7 and 9 gives that there exist constants $C > c > 0$ such that*

$$cn^{-\min\left\{\frac{1}{2}, \frac{s+t}{2t+d}\right\}} \leq M\left(\mathcal{W}^{s,2}, \mathcal{W}^{t,2}\right) \leq Cn^{-\min\left\{\frac{1}{2}, \frac{s+t}{2t+d}\right\}}. \tag{3.5}$$

Combining the observation that the s-Hölder space $\mathcal{W}^{s,\infty} \subseteq \mathcal{W}^{s,2}$ with the lower bound (over $\mathcal{W}^{s,\infty}$) in Theorem 3.1 of (Liang, 2017), we have that Eq. (3.5) also holds when $\mathcal{W}^{s,2}$ is replaced with $\mathcal{W}^{s,p}$ for any $p \in [2, \infty]$ (e.g., in the case of the Wasserstein metric $d_{\mathcal{W}^{1,\infty}}$).

3.1.4 Experiments

Here we present some empirical results in Figure 3.1 supporting the theoretical bounds above. First, we consider an example with a finite basis, which should yield the parametric $n^{-1/2}$ rate. In particular, we construct the true distribution $\mathbb{P}$ to consist of 6 randomly chosen basis functions in the Fourier basis. We employ the truncated series estimator $\widehat{\mathbb{P}}$ of Eq. (3.2) in the same basis using different number of samples n and compute the distance $d_{\mathcal{F}_D}\left(\mathbb{P}, \widehat{\mathbb{P}}\right)$. Under this setting, the maximization problem of Eq. (3.1) needed to evaluate this distance can be solved in closed form. The risk empirically appears to closely follow our derived minimax rate of $n^{-1/2}$, as shown in Figure 3.1a. Next, we consider a non-parametric case, in which the number of active basis elements increases as the function of n, weighted such that Eq. (3.5) predicts a rate of $n^{-1/3}$. As expected, the estimated risk, shown in Figure 3.1b, closely resembles the rate of $n^{-1/3}$.

3.1.5 Adversarial Density Estimation with Neural Networks

This section discusses implications of our minimax bounds for GANs. Neural networks in this section are assumed to be fully-connected, with rectified linear unit (ReLU) activations. Key ingredients are an oracle inequality proven in Liang (2017), an upper bound such as Theorem 7 and bounds of Yarotsky (2017) on the size of a neural network needed to approximate functions in a Sobolev class. We note that the adversarial density estimation with neural networks discussed here is different from IGM settings in Chapter 1. The generator network here is used to approximate density functions, which is still an *explicit generative model*. The detailed connection between adversarial density estimation (explicit generative models) and IGM will be discussed next.

In the following, $\mathcal{F}_D$ denotes the set of functions that can be encoded by the discriminator network and $\mathcal{F}_G$ denotes the set of distributions that can be encoded by the generator network. $\mathbb{P}_n := \frac{1}{n}\sum_{i=1}^n 1_{\{X_i\}}$ denotes the empirical distribution of the observed data $X_{1:n} \overset{IID}{\sim} \mathbb{P}$.

Theorem 10 (Improvement of Theorem 3.1 in Liang (2017))**.** *Let $s, t > 0$, and fix a desired approximation accuracy $\epsilon > 0$. Then, there exists a GAN architecture, in which*

1. *the discriminator $\mathcal{F}_D$ has at most $O(\log(1/\epsilon))$ layers and $O(\epsilon^{-d/s}\log(1/\epsilon))$ parameters,*
2. *and the generator $\mathcal{F}_G$ has at most $O(\log(1/\epsilon))$ layers and $O(\epsilon^{-d/t}\log(1/\epsilon))$ parameters,*

$$\textit{such that, if } \widehat{\mathbb{P}}_*(X_{1:n}) := \operatorname*{argmin}_{\widehat{\mathbb{P}}\in\mathcal{F}_G} d_{\mathcal{F}_D}\left(\mathbb{P}_n, \widehat{\mathbb{P}}\right), \textit{ is the optimized GAN estimate of } \mathbb{P},$$

$$\textit{then} \quad \sup_{\mathbb{P}\in\mathcal{W}^{t,2}} \mathop{\mathbb{E}}_{X_{1:n}} \left[d_{\mathcal{W}^{s,2}}\left(\mathbb{P}, \widehat{\mathbb{P}}_*(X_{1:n})\right)\right] \leq C\left(\epsilon + n^{-\min\left\{\frac{1}{2}, \frac{s+t}{2t+d}\right\}}\right).$$

The discriminator and generator in the above theorem can be implemented as described in Yarotsky (2017). The assumption that the GAN is perfectly optimized may be strong; see Liang and Stokes (2018); Nagarajan and Kolter (2017) for discussion of this.

3.2 Minimax Comparison of Explicit and Implicit Generative Models

In this section, we draw formal connections between adversarial density estimation (explicit generative modeling) and learning IGMs under an appropriate measure of risk. In the sequel, we fix a class $\mathcal{F}_G$ of probability measures on a sample space $\mathcal{X}$ and a loss function $\ell : \mathcal{F}_G \times \mathcal{F}_G \to [0, \infty]$ measuring the distance of an estimate $\widehat{\mathbb{P}}$ from the true distribution $\mathbb{P}$. ℓ need not be an adversarial loss $d_{\mathcal{F}_D}$, but our discussion does apply to all ℓ of this form.

3.2.1 A Minimax Framework for Implicit Generative Models

Thus far, we have discussed the *minimax risk of density estimation* in Section 3.1, namely

$$M_D(\mathcal{F}_G, \ell, n) = \inf_{\widehat{\mathbb{P}}} \sup_{\mathbb{P} \in \mathcal{F}_G} R_D(\mathbb{P}, \widehat{\mathbb{P}}), \text{ where } R_D(\mathbb{P}, \widehat{\mathbb{P}}) = \mathop{\mathbb{E}}_{X_{1:n} \overset{IID}{\sim} \mathbb{P}} \left[\ell(\mathbb{P}, \widehat{\mathbb{P}}(X_{1:n})) \right] \tag{3.6}$$

denotes the *density estimation risk of* $\widehat{\mathbb{P}}$ *at* $\mathbb{P}$ and the infimum is taken over all estimators (i.e., (potentially randomized) functions $\widehat{\mathbb{P}} : \mathcal{X}^n \to \mathcal{F}_G$). Whereas density estimation is a classical statistical problem to which we have already contributed novel results, our motivations for studying this problem arose from a desire to better understand recent work on IGMs.

IGMs address the problem of *sampling* as we discussed in Chapter 1, in which we seek to construct a *generator* (transformation function) that produces novel samples from the distribution $\mathbb{P}$ (Mohamed and Lakshminarayanan, 2016). In our context, a generator is a function $\widehat{X} : \mathcal{X}^n \times \mathcal{Z} \to \mathcal{X}$ that takes in n IID samples $X_{1:n} \sim \mathbb{P}$ and a source of randomness $Z \sim \mathbb{P}_{\mathcal{Z}}$ with known distribution $\mathbb{P}_{\mathcal{Z}}$ (independent of $X_{1:n}$) on a space $\mathcal{Z}$, and returns a novel sample $\widehat{X}(X_{1:n}, Z) \in \mathcal{X}$.

The evaluating the performance of implicit generative models, both in theory and in practice, is difficult, with solutions continuing to be proposed (Bińkowski et al., 2018; Lucic et al., 2018; Salimans et al., 2016; Sutherland et al., 2017), some of which have proven controversial. Some of this controversy stems from the fact that many of the most straightforward evaluation objectives are optimized by a trivial generator that 'memorizes' the training data (e.g., $\widehat{X}(X_{1:n}, Z) = X_Z$, where Z is uniformly distributed on $[n]$). One objective that can avoid this problem is as follows. For simplicity, fix the distribution $\mathbb{P}_{\mathcal{Z}}$ of the source randomness $Z \sim \mathbb{P}_{\mathcal{Z}}$ (e.g., $\mathbb{P}_{\mathcal{Z}} = \mathcal{N}(0, I)$). For a fixed training set $X_{1:n} \overset{IID}{\sim} \mathbb{P}$ and base distribution $Z \sim \mathbb{P}_{\mathcal{Z}}$, we define the *implicit distribution of a generator* $\widehat{X}$ as the conditional distribution $\mathbb{P}_{\widehat{X}(X_{1:n},Z)|X_{1:n}}$ over $\mathcal{X}$ of the random variable $\widehat{X}(X_{1:n}, Z)$ given the training data. Then, for any $\mathbb{P} \in \mathcal{F}_G$, we define the *implicit risk of* $\widehat{X}$ *at* $\mathbb{P}$ by

$$R_I(\mathbb{P}, \widehat{X}) := \mathop{\mathbb{E}}_{X_{1:n} \sim \mathbb{P}} \left[\ell(\mathbb{P}, \mathbb{P}_{\widehat{X}(X_{1:n},Z)|X_{1:n}}) \right].$$

We can then study the *minimax risk of sampling*, $M_I(\mathcal{F}_G, \ell, n) := \inf_{\widehat{X}} \sup_{\mathbb{P} \in \mathcal{F}_G} R_I(\mathbb{P}, \widehat{X})$.

Second, since the risk $R_I(\mathbb{P}, \widehat{X})$ depends on the unknown true distribution $\mathbb{P}$, we cannot calculate it in practice. Third, for the same reason (because $R_{\mathbb{P}}(\mathbb{P}, \widehat{X})$ depends directly on $\mathbb{P}$ rather than particular data $X_{1:n}$), it detect lack-of-diversity issues such as mode collapse.

3.2.2 Comparison of Explicit and Implicit Generative Models

Algorithmically, sampling is a very distinct problem from density estimation; for example, many computationally efficient Monte Carlo samplers rely on the fact that a function *proportional* to the density of interest can be computed much more quickly than the exact (normalized) density function (Chib and Greenberg, 1995). In this section, we show that, given unlimited computational resources, the problems of density estimation and sampling are equivalent in a minimax statistical sense. Since exactly minimax estimators ($\text{argmin}_{\widehat{\mathbb{P}}} \sup_{\mathbb{P} \in \mathcal{F}_G} R_D(\mathbb{P}, \widehat{\mathbb{P}})$) often need not exist, the following weaker notion is useful for stating our results:

Definition 11 (Nearly Minimax Sequence)**.** *A sequence $\{\widehat{\mathbb{P}}_k\}_{k \in \mathbb{N}}$ of density estimators (resp., $\{\widehat{X}_k\}_{k \in \mathbb{N}}$ of generators) is called nearly minimax over $\mathcal{F}_G$ if*

$$\lim_{k \to \infty} \sup_{\mathbb{P} \in \mathcal{F}_G} R_{\mathbb{P},D}(\widehat{\mathbb{P}}_k) = M_D(\mathcal{F}_G, \ell, n)$$

(resp., $\lim_{k \to \infty} \sup_{\mathbb{P} \in \mathcal{F}_G} R_{\mathbb{P},I}(\widehat{X}_k) = M_I(\mathcal{F}_G, \ell, n)$).

The following theorem identifies sufficient conditions under which, in the statistical minimax framework described above, density estimation is no harder than sampling. The idea behind the proof is as follows: If we have a good sampler $\widehat{X}$ (i.e., with $R_I(\widehat{X})$ small), then we can draw m 'fake' samples from $\widehat{X}$. We can use these 'fake' samples to construct a density estimate $\widehat{\mathbb{P}}$ of the implicit distribution of $\widehat{X}$ such that, under the technical assumptions below, $R_D(\widehat{\mathbb{P}}) - R_I(\widehat{X}) \to 0$ as $m \to \infty$.

Theorem 12 (Conditions under which Density Estimation is Statistically no harder than Sampling)**.** *Let $\mathcal{F}_G$ be a family of probability distributions on a sample space $\mathcal{X}$. Suppose*

(A1) *$\ell : \mathcal{P} \times \mathcal{P} \to [0, \infty]$ is non-negative, and there exists $C_\triangle > 0$ such that, for all $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3 \in \mathcal{F}_G$, $\ell(\mathbb{P}_1, \mathbb{P}_3) \leq C_\triangle \left(\ell(\mathbb{P}_1, \mathbb{P}_2) + \ell(\mathbb{P}_2, \mathbb{P}_3)\right)$.*

(A2) *$M_D(\mathcal{F}_G, \ell, m) \to 0$ as $m \to \infty$.*

(A3) *For all $m \in \mathbb{N}$, we can draw m IID samples $Z_1, ..., Z_m \overset{IID}{\sim} \mathbb{P}_{\mathcal{Z}}$ of the latent variable Z.*

(A4) *there exists a nearly minimax sequence of samplers $\widehat{X}_k : \mathcal{X}^n \times \mathcal{Z} \to \mathcal{X}$ such that, for each $k \in \mathbb{N}$, almost surely over $X_{1:n}$, $\mathbb{P}_{\widehat{X}_k(X_{1:n}, Z) | X_{1:n}} \in \mathcal{F}_G$.*

Then, $M_D(\mathcal{F}_G, \ell, n) \leq C_\triangle M_I(\mathcal{F}_G, \ell, n)$.

Assumption (A1) is a generalization of the triangle inequality (and reduces to the triangle inequality when $C_\triangle = 1$). This weaker assumption applies, for example, when ℓ is the Jensen-Shannon divergence (with $C_\triangle = 2$) used in the original GAN formulation of Goodfellow et al. (2014), even though this does not satisfy the triangle inequality (Endres and Schindelin, 2003)). Assumption (A2) is equivalent to the existence of a uniformly ℓ-risk-consistent estimator over $\mathcal{F}_G$, a standard property of most distribution classes $\mathcal{F}_G$ over which density estimation is studied (e.g., our Theorem 7). Assumption (A3) is a natural design criterion of IGMs; usually, $\mathbb{P}_{\mathcal{Z}}$ is a simple parametric distribution such as a standard normal.

Finally, Assumption (A4) is the most mysterious, because, currently, little is known about the minimax theory of samplers when $\mathcal{F}_G$ is a large space. On one hand, since $M_I(\mathcal{F}_G, \ell, n)$ is an infimum over $\widehat{X}$, Theorem 12 continues to hold if we restrict the class of samplers (e.g., to those satisfying Assumption (A4) or those we can compute). On the other hand, even without restricting $\widehat{X}$, this assumption may not be too restrictive, because nearly minimax samplers are necessarily close to $\mathbb{P} \in \mathcal{F}_G$. For example, if $\mathcal{F}_G$ contains only smooth distributions but $\widehat{X}$ is the trivial empirical sampler described above, then $\ell(\mathbb{P}, \mathbb{P}_{\widehat{X}})$ should be large and $\widehat{X}$ is unlikely to be minimax optimal.

Finally, in practice, we often do not know estimators that are nearly minimax for finite samples, but may have estimators that are rate-optimal (e.g., as given by Theorem 7), i.e., that satisfy

$$C := \limsup_{n \to \infty} \frac{\sup_{\mathbb{P} \in \mathcal{F}_G} R_I(\mathbb{P}, \widehat{X})}{M_I(\mathcal{F}_G, \ell, n)} < \infty.$$

Under this weaker assumption, it is straightforward to modify our proof to conclude that

$$\limsup_{n \to \infty} \frac{M_D(\mathcal{F}_G, \ell, n)}{M_I(\mathcal{F}_G, \ell, n)} \leq C_\triangle C.$$

The converse result ($M_D(\mathcal{F}_G, \ell, n) \geq M_I(\mathcal{F}_G, \ell, n)$) is simple to prove in many cases, and is related to the well-studied problem of Monte Carlo sampling (Robert, 2004).

Theorem 13 (Conditions under which Sampling is Statistically no harder than Density Estimation)**.** *Suppose that, there exists as nearly minimax sequence $\{\widehat{\mathbb{P}}_k\}_{k \in \mathbb{N}}$ such that, for any $k \in \mathbb{N}$, we can draw a random sample $\widehat{X}$ from $\widehat{\mathbb{P}}_k(X_{1:n})$. Then,*

$$M_D(\mathcal{F}_G, \ell, n) \geq M_I(\mathcal{F}_G, \ell, n).$$

The assumption above that we can draw samples from a nearly minimax sequence of estimators if not particularly insightful, but techniques for drawing such samples have been widely studied in the vast literature of Monte Carlo sampling (Robert, 2004). As an example, if $\widehat{\mathbb{P}}$ is a kernel density estimator with kernel K, then, recalling that K is itself a probability density, of which $\widehat{\mathbb{P}}$ is a mixture, we can sample from $\widehat{\mathbb{P}}$ simply by choosing a sample uniformly from $X_{1:n}$ and adding noise $\epsilon \sim K$. Alternatively, if $\widehat{\mathbb{P}}$ is bounded and has bounded support, then one can perform rejection sampling.

Proof. Since, by definition of the implicit distribution of $\widehat{X}$,

$$\mathbb{P}_{\widehat{X}(X_{1:n}, Z) | X_{1:n}} = \widehat{\mathbb{P}}(X_{1:n})$$

is precisely the implicit distribution of $\widehat{X}$, we trivially have

$$M_I(\mathcal{F}_G, \ell, n) \leq \sup_{\mathbb{P} \in \mathcal{F}_G} \mathop{\mathbb{E}}_{X_{1:n} \overset{IID}{\sim} \mathbb{P}} \left[\ell\left(\mathbb{P}, \mathbb{P}_{\widehat{X}(X_{1:n}, Z) | X_{1:n}} \right) \right]$$

□

3.3 Summary

Given the recent popularity of IGMs in many applications, it is important to theoretically understand why these models appear to outperform classical methods for similar problems. We review new minimax bounds for density estimation under adversarial losses, both with and without adaptivity to smoothness, and gave applications, including both traditional statistical settings and perfectly optimized GANs. We also gave simple conditions under which minimax bounds for density estimation imply bounds for the problem of implicit generative modeling, suggesting that sampling is typically not *statistically* easier than density estimation. Thus, for example, the strong curse of dimensionality that is known to afflict to nonparametric density estimation (Wasserman, 2006) should also limit the performance of implicit generative models such as GANs. One possible avenues for further investigation is if the curse of dimensionality can be avoided when data lie on a low-dimensional manifold.

3.4 Appendix: Proof of Theorem 12

Here, we prove Theorem 12 from the main text, provide some discussion of when the converse direction $M_I(\mathcal{P}, \ell, n) \leq M_D(\mathcal{P}, \ell, n)$ holds.

The assumption (A2) implies that there exists a sequence $\{\widehat{\mathbb{P}}_m\}_{m \in \mathbb{N}}$ of density estimators $\widehat{\mathbb{P}}_m : \mathcal{X}^m \to \mathcal{P}$ that is uniformly consistent in ℓ over $\mathcal{P}$; that is,

$$\lim_{m \to \infty} \sup_{\mathbb{P} \in \mathcal{P}} \mathop{\mathbb{E}}_{Y_{1:m} \overset{IID}{\sim} \mathbb{P}} \left[\ell \left(\mathbb{P}, \widehat{\mathbb{P}}_m(Y_{1:m}) \right) \right]. \tag{3.7}$$

For brevity, we use the abbreviation $\mathbb{P}_{\widehat{X}_k} = \mathbb{P}_{\widehat{X}_k(X_{1:n},Z)|X_{1:n}}$ in the rest of this proof to denote the conditional distribution of the 'fake data' generated by $\widehat{X}_k$ given the true data. Recalling that the minimax risk is at most the risk of any particular sampler, we have

$$\begin{aligned} M_D(\mathcal{P}, \ell, n) &:= \inf_{\widehat{\mathbb{P}}} \sup_{\mathbb{P} \in \mathcal{P}} \mathop{\mathbb{E}}_{\substack{X_{1:n} \overset{IID}{\sim} \mathbb{P} \\ Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z}} \left[\ell \left(\mathbb{P}, \widehat{\mathbb{P}}(X_{1:n}) \right) \right] \\ &\leq \sup_{\mathbb{P} \in \mathcal{P}} \mathop{\mathbb{E}}_{\substack{X_{1:n} \overset{IID}{\sim} \mathbb{P} \\ Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z}} \left[\ell \left(\mathbb{P}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right]. \end{aligned}$$

Taking $\lim_{m\to\infty}$ gives, by Tonelli's theorem and non-negativity of ℓ,

$$\begin{aligned}
&M_D(\mathcal{P}, \ell, n) \\
&\leq \lim_{m\to\infty} \sup_{\mathbb{P}\in\mathcal{P}} \mathop{\mathbb{E}}_{\substack{X_{1:n} \overset{IID}{\sim} \mathbb{P} \\ Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z}} \left[\ell\left(\mathbb{P}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right] \\
&\leq C_\triangle \lim_{m\to\infty} \sup_{\mathbb{P}\in\mathcal{P}} \mathop{\mathbb{E}}_{\substack{X_{1:n} \overset{IID}{\sim} \mathbb{P} \\ Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z}} \left[\ell\left(\mathbb{P}, \mathbb{P}_{\widehat{X}_k} \right) + \ell\left(\mathbb{P}_{\widehat{X}_k}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right] \\
&\leq C_\triangle \lim_{m\to\infty} \sup_{\mathbb{P}\in\mathcal{P}} \mathop{\mathbb{E}}_{\substack{X_{1:n} \overset{IID}{\sim} \mathbb{P} \\ Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z}} \left[\ell\left(\mathbb{P}, \mathbb{P}_{\widehat{X}_k} \right) + \ell\left(\mathbb{P}_{\widehat{X}_k}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right] \\
&\leq C_\triangle \sup_{\mathbb{P}\in\mathcal{P}} \mathop{\mathbb{E}}_{X_{1:n} \overset{IID}{\sim} \mathbb{P}} \left[\ell\left(\mathbb{P}, \mathbb{P}_{\widehat{X}_k} \right) \right] \qquad (3.8) \\
&+ C_\triangle \lim_{m\to\infty} \sup_{\mathbb{P}\in\mathcal{P}} \mathop{\mathbb{E}}_{\substack{X_{1:n} \overset{IID}{\sim} \mathbb{P} \\ Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z}} \left[\ell\left(\mathbb{P}_{\widehat{X}_k}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right]. \qquad (3.9)
\end{aligned}$$

In the above, we upper bounded $M_D(\mathcal{P}, \ell, n)$ by the sum of two terms, (3.8) and (3.9). Since the sequence $\{\widehat{X}_k\}_{k\in\mathbb{N}}$ is nearly minimax, if we were to take an infimum over $k \in \mathbb{N}$ on both sides, the term (3.8) would become precisely $C_\triangle M_I(\mathcal{P}, \ell, n)$. Therefore, it suffices to observe that the second term (3.9) is 0. Indeed, by the assumption that $\mathbb{P}_{\widehat{X}_k} \in \mathcal{P}$ for all $X_{1:n} \in \mathcal{X}$ and the uniform consistency assumption (3.7),

$$\begin{aligned}
&\lim_{m\to\infty} \sup_{\mathbb{P}\in\mathcal{P}} \mathop{\mathbb{E}}_{\substack{X_{1:n} \overset{IID}{\sim} \mathbb{P} \\ Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z}} \left[\ell\left(\mathbb{P}_{\widehat{X}_k}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right] \\
&\leq \lim_{m\to\infty} \sup_{\mathbb{P}\in\mathcal{P}, X_{1:n} \overset{IID}{\sim} \mathbb{P}} \mathop{\mathbb{E}}_{Z_{1:m} \overset{IID}{\sim} \mathbb{Q}_Z} \left[\ell\left(\mathbb{P}_{\widehat{X}_k}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right] \\
&\leq \lim_{m\to\infty} \sup_{\mathbb{P}'\in\mathcal{P}} \mathop{\mathbb{E}}_{X_{n+1:n+m} \overset{IID}{\sim} \mathbb{P}'} \left[\ell\left(\mathbb{P}, \widehat{\mathbb{P}}_m(X_{n+1:n+m}) \right) \right] = 0.
\end{aligned}$$

Part II

Implicit Generative Models by Leveraging Structures and Priors of Data

Chapter 4

Text Generation with Sobolev IPM

In order to learn implicit generative models, it is now well established that the generator should mimic the distribution of real data, in the sense of a certain discrepancy measure. Discrepancies between distributions that measure the goodness of the fit of the neural generator to the real data distribution has been the subject of many recent studies (Arjovsky and Bottou, 2017; Arjovsky et al., 2017; Goodfellow et al., 2014; Gulrajani et al., 2017; Kaae Sønderby et al., 2017; Li et al., 2017; Mao et al., 2017; Mroueh and Sercu, 2017; Mroueh et al., 2017; Nowozin et al., 2016), most of which focus on training stability.

In terms of data modalities, most success was booked in plausible natural image generation after Radford et al. (2016). This success is not only due to advances in training generative adversarial networks in terms of loss functions (Arjovsky et al., 2017) and stable algorithms, but also to the representation power of convolutional neural networks in modeling images and in finding sufficient statistics that capture the *continuous* density function of natural images. When moving to neural generators of *discrete sequences*, compared with extensively studied MLE training (Sutskever et al., 2014), theory and practice of IGMs are still not very well understood. Maximum likelihood pre-training or augmentation, in conjunction with the use of reinforcement learning techniques were proposed in many recent works for training GANs for discrete sequences generation (Che et al., 2017; Hjelm et al., 2018; Rajeswar et al., 2017; Yu et al., 2017). Other methods included using the Gumbel Softmax trick (Kusner and Hernández-Lobato, 2016) and the use of autoencoders to generate adversarially discrete sequences from a continuous space (Zhao et al., 2017). *End to end* training of IGMs (GANs) for discrete sequence generation is still an open problem (Press et al., 2017). Empirical successes of end to end training have been reported within the framework of WGAN-GP (Gulrajani et al., 2017), using a proxy for the Wasserstein distance via a *point-wise gradient penalty* on the critic. Inspired by this success, we propose in this chapter a new Integral Probability Metric (IPM) between distributions that we coin *Sobolev IPM*. Intuitively an IPM (Müller, 1997) between two probability distributions looks for a witness function f, called critic, that maximally discriminates between samples coming from the two distributions:

$$\sup_{f\in\mathcal{F}} \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x).$$

Traditionally, the function f is defined over a function class $\mathcal{F}$ that is independent to the distributions at hand (Sriperumbudur et al., 2012). The Wasserstein-1 distance corresponds for instance to an IPM where the witness functions are defined over the space of Lipschitz functions; The MMD distance (Gretton et al., 2012a) corresponds to witness functions defined over a ball in a Reproducing Kernel Hilbert Space (RKHS). Detailed discussions of IPM are in Chapter 3.

We will firstly revisit Fisher IPM defined in Mroueh and Sercu (2017), which extends the IPM definition to function classes defined with norms that depend on the distributions. Fisher IPM can be seen as restricting the *critic* to a Lebsegue ball defined with respect to a dominant measure μ. The Lebsegue norm is defined as follows:

$$\int_{\mathcal{X}} f^2(x)\mu(x)dx.$$

where μ is a dominant measure of $\mathbb{P}$ and $\mathbb{Q}$.

In Chapter 2, we propose an MMD extension for learning IGMs without leveraging properties of data. In this chapter, we design a new probability distance for matching discrete sequences. We extend the IPM framework to critics bounded in from the Lebsegue norm to the Sobolev norm:

$$\int_{\mathcal{X}} \|\nabla_x f(x)\|_2^2 \, \mu(x)dx,$$

In contrast to Fisher IPM, which compares joint *probability density functions* of all coordinates between two distributions, we will show that Sobolev IPM compares *weighted (coordinate-wise) conditional Cumulative Distribution Functions* for all coordinates on a leave on out basis. Matching conditional dependencies between coordinates is crucial for **sequence modeling**. We note that, in Chapter 3, we analyze the case when the critic functions are in Sobolev space, which focuses on the smoothness assumption of function spaces and the sample complexity only.

Our analysis and empirical verification show that modeling the conditional dependencies can be built in to the metric used to learn IGMs as in Sobolev IPM. For instance, this gives an advantage to Sobolev IPM in comparing sequences over Fisher IPM. Nevertheless, in sequence modeling when we parametrize the critic and the generator with a neural network, we find an interesting tradeoff between the metric used and the architectures used to parametrize the critic and the generator as well as the conditioning used in the generator. The burden of modeling the conditional long term dependencies can be handled by the IPM loss function as in Sobolev IPM (more accurately the choice of the data dependent function class of the critic) or by a simpler metric such as Fisher IPM together with a powerful architecture for the critic that models conditional long term dependencies such as LSTM or GRUs in conjunction with a curriculum conditioning of the generator as done in Press et al. (2017). Highlighting those interesting tradeoffs between metrics, data dependent functions classes for the critic (Fisher or Sobolev) and architectures is crucial to advance sequence modeling and more broadly structured data generation using GANs.

On the other hand, Sobolev norms have been widely used in manifold regularization in the so called Laplacian framework for semi-supervised learning (SSL) (Belkin et al., 2006). GANs have shown success in semi-supervised learning (Dai et al., 2017; Dumoulin et al., 2017; Kumar et al., 2017; Salimans et al., 2016). Nevertheless, many normalizations and additional tricks were needed. We show in the end of this chapter that a variant of Sobolev GAN achieves strong results in semi-supervised learning on CIFAR-10, without the need of any activation normalization in the critic.

4.1 Notations

In what follows, we consider probability measures with a positive weakly differentiable probability density functions (PDF). Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability measures. Let $F_{\mathbb{P}}$ and $F_{\mathbb{Q}}$ be the Cumulative Distribution Functions (CDF) of $\mathbb{P}$ and $\mathbb{Q}$ respectively:

$$F_{\mathbb{P}}(x) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_d} \mathbb{P}(x_1, \dots x_d) dx.$$

The score function of a density function is defined as: $s_{\mathbb{P}}(x) = \nabla_x \log(\mathbb{P}(x)) \in \mathbb{R}^d$.

4.2 Review Fisher IPM: PDF Comparison

Imposing data-independent constraints on the function class in the IPM framework, such as the Lipschitz constraint in the Wasserstein distance is computationally challenging and intractable for the general case. In this section, we generalize the Fisher IPM introduced in Mroueh and Sercu (2017), where the function class is relaxed to a *tractable data dependent constraint* on the second order moment of the critic, in other words the critic is constrained to be in a Lebsegue ball.

Fisher IPM. Let $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{P}(\mathcal{X})$ be the space of distributions defined on $\mathcal{X}$. Let $\mathbb{P}, \mathbb{Q} \in \mathcal{P}(\mathcal{X})$, and μ be a dominant measure of $\mathbb{P}$ and $\mathbb{Q}$, in the sense that

$$\mu(x) = 0 \implies \mathbb{P}(x) = 0 \text{ and } \mathbb{Q}(x) = 0.$$

We assume μ to be also a distribution in $\mathcal{P}(\mathcal{X})$, and assume $\mu(x) > 0$, $\forall x \in \mathcal{X}$. Let $\mathcal{L}_2(\mathcal{X}, \mu)$ be the space of μ-measurable functions. For $f, g \in \mathcal{L}_2(\mathcal{X}, \mu)$, we define the following dot product and its corresponding norm:

$$\langle f, g \rangle_{\mathcal{L}_2(\mathcal{X},\mu)} = \int_{\mathcal{X}} f(x) g(x) \mu(x) dx, \quad \|f\|_{\mathcal{L}_2(\mathcal{X},\mu)} = \sqrt{\int_{\mathcal{X}} f^2(x) \mu(x) dx}.$$

Note that $\mathcal{L}_2(\mathcal{X}, \mu)$, can be formally defined as follows:

$$\mathcal{L}_2(\mathcal{X}, \mu) = \{f : \mathcal{X} \to \mathbb{R} \text{ s.t } \|f\|_{\mathcal{L}_2(\mathcal{X},\mu)} < \infty\}.$$

We define the unit Lebesgue ball as follows:

$$B_2(\mathcal{X}, \mu) = \{f \in \mathcal{L}_2(\mathcal{X}, \mu), \|f\|_{\mathcal{L}_2(\mathcal{X},\mu)} \leq 1\}.$$

Fisher IPM defined in Mroueh and Sercu (2017), searches for the critic function *in the Lebesgue Ball* $B_2(\mathcal{X}, \mu)$ that maximizes the mean discrepancy between $\mathbb{P}$ and $\mathbb{Q}$. Fisher

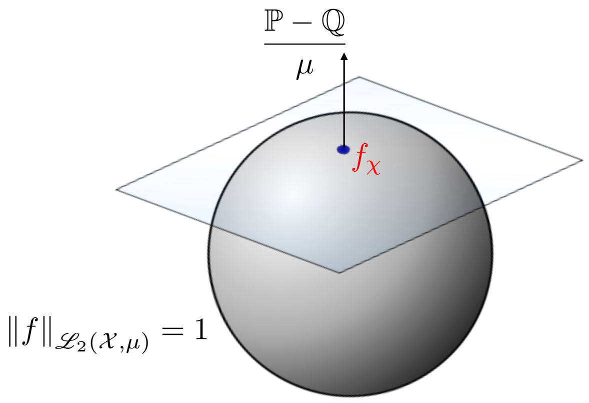

$\frac{\mathbb{P}-\mathbb{Q}}{\mu}$
f_χ
$\|f\|_{\mathscr{L}_2(\mathcal{X},\mu)} = 1$

Figure 4.1: An example of Fisher ball constraints.

GAN (Mroueh and Sercu, 2017) was originally formulated specifically for $\mu = \frac{1}{2}(\mathbb{P}+\mathbb{Q})$. We consider here a general μ as long as it dominates $\mathbb{P}$ and $\mathbb{Q}$. We define Generalized Fisher IPM as follows:

$$\mathcal{F}_\mu(\mathbb{P},\mathbb{Q}) = \sup_{f\in B_2(\mathcal{X},\mu)} \mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x) \tag{4.1}$$

Note that:

$$\mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x) = \left\langle f, \frac{\mathbb{P}-\mathbb{Q}}{\mu}\right\rangle_{\mathcal{L}_2(\mathcal{X},\mu)}.$$

Hence Fisher IPM can be written as follows:

$$\mathcal{F}_\mu(\mathbb{P},\mathbb{Q}) = \sup_{f\in B_2(\mathcal{X},\mu)} \left\langle f, \frac{\mathbb{P}-\mathbb{Q}}{\mu}\right\rangle_{\mathcal{L}_2(\mathcal{X},\mu)} \tag{4.2}$$

We have the following result:

Theorem 14 (Generalized Fisher IPM)**.** *The Fisher distance and the optimal critic are as follows:*

1. *The Fisher distance is given by:*

$$\mathcal{F}_\mu(\mathbb{P},\mathbb{Q}) = \left\|\frac{\mathbb{P}-\mathbb{Q}}{\mu}\right\|_{\mathcal{L}_2(\mathcal{X},\mu)} = \sqrt{\mathbb{E}_{x\sim\mu}\left(\frac{\mathbb{P}(x)-\mathbb{Q}(x)}{\mu(x)}\right)^2}.$$

2. *The optimal f_χ achieving the Fisher distance $\mathcal{F}_\mu(\mathbb{P},\mathbb{Q})$ is:*

$$f_\chi = \frac{1}{\mathcal{F}(\mathbb{P},\mathbb{Q})}\frac{\mathbb{P}-\mathbb{Q}}{\mu}, \boldsymbol{\mu} \textit{ almost surely.}$$

Proof of Theorem 14. From Eq. (4.2), the optimal f_χ belong to the intersection of the hyperplane that has normal $n = \frac{\mathbb{P}-\mathbb{Q}}{\mu}$, and the ball $B_2(\mathcal{X},\mu)$, hence $f_\chi = \frac{n}{\|n\|_{\mathcal{L}_2(\mathcal{X},\mu)}}$. Hence $\mathcal{F}(\mathbb{P},\mathbb{Q}) = \|n\|_{\mathcal{L}_2(\mathcal{X},\mu)}$. An example is shown in Figure 4.1. □

We see from Theorem 14 the role of the dominant measure μ: the optimal critic is defined with respect to this measure and the overall Fisher distance can be seen as an average weighted distance between *probability density functions*, where the average is taken on points sampled from μ. We give here some choices of μ:

1. For $\mu = \frac{1}{2}(\mathbb{P}+\mathbb{Q})$, we obtain the symmetric chi-squared distance as defined in Mroueh and Sercu (2017).
2. μ_{GP}, the implicit distribution defined by the interpolation lines between $\mathbb{P}$ and $\mathbb{Q}$ as in Gulrajani et al. (2017).
3. When μ does not dominate $\mathbb{P}$, and $\mathbb{Q}$, we obtain a non symmetric divergence. For example for $\mu = \mathbb{P}$, $\mathcal{F}^2_{\mathbb{P}}(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} \frac{(\mathbb{P}(x)-\mathbb{Q}(x))^2}{\mathbb{P}(x)}dx$. We see here that for this particular choice we obtain the Pearson divergence.

4.3 Sobolev IPM

In this section, we introduce the Sobolev IPM. In a nutshell, the Sobolev IPM constrains the critic function to belong to a ball in the restricted Sobolev space. In other words we constrain the norm of the gradient of the critic $\nabla_x f(x)$. We will show that by moving from a Lebesgue constraint as in Fisher IPM to a Sobolev constraint as in Sobolev IPM, the metric changes from a joint PDF matching to weighted (ccordinate-wise) conditional CDFs matching. The intrinsic conditioning built in to the Sobolev IPM and the comparison of cumulative distributions makes Sobolev IPM suitable for comparing discrete sequences.

4.3.1 Definition and Expression of Sobolev IPM in terms of Coordinate Conditional CDFs

We will start by recalling some definitions on Sobolev spaces. We assume in the following that $\mathcal{X}$ is compact and consider functions in the Sobolev space $\mathcal{W}^{1,2}(\mathcal{X},\mu)$:

$$\mathcal{W}^{1,2}(\mathcal{X},\mu) = \left\{ f : \mathcal{X} \to \mathbb{R}, \int_{\mathcal{X}} \left\| \nabla_x f(x) \right\|^2 \mu(x)dx < \infty \right\},$$

We restrict ourselves to functions in $\mathcal{W}^{1,2}(\mathcal{X},\mu)$ vanishing at the boundary, and note this space $\mathcal{W}^{1,2}_0(\mathcal{X},\mu)$. Note that in this case:

$$\|f\|_{\mathcal{W}^{1,2}_0(\mathcal{X},\mu)} = \sqrt{\int_{\mathcal{X}} \left\| \nabla_x f(x) \right\|^2 \mu(x)dx}$$

defines a semi-norm. We can similarly define a dot product in $\mathcal{W}^{1,2}_0(\mathcal{X},\mu)$, for $f,g \in \mathcal{W}^{1,2}_0(\mathcal{X},\mu)$:

$$\langle f,g \rangle_{\mathcal{W}^{1,2}_0(\mathcal{X},\mu)} = \int_{\mathcal{X}} \langle \nabla_x f(x), \nabla_x g(x) \rangle_{\mathbb{R}^d} \mu(x)dx.$$

Hence we define the following Sobolev IPM, by restricting the critic of the mean discrepancy to the Sobolev unit ball :

$$\mathcal{S}_{\boldsymbol{\mu}}(\mathbb{P},\mathbb{Q}) = \sup_{f\in\mathcal{W}_0^{1,2}, \|f\|_{\mathcal{W}_0^{1,2}(\mathcal{X},\boldsymbol{\mu})}\leq 1} \Big\{\mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x)\Big\}. \tag{4.3}$$

When compared to the Wasserstein distance, the Sobolev IPM given in Eq. (4.3) uses a data dependent gradient constraint (depends on μ) rather than a data independent Lipchitz constraint. Let $F_{\mathbb{P}}$ and $F_{\mathbb{Q}}$ be *the cumulative distribution functions* of $\mathbb{P}$ and $\mathbb{Q}$ respectively. We have:

$$\mathbb{P}(x) = \frac{\partial^d}{\partial x_1 \dots \partial x_d} F_{\mathbb{P}}(x), \tag{4.4}$$

and we define

$$D^{-i} = \frac{\partial^{d-1}}{\partial x_1 \dots \partial x_{i-1}\partial x_{i+1}\dots\partial x_d}, \text{ for } i = 1\dots d.$$

D^{-i} computes the $(d-1)$ high-order partial derivative excluding the variable i. Our main result is presented in Theorem 15.

Theorem 15 (Sobolev IPM)**.** *Assume that $F_{\mathbb{P}}$, and $F_{\mathbb{Q}}$ and its d derivatives exist and are continuous: $F_{\mathbb{P}}$ and $F_{\mathbb{Q}} \in C^d(\mathcal{X})$. Define the differential operator D^- :*

$$D^- = (D^{-1}, \dots D^{-d}).$$

For $x = (x_1, \dots x_{i-1}, x_i, x_{i+1}, \dots x_d)$, let $x^{-i} = (x_1, \dots x_{i-1}, x_{i+1}, \dots x_d)$.

The Sobolev IPM given in Eq. (4.3) has the following equivalent forms:

1. *Sobolev IPM as comparison of high order partial derivatives of CDFs. The Sobolev IPM has the following form:*

$$\mathcal{S}_{\mu}(\mathbb{P},\mathbb{Q}) = \frac{1}{d}\sqrt{\int_{\mathcal{X}} \frac{\sum_{i=1}^d (D^{-i}F_{\mathbb{P}}(x) - D^{-i}F_{\mathbb{Q}}(x))^2}{\mu(x)} dx}.$$

2. *Sobolev IPM as comparison of weighted (coordinate-wise) conditional CDFs. The Sobolev IPM can be written in the following equivalent form:*

$$\mathcal{S}^2_{\mu}(\mathbb{P},\mathbb{Q}) = \frac{1}{d^2}\mathbb{E}_{x\sim\mu}\sum_{i=1}^d \left(\frac{\mathbb{P}_{X^{-i}}(x^{-i})F_{\mathbb{P}_{[X_i|X^{-i}=x^{-i}]}}(x_i) - \mathbb{Q}_{X^{-i}}(x^{-i})F_{\mathbb{Q}_{[X_i|X^{-i}=x^{-i}]}}(x_i)}{\mu(x)}\right)^2. \tag{4.5}$$

3. *The optimal critic f^* satisfies the following identity:*

$$\nabla_x f^*(x) = \frac{1}{d\mathcal{S}_{\mu}(\mathbb{P},\mathbb{Q})}\frac{D^-F_{\mathbb{Q}}(x) - D^-F_{\mathbb{P}}(x)}{\mu(x)}, \boldsymbol{\mu} - \textit{almost surely.} \tag{4.6}$$

We make the following remarks on Theorem 15:

1. From Theorem 15, we see that the Sobolev IPM compares d higher order partial derivatives of the cumulative distributions $F_{\mathbb{P}}$ and $F_{\mathbb{Q}}$, while Fisher IPM compares the probability density functions.

2. The dominant measure μ plays a similar role to Fisher:

$$\mathcal{S}^2_{\mu}(\mathbb{P},\mathbb{Q}) = \frac{1}{d^2}\sum_{i=1}^{d}\mathbb{E}_{x\sim\mu}\left(\frac{D^{-i}F_{\mathbb{P}}(x) - D^{-i}F_{\mathbb{Q}}(x)}{\mu(x)}\right)^2,$$

   the average distance is defined with respect to points sampled from μ.

3. **Comparison of coordinate-wise Conditional CDFs.** We note in the following $x^{-i} = (x_1, \dots x_{i-1}, x_{i+1}, \dots x_d)$. Note that we have:

$$\begin{aligned} D^{-i}F_{\mathbb{P}}(x) &= \frac{\partial^{d-1}}{\partial x_1 \dots \partial x_{i-1}\partial x_{i+1}\dots \partial x_d}\int_{-\infty}^{x_1}\cdots\int_{-\infty}^{x_d}\mathbb{P}(u_1\dots u_d)du_1\dots du_d \\ &= \int_{-\infty}^{x_i}\mathbb{P}(x_1,\dots,x_{i-1},u,x_{i+1},\dots,x_d)du \\ &= \mathbb{P}_{X^{-i}}(x_1,\dots,x_{i-1},x_{i+1},\dots x_d)\int_{-\infty}^{x_i}\mathbb{P}_{[X_i|X^{-i}=x^{-i}]}(u|x_1,\dots,x_{i-1},x_{i+1},\dots x_d)du \\ &\text{(Using Bayes rule)} \\ &= \mathbb{P}_{X^{-i}}(x^{-i})F_{\mathbb{P}_{[X_i|X^{-i}=x^{-i}]}}(x_i), \end{aligned}$$

   Note that for each i, $D^{-i}F_{\mathbb{P}}(x)$ is the cumulative distribution of the variable X_i given the other variables $X^{-i} = x^{-i}$, weighted by the density function of X^{-i} at x^{-i}. This leads us to the form given in Eq. (4.5).

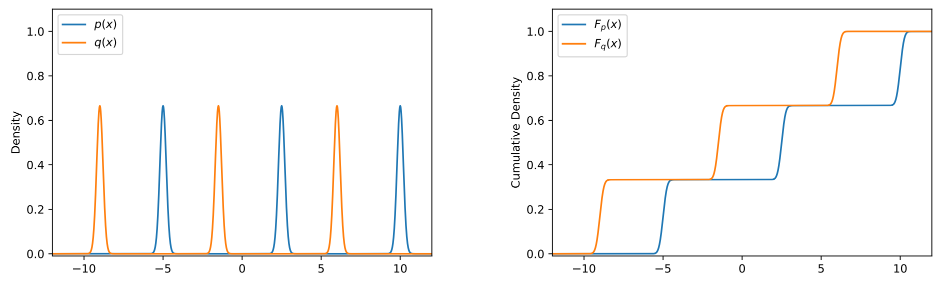
   We see that the Sobolev IPM compares for each dimension i the conditional cumulative distribution of each variable given the other variables, weighted by their density function. We refer to this as comparison of coordinate-wise CDFs on a leave one out basis. From this we see that we are comparing CDFs, which are better behaved on discrete distributions. Moreover, the conditioning built in to this metric will play a crucial role in comparing sequences as the conditioning is important in this context (See Section 4.6.1).

4.3.2 Sobolev IPM Approximation

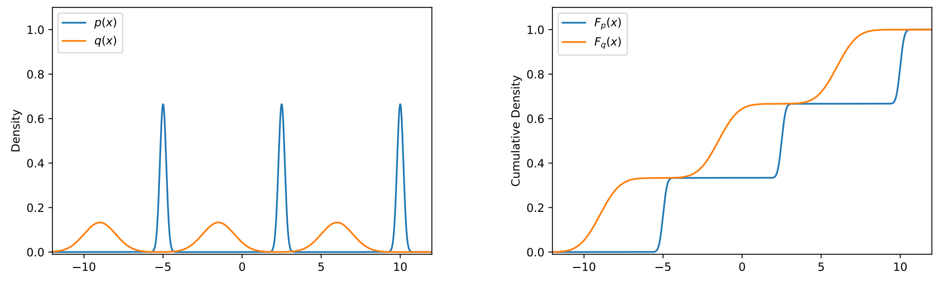
Learning in the whole Sobolev space $\mathcal{W}^{1,2}_0$ is challenging hence we need to restrict our function class to a hypothesis class $\mathcal{H}$, such as neural networks. We assume in the following that functions in $\mathcal{H}$ vanish on the boundary of $\mathcal{X}$, and restrict the optimization to the function space $\mathcal{H}$. $\mathcal{H}$ can be a Reproducing Kernel Hilbert Space as in the MMD case or parametrized by a neural network. Define the Sobolev IPM approximation in $\mathcal{H}$:

$$\mathcal{S}_{\mathcal{H},\mu}(\mathbb{P},\mathbb{Q}) = \sup_{f\in\mathcal{H},\|f\|_{\mathcal{W}^{1,2}_0}\leq 1}\left\{\mathbb{E}_{x\sim\mathbb{P}}f(x) - \mathbb{E}_{x\sim\mathbb{Q}}f(x)\right\} \tag{4.7}$$

The following Lemma shows that the Sobolev IPM approximation in $\mathcal{H}$ is proportional to Sobolev IPM. The tightness of the approximation of the Sobolev IPM is governed by the tightness of the approximation of the optimal Sobolev Critic f^* in $\mathcal{H}$. This approximation is measured in the Sobolev sense, using the Sobolev dot product.



(a) Smoothed discrete densities: PDF versus CDF of smoothed discrete densities with non overlapping supports.



(b) Smoothed Discrete and Continuous densities: PDF versus CDF of a smoothed discrete density and a continuous density with non overlapping supports.

Figure 4.2: In the GAN context for example in text generation, we have to match a (smoothed) discrete real distribution and a continuous generator. In this case, the CDF matching enabled by Sobolev IPM gives non zero discrepancy between a (smoothed) discrete and a continuous density even if the densities have disjoint supports. This ensures non vanishing gradients of the critic.

Lemma 16 (Sobolev IPM Approximation in a Hypothesis Class)**.** *Let $\mathcal{H}$ be a function space with functions vanishing at the boundary. For any $f \in \mathcal{H}$ and for f^* the optimal critic in $\mathcal{W}_0^{1,2}$, we have:*

$$\mathcal{S}_{\mathcal{H},\mu}(\mathbb{P},\mathbb{Q}) = \mathcal{S}_{\mu}(\mathbb{P},\mathbb{Q}) \sup_{f\in\mathcal{H}, \|f\|_{\mathcal{W}_0^{1,2}(\mathcal{X},\mu)}\leq 1} \int_{\mathcal{X}} \langle \nabla_x f(x), \nabla_x f^*(x)\rangle_{\mathbb{R}^d}\, \mu(x)dx.$$

Note that Lemma 16 means that the Sobolev IPM is well approximated if the space $\mathcal{H}$ has an enough representation power to express $\nabla_x f^*(x)$. This is parallel to the Fisher IPM approximation (Mroueh and Sercu, 2017) where it is shown that the Fisher IPM approximation error is proportional to the critic approximation in the Lebesgue sense. Having in mind that the gradient of the critic is the information that is passed on to the generator, we see that this convergence in the Sobolev sense to the optimal critic is an important property for GAN training.

4.4 Sobolev IPM and Other Discrepancies

In this section, we present the theoretical properties of Sobolev IPM and how it relates to distributions transport theory and other known metrics between distributions.

4.4.1 Sobolev IPM / Cramér Distance and Wasserstein-1 in One Dimension

In one dimension, Sobolev IPM is the Cramér Distance (for μ uniform on $\mathcal{X}$, we note this $\mu := 1$). While Sobolev IPM in one dimension measures the discrepancy between CDFs, the one dimensional Wasserstein-p distance measures the discrepancy between inverse CDFs:

$$\mathcal{S}^2_{\mu:=1}(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} (F_{\mathbb{P}}(x) - F_{\mathbb{Q}}(x))^2 dx \text{ versus } \mathcal{W}^p_p(\mathbb{P},\mathbb{Q}) = \int_0^1 |F^{-1}_{\mathbb{P}}(u) - F^{-1}_{\mathbb{Q}}(u)|^p du,$$

Recall also that the Fisher IPM for uniform μ is given by :

$$\mathcal{F}^2_{\mu:=1}(\mathbb{P},\mathbb{Q}) = \int_{\mathcal{X}} (\mathbb{P}(x) - \mathbb{Q}(x))^2 dx.$$

Consider for instance two point masses $\mathbb{P} = \delta_{a_1}$ and $\mathbb{Q} = \delta_{a_2}$ with $a_1, a_2 \in \mathbb{R}$. The rationale behind using Wasserstein distance for GAN training is that since it is a weak metric, for far distributions Wasserstein distance provides some signal (Arjovsky et al., 2017). In this case, it is easy to see that $\mathcal{W}^1_1(\mathbb{P},\mathbb{Q}) = \mathcal{S}^2_{\mu:=1} = |a_1 - a_2|$, while $\mathcal{F}^2_{\mu:=1}(\mathbb{P},\mathbb{Q}) = 2$. As we see from this simple example, *CDF* comparison is more suitable than PDF for comparing distributions on *discrete spaces*. See Figure 4.2, for a further discussion of this effect in the GAN context.

4.4.2 Distribution Transport Perspective on Sobolev IPM

We characterize the optimal critic of the Sobolev IPM as a solution of a non linear PDE. The solution of the variational problem of the Sobolev IPM satisfies a non linear PDE that can be derived using standard tools from calculus of variations (Alaoui et al., 2016; Ekeland and Turnbull, 1983).

