SPECTRAL PROBABILISTIC MODELING AND APPLICATIONS TO NATURAL LANGUAGE PROCESSING

Ankur Parikh

August 2015
CMU-ML-15-102

SPECTRAL PROBABILISTIC MODELING AND APPLICATIONS TO NATURAL LANGUAGE PROCESSING

Ankur Parikh

August 2015
CMU-ML-15-102

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

Submitted in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

This research was sponsored by: the National Science Foundation Graduate Research Fellowship Program under grant numbers DGE0750271, 0946825, DGE1252522; National Science Foundation grant numbers IIS0713379 and IIS1218282; National Institutes of Health grant numbers R01GM093156 and P30DA035778; Defense Advanced Research Projects Agency grant number NBCH1080007; and the Pittsburgh Life Sciences Greenhouse.

Keywords: probabilistic graphical models, spectral methods, kernels, unsupervised parsing, language modeling

Acknowledgements

This thesis would not have been possible without the help of a great number of people. I would first like to thank my undergraduate advisors Profs. Sharad Malik and Rob Schapire for introducing me to the world of research and inspiring me to commit several years of my life to being a PhD student. It was definitely worth it.

I am very grateful to have Prof. Eric Xing as my thesis advisor, who played an instrumental role in helping me to pick impactful problems and construct a long term vision for my thesis that spanned both theory and application. With his guidance and the breadth of expertise represented by the SAILING lab, I had the opportunity to be able to do research in a diverse range of fields spanning machine learning, natural language processing, and computational biology.

I'm also very thankful to my other thesis committee members Le Song, Noah Smith, Geoff Gordon, John Platt, and previously Ben Taskar for their time and feedback that have helped to improve many parts of this thesis. Le's mentorship taught me how to tackle technically challenging problems and introduced me to the field of spectral learning. Noah has always made time to listen to my practice talks and give honest and useful feedback. Geoff and John have been really great to discuss machine learning ideas with, and Ben gave me some key suggestions on how to keep my thesis more focused.

In natural language processing (NLP), I have been fortunate to collaborate with and have learned a lot from Chris Dyer, Shay Cohen, Hoifung Poon, and Kristina Toutanova. Their guidance exposed me to an amazing area that I've become very passionate about. I'm also grateful to Avneesh Saluja who has been great to collaborate and discuss ideas with, and Brendan O'Connor who first sparked my interest in NLP by inviting me to Noah's reading group.

Although not part of my thesis, I have benefited greatly from my computational biology collaborations with Prof. Wei Wu and Micol Marchetti-Bowick. I'm also grateful to Mladen Kolar who helped me write my first publication and my other collaborators Mariya Ishteva, Qirong Ho, Asela Gunawardana, and Chris Meek.

The Machine Learning Department has been an awesome experience, and I'm very thankful to Diane Stidle, Mallory Deptola, and Michelle Martin for all their help, support, and patience in the many years I've been here.

Lastly, I'm very grateful to my parents, Prashant and Manjari Parikh, whose unwavering support for me to pursue my passion has been crucial to being able to complete this thesis. Without their advice and support, it would have been difficult to navigate the ups and downs of a PhD.

Abstract

Probabilistic modeling with latent variables is a powerful paradigm that has led to key advances in many applications such natural language processing, text mining, and computational biology. Unfortunately, while introducing latent variables substantially increases representation power, learning and modeling can become considerably more complicated. Most existing solutions largely ignore non-identifiability issues in modeling and formulate learning as a nonconvex optimization problem, where convergence to the optimal solution is not guaranteed due to local minima.

In this thesis, we propose to tackle these problems through the lens of linear/multi-linear algebra. Viewing latent variable models from this perspective allows us to approach key problems such as structure learning and parameter learning using tools such as matrix/tensor decompositions, inversion, and additive metrics. These new tools enable us to develop novel solutions to learning in latent variable models with theoretical and practical advantages. For example, our spectral parameter learning methods for latent trees and junction trees are provably consistent, local-optima-free, and 1-2 orders of magnitude faster than EM for large sample sizes.

In addition, we focus on applications in Natural Language Processing, using our insights to not only devise new algorithms, but also to propose new models. Our method for unsupervised parsing is the first algorithm that has both theoretical guarantees and is also practical, performing favorably to the CCM method of Klein and Manning. We also developed power low rank ensembles, a framework for language modeling that generalizes existing n-gram techniques to non-integer n. It consistently outperforms state-of-the-art Kneser Ney baselines and can train on billion-word datasets in a few hours.

Contents

1 Introduction **7**

1.1 Challenges . . . 8

1.2 A Linear Algebra Approach To Graphical Models . . . 9

1.3 Applications to Natural Language Processing . . . 9

1.4 Thesis Statement . . . 11

1.5 Related Work . . . 12

1.6 Outline . . . 12

2 Background: Modeling with Probabilistic Graphical Models **13**

2.1 Tasks in a graphical model . . . 16

2.2 Latent Variable Graphical Models . . . 20

2.3 Review of Hilbert Space Embedding of Distributions . . . 24

3 A Linear Algebra View of Latent Variable Models **33**

3.1 Tensor Notation . . . 33

3.2 The Spectral View . . . 35

I Spectral Learning Algorithms for Graphical Models **38**

4 A Spectral Algorithm for Latent Tree Graphical Models **39**

4.1 Intuition . . . 40

4.2 Notation for Latent Tree Graphical Models 41

4.3 Derivation of Spectral Algorithm 42

4.4 Dealing with $S_H \neq S_O$ 48

4.5 Sample Complexity Analysis 51

4.6 Empirical Results 51

4.7 Connections with Tensor Decomposition 52

4.8 Conclusion 55

5 A Spectral Algorithm for Latent Junction Trees **56**

5.1 Additional Tensor Notation 57

5.2 Latent Junction Trees 58

5.3 Tensor Representation for Message Passing 59

5.4 Transformed Representation 62

5.5 Observable Representation 63

5.6 Discussion 65

5.7 Sample Complexity 66

5.8 Experiments 66

5.9 Conclusion 68

5.10 Appendix 69

6 Nonparametric Latent Trees with Kernel Embeddings **83**

6.1 Notation 84

6.2 Kernel Density Estimation 85

6.3 Connection to Hilbert Space Embeddings 86

6.4 Deriving the Spectral Algorithm 87

6.5 Structure Learning of Latent Tree Graphical Models 90

6.6 Experiments 94

6.7 Conclusion 97

6.8 Appendix 98

7 Alternative Spectral Representation of Latent Tree Graphical Models 101

7.1 Intuition 102

7.2 Notation 103

7.3 Derivation of Alternate Spectral Algorithm 103

7.4 Observable Representation 105

7.5 Sample Complexity 107

7.6 Experiments 109

7.7 Discussion 112

7.8 Appendix 112

7.9 Eigenvalue Bounds 115

7.10 Putting it all together 123

II Spectral Models for Natural Language Processing 125

8 Spectral Unsupervised Parsing with Additive Tree Metrics 126

8.1 Introduction 126

8.2 Learning Setting and Model 128

8.3 Spectral Learning Algorithm based on Additive Tree Metrics 131

8.4 Experiments 139

8.5 Conclusion 143

8.6 Appendix 143

9 Language Modeling via Power Low Rank Ensembles 155

9.1 Introduction 155

9.2 Discount-based Smoothing 157

9.3 Power Low Rank Ensembles 158

9.4 Creating the Ensemble 161

9.5 Experiments 167

9.6 Machine Translation Task 170

9.7 Related Work 170

9.8 Conclusion 171

9.9 Appendix 171

10 Conclusion **177**

Chapter 1

Introduction

Probabilistic graphical models have become an indispensable framework in artificial intelligence (Pearl, 1988; Koller and Friedman, 2009; Murphy, 2012). Their ability to model and reason about complex uncertainty among large sets of variables has been an important driving force behind advances in many applications such as natural language processing(Manning and Schütze, 1999) and computational biology(Baldi et al., 2001) in the last two decades.

Four central themes in graphical models are:

1. **Structure Learning**: How do we determine the structural dependencies among a set of random variables?
2. **Parameter Learning**: Once the model structure has been determined, how can the parameters be learned from the data?
3. **Inference**: How can we reason or make predictions about a set of variables conditioned on the values of other variables?
4. **Modeling**: How to leverage these tools to construct effective models for real world phenomena?

For graphical models where all variables are observed in the data, many of the solutions to these algorithmic challenges are well studied. Often for simple structures, many of these problems have tractable solutions e.g. the Chow-Liu algorithm for structure learning of tree graphical models, maximum likelihood estimation with fully observed data for parameter learning, max-product for inference on trees. For more complex graphical models, structure learning, parameter learning, and inference are generally NP-hard but approximate solutions have been developed (Wainwright and Jordan, 2008; Koller and Friedman, 2009).

However, limiting ourselves to only using observed variables can be quite restrictive from the perspective of modeling. This is because, in many cases, the observed variables alone may not suffice to provide a concise explanation of the data. Consider the machine translation example shown in Figure 1.1. Here the goal is to translate the simple English sentence *"spectral learning is awesome "* into Spanish, which is *"aprendizaje espectral es impresionante"*. At first this task can seem daunting since the i^{th} word in the English sentence does not necessarily correspond to the i^{th} word in the Spanish sentence. For example, here "*spectral*", the first word in the English sentence corresponds to "*espectral*", the second word in the Spanish sentence. For more complicated sentences, whole phrases may be moved/inserted when translating to another language. Therefore, strictly modeling with only the observed variables, as informally shown in Figure 1.1(a), can produce very complicated dependencies.

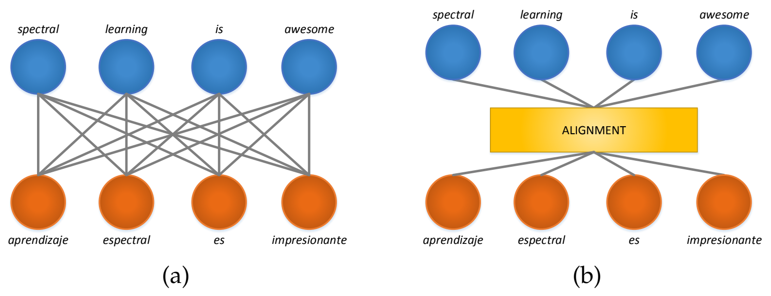


Figure 1.1: Translation example of how latent variables can provide simpler solutions to problems

Intuitively, the problem would be much simpler if we knew which English word mapped to each Spanish word i.e. the *word alignment*, as (informally) shown in Figure 1.1(b). Given the alignment function $\boldsymbol{a}$, translation becomes much easier, since to get the i^{th} spanish word, we can simply look up the Spanish translation of the $\boldsymbol{a}(i)^{th}$ english word. However, the alignment is not provided in the training data (which generally just consists of English/Spanish sentence pairs), and therefore is a *latent* (or hidden) variable.

1.1 Challenges

Unfortunately, while latent variables provide advantages for model design, they pose substantial challenges for structure learning, parameter learning, and modeling. There exist many problems that are easy for fully observed models but are considerably more difficult for latent variable models. For example, maximum likelihood estimation with latent variables for structure/parameter learning is generally NP-hard. Similarly, while observed models are always identifiable, the same is generally not true for latent variable models, making evaluation and interpretation complicated.

The vast majority of existing approaches addressing these challenges rely on local-search heuristics based on either greedy search or non-convex optimization. While in many cases these methods work well in practice, they often require careful initialization and problem-specific heuristics to yield good results. For example, unsupervised parsing performance with random initializations can vary greatly, making carefully crafted initializers essential to reliable results. In addition, the sequential nature of these existing methods can often make parallelization challenging and thus scalability non-trivial in today's distributed computing paradigm.

From the theoretical perspective, these approaches often do not shed light into the nature and complexity of latent models making it difficult to ascertain what makes the problem difficult, and whether certain relaxations can lead to tractable solutions.

1.2 A Linear Algebra Approach To Graphical Models

In this thesis, we tackle these challenges from a different perspective revolving around linear algebra. From the linear algebra point of view, latent variables induce low rank dependencies among the observed variables. The *rank* of the latent space can then be theoretically used to quantify the complexity/hardness of learning in the latent variable model. As we will see, when the rank becomes very large, the problem will become intractable, and we will be forced to resort to heuristic search methods. However, in the *low rank* scenario, which we will argue occurs more often in practice, we can leverage tools from linear algebra to provide provably consistent solutions in many cases.

Our work is inspired by recent theoretical results from different communities such as dynamical systems (Katayama, 2005), theoretical computer science (Dasgupta, 1999), statistical machine learning (Hsu et al., 2009; Bailly et al., 2009), and phylogenetics (Mossel and Roch, 2005). Before this thesis however, the majority of spectral learning methods were limited to hidden Markov models (Hsu et al., 2009; Bailly et al., 2009; Siddiqi et al., 2010; Song et al., 2010a).

In this dissertation, we take a more general view, proposing not just to leverage low rank matrix factorization (although that is certainly a major component), but also higher order tensor operations, additive tree metrics, and other tools to present theoretically principled and practical spectral solutions to a broad class of problems. In particular, we seek to focus on parameter learning, structure learning, and modeling with latent variables from the linear algebra point of view.

1.3 Applications to Natural Language Processing

From the application standpoint, we seek to use these theoretical insights to develop new models and algorithms for Natural Language Processing (NLP) tasks. We focus on unsupervised parsing and language modeling, two probabilistic modeling problems which we believe can benefit substantially from the linear algebra perspective.

1.3.1 Unsupervised Parsing

Unsupervised parsing (also known as grammar induction) is the problem of discovering syntactic structure in sentences without the help of annotated training examples marked with syntactic trees. Solutions to the problem of grammar induction have been long sought after since the early days of computational linguistics and are interesting both from cognitive and engineering perspectives. Cognitively, it is more plausible to assume that children obtain only terminal strings of parse trees and not the actual parse trees, which means the unsupervised setting may be a better model for studying language acquisition.

From the engineering perspective, training data for unsupervised parsing exists in abundance (i.e. sentences and part-of-speech tags), and is much cheaper than data required for supervised training, which requires manual syntactic annotation.

Most of the solutions suggested treat the problem of unsupervised parsing by assuming a parametric model, which is then estimated by identifying a local maximum of an objective function such as the likelihood (Klein and Manning, 2004) or a variant of it (Cohen and Smith, 2009; Headden et al., 2009; Spitkovsky et al., 2010b; Gillenwater et al., 2010; Golland and DeNero, 2012). Unfortunately, finding the global maximum for these objective functions is usually intractable (Cohen and Smith, 2010). As a result, many of these methods suffer from severe local-optima and initializers are crafted to obtain good solutions.

In this thesis, we take a very different approach to unsupervised parsing. We propose to formulate unsupervised parsing as a latent structure learning problem where the latent structure (parse tree) varies for each example. Our goal is to then leverage linear algebra tools to derive a provably correct learning algorithm that also works empirically well in practice.

1.3.2 Language Modeling

Language modeling - the task of assigning probabilities to sequences of words, is an important component of many NLP and speech systems. It is a seemingly simple, yet actually very challenging problem that is useful in a number of applications e.g. machine translation (Koehn, 2010) since it allows one to determine which candidate sequences are more likely than others. The predominant approach to language modeling is the n-gram language model, wherein the probability of a word sequence $P(w_1, \ldots, w_m)$ is first factored and then approximated (with the Markov assumption) as:

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i | w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i | w_{i-n+1}^{i-1})$$

In other words, one only needs to take into account the previous $n-1$ words when computing the probability of a word w_i given its word history. This assumption reduces parameters significantly, but it is not enough. Due to the large vocabulary space, maximum likelihood approaches will often assign zero probability to word sequences that were unseen in the training data (but can be in the test data).

A rich literature in language model (LM) *smoothing* has thus arisen in response to this core issue, with the basic idea behind most approaches being to *interpolate* with or *back off* to lower order n-gram models (which are less sparse) as the need arises (Chen and Goodman, 1999). While surprisingly simple, these techniques, in particular Kneser-Ney smoothing (Kneser and Ney, 1995) and variants, have often been the state of the art for more than a decade. However, these approaches fail to exploit the semantic/syntactic relatedness among words that can intuitively be exploited to help alleviate the data sparsity problem in a more efficient manner.

As part of this dissertation, we examine how ensembles of specially constructed low rank matrices/tensors can be leveraged to provide a novel solution to the language modeling problem. Our proposed solution contains Absolute Discounting(Ney et al., 1994) and Kneser Ney (Kneser and Ney, 1995) as special cases of our general framework. Our approach can take incorporate semantic relatedness among words to improve performance while preserving scalability to large datasets.

1.4 Thesis Statement

The central theme of this thesis revolves around the following statement:

Viewing latent variable models through the lens of linear/multi-linear algebra allows us to approach key problems such as structure and parameter learning using tools such as matrix/tensor decompositions, inversion, and hilbert space operators. These new tools enable us to develop novel solutions for learning and inference in latent variable models that have both theoretical and practical advantages. In addition, these insights aid us in designing new models and algorithms for language modeling and unsupervised parsing in Natural Language Processing.

Key Contributions

1. **Chapter 4: A Spectral Algorithm for Latent Tree Graphical Models** - We leverage tensor algebra to derive a provably consistent parameter learning algorithm for latent trees. Our approach gives comparable or better accuracy to Expectation Maximization, while being 10-100x faster in a variety of settings (*Key theme: parameter learning*). Preliminary version: Parikh et al. (2011).
2. **Chapter 5: A Spectral Algorithm for Latent Junction Trees** - We generalize the work from the previous chapter to low-treewidth graphical models via junction trees (*Key theme: parameter learning*). Preliminary version: Parikh et al. (2012)
3. **Chapter 6:Kernel Embeddings of Latent Tree Graphical Models** - Using Hilbert Space Embeddings (Smola et al., 2007), we develop algorithms for latent variable parameter and structure learning in latent tree graphical models with continuous, non-Gaussian variables (*Key themes: parameter and structure learning*). Preliminary version: Song et al. (2011b).
4. **Chapter 7: Alternate Spectral Algorithm for Latent Tree Models** - We show that the spectral algorithm derived in Chapter 4 is not unique and there exists another spectral representation that only requires tensors of order 3 regardless of the tree topology. Both factorizations are compared empirically in different settings (*Key theme: parameter learning*). Preliminary version: Parikh et al. (2011)
5. **Chapter 8: Spectral Unsupervised Parsing with Additive Tree Metrics** - We propose a novel model for unsupervised parsing that revolves around structure learning of projective latent trees where the latent structure changes for each example. Unlike existing approaches, our method has provable guarantees on structure recovery and also performs well empirically (*Key themes: structure learning and modeling*). Preliminary version: Parikh et al. (2014a)
6. **Chapter 9: Language Modeling with Power Low Rank Ensembles** - We develop a novel low rank framework of n-gram language models where existing methods such as Absolute Discounting and Kneser Ney are special cases. Our approach allows n-grams of non-integer n i.e. 2.5-grams, 1.5-grams etc. and consistently outperforms state-of-the-art Kneser Ney baselines without sacrificing the computational advantages of n-gram based methods (*Key theme: modeling*). Preliminary version: Parikh et al. (2014b)

.

1.5 Related Work

As mentioned previously, the basis for this dissertation is built on work from many different communities such as dynamical systems, (Katayama, 2005), theoretical computer science (Dasgupta, 1999), statistical machine learning (Hsu et al., 2009; Bailly et al., 2009), and phylogenetics (Mossel and Roch, 2005). Before this thesis however, the majority of spectral learning methods were limited to hidden Markov models (Hsu et al., 2009; Bailly et al., 2009; Siddiqi et al., 2010; Song et al., 2010a).

In this work we take a more general graphical models point of view, leveraging tensor algebra to show that spectral approaches can lead to principled and practical solutions for a wide class of problems in graphical models and Natural Language Processing. Our approach allows us to gain a deeper understanding of spectral learning and its connections to tensor algebra. Furthermore, it allows us to use these insights to develop novel models and solutions for new domains.

Concurrently with this thesis, other researchers have also developed spectral learning methods for other problems such as predictive state representations (Boots et al., 2010), topic models (Anandkumar et al., 2012, 2013), latent probabilistic context free grammars (Cohen et al., 2012; Cohen and Collins, 2012) and other models (Bailly et al., 2010; Balle et al., 2011; Luque et al., 2012; Dhillon et al., 2012a; Balle et al., 2012; Boots and Gordon, 2013; Zhang et al., 2014; Chaganty and Liang, 2014; Quattoni et al., 2014; Wang and Zhu, 2014; Subakan et al., 2014; Saluja et al., 2014).

In particular, the works of Cohen et al. (2012); Cohen and Collins (2012) and Dhillon et al. (2012a); Melnyk and Banerjee (2014) leverage the tensor formulation we developed in this thesis.

Another line of work that seeks to develop alternative learning methods for latent variable models is that of anchor-based methods, first proposed by Arora et al. (2012b), and followed up by Arora et al. (2012a); Ding et al. (2013); Cohen and Collins (2014); Nguyen et al. (2014). Compared to spectral approaches, these methods make a rather different set of assumptions, assuming that for each latent state there exists a feature that can only be generated by that state. While it would be very interesting to compare these approaches directly with spectral methods, it is beyond the scope of the thesis.

1.6 Outline

Before describing the primary contributions of this thesis in more detail, some background on graphical models and kernel methods is presented in Chapter 2. We then discuss a linear algebra perspective on latent variable models in Chapter 3 which serves as a foundation for our work. Chapters 4, 5, 6, 7, 8, and 9 then present the main contributions of the thesis.

Chapter 2

Background: Modeling with Probabilistic Graphical Models

This chapter largely serves to give a brief background of probabilistic graphical models (Pearl, 1988; Koller and Friedman, 2009; Murphy, 2012), a popular and elegant paradigm for modeling random variables, that is the basis for this thesis. We also give a brief tutorial on Hilbert space embeddings (Smola et al., 2007), a concept central to Chapter 6.

2.0.1 Motivation

Consider p random variables $\mathcal{X} = \{X_1, ..., X_p\}$, where each variable is allowed to take on m states i.e. $X_i \in \{0, ..., m-1\}$ $\forall X_i \in \mathcal{X}$. Then there are m^p different settings of the random variables, each of which can be assigned a different probability.

First consider making no assumptions on the underlying distribution $p(X_1, ..., X_p)$ so that there are no constraints among the probabilities of different assignments (with the exception that they must sum to one). For example, for $p = 3, m = 2$ we would have:

$$\begin{aligned}
P(X_1 = 0, X_2 = 0, X_3 = 0) &= \theta_{000} \\
P(X_1 = 0, X_2 = 0, X_3 = 1) &= \theta_{001} \\
P(X_1 = 0, X_2 = 1, X_3 = 0) &= \theta_{010} \\
P(X_1 = 0, X_2 = 1, X_3 = 1) &= \theta_{011} \\
P(X_1 = 1, X_2 = 0, X_3 = 0) &= \theta_{100} \\
P(X_1 = 1, X_2 = 0, X_3 = 1) &= \theta_{101} \\
P(X_1 = 1, X_2 = 1, X_3 = 0) &= \theta_{110} \\
P(X_1 = 1, X_2 = 1, X_3 = 1) &= \theta_{111}
\end{aligned} \tag{2.1}$$

where the parameters $\theta_{000}, ...\theta_{111}$ must be estimated from data. Note that all but one of the parameters can be set arbitrarily, since the only constraints are that the parameters are non-negative and that the distribution must sum to one.

However, in this scenario, statistical estimation of parameters (learning) becomes very challenging. Consider the following lemma from (Roy, 2011):

Lemma 1. *Let $p_{\boldsymbol{\theta}}$ be a multinomial distribution with B bins and parameters $\boldsymbol{\theta}$ i.e. bin i is associated with probability $\boldsymbol{\theta}_i$. Let $\hat{\boldsymbol{\theta}}$ be the maximum likelihood estimate (MLE) of the multinomial parameters from N iid samples from $p_{\boldsymbol{\theta}}$. Then,*

$$\mathbb{E}[KL(p_{\boldsymbol{\theta}}|p_{\hat{\boldsymbol{\theta}}})] \leq \frac{B-1}{N} \tag{2.2}$$

Since $B = m^p$, Lemma 1 implies that $N \in \Omega(m^p)$ for $\mathbb{E}[KL(p_{\boldsymbol{\theta}}|p_{\hat{\boldsymbol{\theta}}})] \to 0$, making statistical estimation challenging for more than a handful of variables.

Similarly, this approach also leads to severe computational problems when attempting to do **inference**, e.g. computing lower order marginals. Let S and T be two sets of variables $\mathcal{X}_S, \mathcal{X}_T \subseteq \mathcal{X}$ s.t. $\mathcal{X}_S \cup \mathcal{X}_T = \mathcal{X}$ and let $\boldsymbol{x}_S, \boldsymbol{x}_T$ indicate a possible set of instantiations.

$$P(\mathcal{X}_S = \boldsymbol{x}_S) = \sum_{\boldsymbol{x}_T} P(\mathcal{X}_S = \boldsymbol{x}_S, \boldsymbol{x}_T) \tag{2.3}$$

where the summation would have computational complexity $\Theta(m^{|\mathcal{X}_T|})$ i.e. exponential with respect to the number of variables being summed out.

As a result, the brute force representation of probability distributions, while very expressive from the modeling point of view, is statistically and computationally prohibitive.

2.0.2 Leveraging Independencies

Let us now consider the other extreme, showing that we can severely restrict modeling power to increase computational and statistical efficiency.

Assume that all $X_1, ..., X_p$ are mutually independent, implying that the factorization below:

$$P(X_1, ..., X_p) = \prod_{i=1}^{p} P(X_i) \tag{2.4}$$

Considering our $m = 2, p = 3$ example again we have that:

$$\begin{aligned}
p(X_1 = 0, X_2 = 0, X_3 = 0) &= \theta_{X_1=0}\theta_{X_2=0}\theta_{X_3=0} \\
p(X_1 = 0, X_2 = 0, X_3 = 1) &= \theta_{X_1=0}\theta_{X_2=0}\theta_{X_3=1} \\
p(X_1 = 0, X_2 = 1, X_3 = 0) &= \theta_{X_1=0}\theta_{X_2=1}\theta_{X_3=0} \\
p(X_1 = 0, X_2 = 1, X_3 = 1) &= \theta_{X_1=0}\theta_{X_2=1}\theta_{X_3=1} \\
p(X_1 = 1, X_2 = 0, X_3 = 0) &= \theta_{X_1=1}\theta_{X_2=0}\theta_{X_3=0} \\
p(X_1 = 1, X_2 = 0, X_3 = 1) &= \theta_{X_1=1}\theta_{X_2=0}\theta_{X_3=1} \\
p(X_1 = 1, X_2 = 1, X_3 = 0) &= \theta_{X_1=1}\theta_{X_2=1}\theta_{X_3=0} \\
p(X_1 = 1, X_2 = 1, X_3 = 1) &= \theta_{X_1=1}\theta_{X_2=1}\theta_{X_3=1}
\end{aligned} \tag{2.5}$$

where we have 3 constraints (in addition to non-negativity):

$$\begin{aligned}
\theta_{X_1=0} + \theta_{X_1=1} &= 1 \\
\theta_{X_2=0} + \theta_{X_2=1} &= 1 \\
\theta_{X_3=0} + \theta_{X_3=1} &= 1
\end{aligned} \tag{2.6}$$

Generally, p independent variables mean $O(pm)$ parameters which is much fewer than the $O(m^p)$ required previously. This results in statistical and computational tractability. Statistically, Lemma 1 means $N = \Omega(mp)$ is sufficient for $\mathbb{E}[KL(p_{\boldsymbol{\theta}}|p_{\hat{\boldsymbol{\theta}}})] \to 0$.

Computationally, marginalization can be computed efficiently since the sum can be distributed inside the factors. Again let S and T be two sets of variables $\mathcal{X}_S, \mathcal{X}_T \subseteq \mathcal{X}$ s.t. $\mathcal{X}_S \cup \mathcal{X}_T = \mathcal{X}$ and let $\boldsymbol{x}_S, \boldsymbol{x}_T$ indicate a possible set of instantiations. Then,

$$P(\mathcal{X}_S = \boldsymbol{x}_S) = \sum_{\boldsymbol{x}_T \in \mathcal{X}_T} P(\mathcal{X}_S = \boldsymbol{x}_S, \boldsymbol{x}_T) = \prod_{s \in S} P(X_s = x_s) \prod_{t \in T} \sum_{x_t} P(X_t = x_t) \tag{2.7}$$

2.0.3 Probabilistic Graphical Models

Graphical models provide a formal framework for trading off model expressivity with statistical/computational tractability of learning and inference.

A graphical model is a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where the vertices ($\mathcal{V}$) correspond to random variables and the edges ($\mathcal{E}$) indicate dependencies. For this work we will assume that the graph is directed and acyclic (a DAG) and thus is also known as a Bayesian network.

The connectivity of the graph indicates the independence assumptions made by the graph. No edges as shown in Figure 2.1(a) indicates that all the variables are independent while a clique (all edges present) as shown in Figure 2.1(b) implies no conditional independences.

Most of the rest of the thesis (with the exception of Chapter 5) is focused on graphical models where $\mathcal{G}$ is a tree. In tree-shaped graphical models, two nodes X_i and X_j become conditionally independent conditioned on any variable in the path between X_i and X_j. The rigorous definition

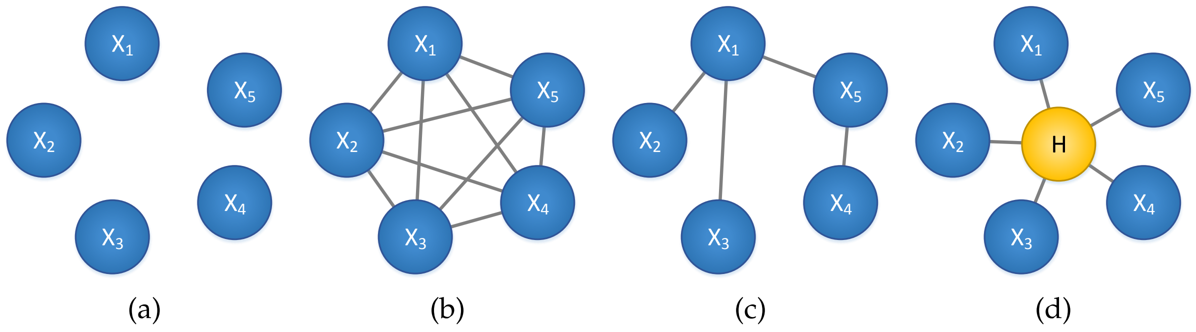


Figure 2.1: Example graphical model structures

is provided below:

Definition 1. *Given a tree graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $X_i, X_j \in \mathcal{V}$ and let $\mathcal{Z}$ denote the set of variables on the (unique) path between X_i and X_j. Let $\mathcal{A}$ be some subset of $\mathcal{V}$. Then $X_i \perp X_j | \mathcal{A}$ if and only if $A \cap \mathcal{Z} \neq \emptyset$.*

Note that the independences implied by the structure are unrelated to the direction of the edges in a tree graphical model. (For conditional independencies induced by more complex graph structures such as v-structures, please see Koller and Friedman (2009)). This gives the following corollary:

Corollary 1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a tree graphical model directed via an arbitrary root X_r. A particular distribution p respects the independences in $\mathcal{G}$ if and only if:*

$$P(X_1, ..., X_p) = \prod_{i=1}^{p} P(X_i | X_{\pi(i)}) \tag{2.8}$$

where $X_\pi(i)$ denotes the parent of X_i and $X_{\pi(r)} = \emptyset$ for notational convenience.

For example, consider the tree in Figure 2.1(c). Choosing X_1 as the root gives the following factorization:

$$\begin{aligned} P(X_1, X_2, X_3, X_4, X_5) &= \prod_{i=1}^{p} P(X_i | X_{\pi(i)}) \\ &= P(X_1)P(X_2|X_1)P(X_3|X_1)P(X_5|X_1)P(X_4|X_5) \end{aligned} \tag{2.9}$$

2.1 Tasks in a graphical model

Structure learning, parameter learning, and inference are the three main algorithmic tasks in graphical models. As we will show below, when $X_1, .., X_p$ are all observed in the training data, these tasks have elegant and principled solutions (atleast for tree-shaped models).

2.1.1 Parameter Learning

Typically parameter learning is done via maximum likelihood estimation i.e. finding the set of parameters that maximize the log-likelihood of the data:

$$\hat{\boldsymbol{\theta}}_{MLE} = \operatorname*{argmax}_{\theta} \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)}|\boldsymbol{\theta}) \tag{2.10}$$

where $n = 1, .., N$ indexes the N data points and $\boldsymbol{x}^{(n)} = \{x_1^{(n)}, ..., x_p^{(n)}\}$ is the n^{th} sample. Depending on the graph structure, $p(\boldsymbol{x}^{(n)}, \boldsymbol{\theta})$ factorizes into local probabilities, making the problem statistically tractable. For instance, for tree shaped models, the log-likelihood decomposes as follows:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{MLE} &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{i \in \mathcal{V}} \log P(x_i^{(n)}|x_{\pi(i)}^{(n)}, \boldsymbol{\theta}) \\
\hat{\boldsymbol{\theta}}_{MLE} &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{i \in \mathcal{V}} \sum_{a=1}^{m} \sum_{b=1}^{m} \mathbb{I}[x_i^{(n)} = a \wedge x_{\pi(i)}^{(n)} = b] \log \theta_{X_i=a|X_{\pi(i)}=b}
\end{aligned} \tag{2.11}$$

with the constraints that θ is non-negative and that $\sum_a \boldsymbol{\theta}_{X_i=a|X_{\pi(i)=b}} = 1 \, \forall i, b$. Taking the derivative and setting equal to zero will show that this reduces to simply computing the conditional probability estimates from the data:

$$\hat{\theta}_{X_i=a|X_{\pi(i)}=b} = \hat{P}(X_i = a|X_{\pi(i)} = b) = \frac{\text{count}(X_i = a, X_{\pi(i)} = b)}{\text{count}(X_{\pi(i)} = b)} \tag{2.12}$$

where $\hat{P}$ denotes the empirical probability estimate and count$(\cdot)$ returns the number of data instances for which the argument $\cdot$ holds true.

2.1.2 Structure Learning

Given N samples of p variables $X_1, ..., X_p$ the goal of structure learning is to find the structure (set of edges $\mathcal{E}$) that maximizes the log-likelihood of the data $\mathcal{D}$. This problem is in NP-hard if arbitrary structures are permitted (Koller and Friedman, 2009).

However, if we limit the class of structures considered to trees, then a provably optimal algorithm, typically called the Chow-Liu algorithm, exists (Edmonds, 1967; Chow and Liu, 1968a). The key to the algorithm is the following likelihood decomposition:

$$\begin{aligned}
\log P(\mathcal{D}|\hat{\theta}, \mathcal{G}) &= \sum_{n=1}^{N} \sum_{i=1}^{p} \log \hat{P}(x_i^{(n)}|x_{\pi(i)}^{(n)}) \\
&\propto \sum_{x_i, x_{\pi(i)}} \sum_{i=1}^{p} \hat{P}(x_i, x_{\pi(i)}) \log \hat{P}(x_i|x_{\pi(i)}) \\
&= \sum_{i=1}^{p} \sum_{x_i, x_{\pi(i)}} \hat{P}(x_i, x_{\pi(i)}) \log \left(\frac{\hat{P}(x_i, x_{\pi(i)})}{\hat{P}(x_{\pi(i)})} \right) \\
&= \sum_{i=1}^{p} \sum_{x_i, x_{\pi(i)}} \hat{P}(x_i, x_{\pi(i)}) \log \left(\frac{\hat{P}(x_i, x_{\pi(i)})\hat{P}(x_i)}{\hat{P}(x_{\pi(i)})\hat{P}(x_i)} \right) \\
&= \sum_{i=1}^{p} \sum_{x_i, x_{\pi(i)}} \hat{P}(x_i, x_{\pi(i)}) \log \left(\frac{\hat{P}(x_i, x_{\pi(i)})}{\hat{P}(x_{\pi(i)})\hat{P}(x_i)} \right) + \sum_{i=1}^{p} \sum_{x_i, x_{\pi(i)}} \hat{P}(x_i, x_{\pi(i)}) \log \hat{P}(x_i) \\
&= \sum_{i=1}^{p} MI(x_i, x_{\pi(i)}) - \sum_{i=1}^{p} H(x_i) \qquad (2.13)
\end{aligned}$$

where $H(\cdot)$ denotes entropy and $MI(\cdot)$ denotes mutual information. Since $H(x_i)$ does not depend on the graph structure, the best structure is the one that maximizes $\sum_{i=1}^{p} MI(X_i, X_{\pi(i)})$. This can be done using a maximum spanning tree algorithm since we are limiting the structure to a tree. The resulting method is shown in Algorithm 1.

Algorithm 1 Calculate most likely tree structure $\mathcal{G}$ given data (Edmonds, 1967; Chow and Liu, 1968a)

```
In: N samples of p discrete variables: $\mathcal{D} = \{x_1^{(n)}, ..., x_p^{(n)}\}_{n=1}^{N}$
Out: tree structure $\mathcal{G}$ with p nodes
  for each $(X_i, X_j)$ s.t. $1 \leq i, j \leq p$ and $i \neq j$ do
    Compute mutual information estimate $MI(X_i, X_j) = \sum_{x_i, x_j} \hat{P}(x_i, x_j) \log\left(\frac{\hat{P}(x_i, x_j)}{\hat{P}(x_i)\hat{P}(x_j)}\right)$
  end for
  Let $\mathcal{G}_{full}$ be the complete undirected graph where each node i corresponds to variable $X_i$ and
  the weight of each edge (i, j) equals $MI(X_i, X_j)$
  Return $\mathcal{G}$, a maximum spanning tree of $\mathcal{G}_{full}$
```

2.1.3 Inference

Mathematically, the goal of inference is to compute the probabilities a subset of variables conditioned on the values of other variables. As mentioned earlier, attempting to brute force this summation can result in exponential computational complexity.

However, for trees the inference problem is tractable and can be solved exactly via message passing / belief propagation (Pearl, 1988). The key insight is to use the conditional independence statements of the model to push some of the factors outside certain summations that they do

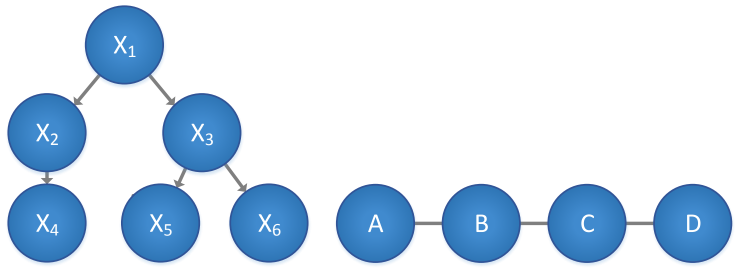


Figure 2.2: Example graphical models for belief propagation.

not depend on. For simplicity, let us focus on the graphical model in Figure 2.2(a) and consider computing the marginal probability of all the leaf nodes fixed to certain evidence values while summing out all the internal variables.

The algorithm works bottom up from the leafs, with each node gathering all the messages from its children and passing a new message to its parent. The message at the root equals the desired probability.

The outgoing message from the leaf to its parent is:

$$\boldsymbol{m}_{i\to\pi(i)}(x_{\pi(i)}) = P(\bar{x}_i|X_\pi) = \sum_{x_i} P(X_i|X_{\pi(i)})\mathbb{I}[x_i = \bar{x}_i] \tag{2.14}$$

where the $\mathbb{I}[x_i = \bar{x}_i]$ (indicator) function is 1 if and only if x_i equals the corresponding evidence value $\bar{x}_i$.

Example: Assume all the leaf nodes in Figure 2.2(a) are evidence variables gives:

$$\begin{aligned}
\boldsymbol{m}_{4\to 2}(x_2) &= P(\bar{x}_4|x_2) \\
\boldsymbol{m}_{5\to 3}(x_3) &= P(\bar{x}_5|x_3) \\
\boldsymbol{m}_{6\to 3}(x_3) &= P(\bar{x}_6|x_3)
\end{aligned} \tag{2.15}$$

At internal nodes, the outgoing message is:

$$\boldsymbol{m}_{i\to\pi(i)}(x_{\pi(i)}) = P(\bar{\boldsymbol{x}}_{\ell(i)}|X_\pi) = \sum_{x_i} P(X_i|X_{\pi(i)}) \prod_{j\in c(i)} \boldsymbol{m}_{j\to i}(x_i) \tag{2.16}$$

where $c(i)$ is the set of the children of i and $\boldsymbol{x}_{\ell(i)}$ is the set of leaves in the subtree rooted at X_i. Note that the message from node X_i to its parent $X_{\pi(i)}$ can only be computed after all the messages of its children have been computed.

Example: Assuming the internal non-root nodes in Figure 2.2(a) are non-evidence variables gives:

$$\begin{aligned}
\boldsymbol{m}_{2\to 1}(x_1) &= \sum_{x_2} P(x_2|x_1)\boldsymbol{m}_{4\to 2}(x_2) = \sum_{x_2} P(x_2, \bar{x}_4|x_1) \\
\boldsymbol{m}_{3\to 1}(x_1) &= \sum_{x_3} P(x_3|x_1)\boldsymbol{m}_{5\to 3}(x_3)m_{6\to 3}(x_3) = \sum_{x_3} P(x_3, \bar{x}_5, \bar{x}_6|x_1)
\end{aligned} \tag{2.17}$$

Finally at the root,

$$P(\boldsymbol{x}_{\mathcal{E}} = \bar{\boldsymbol{x}}_{\mathcal{E}}) = \sum_{x_i} P(X_i) \prod_{j \in c(i)} \boldsymbol{m}_{j\to i}(x_i) \tag{2.18}$$

where $\mathcal{E}$ denotes the set of evidence variables (assumed to be all the leaves in this case for simplicity).

Example: For the root node in Figure 2.2(a) we have:

$$P(\bar{x}_4, \bar{x}_5, \bar{x}_6) = \sum_{x_1} P(x_1)m_{2\to 1}(x_1)m_{3\to 1}(x_1) = \sum_{x_1} P(x_1, \bar{x}_4\bar{x}_5, \bar{x}_6) \tag{2.19}$$

Note here that each message can be computed in polynomial time: $O(m^2)$ if all the variables take on m states.

2.1.4 Consistency

The procedures we have described above have a very desirable statistical property called *consistency*. In particular, if the data is generated by assumed model, then structure learning, parameter learning, and inference (as described in the previous sections for trees) will return the correct structures/parameters/probabilities respectively as the training data size goes to infinity. This can be mathematically expressed as follows:

Lemma 2. *Assume that underlying distribution $p(\boldsymbol{x})$ is generated by a tree graphical model with graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and parameters $\boldsymbol{\theta}$. Given N samples generated from $p(\boldsymbol{x})$, we learn $\widehat{\mathcal{G}}$ from the method in 2.1.2, $\hat{\boldsymbol{\theta}}$ from 2.1.1, and estimate the probability $\hat{P}(\boldsymbol{x})$ from message passing from 2.1.3. Then,*

$$\begin{aligned}
&\widehat{\mathcal{G}} \to \mathcal{G} \textit{ as } N \to \infty \\
&\hat{\boldsymbol{\theta}} \to \boldsymbol{\theta} \textit{ as } N \to \infty \\
&\hat{P}(\boldsymbol{x}) \to P(\boldsymbol{x}) \ \forall \boldsymbol{x} \textit{ as } N \to \infty
\end{aligned} \tag{2.20}$$

2.2 Latent Variable Graphical Models

Let us now attempt to model a real world scenario using these techniques. Assume that there are patients entering a doctor's office with various symptoms e.g. sneezing, coughing etc. These symptoms can be modeled as observed variables since they are easily measured. Consider reasoning about these variables under the graphical model structures shown in Figure 2.1(a)-(c) where we let each of the X_i's indicate a different symptom. The model in Figure 2.1(a) is clearly unreasonable since it assumes all the symptoms are independent. Figure 2.1(b) doesn't make any modeling assumptions, but its computationally and statistically intractable. Figure 2.1(c) also seems suboptimal since in reality all the variables are quite dependent even if we know what a few of them are.

However, there is intuitively a simple explanation underlying the data. In particular if we know the the underlying illness the patient has then it is reasonable to assume the symptoms are conditionally independent given this illness. This "illness" is a latent (hidden) variable since it is not something that appears in the data but something we assume to exist.

Incorporating this latent variable leads to the graphical model in Figure 2.1(d) where the orange node H intuitively represents the illness, and all the symptoms are conditionally independent given H. Note that the model is rather compact since the number of edges is linear in the number of variables.

More generally, latent variables offer expressive modeling power without sacrificing statistical efficiency since they can model complicated dependencies with a compact number of edges. However, as we will see below, the cost usually comes in the form of computational efficiency. While learning in observed variables is often based on strong theoretical foundations (e.g. consistency), learning in latent variable models is challenging and largely tackled with local search methods that do generally have any guarantees on consistency.

2.2.1 Parameter Learning

The challenge in learning is most easily apparent in parameter learning of latent variable models. Since a subset of the variables are no longer observed, the goal in parameter learning is now to learn the set of parameters that maximizes the *marginal* log-likelihood:

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{MLE} &= \max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)}, \boldsymbol{\theta}) \\ &= \max_{\theta} \sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} p(\boldsymbol{x}^{(n)}, \boldsymbol{h}, \boldsymbol{\theta}) \qquad (2.21)\end{aligned}$$

where $\boldsymbol{x}$ denotes the observed variables and $\boldsymbol{h}$ denotes the latent variables. Unlike fully observed models, the function above is non-concave due to the summation inside the logarithm, which makes the optimization problem NP-hard. Existing optimization strategies mostly revolve around local search and are therefore prone to being trapped in a local optima. One popular approach is Expectation Maximization (EM) (Dempster et al., 1977).

EM aims to maximize a surrogate objective function that is a lower bound on the original objective. Let $Q(\boldsymbol{h}|\boldsymbol{x}^{(n)})$ be some probability distribution. Then the surrogate function can be derived via Jensen's Inequality:

$$\begin{aligned}
\sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} p(\boldsymbol{x}^{(n)}, \boldsymbol{h}, \boldsymbol{\theta}) &= \sum_{n=1}^{N} \log \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \frac{p(\boldsymbol{x}^{(n)}, \boldsymbol{h}, \boldsymbol{\theta})}{Q^{(t+1)}(\boldsymbol{h}|\boldsymbol{x}^{(n)})} \\
&\geq \sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log \frac{p(\boldsymbol{x}^{(n)}, \boldsymbol{h}, \boldsymbol{\theta})}{Q(\boldsymbol{h}|\boldsymbol{x}^{(n)})} \quad \text{(using Jensen's Inequality)} \\
&= \sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log p(\boldsymbol{x}^{(n)}, \boldsymbol{h}, \boldsymbol{\theta}) - \sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \qquad (2.22)
\end{aligned}$$

EM performs coordinate descent on Eq. 2.22, alternatively maximizing $\boldsymbol{\theta}$ where Q is fixed (called the M-step) and maximizing Q when $\boldsymbol{\theta}$ is fixed (called the E-step).

M-step (optimize in terms of $\boldsymbol{\theta}$): The second term in Eq. 2.22 is fixed as a function of $\boldsymbol{\theta}$ so we are left with only optimizing the first term, which is essentially a weighted log-likelihood that is concave:

$$\hat{\boldsymbol{\theta}}_{MLE} \leftarrow \max_{\boldsymbol{\theta}} \sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log p(\boldsymbol{x}^{(n)}, \boldsymbol{h}, \boldsymbol{\theta}) \qquad (2.23)$$

E-step (optimize in terms of Q): Eq. 2.22 can be rewritten as

$$\begin{aligned}
\sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log \frac{p(\boldsymbol{x}^{(n)}, \boldsymbol{h}, \boldsymbol{\theta})}{Q(\boldsymbol{h}|\boldsymbol{x}^{(n)})} &= \sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log \frac{p(\boldsymbol{h}|\boldsymbol{x}^{(n)}, \theta) p(\boldsymbol{x}^{(n)}, \boldsymbol{\theta})}{Q(\boldsymbol{h}|\boldsymbol{x}^{(n)})} \\
&= \sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log p(\boldsymbol{x}^{(n)}, \boldsymbol{\theta}) - \sum_{n=1}^{N} \sum_{\boldsymbol{h}} Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) \log \frac{p(\boldsymbol{h}|\boldsymbol{x}^{(n)}, \boldsymbol{\theta})}{Q(\boldsymbol{h}|\boldsymbol{x}^{(n)})} \\
&= \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)}, \boldsymbol{\theta}) - \sum_{n=1}^{N} KL\left(Q(\boldsymbol{h}|\boldsymbol{x}^{(n)}) || p(\boldsymbol{h}|\boldsymbol{x}^{(n)}, \boldsymbol{\theta})\right) \qquad (2.24)
\end{aligned}$$

where KL indicates KL divergence. Thus, maximizing Eq. 2.24 in terms of Q means setting $Q(\boldsymbol{h}|\boldsymbol{x}^{(n)})$ to $p(\boldsymbol{h}|\boldsymbol{x}^{(n)}, \boldsymbol{\theta})$. The latter term requires performing inference (e.g. message passing) to compute.

Advantages/Shortcomings: It can be shown that EM never decreases the objective function, and thus will eventually converge to a local optimum. Moreover, it was recently shown that given a proper initialization, EM will converge to the maximum likelihood estimate (Balakrishnan et al., 2014). This makes EM highly statistically efficient since it aims to find the maximum likelihood solution, which is the most statistically efficient estimator (Casella and Berger, 2002).

Unfortunately since the search space typically contains many local optima, and thus often EM will get trapped in sub-optimal solutions as with any other local search method. With arbitrary initialization, there is no guarantee that EM will reach the optimal (or even a good) solution, and therefore is not consistent.

Moreover, EM can be quite slow since the E-step requires performing inference over all the training examples which can become very computationally expensive for large datasets. While there do exist methods to speed up EM such as online EM, these approaches introduce additional parameters such as the learning rate, that performance is often sensitive to.

2.2.2 Structure Learning

Structure learning of latent variable models is also considerably more challenging than the fully observed case. Recall from the previous section that when learning observed trees we can write the likelihood as a sum of mutual information and entropy (Eq. 2.13).

Unfortunately when doing structure learning with latent variable models it is difficult to compute mutual information among pairs of variables when atleast one is not observed. Thus it has largely been tackled by heuristics that do not have any consistency guarantees. Examples from the phylogenetic community include maximum parisinomius and maximum likelihood methods (Semple and Steel, 2003). In the machine learning community, Zhang (2004) proposed a search heuristic for hierarchical latent class models by defining a series of local search operations and using EM to compute the likelihood of candidate structures. Harmeling and Williams (2011) proposed a greedy algorithm to learn binary trees by joining two nodes with a high mutual information and iteratively performing EM to compute the mutual information among newly added hidden nodes. Alternatively, Bayesian hierarchical clustering (Heller and Ghahramani, 2005) is an agglomerative clustering technique that merges clusters based on a statistical hypothesis test.

There do exist a class of methods called distance-based approaches (Saitou and Nei, 1987; Semple and Steel, 2003; Choi et al., 2010; Anandkumar et al., 2011) that leverage the concept of additive tree metrics (Buneman, 1971; Erdõs et al., 1999) from the phylogenetics community to give structure learning algorithms with theoretical guarantees. We will be using these ideas in Chapters 6 and 8.

2.2.3 Inference

Once the latent variable model has been learned, then inference can proceed as described in 2.1.3. There are some interesting variants like marginal MAP (Jiang et al., 2011; Liu and Ihler, 2013) that are more challenging but they are outside the scope of this thesis.

2.2.4 Motivation for Linear Algebra Point of View

The previous subsections demonstrate that while latent variable models provide expensive modeling power, existing learning methods, in an attempt to match the statistical efficiency of learning in observed models (i.e. maximum likelihood estimation), suffer from severe computational problems and do not guarantee statistical consistency. By formulating the problem using linear algebra, we will show in this thesis that it is possible to trade-off statistical and computational complexity while preserving modeling power, thus providing a different approach to latent variable learning.

2.3 Review of Hilbert Space Embedding of Distributions

So far the focus has been on modeling discrete variables. However, in Chapter 6 we will discuss generalizing some of our proposed approaches to continuous, non-Gaussian settings where the variables do not easily fit into a parametric family. For example, demographics data is highly skewed while image data often is multimodal.

To do this, we will leverage a recent idea called Hilbert space embeddings of distributions (Smola et al., 2007; Song et al., 2009) and its applications to kernel graphical models (Song et al., 2010c,b, 2011a). These techniques provide an elegant way to do probabilistic inference for continuous, nonparametric distributions. We give a brief review of these ideas below.

2.3.1 Intuition

For intuition, consider the Gaussian distribution, which is one continuous distribution that is easily handled by conventional probabilistic modeling methods. Gaussians are easy to model for two reasons:

1. They have a compact set of sufficient statistics: the mean and variance (μ, σ^2)
2. It is easy to perform marginalization / sum-product (probabilistic inference) with gaussian distributions since marginals/conditionals of a multivariate Gaussian distribution are also Gaussian.

Hilbert space embeddings provide a means to do the same for a variable X from an arbitrary (smooth) distribution D, by constructing a sufficient statistic $\boldsymbol{\mu}_X$ for this distribution that allows for efficient probabilistic inference.

One strategy would be to take just the mean or the first order moment of the distribution:

$$\boldsymbol{\mu}_x^1 = \begin{pmatrix} \mathbb{E}[X] \end{pmatrix} \tag{2.25}$$

However, this statistic is not sufficient since there are many distributions that can have the same mean. We could also take more moments e.g.

$$\mu_x^2 = \begin{pmatrix} \mathbb{E}[X] \\ \mathbb{E}[X^2] \end{pmatrix} \qquad \mu_x^3 = \begin{pmatrix} \mathbb{E}[X] \\ \mathbb{E}[X^2] \\ \mathbb{E}[X^3] \end{pmatrix} \tag{2.26}$$

This allows us to distinguish among a wider range of distributions. For example if distribution D_1 and D_2 do not share the same mean, variance, *or* skewness then $\boldsymbol{\mu}_x^3 \neq \boldsymbol{\mu}_x^3$.

Note that we can keep adding moments, possibly even constructing an infinite dimensional statistic $\boldsymbol{\mu}_x^\infty$ that would enable us to distinguish among a large class of distributions. But how will we perform computations with this statistic! This is where kernels come in. While $\boldsymbol{\mu}_x^\infty$ may be infinite dimensional it can be constructed so that taking inner products is easy to compute (i.e. the

RKHS term	(informal) analog in vector space of p-dimensional vectors in $\mathbb{R}^p$
function	vector
operator	matrix
adjoint	transpose
reproducing property	dot product with indicator vector

Table 2.1: Summary of general notation convention used throughout the thesis

"kernel trick") [1].

Before formally defining Hilbert space embeddings, we review relevant concepts in functional analysis and reproducing kernel Hilbert spaces that are key to the exposition. Table 2.1 provides some informal notation analogs for easier understanding.

2.3.2 Hilbert Spaces

Vector space: A vector space is a set of objects $\mathcal{V}$ that are closed under linear operations i.e. $v, w \in \mathcal{V} \to \alpha v + \beta w \in \mathcal{V}$. In elementary linear algebra $\mathcal{V}$ is typically $\mathbb{R}^p$ and each element is a p-dimensional real valued vector. However, when discussing Hilbert space embeddings we will take a more general point of view and $\mathcal{V}$ may be a space of functions (which can be informally be thought of as infinite dimensional vectors).

Hilbert space: A Hilbert space is a complete vector space endowed with an inner product $\langle \cdot, \cdot \rangle$. This inner product $\langle f, g \rangle$ satisfies the following properties:

- Symmetry: $\langle f, g \rangle = \langle g, f \rangle$
- Linearity $\langle \alpha_1 f_1 + \alpha_2 f_2, g \rangle = \alpha_1 \langle f_1, g \rangle + \alpha_2 \langle f_2, g \rangle$
- Nonnegativity $\langle f, f \rangle \geq 0$
- Zero $\langle f, f \rangle = 0 \implies f = 0$

Some examples of inner products are the the vector dot product $\langle v, w \rangle = v^\top w$, or the integral $\langle f, g \rangle = \int f(x) g(x)\, dx$.

Operator: An operator $C : \mathcal{F} \to \mathcal{G}$ maps a a function $f \in \mathcal{F}$ in one Hilbert Space to a function g in another i.e. $g = Cf$. For the special case when $\mathcal{F}$ is the space of p dimensional vectors in $\mathbb{R}^p$ and $\mathcal{G}$ is the space of q dimensional vectors in $\mathbb{R}^q$, an operator would be a $p \times q$ matrix.

Adjoint: The adjoint $C^\top : \mathcal{G} \to \mathcal{F}$ of an operator $C : \mathcal{F} \to \mathcal{G}$ is defined such that $\langle g, Cf \rangle = \langle C^\top g, f \rangle \forall f \in \mathcal{F}, g \in \mathcal{G}$. This is the generalization of transpose/conjugate transpose for matrices i.e. $w^\top M v = (M^\top w)^\top v$.

Outer product: $f \otimes g$ is defined implicitly such that $f \otimes g(h) = \langle g, h \rangle f$. It is the generalization of the traditional vector outer product $v \otimes w = v w^\top$.

[1]Hilbert space embeddings actually do not use the moments to construct sufficient statistics. This is merely an example for intuition

Reproducing Kernel Hilbert Space: A reproducing kernel Hilbert space is a special type of Hilbert space that is characterized using a mercer kernel. A mercer kernel $\boldsymbol{K}(x, y)$ is a function of two variables such that

$$\int\int \boldsymbol{K}(x, y)\boldsymbol{f}(x)\boldsymbol{f}(y)\, dx\, dy > 0 \ \ \forall \boldsymbol{f} \in \mathcal{F} \tag{2.27}$$

Note that this a generalization of a positive definite matrix. The most common kernel we will use is the Gaussian RBF kernel:

$$\boldsymbol{K}(x, y) = \exp\left(\frac{||x - y||^2}{\sigma^2}\right) \tag{2.28}$$

Consider holding one element of the kernel fixed i.e. $\boldsymbol{\phi}_x = \boldsymbol{K}(x, :)$, which is a function of one variable that we call the feature function. The collection of feature functions is called the **feature map**.

For a Gaussian kernel, the feature functions are unnormalized Gaussians:

$$\boldsymbol{\phi}_1(y) = \exp\left(\frac{||1 - y||^2}{\sigma^2}\right) \qquad\qquad \boldsymbol{\phi}_{1.5}(y) = \exp\left(\frac{||1.5 - y||^2}{\sigma^2}\right) \tag{2.29}$$

and the inner product is defined as $\langle \boldsymbol{\phi}_x, \boldsymbol{\phi}_y \rangle = \langle \boldsymbol{K}(x, :), \boldsymbol{K}(y, :) \rangle := \boldsymbol{K}(x, y)$. Note that $\boldsymbol{\phi}_x(y) = \boldsymbol{\phi}_y(x) = \boldsymbol{K}(x, y)$.

Consider the set of functions that can be formed with linear combinations of these feature functions:

$$\mathcal{F}_0 = \left\{ \boldsymbol{f}(z) : \sum_{j=1} k\alpha_j \boldsymbol{\phi}_{x_j}(z), \forall k \in \mathbb{N}_+ \text{ and } \forall x_j \in \mathcal{X} \right\} \tag{2.30}$$

The RKHS $\mathcal{F}$ is defined as the completion of $\mathcal{F}_0$[2]. Intuitively one can think of the feature functions as an overcomplete basis for $\mathcal{F}$.

Reproducing Property: An RKHS has the key property that taking the inner product of a function $\boldsymbol{f}$ with $\boldsymbol{\phi}_x$ evaluates $\boldsymbol{f}$ at point x.

$$\langle \boldsymbol{f}, \boldsymbol{\phi}_x \rangle = \langle \sum_j \alpha_j \boldsymbol{\phi}_{x_j}, \boldsymbol{\phi}_x \rangle = \sum_j \alpha_j \langle \boldsymbol{\phi}_{x_j}, \boldsymbol{\phi}_x \rangle = \sum_j \alpha_j \boldsymbol{K}(x_j, x) = \boldsymbol{f}(x) \tag{2.31}$$

This is the generalization for $\boldsymbol{v}(i) = \boldsymbol{v}^\top \boldsymbol{\delta}_i$ in the vector space of p-dimensional vectors in $\mathbb{R}^p$ where $\boldsymbol{\delta}_i$ is an indicator vector that has a 1 in position i and 0 in all other positions.

[2]A metric space $\mathcal{F}$ is complete if for any cauchy sequence of elements in $\mathcal{F}$ the limit of this sequence is also in $\mathcal{F}$

2.3.3 Hilbert Space Embeddings

Given a kernel $\boldsymbol{K}$ and associated feature map $\boldsymbol{\phi}$ and RKHS $\mathcal{F}$, the Hilbert space embedding of a univariate random variable X is defined as (Smola et al., 2007):

$$\boldsymbol{\mu}_X(\cdot) = \mathbb{E}_X[\boldsymbol{\phi}_X] = \int P(x)\boldsymbol{\phi}_x(\cdot)\,dx \tag{2.32}$$

If the kernel is universal then the map from distributions to embeddings is one-to-one. Examples of universal kernels include the Gaussian RBF Kernel and the Laplace Kernel.

The empirical estimate is defined as

$$\widehat{\mu}_X = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}_{x^{(n)}} \tag{2.33}$$

However, note that since $\boldsymbol{\phi}$ is infinite dimensional, it is not possible to directly compute this estimate from data (this will be handled in 2.3.5).

Example (discrete): Let us show how this reduces to the traditional probability vector in the discrete case. Consider a random variable X that takes the values in the set $\{1, 2, 3, 4\}$. We want to embed it into an RKHS of 4 dimensional vectors in $\mathbb{R}^4$. The feature functions in this RKHS are:

$$\boldsymbol{\phi}_1 = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \qquad \boldsymbol{\phi}_2 = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \qquad \boldsymbol{\phi}_3 = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \qquad \boldsymbol{\phi}_4 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} \tag{2.34}$$

Thus,

$$\boldsymbol{\mu}_X = \mathbb{E}_X[\boldsymbol{\phi}_X] = \sum_i^4 P(X = i)\boldsymbol{\phi}_i = \begin{pmatrix}P(X=1)\\P(X=2)\\P(X=3)\\P(X=4)\end{pmatrix} \tag{2.35}$$

Evaluating the embedding: $\boldsymbol{\mu}_X$ is a function in $\mathcal{F}$ that characterizes the distribution of X. But how do we obtain probability estimates from $\boldsymbol{\mu}_X$? Consider again the discrete example above. In this case, we can obtain $P(X = i)$ by taking the inner product of $\boldsymbol{\mu}_X$ with $\boldsymbol{\phi}_i$ in Eq. 2.34 (the indicator vector) i.e.

$$P(X = i) = \boldsymbol{\mu}_X(i) = \langle \boldsymbol{\mu}_X, \boldsymbol{\phi}_i \rangle \tag{2.36}$$

For the general case, we follow the same strategy leveraging the reproducing property:

$$\boldsymbol{\mu}_X(i) = \langle \boldsymbol{\mu}_X, \boldsymbol{\phi}_i \rangle = \mathbb{E}[\langle \boldsymbol{\phi}_X, \boldsymbol{\phi}_i \rangle] = \mathbb{E}[\boldsymbol{K}(X, i)] \tag{2.37}$$

For a gaussian RBF kernel, this is equal to:

$$\mathbb{E}\left[\exp\left(\frac{\|i - X\|^2}{\sigma^2}\right)\right] \approx \frac{1}{N}\sum_{n=1}^{N} \exp\left(\frac{\|i - x^{(n)}\|^2}{\sigma^2}\right) \tag{2.38}$$

Note this is exactly equal to the (unnormalized) kernel density estimate under a gaussian RBF kernel! Thus, in one dimension the Hilbert space embedding of a distribution is equivalent to kernel density estimation. Now let us consider embedding multiple variables.

Cross-covariance operator: $\mathcal{C}_{YX} = \mathbb{E}_{YX}[\boldsymbol{\psi}_Y \otimes \boldsymbol{\phi}_X]$ for $\boldsymbol{\phi} \in \mathcal{F}, \boldsymbol{\psi} \in \mathcal{G}$. This is the embedding of the joint distribution among two variables. Just like with the one dimensional case, we can evaluate this operator at (i, j) using $\boldsymbol{\psi}_i, \boldsymbol{\phi}_j$ to give the the unormalized (2-dimensional) kernel density estimate of $P(Y = i, X = j)$:

$$\begin{aligned}\langle \boldsymbol{\psi}_i, \mathcal{C}_{YX}\boldsymbol{\phi}_j \rangle &= \mathbb{E}_{YX}[\boldsymbol{K}_{\mathcal{G}}(Y, i)\boldsymbol{K}_{\mathcal{F}}(X, j)] \\ &\approx \frac{1}{N}\sum_{n=1}^{N} \exp\left(\frac{\|i - y^{(n)}\|^2}{\sigma^2}\right)\exp\left(\frac{\|j - x^{(n)}\|^2}{\sigma^2}\right) \quad \text{(empirical estimate under gaussian RBF kernels)}\end{aligned} \tag{2.39}$$

Note that we could represent p variables using a huge cross-covariance operator $\mathcal{C}_{1...p} = \mathbb{E}_{X_1...X_p}[\boldsymbol{\phi}_{X_1} \otimes ... \otimes \boldsymbol{\phi}_{X_p}]$. This would be analogous to p-dimensional kernel density estimation. Consequently, this would be pointless since due to the curse of dimensionality, kernel density estimation works poorly when p is even moderately large.

However, the key advantage of Hilbert space embeddings over kernel density estimation in high dimensions is that we will be able to "factorize" the joint cross-covariance operator into smaller operators analogous to how a large discrete probability table can be factorized into smaller factors based on conditional independence assumptions. To do this, we need to define the conditional embedding operator.

Conditional embedding operator (Song et al., 2009): The conditional embedding operator is defined as

$$\mathcal{C}_{X|Y} = \mathcal{C}_{XY}\mathcal{C}_{YY}^{-1} \tag{2.40}$$

and has the following property:

$$\mathbb{E}_{X|y}[\boldsymbol{\phi}_X | y] = \mathcal{C}_{X|Y}\boldsymbol{\phi}_y \tag{2.41}$$

This is analogous to "slicing" the conditional probability table in the discrete case i.e. $\boldsymbol{\mathcal{P}}_{X|Y=1} = \boldsymbol{\mathcal{P}}_{X|Y}\boldsymbol{\delta}_1$ where $\boldsymbol{\mathcal{P}}_{X|Y}$ is the conditional probability matrix of $X|Y$ and δ_i is the indicator vector. Furthermore, using this property it is possible to derive two key operations that are essential to probabilistic inference:

Lemma 3. *The sum rule:* $\mathcal{C}_{YX} = \mathcal{C}_{Y|X}\boldsymbol{\mu}_X$

Proof. We first show the proof for the discrete case in terms of expectations since it directly generalizes to the kernel scenario. Let $\boldsymbol{\mathcal{P}}_{X|Y}$ be the conditional probability matrix of $X|Y$ i.e. $\boldsymbol{\mathcal{P}}_{X|Y}(i,j) = P(X=i|Y=j)$ and $\boldsymbol{\mathcal{P}}_Y$ be the marginal probability vector of Y i.e. $\boldsymbol{\mathcal{P}}_Y(i) = \boldsymbol{\mathcal{P}}(Y=i)$. Then,

$$\begin{aligned}
\boldsymbol{\mathcal{P}}_{X|Y}\boldsymbol{\mathcal{P}}_Y &= \boldsymbol{\mathcal{P}}_{X|Y}\mathbb{E}_Y[\boldsymbol{\delta}_Y] \\
&= \mathbb{E}_Y[\boldsymbol{\mathcal{P}}_{X|Y}\boldsymbol{\delta}_Y] \\
&= \mathbb{E}_Y[\mathbb{E}_{X|Y}[\boldsymbol{\delta}_X]] \\
&= \mathbb{E}_{XY}[\boldsymbol{\delta}_X] \\
&= \boldsymbol{\mathcal{P}}_X
\end{aligned} \tag{2.42}$$

Now the kernel case follows analogously:

$$\begin{aligned}
\boldsymbol{C}_{X|Y}\mu_Y &= \boldsymbol{C}_{X|Y}\mathbb{E}_Y[\phi_Y] \quad \text{(definition of } \mu_Y\text{)} \\
&= \mathbb{E}_Y[\boldsymbol{C}_{X|Y}\boldsymbol{\phi}_Y] \\
&= \mathbb{E}_Y[\mathbb{E}_{X|Y}[\boldsymbol{\phi}_X]] \quad \text{(property of conditional embedding operator)} \\
&= \mathbb{E}_{XY}[\boldsymbol{\phi}_X] \\
&= \boldsymbol{\mu}_X
\end{aligned} \tag{2.43}$$

□

Lemma 4. ***Product rule:*** $\boldsymbol{C}_{YX} = \boldsymbol{C}_{Y|X}\boldsymbol{C}_{XX}$. *(The proof follows similarly to the one for the sum rule so is not shown)*

2.3.4 Kernel Graphical Models

The tools from the previous section allow us to factorize arbitrary multivariate continuous distributions in an analogous manner to discrete probability tables, thus generalizing graphical models to the continuous case as done by (Song et al., 2010c,b, 2011a).

Consider the graphical model in Figure 2.2(b) with 4 variables, A, B, C, D. We represent the conditional probability distributions as follows:

$\boldsymbol{C}_{AA}$	marginal distribution of A
$\boldsymbol{C}_{B\|A}$	conditional distribution of $B\|A$
$\boldsymbol{C}_{C\|B}$	conditional distribution of $C\|B$
$\boldsymbol{C}_{D\|C}$	conditional distribution of $D\|C$

Now by applying the sum and product rules we obtain:

$$\boldsymbol{C}_{A,D} = \boldsymbol{C}_{AA}\boldsymbol{C}_{B|A}^{\top}\boldsymbol{C}_{C|B}^{\top}\boldsymbol{C}_{D|C}^{\top} \tag{2.44}$$

(Song et al., 2010c, 2011a) generalized this strategy to derive kernel message passing algorithms to perform nonparametric inference in tree and non-tree models.

2.3.5 Empirical computation of Hilbert space embeddings

Note that empirical estimates require

$$\widehat{\mu}_X = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}_{x^{(n)}}$$
$$\widehat{\boldsymbol{C}}_{YX} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\phi}_{y^{(n)}} \otimes \boldsymbol{\phi}_{x^{(n)}} \tag{2.45}$$

These empirical estimates are difficult to directly compute since $\boldsymbol{\phi}$ is not required to be a finite dimensional vector. However, note that because of the kernel trick it is possible to evaluate these operators at specific points:

$$\widehat{\mu}_X(\bar{x}) = \langle \widehat{\boldsymbol{\mu}}_X, \boldsymbol{\phi}_{\bar{x}} \rangle = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{K}(x^{(n)}, \bar{x})$$
$$\widehat{\boldsymbol{C}}_{YX}(\bar{y}, \bar{x}) = \langle \boldsymbol{\phi}_{\bar{y}}, \widehat{\boldsymbol{C}}_{YX}\boldsymbol{\phi}_{\bar{x}} \rangle = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{K}(y^{(n)}, \bar{y})\boldsymbol{K}(x^{(n)}, \bar{x}) \tag{2.46}$$

However, we will find this unwieldy and slow in practice because performing the sum/chain rule will require manipulation using the kernel matrix $\boldsymbol{K}$ which is $N \times N$. Below we will show it is possible to build a low rank approximation of the kernel matrix and build approximate features $\widehat{\boldsymbol{\phi}}$ that are much smaller than $N \times 1$.

Approximate feature maps using incomplete cholesky decomposition[3]: It is more common (and more computationally efficient) to construct an finite dimensional feature map via incomplete cholesky decomposition (Fine and Scheinberg, 2002; Bach and Jordan, 2003). Note that using QR decomposition,

$$\boldsymbol{K} = \boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \boldsymbol{R}^\top \boldsymbol{Q}^\top \boldsymbol{Q}\boldsymbol{R} = \boldsymbol{R}^\top \boldsymbol{R} \tag{2.47}$$

where $\boldsymbol{\Phi} := \{\boldsymbol{\phi}_{x^{(1)}}, ..., \boldsymbol{\phi}_{x^{(N)}}\}$ is the feature map, $\boldsymbol{Q} = \{\boldsymbol{q}_1, ..., \boldsymbol{q}_n\}$ is an orthogonal matrix and $\boldsymbol{R} = \{\boldsymbol{r}_1, ..., \boldsymbol{r}_n\}$ is a matrix of coordinates in the new basis. Note that each $\boldsymbol{q}_i$ may be infinite-dimensional, but each $\boldsymbol{r}_i$ is $N \times 1$.