Theorem 17 (PDE satisfied by the Sobolev Critic)**.** *The optimal critic of Sobolev IPM f^* satisfies the following PDE:*

$$\Delta f^*(x) + \langle \nabla_x \log \mu(x), \nabla_x f^*(x) \rangle + \frac{\mathbb{P}(x) - \mathbb{Q}(x)}{\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})\mu(x)} = 0. \tag{4.8}$$

Define the Stein Operator: $T(\mu)\vec{g}(x) = \frac{1}{2}\Big(\langle \nabla_x \log(\mu(x)), \vec{g}(x) \rangle + div(\vec{g}(x)) \Big)$. Hence we have the following Transport Equation of $\mathbb{P}$ to $\mathbb{Q}$:

$$\mathbb{Q}(x) = \mathbb{P}(x) + 2\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})\mu(x)T(\mu)\nabla_x f^*(x).$$

Recall the definition of Stein Discrepancy :

$$\mathbb{S}(\mathbb{Q},\mu) = \sup_{\vec{g}} |\mathbb{E}_{x\sim\mathbb{Q}}[T(\mu)\vec{g}(x)]|, \vec{g} : \mathcal{X} \to \mathbb{R}^d.$$

Theorem 18 (Sobolev and Stein Discrepanices)**.** *The following inequality holds true:*

$$\left|\mathbb{E}_{x\sim\mathbb{Q}}\left[\frac{\mathbb{Q}(x)-\mathbb{P}(x)}{\mu(x)}\right]\right| \leq 2 \underbrace{\mathbb{S}(\mathbb{Q},\mu)}_{\textit{Stein Good fitness of the model } \mathbb{Q} \textit{ w.r.t to } \mu} \underbrace{\mathcal{S}_{\mu}(\mathbb{P},\mathbb{Q})}_{\textit{Sobolev Distance}} \tag{4.9}$$

Consider for example $\mu=\mathbb{P}$, and sequence $\mathbb{Q}_n$. If the Sobolev distance goes $\mathcal{S}_{\mathbb{P}}(\mathbb{P},\mathbb{Q}_n)\to 0$, the ratio $r_n(x)=\frac{\mathbb{Q}_n(x)}{\mathbb{P}(x)}$ converges in expectation (w.r.t to $\mathbb{Q}$) to 1. The speed of the convergence is given by the Stein Discrepancy $\mathbb{S}(\mathbb{Q}_n,\mathbb{P})$.

4.4.3 Relation to Fokker-Planck Diffusion Equation and Particles Dynamics

Note that PDE satisifed by the Sobolev critic given in Eq. (4.8) can be equivalently written:

$$\frac{\mathbb{P}-\mathbb{Q}}{\mathcal{S}_{\mu}(\mathbb{P},\mathbb{Q})} = -\mathrm{div}(\mu(x)\nabla_x f^*(x)), \tag{4.10}$$

written in this form, we draw a connection with the Fokker-Planck Equation for the evolution of a density function q_t that is the density of particles $X_t\in\mathbb{R}^d$ evolving with a drift (a velocity field) $V(x,t):\mathcal{X}\times[0,\infty[\to\mathbb{R}^d$:

$$dX_t = V(X_t,t)dt, \text{where the density of } X_0 \text{ is given by } q_0(x)=\mathbb{Q}(x),$$

The Fokker-Planck Equation states that the evolution of the particles density q_t satisfies:

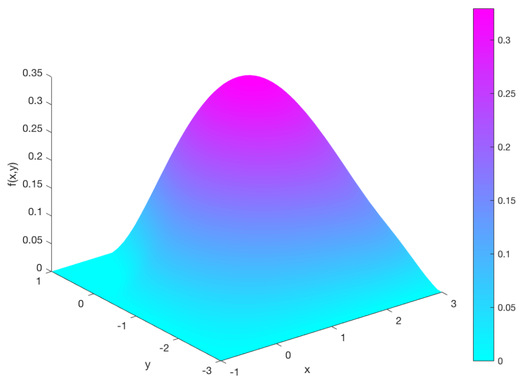
$$\frac{dq_t}{dt}(x) = -\mathrm{div}(q_t(x)V(x,t)) \tag{4.11}$$

Comparing Eq. (4.10) and Eq. (4.11), we identify then the gradient of Sobolev critic as a drift. This suggests that one can define "Sobolev descent" as the evolution of particles along the gradient flow:

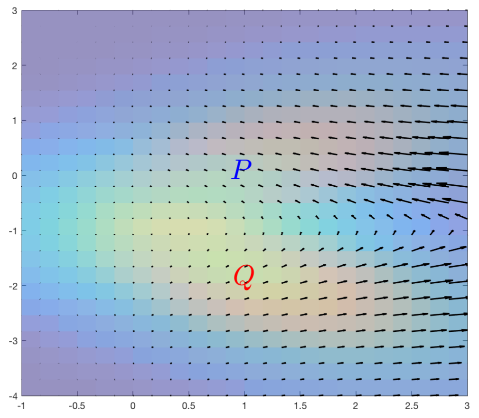
$$dX_t = \nabla_x f_t^*(X_t)dt, \text{where the density of } X_0 \text{ is given by } q_0(x)=\mathbb{Q}(x),$$

where f_t^* is the Sobolev critic between q_t and $\mathbb{P}$. One can show that the limit distribution of the particles is $\mathbb{P}$. The analysis of "Sobolev descent" and its relation to Stein Descent (Liu, 2017; Liu and Wang, 2016; Mroueh et al., 2019) is beyond the scope of this thesis. Hence we see that the gradient of the Sobolev critic defines a transportation plan to move particles whose distribution is $\mathbb{Q}$ to particles whose distribution is $\mathbb{P}$ (See Figure 4.3). This highlights the role of the gradient of the critic in the context of GAN training in term of transporting the distribution of the generator to the real distribution.

Example: Sobolev IPM between two 2D Gaussians. We consider $\mathbb{P}$ and $\mathbb{Q}$ to be two dimensional Gaussians with means μ_1 and μ_2 and covariances Σ_1 and Σ_2. Let (x,y) be the coordinates in 2D. We note $F_{\mathbb{P}}$ and $F_{\mathbb{Q}}$ the CDFs of $\mathbb{P}$ and $\mathbb{Q}$ respectively. We consider



(a) Numerical solution of the PDE satisfied by the optimal Sobolev critic.



(b) Optimal Sobolev Transport Vector Field $\nabla_x f^*(x)$ (arrows are the vector field $\nabla_x f^*(x)$ evaluated on the 2D grid. Magnitude of arrows was rescaled for visualization.)

Figure 4.3: Numerical solution of the PDE satisfied by the optimal Sobolev critic and the transportation plan induced by the gradient of Sobolev critic. The *gradient of the critic* (wrt to the input), defines on the support of $\mu = \frac{\mathbb{P}+\mathbb{Q}}{2}$ a *transportation plan for moving the distribution mass from* $\mathbb{Q}$ *to* $\mathbb{P}$, which is an example for showing our analysis of transportation plan and its relation to Fokker-Planck diffusion.

in this example $\mu = \frac{\mathbb{P}+\mathbb{Q}}{2}$. We know from Theorem 15 that the gradient of the Sobolev optimal critic is proportional to the following vector field:

$$\nabla f^*(x,y) \ \alpha \ \frac{1}{\mu(x,y)} \begin{bmatrix} \frac{\partial}{\partial y}(F_{\mathbb{Q}}(x,y) - F_{\mathbb{P}}(x,y)) \\ \frac{\partial}{\partial x}(F_{\mathbb{Q}}(x,y) - F_{\mathbb{P}}(x,y)) \end{bmatrix} \tag{4.12}$$

In Figure 4.3 we consider $\mu_1 = [1,0], \Sigma_1 = \begin{bmatrix} 1.9 & 0.8 \\ 0.8 & 1.3 \end{bmatrix} \mu_2 = [1,-2], \Sigma_2 = \begin{bmatrix} 1.9 & -0.8 \\ -0.8 & 1.3 \end{bmatrix}$. In Figure 4.3a we plot the numerical solution of the PDE satisfied by the optimal Sobolev critic given in Eq. (4.10), using MATLAB solver for elliptic PDEs (more accurately we solve $-div(\mu(x)\nabla_x f(x)) = \mathbb{P}(x) - \mathbb{Q}(x)$, hence we obtain the solution of Eq. (4.10) up to a normalization constant ($\frac{1}{\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})}$)). We numerically solve the PDE on a rectangle with zero boundary conditions. We see that the optimal Sobolev critic separates the two distributions well. In Figure 4.3b we then numerically compute the gradient of the optimal Sobolev critic on a 2D grid as given in Eq. (4.12) (using numerical evaluation of the CDF and finite difference for the evaluation of the partial derivatives). We plot in Figure 4.3b the density functions of $\mathbb{P}$ and $\mathbb{Q}$ as well as the vector field of the gradient of the optimal Sobolev critic. As discussed above, we see that the gradient of the critic (wrt to the input), defines on the support of $\mu = \frac{\mathbb{P}+\mathbb{Q}}{2}$ a transportation plan for moving the distribution mass from $\mathbb{Q}$ to $\mathbb{P}$.

4.4.4 Comparison with Other Discrepancies

In Table 4.1, we give a comparison of different discrepancies Δ and function spaces $\mathcal{F}$ used in the literature for IGM training together with our proposed Sobolev IPM. We see from Table 4.1 that Sobolev IPM, compared to Wasserstein Distance, imposes a tractable smoothness constraint on the critic on points sampled from a distribution μ, rather then imposing a Lipschitz constraint on all points in the space $\mathcal{X}$. We also see that Sobolev IPM is the natural generalization of the Cramér Von-Mises Distance from one dimension to high dimensions as we discussed above. We note that the Energy Distance, a form of Maximum Mean Discrepancy for a special kernel, was used in Bellemare et al. (2017) as a generalization of the Cramér distance in GAN training but still needed a gradient penalty in its algorithmic counterpart leading to a mis-specified distance between distributions. We note that the Stein metric (Liu, 2017; Liu et al., 2016) uses the score function to match distributions and we have shown that how Sobolev IPM relates to the Stein discrepancy. Finally, Fisher IPM compares joint PDF of the distributions, Sobolev IPM compares weighted (coordinate-wise) conditional CDFs.

4.5 Sobolev GAN: IGM with Sobolev IPM

Now we turn to the problem of learning IGMs with Sobolev IPM. Given the "real distribution" $\mathbb{P}_{\mathcal{X}} \in \mathcal{P}(\mathcal{X})$, our goal is to learn a generator $g_\theta : \mathcal{Z} \subset \mathbb{R}^{n_z} \to \mathcal{X}$, such that for $z \sim p_z$, the distribution of $g_\theta(z)$ is close to the real data distribution $\mathbb{P}_{\mathcal{X}}$, where p_z is a fixed distribution on $\mathcal{Z}$ (for instance $z \sim \mathcal{N}(0, I_{n_z})$). We note $\mathbb{Q}_\theta$ for the "fake distribution" of $g_\theta(z), z \sim p_z$. Consider $\{x_i, i = 1 \dots N\} \sim \mathbb{P}_{\mathcal{X}}$, $\{z_i, i = 1 \dots N\} \sim \mathcal{N}(0, I_{n_z})$, and $\{\widetilde{x}_i, i = 1 \dots N\} \sim \mu$. We consider these choices for μ:

1. $\mu = \frac{\mathbb{P}_{\mathcal{X}} + \mathbb{Q}_\theta}{2}$ i.e $\widetilde{x} \sim \mathbb{P}_{\mathcal{X}}$ or $\widetilde{x} = g_\theta(z), z \sim \mathbb{P}_{\mathcal{Z}}$ with equal probability $\frac{1}{2}$.
2. μ_{GP} is the implicit distribution defined by the interpolation lines between $\mathbb{P}_{\mathcal{X}}$ and $\mathbb{Q}_\theta$ as in Gulrajani et al. (2017) i.e : $\widetilde{x} = ux + (1-u)y, x \sim \mathbb{P}_{\mathcal{X}}, y = g_\theta(z), z \sim p_z$ and $u \sim \text{Unif}[0, 1]$.

Sobolev GAN can be written as follows:

$$\min_{\theta} \sup_{\phi, \frac{1}{N}\sum_{i=1}^N \left\|\nabla_x f_\phi(\widetilde{x}_i)\right\|^2 = 1} \widehat{\mathcal{E}}(f_\phi, g_\theta) = \frac{1}{N}\sum_{i=1}^N f_\phi(x_i) - \frac{1}{N}\sum_{i=1}^N f_\phi(g_\theta(z_i))$$

For any choice of the parametric function class $\mathcal{H}_\phi$, note the constraint by $\widehat{\Omega}_S(f_\phi, g_\theta) = \frac{1}{N}\sum_{i=1}^N \|\nabla_x f_\phi(\widetilde{x}_i)\|^2$. For example if $\mu = \frac{\mathbb{P}_{\mathcal{X}} + \mathbb{Q}_\theta}{2}$, $\widehat{\Omega}_S(f_\phi, g_\theta) = \frac{1}{2N}\sum_{i=1}^N \|\nabla_x f_\phi(x_i)\|^2 + \frac{1}{2N}\sum_{i=1}^N \|\nabla_x f_\phi(g_\theta(z_i))\|^2$. Note that, since the optimal theoretical critic is achieved on the sphere, we impose a sphere constraint rather than a ball constraint. Similar to Mroueh and Sercu (2017) we define the Augmented Lagrangian corresponding to Sobolev GAN objective and constraint

$$\mathcal{L}_S(p, \theta, \lambda) = \widehat{\mathcal{E}}(f_\phi, g_\theta) + \lambda(1 - \widehat{\Omega}_S(f_\phi, g_\theta)) - \frac{\rho}{2}(\widehat{\Omega}_S(f_\phi, g_\theta) - 1)^2 \tag{4.13}$$

where λ is the Lagrange multiplier and $\rho > 0$ is the quadratic penalty weight. We alternate between optimizing the critic and the generator. We impose the constraint when training

	$\boldsymbol{\Delta(f;\mathbb{P},\mathbb{Q})}$	$\boldsymbol{\mathcal{F}}$ Function class	$\boldsymbol{d_{\mathcal{F}}(\mathbb{P},\mathbb{Q})}$ Closed Form
f-Divergence (Goodfellow et al., 2014) (Nowozin et al., 2016)	$\mathbb{E}_{x\sim\mathbb{P}}f(x)-\mathbb{E}_{x\sim\mathbb{Q}}\varphi^*(f(x))$ φ^* Fenchel Conjugate	$\left\{f:\mathcal{X}\to\mathbb{R}, f\in\text{dom}_{\varphi^*}\right\}$	$\mathbb{E}_{x\sim\mathbb{Q}}\left[\varphi(\frac{\mathbb{P}(x)}{\mathbb{Q}(x)})\right]$
Wasserstein -1 (Arjovsky et al., 2017) (Gulrajani et al., 2017)	$\mathbb{E}_{x\sim\mathbb{P}}f(x)-\mathbb{E}_{x\sim\mathbb{Q}}f(x)$	$\left\{f:\mathcal{X}\to\mathbb{R}, \left\Vert f\right\Vert_{\text{lip}}\leq 1\right\}$	$\inf_{\pi\in\Pi(\mathbb{P},\mathbb{Q})}\int_{\mathcal{X}}\left\Vert x-y\right\Vert_1 d\pi(x,y)$ Sinkhorn Divergence (Genevay et al., 2018)
MMD (Li et al., 2017) (Li et al., 2015b) (Dziugaite et al., 2015)	$\mathbb{E}_{x\sim\mathbb{P}}f(x)-\mathbb{E}_{x\sim\mathbb{Q}}f(x)$	$\left\{f:\mathcal{X}\to\mathbb{R}, \left\Vert f\right\Vert_{\mathcal{H}_k}\leq 1\right\}$	$\left\Vert\mathbb{E}_{x\sim\mathbb{P}}k_x-\mathbb{E}_{x\sim\mathbb{Q}}k_x\right\Vert_{\mathcal{H}_k}$
Stein Discrepancy (Wang and Liu, 2016)	$\mathbb{E}_{x\sim\mathbb{Q}}\left[T(\mathbb{P})f(x)\right]$ $T(\mathbb{P})=(\nabla_x\log(\mathbb{P}(x))^\top+\nabla_x.$	$\Big\{f:\mathcal{X}\to\mathbb{R}^d$ f smooth with zero boundary condition $\Big\}$	NA in general has a closed form in RKHS
Cramér for $\boldsymbol{d=1}$ (Bellemare et al., 2017)	$\mathbb{E}_{x\sim\mathbb{P}}f(x)-\mathbb{E}_{x\sim\mathbb{Q}}f(x)$	$\Big\{f:\mathcal{X}\to\mathbb{R}, \mathbb{E}_{x\sim\mathbb{P}}(\frac{df(x)}{dx})^2\leq 1,$ f smooth with zero boundary condition $\Big\}$	$\sqrt{\mathbb{E}_{x\sim\mathbb{P}}\left(\frac{F_{\mathbb{P}}(x)-F_{\mathbb{Q}}(x)}{\mathbb{P}(x)}\right)^2}$ $x\in\mathbb{R}$
μ-Fisher IPM (Mroueh and Sercu, 2017)	$\mathbb{E}_{x\sim\mathbb{P}}f(x)-\mathbb{E}_{x\sim\mathbb{Q}}f(x)$	$\Big\{f:\mathcal{X}\to\mathbb{R}, f\in\mathcal{L}_2(\mathcal{X},\mu),$ $\mathbb{E}_{x\sim\mu}f^2(x)\leq 1\Big\}$	$\sqrt{\mathbb{E}_{x\sim\mu}\left(\frac{\mathbb{P}(x)-\mathbb{Q}(x)}{\mu(x)}\right)^2}$
μ-Sobolev IPM (This work)	$\mathbb{E}_{x\sim\mathbb{P}}f(x)-\mathbb{E}_{x\sim\mathbb{Q}}f(x)$	$\Big\{f:\mathcal{X}\to\mathbb{R}, f\in\mathcal{W}_0^{1,2}(\mathcal{X},\mu),$ $\mathbb{E}_{x\sim\mu}\left\Vert\nabla_x f(x)\right\Vert^2\leq 1,$ with zero boundary condition $\Big\}$	$\frac{1}{d}\sqrt{\mathbb{E}_{x\sim\mu}\sum_{i=1}^d\left(\frac{\phi_i(\mathbb{P})-\phi_i(\mathbb{Q})}{\mu(x)}\right)^2}$ where $\phi_i(\mathbb{P})=$ $\mathbb{P}_{X^{-i}}(x^{-i})F_{\mathbb{P}_{[X_i\vert X^{-i}=x^{-i}]}}(x_i)$ $x^{-i}=(x_1,\dots x_{i-1},x_{i+1},\dots x_d)$

Table 4.1: Comparison of different metrics between distributions used for GAN training. References are for papers using those metrics for GAN training.

the critic only. Given θ, we solve $\max_\phi\min_\lambda\mathcal{L}_S(\phi,\theta,\lambda)$, for training the critic. Then given the critic parameters ϕ we optimize the generator weights θ to minimize the objective $\min_\theta\widehat{\mathcal{E}}(f_\phi,g_\theta)$. See Algorithm 2.

Remark 19. *Note that in Algorithm 2, we obtain a biased estimate since we are using same samples for the cost function and the constraint, but the incurred bias can be shown to be small and vanishing as the number of samples increases as shown and justified in Shivaswamy and Jebara (2010).*

Algorithm 2 Sobolev GAN

```
Input: $\rho$ penalty weight, $\eta$ Learning rate, $n_c$ number of iterations for training the critic,
N batch size
Initialize $p, \theta, \lambda = 0$
repeat
    for $j = 1$ to $n_c$ do
        Sample a minibatch $x_i, i = 1 \dots N, x_i \sim \mathbb{P}_{\mathcal{X}}$
        Sample a minibatch $z_i, i = 1 \dots N, z_i \sim \mathbb{P}_{\mathcal{Z}}$
        $(g_\phi, g_\lambda) \leftarrow (\nabla_\phi \mathcal{L}_S, \nabla_\lambda \mathcal{L}_S)(p, \theta, \lambda)$
        $p \leftarrow \phi + \eta$ ADAM $(p, g_\phi)$

        $\lambda \leftarrow \lambda - \rho g_\lambda$                    ▷ SGD rule on $\lambda$ with learning rate $\rho$
    end for
    Sample $z_i, i = 1 \dots N, z_i \sim \mathbb{P}_{\mathcal{Z}}$
    $d_\theta \leftarrow \nabla_\theta \widehat{\mathcal{E}}(f_\phi, g_\theta) = -\nabla_\theta \frac{1}{N} \sum_{i=1}^N f_\phi(g_\theta(z_i))$
    $\theta \leftarrow \theta - \eta$ ADAM $(\theta, d_\theta)$
until $\theta$ converges
```

Relation to WGAN-GP. WGAN-GP can be written as follows:

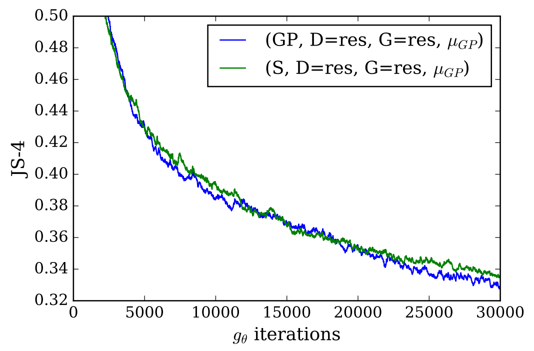
$$\min_\theta \sup_{\phi, \left\|\nabla_x f_\phi(\widetilde{x}_i)\right\|=1, \widetilde{x}_i \sim \mu_{GP}} \widehat{\mathcal{E}}(f_\phi, g_\theta) = \frac{1}{N}\sum_{i=1}^N f_\phi(x_i) - \frac{1}{N}\sum_{i=1}^N f_\phi(g_\theta(z_i))$$

The main difference between WGAN-GP and our setting, is that WGAN-GP enforces *pointwise constraints* on points drawn from $\mu = \mu_{GP}$ via a point-wise quadratic penalty $(\widehat{\mathcal{E}}(f_\phi, g_\theta) - \lambda \sum_{i=1}^N (1 - \|\nabla_x f(\widetilde{x}_i)\|)^2)$ while we enforce that constraint on average as a Sobolev norm, allowing us the coordinate weighted conditional CDF interpretation of the IPM.

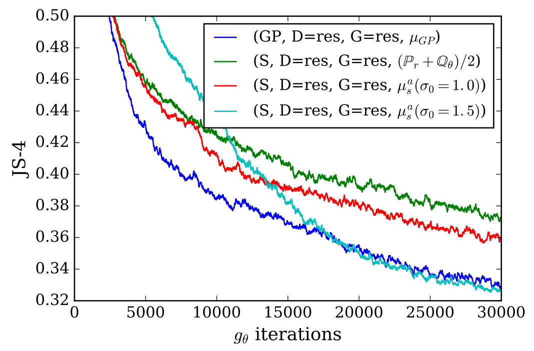
4.6 Experiments

Sobolev IPM has two important properties; The first stems from the *conditioning* built in to the metric through the weighted conditional CDF interpretation. The second stems from the *diffusion* properties that the critic of Sobolev IPM satisfies that has theoretical and practical ties to the Laplacian regularizer and diffusion on manifolds used in semi-supervised learning (Belkin et al., 2006).

In this section, we exploit those two important properties in two applications of Sobolev GAN: text generation and semi-supervised learning. First in *text generation*, which can be seen as a discrete sequence generation, Sobolev GAN (and WGAN-GP) enable training GANs without need to do explicit brute-force conditioning. We attribute this to the built-in conditioning in Sobolev IPM (for the sequence aspect) and to the CDF matching (for the discrete aspect). Secondly using GANs in semi-supervised learning is a promising avenue for learning using unlabeled data. We show that a variant of Sobolev GAN can achieve



(a) Comparing Sobolev with μ_{GP} and WGAN-GP. The JS-4 are 0.3363 and 0.3302 respectively.



(b) Result of Sobolev GAN for various dominating measure μ, for resnets as architectures of the critic and the generator.

Figure 4.4: Result of Sobolev GAN for various dominating measure μ, for resnets as architectures of the critic and the generator.

strong SSL results on the CIFAR-10 dataset, without the need of any form of activation normalization in the networks or any extra ad hoc tricks.

4.6.1 Text Generation with Sobolev GAN

In this Section, we present an empirical study of Sobolev GAN in character level text generation. Our empirical study on end to end training of character-level GAN for text generation is articulated on four dimensions **(loss, critic, generator, $\boldsymbol{\mu}$)**. (1) the loss used (**GP**: WGAN-GP (Gulrajani et al., 2017), **S**: Sobolev or **F**: Fisher) (2) the architecture of the critic (Resnets or RNN) (3) the architecture of the generator (Resnets or RNN or RNN with curriculum learning) (4) the sampling distribution $\boldsymbol{\mu}$ in the constraint.

Text Generation Experiments. We train a character-level GAN on Google Billion Word dataset and follow the same experimental setup used in Gulrajani et al. (2017). The generated sequence length is 32 and the evaluation is based on Jensen-Shannon divergence on empirical 4-gram probabilities (JS-4) of validation data and generated data. JS-4 may not be an ideal evaluation criteria, but it is a reasonable metric for current character-level GAN results, which is still far from generating meaningful sentences.

Annealed Smoothing of discrete $\mathbb{P}_{\mathcal{X}}$ in the constraint $\boldsymbol{\mu}$. Since the generator distribution will always be defined on a continuous space, we can replace the discrete "real" distribution $\mathbb{P}_{\mathcal{X}}$ with a smoothed version (Gaussian kernel smoothing) $\mathbb{P}_{\mathcal{X}} \star \mathcal{N}(0, \sigma^2 I_d)$. This corresponds to doing the following sampling for $\mathbb{P}_{\mathcal{X}} : x + \xi, x \sim \mathbb{P}_{\mathcal{X}}$, and $\xi \sim \mathcal{N}(0, \sigma^2 I_d)$. Note that we only inject noise to the "real" distribution with the goal of smoothing the support of the discrete distribution, as opposed to instance noise on both "real" and "fake" to stabilize the training, as introduced in Arjovsky and Bottou (2017); Kaae Sønderby et al. (2017). As it is common in optimization by continuation (Mobahi and III, 2015), we also anneal the noise level σ as the training progresses on a linear schedule.

Sobolev GAN versus WGAN-GP with Resnets. In this setting, we compare

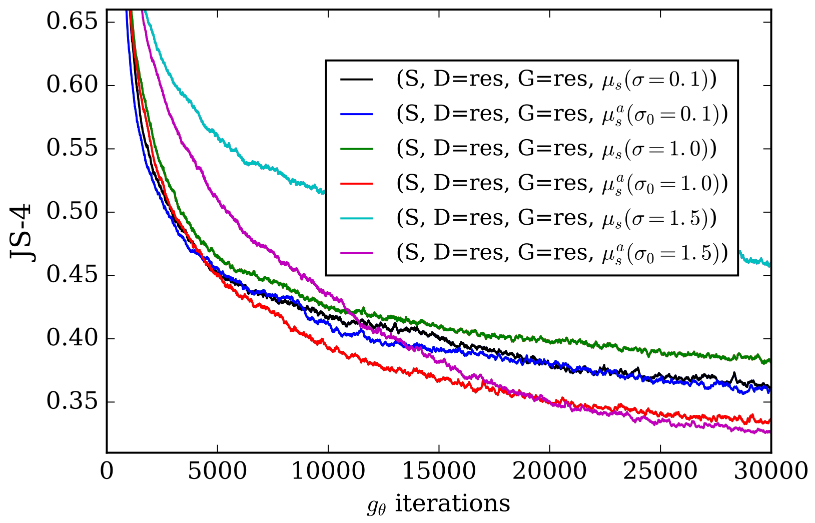


Figure 4.5: Comparison of annealed versus non annealed smoothing of $\mathbb{P}_{\mathcal{X}}$ in Sobolev GAN. We see that annealed smoothing outperforms the non annealed smoothing experiments.

(WGAN-GP,G=Resnet,D=Resnet,$\mu = \mu_{GP}$) to (Sobolev,G=Resnet,D=Resnet,μ) where μ is one of: (1) μ_{GP}, (2) the noise smoothed $\mu_s(\sigma) = \frac{\mathbb{P}_{\mathcal{X}} \star \mathcal{N}(0,\sigma^2 I_d) + \mathbb{Q}_\theta}{2}$ or (3) noise smoothed with annealing $\mu_s^a(\sigma_0)$ with σ_0 the initial noise level. We use the same architectures of Resnet with 1D convolution for the critic and the generator as in Gulrajani et al. (2017). In order to implement the noise smoothing we transform the data into one-hot vectors. Each one hot vector x is transformed to a probability vector p with 0.9 replacing the one and $0.1/(dict_{size} - 1)$ replacing the zero. We then sample ϵ from a Gaussian distribution $\mathcal{N}(0, \sigma^2)$, and use softmax to normalize $\log p + \epsilon$. We use Algorithm 2 for Sobolev GAN and fix the learning rate to 10^{-4} and ρ to 10^{-5}. The noise level σ was annealed following a linear schedule starting from an initial noise level σ_0 (at iteration i, $\sigma_i = \sigma_0(1 - \frac{i}{Maxiter})$, Maxiter=30K). Results are given in Figure 4.4a for the JS-4 evaluation of both WGAN-GP and Sobolev GAN for $\mu = \mu_{GP}$. In Figure 4.4b we show the JS-4 evaluation of Sobolev GAN with the annealed noise smoothing $\mu_s^a(\sigma_0)$, for various values of the initial noise level σ_0. We see that the training succeeds in both cases. Sobolev GAN achieves slightly better results than WGAN-GP for the annealing that starts with high noise level $\sigma_0 = 1.5$. We note that without smoothing and annealing i.e using $\mu = \frac{\mathbb{P}_{\mathcal{X}} + \mathbb{Q}_\theta}{2}$, Sobolev GAN is behind. Annealed smoothing of $\mathbb{P}_{\mathcal{X}}$, helps the training as the real distribution is slowly going from a continuous distribution to a discrete distribution. Figure 4.5 gives a comparison between annealed and non annealed smoothing.

Fisher GAN Curriculum Conditioning versus Sobolev GAN: Explicit versus Implicit conditioning. We analyze how Fisher GAN behaves under different architectures of generators and critics. We first fix the generator to be ResNet. We study 3 different architectures of critics: ResNet, GRU (we follow the experimental setup from Press et al. (2017)), and hybrid ResNet+GRU (Reed et al., 2016). We notice that RNN is unstable, we need to clip the gradient values of critics in $[-0.5, 0.5]$, and the gradient of the

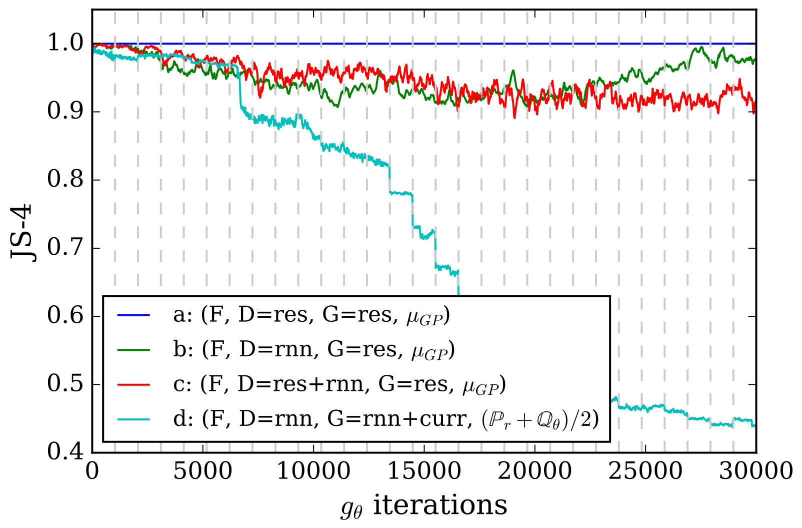
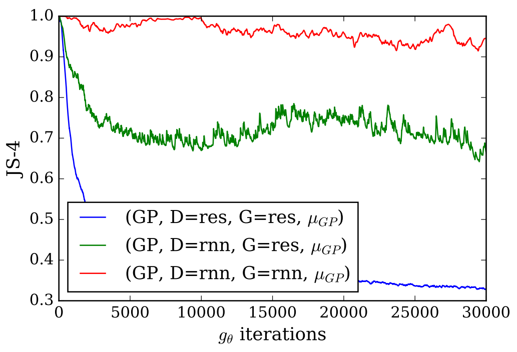


Figure 4.6: Fisher GAN with different architectures for critics: (a-c) We see that for $\mu = \mu_{GP}$ and $G =$ Resnet for various critic architectures, Fisher GAN fails at the task of text generation. We notice small improvements for RNN critics (b-c) due to the conditioning and factoring of the distribution. (d) Fisher GAN with recurrent generator and critic, trained on a curriculum conditioning for increasing lengths ℓ, increments indicated by gridlines. In this curriculum conditioning setup, with recurrent critics and generators, the training of Fisher GAN succeeds and reaches similar levels of Sobolev GAN (and WGAN-GP). It is important to note that by doing this *explicit curriculum conditioning* for Fisher GAN, we highlight the *implicit conditioning* induced by Sobolev GAN, via the gradient regularizer.

Lagrange multiplier λ_F to $[-10^4, 10^4]$. We fix $\rho_F = 10^{-7}$ and we use $\mu = \mu_{GP}$. We search the value for the learning rate in $[10^{-5}, 10^{-4}]$. We see that for $\mu = \mu_{GP}$ and $G =$ Resnet for various critic architectures, Fisher GAN fails at the task of text generation (Figure 4.6 a-c). Nevertheless, when using RNN critics (Fig 4.6 b, c) a marginal improvement happens over the fully collapsed state when using a resnet critic (Fig 4.6 a). We hypothesize that RNN critics enable some conditioning and factoring of the distribution, which is lacking in Fisher IPM.

Finally Figure 4.6 (d) shows the result of training with recurrent generator and critic. We follow Press et al. (2017) in terms of GRU architecture, but differ by using Fisher GAN rather than WGAN-GP. We use $\mu = \frac{\mathbb{P}_\mathcal{X} + \mathbb{Q}_\theta}{2}$ i.e. without annealed noise smoothing. We train (F, D=RNN,G=RNN,$\frac{\mathbb{P}_\mathcal{X} + \mathbb{Q}_\theta}{2}$) using curriculum conditioning of the generator for all lengths ℓ as done in Press et al. (2017): the generator is conditioned on $32 - \ell$ characters and predicts the ℓ remaining characters. We increment $\ell = 1$ to 32 on a regular schedule (every 15k updates). JS-4 is only computed when $\ell > 4$. We see in Figure 4.6 that under curriculum conditioning with recurrent critics and generators, the training of Fisher GAN succeeds and reaches similar levels of Sobolev GAN (and WGAN-GP). Note that the need of this *explicit brute force conditioning* for Fisher GAN, highlights the *implicit*



(a) WGAN-GP

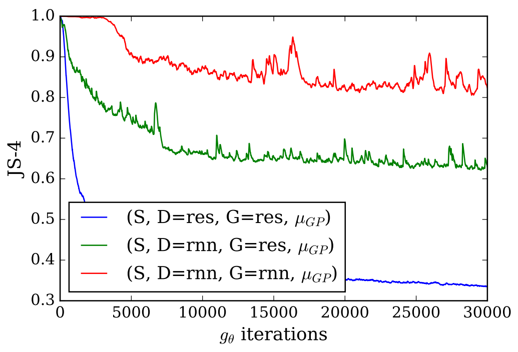


Figure 4.7: Result of WGAN-GP and Sobolev with RNNs.

conditioning induced by Sobolev GAN via the gradient regularizer, without the need for curriculum conditioning.

Sobolev GAN versus WGAN-GP with RNN. We fix the generator architecture to Resnets. The experiments of using RNN (GRU) as the critic architecture for WGAN-GP and Sobolev is shown in Figure 4.7 where we used $\mu = \mu_{GP}$ for both cases. We only apply gradient clipping to stabilize the performance without other tricks. We can observe that using RNN degrades the performance. We think that this is due to an optimization issue and a difficulty in training RNN under the GAN objective without any pre-training or conditioning. Finally, we show generated examples in Figure 4.8.

```
That time out very came of their
But it Gaylen was strosd of the
The case had caurgr thing it las
Gropate evong hould exficioul pa
The See qust , so make starter S
Cauntsrs of oprnnd accused there
Compara Tizo is thene ano hastin
With Earaie Ïpptaring very woutd
When livht think not Braoph SPec
The phan teiled " Policy , tor e
Coydey GN11 ) -s pail is uniled
That 's conpect d larce antin-iu
But it 's familions a, IHican er
Nit was bad a year hitoloy hodat
And prenches gless fram Avers aa
If 's might , comp-rime at overg
Jeads years lead of gonguied to
Asong he into get his ressson '
Nou projecti y bated with te de
CradfsCuel sad out the Gutoor .`
```

(WGAN-GP,D=res,G=res,μ_{GP})

```
The Loraia arnup to Nou ands in
Nany tecalliexpeace in that veel
" It not has allown ourn Ehough
This bastly , suphoriation almo
" The pasts of a nummers said Nh
A loved the Cam feal switht with
Apenole 's no. on walling any Cc
Furaymand chainting suppinally s
Larts Ginis , R-Ra tarkedment st
It what wowed night a chiols as
Overy shy really -- " Cyphil mad
She wore will be also a marged w
But Rere tained the sian hoy at
Hends to Won) ).--u2- 2y this he
Felecter indoadoy is ne rtlayne
Pet isser juastivus also first i
But you had not of hiscered tnd
Thoir Taray taked an intervatter
She vagger conmisurespis herkied
His juestor foy not ar oreeoon t
Buile president up thepunsit an
In ealled osficers in a rould a
The mome of tot of not shodld ye
It 'snsacopprctionialso muss y ,
```

(Sobolev GAN,D=res,G=res,μ_{GP})

```
In reperted a "aMametan 's Gegtn
Sime Vmone onerge recighed a an
Rechardty ) " Gush 's womes it l
Catious paice of an sepurying or
The vews dolerated badds to appe
Orgarda of to the cheek-ng nees
You all tnl torgave takely his e
" Ancrops than , Mumine of the s
" The Bontement is shouts will b
One if the rops of the Cutlent h
While paliless streaghal dustist
Everyial with a Ecenbers are car
It has an atton<ent ligges of ha
Hçwnton , ¡- one in aroed that I
Anmithingly country cestents toa
The odtitians exptolises , began
The may last stoct was anso stad
But the antinf moted chapimabie
The saysuthat yearthand on the d
Even the gime was shopld on pist
```

```
If you ad someone bidding at a t
ITas t ian at train , who is a m
" Be pls sahs this car and , you
 I can reminere several wok sine
" I tihne the animal soun like a
No , I hsn hen am afra i tak th
Wel , maybe the a lost good tal
Binan and I han an met is to boo
Since tins the the and shime bro
I tihne thn aimar thin you wasn
```

(Sobolev GAN,D=res,G=res,$\mu_s^a(\sigma_0 = 1.5)$) (Fisher GAN,D=GRU,G=GRU+curr,$\frac{\mathbb{P}_r+\mathbb{Q}_\theta}{2}$)

Figure 4.8: Text samples from various GANs considered in this paper.

4.6.2 Semi-Supervised Learning with Sobolev GAN

A proper and promising framework for evaluating GANs consists in using it as a regularizer in the semi-supervised learning setting (Dumoulin et al., 2017; Kumar et al., 2017; Salimans et al., 2016). As mentioned before, the Sobolev norm as a regularizer for the Sobolev IPM draws connections with the Laplacian regularization in manifold learning (Belkin et al., 2006). In the Laplacian framework of semi-supervised learning, the classifier satisfies a smoothness constraint imposed by controlling its Sobolev norm: $\int_{\mathcal{X}} \left\|\nabla_x f(x)\right\|^2 \mu^2(x)dx$ (Alaoui et al., 2016). In this Section, we present a variant of Sobolev GAN that achieves competitive performance in semi-supervised learning on the CIFAR-10 dataset (Krizhevsky and Hinton, 2009) without using any internal activation normalization in the critic, such as batch normalization (BN) (Ioffe and Szegedy, 2015), layer normalization (LN) (Ba et al., 2016), or weight normalization (Salimans and Kingma, 2016).

In this setting, a convolutional neural network $\Phi_\omega : \mathcal{X} \to \mathbb{R}^m$ is shared between the cross entropy (CE) training of a K-class classifier ($S \in \mathbb{R}^{K\times m}$) and the critic of GAN (See Figure 4.9). We have the following training equations for the (critic + classifer) and the generator:

$$\text{Critic + Classifier:} \quad \max_{S,\Phi_\omega,f} \mathcal{L}_D = \mathcal{L}^{\text{GAN}}_{\text{alm}}(f, g_\theta) - \lambda_{CE} \sum_{(x,y)\in\text{lab}} CE(p(y|x), y) \tag{4.14}$$

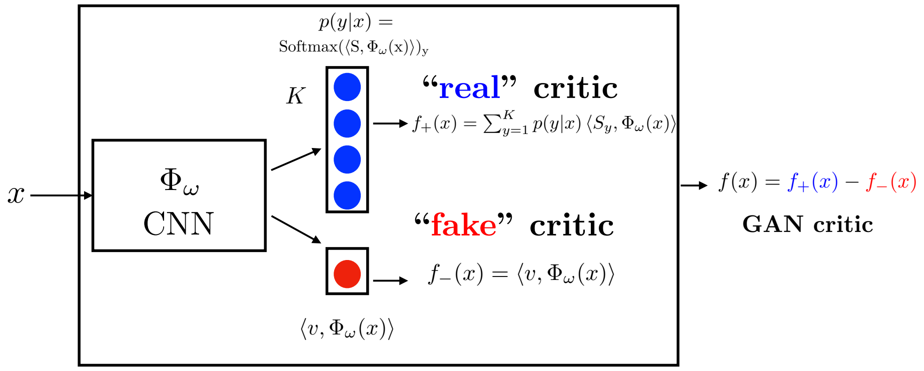
$$\text{Generator:} \ \max_{\theta} \mathcal{L}_G = \widehat{\mathcal{E}}(f, g_\theta) \tag{4.15}$$

where the main IPM objective with N samples: $\widehat{\mathcal{E}}(f, g_\theta) = \frac{1}{N}\left(\sum_{x\in\text{unl}} f(x) - \sum_{z\sim p_z} f(g_\theta(z))\right)$.

Following Mroueh and Sercu (2017), we use the following "$K+1$ parametrization" for the critic (See Figure 4.9) :

$$f(x) = \underbrace{\sum_{y=1}^{K} p(y|x)\left\langle S_y, \Phi_\omega(x)\right\rangle}_{f_+\text{: "real" critic}} - \underbrace{\left\langle v, \Phi_\omega(x)\right\rangle}_{f_-\text{:"fake" critic}}$$

Note that $p(y|x) = \mathrm{Softmax}(\langle \mathrm{S}, \Phi_\omega(\mathrm{x})\rangle)_\mathrm{y}$ appears both in the critic formulation and in the Cross-Entropy term in Eq. (4.14). Intuitively this critic uses the K class directions of the classifier S_y to define the "real" direction, which competes with another K+1$^{\text{th}}$ direction v that indicates fake samples. This parametrization adapts the idea of Salimans et al. (2016), which was formulated specifically for the classic KL / JSD based GANs, to IPM-based GANs. We saw consistently better results with the $K+1$ formulation over the regular formulation where the classification layer S doesn't interact with the critic direction v. We also note that when applying a gradient penalty based constraint (either WGAN-GP or Sobolev) on the full critic $f = f_+ - f_-$, it is impossible for the network to fit even the small labeled training set (underfitting), causing bad SSL performance. This leads us to the formulation below, where we apply the Sobolev constraint only on f_-. Throughout this Section we fix $\mu = \frac{\mathbb{P}_{\mathcal{X}}+\mathbb{Q}_\theta}{2}$.



$p(y|x) =$
$\mathrm{Softmax}(\langle S, \Phi_\omega(x)\rangle)_y$
K
"real" critic
$f_+(x) = \sum_{y=1}^{K} p(y|x)\,\langle S_y, \Phi_\omega(x)\rangle$
x
Φ_ω
CNN
$f(x) = f_+(x) - f_-(x)$
GAN critic
"fake" critic
$f_-(x) = \langle v, \Phi_\omega(x)\rangle$
$\langle v, \Phi_\omega(x)\rangle$

Figure 4.9: "K+1" parametrization of the critic for semi-supervised learning.

We propose the following two schemes for constraining the K+1 critic $f(x) = f_+(x) - f_-(x)$:

1) **Fisher constraint on the critic**: We restrict the critic to the following set:

$$f \in \left\{ f = f_+ - f_-, \quad \widehat{\Omega}_F(f, g_\theta) = \frac{1}{2N}\left(\sum_{x\in\text{unl}} f^2(x) + \sum_{z\sim p_z} f^2(g_\theta(z))\right) = 1 \right\}.$$

This constraint translates to the following ALM objective in Eq. (4.14):

$$\mathcal{L}_{\text{alm}}^{\text{GAN}}(f, g_\theta) = \widehat{\mathcal{E}}(f, g_\theta) + \lambda_F(1 - \widehat{\Omega}_F(f, g_\theta)) - \frac{\rho_F}{2}(\widehat{\Omega}_F(f, g_\theta) - 1)^2,$$

where the Fisher constraint ensures the stability of the training through an implicit whitened mean matching (Mroueh and Sercu, 2017).

2) **Fisher+Sobolev constraint:** We impose 2 constraints on the critic: Fisher on f & Sobolev on f_-

$$f \in \left\{ f = f_+ - f_-, \quad \widehat{\Omega}_F(f, g_\theta) = 1 \text{ and } \widehat{\Omega}_S(f_-, g_\theta) = 1 \right\},$$

where $\widehat{\Omega}_S(f_-, g_\theta) = \frac{1}{2N}\left(\sum_{x\in\text{unl}} \|\nabla_x f_-(x)\|^2 + \sum_{z\sim p_z} \|\nabla_x f_-(g_\theta(z))\|^2\right)$.

This constraint translates to the following ALM in Eq. (4.14):

$$\begin{aligned}\mathcal{L}_{\text{alm}}^{\text{GAN}}(f, g_\theta) = \widehat{\mathcal{E}}(f, g_\theta) &+ \lambda_F(1 - \widehat{\Omega}_F(f, g_\theta)) + \lambda_S(1 - \widehat{\Omega}_S(f_-, g_\theta)) \\ &- \frac{\rho_F}{2}(\widehat{\Omega}_F(f, g_\theta) - 1)^2 - \frac{\rho_S}{2}(\widehat{\Omega}_S(f_-, g_\theta) - 1)^2.\end{aligned}$$

Note that the fisher constraint on f ensures the stability of the training, and the Sobolev constraints on the "fake" critic f_- enforces smoothness of the "fake" critic and thus the shared CNN $\Phi_\omega(x)$. This is related to the classic Laplacian regularization in semi-supervised learning (Belkin et al., 2006).

Table 4.2 shows results of SSL on CIFAR-10 comparing the two proposed formulations. Similar to the standard procedure in other GAN papers, we do hyperparameter and model selection on the validation set. We present baselines with a similar model architecture and leave out results with significantly larger convnets. Φ_ω is similar to Dumoulin et al. (2017); Mroueh and Sercu (2017); Salimans et al. (2016) in architecture, but note that we do not use any batch, layer, or weight normalization yet obtain strong competitive accuracies. We hypothesize that we do not need any normalization in the critic, because of the implicit whitening of the feature maps introduced by the Fisher and Sobolev constraints as explained in Mroueh and Sercu (2017).

Table 4.2: CIFAR-10 error rates for varying number of labeled samples in the training set. Mean and standard deviation computed over 5 runs. We only use the $K+1$ formulation of the critic. Note that we achieve strong SSL performance without any additional tricks, and even though the critic does not have any batch, layer or weight normalization. Baselines with * use either additional models like PixelCNN, or do data augmentation (translations and flips), or use a much larger model, either of which gives an advantage over our plain simple training method. † is the result we achieved in our experimental setup under the same conditions but without "K+1" critic, since Gulrajani et al. (2017) do not have SSL results.

Number of labeled examples	1000	2000	4000	8000
Model		Misclassification rate		
CatGAN (Springenberg, 2015)			19.58	
FM (Salimans et al., 2016)	21.83 ± 2.01	19.61 ± 2.09	18.63 ± 2.32	17.72 ± 1.82
ALI (Dumoulin et al., 2017)	19.98 ± 0.3	19.09 ± 0.15	17.99 ± 0.54	17.05 ± 0.50
Tangents Reg (Kumar et al., 2017)	20.06 ± 0.5		16.78 ± 0.6	
Π-model (Laine and Aila, 2016) *			16.55 ± 0.29	
VAT (Miyato et al., 2017)			14.87	
Bad Gan (Dai et al., 2017) *			14.41 ± 0.30	
VAT+EntMin+Large (Miyato et al., 2017) *			13.15	
Sajjadi (Sajjadi et al., 2016) *			11.29	
WGAN-GP (Gulrajani et al., 2017) †	44.85 ± 0.28	37.62 ± 0.56	32.66 ± 0.48	30.38 ± 0.22
Fisher, layer norm (Mroueh and Sercu, 2017)	19.74 ± 0.21	17.87 ± 0.38	16.13 ± 0.53	14.81 ± 0.16
Fisher, no norm (Mroueh and Sercu, 2017)	21.49 ± 0.18	19.20 ± 0.46	17.30 ± 0.30	15.57 ± 0.33
Sobolev + Fisher, no norm (This Work)	20.14 ± 0.21	17.38 ± 0.10	15.77 ± 0.19	14.20 ± 0.08

4.7 Summary

We introduce the Sobolev IPM and showe that it amounts to a comparison between weighted (coordinate-wise) CDFs. The intrinsic conditioning implied by the Sobolev IPM explains the success of gradient regularization in Sobolev GAN and WGAN-GP on discrete sequence data, and particularly in text generation. We highlighted the important tradeoffs between the implicit conditioning introduced by the gradient regularizer in Sobolev IPM, and the explicit conditioning of Fisher IPM via recurrent critics and generators in conjunction with the curriculum conditioning. Both approaches succeed in text generation. We showed that Sobolev GAN achieves competitive semi-supervised learning results without the need of any normalization, thanks to the smoothness induced by the gradient regularizer. We think the Sobolev IPM point of view will open the door for designing new regularizers that induce different types of conditioning for general structured/discrete/graph data beyond sequences. Lastly, we remark that although Sobolev IPM encourages conditioning matching for sequence modeling, the performance is still not competitive with MLE training (Sutskever et al., 2014). A better IGM algorithm for sequence/text generation is still an open problem.

4.8 Appendix: Proofs

Proof of Theorem 15. Let $F_{\mathbb{P}}$ and $F_{\mathbb{Q}}$, be *the cumulative distribution functions* of $\mathbb{P}$ and $\mathbb{Q}$ respectively. We have:

$$\mathbb{P}(x) = \frac{\partial^d}{\partial x_1 \dots \partial x_d} F_{\mathbb{P}}(x), \tag{4.16}$$

We note $D = \frac{\partial^d}{\partial x_1 \dots \partial x_d}$, and $D^{-i} = \frac{\partial^{d-1}}{\partial x_1 \dots \partial x_{i-1} \partial x_{i+1} \dots \partial x_d}$, for $i = 1 \dots d$.