Interestingly, $\boldsymbol{R}$ presents an alternate feature map (just transformed to be in the basis of $\boldsymbol{Q}$). In particular we can set $\widehat{\boldsymbol{\phi}}_{x^{(i)}} := \boldsymbol{r}_i$. While the original $\boldsymbol{\phi}_{x^{(i)}}$ was infinite dimensional $\boldsymbol{r}_i$ must be finite because the kernel matrix $\boldsymbol{K}$ is $N \times N$ and therefore has at most rank N.

Thus, our goal will be to compute $\boldsymbol{R}$. A first strategy would just be to use standard QR

[3]We follow the exposition in http://www.gatsby.ucl.ac.uk/~gretton/coursefiles/incompleteCholesky.pdf

decomposition. In this procedure, $\boldsymbol{q}_1 := \frac{\boldsymbol{\phi}_{x^{(1)}}}{||\boldsymbol{\phi}_{x^{(1)}}||}$ and then each subsequent $\boldsymbol{q}_i$ is updated as follows:

$$\tilde{\boldsymbol{q}}_j = \boldsymbol{\phi}_{x^{(j)}} - \sum_{t=1}^{j-1} \langle \boldsymbol{q}_t, \boldsymbol{\phi}_{x^{(t)}} \rangle \boldsymbol{q}_t \tag{2.48}$$

followed by setting $\boldsymbol{q}_j = \frac{\tilde{\boldsymbol{q}}_j}{||\tilde{\boldsymbol{q}}_j||_2}$. This can be rewritten as:

$$\boldsymbol{q}_j = \frac{1}{\nu_j} \left(\boldsymbol{\phi}_{x^{(j)}} - \sum_{t=1}^{j-1} \langle \boldsymbol{q}_t, \boldsymbol{\phi}_{x^{(t)}} \rangle \boldsymbol{q}_t \right) \tag{2.49}$$

where ν_j is the normalization constant. $\boldsymbol{R}$ can then be recovered by noting that $\boldsymbol{R}(j,i) = \langle \boldsymbol{q}_j, \boldsymbol{\phi}_{x^{(i)}} \rangle$. However, this procedure is not feasible to run since the feature function $\boldsymbol{\phi}_{x^{(j)}}$ (and subsequently $\boldsymbol{q}$) can be infinite dimensional.

Below we show one can recover $\boldsymbol{R}$ directly without $\boldsymbol{Q}$! By definition of $\boldsymbol{R}(j,i)$ we can recover it by taking the inner product with $\boldsymbol{\phi}_{x^{(i)}}$ on both sides:

$$\begin{aligned} \boldsymbol{R}(j,i) = \langle \boldsymbol{q}_j, \boldsymbol{\phi}_{x_i} \rangle &= \frac{1}{\nu_j} \left(\langle \boldsymbol{\phi}_{x_j}, \boldsymbol{\phi}_{x_i} \rangle - \sum_{t=1}^{j-1} \langle \boldsymbol{q}_t, \boldsymbol{\phi}_{x_t} \rangle \langle \boldsymbol{q}_t, \boldsymbol{\phi}_{x_i} \rangle \right) \\ \boldsymbol{R}(j,i) &= \frac{1}{\nu_j} \left(\boldsymbol{K}(j,i) - \sum_{t=1}^{j-1} \boldsymbol{R}(t,j) \boldsymbol{R}(t,i) \right), i = j+1, ..., N \end{aligned} \tag{2.50}$$

Since $t < j$, $\boldsymbol{R}(t,j)$ and $\boldsymbol{R}(t,i)$ have already been computed. The only question is how to compute ν_j. To do this recall that $d_i := \boldsymbol{K}(i,i) = \langle \boldsymbol{\phi}_{x^{(i)}}, \boldsymbol{\phi}_{x^{(i)}} \rangle = ||\boldsymbol{\phi}_{x^{(i)}}||_2^2$. If we interpret d_i to be the residual norm squared, then at each step we update d_i by subtracting $R(j,i) = \langle \boldsymbol{q}_j, \boldsymbol{\phi}_{x_i} \rangle$ i.e.:

$$d_i \leftarrow d_i - \boldsymbol{R}(j,i)^2 \tag{2.51}$$

Then we set $\nu_j = \sqrt{d_j}$.

Note that when $||\boldsymbol{d}||_2 \leq \epsilon$, then all the data points are approximately represented by the existing basis, suggesting that this criterion presents a principled way to find an approximate solution that can result in considerable computational gains. In particular it can reduce the dimension $\widehat{\boldsymbol{\phi}}$ to be considerably smaller than N. This strategy is called the incomplete cholesky decomposition and is shown in Algorithm 2. Using this approximate feature map, we can directly compute the embedding operators:

$$\begin{aligned} \widehat{\boldsymbol{\mu}}_X &= \frac{1}{N} \sum_{n=1}^{N} \widehat{\boldsymbol{\phi}}_{x^{(n)}} \\ \widehat{\boldsymbol{C}}_{YX} &= \frac{1}{N} \sum_{n=1}^{N} \widehat{\boldsymbol{\phi}}_{y^{(n)}} \otimes \widehat{\boldsymbol{\phi}}_{x^{(n)}} \end{aligned} \tag{2.52}$$

Note that this is much more efficient for inference since $\boldsymbol{\mu}_X$ will have length smaller much for N if the value of ϵ in Algorithm 2 is chosen wisely.

Algorithm 2 Calculate approximate feature map $\widehat{\mathbf{\Phi}}$ via incomplete cholesky decomposition of $\boldsymbol{K}$

```
In: Kernel matrix $\boldsymbol{K}$ (assumed to be $N \times N$), tolerance $\epsilon$
Out: approximate feature map $\widehat{\mathbf{\Phi}} := \{\widehat{\boldsymbol{\phi}}_{x^{(1)}}, ....\widehat{\boldsymbol{\phi}}_{x^{(N)}}\}$
  $\boldsymbol{d} \leftarrow \mathrm{diag}(\boldsymbol{K})$
  $j \leftarrow 1$
  while $||\boldsymbol{d}||_2 \leq \epsilon$ do
    $\nu_j = \sqrt{\boldsymbol{d}(j)}$
    for $i = j + 1; i \leq N; i + +$ do
      $\boldsymbol{R}(j, i) = \frac{1}{\nu_j}\left(\boldsymbol{K}(j, i) - \sum_{t=1}^{j-1} \boldsymbol{R}(t, j)\boldsymbol{R}(t, i)\right)$
      $\boldsymbol{d}_i \leftarrow \boldsymbol{d}_i - \boldsymbol{R}(j, i)^2$
    end for
    $j \leftarrow j + 1$
  end while
  $\widehat{\mathbf{\Phi}} \leftarrow \boldsymbol{R}$
```

Chapter 3

A Linear Algebra View of Latent Variable Models

Before, delving into the rest of the thesis, we describe latent variable models from the point of view of linear algebra, a formulation that is key to the rest of the thesis.

3.1 Tensor Notation

We first give an introduction to the tensor notation tailored to this thesis.

An Nth order tensor is a multiway array with N "modes", *i.e.*, N indices $\{i_1, i_2, \dots, i_N\}$ are needed to access its entries. Subarrays of a tensor are formed when a subset of the indices is fixed, and we use a colon to denote all elements of a mode. For instance, $\mathcal{A}(i_1, \dots, i_{n-1}, :, i_{n+1}, \dots, i_N)$ are all elements in the nth mode of a tensor $\mathcal{A}$ with indices from the other $N-1$ modes fixed to $\{i_1, \dots, i_{n-1}, i_{n+1}, \dots, i_N\}$ respectively. Furthermore, we also use the shorthand $\boldsymbol{i}_{p:q} = \{i_p, i_{p+1}, \dots, i_{q-1}, i_q\}$ for consecutive indices, *e.g.*, $\mathcal{A}(i_1, \dots, i_{n-1}, :, i_{n+1}, \dots, i_N) = \mathcal{A}(\boldsymbol{i}_{1:n-1}, :, \boldsymbol{i}_{n+1:N})$.

Furthermore, let $\boldsymbol{\mathcal{P}}(X)$ denote probability vectors/matrices/tensors. For example $\boldsymbol{\mathcal{P}}(X)$ is the marginal probability vector of X i.e. $\boldsymbol{\mathcal{P}}(X)_i = P(X = i)$. Similarly $\boldsymbol{\mathcal{P}}(X, Y|Z)$ encodes the conditional probability tensor of X, Y given Z i.e. $\boldsymbol{\mathcal{P}}(X, Y|Z)_{i,j,k} = P(X = i, Y = j|Z = k)$.

Mode-specific diagonal matrices/tensors. We use $\boldsymbol{\delta}$ to denote an N-way relation: its entry $\boldsymbol{\delta}(\boldsymbol{i}_{1:N})$ at position $\boldsymbol{i}_{1:N}$ equals 1 when all indexes are the same ($i_1 = i_2 = \ldots = i_N$), and 0 otherwise. We will use $\oslash_d$ to denote repetition of an index d times. For instance, we use $\mathbb{P}(\oslash_d X)$ to denote a dth order tensor where its entries at $(\boldsymbol{i}_{1:d})$th position are specified by $\boldsymbol{\delta}(\boldsymbol{i}_{1:d})\mathbb{P}(X = x_{i_1})$. A diagonal matrix with its diagonal equal to $\mathbb{P}(X)$ is then denoted as $\mathbb{P}(\oslash_2 X)$. Similarly, we can define a $(d + d')$th order tensor $\mathbb{P}(\oslash_d X| \oslash_{d'} Y)$ where its $(\boldsymbol{i}_{1:d}\boldsymbol{j}_{1:d'})$th entry corresponds to $\boldsymbol{\delta}(\boldsymbol{i}_{1:d})\boldsymbol{\delta}(\boldsymbol{j}_{1:d'})\mathbb{P}(X = x_{i_1}|Y = y_{j_1})$. By default $\oslash$ is equivalent to $\oslash_2$.

Matrix Sum Rule: Note that $\boldsymbol{\mathcal{P}}(Y) = \boldsymbol{\mathcal{P}}(Y|X)\boldsymbol{\mathcal{P}}(X)$ since the matrix multiplication marginalizes out X. This is the matrix formulation of $P(Y) = \sum_X P(Y|X)P(X)$.

Matrix Chain Rule: If we put X on the diagonal i.e. $\boldsymbol{\mathcal{P}}(\oslash_2 X)$ then X will not be marginalized out. This gives us $\boldsymbol{\mathcal{P}}(Y, X) = \boldsymbol{\mathcal{P}}(Y|X)\boldsymbol{\mathcal{P}}(\oslash_2 X)$, which is the matrix formulation of $P(Y, X) = P(Y|X)P(X)$.

Matricization: Sometimes we will find it more convenient to rearrange a tensor into a matrix by placing a set of modes on the rows and the rest on the columns. For example, consider the probability tensor $\boldsymbol{\mathcal{P}}(X_1, X_2, X_3, X_4)$ where each of the X_i take on n states. Then one matricization may be the $n^2 \times n^2$ matrix $\boldsymbol{\mathcal{P}}(\{X_1, X_2\}, \{X_3, X_4\})$ where X_1, X_2 are on the rows and X_3, X_4 are on the columns. Other possible matricizations include $\boldsymbol{\mathcal{P}}(\{X_1, X_3\}, \{X_2, X_4\})$ and $\boldsymbol{\mathcal{P}}(\{X_1\}, \{X_2, X_3, X_4\})$.

Labeled Tensors: In contrast to the conventional tensor notation such as the one described in Kolda and Bader (2009a), the ordering of the modes of a tensors will generally not be essential in this thesis. We define labeled tensors to be tensors such that the modes of the tensor are labeled with random variables. Each mode will correspond to a random variable and what is important is to keep track of this correspondence. Therefore, two labeled tensors are equivalent if they have the same set of labels and they can be obtained from each other by a permutation of the modes for which the labels are aligned.

In the matrix case this translates to $\boldsymbol{A}$ and $\boldsymbol{A}^\top$ being equivalent in the sense that $\boldsymbol{A}^\top$ carries the same information as $\boldsymbol{A}$, as long as we remember that the rows of $\boldsymbol{A}^\top$ are the columns of $\boldsymbol{A}$ and vice versa. We will use the following notation to denote this equivalence

$$\boldsymbol{A} \cong \boldsymbol{A}^\top \tag{3.1}$$

Under this notation, the dimension (or the size) of a mode labeled by variable X will be the same as the number of possible values for variable X. Furthermore, when we multiply two labeled tensors together, we will always carry out the operation along (a set of) modes with matching labels.

Tensor multiplication for labeled tensors. Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ be an Nth order tensor and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_M}$ be an Mth order tensor. If X is a common mode label for both $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ (w.l.o.g. we assume that this is the first mode, implying also that $I_1 = J_1$), multiplying along this mode will give

$$\boldsymbol{C} = \boldsymbol{\mathcal{A}} \times_X \boldsymbol{\mathcal{B}} \;\in\; \mathbb{R}^{I_2 \times \cdots \times I_N \times J_2 \times \cdots \times J_M}, \tag{3.2}$$

where the entries of $\boldsymbol{C}$ is defined as

$$\boldsymbol{C}(\boldsymbol{i}_{2:N}, \boldsymbol{j}_{2:M}) = \sum_{i=1}^{I_1} \boldsymbol{\mathcal{A}}(i, \boldsymbol{i}_{2:N})\, \boldsymbol{\mathcal{B}}(i, \boldsymbol{j}_{2:M})$$

Further notation/operations for labeled tensors such as multi-mode multiplication and tensor inversion will be introduced in Chapter 5. Table 3.1 gives provides a quick reference for the notation conventions.

Symbol	*Definition*
$P(\cdot)$	probability
$p(\cdot)$	probability mass/density function
$\mathcal{P}(X_1, ..., X_n)$	probability vector/matrix/tensor
$\mathcal{P}(\{X_1, X_2, X_4\}, \{X_3, X_5\})$	matricization of probability tensor
$\oslash_i$	diagonal on i modes

Table 3.1: Summary of general notation convention used throughout the thesis

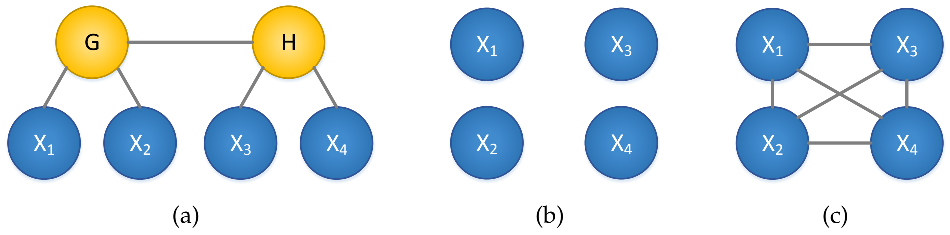


Figure 3.1: Different ways of modeling 4 observed variables X_1, X_2, X_3, X_4. G and H are latent variables.

3.2 The Spectral View

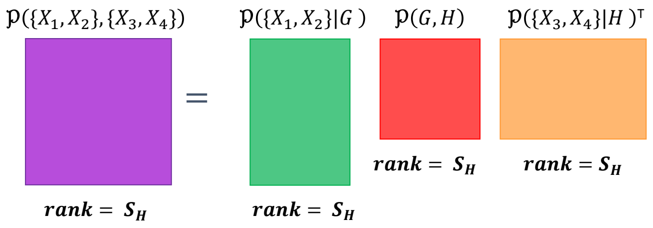
Consider Figure 3.1(a) where there are four observed variables X_1, X_2, X_3, and X_4 (indicated in blue) and two latent variables G and H (in yellow). Let S_O be the number of observed states (i.e. the number of states each of X_1, X_2, X_3, X_4 take on) and S_H be the number of latent states (i.e. the number of states that each of G, H can take on).

Consider the problem of structure learning: recovering the structural relationship among these variables given only samples of X_1, X_2, X_3, X_4. One strategy may be to greedily merge the observed variables (Harmeling and Williams, 2011). While this may work sometimes, it will not lead to a consistent solution in general. Similarly, once the structure is known, learning parameters (i.e. the conditional probability tables) is commonly done with the Expectation Maximization (EM) algorithm. EM is essentially coordinate descent on a nonconvex objective and is therefore not guaranteed to return the optimal answer.

However, are these problems really intractable, or are they simply difficult from the likelihood optimization from the point of view? For intuition, let us first examine parameter learning. Consider setting $S_G = S_H = 1$. In this case, X_1, X_2, X_3, and X_4 are independent as shown in Figure 3.1(b), since no information can travel through the latent variables. The learning problem becomes trivial in this scenario.

On the other side of the spectrum, let $S_H = S_O^4$. In this case the problem becomes equivalent to learning the clique model in Figure 3.1(c) which has $O(S_O^4)$ parameters. In general, if we had p observed variables all connected to one latent variable, then setting $S_H = S_O^p$ would be equivalent to p-way clique. This leads to an exponential increase in parameters and defeats the point of a graphical model since there are no conditional independence statements.

However, what about $1 < S_H < S_O^p$. From the optimization point of view these values of S_H lead



$\mathcal{P}(\{X_1,X_2\},\{X_3,X_4\})$
$=$
$\mathcal{P}(\{X_1,X_2\}|G)$
$\mathcal{P}(G,H)$
$\mathcal{P}(\{X_3,X_4\}|H)^\top$
$rank = S_H$
$rank = S_H$
$rank = S_H$
$rank = S_H$

Figure 3.2: Depiction of one low rank relationship encoded in Figure 3.1(a).

to nonconvex objectives and therefore are difficult. However, how much harder is $S_H = S_O^p - 1$ than $S_H = 2$? These are the types of questions that likelihood optimization cannot quantify.

Now, let us approach the problem from a linear algebra point of view. First, consider two variables X_1 and X_2. In Figure 3.1(c) there is in general nothing we can say is special about the matrix $\mathcal{P}(X_1, X_2)$. However, in the case of Figure 3.1(b), $\mathcal{P}(X_1, X_2) = \mathcal{P}(X_1)\mathcal{P}(X_2)^\top$. Therefore $\mathcal{P}(X_1, X_2)$ is rank one.

More generally, for Figure 3.1(a), $\mathcal{P}(X_1, X_2) = \mathcal{P}(X_1|G)\mathcal{P}(\oslash G)\mathcal{P}(X_2|G)^\top$. Assuming all the matrices on the right hand side are full rank, this implies that $\mathcal{P}(X_1, X_2)$ has rank S_H, implying that our factorization is a *low rank* factorization.

Lets now extend this logic to all 4 variables in Figure 3.1(a). Let $\mathcal{P}(\{X_1, X_2\}, \{X_3, X_4\})$ be the $S_O^2 \times S_O^2$ joint probability matrix of X_1, X_2, X_3, X_4 where the values of X_1, X_2 are on the rows and X_3, X_4 is on the columns. This matrix has the following low rank factorization:

$$\mathcal{P}(\{X_1, X_2\}, \{X_3, X_4\}) = \mathcal{P}(\{X_1, X_2\}|G)\mathcal{P}(G, H)\mathcal{P}(\{X_3, X_4\}|H)^\top$$

A graphical depiction is shown in Figure 3.2. Moreover, different matricizations lead to different factorizations i.e.

$$\mathcal{P}(\{X_1\}, \{X_2, X_3, X_4\}) = \mathcal{P}(X_1|G)\mathcal{P}(\oslash G)\mathcal{P}(\{X_2, X_3, X_4\}|G)^\top$$
$$\mathcal{P}(\{X_1, X_2, X_3\}, \{X_4\}) = \mathcal{P}(\{X_1, X_2, X_3|H)\mathcal{P}(\oslash H)\mathcal{P}(X_4|H)^\top$$

However, note all these low rank factorizations still have rank S_H. Thus, while connections among observed variables in a graphical model specify "hard" conditional independences/dependencies, latent variables induce low rank dependencies among observed variables. Small S_H translates into small rank, implying simple dependencies among observed variables. As S_H gets larger, the rank of the model increases and the dependencies become more complicated. As we will see later, the cardinality of S_H plays a key role in determining the difficulty of learning with latent variables.

While this linear algebra point of view may seem to be just a simple reformulation, it leads to an interesting perspective that inspires solutions to a variety of problems that we will explore in later chapters:

- **Parameter Learning**: The particular low rank factorization we showed above is special in that it is composed of conditional probability matrices. However, low rank factorizations are in general not unique. For any matrix factorization $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{R}$, $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{S}^{-1}\boldsymbol{S}\boldsymbol{R}$ is also a low rank factorization where $\boldsymbol{S}$ can be any invertible matrix. Thus, a natural question to ask is do there exist other factorizations that only depend on observed variables and therefore do not require EM to learn?

- **Structure Learning**: The fact that $\mathcal{P}(\{X_1, X_2\}, \{X_3, X_4\})$ is low rank but $\mathcal{P}(\{X_1, X_3\}, \{X_2, X_4\})$ is not reveals aspects about the latent structure of the model. How can this intuition be used to learn the latent structure underlying a set of continuous, non-Gaussian variables? In particular, we will leverage the notion of *additive tree metrics* (Saitou and Nei, 1987; Lake, 1994; Choi et al., 2011).
- **Modeling**: Can this connection between linear algebra and probabilistic modeling motivate new models and solutions for language modeling and unsupervised parsing? For example, in language modeling can we create n-grams for non-integer values of n i.e. 1.5-grams, 2.5-grams etc.?

Part I

Spectral Learning Algorithms for Graphical Models

Chapter 4

A Spectral Algorithm for Latent Tree Graphical Models

Using the insights from the previous chapter, we first tackle parameter learning in latent variable models. As mentioned earlier, conventional approaches have primarily relied on likelihood maximization and local search heuristics such as expectation maximization (EM) (Dempster et al., 1977). In addition to the problem of local optima, EM can require many iterations to reach a prescribed training precision, and high dimensional problems can dramatically slow down EM.

While EM tries to recover the full set of parameters in latent variable models, in many applications it is the inference task that is most interesting. For instance, in speech classification, we are interested in estimating the likelihood of a test sequence under different models; in quantitative finance, we are interested in predicting the price of one stock given the prices of other stocks; or in biological analysis, we are interested in forecasting the expression of one gene given perturbations to other genes. In all these examples, the inference task involves estimating either the joint or conditional distribution of a set of observed variables. Ideally, we want to avoid explicitly recovering the parameters related to latent variables (which leads to non-convex problems), and proceed directly to the quantities of interest.

Recently, Hsu et al. (2009) and Bailly et al. (2009) proposed spectral algorithms for learning hidden Markov models (HMM) which directly estimates the joint distribution of the observed variables without recovering the HMM model parameters. The major computation of the algorithms involves a singular value decomposition (SVD) of small marginal probability matrices involving pairs of observed variables. Compared to EM, this spectral algorithm does not have the problem of local optima, and one can formally study its statistical properties. However, this spectral algorithm is specific to HMMs, and it is not clear whether their techniques can be extend to latent variable models with other topologies. Mossel and Roch (2006) also proposed a spectral algorithm for latent variable models which applies to arbitrary tree topologies, but they made very restrictive assumptions: all variables (observed and latent) have exactly the same number of states, and all conditional probability tables (CPT) are invertible. Under these conditions, they derived a spectral algorithm that can explicitly recover all CPTs from marginals of triples of observed variables. In many applications, however, latent variables can represent factors simpler than the noisy observations, and the number of hidden states can be smaller than that of the observed states. In these cases, the CPTs are no longer invertible, which renders this spectral algorithm no longer applicable. Moreover, as we show in Chapter 7 even when in settings when this method is applicable it performs poorly in practice.

Contribution of this chapter: In this chapter, we propose a novel spectral algorithm for latent variable models with arbitrary tree topologies that allows the number of hidden states to be smaller or larger than the number of observed states. Instead of learning the original conditional probability tables (like Mossel and Roch (2006) do), we learn a linear transformation of these parameters that still enables efficient inference. This enables a provably consistent and local-optima-free learning algorithm with sample complexity analysis. Empirical results indicate that our spectral approach performs comparably or better than EM, while being 1-2 orders of magnitude faster.

Key to approach is our extensive use of tensor algebra that enables generalization of spectral learning to models beyond HMMs. Later works such as Cohen et al. (2012); Cohen and Collins (2012); Dhillon et al. (2012a) have leveraged our tensor formulation for supervised parsing with latent variables. Our work is also closely related with tensor decomposition (Kolda and Bader, 2009b), that we discuss at the end of the chapter.

Outline: Some intuition is presented using a small example. We then establish a tensor message passing scheme for latent tree graphical models. Subsequently the spectral representation is derived and different aspects are discussed including sample complexity analysis and practical considerations. Finally, empirical results are presented.

Prerequisites: This chapter assumes a general understanding of latent variable models as presented in 2.2, the connection between latent variable models and low rank factorization in Chapter 3, and the tensor notation in 3.1.

4.1 Intuition

Recall that in Chapter 3 we had established that latent variables introduce low rank factorizations, i.e. $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{R}$, over the marginal probability of the observed variables. For the example in Figure 3.1 we can set,

$$\begin{aligned}\boldsymbol{M} &:= \boldsymbol{\mathcal{P}}(\{X_1, X_2\}, \{X_3, X_4\}) \\ \boldsymbol{L} &:= \boldsymbol{\mathcal{P}}(\{X_1, X_2\}|H)\boldsymbol{\mathcal{P}}(\oslash_2 H) = \boldsymbol{\mathcal{P}}(\{X_1, X_2\}, H) \\ \boldsymbol{R} &:= \boldsymbol{\mathcal{P}}(\{X_3, X_4\}|H)\end{aligned} \tag{4.1}$$

However, low rank factorizations are in general not unique. For any matrix factorization, $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{R}$, we also have that

$$\boldsymbol{M} = \underbrace{\boldsymbol{L}\boldsymbol{S}}_{\widetilde{\boldsymbol{L}}}\ \underbrace{\boldsymbol{S}^{-1}\boldsymbol{R}}_{\widetilde{\boldsymbol{R}}} \tag{4.2}$$

and thus an *alternate* factorization $\boldsymbol{M} = \widetilde{\boldsymbol{L}}\widetilde{\boldsymbol{R}}$. However, note that while $\boldsymbol{L}$ and $\boldsymbol{R}$ may be probability tables, while $\widetilde{\boldsymbol{L}}$ and $\widetilde{\boldsymbol{R}}$ can may have negative values (since even if $\boldsymbol{S}$ is non-negative, $\boldsymbol{S}^{-1}$ may have negative entries).

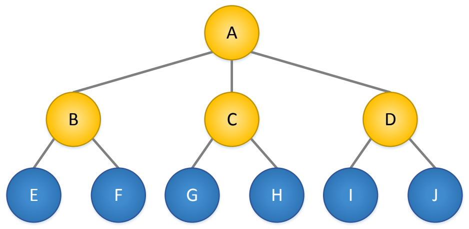


Figure 4.1: Example of a latent tree model with six observed nodes

The natural question to ask is that does there exist a low rank factorization that is only a function of the observed variables X_1, X_2, X_3, X_4? Interestingly, the answer is *yes*!

To see why, consider the following expansions:

$$\begin{aligned} \mathcal{P}(\{X_1, X_2\}, X_3) &= \underline{\mathcal{P}(\{X_1, X_2\}|H)\mathcal{P}(\oslash_2 H)}\mathcal{P}(X_3|H)^\top \\ \mathcal{P}(X_2, \{X_3, X_4\}) &= \mathcal{P}(X_2|H)\mathcal{P}(\oslash_2 H)\underline{\mathcal{P}(\{X_3, X_4\}|H)^\top} \end{aligned} \tag{4.3}$$

The product of the green terms (underlined), in some order, is $\mathcal{P}(\{X_1, X_2\}, \{X_3, X_4\})$ (what we want!). The product of the red terms (not underlined), in some order, is $\mathcal{P}(X_2, X_3)$ (what we need to eliminate). This leads us the following alternate factorization that only depends on observed variables:

$$\mathcal{P}(\{X_1, X_2\}, \{X_3, X_4\}) = \mathcal{P}(\{X_1, X_2\}, X_3)\mathcal{P}(X_2, X_3)^{-1}\mathcal{P}(X_2, \{X_3, X_4\}) \tag{4.4}$$

Note that while the original (CPT) factorization was a product of conditional probability matrices, the alternate factorization is a product of marginal probability matrices and their inverses.

How does this relate to our original factorization? Consider setting $\boldsymbol{S} = \mathcal{P}(X_3|H)$. Then we can prove that

$$\begin{aligned} \boldsymbol{L}\boldsymbol{S} &= \mathcal{P}(\{X_1, X_2\}, X_3) \\ \boldsymbol{S}^{-1}\boldsymbol{R} &= \mathcal{P}(X_2, X_3)^{-1}\mathcal{P}(X_2, \{X_3, X_4\}) \end{aligned} \tag{4.5}$$

4.2 Notation for Latent Tree Graphical Models

We now generalize this intuition to arbitrary latent tree graphical models. A latent tree model defines a joint probability distribution over a set of O observed variables $\mathscr{O} = \{X_1, \ldots, X_O\}$ and a set of H hidden variables $\mathscr{H} = \{X_{O+1}, \ldots, X_{O+H}\}$. The complete set of variables is denoted by $\mathscr{X} = \mathscr{O} \cup \mathscr{H}$. For simplicity, we assume that all observed variables have S_O states and all hidden variables have S_H states. For now assume $S_H = S_O$ and all conditional probability tables have full rank (we address the more general case later).

The joint distribution of $\mathscr{X}$ in a latent tree model is fully characterized by a set of conditional probability tables (CPTs). More specifically, we can select an arbitrary node in the tree as the root, and sort the nodes in the tree in topological order. Then the set of CPTs between nodes and their parents $P(X_i|X_{\pi(i)})$ are sufficient to characterize the joint distribution (the root node X_r has no parent, *i.e.*, $P(X_r|X_{\pi(r)}) = P(X_r)$),

$$P(x_1, \ldots, x_{O+H}) = \prod_{i=1}^{O+H} P(x_i|x_{\pi(i)}) \tag{4.6}$$

Compared to tree models which are defined solely on observed variables (*e.g.*, models obtained from the (Chow and Liu, 1968b) algorithm), latent tree models encompass a much larger classes of models, allowing more flexibility in modeling observed variables. This is evident if we compute the marginal distribution of the observed variables by summing out the latent ones, which gives us a clique over the observed nodes:

$$P(x_1, \ldots, x_O) = \sum_{x_{O+1}} \cdots \sum_{x_{O+H}} \prod_{i=1}^{O+H} P(x_i|x_{\pi(i)}) \tag{4.7}$$

For simplicity, assume that all leaves are observed variables and all internal nodes are latent. For further notation, let X_{i^*} be some leaf in the subtree rooted at X_i, and let T_i denote the set of all such leaves. and X_{-i^*} be a leaf *not* in the subtree rooted at X_i and let T_{-i} denote the set of all those leaves. Let $c_j(i)$ denote the j^{th} child of node X_i, and where X_i has α_i children.

4.3 Derivation of Spectral Algorithm

Our spectral derivation has three main components:

1. Showing how the marginal probability tensor $\boldsymbol{\mathcal{P}}(X_1, \ldots, X_O)$ can be factorized into a collection of lower order tensors where the maximum tensor order is equal to the maximum degree of the tree. This can be shown to be equivalent to a tensor message passing scheme.
2. Inserting the invertible transformations $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ into the message passing scheme to define an alternate low rank factorization that still yields the same marginal probabilities as the original representation.
3. Setting $\boldsymbol{F}$ such that the factors in this alternate factorization only depend on small groups of observed variables.

For simplicity of exposition, we assume all internal nodes have exactly three neighbors. The factorization generalizes to trees of arbitrary topology, but this comes at the cost of memory since the maximum order of the tensor in the factorization is equal to the maximum degree of a node in the tree. However, we show that in Chapter 7 that there exists a representation where the maximum tensor order is 3 regardless of the order of the tree [1]

[1]While it is possible to binarize a tree while adding additional latent variables and setting the factors to enforce certain constraints, these constraints are difficult to enforce in the spectral algorithm.

4.3.1 Factorizing the Marginal Probability Tensor

Consider Figure 4.1. First let us exploit the low rank structure implied by the root A. We can get the following factorizations of the marginal probability tensor $\boldsymbol{\mathcal{P}}(E,F,G,H,I,J)$:

$$\boldsymbol{\mathcal{P}}(E,F,G,H,I,J) = \boldsymbol{\mathcal{P}}(E,F|A)\boldsymbol{\mathcal{P}}(\oslash_2 A)\boldsymbol{\mathcal{P}}(G,H,I,J|A)^\top$$
$$\boldsymbol{\mathcal{P}}(E,F,G,H,I,J) = \boldsymbol{\mathcal{P}}(E,F,G,H|A)\boldsymbol{\mathcal{P}}(\oslash_2 A)\boldsymbol{\mathcal{P}}(I,J|A)^\top$$
$$\boldsymbol{\mathcal{P}}(E,F,G,H,I,J) = \boldsymbol{\mathcal{P}}(E,F,I,J|A)\boldsymbol{\mathcal{P}}(\oslash_2 A)\boldsymbol{\mathcal{P}}(G,H|A)^\top$$

But how to combine all of these into one factorization? With tensors! Let the root probability $P(A)$ be embedded in a third order labeled tensor: $\boldsymbol{\mathcal{P}}(\oslash_3 A)$. One can see that by using (labeled) tensor-matrix multiplication,

$$\boldsymbol{\mathcal{P}}(E,F,G,H,I,J) = \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{\mathcal{P}}(E,F|A) \times_A \boldsymbol{\mathcal{P}}(G,H|A) \times_A \boldsymbol{\mathcal{P}}(I,J|A) \tag{4.8}$$

We then proceed to factorize recursively e.g. $\boldsymbol{\mathcal{P}}(E,F|A) = \boldsymbol{\mathcal{P}}(\oslash_2 B|A) \times_B \boldsymbol{\mathcal{P}}(E|B) \times_B \boldsymbol{\mathcal{P}}(F|B)$.

4.3.2 Tensor Message Passing

The recursive factorization procedure described above can be expressed as message passing, a form that will more be notationally convenient for our subsequent derivation. Instead of attempting to reconstruct the entire marginal probability tensor let us simply focus on a single element of this tensor (e.g. $P(\bar{e},\bar{f},\bar{g},\bar{h},\bar{i},\bar{j})$). Again Figure 4.1 will be used as a running example. Associate each node with the following labeled tensor (which are essentially the original conditional probability tables (CPTs) embedded into higher-order tensors):

- *root*: $\boldsymbol{\mathcal{R}} := \boldsymbol{\mathcal{P}}(\oslash_3 X_r)$
- *internal node*: $\boldsymbol{\mathcal{T}}_i := \boldsymbol{\mathcal{P}}(\oslash_2 X_i|X_{\pi(i)})$
- *leaf*: $\boldsymbol{\mathcal{L}}_i = \boldsymbol{\mathcal{P}}(X_i|X_{\pi(i)})$

Assuming the leaf node is an evidence variable, the message it passes to its parent is the following vector:

$$\boldsymbol{m}_i = \boldsymbol{\mathcal{L}}_i \times_i \delta_{\bar{x}_i} \tag{4.9}$$

Example: $\boldsymbol{M}_E = \boldsymbol{\mathcal{P}}(\bar{e}|B) = \boldsymbol{\mathcal{P}}(E|B) \times_E \delta_{\bar{e}}$

Similarly, the message from a non-root internal node sends to its parent is:

$$\boldsymbol{m}_i = \boldsymbol{\mathcal{T}}_i \times_i \boldsymbol{m}_{c_1(i)} \times_i \boldsymbol{m}_{c_2(i)} \tag{4.10}$$

Example: $\boldsymbol{m}_B = \boldsymbol{\mathcal{P}}(\oslash_2 B|A) \times_B \boldsymbol{m}_E \times_B \boldsymbol{m}_F$

and finally the root agglomerates the messages to give the final probability:

$$P(\bar{x}_1, ..., \bar{x}_O) = \boldsymbol{\mathcal{R}} \times_r \boldsymbol{m}_{c_1(r)} \times_r \boldsymbol{m}_{c_2(r)} \times_r \boldsymbol{m}_{c_3(r)} \tag{4.11}$$

Example: $P(\bar{e}, \bar{f}, \bar{g}, \bar{h}, \bar{i}, \bar{j}) = \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{m}_B \times_A \boldsymbol{m}_C \times_A \boldsymbol{m}_D$.

4.3.3 Transformed Representation

Next, note we do not need to recover the tensor representation explicitly if our focus is to perform inference using the message passing algorithm as in (7.6)–(7.8). As long as we can recover the tensor representation up to some invertible transformation, we can still obtain the correct marginal probability $P(\bar{x}_1, ..., \bar{x}_O)$. More specifically, we can insert (labeled) identity matrices $\boldsymbol{I}$ with mode labels $\{A, A\}$ into the message update equations without changing the final probability. Continuing the running example in Figure 4.1,

$$\begin{aligned} P(\bar{e}, \bar{f}, \bar{g}, \bar{h}, \bar{i}, \bar{j}) &= \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{m}_B \times_A \boldsymbol{m}_C \times_A \boldsymbol{m}_D \\ &= \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{I} \times_A \boldsymbol{m}_B \times_A \boldsymbol{I} \times_A \boldsymbol{m}_C \times_A \boldsymbol{I} \times_A \boldsymbol{m}_D \end{aligned}$$

Next, expand $\boldsymbol{I}$ as a matrix inversion pair $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ and regroup the terms:

$$\begin{aligned} P(\bar{e}, \bar{f}, \bar{g}, \bar{h}, \bar{i}, \bar{j}) &= (\boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{F}_B \times_A \boldsymbol{F}_C \times_A \boldsymbol{F}_D) \\ &\quad \times_{\omega_B} \left(\boldsymbol{m}_B \times_A \boldsymbol{F}_B^{-1}\right) \\ &\quad \times_{\omega_C} \left(\boldsymbol{m}_C \times_A \boldsymbol{F}_C^{-1}\right) \\ &\quad \times_{\omega_D} \left(\boldsymbol{m}_D \times_A \boldsymbol{F}_D^{-1}\right) \end{aligned}$$

where ω_B, ω_C, ω_D are mode labels that depend on the definitions of $\boldsymbol{F}_B, \boldsymbol{F}_C, \boldsymbol{F}_D$ respectively and are defined in the next section. This proceeds recursively e.g.

$$\boldsymbol{m}_D = \boldsymbol{\mathcal{P}}(\bar{i}, \bar{j}|A) = \left(\boldsymbol{\mathcal{P}}(\oslash_2 D|A) \times_D \boldsymbol{F}_I \times_D \boldsymbol{F}_J\right) \times_{\omega_I} \left(\boldsymbol{\mathcal{P}}(\bar{i}|D) \times_D \boldsymbol{F}_I^{-1}\right) \times_{\omega_J} \left(\boldsymbol{\mathcal{P}}(\bar{j}|D) \times_D \boldsymbol{F}_J^{-1}\right)$$

In general, we can define the following transformed tensor representation:

- **root**: $\widetilde{\boldsymbol{\mathcal{R}}} = \boldsymbol{\mathcal{P}}(\oslash_3 X_r) \times_r \boldsymbol{F}_{c_1(r)} \times_r \boldsymbol{F}_{c_2(r)} \times_r \boldsymbol{F}_{c_3(r)}$
- **internal**: $\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{P}}(\oslash_2 X_i|X_{\pi(i)}) \times_{\pi(i)} \boldsymbol{F}_i^{-1} \times_i \boldsymbol{F}_{c_1(i)} \times_i \boldsymbol{F}_{c_2(i)}$
- **leaf**: $\widetilde{\boldsymbol{\mathcal{L}}}_i = \boldsymbol{\mathcal{P}}(X_i|X_{\pi(i)}) \times_{\pi(i)} \boldsymbol{F}_i^{-1}$

4.3.4 Observable Representation

We now derive the observable representation by choosing $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ systematically, so that we can recover each transformed parameter using the marginal probability of a small set of observed variables. This is summarized by the lemma below, generalizing our intuition from Section 4.1.

Lemma 5. *For each X_i, set $\boldsymbol{F}_i = \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})$ with mode labels $\{\omega_i = X_{i^*}, X_{\pi(i)}\}$. Then we have the following:*

- $\widetilde{\boldsymbol{\mathcal{R}}} = \boldsymbol{\mathcal{P}}(X_{c_1(r)^*}, X_{c_2(r)^*}, X_{c_3(r)^*})$

Original	Tensor	Observable(Spectral)
$P(A)$	$\mathcal{P}(\oslash_3 A)$	$\mathcal{P}(E,G,I)$
$P(B\|A)$	$\mathcal{P}(\oslash_2 B\|A)$	$\mathcal{P}(E,F,G) \times_G \mathcal{P}(G,E)^{-1}$
$P(E\|B)$	$\mathcal{P}(E\|B)$	$\mathcal{P}(E,G) \times_G \mathcal{P}(G,E)^{-1} = \boldsymbol{I}$

Table 4.1: Original, tensor, and observable parameters for nodes A, B, and E in the example in Figure 4.1 for the case where $S_H = S_O$. Note that the observable parameters are not unique, but this is one valid set.

- $\mathcal{T}_i = \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{-i^*} \mathcal{P}(X_{-i^*}, X_{i^*})^{-1}$
- $\mathcal{L}_i = \mathcal{P}(X_i, X_{-i^*}) \times_{-i^*} \mathcal{P}(X_{-i^*}, X_{i^*})^{-1} = \boldsymbol{I}$

Proof. Below is the derivation for internal nodes (the root and leaf are just special cases). Recall that X_{i^*} is some leaf in the subtree rooted at X_i and X_{-i^*} is a leaf *not* in the subtree rooted at X_i. First note that,

$$\begin{aligned}
\widetilde{\mathcal{T}}_i &= \mathcal{P}(\oslash_2 X_i|X_{\pi(i)}) \times_{\pi(i)} \boldsymbol{F}_i^{-1} \times_i \boldsymbol{F}_{c_1(i)} \times_i \boldsymbol{F}_{c_2(i)} \\
&= \mathcal{P}(\oslash_2 X_i|X_{\pi(i)}) \times_{\pi(i)} \mathcal{P}(X_{i^*}|X_{\pi(i)})^{-1} \times_i \mathcal{P}(X_{c_1(i)^*}|X_i) \times_i \mathcal{P}(X_{c_2(i)^*}|X_i) \\
&= \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}|X_{\pi(i)}) \times_{\pi(i)} \mathcal{P}(X_{i^*}|X_{\pi(i)})^{-1} \qquad (4.12)
\end{aligned}$$

We now prove the following relation:

$$\widetilde{\mathcal{T}}_i \times_{i^*} \mathcal{P}(X_{i^*}, X_{-i^*}) = \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \qquad (4.13)$$

Note that $\mathcal{P}(X_{i^*}, X_{-i^*}) = \mathcal{P}(X_{i^*}|X_{\pi(i)})\mathcal{P}(\oslash X_{\pi(i)})\mathcal{P}(X_{-i^*}|X_{\pi(i)})^\top$. Thus,

$$\begin{aligned}
\widetilde{\mathcal{T}}_i \times_{i^*} \mathcal{P}(X_{i^*}, X_{-i^*}) &= \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}|X_{\pi(i)}) \times_{\pi(i)} \mathcal{P}(X_{i^*}|X_{\pi(i)})^{-1} \times_{i^*} (\mathcal{P}(X_{i^*}|X_{\pi(i)})\mathcal{P}(\oslash X_{\pi(i)})\mathcal{P}(X_{-i^*}|X_{\pi(i)})^\top) \\
&= \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}|X_{\pi(i)}) \times_{\pi(i)} \mathcal{P}(\oslash X_{\pi(i)}) \times_{\pi(i)} \mathcal{P}(X_{-i^*}|X_{\pi(i)})^\top \\
&= \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*})
\end{aligned}$$

From here, one can conclude that

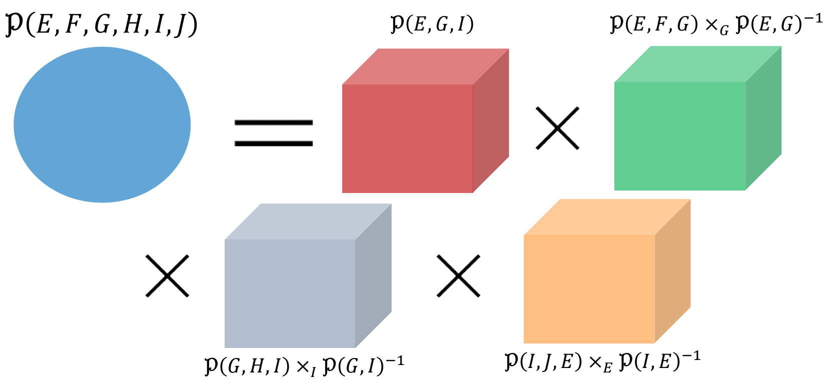
$$\widetilde{\mathcal{T}}_i = \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{-i^*} \mathcal{P}(X_{-i^*}, X_{i^*})^{-1} \qquad (4.14)$$

□

Example: In the example in Figure 4.1, one possible value of $\widetilde{\mathcal{T}}_B$ is $\mathcal{P}(E,F,G) \times_G \mathcal{P}(G,E)^{-1}$. Table 4.1 shows a table comparing the original, tensor, and observable (spectral) parameters, and Figure 4.2 shows a graphical illustration of the observable representation.

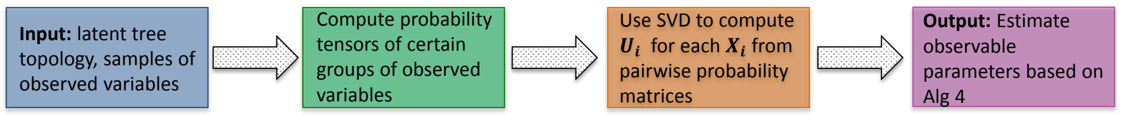
4.3.5 Training and Test

Training: In training, we compute the observable (spectral) parameters as shown in Algorithm 4. We will describe the purpose of $\boldsymbol{U}_i$ and the thin SVD in the next section. A high level flowchart



$\mathcal{P}(E,F,G,H,I,J)$
$\mathcal{P}(E,G,I)$
$\mathcal{P}(E,F,G) \times_G \mathcal{P}(E,G)^{-1}$
$\mathcal{P}(G,H,I) \times_I \mathcal{P}(G,I)^{-1}$
$\mathcal{P}(I,J,E) \times_E \mathcal{P}(I,E)^{-1}$

Figure 4.2: Illustration of spectral decomposition of example in Figure 4.1.



Input: latent tree topology, samples of observed variables
Compute probability tensors of certain groups of observed variables
Use SVD to compute U_i for each X_i from pairwise probability matrices
Output: Estimate observable parameters based on Alg 4

Figure 4.3: Flowchart that gives an overview of Algorithm 4

is shown in Figure 4.3 and an algorithm box tailored for the example in Figure 4.1 is shown in Algorithm 3.

Test: In test time, the goal is that given a set of evidence variables $\mathcal{E} = \{\bar{x}_{e_1}, ..., \bar{x}_{e_{|\mathcal{E}|}}\}$ to compute the probability $P(\bar{x}_{e_1}, ..., \bar{x}_{|\mathcal{E}|})$. We can do this using the tensor message passing scheme shown before except with the transformed parameters instead of the original parameters. The only difference is that depending on if $\bar{x}_i$ is an evidence variable or not the leaf message changes.

Algorithm 3 Spectral learning algorithm for example in Figure 4.1 when $S_H = S_O$

In: Tree topology and N *i.i.d.* samples of E, F, G, H, I, J

Out: Estimated observable root, internal, and leaf parameters: $\widehat{\mathcal{R}}_A, \widehat{\mathcal{T}}_B, \widehat{\mathcal{T}}_C, \widehat{\mathcal{T}}_D, \widehat{\mathcal{L}}_E, \widehat{\mathcal{L}}_F, \widehat{\mathcal{L}}_G, \widehat{\mathcal{L}}_H, \widehat{\mathcal{L}}_I, \widehat{\mathcal{L}}_J$

```
1: Compute the following triple/pairwise probability tensors:
       $\widehat{\mathcal{P}}(E,G,I), \widehat{\mathcal{P}}(E,F,G), \widehat{\mathcal{P}}(G,H,I), \widehat{\mathcal{P}}(I,J,E),$
       $\widehat{\mathcal{P}}(E,G), \widehat{\mathcal{P}}(G,I), \widehat{\mathcal{P}}(I,E)$
2: Compute observable parameters as:
       $\widehat{\mathcal{R}}_A = \widehat{\mathcal{P}}(E,G,I)$
       $\widehat{\mathcal{T}}_B = \widehat{\mathcal{P}}(E,F,G) \times_G (\mathcal{P}(E,G))^{-1}$
       $\widehat{\mathcal{T}}_C = \widehat{\mathcal{P}}(G,H,I) \times_I (\mathcal{P}(G,I))^{-1}$
       $\widehat{\mathcal{T}}_D = \widehat{\mathcal{P}}(I,J,E) \times_E (\mathcal{P}(I,E))^{-1}$
       $\widehat{\mathcal{L}}_E = \boldsymbol{I}, \widehat{\mathcal{L}}_F = \boldsymbol{I}, \widehat{\mathcal{L}}_G = \boldsymbol{I}, \widehat{\mathcal{L}}_H = \boldsymbol{I}, \widehat{\mathcal{L}}_I = \boldsymbol{I}, \widehat{\mathcal{L}}_J = \boldsymbol{I}$
```

If leaf is an evidence variable multiply by the indicator vector to select the appropriate row:

$$\boldsymbol{m}_i = \widetilde{\boldsymbol{\mathcal{L}}}_i \times_i \boldsymbol{\delta}_{\bar{x}_i} \tag{4.15}$$

If leaf is not an evidence variable sum over X_i:

$$\boldsymbol{m}_i = \widetilde{\boldsymbol{\mathcal{L}}}_i \times_i \mathbf{1} \tag{4.16}$$

The general algorithm is in given in Algorithm 5.

4.3.6 Expectation Form

We will find it useful later in this chapter, as well as in Chapter 6 to consider an equivalent form of the spectral algorithm in terms of expectations. We simply give the algorithm below again in this form for future use. As noted in 2.3, probability matrices can be written in expectation form:

$$\begin{aligned}
\boldsymbol{\mathcal{R}} &= \boldsymbol{\mathcal{P}}(\oslash_3 X_r) = \mathbb{E}_{X_r}[\boldsymbol{\delta}_{X_r} \otimes \boldsymbol{\delta}_{X_r} \otimes \boldsymbol{\delta}_{X_r}] \\
\boldsymbol{\mathcal{T}}_i &= \boldsymbol{\mathcal{P}}(\oslash_2 X_i | X_{\pi(i)}) = \mathbb{E}_{X_i|X_{\pi(i)}}[\boldsymbol{\delta}_{X_i} \otimes \boldsymbol{\delta}_{X_i} | X_{\pi(i)}] \\
\boldsymbol{\mathcal{L}}_i &= \boldsymbol{\mathcal{P}}(X_i | X_{\pi(i)}) = \mathbb{E}_{X_i|X_{\pi(i)}}[\boldsymbol{\delta}_{X_i} | X_{\pi(i)}]
\end{aligned}$$

where $\boldsymbol{\delta}_{X_i}$ is the indicator vector with a one in the position of the value of X_i.

Algorithm 4 Spectral learning algorithm for latent tree graphical models

```
In: Tree topology and $N$ i.i.d. samples $\left\{x_1^n, \dots, x_{|O|}^n\right\}_{n=1}^N$
Out: Estimated observable root, internal, and leaf parameters, $\widehat{\boldsymbol{\mathcal{R}}}, \widehat{\boldsymbol{\mathcal{T}}}_i$ for each internal node, $\widehat{\boldsymbol{\mathcal{L}}}_i$ for each leaf node,
 1: For each node $X_i$, perform a "thin" singular value decomposition of $\widehat{\boldsymbol{\mathcal{P}}}(X_{i^*}, X_{-i^*}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$; let $\widehat{\boldsymbol{U}}_i = \boldsymbol{U}(:, 1 : S_H)$ be the
    the first $S_H$ principal left singular vectors.
 2: Compute observable parameters as:
      $\widehat{\boldsymbol{\mathcal{R}}} = \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(r)^*}, X_{c_2(r)^*}, X_{c_3(r)^*}) \times_{c_1(r)} \widehat{\boldsymbol{U}}_{c_1(r)} \times_{c_2(r)} \widehat{\boldsymbol{U}}_{c_2(r)} \times_{c_3(r)} \widehat{\boldsymbol{U}}_{c_3(r)}$
      $\widehat{\boldsymbol{\mathcal{T}}}_i = \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{-i^*} (\widehat{\boldsymbol{\mathcal{P}}}(X_{-i^*}, X_{i^*}) \times_{i^*} \widehat{\boldsymbol{U}}_i)^\dagger \times_{c_1(r)} \widehat{\boldsymbol{U}}_{c_1(r)} \times_{c_2(r)} \widehat{\boldsymbol{U}}_{c_2(r)}$
      $\widehat{\boldsymbol{\mathcal{L}}}_i = \widehat{\boldsymbol{\mathcal{P}}}(X_i, X_{-i^*}) \times_{-i^*} (\widehat{\boldsymbol{\mathcal{P}}}(X_{-i^*}, X_{i^*}) \times_{i^*} \widehat{\boldsymbol{U}}_i)^\dagger = \widehat{\boldsymbol{U}}_i$
```

Algorithm 5 Inference with Spectral Parameters

```
In: Tree topology, set of spectral parameters $\widehat{\boldsymbol{\mathcal{R}}}, \widehat{\boldsymbol{\mathcal{T}}}_i foreachinternalnode, \widehat{\boldsymbol{\mathcal{L}}}_i$ for each leaf node, and set of evidence
$\mathcal{E} = \{\bar{x}_{e_1}, ..., \bar{x}_{e_{|\mathcal{E}|}}\}$
Out: estimated probability $P(\bar{x}_{e_1}, ..., \bar{x}_{|\mathcal{E}|})$

 1: In reverse topological order, each node accumulates at message at leaf and sends to parent
      • Evidence Leaf: $\widehat{\boldsymbol{m}}_i = \widehat{\boldsymbol{\mathcal{L}}}_i \times_i \delta_{\bar{x}_i}$
      • Non-Evidence Leaf: $\widehat{\boldsymbol{m}}_i = \widehat{\boldsymbol{\mathcal{L}}}_i \times_i \mathbf{1}$
      • Internal Node: $\widehat{\boldsymbol{m}}_i = \widehat{\boldsymbol{\mathcal{T}}}_i \times_i \widehat{\boldsymbol{m}}_{c_1(i)} \times_i \widehat{\boldsymbol{m}}_{c_2(i)}$
      • Root: $\widehat{P}(\bar{x}_{e_1}, ..., \bar{x}_{|\mathcal{E}|}) = \widehat{\boldsymbol{\mathcal{R}}} \times_i \boldsymbol{m}_{c_1(i)} \times_i \widehat{\boldsymbol{m}}_{c_2(i)} \times_i \widehat{\boldsymbol{m}}_{c_3(i)}$
```

Setting $\boldsymbol{F}_i = \boldsymbol{\mathcal{P}}(X_i|H) = \mathbb{E}[\boldsymbol{\delta}_{X_i}|H]$ gives the expectation forms of the observable representation:

- $\widetilde{\boldsymbol{\mathcal{R}}} = \mathbb{E}[\boldsymbol{\delta}_{X_{c_1(i)^*}} \otimes \boldsymbol{\delta}_{X_{c_2(i)^*}} \otimes \boldsymbol{\delta}_{X_{c_3(i)^*}}]$
- $\boldsymbol{\mathcal{T}}_i = \mathbb{E}[\boldsymbol{\delta}_{X_{c_1(i)^*}} \otimes \boldsymbol{\delta}_{X_{c_2(i)^*}} \otimes \boldsymbol{\delta}_{X_{-i^*}}] \times_{-i^*} \mathbb{E}[\boldsymbol{\delta}_{X_{-i^*}} \otimes \boldsymbol{\delta}_{X_{i^*}}]^{-1}$
- $\boldsymbol{\mathcal{L}}_i = \mathbb{E}[\boldsymbol{\delta}_{X_{-i^*}} \otimes \boldsymbol{\delta}_{X_{i^*}}] \times_{-i^*} \mathbb{E}[\boldsymbol{\delta}_{X_{-i^*}} \otimes \boldsymbol{\delta}_{X_{i^*}}]^{-1}$

4.4 Dealing with $S_H \neq S_O$

In the above derivations we have assumed $S_H = S_O$ and all underlying probability tables are full rank. This guarantees that the inverse of $\boldsymbol{\mathcal{P}}(X_i^*, X_{-i^*})$ exists and that:

$$\boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*})^{-1} = \left(\boldsymbol{\mathcal{P}}(X_{-i^*}|X_{\pi(i)})^{-1}\right)^{\top} \boldsymbol{\mathcal{P}}(\oslash X_{\pi(i)})^{-1} \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})^{-1} \tag{4.17}$$

which is essential for the correctness of the algorithm (Lemma 5). However, in general, $S_H \neq S_O$ causing either the inverse to not exist or the above equality to not hold. We detail these two cases below.

4.4.1 What if $S_H < S_O$?

Based on our intuition in Chapter 3, rank (equivalent to the number of latent states if we assume full rank CPTs) is a measure of the amount of long range dependency in a latent model. Thus, $S_H < S_O$ actually means shorter range dependencies and should be easier to solve than the $S_H = S_O$ case. However, the algorithm described in the previous section does not directly apply since $\boldsymbol{\mathcal{P}}(X_i^*, X_{-i^*})$ is no longer invertible. The solution is to simply project all the matrices/tensors to the S_H dimensional space where the inverse exists. As a result, instead of choosing $\boldsymbol{F} = \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})$ we choose $\boldsymbol{F}_i = \boldsymbol{U}_i^{\top}\boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})$ where $\boldsymbol{U}_i$ is the top S_H left singular vectors of $\boldsymbol{\mathcal{P}}(X_i^*, X_{-i^*})$ and replacing $\boldsymbol{F}^{-1}$ with $\boldsymbol{F}^{\dagger}$ where $\dagger$ indicates pseudo-inverse. We show why this works below using the derivation of the internal node (root and leaf are special cases):

$$\begin{aligned}
\widetilde{\boldsymbol{\mathcal{T}}}_i &= \boldsymbol{\mathcal{P}}(\oslash_2 X_i|X_{\pi(i)}) \times_{\pi(i)} \boldsymbol{F}_i^{\dagger} \times_i \boldsymbol{F}_{c1(i)} \times_i \boldsymbol{F}_{c2(i)} \\
&= \boldsymbol{\mathcal{P}}(\oslash_2 X_i|X_{\pi(i)}) \times_{\pi(i)} \left(\boldsymbol{U}_i^{\top}\boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})\right)^{\dagger} \times_i \boldsymbol{U}_{c_1(i)}^{\top}\boldsymbol{\mathcal{P}}(X_{c_1(i)^*}|X_i) \times_i \boldsymbol{U}_{c_2(i)}^{\top}\boldsymbol{\mathcal{P}}(X_{c_2(i)^*}|X_i) \\
&= \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*}|X_{\pi(i)}) \times_{\pi(i)} \left(\boldsymbol{U}_i^{\top}\boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})\right)^{\dagger} \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)}^{\top} \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)}^{\top}
\end{aligned} \tag{4.18}$$

Again we prove the following relation:

$$\widetilde{\boldsymbol{\mathcal{T}}}_i \times_{i^*} \boldsymbol{U}_i^{\top}\boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*}) = \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)}^{\top} \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)}^{\top} \tag{4.19}$$

where again, X_{-i^*} is an observed leaf that is not a descendant of X_i.

The proof follow similarly as before by expanding $\boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*})$:

$$\begin{aligned}
\widetilde{\boldsymbol{\mathcal{T}}}_i \times_{i^*} \boldsymbol{U}_i^T \boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*}) &= \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*} | X_{\pi(i)}) \times_{\pi(i)} \left(\boldsymbol{U}_i^T \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})\right)^{\dagger} \\
&\quad \times_{i^*} (\boldsymbol{U}_i^T \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)}) \boldsymbol{\mathcal{P}}(\oslash X_{\pi(i)}) \boldsymbol{\mathcal{P}}(X_{-i^*}|X_{\pi(i)})^\top) \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)}^\top \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)}^\top \\
&= \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*} | X_{\pi(i)}) \times_{\pi(i)} \boldsymbol{\mathcal{P}}(\oslash X_{\pi(i)}) \times_{\pi(i)} \boldsymbol{\mathcal{P}}(X_{-i^*}|X_{\pi(i)})^\top \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)}^\top \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)}^\top \\
&= \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)}^\top \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)}^\top \qquad (4.20)
\end{aligned}$$

From here, one can conclude that

$$\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{-i^*} \left(\boldsymbol{U}_i^T \boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*})\right)^{\dagger} \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)}^\top \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)}^\top \qquad (4.21)$$

This explains the presence of $\boldsymbol{U}_i$ in Algorithm 4.

4.4.2 What if $S_H > S_O$?

On the other hand, when $S_H > S_O$, $\boldsymbol{\mathcal{P}}(X_i^*, X_{-i^*})$ is full rank but the relation $\boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*})^{-1} = (\boldsymbol{\mathcal{P}}(X_{-i^*}|X_{\pi(i)})^{-1})^\top \boldsymbol{\mathcal{P}}(\oslash X_{\pi(i)})^{-1} \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})^{-1}$ no longer holds. This scenario indicates longer range dependencies and therefore a more challenging learning scenario. For intuition on why this is the case, consider the following generative process:

1. With probability 0.5, let $S = X$, and with probability 0.5 let $S = Y$.
2. Print A m times.
3. Print S
4. Go back to step (2)

With $m = 1$ we either generate $AXAXAXA$ or $AYAYAYA$... With $m = 2$ we either generate $AAXAAXAA$.. or $AAYAAYAA$..

Note that an HMM, which is a natural way to model this generative process, needs $2m$ states to be able to correctly capture the long range dynamics. This is because it needs to remember the count as well as whether we picked $S = X$ or $S = Y$ in Step 1. However, the number of observed states m does not change, so spectral learning as we have described it so far, will break for $m > 2$. How to deal with this in the spectral framework?

Siddiqi et al. (2010) and Cohen et al. (2012) proposed solutions to this problem by constructing features out of groups of observations. In our case, this would mean constructing features out of all the observed descendants in the subtree rooted at X_i (denoted as T_i) instead of using just one descendant X_i^*.

This can be accomplished using the expectation form we derived in 4.3.6 and setting $\boldsymbol{F}_i = \mathbb{E}[\boldsymbol{\phi}_{T_i}|X_{\pi(i)}]$ where $\boldsymbol{\phi}_{T_i}$ is an arbitrary feature vector that can be a function of any of the variables in T_i. Note that $\boldsymbol{U}_i$ would now be the SVD of $\mathbb{E}[\boldsymbol{\phi}_{T_i} \otimes \boldsymbol{\phi}_{T_{-i}}]$ instead of $\mathbb{E}[\boldsymbol{\delta}_{X_i^*} \otimes \boldsymbol{\delta}_{X_{-i}^*}]$. For the example in Figure 4.1, we could set $\boldsymbol{F}_B = \mathbb{E}[\boldsymbol{\phi}_{T_B}|A]$ where $\boldsymbol{\phi}_{T_B} := [\boldsymbol{\delta}_E; \boldsymbol{\delta}_F]$ (; represents vertical concatenation) and $\boldsymbol{\phi}_{T_{-B}} = [\boldsymbol{\delta}_G; \boldsymbol{\delta}_H; \boldsymbol{\delta}_I; \boldsymbol{\delta}_J]$. This would give the following observable representation:

- $\widetilde{\mathcal{R}} = \mathbb{E}[\boldsymbol{\phi}_{T_{c_1(i)}} \otimes \boldsymbol{\phi}_{T_{c_1(i)}} \otimes \boldsymbol{\phi}_{T_{c_1(i)}}] \times_{T_{c_1(i)}} \boldsymbol{U}_{c1(i)} \times_{T_{c_2(i)}} \boldsymbol{U}_{c2(i)} \times_{T_{c_3(i)}} \boldsymbol{U}_{c3(i)}$
- $\mathcal{T}_i = \mathbb{E}[\boldsymbol{\phi}_{T_{c_1(i)}} \otimes \boldsymbol{\phi}_{T_{c_2(i)}} \otimes \boldsymbol{\phi}_{T_{-i^*}}] \times_{T_{-i^*}} \left(\boldsymbol{U}_i^\top \mathbb{E}[\boldsymbol{\phi}_{T_{i^*}} \otimes \boldsymbol{\phi}_{T_{-i^*}}]\right)^{-\dagger} \times_{T_{c_1(i)}} \boldsymbol{U}_{c1(i)} \times_{T_{c_2(i)}} \boldsymbol{U}_{c2(i)}$
- $\mathcal{L}_i = \mathbb{E}[\boldsymbol{\phi}_{T_{-i^*}} \otimes \boldsymbol{\phi}_{T_{i^*}}] \times_{T_{-i^*}} \left(\boldsymbol{U}_i^\top \mathbb{E}[\boldsymbol{\phi}_{T_{i^*}} \otimes \boldsymbol{\phi}_{T_{-i^*}}]\right)^{\dagger}$

4.4.3 Linear Systems to Improve Stability

There is another technique we can use to improve performance in practice. Consider the below relation for non-root internal nodes, which is key to the derivation of the algorithm (same as Eq. 4.13):

$$\widetilde{\mathcal{T}}_i \times_{i^*} \mathcal{P}(X_{i^*}, X_{-i^*}) = \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \tag{4.22}$$

which then implies that

$$\widetilde{\mathcal{T}}_i = \mathcal{P}(X_{c_1(i)}, X_{c_2(i)}, X_{-i^*}) \times_{-i^*} \mathcal{P}(X_{-i^*}, X_{i^*})^{-1} \tag{4.23}$$

However, there may be many choices of X_{-i^*} (which we denote with $X_{-i^*}^{(1)}, ..., X_{-i^*}^{(Z)}$) for which the above equality is true:

$$\begin{array}{rcl}
\widetilde{\mathcal{T}}_i \times_{i^*} \mathcal{P}(X_{i^*}, X_{-i^*}^{(1)}) & = & \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}^{(1)}) \\
\widetilde{\mathcal{T}}_i \times_{i^*} \mathcal{P}(X_{i^*}, X_{-i^*}^{(2)}) & = & \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}^{(2)}) \\
 & ... & \\
\widetilde{\mathcal{T}}_i \times_{i^*} \mathcal{P}(X_{i^*}, X_{-i^*}^{(Z}) & = & \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}^{(Z)})
\end{array}$$

This defines an over-constrained linear system, and we can solve for $\widetilde{\mathcal{T}}_i$ using least squares. In the case where $S_O > S_H$, and the projection matrix $\boldsymbol{U}_i$ is needed, our system of equations becomes:

$$\begin{array}{rcl}
\widetilde{\mathcal{T}}_i \times_{i^*} (\boldsymbol{U}_i^\top \mathcal{P}(X_{i^*}, X_{-i^*}^{(1)})) & = & \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}^{(1)}) \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)} \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)} \\
\widetilde{\mathcal{T}}_i \times_{i^*} (\boldsymbol{U}_i^\top \mathcal{P}(X_{i^*}, X_{-i^*}^{(2)})) & = & \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}^{(1)}) \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)} \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)} \\
 & ... & \\
\widetilde{\mathcal{T}}_i \times_{i^*} (\boldsymbol{U}_i^\top \mathcal{P}(X_{i^*}, X_{-i^*}^{(Z)})) & = & \mathcal{P}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}^{(1)}) \times_{c_1(i)^*} \boldsymbol{U}_{c_1(i)} \times_{c_2(i)^*} \boldsymbol{U}_{c_2(i)}
\end{array}$$

In this case, one good choice of $\boldsymbol{U}_i$ is the top singular vectors of the matrix formed by the horizontal concatenation of $\mathcal{P}(X_{i^*}, X_{-i^*}^{(1)}), ..., \mathcal{P}(X_{i^*}, X_{-i^*}^{(Z)})$.

This linear system method allows for more robust estimation, especially in smaller sample sizes (at the cost of more computation). It can be applied to the leaf nodes as well. One does not need to set up the linear system with all the valid choices; a subset is also acceptable.

4.5 Sample Complexity Analysis

We analyze the sample complexity of Algorithm 4 and show that it depends on the tree topology and the spectral properties of the true model. Let d_i be the degree of node i in the tree.

Theorem 1. *Let $d_{\max} = \max_i d_i$. Then, for any $\epsilon > 0, 0 < \delta < 1$, if*

$$N \geq O\left(\left(\frac{4S_H^2}{3\beta^2} \right)^{d_{\max}} \frac{S_O \ln \frac{|O+H|}{\delta} |O+H|^2}{\epsilon^2 \alpha^4} \right)$$

where $\sigma_\tau(*)$ returns the τ^{th} largest singular value and

$$\alpha = \min_i \ \sigma_{S_H}(\boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*})), \quad \beta = \min_i \ \sigma_{S_H}(\boldsymbol{F}_i)$$

Then with probability $1 - \delta$,
$\sum_{x_1,\dots,x_O} \left| \widehat{P}_{spectral}(x_1, \dots, x_O) - P(x_1, \dots, x_O) \right| \leq \epsilon$, where $\widehat{P}_{spectral}$ indicates the probability returned by the spectral algorithm.

Proof. The proof is a special case of Theorem 3 in Chapter 5 for latent junction trees. □

Note the dependence on the singular values of certain probability tensors. In fully observed models, the accuracy of the learned parameters depends only on how close the empirical estimates of the factors are to the true factors. However, our spectral algorithm also depends on how close the inverses of these empirical estimates are to the true inverses, which depends on the spectral properties of the matrices (Stewart and Sun, 1990).

4.6 Empirical Results

We present a set of synthetic experiments to evaluate our method in a variety of settings. We compare our method to the Expectation Maximization (EM) algorithm (Dempster et al., 1977). Convergence of EM is determined by measuring the change in the log likelihood at iteration t (denoted by $f(t)$) over the average: $\frac{|f(t)-f(t-1)|}{\text{avg}(f(t),f(t-1))} \leq \gamma$. To explore the computational / accuracy trade-off we consider a variety of choices of γ. We denote *EM*− to indicate convergence threshold $\gamma = 1e-3$ (low precision EM), *EM* to indicate convergence threshold $\gamma = 1e-4$ (standard precision as used in Murphy (2005)), and *EM*+ to indicate convergence threshold $\gamma = 1e-5$ (high precision EM). All variants given 5 restarts by default, and our method is given a linear system size of 10 (as discussed in 4.4.3) unless specified otherwise.

We compare the methods on both accuracy and training time (All approaches have similar test runtime). For accuracy, we measure the performance of joint estimation using $\epsilon = \frac{|\widehat{P}(x_1,\dots,x_O) - P(x_1,\dots,x_O)}{P(x_1,\dots,x_O)}$ averaged over 1000 test points randomly drawn from the underlying model. For runtime, the training time of each method is reported in seconds. Results are reported as the training set size size is varied from 100 to 100,000.

The first set of experiments demonstrates performance as a function of tree depth (Figure 4.4). All trees generated were binary, and we experimented with depths 4,5,6, and 7. S_O was set to 4 and S_H was set to 2. In terms of accuracy, we generally observe 3 distinct regions, low-sample size, mid-sample size, and large sample size. In the low sample size region, *EM* and *EM*+ tend to overfit to the training data, and our spectral algorithm and EM- perform the best. In the mid-sample size region, *EM*− does poorly since it does not converge to a high enough precision. Our method does well but *EM* and *EM*+ tend to do better since they benefit from a smaller number of parameters and lack of dependence on singular values. However, once a certain sample size is reached (the large sample size region), our spectral algorithm consistently outperforms all the EM variants which suffer from local minima and convergence issues. From the perspective of runtime our method is considerably faster than all the variants, and is almost two orders of magnitude faster than *EM*+.

The second set of experiments (Figure 4.5) evaluates the effect of the number of restarts on EM. The tree structure used is a binary tree with depth 6 (64 total nodes), and as in the previous experiment S_O was set to 4 and S_H was set to 2. The results are as expected. More restarts tends to improve the performance of EM while also increasing runtime. However, we see that the effect of restarts is considerably smaller than the effect of the precision threshold for this family of models.

Lastly we evaluate the effect of the linear system size on the spectral algorithm as shown in Figure 4.6. In some sense this is the parameter that trades off accuracy and runtime for our method analogous to how the convergence threshold / number of restarts trade off accuracy and speed for EM. We fix the tree topology to a binary tree of depth 6 (64 nodes) and fix $S_O = 5$. S_H is varied from 2 to 5 and we compare linear system sizes of 1,2,5, and 10. When $S_H = 2$ all the variants perform similarly, but as the number of hidden states is increased, increasing the size of the linear system produces a considerably more accurate solution. Runtime is not considerably different across the different variants due to the efficiacy of MATLAB linear algebra libraries (although of course a larger linear system involves more computational cost).

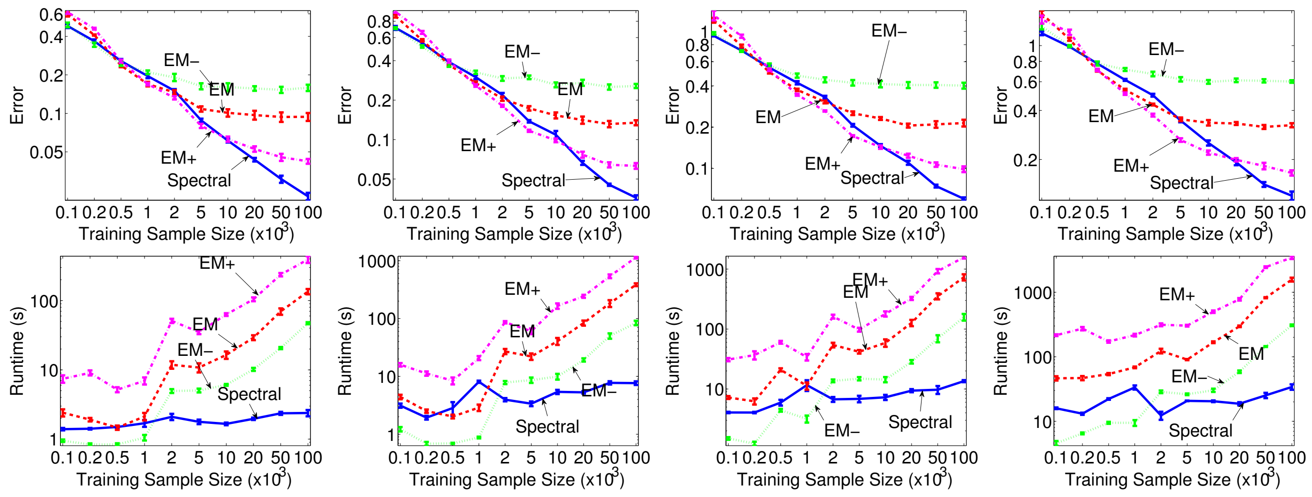
It should also be noted that while all the implementations in this chapter are single core, spectral learning easily parallelizes and can be run on very large datasets. Note that the only part of Algorithm 4 that explicitly depends on the data is computation of the probability / feature tensors, which can be done in parallel. After that, the SVD is the dominant step and can be done efficiently using randomized methods e.g. Halko et al. (2011).

4.7 Connections with Tensor Decomposition

Before moving on to the next chapter, we briefly discuss how spectral learning and EM are related to tensor decomposition methods in the applied mathematics community.

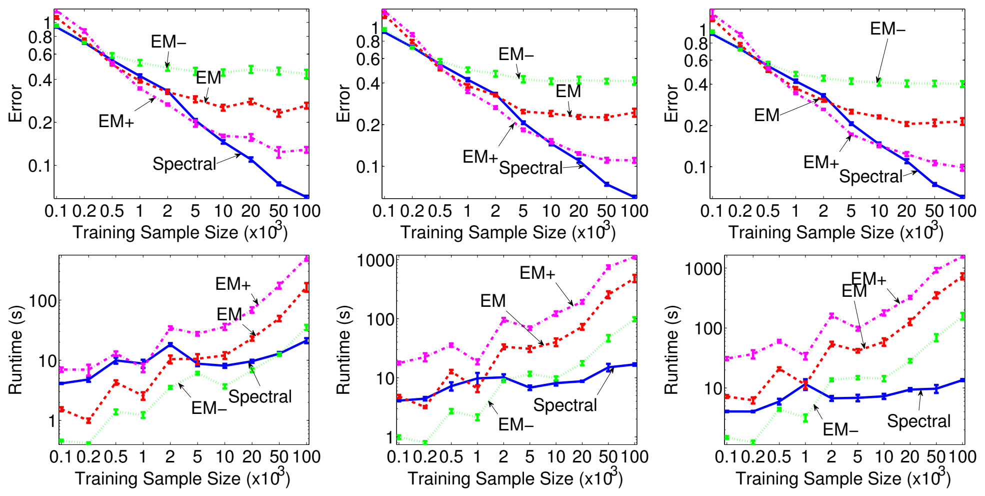
PARAFAC Decomposition (Harshman, 1970): Given an n^{th} order tensor $\mathcal{T}$ of dimensions $D_1 \times ....D_n$, PARAFAC decomposition of rank K consists of one $K \times 1$ vector $\boldsymbol{z}$, and n matrices $\boldsymbol{M}_1, ..., \boldsymbol{M}_n$, each of dimension $D_i \times K$. $\mathcal{T}$ is then approximated as:

$$\mathcal{T}_{parafac}(i_1, ..., i_n) = \sum_{k=1}^{K} \boldsymbol{z}(k)\boldsymbol{M}_1(i_1, k)....\boldsymbol{M}_n(i_n, k) \tag{4.24}$$



(a) Depth 4 (b) Depth 5 (c) Depth 6 (d) Depth 7

Figure 4.4: Comparison of our spectral algorithm (blue) to standard EM with convergence set at $1e-4$ (red), low precision EM with convergence set at $1e-3$ (green) and high precision EM with converge set at $1e-5$ (magenta) for binary trees of various depth. All variants of EM are given 5 random restarts and we use a max linear system size of 10 for the spectral algorithm. Number of observed states (S_O) is fixed to 4 and number of latent states (S_H) is fixed to 2. Both errors and runtimes in log scale.



(a) 1 restart (b) 3 restarts (c) 5 restarts

Figure 4.5: Comparison of our spectral algorithm (blue) to standard EM with convergence set at $1e-4$ (red), low precision EM with convergence set at $1e-3$ (green) and high precision EM with converge set at $1e-5$ (magenta) for different numbers of restarts provided given to EM. The latent tree structure is a binary tree with depth 6 (64 total nodes). Number of observed states (S_O) is fixed to 4 and number of latent states (S_H) is fixed to 2. For the spectral algorithm we use a max linear system size of 10. Both errors and runtimes in log scale.

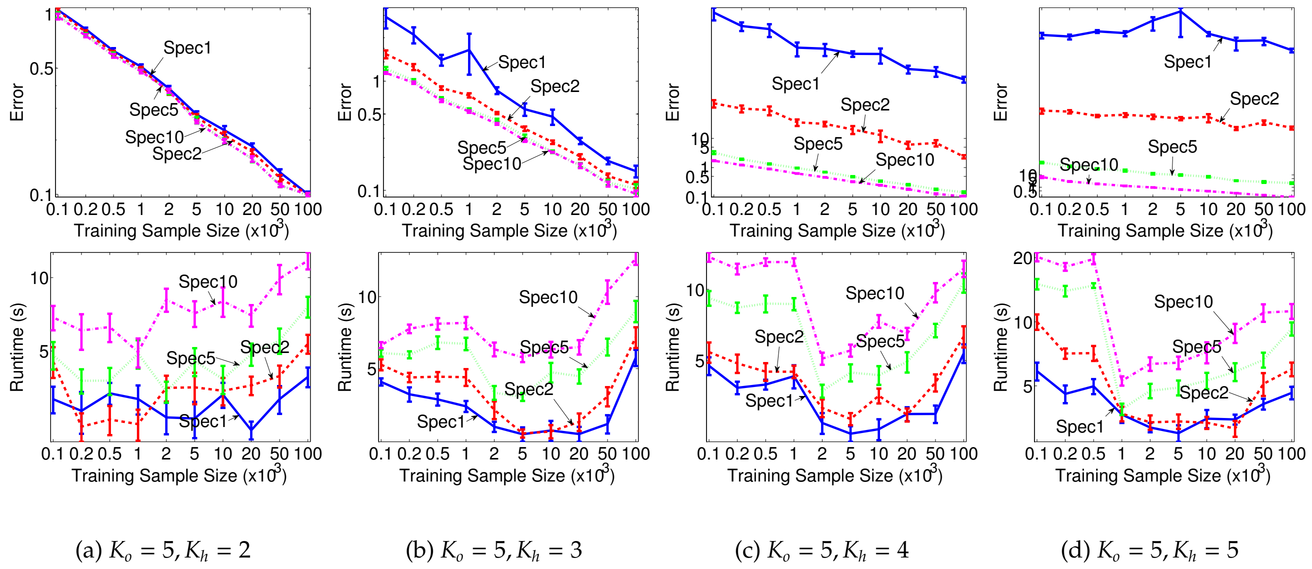


Figure 4.6: Evaluation of the spectral algorithm for different values of the max linear system size: 1-blue, 2-red, 5-green, 10-magenta. The latent tree structure is a binary tree with depth 6 (64 total nodes). Number of observed states (S_O) is fixed to 5 and number of latent states (S_H) is varied from 2 to 5 across (a), (b), (c), (d). Both errors and runtimes in log scale.

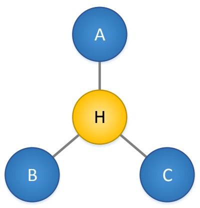


Figure 4.7: Example of a latent tree model with three observed nodes

$\boldsymbol{z}, \boldsymbol{M}_1, ..., \boldsymbol{M}_k$ are found by attempting to minimize a certain loss (traditional PARAFAC decomposition uses the ℓ_2 loss). This is non-convex, and is thus done typically with alternating minimization. But this makes it easy to constrain the probability tables to sum to one. Moreover, if the KL-divergence is used for the loss it is equivalent to using EM to learn the graphical model shown in Figure 4.7.

Tucker Decomposition (Hitchcock, 1927; Tucker, 1966): We have seen that PARAFAC decomposition is related to EM. We now describe a tensor decomposition that is related to spectral learning.

Given an n^{th} order tensor $\mathcal{T}$ of dimensions $D_1 \times ....D_n$, TUCKER decomposition of rank K consists of one $K \times K \times ... \times K$ core tensor $\boldsymbol{Z}$, and n matrices $\boldsymbol{M}_1, ..., \boldsymbol{M}_n$, each of dimension $D_i \times K$. T is then approximated as:

$$\mathcal{T}_{tucker}(i_1, ..., i_n) = \sum_{k_1=1}^{K} ... \sum_{k_n=1}^{K} \boldsymbol{Z}(k_1, ..., k_n)\boldsymbol{M}_1(i_1, k_1)....\boldsymbol{M}_n(i_n, k_n) \tag{4.25}$$

Note that unlike the PARAFAC Decomposition, the Tucker decomposition has a provably consistent learning algorithm using SVD. Recall that $\mathcal{T}_{\{A\},\{B\}}$ is the unfolding of $\mathcal{T}$ such that the modes in the set A are on the rows and the modes of set B are on the columns. Then the following algorithm recovers the Tucker decomposition:

- Set M_i to the top k singular vectors of $\mathcal{T}_{\{i\},\{1,\ldots,i-1,i+1,\ldots,n\}}$ for each i
- Define $\mathcal{Z} := \mathcal{T} \times_1 M_1 \ldots \times_n M_n$

Note that this is equivalent to the spectral algorithm for the graphical model shown in Figure 4.7.

4.8 Conclusion

This chapter presents a spectral learning algorithm for latent tree graphical models that is provably consistent unlike traditional methods. Empirical results show the algorithm gives a favorable statistical/computational trade-off to the EM algorithm across a variety of settings often giving a 1-2 order of magnitude speed up.

This chapter also serves a foundation for the rest of Part I of this thesis. The following 3 chapters will extend this work in a variety of ways.

Chapter 5

A Spectral Algorithm for Latent Junction Trees

The previous chapter proposed a provably consistent spectral learning algorithm for latent tree models. However, latent structures beyond trees, such as higher order HMMs (Kundu et al., 1989), factorial HMMs (Ghahramani and Jordan, 1997) and Dynamic Bayesian Networks (Murphy, 2002), are needed and have been proven useful in many real world problems. The challenges for generalizing spectral algorithms to general latent structured models include the larger factors, more complicated conditional independence structures, and the need to sum out multiple variables simultaneously.

Contribution of this chapter: In this thesis, we take one step toward developing spectral methods for more general graphical models by proposing a spectral algorithm for latent junction trees. The key idea of our approach is to embed the clique potentials of the junction tree into higher order tensors such that the computation of the marginal probability of observed variables can be carried out via tensor operations. While this novel representation leads only to a moderate increase in the number parameters for junction trees of low treewidth, it allows us to design an algorithm that can recover a transformed version of the tensor parameterization and ensure that the joint probability of observed variables are computed correctly and consistently. As with the spectral algorithm described in the previous chapter, the main computation of the algorithm involves only tensor operations and singular value decompositions which can be orders of magnitude faster than EM algorithms for large datasets. To our knowledge, this is the first provably consistent parameter learning technique for latent variable models beyond trees. In our experiments, we show that our spectral algorithm can be almost 2 orders of magnitude faster than EM while at the same achieving comparable or better accuracy. Our spectral algorithm also achieves comparable accuracy to EM on real data.

Outline: A high level overview of our approach is given in Figure 5.1. We first provide some additional notation and background on latent junction trees. We then derive the spectral algorithm by representing junction tree message passing with tensor operations, and then transform this representation into one that only depends on observed variables. Finally, we analyze the sample complexity of our method and evaluate it on synthetic and real datasets.

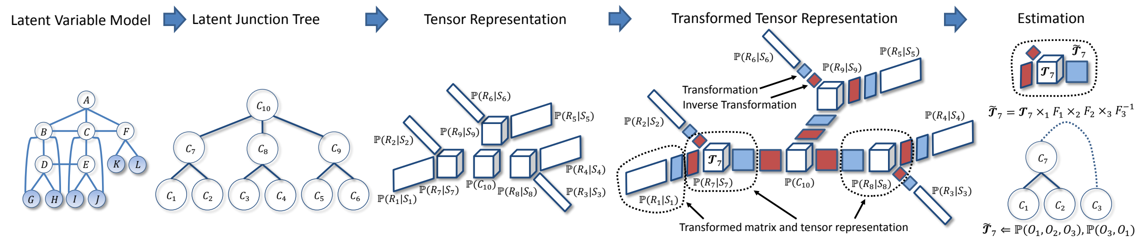


Figure 5.1: Our algorithm for local-minimum-free learning of latent variable models consist of four major steps. (1) First, we transform a model into a junction tree, such that each node in the junction tree corresponds to a maximal clique of variables in the triangulated graph of the original model. (2) Then we embed the clique potentials of the junction tree into higher order tensors and express the marginal distribution of the observed variables as a tensor-tensor/matrix multiplication according to the message passing algorithm. (3) Next we transform the tensor representation by inserting a pair of transformations between those tensor-tensor/matrix operations. Each pair of transformations is chosen so that they are inversions of each other. (4) Lastly, we show that each transformed representation is a function of only observed variables. Thus, we can estimate each individual transformed tensor quantity using samples from observed variables.

Prerequisites: This chapter assumes a general understanding of latent variable models as presented in 2.2, the connection between latent variable models and low rank factorization in Chapter 3, and the tensor notation in 3.1. It is also recommended to read Chapter 4 first.

5.1 Additional Tensor Notation

In addition to the notation in 3.1 we define some additional operations that are unique to this chapter.

Tensor multi-mode multiplication for labeled tensors Let $\boldsymbol{\sigma} = \{X_1, \ldots, X_k\}$ be an arbitrary set of k modes (k variables) shared by $\mathcal{A}$ and $\mathcal{B}$ (w.l.o.g. we assume these labels correspond to the first k modes, and $I_1 = J_1, \ldots, I_k = J_k$ holds for the corresponding dimensions). Then multiplying two labeled tensors $\mathcal{A}$ and $\mathcal{B}$ along $\boldsymbol{\sigma}$ results in

$$\mathcal{D} = \mathcal{A} \times_{\sigma} \mathcal{B} \in \mathbb{R}^{I_{k+1} \times \ldots \times I_N \times J_{k+1} \times \ldots \times J_M}, \tag{5.1}$$

where the entries of $\mathcal{D}$ are defined as

$$\mathcal{D}(\boldsymbol{i}_{k+1:N}, \boldsymbol{j}_{k+1:M}) = \sum_{\boldsymbol{i}_{1:k}} \mathcal{A}(\boldsymbol{i}_{1:k}, \boldsymbol{i}_{k+1:N}) \mathcal{B}(\boldsymbol{i}_{1:k}, \boldsymbol{j}_{k+1:M}).$$

Multi-mode multiplication can also be interpreted as reshaping the $\boldsymbol{\sigma}$ modes of $\mathcal{A}$ and $\mathcal{B}$ into a single mode and doing single-mode tensor multiplication. Furthermore, tensor multiplication with labels is symmetric in its arguments, *i.e.*, $\mathcal{A} \times_{\sigma} \mathcal{B} \cong \mathcal{B} \times_{\sigma} \mathcal{A}$.

Mode-specific labeled identity tensor. We now define our notion of identity tensor with respect to a set of modes $\boldsymbol{\sigma} = \{X_1, \ldots, X_K\}$. Let $\mathcal{A}$ be a labeled tensor with mode labels containing $\boldsymbol{\sigma}$, and $\mathcal{I}_{\sigma}$ be a tensor with $2K$ modes with mode labels $\{X_1, \ldots, X_K, X_1, \ldots, X_K\}$. Then $\mathcal{I}_{\sigma}$ is an identity tensor with respect to modes $\boldsymbol{\sigma}$ if

$$\mathcal{A} \times_{\sigma} \mathcal{I}_{\sigma} \cong \mathcal{A}. \tag{5.2}$$

One can also understand $\mathcal{I}_{\sigma}$ using its matrix representation: flattening $\mathcal{I}_{\sigma}$ with respect to $\boldsymbol{\sigma}$ (the first $\boldsymbol{\sigma}$ modes mapped to rows and the second $\boldsymbol{\sigma}$ modes mapped to columns) results in an identity matrix.

Mode-specific labeled tensor inversion. Let $\mathcal{F}, \mathcal{F}^{-1} \in \mathbb{R}^{c_1(i) \times \cdots \times I_K \times I_{K+1} \times \cdots \times I_{K+K'}}$ be labeled tensors of order $K + K'$, and both have two sets of mode labels $\boldsymbol{\sigma} = \{X_1, \ldots, X_K\}$ and $\boldsymbol{\omega}' = \{X_{K+1}, \ldots, X_{K+K'}\}$. Then $\mathcal{F}^{-1}$ is the inverse of $\mathcal{F}$ w.r.t. modes $\boldsymbol{\omega}$ if and only if

$$\mathcal{F} \times_{\omega} \mathcal{F}^{-1} \cong \mathcal{I}_{\sigma}. \tag{5.3}$$

Multimode inversion can also be interpreted as reshaping $\mathcal{F}$ with respect to $\boldsymbol{\omega}$ into a matrix of size $(c_1(i) \ldots I_K) \times (I_{K+1} \ldots I_{K+K'})$, taking the inverse, and then rearranging back into a tensor. Thus the existence and uniqueness of this inverse can be characterized by the rank of the matricized version of $\mathcal{F}$.

5.2 Latent Junction Trees

In this chapter, we will focus on discrete latent variable models where the number of states, S_H, for each hidden variable is much smaller than the number of states, S_O, for each observed variable. A latent variable model defines a joint probability distribution over a set of variables $\mathscr{X} = \mathscr{O} \cup \mathscr{H}$. Let $O := |\mathscr{O}|$ and $H := |\mathscr{H}|$. Here, $\mathscr{O}$ denotes the set of observed variables, $\{X_1, \ldots, X_O\}$ and $\mathscr{H}$ denotes the set of hidden variables, $\{X_{O+1}, \ldots, X_{O+H}\}$

We will focus on latent variable models where the structure of the model is a junction tree of low treewidth (Cowell et al., 1999). Each node C_i in a junction tree corresponds to a subset (clique) of variables from the original graphical model. We will also use C_i to denote the collection of variables contained in the node, *i.e.* $C_i \subset \mathscr{X}$. Let $\mathbb{C}$ denote the set of all clique nodes. The treewidth is then the size of a largest clique in a junction tree minus one, that is $t = \max_{C_i \in \mathbb{C}} |C_i| - 1$. Furthermore, we associate each edge in a junction tree with a separator set $S_{ij} := C_i \cap C_j$ which contains the common variables of the two cliques C_i and C_j it is connected to. If we condition on all variables in any S_{ij}, the variables on different sides of S_{ij} will become independent.

As in the previous chapter, without loss of generality, we assume that each internal clique node in the junction tree has exactly 3 neighbors.[1] Then we can pick a clique C_r as the root of the tree and reorient all edges away from the root to induce a topological ordering of the clique nodes. Given the ordering, the root node will have 3 children nodes, denoted as $C_{c_1(r)}, C_{c_2(r)}$ and $C_{c_3(r)}$. Each other internal node C_i will have a unique parent node, denoted as $C_{\pi(i)}$, and 2 children nodes denoted as $C_{c_1(i)}$ and $C_{c_2(i)}$. Each leaf node C_l is only connected with its unique parent node $C_{\pi(l)}$. Furthermore, we can simplify the notation for the separator set between a node C_i and its parent $C_{\pi(i)}$ as $S_i = C_i \cap C_{\pi(i)}$, omitting the index for the parent node. Then the remainder set of a node is defined as $R_i = C_i \setminus S_i$. We also assume w.l.o.g. that if C_i is a leaf in the junction tree, R_i consists of

[1]If this is not the case, the derivation is similar but notationally much heavier.

only observed variables. We will use r_i to denote an instantiation of the set of variables in R_i. See Figure 5.2 for an illustration of notation.

Given a root and a topological ordering of the nodes in a junction tree, the joint distribution of all variables $\mathscr{X}$ can be factorized according to

$$P(\mathscr{X}) = \prod_{i=1}^{|\mathscr{X}|} P(R_i|S_i), \tag{5.4}$$

where each CPT $P(R_i|S_i)$, also called a clique potential, corresponds to a node C_i. The number of parameters needed to specify the model is $O(|\mathbb{C}|S_O^t)$, linear in the number of cliques but exponential in the tree width t. Then the marginal distribution of the observed variables can be obtained by summing over the latent variables,

$$P(x_1, ..., x_O) = \sum_{X_{O+1}} \cdots \sum_{X_{O+H}} \left[\prod_{i=1}^{|\mathscr{X}|} P(R_i|S_i) \right], \tag{5.5}$$

Note that each (non-leaf) remainder set R_i contains a small subset of all latent variables. The presence of latent variables introduces complicated dependency between observed variables, while at the same time only a small number of parameters corresponding to the entries in the CPTs are needed to specify the model.

The process of eliminating the latent variables in (5.5) can be carried out efficiently via message passing. Each node only needs to sum out a small number of variables and then the intermediate result, called the message, is passed to its parent for further processing. In the end the root node incorporates all messages from its children and produces the final result $P(x_1, ..., x_O)$. The local summation step, called the message update, can be generically written as[2]

$$\boldsymbol{\mathcal{M}}(S_i) = \sum\nolimits_{R_i} P(R_i|S_i)\boldsymbol{\mathcal{M}}(S_{c_1(i)})\boldsymbol{\mathcal{M}}(S_{c_2(i)}) \tag{5.6}$$

where we use $\boldsymbol{\mathcal{M}}(S_i)$ to denote the intermediate results of eliminating variables in the remainder set R_i. This message update is then carried out recursively according the reverse topological order of the junction tree until we reach the root node. The local summation step for the leaf nodes and root node can be viewed as special cases of (5.6). For a leaf node C_l, there is no incoming message from children nodes, and hence $\boldsymbol{\mathcal{M}}(S_l) = P(r_l|S_l)$; for the root node C_r, $S_r = \emptyset$ and $R_i = C_i$, and hence $P(x_1, ..., x_O) = \boldsymbol{\mathcal{M}}(\emptyset) = \sum_{\forall X \in C_r} P(C_r)\boldsymbol{\mathcal{M}}(S_{c_1(r)})\boldsymbol{\mathcal{M}}(S_{c_2(r)})\boldsymbol{\mathcal{M}}(S_{c_3(r)})$.

Example. The message update at the internal node C_{BCDE} in Figure 5.2 is

$$\boldsymbol{\mathcal{M}}(\{C, E\}) = \sum\nolimits_{B,D} P(B, D|C, E)P(f|B, C)P(g|B, D).$$

5.3 Tensor Representation for Message Passing

Although the parametrization of latent junction trees using CPTs is very compact and inference (message passing) can be carried out efficiently, parameters in this representation can be difficult to learn. Since the likelihood of the observed data is no longer concave in the latent parameters,

[2]For simplicity of notation, assume $C_i = S_i \cup S_{c_1(i)} \cup S_{c_2(i)}$.

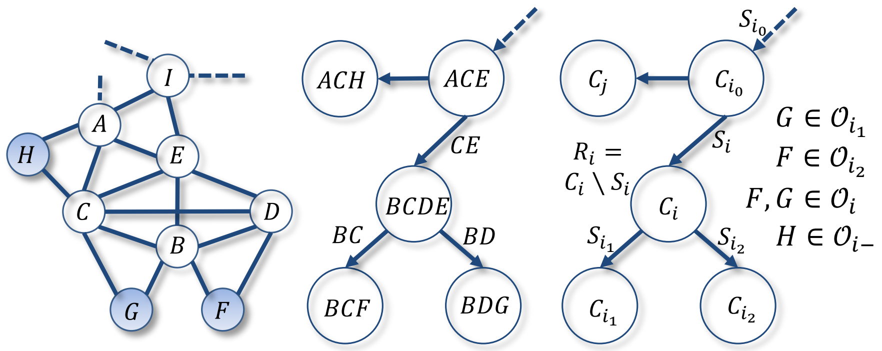


Figure 5.2: Example latent variable models with variables $\mathscr{X} = \{A, B, C, D, E, F, G, H, I, \ldots\}$, the observed variables are $\mathscr{O} = \{F, G, H, \ldots\}$ (only partially drawn). Its corresponding junction tree is shown in the middle panel. Corresponding to this junction tree, we also show the general notation for it in the rightmost panel.

local search heuristics, such as EM, are often employed to learning the parameters. Therefore, our goal is to design a new representation for latent junction trees, such that subsequent learning can be carried out in a local-optima-free fashion.

In this section, we will develop a new representation for the message update in (5.6) by embedding each CPT $P(R_i|S_i)$ into a higher order tensor $\boldsymbol{\mathcal{P}}(C_i)$. This form is a strict generalization of the tensor representation proposed in Chapter 4.

5.3.1 Embed CPTs to higher order tensors

As we can see from (7.4), the joint probability distribution of all variables can be represented by a set of conditional distributions over just subsets of variables. Each one of this conditionals is a low order tensor. For example in Figure 5.2, the CPT corresponding to the clique node C_{BCDE} would be a 4th order tensor $\boldsymbol{\mathcal{P}}(B, D|C, E)$ where each variable corresponds to a different mode of the tensor. However, this representation is not suitable for deriving the observable representation since message passing cannot be defined easily using the tensor multiplication/sum connection shown above. Instead we will embed these tensors into even higher order tensors to facilitate the computation. The key idea is to introduce duplicate indexes using the mode-specific identity tensors, such that the sum-product steps in message updates can be expressed as tensor multiplications.

More specifically, the number of times a mode of the tensor is duplicated will depend on how many times the corresponding variable in the clique C_i appears in the separator sets incident to C_i. We can define the count for a variable $X_j \in C_i$ as

$$d_{j,i} = \mathbb{I}[X_j \in S_i] + \mathbb{I}[X_j \in S_{c_1(i)}] + \mathbb{I}[X_j \in S_{c_2(i)}], \tag{5.7}$$

where $\mathbb{I}[\cdot]$ is an indicator function taking value 1 if its argument is true and 0 otherwise. Then the labeled tensor representation of the node C_i is

$$\mathcal{P}(C_i) := \mathcal{P}(\underbrace{\ldots,(\oslash_{d_{j,i}}X_j),\ldots}_{\forall X_j\in R_i}|\underbrace{\ldots,(\oslash_{d_{j',i}}X_{j'}),\ldots}_{\forall X_{j'}\in S_i}), \tag{5.8}$$

where the labels for the modes of the tensor are the combined labels of the separator sets, *i.e.*, $\{S_i, S_{c_1(i)}, S_{c_2(i)}\}$. The number of times a variable is repeated in the label set is exactly equal to $d_{j,i}$.

Essentially, the labeled tensor $\mathcal{P}(C_i)$ contains exactly the same information as the original CPT $P(R_i|S_i)$. Furthermore, $\mathcal{P}(C_i)$ has a lot of zero entries, and the entries from $P(R_i|S_i)$ are simply embedded in the higher order tensor $\mathcal{P}(C_i)$. Suppose all variables in node C_i are latent variables each taking S_H values. Then the number of entries needed to specify $P(R_i|S_i)$ is $S_H^{|C_i|}$, while the high order tensor $\mathcal{P}(C_i)$ has $S_H^{d_i}$ entries where $d_i := \sum_{j:X_j\in C_i} d_{j,i}$ which is never smaller than $S_H^{|C_i|}$. In a sense, the parametrization using higher order tensor $\mathcal{P}(C_i)$ is less compact than the parametrization using the original CPTs. However, constructing the tensor $\mathcal{P}$ this way allows us to express the junction tree message update step in (5.6) as tensor multiplications (more details in the next section), and then we can leverage tools from tensor analysis to design a local-minimum-free learning algorithm.

The tensor representation for the leaf nodes and the root node are special cases of the representation in (5.8). The tensor representation at a leaf node C_l is simply equal to $\mathcal{P}(C_l) = \mathcal{P}(R_l|S_l)$. The root node C_r has no parent, so $\mathcal{P}(C_r) = \mathcal{P}(\ldots,(\oslash_{d_{j,r}}X_j),\ldots)$, $\forall X_j \in C_r$. Furthermore, since $d_{j,i}$ is simply a count of how many times a variable in C_i appears in each of the incident separators, the size of each tensor does not depend on which clique node was selected as the root.

Example. In Figure 5.2, node C_{BCDE} corresponds to CPT $P(B,D|C,E)$. Its high order tensor representation is $\mathcal{P}(C_{BCDE}) = P(\oslash_2 B, D|\oslash_2 C, E)$, since both B and C occur twice in the separator sets incident to C_{BCDE}. Therefore the tensor $\mathcal{P}(\{B,C,D,E\})$ is a 6th order tensor with mode labels $\{B,B,D,C,C,E\}$.

5.3.2 Tensor message passing

With the higher order tensor representation for clique potentials in the junction tree as in (5.8), we can express the message update step in (5.6) as tensor multiplications. Consequently, we can compute the marginal distribution of the observed variables $\mathcal{O}$ in equation (5.5) recursively using a sequence of tensor multiplications. More specifically the general message update equation for a node in a junction tree can be expressed as

$$\mathcal{M}(S_i) = \mathcal{P}(C_i) \times_{S_{c_1(i)}} \mathcal{M}(S_{c_1(i)}) \times_{S_{c_2(i)}} \mathcal{M}(S_{c_2(i)}). \tag{5.9}$$

Here the modes of the tensor $\mathcal{P}(C_i)$ are labeled by the variables, and the mode labels are used to carry out tensor multiplications as explained in Section 5.1.

The tensor message passing steps in leaf nodes and the root node are special cases of the tensor message update in equation (5.9). The outgoing message $\mathcal{M}(S_l)$ at a leaf node C_l can be computed by simply setting all variables in R_l to the actual observed values r_l, *i.e.*,

$$\mathcal{M}(S_l) = \mathcal{P}(C_l)_{R_l=r_l} = P(R_l = r_l|S_l). \tag{5.10}$$

At the root, we obtain the marginal probability of the observed variables by aggregating all incoming messages from its 3 children, *i.e.*,

$$P(x_1, ..., x_O) = \mathcal{P}(C_r) \times_{S_{c_1(r)}} \mathcal{M}(S_{c1(r)}) \times_{S_{c_2(r)}} \mathcal{M}(S_{c2(r)}) \times_{S_{c_3(r)}} \mathcal{M}(S_{c3(r)}) \tag{5.11}$$

Example. For Figure 5.2, using the following tensors

$$\begin{aligned} \mathcal{P}(\{B, C, D, E\}) &= P(\oslash_2 B, D \mid \oslash_2 C, E) \\ \mathcal{M}(\{B, C\}) &= P(f|B, C) \\ \mathcal{M}(\{B, D\}) &= P(g|B, D), \end{aligned}$$

we can write the message update for node C_{BCDE} in the form of equation (5.9) as

$$\begin{aligned} \mathcal{M}(\{C, E\}) = \mathcal{P}(\{B, C, D, E\}) \times_{\{B,C\}} \mathcal{M}(\{B, C\}) \\ \times_{\{B,D\}} \mathcal{M}(\{B, D\}). \end{aligned}$$

Note how the tensor multiplication sums out B and D: $\mathcal{P}(\{B, C, D, E\})$ has two B labels, and it appears in the subscripts of tensor multiplication twice; D appears once in the label and in the subscript of tensor multiplication respectively. Similarly, C is not summed out since there are two C labels but it appears only once in the subscript of tensor multiplication.

5.4 Transformed Representation

Explicitly learning the tensor representation in (5.8) is still an intractable problem. Our key observation, as that in Chapter 4 is that we do not need to recover the tensor representation explicitly if our focus is to perform inference using the message passing algorithm as in (5.9)–(5.11). As long as we can recover the tensor representation up to some invertible transformation, we can still obtain the correct marginal probability $P(x_1, ..., x_O)$.

More specifically, we can insert a mode-specific identity tensor $\mathcal{I}_\sigma$ into the message update equation in (5.9) without changing the outgoing message. Subsequently, we can then replace the mode-specific identity tensor by a pair of labeled tensors, $\mathcal{F}$ and $\mathcal{F}^{-1}$, which are mode-specific inversions of each other ($\mathcal{F} \times_\omega \mathcal{F}^{-1} \cong \mathcal{I}_\sigma$). Then we can group these inserted tensors with the representation $\mathcal{P}(C)$ from (5.8), and obtain a transformed version $\widetilde{\mathcal{P}}(C)$ (also see Figure 5.1). Furthermore, we have the freedom in choosing these collections of tensor inversion pairs. We will show that if we choose them systematically, we will be able to estimate each transformed tensor $\widetilde{\mathcal{P}}(C)$ using the marginal probability of a small set of observed variables (observable representation). In this section, we will first explain the transformed tensor representation.

Let us first consider a node C_i and its parent node $C_{\pi(i)}$. Then the outgoing message of $C_{\pi(i)}$ can be computed recursively as

$$\mathcal{M}(S_{\pi(i)}) = \mathcal{P}(C_{\pi(i)}) \times_{S_i} \underbrace{\mathcal{M}(S_i)}_{\mathcal{P}(C_i) \times_{S_{c_1(i)}} \mathcal{M}(S_{c_1(i)}) \times_{S_{c_2(i)}} \mathcal{M}(S_{c_2(i)})} \times \dots$$

Inserting a mode specific identity tensor $\mathcal{I}_{S_i}$ with labels $\{S_i, S_i\}$ and similarly defined mode specific identity tensors $\mathcal{I}_{S_{c_1(i)}}$ and $\mathcal{I}_{S_{c_2(i)}}$ into the above two message updates, we obtain

$$\mathcal{M}(S_{\pi(i)}) = \mathcal{P}(C_{\pi(i)}) \times_{S_i} (\mathcal{I}_{S_i} \times_{S_i} \underbrace{\mathcal{M}(S_i)}_{\mathcal{P}(C_i) \times_{S_{c_1(i)}} (\mathcal{I}_{S_{c_1(i)}} \times_{S_{c_1(i)}} \mathcal{M}(S_{c_1(i)})) \times_{S_{c_2(i)}} (\mathcal{I}_{S_{c_2(i)}} \times_{S_{c_2(i)}} \mathcal{M}(S_{c_1(i)}))}) \times \ldots$$

Then we can further expand $\mathcal{I}_{S_i}$ using tensor inversion pairs $\mathcal{F}_i, \mathcal{F}_i^{-1}$, *i.e.*, $\mathcal{I}_{S_i} = \mathcal{F}_i \times_{\omega_i} \mathcal{F}_i^{-1}$. Note that both $\mathcal{F}$ and $\mathcal{F}^{-1}$ have two set of mode labels, S_i and another set $\boldsymbol{\omega}_i$ which is related to the observable representation and explained in the next section. Similarly, we expand $\mathcal{I}_{S_{c_1(i)}}$ and $\mathcal{I}_{S_{c_2(i)}}$ using their corresponding tensor inversion pairs.

After expanding tensor identities $\mathcal{I}$, we can regroup terms, and at node C_i we have

$$\mathcal{M}(S_i) = (\mathcal{P}(C_i) \times_{S_{c_1(i)}} \mathcal{F}_{c_1(i)} \times_{S_{c_2(i)}} \mathcal{F}_{c_2(i)}) \times_{\omega_{c_1(i)}} (\mathcal{F}_{c_1(i)}^{-1} \times_{S_{c_1(i)}} \mathcal{M}(S_{c_1(i)})) \times_{\omega_{c_2(i)}} (\mathcal{F}_{c_2(i)}^{-1} \times_{S_{c_2(i)}} \mathcal{M}(S_{c_2(i)})) \tag{5.12}$$

and at the parent node $C_{\pi(i)}$ of C_i

$$\begin{aligned}\mathcal{M}(S_{\pi(i)}) =& (\mathcal{P}(C_{\pi(i)}) \times_{S_i} \mathcal{F}_i \times \ldots) \\ & \times_{\omega_i} (\mathcal{F}_i^{-1} \times_{S_i} \mathcal{M}(S_i)) \times \ldots\end{aligned} \tag{5.13}$$

Now we can define the transformed tensor representation for $\mathcal{P}(C_i)$ as

$$\widetilde{\mathcal{P}}(C_i) := \mathcal{P}(C_i) \times_{S_{c_1(i)}} \mathcal{F}_{c_1(i)} \times_{S_{c_2(i)}} \mathcal{F}_{c_2(i)} \times_{S_i} \mathcal{F}_i^{-1}, \tag{5.14}$$

where the two transformations $\mathcal{F}_{c_1(i)}$ and $\mathcal{F}_{c_2(i)}$ are obtained from the children side and the transformation $\mathcal{F}_i^{-1}$ is obtained from the parent side. Similarly, we can define the transformed representation for a leaf node and for the root node as

$$\widetilde{\mathcal{P}}(C_l) = \mathcal{P}(C_l) \times_{S_l} \mathcal{F}_l^{-1} \tag{5.15}$$

$$\widetilde{\mathcal{P}}(C_r) = \mathcal{P}(C_r) \times_{S_{c_1(r)}} \mathcal{F}_{c_1(r)} \times_{S_{c_2(r)}} \mathcal{F}_{c_2(r)} \times_{S_{c_3(r)}} \mathcal{F}_{c_3(r)} \tag{5.16}$$

Applying these definitions of the transformed representation recursively, we can perform message passing based purely on these transformed representations

$$\widetilde{\mathcal{M}}(S_{\pi(i)}) = \widetilde{\mathcal{P}}(C_{\pi(i)}) \times_{\omega_i} \underbrace{\widetilde{\mathcal{M}}(S_i)}_{\widetilde{\mathcal{P}}(C_i) \times_{\omega_{c_1(i)}} \widetilde{\mathcal{M}}(S_{c_1(i)}) \times_{\omega_{c_2(i)}} \widetilde{\mathcal{M}}(S_{c_2(i)})} \times \ldots \tag{5.17}$$

5.5 Observable Representation

We now derive the observable representation by choosing the collection of tensor pairs $\mathcal{F}$ and $\mathcal{F}^{-1}$ systematically, so that we can recover each transformed tensor $\widetilde{\mathcal{P}}(C)$ using the marginal probability of a small set of observed variables.

We will focus on the transformed tensor representation in (5.14) for an internal node C_i (other cases follow as special cases). Generalizing the idea in Chapter 4, we choose $\mathcal{F}_i = \mathcal{P}(\mathcal{O}_i | S_i)$ where

$\mathcal{O}_i$ is a set of observed variables in the subtree rooted at C_i. Plugging this in gives,

$$\widetilde{\mathcal{P}}(C_i) = \mathcal{P}(C_i) \times_{S_{c_1(i)}} \mathcal{P}(\mathcal{O}_{c_1(i)}|S_{c_1(i)}) \times_{S_{c_2(i)}} \mathcal{P}(\mathcal{O}_{c_2(i)}|S_{c_2(i)}) \times_{S_i} \mathcal{P}(\mathcal{O}_i|S_i)^{-1} \tag{5.18}$$

where the first two tensor multiplications essentially eliminate the latent variables in $S_{c_1(i)}$ and $S_{c_2(i)}$.[3] With these choices, we also fix the mode labels $\boldsymbol{\omega}_i$, $\boldsymbol{\omega}_{c_1(i)}$ and $\boldsymbol{\omega}_{c_2(i)}$ in (5.12) (5.13) and (5.17). That is $\boldsymbol{\omega}_i = \mathcal{O}_i$, $\boldsymbol{\omega}_{c_1(i)} = \mathcal{O}_{c_1(i)}$ and $\boldsymbol{\omega}_{c_2(i)} = \mathcal{O}_{c_2(i)}$.

To remove all dependencies on latent variables in $\widetilde{\mathcal{P}}(C_i)$ and relate it to observed variables, we need to eliminate the latent variables in S_i and the tensor $\mathcal{P}(\mathcal{O}_i|S_i)^{-1}$. For this, we multiply the transformed tensor $\widetilde{\mathcal{P}}(C_i)$ by $\mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i})$, where $\mathcal{O}_{-i}$ denotes some set of observed variables which do *not* belong to the subtree rooted at node C_i. Furthermore, $\mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i})$ can be re-expressed using the conditional distribution of $\mathcal{O}_i$ and $\mathcal{O}_{-i}$ respectively, conditioning on the separator set S_i, *i.e.*,

$$\mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i}) = \mathcal{P}(\mathcal{O}_i|S_i) \times_{S_i} \mathcal{P}(\oslash_2 S_i) \times_{S_i} \mathcal{P}(\mathcal{O}_{-i}|S_i).$$

Therefore, we have

$$\mathcal{P}(\mathcal{O}_i|S_i)^{-1} \times_{\mathcal{O}_i} \mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i}) = \mathcal{P}(\oslash_2 S_i) \times_{S_i} \mathcal{P}(\mathcal{O}_{-i}|S_i),$$

and plugging this into (5.18), we have

$$\begin{aligned}\widetilde{\mathcal{P}}(C_i) \times_{\mathcal{O}_i} \mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i}) &= \mathcal{P}(C_i) \times_{S_{c_1(i)}} \mathcal{P}(\mathcal{O}_{c_1(i)}|S_{c_1(i)}) \times_{S_{c_2(i)}} \mathcal{P}(\mathcal{O}_{c_2(i)}|S_{c_2(i)}) \times_{S_i} \mathcal{P}(\oslash_2 S_i) \times_{S_i} \mathcal{P}(\mathcal{O}_{-i}|S_i) \\ &= \mathcal{P}(\mathcal{O}_{c_1(i)}, \mathcal{O}_{c_2(i)}, \mathcal{O}_{-i}) \end{aligned} \tag{5.19}$$

where $\widetilde{\mathcal{P}}(C_i)$ is now related to only marginal probabilities of observed variables. From the equivalent relation, we can inverting $\mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i})$, and obtain the observable representation for $\widetilde{\mathcal{P}}(C_i)$

$$\widetilde{\mathcal{P}}(C_i) = \mathcal{P}(\mathcal{O}_{c_1(i)}, \mathcal{O}_{c_2(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} \mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i})^{-1}. \tag{5.20}$$

Example. For node C_{BCDE} in Figure 5.2, the choices of $\mathcal{O}_i$, $\mathcal{O}_{c_1(i)}$, $\mathcal{O}_{c_2(i)}$ and $\mathcal{O}_{-i}$ are $\{F, G\}$, G, F and H respectively.

There are many valid choices of $\mathcal{O}_{-i}$. As in Chapter 4 these different choices can be combined via a linear system using Eq. 5.19. This can substantially increase performance.

For the leaf nodes and the root node, the derivation for their observable representations can be viewed as special cases of that for the internal nodes. We provide the results for their observable representation below:

$$\widetilde{\mathcal{P}}(C_r) = \mathcal{P}(\mathcal{O}_{c_1(r)}, \mathcal{O}_{c_2(r)}, \mathcal{O}_{c_3(r)}), \tag{5.21}$$

$$\widetilde{\mathcal{P}}(C_l) = \mathcal{P}(\mathcal{O}_l, \mathcal{O}_{l-}) \times_{\mathcal{O}_{l-}} \mathcal{P}(\mathcal{O}_l, \mathcal{O}_{l-})^{-1}. \tag{5.22}$$

If $\mathcal{P}(\mathcal{O}_l, \mathcal{O}_{l-})$ is invertible, then $\widetilde{\mathcal{P}}(C_l) = \mathcal{I}_{\mathcal{O}_l}$. Otherwise we need to project $\mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i})$ using a tensor $\mathcal{U}_i$ to make it invertible, as discussed in the next section. The overall training and test algorithms are given in Algorithms 6 and 7. Given N *i.i.d.* samples of the observed nodes, we simply replace $\mathcal{P}(\cdot)$ by the empirical estimate $\widehat{\mathcal{P}}(\cdot)$. A high level flowchart of the training procedure is given in

[3]If a latent variable in $S_{c_1(i)} \cup S_{c_2(i)}$ is also in S_i, it is not eliminated in this step but in another step.

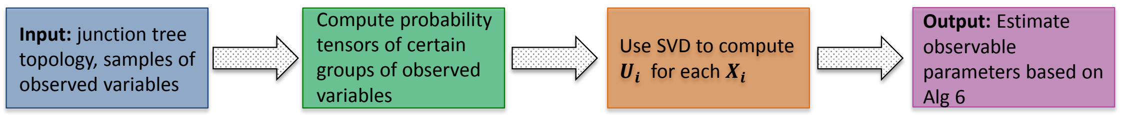

Input: junction tree topology, samples of observed variables
Compute probability tensors of certain groups of observed variables
Use SVD to compute $\boldsymbol{U_i}$ for each $\boldsymbol{X_i}$
Output: Estimate observable parameters based on Alg 6

Figure 5.3: Flowchart that gives an overview of Algorithm 6

Figure 5.3.

5.6 Discussion

The observable representation exists only if there exist tensor inversion pairs $\mathcal{F}_i = P(\mathcal{O}_i|S_i)$, and $\mathcal{F}_i^{-1}$. This is equivalent to requiring that the rank of the matricized version of $\mathcal{F}_i$ (rows corresponds to modes $\mathcal{O}_i$ and column to modes S_i) has rank $\tau_i := S_H \times |S_i|$. Similarly. the matricized version of $\boldsymbol{\mathcal{P}}(\mathcal{O}_{-i}|S_i)$ also needs to have rank τ_i, so that the matricized version of $\boldsymbol{\mathcal{P}}(\mathcal{O}_i, \mathcal{O}_{-i})$ has rank τ_i and is invertible. Thus, it is required that #states$(\mathcal{O}_i) \geq$ #states(S_i). This can be achieved by either making $\mathcal{O}_i$ consist of a few high dimensional observations, or of many smaller dimensional ones. In the case when #states$(\mathcal{O}_i) >$ #states(S_i), we need to project $\mathcal{F}_i$ to a lower dimensional space using a tensor $\boldsymbol{\mathcal{U}}_i$ so that it can be inverted. In this case, we define $\mathcal{F}_i := \boldsymbol{\mathcal{P}}(\mathcal{O}_i|S_i) \times_{\mathcal{O}_i} \boldsymbol{\mathcal{U}}_i$. For example, following this through the computation for the leaf gives us that $\widetilde{\boldsymbol{\mathcal{P}}}(C_l) = \boldsymbol{\mathcal{P}}(\mathcal{O}_l, \mathcal{O}_{l-}) \times_{\mathcal{O}_{l-}} (\boldsymbol{\mathcal{P}}(\mathcal{O}_l, \mathcal{O}_{l-}) \times_{\mathcal{O}_l} \boldsymbol{\mathcal{U}}_l)^{-1}$. A good choice of $\boldsymbol{\mathcal{U}}_i$ can be obtained by performing a singular value decomposition of the matricized version of $\boldsymbol{\mathcal{P}}(\mathcal{O}_i, \mathcal{O}_{-i})$ (variables in $\mathcal{O}_i$ are arranged to rows and those in $\mathcal{O}_{-i}$ to columns).

For HMMs and latent trees, this rank condition can be expressed simply as requiring the condi-

Algorithm 6 Spectral learning algorithm for latent junction trees

```
In: Junction tree topology and $N$ i.i.d. samples $\left\{x_1^{(n)}, \ldots, x_O^{(n)}\right\}_{n=1}^N$
Out: Spectral parameters $\widehat{\boldsymbol{\mathcal{P}}}(C_r)$, $\widehat{\boldsymbol{\mathcal{P}}}(C_l)$ for all leaves, $\widehat{\boldsymbol{\mathcal{P}}}(C_i)$ for all non-root internal nodes
1: For each node $C_i$, perform a "thin" singular value decomposition of $\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_i, \mathcal{O}_{-i}) = \boldsymbol{\mathcal{U}}\boldsymbol{\Sigma}\boldsymbol{\mathcal{V}}^\top$; let $\widehat{\boldsymbol{\mathcal{U}}}_i = \boldsymbol{\mathcal{U}}(:, 1 : S_H^{\tau_i})$ be
   the the first $S_H^{\tau_i}$ principal left singular vectors where $\tau_i = |S_i|$.
2: Estimate $\widehat{\boldsymbol{\mathcal{P}}}(C_i)$ for the root, leaf and internal nodes
   $\widehat{\boldsymbol{\mathcal{P}}}(C_r) = \widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_{c_1(r)}, \mathcal{O}_{c_2(r)}, \mathcal{O}_{c_3(r)}) \times_{\mathcal{O}_{c_1(r)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_1(r)} \times_{\mathcal{O}_{c_2(r)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_2(r)} \times_{\mathcal{O}_{c_3(r)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_3(r)}$
   $\widehat{\boldsymbol{\mathcal{P}}}(C_l) = \widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_l, \mathcal{O}_{l-}) \times_{\mathcal{O}_{l-}} (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_l, \mathcal{O}_{l-}) \times_{\mathcal{O}_l} \widehat{\boldsymbol{\mathcal{U}}}_l)^{-1}$
   $\widehat{\boldsymbol{\mathcal{P}}}(C_i) = \widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_{c_1(i)}, \mathcal{O}_{c_2(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{c_1(i)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_1(i)} \times_{\mathcal{O}_{c_2(i)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_2(i)}$
       $\times_{\mathcal{O}_{-i}} (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_i} \widehat{\boldsymbol{\mathcal{U}}}_i)^{-1}$
```

Algorithm 7 Inference with spectral parameters for latent junction trees

```
In: Spectral parameters $\widehat{\boldsymbol{\mathcal{P}}}(C_r)$, $\widehat{\boldsymbol{\mathcal{P}}}(C_l)$ for all leaves, $\widehat{\boldsymbol{\mathcal{P}}}(C_i)$ for all non-root internal nodes and evidence $\{\bar{x}_1, \ldots, \bar{x}_O\}$
Out: Estimated marginal $\widehat{P}(\bar{x}_1, \ldots, \bar{x}_O)$
1: In reverse topological order, leaf and internal nodes send messages
   $\widehat{\boldsymbol{\mathcal{M}}}(S_l) = \widehat{\boldsymbol{\mathcal{P}}}(C_l)_{\mathcal{O}_l = \bar{o}_l}$
   $\widehat{\boldsymbol{\mathcal{M}}}(S_i) = \widehat{\boldsymbol{\mathcal{P}}}(C_i) \times_{\mathcal{O}_{c_1(i)}} \widehat{\boldsymbol{\mathcal{M}}}(S_{c_1(i)}) \times_{\mathcal{O}_{c_2(i)}} \widehat{\boldsymbol{\mathcal{M}}}(S_{c_2(i)})$
2: At the root, obtain $\widehat{P}(\bar{x}_1, \ldots, \bar{x}_O)$ by
   $\widehat{\boldsymbol{\mathcal{P}}}(C_r) \times_{\mathcal{O}_{c_1(r)}} \widehat{\boldsymbol{\mathcal{M}}}(S_{c_1(r)}) \times_{\mathcal{O}_{c_2(r)}} \widehat{\boldsymbol{\mathcal{M}}}(S_{c_2(r)}) \times_{\mathcal{O}_{c_3(r)}} \widehat{\boldsymbol{\mathcal{M}}}(S_{c_3(r)})$
```

tional probability tables of the underlying model to not be rank-deficient. However, junction trees encode more complex latent structures that introduce subtle considerations. A general characterization of the existence condition for observable representation with respect to the graph topology will be our future work.

The other practical aspects of spectral learning such as features, the expectation form, and linear systems as discussed in Chapter 4 also apply to our junction tree algorithm.

5.7 Sample Complexity

We analyze the sample complexity of Algorithm 6 and show that it depends on the junction tree topology and the spectral properties of the true model. Let d_i be the order of $\mathcal{P}(C_i)$ and e_i be the number of modes of $\mathcal{P}(C_i)$ that correspond to observed variables.

Theorem 2. *Let $\tau_i = S_H \times |S_i|$, $d_{\max} = \max_i d_i$, and $e_{\max} = \max_i e_i$. Then, for any $\epsilon > 0, 0 < \delta < 1$, if*

$$N \geq O\left(\left(\frac{4S_H^2}{3\beta^2}\right)^{d_{\max}} \frac{S_O^{e_{\max}} \ln \frac{|\mathbb{C}|}{\delta} |\mathbb{C}|^2}{\epsilon^2 \alpha^4}\right)$$

where $\sigma_\tau(*)$ returns the τ^{th} largest singular value and

$$\alpha = \min_i \ \sigma_{\tau_i}(P(\mathcal{O}_i, \mathcal{O}_{-i})), \quad \beta = \min_i \ \sigma_{\tau_i}(\mathcal{F}_i)$$

Then with probability $1 - \delta$,
$\sum_{x_1,\ldots,x_{|\mathcal{O}|}} \left|\widehat{P}(x_1, \ldots, x_{|\mathcal{O}|}) - P(x_1, \ldots, x_{|\mathcal{O}|})\right| \leq \epsilon$.

See the Appendix (5.10) for a proof. The result implies that the estimation problem depends exponentially on d_{max} and e_{max}, but note that $e_{max} \leq d_{max}$. Furthermore, d_{max} is always greater than or equal to the treewidth. Note the dependence on the singular values of certain probability tensors. In fully observed models, the accuracy of the learned parameters depends only on how close the empirical estimates of the factors are to the true factors. However, our spectral algorithm also depends on how close the inverses of these empirical estimates are to the true inverses, which depends on the spectral properties of the matrices (Stewart and Sun, 1990).

5.8 Experiments

We now evaluate our method on synthetic and real data and compare it with both standard EM (Dempster et al., 1977) and stepwise online EM (Liang and Klein, 2009). All methods were implemented in C++, and the matrix library Eigen (Guennebaud et al., 2010) was used for computing SVDs and solving linear systems. For all experiments, standard EM is given 5 random restarts. Online EM tends to be sensitive to the learning rate, so it is given one restart for each of 5 choices of the learning rate $\{0.6, 0.7, 0.8, 0.9, 1\}$ (the one with highest likelihood is selected). Convergence is determined by measuring the change in the log likelihood at iteration t (denoted by $f(t)$) over the average: $\frac{|f(t)-f(t-1)|}{\text{avg}(f(t), f(t-1))} \leq 10^{-4}$ (the same precision as used in (Murphy, 2005)).

For large sample sizes our method is almost two orders of magnitude faster than both EM and online EM. This is because EM is iterative and every iteration requires inference over all the training examples which can become expensive. On the other hand, the computational cost of our method is dominated by the SVD/linear system. Thus, it is primarily dependent only on the number of observed states and maximum tensor order, and can easily scale to larger sample sizes.

In terms of accuracy, we generally observe 3 distinct regions, low-sample size, mid-sample size, and large sample size. In the low sample size region, EM/online EM tend to overfit to the training data and our spectral algorithm usually performs better. In the mid-sample size region EM/online EM tend to perform better since they benefit from a smaller number of parameters. However, once a certain sample size is reached (the large sample size region), our spectral algorithm consistently outperforms EM/online EM which suffer from local minima and convergence issues.

5.8.1 Synthetic Evaluation

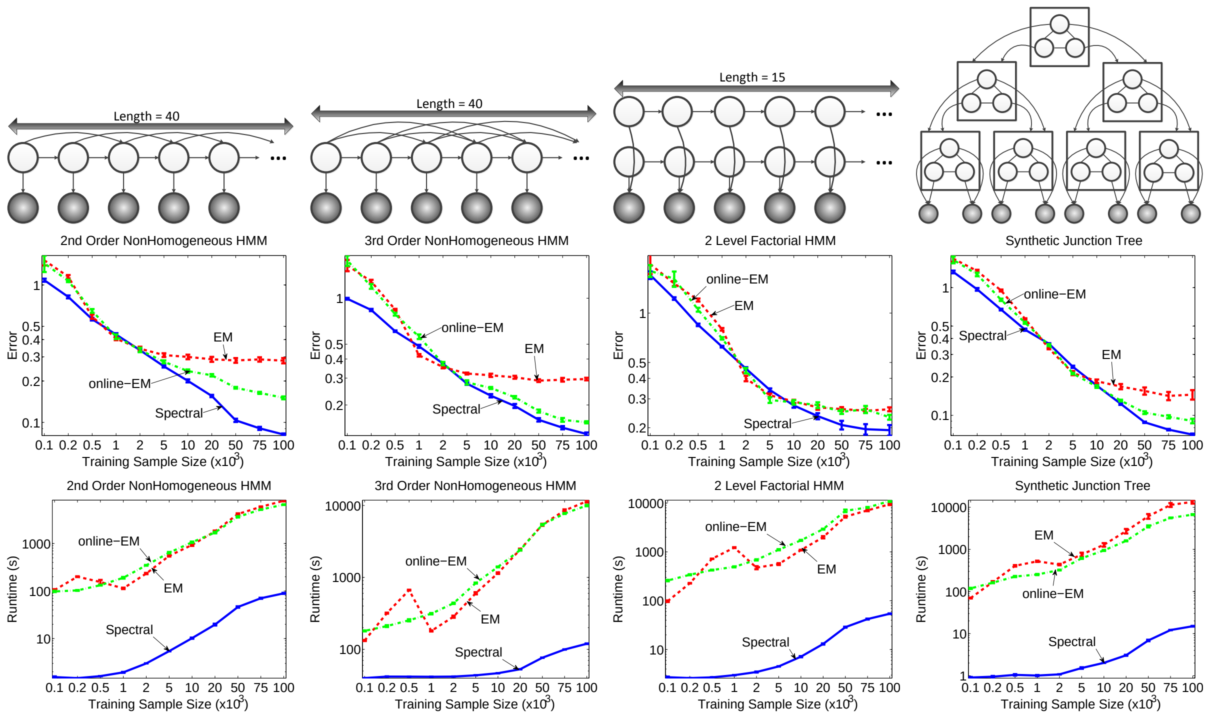
We first perform a synthetic evaluation. 4 different latent structures are used (see Figure 5.4): a second order nonhomogenous (NH) HMM, a third order NH HMM, a 2 level NH factorial HMM, and a complicated synthetic junction tree. The second/third order HMMs have $S_H = 2$ and $S_O = 4$, while the factorial HMM and synthetic junction tree have $S_H = 2$, and $S_O = 16$. For each latent structure, we generate 10 sets of model parameters, and then sample N training points and 1000 test points from each set, where N is varied from 100 to $100,000$. For evaluation, we measure the accuracy of joint estimation using $error = \frac{|\widehat{P}(x_1,...,x_O) - P(x_1,...,x_O)|}{P(x_1,...,x_O)}$. We also measure the training time of both methods.

Figure 5.4 shows the results. As discussed earlier, our algorithm is between one and two orders of magnitude faster than both EM and online EM for all the latent structures. EM is actually slower for very small sample sizes than for mid-range sample sizes because of overfitting. Also, in all cases, the spectral algorithm has the lowest error for large sample sizes. Moreover, critical sample size at which spectral overtakes EM/online EM is largely dependent on the number of parameters in the observable representation compared to that in the original parameterization of the model. In higher order/factorial HMM models, this increase is small, while in the synthetic junction tree it is larger.

5.8.2 Splice dataset

We next consider the task of determining splicing sites in DNA sequences (Asuncion and Newman, 2007). Each example consists of a DNA sequence of length 60, where each position in the sequence is either an A, T, C, or G. The goal is to classify whether the sequence is an Intron/Exon site, Exon/Intron site, or neither. During training, for each class a different second order nonhomogeneous HMM with $S_H = 2$ and $S_O = 4$ is trained. At test, the probability of the test sequence is computed for each model, and the one with the highest probability is selected (which we found to perform better than a homogeneous one).

Figure 5.5, shows our results, which are consistent with our synthetic evaluation. Spectral performs the best in low sample sizes, while EM/online EM perform a little better in the mid-sample size range. The dataset is not large enough to explore the large sample size regime. Moreover, we note that spectral algorithm is much faster for all the sample sizes.



(a) 2nd Order HMM (b) 3rd Order HMM (c) 2 Level Factorial HMM (d) Synthetic Junction Tree

Figure 5.4: Comparison of our spectral algorithm (blue) to EM (red) and online EM (green) for various latent structures. Both errors and runtimes in log scale.

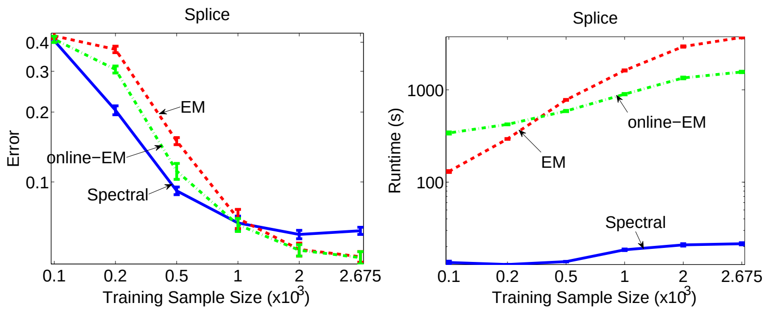


Figure 5.5: Results on Splice dataset

5.9 Conclusion

We have developed an alternative parameterization that allows fast, local minima free, and consistent parameter learning of latent junction trees that generalizes the algorithm we presented in Chapter 4. Our approach generalizes spectral algorithms to a much wider range of structures such as higher order, factorial, and semi-hidden Markov models. Unlike traditional nonconvex optimization formulations, spectral algorithms allow us to theoretically explore latent variable models in more depth. The spectral algorithm depends not only on the junction tree topology but also on the spectral properties of the parameters. Thus, two models with the same structure may pose different degrees of difficulty based on the underlying singular values. This is very different from learning fully observed junction trees, which is primarily dependent on only the topology/treewidth. Future directions include learning discriminative models and structure learning.

5.10 Appendix

We prove the sample complexity theorem.

Notation

For simplicity, the main text describes the algorithm in the context of a binary tree. However, in the sample complexity proof, we adopt a more general notation. Let C_i be a clique, and $C_{c_1(i)}$,...,$C_{c_{\alpha_i}(i)}$ denote its α_i children. Let $\mathbb{C}$ denote the set of all the cliques in the junction tree, and $|\mathbb{C}|$ denote its size. Define $d_{\max}$ to be maximum degree of a tensor in the observable representation and e_{max} to be maximum number of observed variables in any tensor. Furthermore, let $\tau_i = |S_H| \times |S_i|$ (i.e. the number of states associated with the separator).

We can now write the transformed representation for junction trees with more than 3 neighbors as:

Root:
$$\widetilde{\mathcal{P}}(C_i) = \mathcal{P}(C_i) \times_{S_{c_1(i)}} \mathcal{F}_{c_1(i)} \times \ldots \times_{S_{c_{\alpha_i}(i)}} \mathcal{F}_{c_{\alpha_i}(i)}$$
Internal nodes:
$$\widetilde{\mathcal{P}}(C_i) = \mathcal{P}(C_i) \times_{S_i} \mathcal{F}_i^{\dagger} \times_{S_{c_1(i)}} \mathcal{F}_{c_1(i)} \times \ldots \times_{S_{c_{\alpha_i}(i)}} \mathcal{F}_{c_{\alpha_i}(i)}$$
Leaf:
$$\widetilde{\mathcal{P}}(C_i) = \mathcal{P}(C_i) \times_{S_i} \mathcal{F}_i^{\dagger}$$

and the observable representation as:

$$\textbf{root: } \mathcal{P}(C_i) = \mathcal{P}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}) \times_{\mathcal{O}_{c_1(i)}} \mathcal{U}_{c_1(i)} \times \ldots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \mathcal{U}_{c_{\alpha_i}(i)}$$
$$\textbf{internal: } \mathcal{P}(C_i) = \mathcal{P}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_i} \mathcal{U}_i)^{\dagger} \times_{\mathcal{O}_{c_1(i)}} \mathcal{U}_{c_1(i)} \times \ldots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \mathcal{U}_{c_{\alpha_i}(i)}$$
$$\textbf{leaf: } \mathcal{P}(C_i) = \widehat{\mathcal{P}}(R_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\widehat{\mathcal{P}}(\mathcal{O}_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_i} \mathcal{U}_i)^{\dagger}$$

Sometimes when the multiplication indices are clear, we will omit it to make things simpler.

Rearranged version:
We will often find it convenient to rearrange the tensors into lower order tensors for the purpose of taking some norms (such as the spectral norm). We define $\boldsymbol{R}(\cdot)$ as the "rearranging" operation. For example, $\boldsymbol{R}(\widehat{\mathcal{P}}(\mathcal{O}_i, \mathcal{O}_{-i}))$ is the matricized version of $\widehat{\mathcal{P}}(\mathcal{O}_i, \mathcal{O}_{-i})$ with $\mathcal{O}_i$ being mapped to the rows and $\mathcal{O}_{-i}$ being mapped to the columns.
More generally, if $\mathcal{P}(C_i)$ (which has order d_i) then $\boldsymbol{R}(\mathcal{P}(C_i))$ is a rearrangement of of $\mathcal{P}(C_i)$ such that it has order equal to the number of neighbors of C_i. The set of modes of $\mathcal{P}(C_i)$ corresponding to a single separator of C_i get mapped to a single mode in $\boldsymbol{R}(\mathcal{P}(C_i))$. We let $\boldsymbol{R}(S_{\alpha_i(j)})$ denote the mode

that $S_{\alpha_i(j)}$ maps to in $\boldsymbol{R}(\boldsymbol{\mathcal{P}}(C_i))$.

In the case of the root, $\boldsymbol{R}(\boldsymbol{\mathcal{P}}(C_i))$ rearranges $\boldsymbol{\mathcal{P}}(C_i)$ into a tensor of order α_i. For other clique nodes, $\boldsymbol{R}(\boldsymbol{\mathcal{P}}(C_i))$ rearranges $\boldsymbol{\mathcal{P}}(C_i)$ into a tensor of order $\alpha_i + 1$.

Example: In Figure 5.2 $\boldsymbol{\mathcal{P}}(C_{BCDE}) = \boldsymbol{\mathcal{P}}(\oslash_2 B, D | \oslash_2 C, E)$. C_{BCDE} has 3 neighbors and so $\boldsymbol{R}(\boldsymbol{\mathcal{P}}(C_{BCDE}))$ is of order 3 where each of the modes correspond to $\{B, C\}$, $\{B, D\}$, $\{C, E\}$ respectively.

This rearrangement can be applied to the other quantities. For example, $\boldsymbol{R}(\boldsymbol{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}))$ is a rearranging of $\boldsymbol{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)})$ into a tensor of order α_i where the $\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}$ correspond to one mode each. $\boldsymbol{R}(\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_i, \mathcal{O}_{-i}))$ is the matricized version of $\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_i, \mathcal{O}_{-i})$ with $\mathcal{O}_i$ being mapped to the rows and $\mathcal{O}_{-i}$ being mapped to the columns. Similarly, $\boldsymbol{R}(\boldsymbol{\mathcal{F}}_i)$ is the matricized version of $\boldsymbol{\mathcal{F}}_i$ and $\boldsymbol{R}(\boldsymbol{\mathcal{U}}_i)$ is the matricized version of $\boldsymbol{\mathcal{U}}_i$.

Thus, we can define the rearranged quantities:

Root:

$$\boldsymbol{R}(\widetilde{\mathcal{P}}(C_i)) = \boldsymbol{R}(\boldsymbol{\mathcal{P}}(C_i)) \times_{\boldsymbol{R}(S_{c_1(i)})} \boldsymbol{R}(\boldsymbol{\mathcal{F}}_{c_1(i)}) \times \ldots \times_{\boldsymbol{R}(S_{c_{\alpha_i}(i)})} \boldsymbol{R}(\boldsymbol{\mathcal{F}}_{c_{\alpha_i}(i)})$$

$$\boldsymbol{R}(\widetilde{\mathcal{P}}(C_i)) = \boldsymbol{R}(\boldsymbol{\mathcal{P}}(C_i)) \times_{\boldsymbol{R}(S_i)} \boldsymbol{R}(\boldsymbol{\mathcal{F}}_i^{\dagger}) \times_{\boldsymbol{R}(S_{c_1(i)})} \boldsymbol{\mathcal{F}}_{c_1(i)} \times \ldots \times_{\boldsymbol{R}(S_{c_{\alpha_i}(i)})} \boldsymbol{R}(\boldsymbol{\mathcal{F}}_{\alpha_i})$$

Leaf:

$$\boldsymbol{R}(\widetilde{\mathcal{P}}(C_i)) = \boldsymbol{R}(\boldsymbol{\mathcal{P}}(C_i)) \times_{\boldsymbol{R}(S_i)} \boldsymbol{R}(\boldsymbol{\mathcal{F}}_i^{\dagger})$$

and the observable representation as:

$$\begin{aligned}
\textbf{root: } R(\boldsymbol{\mathcal{P}}(C_i)) &= \boldsymbol{R}(\boldsymbol{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)})) \times_{\boldsymbol{R}(S_{c_1(i)})} \boldsymbol{R}(\boldsymbol{\mathcal{U}}_{c_1(i)}) \times \ldots \times_{\boldsymbol{R}(S_{c_{\alpha_i}(i)})} \boldsymbol{R}(\boldsymbol{\mathcal{U}}_{c_{\alpha_i}(i)}) \\
\textbf{internal: } R(\boldsymbol{\mathcal{P}}(C_i)) &= \boldsymbol{R}(\boldsymbol{\mathcal{P}}(\mathcal{O}_{S_{c_1(i)}}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i})) \times_{\boldsymbol{R}(S_{-i})} \boldsymbol{R}((\boldsymbol{\mathcal{P}}(\mathcal{O}_i, \mathcal{O}_{-i}) \times_{S_i} \boldsymbol{\mathcal{U}}_i)^{\dagger}) \\
&\quad \times_{\boldsymbol{R}(S_{c_1(i)})} \boldsymbol{R}(\boldsymbol{\mathcal{U}}_{c_1(i)}) \times \ldots \times_{\boldsymbol{R}(S_{c_{\alpha_i}(i)})} \boldsymbol{R}(\boldsymbol{\mathcal{U}}_{c_{\alpha_i}(i)}) \\
\textbf{leaf: } R(\boldsymbol{\mathcal{P}}(C_i)) &= \boldsymbol{R}(\widehat{\boldsymbol{\mathcal{P}}}(R_i, \mathcal{O}_{-i})) \times_{\boldsymbol{R}(\mathcal{O}_{-i})} \boldsymbol{R}((\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_i} \boldsymbol{\mathcal{U}}_i)^{\dagger})
\end{aligned} \tag{5.23}$$

Furthermore define the following:

$$\alpha = \min_{C_i \in \mathbb{C}} \sigma_{\tau_i}(\mathbb{P}[\mathcal{O}_i, \mathcal{O}_{-i}]) \tag{5.24}$$

$$\beta = \min_{C_i \in \mathbb{C}} \sigma_{\tau_i}(\boldsymbol{\mathcal{F}}_i) \tag{5.25}$$

The proof is based on the technique of HKZ (Hsu et al., 2009) but has differences due to the junction tree topology and higher order tensors.

Tensor Norms

We briefly define several tensor norms that are a generalization of matrix norms. For more information about matrix norms see (Horn and Johnson, 1990).

Frobenius Norm: Just like for matrices, the frobenius norm of a tensor of order N is defined as:

$$||\boldsymbol{T}||_F = \sqrt{\sum_{c_1(i),...,i_N} \boldsymbol{T}(c_1(i),...,i_N)^2} \tag{5.26}$$

Elementwise One Norm: Similarly, the elementwise one norm of a tensor of order N is defined as:

$$||\boldsymbol{T}||_1 = \sum_{c_1(i),...,i_N} |\boldsymbol{T}(c_1(i),...,i_N)| \tag{5.27}$$

Spectral Norm: For tensors of order N the spectral norm (Nguyen et al., 2010) can be defined as as

$$|||\boldsymbol{T}|||_2 = \sup_{v_i \, s.t. \, ||v_i||_2 \leq 1 \, \forall 1 \leq i \leq N} \boldsymbol{T} \times_N \boldsymbol{v}_N, ..., \times_2 \boldsymbol{v}_2 \times_1 \boldsymbol{v}_1 \tag{5.28}$$

In our case, we will find it more convenient to use the rearranged spectral norm, which we define as:

$$|||\boldsymbol{T}|||_{2R} = |||\boldsymbol{R}(\boldsymbol{T})|||_2 \tag{5.29}$$

where $\boldsymbol{R}(\cdot)$ was defined in the previous section.