D^{-i} computes the $d-1$ partial derivative excluding the variable i.
In the following we assume that $F_{\mathbb{P}}$, and $F_{\mathbb{Q}}$ and its d derivatives exist and are continuous meaning that $F_{\mathbb{P}}$ and $F_{\mathbb{Q}} \in C^d(\mathcal{X})$. The objective function in Eq. (4.3) can be written as follows:

$$\begin{aligned}
\mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x) &= \int_{\mathcal{X}} f(x) D\Big(F_{\mathbb{P}}(x) - F_{\mathbb{Q}}(x)\Big) dx \\
&= \int_{\mathcal{X}} f(x) \frac{\partial}{\partial x_i} D^{-i}(F_{\mathbb{P}}(x) - F_{\mathbb{Q}}(x)) dx \\
&\quad \text{(for any } i\text{, since } F_{\mathbb{P}} \text{ and } F_{\mathbb{Q}} \in C^d(\mathcal{X})) \\
&= -\int_{\mathcal{X}} \frac{\partial f}{\partial x_i} D^{-i}(F_{\mathbb{P}}(x) - F_{\mathbb{Q}}(x)) dx \\
&\quad (f \text{ vanishes at the boundary in } \mathcal{W}_0^{1,2}(\mathcal{X}, \mu)\ )
\end{aligned}$$

Let $D^{-} = (D^{-1}, \dots, D^{-d})$ it follows that:

$$\begin{aligned}
\mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x) &= \frac{1}{d} \sum_{i=1}^{d} \int_{\mathcal{X}} \frac{\partial f}{\partial x_i} D^{-i}(F_{\mathbb{Q}}(x) - F_{\mathbb{P}}(x)) dx \\
&= \frac{1}{d} \int_{\mathcal{X}} \Big\langle \nabla_x f(x), D^{-}(F_{\mathbb{Q}}(x) - F_{\mathbb{P}}(x)) \Big\rangle_{\mathbb{R}^d} dx
\end{aligned} \tag{4.17}$$

Let us define $\mathcal{L}_2(\mathcal{X}, \mu)^{\otimes d}$ the space of measurable functions from $\mathcal{X} \to \mathbb{R}^d$. For $g, h \in \mathcal{L}_2(\mathcal{X}, \mu)^{\otimes d}$ the dot product is defined as follows:

$$\langle g, h \rangle_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} = \int_{\mathcal{X}} \langle g(x), h(x) \rangle_{\mathbb{R}^d} \mu(x) dx$$

and the norm is given :

$$\|g\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} = \int_X \|g\|^2_{\mathbb{R}^d} \mu(x)dx.$$

We can write the objective in Eq. (4.17) in term of the dot product in $\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}$:

$$\mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x) = \frac{1}{d}\left\langle \nabla_x f, \frac{D^-(F_{\mathbb{Q}} - F_{\mathbb{P}})}{\mu}\right\rangle_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}}. \tag{4.18}$$

On the other hand the constraint in Eq. (4.3) can be written in terms of the norm in $\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}$:

$$\|f\|_{\mathcal{W}_0^{1,2}(\mathcal{X},\mu)} = \|\nabla_x f\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} \tag{4.19}$$

Replacing the objective and constraint given in Eq. (4.18) and (4.19) in Eq. (4.3), we obtain:

$$\begin{aligned}
\mathcal{S}(\mathbb{P},\mathbb{Q}) &= \frac{1}{d} \sup_{f, \|\nabla_x f\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} \leq 1} \left\langle \nabla_x f, \frac{D^-(F_{\mathbb{Q}} - F_{\mathbb{P}})}{\mu}\right\rangle_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} \\
&= \frac{1}{d} \sup_{g\in\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}, \|g\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} \leq 1} \left\langle g, \frac{D^-(F_{\mathbb{Q}} - F_{\mathbb{P}})}{\mu}\right\rangle_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} \\
&= \frac{1}{d}\left\| \frac{D^-(F_{\mathbb{Q}} - F_{\mathbb{P}})}{\mu}\right\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} \\
&\left(\text{By definition of } \|.\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}}, g^* = \frac{D^-F_{\mathbb{Q}}(x) - D^-F_{\mathbb{P}}(x)}{\mu(x)} \frac{1}{\left\|\frac{D^-(F_{\mathbb{Q}} - F_{\mathbb{P}})}{\mu}\right\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}}}\right) \\
&= \frac{1}{d}\sqrt{\int_{\mathcal{X}} \frac{\left\|D^-F_{\mathbb{Q}}(x) - D^-F_{\mathbb{P}}(x)\right\|^2}{\mu(x)} dx}.
\end{aligned}$$

Hence we find also that the optimal critic f^* satisfies:

$$\nabla_x f^*(x) = \frac{D^-F_{\mathbb{Q}}(x) - D^-F_{\mathbb{P}}(x)}{\mu(x)} \frac{1}{\left\|\frac{D^-(F_{\mathbb{Q}} - F_{\mathbb{P}})}{\mu}\right\|_{\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}}}.$$

□

Proof of Lemma 16.

$$\begin{aligned}
\mathbb{E}_{x\sim\mathbb{P}} f(x) - \mathbb{E}_{x\sim\mathbb{Q}} f(x) &= \frac{1}{d}\int_{\mathcal{X}} \left\langle \nabla_x f(x), D^-(F_{\mathbb{Q}}(x) - F_{\mathbb{P}}(x))\right\rangle_{\mathbb{R}^d} dx \\
&= \mathcal{S}_\mu(\mathbb{P},\mathbb{Q}) \int_{\mathcal{X}} \left\langle \nabla_x f(x), \frac{D^-(F_{\mathbb{Q}}(x) - F_{\mathbb{P}}(x))}{\mu(x) d \mathcal{S}_\mu(\mathbb{P},\mathbb{Q})}\right\rangle_{\mathbb{R}^d} \mu(x)dx \\
&= \mathcal{S}_\mu(\mathbb{P},\mathbb{Q}) \int_{\mathcal{X}} \left\langle \nabla_x f(x), \nabla_x f^*(x)\right\rangle \mu(x)dx \\
&= \mathcal{S}_\mu(\mathbb{P},\mathbb{Q}) \left\langle f, f^*\right\rangle_{\mathcal{W}_0^{1,2}}
\end{aligned}$$

Hence we have:

$$\sup_{f\in\mathcal{H},\|f\|_{\mathcal{W}_0^{1,2}}\leq 1} \mathbb{E}_{x\sim\mathbb{P}}f(x)-\mathbb{E}_{x\sim\mathbb{Q}}f(x)=\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})\sup_{f\in\mathcal{H},\|f\|_{\mathcal{W}_0^{1,2}}\leq 1}\langle f,f^*\rangle_{\mathcal{W}_0^{1,2}},$$

It follows therefore that:

$$\mathcal{S}_{\mathcal{H}}(\mathbb{P},\mathbb{Q})=\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})\sup_{f\in\mathcal{H},\|f\|_{\mathcal{W}_0^{1,2}}\leq 1}\langle f,f^*\rangle_{\mathcal{W}_0^{1,2}}$$

□

We conclude that the Sobolev IPM can be approximated in arbitrary space as long as it has enough capacity to approximate the optimal critic. Interestingly the approximation error is measured now with the Sobolev semi-norm, while in Fisher it was measured with the Lebesgue norm. Approximations with Sobolev Semi-norms are stronger then Lebesgue norms as given by the Poincare inequality ($||f||_{\mathcal{L}_2}\leq C\,\|f\|_{\mathcal{W}_0^{1,2}}$), meaning if the error goes to zero in Sobolev sense it also goes to zero in the Lebesgue sense , but the converse is not true.

Proof of Theorem 17. The proof follows similar arguments in the proofs of the analysis of Laplacian regularization in semi-supervised learning studied by Alaoui et al. (2016).

$$\begin{aligned}\mathcal{S}_\mu(\mathbb{P},\mathbb{Q}) \quad &= \sup_{f\in\mathcal{W}_0^{1,2}}\Big\{\mathbb{E}_{x\sim\mathbb{P}}\,[f(x)]-\mathbb{E}_{x\sim\mathbb{Q}}\,[f(x)]\Big\}\\ s.t. \quad & \qquad \mathbb{E}_{x\sim\mu}\|\nabla f(x)\|_2^2\leq 1,\end{aligned}\tag{4.20}$$

Note that this problem is convex in f (Ekeland and Turnbull, 1983). Writing the lagrangian for Eq. (4.20) we get :

$$\begin{aligned}L(f,\lambda)&=\mathbb{E}_{x\sim\mathbb{P}}\,[f(x)]-\mathbb{E}_{x\sim\mathbb{Q}}\,[f(x)]+\frac{\lambda}{2}\Big(1-\mathbb{E}_{x\sim\mu}\|\nabla_x f(x)\|_2^2\Big)\\ &=\int_{\mathcal{X}}f(x)\,\big(\mathbb{P}(x)-\mathbb{Q}(x)\big)\,dx+\frac{\lambda}{2}\Big(1-\int_{\mathcal{X}}\|\nabla_x f(x)\|_2^2\mu(x)dx\Big)\\ &=\int_{\mathcal{X}}f(x)\,\mu_1(x)\,dx+\frac{\lambda}{2}\Big(1-\int_{\mathcal{X}}\|\nabla_x f(x)\|_2^2\;\mu(x)\,dx\Big)\end{aligned}$$

We denote $\big(\mathbb{P}(x)-\mathbb{Q}(x)\big)$ as $\mu_1(x)$.To get the optimal f, we need to apply KKT conditions on the above equation.

$$L(f,\lambda)=\int_{\mathcal{X}}f(x)\,\mu_1(x)\,dx+\frac{\lambda}{2}\Big(1-\int_{\mathcal{X}}\|\nabla_x f(x)\|_2^2\;\mu(x)\,dx\Big)$$

From the calculus of variations:

$$\begin{aligned}
L(f+\epsilon h,\lambda) &= \int_{\mathcal{X}} (f+\epsilon h)(x)\ \mu_1(x)\ dx + \frac{\lambda}{2}\Big(1-\int_{\mathcal{X}} \|\nabla_x\big(f+\epsilon h\big)(x)\|_2^2\ \ \mu(x)\ dx\Big)\\
&= \int_{\mathcal{X}} (f(x)+\epsilon h(x))\ \mu_1(x)\ dx + \frac{\lambda}{2}\Big(1-\int_{\mathcal{X}} \big\langle \nabla_x\big(f+\epsilon h\big)(x), \nabla_x\big(f+\epsilon h\big)(x)\big\rangle\ \ \mu(x)\ dx\Big)\\
&= \int_{\mathcal{X}} (f(x)+\epsilon h(x))\ \mu_1(x)\ dx\\
&+ \frac{\lambda}{2}\Big(1-\int_{\mathcal{X}} \big[\|\nabla_x f(x)\|_2^2 + 2\epsilon\langle \nabla_x f(x), \nabla_x h(x)\rangle + \mathcal{O}(\epsilon^2)\big]\ \mu(x)dx\Big)\\
&= L(f,\lambda) + \epsilon\int_{\mathcal{X}} h(x)\ \mu_1(x)\ dx - \lambda\epsilon\int_{\mathcal{X}} \langle \nabla_x f(x), \nabla_x h(x)\rangle\ \mu(x)\ dx + \mathcal{O}(\epsilon^2)\\
&= L(f,\lambda) + \epsilon\Big[\int_{\mathcal{X}} h(x)\ \mu_1(x)\ dx - \lambda\int_{\mathcal{X}} \langle \nabla_x f(x), \nabla_x h(x)\rangle\ \mu(x)\ dx\Big] + \mathcal{O}(\epsilon^2)
\end{aligned}$$

Now we apply integration by part and set h to be zero at boundary as in Alaoui et al. (2016). We get :

$$\begin{aligned}
\int_{\mathcal{X}} \langle \nabla_x f(x), \nabla_x h(x)\rangle\ \mu(x)\ dx &= \int_{\mathcal{X}} \langle \nabla_x f(x)\ \mu(x), \nabla_x h(x)\rangle\ dx\\
&= \oint_{\partial\mathcal{X}} h(x)\mu(x)\nabla_x f(x).n(x)dS(x)\ \ - \int_{\mathcal{X}} div\big(\mu(x)\nabla_x f(x)\big)\ h(x)\ dx\\
&= -\int_{\mathcal{X}} div\big(\mu(x)\nabla_x f(x)\big)\ h(x)\ dx
\end{aligned}$$

Hence,

$$\begin{aligned}
L(f+\epsilon h,\lambda) &= L(f,\lambda) + \epsilon\Big[\int_{\mathcal{X}} \mu_1(x)\ h(x)\ dx + \lambda\int_{\mathcal{X}} div\big(\mu(x)\nabla_x f(x)\big)\ h(x)\ dx\Big] + \mathcal{O}(\epsilon^2)\\
&= L(f,\lambda) + \epsilon\int_{\mathcal{X}} \Big(\mu_1(x) + \lambda\ \ div\big(\mu(x)\nabla_x f(x)\big)\Big)\ h(x)\ dx\ \ + \mathcal{O}(\epsilon^2)
\end{aligned}$$

The functional derivative of $L(f,\lambda)$, at any test function h vanishing on the boundary:

$$\begin{aligned}
\int_{\mathcal{X}} \frac{\partial L(f,\lambda)}{\partial f}(x)h(x)dx &= \lim_{\epsilon\to 0}\frac{L(f+\epsilon h,\lambda)-L(f,\lambda)}{\epsilon}\\
&= \int_{\mathcal{X}} \Big(\mu_1(x) + \lambda\ \ div\big(\mu(x)\nabla_x f(x)\big)\Big)\ h(x)\ dx
\end{aligned}$$

Hence we have:

$$\frac{\partial L(f,\lambda)}{\partial f}(x) = \mu_1(x) + \lambda\ \ div\big(\mu(x)\nabla_x f(x)\big)$$

For the optimal f^*, λ^* first order optimality condition gives us:

$$\mu_1(x) + \lambda^*\ \ div\big(\mu(x)\nabla_x f^*(x)\big) = 0 \tag{4.21}$$

and

$$\int_{\mathcal{X}} \left\|\nabla_x f^*(x)\right\|^2 \mu(x)dx = 1 \tag{4.22}$$

Note that (See Alaoui et al. (2016)) :

$$div\big(\mu(x)\nabla_x f^*(x)\big) = \mu(x)\Delta_2 f^*(x) + \langle \nabla_x \mu(x), \nabla_x f^*(x)\rangle,$$

since $div(\nabla_x f^*(x)) = \Delta_2 f^*(x)$. Hence from Eq. (4.21)

$$\begin{aligned}
&\mu_1(x) + \lambda^*\ div\ \big(\mu(x)\nabla_x f^*(x)\big) = 0\\
\Rightarrow\ &\mu_1(x) + \lambda^*\ \big(\mu(x)\Delta_2 f^*(x) + \langle \nabla_x \mu(x), \nabla_x f^*(x)\rangle\big) = 0\\
\Rightarrow\ &\mu_1(x) + \lambda^*\ \mu(x)\Delta_2 f^*(x) + \lambda^*\langle \nabla_x \mu(x), \nabla_x f^*(x)\rangle = 0\\
\Rightarrow\ &\Delta_2 f^*(x) + \left\langle \frac{\nabla_x \mu(x)}{\mu(x)}, \nabla_x f^*(x)\right\rangle + \frac{\mu_1(x)}{\lambda^*\mu(x)} = 0\\
\Rightarrow\ &\Delta_2 f^*(x) + \langle \nabla_x \log\mu(x), \nabla_x f^*(x)\rangle + \frac{\mathbb{P}(x)-\mathbb{Q}(x)}{\lambda^*\mu(x)} = 0
\end{aligned} \tag{4.23}$$

Hence f^*, λ^* satisfies :

$$\Delta_2 f^*(x) + \langle \nabla_x \log\mu(x), \nabla_x f^*(x)\rangle + \frac{\mathbb{P}(x)-\mathbb{Q}(x)}{\lambda^*\mu(x)} = 0 \tag{4.24}$$

and

$$\int_{\mathcal{X}} \left\|\nabla_x f^*(x)\right\|^2 \mu(x)dx = 1. \tag{4.25}$$

Let us verify that the optimal critic as found in the geometric definition (Theorem 15) of Sobolev IPM that satisfies:

$$\nabla_i f^*(x) = \frac{\partial f^*(X)}{\partial x_i} = \frac{D^{-i}F_{\mathbb{Q}}(x) - D^{-i}F_{\mathbb{P}}(x)}{\lambda^* d\ \ \mu(x)} \qquad \forall\ i \in [d], \tag{4.26}$$

satisfies indeed the PDE.

From Eq. (4.26), we want to compute $\frac{\partial^2 f(x)}{\partial x_i^2}$ for all i:

$$\begin{aligned}
\frac{\partial^2 f(x)}{\partial x_i^2} &= \frac{1}{\lambda^* d}\left[\frac{\mu(x)\big[\frac{\partial}{\partial x_i}(D^{-i}F_{\mathbb{Q}}(x) - D^{-i}F_{\mathbb{P}}(x))\big] - \big[D^{-i}F_{\mathbb{Q}}(x) - D^{-i}F_{\mathbb{P}}(x)\big]\nabla_i\mu(X)}{\mu^2(x)}\right]\\
&= \frac{1}{\lambda^* d}\left[\frac{\mu(x)\big[\mathbb{Q}(x) - \mathbb{P}(x)\big] - \big[D^{-i}F_{\mathbb{Q}}(x) - D^{-i}F_{\mathbb{P}}(x)\big]\nabla_i\mu(X)}{\mu^2(x)}\right]\\
&= \frac{\mathbb{Q}(x)-\mathbb{P}(x)}{\lambda^* d\ \mu(x)} - \frac{\nabla_i\mu(x)}{\mu(x)}\nabla_i f^*(x)
\end{aligned}$$

Hence,

$$\frac{\partial^2 f(x)}{\partial x_i^2} + \frac{\nabla_i \mu(x)}{\mu(x)} \nabla_i f(x) + \frac{\big(\mathbb{P}(x) - \mathbb{Q}(x)\big)}{\lambda^* d \;\; \mu(x)} = 0 \tag{4.27}$$

Adding Eq. (4.27) for all $i \in [d]$, we get :

$$\sum_{i=1}^{d} \left(\frac{\partial^2 f(x)}{\partial x_i^2} + \frac{\nabla_i \mu(x)}{\mu(x)} \nabla_i f(x) + \frac{\big(\mathbb{P}(x) - \mathbb{Q}(x)\big)}{\lambda^* d \;\; \mu(x)} \right) = 0$$

As a result, the solution f^* of the partial differential equation given in Eq. (4.24) satisfies the following :

$$\frac{\partial f^*(x)}{\partial x_i} = \frac{D^{-i} F_{\mathbb{Q}}(x) - D^{-i} F_{\mathbb{P}}(x)}{\lambda^* d \;\; \mu(x)} \qquad \forall\, i \in [d]$$

Using the constraint in (4.25) we can get the value of λ^* :

$$\begin{aligned} &\int \|\nabla f^*(x)\|^2 \; \mu(x) \; dx = 1 \\ \Rightarrow &\int \sum_{i=1}^{d} \Big(\frac{\partial f^*(x)}{\partial x_i} \Big)^2 \; \mu(x) \; dx = 1 \\ \Rightarrow &\lambda^* = \frac{1}{d} \sqrt{\sum_{i=1}^{d} \int \frac{\big(D^{-i} F_{\mathbb{Q}}(x) - D^{-i} F_{\mathbb{P}}(x)\big)^2}{\mu(x)} \; dx} = \mathcal{S}_\mu(\mathbb{P}, \mathbb{Q}). \end{aligned}$$

□

Proof of Theorem 18. Define the Stein operator (Oates et al., 2017):

$$\begin{aligned} T(\mu)[\nabla_x f(x)] &= \frac{1}{2} \langle \nabla_x f(x), \nabla_x \log \mu(x) \rangle + \frac{1}{2} \langle \nabla_x, \nabla_x f(x) \rangle \\ &= \frac{1}{2} \langle \nabla_x f(x), \nabla_x \log \mu(x) \rangle + \frac{1}{2} \Delta_2 f(x). \end{aligned}$$

This operator was later used in defining the Stein discrepancy (Chwialkowski et al., 2016; Gorham and Mackey, 2015; Liu, 2017; Liu et al., 2016).

Recall that Barbour generator theory provides us a way of constructing such operators that produce mean zero function under μ. It is easy to verify that:

$$\mathbb{E}_{x \sim \mu} T(\mu) \nabla_x f(x) = 0.$$

Recall that this operator arises from the overdamped Langevin diffusion, defined by the stochastic differential equation:

$$dx_t = \frac{1}{2} \nabla_x \log \mu(x_t) + dW_t$$

where $(W_t)_{t\geq 0}$ is a Wiener process. This is related to plug and play networks for generating samples if the distribution is known, using the stochastic differential equation.

From Theorem 17, it is easy to see that the PDE the Sobolev Critic $(f^*, \lambda^* = \mathcal{S}_\mu(\mathbb{P},\mathbb{Q}))$ can be written in term of Stein Operator as follows:

$$T(\mu)[\nabla_x f^*](x) = \frac{1}{2\lambda^*}\frac{\mathbb{Q}(x)-\mathbb{P}(x)}{\mu(x)}$$

Taking absolute values and the expectation with respect to $\mathbb{Q}$:

$$\left|\mathbb{E}_{x\sim\mathbb{Q}}\left[T(\mu)\nabla_x f^*(x)\right]\right| = \frac{1}{2\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})}\left|\mathbb{E}_{x\sim\mathbb{Q}}\left[\frac{\mathbb{Q}(x)-\mathbb{P}(x)}{\mu(x)}\right]\right|$$

Recall that the definition of Stein Discrepancy :

$$\mathbb{S}(\mathbb{Q},\mu) = \sup_{\vec{g}\in\mathcal{L}_2(\mathcal{X},\mu)^{\otimes d}} \left|\mathbb{E}_{x\sim\mathbb{Q}}\left[T(\mu)\vec{g}(x)\right]\right|$$

It follows that Sobolev IPM critic satisfies:

$$\left|\mathbb{E}_{x\sim\mathbb{Q}}\left[T(\mu)\nabla_x f^*(x)\right]\right| \leq \mathbb{S}(\mathbb{Q},\mu),$$

Hence we have the following inequality:

$$\frac{1}{2\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})}\left|\mathbb{E}_{x\sim\mathbb{Q}}\left[\frac{\mathbb{Q}(x)-\mathbb{P}(x)}{\mu(x)}\right]\right| \leq \mathbb{S}(\mathbb{Q},\mu)$$

This is equivalent to:

$$\left|\mathbb{E}_{x\sim\mathbb{Q}}\left[\frac{\mathbb{Q}(x)-\mathbb{P}(x)}{\mu(x)}\right]\right| \leq 2 \underbrace{\mathbb{S}(\mathbb{Q},\mu)}_{\text{Stein Good fitness of the model } \mathbb{Q} \text{ w.r.t to } \mu} \underbrace{\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})}_{\text{Sobolev Distance}}$$

Similarly we obtain:

$$\left|\mathbb{E}_{x\sim\mathbb{P}}\left[\frac{\mathbb{Q}(x)-\mathbb{P}(x)}{\mu(x)}\right]\right| \leq 2 \underbrace{\mathbb{S}(\mathbb{P},\mu)}_{\text{Stein Good fitness of } \mu \text{ w.r.t to } \mathbb{P}} \underbrace{\mathcal{S}_\mu(\mathbb{P},\mathbb{Q})}_{\text{Sobolev Distance}}$$

For instance consider $\mu = \mathbb{P}$, we have therefore:

$$\frac{1}{2}\left|\mathbb{E}_{x\sim\mathbb{Q}}\left[\frac{\mathbb{Q}(x)}{\mathbb{P}(x)}\right]-1\right| \leq \mathbb{S}(\mathbb{Q},\mathbb{P})\mathcal{S}_{\mathbb{P}}(\mathbb{P},\mathbb{Q}).$$

Note that the left hand side of the inequality is not the total variation distance. Hence for a sequence $\mathbb{Q}_n$ if the Sobolev distance goes $\mathcal{S}_{\mathbb{P}}(\mathbb{P},\mathbb{Q}_n)\to 0$, the ratio $r_n(x) = \frac{\mathbb{Q}_n(x)}{\mathbb{P}(x)}$ converges in expectation (w.r.t to $\mathbb{Q}$) to 1. The speed of the convergence is given by the Stein Discrepancy $\mathbb{S}(\mathbb{Q}_n,\mathbb{P})$.

One important observation here is that convergence of PDF ratio is weaker than the conditional CDF as given by the Sobolev distance and of the good fitness of score function as given by Stein discrepancy.

□

4.9 Appendix: Hyperparameters for SSL

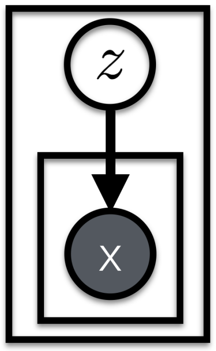
We use Adam with learning rate $\eta = 2\text{e}{-4}$, $\beta_1 = 0.5$ and $\beta_2 = 0.999$, both for critic f (without BN) and Generator (with BN). We selected $\lambda_{CE} = 1.5$ from $[0.8, 1.5, 3.0, 5.0]$. We train all models for 350 epochs. We used some L2 weight decay: $1\text{e}{-6}$ on ω, S (i.e. all layers except last) and $1\text{e}{-3}$ weight decay on the last layer v. For formulation 1 (Fisher only) we have $\rho_F = 1\text{e}{-7}$, modified critic learning rate $\eta_D = 1\text{e}{-4}$, critic iters $n_c = 2$. For formulation 2 (Sobolev + Fisher) we have $\rho_F = 5\text{e}{-8}$, $\rho_S = 2\text{e}{-8}$, critic iters $n_c = 1$. For the WGAN-GP (Gulrajani et al., 2017) baseline SSL experiment we followed the original paper with critic iters $n_c = 5$, $\eta_G = \eta_D = 1\text{e}{-4}$, Adam β_2=0.9 and GP weight $\lambda_{GP} = 10.0$. Architectures are as below. We determined $\lambda_{CE} = 0.3$ to be optimal from $[0.03, 0.1, 0.3, 1.0, 3.0]$. As mentioned in Section 4.6.2, the K+1 critic formulation is not able to fit the training set with the GP constraint, so we fall back to the plain critic formulation where the critic $\langle v, \Phi_\omega(x)\rangle$ does not interact with the classifier $\langle S, \Phi_\omega(x)\rangle$.

Chapter 5

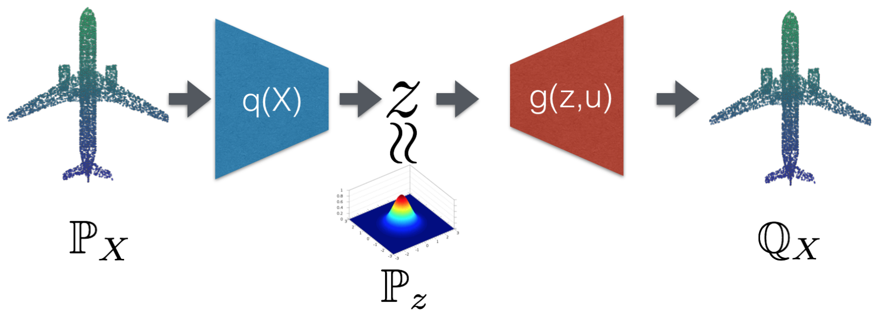
Point Cloud Generation with IGMs

A fundamental problem in machine learning is that given a data set, learn a generative model that can efficiently generate *arbitrary many new sample points* from the domain of the underlying distribution (Bishop, 2006). Recently, capturing 3D information is garnering attention. There are many different data types for 3D information, such as CAD, 3D meshes, and point clouds. 3D point clouds are getting popular since these store more information than 2D images and sensors capable of collecting point clouds have become more accessible. These include Lidar on self-driving cars, Kinect for Xbox, and face identification sensor on phones. Compared to other formats, point clouds can be easily represented as a set of points, which has several advantages, such as permutation invariance of the set members. The algorithms which can effectively learn from this type of data is an emerging field (Fan et al., 2017; Kalogerakis et al., 2017; Qi et al., 2017a,b; Zaheer et al., 2017a). However, compared to supervised learning, unsupervised generative models for 3D data are still underexplored (Achlioptas et al., 2018; Oliva et al., 2018).

In this chapter, we we propose an implicit generative model with a hierarchical sampling and an inference network for point clouds. The proposed algorithm learns a stochastic procedure which can generate new point clouds and draw samples from the generated point clouds without explicitly modeling the underlying density function. The proposed algorithm, PC-GAN, is a generic framework which can incorporate many existing IGM variants, such as GANs. The algorithm can also be treated as an autoencoder extension with an IGM decoder. By utilizing the low dimensionality of point clouds, we further propose a *sandwiching* objective by considering both upper and lower bound estimators of Wasserstein distance, which can lead to tighter approximation. Evaluation on ModelNet40 shows excellent generalization capability of PC-GAN. We first demonstrate that we can sample from the learned model to generate new point clouds and the latent representations learned by the inference network provide meaningful interpolations between point clouds. Then we show the conditional generation results on *unseen* classes of objects, which demonstrates the superior generalization ability of PC-GAN.



(a) The graphical model diagram of generating point clouds, where z is the latent variable indicating which point cloud to be generated.



(b) The implementation of PC-GAN, which is an extension of the autoencoder. The inference network q can be treated as an encoder, while the point cloud generator is an IGM decoder, which allows to generate (sample) arbitrarily many points.

Figure 5.1: Overview of PC-GAN.

5.1 Learning to Generate Point Clouds

A point cloud for an object z is a *set* of n low dimensional vectors $X = \{x_1, ..., x_n\}$ with $x_i \in \mathbb{R}^d$, where d is usually 3 and n can be infinite. M different objects can be described as a collection of point clouds $X^{(1)}, ..., X^{(M)}$. The difficulties of modeling point clouds are

1. the points in a point cloud are orderless (permutation invariant), which are different from pixels in images, and
2. the number of points in each point clouds can be different.

To model point clouds by taking these two difficulties into account, the key assumption is treating each point cloud as finite samples from a 3-dimensional distribution. Based on the De-Finetti theorem, we could factor the probability with some suitably defined z, such as object representation of point clouds, as $p(X) = \int_z \prod_{i=1}^n p(x_i|z)p(z)dz$.

Given z, learning a 3-dimensional distribution $p(x|z)$ is an easier task. Instead of learning explicit densities (Eckart et al., 2015; Jian and Vemuri, 2005; Strom et al., 2010) to model this 3-dimensional distribution, we are interested in implicit generative models with an IGM, where we can generate point clouds via sampling from the learned IGM.

Formally, given z, we train an IGM by modeling a transformation function $g(z, u)$ such that $x = g(z, u)$, where $u \sim \mathbb{P}_u$ is an initial randomness. The generator $g(z, z)$ follows $\mathbb{Q}_X$ by optimizing a probabilistic divergence $D(\mathbb{P}_X \| \mathbb{Q}_X)$ between $p(x|z)$, which is denoted as $\mathbb{P}_X$, and the distribution $\mathbb{Q}_X$ of $g(z, u)$. We can adopt any probability distance or divergence for modeling $D(\mathbb{P}_X \| \mathbb{Q}_X)$, such as the distances defined for training GANs as we discussed in the previous chapters. The full objective can be written as $\mathbb{E}_X \left[\min_g D(\mathbb{P}_X \| \mathbb{Q}_X) \right]$.

Inference Since z is an unobserved latent variable for modeling different objects, we need to infer z during training. The proposed algorithm has to concurrently learn the inference network $q(X) \approx z$ while we learn $p(x|z)$.

Hierarchical Sampling The overview of proposed algorithm for point cloud generation (PC-GAN) is shown in Figure 5.1. The sampling process can be better illustrated via a graphical diagram as shown in Figure 5.1a. We first sample the unobserved latent variable $z \sim \mathbb{P}_z$, then sample the points of X given z. In Figure 5.1b, we show the implementation of the proposed algorithm. To sample z in the generation process, during the training, we have to enforce $\mathbb{P}_z$ to be closed to a prior distribution (Kingma and Welling, 2013), or learn to sample from $\mathbb{P}_z$ via the other IGM (Tomczak and Welling, 2017; Zhao et al., 2017).

5.1.1 Auto-Encoding Perspective

The proposed PC-GAN can be viewed as an auto-encoding-based generative algorithm for distributions of distributions (point clouds). The inference network $q(X)$ encode the object into the latent representation, which can be treated as encoders or recognition networks in the existing literature (Kingma and Welling, 2013; Salakhutdinov and Hinton, 2009). The generator g is an “IGM decoder”, which is an extension of typical decoders or generation networks. Instead of generating a single points (Kingma and Welling, 2013), g models a sampling process to sample a point cloud. The hierarchical sampling follows the ideas of matching or learning priors (Makhzani et al., 2015; Tolstikhin et al., 2017a; Tomczak and Welling, 2017; Zhao et al., 2017).

Achlioptas et al. (2018) is a similar work, which also explore an AAE variant (Makhzani et al., 2015; Tolstikhin et al., 2017a) for point cloud. They use a specially-designed encoder network (Qi et al., 2017a) for learning a compressed representation for point clouds before training GAN on the latent space. However, their decoder is restricted to be an MLP which generates m fixed number of points, where m has to be pre-defined. That is, the output of their decoder is fixed to be $3m$ for 3D point clouds, while the output of the proposed g is only 3 dimensional and g can generate arbitrarily many points by sampling different random noise z as input. Groueix et al. (2018b); Yang et al. (2018) propose similar decoders to g with fixed grids to break the limitation of Achlioptas et al. (2018) aforementioned, but they use heuristic Chamfer distance and do not exploit generative models for point clouds.

5.1.2 Different Discrepancies for Matching Point Clouds

To train the generator g using a GAN-like objective as reconstruction loss in autoencoder for point clouds, we need a discriminator $f(\cdot)$ to distinguishes generated samples and true samples conditioned on z. Combining with the inference network $q(X)$ discussed aforementioned, the objecitve with IPM-based GANs can be written as

$$\mathbb{E}_{z\sim\mathbb{P}_z}\left[\min_{g,q}\underbrace{\max_{f\in\mathcal{F}}\mathbb{E}_{x\sim p(X|z)}\left[f(x)\right]-\mathbb{E}_{X\sim p(X|z)}\left[f(g(q(X),u))\right]}_{D(\mathbb{P}_X\|\mathbb{Q}_X)}\right], \tag{5.1}$$

where $\mathcal{F}$ is the constraint for different probabilistic distances, such as 1-Lipschitz (Arjovsky et al., 2017), L^2 ball (Mroueh and Sercu, 2017) or Sobolev ball (Mroueh et al., 2018).

5.1.3 Tighter Solutions via Sandwiching

We begin by noting that the popular Wasserstein GAN (Arjovsky et al., 2017), aims to optimize g by $\min w(\mathbb{P}_X, \mathbb{Q}_X)$, where $w(\mathbb{P}_X, \mathbb{Q}_X)$ is the Wasserstein distance between the truth $\mathbb{P}_X$ and the generated distribution $\mathbb{Q}_X$ of g. Many GAN works (e.g. Arjovsky et al. (2017)) approximate $w(\mathbb{P}_X, \mathbb{Q}_X)$ in dual (a maximization problem), such as (5.1), by neural networks. The resulting estimator $W_L(\mathbb{P}_X, \mathbb{Q}_X)$ is a lower bound of the true Wasserstein distance, as neural networks can only recover a subset of 1-Lipschitz functions (Arora et al., 2017). However, finding a lower bound $W_L(\mathbb{P}_X, \mathbb{Q}_X)$ for $w(\mathbb{P}_X, \mathbb{Q}_X)$ may not be an ideal surrogate for solving a minimization problem $\min w(\mathbb{P}_X, \mathbb{Q}_X)$. In optimal transport literature, Wassertein distance is usually estimated by approximating matching cost, $W_U(\mathbb{P}_X, \mathbb{Q}_X)$, which gives us an upper bound of the true Wasserstein distance.

We propose to combine, in general, a lower bound W_L and upper bound estimator W_U by sandwiching the solution between the two, i.e. we solve the following minimization problem:

$$\min_G \quad W_U(\mathbb{P}_X, \mathbb{Q}_X) \qquad \text{s.t.} \quad W_U(\mathbb{P}_X, \mathbb{Q}_X) - W_L(\mathbb{P}_X, \mathbb{Q}_X) < \lambda \tag{5.2}$$

The problem can be simplified and solved using method of lagrange multipliers as follows:

$$\min_G W_s(\mathbb{P}_X, \mathbb{Q}_X) := (1-s)W_U(\mathbb{P}_X, \mathbb{Q}_X) + sW_L(\mathbb{P}_X, \mathbb{Q}_X) \tag{5.3}$$

By solving the new *sandwiched problem* (5.3), we show that under certain conditions we obtain a better estimator of Wasserstein distance in the following lemma:

Lemma 20. *Suppose we have two approximators to Wasserstein distance: an upper bound W_U and a lower W_L, such that $\forall \mathbb{P}_X, \mathbb{Q}_X : (1+\epsilon_1)w(\mathbb{P}_X, \mathbb{Q}_X) \leq W_U(\mathbb{P}_X, \mathbb{Q}_X) \leq (1+\epsilon_2)w(\mathbb{P}_X, \mathbb{Q}_X)$ and $\forall P, G : (1-\epsilon_2)w(\mathbb{P}_X, \mathbb{Q}_X) \leq W_L(\mathbb{P}_X, \mathbb{Q}_X) \leq (1-\epsilon 1)w(\mathbb{P}_X, \mathbb{Q}_X)$ respectively, for some $\epsilon_2 > \epsilon_1 > 0$ and $\epsilon_1 > \epsilon_2/3$. Then, using the sandwiched estimator W_s from (5.3), we can achieve a tighter estimator of the Wasserstein distance than using either one estimator, i.e.*

$$\exists s : |W_s(\mathbb{P}_X, \mathbb{Q}_X) - w(\mathbb{P}_X, \mathbb{Q}_X)| < \min\{|W_U(\mathbb{P}_X, \mathbb{Q}_X) - w(\mathbb{P}_X, \mathbb{Q}_X)|, |W_L(\mathbb{P}_X, \mathbb{Q}_X) - w(\mathbb{P}_X, \mathbb{Q}_X)|\} \tag{5.4}$$

Proof. We prove the claim by show that LHS is at most ϵ_1, which is the lower bound for

RHS.

$$
\begin{aligned}
&|W_s(\mathbb{P}_X,\mathbb{Q}_X) - w(\mathbb{P}_X,\mathbb{Q}_X)| \\
&\quad = |(1-s)W_U(\mathbb{P}_X,\mathbb{Q}_X) + sW_L(\mathbb{P}_X,\mathbb{Q}_X) - w(\mathbb{P}_X,\mathbb{Q}_X)| \\
&\quad = |(1-s)(W_U(\mathbb{P}_X,\mathbb{Q}_X) - w(\mathbb{P}_X,\mathbb{Q}_X)) - s(w(\mathbb{P}_X,\mathbb{Q}_X) - W_L(\mathbb{P}_X,\mathbb{Q}_X))| \\
&\quad \le \max\{(1-s)\underbrace{(W_U(\mathbb{P}_X,\mathbb{Q}_X) - w(\mathbb{P}_X,\mathbb{Q}_X))}_{\le\epsilon_2}, s\underbrace{(w(\mathbb{P}_X,\mathbb{Q}_X) - W_L(\mathbb{P}_X,\mathbb{Q}_X))}_{\le\epsilon_2}\} \\
&\qquad\qquad - \min\{(1-s)\underbrace{(W_U(\mathbb{P}_X,\mathbb{Q}_X) - w(\mathbb{P}_X,\mathbb{Q}_X))}_{\ge\epsilon_1}, s\underbrace{(w(\mathbb{P}_X,\mathbb{Q}_X) - W_L(\mathbb{P}_X,\mathbb{Q}_X))}_{\ge\epsilon_1}\} \\
&\quad \le \max\{(1-s), s\}\epsilon_2 - \min\{(1-s), s\}\epsilon_1
\end{aligned}
\tag{5.5}
$$

Without loss of generality we can assume $\lambda < 0.5$, which brings us to

$$|W_s(\mathbb{P}_X,\mathbb{Q}_X) - w(\mathbb{P}_X,\mathbb{Q}_X)| \le (1-\lambda)\epsilon_2 - \lambda\epsilon_1 \tag{5.6}$$

Now if we chose $\frac{\epsilon_2-\epsilon_1}{\epsilon_2+\epsilon_1} < \lambda < 0.5$, then $|W_s(\mathbb{P}_X,\mathbb{Q}_X) - w(\mathbb{P}_X,\mathbb{Q}_X)| < \epsilon_1$ as desired. $\square$

Upper Bound Implementation The primal of Wasserstein distance is defined as

$$w(\mathbb{P}_X,\mathbb{Q}_X) = \inf_{\gamma\in\Gamma(\mathbb{P}_X,\mathbb{Q}_X)} \int \|x-y\|_1 d\gamma(x,y),$$

where Γ is the *coupling* of $\mathbb{P}_X$ and $\mathbb{Q}_X$. The Wasserstein distance is also known as optimal transport (OT) or earth moving distance (EMD). As the name suggests, when $w(\mathbb{P}_X,\mathbb{Q}_X)$ is estimated with finite number of samples $X = x_1,\dots,x_n$ and $Y = y_1,\dots,y_n$, we find the one-to-one matching between X and Y such that the total pairwise distance is minimal. The resulting minimal total (average) pairwise distance is $w(X,Y)$. In practice, finding the exact matching efficiently is non-trivial and still an open research problem (Peyré et al., 2019). Instead, we consider an approximation provided by Bertsekas (1985). One can show that algorithm terminates with a valid matching and the resulting matching cost $W_U(X,Y)$ is an ϵ-approximation of $w(X,Y)$. Thus, the estimator can serve as an upper bound, i.e.

$$w(X,Y) \le W_U(X,Y) \le (1+\epsilon)w(X,Y),$$

We remark estimating Wasserstein distance $w(\mathbb{P}_X,\mathbb{Q}_X)$ with finite sample in primal is only favorable in low dimensional data, such as point clouds. The error between $w(\mathbb{P}_X,\mathbb{Q}_X)$ and $w(X,Y)$ is $O(1/n^{1/d})$, where d is the dimension of data (Weed and Bach, 2017). Therefore, for high dimensional data, such as images, we cannot accurately estimate Wasserstein distance in primal as well as its upper bound with a small minibatch. A modified primal with low sample complexity is still an open research problem (Cuturi, 2013; Genevay et al., 2018).

Lower Bound Implementation The dual of Wasserstein distance is defined as

$$w(\mathbb{P}_X,\mathbb{Q}_X) = \sup_{f\in\mathcal{L}_1} \mathbb{E}_{x\sim\mathbb{P}_X} f(x) - \mathbb{E}_{x\sim\mathbb{Q}_X} f(x), \tag{5.7}$$

where $\mathcal{L}_k$ is the set of k-Lipschitz functions whose Lipschitz constant is no larger than k. In practice, deep neural networks parameterized by ϕ with constraints $f_\phi \in \Omega_\phi$ (Arjovsky et al., 2017), result in a distance approximation

$$W_L(\mathbb{P}_X, \mathbb{Q}_X) = \max_{f_\phi \in \Omega_\phi} \mathbb{E}_{x \sim \mathbb{P}_X} f_\phi(x) - \mathbb{E}_{x \sim \mathbb{Q}_X} f_\phi(x). \tag{5.8}$$

If there exists k such that $\Omega_f \subseteq \mathcal{L}_k$, then $W_L(\mathbb{P}_X, \mathbb{Q}_X)/k \leq w(\mathbb{P}_X, \mathbb{Q}_X)\ \forall \mathbb{P}_X, \mathbb{Q}_X$ is a lower bound. To enforce $\Omega_\phi \subseteq \mathcal{L}_k$, Arjovsky et al. (2017) propose a weight clipping constraint Ω_c, which constrains every weight to be in $[-c, c]$ for some c.

5.2 Experiments

In this section we demonstrate the point cloud generation capabilities of PC-GAN. As discussed in Section 5.1, we refer Achlioptas et al. (2018) as AAE-Fix as it could be treated as an AAE extension with fix number of output points. The sandwiching objective W_s for PC-GAN combines W_L and W_U with the mixture 1:20 without tunning for all experiments. W_L is a GAN loss by combining Arjovsky et al. (2017) and Mroueh and Sercu (2017) and we adopt Bertsekas (1985) for W_U. We parametrize q in PC-GAN by Zaheer et al. (2017a).

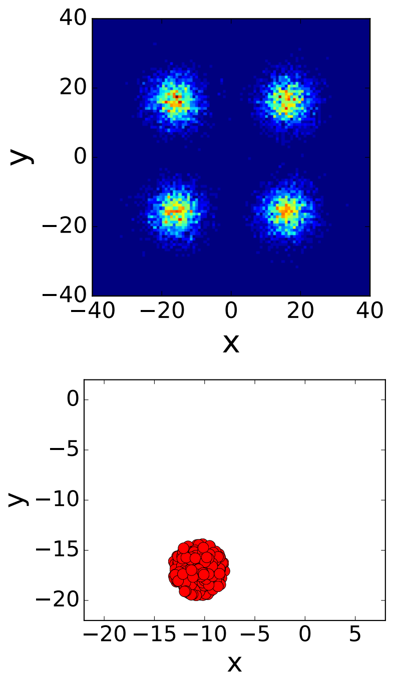
5.2.1 Synthetic Datasets

We generate 2D circle point clouds. The center of circles follows a mixture of Gaussians $\mathcal{N}(\{\pm 16\} \times \{\pm 16\}, 16I)$ with equal mixture weights. The radius of the circles was drawn from a uniform distribution *Unif*$(1.6, 6.4)$. One sampled circle is shown in Figure 5.2a.

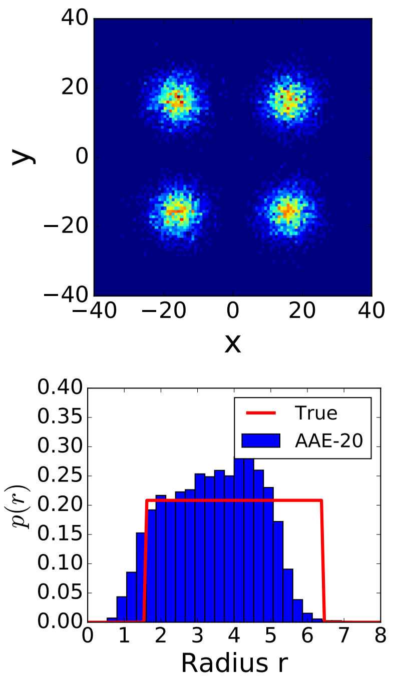
The output size of the AAE-Fix decoder is 500×2 for 500 points, and the output size of its encoder (latent code) is 20. The total number of parameters are $24K$. For PC-GAN, the inference network (encoder) output size is 15. The total nuumber of parameters of PC-GAN is only $12K$. We evaluated the conditional distributions on the $10,000$ testing circles. We measured the empirical distributions of the centers and the radius of the reconstructed circles of the testing data as shown in Figure 5.2. From Figure 5.2, both AAE-Fix and PC-GAN can successfully recover the center distribution, but AAE-Fix does not learn the radius distribution well even with larger latent code (20) and more parameters ($24K$). The gap of memory usage could be larger if we configure AAE-Fix to generate more points, while the model size required for PC-GAN is independent of the number of points. The reason is an MLP decoder adopted by Achlioptas et al. (2018) wastes parameters for nearby points.

5.2.2 Study on ModelNet40

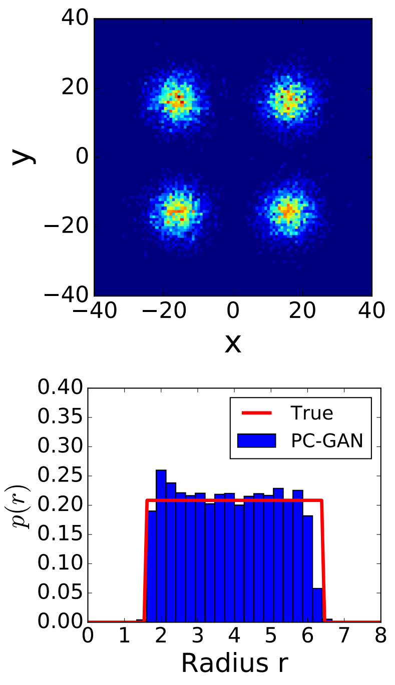
We consider ModelNet40 (Wu et al., 2015) benchmark, which contains 40 classes of objects. There are $9,843$ training and $2,468$ testing instances. We follow Achlioptas et al. (2018) to consider two settings. One is training on single class of objects. The other is training on all $9,843$ objects in the training set. Achlioptas et al. (2018) set the latent code size of



(a) True Data



(b) AAE-Fix



(c) PC-GAN

Figure 5.2: (a) (top) the true center distribution and (bottom) one example of a circle point cloud. (b-d) are the reconstructed center and radius distributions.

AAE-Fix to be 128 and 256 for these two settings, with the total number of parameters to be $15M$ and $15.2M$, respectively. Similarly, we set the output dimension of q in PC-GAN to be 128 and 256 for single-class and all-classes. The total number of parameters are $1M$ and $3M$, respectively.

Metrics for Quantitative Comparison Firstly, we are interested in whether the learned g and q can model the distribution of unseen test data. For each test point cloud, we infer the latent variable $q(X)$, then use g to generate points. We then compare the distribution between the input point cloud and the conditionally generated point clouds.

There are many finite sample estimators for f-divergence and IPM can be used for evaluation. However, those estimators with finite samples are either biased or with high variance (Peyré et al., 2019; Póczos et al., 2012; Wang et al., 2009; Weed and Bach, 2017). Also, it is usually computationally infeasible 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 et al., 2012; Wang et al., 2009). Therefore, we propose to measure the performance with the following criteria. Given a point cloud $\{x_i\}_{i=1}^n$ and

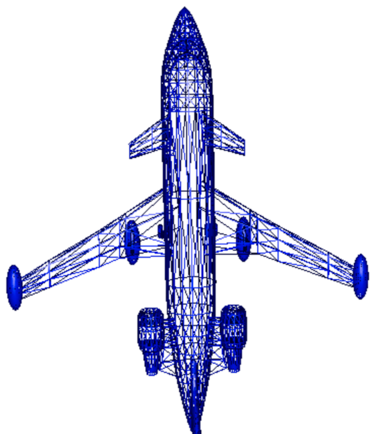


Table 5.1: Quantitative results of different models trained on different subsets of ModelNet40 and evaluated on the corresponding test set. ModelNet10 is a subset containing 10 classes of objects, while ModelNet40 is a full training set. AAE-Fix is trained using the code from Achlioptas et al. (2018). The PC-GAN variants are trained via upper bound W_U, lower bound W_L and sandwiching loss W_s.

Data	Distance to Face (D2F ↓)				Coverage (↑)			
	PC-GAN (W_s)	AAE-Fix	PC-GAN (W_U)	PC-GAN (W_L)	PC-GAN (W_s)	AAE-Fix	PC-GAN (W_U)	PC-GAN (W_L)
Aeroplanes	1.89E+01	1.99E+01	1.53E+01	2.49E+01	1.95E-01	2.99E-02	1.73E-01	1.88E-01
Benches	1.09E+01	1.41E+01	1.05E+01	2.46E+01	4.44E-01	2.35E-01	2.58E-01	3.83E-01
Cars	4.39E+01	6.23E+01	4.25E+01	6.68E+01	2.35E-01	4.98E-02	1.78E-01	2.35E-01
Chairs	1.01E+01	1.08E+01	1.06E+01	1.08E+01	3.90E-01	1.82E-01	3.57E-01	3.95E-01
Cups	1.44E+03	1.79E+03	1.28E+03	3.01E+03	6.31E-01	3.31E-01	4.32E-01	5.68E-01
Guitars	2.16E+02	1.93E+02	1.97E+02	1.81E+02	2.25E-01	7.98E-02	2.11E-01	2.27E-01
Lamps	1.47E+03	1.60E+03	1.64E+03	2.77E+03	3.89E-01	2.33E-01	3.79E-01	3.66E-01
Laptops	2.43E+00	3.73E+00	2.65E+00	2.58E+00	4.31E-01	2.56E-01	3.93E-01	4.55E-01
Sofa	1.71E+01	1.64E+01	1.45E+01	2.76E+01	3.65E-01	1.62E-01	2.94E-01	3.47E-01
Tables	2.79E+00	2.96E+00	2.44E+00	3.69E+00	3.82E-01	2.59E-01	3.20E-01	3.53E-01
ModelNet10	5.77E+00	6.89E+00	6.03E+00	9.19E+00	3.47E-01	1.90E-01	3.36E-01	3.67E-01
ModelNet40	4.84E+01	5.86E+01	5.24E+01	7.96E+01	3.80E-01	1.85E-01	3.65E-01	3.71E-01

a mesh, which is a collection of faces $\{F_j\}_{j=1}^m$, we measure the *distance to face (D2F)* as

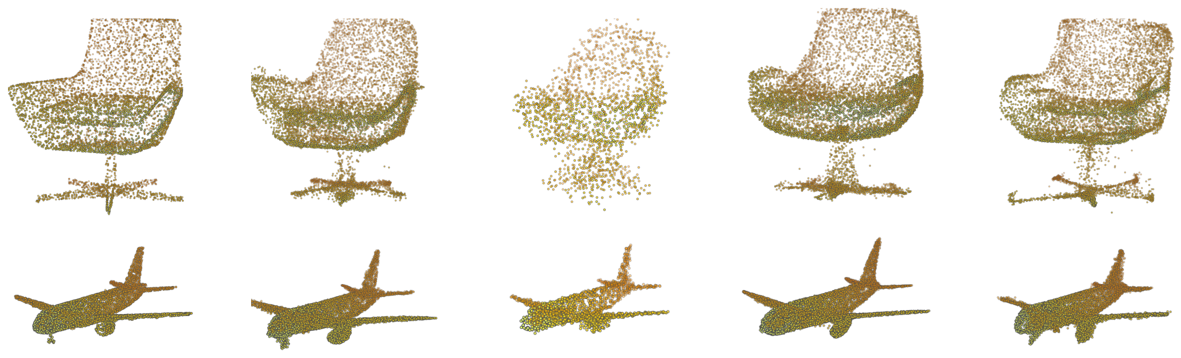
$$D2F\left(\{x_i\}_{i=1}^n, \{F_j\}_{j=1}^m\right) = \frac{1}{n}\sum_{i=1}^n \min_j \mathcal{D}(x_i, F_j),$$

where $\mathcal{D}(x_i, F_j)$ 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., 2018; Fan et al., 2017), with infinitely samples from true distributions (meshes).

Nevertheless, the algorithm can have low or zero D2F by only focusing a small portion of the point clouds (mode collapse). Therefore, we are also interested in whether the generated points recover enough supports of the distribution. We compute the *Coverage* ratio as follows. For each point, we find the its nearest face, we then treat this face is covered[1]. We then compute the ratio of number of faces of a mesh is covered. A sampled mesh is showed in Figure 5.3, where the details have more faces (non-uniform). Thus, it is difficult to get high coverage for AAE-Fix or PC-GAN trained by limited number of sampled points. However, the coverage ratio, on the other hand, serve as an indicator about how much details the model recovers. The idea of the proposed metrics, D2F and Convergae, is simultaneously proposed by Lucic et al. (2018) for general IGM performance evaluation, while they call these as *precision* and *recall*, respectively.