Induced One Norm: For matrices the induced one norm is defined as the max column sum of the matrix: $|||\boldsymbol{M}|||_1 = \sup_{v \, s.t. \, ||v||_1 \leq 1} ||\boldsymbol{M}\boldsymbol{v}||_1$. We can generalize this to be the max slice sum of a tensor on a tensor where some modes are fixed and others are summed over. Let σ denote the modes that will be maxed over. Then:

$$|||\boldsymbol{T}|||_1^{\sigma} = \sup_{v_i \, s.t. \, ||v_i||_1 \leq 1, \, \forall 1 \leq i \leq |\sigma|} ||\boldsymbol{T} \times_{\sigma_1} \boldsymbol{v}_1, ..., \times_{\sigma_{|\sigma|}} \boldsymbol{v}_{|\sigma|}||_1 \tag{5.30}$$

In the Appendix, we prove several lemmas regarding these tensor norms.

Concentration Bounds

$$\epsilon(\mathcal{O}_1, ..., \mathcal{O}_d) = \left\| \widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_1, ..., \mathcal{O}_d) - \boldsymbol{\mathcal{P}}(\mathcal{O}_1, ..., \mathcal{O}_d) \right\|_F \tag{5.31}$$

$$\epsilon(\mathcal{O}_1, ..., \mathcal{O}_{d-e}, \boldsymbol{o}_{d-e+1}, ..., \boldsymbol{o}_d) = \left\| \widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_1, ..., \mathcal{O}_{d-e}, \boldsymbol{o}_{d-e+1}, ..., \boldsymbol{o}_d) - \boldsymbol{\mathcal{P}}(\mathcal{O}_1, ..., \mathcal{O}_{d-e}, \boldsymbol{o}_{d-e+1}, ..., \boldsymbol{o}_d) \right\|_F \tag{5.32}$$

$1, ..., d-e$ denote the $d-e$ non-evidence variables while $d-e+1, ..., d$ denote the e evidence variables. d indicates the total number of modes of the tensor. As the number of samples N gets large, we expect these quantities to be small.

Lemma 6 (variant of HKZ (Hsu et al., 2009))**.** *If the algorithm independently samples N of the observa-*

tions, then with probability at least $1-\delta$*.*

$$\epsilon(\mathscr{O}_1, ..., \mathscr{O}_d) \leq \sqrt{\frac{1}{N}\ln\frac{2|\mathbb{C}|}{\delta}} + \sqrt{\frac{1}{N}} \tag{5.33}$$

$$\sum_{\boldsymbol{o}_{d-e+1},...,\boldsymbol{o}_d} \epsilon(\mathscr{O}_1, ..., \mathscr{O}_{d-e}, \boldsymbol{o}_{d-e+1}, ..., \boldsymbol{o}_d) \leq \sqrt{\frac{S_O^{e_{\max}}}{N}\ln\frac{2|\mathbb{C}|}{\delta}} + \sqrt{\frac{S_O^{e_{\max}}}{N}} \tag{5.34}$$

for ***all*** *tuples* $(\mathscr{O}_1, ..., \mathscr{O}_d)$ *that are used in the spectral algorithm.*

The proof is the same as that of HKZ (Hsu et al., 2009) except the union bound is taken over $2|\mathbb{C}|$ instead of 3 (since each transformed quantity in the spectral algorithm is composed of at most two such terms). The last bound can be made tighter, identical to HKZ, but for simplicity we do not pursue that approach here.

Singular Value Bounds

Basically this is the generalized version of Lemma 9 in HKZ (Hsu et al., 2009), which is stated below for completeness:

Lemma 7 (variant of HKZ (Hsu et al., 2009))**.** *Suppose* $\epsilon(\mathscr{O}_i, \mathscr{O}_{-i}) \leq \varepsilon \times \sigma_{\tau_i}(P(\mathscr{O}_i, \mathscr{O}_{-i}))$ *for some* $\varepsilon < 1/2$*. Let* $\varepsilon_0 = \epsilon(\mathscr{O}_i, \mathscr{O}_{-i})^2/((1-\varepsilon)\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}))^2$*. Then:*

1. $\varepsilon_0 < 1$
2. $\sigma_{\tau_i}(\widehat{\boldsymbol{\mathcal{P}}}(\mathscr{O}_i, \mathscr{O}_{-i}) \times_{\mathscr{O}_i} \widehat{\boldsymbol{\mathcal{U}}}_i) \geq (1-\varepsilon_0)\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}))$
3. $\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}) \times_{\mathscr{O}_i} \widehat{\boldsymbol{\mathcal{U}}}_i) \geq \sqrt{1-\varepsilon_0}\,\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}))$
4. $\sigma_{\tau_i}(\widehat{\boldsymbol{\mathcal{P}}}(\mathscr{O}_i|S_i) \times_{\mathscr{O}_i} \widehat{\boldsymbol{\mathcal{U}}}_i) \geq \sqrt{1-\varepsilon_0}\,\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i|S_i)))$

It follows that if $\epsilon(\mathscr{O}_i, \mathscr{O}_{-i}) \leq \sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}))/3$ then this implies that $\varepsilon_0 \leq \frac{1/9}{4/9} = \frac{1}{4}$. It then follows that,

1. $\sigma_{\tau_i}(\widehat{\boldsymbol{\mathcal{P}}}(\mathscr{O}_i, \mathscr{O}_{-i}) \times_{\mathscr{O}_i} \widehat{\boldsymbol{\mathcal{U}}}_i) \geq \frac{3}{4}\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}))$
2. $\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}) \times_{\mathscr{O}_i} \widehat{\boldsymbol{\mathcal{U}}}_i) \geq \frac{\sqrt{3}}{2}\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i, \mathscr{O}_{-i}))$
3. $\sigma_{\tau_i}(\widehat{\boldsymbol{\mathcal{P}}}(\mathscr{O}_i|S_i) \times_{\mathscr{O}_i} \widehat{\boldsymbol{\mathcal{U}}}_i) \geq \frac{\sqrt{3}}{2}\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}(\mathscr{O}_i|S_i))$

Bounding the Transformed Quantities

Define,

1. **root:** $\breve{\mathcal{P}}(C_i) := P(\mathcal{O}_{c_1(i)}, \dots, \mathcal{O}_{c_{\alpha_i}(i)}) \times_{\mathcal{O}_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \dots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}$
2. **leaf**: $\breve{\mathcal{P}}_{r_i}(C_i) = P(R_i = r_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger}$
3. **internal:** $\breve{\mathcal{P}}(C_i) = P(\mathcal{O}_{S_{c_1(i)}}, \dots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} \times_{\mathcal{O}_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \dots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}$

(which are the observable quantities with the true probabilities, but empirical $\widehat{\mathcal{U}}$'s). Similarly, we can define $\breve{\mathcal{F}}_i = \mathcal{P}[\mathcal{O}_i|S_i] \times_{\mathcal{O}_i} \widehat{\mathcal{U}}_i$. We have also abbreviated $\mathcal{P}(\mathcal{O}_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_i} \widehat{\mathcal{U}}_i$ with $\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i})$.

We seek to bound the following three quantities:

$$\delta_i^{root} := \|(\widehat{\mathcal{P}}(C_i) - \breve{\mathcal{P}}(C_i)) \times_{\mathcal{O}_{\alpha_i(1)}} \breve{\mathcal{F}}^{-1}_{c_1(i)}, \dots, \times_{\mathcal{O}_{\alpha_i(\alpha_i)}} \breve{\mathcal{F}}^{-1}_{c_{\alpha_i}(i)}\|_1 \tag{5.35}$$

$$\delta_i^{internal} := \left\| \left| (\widehat{\mathcal{P}}(C_i) - \breve{\mathcal{P}}(C_i)) \times_{\mathcal{O}_{-i}} \breve{\mathcal{F}}_i \times_{\mathcal{O}_{c_1(i)}} \breve{\mathcal{F}}_i^{-1}, \dots, \times_{\mathcal{O}_{\alpha_i(\alpha_i)}} \breve{\mathcal{F}}^{-1}_{c_{\alpha_i}(i)} \right| \right\|_1^{S_i} \tag{5.36}$$

$$\xi_i^{leaf} := \sum_{r_i} \|(\widehat{\mathcal{P}}_{r_i}(C_i) - \breve{\mathcal{P}}_{r_i}(C_i)) \times_{\mathcal{O}_{-i}} \breve{\mathcal{F}}_i\|_1 \tag{5.37}$$

Lemma 8. *If $\epsilon(\mathcal{O}_i, \mathcal{O}_{-i}) \le \sigma_{\tau_i}(P(\mathcal{O}_i, \mathcal{O}_{-i}))/3$ then*

$$\delta_i^{root} \le \frac{2^{d_{\max}} S_H^{d_{\max}} \epsilon(\mathcal{O}_{c_1(i)}, \dots, \mathcal{O}_{c_{\alpha_i}(i)})}{3^{d/2}\beta^d} \tag{5.38}$$

$$\delta_i^{internal} \le \frac{2^{d_{\max}+3} S_H^{d_{\max}}}{3\sqrt{3}^d \beta^d} \left(\frac{\epsilon(\mathcal{O}_{c_1(i)}, \dots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i})}{\sigma_{\tau_i}(P(\mathcal{O}_i, \mathcal{O}_{-i}))} + \frac{\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{(\sigma_{\tau_i}(P(\mathcal{O}_i, \mathcal{O}_{-i})))^2} \right) \tag{5.39}$$

$$\xi_i^{leaf} \le \frac{8 S_H^{d_{\max}}}{3} \left(\frac{\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{\sigma_{\tau_i}(P(\mathcal{O}_i, \mathcal{O}_{-i}))^2} + \frac{\sum_{r_i} \epsilon(R_i = r_i, \mathcal{O}_{-i})}{\sigma_{\tau_i}(\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))} \right) \tag{5.40}$$

We define $\triangle = \max(\delta_i^{root}, \delta_i^{internal}, \xi_i^{leaf})$ (over all i).

Proof.

Case δ_i^{root} :
For the root, $\widehat{\mathcal{P}}(C_i) = \widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \dots, \mathcal{O}_{c_{\alpha_i}(i)}) \times_{O_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \dots \times_{O_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}$ and similarly $\breve{\mathcal{P}}(C_i) = \mathcal{P}(\mathcal{O}_{c_1(i)}, \dots, \mathcal{O}_{c_{\alpha_i}(i)}) \times_{O_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \dots \times_{O_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}$.

$$\begin{aligned}
\delta^{root} &= \Big\|\Big(\widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)}) \times_{O_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{O_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)} \\
&\quad - \mathcal{P}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)}) \times_{O_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{O_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\Big) \times_{O_{\alpha_i(1)}} \breve{\mathcal{F}}^{-1}_{c_1(i)}, \ldots., \times_{O_{\alpha_i(\alpha_i)}} \breve{\mathcal{F}}^{-1}_{\alpha_i(1)}\Big\|_1 \\
&\le S_H^{d_{\max}} \Big|\Big|\Big|(\widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)}) - \mathcal{P}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)})) \times_{O_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{O_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\Big|\Big|\Big|_{2R} \\
&\quad \times \Big|\Big|\Big|\breve{\mathcal{F}}^{-1}_{c_1(i)}\Big|\Big|\Big|_{2R} \ldots. \Big|\Big|\Big|\breve{\mathcal{F}}_{c_1(i)})^{-1}\Big|\Big|\Big|_{2R} \\
&\le \frac{S_H^{d_{\max}}}{\prod_{j=1}^{\alpha_i} \sigma_{\tau_{\alpha_i(j)}}(\breve{\mathcal{F}}_{\alpha_i(j)})} \Big|\Big|\Big|(\widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)}) - \mathcal{P}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)})) \times_{O_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{O_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\Big|\Big|\Big|_{2R} \\
&\le \frac{S_H^{d_{\max}}}{\prod_{j=1}^{\alpha_i} \sigma_{\tau_{\alpha_i(j)}}(\breve{\mathcal{F}}_{\alpha_i(j)})} \Big|\Big|\Big|(\widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)}) - \mathcal{P}(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)}))\Big|\Big|\Big|_{2R} \times \\
&\qquad\qquad \Big|\Big|\Big|\widehat{\mathcal{U}}_{c_1(i)}\Big|\Big|\Big|_{2R} \Big|\Big|\Big|\widehat{\mathcal{U}}_{\alpha_i(2)}\Big|\Big|\Big|_{2R} \cdots \Big|\Big|\Big|\widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\Big|\Big|\Big|_{2R} \\
&= \frac{S_H^{d_{\max}}}{\prod_{j=1}^{\alpha_i} \sigma_{\tau_{\alpha_i(j)}}(\breve{\mathcal{F}}_{\alpha_i(j)})} \epsilon(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)})
\end{aligned}$$

Between the first and second line we use Lemma 13 to convert from elementwise one norm to spectral norm, and Lemma 11 (submultiplicativity). Lemma 11 (submultiplicativity) is applied again to get to the second-to-last line. We also use the fact that $\Big|\Big|\Big|\widehat{\mathcal{U}}_{c_1(i)}\Big|\Big|\Big|_{2R} = 1$.

Thus, by Lemma 7

$$\delta_i^{root} \le \frac{2^{d_{max}} S_H^{d_{max}} \epsilon(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)})}{3^{d_{max}/2} \beta^{d_{max}}} \tag{5.41}$$

Case ξ_i^{leaf}:
We note that $\widehat{\mathcal{P}}_{r_i}(C_i) = \widehat{\mathcal{P}}(R_i = r_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger}$ and similarly $\breve{\mathcal{P}}_{r_i}(C_i) = \mathcal{P}(R_i = r_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger}$.

Again, we use Lemma 13 to convert from the one norm to the spectral norm, and Lemma 11 for submultiplicativity.

$$\begin{aligned}
\xi_i^{leaf} &= \sum_{r_i} \Big|\Big|\Big|(\widehat{\mathcal{P}}_{r_i}(C_i) - \breve{\mathcal{P}}_{r_i}(C_i)) \times_{\mathcal{O}_{-i}} \breve{\mathcal{F}}_i\Big|\Big|\Big|_1^{S_i} \\
&= \sum_{r_i} \Big|\Big|\Big|(\widehat{\mathcal{P}}_{r_i}(C_i) - \breve{\mathcal{P}}_{r_i}(C_i)) \times_{\mathcal{O}_{-i}} \widehat{\mathcal{U}}_i\Big|\Big|\Big|_1^{S_i} |||\mathcal{P}[\mathcal{O}_i|S_i]|||_1^{S_i} \\
&\le \sum_{r_i} S_H^{d_{\max}} \Big|\Big|\Big|(\widehat{\mathcal{P}}_{r_i}(C_i) - \breve{\mathcal{P}}_{r_i}(C_i))\Big|\Big|\Big|_{2R} \Big|\Big|\Big|\widehat{\mathcal{U}}_i\Big|\Big|\Big|_{2R} \\
&\le \sum_{r_i} S_H^{d_{\max}} \Big|\Big|\Big|(\widehat{\mathcal{P}}_{r_i}(C_i) - \breve{\mathcal{P}}_{r_i}(C_i))\Big|\Big|\Big|_{2R}
\end{aligned} \tag{5.42}$$

Note that $|||\boldsymbol{\mathcal{P}}[\mathcal{O}_i|S_i]|||_1^{S_i} = 1$, and $\left|\left|\left|\widehat{\boldsymbol{\mathcal{U}}}_i\right|\right|\right|_{2R} = 1$.

$$\begin{aligned}
\left|\left|\left|\widehat{\boldsymbol{\mathcal{P}}}_{r_i}(C_i) - \check{\boldsymbol{\mathcal{P}}}_{r_i}(C_i)\right|\right|\right|_{2R} &= \left|\left|\left|\widehat{\boldsymbol{\mathcal{P}}}(R_i = r_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\widehat{\boldsymbol{\mathcal{P}}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} - P(R_i = r_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger}\right|\right|\right|_{2R} \\
&\leq \left|\left|\left|\boldsymbol{\mathcal{P}}(R_i = r_i, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\widehat{\boldsymbol{\mathcal{P}}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})^{\dagger} - \boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})^{\dagger})\right|\right|\right|_{2R} + \\
&\quad \left|\left|\left|(\widehat{\boldsymbol{\mathcal{P}}}(R_i = r_i, \mathcal{O}_{-i}) - \boldsymbol{\mathcal{P}}(R_i = r_i, \mathcal{O}_{-i})) \times_{\mathcal{O}_{-i}} \boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})^{\dagger}\right|\right|\right|_{2R} \\
&\leq |||\boldsymbol{\mathcal{P}}(R_i = r_i, \mathcal{O}_{-i})|||_{2R} \left|\left|\left|(\widehat{\boldsymbol{\mathcal{P}}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})^{\dagger} - \boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})^{\dagger})\right|\right|\right|_{2R} + \\
&\quad \left|\left|\left|\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})^{\dagger}\right|\right|\right|_{2R} \left|\left|\left|(\widehat{\boldsymbol{\mathcal{P}}}(R_i = r_i, \mathcal{O}_{-i}) - \boldsymbol{\mathcal{P}}(R_i = r_i, \mathcal{O}_{-i}))\right|\right|\right|_{2R} \\
&\leq P(R_i = r_i)\frac{1+\sqrt{5}}{2}\frac{\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{\min(\sigma_{\tau_i}(\widehat{\boldsymbol{\mathcal{P}}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})), \sigma_{\tau_{ij}}(\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})))^2} \\
&\quad + \frac{\epsilon(R_i = r_i, \mathcal{O}_{-i})}{\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))}
\end{aligned}$$

The last line follows from Eq. 5.55. We have also used the fact that $|||\boldsymbol{\mathcal{P}}(R_i = r_i, \mathcal{O}_{-i})|||_{2R} \leq ||\boldsymbol{\mathcal{P}}(R_i = r_i, \mathcal{O}_{-i})||_F \leq P(R_i = r)$ by Lemma 12. Thus, using Lemma 7,

$$\xi_i^{leaf} \leq \sum_{r_i} S_H^{d_{\max}} \left(P(R_i = r_i)\frac{1+\sqrt{5}}{2}\frac{\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{\min(\sigma_{\tau_i}(\widehat{\boldsymbol{\mathcal{P}}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})), \sigma_{\tau_i}(\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i})))^2} + \frac{\epsilon(R_i = r_i, \mathcal{O}_{-i})}{\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))} \right) \quad (5.43)$$

$$\begin{aligned}
\xi_i^{leaf} &\leq \sum_{r_i} S_H^{d_{max}} \left(\frac{1+\sqrt{5}}{2}\frac{16P(R_i = r_i)\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{9\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^2} + \frac{2\epsilon(R_i = r_i, \mathcal{O}_{-i})}{\sqrt{3}\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))} \right) \\
&\leq \frac{8S_H^{d_{max}}}{3} \left(\frac{\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{\sigma_{\tau_i}(P(\mathcal{O}_i, \mathcal{O}_{-i}))^2} + \frac{\sum_{r_i}\epsilon(R_i = r_i, \mathcal{O}_{-i})}{\sigma_{\tau_i}(\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))} \right) \qquad (5.44)
\end{aligned}$$

$\delta_i^{internal}$

$\check{\boldsymbol{\mathcal{P}}}(C_i) = \boldsymbol{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \dots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\boldsymbol{\mathcal{P}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} \times_{\mathcal{O}_{c_1(i)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_1(i)} \times \dots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_{\alpha_i}(i)}$.

Similarly, $\widehat{\boldsymbol{\mathcal{P}}}(C_i) = \widehat{\boldsymbol{\mathcal{P}}}(\mathcal{O}_{c_1(i)}, \dots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\widehat{\boldsymbol{\mathcal{P}}}_{\widehat{\boldsymbol{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} \times_{\mathcal{O}_{c_1(i)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_1(i)} \times \dots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\boldsymbol{\mathcal{U}}}_{c_{\alpha_i}(i)}$.

Again, we use Lemma 13 to convert from one norm to spectral norm and Lemma 11 for submultiplicativity.

$$\begin{aligned}
\delta_i^{internal} &= \left|\left|\left|(\widehat{\mathcal{P}}(C_i) - \breve{\mathcal{P}}(C_i)) \times_{\mathcal{O}_{-i}} \breve{\mathcal{F}}_i \times_{\mathcal{O}_{\alpha_i(1)}} \breve{\mathcal{F}}^{-1}_{\alpha_i(1)}, ..., \times_{\mathcal{O}_{\alpha_i(\alpha_i)}} \breve{\mathcal{F}}^{-1}_{c_{\alpha_i}(i)}\right|\right|\right|_1^{S_i} \\
&\le \left|\left|\left|(\widehat{\mathcal{P}}(C_i) - \breve{\mathcal{P}}(C_i)) \times_{\mathcal{O}_{-i}} \widehat{\mathcal{U}}_i \times_{\mathcal{O}_{\alpha_i(1)}} \breve{\mathcal{F}}^{-1}_{\alpha_i(1)}, ..., \times_{\mathcal{O}_{\alpha_i(\alpha_i)}} \breve{\mathcal{F}}^{-1}_{c_{\alpha_i}(i)}\right|\right|\right|_1^{S_i} |||\mathcal{P}[\mathcal{O}_i|S_i]|||_1^{S_i} \\
&\le S_H^{d_{max}} \left|\left|\left|(\widehat{\mathcal{P}}(C_i) - \breve{\mathcal{P}}(C_i))\right|\right|\right|_{2R} \left|\left|\left|\widehat{\mathcal{U}}_i\right|\right|\right|_{2R} \left|\left|\left|\breve{\mathcal{F}}^{-1}_{c_1(i)}\right|\right|\right|_{2R} \cdots \left|\left|\left|\breve{\mathcal{F}}^{-1}_{c_{\alpha_i}(i)}\right|\right|\right|_{2R} \\
&\le \frac{S_H^{d_{max}}}{\prod_{j=1}^{\alpha_i} \sigma_{\tau_{\alpha_i(j)}}(\breve{\mathcal{F}}_{\alpha_i(j)})} \left|\left|\left|(\widehat{\mathcal{P}}(C_i) - \breve{\mathcal{P}}(C_i))\right|\right|\right|_{2R} \qquad (5.45)
\end{aligned}$$

Note that $|||\mathcal{P}[\mathcal{O}_i|S_i]|||_1^{S_i} = 1$, and $\left|\left|\left|\widehat{\mathcal{U}}_i\right|\right|\right|_{2R} = 1$.

$$\begin{aligned}
&\left|\left|\left|\widehat{\mathcal{P}}(C_i) - \breve{\mathcal{P}}(C_i)\right|\right|\right|_{2R} \\
= &\left|\left|\left|\widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\widehat{\mathcal{P}_{\widehat{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} \times_{\mathcal{O}_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)} - \right.\right.\right. \\
&\quad \left.\left.\left. P(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} (\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} \times_{\mathcal{O}_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\right|\right|\right|_{2R} \\
\le &\left|\left|\left|(\widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) - P(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i})) \times_{\mathcal{O}_{-i}} \mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger}\right.\right.\right. \\
&\quad \left.\left.\left. \times_{\mathcal{O}_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\right|\right|\right|_{2R} \\
&\quad + \left|\left|\left|P(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) \times_{\mathcal{O}_{-i}} ((\widehat{\mathcal{P}_{\widehat{\mathcal{U}}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} - (\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger})\right.\right.\right. \\
&\quad \left.\left.\left. \times_{\mathcal{O}_{c_1(i)}} \widehat{\mathcal{U}}_{c_1(i)} \times \ldots \times_{\mathcal{O}_{c_{\alpha_i}(i)}} \widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\right|\right|\right|_{2R} \\
\le &\left|\left|\left|(\widehat{\mathcal{P}}(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}) - P(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}))\right|\right|\right|_{2R} |||\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger}|||_{2R} \left|\left|\left|\widehat{\mathcal{U}}_{c_1(i)}\right|\right|\right|_{2R} \cdots \left|\left|\left|\widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\right|\right|\right|_{2R} \\
&\quad + \left|\left|\left|P(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i})\right|\right|\right|_{2R} \left|\left|\left|\widehat{\mathcal{P}}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger} - \mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))^{\dagger}\right|\right|\right|_{2R} \left|\left|\left|\widehat{\mathcal{U}}_{c_1(i)}\right|\right|\right|_{2R} \cdots \left|\left|\left|\widehat{\mathcal{U}}_{c_{\alpha_i}(i)}\right|\right|\right|_{2R} \\
\le &\frac{\epsilon(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i})}{\sigma_{\tau_i}(\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))} + \frac{1+\sqrt{5}}{2} \frac{\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{\min(\sigma_{\tau_i}(\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i})), \sigma_{\tau_i}(\widehat{\mathcal{P}}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i})))^2}
\end{aligned}$$

The last line follows from Eq. 5.55. We have also used the fact that $\left|\left|\left|P(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i})\right|\right|\right|_{2R} \le ||P(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i}))||_F \le 1$ via Lemma 12. Using Lemma 7,

$$\delta_i^{internal} \le \frac{(2S_H)^{d_{max}}}{(\beta\sqrt{3})^{d_{max}}} \left(\frac{\epsilon(\mathcal{O}_{c_1(i)}, \ldots, \mathcal{O}_{c_{\alpha_i}(i)}, \mathcal{O}_{-i})}{\sigma_{\tau_i}(\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i}))} + \frac{1+\sqrt{5}}{2} \frac{\epsilon(\mathcal{O}_i, \mathcal{O}_{-i})}{\min(\sigma_{\tau_i}(\mathcal{P}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i})), \sigma_{\tau_i}(\widehat{\mathcal{P}}_{\widehat{\mathcal{U}}}(\mathcal{O}_i, \mathcal{O}_{-i})))^2} \right) \qquad (5.46)$$

$$\begin{aligned}\delta_i^{internal} &\leq \frac{(2S_H)^{d_{max}}}{(\beta\sqrt{3})^{d_{max}}}\left(\frac{2\epsilon(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)},\mathcal{O}_{-i})}{\sqrt{3}\sigma_{\tau_i}(P(\mathcal{O}_i,\mathcal{O}_{-i}))}+\frac{8\epsilon(\mathcal{O}_i,\mathcal{O}_{-i})}{3(\sigma_{\tau_i}(P(\mathcal{O}_i,\mathcal{O}_{-i})))^2}\right)\\ &\leq \frac{8(2S_H)^{d_{max}}}{3(\beta\sqrt{3})^{d_{max}}}\left(\frac{\epsilon(\mathcal{O}_{c_1(i)},\ldots,\mathcal{O}_{c_{\alpha_i}(i)},\mathcal{O}_{-i})}{\sigma_{\tau_i}(P(\mathcal{O}_i,\mathcal{O}_{-i}))}+\frac{\epsilon(\mathcal{O}_i,\mathcal{O}_{-i})}{(\sigma_{\tau_i}(P(\mathcal{O}_i,\mathcal{O}_{-i})))^2}\right) \qquad (5.47)\end{aligned}$$

□

Bounding the Propagation of Error

We now show all these errors propagate on the junction tree. For this section, assume the clique nodes are numbered $1, 2, ..., |\mathbb{C}|$ in breadth first order (such that 1 is the root). Moreover let $\Phi_{1:c}(C)$ be the transformed factors accumulated so far if we computed the joint probability from the root down (instead of the bottom up). For example,

$$\begin{aligned}\Phi_{1:1}(C) &= \mathcal{P}(C_1) & (5.48)\\ \Phi_{1:2}(C) &= \mathcal{P}(C_1)\times_{S_2}\mathcal{P}(C_2) & (5.49)\\ \Phi_{1:2}(C) &= \mathcal{P}(C_1)\times_{S_2}\mathcal{P}(C_2)\times_{S_3}\mathcal{P}(C_3) & (5.50)\end{aligned}$$

(Note how this is very computationally inefficient: the tensors get very large. However, it is useful for proving statistical properties). Then the modes of $\Phi_{1:c}$ can be partitioned into mode groups, $\mathcal{M}_1,...,\mathcal{M}_{d_c}$ (where each mode group consists of the variables on the corresponding separator edge). We now prove the following lemma,

Lemma 9. *Define* $\triangle = \max(\delta_i^{root}, \delta_i^{internal}, \xi_i^{leaf})$. *Then,*

$$\sum_{\boldsymbol{x}_{1:c}} \|(\breve{\Phi}_{1:c}(C) - \widehat{\Phi}_{1:c}(C)) \times_{\mathcal{M}_1} \breve{\mathcal{F}}_1^{-1}, ..., \times_{\mathcal{M}_{d_c}} \breve{\mathcal{F}}_{d_c}^{-1}\|_1 \leq (1+\triangle)^{c-1}\delta_i^{root} + (1+\triangle)^{c-1} - 1 \qquad (5.51)$$

$\boldsymbol{x}_{1:c}$ is all the observed variables in cliques 1 through c. Note that when $c = |\mathbb{C}|$ then this implies that $\widehat{\Phi}_{1:c}(C) = \widehat{\mathcal{P}}[x_1, ..., x_O]$ and thus,

$$\sum_{\boldsymbol{x}} |\widehat{P}(x_1,...,x_O) - P(x_1,...,x_O)| \leq (1+\triangle)^{|\mathbb{C}|-1}\delta_i^{root} + (1+\triangle)^{|\mathbb{C}|-1} - 1 \qquad (5.52)$$

Proof. The proof is by induction on c. The base case follows trivially from the definition of δ_i^{root}. For the induction step, assume the claim holds for $c \geq 1$. Then we prove it holds for $c + 1$.

$$
\begin{aligned}
& \sum_{\boldsymbol{x}_{1:c+1}} \|(\breve{\Phi}_{1:c+1}(\mathcal{C}) - \widehat{\Phi}_{1:c+1}(\mathcal{C})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 \\
= & \sum_{\boldsymbol{x}_{1:c+1}} \Big\|\Big(\breve{\Phi}_{1:c}(\mathcal{C}) \times (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1})) + (\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1})) + (\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1} \\
& \qquad \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\Big\|_1 \\
\leq & \sum_{\boldsymbol{x}_{1:c+1}} \|(\breve{\Phi}_{1:c}(\mathcal{C}) \times (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}))) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 + \\
& \quad \sum_{\boldsymbol{x}_{1:c+1}} \|((\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}))) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 + \\
& \quad \sum_{\boldsymbol{x}_{1:c+1}} \|((\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C}) \times \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1
\end{aligned}
$$

Now we consider two cases, when $c + 1$ is a leaf clique and when it is an internal clique.

Case 1: Internal Clique

Note here the summation over $\boldsymbol{x}_{1:c+1}$ is irrelevant since there is no evidence. We use Lemma 10 to break up the three terms:

The first term,

$$
\begin{aligned}
& \|(\breve{\Phi}_{1:c}(\mathcal{C}) \times (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}))) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 \\
\leq & \Big\|\Big\|(\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1})) \times_{\mathscr{O}_{-(c+1)}} \breve{\boldsymbol{\mathcal{F}}}_{c+1} \times_{\mathscr{O}_{\alpha_{c+1}(1)}} \breve{\boldsymbol{\mathcal{F}}}_{\alpha_{c+1}(1)}^{-1}, ..., \times_{\mathscr{O}_{\alpha_{c+1}(\gamma_{c+1})}} \breve{\boldsymbol{\mathcal{F}}}_{\alpha_{c+1}(\gamma_{c+1})}^{-1}\Big\|\Big\|_1^{S_i} \\
\times & \|\breve{\Phi}_{1:c}(\mathcal{C}) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_c}^{-1}\|_1 \\
\leq & \triangle \times 1
\end{aligned}
$$

The first term above is simply $\delta_i^{internal} \leq \triangle$ while the second equals one since it is a joint distribution.

Now for the second term,

$$
\begin{aligned}
& \|(\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times (\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}))) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 \\
\leq & \|(\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_c}^{-1}\|_1 \times \\
& \quad \Big\|\Big\|(\widehat{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1})) \times_{\mathscr{O}_{-(c+1)}} \breve{\boldsymbol{\mathcal{F}}}_{c+1} \times_{\mathscr{O}_{\alpha_{c+1}(1)}} \breve{\boldsymbol{\mathcal{F}}}_{\alpha_{c+1}(1)}^{-1}, ..., \times_{\mathscr{O}_{\alpha_{c+1}(\gamma_{c+1})}} \breve{\boldsymbol{\mathcal{F}}}_{\alpha_{c+1}(\gamma_{c+1})}^{-1}\Big\|\Big\|_1^{S_{c+1}} \\
\leq & ((1+\triangle)^{c-1}\delta^{root} + (1+\triangle)^{c-1} - 1) \times \delta_{c+1}^{internal} \qquad \text{(via induction hypothesis)} \\
\leq & ((1+\triangle)^{c-1}\delta^{root} + (1+\triangle)^{c-1} - 1) \times \triangle
\end{aligned}
$$

The third term,

$$\begin{aligned}
&\|((\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times \breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 \\
&\leq \|(\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_c}^{-1}\|_1 \times \\
&\left\|\left|\breve{\boldsymbol{\mathcal{P}}}(\mathcal{C}_{c+1}) \times_{\mathscr{O}_{-(c+1)}} \breve{\boldsymbol{\mathcal{F}}}_{c+1} \times_{\mathscr{O}_{\alpha_{c+1}(1)}} \breve{\boldsymbol{\mathcal{F}}}_{\alpha_{c+1}(1)}^{-1}, ..., \times_{\mathscr{O}_{\alpha_{c+1}(\gamma_{c+1})}} \breve{\boldsymbol{\mathcal{F}}}_{\alpha_{c+1}(\gamma_{c+1})}^{-1}\right|\right\|_1^{S_{c+1}} \\
&\leq ((1+\triangle)^{c-1}\delta^{root} + (1+\triangle)^{c-1} - 1) \times 1
\end{aligned}$$

Case 2: Leaf Clique

Again we use Lemma 10 to break up the three terms:

The first term,

$$\begin{aligned}
&\sum_{\boldsymbol{x}_{1:c+1}} \|(\breve{\Phi}_{1:c}(\mathcal{C}) \times (\widehat{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}))) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 \\
\leq\ &\sum_{\boldsymbol{x}_{1:c+1}} \|\breve{\Phi}_{1:c}(\mathcal{C}) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_c}^{-1}\|_1 \left\|\left|(\widehat{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1})) \times_{\mathscr{O}_{-(c+1)}} \breve{\boldsymbol{\mathcal{F}}}_{c+1}\right|\right\|_1^{S_{c+1}} \\
\leq\ &\sum_{\boldsymbol{x}_{1:c}} \|\breve{\Phi}_{1:c}(\mathcal{C}) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_c}^{-1}\|_1 \sum_{\boldsymbol{x}_{c+1}} \left\|\left|(\widehat{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1})) \times_{\mathscr{O}_{-(c+1)}} \breve{\boldsymbol{\mathcal{F}}}_{c+1}\right|\right\|_1^{S_{c+1}} \\
\leq\ &1 \times \triangle
\end{aligned}$$

The first term above equals 1 because it is a joint distribution and the second is the bound on the transformed quantity we had proved earlier (since $r_i = \boldsymbol{x}_{c+1}$).

The second term,

$$\begin{aligned}
&\sum_{\boldsymbol{x}_{1:c+1}} \|(\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times (\widehat{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}))) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 \\
\leq\ &\sum_{\boldsymbol{x}_{1:c}} \|(\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_c}^{-1}\|_1 \sum_{\boldsymbol{x}_{c+1}} \left\|\left|(\widehat{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}) - \breve{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1})) \times_{\mathscr{O}_{-(c+1)}} \breve{\boldsymbol{\mathcal{F}}}_{c+1}\right|\right\|_1 \\
\leq\ &((1+\triangle)^{c-1}\delta^{root} + (1+\triangle)^{c-1} - 1) \times \xi_{c+1}^{leaf} \\
\leq\ &((1+\triangle)^{c-1}\delta^{root} + (1+\triangle)^{c-1} - 1) \times \triangle
\end{aligned}$$

The third term,

$$\begin{aligned}
&\sum_{\boldsymbol{x}_{1:c+1}} \|((\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C}) \times \breve{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_{c+1}}} \breve{\boldsymbol{\mathcal{F}}}_{d_{c+1}}^{-1}\|_1 \\
&\leq \sum_{\boldsymbol{x}_{1:c}} \|(\breve{\Phi}_{1:c}(\mathcal{C}) - \widehat{\Phi}_{1:c}(\mathcal{C})) \times_{\mathscr{M}_1} \breve{\boldsymbol{\mathcal{F}}}_1^{-1}, ..., \times_{\mathscr{M}_{d_c}} \breve{\boldsymbol{\mathcal{F}}}_{d_c}^{-1}\|_1 \sum_{\boldsymbol{x}_{c+1}} \left\|\left|\breve{\boldsymbol{\mathcal{P}}}_{r_i}(\mathcal{C}_{c+1}) \times_{\mathscr{O}_{-(c+1)}} \breve{\boldsymbol{\mathcal{F}}}_{c+1}\right|\right\|_1^{S_{c+1}} \\
&\leq ((1+\triangle)^{c-1}\delta^{root} + (1+\triangle)^{c-1} - 1) \times 1
\end{aligned}$$

Combining these terms proves the induction step. □

Putting it all together

We use the fact from HKZ (Hsu et al., 2009) that $(1 + a/t)^t \leq 1 + 2a$ for $a \leq 1/2$. Now $\triangle$ is the main source of error. We set $\triangle \leq O(\epsilon_{total}/|\mathbb{C}|)$.

Note that

$$\triangle \leq \frac{2^{d_{\max}+3} S_H^{d_{\max}}}{3\sqrt{3}^{d_{max}} \beta^{d_{\max}}} \left(\frac{\sum_{\boldsymbol{o}_{d-e+1},\ldots,\boldsymbol{o}_d} \epsilon(\mathscr{O}_1, \ldots., \mathscr{O}_{d-e}, \boldsymbol{o}_{d-e+1}, \ldots, \boldsymbol{o}_d)}{\alpha} + \frac{\sum_{\boldsymbol{o}_{d-e+1},\ldots,\boldsymbol{o}_d} \epsilon(\mathscr{O}_1, \ldots., \mathscr{O}_{d-e}, \boldsymbol{o}_{d-e+1}, \ldots, \boldsymbol{o}_d)}{\alpha^2} \right)$$

This gives,

$$\frac{2^{d_{\max}+3} S_H^{d_{\max}}}{3\sqrt{3}^{d_{max}} \beta^{d_{max}}} \left(\frac{\sum_{\boldsymbol{o}_{d-e+1},\ldots,\boldsymbol{o}_d} \epsilon(\mathscr{O}_1, \ldots., \mathscr{O}_{d-e}, \boldsymbol{o}_{d-e+1}, \ldots, \boldsymbol{o}_d)}{\alpha} + \frac{\sum_{\boldsymbol{o}_{d-e+1},\ldots,\boldsymbol{o}_d} \epsilon(\mathscr{O}_1, \ldots., \mathscr{O}_{d-e}, \boldsymbol{o}_{d-e+1}, \ldots, \boldsymbol{o}_d)}{\alpha^2} \right) \leq K\epsilon_{total}/|\mathbb{C}|$$

where K is some constant.

This implies,

$$\sum_{\boldsymbol{o}_{d-e+1},\ldots,\boldsymbol{o}_d} \epsilon(\mathscr{O}_1, \ldots., \mathscr{O}_{d-e}, \boldsymbol{o}_{d-e+1}, \ldots, \boldsymbol{o}_d) \leq K \frac{3^{d_{max}/2+1} \epsilon_{total} \alpha^2 \beta^{d_{max}}}{2^{d_{max}+3} S_H^{d_{max}} |\mathbb{C}|} \tag{5.53}$$

Now using the concentration bound (Lemma 6) will give,

$$K \frac{3^{d_{max}/2+1} \epsilon_{total} \alpha^2 \beta^{d_{max}}}{2^{d_{max}+3} S_H^{d_{max}} |\mathbb{C}|} \leq \sqrt{\frac{S_O^{e_{\max}}}{N} \ln \frac{2|\mathbb{C}|}{\delta}} + \sqrt{\frac{S_O^{e_{max}}}{N}}$$

Solving for N:

$$N \geq O\left(\left(\frac{4S_H^2}{3\beta^2} \right)^{d_{max}} \frac{S_O^{e_{max}} \ln \frac{|\mathbb{C}|}{\delta} |\mathbb{C}|^2}{\epsilon_{total}^2 \alpha^4} \right) \tag{5.54}$$

and this completes the proof.

Matrix Perturbation Bounds

This is Theorem 3.8 from pg. 143 in Stewart and Sun, 1990 (Stewart and Sun, 1990). Let $A \in \boldsymbol{R}^{m \times n}$, with $m \geq n$ and let $\tilde{A} = A + E$. Then

$$\left\| \widetilde{A}^{+} - A^{+} \right\|_2 \leq \frac{1+\sqrt{5}}{2} \max(\left\| A^{+} \right\|_2^2, \left\| \widetilde{A} \right\|_2^2) \, \|E\|_2 \tag{5.55}$$

Tensor Norm Bounds

For matrices it is true that $\|\boldsymbol{M}\boldsymbol{v}\|_1 \leq |||\boldsymbol{M}|||_1 \, \|\boldsymbol{v}\|_1$. We prove the generalization to tensors.

Lemma 10. *Let $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ be tensors $\boldsymbol{\sigma}$ a set of (labeled) modes.*

$$\|\boldsymbol{T}_1 \times_\sigma \boldsymbol{T}_2\|_1 \leq |||\boldsymbol{T}_2|||_1^\sigma \, \|\boldsymbol{T}_1\|_1 \tag{5.56}$$

Proof.

$$\begin{aligned}
\|\boldsymbol{T}_1 \times_\sigma \boldsymbol{T}_2\|_1 &= \sum_{\boldsymbol{i}_{1:N}\setminus\sigma} \sum_{\boldsymbol{j}_{1:M\setminus\sigma}} \sum_{\boldsymbol{x}} \boldsymbol{T}_1(\boldsymbol{i}_{1:N} \setminus \boldsymbol{\sigma}, \boldsymbol{\sigma} = \boldsymbol{x}) \boldsymbol{T}_2(\boldsymbol{j}_{1:N} \setminus \boldsymbol{\sigma}, \boldsymbol{\sigma} = \boldsymbol{x}) \\
&= \sum_{\boldsymbol{i}_{1:N}\setminus\sigma} \sum_{\boldsymbol{x}} \sum_{\boldsymbol{j}_{1:M\setminus\sigma}} \boldsymbol{T}_1(\boldsymbol{i}_{1:N} \setminus \boldsymbol{\sigma}, \boldsymbol{\sigma} = \boldsymbol{x}) \boldsymbol{T}_2(\boldsymbol{j}_{1:N} \setminus \boldsymbol{\sigma}, \boldsymbol{\sigma} = \boldsymbol{x}) \\
&= \sum_{\boldsymbol{i}_{1:N}\setminus\sigma} \sum_{\boldsymbol{\sigma}} \boldsymbol{T}_1(\boldsymbol{i}_{1:N} \setminus \boldsymbol{\sigma}, \boldsymbol{\sigma} = \boldsymbol{x}) \sum_{\boldsymbol{j}_{1:M\setminus\sigma}} \boldsymbol{T}_2(\boldsymbol{j}_{1:N} \setminus \boldsymbol{\sigma}, \boldsymbol{\sigma} = \boldsymbol{x}) \\
&\leq \max_{\boldsymbol{x}} \left(\sum_{\boldsymbol{j}_{1:M\setminus\sigma}} \boldsymbol{T}_2(\boldsymbol{j}_{1:N} \setminus \boldsymbol{\sigma}, \boldsymbol{\sigma} = \boldsymbol{x}) \right) \|\boldsymbol{T}_1\|_1 \\
&= |||\boldsymbol{T}_2|||_1^\sigma \, \|\boldsymbol{T}_1\|_1
\end{aligned}$$

□

We prove a restricted analog of the fact that spectral norm is submultiplicative for matrices i.e. $|||AB|||_2 \leq |||A|||_2 \, |||B|||_2$.

Lemma 11. *Let $\boldsymbol{T}$ be a tensor of order N and let $\boldsymbol{M}$ be a matrix. Then,*

$$|||\boldsymbol{T} \times_1 \boldsymbol{M}|||_2 \leq |||\boldsymbol{T}|||_2 \, |||\boldsymbol{M}|||_2 \tag{5.57}$$

Proof.

$$\begin{aligned}
|||\boldsymbol{T} \times_1 \boldsymbol{M}|||_2 &= \sup_{\boldsymbol{v}_m, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N} \sum_{c_1(i), \ldots, i_N, m} \boldsymbol{T}(c_1(i), c_2(i), \ldots, i_N) \boldsymbol{M}(c_1(i), m) \boldsymbol{v}_m(m) \boldsymbol{v}_{c_2(i)}(c_2(i)) \boldsymbol{v}_{i_3}(i_3) \ldots . \boldsymbol{v}_{i_n}(i_n) \\
&= \sup_{\boldsymbol{v}_m, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N} \sum_{c_2(i), \ldots, i_N} \sum_{c_1(i)} \boldsymbol{T}(c_1(i), c_2(i), \ldots, i_N) \sum_m \boldsymbol{M}(c_1(i), m) \boldsymbol{v}_m(m) \\
&\leq \sup_{\boldsymbol{v}_m, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N} \left(\sup_{c_1(i)} \left\| \sum_m \boldsymbol{M}(c_1(i), m) \boldsymbol{v}_m(m) \right\|_2 \right) \times \\
&\quad \sum_{c_1(i), \ldots, i_N} \boldsymbol{T}(c_1(i), c_2(i), \ldots, i_N) \frac{1}{\left\| \sum_m \boldsymbol{M}(c_1(i), m) \boldsymbol{v}_m(m) \right\|_2} \left(\sum_m \boldsymbol{M}(c_1(i), m) \boldsymbol{v}_m(m) \right) \boldsymbol{v}_{c_2(i)}(c_2(i)) \boldsymbol{v}_{i_3}(i_3) \ldots . \boldsymbol{v}_{i_n}(i_n) \\
&\leq |||\boldsymbol{M}|||_2 \, |||\boldsymbol{T}|||_2
\end{aligned}$$

□

Lemma 12. *Let $\boldsymbol{T}$ be a tensor of order N. Then,*

$$|||\boldsymbol{T}|||_2 \leq ||\boldsymbol{T} \times_1 v_1, ..., \times_{n-1} v_{n-1}||_F \leq ||T \times_1 v_1, ..., \times_{n-2} v_{n-2}||_F \leq ... \leq ||T||_F \tag{5.58}$$

Proof. It suffices to show that $\sup_{v\, s.t.\, ||v|| \leq 1} ||\boldsymbol{T} \times_1 \boldsymbol{v}||_F \leq ||\boldsymbol{T}||_F$. By submultiplicativity of the frobenius norm: $||\boldsymbol{T} \times_1 \boldsymbol{v}||_F \leq ||\boldsymbol{T}||_F \, ||\boldsymbol{v}||_F \leq ||\boldsymbol{T}||$, since $||\boldsymbol{v}||_F = ||\boldsymbol{v}||_2 \leq 1$. □

Lemma 13. *Let $\boldsymbol{T}$ be a tensor of order N, where each mode is of dimension k. Then,*

$$|||\boldsymbol{T}|||_1^{\sigma} \leq k^N \, |||\boldsymbol{T}|||_2 \tag{5.59}$$

For any $\boldsymbol{\sigma}$.

Proof. We simply prove this for $\boldsymbol{\sigma} = \emptyset$ (which corresponds to elementwise one norm) since $|||\boldsymbol{T}|||_1^{\sigma_1} \leq |||\boldsymbol{T}|||_1^{\sigma_2}$ if $\sigma_2 \subseteq \sigma_1$. Note that $||\boldsymbol{T}||_1 \leq k^N \max(|\boldsymbol{T}|)$ (where $\max(\boldsymbol{T})$ is the maximum element of $|\boldsymbol{T}|$). Similarly, $\max(|\boldsymbol{T}|) \leq |||\boldsymbol{T}|||_2$ which implies that $||\boldsymbol{T}||_1 \leq k^N \, |||\boldsymbol{T}|||_2$. □

Chapter 6

Nonparametric Latent Trees with Kernel Embeddings

In Chapter 4, we proposed linear-algebra based solutions to parameter learning in latent tree graphical models where the variables are discrete. However, in many real world scenarios, the variables under study take on continuous values. One example is demographics where variables like income, crime rate, and age are highly skewed and rarely follow bell curves. For example, Figure 6.1 shows the age distributions of two countries, Egypt, and Japan neither of which follow Gaussian distributions.

It is often difficult to apply traditional approaches to these datasets, since most algorithms for probabilistic modeling assume that the marginal/conditional distributions easily fit into a parametric family. Discretizing the data is often suboptimal, since it achieves a weaker convergence rate than kernel density estimation. For example for one-dimensional density estimation, the integrated square error of a histogram estimator decreases at a rate of $O(N^{-2/3})$ while the integrated square error of kernel density estimation decreases at a rate of $O(N^{-4/5})$(Shalizi, 2015). Unfortunately, kernel density estimation and other conventional nonparametric techniques severely suffer from the curse of dimensionality, making them impractical for modeling more than a handful of variables.

As a result, typical parameter learning and inference methods approximate the underlying distributions with Gaussians (or mixtures of Gaussians) so that traditional learning methods like

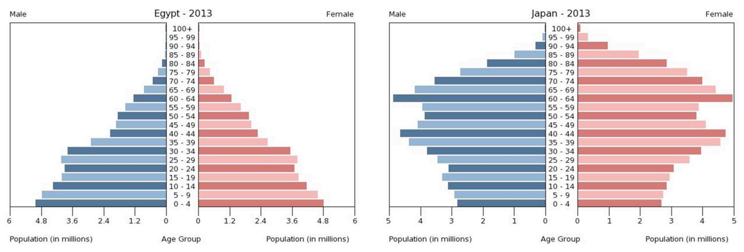


Figure 6.1: Examples of real world non-Gaussian distributions: Age distributions of Egypt (left) and Japan (right). (Photo Source : www.census.gov)

Expectation Maximization can be used (Dempster et al., 1977). Structure learning of latent trees is even more challenging and has largely been tackled by heuristics since the search space of structures is intractable. Examples from the phylogenetic community include maximum parisinomius and maximum likelihood methods (Semple and Steel, 2003). In the machine learning community,Zhang (2004) proposed a search heuristic for hierarchical latent class models by defining a series of local search operations and using EM to compute the likelihood of candidate structures. Harmeling and Williams (2011) proposed a greedy algorithm to learn binary trees by joining two nodes with a high mutual information and iteratively performing EM to compute the mutual information among newly added hidden nodes. Alternatively, Bayesian hierarchical clustering (Heller and Ghahramani, 2005) is an agglomerative clustering technique that merges clusters based on a statistical hypothesis test. In addition to lacking theoretical guarantees, these methods do not apply to non-Gaussian continuous distributions.

Contributions of this chapter: In this work, we propose a method for latent tree models with continuous and non-Gaussian observations based on the concept of kernel embedding of distributions (Smola et al., 2007). Kernel embeddings enable efficient representation of large multivariate continuous distributions avoiding the curse of dimensionality inherent in naive kernel density estimation. We present principled methods for parameter learning/inference as well as structure learning:

- **Parameter learning/inference**: We generalize the spectral algorithm from Chapter 4 to the continuous scenario using kernel embeddings. Previous work on nonparametric latent models was limited to hidden Markov models (Song et al., 2010a).
- **Structure learning**: We define a distance measure between variables based on the singular value decomposition of covariance operators. This allows us to generalize some of the distance based latent tree learning procedures from the phylogenetics community such as neighbor joining (Saitou and Nei, 1987) and recursive grouping methods (Choi et al., 2010) to the nonparametric setting. These distance based methods come with strong statistical guarantees, but have previously been unexplored in the nonparametric setting.

Outline: We first review multivariate kernel density estimation to motivate our approach. We then assume the latent structure is known and propose a spectral algorithm for parameter learning (and inference), followed by a structure learning algorithm. Lastly, we demonstrate the efficacy of our approach on synthetic and real datasets.

Prerequisites: This chapter assumes a general understanding of latent variable models as presented in 2.2, knowledge of Hilbert space embeddings as discussed in 2.3, the connection between latent variable models and low rank factorization in Chapter 3, and the tensor notation in 3.1.

6.1 Notation

So that the chapter is standalone, we briefly review notation although it is the same as that in Chapter 4. A latent tree model defines a joint probability distribution over a set of O observed variables $\mathscr{O} = \{X_1, \ldots, X_O\}$ and a set of H hidden variables $\mathscr{H} = \{X_{O+1}, \ldots, X_{O+H}\}$. The complete

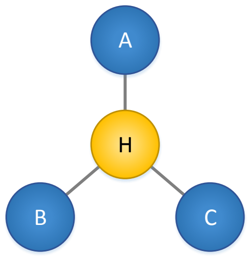


Figure 6.2: 4-node graphical model example

set of variables is denoted by $\mathscr{X} = \mathscr{O} \cup \mathscr{H}$. We assume that all hidden variables have dimension S_H (the observed variables are continuous). The joint distribution of $\mathscr{X}$ in a latent tree model is fully characterized by the following equation:

$$P(x_1, \ldots, x_{O+H}) = \prod_{s=1}^{O+H} P(x_s | x_{\pi(s)}). \tag{6.1}$$

where $X_{\pi(s)}$ denotes the parent of X_s. For simplicity, assume that all leaves are observed variables and all internal nodes are latent. For further notation, let X_{s^*} be some observed node in the subtree rooted at X_s. Let $c_j(s)$ denote the j^{th} child of node X_s where X_s has α_s children. For ease of exposition we will assume that all internal nodes have exactly 3 neighbors and that all internal nodes are latent while all leaf nodes are observed.

6.2 Kernel Density Estimation

To motivate our approach we first discuss how kernel density estimation(KDE) could be used to tackle our problem. KDE is a nonparametric way of fitting the density of continuous random variables with non-Gaussian statistical features such as multi-modality and skewness. Given a set of N *i.i.d.* samples $\left\{(x_1^{(n)}, \ldots, x_O^{(n)})\right\}_{i=1}^N$ from $p(X_1, \ldots, X_O)$, KDE estimates the density via

$$\widehat{P}(\bar{x}_1, \ldots, \bar{x}_O) \propto b_r := \mathbb{E}_{X_1,\ldots,X_O}\left[\prod_{j=1}^O k(\bar{x}_j, X_j)\right] \approx \frac{1}{N}\sum_{n=1}^N \prod_{j=1}^O k(\bar{x}_j, x_j^{(n)}) \tag{6.2}$$

where $k(x, x') = \langle \boldsymbol{\phi}_x, \boldsymbol{\phi}_{x'} \rangle$ is a kernel function and $\boldsymbol{\phi}_x \in \mathcal{F}$ where $\mathcal{F}$ is the underlying reproducing kernel hilbert space (RKHS). A commonly used kernel function, which we will focus on, is the Gaussian RBF kernel $k(x, x') = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\|x - x'\|^2/2\sigma^2)$. However, traditional KDE cannot model the underlying latent dependency structure among $X_1, ..., X_O$, and therefore scales poorly to beyond a handful of variables due to the curse of dimensionality [1].

However, incorporating latent structure into KDE could greatly increase its statistical feasibility in high dimensions. In the example in Figure 6.2 consider computing the marginal probability

[1] In particular the integrated square error of traditional KDE goes to zero at a rate of $O(N^{\frac{-4}{4+O}})$ (Casella and Berger, 2002)

$P(A = \bar{a}, B = \bar{b}, C = \bar{c})$:

$$\begin{aligned}\widehat{P}(\bar{a}, \bar{b}, \bar{c}) \propto b_r &:= \mathbb{E}_{A,B,C,H}\left[k(\bar{a}, A)k(\bar{b}, B)k(\bar{c}, C)\right] \\ &= \mathbb{E}_H[\mathbb{E}_{A,B,C|H}[k(\bar{a}, A)k(\bar{b}, B)k(\bar{c}, C)]] \\ &= \mathbb{E}_H[\mathbb{E}_{A|H}[k(\bar{a}, A)]\mathbb{E}_{B|H}[k(\bar{b}, B)]\mathbb{E}_{C|H}[k(\bar{c}, C)] \end{aligned} \tag{6.3}$$

where in the last line we have used the conditional independence implied by the graphical model structure. For general latent tree models, this immediately suggests a message passing scheme to compute the marginal probability of the evidence $(\bar{x}_1, ..., \bar{x}_O)$ at the leaf nodes:

- A leaf node X_s passes the following message to its parent: $\boldsymbol{m}_s(x_{\pi(s)}) = \mathbb{E}_{X_s|x_{\pi(s)}}[k(\bar{x}_s, X_s)]$.
- An internal latent node X_s aggregates incoming messages from its two children and then sends an outgoing message to its own parent $\boldsymbol{m}_s(x_{\pi(s)}) = \mathbb{E}_{X_s|x_{\pi(s)}}[\boldsymbol{m}_{c_1(s)}(X_s)\boldsymbol{m}_{c_2(s)}(X_s)]$.
- Finally, at the root node X_r, all incoming messages are multiplied together and the root variable is integrated out $b_r := \mathbb{E}_{\mathscr{O}\cup\mathscr{H}}\left[\prod_{j=1}^{O} k(\bar{x}_j, X_j)\right] = \mathbb{E}_{X_r}[\boldsymbol{m}_{c_1(r)}(X_r)m_{c_2(r)}(X_r)\boldsymbol{m}_{c_3(r)}(X_r)]$

Unfortunately, under the conventional KDE formulation it is difficult to define or manipulate these messages in practice. This motivates us to use Hilbert Space Embeddings, which provide us with the representation power we need (Smola et al., 2007).

6.3 Connection to Hilbert Space Embeddings

Consider Eq. 6.2 again. Products of kernels are also kernels, which allow us to rewrite $\prod_{j=1}^{O} k(\bar{x}_j, X_j)$ as a single inner product $\left\langle \otimes_{j=1}^{O} \boldsymbol{\phi}(\bar{x}_j), \otimes_{j=1}^{O} \boldsymbol{\phi}(X_j)\right\rangle$. Here $\otimes_{j=1}^{O}\star$ denotes the tensor product of O feature vectors which results in a rank-1 tensor of order O.

Now from the background in Chapter 2.3, $\boldsymbol{C}_{X_1,...,X_O} := \mathbb{E}_{\mathscr{O}}\left[\otimes_{j=1}^{O}\boldsymbol{\phi}(X_j)\right]$ is the cross-covariance operator that serves as the Hilbert space embedding of distribution $p(X_1, ..., X_O)$ with tensor features $\otimes_{j=1}^{O}\boldsymbol{\phi}(X_j)$. Moreover, b_r is simply evaluating this operator using $\boldsymbol{\phi}_{\bar{x}_1}, ..., \boldsymbol{\phi}_{\bar{x}_O}$:

$$b_r := \mathbb{E}_{X_1,...,X_O}\left[\prod_{j=1}^{O} k(\bar{x}_j, X_j)\right] = \boldsymbol{C}_{X_1,..,X_O} \times_1 \boldsymbol{\phi}_{\bar{x}_1} ... \times_O \boldsymbol{\phi}_{\bar{x}_O} \tag{6.4}$$

Thus the message passing scheme in Section 6.2 also serves to evaluate the cross covariance operator $\boldsymbol{C}_{X_1,..,X_O}$ at a particular value. Furthermore, following Song et al. (2010c), we can recast this message passing scheme in terms of (smaller) cross-covariance and conditional operators. Define the following parameters for each node in the latent tree:

- **root** (X_r): $\boldsymbol{C}_{rrr} := \mathbb{E}[\boldsymbol{\phi}_r \otimes \boldsymbol{\phi}_r \otimes \boldsymbol{\phi}_r]$ (embedding of X_r into the RKHS $\mathcal{F} \otimes \mathcal{F} \otimes \mathcal{F}$)
- **internal node** (X_s): $\boldsymbol{C}_{ss|\pi(s)} := \boldsymbol{C}_{ss\pi(s)}\boldsymbol{C}^{-1}_{\pi(s)\pi(s)} = \mathbb{E}[\boldsymbol{\phi}_s \otimes \boldsymbol{\phi}_s \otimes \boldsymbol{\phi}_{\pi(s)}]\mathbb{E}[\boldsymbol{\phi}_{\pi(s)} \otimes \boldsymbol{\phi}_{\pi(s)}]^{-1}$ (conditional embedding of $X_s|X_{\pi(s)}$ into the RKHS $\mathcal{F} \otimes \mathcal{F} \otimes \mathcal{F}$)

- **leaf** (X_s): $\boldsymbol{C}_{s|\pi(s)}$ (standard conditional embedding of $X_s|X_{\pi(s)}$)

Then, we can reformulate the message passing scheme. First at the leaf:

$$\begin{aligned}\boldsymbol{m}_s(x_{\pi(s)}) &= \mathbb{E}_{X_s|x_{\pi(s)}}[k(\bar{x}_s, X_s)] \\ &= \mathbb{E}_{X_s|x_{\pi(s)}}[\langle \boldsymbol{\phi}_{\bar{x}_s}, \boldsymbol{\phi}_{X_s}\rangle] \\ &= \langle \boldsymbol{\phi}_{\bar{x}_s}, \mathbb{E}_{X_s|x_{\pi(s)}}[\boldsymbol{\phi}_{X_s}]\rangle \\ &= \boldsymbol{C}_{s|\pi(s)} \times_s \boldsymbol{\phi}_{\bar{x}_s} \times_{\pi(s)} \boldsymbol{\phi}_{x_{\pi(s)}} \end{aligned} \tag{6.5}$$

where in the last line we have used Eq. 2.41. Thus, since $\boldsymbol{m}_s(x_{\pi(s)}) = \langle \boldsymbol{m}_s(\cdot), \boldsymbol{\phi}_{x_{\pi(s)}}\rangle$ via the reproducing property, we can conclude that $\boldsymbol{m}_s(\cdot) = \boldsymbol{C}_{s|\pi(s)} \times_s \boldsymbol{\phi}_{\bar{x}_s}$.

For the internal node,

$$\begin{aligned}\boldsymbol{m}_s(x_{\pi(s)}) &= \mathbb{E}_{X_s|x_{\pi(s)}}[\boldsymbol{m}_{c_1(s)}(X_s)\boldsymbol{m}_{c_2(s)}(X_s)] \\ &= \mathbb{E}_{X_s|x_{\pi(s)}}[\langle \boldsymbol{m}_{c_1(s)}, \boldsymbol{\phi}_{X_s}\rangle\langle \boldsymbol{m}_{c_2(s)}, \boldsymbol{\phi}_{X_s}\rangle] \\ &= \mathbb{E}_{X_s|x_{\pi(s)}}[\boldsymbol{\phi}_{X_s} \otimes \boldsymbol{\phi}_{X_s}] \times_s \boldsymbol{m}_{c_1(s)} \times_s \boldsymbol{m}_{c_2(s)} \\ &= \boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{m}_{c_1(s)} \times_s \boldsymbol{m}_{c_2(s)} \times_{\pi(s)} \boldsymbol{\phi}_{x_{\pi(s)}} \end{aligned} \tag{6.6}$$

Thus, $\boldsymbol{m}_s(\cdot) = \boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{m}_{c_1(s)} \times_s \boldsymbol{m}_{c_2(s)}$. Similarly for the root, we have that

$$b_r = \boldsymbol{C}_{rrr} \times_r \boldsymbol{m}_{c_1(r)} \times_r \boldsymbol{m}_{c_2(r)} \times_r \boldsymbol{m}_{c_3(r)} \tag{6.7}$$

Note this shows how hilbert space embeddings enable us to factorize a kernel density estimate according to a conditional independence structure analogous to conditional probability tables in discrete graphical models.

6.4 Deriving the Spectral Algorithm

The drawback of the representations in (6.5), (6.6) and (6.7) is that they require exact knowledge of conditional embedding operators associated with latent variables, but none of these are available in training. Next we will show that we can still make use of the tensor decomposition representation without the need for recovering the latent variables explicitly.

To derive our alternate factorization, we now insert invertible transformations $\boldsymbol{F}$. At the root,

$$\begin{aligned} b_r &= \boldsymbol{C}_{rrr} \times_r \boldsymbol{I} \times_r \boldsymbol{m}_{c_1(s)} \times_r \boldsymbol{I} \times_r \boldsymbol{m}_{c_2(s)} \times_r \boldsymbol{I} \times_r \boldsymbol{m}_{c_3(s)} \\ &= \boldsymbol{C}_{rrr} \times_r (\boldsymbol{F}_{c_1(s)} \times_{\omega_{c_1(s)}} \boldsymbol{F}^{\dagger}_{c_1(s)}) \times_r \boldsymbol{m}_{c_1(s)} \times_r (\boldsymbol{F}_{c_2(s)} \times_{\omega_{c_2(s)}} \boldsymbol{F}^{\dagger}_{c_2(s)}) \times_r \boldsymbol{m}_{c_2(s)} \times_r (\boldsymbol{F}_{c_3(s)} \times_{\omega_{c_3(s)}} \boldsymbol{F}^{\dagger}_{c_3(s)}) \times_r \boldsymbol{m}_{c_3(s)} \\ &= (\boldsymbol{C}_{rrr} \times_r \boldsymbol{F}_{c_1(s)} \times_r \boldsymbol{F}_{c_2(s)} \times_r \boldsymbol{F}_{c_3(s)}) \times_{\omega_{c_1(s)}} (\boldsymbol{m}_{c_1(s)} \times_r \boldsymbol{F}^{\dagger}_{c_1(s)}) \times_{\omega_{c_2(s)}} (\boldsymbol{m}_{c_2(s)} \times_r \boldsymbol{F}^{\dagger}_{c_2(s)}) \times_{\omega_{c_3(s)}} (\boldsymbol{m}_{c_3(s)} \times_{\omega_{c_3(s)}} \boldsymbol{F}^{\dagger}_{c_3(s)}) \end{aligned}$$

This continues recursively e.g.

$$\begin{aligned}
\boldsymbol{m}_{c_1} &= \boldsymbol{C}_{ss|\pi(s)} \times_s (\boldsymbol{F}_{c_1} \times_{\omega_{c_1}} \boldsymbol{F}_{c_1}^{\dagger}) \times_s \boldsymbol{m}_{c_1(s)} \times_s (\boldsymbol{F}_{c_2} \times_{\omega_{c_2}} \boldsymbol{F}_{c_2}^{\dagger}) \times_s \boldsymbol{m}_{c_2(s)} \\
&= (\boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{F}_{c_1(s)} \times_s \boldsymbol{F}_{c_2(s)}) \\
&\times_{\omega_{c_1(s)}} (\boldsymbol{m}_{c_1(s)} \times_s \boldsymbol{F}_{c_1(s)}^{\dagger}) \times_{\omega_{c_2(s)}} (\boldsymbol{m}_{c_2(s)} \times_s \boldsymbol{F}_{c_2(s)})^{\dagger})
\end{aligned}$$

where $\omega_{c_1(s)}, \omega_{c_2(s)}, \omega_{c_3(s)}$ depend on the definition of $\boldsymbol{F}$ and will be defined in the section. This leads to the transformed representation:

- **root**: $\widetilde{\mathcal{R}} = \boldsymbol{C}_{rrr} \times_r \boldsymbol{F}_{c_1(s)} \times_r \boldsymbol{F}_{c_2(s)} \times_r \boldsymbol{F}_{c_3(s)}$
- **internal**: $\widetilde{\mathcal{T}}_s = \boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{F}_{c_1(s)} \times_s \boldsymbol{F}_{c_2(s)} \times_{\pi(s)} \boldsymbol{F}_s^{\dagger}$
- **leaf**: $\widetilde{\mathcal{L}}_s = \boldsymbol{C}_{s|\pi(s)} \times_{\pi(s)} \boldsymbol{F}_s^{\dagger}$

6.4.1 Observable Representation

Lemma 14. *If we set $\boldsymbol{F}_s = \boldsymbol{U}^{\top}\boldsymbol{C}_{s^*|\pi(s)}$ where $\boldsymbol{U}_s$ are the top left singular vectors of $\boldsymbol{C}_{s^*,-s^*}$ and $\boldsymbol{U}_s$ has mode labels $\{s, \omega_s\}$ then we have that*

- $\widetilde{\mathcal{R}} = \boldsymbol{C}_{c_1(s)^*c_2(s)^*c_3(s)^*} \times_{c_1(s)^*} \boldsymbol{U}_{c_1(s)} \times_{c_2(s)^*} \boldsymbol{U}_{c_2(s)} \times_{c_3(s)^*} \boldsymbol{U}_{c_3(s)}$
- $\widetilde{\mathcal{T}}_s = \boldsymbol{C}_{c_1(s)^*c_2(s)^*-s^*} \times_{-s^*} (\boldsymbol{U}_s^{\top}\boldsymbol{C}_{s^*,-s^*})^{\dagger} \times_{c_1(s)^*} \boldsymbol{U}_{c_1(s)} \times_{c_2(s)^*} \boldsymbol{U}_{c_2(s)}$
- $\widetilde{\mathcal{L}}_s = \boldsymbol{C}_{s,-s^*} \times_{-s^*} (\boldsymbol{U}_s^{\top}\boldsymbol{C}_{s,-s^*})^{\dagger} = \boldsymbol{U}_s$

Proof. **Root**: We first prove that $\boldsymbol{C}_{rrr} \times_r \boldsymbol{C}_{c_1(r)^*|r} \times_r \boldsymbol{C}_{c_2(r)^*|r} \times_r \boldsymbol{C}_{c_3(r)^*|r} = \boldsymbol{C}_{c_1(r)^*c_2(r)^*c_3(r)^*}$: Consider any $\boldsymbol{f}, \boldsymbol{g}, \boldsymbol{h} \in \mathcal{F}$. Then,

$$\begin{aligned}
\boldsymbol{C}_{rrr} \quad \times_r \quad & \boldsymbol{C}_{c_1(r)^*|r} \times_r \boldsymbol{C}_{c_2(r)^*|r} \times_r \boldsymbol{C}_{c_3(r)^*|r} \times_{c_3(r)^*} \boldsymbol{h} \times_{c_2(r)^*} \boldsymbol{g} \times_{c_1(r)^*} \boldsymbol{f} \\
= \quad & \left\langle \boldsymbol{f} \otimes \boldsymbol{g} \otimes \boldsymbol{h}, \boldsymbol{C}_{rrr} \times_r \boldsymbol{C}_{c_1(r)^*|r} \times_r \boldsymbol{C}_{c_2(r)^*|r} \times_r \boldsymbol{C}_{c_3(r)^*|r} \right\rangle \\
= \quad & \mathbb{E}_{X_r}\left[\left\langle \boldsymbol{C}_{c_1(r)^*|r}^{\top}\boldsymbol{f}, \boldsymbol{\phi}(X_r) \right\rangle \left\langle \boldsymbol{C}_{c_2(r)^*|r}^{\top}\boldsymbol{g}, \boldsymbol{\phi}(X_r) \right\rangle \left\langle \boldsymbol{C}_{c_3(r)^*|r}^{\top}\boldsymbol{h}, \boldsymbol{\phi}(X_r) \right\rangle \right] \\
= \quad & \mathbb{E}_{X_r}\left[\left\langle \boldsymbol{f}, \boldsymbol{C}_{c_1(r)^*|r}\boldsymbol{\phi}(X_r) \right\rangle \left\langle \boldsymbol{g}, \boldsymbol{C}_{c_2(r)^*|r}\phi(X_r) \right\rangle \left\langle \boldsymbol{h}, \boldsymbol{C}_{c_3(r)^*|r}\boldsymbol{\phi}(X_r) \right\rangle \right] \\
= \quad & \mathbb{E}_{X_r}\left[\mathbb{E}_{X_{c_1(r)^*}|X_r}\left[\boldsymbol{f}(X_{c_1(r)^*}) \right] \mathbb{E}_{X_{c_2(r)^*}|X_r}\left[\boldsymbol{g}(X_{c_2(r)^*}) \right] \mathbb{E}_{X_{c_3(r)^*}|X_r}\left[\boldsymbol{h}(X_{c_3(r)^*}) \right] \right] \\
= \quad & \mathbb{E}_{X_{c_1(r)^*}, X_{c_2(r)^*}, X_{c_3(r)^*}}\left[\boldsymbol{f}(X_{c_1(r)^*}) \boldsymbol{g}(X_{c_2(r)^*}) \boldsymbol{h}(X_{c_3(r)^*}) \right] \\
= \quad & \left\langle \boldsymbol{f} \otimes \boldsymbol{g} \otimes \boldsymbol{h}, \mathbb{E}_{X_{c_1(r)^*}, X_{c_2(r)^*}, X_{c_3(r)^*}}\left[\boldsymbol{\phi}(X_{c_1(r)^*}) \otimes \boldsymbol{\phi}(X_{c_2(r)^*}) \otimes \boldsymbol{\phi}(X_{c_3(r)^*}) \right] \right\rangle \\
= \quad & \boldsymbol{C}_{c_1(r)^*c_2(r)^*c_3(r)^*} \times_{c_3(r)^*} \boldsymbol{h} \times_{c_2(r)^*} \boldsymbol{g} \times_{c_1(r)^*} \boldsymbol{f} \qquad (6.8)
\end{aligned}$$

We can thus conclude that

$$\widetilde{\mathcal{R}} = \boldsymbol{C}_{c_1(s)^*c_2(s)^*c_3(s)^*} \times_{c_1(s)^*} \boldsymbol{U}_{c_1(s)} \times_{c_2(s)^*} \boldsymbol{U}_{c_2(s)} \times_{c_3(s)^*} \boldsymbol{U}_{c_3(s)} \qquad (6.9)$$

Leaf: Consider expanding the related quantity $\widetilde{\mathcal{L}}_s(\boldsymbol{U}_s^\top \boldsymbol{C}_{s\pi(s)^*})$:

$$\begin{aligned}
\widetilde{\mathcal{L}}_s(\boldsymbol{U}_s^\top \boldsymbol{C}_{s\pi(s)^*}) &= \boldsymbol{C}_{s|\pi(s)}(\boldsymbol{U}_s^\top \boldsymbol{C}_{s|\pi(s)})^\dagger(\boldsymbol{U}_s^\top \boldsymbol{C}_{s\pi(s)^*}) \\
&= \boldsymbol{C}_{s|\pi(s)}(\boldsymbol{U}_s^\top \boldsymbol{C}_{s|\pi(s)})^\dagger(\boldsymbol{U}_s^\top \boldsymbol{C}_{s|\pi(s)}\boldsymbol{C}_{\pi(s)\pi(s)}\boldsymbol{C}^\top_{\pi(s)^*|\pi(s)}) \\
&= \boldsymbol{C}_{s|\pi(s)}\boldsymbol{C}_{\pi(s)\pi(s)}\boldsymbol{C}^\top_{\pi(s)^*|\pi(s)} \\
&= \boldsymbol{C}_{s\pi(s)^*} \qquad (6.10)
\end{aligned}$$

where we have used the fact that $\boldsymbol{C}_{s|\pi(s)}\boldsymbol{C}_{\pi(s)^2}\boldsymbol{C}^\top_{\pi(s)^*|\pi(s)} = \boldsymbol{C}_{s\pi(s)^*}$ (which is proved using the same technique as used for the proof of the root).

This implies that $\widetilde{\mathcal{L}}_s = (\boldsymbol{C}_{\pi(s)^*s}\boldsymbol{U}_s)^\dagger \boldsymbol{C}_{\pi(s)^*s} = \boldsymbol{U}_s$.

Intermediate Node: Consider expanding the quantity $\widetilde{\mathcal{T}}_s \times_{\omega_s} (\boldsymbol{C}_{\pi(s)^*s^*}\boldsymbol{U}_s)$:

$$\begin{aligned}
&\widetilde{\mathcal{T}}_s \times_{\omega_s} (\boldsymbol{C}_{\pi(s)^*s^*}\boldsymbol{U}_s) \\
=\ &\boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{U}^\top_{c_1(s)}\boldsymbol{C}_{c_1(s)^*|s} \times_s \boldsymbol{U}^\top_{c_2(s)}\boldsymbol{C}_{c_2(s)^*|s} \times_{\pi(s)} (\boldsymbol{C}^\top_{s^*|\pi(s)}\boldsymbol{U}_s)^\dagger \times_{\omega_s} (\boldsymbol{C}_{\pi(s)^*s^*}\boldsymbol{U}_s) \\
=\ &\boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{C}_{c_1(s)^*|s} \times_s \boldsymbol{C}_{c_2(s)^*|s} \times_{\pi(s)} (\boldsymbol{C}_{\pi(s)^*s^*}\boldsymbol{U}_s)(\boldsymbol{C}^\top_{s^*|\pi(s)}\boldsymbol{U}_s)^\dagger \times_{c_1(s)^*} \boldsymbol{U}^\top_{c_1(s)} \times_{c_2(s)^*} \boldsymbol{U}^\top_{c_2(s)} \\
=\ &\boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{C}_{c_1(s)^*|s} \times_s \boldsymbol{C}_{c_2(s)^*|s} \times_{\pi(s)} (\boldsymbol{C}_{\pi(s)^*|\pi(s)}\boldsymbol{C}_{\pi(s)\pi(s)})(\boldsymbol{C}^\top_{s^*|\pi(s)}\boldsymbol{U}_s)(\boldsymbol{C}^\top_{s^*|\pi(s)}\boldsymbol{U}_s)^\dagger \times_{c_1(s)^*} \boldsymbol{U}^\top_{c_1(s)} \times_{c_2(s)^*} \boldsymbol{U}^\top_{c_2(s)} \\
=\ &\boldsymbol{C}_{c_1(s)^*c_2(s)^*\pi(s)^*} \times_{c_1(s)^*} \boldsymbol{U}^\top_{c_1(s)^*} \times_{c_2(s)^*} \boldsymbol{U}^\top_{c_2(s)^*} \qquad (6.11)
\end{aligned}$$

where in the last line we have claimed that $\boldsymbol{C}_{c_1(s)^*c_2(s)^*\pi(s)^*} = \boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{C}_{c_1(s)^*|s} \times_s \boldsymbol{C}_{c_2(s)^*|s} \times_{\pi(s)} \boldsymbol{C}_{\pi(s)^*|\pi(s)}\boldsymbol{C}_{\pi(s)\pi(s)}$. To prove this assertion, first consider the $\boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{C}_{c_1(s)^*|s} \times_s \boldsymbol{C}_{c_2(s)^*|s}$ part. For any $f, g \in \mathcal{F}$:

$$\begin{aligned}
\left\langle f \otimes g, \boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{C}_{c_1(s)^*|s} \times_s \boldsymbol{C}_{c_2(s)^*|s} \times_{\pi(s)} \phi(x_{\pi(s)})\right\rangle &= \left\langle (\boldsymbol{C}^\top_{c_1(s)^*|s}f) \otimes (\boldsymbol{C}^\top_{c_2(s)^*|s}g), \boldsymbol{C}_{ss|\pi(s)} \times_{\pi(s)} \phi(x_{\pi(s)})\right\rangle \\
&= \left\langle (\boldsymbol{C}^\top_{c_1(s)^*|s}f) \otimes (\boldsymbol{C}^\top_{c_2(s)^*|s}g), \mathbb{E}_{X_s|x_{\pi(s)}}[\phi(X_s) \otimes \phi(X_s)]\right\rangle \\
&= \mathbb{E}_{X_s|x_{\pi(s)}}\left[\left\langle (\boldsymbol{C}^\top_{c_1(s)^*|s}f) \otimes (\boldsymbol{C}^\top_{c_2(s)^*|s}g), \phi(X_s) \otimes \phi(X_s)\right\rangle\right] \\
&= \mathbb{E}_{X_s|x_{\pi(s)}}\left[\left\langle f, \boldsymbol{C}_{c_1(s)^*|s}\phi(X_s)\right\rangle\left\langle g, \boldsymbol{C}_{c_2(s)^*|s}\phi(X_s)\right\rangle\right] \\
&= \mathbb{E}_{X_s|x_{\pi(s)}}\left[\mathbb{E}_{X_{c_1(s)^*}|X_s}[f(X_{c_1(s)^*})]\mathbb{E}_{X_{c_2(s)^*}|X_s}[g(X_{c_2(s)^*})]\right] \\
&= \mathbb{E}_{c_1(s)^*, c_2(s)^*|x_{\pi(s)}}\left[f(X_{c_1(s)^*})g(X_{c_2(s)^*})\right] \\
&= \left\langle f \otimes g, \boldsymbol{C}_{c_1(s)^*, c_2(s)^*|\pi(s)} \times_{\pi(s)} \phi(x_{\pi(s)})\right\rangle \qquad (6.12)
\end{aligned}$$

Thus, $\boldsymbol{C}_{c_1(s)^*c_2(s)^*|\pi(s)} = \boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{C}_{c_1(s)^{**}|s} \times_s \boldsymbol{C}_{c_2(s)^*|s}$. We can then conclude (using a similar derivation to that for the root) that $\boldsymbol{C}_{c_1(s)^*, c_2(s)^*, \pi(s)^*} = \boldsymbol{C}_{c_1(s)^{**}c_2(s)^*|\pi(s)} \times_{\pi(s)} \boldsymbol{C}_{\pi(s)^*|\pi(s)}\boldsymbol{C}_{\pi(s)\pi(s)}$. Thus,

$$\boldsymbol{C}_{c_1(s)^*, c_2(s)^*, \pi(s)^*} = \boldsymbol{C}_{ss|\pi(s)} \times_s \boldsymbol{C}_{c_1(s)^*|s} \times_s \boldsymbol{C}_{c_2(s)^*|s} \times_{\pi(s)} \boldsymbol{C}_{\pi(s)^*|\pi(s)}\boldsymbol{C}_{\pi(s)\pi(s)} \qquad (6.13)$$

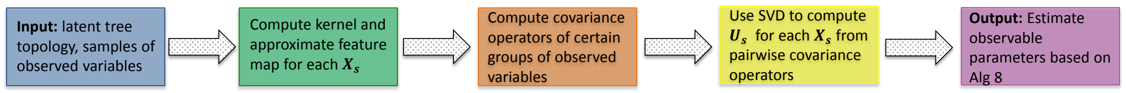

Input: latent tree topology, samples of observed variables
Compute kernel and approximate feature map for each X_s
Compute covariance operators of certain groups of observed variables
Use SVD to compute U_s for each X_s from pairwise covariance operators
Output: Estimate observable parameters based on Alg 8

Figure 6.3: Flowchart that gives an overview of Algorithm 8

Now, returning to Eq. 6.11 we get that

$$\widetilde{\mathcal{T}}_s = \mathcal{C}_{c_1(s)^{**}c_2(s)^*\pi(s)^*} \times_{c_1(s)^*} \mathcal{U}_{c_1(s)}^\top \times_{c_2(s)^*} \mathcal{U}_{c_2(s)}^\top \times_{\pi(s)^*} (\mathcal{C}_{\pi(s)^*s^*}\mathcal{U}_s)^\dagger \tag{6.14}$$

where one valid choice for s^* is $c_1(s)^*$. □

6.4.2 Training and Test

Empirically, during training, our method needs to compute estimates of the following cross-covariance operators:

$$\widehat{\mathcal{C}}_{s,t,u} = \frac{1}{N}\sum_{n=1}^{N} \phi_{x_s^{(n)}} \otimes \phi_{x_t^{(n)}} \otimes \phi_{x_u^{(n)}}$$
$$\widehat{\mathcal{C}}_{s,t} = \frac{1}{N}\sum_{n=1}^{N} \phi_{x_s^{(n)}} \otimes \phi_{x_t^{(n)}} \tag{6.15}$$

Please see 2.3.5 for how to do this efficiently using incomplete cholesky decompositions.

The training and test algorithms are shown in Algorithms 8 and 9. A high level flowchart of training is given in Figure 6.3. As one can see, although the derivation is different, the end result is very similar to Algorithms 4 and Algorithms 5 in Chapter 4 (the only difference being the probability tensors are replaced with hilbert space operators). Thus, spectral approaches elegantly generalize to nonparametric settings unlike traditional nonconvex-optimization based methods.

6.5 Structure Learning of Latent Tree Graphical Models

The last section focused on parameter learning and inference where the structure of the latent tree is known. In this section, we focus on learning the structure of the latent tree. Structure learning of latent trees is a challenging problem that has largely been tackled by heuristics since the search space of structures is intractable. The additional challenge in our case is that the observed variables are continuous and non-Gaussian, which we are not aware of any existing methods for this problem.

6.5.1 Kernel Tree Metric

We develop a distance based method for constructing latent trees of continuous, non-Gaussian variables. The idea is that if we have a tree metric (distance) between distributions on observed nodes, we can use the property of the tree metric to reconstruct the latent tree structure using algorithms such as neighbor joining (Saitou and Nei, 1987) and the recursive grouping algorithm (Choi et al., 2010). These methods take a distance matrix among all pairs of observed variables as input and output a tree by iteratively adding hidden nodes. While these methods are iterative, they have strong theoretical guarantees on structure recovery when the true distance matrix forms an additive tree metric. However, most previously known tree metrics are defined for discrete and Gaussian variables. The additional challenge in our case is that the observed variables are continuous and non-Gaussian. We propose a tree metric below which works for continuous non-Gaussian cases.

Algorithm 8 Kernel spectral learning algorithm for latent tree graphical model

```
In: Tree topology and $N$ i.i.d. samples $\left\{x_1^{(n)}, \ldots, x_O^{(n)}\right\}_{n=1}^N$
Out: Estimated observable root, internal, and leaf parameters, $\widehat{\mathcal{R}}$, $\widehat{\mathcal{T}}_s$ for each non-root internal node, $\widehat{\mathcal{L}}_s$ for each leaf node
1: For each node $X_s$, compute the kernel matrix $\boldsymbol{K}_s$ where $\boldsymbol{K}_s(m,n) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\|x_s^{(m)} - x_s^{(n)}\|^2/2\sigma^2)$.
2: For each node $X_s$, recover the approximate feature map $\left\{\widehat{\boldsymbol{\phi}}_{x_s^{(1)}}, ..., \widehat{\boldsymbol{\phi}}_{x_s^{(N)}}\right\}$ via incomplete cholesky decomposition on $\boldsymbol{K}_s$.
3: Compute empirical estimates of covariance operators for all observed pairs and triples:
       $$\widehat{\boldsymbol{C}}_{stu} = \frac{1}{N}\sum_{n=1}^{N} \widehat{\boldsymbol{\phi}}_{x_s^{(n)}} \otimes \widehat{\boldsymbol{\phi}}_{x_t^{(n)}} \otimes \widehat{\boldsymbol{\phi}}_{x_u^{(n)}}$$
       $$\widehat{\boldsymbol{C}}_{st} = \frac{1}{N}\sum_{n=1}^{N} \widehat{\boldsymbol{\phi}}_{x_s^{(n)}} \otimes \widehat{\boldsymbol{\phi}}_{x_t^{(n)}} \qquad (6.16)$$
4: For each node $X_s$, perform a "thin" singular value decomposition of $\boldsymbol{C}_{s^*-s^*} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$; let $\widehat{\boldsymbol{U}}_i = \boldsymbol{U}(:, 1 : S_H)$ be the the first $S_H$ principal left singular vectors.
5: Compute observable parameters as:
       $$\widehat{\widehat{\mathcal{R}}} = \widehat{\boldsymbol{C}}_{c_1(s)^* c_2(s)^* c_3(s)^*} \times_{c_1(s)^*} \widehat{\boldsymbol{U}}_{c_1(s)} \times_{c_2(s)^*} \widehat{\boldsymbol{U}}_{c_2(s)} \times_{c_3(s)^*} \widehat{\boldsymbol{U}}_{c_3(s)}$$
       $$\widehat{\mathcal{T}}_s = \widehat{\boldsymbol{C}}_{c_1(s)^* c_2(s)^* -s^*} \times_{-s^*} (\widehat{\boldsymbol{U}}_s^\top \widehat{\boldsymbol{C}}_{s^*,-s^*})^\dagger \times_{c_1(s)^*} \widehat{\boldsymbol{U}}_{c_1(s)} \times_{c_2(s)^*} \widehat{\boldsymbol{U}}_{c_2(s)}$$
       $$\widehat{\mathcal{L}}_s = \widehat{\boldsymbol{C}}_{s,-s^*} \times_{-s^*} (\widehat{\boldsymbol{U}}_s^\top \widehat{\boldsymbol{C}}_{s,-s^*})^\dagger = \widehat{\boldsymbol{U}}_s$$
```

Algorithm 9 Inference with Kernel Spectral Parameters

```
In: Tree topology, set of spectral parameters $\widehat{\mathcal{R}}$, $\widehat{\mathcal{T}}_s$ for all non-root internal nodes, $\widehat{\mathcal{L}}_s$ for all leaf nodes, evidence $\{\bar{x}_1, ..., \bar{x}_O\}$, and associated feature functions $\{\widehat{\boldsymbol{\phi}}_{\bar{x}_1}, ...., \widehat{\boldsymbol{\phi}}_{\bar{x}_O}\}$
Out: estimated unnormalized probability $b_r \propto \widehat{P}(\bar{x}_1, ..., \bar{x}_O)$

1: In reverse topological order, each node accumulates at message at leaf and sends to parent
       • Leaf: $\widehat{\boldsymbol{m}}_s = \widehat{\mathcal{L}}_s \times_s \boldsymbol{\phi}_{\bar{x}_s}$
       • Internal Node: $\widehat{\boldsymbol{m}_s} = \widehat{\mathcal{T}}_s \times_s \widehat{\boldsymbol{m}}_{c_1(s)} \times_s \widehat{\boldsymbol{m}}_{c_2(s)}$
       • Root: $b_r = \widehat{\mathcal{R}} \times_i \boldsymbol{m}_{c_1(r)} \times_r \widehat{\boldsymbol{m}}_{c_2(r)} \times_r \widehat{\boldsymbol{m}}_{c_3(r)}$
```

Tree metric and pseudo-determinant We will first explain some basic concepts of a tree metric. If the joint probability distribution $P(\mathscr{X})$ has a latent tree structure, then a distance measure $d(s,t)$ between an arbitrary variables pairs X_s and X_t are called tree metric if it satisfies the following path additive condition: $d(s,t) = \sum_{(u,v)\in Path(s,t)} d(u,v)$. For discrete and Gaussian variables, tree metric can be defined via the determinant $|\cdot|$ (Choi et al., 2010)

$$d(s,t) = -\tfrac{1}{2}\log|\boldsymbol{\mathcal{P}}(X_s,X_t)\boldsymbol{\mathcal{P}}(X_s,X_t)^\top| + \tfrac{1}{4}\log|\boldsymbol{\mathcal{P}}(\oslash_2 X_s)\boldsymbol{\mathcal{P}}(\oslash_2 X_s)^\top| + \tfrac{1}{4}\log|\boldsymbol{\mathcal{P}}(\oslash_2 X_t)\boldsymbol{\mathcal{P}}(\oslash_2 X_t)^\top| \quad (6.17)$$

However, this definition of tree metric is restricted in the sense that it requires all discrete variables to have the same number of states and all Gaussian variables have the same dimension. This is because determinant is only defined (and non-zero) for square and non-singular matrices. For our more general scenario, where the observed variables are continuous non-Gaussian but the hidden variables have dimension S_H, we will define a tree metric based on pseudo-determinant which works for our operators.

Nonparametric tree metric The pseudo-determinant is defined as the product of non-zero singular values of an operator $|C|_\star = \prod_{i=1}^{S_H}\sigma_i(C)$. In our case, since we assume that the dimension of the hidden variables is S_H, the pseudo-determinant is simply the product of top S_H singular values. Then we define the distance metric between as two continuous non-Gaussian variables X_s and X_t as follows:

$$d(s,t) = -\tfrac{1}{2}\log\left|\boldsymbol{C}_{st}\boldsymbol{C}_{st}^\top\right|_\star + \tfrac{1}{4}\log|\boldsymbol{C}_{ss}\boldsymbol{C}_{ss}^\top|_\star + \tfrac{1}{4}\log|\boldsymbol{C}_{tt}\boldsymbol{C}_{tt}^\top|_\star. \quad (6.18)$$

Lemma 15. *The distance measure in Eq. 6.18 is an additive tree metric.*

Proof. Here we just show why the lemma holds for the simple path $s \leftarrow u \rightarrow t$ where u is latent and s and t are observed. The full proof is in the Appendix. Note that

$$\boldsymbol{C}_{st} = \boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{t|u}^\top \quad (6.19)$$

Thus,

$$\boldsymbol{C}_{st}\boldsymbol{C}_{st}^\top = \boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{t|u}^\top\boldsymbol{C}_{t|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{s|u}^\top \quad (6.20)$$

Combining this definition with Sylvester's Determinant Theorem (Akritas et al., 1996), gives us that:

$$\begin{aligned}|\boldsymbol{C}_{st}\boldsymbol{C}_{st}^\top|_+ &= |\boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{t|u}^\top\boldsymbol{C}_{t|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{s|u}^\top|_+ \\ &= |\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{t|u}^\top\boldsymbol{C}_{t|u}\boldsymbol{C}_{uu}|_+ \end{aligned} \quad (6.21)$$

(i.e. we can move $\boldsymbol{C}_{s|u}^\top$ the to the front).

Now $\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{t|u}^\top\boldsymbol{C}_{t|u}\boldsymbol{C}_{uu}$ has rank S_H so the pseudo-determinant equals the normal determinant in this case. Using the fact that $|\boldsymbol{AB}| = |\boldsymbol{A}||\boldsymbol{B}|$ if $\boldsymbol{A}$ and $\boldsymbol{B}$ are square, we get

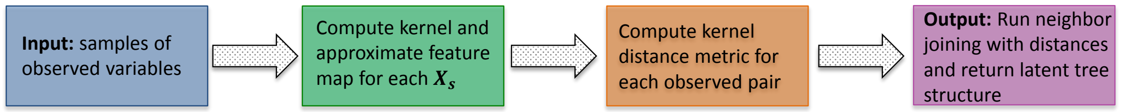

Input: samples of observed variables
Compute kernel and approximate feature map for each X_s
Compute kernel distance metric for each observed pair
Output: Run neighbor joining with distances and return latent tree structure

Figure 6.4: Flowchart that gives an overview of Algorithm 10

$$\begin{aligned} |C_{st}C_{st}^\top|_+ &= |C_{s|u}^\top C_{s|u}C_{uu}C_{t|u}^\top C_{t|u}C_{uu}| \\ &= |C_{s|u}^\top C_{s|u}C_{uu}||C_{t|u}^\top C_{t|u}C_{uu}| \\ &= \frac{|C_{uu}||C_{s|u}^\top C_{s|u}C_{uu}|}{|C_{uu}|} \times \frac{|C_{uu}||C_{t|u}^\top C_{t|u}C_{uu}|}{|C_{uu}|} \\ &= \frac{|C_{uu}C_{s|u}^\top C_{s|u}C_{uu}|}{|C_{uu}|} \times \frac{|C_{uu}C_{t|u}^\top C_{t|u}C_{uu}|}{|C_{uu}|} \end{aligned} \tag{6.22}$$

Furthermore, note that

$$|C_{uu}C_{s|u}^\top C_{s|u}C_{uu}| = |C_{s|u}C_{uu}C_{uu}C_{s|u}^\top|_+ = |C_{su}C_{su}^\top|_+ \tag{6.23}$$

This gives,

$$|C_{st}C_{st}^\top|_+ = \frac{|C_{su}C_{su}^\top|_+}{|C_{uu}|} \times \frac{|C_{tu}C_{tu}^\top|_+}{|C_{uu}|} \tag{6.24}$$

Substituting back into Eq. (6.18) proves that

$$\begin{aligned} d(s,t) &= -\frac{1}{2}\log|C_{su}C_{su}^\top|_+ - \frac{1}{2}\log|C_{ut}C_{ut}^\top|_+ + \frac{1}{2}\log|C_{uu}C_{uu}^\top|_+ \\ &+ \tfrac{1}{4}\log|C_{ss}C_{ss}^\top|_+ + \tfrac{1}{4}\log|C_{tt}C_{tt}^\top|_+ \\ &= d(s,u) + d(u,t) \end{aligned} \tag{6.25}$$

□

6.5.2 Structure learning algorithm

Our structure learning algorithm, shown in Algorithm 10 with a high level flowchart in Figure 6.4, works by first computing a kernel and approximate feature map (via incomplete cholesky decomposition) for each variable X_s. Then for each pair of variances (X_s, X_t), the empirical covariance operators $\widehat{C}_{st}$, $\widehat{C}_{ss}$, and $\widehat{C}_{tt}$ are computed, and $d(s,t)$ is computed using Eq. 6.18. The distances are then used as input to the neighbor joining algorithm in Algorithm 11 described in the Appendix.

6.6 Experiments

We evaluate our method on synthetic data as well as a real-world crime/communities dataset (Asuncion and Newman, 2007; Redmond and Baveja, 2002). We primarily compare to 2 existing approaches. The first is to assume the data is multivariate Gaussian and use the tree metric defined in Choi et al. (2010) (which is essentially a function of the correlation coefficient). The second existing approach we compare to is the Nonparanormal (NPN) (Liu et al., 2009) which assumes that there exist marginal transformations $\boldsymbol{f}_1, \dots, \boldsymbol{f}_p$ such that $\boldsymbol{f}(X_1), \dots, \boldsymbol{f}(X_p) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. If the data comes from a Nonparanormal distribution, then the transformed data are assumed to be multivariate Gaussian and the same tree metric as the Gaussian case can be used on the transformed data. Our approach makes much fewer assumptions about the data than either of these two methods which can be more favorable in practice.

To perform inference in our approach, we use the spectral algorithm described earlier in the paper. For inference in the Gaussian (and nonparanormal) cases, we use the technique in Choi et al. (2010) to learn the model parameters (covariance matrix). Once the covariance matrix has been estimated, marginalization in a Gaussian graphical model reduces to solving a linear equation of one variable if we are only computing the marginal of one variable given a set of evidence (Bickson, 2008).

6.6.1 Synthetic data: density estimation

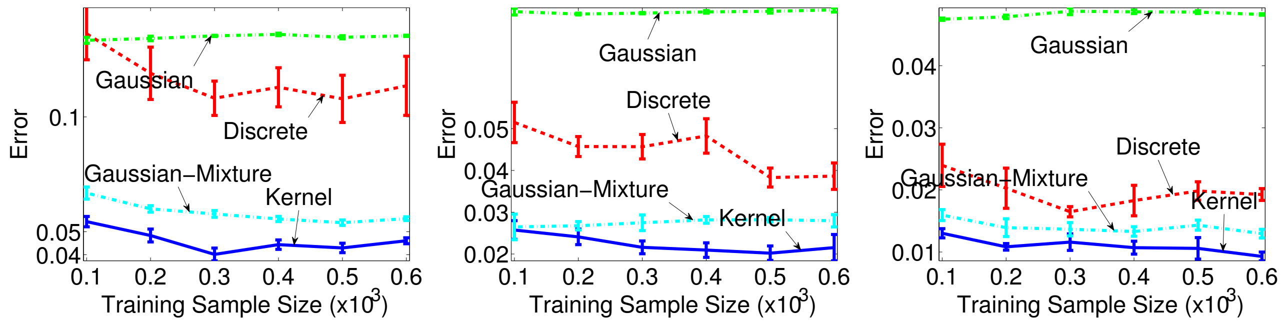
Before moving on to larger experiments, we first show that our model assumptions can result in more accurate density estimation of non-Gaussian distributions. The underlying data is generated as a two dimensional mixture of exponentials:

$$P(x_1, x_2) \propto \exp(\|x_1 - \mu_1\| + \|x_2 - \mu_2\|) + \exp(\|x_1 + \mu_1\| + \|x_2 + \mu_2\|) \tag{6.26}$$

Note that the first component has mean (μ_1, μ_2) while the second component has mean $(-\mu_1, -\mu_2)$. We experiment with the different values $(\mu_1, \mu_2) = (2, 2)$, $(\mu_1, \mu_2) = (4, 4)$, and $(\mu_1, \mu_2) = (6, 6)$. For all methods we evaluate the density on a grid $\boldsymbol{G}$ of evenly spaced points in $[-2\mu_1, 2\mu_1] \times [-2\mu_2, 2\mu_2]$.

Algorithm 10 Kernel structure learning algorithm for latent tree graphical model

```
In: $N$ i.i.d. samples $\left\{x_1^{(n)}, \dots, x_O^{(n)}\right\}_{n=1}^N$
Out: estimated tree topology $\mathscr{T}$
1: For each node $X_s$, compute the kernel matrix $\boldsymbol{K}_s$ where $\boldsymbol{K}_s(m,n) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\|x_s^{(m)} - x_s^{(n)}\|^2/2\sigma^2)$.
2: For each node $X_s$, recover the approximate feature map $\left\{\widehat{\boldsymbol{\phi}}_{x_s^{(1)}}, \dots, \widehat{\boldsymbol{\phi}}_{x_s^{(N)}}\right\}$ via incomplete cholesky decomposition on $\boldsymbol{K}_s$.
3: for each $(X_s, X_t)$ s.t. $1 \le s, t \le O$ and $s \ne t$ do
4:     Compute $\widehat{\boldsymbol{C}}_{st} = \frac{1}{N}\sum_{n=1}^N \widehat{\boldsymbol{\phi}}_{x_s^{(n)}} \otimes \widehat{\boldsymbol{\phi}}_{x_t^{(n)}}$
5:     Compute $\widehat{\boldsymbol{C}}_{ss} = \frac{1}{N}\sum_{n=1}^N \widehat{\boldsymbol{\phi}}_{x_s^{(n)}} \otimes \widehat{\boldsymbol{\phi}}_{x_s^{(n)}}$
6:     Compute $\widehat{\boldsymbol{C}}_{tt} = \frac{1}{N}\sum_{n=1}^N \widehat{\boldsymbol{\phi}}_{x_t^{(n)}} \otimes \widehat{\boldsymbol{\phi}}_{x_t^{(n)}}$
7:     Compute $d(s,t)$ according to Eq. 6.18
8: end for
9: Call Neighbor Joining (Algorithm 11) with distances $d(s,t)$ and return resulting tree $\mathscr{T}$
```



(a) $(\mu_1, \mu_2) = (2, 2)$ (b) $(\mu_1, \mu_2) = (4, 4)$ (c) $(\mu_1, \mu_2) = (6, 6)$

Figure 6.5: Density estimation of 2-dimensional mixture of laplace distributions.

The total error is measured as: $err = \sqrt{\sum_{(x_1,x_2)\in \mathbf{G}} ||P(x_1, x_2) - \widehat{P}(x_1, x_2)||^2}$.

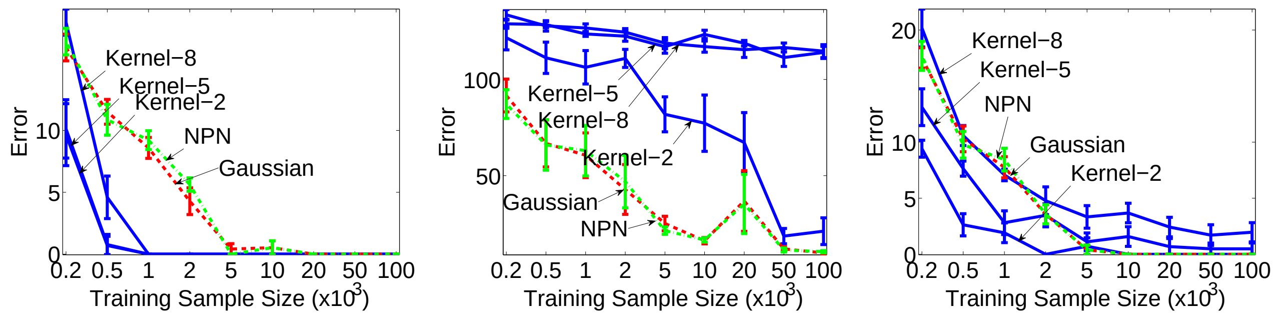
Figure 6.5, shows the results where we compare our approach with the Gaussian and Gaussian mixture distributions as well as a histogram-based approach (called "Discrete"). As expected, the problem is more difficult when the components are closer together. Our method performs the best for all the cases.

6.6.2 Synthetic data: structure recovery.

The second experiment is to demonstrate how our method compares to the Gaussian and Nonparanormal methods in terms of structure recovery for larger trees. We experiment with 3 different tree types (each with 64 leaves or observed variables): a balanced binary tree, a completely binary skewed tree (like an HMM), and randomly generated binary trees. Furthermore we explore with two types of underlying distributions: **(1)** A multivariate Gaussian with mean zero and inverse covariance matrix that respects the tree structure. **(2)** A highly non-Gaussian distribution that uses the following generative process to generate the n-th sample from a node s in the tree (denoted $x_s^{(n)}$): If s is the root, sample from a mixture of 2 Gaussians. Else, with probability $\frac{1}{2}$ sample from a Gaussian with mean $-x_{\pi_s}^{(n)}$ and with probability $\frac{1}{2}$ sample from a Gaussian with mean $x_{\pi_s}^{(n)}$.

We vary the training sample size from 200 to 100,000. Once we have computed the empirical tree distance matrix for each algorithm, we use the neighbor joining algorithm (Saitou and Nei, 1987) to learn the trees. For evaluation we compare the number of hops between each pair of leaves in the true tree to the estimated tree. For a pair of leaves i, j the error is defined as: $error(i, j) = \frac{|hops^*(i,j) - \widehat{hops}(i,j)|}{hops^*(i,j)} + \frac{|hops^*(i,j) - \widehat{hops}(i,j)|}{\widehat{hops}(i,j)}$, where $hops^*$ is the true number of hops and $\widehat{hops}$ is the estimated number of hops. The total error is then computed by adding the error for each pair of leaves.

The performance of our method depends on the number of singular values chosen and we experimented with 2, 5 and 8 singular values. Furthermore, we choose the bandwidth σ for the Gaussian RBF kernel needed for the covariance operators using median distance between pairs of training points.



(a) balanced binary tree (b) skewed HMM-like tree (c) random trees

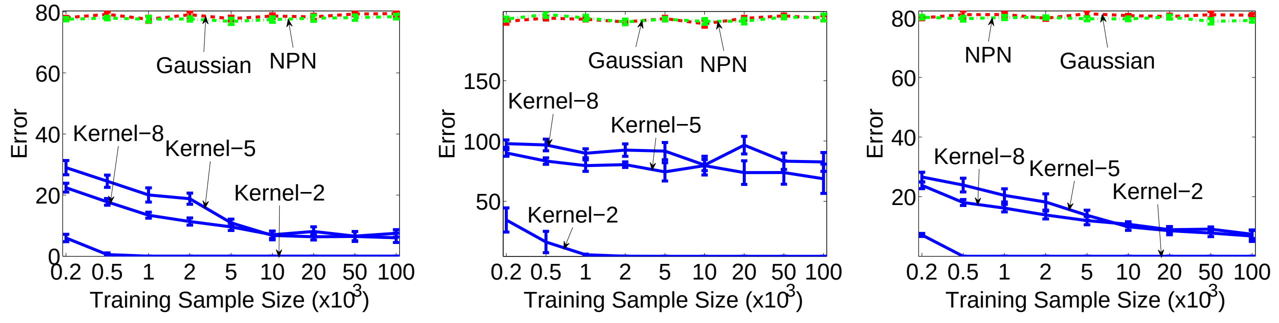


Figure 6.6: Comparison of our kernel structure learning method to the Gaussian and Nonparanormal methods on different tree structures. Top row: data points are generated from Gaussian distributions with latent variables connected by a tree structure. Bottom row: data points are generated from mixture of Gaussian distributions with latent variables connected by a tree structure. Especially in the latter case, our kernel structure learning method is able to adapt the data distributions and recover the structure in a much more accurate way.

When the underlying distribution is not Gaussian, our method performs better than the Gaussian and Nonparanormal methods for all the tree structures. This is to be expected, since the non-Gaussian data we generated is neither Gaussian or Nonparanormal, yet our method is able to learn the structure correctly. We also note that balanced binary trees are the easiest to learn while the skewed trees are the hardest (Figure 6.6).

Even when the underlying distribution is Gaussian, our method still performs very well compared to the Gaussian and NPN approaches and outperforms them for the binary and balanced trees. It performs worse for the skewed case likely due to the fact that the eigenvalues (dependence) decay along the length of the tree leading to larger errors in the empirical distance matrix.

Although it would be interesting to compare to the pouch latent tree model (Poon et al., 2010), their model assumes multiple observed variables can exist in the same leaf of the latent tree (unlike our approach) which makes a direct structure comparison difficult.

6.6.3 Crime Dataset.

Finally, we explore the performance of our method on a communities and crime dataset from the UCI repository (Asuncion and Newman, 2007; Redmond and Baveja, 2002). In this dataset several real valued attributes are collected for several communities, such as ethnicity proportions, income, poverty rate, divorce rate etc., and the goal is to predict the number of violent crimes (proportional to size of community) that occur based on these attributes. In general these attributes are highly skewed and therefore not well characterized by a Gaussian model.

We divide the data into 1400 samples for training, 300 samples for model selection (held-out likelihood), and 300 samples for testing. We pick the first 50 of these attributes, plus the violent crime variable and construct a latent tree using our tree metric and neighbor joining algorithm (Saitou and Nei, 1987). We depict the tree in Figure 6.7 and highlight a few coherent groupings. For example, the "elderly" group attributes are those related to retirement and social security (and thus correlated). The large clustering in the center is where the class variable (violent crimes) is located next to the poverty rate, and the divorce rate among other relevant variables. Other groupings include type of occupation and education level as well as ethnic proportions. Thus, overall our method is able to capture sensible relationships.

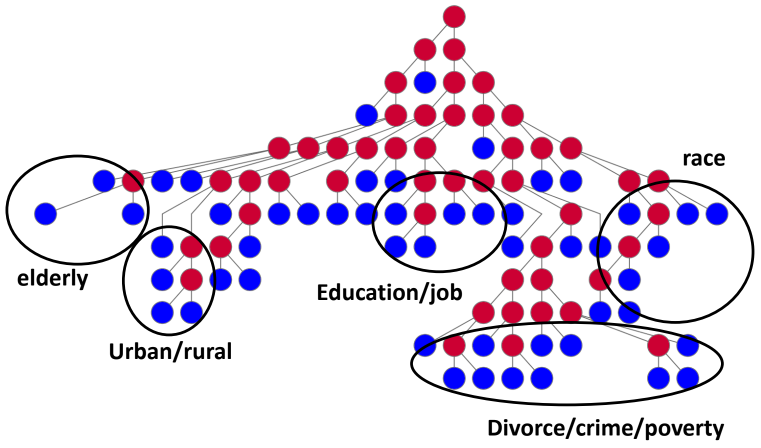
For a more quantitative evaluation, we condition on a set of $\mathcal{E}$ evidence variables where $|\mathcal{E}| = 30$ and predict the violent crimes class label. We experiment with a varying number of sizes of the training set from 200 to 1400. At test, we evaluate on all the 300 test examples for 10 randomly chosen evidence sets of evidence variables. Since the crime variable is a number between 0 and 1, our error measure is simply $err(\widehat{c}) = |\widehat{c} - c^*|$ (where $\widehat{c}$ is the predicted value and c^* is the true value).

In this experiment, in addition to comparing with the Gaussian and the Nonparanormal, we also compare with two standard classifiers, (Gaussian) Naive Bayes and Linear Regression. Although Naive Bayes can easily handle missing values, linear regression cannot. To deal with this problem, we simply use the mean value of a variable if it is not in $\mathcal{E}$ (this performed much better than setting it to zero).

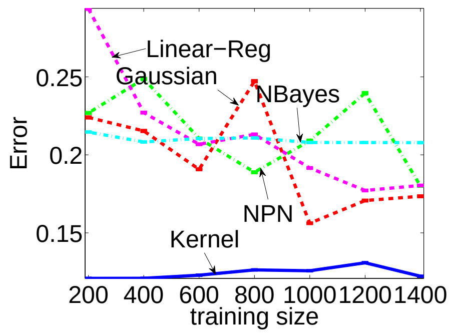
As one can see in Figure 6.7 our method outperforms all the other approaches. We find that our method performs similarly for different choices of $\mathcal{E}$. Moreover, the accuracy of the Gaussian and nonparanormal vary widely for different evidence sets (and thus the more erratic overall performance). Thus, in this case our method is better able to capture the skewed distributions of the variables than the other methods.

6.7 Conclusion

We present a distribution embedding framework for nonparametric latent tree graphical models. Our approach be used to recover the latent tree structures, and perform local-mininum-free spectral learning and inference for continuous and non-Gaussian variables. Both simulation and results on real datasets show the advantage of our proposed approach for non-Gaussian data.



(a)



(b)

Figure 6.7: (a) visualization of kernel latent tree learned from crime data (b) Comparison of our method to Gaussian and NPN in predictive task.

6.8 Appendix

6.8.1 Proof of Lemma 15

Proof. For conciseness, we simply prove the property for paths of length 2. The proof for more general cases follows similarly (e.g. see (Anandkumar et al., 2011)). For paths of length 2, $s - u - t$, there are three cases[2]:

- $s \leftarrow u \rightarrow t$
- $s \leftarrow u \leftarrow t$
- $s \rightarrow u \rightarrow t$

Case 1 was already shown in the main text so we just show Cases 2 and 3 here.

Case 2: $s \leftarrow u \leftarrow t$ Since only leaf nodes can be observed, u and t must be latent but s can be either observed or latent. We assume it is observed, the latent case follows similarly.

Using the conditional independence relationship,

$$\mathcal{C}_{st} = \mathcal{C}_{s|u}\mathcal{C}_{u|t}\mathcal{C}_{tt} \tag{6.27}$$

[2]although the additive distance metric is undirected, the conditional embedding operators are defined with respect to parent-child relationships, so we must consider direction

Thus, via Sylvester's Determinant Theorem (Akritas et al., 1996) as before,

$$\begin{aligned} |\boldsymbol{C}_{st}\boldsymbol{C}_{st}^\top|_+ &= |\boldsymbol{C}_{s|u}\boldsymbol{C}_{u|t}\boldsymbol{C}_{tt}\boldsymbol{C}_{tt}\boldsymbol{C}_{u|t}^\top\boldsymbol{C}_{s|u}^\top|_+ \\ &= |\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}\boldsymbol{C}_{u|t}\boldsymbol{C}_{tt}\boldsymbol{C}_{tt}\boldsymbol{C}_{u|t}^\top|_+ \end{aligned} \tag{6.28}$$

Now $\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}\boldsymbol{C}_{u|t}\boldsymbol{C}_{tt}\boldsymbol{C}_{tt}\boldsymbol{C}_{u|t}^\top$ has rank S_H, the pseudo-determinant equals the normal determinant in this case. Using the fact that $|\boldsymbol{AB}| = |\boldsymbol{A}||\boldsymbol{B}|$ if $\boldsymbol{A}$ and $\boldsymbol{B}$ are square, we get

$$\begin{aligned} |\boldsymbol{C}_{st}\boldsymbol{C}_{st}^\top|_+ &= |\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}\boldsymbol{C}_{u|t}\boldsymbol{C}_{tt}\boldsymbol{C}_{tt}\boldsymbol{C}_{u|t}^\top| \\ &= |\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}||\boldsymbol{C}_{u|t}\boldsymbol{C}_{tt}\boldsymbol{C}_{tt}\boldsymbol{C}_{u|t}^\top| \\ &= |\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}||\boldsymbol{C}_{ut}\boldsymbol{C}_{ut}^\top| \\ &= \frac{|\boldsymbol{C}_{uu}||\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}||\boldsymbol{C}_{uu}|}{|\boldsymbol{C}_{uu}||\boldsymbol{C}_{uu}|} \times |\boldsymbol{C}_{ut}\boldsymbol{C}_{ut}^\top| \\ &= \frac{|\boldsymbol{C}_{uu}\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}|}{|\boldsymbol{C}_{uu}||\boldsymbol{C}_{uu}|} \times |\boldsymbol{C}_{ut}\boldsymbol{C}_{ut}^\top| \end{aligned} \tag{6.29}$$

Furthermore, note that

$$|\boldsymbol{C}_{uu}\boldsymbol{C}_{s|u}^\top\boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}| = |\boldsymbol{C}_{s|u}\boldsymbol{C}_{uu}\boldsymbol{C}_{uu}\boldsymbol{C}_{s|u}^\top|_+ = |\boldsymbol{C}_{su}\boldsymbol{C}_{su}^\top|_+ \tag{6.30}$$

This gives,

$$|\boldsymbol{C}_{st}\boldsymbol{C}_{st}^\top|_+ = \frac{|\boldsymbol{C}_{su}\boldsymbol{C}_{su}^\top|_+}{|\boldsymbol{C}_{uu}||\boldsymbol{C}_{uu}|} \times |\boldsymbol{C}_{ut}\boldsymbol{C}_{ut}^\top|_+ \tag{6.31}$$

Substituting back into Eq. (6.18) proves that

$$\begin{aligned} d(s,t) &= -\frac{1}{2}\log|\boldsymbol{C}_{su}\boldsymbol{C}_{su}^\top|_+ - \frac{1}{2}\log|\boldsymbol{C}_{ut}\boldsymbol{C}_{ut}^\top|_+ + \frac{1}{2}\log|\boldsymbol{C}_{uu}\boldsymbol{C}_{uu}^\top|_+ + \tfrac{1}{4}\log|\boldsymbol{C}_{ss}\boldsymbol{C}_{ss}^\top|_+ + \tfrac{1}{4}\log|\boldsymbol{C}_{tt}\boldsymbol{C}_{tt}^\top|_+ \\ &= d(s,u) + d(u,t) \end{aligned} \tag{6.32}$$

Case 3: $s \to u \to t$ The same argument as case 2 holds here.

□

6.8.2 Neighbor Joining Algorithm

Neighbor joining (Saitou and Nei, 1987) is a distance based algorithm that given a distance matrix among pairs of observed (leaf) nodes, returns a binary latent tree. Although it is greedy, if the distance matrix satisfies the tree additivity property, neighbor joining is provably consistent (Atteson,

Algorithm 11 Neighbor Joining Algorithm (Saitou and Nei, 1987)

```
Input: Pairwise distances $d(s,t)$ between observed variables in $\mathcal{O}$
Output: Latent tree structure $\mathcal{T} = (\mathcal{V}, \mathcal{E})$
  Initialize the latent tree structure: $\mathcal{W} = \{1, ..., O\}, \mathcal{E} = \emptyset$
  Define a working set: $\mathcal{W} = \{1, ..., O\}$
  $u = |\mathcal{O}| + 1$
  while $|\mathcal{W}| \geq 2$ do
    Create $\boldsymbol{Q}$ matrix according to Eq. 6.33
    $(f^*, g^*) = \text{argmin}_{(f,g)\in\mathcal{W}, f\neq g} Q(f,g)$
    Create a new node with index $u$ to join node $f^*$ and $g^*$
    Update distances to the new node $u$ according to Eqs. 6.36
    $\mathcal{W} \leftarrow \mathcal{W} \setminus \{f^*, g^*\} \cup \{u\}$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{u\}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(u, f^*), (u, g^*)\}$
    $u \leftarrow u + 1$
  end while
```

1997). We describe it below[3]:

Let $\mathcal{W}$ be the active working set. Initially $\mathcal{W} = \{X_1, ..., X_O\}$. To decide which two nodes to merge, neighbor joining creates a $\boldsymbol{Q}$ matrix defined as follows:

$$\boldsymbol{Q}(i,j) = (|\mathcal{W}| - 2)d(i,j) - \sum_{k\in\mathcal{W}} d(i,k) - \sum_{k\in\mathcal{W}} d(j,k) \tag{6.33}$$

It then selects the pair of nodes f^*, g^* such that $Q(f^*, g^*)$ is the lowest.

If f^* and g^* are joined to a parent u, compute the distances to u as follows.

$$d(f^*, u) = \frac{1}{2}d(f^*, g^*) + \frac{1}{2(|\mathcal{W}| - 2)}\left[\sum_{k\in\mathcal{W}} d(f^*, k) - \sum_{k\in\mathcal{W}} d(g^*, k)\right] \tag{6.34}$$

$$d(g^*, u) = d(f^*, g^*) - d(f^*, u) \tag{6.35}$$

$$d(k, u) = \frac{1}{2}[d(f^*, k) + d(g^*, k) - d(f^*, g^*)] \;\; \forall k \in \mathcal{W} \setminus \{f^*, g^*\} \tag{6.36}$$

After these distances are computed, f^* and g^* are merged into a node u. f^*, g^* are removed from $\mathcal{W}$ and u is added to $\mathcal{W}$. $\boldsymbol{Q}$ is then recomputed and the algorithm picks another pair of nodes to merge until the tree is complete. The algorithm is shown in Algorithm 11.

[3]We follow the exposition in http://en.wikipedia.org/wiki/Neighbor_joining

Chapter 7

Alternative Spectral Representation of Latent Tree Graphical Models

The spectral representation that we presented in Chapter 4 is not suitable for trees where certain nodes have large numbers of children, because the order of the parameter tensor required is equal to the degree of the associated node. This limits the applicability of this method in practice, a problem that conventional learning methods such as Expectation Maximization (EM) (Dempster et al., 1977) do not face.

Moreover, Mossel and Roch (Mossel and Roch, 2006) also proposed a spectral algorithm for latent variable models which applies to arbitrary tree topologies and only requires tensors of order 3. While they made very restrictive assumptions, and their method does not perform well empirically, it still hints that a more efficient observable representation may be possible.

Contribution of this chapter: We derive an alternate spectral learning algorithm for latent tree graphical models that only requires tensors of order 3 regardless of the topology of the tree. Key to our derivation is the use of an alternate representation for tensor message passing that is more compact. Sample complexity results are provided and empirically we show that this new representation performs favorably than the representation in Chapter 4, especially for trees with larger degree. We also compare with the algorithm of Mossel and Roch (2006) showing our method gives considerably more stable results across a variety of tree topologies. Unfortunately, this representation doesn't extend easily to junction trees or kernel embeddings in Chapters 5 and 6.

Outline: We first provide an example for intuition and then derive the spectral algorithm by representing message passing in a compact tensor form, followed by transforming this representation into one that only depends on observed variables. Finally, we analyze the sample complexity of our method and compare it empirically to the spectral algorithm derived in Chapter 4 and that of Mossel and Roch (2006).

Prerequisites: This chapter assumes a general understanding of latent variable models as presented in 2.2, the connection between latent variable models and low rank factorization in Chapter 3, and the tensor notation in 3.1. It is also recommended to read Chapter 4 first.

7.1 Intuition

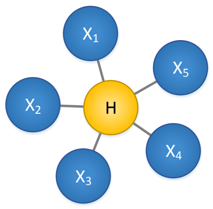


Figure 7.1: Example latent variable model

We first present a small example for intuition. Consider the graphical model in Figure 7.1. The original spectral algorithm in Chapter 4 first proposes the following factorization of the marginal probability tensor $\mathcal{P}(X_1, X_2, X_3, X_4, X_5)$:

$$\mathcal{P}(X_1, X_2, X_3, X_4, X_5) = \mathcal{P}(\oslash_5 H) \times_H \mathcal{P}(X_1|H) \times_H \mathcal{P}(X_2|H) \times_H \mathcal{P}(X_3|H) \times_H \mathcal{P}(X_4|H) \times_H \mathcal{P}(X_5|H)$$

Although $\mathcal{P}(\oslash_5 H)$ is a fifth order tensor, it is diagonal, and therefore very compact. However, the observable representation is:

$$\mathcal{P}(X_1, X_2, X_3, X_4, X_5) = \mathcal{P}(X_1, X_2, X_3, X_4, X_5) \times_{X_1} \boldsymbol{U}_1^T \times_{X_2} \boldsymbol{U}_2^T \times_{X_3} \boldsymbol{U}_3^T \times_{X_4} \boldsymbol{U}_4^T \times_{X_5} \boldsymbol{U}_5^T$$

Now the fifth order tensor is fully dense, which is the central cause of the parameter explosion in the original spectral algorithm.

Let us consider an alternate factorization of the marginal probability tensor. Associate the conditional probability table of each child X_i with the 3rd order labeled tensor $\mathcal{P}(X_i| \oslash_2 H)$. Instead of directly multiplying each of these tensors with the root tensor that is a function of H, we will instead first agglomerate them:

$$\mathcal{P}(X_1, X_2, X_3, X_4, X_5| \oslash_2 H) = \mathcal{P}(X_1| \oslash_2 H) \times_H \mathcal{P}(X_2| \oslash_2 H) \times_H \mathcal{P}(X_3| \oslash_2 H) \times_H \mathcal{P}(X_4| \oslash_2 H) \times_H \mathcal{P}(X_5| \oslash_2 H) \tag{7.1}$$

The $\oslash_2 H$ prevents H from being marginalized out while the tensors are multiplied. We then let the root be associated with a vector $\mathcal{P}(H)$ and recover the marginal probability tensor as follows :

$$\mathcal{P}(H) \times_H (\mathcal{P}(X_1, X_2, X_3, X_4, X_5| \oslash_2 H) \times_H \mathbf{1}) \tag{7.2}$$

This gives us the following tensor representation for the graphical model in Figure 7.1.

$$\begin{aligned} \mathcal{P}(X_1, X_2, X_3, X_4, X_5) = \mathcal{P}(H) \times_H \Big(&\mathcal{P}(X_1| \oslash_2 H) \times_H \mathcal{P}(X_2| \oslash_2 H) \\ &\times_H \mathcal{P}(X_3| \oslash_2 H) \times_H \mathcal{P}(X_4| \oslash_2 H) \\ &\times_H \mathcal{P}(X_5| \oslash_2 H) \times_H \mathbf{1}\Big) \end{aligned} \tag{7.3}$$

Note that the maximum tensor order of the right hand side is 3 since $\mathcal{P}(H)$ is a vector. Moreover, in the following sections we will demonstrate that it is possible to transform this representation so that it only depends on observed variables.

Notation	Definition	Example(from Figure 7.3)
X_{i^*}	some observed node in subtree rooted at X_i	$X_B^* = E$, $X_E^* = E$
T_i	set of observed nodes in subtree rooted at X_i	$T_B = \{E, F\}$, $T_E = \{E\}$
$X_{\rho(i)}$	right sibling of X_i (cyclic)	$X_{\rho(B)} = C$, $X_{\rho(C)} = D$, $X_{\rho(D)} = B$
$X_{\lambda(i)}$	left sibling of X_i (cyclic)[1]	$X_{\lambda(B)} = D$, $X_{\lambda(C)} = B$, $X_{\lambda(D)} = C$

Figure 7.2: Notation for this chapter

7.2 Notation

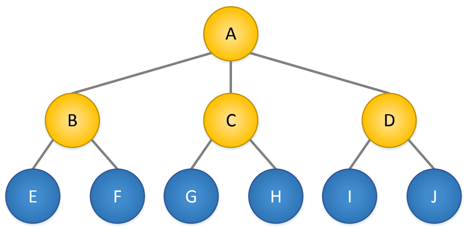


Figure 7.3: Latent tree model with six observed nodes

Before proceeding to present the spectral algorithm, we briefly discuss notation. A latent tree model defines a joint probability distribution over a set of O observed variables $\mathscr{O} = \{X_1, \ldots, X_O\}$ and a set of H hidden variables $\mathscr{H} = \{X_{O+1}, \ldots, X_{O+H}\}$. The complete set of variables is denoted by $\mathscr{X} = \mathscr{O} \cup \mathscr{H}$. For simplicity, we assume that all hidden variables have S_H states and observed variables have S_O states. The joint distribution of $\mathscr{X}$ in a latent tree model is fully characterized by the following equation:

$$P(x_1, \ldots, x_{O+H}) = \prod_{i=1}^{O+H} P(x_i | x_{\pi(i)}). \tag{7.4}$$

where $X_{\pi(i)}$ denotes the parent of X_i. For simplicity, assume that all leaves are observed variables and all internal nodes are latent. For further notation, let X_{i^*} be some observed node in the subtree rooted at X_i. Let $c_j(i)$ denote the j^{th} child of node X_i where X_i has α_i children. For ease of exposition we will assume that all internal nodes have exactly 3 neighbors.

Furthermore, let X_i^* denote some observed node in the subtree rooted at X_i and T_i denote the set of all observed nodes in the subtree rooted at X_i. Let $\lambda(i)$ be the left sibling of X_i and $\rho(i)$ be the right sibling of X_i where the order is cyclic. If X_i only has two children than it can pretend its "uncle" is its left sibling (e.g. set $\lambda(i)$ to a sibling of $X_{\pi(i)}$). A summary of notation is provided in Table 7.2.

7.3 Derivation of Alternate Spectral Algorithm

Our spectral derivation has three main components. First we show how the marginal probability tensor $\boldsymbol{\mathcal{P}}(X_1, \ldots, X_O)$ can be factorized into a collection of lower order tensors where the maximum

[1]if X_i only has one sibling then set to sibling of $X_{\pi(i)}$

tensor order is 3 regardless of the tree topology. This can be shown to be equivalent to a tensor message passing scheme. Secondly, we transform this representation by inserting the invertible transformations $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ to define an alternate low rank factorization that still returns the same probability. Finally, we choose $\boldsymbol{F}$ carefully so that the factors in the transformed representation only depend on observed variables. Figure 7.3 is used as a running example.

7.3.1 Factorizing the Marginal Probability Tensor

In this representation, we associate the root node X_r with the marginal probability (labeled) vector $\boldsymbol{\mathcal{P}}(X_r)$ and non root nodes with third order labeled tensors $\boldsymbol{\mathcal{P}}(X_i| \oslash_2 X_{\pi(i)})$. Consider Figure 7.3. We have that

$$\boldsymbol{\mathcal{P}}(E,F,G,H,I,J) = \boldsymbol{\mathcal{P}}(A) \times_A (\boldsymbol{\mathcal{P}}(E,F| \oslash_2 A) \times_A \boldsymbol{\mathcal{P}}(G,H| \oslash_2 A) \times_A \boldsymbol{\mathcal{P}}(I,J| \oslash_2 A) \times_A \mathbf{1})$$

Decomposing recursively gives

$$\begin{aligned}
\boldsymbol{\mathcal{P}}(E,F| \oslash_2 A) &= P(B| \oslash_2 A) \times_B (\boldsymbol{\mathcal{P}}(E| \oslash_2 B) \times_B \boldsymbol{\mathcal{P}}(F| \oslash_2 B) \times_B \mathbf{1}) \\
\boldsymbol{\mathcal{P}}(G,H| \oslash_2 C) &= P(C| \oslash_2 A) \times_C (\boldsymbol{\mathcal{P}}(G| \oslash_2 C) \times_C \boldsymbol{\mathcal{P}}(H| \oslash_2 C) \times_C \mathbf{1}) \\
\boldsymbol{\mathcal{P}}(I,J| \oslash_2 |D) &= P(D| \oslash_2 A) \times_D (\boldsymbol{\mathcal{P}}(I| \oslash_2 D) \times_B \boldsymbol{\mathcal{P}}(J| \oslash_2 D) \times_D \mathbf{1})
\end{aligned} \tag{7.5}$$

7.3.2 Tensor Message Passing

As in Chapter 4, the recursive factorization procedure described above can be expressed as message passing, a form that will more be notationally convenient for our subsequent derivation. Instead of attempting to reconstruct the entire marginal probability tensor let us simply focus on a single element of this tensor (e.g. $P(\bar{e}, \bar{f}, \bar{g}, \bar{h}, \bar{i}, \bar{j})$). We will associate each leaf with the labeled tensor $\boldsymbol{\mathcal{L}}_i = \boldsymbol{\mathcal{P}}(X_i| \oslash_2 X_{\pi(i)})$. Then, the message that the leaf passes to its parent is:

$$\boldsymbol{M}_i = \boldsymbol{\mathcal{L}}_i \times_i \boldsymbol{\delta}_{\bar{x}_i} \tag{7.6}$$

Example: $\boldsymbol{M}_E = \boldsymbol{\mathcal{P}}(\bar{e}| \oslash_2 B) = \boldsymbol{\mathcal{P}}(E| \oslash_2 B) \times_E \boldsymbol{\delta}_{\bar{e}}$

The internal node, associated with the labeled tensor $\boldsymbol{\mathcal{T}}_i = \boldsymbol{\mathcal{P}}(X_i| \oslash_2 X_{\pi(i)})$ agglomerates the messages from its children and passes the resulting (diagonal) matrix to its parent (where $\mathbf{1}_i$ is a S_H dimensional vector of ones):

$$\boldsymbol{M}_i = \boldsymbol{\mathcal{T}}_i \times_i \left(\boldsymbol{M}_{c_1(i)} \boldsymbol{M}_{c_2(i)} \mathbf{1}_i\right) \tag{7.7}$$

Example: $\boldsymbol{M}_B = \boldsymbol{\mathcal{P}}(\bar{e}, \bar{f}| \oslash_2 A) = \boldsymbol{\mathcal{P}}(B| \oslash_2 A) \times_B (\boldsymbol{M}_E \boldsymbol{M}_F \mathbf{1}_B)$

Finally the root parameter is represented as a labeled vector: $\boldsymbol{r} := \boldsymbol{\mathcal{P}}(X_r)$. Combining this with the messages from the children gives the probability estimate:

$$P(\bar{x}_1, ..., \bar{x}_O) = \boldsymbol{r}^\top \left(\boldsymbol{M}_{c_1(r)} \boldsymbol{M}_{c_2(r)} \boldsymbol{M}_{c_3(r)} \mathbf{1}_r\right) \tag{7.8}$$

Example: $P(\bar{e}, \bar{f}, \bar{g}, \bar{h}, \bar{i}, \bar{j}) = \boldsymbol{r}^\top \boldsymbol{M}_B \boldsymbol{M}_C \boldsymbol{M}_D \mathbf{1}$.

Note that unlike the tensor representation in Chapter 4, the message now takes the form of a matrix $\boldsymbol{M}_i$ instead of a vector $\boldsymbol{m}_i$. Furthermore, the $\mathbf{1}_i$ is needed unlike in the original tensor representation.

7.3.3 Transformed Representation

Next, as in Chapter 4, note we do not need to recover the tensor representation explicitly if our focus is to perform inference using the message passing algorithm as in (7.6)–(7.8). As long as we can recover the tensor representation up to some invertible transformation, we can still obtain the correct marginal probability $P(\bar{x}_1, ..., \bar{x}_O)$.

More specifically, we can insert a identity matrix $\boldsymbol{I}$ into the message update equation in (7.8) without changing the final probability.

$$\begin{aligned} P(\bar{e}, \bar{f}, \bar{g}, \bar{h}, \bar{i}, \bar{j}) &= \boldsymbol{r}^\top \boldsymbol{M}_B \boldsymbol{M}_C \boldsymbol{M}_D \mathbf{1}_A \\ &= \boldsymbol{r}^\top \times \boldsymbol{I} \times \boldsymbol{M}_B \times \boldsymbol{I} \times \boldsymbol{M}_C \times \boldsymbol{I} \times \boldsymbol{M}_D \times \boldsymbol{I} \times \mathbf{1} \end{aligned}$$

where $\boldsymbol{I}$ is the $S_H \times S_H$ identity matrix.

Subsequently, we can then replace this matrix with a pair of matrices $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$, that are inverses of each other, and then regroup the terms:

$$P(\bar{e}, \bar{f}, \bar{g}, \bar{h}, \bar{i}, \bar{j}) = \boldsymbol{r}^\top \boldsymbol{F}_{A,E} \times_{\omega_A} \left((\boldsymbol{F}_{A,B}^{-1} \times \boldsymbol{M}_B \times \boldsymbol{F}_{A,C}) \times (\boldsymbol{F}_{A,C}^{-1} \boldsymbol{M}_C \boldsymbol{F}_{A,D}) \times (\boldsymbol{F}_{A,D}^{-1} \boldsymbol{M}_D \boldsymbol{F}_{A,B}) \times (\boldsymbol{F}_{A,B}^{-1} \mathbf{1}_A) \right) \quad (7.9)$$

where ω_A is a mode label that will be defined in the next section since it depends on the definition of $\boldsymbol{F}$. These process proceeds recursively e.g.

$$\boldsymbol{M}_B = \boldsymbol{\mathcal{P}}(B| \oslash_2 A) \times_B \boldsymbol{F}_{B,E} \times_{\omega_B} \left((\boldsymbol{F}_{B,E}^{-1} \boldsymbol{M}_E \boldsymbol{F}_{B,F}) \times (\boldsymbol{F}_{B,F}^{-1} \boldsymbol{M}_F \boldsymbol{F}_{B,E}) \times (\boldsymbol{F}_{B,E}^{-1} \mathbf{1}_B) \right)$$

In general, we can define the following transformed tensor representation:

- **root**: $\boldsymbol{F}_{i,c_1(i)}^\top \boldsymbol{r}$
- **internal**: $\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{T}}_i \times_i \boldsymbol{F}_{i,c_1(i)} \times_{\pi(i)} \boldsymbol{F}_{\pi(i),i}^{-1} \times_{\pi(i)} \boldsymbol{F}_{\pi(i),\rho(i)}$
- **leaf**: $\widetilde{\boldsymbol{\mathcal{L}}}_i = \boldsymbol{\mathcal{L}}_i \times_{\pi(i)} \boldsymbol{F}_{\pi(i),i}^{-1} \times_{\pi(i)} \boldsymbol{F}_{\pi(i),\rho(i)}$
- **one**: $\tilde{\mathbf{1}}_i = \boldsymbol{F}_{i,c_1(i)}^{-1} \mathbf{1}_i$

7.4 Observable Representation

We now derive the observable representation by choosing $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ systematically, so that we can recover each transformed parameter using the marginal probability of a small set of observed variables. This is summarized by the lemma below:

Lemma 16. *Define $\boldsymbol{F}_{i,j} = \boldsymbol{\mathcal{P}}(X_{j^*}|X_i)^\top$ with mode labels $\{X_i, \omega_i := X_{j^*}\}$.*

- ***root***: $\tilde{\boldsymbol{r}} = \boldsymbol{\mathcal{P}}(X_{c_1(r)^*})$
- ***internal***: $\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{\lambda(i)^*}, X_{\rho(i)^*}) \times_{\lambda(i)^*} \boldsymbol{\mathcal{P}}(X_{i^*}, X_{\lambda(i)^*})^{-1}$
- ***leaf***: $\widetilde{\boldsymbol{\mathcal{L}}}_i = \boldsymbol{\mathcal{P}}(X_i, X_{\lambda(i)^*}, X_{\rho(i)^*}) \times_{\lambda(i)^*} \boldsymbol{\mathcal{P}}(X_i, X_{\lambda(i)^*})^{-1}$
- ***one***: $\tilde{\mathbf{1}} = \boldsymbol{\mathcal{P}}(X_{\lambda(c_1(i))^*}, X_{c_1(i)^*})^{-1}\boldsymbol{\mathcal{P}}(X_{\lambda(c_1(i))^*})$

Proof. The derivation for the root is straightforward:

$$\tilde{\boldsymbol{r}} = \boldsymbol{\mathcal{P}}(X_i) \times_i \boldsymbol{F}_{i,c_1(r)} = \boldsymbol{\mathcal{P}}(X_i) \times_i \boldsymbol{\mathcal{P}}(X_{c_1(r)^*}|X_i) = \boldsymbol{\mathcal{P}}(X_{c_1(r)^*})$$

Example (Figure 7.3): $X^*_{c_1(r)} \in \{E, F\}$ so one possibility is $\tilde{\boldsymbol{r}} = \boldsymbol{\mathcal{P}}(E)$.

For the internal node, first consider the quantity, $\boldsymbol{\mathcal{P}}(X_{i^*}, X_{\lambda(i)^*})$. This can be expanded into,

$$\boldsymbol{\mathcal{P}}(X_{i^*}, X_{\lambda(i)^*}) = \boldsymbol{\mathcal{P}}(X^*_i|X_{\pi(i)})\boldsymbol{\mathcal{P}}(\oslash_2 X_{\pi(i)})\boldsymbol{\mathcal{P}}(X_{\lambda(i)^*}|X_{\pi(i)})^\top \tag{7.10}$$

Combining this with the definition of $\widetilde{\boldsymbol{\mathcal{T}}}_i$ gives

$$\begin{aligned} & \widetilde{\boldsymbol{\mathcal{T}}}_i \times_{\lambda(i)^*} \boldsymbol{\mathcal{P}}(X_{i^*}, X_{\lambda(i)^*}) \\ = \;& \widetilde{\boldsymbol{\mathcal{T}}}_i \times_{\lambda(i)^*} \left(\boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})\boldsymbol{\mathcal{P}}(\oslash_2 X_{\pi(i)})\boldsymbol{\mathcal{P}}(X_{\lambda(i)^*}|X_{\pi(i)})^\top\right) \\ = \;& \boldsymbol{\mathcal{P}}(X_i| \oslash_2 X_{\pi(i)}) \times_i \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}|X_i) \times_{\pi(i)} \boldsymbol{\mathcal{P}}(X_{\rho(i)^*}|X_{\pi(i)}) \times_{X_{\pi(i)}} \left(\boldsymbol{\mathcal{P}}(\oslash_2 X_{\pi(i)})\boldsymbol{\mathcal{P}}(X_{\lambda(i)^*}|X_{\pi(i)})^\top\right) \\ = \;& \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{\rho(i)^*}, X_{\lambda(i)^*}) \end{aligned} \tag{7.11}$$

We can thus conclude that

$$\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{\rho(i)^*}, X_{\lambda(i)^*}) \times_{\lambda(i)^*} \boldsymbol{\mathcal{P}}(X_{i^*}, X_{\lambda(i)^*})^{-1} \tag{7.12}$$

Example (Figure 7.3): For the internal node B, we can set $X^*_i = X_{c_1(i)^*} \in \{E, F\}$, $X_{\rho(i)^*} = \{G, H\}$, $X_{\lambda(i)^*} = \{I, J\}$. This would give $\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{P}}(E, G, I) \times_{\lambda(i)^*} \boldsymbol{\mathcal{P}}(E, I)^{-1}$.

The leaf is just a special case of the internal node. For the $\tilde{\mathbf{1}}_i$ term, multiply by $\boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{\lambda(c_1(i))^*})$:

$$\begin{aligned} \tilde{\mathbf{1}}_i^\top \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{\lambda(c_1(i))^*}) &= \tilde{\mathbf{1}}_i^\top\left(\boldsymbol{\mathcal{P}}(X_{c_1(i)^*}|X_i)\boldsymbol{\mathcal{P}}(\oslash_2 X_i)\boldsymbol{\mathcal{P}}(X_{\lambda(c_1(i))^*}|X_i)^\top\right) \\ &= \mathbf{1}^\top\left(\boldsymbol{\mathcal{P}}(\oslash_2 X_i)\boldsymbol{\mathcal{P}}(X_{\lambda(c_1(i))^*}|X_i)^\top\right) \\ &= \boldsymbol{\mathcal{P}}(X_{\lambda(c_1(i))^*})^{(\top)} \end{aligned} \tag{7.13}$$

leading us to conclude that

$$\tilde{\mathbf{1}} = \boldsymbol{\mathcal{P}}(X_{\lambda(c_1(i))^*}, X_{c_1(i)^*})^{-1}\boldsymbol{\mathcal{P}}(X_{\lambda(c_1(i))^*}) \tag{7.14}$$

Example (Figure 7.3): For $\tilde{\mathbf{1}}_A$, $X_{c_1(i)^*} \in \{E, F\}$, $X_{\lambda(c_1(i))^*} \in \{I, J\}$, so $\tilde{\mathbf{1}} = \boldsymbol{\mathcal{P}}(I, E)^{-1}\boldsymbol{\mathcal{P}}(I)$. □

Algorithm 12 shows the algorithm for training for the example in Figure 7.3 when $S_H = S_O$.

Akin to the algorithm in Chapter 4, if $S_O > S_H$ then $F_{i,j} = \boldsymbol{U}_j^\top \boldsymbol{\mathcal{P}}(X_{j^*}|X_i)$ instead of $F_{i,j} = \boldsymbol{\mathcal{P}}(X_{j^*}|X_i)$

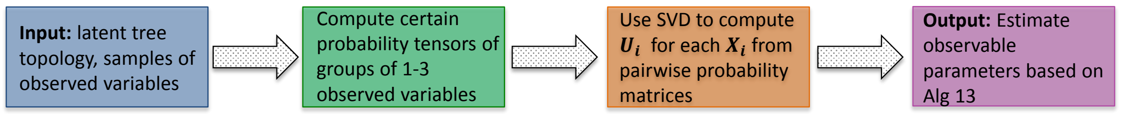

Input: latent tree topology, samples of observed variables
Compute certain probability tensors of groups of 1-3 observed variables
Use SVD to compute $\boldsymbol{U_i}$ for each $\boldsymbol{X_i}$ from pairwise probability matrices
Output: Estimate observable parameters based on Alg 13

Figure 7.4: Flowchart that gives an overview of Algorithm 13

where $\boldsymbol{U}_j$ is set to the top S_H left singular vectors of $\mathcal{P}(X_{j^*}, X_{\lambda(j)^*})$. The more general algorithms for training and testing are given in Algorithms 13 and 14. Figure 7.4 gives a high level flowchart for the general training procedure.

7.5 Sample Complexity

We analyze the sample complexity of Algorithm 13 and find that it depends on the tree topology and the spectral properties of the true model. See the Appendix for a proof.