The results are reported in Table 5.1. We compare four different algorithm, AAE-Fix

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



(a) Data (b) PC-GAN (W_s) (c) AAE-Fix (d) PC-GAN (W_U) (e) PC-GAN (W_L)

Figure 5.4: Example reconstruction (conditional generation) on test objects. PC-GAN with sandwiching (W_s) is better in capturing fine details like wheels of aeroplane or proper chair legs.

and PC-GAN with three objectives, including upper bound W_U (ϵ approximated Wasserstein distance), lower bound W_L (GAN with L^2 ball constraints and weight clipping), and the sandwiching loss W_s as discussed in Section 5.1.3, The study with W_U and W_L also serves as the ablation test of W_s.

Comparison between Upper bound, Lower bound and Sandwiching Since W_U directly optimizes point-wise distance between training and generated point clouds, W_U usually results in smaller D2F than W_L in Table 5.1. Although W_L only recovers lower bound estimates of Wasserstein distance, its discriminator is known to focus on learning support of the distribution (Bengio, 2018), which results in better coverage (support) than W_U.

Theoretically, the proposed sandwiching W_s results in a tighter Wasserstein distance estimation than W_U and W_L (Lemma 20). Based on above discussion, it can also be understood as balancing both D2F and coverage by combining both W_U and W_L to get a desirable middle ground. Empirically, we even observe that W_s results in better coverage than W_L, and competitive D2F with W_U. The intuition is that some discriminative tasks are *off* to W_U objective, so the discriminator can focus more on learning distribution supports. We argue that this difference is crucial for capturing the object details. Some reconstructed point clouds of testing data are shown in Figure 5.4. For aeroplane examples, W_U are failed to capture aeroplane tires while W_s has better tire than W_L. For Chair example, W_s recovers better legs than W_U and better seat cushion than W_L. Lastly, we highlight W_s outperforms others more significantly when training data is larger (ModelNet10 and ModelNet40) in Table 5.1.

Comparison between PC-GAN and AAE-Fix In most cases, PC-GAN with W_s has lower D2F in Table 5.1 with less number of parameters aforementioned. Similar to the argument in Section 5.2.1, although AAE-Fix use larger networks, its decoder wastes parameters for nearby points. AAE-Fix only outperforms PC-GAN (W_s) in Guitar and

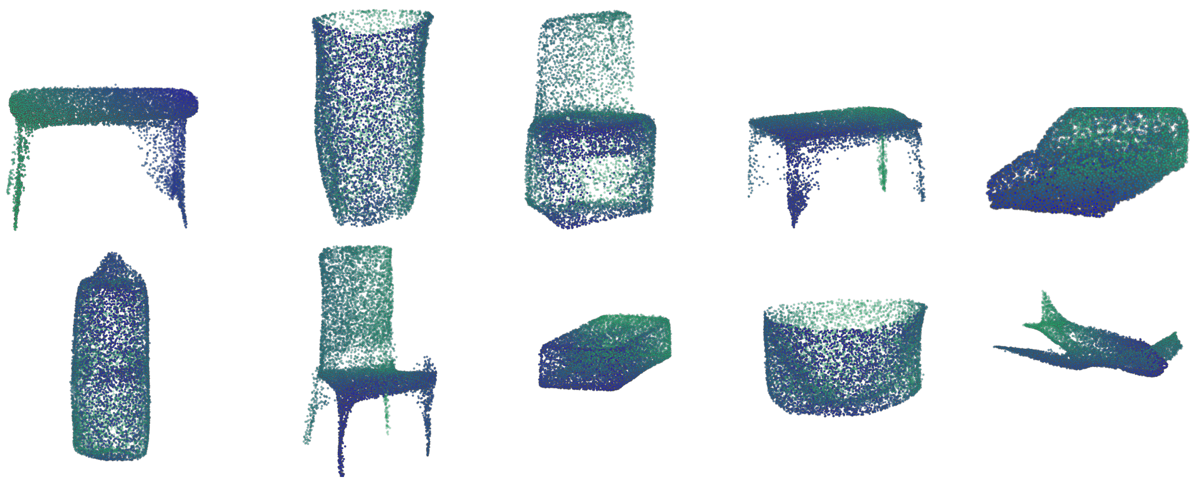


Figure 5.5: Randomly sampled objects and corresponding point cloud from the hierarchical sampling Even if there are some defects, the objects are smooth, symmetric and structured.

Sofa in terms of D2F, since the variety of these two classes are low. It is easier for MLP to learn the shared template (basis) of the point clouds. On the other hand, due to the limitation of the fixed number of output points and Chamfer distance objective, AAE-Fix has worse coverage than PC-GAN as in Figure 5.4, where AAE-Fix is also failed to recover aeroplane tire.

Hierarchical Sampling In Section 5.1, we propose a hierarchical sampling process for sampling point clouds. The randomly sampled results without given any data as input are shown in Figure 5.5. 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.

Interpolation of Learned Manifold We study whether the interpolation between two objects on the latent space results in smooth change. We interpolate the inferred representations of two objects by q, and use the generator g to sample points based on the interpolation. The inter-class result is shown in Figure 5.6.

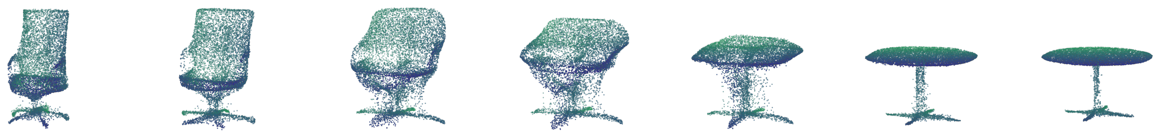
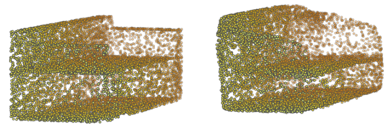
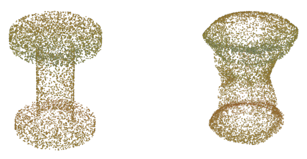
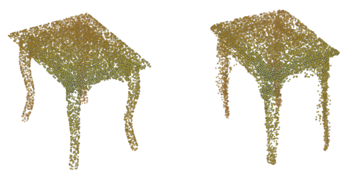


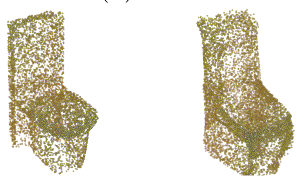
Figure 5.6: Interpolating between latent representations $q(X)$ of a table and a chair point clouds.

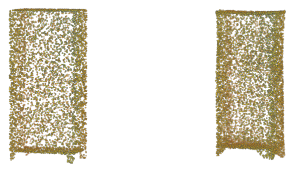
Generalization on Unseen Classes In above, we studied the reconstruction of unseen testing objects, while PC-GAN still saw the point clouds from the same class during training. *Here we study the more challenging task.* We train PC-GAN on first 30 (Alphabetic order) class, and test on the other *fully unseen 10 classes.* Some reconstructed (conditionally generated) point clouds are shown in Figure 5.7. For the object from the unseen classes, the conditionally generated point clouds still recovers main shape and reasonable geometry structure, which confirms the advantage of the proposed PC-GAN: by enforcing the point-wise transformation, the model is forced to learn the underlying geometry structure and the shared building blocks, instead of naively copying the input from the conditioning. The rsulted D2F and coverage are 57.4 and 0.36, which are only slightly worse than 48.4 and 0.38 by training on whole 40 classes in Table 5.1 (ModelNet40), which supports the claims of the good generalization ability of PC-GAN.


(a) Sofa


(b) Stool


(c) Table


(d) Toilet


(e) TV Stand

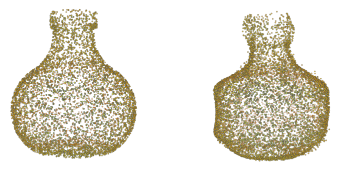

(f) Vase

Figure 5.7: The reconstructed objects from unseen classes (even in training). In each plot, LHS is true data while RHS is PC-GAN. PC-GAN generalizes well as it can match patterns and symmetries from classes seen in the past to new unseen classes.

5.2.3 Classification Results

We evaluate the quality of the representations from the learned inference network q. We train the inference network q and the generator g on the training split of ModelNet40 with data augmentation as mentioned above for learning generative models without label information. We then extract the latent representation $q(X)$ for each point clouds and train linear SVM on the that with its label by following Achlioptas et al. (2018).

We only sample 1000 as input for our inference network q. Benefited by the Deep Sets architecture for the inference network, which is invariant to number of points. Therefore, we are allowed to sample different number of 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 5.2. By using 1000 points, we are already better than Achlioptas et al. (2018) with 2048 points, and competitive with the supervised learning algorithm Deep Sets.

Method	# points	Accuracy
PC-GAN	1000	$87.5 \pm .6\%$
PC-GAN	2048	$87.8 \pm .2\%$
AAE-Fix (Achlioptas et al., 2018)	2048	$85.5 \pm .3\%$
Deep Sets (Zaheer et al., 2017a)	1000	$87 \pm 1\%$
Deep Sets (Zaheer et al., 2017a)	5000	$90 \pm .3\%$

Table 5.2: Classification accuracy results.

5.3 Summary

In this chapter, we propose PC-GAN for learning to generate point clouds. The design is inspired by standard hierarchical samplings with latent variables in graphical models. Previously, every node is modeled by parametric distributions (explicit models) (Koller et al., 2009). We, instead, show how to incorporate IGMs into those frameworks to directly model samplings of nodes with a learned inference. In the next chapter, we will extend this framework further by showing how to encode priors into the model design to generate more specific and highly-structured point clouds.

Chapter 6

Generate Structured Point Clouds with Human Priors

In Chapter 5, we propose an auto-encoding framework, PC-GAN, for learning to generate point clouds with an IGM autoencoder. Although we demonstrate successful examples on toy point clouds (e.g. aeroplanes, chairs, tables), it is challenging to generate highly-structured point clouds, such as human body scans. PC-GAN uses an IGM decoder to transform a zero-prior (e.g. Gaussian (Li et al., 2018) or uniform distributions (Groueix et al., 2018b; Yang et al., 2018)) into the target point clouds. However, we could leverage our prior of target point clouds to replace the object-agnostic zero priors. For example, we could use a uniform distribution over a body template, which defines the geometry of the shape, as the input randomness of the IGM decoder for body point clouds (Groueix et al., 2018a). In computer graphics, many advanced *models* have been extensively studied (e.g. Anguelov et al. (2005); Bogo et al. (2016)), which encode strong human knowledges of different 3D data. In this chapter, we study how to incorporate an expressive yet simple graphics model into IGM designs, which serve as human priors, to learn to generate point clouds of *rigged* objects. We use real body scan point clouds as examples, and achieve state-of-the-art performance on standard benchmarks.

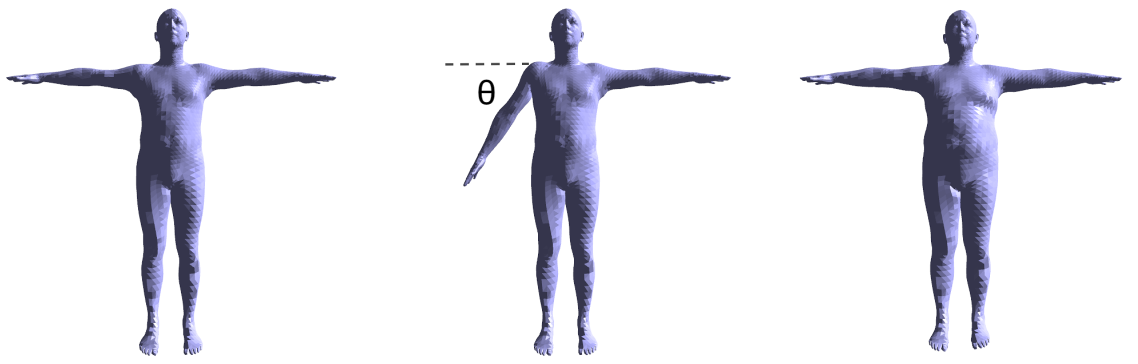
Model Fitting and Registration. Using a graphics model as prior into IGM decoder to reconstruct input point clouds can also be treated as controlling graphics model to *register* (fit) input clouds, which has been extensively studied in past decades in computer vision and computer graphics. Works in this area can be coarsely grouped together based on how much prior knowledge and supervision is incorporated into the fitting method. On one end of the spectrum, there are entirely unsupervised and object-agnostic models, such as Groueix et al. (2018b); Yang et al. (2018) and PC-GAN in Chapter 5. These methods learn to deform a zero-prior to match the target geometry, while making no assumptions about the objects. Adding slightly more prior knowledge, 3D-CODED (Groueix et al., 2018a) uses a template mesh (e.g. hand or body) with a topology better suited to the object of interest. On the other end of the spectrum are highly specialized models for specific objects, such as hands and bodies, including SCAPE (Anguelov et al., 2005), Dyna (Pons-Moll et al., 2015), SMPL (Loper et al., 2015), and MANO (Romero et al., 2017). These models are built using high-resolution 3D scans with correspondence and human curation. They model correctives for different poses and modalities (e.g. body types) and can be used as high-quality generative models of geometry. A number of works learn to manipulate these models to fit data based on different sources of supervision, such as key points (Bogo et al., 2016; Joo et al., 2018; Lassner et al., 2017; Mehta et al., 2017; Tung et al., 2017) and/or prior distributions of model parameters (Kanazawa et al., 2018a,b).

In this chapter, we present an unsupervised/self-supervised algorithm, LBS Autoencoder (LBS-AE), to fit such articulated mesh models to point cloud data, which improves the core auto-encoding components of PC-GAN in Chapter 5. The proposed algorithm is a middle ground of the two ends of spectrum discussed above in two senses.

First, we assume an articulated template model of the object class is available as a prior, but not the statistics of its articulation of data nor the specific shape of the object instance. We argue that this prior information is widely available for many common objects of interest in the form of "rigged" or "skinned" mesh models, which are typically created by artists for use in animation. In addition to a template mesh describing the geometric shape, these prior models have two more components: (1) a kinematic hierarchy of transforms describing the degrees of freedom, and (2) a skinning function that defines how transforms in the hierarchy influence each of the mesh vertices. This enables registration to data by manipulating the transforms in the model. One common example is Linear Blending Skinning (LBS). Therefore, instead of relying on deep networks to learn the full deformation process from a single template (Groueix et al., 2018a), we leverage LBS as the prior of IGM decoders to model coarse joint deformations. Different from hand-crafted models such as SMPL (Loper et al., 2015), LBS by itself does not model pose-dependent correctives between the template and data, nor does it model the space of non-articulated shape variation (e.g. body shape). Instead, we use the transformations learned via IGM decoders to model those correctives.

Second, for fitting models to data during the training, existing works either rely on explicit supervision (e.g. correspondence (Groueix et al., 2018a) and key points (Joo et al., 2018)) or unsupervised nearest neighbors search (e.g. Yang et al. (2018)) to find point correspondence between the model and data for measuring reconstruction loss. A detailed discussion of point clouds matching loss can be found in Chapter 5. Rather than using external supervision, we introduce a "Structured Chamfer Distance" (SCD), which is a probability pseudometric. It improves the blind nearest neighbor based Chamfer distance by inferring coarse correspondences before find matchings. The proposed SCD can also be treated as dividing distribution matching into couples of conditional distribution matchings which conditions on meaningful geometry segmentations. The challenge is we do not assume external supervision to be available for the input point clouds. Instead, we utilize the learned LBS-AE to generate *self-supervision* to learn the segmentation inference from scratch simultaneously under a joint training framework.

In this work, we show that the prior from artist-defined rigs may already be sufficiently constrained and informative to learn effective transformations to generate realistic and highly-structured data without any additional labeling. In addition to learning powerful generative models, such a model-fitting pipeline without additional supervision has the potential to simplify geometric registration tasks by requiring less human labeling effort.



(a) Template mesh U.

(b) LBS deformation $M(\theta, U)$ of the template using joint angles θ.

(c) A deformed template U^d.

Figure 6.1: The example of LBS and template deformations.

6.1 LBS Autoencoder for Point Clouds

We propose to learn an autoencoder that takes as input an unstructured point cloud $X=\{x\}_{i=1}^n$, where each x_i is a 3D point and n is a variable number, and produces as output a fixed number m of corresponded vertices $V=\{v_i\}_{i=1}^m$. The vertices V form a mesh with fixed topology whose geometry should closely match that of the input. We note that although we assume the inputs are point clouds, they could also be the vertices of a mesh without using any topology information. Rather than allowing the autoencoder to be any arbitrary deformation produced by a deep neural network (as in Groueix et al. (2018b); Yang et al. (2018)), we force the output to be produced by Linear Blending Skinning (LBS) to explicitly encode the motion of joints. We allow additional non-linear deformations (also given by a neural network) to model deviations from the LBS approximation. However, an important difference with respect to similar models, such as SMPL (Loper et al., 2015) or MANO (Romero et al., 2017), is that we do not pre-learn the space of non-LBS deformations on a curated set (and then fix them) but rather learn these simultaneously on the data that is to be aligned, without additional supervision.

Linear Blending Skinning (LBS) We start by briefly introducing LBS (Magnenat-Thalmann et al., 1988), which is the core building component of the proposed work. LBS models deformation of a mesh from a rest pose as a weighted sum of the skeleton bone transformations applied to each vertex. We follow the notation outlined in Loper et al. (2015), which is a strong influence on our model. An LBS model with J joints can be defined as follows

$$V = M(\theta, U), \tag{6.1}$$

with V the vertices of the deformed shape after LBS. The LBS function M takes two parameters, one is the vertices $U = \{u_i\}_{i=1}^m$ of a base mesh (template), and the other are the relative joint rotation angles $\theta \in \mathbb{R}^{J\times 3}$ for each joint j with respect to its parents. If $\theta = \mathbf{0}$, then $M(\mathbf{0}, U) = U$. Two additional parameters, the skinning weights w and the joint hierarchy K, are required by LBS. We will consider them fixed by the artist-defined rig. In particular, $w \in \mathbb{R}^{m\times J}$ defines the weights of each vertex contributing to joint j and $\sum_j w_{i,j} = 1$ for all i. K is the joint hierarchy. Each vertex $v_i \in V$ can then be written as

$$v_i = (\mathbf{I}_3, \mathbf{0}) \cdot \sum_{j=1}^{J} w_{i,j} \mathcal{T}_j(\theta, K) \begin{pmatrix} u_i \\ 1 \end{pmatrix},$$

where $\mathcal{T}_j(\theta, K) \in \mathrm{SE}(3)$ is a transformation matrix for each joint j, which encodes the transformation from the rest pose to the posed mesh in world coordinate, constructed by traversing the hierarchy K from the root to j. Since each v_i is constructed by a sequence of linear operations, the LBS $M(\theta, U)$ is differentiable respect to θ and U. A simple example constructed from the LBS component in SMPL (Loper et al., 2015) is shown in Figure 6.1a and 6.1b.

In this work, both the joint angles and the template mesh used in the LBS function are produced by deep networks from the input point cloud data,

$$V = M(q^{\theta}(X), g(X, U)), \tag{6.2}$$

where we identify a *joint angle inference* network q^{θ}, and a *template deformation* network g which we describe below.

Joint Angle (Pose) Inference Given an LBS model defined in Eq. (6.1), the goal is to regress joint angles based on input X via a function $f : X \to \theta$ such that $M(q^{\theta}(X), U) \approx X$. We use a deep neural network, which takes set data (e.g. point cloud) as input (Qi et al., 2017a; Zaheer et al., 2017a) to q^{θ}, but we must also specify how to compare X and V from $M(\cdot)$. In Chapter 5, we discussed several probability (pseudo) distances (e.g. optimal transport) to match two point clouds. After finding *matchings* or *correpsondences*, we learn q^{θ} by back-propagating this point-wise loss through the differentiable LBS $V = M(q^{\theta}(X), U)$. Also note that we only sample a subset of points for estimating Eq. (6.3) under SGD training schemes. In the following, we will refer *matching* in optimal transport as *correspondence*, which is commonly used in registration literature.

However, finding optimal correspondences in optimal transport is usually costly. Also, optimal transport is less robust to missing data, which is usally the case for real-world point cloud data. Instead, Chamfer distance (CD) (Yang et al., 2018), which finds the matching grreedily via nearest neighbors, is defined as $\mathcal{L}_c(X, V) =$

$$\frac{1}{n}\sum_{i=1}^{n} \|x_i - \mathcal{N}_V(x_i)\|^2 + \frac{1}{m}\sum_{j=1}^{m} \|v_j - \mathcal{N}_X(v_j)\|^2, \tag{6.3}$$

where $\mathcal{N}_V(x_i) = \arg\min_{v_j \in V} \|x_i - v_j\|$ is the nearest neighbor of x_i in V. This is also called Iterative Closest Point (ICP) in the registration literature (Besl and McKay, 1992).

Although CD is a probability pseudometric, it is fast to compute. Later, we will discuss how to improve CD to resolve drawbacks of its greedy correspondence finding.

In practice, we observe that it takes many iterations for PointNet (Qi et al., 2017a) or DeepSet (Zaheer et al., 2017a) architectures to improve if the target loss is CD instead of corresponded supervision. Similar behaviors were observed in Li et al. (2018); Yang et al. (2018), where the algorithms may take millions of iterations to converge. To alleviate this problem, we utilize LBS to generate data based on a given θ' for *self-supervision* by optimizing

$$\min_f \mathcal{L}_\theta = \|q^\theta(M(\theta', U)) - \theta'\|^2.$$

It is similar to the *loop-back* loss (Genova et al., 2018) that ensures q^θ can correctly reinterpret the model's own output from M. Different from Genova et al. (2018); Kanazawa et al. (2018a), we do not assume a prior pose distribution is available. Our θ' comes from two sources of randomness. One is uniform distributions within the given joint angle ranges (specified by the artist-defined rig) and the second is we uniformly perturb the inferred angles from input samples with a small uniform noise on the fly, which can gradually adapt to the training data distribution when the estimation is improved as training progresses (see Section 6.2 and Figure 6.5).

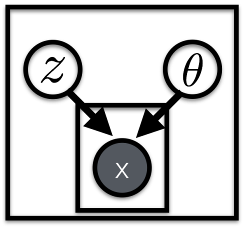
Template Deformation Although LBS can represent large pose deformations, due to limitations of LBS as well as differences between the artist modeled mesh and the real data, there will be a large residual in the fitting. We refer to this residual as a *modality gap* between the model and reality, and alleviate this difference by using neural networks to produce the template mesh to be posed by LBS. The deformation network $g(q^z(X), u_i)$ takes two sources as input, where u_i is each vertex in the template mesh U, and $q^z(X)$ are latent variables, which contains *style* information (e.g. body type) about X. This yields a deformed template $U^d = \{g(q^z(X), u_i)\}_{i=1}^m$. One example is shown in Figure 6.1c. After LBS, we denote the deformed and posed mesh as $V^d = M(q^\theta(X), U^d)$, and denote by $V = M(q^\theta(X), U)$ the posed original template.

If q^z is high-capacity, $q^\theta(X)$ can learn to generate all-zero joint angles for the LBS component (ignoring the input X), and explain all deformations instead with q^z. That is, $M(q^\theta(X), U^d) = M(\mathbf{0}, U^d) = U^d \approx X$, which reduces to the unsupervised version of Groueix et al. (2018a). Instead of using explicit regularization to constrain q^z (e.g. $\|g(q^z(X), U^B)\|$), we propose a composition of two Chamfer distances as

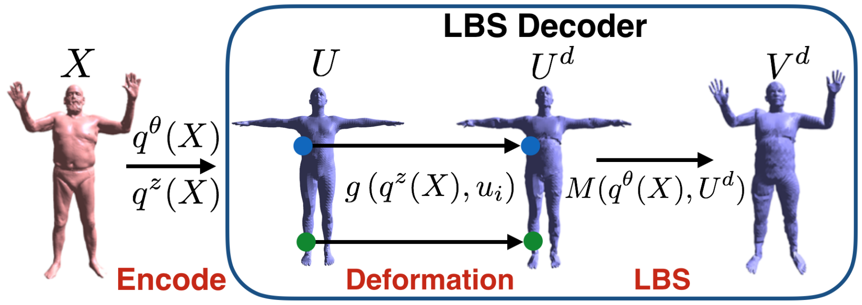
$$\mathcal{L}_{c^2,\lambda} = \mathcal{L}_c\left(X, V^d\right) + \lambda \mathcal{L}_c\left(X, V\right). \tag{6.4}$$

The second term in Eq. (6.4) enforces $q^\theta(X)$ to learn correct joint angles even without template deformation.

Lastly, we follow Groueix et al. (2018a); Kanazawa et al. (2018b) and apply Laplacian regularization $\mathcal{L}_{\texttt{lap}} = \|LV^d\|$ to encourage smoothness of the deformed template, where L is the discrete Laplace-Beltrami operator constructed from mesh U and its faces.



(a) The graphical diagram of LBS-AE, where z is the style latent code and θ is unobserved joint angles.



(b) Given a point cloud X of a input shape, we encode X into a latent code $q^z(X)$ and the inferred joint angles $q^\theta(X)$. The decoder contains a deformation network g to deform the template U into U^d, then uses a LBS to pose U^d into V^d as the reconstruction.

Figure 6.2: Overview of LBS Autoencoder (LBS-AE).

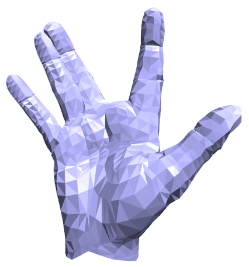
6.1.1 Autoencoding Interpretation

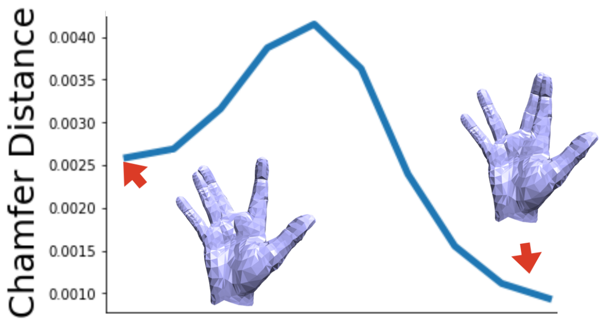
We present the proposed algorithm by following Loper et al. (2015) from a model construction perspective. Here we connect the proposed algorithm with the encoder-decoder scheme under the PC-GAN framework. The inference network $q(X)$ consists of two components, including $q^\theta(X)$ for interpretable joint angle inference and $q^x(X)$ for style code inference. The IGM decoder, different from using zero-prior as presented in Chapter 5, is a pipeline constructed by combining a human designed LBS function and a style deformation (transformation) network g. The transformation network g only models the transformation of styles (correctives) by a neural network, while the initial randomness $\mathbb{P}_u$ is an uniform distribution over the template U. The extended graphical diagram is shown in Figure 6.2a. In Chapter 5, we only have one latent variable z, which contains the identity information of the object. In Figure 6.2a, on the other hand, z is the *style* latent variable inferred by $q^z(X)$. We introduce one more interpretable latent variable θ, which is required by LBS. We call the proposed algorithm LBS-AE as shown in Figure 6.2b.

6.2 Structured Chamfer Distance

To train an autoencoder, we have to define proper reconstruction errors match input and reconstructed point clouds (distributions). In LBS-AE, the objective that provides information about input point clouds can be any distribution matching distance such as CD (6.3). In Chapter 5, we study different probability distance which outperforms CD, but CD enjoys the advantage of being easily and fast computed. In this section, we have a deeper investigation of undesirable local optima, which hinders the algorithm from improving. We then propose an improved modification, Structured Chamfer Distance (SCD), which leverage the prior from LBS model to have a stronger discriminative power but it is still efficient in computation.

(a) Input


(b) Estimate



(c) The Chamfer distance between the input and the target when we move the middle finger.

Figure 6.3: When we try to move the middle finger of the current estimate (b) toward the target (a), the Chamfer distance increases before decreasing, showing a local optimum that is difficult to overcome.

A local optimum example of CD is shown in Figure 6.3. To move the middle finger from the current estimate towards the index finger to fit the input, the Chamfer distance must increase before decreasing. This local optimum is caused by incorrect correspondences found by nearest neighbor search (the nearest neighbor of the middle finger of the current estimate is the ring finger of the input).

6.2.1 High-Level Correspondence

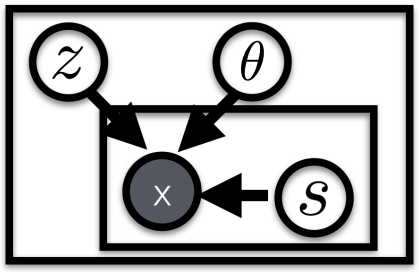
Given a pair of sets (V, X), for each $v \in V$, we want to find its correspondence $\mathcal{C}_X(v)$ in X. In CD, we use the nearest neighbor $\mathcal{N}_X(v)$ to approximate $\mathcal{C}_X(v)$, which can be wrong, as shown in Figure 6.3. Instead of searching for nearest neighbors $\mathcal{N}_X(v)$ over the entire set X, we propose to search within a subset $X' \subset X$, where $\mathcal{C}_X(v) \in X'$, by eliminating irrelevant points in X. Following this idea, we partition X into k subsets, $X^1 \dots X^k$, where we use $s(x; X) \in \{1, \dots, k\}$ to denote which subset x belongs to. A desirable partition should ensure $s(v; V) = s(\mathcal{C}_X(v); X)$; then, to find the nearest neighbor of v, we need only consider $X^{s(v)} \subset X$. We then define the *Structured Chamfer Distance* (SCD) as $\mathcal{L}_s(X, V) =$

$$\frac{1}{n}\sum_{i=1}^{n} \|x_i - \mathcal{N}_{V^{s(x_i)}}(x_i)\|^2 + \frac{1}{m}\sum_{j=1}^{m} \|v_j - \mathcal{N}_{X^{s(v_j)}}(v_j)\|^2, \tag{6.5}$$

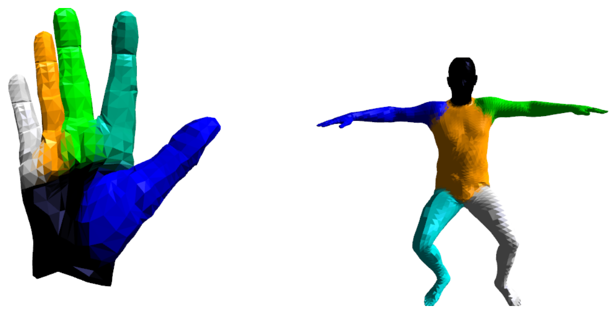
where we ease the notation of $s(x, X)$ and $s(v, V)$ to be $s(x)$ and $s(v)$. Compared with CD, which finds nearest neighbors from *all to all*, SCD uses *region to region* based on the high-level correspondence by leveraging the structure of data. Similar to Eq. (6.4), we define

$$\mathcal{L}_{s^2,\lambda} = \mathcal{L}_s\left(X, V^d\right) + \lambda\mathcal{L}_s\left(X, V\right). \tag{6.6}$$

We propose to partition the vertices based on the underlying geometry which is by LBS



(a) Extended graphical diagram of LBS-AE with the partition latent variable for each point.



(b) Example partitions of human hand and body shapes defined by joints.

Figure 6.4: LBE-AE with partition inference.

skinning weights at a chosen granularity. Example partitions of hand and body data are shown in Figure 6.4b, which use the structure and our prior knowledge of the human body. These satisfy the property that the true correspondence is within the same partition. If we are able infer the partition properly, SCD provides better discriminative power to resolve the local optimum in Figure 6.3. Finally, SCD can also be treated as $\mathbb{E}_s[D(\mathbb{P}_{X|s}\|\mathbb{Q}_{X|s})]$, which divides $D(\mathbb{P}_X\|\mathbb{Q}_X)$ into many probability distances between conditional distributions given each partition s.

6.2.2 Segmentation Inference

We extend the model as shown in Figure 6.4a by introducing one more latent variable s, which denotes the partition of each point x. For the deformed mesh V, we can easily infer the partition $s(v;V)$, because the mapping between vertices and joints is defined by the LBS skinning weights w. We directly use $\text{argmax}_j\, w_{i,j}$ as labels. Without additional labeling or keypoint information, the difficulty is to infer $s(x;X)$ for $x \in X$, which is a point cloud segmentation task (Qi et al., 2017a). However, without labels for X, we are not able to train a segmentation inference on X directly. Instead, similar to the self-supervision technique used for training the joint angle regressor, we propose to train a segmentation network s with the data (V^d, Y) generated by LBS, where Y are the labels for w defined in LBS and $V = M(\theta, U^d)$. Note that θ follows the same distribution as before, which contains uniform sampling for exploration and perturbation of the inferred angles $q^\theta(X)$, as shown in Figure 6.5. Instead of using the base template U only, we use the inferred deformed template U^d to adapt to the real data modality, which improves performance (see Section 6.3.1).

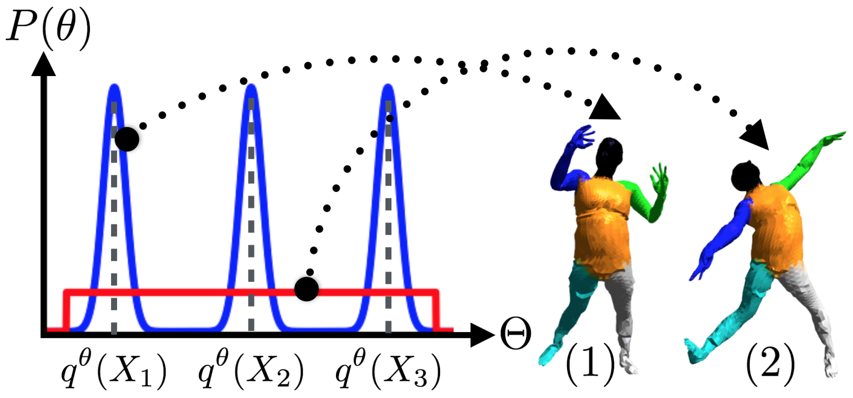


Figure 6.5: The mixture distribution of self-supervision data from the LBS at iteration t. We sample from (1) the perturbed distribution centered at the $q^\theta(X_i)$ and (2) a uniform distribution.

The final objective for training the shape deformation pipeline including q^θ and q^z is[1]

$$\mathcal{L} = \mathcal{L}_{c^2,0.5} + \lambda_s \mathcal{L}_{s^2,0.5} + \lambda_{\texttt{lap}} \mathcal{L}_{\texttt{lap}} + \lambda_\theta \mathcal{L}_\theta, \tag{6.7}$$

and we use standard cross-entropy for training s. In practice, since s is noisy in the first iterations, we pretrain it for $50K$ iterations with poses from uniform distributions over joint angles. Note that, for pretraining, we can only use the base template U to synthesize data. After that, we then learn every component jointly by updating each network alternatively. The final algorithm, LBS-AE with SCD as reconstruction loss, is shown in Algorithm 3.

Algorithm 3 LBS-AE with SCD

```
Inputs: • Point Clouds: $\{X\}$
        • LBS: $M(;w, K, U)$ and angle ranges $(R_l, R_u)$
Pretrain $s$ on uniformly sampled poses from LBS
while $q^\theta$, $q^z$, and $g$ have not converged:
  1. Sample minibatch $\{X_i\}_{i=1}^B$, $\{X_i'\}_{i=1}^B$
  2. $\theta' = \{q^\theta(X_i) + \epsilon_i\}_{i=1}^B \cup \theta_r \sim$ Unif$(R_l, R_u)$
  3. Generate $(V^d, Y)$ based on $\theta'$ to update $s$
  4. Infer segmentation labels $\{s(X_i')\}_{i=1}^B$
  5. Update $q^\theta$, $q^z$ and $g$ based on Eq. (6.7)
```

6.2.3 IGM Losses with Auxiliary Neural Networks

Using auxiliary neural networks to define objectives for training targeted models is also broadly studied in IGM literature (e.g. Arjovsky et al. (2017); Goodfellow et al. (2014); Gulrajani et al. (2017); Li et al. (2017); Mao et al. (2017); Mroueh and Sercu (2017);

[1]We use $\lambda = 0.5$, $\lambda_{\texttt{lap}} = 0.005$, $\lambda_\theta = 0.5$ in all experiments.

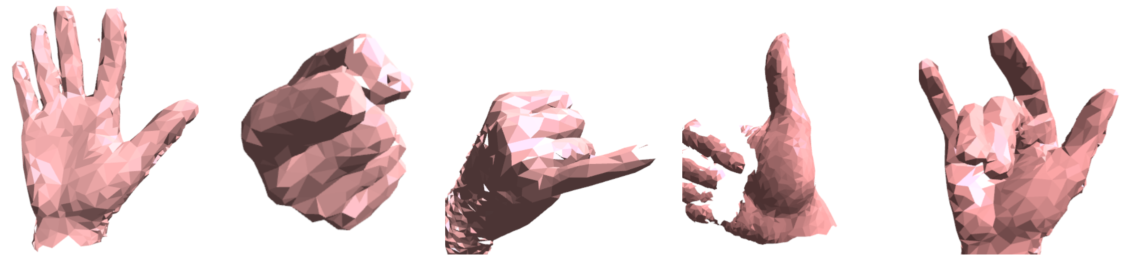


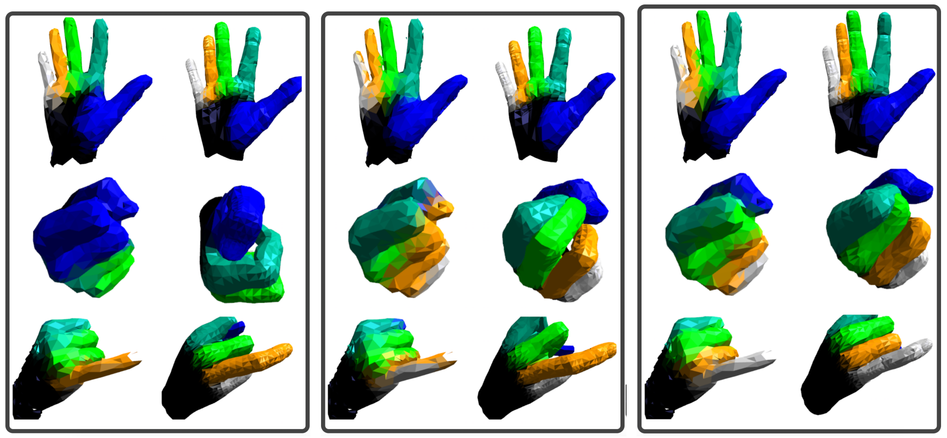
Figure 6.6: Examples of the captured hands.

Nowozin et al. (2016)) as we have discussed extensively in previous chapters. In most of the existing literature, they approximate a probability distance or divergence via neural networks with theoretical guarantees. Although the proposed SCD is a pseudometric, by leveraging prior knowledge, the auxiliary inference network adopted by SCD is an interpretable segmentation network which can be trained without adversarial training and results in superior performance.

6.3 Experiment

Datasets We consider hand and body data. For body data, we test on the FAUST benchmark (Bogo et al., 2014), which captures real human body with correspondence labeling. For hand data, we use a multi-view capture system to captured $1,524$ poses from three people, which have missing area and different densities of points across areas. The examples of reconstructed meshes are shown in Figure 6.6. For numerical evaluation, in addition to FAUST, we also consider synthetic data since we do not have labeling information on the hand data (e.g. key points, poses, correspondence). To generate synthetic hands, we first estimate pose parameters of the captured data under LBS. To model the modality gap, we prepare different base templates with various thickness and length of palms and fingers. We then generate data with LBS based on those templates and the inferred pose parameters. We also generate synthetic human body shapes using SMPL (Bogo et al., 2016). We sample $20,000$ parameter configurations estimated by SURREAL (Varol et al., 2017) and $3,000$ samples of bent shapes from Groueix et al. (2018a). For both synthetic hand and body data, the scale of each shape is in $[-1,1]^3$ and we generate $2,300$ and 300 examples as holdout testing sets.

Architectures The architecture of q^θ follows Chapter 5 to use Zaheer et al. (2017a), which shows competitive performance with PointNet (Qi et al., 2017a) with half the number of parameters. The output is set to be $J \times 3$ dimensions, where J is the number of joints. We use the previous layer's activations as $q^z(X)$ for g. We use a three layer MLP to model g, where the input is the concatenation of v, $q^\theta(X)$ and $q^z(X)$, and the hidden layer sizes are 256 and 128. For the segmentation inference network s, we use Qi et al. (2017a) because of better performance. For hand data, we use an artist-created LBS, while we use the LBS part from SMPL (Loper et al., 2015) for body data.



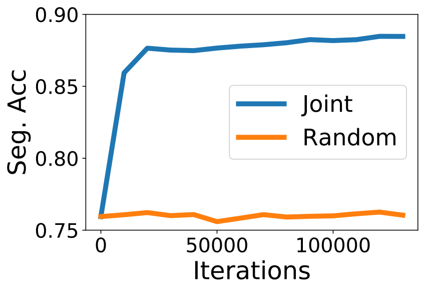
(a) LBS-AE$_{\mathrm{CD}}$ (b) Modality Gap (c) LBS-AE

Figure 6.7: Ablation study of the proposed LBS-AE. For each block, the left column is the inferred segmentations of input shapes while the right column is the reconstruction.

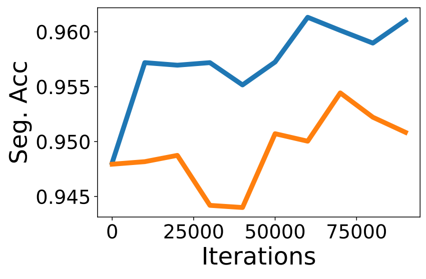
6.3.1 Study on Segmentation Inference

One goal of the proposed LBS-AE is to leverage geometry structures of the shape, by learning segmentation inference jointly to improve correspondence finding via nearest neighbor searching when measuring the distance between two distributions of point clouds (shapes). Different from previous works (e.g. Wei et al. (2016)), we do not rely on any human labels. We study how the segmentation inference with self-supervision interacts with the model fitting to data. We train different variants of LBS-AE to fit the captured hands data. The first is learning LBS-AE with CD only (LBS-AE$_{\mathrm{CD}}$). The objective is Eq. (6.7) without $\mathcal{L}_{s^2,0.5}$. We then train the segmentation network s for SCD with hand poses sampled from uniform distributions based on U instead of U^d. Note that there is no interaction between learning s and the other networks q^θ and g. The segmentation and reconstructed results are shown in Figure 6.7a. We observe that the segmentation network trained on randomly sampled poses from a uniform distribution can only segment easy poses correctly and fail on challenging cases, such as feast poses, because of the difference between true pose distributions and the uniform distribution used as well as the modality gaps between real hands and synthetic hands from LBS. On the other hand, LBS-AE$_{\mathrm{CD}}$ is stuck at different local optimums. For example, it recovers to stretch the ring finger instead of the little finger for the third pose.

Secondly, we study the importance of adapting to different modalities. In Figure 6.7b, we train segmentation and LBS fitting jointly with SCD. However, when we augment the data for training segmentation, we only adapt to pose distributions via $q^\theta(X)$, instead of using the deformed U^d. Therefore, the training data for s for this case has a modality gap between it and the true data. Compared with Figure 6.7a, the joint training benefits the performance, for example, on the feast pose. It suggests how good segmentation learning benefits reconstruction. Nevertheless, it still fails on the third pose. By training LBS-AE and the segmentation inference jointly with inferred modalities and poses, we could fit the poses better as shown in Figure 6.7c. This difference demonstrates the importance of training segmentation inference adapting to the pose distributions and different modalities.



(a) Synthetic Hands



(b) SMPL

Figure 6.8: Segmentation accuracy on the holdout testing sets.

Numerical Results We also quantitatively investigate the learned segmentation inference when the ground truth is available. We train s with (1) randomly sampled shapes from uniform distributions over joint angle ranges (`Random`) and (2) the proposed joint training (`Joint`). We use pretraining as initialization as describing in Section 6.2. We then train these two algorithms on the synthetic hand and body data and evaluate segmentation accuracy on the testing sets. The results are shown in Figure 6.8. `Random` is exactly the same as pretraining. After pretraining, `Random` is almost converged. On the other hand, `Joint` improves the segmentation accuracy in both cases by gradually adapting to the true pose distribution when the joint angle regressor q^{θ} is improved. It justifies the effectiveness of the proposed joint training where we can infer the segmentation in a self-supervised manner. For hand data, as we show in Figure 6.7, there are many *touching-skin* poses where fingers are touched to each other. For those poses, there are strong correlations between joints in each pose, which are hard to be sampled by a simple uniform distribution and results in a significant performance gap in Figure 6.8a. For body data, many poses from SURREAL are with separate limbs, which `Random` can generalize surprisingly well. Although it seems `Joint` only leads to incremental improvement over `Random`, we argue this gap is substantial, especially for resolving challenging touching-skin cases as we will show in Section 6.3.3.

6.3.2 Qualitative Study

We compare the proposed LBS-AE with the unsupervised variant of 3D-CODED (Groueix et al., 2018a), which learns the deformation by entirely relying on neural networks to transform a template prior without LBS. Their objective is similar to Eq. (6.7), but using CD and Laplacian regularization only. For fair comparison, we also generate synthetic data on the fly with randomly sampled poses and correspondence for 3D-CODED, which boosts its performance. We also compare with the simplified version of the proposed algorithm by using CD instead of SCD, which is denoted as LBS-AE$_{\text{CD}}$ as above.

We fit and reconstruct the hand and body data as shown in Figure 6.9. For the thumb-up pose, due to wrong correspondences from nearest neighbor search, both 3D-CODED and LBS-AE$_{\text{CD}}$ reconstruct wrong poses. The wrong correspondence causes problems to 3D-CODED. Since the deformation from templates to targeted shapes fully relies on a deep neural network, when the correspondence is wrong and the network is powerful, it learns distorted deformation even with a Laplacian regularization. On the other hand, since LBS-AE$_{\text{CD}}$ still utilizes LBS, the deformation network g is easier to be regularized, which results in better finger reconstructions. We note that 3D-CODED learns proper deformation if the correspondence can be found correctly, such as the third row in Figure 6.9. In both cases, the proposed LBS-AE can learn segmentation well and recover the poses better.

Lastly, we consider fitting FAUST, with only 200 samples, as shown in Figure 6.10. With limited and diverse poses, we have less hint of how the poses deform (Wei et al., 2016), a nearest neighbor search is easily trapped in bad local optimums as we mentioned in Figure 6.3. The proposed LBS-AE still results in reasonable reconstructions and segmentations, though the right arm in the second row suffers from the local optimum issues within the segmentation. A fix is to learn more fine-grained segmentation, but it brings the trade-off between task difficulties and model capacities as well as the as statistical challenges with limited samples.

6.3.3 Quantitative Study

We conduct quantitative analysis on reconstruction, pose estimation, and correspondence on synthetic hand and body data. We use $\sqrt{\text{CD}}$ as the proxy to reconstructions. Pose estimation compares the average ℓ_2 distance between true joint positions and inferred ones while correspondence also measures the average ℓ_2 between found and true correspondences. We randomly generate 4000 testing pairs from the testing data for correspondence comparison. Given two shapes, we fit the shapes via the trained models. Since we know the correspondence of the reconstructions, we project the data onto the reconstructions to find the correspondence. For more details, we refer readers to Groueix et al. (2018a).

We compare three variants of 3D-CODED, including the supervised version with full correspondence, and the unsupervised version with and without synthetic data augmentation aforementioned. For LBS-AE, we also consider three variants, including a simple CD baseline (LBS-AE$_{\text{CD}}$), a segmentation network s trained on poses from uniform distributions LBS-AE$_{\text{RAND}}$ and joint training version (LBS-AE). The results are shown in Table 6.1.

For LBS-AE variants, the jointly trained LBS-AE is better than LBS-AE$_{\text{CD}}$ and LBS-AE$_{\text{RAND}}$. It supports the hypothesis in Section 6.3.1, that joint training facilitates improving model fitting and segmentation. Also, as shown in Section 6.3.1, the pretrained segmentation network still has reasonable testing accuracy and brings an improvement over using CD loss only. On the other hand, the supervised 3D-CODED trained with full correspondence is worse than the proposed unsupervised LBS-AE due to generalization ability. For correspondence on the SMPL training set, supervised Groueix et al. (2018a) achieves 0.065 while LBS-AE achieve 0.069. If we increase the training data size three times, supervised 3D-CODED improves its correspondence result to be 0.095. For hand

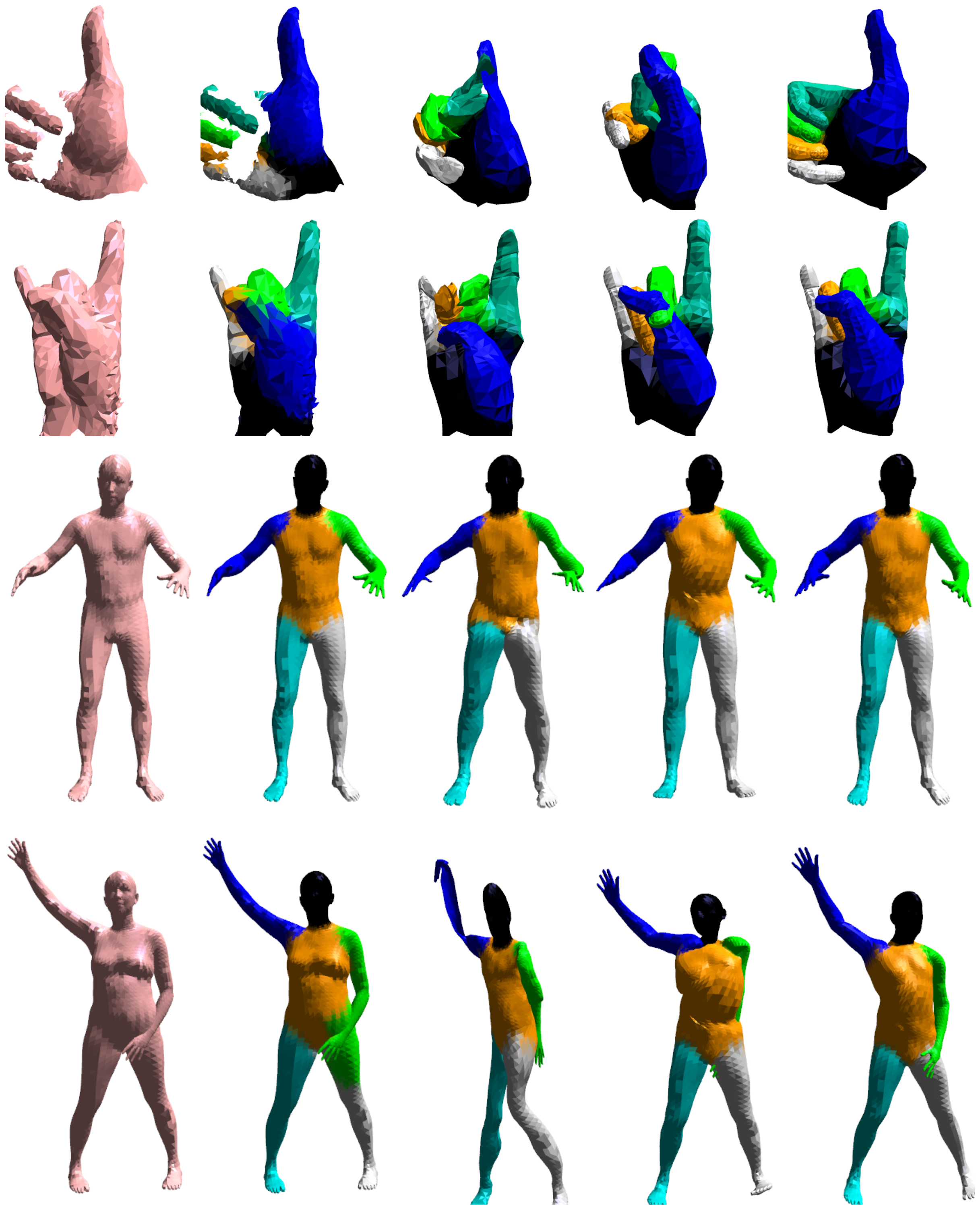
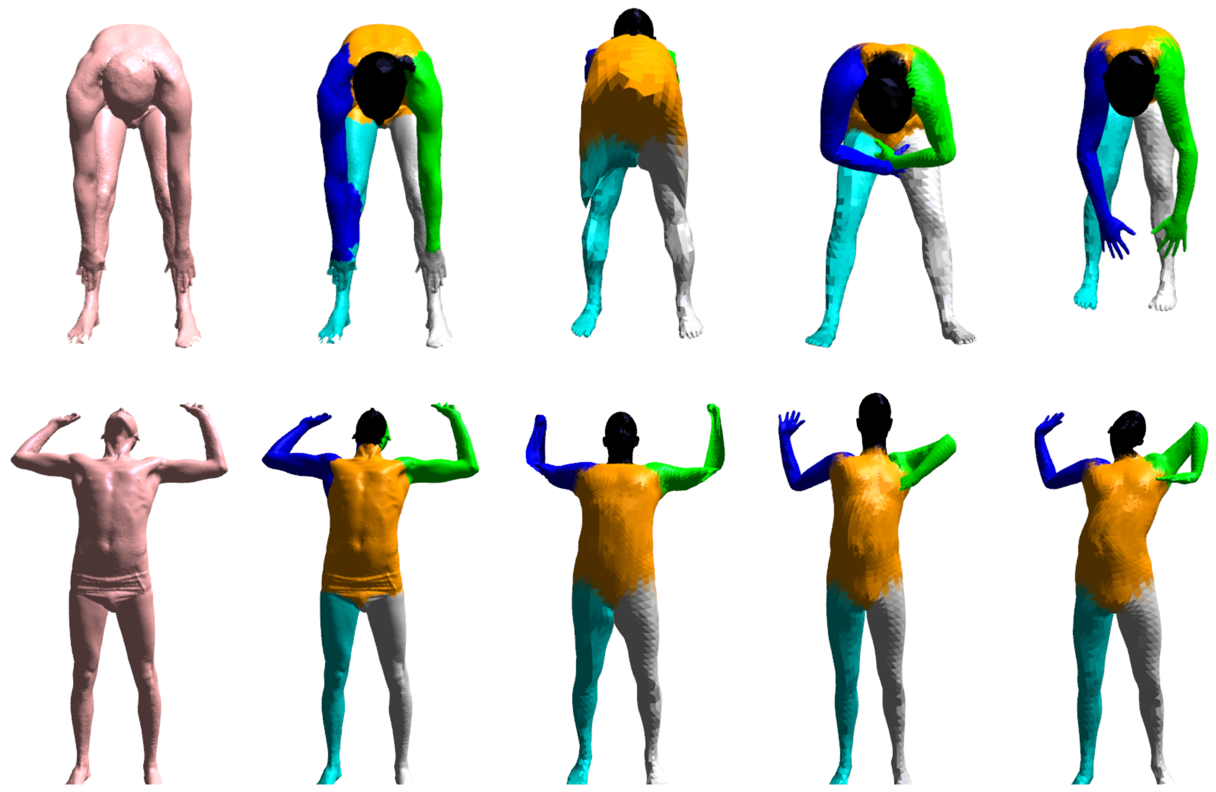


Figure 6.9: Qualitative comparisons on captures hands and SURREAL (SMPL). Given point clouds sampled from the surfaces of input shapes (a), (c-e) are the reconstructions from different algorithms. (b) is the inferred segmentation of LBS-AE on the input shape.



(a) Input (b) Segment (c) Groueix et al. (2018a) (d) CD (e) LBS-AE

Figure 6.10: Qualitative Comparison on FAUST.

data, supervised 3D-CODED generalizes even worse with only 1500 training examples. It suggests that leveraging LBS models into the model can not only use smaller networks but also generalize better than relying on an unconstrained deformation from a deep network.

Deformation Network We also investigate the ability of the deformation in LBS-AE. For data generated via SMPL, we know the ground truth of deformed templates U^{gt} of each shape. The average ℓ_2 distance between corresponding points from U^{gt} and U^d is 0.02, while the average distance between U^{gt} and U is 0.03.

Real-World Benchmark. One representative real-world benchmark is FAUST (Bogo et al., 2014). We follow the protocol used in Groueix et al. (2018a) for comparison, where they train on SMPL with SURREAL parameters and then fine-tune on FAUST. In Groueix et al. (2018a), they use a different number of data from SMPL with SURREAL parameters, while we only use 23K. The numerical results are shown in Table 6.2. With only 23K SMPL data and self-supervision, we are better than unsupervised 3D-CODED with 50K data, supervised 3D-CODED with 10K data, and the supervised learning algorithm

	SMPL			Synthetic Hand		
Algorithm	Recon	Pose	Corre.	Recon	Pose	Corre.
Unsupervised 3D-CODED	0.076	0.082	0.136	0.099	0.035	0.176
Unsupervised 3D-CODED + Augmentation	0.081	0.081	0.132	0.069	0.049	0.140
Supervised 3D-CODED	0.073	0.071	0.104	0.062	0.047	0.135
LBS-AE$_{\text{CD}}$	0.051	0.152	0.147	0.082	0.069	0.168
LBS-AE$_{\text{RAND}}$	0.041	0.058	0.100	0.069	0.050	0.137
LBS-AE	**0.037**	**0.048**	**0.091**	**0.053**	**0.035**	**0.111**

Table 6.1: Quantitative results on synthetic data.

Algorithm	Inter-Subject error (cm)	Intra-Subject error (cm)
FMNet (Litany et al., 2017)	4.826	2.44
Unsupervised 3D-CODED (230K)	4.88	-
Supervised 3D-CODED (10K)	4.70	-
Supervised 3D-CODED (230K)	3.26	1.985
LBS-AE (23K)	4.08	2.161

Table 6.2: Correspondence results on FAUST testing set.

FMNet (Litany et al., 2017). We show some visualization of the inferred correspondence in Figure 6.11.

6.4 Related Works

Finally, since the model fitting and registration is an extensively studied topic, we only briefly review and compare the proposed LBS-AE with existing works.

LBS Extensions Various extensions have been proposed to fix some of the shortcomings of LBS (Bailey et al., 2018; Joshi et al., 2007; Kavan et al., 2008; Kavan and Žára, 2005; Le and Deng, 2012; Lewis et al., 2000; Loper et al., 2015; Rhee et al., 2006; Sloan et al., 2001; Wang and Phillips, 2002; Zuffi and Black, 2015), where we only name afew here. The proposed template deformation follows the idea of Kurihara and Miyata (2004); Loper et al. (2015); Rhee et al. (2006); Zuffi and Black (2015) to model the modalities and corrections of LBS on the base template rest pose. Loper et al. (2015); Zuffi and Black (2015) use PCA-like algorithms to model modalities via a weighted sum of learned shape basis. Instead, our approach is similar to Bailey et al. (2018) by learning modalities via a deformation network. The main difference between LBS-AE and Bailey et al. (2018); Loper et al. (2015); Zuffi and Black (2015) is we do not rely on correspondence information to learn the template deformation d a priori. We simultaneously learn d and infer pose parameters without external labeling.

Deep Learning for 3D Data Many deep learning techniques have been developed for different types of 3D information, such as 3D voxels (Girdhar et al., 2016; Wu et al.,

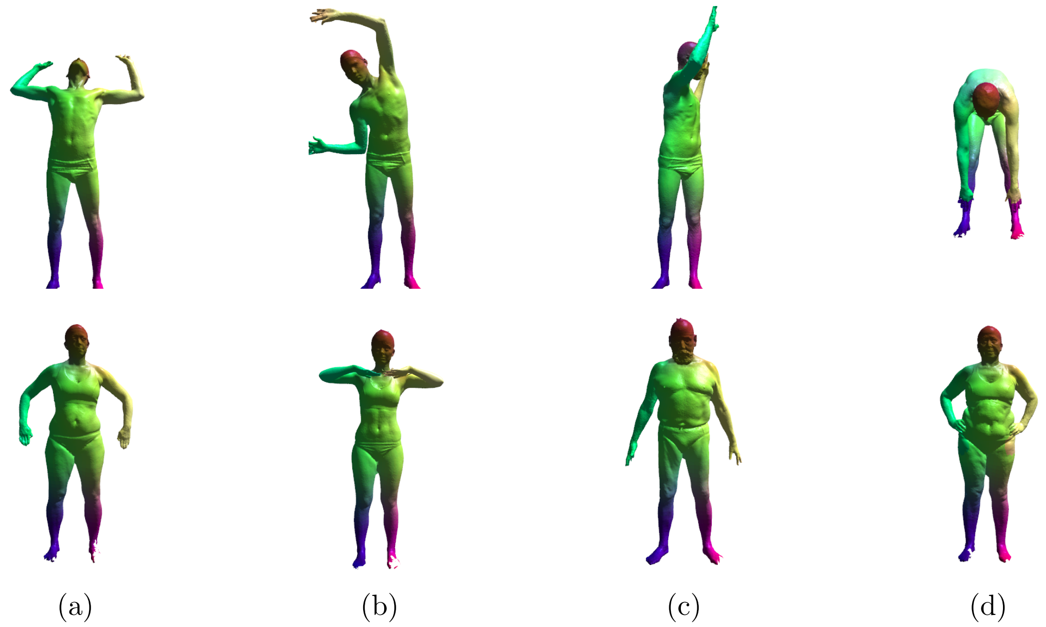


Figure 6.11: Inferred correspondence of FAUST testing data.

2016, 2015), geometry images (Sinha et al., 2016, 2017), meshes (Bronstein et al., 2017), depth maps (Wang et al., 2016) and point clouds (Qi et al., 2017a,b; Zaheer et al., 2017a). Autoencoders for point clouds are explored by Achlioptas et al. (2018); Groueix et al. (2018b); Li et al. (2018); Yang et al. (2018).

Model Fitting with Different Knowledge Different works have studied to registration via fitting a mesh model by leveraging different levels of information about the data. Kanazawa et al. (2018a) use SMPL (Loper et al., 2015) to reconstruct meshes from images by using key points and prior knowledge of distributions of pose parameters. Kanazawa et al. (2018b) explore using a template instead of a controllable model to reconstruct the mesh with key points. Bogo et al. (2016); Joo et al. (2018) also adopt pretrained key point detectors from other sources of data as supervision. Simultaneous training to improve model fitting and key point detection are explored by Lassner et al. (2017); Mehta et al. (2017). The main difference from the proposed joint training in LBS-AE is we do not rely on an additional source of real-world data to pretrain networks, as needed to train these key point detectors. Wei et al. (2016) share a similar idea of using segmentation for nearest neighbor search, but they trained the segmentation from labeled examples. Genova et al. (2018) propose to control morphable models instead of rig models for modeling faces. They also utilize prior knowledge of the 3DMM parameter distributions for real faces. We note that most of the works discussed above aim to recover 3D models from images. Groueix et al. (2018a) is the most related work to the proposed LBS-AE, but doesn't use LBS-based deformation. They use a base template and learn the full deformation process with a neural network trained by correspondences provided a priori or from nearest neighbor search. More comparison between Groueix et al. (2018a) and LBS-AE is studied in Section 6.3.

Lastly, learning body segmentation via SMPL is studied by Varol et al. (2017), but with a focus on learning a segmentation using SMPL with parameters inferred from real-world data to synthesize training examples.

6.5 Summary

We propose an autoencoding algorithm, LBS-AE, which can be used as a building component of PC-GAN, to align articulated mesh models to point clouds. The IGM decoder leverages an artist-defined mesh rig with LBS and only use neural networks to model the template deformation, instead of the full transformation. We constrain the encoder to infer interpretable joint angles. We also propose the structured Chamfer distance for training LBS-AE, defined by inferring a meaningful segmentation of the target data to improve the correspondence finding via nearest neighbor search in the original Chamfer distance. We pave a new way of research in both generative models and computer graphics. In IGM, we show an successful example on how to encode human prior into generators to generate not only structured but also interpretable (e.g. joint angle inference) samples. SCD demonstrate how to define a specialized and powerful probability distance with priors and the learned inference. In graphics, the learned transformation serves as a learning-based corrective modeling while the proposed SCD is a successful example showing how to find correspondences via self-supervision without external labeling.

Part III

Applications of Implicit Generative Models

Chapter 7

Implicit Kernel Learning

Kernel methods are among the essential foundations in machine learning and have been extensively studied in the past decades. In supervised learning, kernel methods allow us to learn non-linear hypothesis. They also play a crucial role in statistics. Kernel maximum mean discrepancy (MMD) (Gretton et al., 2012a) is a powerful two-sample test, which is based on a statistics computed via kernel functions. Even though there is a surge of deep learning in the past years, several successes have been shown by kernel methods and deep feature extraction. Wilson et al. (2016) demonstrate performance improvement by incorporating deep learning, kernel and Gaussian process. In Chapter 2, we show MMD GAN reaches the state-of-the-art performance on generating complex data.

In practice, however, kernel selection is always an important step. Instead of choosing by a heuristic, several works have studied *kernel learning*. Multiple kernel learning (MKL) (Bach, 2009; Bach et al., 2004; Duvenaud et al., 2013; Gönen and Alpaydın, 2011; Lanckriet et al., 2004) is the pioneering framework to combine predefined kernels. One recent kernel learning development is to learn kernels via learning spectral distributions (Fourier transform of the kernel). Wilson and Adams (2013) model spectral distributions via a mixture of Gaussians. Oliva et al. (2016) extend it to Bayesian non-parametric models. In addition to model spectral distribution with *explicit* density models aforementioned, many works optimize the sampled random features or its weights (e.g. Băzăvan et al. (2012); Bullins et al. (2018); Chang et al. (2017b); Sinha and Duchi (2016); Yang et al. (2015)). The other orthogonal approach to modeling spectral distributions is learning feature maps for standard kernels (e.g. Gaussian). Feature maps learned by deep learning lead to state-of-the-art performance on different tasks (Hinton and Salakhutdinov, 2008; Li et al., 2017; Wilson et al., 2016).

In addition to learning effective features, implicit generative models via deep learning also lead to promising performance in learning distributions of complex data as we have discussed in previous chapters. Inspired by its recent success, we propose to model kernel spectral distributions with IGMs in a data-driven fashion, which we call *Implicit Kernel Learning* (IKL). IKL provides a new route to modeling spectral distributions by learning sampling processes of the spectral densities, which is under explored by previous works aforementioned.

In this chapter, we start from studying the generic problem formulation of IKL, and propose an easily implemented, trained and evaluated neural network parameterization which satisfies Bochner's theorem (Section 7.1). We then extend MMD GAN in Chapter 2 with IKL on learning to generate images and text (Section 7.2). In addition to IGMs, the proposed IKL can be applied to standard two-staged supervised learning task with Random Kitchen Sinks (Sinha and Duchi, 2016) (Section 7.4). The conditions required for training IKL and its theoretical guarantees are also studied. In both tasks, we show that IKL leads to competitive or better performance than heuristic kernel selections and existing approaches modeling kernel spectral densities. It demonstrates the potentials of applying IGMs to different machine learning tasks.

7.1 Kernel Learning

Kernels have been used in several applications with success, including supervised learning, unsupervised learning, and hypothesis test. They have also been combined with deep learning in different applications (Dziugaite et al., 2015; Li et al., 2015b; Mairal, 2016; Mairal et al., 2014; Wilson et al., 2016). Given data $x \in \mathbb{R}^d$, kernel methods compute the inner product of the feature transformation $\varphi(x)$ in a high-dimensional Hilbert space H via a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which is defined as $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_H$, where $\varphi(x)$ is usually high or even infinitely dimensional. If k is shift invariant (i.e. $k(x, y) = k(x-y)$), we can represent k as an expectation with respect to a spectral distribution $\mathbb{P}_k(\omega)$.

Bochner's theorem (Rudin, 2011) A continuous, real valued, symmetric and shift-invariant function k on $\mathbb{R}^d$ is a positive definite kernel if and only if there is a positive finite measure $\mathbb{P}_k(\omega)$ such that

$$k(x - x') = \int_{\mathbb{R}^d} e^{i\omega^\top (x-x')} d\mathbb{P}_k(\omega) = \mathbb{E}_{\omega \sim \mathbb{P}_k} \left[e^{i\omega^\top (x-x')} \right].$$

7.1.1 Implicit Kernel Learning

We restrict ourselves to learning shift invariant kernels. According to that, learning kernels is equivalent to learning a spectral distribution by optimizing

$$\begin{aligned} &\arg\max_{k \in \mathcal{K}} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') k(x, x') \right] = \\ &\arg\max_{k \in \mathcal{K}} \sum_{i=1} \mathbb{E}_{x \sim \mathbb{P}_i, x' \sim \mathbb{Q}_i} \left[F_i(x, x') \mathbb{E}_{\omega \sim \mathbb{P}_k} \left[e^{i\omega^\top (x-x')} \right] \right], \end{aligned} \tag{7.1}$$

where F is a task-specific objective function and $\mathcal{K}$ is a set of kernels. Eq. (7.1) covers many popular objectives, such as kernel alignment (Gönen and Alpaydın, 2011) and MMD distance (Gretton et al., 2012a). Existing works (Oliva et al., 2016; Wilson and Adams, 2013) learn the spectral density $\mathbb{P}_k(\omega)$ with *explicit* forms via parametric or non-parametric models. When we learn kernels via Eq. (7.1), the access of spectral density functions $\mathbb{P}_k(\omega)$ is not necessary as long as we are able to estimate kernel evaluations $k(x-x') = \mathbb{E}_\omega[e^{i\omega^\top (x-x')}]$ via sampling from $\mathbb{P}_k(\omega)$ (Rahimi and Recht, 2007). Alternatively, *implicit probabilistic (generative) models* define a stochastic procedure that can generate (sample) data from $\mathbb{P}_k(\omega)$ without modeling $\mathbb{P}_k(\omega)$. Recently, neural implicit generative models (MacKay, 1995) regained attentions with promising results (Goodfellow et al., 2014) and simple sampling procedures. We first sample ν from a base distribution $\mathbb{P}(\nu)$ which is known (e.g. Gaussian distribution), then use a deterministic function h_ψ parametrized by ψ, to transform ν into $\omega = h_\psi(\nu)$, where ω follows the complex target distribution $\mathbb{P}_k(\omega)$. Inspired by the success of implicit generative models, we propose an *Implicit Kernel Learning (IKL)* method by modeling $\mathbb{P}_k(\omega)$ via an implicit generative model $h_\psi(\nu)$, where $\nu \sim \mathbb{P}(\nu)$, which results in

$$k_\psi(x,x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(x-x')}\right], \tag{7.2}$$

and reducing Eq. (7.1) to solve

$$\operatorname*{argmax}_{\psi} \sum_{i=1} \mathbb{E}_{x\sim\mathbb{P}_i, x'\sim\mathbb{Q}_i}\left[F_i(x,x')\mathbb{E}_\nu\left(e^{ih_\psi(\nu)^\top(x-x')}\right)\right]. \tag{7.3}$$

The gradient of Eq. (7.3) can be represented as

$$\sum_{i=1} \mathbb{E}_{x\sim\mathbb{P}_i, x'\sim\mathbb{Q}_i}\mathbb{E}_\nu\left[\nabla_\psi F_i(x,x')e^{ih_\psi(\nu)^\top(x-x')}\right].$$

Thus, Eq. (7.3) can be optimized via sampling x, x' from data and ν from the base distribution to estimate gradient as shown above (SGD) in every iteration. Next, we discuss the parametrization of h_ψ to satisfy Bochner's Theorem, and describe how to evaluate IKL kernels in practice.

Symmetric $\mathbb{P}_k(\omega)$ To result in real valued kernels, the spectral density has to be symmetric, where $\mathbb{P}_k(\omega) = \mathbb{P}_k(-\omega)$. Thus, we parametrize $h_\psi(\nu) = \texttt{sign}(\nu) \circ \widetilde{h}_\psi(\texttt{abs}(\nu))$, where $\circ$ is the Hadamard product and $\widetilde{h}_\psi$ can be any unconstrained function if the base distribution $\mathbb{P}(\nu)$ is symmetric (i.e. $\mathbb{P}(\nu) = \mathbb{P}(-\nu)$), such as standard normal distributions.

Kernel Evaluation Although there is usually no closed form for the kernel evaluation $k_\psi(x,x')$ in Eq. (7.2) with fairly complicated h_ψ, we can evaluate (approximate) $k_\psi(x,x')$ via sampling finite number of random Fourier features $\widehat{k}_\psi(x,x') = \widehat{\varphi}_{h_\psi}(x)^\top\widehat{\varphi}_{h_\psi}(x')$, where $\widehat{\varphi}_{h_\psi}(x)^\top = [\varphi(x;h_\psi(\nu_1)),\ldots,\varphi(x;h_\psi(\nu_m))]$, and $\varphi(x;\omega)$ is the evaluation on ω of the Fourier transformation $\varphi(x)$ (Rahimi and Recht, 2007).

Next, we demonstrate two applications covered by Eq. (7.3) as examples, where we can apply IKL, including maximum mean discrepancy (MMD) and kernel alignment.

7.2 MMD GAN with IKL

Given $\{x_i\}_{i=1}^n \sim \mathbb{P}_\mathcal{X}$, instead of estimating the density $\mathbb{P}_\mathcal{X}$, implicit generative models learns a generative network g_θ (generator), which transforms a base distribution $\mathbb{P}_\mathcal{Z}$ over

$\mathcal{Z}$ into an *implicit* distribution $\mathbb{Q}_\theta$ to approximate $\mathbb{P}_\mathcal{X}$. During the training, IGMs have to alternatively estimate a *distance* $D(\mathbb{P}_\mathcal{X}\|\mathbb{Q}_\theta)$ between $\mathbb{P}_\mathcal{X}$ and $\mathbb{Q}_\theta$, and updates g_θ by minimizing $D(\mathbb{P}_\mathcal{X}\|\mathbb{Q}_\theta)$. Different probability metrics have been studied (Arbel et al., 2018; Arjovsky et al., 2017; Dziugaite et al., 2015; Goodfellow et al., 2014; Gulrajani et al., 2017; Li et al., 2017, 2015b; Mroueh et al., 2018; Mroueh and Sercu, 2017; Mroueh et al., 2017; Nowozin et al., 2016). Kernel maximum mean discrepancy (MMD) is a probability metric, which is commonly used in two-sample-test to distinguish two distributions with finite samples (Gretton et al., 2012a). Given a kernel k, recall that MMD is defined as

$$M_k(\mathbb{P},\mathbb{Q}) = \|\mu_\mathbb{P} - \mu_\mathbb{Q}\|^2_\mathcal{H} = \mathbb{E}_{\mathbb{P},\mathbb{P}}[k(x,x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x,y)] + \mathbb{E}_{\mathbb{Q},\mathbb{Q}}[k(y,y')].$$

Dziugaite et al. (2015); Li et al. (2015b) train the generator g_θ by optimizing $\min_\theta M_k(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta)$ with a Gaussian kernel k. In Chapter 2, we consider the MMD $M_\phi(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta)$ with a composition kernel

$$k_\phi(x,x') = \exp(-\|f_\phi(x) - f_\phi(x)'\|^2), \tag{7.4}$$

which combines Gaussian kernel k and a neural network f_ϕ. The proposed MMD GAN learns g_θ via solving $\min_\theta \max_\phi M_\phi(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta)$. The inner maximization $\max_\phi M_\phi(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta)$ can be interpret as a new distance defined by a variational form or a MMD with kernel learning. Please refer to Chapter 2 for more detailed discussions.

7.2.1 Training MMD GAN with IKL

Although the composition kernel with a learned feature embedding f_ϕ is powerful, choosing a good base kernel k is still crucial in practice (Bińkowski et al., 2018). Different base kernels for MMD GAN, such as rational quadratic kernel (Bińkowski et al., 2018) and distance kernel (Bellemare et al., 2017), have been studied. Instead of choosing it by hands, we propose to learn the base kernel by IKL, which extend Eq. (7.4) to be $k_{\psi,\phi} = k_\psi \circ f_\phi$ with the form

$$k_{\psi,\phi}(x,x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(f_\phi(x)-f_\phi(x'))}\right]. \tag{7.5}$$

We then extend the MMD GAN objective to be

$$\min_\theta \max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta), \tag{7.6}$$

where $M_{\psi,\phi}$ is the MMD distance with the IKL kernel (7.5). Clearly, for a given ϕ, the maximization over ψ in Eq. (7.6) can be represented as Eq. (7.1) by letting $F_1(x,x') = 1$, $F_2(x,y) = -2$ and $F_3(y,y') = 1$. In what follows, we will use for convenience $k_{\psi,\phi}$, k_ψ and k_ϕ to denote kernels defined in Eq. (7.5), (7.2) and Eq. (7.4) respectively.

7.2.2 Property of MMD GAN with IKL

As proven by Arjovsky and Bottou (2017), some probability distances adopted by existing works (e.g. Goodfellow et al. (2014)) are not *weak* (i.e. $\mathbb{P}_n \xrightarrow{D} \mathbb{P}$ then $D(\mathbb{P}_n\|\mathbb{P}) \to 0$), which cannot provide better signal to train g_θ. Also, they usually suffer from discontinuity, hence

Algorithm 4 MMD GAN with IKL

```
**Input:** $\eta$ the learning rate, $B$ the batch size, $n_c$ number of $f, h$ updates per $g$ update, $m$ the number of basis, $\lambda_{GP}$ the coefficient of gradient penalty, $\lambda_h$ the coefficient of variance constraint.
**Initial** parameter $\theta$ for $g$, $\phi$ for $f$, $\psi$ for $h$
**Define** $\mathcal{L}(\psi,\phi) = M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta) - \lambda_{GP}(\|\nabla_{\widehat{x}} f_\phi(\widehat{x})\|_2 - 1)^2 - \lambda_h(\mathbb{E}_\nu[\|h_\psi(\nu)\|^2] - u)^2$
**while** $\theta$ has not converged **do**
    **for** $t = 1,\ldots,n_c$ **do**
        Sample $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X})$, $\{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z})$, $\{\nu_k\}_{k=1}^m \sim \mathbb{P}(\nu)$
        $(\psi,\phi) \leftarrow \phi + \eta\text{Adam}((\psi,\phi), \nabla_{\psi,\phi}\mathcal{L}(\psi,\phi))$
    **end for**
    Sample $\{x_i\}_{i=1}^B \sim \mathbb{P}(\mathcal{X})$, $\{z_j\}_{j=1}^B \sim \mathbb{P}(\mathcal{Z})$, $\{\nu_k\}_{k=1}^m \sim \mathbb{P}(\nu)$
    $\theta \leftarrow \theta - \eta\text{Adam}(\theta, \nabla_\theta M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta))$
**end while**
```

it cannot be trained via gradient descent at certain points. We extend Theorem 3 and Theorem 4 to prove that $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta)$ is a continuous and differentiable objective in θ and *weak* under an extended mild assumptions based on Assumption 21.

Assumption 21. *$g_\theta(z)$ is locally Lipschitz and differentiable in θ; $f_\phi(x)$ is Lipschitz in x and $\phi \in \Phi$ is compact. $f_\phi \circ g_\theta(z)$ is differentiable in θ and there are local Lipschitz constants, which is independent of ϕ, such that $\mathbb{E}_{z\sim\mathbb{P}_z}[L(\theta,z)] < +\infty$. The above assumptions are adopted by Arjovsky et al. (2017). Lastly, assume given any $\psi \in \Psi$, where Ψ is compact, $k_\psi(x,x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(x-x')}\right]$ and $|k_\psi(x,x')| < \infty$ is differentiable and Lipschitz in (x,x') which has an upper bound L_k for Lipschitz constant of (x,x') given different ψ.*

Theorem 22. *Assume function g_θ and kernel $k_{\psi,\phi}$ satisfy Assumption 21, $\max_{\psi,\phi} M_{\psi,\phi}$ is weak, that is, $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{P}_n) \to 0 \iff \mathbb{P}_n \xrightarrow{D} \mathbb{P}_\mathcal{X}$. Also, $\max_{\psi,\phi} M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta)$ is continuous everywhere and differentiable almost everywhere in θ.*

Lemma 23. *Assume $\mathcal{X}$ is bounded. Let $x,x' \in \mathcal{X}$, $k_\psi(x,x') = \mathbb{E}_\nu\left[e^{ih_\psi(\nu)^\top(x-x')}\right]$ is Lipschitz in (x,x') if $\mathbb{E}_\nu\left[\|h_\psi(\nu)\|^2\right] < \infty$.*

We note that $\mathbb{E}_\nu\left[\|h_\psi(\nu)\|^2\right]$ is variance since $\mathbb{E}_\nu\left[h_\psi(\nu)\right] = 0$. We penalize $\lambda_h(\mathbb{E}_\nu\left[\|h_\psi(\nu)\|^2\right] - u)^2$ as an approximation of Lemma 23 in practice to ensure that assumptions in Theorem 22 are satisfied. The algorithm with IKL and gradient penalty (Bińkowski et al., 2018) is shown in Algorithm 4.

7.3 Empirical Study of GAN with IKL

We consider image and text generation tasks for quantitative evaluation. For image generation, we evaluate the inception score (Salimans et al., 2016) and FID score (Heusel et al., 2017) on CIFAR-10 (Krizhevsky and Hinton, 2009). We use DCGAN (Radford et al., 2016) and expands the output of f_ϕ to be 16-dimensional as Bińkowski et al. (2018). For text generation, we consider a length-32 character-level generation task on Google Billion

Words dataset as Chapter 4. The evaluation is based on Jensen-Shannon divergence between empirical 4-gram probabilities (JS-4) of the generated sequence and the validation data (Gulrajani et al., 2017; Heusel et al., 2017; Mroueh et al., 2018). The model architecture follows Gulrajani et al. (2017) in using ResNet with 1D convolutions. We train every algorithm $10,000$ iterations for comparison.

For MMD GAN with fixed base kernels, we consider the mixture of Gaussian kernels $k(x,x') = \sum_q \exp(-\frac{\|x-x'\|^2}{2\sigma_q^2})$ as in Chapter 2 and the mixture of RQ kernels $k(x,x') = \sum_q (1+\frac{\|x-x'\|^2}{2\alpha_q})^{-\alpha_q}$ (Bińkowski et al., 2018). We tuned hyperparameters σ_q and α_q for each kernel as reported in Section 7.6.4.

Lastly, for learning base kernels, we compare IKL with SM kernel (Wilson and Adams, 2013) f_ϕ, which learns mixture of Gaussians to model kernel spectral density. It can be treated as the *explicit generative model counter part* of the proposed IKL.

In both tasks, $\mathbb{P}(\nu)$, the base distribution of IKL, is a standard normal distribution and h_ψ is a 3-layer MLP with 32 hidden units for each layer. Similar to the aforementioned mixture kernels, we consider the mixture of IKL kernels with the variance constraints $\mathbb{E}[\|h_\psi(\nu)\|^2] = 1/\sigma_q$, where σ_q is the bandwidths for the mixture of Gaussian kernels. Note that if h_ψ is an identity map, we recover the mixture of Gaussian kernels. We fix λ_h to be 10 and resample $m = 1024$ random features for IKL in every iteration. For other settings, we follow Bińkowski et al. (2018) and the hyperparameters can be found in Section 7.6.4.

7.3.1 Quantitative and Qualitative Results

We compare MMD GAN with the proposed IKL and different fixed kernels. We repeat the experiments 10 times and report the average result with standard error in Table 7.1. Note that for inception score the larger the better; while FID and JS-4 the smaller the better. We also report WGAN-GP results as a reference. Sampled images on larger datasets are shown in Figure 7.1. Before the discussion, we note that the difference in FID is less significant than inception score and JS-4, but FID score is a biased evaluation metric as discussed in Bińkowski et al. (2018).

Method	Inception Scores (↑)	FID Scores (↓)	JS-4 (↓)
Gaussian	6.726 ± 0.021	32.50 ± 0.07	0.381 ± 0.003
RQ	6.785 ± 0.031	32.20 ± 0.09	0.463 ± 0.005
SM	6.746 ± 0.031	32.43 ± 0.08	0.378 ± 0.003
IKL	$\mathbf{6.876 \pm 0.018}$	$\mathbf{31.98 \pm 0.05}$	$\mathbf{0.372 \pm 0.002}$
WGAN-GP	6.539 ± 0.034	36.413 ± 0.05	0.379 ± 0.002

Table 7.1: Inception scores, FID scores, and JS-4 divergece results.

Pre-defined Kernels Bińkowski et al. (2018) show RQ kernels outperform Gaussian and energy distance kernels on image generation. Our empirical results agree with such finding:

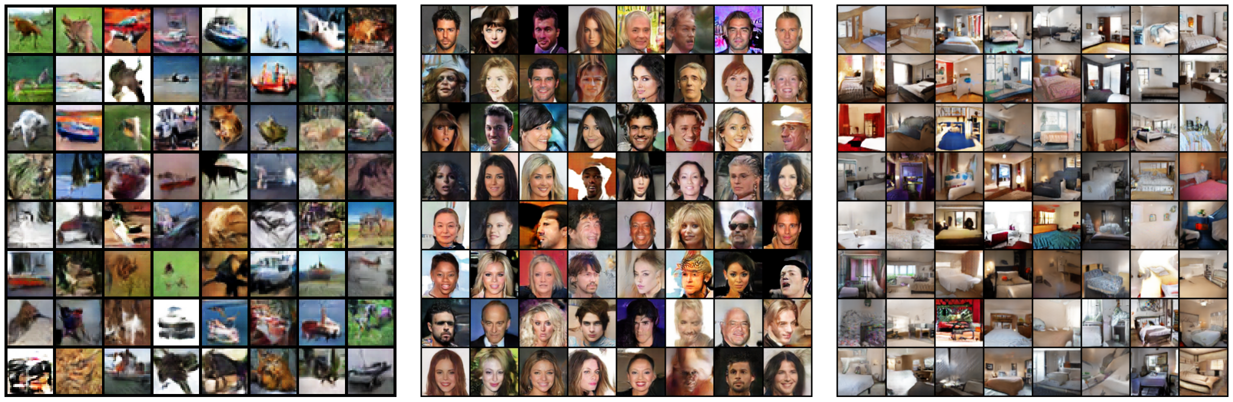


Figure 7.1: Samples generated by MMD GAN-IKL on CIFAR-10, CELEBA and LSUN dataset.

RQ kernels achieve 6.785 inception score while for Gaussian kernel it is 6.726, as shown in the left column of Table 7.1. In text generation, nonetheless, RQ kernels only achieve 0.463 JS-4 score[1] and are not on par with 0.381 acquired by Gaussian kernels, even though it is still slightly worse than WGAN-GP. These results imply *kernel selection is task-specific*. On the other hand, the proposed IKL learns kernels in a data-driven way, which results in the best performance in both tasks. In CIFAR-10, although Gaussian kernel is worse than RQ, IKL is still able to transforms $\mathbb{P}(\nu)$, which is Gaussian, into a powerful kernel, and outperforms RQ on inception scores (6.876 v.s. 6.785). For text generation, from Table 7.1 and Figure 7.2, we observe that IKL can further boost Gaussian into better kernels with substantial improvement. Also, we note that the difference between IKL and pre-defined kernels in Table 7.1 is significant based on the t-test at 95% confidence level.

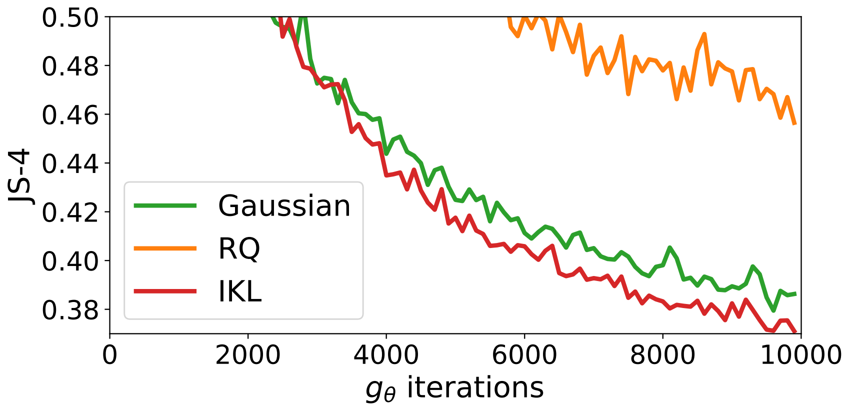


Figure 7.2: Convergence of MMD GANs with different kernels on text generation.

[1]For RQ kernels, we searched 10 possible hyperparameter settings to ensure the unsatisfactory performance is not caused by the improper parameters.

Learned Kernels The SM kernel (Wilson and Adams, 2013), which learns the spectral density via mixture of Gaussians, does not significantly outperforms Gaussian kernel as shown in Table 7.1, since Li et al. (2017) already uses equal-weighted mixture of Gaussian formulation. It suggests that proposed IKL can learn more complicated and effective spectral distributions than simple mixture models.

7.3.2 Study of Variance Constraints

In Lemma 23, we prove bounding variance $\mathbb{E}[\|h_\psi(\nu)\|^2]$ guarantees k_ψ to be Lipschitz as required in Theorem 22. We investigate the importance of this constraint. In Figure 7.3, we show the training objective (MMD), $\mathbb{E}[\|h_\psi(\nu)\|^2]$ and the JS-4 divergence for training MMD GAN (IKL) without variance constraint, i.e. $\lambda_h = 0$. We could observe the variance keeps going up without constraints, which leads exploded MMD values. Also, when the exploration is getting severe, the JS-4 divergence starts increasing, which implies MMD cannot provide meaningful signal to g_θ. The study justifies the validity of Theorem 22 and Lemma 23.

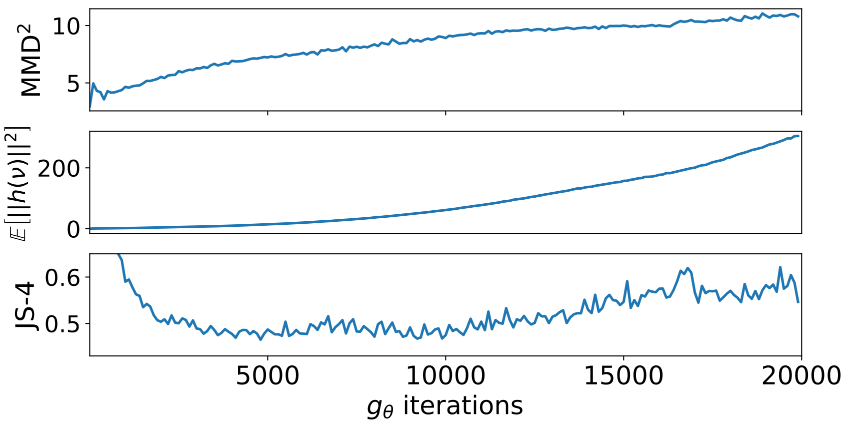


Figure 7.3: Learning MMD GAN (IKL) without the variance constraint on Google Billion Words datasets for text generation.

In Section 7.2, we propose to constrain variance via $\lambda_h(\mathbb{E}_\nu[\|h_\psi(\nu)\|^2] - u)^2$. There are other alternatives, such as using Lagrangian. In practice, we do not observe significant difference. Last, we remark that the proposed constraint is a sufficient condition. For CIFAR-10, without the constraint, we observe that the variance is still bouncing between 1 and 2 without explosion as Figure 7.3. Therefore, the training leads to a satisfactory result with 6.731 ± 0.034 inception score, but it is slightly worse than IKL in Table 7.1. The necessary or weaker sufficient conditions are worth further studying as a future work.

7.3.3 Computational Time

One concern of the proposed IKL is the computational overhead introduced by sampling random features as well as using more parameters to model h_ψ.

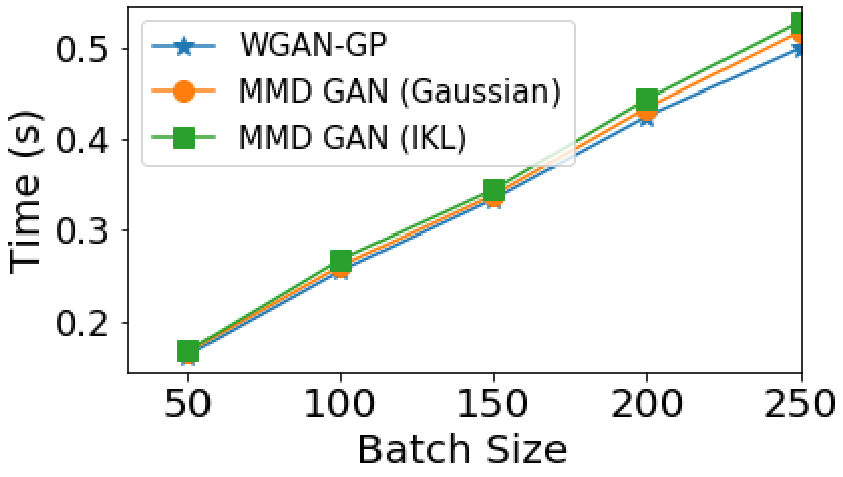


Figure 7.4: Computational time for one generator update iteration with different batch sizes.

Model Capacity For f_ϕ, the number of parameters for DCGAN is around 0.8 million for size 32×32 images and 3 millions for size 64×64 images. The ResNet architecture used in Gulrajani et al. (2017) has around 10 millions parameters. In contrast, in all experiments, we use simple three layer MLP as h_ψ for IKL, where the input and output dimensions are 16, and hidden layer size is 32. The total parameters are just around 2,000. Compared with f_ϕ, the additional number of parameters used for h_ψ is almost negligible.

The other concern of IKL is sampling random features for each examples. In our experiments, we use $m = 1024$ random features for each iteration. We measure the time per iteration of updating critic iterations (f for WGAN-GP and MMD GAN with Gaussian kernel; f and h for IKL) with different batch sizes under Titan X as shown in 7.4. The difference between WGAN-GP, MMD GAN and IKL are not significant. The reason is computing MMD and random feature is highly parallelizable, and other computation, such as evaluating f_ϕ and its gradient penalty, dominates the cost because f_ϕ has much more parameters as aforementioned. Therefore, we believe the proposed IKL is still cost effective in practice.

7.3.4 IKL with and without Neural Networks on GAN training

Instead of learning a transformation function h_ψ for spectral distributions as we proposed in Section 7.1 (IKL-NN), the other realization of IKL is to keep a pool of a finite number of learned random features $\Omega = \{\widehat{\omega}_i\}_{i=1}^m$, and approximate the kernel evaluation by $\widehat{k}_\Omega(x, x') = \widehat{\varphi}_\Omega(x)^\top \widehat{\varphi}_\Omega(x')$, where $\widehat{\varphi}_\Omega(x)^\top = [\varphi(x; \widehat{\omega}_1), \ldots, \varphi(x; \widehat{\omega}_m)]$. During the learning, it directly optimize $\widehat{\omega}_i$. Many existing works study this idea for supervised learning, such as Băzăvan et al. (2012); Bullins et al. (2018); Chang et al. (2017b); Sinha and Duchi (2016); Yang et al. (2015). We call the latter realization as IKL-RFF. Next, we discuss and compare the difference between IKL-NN and IKL-RFF.

The crucial difference between IKL-NN and IKL-RFF is, IKL-NN can sample arbitrary

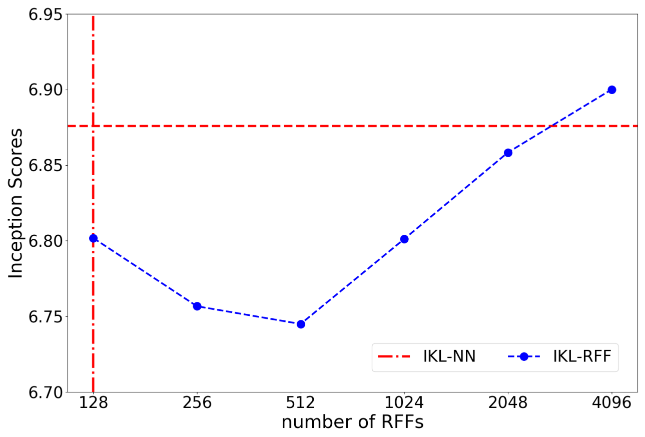


Figure 7.5: The comparison between IKL-NN and IKL-RFF on CIFAR-10 under different number of random features.

number of random features by first sampling $\nu \sim \mathbb{P}(\nu)$ and transforming it via $h_\psi(\nu)$, while IKL-RFF is restricted by the pool size m. If the application needs more random features, IKL-RFF will be memory inefficient. Specifically, we compare IKL-NN and IKL-RFF with different number of random features in Figure 7.5. With the same number of parameters (i.e., $|h_\psi| = m \times dim(\nu)$)[2] , IKL-NN outperforms IKL-RFF of $m = 128$ on inception scores (6.876 versus 6.801). For IKL-RFF to achieve the same or better Inception scores of IKL-NN, the number of random features m needs increasing to 4096, which is less memory efficient than the IKL-NN realization. In particular, h_ψ of IKL-NN is a three-layers MLP with 2048 number of parameters ($16\times32+32\times32+32\times16$), while IKL-RFF has 2048, 65536 number of parameters, for $m = 128, 4096$, respectively.

Algorithm	JS-4
IKL-NN	0.372 ± 0.002
IKL-RFF	0.383 ± 0.002
IKL-RFF (+2)	0.380 ± 0.002
IKL-RFF (+4)	0.377 ± 0.002
IKL-RFF (+8)	0.375 ± 0.002

Table 7.2: The comparison between IKL-NN and IKL-RFF on Google Billion Word.

On the other hand, using large m for IKL-RFF not only increases the number of parameters, but might also enhance the optimization difficulty. Zhang et al. (2017) discuss the difficulty of optimizing RFF directly on different tasks. Here we compare IKL-NN and IKL-RFF on challenging Google Billion Word dataset. We train IKL-RFF with the same

[2] $|h_\psi|$ denotes number of parameters in h_ψ, m is number of random features and $dim(\nu)$ is the dimension of the ν.

setting as Section 7.6.4, where we set the pool size m to be 1024 and the updating schedule between critic and generator to be 10 : 1, but we tune the Adam optimization parameter for IKL-RFF for fair comparison. As discussed above, please note that the number of parameters for h_ψ is 2048 while IKL-RFF uses 16384 when $m = 1024$. The results are shown in Table 7.2. Even IKL-RFF is using more parameters, the performance 0.383 is not competitive as IKL-NN, which achieves 0.372.

In Algorithm 4, we update f_ϕ and h in each iteration with n_c times, where we use $n_c = 10$ here. We keep the number of updating f_ϕ to be 10, but increase the number of update for $\{\widehat{\omega}_i\}_{i=1}^{1024}$ to be $12, 14, 18$ in each iteration. The result is shown in Table 7.2 with symbols +2, +4 and +8 respectively. Clearly, we see IKL-RFF need more number of updates to achieve competitive performance with IKL-NN. The results might implies IKL-RFF is a more difficult optimization problem with more parameters than IKL-NN. It also confirms the effectiveness of learning IGMs with deep neural networks, but the underlying theory is still an open research question. A better optimization algorithm (Zhang et al., 2017) may improve the performance gap between IKL-NN and IKL-RFF, which worth more study as future work.

7.4 Random Kitchen Sinks with IKL

In addition to improving learning IGMs, IKL also benefits non-generative tasks in machine learning. Rahimi and Recht (2009) propose *Random Kitchen Sinks* (RKS) as follows. We sample $\omega_i \sim \mathbb{P}_k(\omega)$ and transform $x \in \mathbb{R}^d$ into

$$\widehat{\varphi}(x) = [\varphi(x;\omega_1), \ldots, \varphi(x;\omega_M)], \text{ where } \sup_{x,\omega} |\varphi(x;\omega)| < 1.$$

We then learn a classifier on the transformed features $\widehat{\varphi}(x;\omega)$. Kernel methods with random features (Rahimi and Recht, 2007) is an example of RKS, where $\mathbb{P}_k(\omega)$ is the spectral distribution of the kernel and $\varphi(x;\omega) = \left[\cos(\omega^\top x), \sin(\omega^\top x)\right]$. We usually learn a model $\mathbf{w}$ by solving

$$\operatorname*{argmin}_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell\left(\mathbf{w}^\top \widehat{\varphi}(x_i)\right). \tag{7.7}$$

If ℓ is a convex loss function, the objective (7.7) can be solved efficiently to global optimum.

Spectral distributions $\mathbb{P}_k$ are usually set as a parameterized form, such as Gaussian distributions, but the selection of $\mathbb{P}_k$ is important in practice. If we consider RKS as kernel methods with random features, then selecting $\mathbb{P}$ is equivalent to the well-known kernel selection (learning) problem for supervised learning (Gönen and Alpaydın, 2011).

Two-Stage Approach We follows Sinha and Duchi (2016) to consider kernel learning for RKS with a two-stage approach. In stage 1, we consider kernel alignment (Cristianini et al., 2002) of the form, $\operatorname{argmax}_{k\in\mathcal{K}} \mathbb{E}_{(x,y),(x',y')} \sum_{i\neq j} yy'k(x,x')$. By parameterizing k via the implicit generative model h_ψ as in Section 7.1, we have the following problem:

$$\operatorname*{argmax}_{\psi} \mathbb{E}_{(x,y),(x',y')} yy' \mathbb{E}_\nu \left[e^{ih_\psi(\nu)^\top (x-x')}\right], \tag{7.8}$$

Algorithm 5 Random Kitchen Sinks with IKL

```
**Stage 1: Kernel Learning**
**Input:** $X = \{(x_i, y_i)\}_{i=1}^n$, the batch size $B$ for data and $m$ for random feature, learning rate $\eta$
Initial parameter $\psi$ for $h$
**while** $\psi$ has not converged or reach maximum iters **do**
    Sample $\{(x_i, y_i)\}_{i=1}^B \subseteq X$. Fresh sample $\{\nu_j\}_{j=1}^m \sim \mathbb{P}(\nu)$
    $g_\psi \leftarrow \nabla_\psi \frac{1}{B(B-1)} \sum_{i \neq i'} y_i y_{i'} \frac{1}{m} \sum_{j=1}^m e^{ih_\psi(\nu_j)^\top (x_i - x_{i'}))}$
    $\psi \leftarrow \psi - \eta \mathrm{Adam}(\psi, g_\psi)$
**end while**
**Stage 2: Random Kitchen Sinks**
Sample $\{\nu_i\}_{i=1}^M \sim \mathbb{P}(\nu)$, note that $M$ is not necessarily equal to $m$
Transform $X$ into $\varphi(X)$ via $h_\psi$ and $\{\nu_i\}_{i=1}^M$
Learn a linear classifier on $(\varphi(X), Y)$
```

which can be treated as Eq. (7.1) with $F_1(x, x') = yy'$. After solving Eq. (7.8), we learn a *sampler* h_ψ where we can easily sample from. In stage 2, we thus have the advantage of solving a convex problem Eq. (7.7) in RKS with IKL. The algorithm is shown in Algorithm 5.

Note that in stage 1, we resample $\{\nu_j\}_{j=1}^m$ in every iteration to train an implicit generative model h_ψ. The advantage of Algorithm 5 is the random features used in kernel learning and RKS can be *different*, which allows us to use *less* random features in kernel learning (stage 1), and sample *more* features for RKS (stage 2).

One can also *jointly* train both feature mapping ω and the model parameters $\mathbf{w}$, such as neural networks. We remark that our intent is not to show state-of-the-art results on supervised learning, on which deep neural networks dominate (He et al., 2016; Krizhevsky et al., 2012). We use RKS as a protocol to study kernel learning and the proposed IKL, which still has competitive performance with neural networks on some tasks (Rahimi and Recht, 2009; Sinha and Duchi, 2016). Also, the simple procedure of RKL with IKL allows us to provide some theoretical guarantees of the performance, which is sill challenging of deep learning models.

Comparison with Existing Works Sinha and Duchi (2016) learn non-uniform weights for M random features via kernel alignments in stage 1 then using these optimized features in RKS in the stage 2. Note that the random features used in stage 1 has to be the same as the ones in stage 2. A joint training of feature mapping and classifier can be treated as a 2-layer neural networks (Alber et al., 2017; Băzăvan et al., 2012; Bullins et al., 2018). Learning kernels with aforementioned works will be more costly if we want to use a large number of random features for training classifiers. In contrast to IGMs, Oliva et al. (2016) learn an explicit Bayesian nonparametric generative model for spectral distributions, which requires specifically designed inference algorithms. Learning kernels for Eq. (7.7) in dual form without random features has also been proposed. It usually require costly steps, such as eigendecomposition of the Gram matrix (Gönen and Alpaydın, 2011).

7.4.1 Consistency and Generalization

The simple two-stages approach, IKL with RKS, allows us to provide the consistency and generalization guarantees. For consistency, it guarantees the solution of finite sample approximations of Eq. (7.8) approach to the optimum of Eq. (7.8) (population optimum), when we increase the number of training data and the number of random features. We firstly define necessary symbols and state the theorem.