Algorithm 12 Spectral learning algorithm for example in Figure 7.3 when $S_H = S_O$

In: Tree topology and N *i.i.d.* samples of E, F, G, H, I, J

Out: Estimated observable root, internal, leaf, and **1** parameters, $\widehat{\mathcal{R}}_A, \widehat{\mathcal{T}}_B, \widehat{\mathcal{T}}_C, \widehat{\mathcal{T}}_D, \widehat{\mathcal{L}}_E, \widehat{\mathcal{L}}_F, \widehat{\mathcal{L}}_G, \widehat{\mathcal{L}}_H, \widehat{\mathcal{L}}_I, \widehat{\mathcal{L}}_J, \widehat{\mathbf{1}}_A, \widehat{\mathbf{1}}_B, \widehat{\mathbf{1}}_C, \widehat{\mathbf{1}}_D$

```
1: Compute the following probability tensors:
$$\widehat{\mathcal{P}}(E), \widehat{\mathcal{P}}(G), \widehat{\mathcal{P}}(I),$$
$$\widehat{\mathcal{P}}(E,I,G), \widehat{\mathcal{P}}(G,E,I), \widehat{\mathcal{P}}(I,G,E), \widehat{\mathcal{P}}(E,G,F), \widehat{\mathcal{P}}(G,I,H), \widehat{\mathcal{P}}(I,E,J),$$
$$\widehat{\mathcal{P}}(E,I), \widehat{\mathcal{P}}(E,G), \widehat{\mathcal{P}}(I,G),$$
$$\widehat{\mathcal{P}}(F,G), \widehat{\mathcal{P}}(H,I), \widehat{\mathcal{P}}(J,E)$$
2: Compute observable parameters as:
$$\widehat{\mathcal{R}}_A = \widehat{\mathcal{P}}(E)$$
$$\widehat{\mathcal{T}}_B = \widehat{\mathcal{P}}(E,I,G) \times_I (\widehat{\mathcal{P}}(E,I))^{-1}$$
$$\widehat{\mathcal{T}}_C = \widehat{\mathcal{P}}(G,E,I) \times_E (\widehat{\mathcal{P}}(G,E))^{-1}$$
$$\widehat{\mathcal{T}}_D = \widehat{\mathcal{P}}(I,G,E) \times_G (\widehat{\mathcal{P}}(I,G))^{-1}$$
$$\widehat{\mathcal{L}}_E = \widehat{\mathcal{P}}(E,G,F) \times_G (\widehat{\mathcal{P}}(E,G))^{-1}$$
$$\widehat{\mathcal{L}}_F = \widehat{\mathcal{P}}(F,G,E) \times_G (\widehat{\mathcal{P}}(F,G))^{-1}$$
$$\widehat{\mathcal{L}}_G = \widehat{\mathcal{P}}(G,I,H) \times_I (\widehat{\mathcal{P}}(G,I))^{-1}$$
$$\widehat{\mathcal{L}}_H = \widehat{\mathcal{P}}(H,I,G) \times_I (\widehat{\mathcal{P}}(H,I))^{-1}$$
$$\widehat{\mathcal{L}}_I = \widehat{\mathcal{P}}(I,E,J) \times_E (\widehat{\mathcal{P}}(I,E))^{-1}$$
$$\widehat{\mathcal{L}}_J = \widehat{\mathcal{P}}(J,E,I) \times_E (\widehat{\mathcal{P}}(J,E))^{-1}$$
$$\widehat{\mathbf{1}}_A = (\widehat{\mathcal{P}}(I,E))^{-1}\mathcal{P}(I)$$
$$\widehat{\mathbf{1}}_B = (\widehat{\mathcal{P}}(G,E))^{-1}\mathcal{P}(G)$$
$$\widehat{\mathbf{1}}_C = (\widehat{\mathcal{P}}(I,G))^{-1}\mathcal{P}(I)$$
$$\widehat{\mathbf{1}}_D = (\widehat{\mathcal{P}}(E,I))^{-1}\mathcal{P}(E)$$
```

Algorithm 13 Alternate spectral learning algorithm for latent tree graphical model

In: Tree topology and N *i.i.d.* samples $\left\{x_1^{(1)}, \ldots, x_O^{(n)}\right\}_{n=1}^{N}$

Out: Estimated observable parameters $\widehat{\boldsymbol{r}}$ for root, $\widehat{\boldsymbol{\mathcal{T}}}_i$ for each non-root internal node, $\widehat{\boldsymbol{\mathcal{L}}}_i$ for each leaf, $\widehat{\mathbf{1}}_i$ for each internal node

```
1: For each node $X_j$, perform a "thin" singular value decomposition of $\widehat{\boldsymbol{\mathcal{P}}}(X_j^*, X_{\Lambda(j)^*}) = \boldsymbol{U\Sigma V}^\top$; let $\widehat{\boldsymbol{U}}_j = \boldsymbol{U}(:, 1:S_H)$ be
   the the first $S_H$ principal left singular vectors.
2: Estimate $\tilde{\boldsymbol{r}}$, and each $\tilde{\boldsymbol{\mathcal{T}}}_i$, $\widehat{\boldsymbol{\mathcal{L}}}_i$, $\tilde{\mathbf{1}}_i$ via
```

$$\widehat{\boldsymbol{r}} = \widehat{\boldsymbol{U}}_r^\top \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(r)^*}) \tag{7.15}$$

$$\widehat{\boldsymbol{\mathcal{T}}}_i = \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(i)^*}, X_{\Lambda(i)^*}, X_{\rho(i)^*}) \times_{\Lambda(i)^*} \left(\widehat{\boldsymbol{U}}_i^\top \widehat{\boldsymbol{\mathcal{P}}}(X_{i^*}, X_{\Lambda(i)^*})\right)^\dagger \times_{c_1(i)^*} \widehat{\boldsymbol{U}}_{c_1(i)} \times_{\rho(i)^*} \widehat{\boldsymbol{U}}_{\rho(i)} \tag{7.16}$$

$$\widehat{\boldsymbol{\mathcal{L}}}_i = \widehat{\boldsymbol{\mathcal{P}}}(X_i, X_{\Lambda(i)^*}, X_{\rho(i)^*}) \times_{\Lambda(i)^*} \left(\widehat{\boldsymbol{U}}_i^\top \widehat{\boldsymbol{\mathcal{P}}}(X_{i^*}, X_{\Lambda(i)^*})\right)^\dagger \times_{\rho(i)^*} \widehat{\boldsymbol{U}}_{\rho(i)} \tag{7.17}$$

$$\widehat{\mathbf{1}}_i = \left(\widehat{\boldsymbol{\mathcal{P}}}(X_{\Lambda(c_1(i))^*}, X_{c_1(i)^*})\widehat{\boldsymbol{U}}_{c_1(i)}\right)^\dagger \widehat{\boldsymbol{\mathcal{P}}}(X_{\Lambda(c_1(i))^*}) \tag{7.18}$$

Theorem 3. *For any $\epsilon > 0, 0 < \delta < 1$, let*

$$N \geq O\left(\frac{(\alpha_{max} S_H)^{2\ell+1}}{\gamma\beta\epsilon^2}\right) \log \frac{|\mathscr{O}|}{\delta}$$

where $\alpha_{max} = \max_i \alpha_i$, $\sigma_{S_H}(*)$ returns the S_H^{th} largest singular value and

$$\gamma = \min_{i \neq j, i, j \in \mathscr{O}} \ \sigma_{S_H}(\boldsymbol{\mathcal{P}}(X_i, X_j))^4$$
$$\beta = \min_{i \in \mathscr{O}} \ \sigma_{S_H}(\boldsymbol{\mathcal{P}}(X_i | X_{\pi(i)}))^2$$

Then $\sum_{x_1, \ldots, x_O} \left|\widehat{P}(x_1, \ldots, x_O) - P(x_1, \ldots, x_O)\right| \leq \epsilon$ with probability $1 - \delta$.

This result implies that the estimation problem gets harder as the maximum degree $\alpha_{\max}$ of the hidden nodes, the number S_H of the hidden states, and the length ℓ of the chain of hidden variables increase. Furthermore, the sample complexity depends exponentially in ℓ, which suggests that we should choose the root of the tree to make ℓ small. However, we believe that such adverse dependence on ℓ is due to the artifact of our analysis.

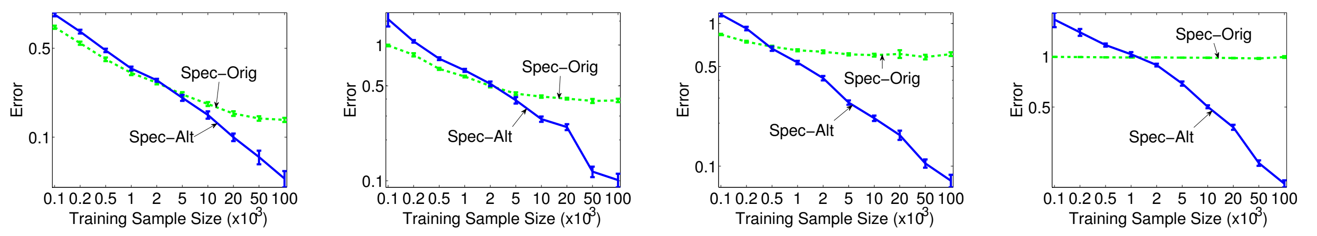
A special case of latent tree models is hidden Markov models (HMMs). Recently, (Hsu et al., 2009)

Algorithm 14 Inference with Alternate Spectral Parameters

In: Tree topology, set of spectral parameters $\widehat{\boldsymbol{r}}$, $\widehat{\boldsymbol{\mathcal{T}}}_i$ for each non-root internal node, $\widehat{\boldsymbol{\mathcal{L}}}_i$ for each leaf node, $\widehat{\mathbf{1}}_i$ for each internal node, and set of evidence $\mathcal{E} = \{\bar{x}_{e_1}, ..., \bar{x}_{e_{|\mathcal{E}|}}\}$

Out: estimated probability $\widehat{P}(\bar{x}_{e_1}, ..., \bar{x}_{|\mathcal{E}|})$

```
1: In reverse topological order, each node accumulates at message at leaf and sends to parent
     • Evidence Leaf: $\widehat{\boldsymbol{M}}_i = \widehat{\boldsymbol{\mathcal{L}}}_i \times_i \boldsymbol{\delta}_{\bar{x}_i}$
     • Non-Evidence Leaf: $\widehat{\boldsymbol{M}}_i = \widehat{\boldsymbol{\mathcal{L}}}_i \times_i \mathbf{1}$
     • Internal Node: $\widehat{\boldsymbol{M}}_i = \widehat{\boldsymbol{\mathcal{T}}}_i \times_i \left(\widehat{\boldsymbol{M}}_{c_1(i)} \widehat{\boldsymbol{M}}_{c_2(i)} \widehat{\mathbf{1}}_i\right)$
     • Root: $\widehat{P}(\bar{x}_{e_1}, ..., \bar{x}_{|\mathcal{E}|}) = \widehat{\boldsymbol{r}}^\top \left(\widehat{\boldsymbol{M}}_{c_1(i)} \widehat{\boldsymbol{M}}_{c_2(i)} \widehat{\boldsymbol{M}}_{c_3(i)} \widehat{\mathbf{1}}_i\right)$
```



(a) NumChildren=3, Depth=3 (b) NumChildren=3, Depth=4 (c) NumChildren=4, Depth=3 (d) NumChildren=4, Depth=4

Figure 7.5: Comparison of original spectral algorithm (green) to alternate spectral algorithm (blue) as a function of degree and depth

derived a spectral algorithm specific to HMMs. Their reasoning relies heavily on a single connected chain of hidden variables and each hidden variable has an observed variable attached. Although this excludes many interesting tree topologies, they obtained a tighter sample complexity bound which is $O(\frac{S_H \ell^2}{\gamma \beta \epsilon^2})$ (polynomial in ℓ). This also suggests that our analysis can be further improved.

7.6 Experiments

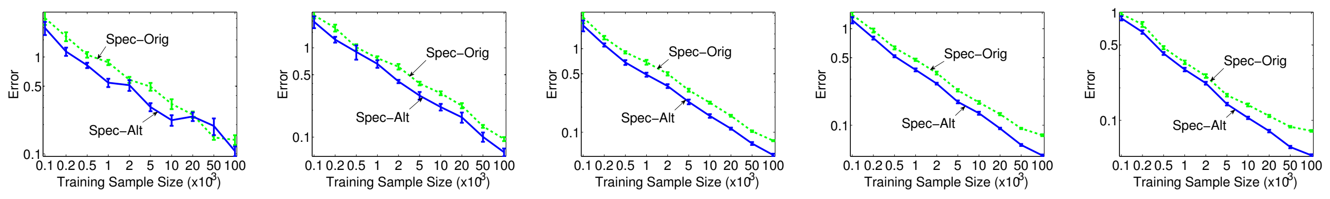
In this section, we empirically evaluate this new algorithm. We first empirically compare to the traditional spectral algorithm described in Chapter 4. For both methods we use a linear system size of 1. Since both methods have similar runtime, we only focus on accuracy, particularly as the topology of the tree and nature of the parameters are changed. We measure the performance of joint estimation using $\epsilon = \frac{|\widehat{P}(x_1,...,x_O) - P(x_1,...,x_O)|}{P(x_1,...,x_O)}$ averaged over 1000 test points randomly drawn from the underlying model. *Spec-Orig* or *original spectral algorithm* denotes the spectral method from Chapter 4 while *Spec-Alt* or *alternate spectral algorithm* denotes the spectral method presented in this chapter.

The first set of experiments (Figure 7.5) shows the performance of the methods as the topology of the tree is varied. S_O and S_H are set to 6 and 2 respectively. In all cases the tree is balanced and NumChildren refers to the number of children each internal node has, and Depth indicates the depth of the tree. As one can see for lower sample sizes, the original spectral algorithm is more robust, while as the sample size increases, the alternate algorithm consistently performs better.

The next two sets of experiments measure performance as positive values are added to the diagonals of the conditional probability tables (before normalization) thus increasing the singular values of the resulting conditional probability tables after normalization. For these experiments S_O is set to 4 and S_H is set to 3. Figure 7.6 shows the effect of this for a tree with NumChildren set to 3 (every internal node has 3 children) and depth also fixed to 3 while Figure 7.7 shows the effect of increasing diagonals for NumChildren=4 and Depth=3 (every internal node has 4 children). In both cases, it appears both methods benefit similarly from the increased singular values.

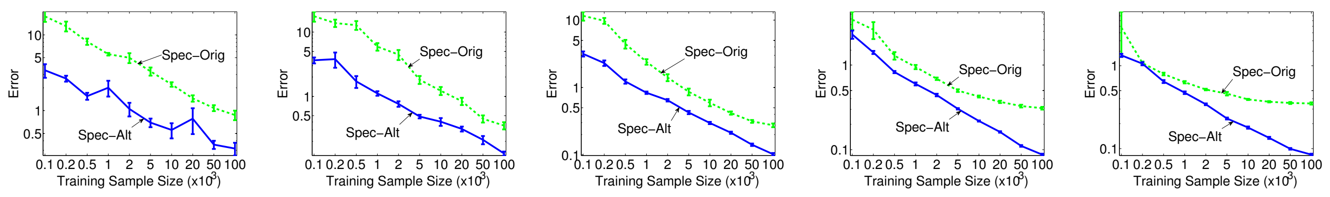
7.6.1 Comparison with Mossel and Roch Algorithm

We now make a separate comparison with the spectral algorithm by (Mossel and Roch, 2006), since it only applies to case where the number of observed states S_O is the same as the number of hidden



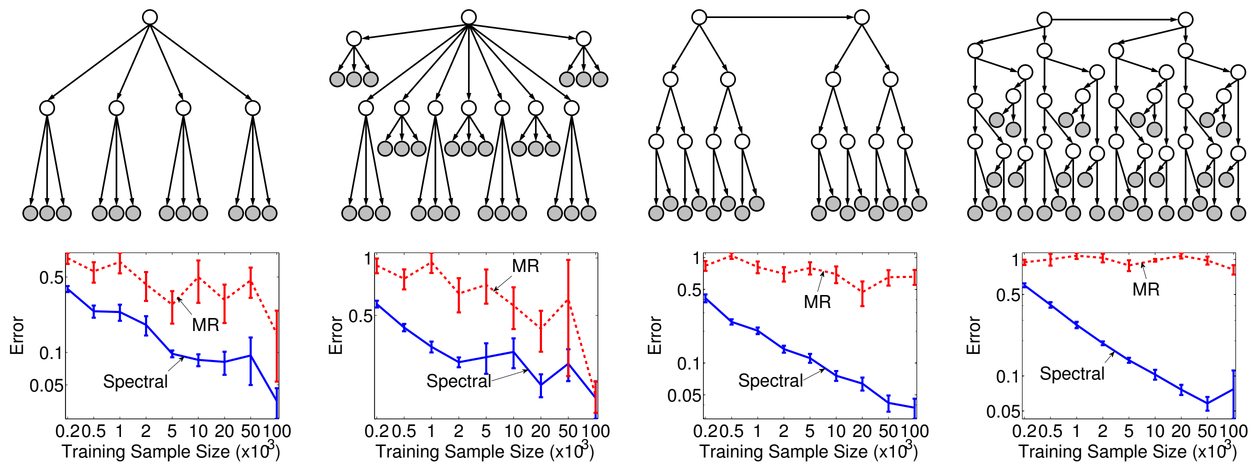
(a) DiagAdd = 0 (b) DiagAdd = 1 (c) DiagAdd = 2 (d) DiagAdd = 5 (e) DiagAdd = 10

Figure 7.6: Comparison of original spectral algorithm (green) to alternate spectral algorithm (blue) for NumChildren=3, Depth=3 for increasingly diagonally dominant conditional probability tables (DiagAdd indicates how much was added to the diagonal before normalization)



(a) DiagAdd = 0 (b) DiagAdd = 1 (c) DiagAdd = 2 (d) DiagAdd = 5 (e) DiagAdd = 10

Figure 7.7: Comparison of original spectral algorithm (green) to alternate spectral algorithm (blue) for NumChildren=4, Depth=3 for increasingly diagonally dominant conditional probability tables (DiagAdd indicates how much was added to the diagonal before normalization)



(a) broad4 (b) broad9 (c) deep4 (d) deep5

Figure 7.8: Comparison of our alternate spectral algorithm (Spectral) with the Mossel and Roch algorithm (MR) for 4 different latent tree topologies. The errors are plotted in log scale.

states S_H. We set $S_O = S_H = 2$ and use a variety of topologies shown in Figure 7.8. Although this method is theoretically interesting, it can perform poorly in practice.

The results are shown in Figure 7.8 (the runtime of both methods are similar, and thus not reported). Our spectral algorithm significantly outperforms the MR algorithm on all trees for practically all sample sizes. This is because our method does not explicitly recover the CPTs, and is thus more robust. We also note that our approach is more general: it can allow for the observation state space to be larger than the hidden state space, which may be preferable in many applications

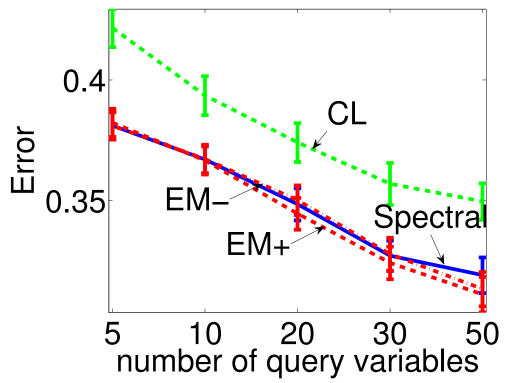


Figure 7.9: Comparison of our spectral algorithm (Spectral, blue line) with EMs (EM+ and EM-, red lines) and Chow-Liu based algorithm (CL, green line) on stock dataset.

where the observation space can be large (*e.g.*, quantization of a continuous variable), but the hidden factors are simple and have lower dimensions.

7.6.2 Stock Trend Prediction

Finally, we evaluate our algorithm on a stock trend prediction problem. Our goal is to predict whether a stock X_i will go up or down on a particular day given the trends of a set $\mathscr{E}$ of other stocks. We acquired closing prices of 59 stocks from 1984 to 2011, which provides us 6800 samples.[2] We randomly partition these samples to 6300 training points and 500 test points. Since we are only predicting whether a stock goes up or down, the data are binarized. From the training data, we learn the latent tree topology using an algorithm by (Choi et al., 2010), and a fully observable Chow-Liu tree (Chow and Liu, 1968b).

We compare our spectral algorithm to *EM+* (high precision EM) and *EM−* (low precision EM) using the latent tree, and with inference over the Chow-Liu tree (*CL*). For the prediction task, we need to estimate the conditional, *i.e.*, $P(X_i|x_{j_1}, \ldots, x_{j_{|\mathscr{E}|}})$ and $j_1, \ldots, j_{|\mathscr{E}|} \in \mathscr{E}$. This can be achieved by estimating $P(x_i, x_{j_1}, \ldots, x_{j_{|\mathscr{E}|}})$ for each instantiation x_i. Then we make prediction by $\hat{x}_i = \operatorname{argmax}_{x_i} P(x_i, x_{j_1}, \ldots, x_{j_{|\mathscr{E}|}})$. We measure the prediction error using $\epsilon = |\hat{x}_i - x_i^\star|$ where $x_i^\star$ is the true label.

We experiment with a varying number of query sizes. For each query size Q, we randomly pick Q stocks and predict the value of one stock conditioned on the other $Q - 1$, (and repeat for 50 trials). The results are shown in Figure 7.9. In general, all methods get better as we increase the number of evidence. Over the entire range of query sizes, the advantage of latent tree approaches (Spectral, EM+/-) is clear over the Chow-Liu tree. Thus, the latent factors help better model the stock data in this case. Due to the small training sample size, the distinction between our method and EM is less clear.

[2]www.finance.yahoo.com

7.7 Discussion

In this chapter an alternate observable representation for latent tree graphical models. This representation is considerably different than the one in Chapter 4 and doesn't reduce to it even when the tree is binary. The distinct characteristic of this representation is that the internal/leaf nodes are associated with a 3rd order tensor where the parent variable is on the diagonal. This feature makes it challenging to generalize this factorization to the kernel case where the analogous operator is challenging to define. Similarly, since different children of a clique in a junction tree may be associated with different variables in the parent clique, it also makes it difficult to generalize this approach to the junction tree scenario.

Empirically we show that while this new representation performs slightly worse than the representation in Chapter 4 for small sample sizes, it does considerably better in mid/large sample sizes, especially for trees with larger degree.

7.8 Appendix

The purpose of this section is to prove Theorem 3. In general, for simplicity of exposition, we assume that all internal nodes in the tree are unobserved, and all leaves are observed (since this is the hardest case). The proof generally follows the technique of HKZ (Hsu et al., 2009), but has key differences due to the tree topology instead of the HMM.

7.8.1 Notation for Proof

To make the notation less cumbersome, the notation for the proof will differ from that of the main chapter in a few ways. In particular, for the sample complexity result it doesn't matter the exact variables that are used to construct the observable representation. For example, recall the observable representation estimates:

$$
\begin{aligned}
\widehat{\boldsymbol{r}} &= \widehat{\boldsymbol{\mathcal{U}}}_r^\top \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(r)^*}) \\
\widehat{\boldsymbol{\mathcal{T}}}_i &= \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(i)^*}, X_{\lambda(i)^*}, X_{\rho(i)^*}) \times_{\lambda(i)^*} \left(\widehat{\boldsymbol{\mathcal{U}}}_i^\top \widehat{\boldsymbol{\mathcal{P}}}(X_{i^*}, X_{\lambda(i)^*})\right)^\dagger \times_{c_1(i)^*} \widehat{\boldsymbol{\mathcal{U}}}_{c_1(i)} \times_{\rho(i)^*} \widehat{\boldsymbol{\mathcal{U}}}_{\rho(i)} \\
\widehat{\boldsymbol{\mathcal{L}}}_i &= \widehat{\boldsymbol{\mathcal{P}}}(X_i, X_{\lambda(i)^*}, X_{\rho(i)^*}) \times_{\lambda(i)^*} \left(\widehat{\boldsymbol{\mathcal{U}}}_i^\top \widehat{\boldsymbol{\mathcal{P}}}(X_{i^*}, X_{\lambda(i)^*})\right)^\dagger \times_{\rho(i)^*} \widehat{\boldsymbol{\mathcal{U}}}_{\rho(i)} \\
\widehat{\mathbf{1}}_i &= \left(\widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(i)^*}, X_{\lambda(c_1(i))^*}) \widehat{\boldsymbol{\mathcal{U}}}_{c_1(i)}\right)^\dagger \widehat{\boldsymbol{\mathcal{P}}}(X_{\lambda(c_1(i))^*})
\end{aligned}
$$

For the proof, we will instead be using the simpler notation:

$$\begin{aligned}
\widehat{\boldsymbol{r}} &= \widehat{\boldsymbol{U}}^\top \widehat{\boldsymbol{\mathcal{P}}}_j \\
\widehat{\boldsymbol{\mathcal{T}}}_i &= \widehat{\boldsymbol{\mathcal{P}}}_{j,k,l} \times_k \left(\widehat{\boldsymbol{U}}_2^\top \widehat{\boldsymbol{\mathcal{P}}}_{m,k}\right)^\dagger \times_j \widehat{\boldsymbol{U}}_1 \times_l \widehat{\boldsymbol{U}}_3 \\
\widehat{\boldsymbol{\mathcal{L}}}_i &= \widehat{\boldsymbol{\mathcal{P}}}_{j,k,l} \times_k \left(\widehat{\boldsymbol{U}}_2^\top \widehat{\boldsymbol{\mathcal{P}}}_{m,k}\right)^\dagger \times_l \widehat{\boldsymbol{U}}_3 \\
\widehat{\mathbf{1}}_i &= \left(\widehat{\boldsymbol{\mathcal{P}}}_{j,k} \widehat{\boldsymbol{U}}\right)^\dagger \widehat{\boldsymbol{\mathcal{P}}}_j
\end{aligned}$$

Here the j, k, l, m refer to variable indices and the $\boldsymbol{U}$'s are just arbitrarily numbered to distinguish them within the same equation. For ease of exposition, multiplying by $\boldsymbol{U}$ does not change the mode label, so for example $\widehat{\boldsymbol{\mathcal{T}}}_i$ has mode labels $\{j, m, l\}$.

Furthermore, recall in Section 7.4 we defined $\boldsymbol{F}_{i,j} = \boldsymbol{\mathcal{P}}(X_{j^*}|X_i)^\top \boldsymbol{U}$. This will simply become $\boldsymbol{F} = \boldsymbol{O}^\top \boldsymbol{U}$ where $\boldsymbol{O} = \boldsymbol{\mathcal{P}}(X_{j^*}|X_i)$. Thus using this notation, we can rewrite the transformed representation in Section 7.3.3 as:

- **root**: $\tilde{\boldsymbol{r}} = (\boldsymbol{U}^\top \boldsymbol{O})\boldsymbol{r}$
- **internal**: $\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{T}}_i \times_i (\boldsymbol{O}_1^\top \boldsymbol{U}_1) \times_{\pi(i)} (\boldsymbol{O}_2^\top \boldsymbol{U}_2)^{-1} \times_{\pi(i)} (\boldsymbol{O}_3^\top \boldsymbol{U}_3)$
- **leaf**: $\widetilde{\boldsymbol{\mathcal{L}}}_i = \boldsymbol{\mathcal{L}}_i \times_k (\boldsymbol{O}_2^\top \boldsymbol{U}_2)^{-1} \times_l (\boldsymbol{O}_3^\top \boldsymbol{U}_3)$
- **one**: $\tilde{\mathbf{1}}_i = (\boldsymbol{O}^\top \boldsymbol{U})^{-1} \mathbf{1}_i$

Again note the $\boldsymbol{U}$'s and $\boldsymbol{O}$'s are arbitrarily numbered to distinguish them within the same equation. Here $(\boldsymbol{O}_1^\top \boldsymbol{U}_1)$ has mode label i for the rows, and mode label j for the columns. Similarly $(\boldsymbol{O}_2^\top \boldsymbol{U}_2)$ has mode label $\pi(i)$ for the rows, and mode label k for the columns and $(\boldsymbol{O}_3^\top \boldsymbol{U}_3)$ has mode label $\pi(i)$ for the rows, and mode label l for the columns (Again for clarity, multiplying by $\boldsymbol{U}$ does not affect the mode label).

Recall that $\boldsymbol{M}_i$ be the outgoing message from node X_i to its parent i.e.

- Evidence Leaf: $\boldsymbol{M}_i = \boldsymbol{\mathcal{L}}_i \times_i \boldsymbol{\delta}_{\bar{x}_i}$
- Non-Evidence Leaf: $\boldsymbol{M}_i = \boldsymbol{\mathcal{L}}_i \times_i \mathbf{1}$
- Internal Node: $\boldsymbol{M}_i = \boldsymbol{\mathcal{T}}_i \times_i \left(\boldsymbol{M}_{c_1(i)} \ldots \boldsymbol{M}_{c_{\alpha_i}(i)} \mathbf{1}_i\right)$
- Root: $P(\bar{x}_{e_1}, \ldots, \bar{x}_{|\mathcal{E}|}) = \boldsymbol{r}^\top \left(\boldsymbol{M}_{c_1(i)} \ldots \boldsymbol{M}_{c_{\alpha_i}(i)} \mathbf{1}_i\right)$

Similarly, let $\widetilde{\boldsymbol{M}}_i$ indicate the transformed outgoing message i.e.

- Evidence Leaf: $\widetilde{\boldsymbol{M}}_i = \widetilde{\boldsymbol{\mathcal{L}}}_i \times_i \boldsymbol{\delta}_{\bar{x}_i}$
- Non-Evidence Leaf: $\widetilde{\boldsymbol{M}}_i = \widetilde{\boldsymbol{\mathcal{L}}}_i \times_i \tilde{\mathbf{1}}$
- Internal Node: $\widetilde{\boldsymbol{M}}_i = \widetilde{\boldsymbol{\mathcal{T}}}_i \times_i \left(\widetilde{\boldsymbol{M}}_{c_1(i)} \ldots \widetilde{\boldsymbol{M}}_{c_{\alpha_i}(i)} \tilde{\mathbf{1}}_i\right)$
- Root: $P(\bar{x}_{e_1}, \ldots, \bar{x}_{|\mathcal{E}|}) = \tilde{\boldsymbol{r}}^\top \left(\widetilde{\boldsymbol{M}}_{c_1(i)} \ldots \widetilde{\boldsymbol{M}}_{c_{\alpha_i}(i)} \tilde{\mathbf{1}}_i\right)$

Note $\boldsymbol{M}_i$ and $\widetilde{\boldsymbol{M}}_i$ are matrices and thus $\widetilde{\boldsymbol{M}} = \boldsymbol{F}^{-1}\boldsymbol{M}_i\boldsymbol{F} = (\boldsymbol{O}^\top\boldsymbol{U})^{-1}\boldsymbol{M}_i(\boldsymbol{O}^\top\boldsymbol{U})$. Let $\widehat{\boldsymbol{M}}_i$ indicate the empirical estimate of $\widetilde{\boldsymbol{M}}_i$.

Furthermore, $\|\cdot\|_2$ refers to spectral norm for matrices and tensors (but normal euclidean norm for vectors). $\|\cdot\|_1$ refers to induced 1 norm for matrices and tensors (max column sum), (but normal l1 norm for vectors). $\|\cdot\|_F$ refers to Frobenius norm.

The tensor spectral norm (for 3 dimensions) is defined in (Nguyen et al., 2010):

$$\|\mathcal{T}\|_2 = \sup_{\|v_i\|_2 \leq 1} \mathcal{T} \times_3 v_3 \times_2 v_2 \times_1 v_1 \tag{7.19}$$

where $\{1, 2, 3\}$ indicate the mode labels of $\mathcal{T}$ and v_1 has mode label 1, v_2 has mode label 2, and v_3 has mode label 3.

We will define the induced 1-norm of a tensor as

$$\|\mathcal{T}\|_{1,1} = \sup_{\|v\|_1 \leq 1} \|\mathcal{T} \bar{\times}_1 v\|_1 \tag{7.20}$$

using the ℓ_1 norm of a matrix (*i.e.*, $\|\boldsymbol{A}\|_1 = \sup_{\|v\|_1 \leq 1} \|\boldsymbol{A}v\|_1$).

For more information about matrix norms see (Horn and Johnson, 1990).

7.8.2 Concentration Bounds

$$\epsilon_j = \left\|\widehat{\mathcal{P}}_j - \mathcal{P}_j\right\|_F \tag{7.21}$$

$$\epsilon_{j,k} = \left\|\widehat{\mathcal{P}}_{j,k} - \mathcal{P}_{j,k}\right\|_F \tag{7.22}$$

$$\epsilon_{j=\bar{x},k,l} = \left\|\widehat{\mathcal{P}}_{j=\bar{x},k,l} - \mathcal{P}_{j=\bar{x},k,l}\right\|_F \tag{7.23}$$

$$\epsilon_{j,k,l} = \left\|\widehat{\mathcal{P}}_{j,k,l} - \mathcal{P}_{j,k,l}\right\|_F \tag{7.24}$$

$j = \bar{x}$ denotes that the mode indicated by j is fixed to the evidence value $\bar{x}$ i.e. $\mathcal{P}_{j=\bar{x},k,l}$ is created by taking a slice of $\mathcal{P}_{j,k,l}$ by fixing the first mode to $\bar{x}$.

As the number of samples N gets large, we expect these quantities to be small.

Lemma 17 (variant of HKZ (Hsu et al., 2009) **).** *If the algorithm independently samples N observation* *triples from the tree, then with probability at least* $1 - \delta$*:*

$$\epsilon_j \leq \sqrt{\frac{C}{N} \ln \frac{|\mathscr{O}|}{\delta}} + \sqrt{\frac{1}{N}} \tag{7.25}$$

$$\epsilon_{j,k} \leq \sqrt{\frac{C}{N} \ln \frac{|\mathscr{O}|}{\delta}} + \sqrt{\frac{1}{N}} \tag{7.26}$$

$$\epsilon_{j,k,l} \leq \sqrt{\frac{C}{N} \ln \frac{|\mathscr{O}|}{\delta}} + \sqrt{\frac{1}{N}} \tag{7.27}$$

$$\max_x \epsilon_{j,k=x,l} \leq \sqrt{\frac{C}{N} \ln \frac{|\mathscr{O}|}{\delta}} + \sqrt{\frac{1}{N}} \tag{7.28}$$

$$\max_x \epsilon_{j=x,k,l} \leq \sqrt{\frac{S_O}{N} \ln \frac{|\mathscr{O}|}{\delta}} + \sqrt{\frac{S_O}{N}} \tag{7.29}$$

where C is some constant (from the union bound over $O(V^3)$). (V is the total number of observed variables in the tree). The proof is the same as that of HKZ (Hsu et al., 2009) except the union bound is larger. The last bound can be made tighter, identical to HKZ, but for simplicity we do not pursue that approach here.

7.9 Eigenvalue Bounds

Basically this is Lemma 9 in HKZ (Hsu et al., 2009), which is stated below for completeness:

Lemma 18. *Suppose* $\epsilon_{j,k} \leq \varepsilon \times \sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})$ *for some* $\varepsilon < 1/2$*. Let* $\varepsilon_0 = \epsilon_{j,k}^2/((1-\varepsilon)\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k}))^2$*. Then:*

1. $\varepsilon_0 < 1$
2. $\sigma_{S_H}(\widehat{\boldsymbol{U}}^\top \widehat{\boldsymbol{\mathcal{P}}}_{j,k}) \geq (1-\varepsilon)\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})$
3. $\sigma_{S_H}(\widehat{\boldsymbol{U}}^\top \boldsymbol{\mathcal{P}}_{j,k}) \geq \sqrt{1-\varepsilon_0}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})$
4. $\sigma_{S_H}(\boldsymbol{O}^\top \widehat{\boldsymbol{U}}) \geq \sqrt{1-\varepsilon}\sigma_{S_H}(\boldsymbol{O})$

The proof is in HKZ (Hsu et al., 2009).

7.9.1 Bounding the Transformed Quantities

If Lemma 18 holds then $(\boldsymbol{O}^\top \widehat{\boldsymbol{U}})$ is invertible. Thus, if we define $\widetilde{\boldsymbol{M}}_i = (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} \boldsymbol{M}_i (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})$. Then clearly, $(\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} \widetilde{\boldsymbol{M}}_i (\boldsymbol{O}^\top \widehat{\boldsymbol{U}}) = \boldsymbol{M}_i$. (We admit this is a slight abuse of notation, since $\widetilde{\boldsymbol{M}}_i$ is previously defined to be $(\boldsymbol{O}^\top \boldsymbol{U})^{-1} \boldsymbol{M}_i (\boldsymbol{O}^\top \boldsymbol{U})$, but as long as $(\boldsymbol{O}^\top \widehat{\boldsymbol{U}})$ is invertible it doesn't really matter whether it equals $(\boldsymbol{O}^\top \boldsymbol{U})$ or not for the purposes of this proof). The other quantities are defined similarly.

We seek to bound the following four quantities:

$$\delta^i_{one} = \|(\boldsymbol{O}^\top\widehat{\boldsymbol{U}})(\widehat{\boldsymbol{1}}_i - \tilde{\boldsymbol{1}}_i)\|_1 \tag{7.30}$$
$$\gamma_i = \|(\widehat{\boldsymbol{\mathcal{T}}}_i - \widetilde{\boldsymbol{\mathcal{T}}}_i) \times_j (\boldsymbol{O}_1^\top\widehat{\boldsymbol{U}}_1)^{-1} \times_m (\boldsymbol{O}_2^\top\widehat{\boldsymbol{U}}_2) \times_l (\boldsymbol{O}_3^\top\widehat{\boldsymbol{U}}_3)^{-1}\|_2 \tag{7.31}$$
$$\delta_{root} = \|(\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^T(\boldsymbol{O}^\top\widehat{\boldsymbol{U}})^{-1}\|_\infty \tag{7.32}$$
$$\triangle_i = \sum_{\boldsymbol{x}_i} \|(\boldsymbol{O}_2^\top\widehat{\boldsymbol{U}}_2)(\widehat{\boldsymbol{M}}_i - \widetilde{\boldsymbol{M}}_i)(\boldsymbol{O}_3^\top\widehat{\boldsymbol{U}}_3)^{-1}\|_1 \tag{7.33}$$

Here $\boldsymbol{x}_i$ denotes all observations that are in the subtree of node i (since i may be hidden or observed). Sometimes we like to distinguish between when i is observed and when i is hidden.Thus, we sometimes refer to the quantity $\triangle_i^{obs}$ and $\triangle_i^{hidden}$ for when i is observed or hidden respectively. Note that we don't need to explicitly have a bound for $\|\widehat{\boldsymbol{\mathcal{L}}}_i - \widetilde{\boldsymbol{\mathcal{L}}}_i\|$ since if i is a leaf then $\widetilde{\boldsymbol{M}}_i$ is just a slice of $\widetilde{\boldsymbol{\mathcal{L}}}_i$.

Lemma 19. *Assume $\epsilon_{j,k} \le \sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})/3$ for all $j \neq k$. Then*

$$\delta_{root} \le \frac{2\epsilon_j}{\sqrt{3}\sigma_{S_H}(\boldsymbol{O})} \tag{7.34}$$
$$\delta^i_{one} \le 4\sqrt{S_H}\left(\frac{\epsilon_{j,k}}{\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})^2} + \frac{\epsilon_j}{\sqrt{3}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})}\right) \tag{7.35}$$
$$\gamma_i \le \frac{4\sqrt{S_H}}{\sigma_{S_H}(\boldsymbol{O})}\left(\frac{\epsilon_{j,k}}{\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})^2} + \frac{\epsilon_{j,k,l}}{\sqrt{3}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})}\right) \tag{7.36}$$
$$\triangle_i^{hidden} \le \left((1+\gamma_i)\prod_{a=1}^{\alpha_i}(1+\triangle_{c_a(i)})\delta^i_{one} + (1+\gamma_i)m\prod_{a=1}^{\alpha_i}(1+\triangle_{c_a(i)}) - m\right) \tag{7.37}$$
$$\triangle_i^{obs} \le \le 4\frac{\sqrt{S_H}}{\sigma_{S_H}(\boldsymbol{O})}\left(\frac{\epsilon_{j,k}}{(\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k}))^2} + \frac{\sum_{x_i}\epsilon_{j,k=x_i,l}}{\sqrt{3}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})}\right) \tag{7.38}$$

The main challenge in this part is $\triangle_i$ and γ_i^{hidden}. The rest are similar to HKZ. However, we go through the other bounds to be more explicit about some of the properties used, since sometimes we have used different norms etc.

δ_{root}

We note that $\widehat{\boldsymbol{r}} = \widehat{\boldsymbol{U}}^\top\widehat{\boldsymbol{\mathcal{P}}}_j$ and similarly $\tilde{\boldsymbol{r}} = \widehat{\boldsymbol{U}}^\top\boldsymbol{\mathcal{P}}_j$.

$$\delta_{root} = \|(\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^\top(\boldsymbol{O}^\top\widehat{\boldsymbol{U}})^{-1}\|_\infty \le \|\widehat{\boldsymbol{\mathcal{P}}}_j^\top - \boldsymbol{\mathcal{P}}_j^\top\|_2\|\widehat{\boldsymbol{U}}_j\|_2\|(\boldsymbol{O}^\top\widehat{\boldsymbol{U}})^{-1}\|_2 \tag{7.39}$$
$$\le \|\widehat{\boldsymbol{\mathcal{P}}}_j^\top - \boldsymbol{\mathcal{P}}_j^\top\|_2\|(\boldsymbol{O}^\top\widehat{\boldsymbol{U}})^{-1}\|_2 \le \frac{\epsilon_j}{\sigma_{S_H}(\boldsymbol{O}^\top\widehat{\boldsymbol{U}})} \tag{7.40}$$

The first inequality follows from the relationship between ℓ_∞ and ℓ_2 norm and submultiplicativity. The second follows from a matrix perturbation bound given in Lemma 7.94. We also use the fact that since $\widehat{\boldsymbol{U}}$ is orthonormal it has spectral norm 1.

Assuming that $\epsilon_{j,k} \leq \sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})/3$ gives $\delta_{root} \leq \frac{2\epsilon_r}{\sqrt{3}\sigma_{S_H}(\boldsymbol{O})}$ by Lemma 18.

δ^i_{one}

$$\begin{aligned}
\delta^i_{one} &= \|(\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\mathbf{1}}_i - \tilde{\mathbf{1}}_i)\|_1 \leq \sqrt{S_H}\,\|\boldsymbol{O}\|_2 \left\|\widehat{\boldsymbol{U}}\right\|_2 \left\|\widehat{\mathbf{1}}_i - \tilde{\mathbf{1}}\right\|_2 && (7.41)\\
&= \sqrt{S_H}\left\|\widehat{\mathbf{1}}_i - \tilde{\mathbf{1}}_i\right\|_2 = \sqrt{S_H}\left\|\widehat{\mathbf{1}}_i - \tilde{\mathbf{1}}_i\right\|_2 && (7.42)
\end{aligned}$$

Here we have converted ℓ_1 norm to ℓ_2 norm, used submultiplicativity, the fact that $\widehat{\boldsymbol{U}}$ is orthonormal so has spectral norm 1, and that $\boldsymbol{O}$ is a conditional probability matrix and therefore also has spectral norm 1.

We note that $\widehat{\mathbf{1}}_i = (\widehat{\boldsymbol{\mathcal{P}}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j$ and similarly $\tilde{\mathbf{1}}_i = (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\boldsymbol{\mathcal{P}}_j$, where j and k are a particular pair of observations described in the main paper.

$$\begin{aligned}
\|\widehat{\mathbf{1}}_i - \tilde{\mathbf{1}}_i\|_2 &= \|(\widehat{\boldsymbol{\mathcal{P}}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j - (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\boldsymbol{\mathcal{P}}_j\|_2 && (7.43)\\
&= \|(\widehat{\boldsymbol{\mathcal{P}}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j - (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j + (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j - (\boldsymbol{\mathcal{P}}^\top_{j,k}\widehat{\boldsymbol{U}})^+\boldsymbol{\mathcal{P}}_j\|_2 && (7.44)\\
&\leq \|(\widehat{\boldsymbol{\mathcal{P}}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j - (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j\|_2 + \|(\widehat{\boldsymbol{\mathcal{P}}}_{j,k}\widehat{\boldsymbol{U}})^+\widehat{\boldsymbol{\mathcal{P}}}_j - (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\boldsymbol{\mathcal{P}}_j\|_2 && (7.45)\\
&\leq \|(\widehat{\boldsymbol{\mathcal{P}}}_{j,k}\widehat{\boldsymbol{U}})^+ - (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\|_2\|\hat{\boldsymbol{\mathcal{P}}}_j\|_1 + \|(\widehat{\boldsymbol{\mathcal{P}}}_{j,k}\widehat{\boldsymbol{U}})^+ - (\boldsymbol{\mathcal{P}}_{j,k}\widehat{\boldsymbol{U}})^+\|_2\|\widehat{\boldsymbol{\mathcal{P}}}_j - \boldsymbol{\mathcal{P}}_j\|_2 && (7.46)\\
&\leq \frac{1+\sqrt{5}}{2} \times \frac{\epsilon_{m,j}}{\min(\sigma_{S_H}(\widehat{\boldsymbol{\mathcal{P}}}_{j,k}), \sigma_{S_H}(\boldsymbol{\mathcal{P}}^\top_{j,k}\widehat{\boldsymbol{U}}))^2} + \frac{\epsilon_j}{\sigma_{S_H}(\boldsymbol{\mathcal{P}}^\top_{j,k}\widehat{\boldsymbol{U}})} && (7.47)
\end{aligned}$$

where we have used the triangle inequality in the first inequality and the submultiplicative property of matrix norms in the second. The last inequality follows by matrix perturbation bounds. Thus using the assumption that $\epsilon_{j,k} \leq \sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})/3$, we get that

$$\delta_{one} \leq 4\sqrt{S_H}\left(\frac{\epsilon_{j,k}}{\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})^2} + \frac{\epsilon_j}{\sqrt{3}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})}\right) \qquad (7.48)$$

Tensor

Recall that $\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{T}}_i \times_i (\boldsymbol{O}_1^\top\widehat{\boldsymbol{U}}_1) \times_{\pi(i)} (\boldsymbol{O}_2^\top\widehat{\boldsymbol{U}}_2)^{-1} \times_{\pi(i)} (\boldsymbol{O}_3^\top\widehat{\boldsymbol{U}}_3) = \boldsymbol{\mathcal{P}}_{j,k,l} \times_j \widehat{\boldsymbol{U}}_1 \times_k (\boldsymbol{\mathcal{P}}_{m,k}\widehat{\boldsymbol{U}}_2)^\dagger \times_l \widehat{\boldsymbol{U}}_3$. Similarly, $\widehat{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{P}}_{j,k,l} \times_j \widehat{\boldsymbol{U}}_1 \times_k (\boldsymbol{\mathcal{P}}_{m,k}\widehat{\boldsymbol{U}}_2)^\dagger \times_l \widehat{\boldsymbol{U}}_3$.

$$\|(\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i) \times_j (O_1^\top \widehat{U}_1)^{-1} \times_m (O_2^\top \widehat{U}_2) \times_l (O_3^\top \widehat{U}_3)^{-1}\|_{1,1} \leq \frac{\sqrt{S_H}}{\sigma_{S_H}(O)} \left\|\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i\right\|_2 \tag{7.49}$$

This is because both $\widehat{U}$ and O have spectral norm one and the $\sqrt{S_H}$ factor is the cost of converting from 1 norm to spectral norm.

$$\begin{aligned}
\left\|\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i\right\|_2 &= \left\|\widehat{\mathcal{P}}_{j,k,l} \times_j \widehat{U}_1 \times_k (\widehat{\mathcal{P}}_{m,k}\widehat{U}_2)^\dagger \times_l \widehat{U}_3 - \mathcal{P}_{j,k,l} \times_j \widehat{U}_1 \times_k (\mathcal{P}_{m,k}\widehat{U}_2)^\dagger \times_l \widehat{U}_3\right\|_2 && (7.50)\\
&= \left\|\widehat{\mathcal{P}}_{j,k,l} \times_j \widehat{U}_1^\top \times_k (\widehat{\mathcal{P}}_{m,k}\widehat{U}_2)^\dagger \times_l \widehat{U}_3 - \widehat{\mathcal{P}}_{j,k,l} \times_j \widehat{U}_1 \times_k (\mathcal{P}_{m,k}\widehat{U}_2)^\dagger \times_l \widehat{U}_3\right\|_2 && (7.51)\\
&\quad + \left\|\widehat{\mathcal{P}}_{j,k,l} \times_j \widehat{U}_1^\top \times_k (\mathcal{P}_{m,k}\widehat{U}_2)^\dagger \times_l \widehat{U}_3 - \mathcal{P}_{j,k,l} \times_j \widehat{U}_1 \times_k (\mathcal{P}_{m,k}\widehat{U}_2)^\dagger \times_l \widehat{U}_3\right\|_2 && (7.52)\\
&= \left\|\widehat{\mathcal{P}}_{j,k,l} \times_j \widehat{U}_1 \times_k ((\widehat{\mathcal{P}}_{m,k}\widehat{U}_2)^\dagger - (\mathcal{P}_{m,k}\widehat{U}_2)^\dagger) \times_l \widehat{U}_3\right\|_2 && (7.53)\\
&\quad + \left\|(\widehat{\mathcal{P}}_{j,k,l} \times_j \widehat{U}_1 \times_l \widehat{U}_3 - \mathcal{P}_{j,k,l} \times_j \widehat{U}_1 \times_l \widehat{U}_3) \times_k (\mathcal{P}_{m,k}\widehat{U}_2)^\dagger\right\|_2 && (7.54)\\
&= \left\|\widehat{\mathcal{P}}_{j,k,l}\right\|_2 \frac{1+\sqrt{5}}{2} \frac{\epsilon_{l,j}}{\min(\sigma_{S_H}(\widehat{\mathcal{P}}_{m,k}), \sigma_{S_H}(\mathcal{P}_{m,k}\widehat{U}))^2} + \frac{\epsilon_{j,k,l}}{\sigma_{S_H}(\mathcal{P}_{m,k}\widehat{U})} && (7.55)
\end{aligned}$$

It is clear that $\left\|\widehat{\mathcal{P}}_{j,k,l}\right\|_2 \leq \left\|\widehat{\mathcal{P}}_{j,k,l}\right\|_F \leq 1$.

Using the fact that $\epsilon_{j,k} \leq \sigma_{S_H}(\mathcal{P}_{j,k})/3$ gives us the following bound:

$$\gamma_v \leq \frac{4\sqrt{S_H}}{\sigma_{S_H}(O)} \left(\frac{\epsilon_{j,k}}{\sigma_{S_H}(\mathcal{P}_{j,k})} + \frac{\epsilon_{j,k,l}}{\sqrt{3}\sigma_{S_H}(\mathcal{P}_{j,k})} \right) \tag{7.56}$$

Bounding $\triangle_i$

We now seek to bound $\triangle_i = \sum_{x_i} \|(O_2^\top \widehat{U}_2)(\widehat{M}_i - \widetilde{M}_i)(\widehat{U}_3^\top O_3)^{-1}\|_1$. There are two cases: either i is a leaf or it is not.

***i* is leaf node**

In this case our proof simply follows from HKZ (Hsu et al., 2009) and is repeated here for convenience.

$$\begin{aligned}
\|(O_2^\top \widehat{U}_2)(\widehat{M} - \widetilde{M})(O_3^\top \widehat{U}_3)^{-1}\|_1 &\leq \sqrt{S_H}\|O_2\|_1 \left\|(\widehat{M}_i - \widetilde{M}_i)(O_3^\top \widehat{U}_3)^{-1}\right\|_2 && (7.57)\\
&\leq \sqrt{S_H}\frac{\left\|\widehat{M}_i - \widetilde{M}_i\right\|_2}{\sigma_{S_H}(O^\top \widehat{U})} && (7.58)
\end{aligned}$$

Note that $\widehat{\boldsymbol{M}}_i = (\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger}\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2$ and $\widetilde{\boldsymbol{M}}_i = (\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger}\boldsymbol{\mathcal{P}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2$.

$$\begin{aligned}
\left\|\widehat{\boldsymbol{M}}_i - \widetilde{\boldsymbol{M}}_i\right\|_2 &= \left\|(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{-1}\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2 - (\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger}\boldsymbol{\mathcal{P}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2\right\|_2 \\
&= \left\|(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger}\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2 - (\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger}\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2 + (\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger}\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2 - \widehat{\boldsymbol{U}}_1^{\top}\boldsymbol{\mathcal{P}}_{j=x_i,k,l}(\widehat{\boldsymbol{U}}_2^{\top}\widehat{\boldsymbol{\mathcal{P}}}_{k,m})^{\dagger}\right\|_2 \\
&\leq \left\|((\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger} - (\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger})\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2\right\|_2 + \left\|(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}_1)^{\dagger}(\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2 - \boldsymbol{\mathcal{P}}_{j=x_i,k,l}\widehat{\boldsymbol{U}}_2)\right\|_2 \\
&\leq \left\|\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\right\|_2 \frac{1+\sqrt{5}}{2}\frac{\epsilon_{k,m}}{\min(\sigma_{S_H}(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}), \sigma_{S_H}(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}))} + \frac{\epsilon_{j=x_i,k,l}}{\sigma_{S_H}(\widehat{\boldsymbol{\mathcal{P}}}_{k,m})\widehat{\boldsymbol{U}}} \\
&\leq P(x_i = x)\frac{1+\sqrt{5}}{2}\frac{\epsilon_{k,m}}{\min(\sigma_{S_H}(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}), \sigma_{S_H}(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}}))^2} + \frac{\epsilon_{j=x_i,k,l}}{\sigma_{S_H}(\widehat{\boldsymbol{\mathcal{P}}}_{k,m}\widehat{\boldsymbol{U}})} \qquad (7.59)
\end{aligned}$$

where the first inequality follows from the triangle inequality, and the second uses matrix perturbation bounds (and the fact that spectral norm of $\widehat{\boldsymbol{U}}$ is 1).

The final inequality follows from the fact that spectral norm is less than frobenius norm which is less than l1 norm:

$$\left\|\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}\right\| \leq \sqrt{\sum_{a,b}[\widehat{\boldsymbol{\mathcal{P}}}_{j=x_i,k,l}]^2_{a,b}} \leq \sum_{a,b}[\boldsymbol{\mathcal{P}}_{j=x_i,k,l}]_{a,b} \leq P(x_i = x) \qquad (7.60)$$

The first inequality follows from relation between 1 operator norm and 2 operator norm. Because $\boldsymbol{O}$ is a conditional probability matrix $\|\boldsymbol{O}\|_1 = 1$ (i.e. the max column sum is 1).

Using the fact that $\epsilon_{j,k} \leq \sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})/3$ gives us the following bound:

$$\triangle_{i,x} \leq 4\frac{\sqrt{S_H}}{\sigma_{S_H}(\boldsymbol{O})}\left(P(x_i = x)\frac{\epsilon_{j,k=x_i,l}}{(\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{l,m}))^2} + \frac{\epsilon_{j,k=x_i,l}}{\sqrt{3}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{l,m})}\right) \qquad (7.61)$$

Summing over x would give

$$\triangle_i \leq 4\frac{\sqrt{S_H}}{\sigma_{S_H}(\boldsymbol{O})}\left(\frac{\epsilon_{l,m}}{(\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{l,m}))^2} + \frac{\sum_{x_i}\epsilon_{j,k=x_i,l}}{\sqrt{3}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{l,m})}\right) \qquad (7.62)$$

***i* is not a leaf node**

Let $\widehat{\boldsymbol{m}}_{\alpha_i:1} = \widehat{\boldsymbol{M}}_{c_{\alpha_i}(i)} ... \widehat{\boldsymbol{M}}_1 \widehat{\mathbf{1}}_i$ and $\tilde{\boldsymbol{m}}_{\alpha_i:1} = \widetilde{\boldsymbol{M}}_{c_{\alpha_i}(i)} ... \widetilde{\boldsymbol{M}}_1 \tilde{\mathbf{1}}_i$

$$\begin{aligned}
& \sum_{\mathbf{x}_i} \|(\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\hat{\boldsymbol{M}}_i - \widetilde{\boldsymbol{M}}_i)(\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1}\|_1 \\
= & \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\widehat{\mathcal{T}}_i \times_j \widehat{\boldsymbol{M}}_{c_{\alpha_i}(i)} ... \widehat{\boldsymbol{M}}_1 \widehat{\mathbf{1}}_i - \widetilde{\mathcal{T}}_i \times_j \widetilde{\boldsymbol{M}}_{c_{\alpha_i}(i)} ... \widetilde{\boldsymbol{M}}_1 \tilde{\mathbf{1}}_i)(\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_1 \\
= & \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\widehat{\mathcal{T}}_i \times_j \widehat{\boldsymbol{m}}_{\alpha_i:1} - \widetilde{\mathcal{T}}_i \times_j \tilde{\boldsymbol{m}}_{\alpha_i:1})(\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_1 \\
= & \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2) \left((\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i) \times_j \tilde{\boldsymbol{m}}_{\alpha_i:1} + (\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i) \times_j (\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) + \widetilde{\mathcal{T}}_i \times_j (\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right) (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_1 \\
\leq & \sum_{\mathbf{x}_i} \left\| (\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i) \times_j (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)^{-1} \times_m (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2) \times_l (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_{1,1} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1) \tilde{\boldsymbol{m}}_{\alpha_i:1} \right\|_1 \\
+ & \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \left\| (\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i) \times_j (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)^{-1} \times_m (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2) \times_l (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_{1,1} \\
+ & \sum_{\mathbf{x}_i} \left\| \widetilde{\mathcal{T}}_i \times_j (\boldsymbol{O}^\top \widehat{\boldsymbol{U}}_1)^{-1} \times_m (\widehat{\boldsymbol{U}}_2^\top \boldsymbol{O}_2) \times_l (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_{1,1} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \qquad (7.63)
\end{aligned}$$

First term is bounded by:

$$\left\| (\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i) \times_j (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)^{-1} \times_m (\widehat{\boldsymbol{U}}_2^\top \boldsymbol{O}_2) \times_l (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_{1,1} S_H \leq S_H \gamma_i \qquad (7.64)$$

Second term is bounded by:

$$\begin{aligned}
& \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \left\| (\widehat{\mathcal{T}}_i - \widetilde{\mathcal{T}}_i) \times_j (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)^{-1} \times_m (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2) \times_l (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_{1,1} \qquad (7.65) \\
& \leq \gamma_i \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \qquad (7.66)
\end{aligned}$$

Third Term is bounded by:

$$\left\| \widetilde{\mathcal{T}}_i \times_j (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)^{-1} \times_m (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2) \times_l (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_{1,1} \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \leq \sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \qquad (7.67)$$

In the next section, we will see that

$$\sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \leq \left(\prod_{a=1}^{\alpha_i} (1 + \Delta_{c_a(i)}) \delta_{one}^i + S_H \prod_{a=1}^{\alpha_i} (1 + \triangle_{c_a(i)}) - S_H \right) \qquad (7.68)$$

So the overall bound is

$$\triangle_i \leq \left((1 + \gamma_i) \prod_{a=1}^{\alpha_i} (1 + \Delta_{c_a(i)}) \delta_{one}^i + (1 + \gamma_i) S_H \prod_{a=1}^{\alpha_i} (1 + \Delta_{c_a(i)}) - S_H \right). \qquad (7.69)$$

Bounding $\sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1$

Lemma 20.

$$\sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{m}}_{\alpha_i:1} - \tilde{\boldsymbol{m}}_{\alpha_i:1}) \right\|_1 \le \prod_{a=1}^{\alpha_i} (1 + \triangle_{c_a(i)}) \delta^i_{one} + S_H \prod_{a=1}^{\alpha_i} (1 + \triangle_{c_a(i)}) - S_H \tag{7.70}$$

The proof is by induction. Base case: $\left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\mathbf{1}}_i - \tilde{\mathbf{1}}_i) \right\|_1 \le \delta^i_{one}$, by definition of δ^i_{one}.

Inductive step: Let us say claim holds up until $u-1$. We show it holds for u. Thus

$$\sum_{\mathbf{x}_i} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{m}}_{(u-1):1} - \tilde{\boldsymbol{m}}_{(u-1):1}) \right\|_1 \le \prod_{a=1}^{u-1} (1 + \triangle_{c_a(i)}) \delta^i_{one} + S_H \prod_{a=1}^{u-1} (1 + \triangle_{c_a(i)}) - S_H \tag{7.71}$$

We now decompose the sum over x as

$$\sum_{\mathbf{x}_{u:1}} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{m}}_{u:1} - \tilde{\boldsymbol{m}}_{u:1}) \right\|_1 \tag{7.72}$$

$$= \sum_{\mathbf{x}_{u:1}} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}}) \left((\widehat{\boldsymbol{M}}_{c_u(i)} - \widetilde{\boldsymbol{M}}_{c_u(i)}) \tilde{\boldsymbol{m}}_{(u-1):1} + (\widehat{\boldsymbol{M}}_{c_u(i)} - \widetilde{\boldsymbol{M}}_{c_u(i)})(\widehat{\boldsymbol{m}}_{(u-1):1} - \tilde{\boldsymbol{m}}_{(u-1):1}) + (\widehat{\boldsymbol{m}}_{(u-1):1} - \tilde{\boldsymbol{m}}_{(u-1):1}) \right) \right\|_1$$

Using the triangle inequality, we get

$$\sum_{\mathbf{x}_{u:1}} \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\widehat{\boldsymbol{M}}_{c_u(i)} - \widetilde{\boldsymbol{M}}_{c_u(i)})(\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_1 \left\| (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3) \tilde{\boldsymbol{m}}_{(u-1):1} \right\|_1 \tag{7.73}$$

$$+ \sum_{\mathbf{x}_{u:1}} \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\widehat{\boldsymbol{M}}_{c_u(i)} - \widetilde{\boldsymbol{M}}_u)(\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)^{-1} \right\|_1 \left\| (\boldsymbol{O}_3^\top \widehat{\boldsymbol{U}}_3)(\widehat{\boldsymbol{m}}_{(u-1):1} - \tilde{\boldsymbol{m}}_{(u-1):1}) \right\|_1 \tag{7.74}$$

$$+ \sum_{\mathbf{x}_{u:1}} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}}) \widetilde{\boldsymbol{M}}_{c_u(i)} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} \right\|_1 \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{m}}_{(u-1):1} - \tilde{\boldsymbol{m}}_{(u-1):1}) \right\|_1 \tag{7.75}$$

Again we are just numbering the $\boldsymbol{U}$'s and $\boldsymbol{O}$'s for clarity to see which corresponds with which. They are omitted in the actual theorem statements since we will take minimums etc. at the end.

We now must bound these terms. First term:

$$\sum_{\mathbf{x}_u} \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\widehat{\boldsymbol{M}}_{c_u(i)} - \widetilde{\boldsymbol{M}}_{c_u(i)})(\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)^{-1} \right\|_1 \sum_{x_{1:u-1}} \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1) \tilde{\boldsymbol{m}}_{(u-1):1} \right\|_1 \le \triangle_u \sum_{\mathbf{x}_{(u-1):1}} \left\| \tilde{\boldsymbol{m}}_{(u-1):1} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}}) \right\|_1 \le S_H \triangle_u$$

since $\triangle_u = \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\widehat{\boldsymbol{M}}_{c_u(i)} - \widetilde{\boldsymbol{M}}_{c_u(i)})(\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)^{-1} \right\|_1$. Second term can be bounded by inductive hypothesis:

$$\sum_{\mathbf{x}_{u:1}} \left\| (\boldsymbol{O}_2^\top \widehat{\boldsymbol{U}}_2)(\widehat{\boldsymbol{M}}_{c_u(i)} - \widetilde{\boldsymbol{M}}_{c_u(i)})(\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)^{-1} \right\|_1 \left\| (\boldsymbol{O}_1^\top \widehat{\boldsymbol{U}}_1)(\widehat{\boldsymbol{m}}_{(u-1):1} - \tilde{\boldsymbol{m}}_{(u-1):1}) \right\|_1$$

$$\leq \triangle_u \left(\prod_{a=1}^{u-1} (1 + \triangle_{c_a(i)}) \delta^i_{one} + S_H \prod_{a=1}^{u-1} (1 + \triangle_{c_a(i)}) - S_H \right) \tag{7.76}$$

The third term is bounded by observing that $(\boldsymbol{O}^\top \widehat{\boldsymbol{U}})\widetilde{\boldsymbol{M}}_{c_u(i)}(\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1}$ = diag($P\mathbf{x}_u$|Parent]). Thus it is diagonal, and $P(\mathbf{x}|Parent)$ has max row or column sum as 1. This means that the third term is bounded by the inductive hypothesis as well:

$$\sum_{\mathbf{x}_{u:1}} \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})\widetilde{\boldsymbol{M}}_{c_u(i)}(\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} \right\|_1 \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{m}}_{(u-1):1} - \tilde{\boldsymbol{m}}_{(u-1):1}) \right\|_1$$

$$\leq \left(\prod_{a=1}^{u-1} (1 + \triangle_{c_a(i)}) \delta^i_{one} + S_H \prod_{a=1}^{u-1} (1 + \triangle_{c_u(i)}) - S_H \right) \tag{7.77}$$

7.9.2 Bounding the propagation of error in tree

We now wrap up the proof based on the approach of HKZ(Hsu et al., 2009).

Lemma 21.

$$\sum_{x_1,\ldots,x_O} \left| \widehat{P}(x_1,\ldots,x_O) - P(x_1,\ldots,x_O) \right|$$

$$\leq S_H \delta_{root} + (1 + \delta_{root}) \left(\prod_{a=1}^{\alpha_i} (1 + \triangle_{c_a(i)}) \delta^r_{one} + S_H \prod_{a=1}^{\alpha_i} (1 + \triangle_{c_a(i)}) - S_H \right) \tag{7.78}$$

$$\begin{aligned}
\sum_{x_1,\ldots,x_O} \left| \widehat{P}(x_1,\ldots,x_O) - P(x_1,\ldots,x_O) \right| &= \sum_{x_1,\ldots,x_O} \left| \widehat{\boldsymbol{r}}^\top \widehat{\boldsymbol{M}}_{c_1(r)} \ldots \widehat{\boldsymbol{M}}_{c_{\alpha_r}(r)} \widehat{\mathbf{1}}_r - \tilde{\boldsymbol{r}}^\top \widetilde{\boldsymbol{M}}_{c_1(r)} \ldots \widetilde{\boldsymbol{M}}_{c_{\alpha_r}(r)} \tilde{\mathbf{1}}_r \right| && (7.79) \\
&\leq \sum_{x_1,\ldots,x_O} \left| (\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}) \right| && (7.80) \\
&+ \sum_{x_1,\ldots,x_O} \left| (\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{M}}_{\alpha_r:1} \widehat{\mathbf{1}}_r - \widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}_r) \right| && (7.81) \\
&+ \sum_{x_1,\ldots,x_O} \left| \tilde{\boldsymbol{r}}^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{M}}_{\alpha_r:1} \widehat{\mathbf{1}}_i - \widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}) \right| && (7.82)
\end{aligned}$$

The first sum is bounded using Holder inequality and noting that the first term is a conditional probability (of all observed variables conditioned on the root)

$$\sum_{x_1,\ldots,x_O} \left| (\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}) \right| \tag{7.83}$$

$$\leq \sum_{x_1,\ldots,x_O} \left\| (\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} \right\|_\infty \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}) \right\|_1 \leq S_H \delta_{root} \tag{7.84}$$

The second sum is bounded by another application of Holder's inequality (and the previous lemma):

$$\sum_{x_1,\ldots,x_O} \left| (\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{M}}_{\alpha_r:1} \widehat{\mathbf{1}}_r - \widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}_r) \right| \tag{7.85}$$

$$\leq \sum_{x_1,\ldots,x_O} \left\| (\widehat{\boldsymbol{r}} - \tilde{\boldsymbol{r}})^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} \right\|_\infty \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{M}}_{\alpha_r:1} \widehat{\mathbf{1}}_r - \widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}_r) \right\|_1 \tag{7.86}$$

$$\leq \delta_{root} \left(\prod_{a=1}^{\alpha_i} (1 + \triangle_{c_a(r)}) \delta_{one}^r + S_H \prod_{a=1}^{\alpha_r} (1 + \triangle_{c_a(r)}) - S_H \right) \tag{7.87}$$

The third sum is also bounded by Holder's Inequality and previous lemmas and noting that $\tilde{\boldsymbol{r}}^\top (\boldsymbol{O}^\top \boldsymbol{U})^{-1} = P(R = r)$:

$$\sum_{x_1,\ldots,x_O} \left| \tilde{\boldsymbol{r}}^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{M}}_{\alpha_r:1} \widehat{\mathbf{1}}_i - \widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}) \right| \tag{7.88}$$

$$\leq \sum_{x_1,\ldots,x_O} \left\| \tilde{\boldsymbol{r}}^\top (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})^{-1} \right\|_\infty \left\| (\boldsymbol{O}^\top \widehat{\boldsymbol{U}})(\widehat{\boldsymbol{M}}_{\alpha_r:1} \widehat{\mathbf{1}}_r - \widetilde{\boldsymbol{M}}_{\alpha_r:1} \tilde{\mathbf{1}}_r) \right\|_1 \tag{7.89}$$

$$\leq \left(\prod_{a=1}^{\alpha_r} (1 + \triangle_{c_a(r)}) \delta_{one}^r + S_H \prod_{a=1}^{\alpha_r} (1 + \triangle_{c_a(r)}) - S_H \right) \tag{7.90}$$

Combining these bounds gives us the desired solution.

7.10 Putting it all together

We seek for

$$\sum_{x_1,\ldots,x_O} \left| \widehat{P}(x_1, \ldots, x_O) - P(x_1, \ldots, x_O) \right| \leq \epsilon \tag{7.91}$$

Using the fact that for $a < .5$, $(1 + a/t)^\top \leq 1 + 2a$, we get that $\triangle_{j_k} \leq O(\epsilon/(S_H J))$. However, $\triangle_j$ is defined recursively, and thus the error accumulates exponential in the longest path of hidden nodes. For example, $\triangle_i^{obs} \leq O(\frac{\epsilon}{(d_{max} S_H)^\ell})$ where ℓ is the longest path of hidden nodes. Tracing this back through will gives the result:

Pick any $\epsilon > 0, \delta < 1$. Let

$$N \geq O\left(\frac{1}{\epsilon^2}\left(\frac{(d_{max}S_H)^{2\ell+1}S_O}{\min_{j,k}\sigma_{S_H}(\boldsymbol{O}_{j,k})^2\min_{i\neq j}\sigma_{S_H}(\boldsymbol{\mathcal{P}}_{j,k})^4}\right)\right)\log\frac{\mathcal{O}}{\delta} \tag{7.92}$$

Then with probability $1-\delta$

$$\sum_{x_1,\ldots,x_O}\left|\widehat{P}(x_1,\ldots,x_O) - P(x_1,\ldots,x_O)\right| \leq \epsilon \tag{7.93}$$

In many cases, if the frequency of the observation symbols follow certain distributions, than the dependence on S_O can be removed as showed in HKZ (Hsu et al., 2009).

7.10.1 Matrix Perturbation Bounds

This is Theorem 3.8 from pg. 143 in Stewart and Sun, 1990 (Stewart and Sun, 1990). Let $A \in \boldsymbol{R}^{m\times n}$, with $m \geq n$ and let $\tilde{A} = A + E$. Then

$$\left\|\widetilde{A}^{\dagger} - A^{\dagger}\right\|_2 \leq \frac{1+\sqrt{5}}{2}\max(\left\|A^{\dagger}\right\|_2^2, \left\|\widetilde{A}\right\|_2^2)\,\|E\|_2 \tag{7.94}$$

Part II

Spectral Models for Natural Language Processing

Chapter 8

Spectral Unsupervised Parsing with Additive Tree Metrics

In this chapter we tackle unsupervised syntactic parsing, an important problem in NLP where existing methods are very sensitive to local optima and therefore require careful initialization. Although traditionally formulated as a parameter learning problem, we take a different approach that revolves around structure learning. Thus, some of the key ideas of this chapter are closely related to the structure learning algorithms discussed in Chapter 6. We discuss this relationship further in 8.5.

Contribution of this chapter: We propose a spectral approach for unsupervised constituent parsing that comes with theoretical guarantees on latent structure recovery. The main algorithm is based on lifting the concept of additive tree metrics for structure learning of latent trees in the phylogenetic and machine learning communities to the case where the tree structure varies across examples. Although finding the "minimal" latent tree is NP-hard in general, for the case of projective trees we find that it can be found using bilexical parsing algorithms. Empirically, our algorithm performs favorably compared to the constituent context model of Klein and Manning (2002) without the need for careful initialization.

Outline: An introduction is first presented, followed by a description of our conditional latent tree model. We then propose a provably consistent learning algorithm and finally present experiments.

Prerequisites: This chapter assumes a general understanding of latent variable models as presented in 2.2, and the connection between latent variable models and low rank factorization in Chapter 3.

8.1 Introduction

Solutions to the problem of grammar induction have been long sought after since the early days of computational linguistics and are interesting both from cognitive and engineering perspectives.