Let $s(x_i, x_j) = y_i y_j$ be a label similarity function, where $|y_i| \leq 1$. We use s_{ij} to denote $s(x_i, x_j)$ interchangeably. Given a kernel k, we define the true and empirical alignment functions as,

$$\begin{aligned} T(k) &= \mathbb{E}\left[s(x,x')k(x,x')\right] \\ \widehat{T}(k) &= \tfrac{1}{n(n-1)} \textstyle\sum_{i\neq j} s_{ij} k(x_i, x_j). \end{aligned}$$

In the following, we abuse the notation k_ψ to be k_h for ease of illustration. Recall the definitions of $k_h(x,x') = \langle \varphi_h(x), \varphi_h(x') \rangle$ and $\widehat{k}_h(x,x') = \widehat{\varphi}_h(x)^\top \widehat{\varphi}_h(x')$. We define two hypothesis sets

$$\begin{aligned} \mathcal{F}_\mathcal{H} &= \{f(x) = \langle w, \varphi_h(x) \rangle_H | h \in \mathcal{H}, \langle w, w \rangle \leq 1\} \\ \widehat{\mathcal{F}}^m_\mathcal{H} &= \{f(x) = w^\top \widehat{\varphi}_h(x) | h \in \mathcal{H}, \|w\| \leq 1, w \in \mathbb{R}^m\}. \end{aligned}$$

Definition 24. *(Rademacher's Complexity) Given a hypothesis set $\mathcal{F}$, where $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ if $f \in \mathcal{F}$, and a fixed sample $X = \{x_1, \ldots, x_n\}$, the empirical Rademacher's complexity of $\mathcal{F}$ is defined as*

$$\mathfrak{R}^n_X(\mathcal{F}) = \frac{1}{n} \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{F}} \sum_{i=1}^n \sigma_i f(x_i) \right],$$

where σ are n i.i.d. Rademacher random variables.

We then have the following theorems showing that the consistency guarantee depends on the complexity of the function class induced by IKL as well as the number of random features. The proof can be found in Section 7.6.3.

Theorem 25. *(Consistency) Let $\widehat{h} = \arg\max_{h\in\mathcal{H}} \widehat{T}(\widehat{k}_h)$, with i.i.d, samples $\{\nu_i\}_{i=1}^m$ drawn from $\mathbb{P}(\nu)$. With probability at least $1 - 3\delta$, we have $|T(k_{\widehat{h}}) - \sup_{h\in\mathcal{H}} T(k_h)| \leq$*

$$2\mathbb{E}_X \left[\mathfrak{R}^{n-1}_X(\mathcal{F}_\mathcal{H}) + \mathfrak{R}^{n-1}_X(\widehat{\mathcal{F}}^m_\mathcal{H}) \right] + \sqrt{\frac{8 \log \frac{1}{\delta}}{n}} + \sqrt{\frac{2 \log \frac{4}{\delta}}{m}}.$$

Applying Cortes et al. (2010), We also have a generalization bound, which depends on the number of training data n, the number of random features m and the Rademacher complexity of IKL kernel, as shown below.

Theorem 26. *(Generalization (Cortes et al., 2010)) Define the true and empirical misclassification for a classifier f as $R(f) = \mathbb{P}(Yf(X) < 0)$ and $\widehat{R}_\gamma(h) = \frac{1}{n}\sum_{i=1}^n \min\{1, [1 - yf(x_i)/\gamma]_+\}$.*

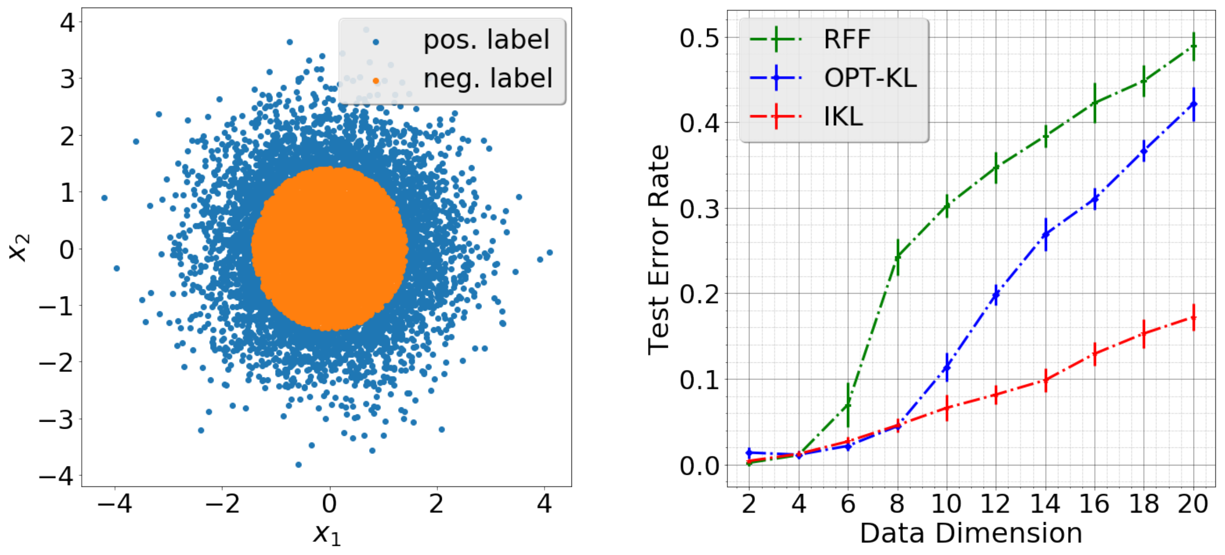


Figure 7.6: Left figure is training examples when $d = 2$. Right figure is the classification error v.s. data dimension.

Then

$$\sup_{f \in \widehat{\mathcal{F}}_{\mathcal{H}}} \{R(f) - \widehat{R}_\gamma(f)\} \leq \frac{2}{\gamma}\mathfrak{R}_X^n(\widehat{\mathcal{F}}_{\mathcal{H}}) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2n}}$$

with probability at least $1 - \delta$.

The Rademacher complexity $\mathfrak{R}_X^n(\mathcal{F}_{\mathcal{H}})$, for example, can be $1/\sqrt{n}$ or even $1/n$ for kernels with different bounding conditions (Cortes et al., 2013). We would expect worse rates for more powerful kernels. It suggests the trade-off between consistency/generalization and using powerful kernels parametrized by neural networks.

7.4.2 Empirical Study

We evaluate the proposed IKL on both synthetic and benchmark binary classification tasks. For IKL, $\mathbb{P}(\nu)$ is standard Normal and h_ψ is a 3-layer MLP for all experiments. The number of random features m to train h_ψ in Algorithm 5 is fixed to be 64. Other experiment details are described in Section 7.6.5.

Kernel learning with a poor choice of $\mathbb{P}_k(\omega)$ We generate $\{x_i\}_{i=1}^n \sim \mathcal{N}(0, I_d)$ with $y_i = \text{sign}(\|x\|_2 - \sqrt{d})$, where d is the data dimension. A two dimensional example is shown in Figure 7.6. Competitive baselines include random features (RFF) (Rahimi and Recht, 2007) as well as OPT-KL (Sinha and Duchi, 2016). In the experiments, we fix $M = 256$ in RKS for all algorithms. Since Gaussian kernels with a bandwidth $\sigma = 1$ is known to be ill-suited for this task (Sinha and Duchi, 2016), we directly use random features from it for RFF and OPT-KL. Similarly, we set $\mathbb{P}(\nu)$ to be standard normal distribution as well.

The test error for different data dimension $d = \{2, 4, \dots, 18, 20\}$ is shown in Figure 7.6. Note that RFF is competitive with OPT-KL and IKL when d is small ($d \leq 6$), while its performance degrades rapidly as d increases, which is consistent with the observation in Sinha and Duchi (2016). More discussion of the reason of failure can be referred to Sinha and Duchi (2016). On the other hand, although using standard normal distributions as spectral distributions is ill-suited for this task, both OPT-KL and IKL can adapt with data and learn to transform it into effective kernels and result in slower degradation with d.

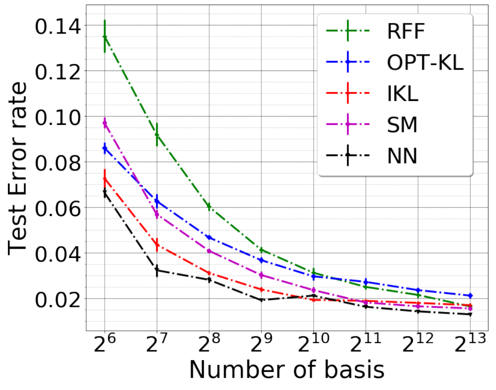
Note that OPT-KL learns the *sparse* weights on the sampled random features ($M = 256$). However, the sampled random features can fail to contain informative ones, especially in high dimension (Bullins et al., 2018). Thus, when using limited amount of random features, OPT-IKL may result in worse performance than IKL in the high dimensional regime in Figure 7.6.

Performance on benchmark datasets Next, we evaluate our IKL framework on standard benchmark binary classification tasks. Challenging label pairs are chosen from MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky and Hinton, 2009) datasets; each task consists of 10000 training and 2000 test examples. For all datasets, raw pixels are used as the feature representation. We set the bandwidth of RBF kernel by the median heuristic. We also compare with Wilson and Adams (2013), the spectral mixture (SM) kernel, which uses Gaussian mixture to learn spectral density and can be seen as an explicit generative model counterpart of IKL. Also, SM kernel is a MKL variant with linear combination (Gönen and Alpaydın, 2011). In addition, we consider the joint training of random features and model parameters, which can be treated as a two-layer neural network (NN) and serve as the lower bound of error for comparing different kernel learning algorithms.

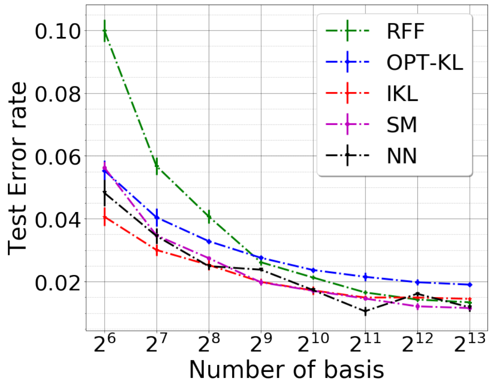
The test error versus different $M = \{2^6, 2^7, \dots, 2^{13}\}$ in the second stage is shown in Figure 7.7. First, in light of computation efficiency, SM and the proposed IKL only sample $m = 64$ random features in each iteration in the first stage, and draws a different number of basis M from the learned $h_\psi(\nu)$ for the second stage. OPT-KL, on the contrary, the random features used in training and testing should be the same. Therefore, OPT-IKL needs to deal with M random features in the training. It brings computation concern when M is large. In addition, IKL demonstrates improvement over the representative kernel learning method OPT-KL, especially significant on the challenging datasets such as CIFAR-10. In some cases, IKL almost reaches the performance of NN, such as MNIST, while OPT-KL degrades to RFF except for a small number of basis ($M = 2^6$). This illustrates the effectiveness of learning kernel spectral distribution via the implicit generative model h_ψ. Also, IKL outperforms SM, which is consistent with the finding in Section 7.2 that IKL can learn more complicated spectral distributions than simple mixture models (SM).

7.5 Summary

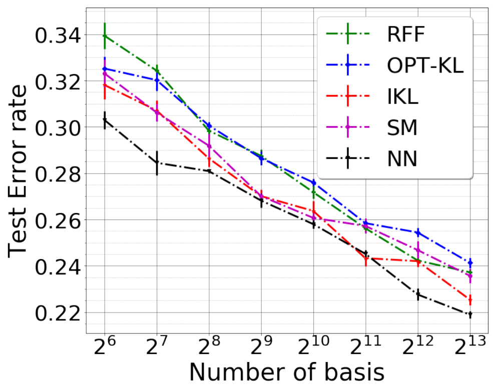
We propose a generic kernel learning algorithm, IKL, which learns sampling processes of kernel spectral distributions via IGMs. An immediate application is using the improved



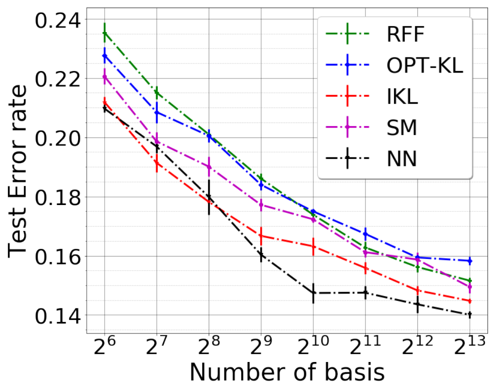
(a) MNIST (4-9)



(b) MNIST (5-6)



(c) CIFAR-10 (auto-truck)



(d) CIFAR-10 (plane-bird)

Figure 7.7: Test error rate versus number of basis in second stage on benchmark binary classification tasks. We report mean and standard deviation over five runs. Our method (IKL) is compared with RFF (Rahimi and Recht, 2009), OPT-KL (Sinha and Duchi, 2016), SM (Wilson and Adams, 2013) and the end-to-end training MLP (NN).

kernel via IGMs to help learning IGMs. We also demonstrate broader usages of IKL on traditional supervised learning tasks with Random Kitchen Sinks (RKS). For these two tasks, the conditions and guarantees of IKL are studied. Empirical studies show IKL is better than or competitive with the state-of-the-art kernel learning algorithms. It proves IKL can learn to transform $\mathbb{P}(\nu)$ into effective kernels even if $\mathbb{P}(\nu)$ is less favorable to the task.

We note that the preliminary idea of IKL is mentioned in Băzăvan et al. (2012), but they ended up with a algorithm that directly optimizes sampled random features (RF), which has many follow-up works (e.g. Bullins et al. (2018); Sinha and Duchi (2016)). The major difference is, by learning the transformation function h_ψ, the RF used in training and evaluation can be different. This flexibility allows a simple training algorithm (SGD) and does not require to keep learned features. In our studies on GAN and RKS, we show using a simple MLP can already achieve better or competitive performance with those works, which suggest IKL can be a new direction for kernel learning and worth more studies.

We highlight that IKL is not conflict with existing works but can be combined with them. In Section 7.2, we show combining IKL with kernel learning via embedding (Wilson et al., 2016) and mixture of spectral distributions (Wilson and Adams, 2013). Therefore, in addition to the examples shown in Section 7.2 and Section 7.4, IKL is directly applicable to many existing works with kernel learning via embedding (e.g. Al-Shedivat et al. (2017); Arbel et al. (2018); Chang et al. (2019); Dai et al. (2014); Jean et al. (2018); Li and Póczos (2016); Wilson et al. (2016)). A possible extension is combining with Bayesian inference (Oliva et al., 2016) under the framework similar to Saatchi and Wilson (2017). The learned sampler from IKL can possibly provide an easier way to do Bayesian inference via sampling.

7.6 Appendix

7.6.1 Proof of Theorem 22

The proof is the straightforward extension of Section 2.5.1 and 2.5.2. We still include the full proof for completeness.

First, we prove $\|k_\psi(f_\phi(x),\cdot)-k_\psi(f_\phi(y),\cdot)\|_{\mathcal{H}_K}$ is Lipschitz. By definition of RKHS, we know that $\|k_\psi(f_\phi(x),\cdot)-k_\psi(f_\phi(y),\cdot)\|_{\mathcal{H}_K}=2(1-k_\psi(f_\phi(x),f_\phi(y)))$. Also, since $k_\psi(0)=1$ and $k_\psi(0)-k_\psi(x,x')\leq L_k\|0-(x-x')\|$ (Lipschtiz assumption of k_ψ), we have

$$\|k_\psi(f_\phi(x),\cdot)-k_\psi(f_\phi(y),\cdot)\|_{\mathcal{H}_K}\leq 2L_k\|f_\phi(x)-f_\phi(y)\|\leq 2L_kL\|x-y\|,$$

where the last inequality is since f_ϕ is also a Lipschitz function with Lipschitz constant L.

The other direction, $\max_{\psi,\phi}M_{\psi,\phi}(\mathbb{P},\mathbb{P}_n)\to 0$ then $\mathbb{P}_n\xrightarrow{D}\mathbb{P}$, is by assuming there exists ψ' and ϕ' such that h_ψ and f_ϕ are identity functions. Then all the remaining claims follows Section 2.5.2.

Next, we show the proof of continuity. We are going to show

$$\max_{\psi,\phi}M_{\psi,\phi}(\mathbb{P}_\mathcal{X},\mathbb{Q}_\theta)=\mathbb{E}_{x,x'}[k_{\psi,\phi}(x,x')]-2\mathbb{E}_{x,z}[k_{\psi,\phi}(x,g_\theta(z))]+\mathbb{E}_{z,z'}[k_{\psi,\phi}(g_\theta(z'),g_\theta(z))] \quad (7.9)$$

is differentiable with respect to ϕ almost everywhere by using Lemma 6. We fist show $\mathbb{E}_{z,z'}[k_{\psi,\phi}(g_\theta(z'),g_\theta(z))]$ in (7.9) is locally Lipschitz in θ. By definition, $k_{\psi,\phi}(x,x')=k_\psi(f_\phi(x)-f_\phi(x'))$, therefore,

$$\begin{aligned}
&\mathbb{E}_{x,x'}\left[k_{\psi,\phi}\Big(g_\theta(z),g_\theta(z')\Big)-k_{\psi,\phi}\Big(g_{\theta'}(z),g_{\theta'}(z')\Big)\right]\\
=&\mathbb{E}_{z,z'}\left[k_\psi\Big(f_\phi\big(g_\theta(z)\big)-f_\phi\big(g_\theta(z')\big)\Big)\right]-\mathbb{E}_{z,z'}\left[k_\psi\Big(f_\phi\big(g_{\theta'}(z)\big)-f_\phi\big(g_{\theta'}(z')\big)\Big)\right]\\
\leq&\mathbb{E}_{z,z'}\left[L_k\Big\|f_\phi\big(g_\theta(z)\big)-f_\phi\big(g_\theta(z')\big)-f_\phi\big(g_{\theta'}(z)\big)+f_\phi\big(g_{\theta'}(z')\big)\Big\|\right]\\
\leq&\mathbb{E}_{z,z'}\Big[L_kL(\theta,z)\|\theta-\theta'\|+L_kL(\theta,z')\|\theta-\theta'\|\Big]\\
=&2L_k\mathbb{E}_z\big[L(\theta,z)\big]\|\theta-\theta'\|.
\end{aligned}$$

The first inequality is followed by the assumption that k is locally Lipschitz in (x, x'), with a upper bound L_k for Lipschitz constants. By Assumption 21, $\mathbb{E}_z\big[L(\theta, z)\big] < \infty$, we prove $\mathbb{E}_{z,z'}\Big[k_\psi\big(f_\phi(g_\theta(z)) - f_\phi(g_\theta(z'))\big)\Big]$ is locally Lipschitz. The similar argument is applicable to other terms in (7.9); therefore, (7.9) is locally Lipschitz in θ.

Last, with the compactness assumption on Φ and Ψ, and differentiable assumption on $M_{\psi,\phi}(\mathbb{P}_{\mathcal{X}}, \mathbb{Q}_\theta)$, applying Lemma 6 proves Theorem 22.

7.6.2 Proof of Lemma 23

Without loss of the generality, we can rewrite the kernel function as $k_\psi(t) = \mathbb{E}_\nu\Big[\cos\big(h_\psi(\nu)^\top t\big)\Big]$, where t is bounded. We then have

$$\begin{aligned}
\|\nabla_t k_\psi(t)\| &= \Big\|\mathbb{E}_\nu\Big[\sin\big(h_\psi(\nu)^\top t\big) h_\psi(\nu)\Big]\Big\| \\
&\leq \mathbb{E}_\nu\Big[\big|\sin(h_\psi(\nu)^\top t)\big| \times \|h_\psi(\nu)\|\Big] \\
&\leq \mathbb{E}_\nu\Big[\|t\| \|h_\psi(\nu)\|^2\Big]
\end{aligned}$$

The last inequality follows by $|\sin(x)| < |x|$. Since t is bounded, if $\mathbb{E}_\nu[\|h_\psi(\nu)\|^2] < \infty$, there exist a constant L such that $\|\nabla_t k_\psi(t)\| < L, \forall t$.

By mean value theorem, for any t and t', there exists $s = \alpha t + (1-\alpha)t'$, where $\alpha \in [0, 1]$, such that

$$k_\psi(t) - k_\psi(t') = \nabla_s k_\psi(s)^\top (t - t').$$

Combining with $\|\nabla_t k_\psi(t)\| < L, \forall t$, we prove

$$k_\psi(t) - k_\psi(t') \leq L\|t - t'\|.$$

7.6.3 Proof of Theorem 25

We first prove two Lemmas.

Lemma 27. *(Consistency with respect to data) With probability at least* $1 - \delta$*, we have*

$$\sup_{h \in \mathcal{H}} |\widehat{T}(k_h) - T(k_h)| \leq 2\mathbb{E}_X\left[\mathfrak{R}_X^{n-1}(\mathcal{F}_\mathcal{H})\right] + \sqrt{\frac{2}{n}\log\frac{1}{\delta}}$$

Proof. Define

$$\rho(x_1, \ldots, x_n) = \sup_{h \in \mathcal{H}} |\widehat{T}(k_h) - T(k_h)|,$$

since $|k_h(x, x')| \leq 1$, it is clearly

$$\begin{aligned}
\sup_{x_1,\ldots,x_i,x_i',\ldots,x_n} \quad & |\rho(x_1, \ldots, x_i, \ldots x_n) - \\
& \rho(x_1, \ldots, x_i', \ldots x_n)| \leq \tfrac{2}{n}.
\end{aligned}$$

Applying McDiarmids Inequality, we get

$$\mathbb{P}\left(\rho(x_1, \ldots, x_n) - \mathbb{E}[\rho(x_1, \ldots, x_n)] \geq \epsilon\right) \leq \exp\left(\frac{-n\epsilon^2}{2}\right).$$

By Lemma 28, we can bound

$$\mathbb{E}[\rho(x_1,\dots,x_n)] \le 2\mathbb{E}_X\left[\mathfrak{R}_X^{n-1}(\mathcal{F}_\mathcal{H})\right]$$

and finish the proof. □

Lemma 28. *Given $X=\{x_1,\dots,x_n\}$, define*

$$\rho(x_1,\dots,x_n)=\sup_{h\in\mathcal{H}}|\widehat{T}(k_h)-T(k_h)|,$$

we have

$$\mathbb{E}\left[\rho(x_1,\dots,x_n)\right] \le 2\mathbb{E}_X\left[\mathfrak{R}_X^{n-1}(\mathcal{F}_\mathcal{H})\right],$$

Proof. The proof is closely followed by Dziugaite et al. (2015). Given h, we first define $t_h(x,x')=s(x,x')k_h(x,x')$ as a new kernel function to simplify the notations. We are then able to write

$$\begin{aligned}
&\mathbb{E}\left[\rho(x_1,\dots,x_n)\right]\\
=\;& \mathbb{E}_X\left[\sup_{h\in\mathcal{H}}\left|\mathbb{E}\left[t_h(z,z')\right]-\frac{1}{n(n-1)}\sum_{i\neq j}t_h(x_i,x_j)\right|\right]\\
\le\;& \mathbb{E}_{X,Z}\left[\sup_{h\in\mathcal{H}}\left|\frac{1}{n(n-1)}\sum_{i\neq j}(t_h(z_i,z_j)-t_h(x_i,x_j))\right|\right]
\end{aligned}$$

by using Jensen's inequality. Utilizing the conditional expectation and introducing the Rademacher random variables $\{\sigma_i\}_{i=1}^{n-1}$, we can write the above bound to be

$$\begin{aligned}
&\frac{1}{n}\sum_i \mathbb{E}_{X_{-i},Z_{-i}}\mathbb{E}_{x_i,z_i}\left[\sup_{h\in\mathcal{H}}\left|\frac{\sum_{i\neq j}t_h(z_i,z_j)-t_h(x_i,x_j)}{n-1}\right|\right]\\
=\;&\mathbb{E}_{X,Z}\mathbb{E}_{X',Z',\sigma}\left[\sup_{h\in\mathcal{H}}\left|\frac{1}{n-1}\sum_{i=1}^{n-1}\sigma_i(t_h(z',z_n)-t_h(x',x_n))\right|\right]
\end{aligned}\tag{7.10}$$

The equality follows by $X-X'$ and $-(X-X')$ has the same distributions if X and X' are independent samples from the same distribution. Last, we can bound it by

$$\begin{aligned}
\le\;& \mathbb{E}_X\mathbb{E}_{\sigma,X'}\left[\sup_{h\in\mathcal{H}}\left|\frac{2}{n-1}\sum_{i=1}^{n-1}\sigma_i t_h(x',x_i)\right|\right]\\
\le\;& \mathbb{E}_X\mathbb{E}_{\sigma}\left[\sup_{f\in\mathcal{F}_{k_\mathcal{H}}}\left|\frac{2}{n-1}\sum_{i=1}^{n-1}\sigma_i f(x_i)\right|\right]\\
=\;& 2\mathbb{E}_X[\mathfrak{R}_X^{n-1}(\mathcal{F}_\mathcal{H})]
\end{aligned}$$

The second inequality follows by $s(x,x')\varphi(x')\in\mathcal{F}$ since $|s(x,x')|\le 1$.

□

Lemma 29. *(Consistency with respect to sampling random features) With probability* $1-\delta$, *we have*

$$|\sup_{h\in\mathcal{H}} \widehat{T}(k_h) - \sup_{h\in\mathcal{H}} \widehat{T}(\widehat{k}_h)| \le \sqrt{\frac{2\log\frac{4}{\delta}}{m}}$$

Proof. Let the optimal solutions be

$$\begin{aligned} h^* &= \arg\max_{h\in\mathcal{H}} \widehat{T}(k_h) \\ \widehat{h} &= \arg\max_{h\in\mathcal{H}} \widehat{T}(\widehat{k}_h), \end{aligned}$$

By definition,

$$\begin{aligned} & \widehat{T}(\widehat{k}_h) \\ = & \frac{1}{n(n-1)}\sum_{i\neq j} s_{ij}\left(\frac{1}{m}\sum_{k=1}^{m}\cos(h(\nu_k)^\top(x_i-x_j))\right) \\ = & \frac{1}{m}\sum_{k=1}^{m}\left(\frac{1}{n(n-1)}s_{ij}\cos(h(\nu_k)^\top(x_i-x_j))\right) \end{aligned}$$

It is true that $|\frac{1}{n(n-1)}s_{ij}\cos(h(\nu_k)^\top(x_i-x_j)| \le 1$ since $|s_{ij}|<1$ and $|\cos(x)|<1$. we then have

$$\begin{aligned} & \mathbb{P}(|\sup_{h\in\mathcal{H}} \widehat{T}(k_h) - \sup_{h\in\mathcal{H}} \widehat{T}(\widehat{k}_h)| > \epsilon) \\ \le & \mathbb{P}(|\widehat{T}(k_{h^*}) - \widehat{T}(\widehat{k}_{h^*})| > \epsilon) + \mathbb{P}(|\widehat{T}(k_{\widehat{h}}) - \widehat{T}(\widehat{k}_{\widehat{h}})| > \epsilon) \\ \le & 4\exp\left(-\frac{m\epsilon^2}{2}\right), \end{aligned}$$

where the last inequality follows from the Hoeffding's inequality. □

With Lemma 27 and Lemma 29, we are ready to prove Theorem 25. We can decompose

$$\begin{aligned} & |T(\widehat{k}_{\widehat{h}}) - \sup_{h\in\mathcal{H}} T(k_h)| \\ \le & |\sup_{h\in\mathcal{H}} T(k_h) - \sup_{h\in\mathcal{H}} \widehat{T}(k_h)| + |\sup_{h\in\mathcal{H}} \widehat{T}(k_h) - \widehat{T}(\widehat{k}_{\widehat{h}})| + |\widehat{T}(\widehat{k}_{\widehat{h}}) - T(\widehat{k}_{\widehat{h}})| \\ \le & \sup_{h\in\mathcal{H}} |T(k_h) - \widehat{T}(k_h)| + |\sup_{h\in\mathcal{H}} \widehat{T}(k_h) - \widehat{T}(\widehat{k}_{\widehat{h}})| + \sup_{h\in\mathcal{H}} |\widehat{T}(\widehat{k}_h) - T(\widehat{k}_h)| \end{aligned}$$

We then bound the first and third terms by Lemma 27 and the second term by Lemma 29. Last, using a union bound completes the proof.

7.6.4 Hyperparameters of GAN Studies

For Gaussian kernels, we use $\sigma_q = \{1,2,4,8,16\}$ for images and $\sigma_q = \{0.5,1,2,4,8\}$ for text; for RQ kernels, we use $\alpha_q = \{0.2,0.5,1,2,5\}$ for images and $\alpha_q = \{0.04,0.1,0.2,0.4,1\}$ for text. We used Adam as optimizer. The learning rate for training both f_ϕ and g_θ is 0.0005 and 0.0001 for image and text experiments, respectively. The batch size B is 64. We set hyperparameter n_c for updating critic to be $n_c = 5$ and $n_c = 10$ for CIFAR10 and Google Billion Word datasets. The learning rate of of h_ψ for Adam is 10^{-6}.

7.6.5 Hyperparameters of Random Kitchen Sinks Studies

For OPT-KL, we use the code provided by Sinha and Duchi (2016)[3]. We tune the hyperparameter $\rho = \{1.25, 1.5, 2, 4, 16, 64\}$ on the validation set. For RFF, OPT-KL, and IKL, the linear classifier is Logistic Regression Fan et al. (2008)[4], as to make reasonable comparison with MLP. We use 3-fold cross validation to select the best C on training set and present the error rate on test set. For CIFAR-10 and MNIST, we normalize data to be zero mean and one standard deviation in each feature dimension. The learning rate for Adam is 10^{-6}. We follow Bullins et al. (2018) to use early stopping when performance on validation set does not gain.

[3] https://github.com/amansinha/learning-kernels

[4] https://github.com/cjlin1/liblinear

Chapter 8

Learning Proximal Operators for Solving Linear Inverse Problems

At the heart of many image processing tasks is a linear inverse problem, where the goal is to reconstruct an image $x \in \mathbb{R}^d$ from a set of measurements $y \in \mathbb{R}^m$ of the form $y = Ax + \epsilon$, where $A \in \mathbb{R}^{m \times d}$ is the measurement operator and $\epsilon \in \mathbb{R}^m$ is the noise. For example, in image inpainting, A is the linear operation of applying a pixelwise mask to the image x. In super-resolution, A downsamples high-resolution images. In compressive sensing, A is a short-fat matrix with fewer rows than columns and is typically a random sub-Gaussian or a sub-sampled orthonormal matrix. Linear inverse problems are often underdetermined, i.e., they involve fewer measurements than unknowns. Such underdetermined systems are difficult to solve since the operator A has a non-trivial null space and there are an infinite number of feasible solutions; however, only a few of the feasible solutions are valid natural images. There are two broad approaches for solving linear underdetermined problems, including hand-crafted regularizations with our priors and learning-based methods.

Hand-Designed Signal Regularizations The first approach regularizes the inverse problem with signal regularizations, which encode our prior knowledges of true solutions from the infinite set of feasible solutions (Chan et al., 2006; Dong et al., 2011; Donoho, 1995; Mairal et al., 2008; Portilla et al., 2003). It is usually in the form

$$\min_x \frac{1}{2} \|y - Ax\|_2^2 + \lambda h(x), \tag{8.1}$$

where $h : \mathbb{R}^d \to \mathbb{R}$ is the signal regularization and λ is the non-negative weighting term. Signal regularizations constraining the sparsity of x in some transformation domain have been widely used in literature. For example, since images are usually sparse after wavelet transformation or after taking gradient operations, a signal prior h can be formulated as $h(x) = \|Wx\|_1$, where W is a operator representing either wavelet transform, taking image gradient, or other hand-designed linear operation that produces sparse features from images (Donoho et al., 1998). Using signal regularizations of ℓ_1-norms provides two advantages. First, it forms a convex optimization problem and provides global optimality. The optimization problem can be solved efficiently with a variety of algorithms for convex optimization. Second, ℓ_1 regularizations enjoy many theoretical guarantees, thanks to results in compressive sensing (Candes et al., 2006). For example, if the linear operator A satisfies conditions like the restricted isometry property and Wx is sufficiently sparse, the optimization problem (8.1) provides the sparsest solution. Despite their algorithmic and theoretical benefits, hand-designed regularizations are often too generic to constrain the solution set of the inverse problem (8.1) to be natural images — we can easily generate noise-like signals that have sparse wavelet coefficients or gradients.

Learning-based Method. The second approach learns a direct mapping from the linear measurement y to the solution x, with the help of large training datasets and deep neural networks. Such methods have achieved promising performance in many challenging image inverse problems like super-resolution (Dong et al., 2014; Ledig et al., 2017), inpainting (Pathak et al., 2016), compressive sensing (Kulkarni et al., 2016; Mousavi and Baraniuk, 2017; Mousavi et al., 2015), and image debluring (Xu et al., 2014). Recently, state-of-the-art performance in many challenging problems (Ledig et al., 2017; Pathak et al., 2016) are acquired via modeling conditional distributions $\mathbb{P}(x|y)$ via deep IGMs. Despite their superior performance, these methods are designed for specific problems and usually cannot solve other problems without retraining the transformation function — even when the problems are similar. For example, a $4\times$-super-resolution network cannot be easily readapted to solve $2\times$ super-resolution problems; a compressive sensing network for Gaussian random measurements is not applicable to sub-sampled Hadamard measurements. Training a new network for every single measurement operator is a wasteful proposition.

One network to solve them all. In comparison, traditional methods using hand-designed signal regularizations can solve any linear inverse problems but they often have poorer performance on an individual problem. Clearly, a middle ground between these two classes of methods is needed. We ask the following question:

> *if we have a large dataset of natural images, can we learn from the dataset a signal priors that can deal with* ***any*** *linear inverse problem involving images?*

Intuitively, the ideal prior should be the distribution of the given image data $\mathbb{P}_{\mathcal{X}}$. The corresponding signal regularization can significantly lower the cost to incorporate inverse algorithms into consumer products, for example, via the form of specialized hardware design. To answer this question, we observe that in optimization algorithms for solving linear inverse problems, signal regularizations usually appear in the form of proximal operators. Geometrically, the proximal operator *projects* the current estimate closer to the feasible sets (natural images) constrained by the signal regularization; statistically, the proximal operator *transforms* the current estimate into the support of $\mathbb{P}_{\mathcal{X}}$. Thus, we propose to learn an *IGM proximal operator* such that the transformed (projected) images follow the prior distribution of data. Once learned, the same network can be integrated into many standard optimization frameworks for solving arbitrary linear inverse problems of natural images.

8.1 One Network to Solve Them All

Signal priors play an important role in constraining underdetermined inverse problems. As mentioned earlier, traditional priors constraining the sparsity of signals in gradient or wavelet bases are often too generic, in that we can easily create non-image signals satisfying these priors. Instead of using traditional priors of images, we propose to learn a prior from a large image dataset. Since the prior is learned directly from the dataset, it is tailored to the statistics of images in the dataset and, in principle, provide stronger regularization to the inverse problem. In addition, similar to traditional signal regularizations, the learned image priors can be used to solve any linear inverse problems pertaining to images.

8.1.1 Problem formulation

The proposed framework is motivated by the optimization technique, alternating direction method of multipliers (ADMM) (Boyd et al., 2011), that is widely used to solve linear inverse problems as defined in Eq. (8.1). A typical first step in ADMM is to separate a complicated objective into several simpler ones by variable splitting, i.e., introducing an additional variable z that is constrained to be equal to x. This gives us the following optimization problem:

$$\min_{x,z} \frac{1}{2} \|y - Az\|_2^2 + \lambda\, h(x) \qquad \text{s.t.} \qquad x = z, \tag{8.2}$$

that is equivalent to the original problem (8.1). The scaled form of the augmented Lagrangian of Eq. (8.2) can be written as

$$\mathcal{L}(x, z, u) = \frac{1}{2}\|y - Az\|_2^2 + \lambda\, h(x) + \frac{\rho}{2} \|x - z + u\|_2^2 ,$$

where $\rho > 0$ is the penalty parameter of the constraint $x = z$, and u represents the dual variables divided by ρ. By alternately optimizing $\mathcal{L}(x, z, u)$ over x, z, and u, ADMM is composed of the following steps:

$$x^{(k+1)} \leftarrow \operatorname*{argmin}_x \frac{\rho}{2} \left\|x - z^{(k)} + u^{(k)}\right\|_2^2 + \lambda\, h(x) \tag{8.3}$$

$$z^{(k+1)} \leftarrow \operatorname*{argmin}_z \frac{1}{2} \|y - Az\|_2^2 + \frac{\rho}{2} \left\|x^{(k+1)} - z + u^{(k)}\right\|_2^2 \tag{8.4}$$

$$u^{(k+1)} \leftarrow u^{(k)} + x^{(k+1)} - z^{(k+1)}.$$

The update of z in Eq. (8.4) is a least squares problem which can be solved efficiently via conjugate gradient descent. The update of x in (8.3) is the proximal operator of the signal regularization h with penalty $\frac{\rho}{\lambda}$, denoted as $\mathbf{prox}_{h,\frac{\rho}{\lambda}}(b)$, where $b{=}z^{(k)}{-}u^{(k)}$. When the signal regularization uses ℓ_1-norm, the proximal operator is simply a soft-thresholding on b. Notice that the ADMM algorithm separates the signal regularization h from the linear operator A. This enables us to learn an *implicit prior* that can be used with any linear operator.

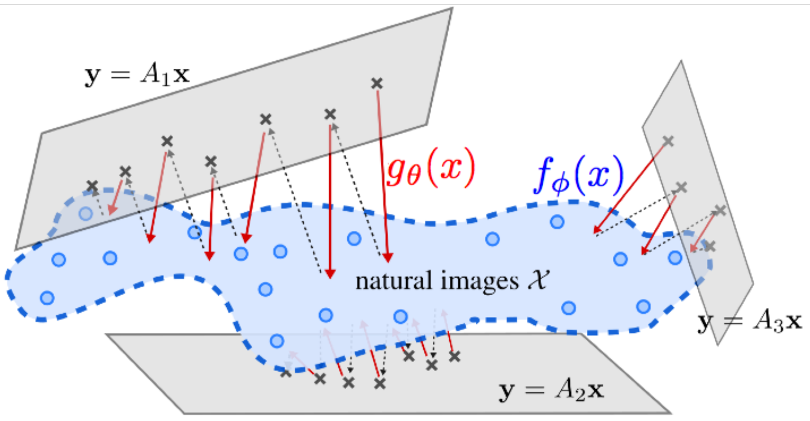

$\mathbf{y} = A_1\mathbf{x}$
$g_\theta(x)$
$f_\phi(x)$
natural images $\mathcal{X}$
$\mathbf{y} = A_3\mathbf{x}$
$\mathbf{y} = A_2\mathbf{x}$

Figure 8.1: Given a large image dataset, the proposed framework learns a classifier f_ϕ that fits the support of the natural image set. Based on f_ϕ, the transformation, which is an IGM projection network g_θ: $\mathbb{R}^d \to \mathbb{R}^d$, is trained to fit the proximal operator of f_ϕ, which enables one to solve a variety of linear inverse problems using ADMM.

8.1.2 Proximal Operations as Transformations via IGMs

Since the regularization (prior) only appears in the form of proximal operators in ADMM, instead of explicitly learning a signal prior $\mathbb{P}_\mathcal{X}$, defining the corresponding h, and solving the proximal operator in each step of ADMM, we propose to directly learn a proximal operator g_θ, which is parametrized by θ. Let $\mathcal{X}$ represent the set of data images which follow the distribution $\mathbb{P}_\mathcal{X}$. The desired proximal operator g_θ has to project b into the support of $\mathbb{P}_\mathcal{X}$. In general, g_θ should be able to deal with any signal in $\mathbb{R}^d$ as b for Eq. (8.3), which is challenging. However, we can initialize the scaled dual variables u with zeros and $z^{(0)}$ with the pseudo-inverse of the least-square term. Since we initialize $u^0 = \mathbf{0}$, the input to the proximal operator $b^{(k)}=z^{(k)}-u^{(k)} = z^{(k)} + \sum_{i=1}^{k} \left(x^{(i)} - z^{(i)}\right) \approx z^{(k)}$ resembles an image. Thereby, even though it is in general difficult to fit a projection function from any signal in $\mathbb{R}^d$ to the natural image space, we expect that the desired projection function only needs to deal with inputs that are close to images, and we train the projection function with slightly perturbed images $\widetilde{x} = x + \epsilon$ from the dataset, where $x \sim \mathbb{P}_\mathcal{X}$. Therefore, we solve g_θ via

$$\min_\theta D(\mathbb{P}_\mathcal{X} \| \mathbb{Q}_\theta),$$

where D is a probability distances, and $\mathbb{Q}_\theta$ is the distribution of $g_\theta(\widetilde{x})$. The process can be viewed as learning an IGM for conditional generation. Given $\widetilde{x}$, the generator g_θ transforms $\widetilde{x}$ into $g_\theta(\widetilde{x}) \sim \mathbb{Q}_\theta$ such that $\mathbb{P}_\mathcal{X} \approx \mathbb{Q}_\theta$. Therefore, we call g_θ as an IGM projection network. We are then able to use a neural network f_ϕ to define powerful $D(\mathbb{P}_\mathcal{X} \| \mathbb{Q}_\theta)$ for distribution matching to train a transformation network as the projection operator.

Geometric Perspective. In addition to the conditional generation viewpoint, we can explain the proposed algorithm from a geometric perspective. The best signal regularization is the indicator function of $\mathcal{X}$, denoted as $\mathcal{I}_\mathcal{X}(\cdot)$, and its corresponding proximal operator $\mathbf{prox}_{\mathcal{I}_\mathcal{X},\rho}(b)$ is a projection operator that projects b onto $\mathcal{X}$ from the geometric perspective— or equivalently, finding a $x \in \mathcal{X}$ such that $\|x - b\|$ is minimized. However, we do not have the oracle indicator function $\mathcal{I}_\mathcal{X}(\cdot)$ in practice, so we cannot evaluate $\mathbf{prox}_{\mathcal{I}_\mathcal{X},\rho}(b)$ to solve the projection operation. Instead, we can approximate it by a classifier f_ϕ with a large dataset whose decision function approximates $\mathcal{I}_\mathcal{X}$ or we can treat f_ϕ as learning the support of $\mathbb{P}_\mathcal{X}$. GANs (Bengio, 2018). Based on the learned classifier f_ϕ, we can learn a projection function g_θ that maps a signal b to the set defined by the classifier. The learned projection function g_θ can then replace the proximal operator (8.3), and we simply update x via

$$x^{(k+1)} \leftarrow g_\theta(z^{(k)} - u^{(k)}). \tag{8.5}$$

An illustration of the idea is shown in Figure 8.1.

There are some caveats for this approach. When the support of $\mathbb{P}_\mathcal{X}$ or the decision function of the classifier f_ϕ is non-convex, the overall optimization becomes non-convex. For solving general non-convex optimization problems, the convergence result is not guaranteed. Based on the theorems for the convergence of non-convex ADMM (Wang et al., 2015), we provide the following theorem to the proposed ADMM framework.

Theorem 30. *Assume that the function g_θ solves the proximal operator* (8.3)*. If the gradient of $h(x)$ is Lipschitz continuous and with large enough ρ, the ADMM algorithm is guaranteed to attain a stationary point.*

The proof follows directly from Wang et al. (2015) and we omit the details here. Although Theorem 30 only guarantees convergence to stationary points instead of the optimal solution as other non-convex formulations, it ensures that the algorithm will not diverge after several iterations. On the other hand, constraining generators via Lipschitz constraints is also shown to lead to stable behaviors (Odena et al., 2018). The iterative update of ADMM can also be treated as a recurrent process as recurrent neural networks, where the projector is the RNN cell. Stabling RNN training via constraining Lipschitz constants of RNN cells is also extensively studied (e.g. Arjovsky et al. (2016)).

Lastly, we remark that techniques like denoising autoencoders learn projection-like operators and, in principle, can be used in place of a proximal operator; however, our empirical findings suggest that ignoring the projection cost $\|b - g_\theta(b)\|^2$ and simply minimizing the reconstruction loss $\|x_0 - g_\theta(b)\|^2$, where b is a perturbed image from x_0, leads to instability in the ADMM iterations.

8.1.3 Implementation details

Choice of activation function. According to Theorem 30, we need the gradient of ϕ to be Lipschitz continuous. Thus, we simply truncate the weights of the network after each iteration as Arjovsky et al. (2017) to bound the gradients of $\mathcal{D}$ w.r.t. x, and choose the smooth exponential linear unit (Clevert et al., 2016) as its activation function instead of rectified linear units.

Image perturbation. While adding Gaussian noise may be the simplest way to perturb an image, we found that the projection network will easily overfit the Gaussian noise and become a dedicated Gaussian denoiser. Since during the ADMM process, the inputs to the projection network, $z^{(k)} - u^{(k)}$, do not usually follow a Gaussian distribution, an overfitted projection network may fail to project the general signals produced by the ADMM process. To avoid overfitting, we generate perturbed images with two methods — adding Gaussian noise with spatially-varying standard deviations and smoothing the input images.

8.1.4 Relationship to other techniques

Many recent works solve linear inverse problems by unrolling the optimization process into the network architecture (Adler and Öktem, 2017; Borgerding and Schniter, 2016; Gregor and LeCun, 2010; Jin et al., 2017). Since the linear operator A is incorporated in the architecture, these networks are problem-specific. The proposed method is also similar to the denoising-based approximate message passing algorithm (Metzler et al., 2016) and plug-and-play priors (Venkatakrishnan et al., 2013), which replace the proximal operator with an image denoiser. Compared to denoising autoencoder, the projection network g_θ is encouraged to project perturbed images $x_0 + \epsilon$ to the closest x in $\mathcal{X}$, instead of the original image x_0. In our empirical experience, the difference helps stabilize the ADMM process.

Other related methods. Many concurrent works also propose to solve generic linear inverse problems by learning proximal operators (Meinhardt et al., 2017; Xiao et al., 2017). Meinhardt et al. (2017) replace the proximal operator with a denoising network. Xiao et al. (2017) use a modified multi-stage non-linear diffusion process (Chen et al., 2015) to learn the proximal operator.

Dave et al. (2017) and Bora et al. (2017) learn explicit generative models of natural images and solve linear inverse problems by performing maximum a posteriori inference. Their algorithms need to compute the gradient of the networks in each iteration, which can be computationally expensive when the networks are very deep and complex. In contrast, the proposed method directly provides the solution to the x-update (8.5) and is thus computationally efficient.

8.1.5 Limitations

Unlike traditional signal regularizations whose weights λ can be adjusted at the time of solving the optimization problem (8.1), the prior weight of the proposed framework is fixed once the projection network is trained. While an ideal projection operator should not be affected by the value of the prior weights, sometimes, it may be preferable to control the effect of the signal regularization to the solution. In our experiments, we find that adjusting ρ sometimes has similar effects as adjusting λ.

The convergence analysis of ADMM in Theorem 30 is based on the assumption that the projection network can provide global optimum of Eq. (8.3). However, in practice the optimality is not guaranteed. While there are convergence analyses with inexact proximal operators, the general properties are too complicated to analyze for deep neural networks. In practice, we find that for problems like pixelwise inpainting, compressive sensing, $2\times$ super-resolution and scattered inpainting the proposed framework converges gracefully, as

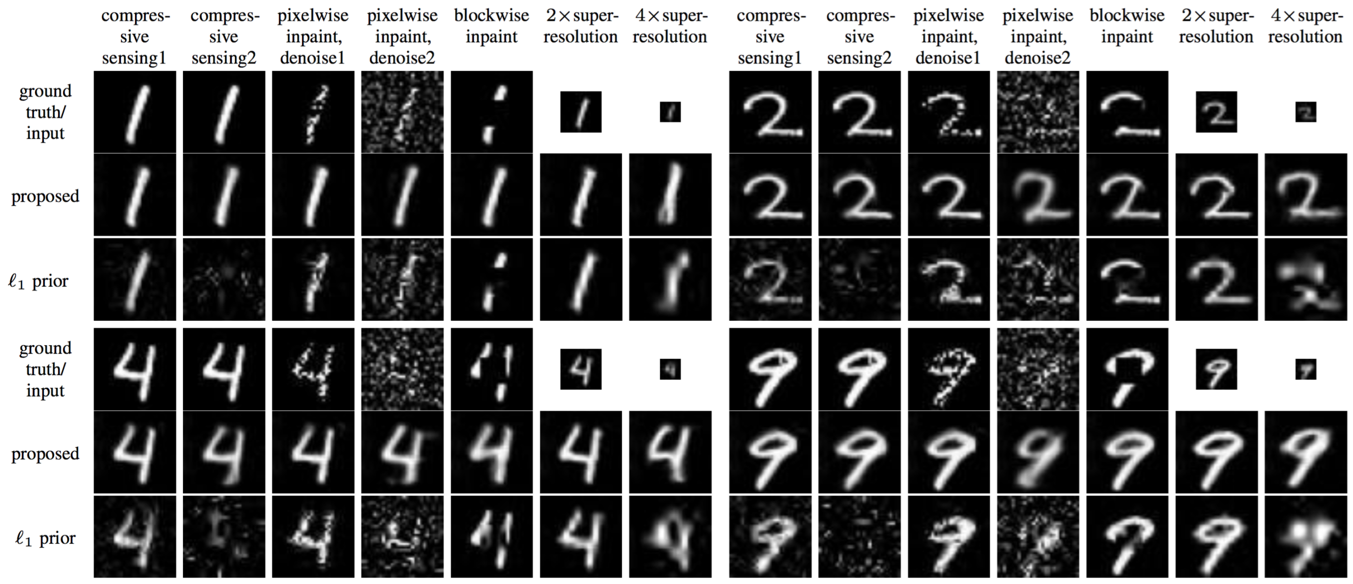


Figure 8.2: Results on MNIST dataset. Since the input of compressive sensing cannot be visualized, we show the ground truth instead. Compressive sensing 1 uses $\frac{m}{d} = 0.3$ and compressive sensing 2 uses $\frac{m}{d} = 0.03$. Pixelwise inpaint 1 drops 50% of the pixels, and pixelwise inpaint 2 drops 70% of the pixels and adds Gaussian noise with $\sigma = 0.3$. We use $\rho = 0.1$ for pixelwise inpainting and $\rho = 0.05$ for blockwise inpainting.

shown in Figure 8.6. For more challenging problems like image inpainting with large blocks and 4×-super-resolution on ImageNet dataset, we sometimes need to stop the ADMM procedure early (by monitoring the residual $\|x^{(k)} - z^{(k)}\|$).

8.2 Experiments

We evaluate the proposed framework on the MNIST (Loosli et al., 2007), MS-Celeb-1M (Guo et al., 2016), ImageNet (Russakovsky et al., 2015), and LabelMe (Russell et al., 2008). For each of the datasets, we perform the following tasks:

(i) *Compressive sensing.* We use $m \times d$ random Gaussian matrices of different compression ($\frac{m}{d}$) as the linear operator A. The images are vectorized into d-dimensional vectors x and multiplied with the random Gaussian matrices to form y.

(ii) *Pixelwise inpainting and denoising.* We randomly drop pixels (independent of channels) by filling zeros and add Gaussian noise with different standard deviations.

(iii) *Scattered inpainting.* We randomly drop 10 small blocks by filling zeros. Each block is of 10% width and height of the input.

(iv) *Blockwise inpainting.* We fill the center 30% region of the input images with zeros.

(v) *Super resolution.* We downsample the images into 50% and 25% of the original width and height using box-averaging algorithm.

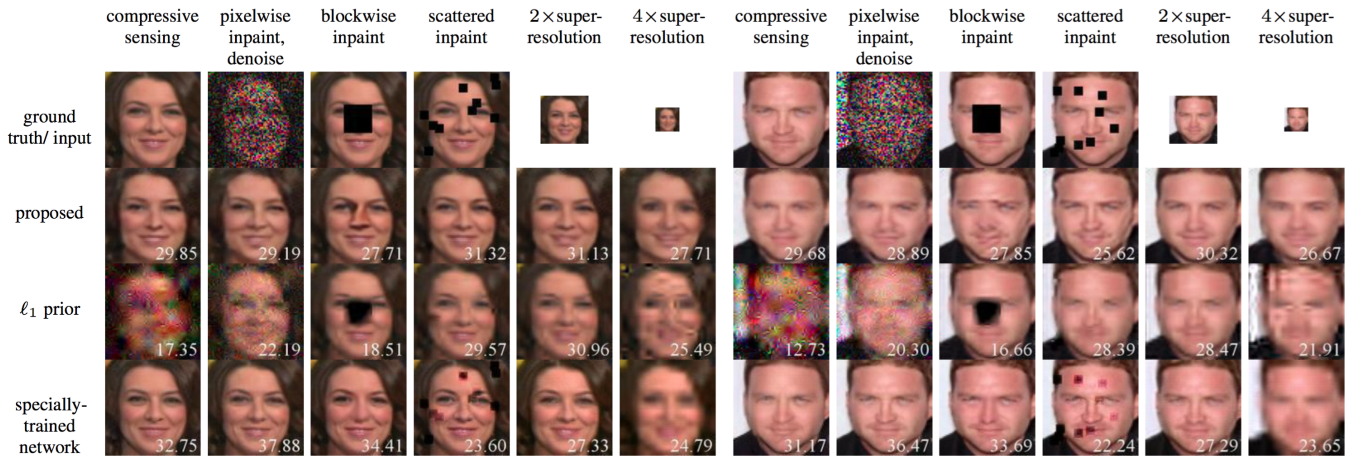


Figure 8.3: Results on MS-Celeb-1M dataset. The PSNR values are shown in the lower-right corner of each image. For compressive sensing, we test on $\frac{m}{d} = 0.1$. For pixelwise inpainting, we drop 50% of the pixels and add Gaussian noise with $\sigma = 0.1$. We use $\rho = 1.0$ on both super resolution tasks.