Cognitively, it is more plausible to assume that children obtain only terminal strings of parse trees and not the actual parse trees. This means the unsupervised setting is a better model for studying language acquisition. From the engineering perspective, training data for unsupervised parsing exists in abundance (i.e. sentences and part-of-speech tags), and is much cheaper than the syntactically annotated data required for supervised training.

Most existing solutions treat the problem of unsupervised parsing by assuming a generative process over parse trees e.g. probabilistic context free grammars (Jelinek et al., 1992), and the constituent context model (Klein and Manning, 2002). Learning then reduces to finding a set of parameters that are estimated by identifying a local maximum of an objective function such as the likelihood (Klein and Manning, 2002) or a variant of it (Smith and Eisner, 2005; Cohen and Smith, 2009; Headden et al., 2009; Spitkovsky et al., 2010b; Gillenwater et al., 2010; Golland and DeNero, 2012). Unfortunately, finding the global maximum for these objective functions is usually intractable (Cohen and Smith, 2012) which often leads to severe local optima problems (but see Gormley and Eisner, 2013). Thus, strong experimental results are often achieved by initialization techniques (Klein and Manning, 2002; Gimpel and Smith, 2012), incremental dataset use (Spitkovsky et al., 2010a) and other specialized techniques to avoid local optima such as count transforms (Spitkovsky et al., 2013). These approaches, while empirically promising, generally lack theoretical justification.

On the other hand, recently proposed spectral methods approach the problem via restriction of the PCFG model (Hsu et al., 2012) or matrix completion (Bailly et al., 2013). These novel perspectives offer strong theoretical guarantees but are not designed to achieve competitive empirical results.

In this chapter, we suggest a different approach, to provide a first step to bridging this theory-experiment gap. More specifically, we approach unsupervised constituent parsing from the perspective of *structure learning* as opposed to parameter learning. We associate each sentence with an undirected latent tree graphical model, which is a tree consisting of both observed variables (corresponding to the words in the sentence) and an additional set of latent variables that are unobserved in the data. This undirected latent tree is then directed via a *direction mapping* to give the final constituent parse.

In our framework, parsing reduces to finding the best latent structure for a given sentence. However, due to the presence of latent variables, structure learning of latent trees is substantially more complicated than in observed models. As before, one solution would be local search heuristics.

Intuitively, however, latent tree models encode low rank dependencies among the observed variables permitting the development of "spectral" methods that can lead to provably correct solutions. In particular we leverage the concept of additive tree metrics (Buneman, 1971, 1974) in phylogenetics and machine learning that can create a special distance metric among the observed variables as a function of the underlying spectral dependencies (Choi et al., 2011; Song et al., 2011b; Anandkumar et al., 2011; Ishteva et al., 2012). Additive tree metrics can be leveraged by "meta-algorithms" such as neighbor-joining (Saitou and Nei, 1987) and recursive grouping (Choi et al., 2011) to provide consistent learning algorithms for latent trees.

Moreover, we show that it is desirable to learn the "minimal" latent tree based on the tree metric ("minimum evolution" in phylogenetics). While this criterion is in general NP-hard (Desper and Gascuel, 2005), for projective trees we find that a bilexical parsing algorithm can be used to find

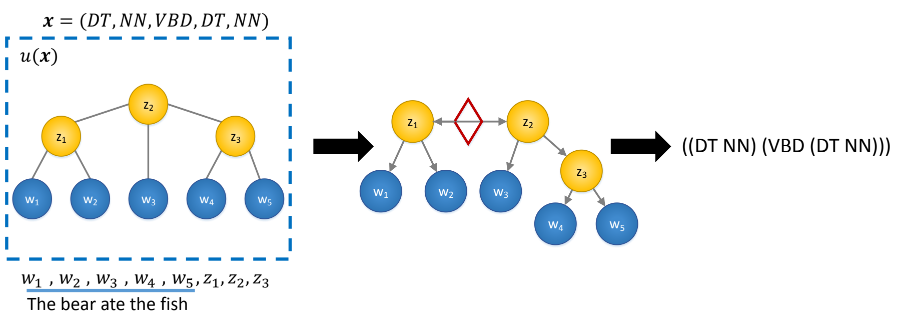


Figure 8.1: Example for the tag sequence (DT, NN, VBD, DT, NN) showing the overview of our approach. We first learn a undirected latent tree for the sequence (left). We then apply a direction mapping h_{dir} to direct the latent tree (center). This can then easily be converted into a bracketing (right).

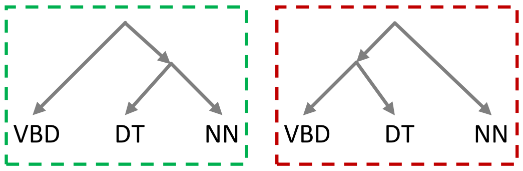


Figure 8.2: Candidate constituent parses for x = (VBD, DT, NN) (left-correct, right -incorrect)

an exact solution efficiently (Eisner and Satta, 1999).

Unlike in phylogenetics and graphical models, where a single latent tree is constructed for all the data, in our case, each part of speech sequence is associated with its own parse tree. This leads to a severe data sparsity problem even for moderately long sentences. To handle this issue, we present a strategy that is inspired by ideas from kernel smoothing in the statistics community (Zhou et al., 2010; Kolar et al., 2010b,a). This allows principled sharing of samples from different but similar underlying distributions.

We provide theoretical guarantees on the recovery of the correct underlying latent tree and characterize the associated sample complexity under our technique. Empirically we evaluate our method on data in English, German and Chinese. Our algorithm performs favorably to Klein and Manning's (2002) constituent-context model (CCM), without the need for careful initialization. In addition, we also analyze CCM's sensitivity to initialization, and compare our results to Seginer's algorithm (Seginer, 2007).

8.2 Learning Setting and Model

In this section, we detail the learning setting and a conditional tree model we learn the structure for.

8.2.1 Learning Setting

Let $\boldsymbol{w} = (w_1, ..., w_\ell)$ be a vector of words corresponding to a sentence of length ℓ. Each w_i is represented by a vector in $\mathbb{R}^p$ for $p \in \mathbb{N}$. The vector is an embedding of the word in some space, chosen from a fixed dictionary that maps word types to $\mathbb{R}^p$. In addition, let $\boldsymbol{x} = (x_1, ..., x_\ell)$ be the associated vector of part-of-speech (POS) tags (i.e. x_i is the POS tag of w_i).

In our learning algorithm, we assume that examples of the form $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$ for $i \in [N] = \{1, \ldots, N\}$ are given, and the goal is to predict a bracketing parse tree for each of these examples. The word embeddings are used during the learning process, but the final decoder that the learning algorithm outputs maps a POS tag sequence $\boldsymbol{x}$ to a parse tree. While ideally we would want to use the word information in decoding as well, much of the syntax of a sentence is determined by the POS tags, and relatively high level of accuracy can be achieved by learning, for example, a supervised parser from POS tag sequences.

Just like our decoder, our model assumes that the bracketing of a given sentence is a function of its POS tags. The POS tags are generated from some distribution, followed by a deterministic generation of the bracketing parse tree. Then, latent states are generated for each bracket, and finally, the latent states at the yield of the bracketing parse tree generate the words of the sentence (in the form of embeddings). The latent states are represented by vectors $z \in \mathbb{R}^m$ where $m < p$.

8.2.2 Intuition

For intuition, consider the simple tag sequence $\boldsymbol{x} = (\texttt{VBD}, \texttt{DT}, \texttt{NN})$. Two candidate constituent parse structures are shown in Figure 8.2 and the correct one is boxed in green (the other in red). Recall that our training data contains word phrases that have the tag sequence $\boldsymbol{x}$ e.g. $\boldsymbol{w}^{(1)} = (\texttt{hit}, \texttt{the}, \texttt{ball})$, $\boldsymbol{w}^{(2)} = (\texttt{ate}, \texttt{an}, \texttt{apple})$.

Intuitively, the words in the above phrases exhibit dependencies that can reveal the parse structure. The determiner (w_2) and the direct object (w_3) are correlated in that the choice of determiner depends on the plurality of w_3. However, the choice of verb (w_1) is mostly independent of the determiner. We could thus conclude that w_2 and w_3 should be closer in the parse tree than w_1 and w_2, giving us the correct structure. Informally, the latent state z corresponding to the (w_2, w_3) bracket would store information about the plurality of z, the key to the dependence between w_2 and w_3. It would then be reasonable to assume that w_2 and w_3 are independent given z.

8.2.3 A Conditional Latent Tree Model

Following this intuition, we propose to model the distribution over the latent bracketing states and words for each tag sequence $\boldsymbol{x}$ as a latent tree graphical model, which encodes conditional independences among the words given the latent states.

Let $\mathcal{V} := \{w_1, ..., w_\ell, z_1, ..., z_H\}$, with w_i representing the word embeddings, and z_i representing the latent states of the bracketings. Then, according to our base model it holds that:

$$P(\boldsymbol{w}, \boldsymbol{z}|\boldsymbol{x}) = \prod_{i=1}^{H} P(z_i|\pi_{\boldsymbol{x}}(z_i), \theta(\boldsymbol{x})) \times \prod_{i=1}^{\ell(\boldsymbol{x})} P(w_i|\pi_{\boldsymbol{x}}(w_i), \theta(\boldsymbol{x})) \tag{8.1}$$

where $\pi_{\boldsymbol{x}}(\cdot)$ returns the parent node index of the argument in the latent tree corresponding to tag sequence $\boldsymbol{x}$.[1] If z is the root, then $\pi_{\boldsymbol{x}}(z) = \emptyset$. All the w_i are assumed to be leaves while all the z_i are internal (i.e. non-leaf) nodes. The parameters $\theta(\boldsymbol{x})$ control the conditional probability tables. We do not commit to a certain parametric family, but see more about the assumptions we make about θ in §8.3.2. The parameter space is denoted Θ. The model assumes a factorization according to a latent-variable tree. The latent variables can incorporate various linguistic properties, such as head information, valence of dependency being generated, and so on. This information is expected to be learned automatically from data.

Our generative model deterministically maps a POS sequence to a bracketing via an undirected latent-variable tree. The orientation of the tree is determined by a *direction mapping* $h_{\text{dir}}(u)$, which is fixed during learning and decoding. This means our decoder first identifies (given a POS sequence) an undirected tree, and then orients it by applying h_{dir} on the resulting tree (see below).

Define $\mathcal{U}$ to be the set of undirected latent trees where all internal nodes have degree exactly 3 (i.e. they correspond to binary bracketing), and in addition $h_{\text{dir}}(u)$ for any $u \in \mathcal{U}$ is projective (explained in the h_{dir} section). In addition, let $\mathcal{T}$ be the set of binary bracketings. The complete generative model that we follow is then:

- Generate a tag sequence $\boldsymbol{x} = (x_1, \ldots, x_\ell)$
- Decide on $u(\boldsymbol{x}) \in \mathcal{U}$, the undirected latent tree that $\boldsymbol{x}$ maps to.
- Set $t \in \mathcal{T}$ by computing $t = h_{\text{dir}}(u)$.
- Set $\theta \in \Theta$ by computing $\theta = \theta(\boldsymbol{x})$.
- Generate a tuple $\boldsymbol{v} = (w_1, \ldots, w_\ell, z_1, ..., z_H)$ where $w_i \in \mathbb{R}^p, z_j \in \mathbb{R}^m$ according to Eq. 8.1.

See Figure 1 (left) for an example.

The Direction Mapping h_{dir}. Generating a bracketing via an undirected tree enables us to build on existing methods for structure learning of latent-tree graphical models (Choi et al., 2011; Anandkumar et al., 2011). Our learning algorithm focuses on recovering the undirected tree based for the generative model that was described above. This undirected tree is converted into a directed tree by applying h_{dir}. The mapping h_{dir} works in three steps:

- It first chooses a top bracket $([1, f-1], [f, \ell])$ where R is the mid-point of the bracket and ℓ is the length of the sentence.

[1] At this point, π refers to an arbitrary direction of the undirected tree $u(\boldsymbol{x})$.

- It marks the edge $e_{i,j}$ that splits the tree according to the top bracket as the "root edge" (marked in red in Figure 8.1(center))
- It then creates t from u by directing the tree outward from $e_{i,j}$ as shown in Figure 8.1(center)

The resulting t is a binary bracketing parse tree. As implied by the above definition of h_{dir}, selecting which edge is the root can be interpreted as determining the top bracket of the constituent parse. For example, in Figure 8.1, the top bracket is $([1,2],[3,5])$ = ([DT, NN], [VBD, DT, NN]). Note that the "root" edge e_{z_1,z_2} partitions the leaves into precisely this bracketing. As indicated in the above section, we restrict the set of undirected trees to be those such that after applying h_{dir} the resulting t is projective i.e. there are no crossing brackets. In §8.4.1, we discuss an effective heuristic to find the top bracket without supervision.

8.3 Spectral Learning Algorithm based on Additive Tree Metrics

Our goal is to recover $t \in \mathcal{T}$ for tag sequence $\boldsymbol{x}$ using the data $\mathcal{D} = [(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})]_{i=1}^{N}$. To get an intuition about the algorithm, consider a partition of the set of examples $\mathcal{D}$ into $\mathcal{D}(\boldsymbol{x}) = \{(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)}) \in \mathcal{D} | \boldsymbol{x}^{(i)} = \boldsymbol{x}\}$, i.e. each section in the partition has an identical sequence of part of speech tags. Assume for this section $|\mathcal{D}(\boldsymbol{x})|$ is large (we address the data sparsity issue in §8.3.4).

We can then proceed by learning how to map a POS sequence $\boldsymbol{x}$ to a tree $t \in \mathcal{T}$ (through $u \in \boldsymbol{\mathcal{U}}$) by focusing only on examples in $\mathcal{D}(\boldsymbol{x})$.

Directly attempting to maximize the likelihood unfortunately results in an intractable optimization problem and greedy heuristics are often employed (Harmeling and Williams, 2011). Instead we propose a method that is provably consistent and returns a tree that can be mapped to a bracketing using h_{dir}.

If all the variables were observed, then the Chow-Liu algorithm (Chow and Liu, 1968c) could be used to find the most likely tree structure $u \in \boldsymbol{\mathcal{U}}$. The Chow-Liu algorithm essentially computes the distances among all pairs of variables (the negative of the mutual information) and then finds the minimum cost tree. However, the fact that the z_i are latent variables makes this strategy substantially more complicated. In particular, it becomes challenging to compute the distances among pairs of latent variables. What is needed is a "special" distance function that allows us to reverse engineer the distances among the latent variables given the distances among the observed variables. This is the key idea behind additive tree metrics that are the basis of our approach.

In the following sections, we describe the key steps to our method. §3.1 and §3.2 largely describe existing background on additive tree metrics and latent tree structure learning, while §3.3 and §3.4 discuss novel aspects that are unique to our problem.

8.3.1 Additive Tree Metrics

Let $u(\boldsymbol{x})$ be the true undirected tree of sentence $\boldsymbol{x}$ and assume the nodes $\mathcal{V}$ to be indexed by $[M] = \{1, \ldots, M\}$ such that $M = |\mathcal{V}| = H + \ell$. Furthermore, let $v \in \mathcal{V}$ refer to a node in the

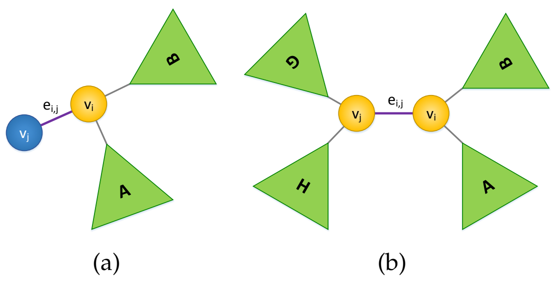


Figure 8.3: Two types of edges in general undirected latent trees. (a) leaf edge, (b) internal edge

undirected tree (either observed or latent). We assume the existence of a distance function that allows us to compute distances between pairs of nodes. For example, as we see in §8.3.2 we will define the distance $d(i, j)$ to be a function of the covariance matrix $\mathbb{E}[v_i v_j^\top | u(\boldsymbol{x}), \theta(\boldsymbol{x})]$. Thus if v_i and v_j are both observed variables, the distance can be directly computed from the data.

Moreover, the metrics we construct are such that they are *tree additive*, defined below:

Definition 2. *A function $d_{u(\boldsymbol{x})} : [M] \times [M] \to \mathbb{R}$ is an* additive tree metric *(Erdõs et al., 1999) for the undirected tree $u(\boldsymbol{x})$ if it is a distance metric,*[2] *and furthermore, $\forall i, j \in [M]$ the following relation holds:*

$$d_{u(\boldsymbol{x})}(i, j) = \sum_{(a,b) \in \text{path}_{u(\boldsymbol{x})}(i,j)} d_{u(\boldsymbol{x})}(a, b) \tag{8.2}$$

where $\text{path}_{u(\boldsymbol{x})}(i, j)$ is the set of all the edges in the (undirected) path from i to j in the tree $u(\boldsymbol{x})$.

As we describe below, given the tree structure, the additive tree metric property allows us to compute "backwards" the distances among the latent variables as a function of the distances among the observed variables.

Define $\boldsymbol{D}$ to be the $M \times M$ distance matrix among the M variables, i.e. $D_{ij} = d_{u(\boldsymbol{x})}(i, j)$. Let $\boldsymbol{D}_{WW}$, $\boldsymbol{D}_{ZW}$ (equal to $\boldsymbol{D}_{WZ}^\top$), and $\boldsymbol{D}_{ZZ}$ indicate the word-word, latent-word and latent-latent sub-blocks of $\boldsymbol{D}$ respectively. In addition, since $u(\boldsymbol{x})$ is assumed to be known from context, we denote $d_{u(\boldsymbol{x})}(i, j)$ just by $d(i, j)$.

Given the fact that the distance between a pair of nodes is a function of the random variables they represent (according to the true model), only $\boldsymbol{D}_{WW}$ can be empirically estimated from data. However, if the underlying tree structure is known, then Definition 2 can be leveraged to compute $\boldsymbol{D}_{ZZ}$ and $\boldsymbol{D}_{ZW}$ as we show below.

We first show how to compute $d(i, j)$ for all i, j such that i and j are adjacent to each other in $u(\boldsymbol{x})$, based only on observed nodes. It then follows that the other elements of the distance matrix can be computed based on Definition 2. To show how to compute distances between adjacent nodes, consider the two cases: **(1)** (i, j) is a leaf edge; **(2)** (i, j) is an internal edge.

[2]This means that it satisfies $d(i, j) = 0$ if and only if $i = j$, the triangle inequality and is also symmetric.

Case 1 (leaf edge, figure 8.3(a)) Assume without loss of generality that j is the leaf and i is an internal latent node. Then i must have exactly two other neighbors $a \in [M]$ and $b \in [M]$. Let A denote the set of nodes that are closer to a than i and similarly let B denote the set of nodes that are closer to b than i. Let A^* and B^* denote all the leaves (word nodes) in A and B respectively. Then using path additivity (Definition 2), it can be shown that for any $a^* \in A^*, b^* \in B^*$ it holds that:

$$d(i, j) = \frac{1}{2}\left(d(j, a^*) + d(j, b^*) - d(a^*, b^*)\right) \tag{8.3}$$

Note that the right-hand side only depends on distances between observed random variables.

Case 2 (internal edge, figure 8.3(b)) Both i and j are internal nodes. In this case, i has exactly two other neighbors $a \in [M]$ and $b \in [M]$, and similarly, j has exactly other two neighbors $g \in [M]$ and $h \in [M]$. Let A denote the set of nodes closer to a than i, and analogously for B, G, and H. Let A^*, B^*, G^*, and H^* refer to the leaves in A, B, G, and H respectively. Then for any $a^* \in A^*$, $b^* \in B^*$, $g^* \in G^*$, and $h^* \in H^*$ it can be shown that:

$$d(i, j) = \tfrac{1}{4}\Big(d(a^*, g^*) + d(a^*, h^*) + d(b^*, g^*) + d(b^*, h^*) - 2d(a^*, b^*) - 2d(g^*, h^*)\Big) \tag{8.4}$$

Empirically, one can obtain a more robust empirical estimate $\widehat{d(i, j)}$ by averaging over all valid choices of a^*, b^* in Eq. 8.3 and all valid choices of a^*, b^*, g^*, h^* in Eq. 8.4 (Desper and Gascuel, 2005).

8.3.2 Constructing a Spectral Additive Metric

In constructing our distance metric, we begin with the following assumption on the distribution in Eq. 8.1 (analogous to the assumptions made in Anandkumar et al., 2011).

Assumption 1 (Linear, Rank m, Means)**.**

$$\mathbb{E}[z_i|\pi_{\boldsymbol{x}}(z_i), \boldsymbol{x}] = \boldsymbol{A}_{(z_i|z_{\pi_{\boldsymbol{x}}(z_i)}, \boldsymbol{x})}\pi_{\boldsymbol{x}}(z_i) \ \forall i \in [H]$$

where $\boldsymbol{A}_{(z_i|\pi_x(z_i), \boldsymbol{x})} \in \mathbb{R}^{m\times m}$ *has rank* m.

$$\mathbb{E}[w_i|\pi_{\boldsymbol{x}}(w_i), \boldsymbol{x}] = \boldsymbol{C}_{(w_i|\pi_x(w_i), \boldsymbol{x})}\pi_{\boldsymbol{x}}(w_i) \ \forall i \in [\ell(\boldsymbol{x})]$$

where $\boldsymbol{C}_{(w_i|\pi_x(w_i), \boldsymbol{x})} \in \mathbb{R}^{p\times m}$ *has rank* m.

Also assume that $\mathbb{E}[z_i z_i^\top|\boldsymbol{x}]$ *has rank* $m \ \forall i \in [H]$.

Note that the matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ are a direct function of $\theta(\boldsymbol{x})$, but we do not specify a model family for $\theta(\boldsymbol{x})$. The only restriction is in the form of the above assumption. If w_i and z_i were discrete, represented as binary vectors, the above assumption would correspond to requiring all conditional probability tables in the latent tree to have rank m. Assumption 1 allows for the w_i to be high dimensional features, as long as the expectation requirement above is satisfied. Similar assumptions are made with spectral parameter learning methods e.g. (Hsu et al., 2009), (Bailly et al., 2009), (Parikh et al., 2011), and (Cohen et al., 2012).

Furthermore, Assumption 1 makes it explicit that regardless of the size of p, the relationships among the variables in the latent tree are restricted to be of rank m, and are thus *low rank* since $p > m$. To leverage this low rank structure, we propose using the following additive metric, a normalized variant of that in Anandkumar et al. (2011):

$$d^{\text{spectral}}(i, j) = -\log \Lambda_m(\boldsymbol{\Sigma}_{\boldsymbol{x}}(i, j)) + \tfrac{1}{2} \log \Lambda_m(\boldsymbol{\Sigma}_{\boldsymbol{x}}(i, i)) + \tfrac{1}{2} \log \Lambda_m(\boldsymbol{\Sigma}_{\boldsymbol{x}}(j, j)) \tag{8.5}$$

where $\Lambda_m(\boldsymbol{A})$ denotes the product of the top m singular values of $\boldsymbol{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}}(i, j) := \mathbb{E}[v_i v_j^\top | \boldsymbol{x}]$, i.e. the uncentered cross-covariance matrix.+

We can then show that this metric is additive:

Lemma 22. *If Assumption 1 holds then, d^{spectral} is an additive tree metric (Definition 2).*

A proof is in the appendix for completeness. From here, we use d to denote d^{spectral}, since that is the metric we use for our learning algorithm.

8.3.3 Recovering the Minimal Projective Latent Tree

It has been shown (Rzhetsky and Nei, 1993) that for any additive tree metric, $u(\boldsymbol{x})$ can be recovered by solving $\arg\min_{u \in \mathcal{U}} c(u)$ for $c(u)$:

$$c(u) = \sum_{(i,j) \in \mathcal{E}_u} d(i, j). \tag{8.6}$$

where $\mathcal{E}_u$ is the set of pairs of nodes which are adjacent to each other in u and $d(i, j)$ is computed using Eq. 8.3 and Eq. 8.4.

Note that the metric d we use in defining $c(u)$ is based on the expectations from the true distribution. In practice, the true distribution is unknown, and therefore we use an approximation for the distance metric $\hat{d}$. As we discussed in §8.3.1 all elements of the distance matrix are functions of observable quantities if the underlying tree u is known. However, only the word-word sub-block $\boldsymbol{D}_{WW}$ can be directly estimated from the data without knowledge of the tree structure.

This subtlety makes solving the minimization problem in Eq. 8.6 NP-hard (Desper and Gascuel, 2005) if u is allowed to be an arbitrary undirected tree. However, if we restrict u to be in $\mathcal{U}$, as we do in the above, then maximizing $\hat{c}(u)$ over $\mathcal{U}$ can be solved using the bilexical parsing algorithm from Eisner and Satta (1999) in $O(\ell(x)^4)$. This is because the computation of the other sub-blocks of the distance matrix only depend on the partitions of the nodes shown in Figure 8.3 into A, B, G, and H, and not on the entire tree structure. For simplicity of implementation, we use a $O(\ell(x)^5)$ CKY-like dynamic programming algorithm described in more detail in §8.6.1.

Summary. We first defined a generative model that describes how a sentence, its sequence of POS tags, and its bracketing is generated (§8.2.3). First an undirected $u \in \mathcal{U}$ is generated (only as a function of the POS tags), and then u is mapped to a bracketing using a direction mapping

h_{dir}. We then showed that we can define a distance metric between nodes in the undirected tree, such that minimizing it leads to a recovery of u. This distance metric can be computed based only on the text, without needing to identify the latent information (§8.3.2). If the true distance metric is known, with respect to the true distribution that generates the words in a sentence, then u can be fully recovered by optimizing the cost function $c(u)$. However, in practice the distance metric must be estimated from data, as discussed below.

8.3.4 Estimation of d from Sparse Data

We now address the data sparsity problem, in particular that $\mathcal{D}(\boldsymbol{x})$ can be very small, and therefore estimating d for each POS sequence separately can be problematic.[3]

In order to estimate d from data, we need to estimate the covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{x}}(i, j)$ (for $i, j \in \{1, \ldots, \ell(\boldsymbol{x})\}$) from Eq. 8.5.

To give some motivation to our solution, consider estimating the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}(1, 2)$ for the tag sequence $\boldsymbol{x} = (\texttt{DT}_1, \texttt{NN}_2, \texttt{VBD}_3, \texttt{DT}_4, \texttt{NN}_5)$. $\mathcal{D}(\boldsymbol{x})$ may be insufficient for an accurate empirical estimate. However, consider another sequence $\boldsymbol{x}' = (\texttt{RB}_1, \texttt{DT}_2, \texttt{NN}_3, \texttt{VBD}_4, \texttt{DT}_5, \texttt{ADJ}_6, \texttt{NN}_7)$. Although $\boldsymbol{x}$ and $\boldsymbol{x}'$ are not identical, it is likely that $\boldsymbol{\Sigma}_{\boldsymbol{x}'}(2, 3)$ is similar to $\boldsymbol{\Sigma}_{\boldsymbol{x}}(1, 2)$ because the determiner and the noun appear in similar syntactic context. $\boldsymbol{\Sigma}_{\boldsymbol{x}'}(5, 7)$ also may be somewhat similar, but $\boldsymbol{\Sigma}_{\boldsymbol{x}'}(2, 7)$ should not be very similar to $\boldsymbol{\Sigma}_{\boldsymbol{x}}(1, 2)$ because the noun and the determiner appear in a different syntactic context.

The observation that the covariance matrices depend on local syntactic context is the main driving force behind our solution. The local syntactic context acts as an "anchor," which enhances or replaces a word index in a sentence with local syntactic context. More formally, an anchor is a function G that maps a word index j and a sequence of POS tags $\boldsymbol{x}$ to a local context $G(j, \boldsymbol{x})$. The anchor we use is $G(j, \boldsymbol{x}) = (j, x_j)$. Then, the covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{x}}$ are estimated using kernel smoothing (Hastie et al., 2009), where the smoother tests similarity between the different anchors $G(j, \boldsymbol{x})$.

Choice of kernel For our experiments, we use the kernel

$$K_\gamma(j, k, j', k' | \boldsymbol{x}, \boldsymbol{x}') = \max\left\{0, 1 - \frac{\kappa(j, k, j', k' | \boldsymbol{x}, \boldsymbol{x}')}{\gamma}\right\} \tag{8.7}$$

where γ denotes the user-specified bandwidth, and $\kappa(j, k, j', k' | \boldsymbol{x}, \boldsymbol{x}') = \dfrac{|j-k| - |j'-k'|}{|j-k| + |j'-k'|}$ if $\boldsymbol{x}(j) = \boldsymbol{x}(j')$ and $\boldsymbol{x}(k') = \boldsymbol{x}(k)$, and $\text{sign}(j-k) = \text{sign}(j'-k')$ (and ∞ otherwise).

The kernel is non-zero if and only if the tags at position j and k in $\boldsymbol{x}$ are identical to the ones in position j' and k' in $\boldsymbol{x}'$, and if the direction between j and k is identical to the one between j' and k'. Note that the kernel is not binary, as opposed to the theoretical kernel in the §8.3.6. Our experiments show that using a non-zero value different than 1 that is a function of the distance

[3]This data sparsity problem is quite severe – for example, the Penn treebank (Marcus et al., 1993) has a total number of 43,498 sentences, with 42,246 *unique* POS tag sequences, averaging $|\mathcal{D}(\boldsymbol{x})|$ to be 1.04.

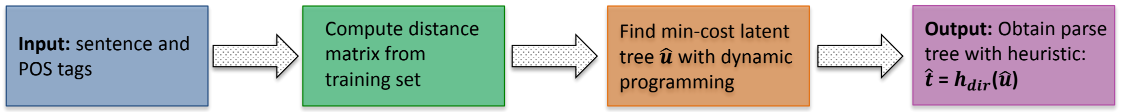

Input: sentence and POS tags
Compute distance matrix from training set
Find min-cost latent tree $\hat{u}$ with dynamic programming
Output: Obtain parse tree with heuristic: $\hat{t} = h_{dir}(\hat{u})$

Figure 8.4: Flowchart that gives an overview of Algorithm 15

between j and k compared to the distance between j' and k' does better in practice. Moreover, we experimented with more complicated kernels that take into account larger context, but empirically these did not affect performance much while increasing computational complexity significantly.

Example: Consider the following tag sequences with associated sentences.

$x = (\texttt{DT}_1, \texttt{NN}_2, \texttt{VBD}_3, \texttt{DT}_4, \texttt{NN}_5)$
(1) *The bear ate the fish*
(2) *An elephant drank the water*

$x' = (\texttt{RB}_1, \texttt{DT}_2, \texttt{NN}_3, \texttt{VBD}_4, \texttt{DT}_5, \texttt{ADJ}_6, \texttt{NN}_7)$
(3) *Slowly, a tortoise ran the rainy race*

Let $\widehat{\Gamma}_x(i,j) := \frac{1}{\sum_{n=1}^N \mathbb{I}[x^{(n)}=x]} \sum_{n=1}^N \mathbb{I}[x^{(n)} = x] w_i^{(n)} (w_j^{(n)})^\top$. This is the empirical estimate without any kernel smoothing. For example, $\widehat{\Gamma}_x(1,2) = \frac{1}{2} w_1^{(1)} (w_2^{(1)})^{(\top)} + \frac{1}{2} w_1^{(2)} (w_2^{(2)})^{(\top)}$

Now consider the kernel in Eq. 8.7. This kernel only has a non-zero value for pairs (i,j) s.t. $i < j$ where the tag at position i is DT and the tag at position j is NN. This leaves us with the terms $x(1,2)$, $x(1,5)$, $x(4,5)$, $x'(2,3)$, $x'(2,7)$, $x'(5,7)$:

$$\begin{aligned}\widehat{\Sigma}_x(1,2) = &\, K_\gamma(1,2,1,2|x,x)\widehat{\Gamma}_x(1,2) + K_\gamma(1,2,1,5|x,x)\widehat{\Gamma}_x(1,5) \\ &+ K_\gamma(1,2,4,5|x,x)\widehat{\Gamma}_x(4,5) + K_\gamma(1,2,2,3|x,x')\widehat{\Gamma}_{x'}(2,3) \\ &+ K_\gamma(1,2,2,7|x,x')\widehat{\Gamma}_{x'}(2,7) + K_\gamma(1,2,5,7|x,x')\widehat{\Gamma}_{x'}(5,7) \qquad (8.8)\end{aligned}$$

8.3.5 Overall algorithm

The full learning algorithm is given in Algorithm 15 and a high-level flow chart is given in Figure 8.4. The first step in the algorithm is to estimate distances $\hat{d}^{\text{spectral}}(j,k)$ for all position pairs j,k. This involves estimating the covariance matrix block $\widehat{\Sigma}_{x^{(i)}}(j,k)$ for each training example $x^{(i)}$ and each pair of preterminal positions (j,k) in $x^{(i)}$. Instead of computing this block by computing the empirical covariance matrix for positions (j,k) in the data $\mathcal{D}(x)$, the algorithm uses all of the pairs (j',k') from all of N training examples using kernel smoothing with bandwidth γ to obtain a more robust estimate. Then, the minimum cost latent tree is computed using dynamic programming and directed into a parse tree using the heuristic h_{dir}. The dynamic programming algorithm is described in more detail in §8.6.1.

Algorithm 15 The learning algorithm for finding the latent structure from a set of examples $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$, $i \in [N]$.

Inputs: Set of examples $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$ for $i \in [N]$, a kernel $K_\gamma(j, k, j', k' | \boldsymbol{x}, \boldsymbol{x}')$, an integer m

Data structures: For each $i \in [N]$, $j, k \in \ell(\boldsymbol{x}^{(i)})$ there is a (uncentered) covariance matrix $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(i)}}(j, k) \in \mathbb{R}^{p \times p}$, and a distance $\hat{d}^{\text{spectral}}(j, k)$.

Algorithm:

(Distance matrix estimation) $\forall i \in [N]$, $j, k \in \ell(\boldsymbol{x}^{(i)})$

- Let $C_{j',k'|i'} = w_{j'}^{(i')}(w_{k'}^{(i')})^\top$, $k_{j,k,j',k',i,i'} = K_\gamma(j, k, j', k' | \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i')})$ and $\ell_{i'} = \ell(\boldsymbol{x}^{(i')})$, and estimate each $p \times p$ covariance matrix as:

$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j, k) = \frac{\sum_{i'=1}^{N} \sum_{j'=1}^{\ell_{i'}} \sum_{k'=1}^{\ell_{i'}} k_{j,k,j',k',i,i'} C_{j',k'|i'}}{\sum_{i'=1}^{N} \sum_{j'=1}^{\ell_{i'}} \sum_{k'=1}^{\ell_{i'}} k_{j,k,j',k',i,i'}}$$

- Compute $\hat{d}^{\text{spectral}}(j, k)$ $\forall j, k \in \ell(\boldsymbol{x}^{(i)})$ using Eq. 8.5.

(Uncover structure) $\forall i \in [N]$ Find $\hat{u}^{(i)} = \arg\min_{u \in \mathcal{U}} \hat{c}(u)$, and for the ith example, return the structure $h_{\text{dir}}(\hat{u}^{(i)})$. (This step is described in more detail in Algorithm 17 and Algorithm 18 in §8.6.1)

8.3.6 Theoretical Guarantees

Our main theoretical guarantee is that under the model assumptions, the learning Algorithm 15 will recover the correct undirected tree $u \in \mathcal{U}$ with high probability, if the given top bracket is correct and if we obtain a sufficient number of examples $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})$ being generated from the model in §2. Note if the top bracket is correct this also implies that the algorithm recovers the correct directed tree $t \in \mathcal{T}$.

Our kernel is controlled by its "bandwidth" γ, a typical kernel parameter that appears when using kernel smoothing. The larger this positive number is, the more inclusive the kernel will be with respect to examples in $\mathcal{D}$ in order to estimate a given covariance matrix.

In order for the learning algorithm to be consistent the kernel must be able to effectively manage the bias-variance trade-off with the bandwidth parameter. Intuitively, if the sample size is small, then the bandwidth should be relatively large to control the variance. As the sample size increases, the bandwidth should be decreased at a certain rate to reduce the bias.

In kernel density estimation in $\mathbb{R}^p$, one can put smoothness assumptions on the density being estimated, such as bounded second derivatives. With these conditions, it can be shown that setting $\gamma = O(N^{-1/5})$ will optimally trade-off the bias and the variance in a way that leads to a consistent estimator.

However, our space of possible sequences is discrete and thus it is much more difficult to define these analogous smoothness conditions. Therefore, giving an asymptotic rate to set the bandwidth as a function of the sample size is difficult. To solve this issue, we consider a specific theoretical kernel where the bandwidth directly relates to the bias of the estimator. Define the following γ-ball

$$B_{j,k,\boldsymbol{x}}(\gamma) = \{(j',k',\boldsymbol{x}') : ||\boldsymbol{\Sigma}_{\boldsymbol{x}'}(j',k') - \boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)||_F \leq \gamma\} \tag{8.9}$$

where $||A||_F$ for a matrix A is the Frobenius norm of that matrix, i.e.: $||A||_F = \sqrt{\sum_{j,k} A_{jk}^2}$.

We then define the following "theoretical" kernel:

$$K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}') = \begin{cases} 1 & : (j',k',\boldsymbol{x}') \in B_{(j,k,\boldsymbol{x})}(\gamma) \\ 0 & : (j',k',\boldsymbol{x}') \notin B_{(j,k,\boldsymbol{x})}(\gamma) \end{cases} \tag{8.10}$$

Note that this kernel is for theoretical purposes only and is not the same kernel that is used in our experiments. It remains unclear how to generalize our theory to encompass our empirical kernel.

Furthermore, to quantify the expected effective sample size using kernel smoothing we define the following quantity:

$$\nu_{j,k,\boldsymbol{x}}(\gamma) = \left(\sum_{\boldsymbol{x}' \in \mathcal{T}^*} \frac{p(\boldsymbol{x}')}{\ell(\boldsymbol{x}')^2} \sum_{j',k' \in [\ell(\boldsymbol{x}')]} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}') \right) \tag{8.11}$$

where $p(\boldsymbol{x})$ the prior distribution over tag sequences. Let $\nu_{\boldsymbol{x}}(\gamma) = \min_{j,k} \nu_{j,k,\boldsymbol{x}}(\gamma)$. Intuitively, $\nu_{j,k,\boldsymbol{x}}(\gamma)$ represents the probability of finding contexts that are "similar" to $(j,k,\boldsymbol{x})$ as defined by the kernel. Consequently, $N\nu_{\boldsymbol{x}}(\gamma)$ represents a lower bound on the expected effective sample size for tag sequence $\boldsymbol{x}$.

Denote $\sigma_{\boldsymbol{x}}(j,k)^{(r)}$ as the r^{th} singular value of $\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)$. Let $\sigma^*(x) := \min_{j,k \in \ell(x)} \min\left(\sigma_{\boldsymbol{x}}(j,k)^{(m)}\right)$. Finally, define ϕ as the difference of the maximum possible entry in $\boldsymbol{w}$ and the minimum possible entry in $\boldsymbol{w}$ (i.e. we assume that the embeddings are bounded vectors).

Using the above definitions and leveraging the proof technique in Zhou et al. (2010), we establish that our strategy will result in a consistent estimation of the word-word distance subblock $\boldsymbol{D}_{WW}$.

Lemma 23. *Assume the kernel used is that in Eq. (8.10) with bandwidth $\gamma = \frac{C_1 \epsilon \sigma^*(x)}{m}$ and that*

$$N \geq \frac{m^2 \phi^2}{C_3 \epsilon^2 \sigma^*(\boldsymbol{x})^2 \nu_{\boldsymbol{x}}(\gamma)^2} \log\left(\frac{C_2 p^2 \ell(\boldsymbol{x})^2}{\delta}\right) \tag{8.12}$$

Then, for a fixed tag sequence $\boldsymbol{x}$, and any $\epsilon < \frac{\sigma^*(x)}{2}$:

$$|\widehat{d^{\text{spectral}}}(j,k) - d^{\text{spectral}}(j,k)| \leq \epsilon \quad \forall j \neq k \in \ell(\boldsymbol{x}) \tag{8.13}$$

with probability $1 - \delta$.

The proof is in §8.6.3. We then have the following theorem.

Theorem 4. *Define*

$$\triangle(\boldsymbol{x}) := \frac{\min_{u' \in \mathcal{U}: u' \neq u(\boldsymbol{x})}(c(u(\boldsymbol{x})) - c(u'))}{8|\ell(\boldsymbol{x})|} \tag{8.14}$$

where $u(\boldsymbol{x})$ is the correct tree and u' is any other tree.

Let $\hat{u}$ be the estimated tree for tag sequence $\boldsymbol{x}$. Assume that the kernel in Eq. (8.10) is used with bandwidth $\gamma = \frac{\triangle(x)\sigma^*(x)}{C_4 m}$ and that

$$N \geq \frac{C_5 m^2 \phi^2 \log\left(\frac{C_2 p^2 \ell(x)^2}{\delta}\right)}{\min(\sigma^*(\boldsymbol{x})^2 \triangle(\boldsymbol{x})^2, \sigma^*(\boldsymbol{x})^2)\nu_{\boldsymbol{x}}(\gamma)^2} \tag{8.15}$$

Then with probability $1 - \delta$, $\hat{u} = u(\boldsymbol{x})$. (If the top bracket is correct, this also implies that $\hat{t} = t(\boldsymbol{x})$)

Proof. By Lemma 23, and this choice of N and γ,

$$|d^{\text{spectral}}(j,k) - d^{\text{spectral}}(j,k)| \leq \triangle(\boldsymbol{x}) \quad \forall j \neq k \in \ell(\boldsymbol{x}) \tag{8.16}$$

Now for any tree $u \in \mathcal{U}$, we can compute $c(u)$ by summing over the distances $d(i,j)$ where i and j are adjacent in u.

Using Eqs. 8.3 and 8.4, Eq. (8.16) implies that

$$|\widehat{d(i,j)} - d(i,j)| \leq 2\triangle(\boldsymbol{x}) \quad \forall j \neq k \in [M] \tag{8.17}$$

Since there are $\leq 2|\ell(\boldsymbol{x})|$ edges in the tree, this is sufficient to guarantee that $|\hat{c}(u) - c(u)| \leq 4|\ell(\boldsymbol{x})|\triangle(\boldsymbol{x})$ $\forall u \in \mathcal{U}$. Thus,

$$\hat{c}(u(\boldsymbol{x})) - \hat{c}(u') < 0 \quad \forall u' \in \mathcal{U} : u' \neq u(\boldsymbol{x}) \tag{8.18}$$

Thus, the correct tree is the one with minimum estimated cost and Algorithm 15 will return the correct tree. □

8.4 Experiments

We report results on three different languages: English, German, and Chinese. For English we use the Penn treebank (Marcus et al., 1993), with sections 2–21 for training and section 23 for final testing. For German and Chinese we use the Negra treebank and the Chinese treebank respectively and the first 80% of the sentences are used for training and the last 20% for testing. All punctuation from the data is removed.[4]

We primarily compare our method to the constituent-context model (CCM) of Klein and Manning (2002). We also compare our method to the algorithm of Seginer (2007).

[4]We make brief use of punctuation for our top bracket heuristic detailed below before removing it.

Length	CCM	CCM-U	CCM-OB	CCM-UB
≤ 10	72.5	57.1	58.2	62.9
≤ 15	54.1	36	24	23.7
≤ 20	50	34.7	19.3	19.1
≤ 25	47.2	30.7	16.8	16.6
≤ 30	44.8	29.6	15.3	15.2
≤ 40	26.3	13.5	13.9	13.8

Table 8.1: Comparison of different CCM variants on English (training). U stands for universal POS tagset, OB stands for conjoining original POS tags with Brown clusters and UB stands for conjoining universal POS tags with Brown clusters. The best setting is just the vanilla setting, CCM.

8.4.1 Experimental Settings

Top bracket heuristic Our algorithm requires the top bracket in order to direct the latent tree. In practice, we employ the following heuristic to find the bracket using the following three steps:

- If there exists a comma/semicolon/colon at index i that has at least a verb before i and both a noun followed by a verb after i, then return $([0, i-1], [i, \ell(x)])$ as the top bracket. (Pick the rightmost comma/semicolon/colon if multiple satisfy the criterion).
- Otherwise find the first non-participle verb (say at index j) and return $([0, j-1], [j, \ell(\boldsymbol{x})])$.
- If no verb exists, return $([0, 1], [1, \ell(\boldsymbol{x})])$.

Word embeddings As mentioned earlier, each w_i can be an arbitrary feature vector. For all languages we use Brown clustering (Brown et al., 1992) to construct a $\log(C) + C$ feature vector where the first $\log(C)$ elements indicate which mergable cluster the word belongs to, and the last C elements indicate the cluster identity. For English, more sophisticated word embeddings are easily obtainable, and we experiment with neural word embeddings (Turian et al., 2010) of length 50. We also explored two types of CCA embeddings: OSCCA and TSCCA, given in Dhillon et al. (2012b). The OSCCA embeddings behaved better on the English the dataset, so we only report their results[5].

Choice of data For CCM, we found that if the full dataset (all sentence lengths) is used in training, then performance degrades when evaluating on sentences of length ≤ 10. We therefore restrict the data used with CCM to sentences of length $\leq \ell$, where ℓ is the maximal sentence length being evaluated. This does not happen with our algorithm, which manages to leverage lexical information whenever more data is available. We therefore use the full data for our method for all lengths.

We also experimented with the original POS tags and the universal POS tags of Petrov et al. (2011). Here, we found out that our method does better with the universal part of speech tags. For

[5]The brown clusters and neural word embeddings were obtained from http://metaoptimize.com/projects/wordreprs/. Unfortunately this website is no longer online. The CCA embeddings were obtained from Paramveer Dhillon.

	ℓ	English							German			Chinese		
		NN-O	NN	CC-O	CC	BC-O	BC	CCM	BC-O	BC	CCM	BC-O	BC	CCM
train	≤ 10	70.9	69.2	70.4	68.7	71.1	69.3	72.5	64.6	59.9	62.6	64.9	57.3	46.1
	≤ 20	55.1	53.5	53.2	51.6	53.0	51.5	50	52.7	48.7	47.9	51.4	46	22.4
	≤ 40	46.1	44.5	43.6	41.9	43.3	41.8	26.3	46.7	43.6	19.8	42.6	38.6	15
test	≤ 10	69.2	66.7	68.3	65.5	68.9	66.1	70.5	66.4	61.6	64.7	58.0	53.2	40.7
	≤ 15	60.3	58.3	58.6	56.4	58.6	56.5	53.8	57.5	53.5	49.6	54.3	49.4	35.9
	≤ 20	54.1	52.3	52.3	50.3	51.9	50.2	50.4	52.8	49.2	48.9	49.7	45.5	20.1
	≤ 25	50.8	49.0	48.6	46.6	48.3	46.6	47.4	50.0	46.8	45.6	46.7	42.7	17.8
	≤ 30	48.1	46.3	45.6	43.7	45.4	43.8	44.9	48.3	45.4	21.9	44.6	40.7	16.1
	≤ 40	45.5	43.8	43.0	41.1	42.7	41.1	26.1	46.9	44.1	20.1	42.2	38.6	14.3

Table 8.2: F_1 bracketing measure for the test sets and train sets in three languages. NN, CC, and BC indicate the performance of our method for neural embeddings, CCA embeddings, and Brown clustering respectively, using the heuristic for h_{dir} described in § 8.4.1. NN-O, CC-O, and BC-O indicate that the oracle (i.e. true top bracket) was used for h_{dir}.

CCM, we also experimented with the original parts of speech, universal tags (CCM-U), the cross-product of the original parts of speech with the Brown clusters (CCM-OB), and the cross-product of the universal tags with the Brown clusters (CCM-UB). The results in Table 8.1 indicate that the vanilla setting is the best for CCM.

Thus, for all results, we use universal tags for our method and the original POS tags for CCM. We believe that our approach substitutes the need for fine-grained POS tags with the lexical information. CCM, on the other hand, is fully unlexicalized.

Parameter Selection Our method requires two parameters, the latent dimension m and the bandwidth γ. CCM also has two parameters, the number of extra constituent/distituent counts used for smoothing. For both methods we chose the best parameters for sentences of length $\ell \leq 10$ on the English Penn Treebank (training) and used this set for all other experiments. This resulted in $m = 7, \gamma = 0.4$ for our method and 2, 8 for CCM's extra constituent/distituent counts respectively. We also tried letting CCM choose different hyperparameters for different sentence lengths based on dev-set likelihood, but this gave worse results than holding them fixed.

8.4.2 Results

Test I: Accuracy Table 8.2 summarizes our results. CCM is used with the initializer proposed in Klein and Manning (2002).[6] NN, CC, and BC indicate the performance of our method for neural embeddings, CCA embeddings, and Brown clustering respectively, using the heuristic for h_{dir} described in § 8.4.1. NN-O, CC-O, and BC-O indicate that the oracle (i.e. true top bracket) was used for h_{dir}. For our method, test set results can be obtained by using Algorithm 15 (except the distances are computed using the training data).

For English, while CCM behaves better for short sentences ($\ell \leq 10$), our algorithm is more robust

[6]We used the implementation available at http://tinyurl.com/lhwk5n6.

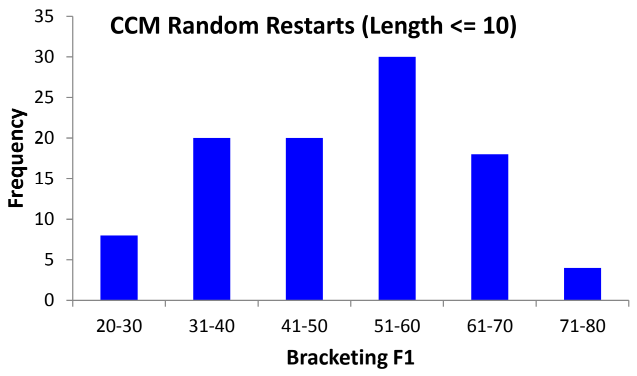


Figure 8.5: Histogram showing performance of CCM across 100 random restarts for sentences of length ≤ 10.

with longer sentences. This is especially noticeable for length ≤ 40, where CCM breaks down and our algorithm is more stable. We find that the neural embeddings modestly outperform the CCA and Brown cluster embeddings.

The results for German are similar, except CCM breaks down earlier at sentences of $\ell \leq 30$. For Chinese, our method substantially outperforms CCM for all lengths. Note that CCM performs very poorly, obtaining only around 20% accuracy even for sentences of $\ell \leq 20$. We didn't have neural embeddings for German and Chinese (which worked best for English) and thus only used Brown cluster embeddings.

For English, the disparity between NN-O (oracle top bracket) and NN (heuristic top bracket) is rather low suggesting that our top bracket heuristic is rather effective. However, for German and Chinese note that the "BC-O" performs substantially better, suggesting that if we had a better top bracket heuristic our performance would increase.

Test II: Sensitivity to initialization The EM algorithm with the CCM requires very careful initialization, which is described in Klein and Manning (2002). If, on the other hand, random initialization is used, the variance of the performance of the CCM varies greatly. Figure 8.5 shows a histogram of the performance level for sentences of length ≤ 10 for different random initializers. As one can see, for some restarts, CCM obtains accuracies lower than 30% due to local optima. Our method does not suffer from local optima and thus does not require careful initialization.

Test III: Comparison to Seginer's algorithm Our approach is not directly comparable to Seginer's because he uses punctuation, while we use POS tags. Using Seginer's parser we were able to get results on the training sets. On English: 75.2% ($\ell \leq 10$), 64.2% ($\ell \leq 20$), 56.7% ($\ell \leq 40$). On German: 57.8% ($\ell \leq 10$), 45.0% ($\ell \leq 20$), and 39.9% ($\ell \leq 40$). On Chinese: 56.6% ($\ell \leq 10$), 45.1% ($\ell \leq 20$), and 38.9% ($\ell \leq 40$).

Thus, while Seginer's method performs better on English, our approach performs 2-3 points better on German, and both methods give similar performance on Chinese.

8.5 Conclusion

We described a spectral approach for unsupervised constituent parsing that comes with theoretical guarantees on latent structure recovery. Empirically, our algorithm performs favorably to the CCM of Klein and Manning (2002) without the need for careful initialization.

The tree metric used in this work is closely related to that used in Chapter 6 except it is finite dimensional. However, the meta-algorithm used to learn the tree structure is rather different. Because of the domain, we are able to restrict the space of tree structures to those that are projective, thus enabling us to find the minimum cost tree. In Chapter 6, however, we did not have any restrictions on the tree structure, and thus solving for the minimum cost tree is NP-hard. Therefore in Chapter 6, we resorted to using neighbor joining (Saitou and Nei, 1987), which is consistent but does not have any guarantees when the model assumptions do not hold (since it is greedy). On the other hand, our meta-algorithm will find the minimum cost tree regardless of whether the data was generated by the assumed model or not.

In the future, it would be interesting to explore dependency parsing with these ideas. There has been considerably more existing work on dependency parsing methods (Cohen and Smith, 2009; Headden et al., 2009; Spitkovsky et al., 2010b,a; Gillenwater et al., 2010; Golland and DeNero, 2012), and it would be interesting to see how our approach compares to these. As an initial attempt, we tried learning dependency trees using the Chow Liu algorithm for fully observed trees (see Algorithm 1 in Chapter 2), instead of the latent trees / additive metrics, but performance was not great.

8.6 Appendix

8.6.1 Details on recovering minimum cost latent tree

In this section we give more details on the dynamic programming algorithm used to recover the projective latent tree. We describe a computational algorithm with $O(\ell(\boldsymbol{x})^5)$ complexity based on the CKY algorithm (Manning and Schütze, 1999). A faster algorithm ($O(\ell(\boldsymbol{x})^4)$) is possible based on bilexical parsing algorithms (Eisner and Satta, 1999) but more difficult to implement.

Note that the distance function d is invariant to the direction of the tree. However, since we are restricting $u \in \mathcal{U}$ (set of projective trees with respect to h_{dir}) we must perform the optimization directly over directed trees $t \in \mathcal{T}$ since projectivity is only a property of a directed tree.

Define $c_{opt}([i, f-1], [f, j])$ to be the optimal cost for a subtree that spans $[i, j]$ with the top split $[f-1, f]$, and $t_{opt}([i, f-1], [f, j])$ be the associated tree. Similarly let $t([i, f-1], [f, j])$ denote some tree (not necessarily optimal) that spans $[i, j]$ with the top split $[f-1, f]$. A tree is considered complete if it spans $[1, \ell]$ where ℓ is the length of the sentence. Otherwise it is considered to be incomplete.

If a tree $t([i, f-1], [f, j])$ is incomplete, then it has a root node that we denote as $r_{[i,f-1],[f,j]}$ [7].

[7] note that the complete tree does not have a root node, but rather a root edge

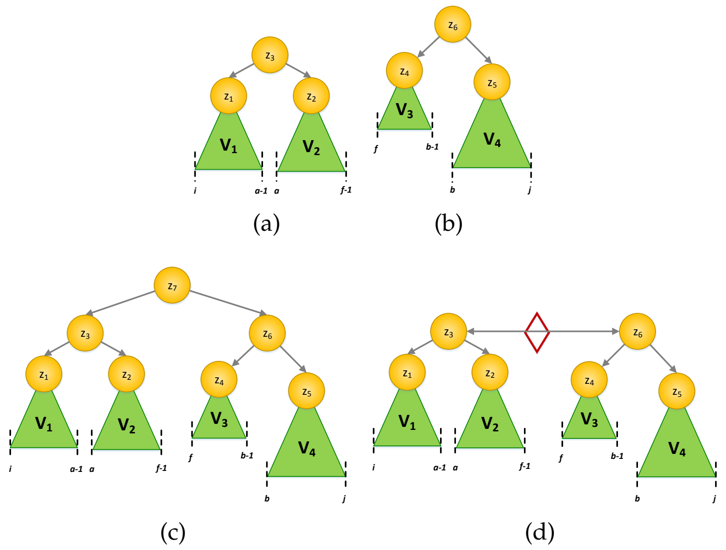


Figure 8.6: Example of merging: (a) tree 1: $t([i, a-1], [a, f-1])$, (b) tree 2: $t([f, b-1], [b, j])$ (c) resulting tree from merging tree 1 and tree 2 (incomplete case): $t([i, f-1], [f, j])$ (d) merged tree from merging tree 1 and tree 2 (complete case): $t([i, f-1], [f, j])$

Algorithm 16 Merge two latent trees

In: tag sequence $\boldsymbol{x}$, and two trees to merge: $t([i, a-1, [a, f-1])$ and $t([f, b-1], [b, j])$
Out: merged tree $t([i, f-1], [f, j])$

```
1: if $i = 1$ and $j = \ell(\boldsymbol{x})$ then
2:     Add edge $(r_{[i,a-1],[a,f]}, r_{[f,b-1],[b,j]})$
3: else
4:     Add node $r_{[i,f-1],[f,j]}$
5:     Add edges $(r_{[i,a-1],[a,f]}, r_{[i,f-1],[f,j]})$ and $(r_{[f,b-1],[b,j]}, r_{[i,f-1],[f,j]})$
6: end if
```

An example of two incomplete trees are shown Figure 8.6(a) and Figure 8.6(b). In Figure 8.6(a), $r_{[i,a-1],[a,f]} = z_3$ and $r_{[f,b-1],[b,j]} = z_6$

Two trees with adjacent spans, e.g. $t([i, a-1, [a, f-1])$ and $t([f, b-1], [b, j])$, can be merged into one tree $t([i, f-1], [f, j])$. The merging process is slightly different depending on whether $t([i, f-1], [f, j])$ is an incomplete or a complete tree. If $t([i, f-1], [f, j])$ is incomplete then a new root node $r_{[i,f-1],[f,j]}$ is added and then connected to the subtrees. For example in Figure 8.6(c), z_7 is the new root and the edges (z_3, z_7) and (z_6, z_7) have been added.

On the other hand, if $t([i, f-1], [f, j])$ then a new root node is not added. Instead we simply add an edge between the two roots of the subtrees being merged. For example in Figure 8.6(d), the edge (z_3, z_6) is added[8].

We now decompose the cost function recursively in order to reach the dynamic programming

[8]The reason for this is technical: in a graphical model all internal nodes with less than 3 neighbors are redundant, since they can be marginalized out and the resulting model will still be a tree.

solution. If $i = 1$ and $j = \ell(x)$, then we can see that $c_{opt}([i, f-1], [f, j])$ can be recursively written as:

$$c_{opt}([i,f-1],[f,j]) = \min_{a,b}\Big(d(r_{[i,a-1],[a,f-1]}, r_{[f,b-1],[b,j]}) + c_{opt}([i,a-1],[a,f-1]) + c_{opt}([f,b-1],[b,j])\Big) \tag{8.19}$$

Note that because of Eq. 8.4, $d(r_{[i,a-1],[a,f-1]}, r_{[f,b-1],[f,j]})$ is only a function of the partition of the leaves into the sets A^*, B^*, G^* and H^*. In this case the four sets are $A^* = [i, a-1]$, $B^* = [a, f-1]$, $G^* = [f, b-1]$, and $H^* = [b, j]$. In Figure 8.6(d), $A^* = V_1^*$, $B^* = V_2^*$, $G^* = V_3^*$, $H^* = V_4^*$.

Decomposing recursively gives

$$\begin{aligned} c_{opt}([i,f-1],[f,j]) = \min_{a,b}\Big(&d(r_{[i,a-1],[a,f-1]}, r_{[i,f-1],[f,j]}) + d(r_{[i,f-1],[f,j]}, r_{[f,b-1],[b,j]}) \\ &+ c_{opt}([i,a-1],[a,f-1]) + c_{opt}([f,b-1],[b,j])\Big) \end{aligned} \tag{8.20}$$

Based on Eq. 8.4, $d(r_{[i,a-1],[a,f-1]}, r_{[i,f-1],[f,j]})$ depends on the partitions $A^* = [i, a-1], B^* = [a, f-1]$, $G^* = [f, j]$, and $H^* = \{x_1, ..., x_\ell\} \setminus \{[i, j]\}$. Similarly, using Eq. 8.4, $d(r_{[i,f-1],[f,j]}, r_{[f,b-1],[b,j]})$ is a function of the partitions $A^* = [f, b-1], B^* = [b, j]$, $G^* = [i, f-1]$, and $H^* = \{x_1, ..., x_\ell\} \setminus \{[i, j]\}$.

In the base case $c_{opt}([i, i], [j, j]) = d(i, j)$ where both i and j are leaves and adjacent words in the sentence.

Reformulating this recursion into a dynamic program gives Algorithm 17. Note that unlike the traditional CKY algorithm that only stores the cost for a span $c(i, j)$, here we are storing the cost for $c([i, f-1], [f, j])$. As a result the computational algorithm is $O(\ell(\boldsymbol{x})^5)$. A faster algorithm ($O(\ell(\boldsymbol{x})^4)$ is possible based on bilexical parsing algorithms (Eisner and Satta, 1999) but more difficult to implement.

To aid implementation, a more detailed version of Algorithm 15 is provided in Algorithm 18.

8.6.2 Proof of Lemma 22

In this section, we prove that our proposed tree metric is path additive based on the proof technique of Lemma 15 in Chapter 6.

Proof. For conciseness, we simply prove the property for paths of length 2. The proof for more general cases follows similarly (e.g. see (Anandkumar et al., 2011)).

First note that the relationship between eigenvalues and singular values allows us to rewrite the distance metric as

$$d^{\text{spectral}}(i,j) = -\frac{1}{2}\log\Lambda_m(\boldsymbol{\Sigma_x}(i,j)\boldsymbol{\Sigma_x}(i,j)^\top) + \tfrac{1}{4}\log\Lambda_m(\boldsymbol{\Sigma_x}(i,i)\boldsymbol{\Sigma_x}(i,i)^\top) + \tfrac{1}{4}\log\Lambda_m(\boldsymbol{\Sigma_x}(j,j)\boldsymbol{\Sigma_x}(j,j)^\top) \tag{8.21}$$

Furthermore, $\boldsymbol{\Sigma_x}(i,j)\boldsymbol{\Sigma_x}(i,j)^\top$ is rank m by Assumption 1 and the conditional independence

Algorithm 17 Compute minimum cost projective parse tree

In: sentence/tag sequence $(\boldsymbol{w}, \boldsymbol{x})$, distances $\hat{d}^{\text{spectral}}(j,k)\ \forall j,k \in \ell(\boldsymbol{x})$, top bracket $[(1, f^{top}-1), (f^{top}, \ell)]$
Out: parse tree $t \in \mathcal{T}$

```
1: Initialize $t_{opt}([i,i],[j,j]) = \text{Merge}(\boldsymbol{x},[w_i],[w_j])$ (via Algorithm 16) and set cost values
   $c_{opt}([i,i],[j,j]) = \hat{d}^{\text{spectral}}(i,j)\ \forall i,j$
2: for $(k = 2; k < \ell; k++)$ do
3:   for each pair $(i,j)$ s.t. $j = i + k$ and $i,j > 0$ do
4:     for $f \in \{i+1, ....j\}$ do
5:       if $i = 1$ and $j = \ell(\boldsymbol{x})$ then
6:         if $f = f^{top}$ then
7:           Compute optimal cost $c_{opt}([i,f-1],[f,j])$ according to Eq. 8.19 and let $a^{min}, b^{min}$ be
             the values of $a, b$ that achieve the minimum.
8:         else
9:           Continue
10:        end if
11:      else
12:        Compute optimal cost $c_{opt}([i,f-1],[f,j])$ according to Eq. 8.20 and let $a^{min}, b^{min}$ be the
           values of $a, b$ that achieve the minimum.
13:      end if
14:      Set $t_{opt}([i,f-1],[f,j]) = \text{Merge}(\boldsymbol{x}, t_{opt}([i,a^{min}-1],[a^{min},f-1]), t_{opt}([f,b^{min}-1],[b^{min},j]))$
         (via Algorithm 16)
15:    end for
16:  end for
17: end for
18: return $t_{opt}([(1, f^{top}-1), (f^{top}, \ell)])$
```

statements implied by the latent tree model. Thus $\Lambda_m(\boldsymbol{\Sigma_x}(i,j)\boldsymbol{\Sigma_x}(i,j)^\top)$ is equivalent to taking the pseudo-determinant $|\cdot|_+$ of $(\boldsymbol{\Sigma_x}(i,j)\boldsymbol{\Sigma_x}(i,j)^\top)$, which is the product of the non-zero eigenvalues. The pseudo-determinant can be alternatively defined in the following limit form:

$$|\boldsymbol{B}|_+ = \lim_{\alpha \to 0} \frac{|\boldsymbol{B} + \alpha \boldsymbol{I}|}{\alpha^{p-m}} \tag{8.22}$$

where $\boldsymbol{B}$ is a $p \times p$ matrix of rank m and $\boldsymbol{I}$ is the $p \times p$ identity and $|\cdot|$ equals the standard determinant. Note that if $m = p$ then $|\boldsymbol{B}|_+ = |\boldsymbol{B}|$.

Thus, our distance metric can be further rewritten as:

$$d^{\text{spectral}}(i,j) = -\frac{1}{2}\log|\boldsymbol{\Sigma_x}(i,j)\boldsymbol{\Sigma_x}(i,j)^\top|_+ + \tfrac{1}{4}\log|\boldsymbol{\Sigma_x}(i,i)\boldsymbol{\Sigma_x}(i,i)^\top|_+ + \tfrac{1}{4}\log|\boldsymbol{\Sigma_x}(j,j)\boldsymbol{\Sigma_x}(j,j)^\top|_+ \tag{8.23}$$

We now proceed with the proof. For paths of length 2, $i - j - k$, there are three cases[9]:

- $i \leftarrow j \rightarrow k$

[9] although the additive distance metric is undirected, $\boldsymbol{A}$ and $\boldsymbol{C}$ in Assumption 1 are defined with respect to parent-child relationships, so we must consider direction

Algorithm 18 The learning algorithm for finding the latent structure from a set of examples $(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)}), i \in [N]$.

```
Inputs: Set of examples $(w^{(i)}, x^{(i)})$ for $i \in [N]$, a kernel $K_\gamma(j,k,j',k'|x,x')$, an integer $m$
Data structures: For each $i \in [N]$, $j,k \in \ell(x^{(i)})$ there is a (uncentered) covariance matrix $\widehat{\Sigma}_{x^{(i)}}(j,k) \in \mathbb{R}^{p \times p}$, and a distance $\hat{d}^{\text{spectral}}(j,k)$.
Algorithm:
(Covariance estimation) $\forall i \in [N], j,k \in \ell(x^{(i)})$
    • Let $C_{j',k'|i'} = w_{j'}^{(i')}(w_{k'}^{(i')})^\top$, $k_{j,k,j',k',i,i'} = K_\gamma(j,k,j',k'|x^{(i)},x^{(i')})$ and $\ell_{i'} = \ell(x^{(i')})$, and estimate each $p \times p$ covariance matrix as:
      $\widehat{\Sigma}_x(j,k) = \frac{\sum_{i'=1}^{N}\sum_{j'=1}^{\ell_{i'}}\sum_{k'=1}^{\ell_{i'}} k_{j,k,j',k',i,i'} C_{j',k'|i'}}{\sum_{i'=1}^{N}\sum_{j'=1}^{\ell_{i'}}\sum_{k'=1}^{\ell_{i'}} k_{j,k,j',k',i,i'}}$
    • Compute $\hat{d}^{\text{spectral}}(j,k)$ $\forall j,k \in \ell(x^{(i)})$ using Eq. 8.5.
(Uncover structure) $\forall i \in [N]$
    • Select top bracket $[(1, f^{top}-1),(f^{top},\ell)]$ via the heuristic described in Section 8.4.1.
    • Find $\hat{t}^{(i)}$ by calling Algorithm 17 with arguments $(w^{(i)}, x^{(i)})$, distances $\hat{d}^{\text{spectral}}(j,k)$ $\forall j,k \in \ell(x^{(i)})$, and top bracket $[(1, f^{top}-1),(f^{top},\ell)]$ (Let $\hat{u}^{(i)}$ is the undirected version of $\hat{t}^{(i)}$)
```

- $i \leftarrow j \leftarrow k$
- $i \rightarrow j \rightarrow k$

Case 1: ($i \leftarrow j \rightarrow k$) Note that in this case j must be latent but i and k can be either observed or latent. We assume below that both i and k are observed. The same proof strategy works for the other cases.

Due to Assumption 1,

$$\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k) = \mathbb{E}[v_i v_k^\top|\boldsymbol{x}] = \boldsymbol{C}_{i|j,\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{C}_{k|j,\boldsymbol{x}}^\top \tag{8.24}$$

Thus,

$$\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)^\top = \boldsymbol{C}_{i|j,\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{C}_{k|j,\boldsymbol{x}}^\top \boldsymbol{C}_{k|j,\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \tag{8.25}$$

Combining this definition with Sylvester's Determinant Theorem (Akritas et al., 1996), gives us that:

$$
\begin{aligned}
|\boldsymbol{\Sigma}_x(i,k)\boldsymbol{\Sigma}_x(i,k)^\top|_+ &= |\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{k|j,x}^\top\boldsymbol{C}_{k|j,x}\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{i|j,x}^\top|_+ \\
&= |\boldsymbol{C}_{i|j,x}^\top\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{k|j,x}^\top\boldsymbol{C}_{k|j,x}\boldsymbol{\Sigma}_x(j,j)|_+ \qquad (8.26)
\end{aligned}
$$

(i.e. we can move $\boldsymbol{C}_{i|j,x}^\top$ the to the front).

Now $\boldsymbol{C}_{i|j,x}^\top\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{k|j,x}^\top\boldsymbol{C}_{k|j,x}\boldsymbol{\Sigma}_x(j,j)$ is $m \times m$ and has rank m. Thus, the pseudo-determinant equals the normal determinant in this case. Using the fact that $|\boldsymbol{AB}| = |\boldsymbol{A}||\boldsymbol{B}|$ if $\boldsymbol{A}$ and $\boldsymbol{B}$ are square, we get

$$
\begin{aligned}
|\boldsymbol{\Sigma}_x(i,k)\boldsymbol{\Sigma}_x(i,k)^\top|_+ &= |\boldsymbol{C}_{i|j,x}^\top\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{k|j,x}^\top\boldsymbol{C}_{k|j,x}\boldsymbol{\Sigma}_x(j,j)| \\
&= |\boldsymbol{C}_{i|j,x}^\top\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)||\boldsymbol{C}_{k|j,x}^\top\boldsymbol{C}_{k|j,x}\boldsymbol{\Sigma}_x(j,j)| \\
&= \frac{|\boldsymbol{\Sigma}_x(j,j)||\boldsymbol{C}_{i|j,x}^\top\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)|}{|\boldsymbol{\Sigma}_x(j,j)|} \times \frac{|\boldsymbol{\Sigma}_x(j,j)||\boldsymbol{C}_{k|j,x}^\top\boldsymbol{C}_{k|j,x}\boldsymbol{\Sigma}_x(j,j)|}{|\boldsymbol{\Sigma}_x(j,j)|} \\
&= \frac{|\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{i|j,x}^\top\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)|}{|\boldsymbol{\Sigma}_x(j,j)|} \times \frac{|\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{k|j,x}^\top\boldsymbol{C}_{k|j,x}\boldsymbol{\Sigma}_x(j,j)|}{|\boldsymbol{\Sigma}_x(j,j)|} \qquad (8.27)
\end{aligned}
$$

Furthermore, note that

$$
\begin{aligned}
|\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{i|j,x}^\top\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)| &= |\boldsymbol{C}_{i|j,x}\boldsymbol{\Sigma}_x(j,j)\boldsymbol{\Sigma}_x(j,j)\boldsymbol{C}_{i|j,x}^\top|_+ \\
&= |\boldsymbol{\Sigma}_x(i,j)\boldsymbol{\Sigma}_x(i,j)^\top|_+ \qquad (8.28)
\end{aligned}
$$

This gives,

$$
|\boldsymbol{\Sigma}_x(i,k)\boldsymbol{\Sigma}_x(i,k)^\top|_+ = \frac{|\boldsymbol{\Sigma}_x(i,j)\boldsymbol{\Sigma}_x(i,j)^\top|_+}{|\boldsymbol{\Sigma}_x(j,j)|} \times \frac{|\boldsymbol{\Sigma}_x(k,j)\boldsymbol{\Sigma}_x(k,j)^\top|_+}{|\boldsymbol{\Sigma}_x(j,j)|} \qquad (8.29)
$$

Substituting back into Eq. (8.23) proves that

$$
\begin{aligned}
d^{\text{spectral}}(i,k) &= -\frac{1}{2}\log|\boldsymbol{\Sigma}_x(i,j)\boldsymbol{\Sigma}_x(i,j)^\top|_+ - \frac{1}{2}\log|\boldsymbol{\Sigma}_x(j,k)\boldsymbol{\Sigma}_x(j,k)^\top|_+ + \frac{1}{2}\log|\boldsymbol{\Sigma}_x(j,j)\boldsymbol{\Sigma}_x(j,j)^\top|_+ \\
&+ \tfrac{1}{4}\log|\boldsymbol{\Sigma}_x(i,i)\boldsymbol{\Sigma}_x(i,i)^\top|_+ + \tfrac{1}{4}\log|\boldsymbol{\Sigma}_x(k,k)\boldsymbol{\Sigma}_x(k,k)^\top|_+ \\
&= d^{\text{spectral}}(i,j) + d^{\text{spectral}}(j,k) \qquad (8.30)
\end{aligned}
$$

Case 2: $i \leftarrow j \leftarrow k$ The proof is similar to above. Here since only leaf nodes can be observed,j and k must be latent but i can be either observed or latent. We assume it is observed, the latent case follows similarly.