Configurations of specially-trained networks. For each task (except for 4×-super resolution and for scattered inpainting), we train a neural network using context encoder (Pathak et al., 2016) with adversarial training. For compressive sensing, we design the network based on Mousavi and Baraniuk (2017), which applies $A^\top$ to the linear measurements and resize it into the image size to operate in image space. The measurement matrix A is a random Gaussian matrix and is fixed. For pixelwise inpainting and denoise, we randomly drop 50% of the pixels and add Gaussian noise with $\sigma = 0.5$ for each training instances. For blockwise inpainting, we drop a block with 30% size of the images at a random location in the images. For 2×-super resolution, we follow Dong et al. (2014) which first upsamples the low-resolution images to the target resolution using bicubic interpolation. We do not train a network for 4×-super resolution and for scattered inpainting — to demonstrate that the specially-trained networks do not generalize well to similar tasks. Since the inputs to the 2×-super resolution network are bicubic-upsampled images, we also apply the upsampling to $\frac{1}{4}$-resolution images and feed them to the same network. We also feed scattered inpainting inputs to the blockwise inpainting network.

Configurations of wavelet sparsity prior. We compare the proposed framework with the traditional signal regularization using ℓ_1-norm of wavelet coefficients. We tune the weight of the ℓ_1 norm, λ, based on the dataset.

Results. For each of the experiments, we use $\rho = 0.3$ if not mentioned. The results on MNIST, MS-Celeb-1M, and ImageNet datasets are shown in Figures 8.2, 8.3, and 8.4, respectively. We also apply the same IGM projection network trained on ImageNet dataset on the test set of LabelMe dataset. We list the statistics of peak-to-noise ratio (PSNR) values of the reconstruction outputs in Table 8.1. In addition, we use the same IGM

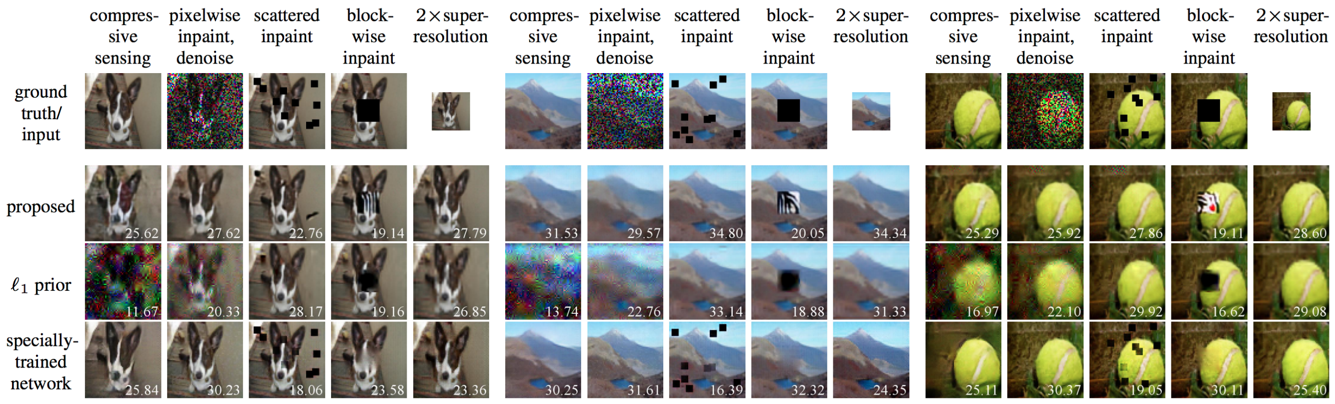


Figure 8.4: Results on ImageNet dataset. The PSNR values are shown in the lower-right corner of each image. Compressive sensing uses $\frac{m}{d} = 0.1$. For pixelwise inpainting, we drop 50% of the pixels and add Gaussian noise with $\sigma = 0.1$. We use $\rho = 0.05$ on scattered inpainting and $\rho = 0.5$ on super resolution.

projection network on the image shown in Figure 8.5[1]. To deal with the 384×512 image, when solving the projection operation (8.3), we apply the IGM projection network on 64×64 patches and stitch the results. The reconstruction outputs are shown in Figure 8.5, and their statistics of each iteration of ADMM are shown in Figure 8.6.

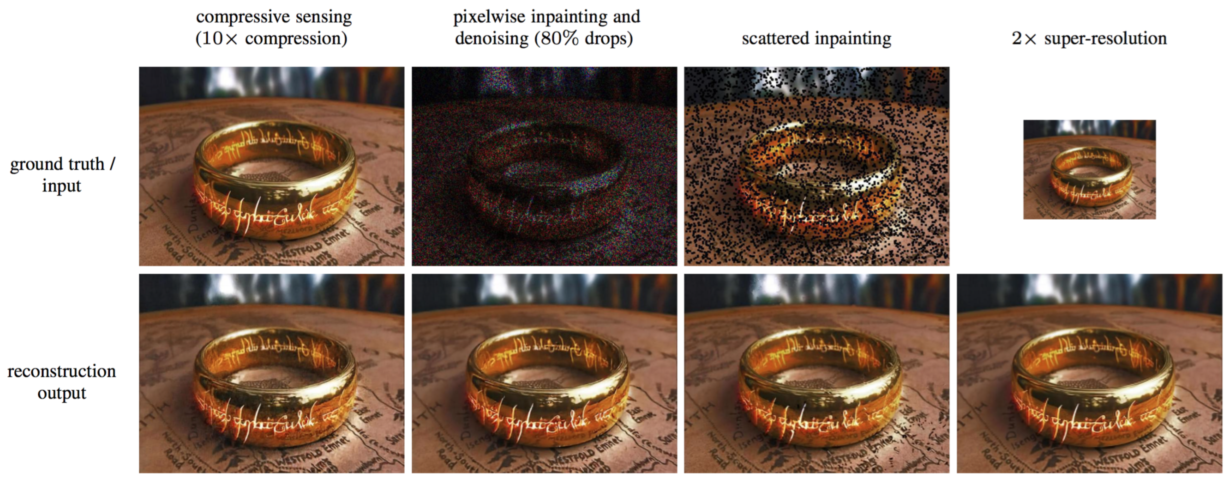

compressive sensing (10× compression)
pixelwise inpainting and denoising (80% drops)
scattered inpainting
2× super-resolution
ground truth / input
reconstruction output

Figure 8.5: The same IGM projection network is used to solve the following tasks: compressive sensing problem with 10× compression, pixelwise random inpainting with 80% dropping rate, scattered inpainting, and 2×-super-resolution. Note that even though the nature and input dimensions of the problems are very different, the proposed framework is able to use a single network to solve them all without retraining.

[1]https://flic.kr/p/mGjhs7

task	**ℓ_1 prior**	**proposed**	**specially-trained**
compressive sensing (10×)	13.01 (±2.75)	25.43 (±3.74)	25.18 (±2.82)
pixelwise inpaint, denoise	20.68 (±1.65)	26.29 (±1.98)	30.13 (±1.66)
2× super-resolution	27.30 (±2.50)	27.11 (±3.21)	22.59 (±2.89)
scattered inpaint	27.85 (±2.58)	25.69 (±3.45)	18.30 (±2.55)

(a) ImageNet

task	**ℓ_1 prior**	**proposed**	**specially-trained**
compressive sensing (10×)	13.79 (±3.67)	27.34 (±5.15)	27.49 (±4.16)
pixelwise inpaint, denoise	21.72 (±2.17)	27.71 (±3.05)	30.93 (±1.96)
2× super-resolution	29.00 (±4.08)	28.52 (±4.64)	20.79 (±4.08)
scattered inpaint	30.17 (±3.96)	28.71 (±5.26)	18.65 (±3.12)

(b) LabelMe

Table 8.1: Average and standard deviation of PSNR values on 100k randomly chosen test images from ImageNet and the whole LabelMe test dataset. Note that we apply the same projection network trained with ImageNet on LabelMe. The similarity in the performance across the two datasets shows the robustness of the projection network.

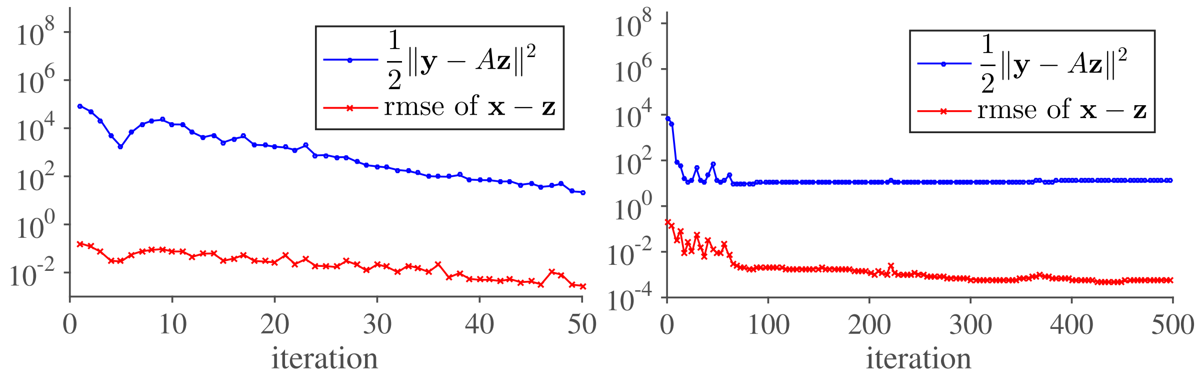


Figure 8.6: Convergence of the ADMM algorithms for compressive sensing (left) and scattered inpainting (right) of Figure 8.5.

As can be seen from the results, using the proposed projection operator/network learning from datasets enables us to solve more challenging problems than using the traditional wavelet sparsity prior. In Figures 8.2 and 8.3, while the traditional ℓ_1-prior of wavelet coefficients is able to reconstruct images from compressive measurements with $\frac{m}{d} = 0.3$, it fails to handle larger compression ratios like $\frac{m}{d} = 0.1$ and 0.03. Similar observations can be seen on pixelwise inpainting of different dropping probabilities and scattered and blockwise inpainting. In contrast, since the proposed IGM projection network is tailored to the images in the datasets, it enables the ADMM algorithm to solve challenging problems like compressive sensing with small $\frac{m}{d}$ and blockwise inpainting on MS-Celeb dataset.

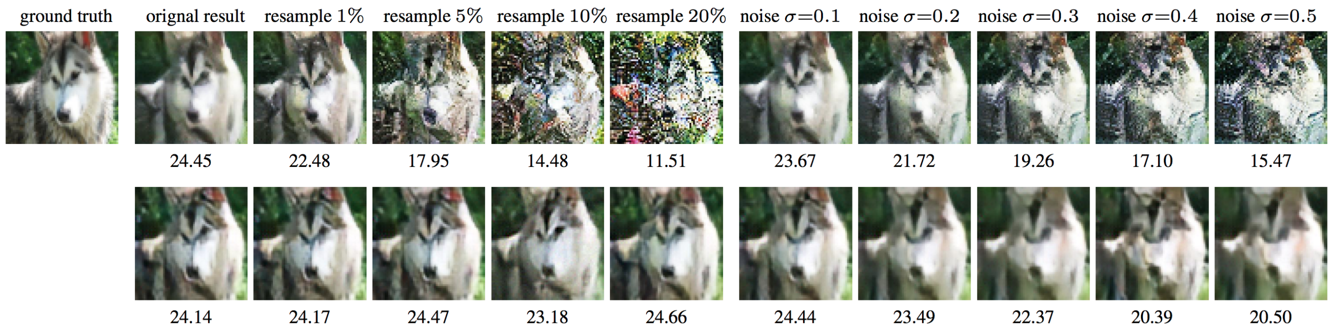


Figure 8.7: Comparison on the robustness to the linear operator A and noise on compressive sensing. The results of the specially-trained network and the proposed method are shown at the top and bottom row, respectively, along with their PSNR values. We use $\rho = 0.5$ for $\sigma = 0.2$, $\rho = 0.7$ for $\sigma = 0.3$, $\rho = 1.0$ for $\sigma = 0.4$, $\rho = 1.1$ for $\sigma = 0.5$, and $\rho = 0.3$ for all other cases.

Robustness to changes in linear operator and to noise. Even though the specially-trained networks are able to generate state-of-the-art results on their designing tasks, they are unable to deal with similar problems, even with a slight change of the linear operator A. For example, as shown in Figure 8.3, the blockwise inpainting network is able to deal with much larger vacant regions; however, it overfits the problem and fails to fill contents to smaller blocks in scattered inpainting problems. The 2×-super resolution network also fails to reconstruct higher resolution images for 4×-super resolution tasks, even though both inputs are upsampled using bicubic algorithm beforehand. We extend this argument with a compressive sensing example. We start from the random Gaussian matrix A_0 used to train the compressive sensing network, and we progressively resample elements in A_0 from the same distribution constructing A_0. As shown in Figure 8.7, once the portion of resampled elements increases, the specially-trained network fails to reconstruct the inputs, even though the new matrices are still Gaussian. The network also shows lower tolerance to Gaussian noise added to the clean linear measurements $y = A_0 x_0$. In comparison, the proposed projector network is robust to changes of linear operators and noise.

Failure cases. The proposed IGM projection network can fail to solve very challenging problems like the blockwise inpainting on ImageNet dataset, which has higher varieties in image contents than the other two datasets we test on. As shown in Figure 8.4, the proposed IGM projection network tries to fill in random edges in the missing regions. In these cases, the IGM projection network fails to project inputs to the natural image set, and thereby, violates our assumption in Theorem 30 and affects the overall ADMM framework. Even though increasing ρ can improve the convergence, it may produce low-quality, overly smoothed outputs.

8.3 Summary

In this chapter, we propose a general framework to implicitly learn a *signal prior* for solving generic linear inverse problems. The application can be explained in two views. From the statistical perspective, the problem is learning conditional distributions via IGMs such that conditional generation follow the data distribution. From the geometric perspective, we follow the projection property of proximal operations and connect the traditional optimization procedures with a learning based method. This application demonstrates the versatility of IGMs that we can replace core hand-crafted component of existing algorithms via IGMs. Therefore, we envision to apply IGMs to improve building components of existing algorithms for a variety of applications.

Chapter 9

Change-Point Detection with Auxiliary Implicit Generative Models

So far we have seen different applications of IGMs from image, text to point cloud generations. In Chapter 8, we show how to use IGM to replace a hand-crafted component and improve the performance of existing machine learning algorithms. In most of the applications, the task can be formulated as a generative task and the output of the algorithm is the learnt sampler of target distributions or the generator network. In this chapter, we present a different usage of IGMs in non-generative problems by constrcuting *auxiliary distributions* to benefit non-generative machine learning algorithms. As an example, we show how IGM can be used to benefit change point detection, which is a variant of anomaly detection, by resolving the issue of data insufficiency.

Change Point Detection. Detecting changes in the temporal evolution of a system (biological, physical, mechanical, etc.) in time series analysis has attracted considerable attention in machine learning and data mining for decades (Basseville et al., 1993; Brodsky and Darkhovsky, 2013). This task, commonly referred to as change-point detection (CPD) or anomaly detection in the literature, aims to predict significant changing points in a temporal sequence of observations. CPD has a broad range of real-world applications such as medical diagnostics (Gardner et al., 2006), industrial quality control (Basu and Meckesheimer, 2007), financial market analysis (Pepelyshev and Polunchenko, 2015) and more. As shown in Figure 9.1, we focus on the retrospective CPD (Li et al., 2015a; Takeuchi and Yamanishi, 2006), which allows a flexible time window to react on the change-points. Retrospective CPD not only enjoys more robust detection (Chandola et al., 2009) but embraces many applications such as climate change detection (Reeves et al., 2007), genetic sequence analysis (Wang et al., 2011), networks intrusion detection (Yamanishi et al., 2004), to name just a few. Various methods have been developed (Gustafsson and Gustafsson, 2000), and many of them are parametric with strong assumptions on the distributions (Basseville et al., 1993; Gustafsson, 1996), including auto-regressive models (Yamanishi and Takeuchi, 2002) and state-space models (Kawahara et al., 2007) for tracking changes in the mean, the variance, and the spectrum.

Ideally, the detection algorithm should be free of distributional assumptions to have

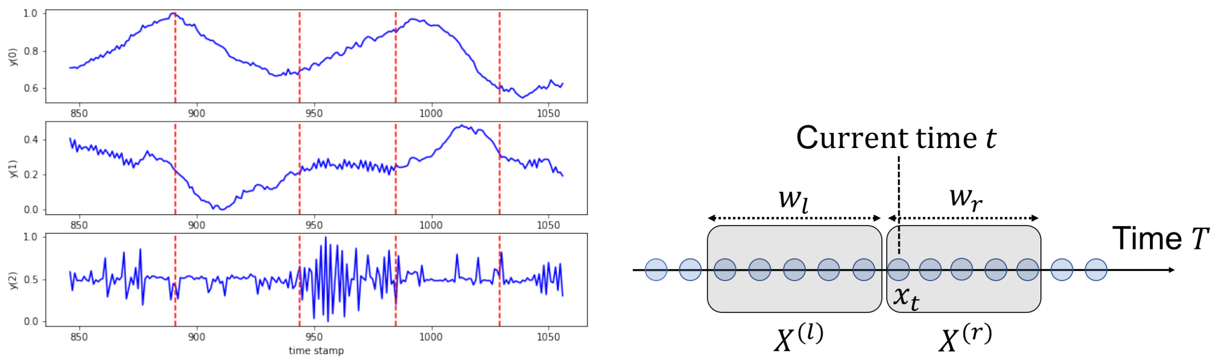


Figure 9.1: A Three-variables time series of Bee-dance dataset with changing points marked by red vertical bars and a sliding window over the time series input with two intervals: the past and the current, where w_l, w_r are the size of the past and current interval, respectively. $X^{(l)}, X^{(r)}$ consists of the data in the past and current interval, respectively.

robust performance as neither true data distributions nor anomaly types are known a priori. Thus the parametric assumptions in many works are unavoidably a limiting factor in practice. As an alternative, nonparametric and kernel approaches are free of distributional assumptions and hence enjoy the advantage of producing more robust performance over a broader class of data distributions.

Kernel two-sample test has been applied to time series CPD with some success. For example, Harchaoui et al. (2009) present a test statistic based upon the maximum kernel fisher discriminant ratio for hypothesis testing and Li et al. (2015a) propose a computationally efficient test statistic based on maximum mean discrepancy with block sampling techniques. The performance of kernel methods, nevertheless, relies heavily on the choice of kernels. Gretton et al. (2007, 2012a) conduct kernel selection for RBF kernel bandwidths via median heuristic without guarantees of optimality regarding to the statistical test power of hypothesis testing. Gretton et al. (2012b) show explicitly optimizing the test power leads to better kernel choice for hypothesis testing under mild conditions. Kernel selection by optimizing the test power, however, is not directly applicable for time series CPD due to insufficient samples of anomalies.

In this chapter, we show how to use a learned generative model to resolve the problem of insufficient data, which leads to effective kernel learning, **KL-CPD**, for maximizing testing power. We start from showing the inaptness of existing kernel learning approaches in a simulated example. We then propose to optimize a lower bound of the test power via an auxiliary generative model, which is parametrized via an IGM to serve as a surrogate of the abnormal events. For hypothesis test with kernels, we present a deep kernel parametrization of our framework, which endows a data-driven kernel for the kernel two-sample test. **KL-CPD** induces composition kernels by combining RNNs and RBF kernels that are suitable for the time series applications. Finally, we conduct extensive benchmark evaluation showing the outstanding performance of **KL-CPD** in real-world CPD applications. With simulation-based analysis, in addition, the proposed algorithm not only boosts the test power but also evades the performance degradation as data dimensionality of time series increases.

9.1 Problem Formulation

Given a sequence of d-dimensional observations $\{x_1, \ldots, x_t, \ldots\}, x_i \in \mathbb{R}^d$, our goal is to detect the existence of a change-point, such that before the change-point, samples are i.i.d from a distribution $\mathbb{P}$, while after the change-point, samples are i.i.d from a different distribution $\mathbb{Q}$. Suppose at current time t and the window size w, denote the past window segment $X^{(l)} = \{x_{t-w}, \ldots, x_{t-1}\}$ and the current window segment $X^{(r)} = \{x_t, \ldots, x_{t+w-1}\}$, We compute the maximum mean discrepancy (MMD) between $X^{(l)}$ and $X^{(r)}$, and use it as the plausibility of change-points: The higher the distribution discrepancy, the more likely the point is a change-point.

9.1.1 MMD and Test Power

We review maximum mean discrepancy (MMD) and its use to two-sample test, which are two cornerstones in this work. Let k be the kernel of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ of functions on a set $\mathcal{X}$. We assume that k is measurable and bounded, $\sup_{x\in\mathcal{X}} k(x,x) < \infty$. MMD is a nonparametric probabilistic distance commonly used in two-sample-test (Gretton et al., 2007, 2012a) as we have seen in Chapter 2 for learning powerful IGM. Given a kernel k, the MMD distance between two distributions $\mathbb{P}$ and $\mathbb{Q}$ is defined as

$$M_k(\mathbb{P}, \mathbb{Q}) := \|\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\|^2_{\mathcal{H}_k} = \mathbb{E}_{\mathbb{P}}[k(x,x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x,y)] + \mathbb{E}_{\mathbb{Q}}[k(y,y')],$$

where $\mu_{\mathbb{P}} = \mathbb{E}_{x\sim\mathbb{P}}[k(x,\cdot))]$ and $\mu_{\mathbb{Q}} = \mathbb{E}_{y\sim\mathbb{Q}}[k(y,\cdot))]$ are the kernel mean embeddings for $\mathbb{P}$ and $\mathbb{Q}$, respectively. In practice we use finite samples from distributions to estimate MMD distance. Given $X = \{x_1, \ldots, x_m\} \sim \mathbb{P}$ and $Y = \{y_1, \ldots, y_m\} \sim \mathbb{Q}$, one unbiased estimator of $M_k(\mathbb{P}, \mathbb{Q})$ is

$$\widehat{M}_k(X,Y) := \frac{1}{\binom{m}{2}} \sum_{i\neq i'} k(x_i, x_{i'}) - \frac{2}{m^2} \sum_{i,j} k(x_i, y_j) + \frac{1}{\binom{m}{2}} \sum_{j\neq j'} k(y_j, y_{j'}).$$

which has nearly minimal variance among unbiased estimators (Gretton et al., 2012a, Lemma 6).

In Chapter 2, we focus on extending MMD to define a valid probability distance. In this chapter, we focus on the *test power* with finite samples. For any characteristic kernel k, $M_k(\mathbb{P}, \mathbb{Q})$ is non-negative and in particular $M_k(\mathbb{P}, \mathbb{Q}) = 0$ *iff* $\mathbb{P} = \mathbb{Q}$. However, the estimator $\widehat{M}_k(X, X')$ may not be 0 even though $X, X' \sim \mathbb{P}$ due to finite sample size. Hypothesis test instead offers thorough statistical guarantees of whether two finite sample sets are the same distribution. Following Gretton et al. (2012a), the hypothesis test is defined by the null hypothesis $H_0 : \mathbb{P} = \mathbb{Q}$ and alternative $H_1 : \mathbb{P} \neq \mathbb{Q}$, using test statistic $m\widehat{M}_k(X,Y)$. For a given allowable false rejection probability α (i.e., false positive rate or Type I error), we choose a test threshold c_α and reject H_0 if $m\widehat{M}_k(X,Y) > c_\alpha$.

We now describe the objective to choose the kernel k for maximizing the test power (Gretton et al., 2012b; Sutherland et al., 2017). First, note that, under the alternative $H_1 : \mathbb{P} \neq \mathbb{Q}$, $\widehat{M}_k$ is asymptotically normal,

$$\frac{\widehat{M}_k(X, Y) - M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} \xrightarrow{\mathcal{D}} \mathcal{N}(0, 1), \tag{9.1}$$

where $V_m(\mathbb{P}, \mathbb{Q})$ denotes the asymptotic variance of the $\widehat{M}_k$ estimator. The test power is then

$$\Pr\left(m\widehat{M}_k(X, Y) > c_\alpha\right) \longrightarrow \Phi\Big(\frac{M_k(\mathbb{P}, \mathbb{Q})}{\sqrt{V_m(\mathbb{P}, \mathbb{Q})}} - \frac{c_\alpha}{m\sqrt{V_m(\mathbb{P}, \mathbb{Q})}}\Big) \tag{9.2}$$

where Φ is the CDF of the standard normal distribution. Given a set of kernels $\mathcal{K}$, We aim to choose a kernel $k \in \mathcal{K}$ to maximize the test power, which is equivalent to maximizing the argument of Φ.

9.2 Optimizing Test Power for Change-Point Detection

In time series CPD, we denote $\mathbb{P}$ as the distribution of usual events and $\mathbb{Q}$ as the distribution for the event when change-points happen. The difficulty of choosing kernels via optimizing test power in Eq. (9.2) is that we have very limited samples from the abnormal distribution $\mathbb{Q}$. Kernel learning in this case can easily overfit, which leads to sub-optimal performance in time series CPD.

9.2.1 Difficulties of Optimizing Kernels for CPD

To demonstrate how limited samples of $\mathbb{Q}$ would affect optimizing test power, we consider kernel selection for Gaussian RBF kernels on the Blobs dataset (Gretton et al., 2012b; Sutherland et al., 2017), which is considered hard for kernel two-sample test. $\mathbb{P}$ is a 5×5 grid of two-dimensional standard normals, with spacing 15 between the centers. $\mathbb{Q}$ is laid out identically, but with covariance $\frac{\epsilon_q - 1}{\epsilon_q + 1}$ between the coordinates (so the ratio of eigenvalues in the variance is ϵ_q). Left panel of Fig. 9.2 shows $X \sim \mathbb{P}$ (red samples), $Y \sim \mathbb{Q}$ (blue dense samples), $\widetilde{Y} \sim \mathbb{Q}$ (blue sparse samples) with $\epsilon_q = 6$. Note that when $\epsilon_q = 1$, $\mathbb{P} = \mathbb{Q}$.

For $\epsilon_q \in \{4, 6, 8, 10, 12, 14\}$, we take 10000 samples for X, Y and 200 samples for $\widetilde{Y}$. We consider two objectives for choosing kernels:

i *median heuristic*

ii *max-ratio* $\eta_{k^*}(X, Y) = \arg\max_k \widehat{M}_k(X, Y)/\sqrt{V_m(X, Y)}$

among 20 kernel bandwidths. We repeat this process 1000 times and report the test power under false rejection rate $\alpha = 0.05$. As shown in the right panel of Fig. 9.2, optimizing kernels using limited samples $\widetilde{Y}$ significantly decreases the test power compared to Y (blue curve down to the cyan curve). This result not only verifies our claim on the inaptness of existing kernel learning objectives for CPD task, but also stimulates us with the following question,

How to optimize kernels with very limited samples from $\mathbb{Q}$, even none in an extreme?

9.2.2 A Practical Lower Bound on Optimizing Test Power

We first assume there exist a surrogate distribution $\mathbb{G}$ that we can easily draw samples from ($Z \sim \mathbb{G}$, $|Z| \gg |\widetilde{Y}|$), and also satisfies the following property:

$$M_k(\mathbb{P},\mathbb{P}) < M_k(\mathbb{P},\mathbb{G}) < M_k(\mathbb{P},\mathbb{Q}), \forall k \in \mathcal{K}, \tag{9.3}$$

Besides, we assume dealing with non trivial case of $\mathbb{P}$ and $\mathbb{Q}$ where a lower bound $\frac{1}{m}v_l \leq V_{m,k}(\mathbb{P},\mathbb{Q}), \forall k$ exists. Since $M_k(\mathbb{P},\mathbb{Q})$ is bounded, there exists an upper bound v_u. With bounded variance $\frac{v_l}{m} \leq V_{m,k}(\mathbb{P},\mathbb{Q}) \leq \frac{v_u}{m}$ condition, we derive an lower bound $\gamma_{k*}(\mathbb{P},\mathbb{G})$ of the test power

$$\begin{aligned} \max_{k\in\mathcal{K}} \frac{M_k(\mathbb{P},\mathbb{Q})}{\sqrt{V_m(\mathbb{P},\mathbb{Q})}} - \frac{c_\alpha/m}{\sqrt{V_m(\mathbb{P},\mathbb{Q})}} &\geq \max_{k\in\mathcal{K}} \frac{M_k(\mathbb{P},\mathbb{Q})}{\sqrt{v_u/m}} - \frac{c_\alpha}{\sqrt{m v_l}} \\ &\geq \max_{k\in\mathcal{K}} \frac{M_k(\mathbb{P},\mathbb{G})}{\sqrt{v_u/m}} - \frac{c_\alpha}{\sqrt{m v_l}} \\ &= \gamma_{k*}(\mathbb{P},\mathbb{G}). \end{aligned} \tag{9.4}$$

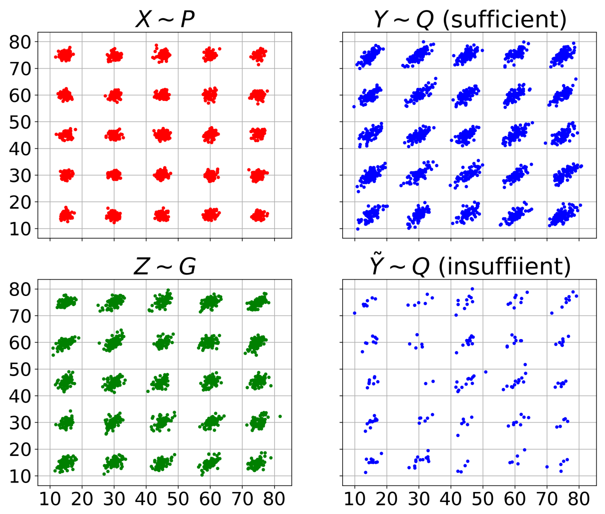
Just for now in the blob toy experiment, we artifact this distribution $\mathbb{G}$ by mimicking $\mathbb{Q}$ with the covariance $\epsilon_g = \epsilon_q - 2$. We defer the discussion on how to find $\mathbb{G}$ in Section 9.2.3. Choosing kernels via $\gamma_{k^*}(X, Z)$ using surrogate samples $Z \sim \mathbb{G}$, as represented by the green curve in Fig. 9.2, substantially boosts the test power compared to $\eta_{k*}(X, \widetilde{Y})$ with sparse samples $\widetilde{Y} \sim \mathbb{Q}$.

Test Threshold Approximation Under $H_0 : \mathbb{P} = \mathbb{Q}$, $m\widehat{M}_k(X, Y)$ converges asymptotically to a distribution that depends on the unknown data distribution $\mathbb{P}$ (Gretton et al., 2012a, Theorem 12); we thus cannot evaluate the test threshold c_α in closed form. Common ways of estimating threshold includes the permutation test and a estimated null distribution based on approximating the eigenspectrum of the kernel. Nonetheless, both are still computational demanding in practice. Even with the estimated threshold, it is difficult to optimize c_α because it is a function of k and $\mathbb{P}$.

For $X, X' \sim \mathbb{P}$, we know that c_α is a function of the empirical estimator $\widehat{M}_k(X, X')$ that controls the Type I error. Bounding $\widehat{M}_k(X, X')$ could be an approximation of bounding c_α. Therefore, we propose the following objective that maximizing a lower bound of test power

$$\operatorname*{argmax}_{k\in\mathcal{K}} M_k(\mathbb{P},\mathbb{G}) - \lambda \widehat{M}_k(X, X'), \tag{9.5}$$

where λ is a hyper-parameter to control the trade-off between Type-I and Type-II errors, as well as absorbing the constants m, v_l, v_u in variance approximation. Note that in experiment, the optimization of Eq. (9.5) is solved using the unbiased estimator of $M_k(\mathbb{P},\mathbb{G})$ with empirical samples.



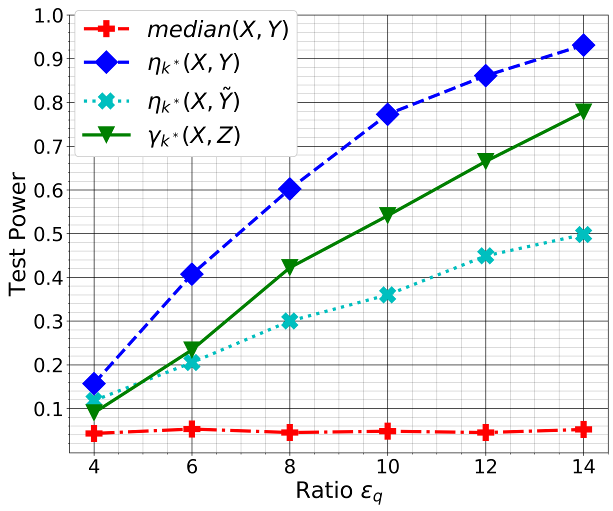


Figure 9.2: **Left**: 5×5 Gaussian grid, samples from $\mathbb{P}$, $\mathbb{Q}$ and $\mathbb{G}$. We discuss two cases of $\mathbb{Q}$, one of sufficient samples, the other of insufficient samples. **Right**: Test power of kernel selection versus ϵ_q. Choosing kernels by $\gamma_{k^*}(X, Z)$ using a surrogate distribution $\mathbb{G}$ is advantageous when we do not have sufficient samples from $\mathbb{Q}$, which is typically the case in time series CPD task.

9.2.3 Learning Surrogate Distributions with IGMs

The remaining question is how to construct the surrogate distribution $\mathbb{G}$ without any sample from $\mathbb{Q}$. Although injecting random noise to $\mathbb{P}$ is a simple way to construct $\mathbb{G}$, it may result in a sub-optimal $\mathbb{G}$ because of sensitivity to the level of injected random noise. Without prior knowledge of $\mathbb{Q}$, to ensure (9.3) hold for any possible $\mathbb{Q}$ (e.g. $\mathbb{Q} \neq \mathbb{P}$ but $\mathbb{Q} \approx \mathbb{P}$), intuitively, we have to make $\mathbb{G}$ as *closed* to $\mathbb{P}$ as possible. We propose to learn an *auxiliary IGM* $\mathbb{G}_\theta$ parameterized by θ such that

$$\widehat{M}_k(X, X') < M_k(\mathbb{P}, \mathbb{G}_\theta) < M_k(\mathbb{P}, \mathbb{Q}), \forall k \in \mathcal{K}.$$

To ensure the first inequality hold, we adopt early stopping when solving $\mathbb{G}_\theta$ in practice. Also, if $\mathbb{P}$ is sophisticate, which is common in time series cases, limited capacity of parametrization of $\mathbb{G}_\theta$ with finite size model (e.g. neural networks) (Arora et al., 2017) and finite samples of $\mathbb{P}$ also hinder us to fully recover $\mathbb{P}$. Therefore, we result in a min-max formulation to consider all possible $k \in \mathcal{K}$ when we learn $\mathbb{G}$,

$$\min_{\theta} \max_{k \in \mathcal{K}} \quad M_k(\mathbb{P}, \mathbb{G}_\theta) - \lambda \widehat{M}_k(X, X'), \tag{9.6}$$

and solve the kernel for the hypothesis test in the mean time. In experiment, we use simple alternative (stochastic) gradient descent to solve each other.

9.2.4 Comparison with MMD GAN

We remark that although the resulted objective (9.6) is similar to MMD GAN in Chapter 2, *the motivation and explanation are different.*

The first difference is the interpretation of inner maximization problem $\max_k M_k(\mathbb{P}, \mathbb{G})$. MMD GANs (Bińkowski et al., 2018; Li et al., 2017) treat whole maximization problem $\max_k M_k(\mathbb{P}, \mathbb{G})$ as a new probabilistic distance, which can also be viewed as an extension of integral probability metric (IPM). The properties of the *distance* is also studied in Arbel et al. (2018); Li et al. (2017). However, the maximization problem (9.4) of this paper defines the lower bound of the test power, which also takes the variance of the empirical estimate into account, instead of the distance.

Regarding the goals, MMD GAN aims to learn a generative model that approximates the underlying data distribution $\mathbb{P}$ of interests. All the works (Arbel et al., 2018; Bińkowski et al., 2018; Dziugaite et al., 2015; Li et al., 2017, 2015b; Sutherland et al., 2017) use MMD or $\max_k M_k(\mathbb{P}, \mathbb{G})$ to define distance, then try to optimize $\mathbb{G}$ to be as closed to $\mathbb{P}$ as possible. However, that is not the goal of this paper, where $\mathbb{G}$ is just an *auxiliary distribution* which needs to satisfies Eq. (9.3). Instead, we aim to find the most powerful k for conducting hypothesis test. In practice, we still optimize $\mathbb{G}$ toward $\mathbb{P}$ because we usually have no prior knowledge (sufficient samples) about $\mathbb{Q}$, and we want to ensure the lower bound still hold for many possible $\mathbb{Q}$ (e.g. $\mathbb{Q}$ can be also similar to $\mathbb{P}$). However, even with this reason, we still adopt early stopping to prevent the auxiliary $\mathbb{G}$ from being exactly the same as $\mathbb{P}$.

9.3 KLCPD: Realization for Time Series Applications

In this section, we present a practical realization of the kernel learning framework for time series CPD with an auxiliary IGM for learning surrogate distributions.

Compositional Kernels. To have a more expressive kernel for complex time series, we consider compositional kernels $\widetilde{k} = k \circ f$ that combines RBF kernels k with injective functions f_ϕ:

$$K = \left\{ \widetilde{k} \mid \widetilde{k}(x, x') = \exp(-\|f_\phi(x) - f_\phi(x)'\|^2) \right\}. \tag{9.7}$$

The resulted kernel $\widetilde{k}$ is still characteristic if f is an injective function and k is characteristic (Gretton et al., 2012a). This ensures the MMD endowed by $\widetilde{k}$ is still a valid probabilistic distance. One example function class is $\{f_\phi | f_\phi(x) = \phi x, \phi > 0\}$, equivalent to the kernel bandwidth tuning. Inspired by the recent success of combining deep neural networks into kernels (Al-Shedivat et al., 2017; Li et al., 2017; Wilson et al., 2016), we parameterize the injective functions f_ϕ by recurrent neural networks (RNNs) to capture the temporal dynamics of time series.

For an injective function f, there exists a function F such that $F(f(x)) = x, \forall x \in \mathcal{X}$, which can be approximated by an auto-encoder via sequence-to-sequence architecture for time series. One practical realization of f would be a RNN encoder parametrized by ϕ while the function F is a RNN decoder parametrized by ψ trained to minimize the reconstruction loss. Thus, our final objective is

$$\min_\theta \max_\phi \quad M_\phi\big(\mathbb{P}, \mathbb{G}_\theta\big) - \lambda \cdot \widehat{M}_{f_\phi}\big(X, X'\big) - \beta \cdot \mathbb{E}_{\nu \in \mathbb{P} \cup \mathbb{G}_\theta} \|\nu - F_\psi\big(f_\phi(\nu)\big)\|_2^2. \tag{9.8}$$

Practical Implementation. In practice, we consider two consecutive windows in mini-batch to estimate $\widehat{M}_{f_\phi}\big(X, X'\big)$ in an online fashion for the sake of efficiency. Specifically, the sample $X \sim \mathbb{P}$ is divided into the left window segment $X^{(l)} = \{x_{t-w}, \dots, x_{t-1}\}$ and the right window segment $X^{(r)} = \{x_t, \dots, x_{t+w-1}\}$ such that $X = \{X^{(l)}, X^{(r)}\}$. We now reveal implementation details of the auxiliary generative model and the deep kernel.

Generator g_θ. Instead of modeling the explicit density of $\mathbb{G}_\theta$, we model an IGM generator g_θ where we can draw samples from. The goal of g_θ is to generate plausibly counterfeit but natural samples based on historical $X \sim \mathbb{P}$, which is similar to the conditional GANs (Isola et al., 2017; Mirza and Osindero, 2014). We use sequence-to-sequence (Seq2Seq) architectures (Sutskever et al., 2014) where g_θ encodes time series into hidden states, and g_{θ_d} decodes it with the distributional autoregressive process to approximate the surrogate sample Z:

$$H = g_\theta\big(X^{(l)}, \mathbf{0}\big), \quad \widetilde{h} = h_{t-1} + \omega, \quad Z = g_{\theta_d}\big(X^{(r)}_{\gg 1}, \widetilde{h}\big).$$

where $\omega \sim \mathbb{P}(W)$ is a d_h-dimensional random noise sampled from a base distribution $\mathbb{P}(W)$ (e.g., uniform, Gaussian). $H = [h_{t-w}, \dots, h_{t-1}] \in \mathbb{R}^{d_h \times w}$ is a sequence of hidden states of the generator's encoder. $X^{(r)}_{\gg 1} = \{\mathbf{0}, x_t, x_{t+1}, \dots, x_{t+w-2}\}$ denotes right shift one unit operator over $X^{(r)}$.

Deep Kernel Parametrization. We aim to maximize a lower bound of test power via back-propagation on ϕ using the deep kernel form $\widetilde{k} = k \circ f_\phi$. On the other hand, we can also view the deep kernel parametrization as an **embedding learning** on the injective function $f_\phi(x)$ that can be distinguished by MMD. Similar to the design of generator, the deep kernel is a Seq2Seq framework with one GRU layer of the follow form:

$$H_\nu = f_\phi\big(\nu\big), \ \widehat{\nu} = F_\psi\big(H_\nu\big).$$

where $\nu \sim \mathbb{P} \cup \mathbb{G}_\theta$ are from either the time series data X or the generated sample $Z \sim g_\theta(\omega|X)$.

We present an realization of **KL-CPD** in Algorithm 6 with the weight-clipping technique. The stopping condition is based on a maximum number of epochs or the detecting power of kernel MMD $M_\phi\big(\mathbb{P}, \mathbb{G}_\theta\big) \leq \epsilon$. This ensure the surrogate $\mathbb{G}_\theta$ is not too close to $\mathbb{P}$, as motivated in Sec. 9.2.2.

9.4 Evaluation on Real-world Data

The section presents a comparative evaluation of the proposed **KL-CPD** and seven representative baselines on benchmark datasets from real-world applications of CPD, including the domains of biology, environmental science, human activity sensing, and network traffic loads. Detailed descriptions of all real-world datasets are itemized as follows.

Algorithm 6 KL-CPD, Learning Kernel with Auxiliary IGM

```
1: Input:$\alpha$ the learning rate, $c$ the clipping parameter, $w$ the window size, $n_c$ the number
   of iterations of deep kernels training per generator update.
2: while $M_\phi(\mathbb{P}, \mathbb{G}_\theta) > \epsilon$ do
3:     for $t = 1, \ldots, n_c$ do
4:         Sample a minibatch $X_t \sim \mathbb{P}$, denote $X_t = \{X_t^{(l)}, X_t^{(r)}\}$, and $\omega \sim \mathbb{P}(\Omega)$
5:         gradient($\phi$) $\leftarrow \nabla_\phi M_\phi(\mathbb{P}, \mathbb{G}_\theta) - \lambda \widehat{M}_\phi(X_t^{(l)}, X_t^{(r)}) - \beta \mathbb{E}_{\nu \sim \mathbb{P} \cup \mathbb{G}_\theta} \|\nu - F_\psi(f_\phi(\nu))\|_2^2$
6:         $\phi \leftarrow \phi + \alpha \cdot \text{RMSProp}(\phi, \text{gradient}(\phi))$
7:         $\phi \leftarrow \text{clip}(\phi, -c, c)$
8:
9:     end for
10:    Sample a minibatch $X_{t'} \sim \mathbb{P}$, denote $X_{t'} = \{X_{t'}^{(l)}, X_{t'}^{(r)}\}$, and $\omega \sim \mathbb{P}(\Omega)$
11:    gradient($\theta$) $\leftarrow \nabla_\theta M_\phi(\mathbb{P}, \mathbb{G}_\theta)$
12:    $\theta \leftarrow \theta - \alpha \cdot \text{Adam}(\theta, \text{gradient}(\theta))$
13: end while
```

- **Bee-Dance**[1] records the pixel locations in x and y dimensions and angle differences of bee movements. Ethologists are interested in the three-stages bee waggle dance and aim at identifying the change point from one stage to another, where different stages serve as the communication with other honey bees about the location of pollen and water.
- **Fishkiller**[2] records water level from a dam in Canada. When the dam do not function normally, the water level oscillates quickly in a particular pattern, causing trouble for the fish. The beginning and end of every water oscillation (fish kills) are treated as change points.
- **HASC**[3] is a subset of the Human Activity Sensing Consortium (HASC) challenge 2011 dataset, which provides human activity information collected by portable three-axis accelerometers. The task of change point detection is to segment the time series data according to the 6 behaviors: stay, walk, jog, skip, stair up, and stair down.
- **Yahoo**[4] contains time series representing the metrics of various Yahoo services (e.g. CPU utilization, memory, network traffic, etc) with manually labeled anomalies. We select 15 out of 68 representative time series sequences after removing some sequences with duplicate patterns in anomalies.

The data statistics are summarized in Table 9.1. We pre-process all dataset by normalizing each dimension in the range of $[0, 1]$.

Following Lai et al. (2018); Liu et al. (2013); Saatçi et al. (2010), the datasets are split into the training set (60%), validation set (20%) and test set (20%) in chronological order. Note that training is fully unsupervised for all methods while labels in the validation set

[1]http://www.cc.gatech.edu/~borg/ijcv_psslds/
[2]http://mldata.org/repository/data/viewslug/fish_killer/
[3]http://hasc.jp/hc2011
[4]https://webscope.sandbox.yahoo.com/catalog.php?datatype=s

Dataset	T	#sequences	domain	#labels
Bee-Dance	826.66	6	$\mathbb{R}^3$	19.5
Fishkiller	45175	1	$\mathbb{R}^+$	899
HASC	39397	1	$\mathbb{R}^3$	65
Yahoo	1432.13	15	$\mathbb{R}^+$	36.06

Table 9.1: Dataset. T is length of time series, #labels is average number of labeled change points.

are used for hyperparameters tuning.

For quantitative evaluation, we consider receiver operating characteristic (ROC) curves of anomaly detection results, and measure the area-under-the-curve (AUC) as the evaluation metric. AUC is commonly used in CPD literature (Li et al., 2015a; Liu et al., 2013; Xu et al., 2017).

We compare **KL-CPD** with following algorithms

- Autoregressive Moving Average (**ARMA**) (Box, 2013) is the classic statistical model that predicts the future time series based on an Autoregressive (AR) and a moving average (MA), where AR involves linear regression, while MA models the error term as a linear combination of errors in the past.
- Autoregressive Gaussian Process (**ARGP**) (Candela et al., 2003) is a Gaussian Process for time series forecasting. In an ARGP of order p, $x_{t-p:t-1}$ are taken as the GP input while the output is x_t. ARGP can be viewed as a non-linear version of AR model.
- Recurrent Neural Networks (**RNN**) (Cho et al., 2014) are powerful neural networks for learning non-linear temporal dynamical systems. We consider gated recurrent units (GRU) in our implementation.
- **LSTNet** (Lai et al., 2018) is a recent state-of-the-art deep neural network fore time series forecasting. LSTNet combines different architectures including CNN, RNN, residual networks, and highway networks.
- **ARGP-BOCPD** (Saatçi et al., 2010) is an extension of the Bayesian online change point detection (BOCPD) which uses ARGP instead of AR in underlying predictive models of BOCPD framework.
- **RDR-KCPD** (Liu et al., 2013) considers f-divergence as the dissimilarity measure. The f-divergence is estimated by relative density ratio technique, which involves solving an unconstrained least-squares importance fitting problem.
- **Mstats-KCPD** (Li et al., 2015a) consider kernel maximum mean discrepancy (MMD) on *data space* as dissimilarity measure. Specifically, It samples B block of segments from the past time series, and computes B times MMD distance between the past block with the current segment and takes the average as the dissimilarity measure.

which can be categorized as real-time CPD methods (**ARMA**, **ARGP**, **RNN**,**LSTNet**) and retrospective CPD methods (**ARGP-BOCPD**, **RDR-KCPD**, **Mstats-KCPD**).

Note that **OPT-MMD** is a deep kernel learning baseline which optimizes MMD by treating past samples as $\mathbb{P}$ and the current window as $\mathbb{Q}$ (insufficient samples).

Hyperparameter Settings For hyper-parameter tuning in **ARMA**, the time lag p, q are chosen from $\{1, 2, 3, 4, 5\}$. For **ARGP** and **ARGP-BOCPD** the time lag order p is set to the same as **ARMA** and the hyperparameter of kernel is learned by maximizing the marginalized likelihood. For **RDR-KCPD**, the window size w are chosen from $\{25, 50\}$, sub-dim $k = 5$, $\alpha = \{0.01, 0.1, 1\}$. For **Mstats-KCPD** and **KL-CPD**, the window size $w = 25$, and we use RBF kernel with median heuristic setting the kernel bandwidth. The hidden dimension of GRU is $d_h = 10$ for **MMD-codespace**, **MMD-negsample** and **KL-CPD**. For **KL-CPD**, λ is chosen from $\{0.1, 1, 10\}$ and β is chosen from $\{10^{-3}, 10^{-1}, 1, 10\}$.

9.4.1 Main Results

In Table 9.2, the first four rows present the real-time CPD methods, followed by three retrospective-CPD models, and the last is our proposed method. **KL-CPD** shows significant improvement over the other methods on all the datasets, except being in a second place on the **Yahoo** dataset, with 2% lower AUC compared to the leading **ARGP**. This confirms the importance of data-driven kernel selection and effectiveness of our kernel learning framework via an auxiliary IGM. Notice that **OPT-MMD** does not perform well compared with **KL-CPD**, which again verifies our simulated example in Sec. 9.2 that directly applying existing kernel learning approaches with insufficient samples may not be suitable for real-world CPD task.

Method	**Bee-Dance**	**Fishkiller**	**HASC**	**Yahoo**
ARMA (Box, 2013)	0.5368	0.8794	0.5863	0.8615
ARGP (Candela et al., 2003)	0.5833	0.8813	0.6448	**0.9318**
RNN (Cho et al., 2014)	0.5827	0.8872	0.6128	0.8508
LSTNet (Lai et al., 2018)	0.6168	0.9127	0.5077	0.8863
ARGP-BOCPD (Saatçi et al., 2010)	0.5089	0.8333	0.6421	0.9130
RDR-KCPD (Liu et al., 2013)	0.5197	0.4942	0.4217	0.6029
Mstats-KCPD (Li et al., 2015a)	0.5616	0.6392	0.5199	0.6961
OPT-MMD	0.5262	0.7517	0.6176	0.8193
KL-CPD (Proposed method)	**0.6767**	**0.9596**	**0.6490**	0.9146

Table 9.2: AUC on four real-world datasets. **KL-CPD** has the best AUC on three out of four datasets.

Distribution matching approaches like **RDR-KCPD** and **Mstats-KCPD** are not as competitive as **KL-CPD**, and often inferior to real-time CPD methods. One explanation is both **RDR-KCPD** and **Mstats-KCPD** measure the distribution distance in the original data space with simple kernel selection using the median heuristic. The change-points may be hard to detect without the latent embedding learned by neural networks.

KL-CPD, instead, leverages RNN to extract useful contexts and encodes time series in a discriminative embedding (latent space) on which kernel two-sample test is used to detection changing points. This also explains the inferior performance of **Mstats-KCPD** which uses kernel MMD with a fix RBF kernel. That is, using a fixed kernel to detect versatile types of change points is likely to fail.

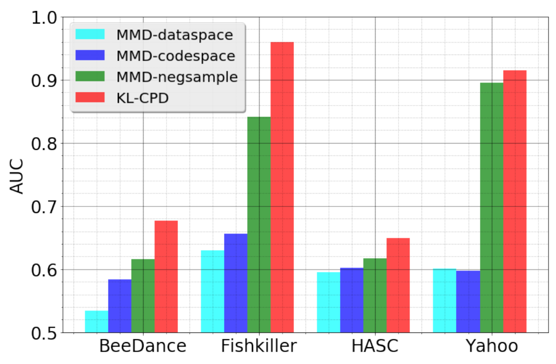


Figure 9.3: Ablation test of **KL-CPD**.

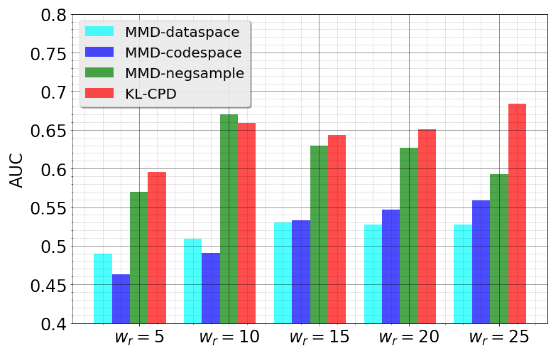


Figure 9.4: AUC vs. different window size w_r on **Bee-Dance**.

9.4.2 Ablation Test on Learning Kernels with Different Encoders

We further examine how different encoders f_ϕ affects **KL-CPD**. For **MMD-dataspace**, f_ϕ is an identity map, equivalent to kernel selection with median heuristic in data space. For **MMD-codespace**, $\{f_\phi, F_\psi\}$ is a Seq2Seq autoencoder minimizing reconstruction loss without optimizing test power. For **MMD-negsample**, the same objective as **KL-CPD** except for replacing the auxiliary generator with injecting Gaussian noise to $\mathbb{P}$.

The results are shown in Figure 9.3. We first notice the mild improvement of **MMD-codespace** over **MMD-dataspace**, showing that using MMD on the induced latent space is effective for discovering beneficial kernels for time series CPD. Next, we see **MMD-negsample** outperforms **MMD-codespace**, showing the advantages of injecting a random perturbation to the current interval to approximate $g_\theta(z|X^{(l)})$. This also justify the validity of the proposed lower bound approach by optimizing $M_k(\mathbb{P}, \mathbb{G})$, which is effective even if we adopt simple perturbed $\mathbb{P}$ as $\mathbb{G}$. Finally, **KL-CPD** models the $\mathbb{G}$ with an auxiliary generator g_θ to obtain conditional samples that are more complex and subtle than the perturbed samples in **MMD-negsample**, resulting in even better performance.

In Figure 9.4, we also demonstrate how the tolerance of delay w_r influences the performance. Due to space limit, results other than **Bee-Dance** dataset are omitted, given they share similar trends. **KL-CPD** shows competitive AUC mostly, only slightly decreases when $w_r = 5$. **MMD-dataspace** and **MMD-codespace**, in contrast, AUC degradation is much severe under low tolerance of delay ($w_r = \{5, 10\}$). The conditional generated samples from **KL-CPD** can be found in Figure 9.5.

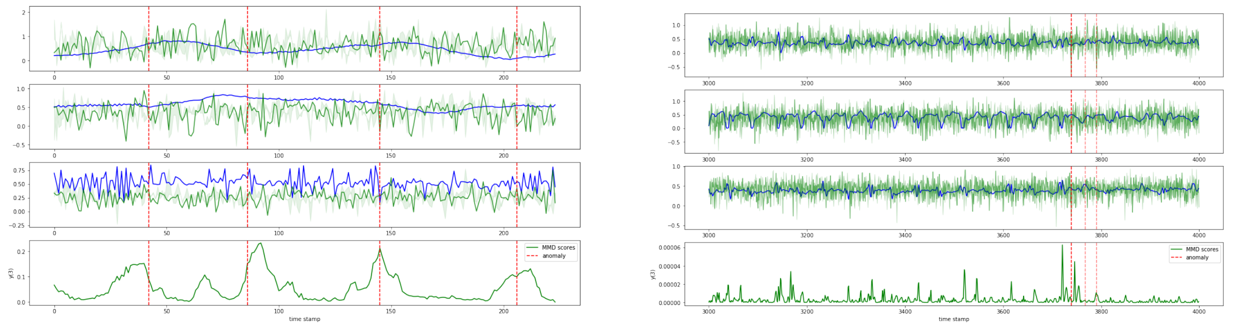


Figure 9.5: Conditionally generated samples by **KL-CPD** and system-predicted CPD scores on **Bee-Dance** (Left) and **HASC** (Right) datasets. In the first three subplots are ground truth signals (blue line), 10 conditional generated samples (green lines) and change points (red vertical line). The last subplot is MMD scores, which peaks around ground truth change points mostly.

9.5 In-depth Analysis on Simulated Data

To further explore the performance of **KL-CPD** with controlled experiments, we follow other time series CPD works (Liu et al., 2013; Matteson and James, 2014; Takeuchi and Yamanishi, 2006) to create three simulated datasets each with a representative change-point characteristic: jumping mean, scaling variance, and alternating between two mixtures of Gaussian (**Gaussian-Mixtures**).

- **Jumping-Mean**: Consider the 1-dimensional auto-regressive model to generate 5000 samples $y(t) = 0.6y(t-1) - 0.5y(t-2) + \epsilon_t$, where $y(1) = y(2) = 0$, $\epsilon_t \sim \mathcal{N}(\mu, 1.5)$ is a Gaussian noise with mean μ and standard deviation 1.5. A change point is inserted at every $100 + \tau$ time stamps by setting the noise mean μ at time t as

$$\mu_n = \begin{cases} 0 & n = 1, \\ \mu_{n-1} + \frac{n}{16} & n = 2, \ldots, 49, \end{cases}$$

  where $\tau \sim \mathcal{N}(0, 10)$ and n is a natural number such that $100(n-1) + 1 \le t \le 100n$.

- **Scaling-Variance**: Same auto-regressive generative model as **Jumping-Mean**, but a change point is inserted at every $100+\tau$ time stamps by setting the noise standard deviation of ϵ_t at time t as

$$\sigma_n = \begin{cases} 1 & n = 1, 3, \ldots, 49, \\ \ln(e + \frac{n}{4}) & n = 2, 4, \ldots, 48, \end{cases}$$

  where $\tau \sim \mathcal{N}(0, 10)$ and n is a natural number such that $100(n-1) + 1 \le t \le 100n$.

- **Gaussian-Mixtures**: Time series data are sampled alternatively between two mixtures of Gaussian $0.5\mathcal{N}(-1, 0.5^2) + 0.5\mathcal{N}(1, 0.5^2)$ and $0.8\mathcal{N}(-1, 1.0^2) + 0.2\mathcal{N}(1, 0.1^2)$ for every 100 time stamps, which is defined as the change points.

9.5.1 Main Results on Simulated data

The results are summarized in Table 9.3. **KL-CPD** achieves the best in all cases. Interestingly, retrospective-CPD (**ARGP-BOCPD**, **RDR-KCPD**, **Mstats-KCPD**) have better results compared to real-time CPD (**ARMA**, **ARGP**, **RNN**,**LSTNet**), which is not the case in real-world datasets. This suggests low reconstruction error does not necessarily lead to good CPD accuracies.

As for why **Mstats-KCPD** does not have comparable performance as **KL-CPD**, given that both of them use MMD as distribution distance? Notice that **Mstats-KCPD** assumes the reference time series (training data) follows the same distribution as the current interval. However, if the reference time series is highly non-stationary, it is more accurate to compute the distribution distance between the latest past window and the current window, which is the essence of **KL-CPD**.

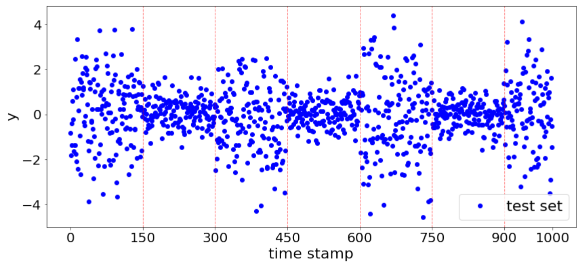
Method	**Jumping-Mean**	**Scaling-Variance**	**Gaussian-Mixtures**
ARMA	0.7731 (0.06)	0.4801 (0.07)	0.5035 (0.08)
ARGP	0.4770 (0.03)	0.4910 (0.07)	0.5027 (0.08)
RNN	0.5053 (0.03)	0.5177 (0.08)	0.5053 (0.08)
LSTNet	0.7694 (0.09)	0.4906 (0.07)	0.4985 (0.07)
ARGP-BOCPD	0.7983 (0.06)	0.4767 (0.08)	0.5027 (0.08)
RDR-KCPD	0.6484 (0.11)	0.7574 (0.06)	0.6022 (0.11)
Mstats-KCPD	0.7309 (0.05)	0.7534 (0.04)	0.6026 (0.08)
KL-CPD	**0.9454** (0.02)	**0.8823** (0.03)	**0.6782** (0.05)

Table 9.3: AUC on three artificial datasets. Mean and standard deviation under 10 random seeds.

9.5.2 MMD versus Dimensionality of Data

We study how different encoders f_ϕ would affect the power of MMD versus the dimensionality of data. We generate an simulated time series dataset by sampling between two multivariate Gaussian $\mathcal{N}(0, \sigma_1^2 I_d)$ and $\mathcal{N}(0, \sigma_2^2 I_d)$ where the dimension $d = \{2, 4, 6, \ldots, 20\}$ and $\sigma_1 = 0.75, \sigma_2 = 1.25$.

Figure 9.6 plots the one-dimension data and AUC results. We see that all methods remain equally strong in low dimensions ($d \leq 10$), while **MMD-dataspace** decreases significantly as data dimensionality increases ($d \geq 12$). An explanation is non-parametric statistical models require the sample size to grow exponentially with the dimensionality of data, which limits the performance of **MMD-dataspace** because of the fixed sample size. On the other hand, **MMD-codespace** and **KL-CPD** are conducting kernel two-sample test on a learned low dimension codespace, which moderately alleviates this issue. Also, **KL-CPD** finds a better kernel (embedding) than **MMD-codespace** by optimizing the lower bound of the test power.



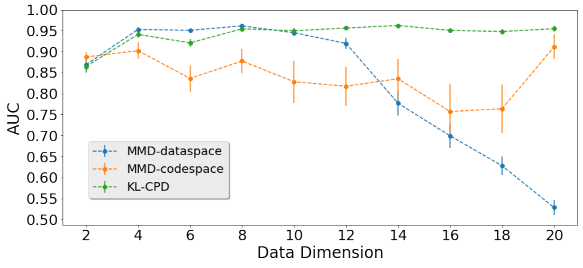


Figure 9.6: MMD with different encoder f_ϕ versus data dimension, under 10 random seeds.

9.6 Summary

In Chapter 8, we use IGMs to replace a non-learning-based building component of existing algorithms, which results in a hybrid approach with better performance. In this chapter, we push the ability of IGMs to the boundary by creating an auxiliary additive via IGMs to boost performance. We propose **KL-CPD**, a new kernel learning framework for two-sample test by optimizing a lower bound of test power defined by an auxiliary IGM generator, to resolve the issue of insufficient samples in change-points detection. The deep kernel parametrization of **KL-CPD** combines the latent space of RNNs with RBF kernels that effectively detect a variety of change-points from different real-world applications. Extensive evaluation of our new approach along with strong baseline methods on benchmark datasets shows the outstanding performance of the proposed method in retrospective CPD. We expect more new algorithms to be studied by adding IGMs as new ingredients.

Part IV

Conclusion

Chapter 10

Conclusion

Learning generative models for capturing distributions of data is a huge topic and has been studied in machine learning and statistics for decades. Implicit generative models (IGMs) do not get attentions in machine learning until the surge of recent deep learning progress. Although there are lots of successful examples of deep IGMs, many developments are weakly justified with mathematical analysis; although IGMs look new in the filed, its essence, learning transformations, has rich connections with several algorithms in other fields and has the potentials for applications underexplored in machine learning before. In this thesis, we focus on these two issues via studying its statistical groundings, and bridging the gaps between modern IGMs and different communities.

We begin from connecting two-sample test, kernel learning and IGMs, which leads to a state-of-the-art generic IGM algorithm MMD GAN. We then shift our focus to analyze sample complexities of learning IGMs from a standard non-parametric point of view. Those studies make a step forward to establishing theoretical foundations of IGMs.

Although we have generic algorithms for learning generative models, different types of data usually require specialized designs by leveraging underlying structures to achieve satisfactory performance. We demonstrate the examples on learning to generate sequence and point clouds (set) data. On the other hand, the essence of IGMs, which models samplings via a transformation function, help us push generative models to different applications and connect with existing algorithms in other fields in a more intuitive way. We learn projections as IGM transformations in proximal methods and we learn deformations as IGM transformations in 3D model registrations. In addition to casting existing processes as transformations, we also introduce new usages of IGMs for non-generative tasks by equipping an auxiliary IGMs as powerful building components.

To summarize, in this thesis, we advance learning implicit generative models via understanding its statistical properties and designing practical algorithms with and without leveraging structures of data. We study different transformations for different generative models, including noise to data, prior to data and data to data transformations, which lead to various applications. We demonstrate the versatilities of IGMs: from natural language and image generations to point clouds generations, from learning kernels to learning proximal operators, and from generative to non-generative tasks.

10.1 Open Problems

Arguably, learning implicit generative models is a growing topic in machine learning and statistics. We discuss some important problems as an avenue for future research.

Evaluation Although there are numerous advancement of IGM algorithms, it is still challenging to evaluate the performance especially on high-dimensional real-world datasets. Although several metrics have be proposed (Bińkowski et al., 2018; Heusel et al., 2017; Salimans et al., 2016), contradictory, there is no conclusive result after tunning hyperparameters (Lucic et al., 2018). One direction is studying synthetic yet fairly complicated datasets, which we have the ground truth for comparison. For high-dimensional real-world data, downstream tasks might be the direction to pursue.

Optimization GAN-based IGM algorithms come with a minmax objective function, which is challenging to optimize. However, most of recent progress of GANs is focusing either statistical properties or neural network architectures. Simple alternative SGD is known to suffer from several drawbacks (Mescheder et al., 2017). Some better algorithms are proposed (Daskalakis et al., 2017; Mescheder et al., 2018, 2017; Nagarajan and Kolter, 2017; Roth et al., 2017), but the analysis is still limited to simple scenarios, such as bilinear games, even though they still bring empirical improvements on complex GAN dynamics. Better algorithms with deeper analysis on more complex systems are desirable.

Connection with Explicit Models Although IGMs focus on learning sampling, it is not conflict with explicit models. In variational Bayes (Kingma and Welling, 2013) and flow models (Kingma and Dhariwal, 2018) with neural networks, the final model is still a transformation function with different constraints. The interesting questions then arise: what is the proper constraints or regularizations for the transformation function for generative models, and which is the ideal objective (adversarial loss v.s. likelihood) of learning generative models.

Structured Data Generation One common and highly structured data is the discrete sequence, such as text. So far, IGMs do not have success of learning on natural languages without relying on MLE pre-training. The important direction is pushing learning IGMs to generate natural languages or proving if MLE is a better objective to optimize than other probability metrics. In addition to text, there are other structured data underexplored for IGMs, such as graphs and 3D meshes.

Human Priors In this thesis, we explore using a differentiable render as priors to model body point clouds. There are more matured renderers developed in computer graphics but they may not be differentiable. A possible route is using REINFROCE (Williams, 1992) or ABC (Louppe and Cranmer, 2019) to get gradient information and studying principal probabilistic frameworks by taking all the components into account.

10.2 Other Publications

In addition to the works covered by this thesis, I have also worked on different topics during my PhD study as listed below.

- Bayesian optimization (Li et al., 2016)
- Random features of kernels (Chang et al., 2017b; Li and Póczos, 2016)
- Polynomial optimization (Wang et al., 2017)
- Video anomaly detection (Liu et al., 2018)
- Adversarial examples (Liu et al., 2019)
- Machine learning in cosmology (Lanusse et al., 2017)

Bibliography

Abbasnejad, E., Shi, J., and van den Hengel, A. (2018). Deep lipschitz networks and dudley GANs. 24

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. (2018). Learning representations and generative models for 3d point clouds. In *International Conference on Machine Learning (ICML)*. xvii, 67, 69, 72, 74, 77, 78, 95

Adler, J. and Öktem, O. (2017). Solving ill-posed inverse problems using iterative deep neural networks. *arXiv preprint arXiv:1704.04058*. 126

Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z., and Xing, E. P. (2017). Learning scalable deep kernels with recurrent structure. *Journal of Machine Learning Research (JMLR)*. 115, 139

Alaoui, A. E., Cheng, X., Ramdas, A., Wainwright, M. J., and Jordan, M. I. (2016). Asymptotic behavior of $_p$-based laplacian regularization in semi-supervised learning. *CoRR*. 45, 56, 61, 62, 63

Alber, M., Kindermans, P.-J., Schütt, K., Müller, K.-R., and Sha, F. (2017). An empirical study on the properties of random bases for kernel methods. In *Advances in Neural Information Processing Systems (NIPS)*. 110

Anelli, G., Antchev, G., Aspell, P., Avati, V., Bagliesi, M., Berardi, V., Berretti, M., Boccone, V., Bottigli, U., Bozzo, M., et al. (2008). The totem experiment at the cern large hadron collider. *Journal of Instrumentation*. 1

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. (2005). Scape: Shape completion and animation of people. *ACM Transactions on Graphics (TOG)*. 79

Arbel, M., Sutherland, D. J., Bińkowski, M., and Gretton, A. (2018). On gradient regularizers for mmd gans. In *Advances in Neural Information Processing Systems (NIPS)*. 22, 102, 115, 139

Arjovsky, M. and Bottou, L. (2017). Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*. 9, 37, 51, 102

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. In *International Conference on Machine Learning (ICML)*. 9, 11, 12, 13, 14, 15, 16, 18, 24, 37, 45, 49, 70, 72, 87, 102, 103, 125

Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*. 125

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American mathematical society*. 27

Arora, S., Ge, R., Liang, Y., Ma, T., and Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (gans). In *International Conference on Machine Learning (ICML)*. 14, 24, 70, 138

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv:1607.06450*. 56

Bach, F. R. (2009). Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems (NIPS)*. 99

Bach, F. R., Lanckriet, G. R., and Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning (ICML)*. 99

Bailey, S. W., Otte, D., Dilorenzo, P., and O'Brien, J. F. (2018). Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)*. 94

Basseville, M., Nikiforov, I. V., et al. (1993). *Detection of abrupt changes: theory and application*. Prentice Hall Englewood Cliffs. 133

Basu, S. and Meckesheimer, M. (2007). Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*. 133

Băzăvan, E. G., Li, F., and Sminchisescu, C. (2012). Fourier kernel learning. In *European Conference on Computer Vision (ECCV)*. 99, 107, 110, 114

Beaumont, M. A. (2010). Approximate bayesian computation in evolution and ecology. *Annual review of ecology, evolution, and systematics*. 1

Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research (JMLR)*. 38, 50, 56, 57

Bellemare, M. G., Danihelka, I., Dabney, W., Mohamed, S., Lakshminarayanan, B., Hoyer, S., and Munos, R. (2017). The cramer distance as a solution to biased wasserstein gradients. *arXiv preprint arXiv:1705.10743*. 9, 20, 48, 49, 102

Bengio, Y. (2018). Gans and unsupervised representation learning. 75, 125

Berlinet, A. and Thomas-Agnan, C. (2011). *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media. 27

Bertsekas, D. P. (1985). A distributed asynchronous relaxation algorithm for the assignment problem. In *Decision and Control*. 71, 72

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*. 82

Biau, G. and Devroye, L. (2015). *Lectures on the nearest neighbor method*. Springer. 23

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. (2018). Demystifying mmd gans. In *International Conference on Learning Representations (ICLR)*. 13, 20, 30, 102, 103, 104, 139, 152

Bishop, C. M., Svensén, M., and Williams, C. K. (1998). Gtm: The generative topographic mapping. *Neural computation*. 2

Bishop, M. C. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York. 1, 67

Bogo, F., Kanazawa, A., Lassner, C., Gehler, P., Romero, J., and Black, M. J. (2016). Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*. 79, 80, 88, 95

Bogo, F., Romero, J., Loper, M., and Black, M. J. (2014). Faust: Dataset and evaluation for 3d mesh registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 88, 93

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. (2017). Compressed sensing using generative models. *arXiv preprint arXiv:1703.03208*. 126

Borgerding, M. and Schniter, P. (2016). Onsager-corrected deep learning for sparse linear inverse problems. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 126

Borisenko, O. and Minchenko, L. (1992). Directional derivatives of the maximum function. *Cybernetics and Systems Analysis*, 28(2):309–312. 21

Bottou, L., Arjovsky, M., Lopez-Paz, D., and Oquab, M. (2017). Geometrical insights for implicit generative modeling. *arXiv preprint arXiv:1712.07822*. 24

Box, G. (2013). Box and jenkins: time series analysis, forecasting and control. *A Very British Affair, ser. Palgrave Advanced Texts in Econometrics. Palgrave Macmillan UK*. 142, 143

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*. 123

Brodsky, E. and Darkhovsky, B. S. (2013). *Nonparametric methods in change point problems*. Springer Science & Business Media. 133

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2017). Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*. 95

Bullins, B., Zhang, C., and Zhang, Y. (2018). Not-so-random features. In *International Conference on Learning Representations (ICLR)*. 99, 107, 110, 113, 114, 119

Candela, J. Q., Girard, A., Larsen, J., and Rasmussen, C. E. (2003). Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting. In *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*. 142, 143

Candes, E. J., Romberg, J. K., and Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Communications on pure and applied mathematics*. 122

Casella, G. and Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA. 2, 3

Chan, T. F., Shen, J., and Zhou, H.-M. (2006). Total variation wavelet inpainting. *Journal of Mathematical imaging and Vision*. 121

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*. 133

Chang, J.-H. R., Li, C.-L., Poczos, B., Kumar, B. V., and Sankaranarayanan, A. C. (2017a). One network to solve them all-solving linear inverse problems using deep projection models. In *IEEE International Conference on Computer Vision (ICCV)*. 6

Chang, W.-C., Li, C.-L., Yang, Y., and Poczos, B. (2017b). Data-driven random fourier features using stein effect. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 99, 107, 153

Chang, W.-C., Li, C.-L., Yang, Y., and Póczos, B. (2019). Kernel change-point detection with auxiliary deep generative models. In *International Conference on Learning Representations (ICLR)*. 1, 6, 115

Che, T., Li, Y., Zhang, R., Hjelm, D. R., Li, W., Song, Y., and Bengio, Y. (2017). Maximum-likelihood augmented discrete generative adversarial networks. *arXiv e-prints*. 37

Chen, Y., Yu, W., and Pock, T. (2015). On learning optimized reaction diffusion processes for effective image restoration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 126

Chib, S. and Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *The american statistician*. 30

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*. 142, 143

Chwialkowski, K., Strathmann, H., and Gretton, A. (2016). A kernel test of goodness of fit. In *International Conference on Machine Learning (ICML)*. 64

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*. 125

Cortes, C., Kloft, M., and Mohri, M. (2013). Learning kernels using local rademacher complexity. In *Advances in Neural Information Processing Systems (NIPS)*. 112

Cortes, C., Mohri, M., and Rostamizadeh, A. (2010). Generalization bounds for learning kernels. In *International Conference on Machine Learning (ICML)*. 111

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., and Kandola, J. S. (2002). On kernel-target alignment. In *International Conference on Machine Learning (ICML)*. 109

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. In

Advances in Neural Information Processing Systems (NIPS). 71

Dai, B., Xie, B., He, N., Liang, Y., Raj, A., Balcan, M.-F. F., and Song, L. (2014). Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems (NIPS)*. 115

Dai, Z., Yang, Z., Yang, F., Cohen, W. W., and Salakhutdinov, R. (2017). Good semi-supervised learning that requires a bad gan. *arXiv preprint arXiv:1705.09783*. 38, 58

Daskalakis, C., Ilyas, A., Syrgkanis, V., and Zeng, H. (2017). Training gans with optimism. In *International Conference on Learning Representations (ICLR)*. 152

Dave, A., Vadathya, A. K., and Mitra, K. (2017). Compressive image recovery using recurrent generative model. In *IEEE International Conference on Image Processing (ICIP)*. 126

Diaconis, P., Holmes, S., and Montgomery, R. (2007). Dynamical bias in the coin toss. *SIAM review*. 1

Diggle, P. J. and Gratton, R. J. (1984). Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*. 1, 2, 23

Dinh, L., Krueger, D., and Bengio, Y. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*. 3

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*. 3

Dong, C., Loy, C. C., He, K., and Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision (ECCV)*. 122, 128

Dong, W., Zhang, L., Shi, G., and Wu, X. (2011). Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization. *IEEE Transactions on Image Processing*. 121

Donoho, D. L. (1995). De-noising by soft-thresholding. *IEEE Transactions on Information Theory*. 121

Donoho, D. L., Vetterli, M., DeVore, R. A., and Daubechies, I. (1998). Data compression and harmonic analysis. *IEEE Transactions on Information Theory*. 121

Dudley, R. (1972). Speeds of metric probability convergence. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*. 24

Dudley, R. M. (2018). *Real Analysis and Probability*. Chapman and Hall/CRC. 22

Dumoulin, V., Belghazi, I., Poole, B., Lamb, A., Arjovsky, M., Mastropietro, O., and Courville, A. (2017). Adversarially learned inference. In *International Conference on Learning Representations (ICLR)*. 17, 18, 38, 56, 58

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning (ICML)*. 99

Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. 9, 10, 15, 49, 100, 102, 117, 139

Eckart, B., Kim, K., Troccoli, A., Kelly, A., and Kautz, J. (2015). MLMD: Maximum likelihood mixture decoupling for fast and accurate point cloud registration. In *International Conference on 3D Vision (3DV)*. 68

Efromovich, S. (2010). Orthogonal series density estimation. *Wiley Interdisciplinary Reviews: Computational Statistics*. 23

Ekeland, I. and Turnbull, T. (1983). *Infinite-dimensional Optimization and Convexity*. The University of Chicago Press. 45, 61

Elgammal, A., Liu, B., Elhoseiny, M., and Mazzone, M. (2017). Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*. 1

Endres, D. M. and Schindelin, J. E. (2003). A new metric for probability distributions. *IEEE Transactions on Information Theory*. 31

Fan, H., Su, H., and Guibas, L. J. (2017). A point set generation network for 3d object reconstruction from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 67, 74

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research (JMLR)*. 119

Finn, C. and Levine, S. (2017). Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation (ICRA)*. 1

Fukumizu, K., Gretton, A., Lanckriet, G. R., Schölkopf, B., and Sriperumbudur, B. K. (2009). Kernel choice and classifiability for rkhs embeddings of probability distributions. In *Advances in Neural Information Processing Systems (NIPS)*. 11

Gardner, A. B., Krieger, A. M., Vachtsevanos, G., and Litt, B. (2006). One-class novelty detection for seizure analysis from intracranial eeg. *JMLR*. 133

Genest, C. and Rivest, L.-P. (2001). On the multivariate probability integral transformation. *Statistics & Probability Letters*. 3

Genevay, A., Peyré, G., and Cuturi, M. (2018). Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 9, 49, 71

Genova, K., Cole, F., Maschinot, A., Sarna, A., Vlasic, D., and Freeman, W. T. (2018). Unsupervised training for 3d morphable model regression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 83, 95

Girdhar, R., Fouhey, D. F., Rodriguez, M., and Gupta, A. (2016). Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision (ECCV)*. 94

Gönen, M. and Alpaydın, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research (JMLR)*. 99, 100, 109, 111, 113

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*. 2, 9, 10, 11, 13, 31, 37, 49, 87, 101, 102

Gorham, J. and Mackey, L. W. (2015). Measuring sample quality with stein's method. In *Advances in Neural Information Processing Systems (NIPS)*. 64

Gregor, K. and LeCun, Y. (2010). Learning fast approximations of sparse coding. In *International Conference on Machine Learning (ICML)*. 126

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J. (2007). A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems (NIPS)*. 134, 135

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012a). A kernel two-sample test. *Journal of Machine Learning Research (JMLR)*. 4, 9, 10, 11, 12, 22, 24, 27, 38, 99, 100, 102, 134, 135, 137, 139

Gretton, A., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., Fukumizu, K., and Sriperumbudur, B. K. (2012b). Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems (NIPS)*. 11, 134, 136

Gretton, A., Sriperumbudur, B., Sejdinovic, D., Strathmann, H., Balakrishnan, S., Pontil, M., and Fukumizu, K. (2012c). Optimal kernel choice for large-scale two-sample tests. In *Advances in Neural Information Processing Systems (NIPS)*. 11

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. (2018a). 3d-coded: 3d correspondences by deep deformation. In *European Conference on Computer Vision (ECCV)*. 79, 80, 83, 88, 90, 91, 92, 93, 95

Groueix, T., Fisher, M., Kim, V. G., Russell, B. C., and Aubry, M. (2018b). A papier-mâché approach to learning 3d surface generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 69, 79, 81, 95

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*. xiii, xvii, 9, 13, 20, 24, 37, 41, 48, 49, 51, 52, 58, 66, 87, 102, 104, 107

Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. (2016). Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *European Conference on Computer Vision (ECCV)*. 127

Gustafsson, F. (1996). The marginalized likelihood ratio test for detecting abrupt changes. *IEEE Transactions on automatic control*. 133

Gustafsson, F. and Gustafsson, F. (2000). *Adaptive filtering and change detection*. Citeseer. 133

Harchaoui, Z., Moulines, E., and Bach, F. R. (2009). Kernel change-point analysis. In *Advances in Neural Information Processing Systems (NIPS)*. 134

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 110

Herbrich, R., Graepel, T., and Obermayer, K. (1999). Support vector learning for ordinal regression. In *International Conference on Artificial Neural Networks (ICANN)*. 14

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*. 103, 104, 152

Hinton, G. E. and Salakhutdinov, R. R. (2008). Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*. 99

Hjelm, R. D., Jacob, A. P., Che, T., Cho, K., and Bengio, Y. (2018). Boundary-seeking generative adversarial networks. In *International Conference on Learning Representations (ICLR)*. 37

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*. 56

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1, 140

Jean, N., Xie, S. M., and Ermon, S. (2018). Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. In *Advances in Neural Information Processing Systems (NIPS)*. 115

Jian, B. and Vemuri, B. C. (2005). A robust algorithm for point set registration using mixture of gaussians. In *IEEE International Conference on Computer Vision (ICCV)*. 68

Jin, K. H., McCann, M. T., Froustey, E., and Unser, M. (2017). Deep convolutional neural network for inverse problems in imaging. *IEEE Transactions on Image Processing*. 126

Joo, H., Simon, T., and Sheikh, Y. (2018). Total capture: A 3d deformation model for tracking faces, hands, and bodies. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 80, 95

Joshi, P., Meyer, M., DeRose, T., Green, B., and Sanocki, T. (2007). Harmonic coordinates for character articulation. In *ACM Transactions on Graphics (TOG)*. 94

Kaae Sønderby, C., Caballero, J., Theis, L., Shi, W., and Huszár, F. (2017). Amortised map inference for image super-resolution. *International Conference on Learning Representations (ICLR)*. 37, 51

Kalogerakis, E., Averkiou, M., Maji, S., and Chaudhuri, S. (2017). 3d shape segmentation with projective convolutional networks. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2. 67

Kanazawa, A., Black, M. J., Jacobs, D. W., and Malik, J. (2018a). End-to-end recovery of human shape and pose. In *IEEE Conference on Computer Vision and Pattern*

Recognition (CVPR). 80, 83, 95

Kanazawa, A., Tulsiani, S., Efros, A. A., and Malik, J. (2018b). Learning category-specific mesh reconstruction from image collections. *arXiv preprint arXiv:1803.07549*. 80, 83, 95

Kantorovich, L. V. and Rubinstein, G. S. (1958). On a space of completely additive functions. *Vestnik Leningrad. Univ.* 24

Kavan, L., Collins, S., Žára, J., and O'Sullivan, C. (2008). Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*. 94

Kavan, L. and Žára, J. (2005). Spherical blend skinning: a real-time deformation of articulated models. In *Symposium on Interactive 3D Graphics and Games*. 94

Kawahara, Y., Yairi, T., and Machida, K. (2007). Change-point detection in time-series data based on subspace identification. In *ICDM*. IEEE. 133

Keller, J. B. (1986). The probability of heads. *The American Mathematical Monthly*. 1

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems (NIPS)*. 3, 152

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*. 2, 3, 9, 69, 152

Koller, D., Friedman, N., and Bach, F. (2009). *Probabilistic graphical models: principles and techniques*. MIT press. 1, 78

Kolmogorov, A. (1933). Sulla determinazione empirica di una lgge di distribuzione. *Inst. Ital. Attuari, Giorn.* 24

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. 14, 56, 103, 113

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 110

Kulkarni, K., Lohit, S., Turaga, P., Kerviche, R., and Ashok, A. (2016). Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 122

Kumar, A., Sattigeri, P., and Fletcher, P. T. (2017). Improved semi-supervised learning with gans using manifold invariances. In *Advances in Neural Information Processing Systems (NIPS)*. 38, 56, 58

Kurihara, T. and Miyata, N. (2004). Modeling deformable human hands from medical images. In *Symposium on Computer Animation*. 94

Kusner, M. J. and Hernández-Lobato, J. M. (2016). Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv:1611.04051*. 37

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. (2017). Grammar variational autoencoder. In *International Conference on Machine Learning (ICML)*. 1

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. (2018). Modeling long-and short-term temporal patterns with deep neural networks. In *SIGIR*. 141, 142, 143

Laine, S. and Aila, T. (2016). Temporal ensembling for semi-supervised learning. *arXiv:1610.02242*. 58

Lanckriet, G. R., Cristianini, N., Bartlett, P., Ghaoui, L. E., and Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research (JMLR)*. 99

Lanusse, F., Ma, Q., Li, N., Collett, T. E., Li, C.-L., Ravanbakhsh, S., Mandelbaum, R., and Póczos, B. (2017). Cmu deeplens: deep learning for automatic image-based galaxy–galaxy strong lens finding. *Monthly Notices of the Royal Astronomical Society (MNRAS)*. 153

Lassner, C., Romero, J., Kiefel, M., Bogo, F., Black, M. J., and Gehler, P. V. (2017). Unite the people: Closing the loop between 3d and 2d human representations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 80, 95

Le, B. H. and Deng, Z. (2012). Smooth skinning decomposition with rigid bones. *ACM Transactions on Graphics (TOG)*. 94

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. 14, 113

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A. P., Tejani, A., Totz, J., Wang, Z., et al. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1, 122

Lei, J. (2018). Convergence and concentration of empirical measures under wasserstein distance in unbounded functional spaces. *arXiv preprint arXiv:1804.10556*. 24

Leoni, G. (2017). *A first course in Sobolev spaces*. American Mathematical Soc. 24, 26

Lewis, J. P., Cordner, M., and Fong, N. (2000). Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Annual Conference on Computer graphics and Interactive Techniques (SIGGRAPH)*. 94

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Póczos, B. (2017). MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems (NIPS)*. 4, 9, 24, 37, 49, 87, 99, 102, 106, 139

Li, C.-L., Chang, W.-C., Mroueh, Y., Yang, Y., and Póczos, B. (2019a). Implicit kernel learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 5

Li, C.-L., Kandasamy, K., Póczos, B., and Schneider, J. (2016). High dimensional bayesian optimization via restricted projection pursuit models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 153

Li, C.-L. and Póczos, B. (2016). Utilize old coordinates: Faster doubly stochastic gradients for kernel methods. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. 115, 153

Li, C.-L., Simon, T., Saragih, J., Póczos, B., and Sheikh, Y. (2019b). Lbs autoencoder: Self-supervised fitting of articulated meshes to point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5

Li, C.-L., Zaheer, M., Zhang, Y., Poczos, B., and Salakhutdinov, R. (2018). Point cloud gan. *arXiv preprint arXiv:1810.05795*. 5, 79, 83, 95

Li, S., Xie, Y., Dai, H., and Song, L. (2015a). M-statistic for kernel change-point detection. In *Advances in Neural Information Processing Systems (NIPS)*. 133, 134, 142, 143

Li, Y., Swersky, K., and Zemel, R. (2015b). Generative moment matching networks. In *International Conference on Machine Learning (ICML)*. 9, 10, 14, 15, 20, 49, 100, 102, 139

Liang, T. (2017). How well can generative adversarial networks (GAN) learn densities: A nonparametric view. *arXiv preprint arXiv:1712.08244*. 24, 28, 29

Liang, T. and Stokes, J. (2018). Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. *arXiv preprint arXiv:1802.06132*. 29

Litany, O., Remez, T., Rodolà, E., Bronstein, A. M., and Bronstein, M. M. (2017). Deep functional maps: Structured prediction for dense shape correspondence. In *IEEE International Conference on Computer Vision (ICCV)*. 94

Liu, H.-T. D., Tao, M., Li, C.-L., Nowrouzezahrai, D., and Jacobson, A. (2019). Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. In *International Conference on Learning Representations (ICLR)*. 153

Liu, Q. (2017). Stein variational descent as a gradient flow. In *Advances in Neural Information Processing Systems (NIPS)*. 46, 48, 64

Liu, Q., Lee, J. D., and Jordan, M. I. (2016). A kernelized stein discrepancy for goodness-of-fit tests. In *International Conference on Machine Learning (ICML)*. 48, 64

Liu, Q. and Wang, D. (2016). Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems (NIPS)*. 46

Liu, S., Yamada, M., Collier, N., and Sugiyama, M. (2013). Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*. 141, 142, 143, 145

Liu, Y., Li, C.-L., and Póczos, B. (2018). Classifier two-sample test for video anomaly detections. In *British Machine Vision Conference (BMVC)*. 153

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 14

Loosli, G., Canu, S., and Bottou, L. (2007). Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*. MIT Press. 127

Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., and Black, M. J. (2015). Smpl: A skinned multi-person linear model. *ACM Transactions on Graphics (TOG)*. 79, 80, 81, 82, 84, 88, 94, 95

Louppe, G. and Cranmer, K. (2019). Adversarial variational optimization of non-differentiable simulators. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1, 152

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are gans created equal? a large-scale study. In *Advances in Neural Information Processing Systems (NIPS)*. 30, 74, 152

MacKay, D. J. (1995). Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. 2, 101

Magnenat-Thalmann, N., Laperrire, R., and Thalmann, D. (1988). Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interface88*. Citeseer. 81

Mairal, J. (2016). End-to-end kernel learning with supervised convolutional kernel networks. In *Advances in Neural Information Processing Systems (NIPS)*. 100

Mairal, J., Koniusz, P., Harchaoui, Z., and Schmid, C. (2014). Convolutional kernel networks. In *Advances in Neural Information Processing Systems (NIPS)*. 100

Mairal, J., Sapiro, G., and Elad, M. (2008). Learning multiscale sparse representations for image and video restoration. *Multiscale Modeling & Simulation*. 121

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*. 9, 69

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., and Paul Smolley, S. (2017). Least squares generative adversarial networks. In *IEEE International Conference on Computer Vision (ICCV)*. 9, 37, 87

Matteson, D. S. and James, N. A. (2014). A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*. 145

Mehta, D., Sridhar, S., Sotnychenko, O., Rhodin, H., Shafiei, M., Seidel, H.-P., Xu, W., Casas, D., and Theobalt, C. (2017). Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*. 80, 95

Meinhardt, T., Möller, M., Hazirbas, C., and Cremers, D. (2017). Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. *arXiv preprint arXiv:1704.03488*. 126

Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for gans do actually converge? *arXiv preprint arXiv:1801.04406*. 152

Mescheder, L., Nowozin, S., and Geiger, A. (2017). The numerics of gans. In *Advances in Neural Information Processing Systems (NIPS)*. 152

Metzler, C. A., Maleki, A., and Baraniuk, R. G. (2016). From denoising to compressed sensing. *IEEE Transactions on Information Theory*. 126

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*. 140

Miyato, T., Maeda, S.-i., Koyama, M., and Ishii, S. (2017). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *arXiv:1704.03976*. 58

Mobahi, H. and III, J. W. F. (2015). A Theoretical Analysis of Optimization by Gaussian Continuation. In *AAAI Conference on Artificial Intelligence*. 51

Mohamed, S. and Lakshminarayanan, B. (2016). Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*. 2, 23, 30

Mousavi, A. and Baraniuk, R. G. (2017). Learning to invert: Signal recovery via deep convolutional networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP)*. 122, 128

Mousavi, A., Patel, A. B., and Baraniuk, R. G. (2015). A deep learning approach to structured signal recovery. In *Allerton Conference on Communication, Control, and Computing*. 122

Mroueh, Y., Li, C.-L., Sercu, T., Raj, A., and Cheng, Y. (2018). Sobolev GAN. *International Conference on Learning Representations (ICLR)*. 5, 24, 70, 102, 104

Mroueh, Y. and Sercu, T. (2017). Fisher gan. In *Advances in Neural Information Processing Systems (NIPS)*. 9, 37, 38, 39, 40, 41, 44, 48, 49, 56, 57, 58, 70, 72, 87, 102

Mroueh, Y., Sercu, T., and Goel, V. (2017). Mcgan: Mean and covariance feature matching gan. In *International Conference on Machine Learning (ICML)*. 14, 37, 102

Mroueh, Y., Sercu, T., and Raj, A. (2019). Sobolev descent. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 46

Müller, A. (1997). Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*. 9, 24, 37

Nagarajan, V. and Kolter, J. Z. (2017). Gradient descent GAN optimization is locally stable. In *Advances in Neural Information Processing Systems (NIPS)*. 29, 152

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NIPS)*. 9, 37, 49, 88, 102

Oates, C. J., Girolami, M., and Chopin, N. (2017). Control functionals for monte carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 64

Odena, A., Buckman, J., Olsson, C., Brown, T. B., Olah, C., Raffel, C., and Goodfellow, I. (2018). Is generator conditioning causally related to gan performance? *arXiv preprint arXiv:1802.08768*. 125

Oliva, J. B., Dubey, A., Póczos, B., Schneider, J., and Xing, E. P. (2018). Transformation autoregressive networks. In *International Conference on Machine Learning (ICML)*. 67

Oliva, J. B., Dubey, A., Wilson, A. G., Póczos, B., Schneider, J., and Xing, E. P. (2016). Bayesian nonparametric kernel-learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 99, 100, 110, 115

Owen, M. (2007). *Practical signal processing.* Cambridge university press. 26

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. (2016). Context encoders: Feature learning by inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 122, 128

Pepelyshev, A. and Polunchenko, A. S. (2015). Real-time financial surveillance via quickest change-point detection methods. *arXiv preprint arXiv:1509.01570.* 133

Peyré, G., Cuturi, M., et al. (2019). Computational optimal transport. *Foundations and Trends® in Machine Learning.* 71, 73

Póczos, B., Xiong, L., and Schneider, J. (2012). Nonparametric divergence estimation with applications to machine learning on distributions. *arXiv preprint arXiv:1202.3758.* 73

Pons-Moll, G., Romero, J., Mahmood, N., and Black, M. J. (2015). Dyna: A model of dynamic human shape in motion. *ACM Transactions on Graphics (TOG).* 79

Portilla, J., Strela, V., Wainwright, M. J., and Simoncelli, E. P. (2003). Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Transactions on Image Processing.* 121

Press, O., Bar, A., Bogin, B., Berant, J., and Wolf, L. (2017). Language generation with recurrent generative adversarial networks without pre-training. *arXiv:1706.01399.* 37, 38, 52, 53

Pritchard, J. K., Seielstad, M. T., Perez-Lezaun, A., and Feldman, M. W. (1999). Population growth of human y chromosomes: a study of y chromosome microsatellites. *Molecular biology and evolution.* 1

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 67, 69, 82, 83, 86, 88, 95

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NIPS).* 67, 95

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR).* 15, 16, 37, 103

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS).* 101, 109, 112

Rahimi, A. and Recht, B. (2009). Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems (NIPS).* xv, 109, 110, 114

Rajeswar, S., Subramanian, S., Dutil, F., Pal, C., and Courville, A. (2017). Adversarial generation of natural language. *arXiv:1705.10929.* 37

Ramdas, A., Reddi, S. J., Póczos, B., Singh, A., and Wasserman, L. A. (2015). On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In *AAAI Conference on Artificial Intelligence*. 28

Ramdas, A., Trillos, N. G., and Cuturi, M. (2017). On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*. 24

Reed, S., Akata, Z., Lee, H., and Schiele, B. (2016). Learning deep representations of fine-grained visual descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 52

Reeves, J., Chen, J., Wang, X. L., Lund, R., and Lu, Q. Q. (2007). A review and comparison of changepoint detection techniques for climate data. *Journal of Applied Meteorology and Climatology*. 133

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*. 3

Rhee, T., Lewis, J. P., and Neumann, U. (2006). Real-time weighted pose-space deformation on the gpu. In *Computer Graphics Forum*. 94

Robert, C. P. (2004). *Monte Carlo methods*. Wiley Online Library. 32

Romero, J., Tzionas, D., and Black, M. J. (2017). Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (TOG)*. 79, 81

Roth, K., Lucchi, A., Nowozin, S., and Hofmann, T. (2017). Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems (NIPS)*. 152

Rudin, W. (2011). *Fourier analysis on groups*. John Wiley & Sons. 100

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*. 127

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision (IJCV)*. 127

Saatçi, Y., Turner, R., and Rasmussen, C. E. (2010). Gaussian process change point models. In *International Conference on Machine Learning (ICML)*. 141, 142, 143

Saatchi, Y. and Wilson, A. G. (2017). Bayesian gan. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3625–3634. 115

Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*. 58

Salakhutdinov, R. and Hinton, G. (2009). Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 69

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*. 1, 17, 18, 30, 38, 56, 58, 103, 152

Salimans, T. and Kingma, D. P. (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 56

Shivaswamy, P. K. and Jebara, T. (2010). Maximum relative margin and data-dependent regularization. *Journal of Machine Learning Research (JMLR)*. 49

Singh, S. and Póczos, B. (2018). Minimax distribution estimation in Wasserstein distance. *arXiv preprint arXiv:1802.08855*. 24

Singh, S., Uppal, A., Li, B., Li, C.-L., Zaheer, M., and Póczos, B. (2018). Nonparametric density estimation under adversarial losses. In *Advances in Neural Information Processing Systems (NIPS)*. 4, 23, 24, 26, 27

Sinha, A., Bai, J., and Ramani, K. (2016). Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision (ECCV)*. 95

Sinha, A. and Duchi, J. C. (2016). Learning kernels with random features. In *Advances in Neural Information Processing Systems (NIPS)*. xv, 99, 100, 107, 109, 110, 112, 113, 114, 119

Sinha, A., Unmesh, A., Huang, Q., and Ramani, K. (2017). Surfnet: Generating 3d shape surfaces using deep residual networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 95

Sloan, P.-P. J., Rose III, C. F., and Cohen, M. F. (2001). Shape by example. In *Symposium on Interactive 3D Graphics and Games*. 94

Smirnov, N. (1948). Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics*. 24

Springenberg, J. T. (2015). Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv:1511.06390*. 58

Sriperumbudur, B. et al. (2016). On the optimal estimation of probability measures in weak and strong topologies. *Bernoulli*. 28

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., and Lanckriet, G. R. (2010). Non-parametric estimation of integral probability metrics. In *IEEE International Symposium on Information Theory Proceedings (ISIT)*. 24

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., Lanckriet, G. R., et al. (2012). On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*. 24, 38

Strom, J., Richardson, A., and Olson, E. (2010). Graph-based segmentation for colored 3d laser point clouds. In *International Conference on Intelligent Robots and Systems (IROS)*. 68

Sutherland, D. J., Tung, H. F., Strathmann, H., De, S., Ramdas, A., Smola, A. J., and Gretton, A. (2017). Generative models and model criticism via optimized maximum mean discrepancy. In *International Conference on Learning Representations (ICLR)*. 15, 30, 136, 139

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 37, 59, 140

Székely, G. J., Rizzo, M. L., Bakirov, N. K., et al. (2007). Measuring and testing dependence by correlation of distances. *The Annals of Statistics*. 24

Takeuchi, J.-i. and Yamanishi, K. (2006). A unifying framework for detecting outliers and change points from time series. *IEEE transactions on Knowledge and Data Engineering*. 133, 145

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*. 15

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. (2017a). Wasserstein auto-encoders. In *International Conference on Learning Representations (ICLR)*. 9, 69

Tolstikhin, I., Sriperumbudur, B. K., and Muandet, K. (2017b). Minimax estimation of kernel mean embeddings. *Journal of Machine Learning Research (JMLR)*. 24

Tomczak, J. M. and Welling, M. (2017). Vae with a vampprior. *arXiv preprint arXiv:1705.07120*. 69

Tsybakov, A. B. (2009). *Introduction to nonparametric estimation*. Springer Series in Statistics. Springer, New York. 1, 23, 24

Tung, H.-Y., Tung, H.-W., Yumer, E., and Fragkiadaki, K. (2017). Self-supervised learning of motion capture. In *Advances in Neural Information Processing Systems (NIPS)*. 80

Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M. J., Laptev, I., and Schmid, C. (2017). Learning from synthetic humans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 88, 96

Venkatakrishnan, S. V., Bouman, C. A., and Wohlberg, B. (2013). Plug-and-play priors for model based reconstruction. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 126

Villani, C. (2008). *Optimal transport: old and new*. Springer Science & Business Media. 24

Wang, D. and Liu, Q. (2016). Learning to draw samples: With application to amortized MLE for generative adversarial learning. *CoRR*. 49

Wang, P., Li, W., Gao, Z., Zhang, J., Tang, C., and Ogunbona, P. O. (2016). Action recognition from depth maps using deep convolutional neural networks. *IEEE Transactions on Human-Machine Systems*. 95

Wang, P.-W., Li, C.-L., and Kolter, J. Z. (2017). Polynomial optimization methods for matrix factorization. In *AAAI Conference on Artificial Intelligence*. 153

Wang, Q., Kulkarni, S. R., and Verdú, S. (2009). Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE International Symposium on Information Theory Proceedings (ISIT)*. 73

Wang, X. C. and Phillips, C. (2002). Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Symposium on Computer Animation*. 94

Wang, Y., Wu, C., Ji, Z., Wang, B., and Liang, Y. (2011). Non-parametric change-point method for differential gene expression detection. *PloS one*. 133

Wang, Y., Yin, W., and Zeng, J. (2015). Global convergence of admm in nonconvex nonsmooth optimization. *arXiv preprint arXiv:1511.06324*. 125

Wasserman, L. (2006). *All of Nonparametric Statistics*. Springer Science & Business Media. 23, 24, 33

Wasserman, L. (2013). *All of statistics: a concise course in statistical inference*. Springer Science & Business Media. 1, 9, 12

Weed, J. and Bach, F. (2017). Sharp asymptotic and finite-sample rates of convergence of empirical measures in wasserstein distance. *arXiv preprint arXiv:1707.00087*. 24, 71, 73

Wei, L., Huang, Q., Ceylan, D., Vouga, E., and Li, H. (2016). Dense human body correspondences using convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 89, 91, 95

Wendland, H. (2004). *Scattered data approximation*. Cambridge university press. 27

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*. 152

Wilson, A. and Adams, R. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *International Conference on Machine Learning (ICML)*. xv, 99, 100, 104, 106, 113, 114, 115

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 11, 99, 100, 115, 139

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NIPS)*. 94

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 72, 95

Xiao, L., Heide, F., Heidrich, W., Schölkopf, B., and Hirsch, M. (2017). Discriminative transfer learning for general image restoration. *arXiv preprint arXiv:1703.09245*. 126

Xu, L., Ren, J. S., Liu, C., and Jia, J. (2014). Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems (NIPS)*. 122

Xu, Z., Kersting, K., and von Ritter, L. (2017). Stochastic online anomaly analysis for streaming time series. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 142

Yamanishi, K. and Takeuchi, J.-i. (2002). A unifying framework for detecting outliers and change points from non-stationary time series data. In *SIGKDD*. ACM. 133

Yamanishi, K., Takeuchi, J.-I., Williams, G., and Milne, P. (2004). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*. 133

Yang, Y., Feng, C., Shen, Y., and Tian, D. (2018). Foldingnet: Point cloud auto-encoder via deep grid deformation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 69, 79, 80, 81, 82, 83, 95

Yang, Z., Wilson, A., Smola, A., and Song, L. (2015). A la carte–learning fast kernels. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 99, 107

Yarotsky, D. (2017). Error bounds for approximations with deep ReLU networks. *Neural Networks*. 29

Yoon, J., Jordon, J., and van der Schaar, M. (2018). Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning (ICML)*. 1

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2015). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*. 14

Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI Conference on Artificial Intelligence*. 37

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017a). Deep sets. In *Advances in Neural Information Processing Systems (NIPS)*. 67, 72, 78, 82, 83, 88, 95

Zaheer, M., Li, C.-L., Póczos, B., and Salakhutdinov, R. (2017b). Gan connoisseur: Can gans learn simple 1d parametric distributions? *NIPS Workshop on Deep Learning: Bridging Theory and Practice*. 3

Zhang, Y., Liang, P., and Charikar, M. (2017). A hitting time analysis of stochastic gradient langevin dynamics. In *COLT*. 108, 109

Zhao, J., Kim, Y., Zhang, K., Rush, A. M., and LeCun, Y. (2017). Adversarially regularized autoencoders. In *International Conference on Machine Learning (ICML)*. 37, 69

Zhao, J., Mathieu, M., and LeCun, Y. (2017). Energy-based Generative Adversarial Network. In *International Conference on Learning Representations (ICLR)*. 9, 11, 19

Zuffi, S. and Black, M. J. (2015). The stitched puppet: A graphical model of 3d human shape and pose. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 94