Due to Assumption 1,

$$\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k) = \mathbb{E}[v_i v_k^\top | x] = \boldsymbol{C}_{i|j,\boldsymbol{x}} \boldsymbol{A}_{j|k,\boldsymbol{x}} \boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k) \tag{8.31}$$

Thus, via Sylvester's Determinant Theorem as before,

$$\begin{aligned} |\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)^\top|_+ &= |\boldsymbol{C}_{i|j,\boldsymbol{x}} \boldsymbol{A}_{j|k,\boldsymbol{x}} \boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{A}_{j|k,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}}^\top|_+ \\ &= |\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}} \boldsymbol{A}_{j|k,\boldsymbol{x}} \boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{A}_{j|k,\boldsymbol{x}}^\top|_+ \qquad (8.32) \end{aligned}$$

Now $\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}} \boldsymbol{A}_{j|k,\boldsymbol{x}} \boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{A}_{j|k,\boldsymbol{x}}^\top$ is $m \times m$ and has rank m. Thus, the pseudo-determinant equals the normal determinant in this case. Using the fact that $|\boldsymbol{AB}| = |\boldsymbol{A}||\boldsymbol{B}|$ if $\boldsymbol{A}$ and $\boldsymbol{B}$ are square, we get

$$\begin{aligned} |\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)^\top|_+ &= |\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}} \boldsymbol{A}_{j|k,\boldsymbol{x}} \boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{A}_{j|k,\boldsymbol{x}}^\top| \\ &= |\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}}||\boldsymbol{A}_{j|k,\boldsymbol{x}} \boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{A}_{j|k,\boldsymbol{x}}^\top| \\ &= |\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}}||\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)^\top| \\ &= \frac{|\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)||\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}}||\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)|}{|\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)||\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)|} \times |\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)^\top| \\ &= \frac{|\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)|}{|\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)||\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)|} \times |\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)^\top| \qquad (8.33) \end{aligned}$$

Furthermore, note that

$$\begin{aligned} |\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top \boldsymbol{C}_{i|j,\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)| &= |\boldsymbol{C}_{i|j,\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{C}_{i|j,\boldsymbol{x}}^\top)|_+ \qquad (8.34) \\ &= |\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)^\top|_+ \qquad (8.35) \end{aligned}$$

This gives,

$$|\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,k)^\top|_+ = \frac{|\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)^\top|_+}{|\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)||\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)|} \times |\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)^\top|_+ \tag{8.36}$$

Substituting back into Eq. (8.23) proves that

$$\begin{aligned} d^{\text{spectral}}(i,k) &= -\frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)^\top|_+ - \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)^\top|_+ + \frac{1}{2}\log|\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)^\top|_+ \qquad (8.37) \\ &+ \tfrac{1}{4}\log|\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,i)\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,i)^\top|_+ + \tfrac{1}{4}\log|\boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)\boldsymbol{\Sigma}_{\boldsymbol{x}}(k,k)^\top|_+ \qquad (8.38) \\ &= d^{\text{spectral}}(i,j) + d^{\text{spectral}}(j,k) \qquad (8.39) \end{aligned}$$

Case 3: $i \to j \to k$ The same argument as case 2 holds here.

□

8.6.3 Proof of Lemma 23

Instead of proving Lemma 23 directly, we divide it into two stages. First, we show that our strategy yields consistent estimation of the covariance matrices (Lemma 24). We then show that a concentration bound on the covariance matrices implies that the empirical distance is close to the true distance (Lemma 25). Putting the results together proves Lemma 23.

Lemma 24. *Assume the kernel used is that in Eq. (8.10) with bandwidth $\gamma = \epsilon/2$ and that*

$$N \geq \frac{\log(\frac{C_1 p^2 \ell(x)^2}{\delta})\phi^2}{C_2 \epsilon^2 \nu_x(\epsilon/2)^2} \tag{8.40}$$

Then for a fixed tag sequence $\boldsymbol{x}$ and $\forall j,k \in \ell(\boldsymbol{x})$,

$$||\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k) - \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k)||_F \leq \epsilon \tag{8.41}$$

with probability $1 - \delta$.

Proof. Define the quantity,

$$\widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k) = \frac{\sum_{i=1}^{M} \sum_{j',k' \in [\ell(\boldsymbol{x}_i)]} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}_i)\boldsymbol{\Sigma}_{\boldsymbol{x}^i}(j',k')}{\sum_{i=1}^{N} \sum_{j',k' \in [\ell(\boldsymbol{x}_i)]} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}_i)} \tag{8.42}$$

Note that via triangle inequality,

$$||\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k) - \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k)||_F \leq ||\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k) - \widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k)||_F + ||\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k) - \widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k)||_F \tag{8.43}$$

The first term (the bias) is bounded by $\epsilon/2$ using the definition of the kernel in Eq. (8.10) with bandwidth $\epsilon/2$. The proof for bound for the second term (the variance) can be derived using the technique of (Zhou et al., 2010) and is in Utility Lemma 1. □

Lemma 24 can then be used to prove that with high probability the estimated mutual information is close to the true mutual information.

Lemma 25. *Assume that*

$$||\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k) - \boldsymbol{\Sigma}_{\boldsymbol{x}}(j,k)||_F \leq \epsilon \leq \frac{\sigma^*(x)}{2} \; \forall j,k \in [\ell(\boldsymbol{x})] \tag{8.44}$$

Then,

$$|d^{\text{spectral}}(j,k) - d^{\text{spectral}}(j,k)| \leq \frac{C_3 m \epsilon}{\sigma^*(x)} \quad \forall j \neq k \in [\ell(\boldsymbol{x})] \tag{8.45}$$

Proof. By triangle inequality,

$$\begin{aligned}
&|d^{\text{spectral}}(j,k) - d^{\text{spectral}}(j,k)| \\
&\qquad \leq |\log \Lambda_m(\mathbf{\Sigma}_x(j,k)) - \log \Lambda_m(\widehat{\mathbf{\Sigma}}_x(j,k))| \\
&\qquad + \frac{1}{2}|\log \Lambda_m(\mathbf{\Sigma}_x(j,j)) - \log \Lambda_m(\widehat{\mathbf{\Sigma}}_x(j,j))| \\
&\qquad + \frac{1}{2}|\log \Lambda_m(\widehat{\mathbf{\Sigma}}_x(k,k)) - \log \Lambda_m(\widehat{\mathbf{\Sigma}}_x(k,k))| \qquad (8.46)
\end{aligned}$$

We show the bound on $|\log \Lambda_m(\mathbf{\Sigma}_x(j,j)) - \log \Lambda_m(\widehat{\mathbf{\Sigma}}_x(j,j))|$, the others follow similarly. By definition of $\Lambda_m(\cdot)$, and triangle inequality we have that,

$$|\log \Lambda_m(\mathbf{\Sigma}_x(j,j)) - \log \Lambda_m(\widehat{\mathbf{\Sigma}}_x(j,j))| \leq \sum_{r=1}^{m} |\log(\sigma_x(j,j)^{(r)}) - \log(\widehat{\sigma}_x(j,j)^{(r)})|$$

To translate our concentration bounds on Frobenius norm to concentration of eigenvalues, we apply Weyl's Theorem. Assume first that $\widehat{\sigma}_x(j,j)^{(r)} \geq \sigma_x(j,j)^{(r)}$. Then,

$$\begin{aligned}
|\log(\sigma_x(j,j)^{(r)}) - \log(\widehat{\sigma}_x(j,j)^{(r)})| &\leq |\log(\sigma_x(j,j)^{(r)}) - \log(\sigma_x(j,j)^{(r)} + \epsilon)| \\
&= |\log(\sigma_x(j,j)^{(r)}) - \log(\sigma_x(j,j)^{(r)}(1 + \frac{\epsilon}{\sigma_x(j,j)^{(r)}}))| \\
&\leq \log(1 + \frac{\epsilon}{\sigma_x(j,j)^{(r)}}) \leq \frac{\epsilon}{\sigma_x(j,j)^{(r)}} \qquad (8.47)
\end{aligned}$$

Similarly, when $\widehat{\sigma}_x(j,j)^{(r)} < \sigma_x(j,j)^{(r)}$, then

$$|\log(\sigma_x(j,j)^{(r)}) - \log(\widehat{\sigma}_x(j,j)^{(r)})| \leq \frac{\epsilon}{\widehat{\sigma}_x(j,j)^{(r)}} = \frac{\epsilon}{\sigma_x(j,j)^{(r)} - \epsilon} \qquad (8.48)$$

Using the fact that $\epsilon \leq \frac{\sigma^*(x)}{2}$ we obtain that,

$$|\log(\sigma_x(j,j)^{(r)}) - \log(\widehat{\sigma}_x(j,j)^{(r)})| \leq \frac{2\epsilon}{\sigma_x(j,j)^{(r)}} \qquad (8.49)$$

As a result,

$$|\log \Lambda_m(\mathbf{\Sigma}_x(j,j)) - \log \Lambda_m(\widehat{\mathbf{\Sigma}}_x(j,j))| \leq m\frac{2\epsilon}{\sigma_x(j,j)^{(m)}} \qquad (8.50)$$

Repeating this for the other terms in Eq. (8.46) gives the bound. □

Utility Lemma 1.

$$P(\|\widehat{\mathbf{\Sigma}}_x(j,k) - \widetilde{\mathbf{\Sigma}}_x(j,k)\|_F \geq \xi) \leq C_1 p^2 \ell(x)^2 \exp\left(-\frac{CN\nu_x(\gamma)^2\xi^2}{\phi^2}\right) \quad \forall j,k \in [\ell(x)] \qquad (8.51)$$

Proof. The proof uses the technique of (Zhou et al., 2010).

Define, $N_{\boldsymbol{x};j,k}(\gamma) = \sum_{i=1}^{N} \sum_{j',k' \in [\ell(\boldsymbol{x}_i)]} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}_i)$ which is the empirical effective sample size under the smoothing kernel.

We first show that the $N_{\boldsymbol{x};j,k}(\gamma)$ is close to the expected effective sample size $N\nu_{\boldsymbol{x}}(\gamma)$.

Using Hoeffding's Inequality, we obtain that

$$P\left(|N_{\boldsymbol{x};j,k}(\gamma) - N\nu_{\boldsymbol{x}}(\gamma)| \geq N\nu_{\boldsymbol{x}}(\gamma)/2\right) \leq C\exp(-N\nu_{\boldsymbol{x}}(\gamma)^2/2) \tag{8.52}$$

Note that

$$\|\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k) - \widetilde{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k)\|_F \leq p^2 \max_{a,b} |\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)| \tag{8.53}$$

where $\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b)$ is the element on the a^{th} row and b^{th} column of $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}}(j,k)$. Thus, it suffices to bound $|\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)|$.

Define the boolean variable $E = \mathbb{I}[N_{\boldsymbol{x};j,k}(\gamma) > N\nu_{\boldsymbol{x}}(\gamma)/2]$. Then,

$$\begin{aligned}
& P(\|\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)\|_F \geq \xi) = \\
& \quad P(\|\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)\|_F \geq \xi | E=1)P(E=1) \\
& \quad + P(\|\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)\|_F \geq \xi | E=0)P(E=0) \\
\leq \; & P(\|\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)\|_F \geq \xi | E=1) + P(E=0)
\end{aligned} \tag{8.54}$$

The second term is bounded by Eq. (8.52). We prove the first term below.

For shorthand, refer to $P(\|\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)\|_F \geq \xi | E=1)$ as $P_E(\|\widehat{\varsigma}_{\boldsymbol{x}}(j,k;a,b) - \widetilde{\varsigma}_{\boldsymbol{x}}(j,k;a,b)\|_F \geq \xi | E=1)$. Define the following quantities:

For convenience define the following quantity, where $w_{j,a}^{(i)}$ is the a^{th} element of the vector $w_j^{(i)}$

$$\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b) = w_{j,a}^{(i)}(w_{k,b}^{(i)})^\top - \varsigma_{\boldsymbol{x}^{(i)}}(j,k;a,b) \tag{8.55}$$

For every $\kappa > 0$, by Markov's inequality,

$$\begin{aligned}
& P_E\left(\sum_{i\in[N]}\sum_{j',k'\in\ell(\boldsymbol{x}^{(i)})} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}^{(i)})\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b) > N_{\boldsymbol{x};j,k}(\gamma)\xi\right) \\
= \; & P_E\left(\exp\left(\kappa\sum_{i\in[N]}\sum_{j',k'\in\ell(\boldsymbol{x}^{(i)})} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}^{(i)})\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b)\right) > \exp\left(\kappa N_{\boldsymbol{x};j,k}(\gamma)\xi\right)\right) \\
\leq \; & \frac{\mathbb{E}\left[\exp\left(\kappa\sum_{i\in[N]}\sum_{j',k'\in\ell(\boldsymbol{x}^{(i)})} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}^{(i)})\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b)\right)\right]}{\exp\left(\kappa N_{\boldsymbol{x};j,k}(\gamma)\xi\right)} \qquad (8.56)
\end{aligned}$$

We now bound the numerator. Using the fact that the samples are independent, we have that

$$\begin{aligned}
& \mathbb{E}\left[\exp\left(\kappa\sum_{i\in[N]}\sum_{j',k'\in\ell(\boldsymbol{x}^{(i)})} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}^{(i)})\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b)\right)\right] && (8.57) \\
= \; & \prod_{i\in[N]}\prod_{j',k'\in\ell(\boldsymbol{x}^{(i)})} \mathbb{E}\left[\exp\left(\kappa K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}^{(i)})\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b)\right)\right] && (8.58) \\
= \; & \prod_{i\in[N]}\prod_{j',k'\in\ell(\boldsymbol{x}^{(i)})} \mathbb{I}[(j',k',\boldsymbol{x}^{(i)}) \in B_\gamma(j,k,\boldsymbol{x})]\mathbb{E}\left[\exp(\kappa\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b))\right] && (8.59) \\
& && (8.60)
\end{aligned}$$

From Utility Lemma 2, we can conclude that

$$\begin{aligned}
& \prod_{i\in[N]}\prod_{j',k'\in\ell(\boldsymbol{x}^{(i)})} \mathbb{I}[(j',k',\boldsymbol{x}^{(i)}) \in B_\gamma(j,k,\boldsymbol{x})]\mathbb{E}\left[\exp(\kappa\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b))\right] && (8.61) \\
\leq \; & \prod_{i\in[N]}\prod_{j',k'\in\ell(\boldsymbol{x}^{(i)})} \mathbb{I}[(j',k',\boldsymbol{x}^{(i)}) \in B_\gamma(j,k,\boldsymbol{x})]\exp(\kappa^2\phi^2/8) && (8.62) \\
\leq \; & \exp(N_{\boldsymbol{x};j,k}(\gamma)\kappa^2\phi^2/8) && (8.63)
\end{aligned}$$

$$\begin{aligned}
& P_E\left(\sum_{i\in[N]}\sum_{j',k'\in\ell(\boldsymbol{x}^{(i)})} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}^{(i)})\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b) > N_{\boldsymbol{x},j,k}(\gamma)\xi\right) && (8.64) \\
\leq \; & \exp(-\kappa N_{\boldsymbol{x};j,k}(\gamma)\xi)\exp(N_{\boldsymbol{x};j,k}(\gamma)\kappa^2\phi^2/8) && (8.65)
\end{aligned}$$

Setting $\kappa = \frac{4\xi}{\phi^2}$, gives us that

$$P_E\left(\sum_{i\in[N]}\sum_{j',k'\in\ell(x^{(i)})} K_\gamma(j,k,j',k'|\boldsymbol{x},\boldsymbol{x}^{(i)})\delta_{\boldsymbol{x}^{(i)}}(j,k;a,b) > N_{\boldsymbol{x},j,k}(\gamma)\xi\right) \tag{8.66}$$

$$\leq \exp(-2N_{\boldsymbol{x};j,k}(\gamma)\xi^2/\phi^2) \tag{8.67}$$

$$\leq \exp(-N\nu_{\boldsymbol{x}}(\gamma)\xi^2/\phi^2) \tag{8.68}$$

Combining the above with Eq. (8.52) and taking some union bounds over a, b, j, k proves the lemma. □

Helper Lemmas

We make use of the following helper lemma that is standard in the proof of Hoeffding's Inequality, e.g. (Casella and Berger, 1990).

Utility Lemma 2. *Suppose that* $\mathbb{E}(X) = 0$ *and that* $a \leq X \leq b$. *Then*

$$\mathbb{E}[e^{tX}] \leq e^{t^2(b-a)^2/8} \tag{8.69}$$

Chapter 9

Language Modeling via Power Low Rank Ensembles

Finally, we propose to use the linear algebra point of view of probabilistic modeling to tackle language modeling, a classic challenge in NLP with many applications such as machine translation and speech recognition.

Contribution of this chapter: We present power low rank ensembles (PLRE), a flexible framework for *n*-gram language modeling consisting of ensembles of low rank matrices and tensors. Our method can be understood as a generalization of *n*-gram modeling to non-integer n, and includes standard techniques such as absolute discounting and Kneser-Ney smoothing as special cases. PLRE training is efficient and our approach outperforms state-of-the-art modified Kneser Ney baselines in terms of perplexity on large corpora as well as on BLEU score in a downstream machine translation task.

Outline: We first present an introduction (§9.1) and then review existing *n*-gram smoothing methods (§9.2). We then present the intuition behind the key components of our technique: **rank** (§9.3.1) and **power** (§9.3.2). We then show how these can be interpolated into an ensemble (§9.4). An experimental evaluation on English and Russian corpora (§9.5) is then presented.

Prerequisites: This chapter assumes a general understanding of the connection between latent variable models and low rank factorization in Chapter 3.

9.1 Introduction

Language modeling is the task of estimating the probability of sequences of words in a language and is an important component in, among other applications, automatic speech recognition (Rabiner and Juang, 1993) and machine translation (Koehn, 2010). The predominant approach to language modeling is the *n*-gram model, wherein the probability of a word sequence $P(w_1, \ldots, w_\ell)$ is decomposed using the chain rule, and then a Markov assumption is made: $P(w_1, \ldots, w_\ell) \approx \prod_{i=1}^{\ell} P(w_i | w_{i-n+1}^{i-1})$. While this assumption substantially reduces the modeling complexity, parameter estimation remains a major challenge. Due to the power-law nature of language (Zipf, 1949), the maximum likelihood estimator massively overestimates the probability of rare events and assigns zero probability to legitimate word sequences that happen not to have been observed in the training data (Manning and Schütze, 1999).

Many smoothing techniques have been proposed to address the estimation challenge. These reassign probability mass (generally from over-estimated events) to unseen word sequences, whose probabilities are estimated by interpolating with or backing off to lower order *n*-gram models (Chen and Goodman, 1999).

Somewhat surprisingly, these widely used smoothing techniques differ substantially from techniques for coping with data sparsity in other domains, such as collaborative filtering (Koren et al., 2009; Su and Khoshgoftaar, 2009) or matrix completion (Candès and Recht, 2009; Cai et al., 2010). In these areas, *low rank* approaches based on matrix factorization play a central role (Lee and Seung, 2001; Salakhutdinov and Mnih, 2008; Mackey et al., 2011). For example, in recommender systems, a key challenge is dealing with the sparsity of ratings from a single user, since typical users will have rated only a few items. By projecting the low rank representation of a user's (sparse) preferences into the original space, an estimate of ratings for new items is obtained. These methods are attractive due to their computational efficiency and mathematical well-foundedness.

In this chapter, we introduce **power low rank ensembles** (PLRE), in which low rank tensors are used to produce smoothed estimates for *n*-gram probabilities. Ideally, we would like the low rank structures to discover semantic and syntactic relatedness among words and *n*-grams, which are used to produce smoothed estimates for word sequence probabilities.

In contrast to the few previous low rank language modeling approaches, PLRE is not orthogonal to *n*-gram models, but rather a general framework where existing *n*-gram smoothing methods such as Kneser-Ney smoothing are special cases. A key insight is that PLRE does not compute low rank approximations of the original joint count matrices (in the case of bigrams) or tensors i.e. multi-way arrays (in the case of 3-grams and above), but instead altered quantities of these counts based on an element-wise power operation, similar to how some smoothing methods modify their lower order distributions.

Moreover, PLRE has two key aspects that lead to easy scalability for large corpora and vocabularies. First, since it utilizes the original *n*-grams, the ranks required for the low rank matrices and tensors tend to be remain tractable (e.g. around 100 for a vocabulary size $V \approx 1 \times 10^6$) leading to fast training times. This differentiates our approach over other methods that leverage an underlying latent space such as neural networks (Bengio et al., 2003; Mnih and Hinton, 2007; Mikolov et al., 2010) or soft-class models (Saul and Pereira, 1997) where the underlying dimension is required to be quite large to obtain good performance. Moreover, at test time, the probability of a sequence can be queried in time $O(\kappa_{max})$ where κ_{max} is the maximum rank of the low rank matrices/tensors used. While this is larger than Kneser Ney's virtually constant query time, it is substantially faster than conditional exponential family models (Chen and Rosenfeld, 2000; Chen, 2009; Nelakanti et al., 2013) and neural networks which require $O(V)$ for exact computation of the normalization constant. See Section 9.7 for a more detailed discussion of related work.

9.2 Discount-based Smoothing

We first provide background on absolute discounting (Ney et al., 1994) and Kneser-Ney smoothing (Kneser and Ney, 1995), two common n-gram smoothing methods. Both methods can be formulated as back-off or interpolated models; we describe the latter here since that is the basis of our low rank approach.

9.2.1 Notation

Let $c(w)$ be the count of word w, and similarly $c(w, w_{i-1})$ for the joint count of words w and w_{i-1}. For shorthand we will define w_i^j to denote the word sequence $\{w_i, w_{i+1}, ..., w_{j-1}, w_j\}$. Let $\hat{P}(w_i)$ refer to the maximum likelihood estimate (MLE) of the probability of word w_i, and similarly $\hat{P}(w_i|w_{i-1})$ for the probability conditioned on a history, or more generally, $\hat{P}(w_i|w_{i-n+1}^{i-1})$.

Let $N_-(w_i) := |\{w : c(w_i, w) > 0\}|$ be the number of distinct words that appear before w_i. More generally, let $N_-(w_{i-n+1}^i) = |\{w : c(w_{i-n+1}^i, w) > 0\}|$. Similarly, let $N_+(w_{i-n+1}^{i-1}) = |\{w : c(w, w_{i-n+1}^{i-1}) > 0\}|$. V denotes the vocabulary size.

9.2.2 Absolute Discounting

Absolute discounting works on the idea of interpolating higher order n-gram models with lower-order n-gram models. However, first some probability mass must be "subtracted" from the higher order n-grams so that the leftover probability can be allocated to the lower order n-grams. More specifically, define the following discounted conditional probability:

$$\hat{P}_D(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_i, w_{i-n+1}^{i-1}) - D, 0\}}{c(w_{i-n+1}^{i-1})} \tag{9.1}$$

Then absolute discounting $P_{\text{abs}}(\cdot)$ uses the following (recursive) equation:

$$P_{\text{abs}}(w_i|w_{i-n+1}^{i-1}) = \hat{P}_D(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1})P_{\text{abs}}(w_i|w_{i-n+2}^{i-1}) \tag{9.2}$$

where $\gamma(w_{i-n+1}^{i-1})$ is the leftover weight (due to the discounting) that is chosen so that the conditional distribution sums to one: $\gamma(w_{i-n+1}^{i-1}) = \frac{D}{c(w_{i-n+1}^{i-1})}N_+(w_{i-n+1}^{i-1})$. For the base case, we set $P_{\text{abs}}(w_i) = \hat{P}(w_i)$.

Discontinuity: Note that if $c(w_{i-n+1}^{i-1}) = 0$, then $\gamma(w_{i-n+1}^{i-1}) = \frac{0}{0}$, in which case $\gamma(w_{i-n+1}^{i-1})$ is set to 1. We will see that this discontinuity appears in PLRE as well.

9.2.3 Kneser Ney Smoothing

Ideally, the smoothed probability should preserve the observed unigram distribution:

$$\hat{P}(w_i) = \sum_{w_{i-n+1}^{i-1}} P_{\text{sm}}(w_i|w_{i-n+1}^{i-1})\hat{P}(w_{i-n+1}^{i-1}) \tag{9.3}$$

where $P_{\text{sm}}(w_i|w_{i-n+1}^{i-1})$ is the smoothed conditional probability that a model outputs. Unfortunately, absolute discounting does not satisfy this property, since it exclusively uses the unaltered MLE unigram model as its lower order model. In practice, the lower order distribution is only utilized when we are unsure about the higher order distribution (i.e., when $\gamma(\cdot)$ is large). Therefore, the unigram model should be altered to condition on this fact.

This is the inspiration behind Kneser-Ney (KN) smoothing, an elegant algorithm with robust performance in n-gram language modeling. KN smoothing defines alternate probabilities $P^{\text{alt}}(\cdot)$:

$$P_D^{\text{alt}}(w_i|w_{i-n'+1}^{i-1}) = \begin{cases} \hat{P}_D(w_i|w_{i-n'+1}^{i-1}), & \text{if } n' = n \\ \frac{\max\{N_-(w_{i-n'+1}^{i})-D,0\}}{\sum_{w_i} N_-(w_{i-n'+1}^{i})}, & \text{if } n' < n \end{cases} \tag{9.4}$$

The base case for unigrams reduces to $P^{\text{alt}}(w_i) = \frac{N_-(w_i)}{\sum_{w_i} N_-(w_i)}$. Intuitively $P^{\text{alt}}(w_i)$ is proportional to the number of unique words that precede w_i. Thus, words that appear in many different contexts will be given higher weight than words that consistently appear after only a few contexts. These alternate distributions are then used with absolute discounting:

$$P_{\text{kn}}(w_i|w_{i-n+1}^{i-1}) = P_D^{\text{alt}}(w_i|w_{i-n+1}^{i-1}) + \gamma(w_{i-n+1}^{i-1})P_{\text{kn}}(w_i|w_{i-n+2}^{i-1}) \tag{9.5}$$

where we set $P_{\text{kn}}(w_i) = P^{\text{alt}}(w_i)$. By definition, KN smoothing satisfies the marginal constraint in Eq. 9.3 (Kneser and Ney, 1995).

9.3 Power Low Rank Ensembles

In n-gram smoothing methods, if a bigram count $c(w_i, w_{i-1})$ is zero, the unigram probabilities are used, which is equivalent to assuming that w_i and w_{i-1} are independent (and similarly for general n). However, in this situation, instead of backing off to a 1-gram, we may like to back off to a "1.5-gram" or more generally an order between 1 and 2 that captures a coarser level of dependence between w_i and w_{i-1} and does not assume full independence.

Inspired by this intuition, our strategy is to construct an ensemble of matrices and tensors that not only consists of MLE-based count information, but also contains quantities that represent levels of dependence in-between the various orders in the model. We call these combinations power low rank ensembles (PLRE), and they can be thought of as n-gram models with non-integer n. Our approach can be recursively formulated as:

$$P_{\text{plre}}(w_i|w_{i-n+1}^{i-1}) = P_{\boldsymbol{D}_0}^{\text{alt}}(w_i|w_{i-n+1}^{i-1})$$
$$+ \gamma_0(w_{i-n+1}^{i-1})\left(\boldsymbol{Z}_{\boldsymbol{D}_1}(w_i|w_{i-n+1}^{i-1}) + \ldots.. + \gamma_{\eta-1}(w_{i-n+1}^{i-1})\left(\boldsymbol{Z}_{\boldsymbol{D}_\eta}(w_i|w_{i-n+1}^{i-1}) + \gamma_\eta(w_{i-n+1}^{i-1})\left(P_{\text{plre}}(w_i|w_{i-n+2}^{i-1})\right)\right)\ldots\right) \tag{9.6}$$

where $\boldsymbol{Z}_1, ..., \boldsymbol{Z}_\eta$ are conditional probability matrices that represent the intermediate n-gram orders[1] and $\boldsymbol{D}$ is a discount function (specified in §9.4).

This formulation begs answers to a few critical questions. How to construct matrices that represent conditional probabilities for intermediate n? How to transform them in a way that generalizes the altered lower order distributions in KN smoothing? How to combine these matrices such that the marginal constraint in Eq. 9.3 still holds? The following propose solutions to these three queries:

1. **Rank** (Section 9.3.1): *Rank* gives us a concrete measurement of the dependence between w_i and w_{i-1}. By constructing low rank approximations of the bigram count matrix and higher-order count tensors, we obtain matrices that represent coarser dependencies, with a rank one approximation implying that the variables are independent.
2. **Power** (Section 9.3.2): In KN smoothing, the lower order distributions are not the original counts but rather altered estimates. We propose a continuous generalization of this alteration by taking the element-wise *power* of the counts.
3. **Creating the Ensemble** (Section 9.4): Lastly, PLRE also defines a way to interpolate the specifically constructed intermediate n-gram matrices. Unfortunately a constant discount, as presented in Section 9.2, will not in general preserve the lower order marginal constraint (Eq. 9.3). We propose a generalized discounting scheme to ensure the constraint holds.

9.3.1 Rank

We first show how rank can be utilized to construct quantities between an n-gram and an $n-1$-gram. In general, we think of an n-gram as an n^{th} order tensor i.e. a multi-way array with n indices $\{i_1, ..., i_n\}$. (A vector is a tensor of order 1, a matrix is a tensor of order 2 etc.) Computing a special rank one approximation of slices of this tensor produces the $n-1$-gram. Thus, taking rank κ approximations in this fashion allows us to represent dependencies between an n-gram and $n-1$-gram.

Consider the bigram count matrix $\boldsymbol{B}$ with N counts which has rank V. Note that $\hat{P}(w_i|w_{i-1}) = \frac{\boldsymbol{B}(w_i,w_{i-1})}{\sum_w \boldsymbol{B}(w,w_{i-1})}$. Additionally, $\boldsymbol{B}$ can be considered a random variable that is the result of sampling N tuples of (w_i, w_{i-1}) and agglomerating them into a count matrix. Assuming w_i and w_{i-1} are independent, the expected value (with respect to the empirical distribution) $\mathbb{E}[\boldsymbol{B}] = NP(w_i)P(w_{i-1})$, which can be rewritten as being proportional to the outer product of the unigram probability vector with itself, and is thus rank one.

[1] with a slight abuse of notation, let $\boldsymbol{Z}_{\boldsymbol{D}_j}$ be shorthand for $\boldsymbol{Z}_{j,\boldsymbol{D}_j}$

This observation extends to higher order n-grams as well. Let $\boldsymbol{C}^n$ be the nth order tensor where $\boldsymbol{C}^n(w_i, ..., w_{i-n+1}) = c(w_i, ..., w_{i-n+1})$. Furthermore denote $\boldsymbol{C}^n(:, \tilde{w}_{i-n+2}^{i-1}, :)$ to be the $V \times V$ matrix slice of $\boldsymbol{C}^n$ where $w_{i-n+2}, ..., w_{i-1}$ are held fixed to a particular sequence $\tilde{w}_{i-n+2}, ..., \tilde{w}_{i-1}$. Then if w_i is conditionally independent of w_{i-n+1} given w_{i-n+2}^{i-1}, then $\mathbb{E}[\boldsymbol{C}^n(:, \tilde{w}_{i-n+2}^{i-1}, :)]$ is rank one $\forall \tilde{w}_{i-n+2}^{i-1}$.

However, it is rare that these matrices are actually rank one, either due to sampling variance or the fact that w_i and w_{i-1} are not independent. What we would really like to say is that the *best* rank one approximation $\boldsymbol{B}^{(1)}$ (under some norm) of $\boldsymbol{B}$ is $\propto \hat{P}(w_i)\hat{P}(w_{i-1})$. While this statement is not true under the ℓ_2 norm, it is true under generalized KL divergence (Lee and Seung, 2001): $gKL(\boldsymbol{A}\|\boldsymbol{B}) = \sum_{ij}\left(\boldsymbol{A}_{ij}\log(\frac{\boldsymbol{A}_{ij}}{\boldsymbol{B}_{ij}}) - \boldsymbol{A}_{ij} + \boldsymbol{B}_{ij})\right)$.

In particular, generalized KL divergence preserves row and column sums: *if $\boldsymbol{M}^{(\kappa)}$ is the best rank κ approximation of $\boldsymbol{M}$ under gKL then the row sums and column sums of $\boldsymbol{M}^{(\kappa)}$ and $\boldsymbol{M}$ are equal* (Ho and Van Dooren, 2008). Leveraging this property, it is straightforward to prove the following lemma:

Lemma 26. *Let $\boldsymbol{B}^{(\kappa)}$ be the best rank κ approximation of $\boldsymbol{B}$ under gKL. Then $\boldsymbol{B}^{(1)} \propto \hat{P}(w_i)\hat{P}(w_{i-1})$ and $\forall w_{i-1}$ s.t. $c(w_{i-1}) \neq 0$:*

$$\hat{P}(w_i) = \frac{\boldsymbol{B}^{(1)}(w_i, w_{i-1})}{\sum_w \boldsymbol{B}^{(1)}(w, w_{i-1})} \tag{9.7}$$

For more general n, let $\boldsymbol{C}^{n,(\kappa)}_{i-1,...,i-n+2}$ be the best rank κ approximation of $\boldsymbol{C}^n(:, \tilde{w}_{i-n+2}^{i-1}, :)$ under gKL. Then similarly, $\forall w_{i-n+1}^{i-1}$ s.t. $c(w_{i-n+1}^{i-1}) > 0$:

$$\hat{P}(w_i|w_{i-1}, ..., w_{i-n+2}) = \frac{\boldsymbol{C}^{n,(1)}_{i-1,...,i-n+2}(w_i, w_{i-n+1}^{i-1})}{\sum_w \boldsymbol{C}^{n,(1)}_{i-1,...,i-n+2}(w, w_{i-n+1}^{i-1})} \tag{9.8}$$

Thus, by selecting $1 < \kappa < V$, we obtain count matrices and tensors between n and $n-1$-grams. The condition that $c(w_{i-n+1}^{i-1}) > 0$ corresponds to the discontinuity discussed in §9.2.2.

9.3.2 Power

Since KN smoothing alters the lower order distributions instead of simply using the MLE, varying the rank is not sufficient in order to generalize this suite of techniques. Thus, PLRE computes low rank approximations of altered count matrices. Consider taking the elementwise power ρ of the bigram count matrix, which is denoted by $\boldsymbol{B}^{\cdot\rho}$. For example, the observed bigram count matrix and associated row sum:

$$\boldsymbol{B}^{\cdot 1} = \begin{pmatrix} 1.0 & 2.0 & 1.0 \\ 0 & 5.0 & 0 \\ 2.0 & 0 & 0 \end{pmatrix} \overset{\text{row sum}}{\longrightarrow} \begin{pmatrix} 4.0 \\ 5.0 \\ 2.0 \end{pmatrix}$$

As expected the row sum is equal to the unigram counts (which we denote as $\boldsymbol{u}$). Now consider

$\boldsymbol{B}^{\cdot 0.5}$:

$$\boldsymbol{B}^{\cdot 0.5} = \begin{pmatrix} 1.0 & 1.4 & 1.0 \\ 0 & 2.2 & 0 \\ 1.4 & 0 & 0 \end{pmatrix} \overset{\text{row sum}}{\longrightarrow} \begin{pmatrix} 3.4 \\ 2.2 \\ 1.4 \end{pmatrix}$$

Note how the row sum vector has been altered. In particular since w_1 (corresponding to the first row) has a more diverse history than w_2, it has a higher row sum (compared to in $\boldsymbol{u}$ where w_2 has the higher row sum). Lastly, consider the case when $p = 0$:

$$\boldsymbol{B}^{\cdot 0} = \begin{pmatrix} 1.0 & 1.0 & 1.0 \\ 0 & 1.0 & 0 \\ 1.0 & 0 & 0 \end{pmatrix} \overset{\text{row sum}}{\longrightarrow} \begin{pmatrix} 3.0 \\ 1.0 \\ 1.0 \end{pmatrix}$$

The row sum is now the number of unique words that precede w_i (since $\boldsymbol{B}^0$ is binary) and is thus equal to the (unnormalized) Kneser Ney unigram. This idea also generalizes to higher order n-grams and leads us to the following lemma:

Lemma 27. *Let $\boldsymbol{B}^{(\rho,\kappa)}$ be the best rank κ approximation of $\boldsymbol{B}^{\cdot\rho}$ under gKL. Then $\forall w_{i-1}$ s.t. $c(w_{i-1}) \neq 0$:*

$$P^{alt}(w_i) = \frac{\boldsymbol{B}^{(0,1)}(w_i, w_{i-1})}{\sum_w \boldsymbol{B}^{(0,1)}(w, w_{i-1})}$$

For more general n, let $\boldsymbol{C}^{n,(\rho,\kappa)}_{i-1,\ldots,i-n+2}$ be the best rank κ approximation of $\boldsymbol{C}^{n,(\rho)}(:, \tilde{w}^{i-1}_{i-n+2}, :)$ under gKL. Similarly, $\forall w^{i-1}_{i-n+1}$ s.t. $c(w^{i-1}_{i-n+1}) > 0$:

$$P^{alt}(w_i | w_{i-1}, \ldots, w_{i-n+2}) = \frac{\boldsymbol{C}^{n,(0,1)}_{i-1,\ldots,i-n+2}(w_i, w^{i-1}_{i-n+1})}{\sum_w \boldsymbol{C}^{n,(0,1)}_{i-1,\ldots,i-n+2}(w, w^{i-1}_{i-n+1})} \tag{9.9}$$

9.4 Creating the Ensemble

Recall our overall formulation in Eq. 9.6; a naive solution would be to set $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_\eta$ to low rank approximations of the count matrices/tensors under varying powers, and then interpolate through constant absolute discounting. Unfortunately, the marginal constraint in Eq. 9.3 will generally not hold if this strategy is used. Therefore, we propose a generalized discounting scheme where each non-zero n-gram count is associated with a different discount $\boldsymbol{D}_j(w_i, w^{i-1}_{i-n'+1})$. The low rank approximations are then computed on the discounted matrices, leaving the marginal constraint intact.

For clarity of exposition, we focus on the special case where $n = 2$ with only one low rank matrix before stating our general algorithm:

$$\begin{aligned} P_{\text{plre}}(w_i | w_{i-1}) &= \hat{P}_{\boldsymbol{D}_0}(w_i | w_{i-1}) \\ &+ \gamma_0(w_{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_1}(w_i | w_{i-1}) + \gamma_1(w_{i-1}) P^{\text{alt}}(w_i)\Big) \end{aligned} \tag{9.10}$$

Our goal is to compute $\boldsymbol{D}_0, \boldsymbol{D}_1$ and $\boldsymbol{Z}_1$ so that the following lower order marginal constraint holds:

$$\hat{P}(w_i) = \sum_{w_{i-1}} P_{\text{plre}}(w_i|w_{i-1})\hat{P}(w_{i-1}) \tag{9.11}$$

Our solution can be thought of as a two-step procedure where we compute the discounts $\boldsymbol{D}_0, \boldsymbol{D}_1$ (and the $\gamma(w_{i-1})$ weights as a by-product), followed by the low rank quantity $\boldsymbol{Z}_1$. First, we construct the following intermediate ensemble of powered, but full rank terms. Let $\boldsymbol{Y}^{\rho_j}$ be the matrix such that $\boldsymbol{Y}^{\rho_j}(w_i, w_{i-1}) := c(w_i, w_{i-1})^{\rho_j}$. Then define

$$P_{\text{pwr}}(w_i|w_{i-1}) := \boldsymbol{Y}^{(\rho_0=1)}_{\boldsymbol{D}_0}(w_i|w_{i-1}) + \gamma_0(w_{i-1})\left(\boldsymbol{Y}^{(\rho_1)}_{\boldsymbol{D}_1}(w_i|w_{i-1}) + \gamma_1(w_{i-1})\boldsymbol{Y}^{(\rho_2=0)}(w_i|w_{i-1})\right) \tag{9.12}$$

where with a little abuse of notation:

$$\boldsymbol{Y}^{\rho_j}_{\boldsymbol{D}_j}(w_i|w_{i-1}) = \frac{c(w_i, w_{i-1})^{\rho_j} - \boldsymbol{D}_j(w_i, w_{i-1})}{\sum_{w_i} c(w_i, w_{i-1})^{\rho_j}}$$

Note that $P^{\text{alt}}(w_i)$ has been replaced with $\boldsymbol{Y}^{(\rho_2=0)}(w_i|w_{i-1})$, based on Lemma 27, and will equal $P^{\text{alt}}(w_i)$ once the low rank approximation is taken as discussed in § 9.4.2).

Since we have only combined terms of different power (but all full rank), it is natural choose the discounts so that the result remains unchanged i.e., $P_{\text{pwr}}(w_i|w_{i-1}) = \hat{P}(w_i|w_{i-1})$, since the low rank approximation (not the power) will implement smoothing. Enforcing this constraint gives rise to a set of linear equations that can be solved (in closed form) to obtain the discounts as we now show below.

9.4.1 Step 1: Computing the Discounts

To ensure the constraint that $P_{\text{pwr}}(w_i|w_{i-1}) = \hat{P}(w_i|w_{i-1})$, it is sufficient to enforce the following two local constraints:

$$\boldsymbol{Y}^{(\rho_j)}(w_i|w_{i-1}) = \boldsymbol{Y}^{(\rho_j)}_{\boldsymbol{D}_j}(w_i|w_{i-1}) + \gamma_j(w_{i-1})\boldsymbol{Y}^{(\rho_{j+1})}(w_i|w_{i-1}) \text{ for } j = 0, 1 \tag{9.13}$$

This allows each $\boldsymbol{D}_j$ to be solved for independently of the other $\{\boldsymbol{D}_{j'}\}_{j' \neq j}$. Let $c_{i,i-1} = c(w_i, w_{i-1})$, $c^j_{i,i-1} = c(w_i, w_{i-1})^{\rho_j}$, and $d^j_{i,i-1} = \boldsymbol{D}_j(w_i, w_{i-1})$. Expanding Eq. 9.13 yields that $\forall w_i, w_{i-1}$:

$$\frac{c^j_{i,i-1}}{\sum_i c^j_{i,i-1}} = \frac{c^j_{i,i-1} - d^j_{i,i-1}}{\sum_i c^j_{i,i-1}} + \left(\frac{\sum_i d^j_{i,i-1}}{\sum_i c^j_{i,i-1}}\right)\frac{c^{j+1}_{i,i-1}}{\sum_i c^{j+1}_{i,i-1}} \tag{9.14}$$

which can be rewritten as:

$$-d^j_{i,i-1} + \left(\sum_i d^j_{i,i-1}\right)\frac{c^{j+1}_{i,i-1}}{\sum_i c^{j+1}_{i,i-1}} = 0 \tag{9.15}$$

Algorithm 19 Compute $\boldsymbol{D}$

```
In: Count tensor $\boldsymbol{C}^n$, powers $\rho_j, \rho_{j+1}$ such that $\rho_j \geq \rho_{j+1}$, and parameter $d_*$.
Out: Discount $\boldsymbol{D}_j$ for powered counts $\boldsymbol{C}^{n,(\rho_j)}$ and associated leftover weight $\gamma_j$
1: Set $\boldsymbol{D}_j(w_i, w_{i-n+1}^{i-1}) = d_* c(w_i, w_{i-n+1}^{i-1})^{\rho_{j+1}}$.
2: $$\gamma_j(w_i, w_{i-n+1}^{i-1}) = \frac{d_* \sum_{w_i} c(w_i, w_{i-n+1}^{i-1})^{\rho_{j+1}}}{\sum_{w_i} c(w_i, w_{i-n+1}^{i-1})^{\rho_j}}$$
```

Note that Eq. 9.15 decouples across w_{i-1} since the only $d^j_{i,i-1}$ terms that are dependent are the ones that share the preceding context w_{i-1}.

It is straightforward to see that setting $d^j_{i,i-1}$ proportional to $c^{j+1}_{i,i-1}$ satisfies Eq. 9.15. Furthermore it can be shown that all solutions are of this form (i.e., the linear system has a null space of exactly one). Moreover, we are interested in a particular subset of solutions where a single parameter d_* (independent of w_{i-1}) controls the scaling as indicated by the following lemma:

Lemma 28. *Assume that $\rho_j \geq \rho_{j+1}$. Choose any $0 \leq d_* \leq 1$. Set $d^j_{i,i-1} = d_* c^{j+1}_{i,i-1}\ \forall i, j$. The resulting discounts satisfy Eq. 9.15 as well as the inequality constraints $0 \leq d^j_{i,i-1} \leq c^j_{i,i-1}$. Furthermore, the leftover weight γ_j takes the form:*

$$\gamma_j(w_{i-1}) = \frac{\sum_i d^j_{i,i-1}}{\sum_i c^j_{i,i-1}} = \frac{d_* \sum_i c^{j+1}_{i,i-1}}{\sum_i c^j_{i,i-1}} \tag{9.16}$$

Proof. Clearly this choice of $d^j_{i,i-1}$ satisfies Eq. 9.15. The largest possible value of $d^j_{i,i-1}$ is $c^{j+1}_{i,i-1}$. $\rho_j \geq \rho_{j+1}$, implies $c^j_{i,i-1} \geq c^{j+1}_{i,i-1}$. Thus the inequality constraints are met. It is then easy to verify that γ takes the above form. □

The above lemma generalizes to longer contexts (i.e. $n > 2$) as shown in Algorithm 19. Note that if $\rho_j = \rho_{j+1}$ then Algorithm 19 is equivalent to scaling the counts e.g. deleted-interpolation/Jelinek Mercer smoothing (Jelinek and Mercer, 1980). On the other hand, when $\rho_{j+1} = 0$, Algorithm 19 is equal to the absolute discounting that is used in Kneser-Ney. Thus, depending on ρ_{j+1}, our method generalizes different types of interpolation schemes to construct an ensemble so that the marginal constraint is satisfied.

9.4.2 Step 2: Computing Low Rank Quantities

The next step is to compute low rank approximations of $\boldsymbol{Y}^{(\rho_j)}_{\boldsymbol{D}_j}$ to obtain $\boldsymbol{Z}_{\boldsymbol{D}_j}$ such that the intermediate marginal constraint in Eq. 9.11 is preserved. This constraint trivially holds for the intermediate ensemble $P_{\text{pwr}}(w_i|w_{i-1})$ due to how the discounts were derived in § 9.4.1. For our running bigram example, define $\boldsymbol{Z}^{(\rho_j,\kappa_j)}_{\boldsymbol{D}_j}$ to be the best rank κ_j approximation to $\boldsymbol{Y}^{(\rho_j,\kappa_j)}_{\boldsymbol{D}_j}$ according

Algorithm 20 Compute $\boldsymbol{Z}$

In: Count tensor $\boldsymbol{C}^n$, power ρ, discounts $\boldsymbol{D}$, rank κ

Out: Discounted low rank conditional probability table $\boldsymbol{Z}_{\boldsymbol{D}}^{(\rho,\kappa)}(w_i|w_{i-n+1}^{i-1})$ (represented implicitly)

```
1: Compute powered counts $\boldsymbol{C}^{n,(\cdot\rho)}$.
2: Compute denominators $\sum_{w_i} c(w_i, w_{i-n+1}^{i-1})^\rho \; \forall w_{i-n+1}^{i-1}$ s.t. $c(w_{i-n+1}^{i-1}) > 0$.
3: Compute discounted powered counts $\boldsymbol{C}_{\boldsymbol{D}}^{n,(\cdot\rho)} = \boldsymbol{C}^{n,(\cdot\rho)} - \boldsymbol{D}$.
4: For each slice $\boldsymbol{M}_{\tilde{w}_{i-n+2}^{i-1}} := \boldsymbol{C}_{\boldsymbol{D}}^{n,(\cdot\rho)}(:, \tilde{w}_{i-n+2}^{i-1}, :)$ compute
       $\boldsymbol{M}^{(\kappa)} := \min_{\boldsymbol{A}\geq 0: rank(\boldsymbol{A})=\kappa} ||\boldsymbol{M}_{\tilde{w}_{i-n+2}^{i-1}} - \boldsymbol{A}||_{KL}$
       (stored implicitly as $\boldsymbol{M}^{(\kappa)} = \boldsymbol{L}\boldsymbol{R}$)
     Set $\boldsymbol{Z}_{\boldsymbol{D}}^{(\rho,\kappa)}(:, \tilde{w}_{i-n+2}^{i-1}, :) = \boldsymbol{M}^{(\kappa)}$
5: Note that
       $\boldsymbol{Z}_{\boldsymbol{D}}^{(\rho,\kappa)}(w_i|w_{i-n+1}^{i-1}) = \frac{\boldsymbol{Z}_{\boldsymbol{D}}^{(\rho,\kappa)}(w_i, w_{i-n+1}^{i-1})}{\sum_{w_i} c(w_i, w_{i-n+1}^{i-1})^\rho}$
```

to gKL and let

$$\boldsymbol{Z}_{\boldsymbol{D}_j}^{\rho_j,\kappa_j}(w_i|w_{i-1}) = \frac{\boldsymbol{Z}_{\boldsymbol{D}_j}^{\rho_j,\kappa_j}(w_i, w_{i-1})}{\sum_{w_i} c(w_i, w_{i-1})^{\rho_j}} \tag{9.17}$$

Note that $\boldsymbol{Z}_{\boldsymbol{D}_j}^{\rho_j,\kappa_j}(w_i|w_{i-1})$ is a valid (discounted) conditional probability since gKL preserves row/column sums so the denominator remains unchanged under the low rank approximation. Then using the fact that $\boldsymbol{Z}^{(0,1)}(w_i|w_{i-1}) = P^{\text{alt}}(w_i)$ (Lemma 27) we can embellish Eq. 9.10 as

$$P_{\text{plre}}(w_i|w_{i-1}) = P_{\boldsymbol{D}_0}(w_i|w_{i-1}) + \gamma_0(w_{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho_1,\kappa_1)}(w_i|w_{i-1}) + \gamma_1(w_{i-1})P_{\text{alt}}(w_i)\Big) \tag{9.18}$$

Leveraging the form of the discounts and row/column sum preserving property of gKL, we then have the following lemma (the proof is in the Appendix):

Lemma 29. *Let $P_{plre}(w_i|w_{i-1})$ indicate the PLRE smoothed conditional probability as computed by Eq. 9.10 and Algorithms 19 and 20. Then, the marginal constraint in Eq. 9.11 holds i.e.:*

$$\hat{P}(w) = \sum_{w_{i-1}} P_{plre}(w_i|w_{i-1})\hat{P}(w_{i-1}) \tag{9.19}$$

To summarize, we show the training and testing algorithms for the bigram power low rank ensemble in Algorithms 21 and 22 respectively. In training, our method first computes discounts and leftover weights using Algorithm 19 and then Algorithm 20 to compute the low rank matrices. Testing simply corresponds to looking up the value for the qury (w_i, w_{i-1}) in each element of the ensemble and combining the terms scaled by the leftover weights.

Figure 9.1 gives a high level overview of training a power low rank ensemble and Algorithms 21 and 22 describe training and testing for a bigram PLRE.

9.4.3 More general algorithm

In general, the principles outlined in the previous sections hold for higher order n-grams. Assume that the discounts are computed according to Algorithm 19 with parameter d_* and $\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}$ is computed according to Algorithm 20. Note that, as shown in Algorithm 20, for higher order n-grams, the $\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}$ are created by taking low rank approximations of slices of the (powered) count tensors (see Lemma 27 for intuition). Eq. 9.6 can now be embellished:

$$P_{\text{plre}}(w_i|w_{i-n+1}^{i-1}) = P_{\boldsymbol{D}_0}^{\text{alt}}(w_i|w_{i-n+1}^{i-1})$$
$$+ \gamma_0(w_{i-n+1}^{i-1})\left(\boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho_1,\kappa_1)}(w_i|w_{i-n+1}^{i-1}) + ... + \gamma_{\eta-1}(w_{i-n+1}^{i-1})\left(\boldsymbol{Z}_{\boldsymbol{D}_\eta}^{(\rho_\eta,\kappa_\eta)}(w_i|w_{i-n+1}^{i-1}) + \gamma_\eta(w_{i-n+1}^{i-1})\left(P_{\text{plre}}(w_i|w_{i-n+2}^{i-1})\right)\right)...\right) \tag{9.20}$$

Lemma 29 also applies in this case and is proved in the Appendix.

9.4.4 Links with KN Smoothing

In this section, we explicitly show the relationship between PLRE and KN smoothing. Rewriting Eq. 9.31 in the following form:

$$P_{\text{plre}}(w_i|w_{i-n+1}^{i-1}) = P_{\text{plre}}^{\text{terms}}(w_i|w_{i-n+1}^{i-1}) + \gamma_{0:\eta}(w_{i-n+1}^{i-1})P_{\text{plre}}(w_i|w_{i-n+2}^{i-1}) \tag{9.21}$$

where $P_{\text{plre}}^{\text{terms}}(w_i|w_{i-n+1}^{i-1})$ contains the terms in Eq. 9.31 except the last, and $\gamma_{0:\eta}(w_{i-n+1}^{i-1}) = \prod_{h=0}^{\eta}\gamma_h(w_{i-n+1}^{i-1})$, we can leverage the form of the discount, and using the fact that $\rho_{\eta+1} = 0$[2]:

$$\gamma_{0:\eta}(w_{i-n-1}^{i-1}) = \frac{d_*{}^{\eta+1}N_+(w_{i-n+1}^{i-1})}{c(w_{i-n+1}^{i-1})} \tag{9.22}$$

With this form of $\gamma(\cdot)$, Eq. 9.21 is remarkably similar to KN smoothing (Eq. 9.5) if KN's discount parameter D is chosen to equal $(d_*)^{\eta+1}$.

The difference is that $P^{\text{alt}}(\cdot)$ has been replaced with the alternate estimate $P_{\text{plre}}^{\text{terms}}(w_i|w_{i-n+1}^{i-1})$, which have been enriched via the low rank structure. Since these alternate estimates were constructed via our ensemble strategy they contain both very fine-grained dependencies (the original n-grams) as well as coarser dependencies (the lower rank n-grams) and is thus fundamentally different than simply taking a single matrix/tensor decomposition of the trigram/bigram matrices.

Moreover, it provides a natural way of setting d_* based on the Good-Turing (GT) estimates employed by KN smoothing. In particular, we can set d_* to be the $(\eta + 1)^{\text{th}}$ root of the KN discount D that can be estimated via the GT estimates.

[2]for derivation see proof of Lemma 29 in the Appendixl

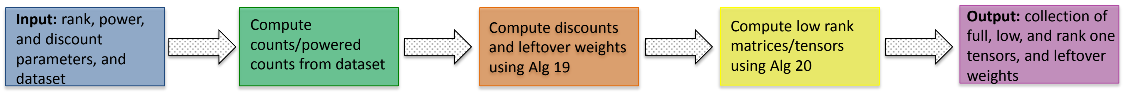

Input: rank, power, and discount parameters, and dataset
Compute counts/powered counts from dataset
Compute discounts and leftover weights using Alg 19
Compute low rank matrices/tensors using Alg 20
Output: collection of full, low, and rank one tensors, and leftover weights

Figure 9.1: Flowchart that gives an overview of training a power low rank ensemble

```
Algorithm 21 Train (Bigram) Power Low Rank Ensemble (PLRE)
In: Dataset $\mathcal{D}$, discount parameter $d_*$, rank $\kappa^*$, power $\rho^*$
Out: Full, low, and rank one bigrams $\{P_{\boldsymbol{D}_0}(w_i|w_{i-1}), \boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho^*,\kappa^*)}(w_i|w_{i-1}), P_{\text{alt}}(w_i)\}$, leftover weights $\{\gamma_0, \gamma_1\}$
1: Compute bigram counts $\boldsymbol{B}$ from dataset $\mathcal{D}$
2: Compute discount $\boldsymbol{D}_0$ and leftover weight $\gamma_0$ using Algorithm 19 with arguments $\boldsymbol{C}^n := \boldsymbol{B}$, $\rho_0 := 1$, $\rho_1 := \rho^*$, $d_*$. Set
       $$P_{\boldsymbol{D}_0}(w_i|w_{i-1}) = \frac{\boldsymbol{B}(w_i, w_{i-1}) - \boldsymbol{D}_0(w_i, w_{i-1})}{\sum_w \boldsymbol{B}(w, w_{i-1})}$$
3: Compute discount $\boldsymbol{D}_1$, and leftover weight $\gamma_1$ using Algorithm 19 with arguments $\boldsymbol{C}^n := \boldsymbol{B}$, $\rho_1 := \rho^*, \rho_2 := 0, d_*$
4: Compute $\boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho^*,\kappa^*)}(w_i|w_{i-1})$ using Algorithm 20 with arguments $\boldsymbol{C}^n := \boldsymbol{B}$, $\rho^*$, $\boldsymbol{D}_1, \kappa^*$
5: Compute $P_{\text{alt}}(w_i)$ using Eq. 9.4 from dataset $\mathcal{D}$
```

```
Algorithm 22 Test (Bigram) Power Low Rank Ensemble (PLRE)
In: current/previous words $(w_i, w_{i-1})$, full, low, and rank one bigrams $\{P_{\boldsymbol{D}_0}(w_i|w_{i-1}), \boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho^*,\kappa^*)}(w_i|w_{i-1}), P_{\text{alt}}(w_i)\}$, leftover weights $\{\gamma_0, \gamma_1\}$
Out: conditional probability estimate $P_{plre}(w_i|w_{i-1})$
1: Compute
       $$P_{\text{plre}}(w_i|w_{i-1}) = P_{\boldsymbol{D}_0}(w_i|w_{i-1}) + \gamma_0(w_{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho^*,\kappa^*)}(w_i|w_{i-1}) + \gamma_1(w_{i-1})P_{\text{alt}}(w_i)\Big) \qquad (9.23)$$
2: Return $P_{\text{plre}}(w_i|w_{i-1})$
```

9.4.5 Computational Considerations

PLRE scales well even as the order n increases. To compute a low rank bigram, one low rank approximation of a $V \times V$ matrix is required. For the low rank trigram, we need to compute a low rank approximation of each slice $C_{\boldsymbol{D}}^{n,(\cdot p)}(:, \tilde{w}_{i-1}, :)\ \forall \tilde{w}_{i-1}$. While this may seem daunting at first, in practice the *size* of each slice (number of non-zero rows/columns) is usually much, much smaller than V, keeping the computation tractable.

Similarly, PLRE also evaluates conditional probabilities at evaluation time efficiently. As shown in Algorithm 20, the normalizer can be precomputed on the sparse powered matrix/tensor. As a result our test complexity is $O(\sum_{i=1}^{\eta_{\text{total}}} \kappa_i)$ where η_{total} is the total number of matrices/tensors in the ensemble. While this is larger than Kneser Ney's practically constant complexity of $O(n)$, it is much faster than other recent methods for language modeling such as neural networks and conditional exponential family models where exact computation of the normalizing constant costs $O(V)$.

Dataset	class-1024(3)	BO-KN(3)	int-KN(3)	BO-MKN(3)	int-MKN(3)	PLRE(3)
Small-English Dev	115.64	99.20	99.73	99.95	95.63	**91.18**
Small-English Test	119.70	103.86	104.56	104.55	100.07	**95.15**
Small-Russian Dev	286.38	281.29	265.71	287.19	263.25	**241.66**
Small-Russian Test	284.09	277.74	262.02	283.70	260.19	**238.96**

Table 9.1: Perplexity results on small corpora for all methods.

9.5 Experiments

To evaluate PLRE, we compared its performance on English and Russian corpora with several variants of KN smoothing, class-based models, and the log-bilinear neural language model (Mnih and Hinton, 2007). We evaluated with perplexity in most of our experiments, but also provide results evaluated with BLEU (Papineni et al., 2002) on a downstream machine translation (MT) task. We have made the code for our approach publicly available [3].

To build the hard class-based LMs, we utilized `mkcls`[4], a tool to train word classes that uses the maximum likelihood criterion (Och, 1995) for classing. We subsequently trained trigram class language models on these classes (corresponding to 2nd-order HMMs) using SRILM (Stolcke, 2002), with KN-smoothing for the class transition probabilities. SRILM was also used for the baseline KN-smoothed models.

For our MT evaluation, we built a hierarchical phrase translation (Chiang, 2007) system using `cdec` (Dyer et al., 2010). The KN-smoothed models in the MT experiments were compiled using KenLM (Heafield, 2011).

9.5.1 Datasets

For the perplexity experiments, we evaluated our proposed approach on 4 datasets, 2 in English and 2 in Russian. In all cases, the singletons were replaced with "<unk>" tokens in the training corpus, and any word not in the vocabulary was replaced with this token during evaluation. There is a general dearth of evaluation on large-scale corpora in morphologically rich languages such as Russian, and thus we have made the processed Large-Russian corpus available for comparison [3].

- **Small-English**: APNews corpus (Bengio et al., 2003): Train - 14 million words, Dev - 963,000, Test - 963,000. Vocabulary- 18,000 types.
- **Small-Russian**: Subset of Russian news commentary data from 2013 WMT translation task[5]: Train- 3.5 million words, Dev - 400,000 Test - 400,000. Vocabulary - 77,000 types.
- **Large-English**: English Gigaword, Training - 837 million words, Dev - 8.7 million, Test - 8.7 million. Vocabulary- 836,980 types.
- **Large-Russian**: Monolingual data from WMT 2013 task. Training - 521 million words, Validation - 50,000, Test - 50,000. Vocabulary- 1.3 million types.

[3]http://www.cs.cmu.edu/~apparikh/plre.html
[4]http://code.google.com/p/giza-pp/
[5]http://www.statmt.org/wmt13/training-monolingual-nc-v8.tgz

For the MT evaluation, we used the parallel data from the WMT 2013 shared task, excluding the Common Crawl corpus data. The newstest2012 and newstest2013 evaluation sets were used as the development and test sets respectively.

9.5.2 Small Corpora

For the class-based baseline LMs, the number of classes was selected from $\{32, 64, 128, 256, 512, 1024\}$ (Small-English) and $\{512, 1024\}$ (Small-Russian). We could not go higher due to the computationally laborious process of hard clustering. For Kneser-Ney, we explore four different variants: back-off (BO-KN) interpolated (int-KN), modified back-off (BO-MKN), and modified interpolated (int-MKN). Good-Turing estimates were used for discounts. All models trained on the small corpora are of order 3 (trigrams).

For PLRE, we used one low rank bigram and one low rank trigram in addition to the MLE n-gram estimates. The powers of the intermediate matrices/tensors were fixed to be 0.5 and the discounts were set to be square roots of the Good Turing estimates (as explained in § 9.4.4). The ranks were tuned on the development set. For Small-English, the ranges were $\{1e-3, 5e-3\}$ (as a fraction of the vocabulary size) for both the low rank bigram and low rank trigram models. For Small-Russian the ranges were $\{5e-4, 1e-3\}$ for both the low rank bigram and the low rank trigram models.

The results are shown in Table 9.1. The best class-based LM is reported, but is not competitive with the KN baselines. PLRE outperforms all of the baselines comfortably. Moreover, PLRE's performance over the baselines is highlighted in Russian. With larger vocabulary sizes, the low rank approach is more effective as it can capture linguistic similarities between rare and common words.

Next we discuss how the maximum n-gram order affects performance. Figure 9.2 shows the relative percentage improvement of our approach over int-MKN as the order is increased from 2 to 4 for both methods. The Small-English dataset has a rather small vocabulary compared to the number of tokens, leading to lower data sparsity in the bigram. Thus the PLRE improvement is small for order = 2, but more substantial for order = 3. On the other hand, for the Small-Russian dataset, the vocabulary size is much larger and consequently the bigram counts are sparser. This leads to similar improvements for all orders (which are larger than that for Small-English).

On both these datasets, we also experimented with tuning the discounts for int-MKN to see if the baseline could be improved with more careful choices of discounts. However, this achieved only marginal gains (reducing the perplexity to 98.94 on the Small-English test set and 259.0 on the Small-Russian test set).

Comparison to neural language models: Although not the focus of this work, we also briefly compare our method to neural language models. Note that neural networks generally have a test complexity of $O(k\sqrt{V})$ using word-classing (Goodman, 2001; Mikolov et al., 2011) which is considerably larger than the test complexity $O(kn)$ of our approach (The test complexity of Kneser-Ney is $O(n)$).

Mnih and Hinton (2007) evaluate on the Small-English dataset (but remove end markers and concatenate the sentences). They obtain perplexities 117.0 and 107.8 using contexts of size 5 and

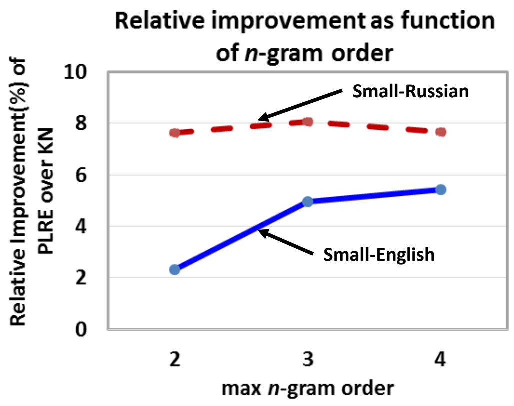


Figure 9.2: Relative percentage improvement of PLRE over int-MKN as the maximum n-gram order for both methods is increased.

Dataset	int-MKN(4)	PLRE(4)
Large-English Dev	73.21	**71.21**
Large-English Test	77.90 ± 0.203	**75.66** ± 0.189
Large-Russian Dev	326.9	**297.11**
Large-Russian Test	289.63 ± 6.82	**264.59** ± 5.839

Table 9.2: Mean perplexity results on large corpora, with standard deviation.

10 respectively. With this preprocessing, a 4-gram (context 3) PLRE achieves 108.4 perplexity. We also obtained results for a recurrent neural network (RNN) model (Mikolov et al., 2010) on the Small-English dataset using the RNNLM toolkit (Mikolov, 2012), and the RNN-ME variant obtains a perplexity of 82.1 (infinite context). Understanding the intuition behind the empirical performance of RNN-ME is the primary inspiration for the future work described in Chapter 10.

9.5.3 Large Corpora

Results on the larger corpora for the top 2 performing methods "PLRE" and "int-MKN" are presented in Table 9.2. Due to the larger training size, we use 4-gram models in these experiments. However, including the low rank 4-gram tensor provided little gain and therefore, the 4-gram PLRE only has additional low rank bigram and low rank trigram matrices/tensors. As above, ranks were tuned on the development set. For Large-English, the ranges were $\{1e-4, 5e-4, 1e-3\}$ (as a fraction of the vocabulary size) for both the low rank bigram and low rank trigram models. For Small-Russian the ranges were $\{1e-5, 5e-5, 1e-4\}$ for both the low rank bigram and the low rank trigram models. For statistical validity, 10 test sets of size equal to the original test set were generated by randomly sampling sentences with replacement from the original test set. Our method outperforms "int-MKN" with gains similar to that on the smaller datasets. As shown in Table 9.3, our method obtains fast training times even for large datasets.

Dataset	PLRE Training Time
Small-English	3.96 min (order 3) / 8.3 min (order 4)
Small-Russian	4.0 min (order 3) / 4.75 min (order 4)
Large-English	3.2 hrs (order 4)
Large-Russian	8.3 hrs (order 4)

Table 9.3: PLRE training times for a fixed parameter setting. 8 Intel Xeon CPUs were used.

Method	BLEU
int-MKN(4)	17.63 ± 0.11
PLRE(4)	17.79 ± 0.07
Smallest Diff	PLRE+0.05
Largest Diff	PLRE+0.29

Table 9.4: Results on English-Russian translation task (mean ± stdev). See text for details.

9.6 Machine Translation Task

Table 9.4 presents results for the MT task, translating from English to Russian[6]. We used MIRA (Chiang et al., 2008) to learn the feature weights. To control for the randomness in MIRA, we avoid retuning when switching LMs - the set of feature weights obtained using int-MKN is the same, only the language model changes. The procedure is repeated 10 times to control for optimizer instability (Clark et al., 2011). Unlike other recent approaches where an additional feature weight is tuned for the proposed model and used in conjunction with KN smoothing (Vaswani et al., 2013), our aim is to show the improvements that PLRE provides as a substitute for KN. On average, PLRE outperforms the KN baseline by 0.16 BLEU, and this improvement is consistent in that PLRE never gets a worse BLEU score.

9.7 Related Work

Recent attempts to revisit the language modeling problem have largely come from two directions: Bayesian nonparametrics and neural networks. Teh (2006) and Goldwater et al. (2006) discovered the connection between interpolated Kneser Ney and the hierarchical Pitman-Yor process. These have led to generalizations that account for domain effects (Wood and Teh, 2009) and unbounded contexts (Wood et al., 2009).

The idea of using neural networks for language modeling is not new (Miikkulainen and Dyer, 1991), but recent efforts (Mnih and Hinton, 2007; Mikolov et al., 2010) have achieved impressive performance. These methods can be quite expensive to train and query (especially as the vocabulary size increases). Techniques such as noise contrastive estimation (Gutmann and Hyvärinen, 2012; Mnih and Teh, 2012; Vaswani et al., 2013), subsampling (Xu et al., 2011), or careful engineering approaches for maximum entropy LMs (which can also be applied to neural networks) (Wu

[5]As described earlier, only the ranks need to be tuned, so only 2-3 low rank bigrams and 2-3 low rank trigrams need to be computed (and combined depending on the setting).

[6]the best score at WMT 2013 was 19.9 (Bojar et al., 2013)

and Khudanpur, 2000) have improved training of these models, but querying the probability of the next word given still requires explicitly normalizing over the vocabulary, which is expensive for big corpora or in languages with a large number of word types. Mnih and Teh (2012) and Vaswani et al. (2013) propose setting the normalization constant to 1, but this is approximate and thus can only be used for downstream evaluation, not for perplexity computation. An alternate technique is to use word-classing (Goodman, 2001; Mikolov et al., 2011), which can reduce the cost of exact normalization to $O(\sqrt{V})$. In contrast, our approach is much more scalable, since it is trivially parallelized in training and does not require explicit normalization during evaluation.

There are a few low rank approaches (Saul and Pereira, 1997; Bellegarda, 2000; Hutchinson et al., 2011), but they are only effective in restricted settings (e.g. small training sets, or corpora divided into documents) and do not generally perform comparably to state-of-the-art models. Roark et al. (2013) also use the idea of marginal constraints for re-estimating back-off parameters for heavily-pruned language models, whereas we use this concept to estimate n-gram specific discounts.

9.8 Conclusion

We presented power low rank ensembles, a technique that generalizes existing n-gram smoothing techniques to non-integer n. By using ensembles of sparse as well as low rank matrices and tensors, our method captures both the fine-grained and coarse structures in word sequences. Our discounting strategy preserves the marginal constraint and thus generalizes Kneser Ney, and under slight changes can also extend other smoothing methods such as deleted-interpolation/Jelinek-Mercer smoothing. Experimentally, PLRE convincingly outperforms Kneser-Ney smoothing as well as class-based baselines.

9.9 Appendix

The primary purpose of this section is to provide a proof of Lemma 29. We also show that Lemma 29 extends to $n > 2$.

9.9.1 Proof of Lemma 29

Proof. Assume the following more general form where multiple low rank matrices can be used i.e.:

$$\begin{aligned} P_{\text{plre}}(w_i|w_{i-1}) = P^{\text{alt}}_{\boldsymbol{D}_0}(w_i|w_{i-1}) + \gamma_0(w_{i-1})\Big(\boldsymbol{Z}^{(\rho_1,\kappa_1)}_{\boldsymbol{D}_1}(w_i|w_{i-1}) + \ldots\ldots \\ + \gamma_{\eta-1}(w_{i-1})\Big(\boldsymbol{Z}^{(\rho_\eta,\kappa_\eta)}_{\boldsymbol{D}_\eta}(w_i|w_{i-1}) + \gamma_\eta(w_{i-1})\Big(\boldsymbol{Z}^{(\rho_{\eta+1}=0,\kappa_{\eta+1}=1)}(w_i|w_{i-1})\Big)\Big)\ldots\Big) \end{aligned} \tag{9.24}$$

where we note that $\boldsymbol{Z}^{(\rho_{\eta+1}=0,\kappa_{\eta+1}=1)}(w_i|w_{i-1})$ is equivalent to $P^{\text{alt}}(w_i)$. It is assumed that $1 \geq \rho_0 \geq \ldots.\rho_{\eta+1} = 0$.

First unroll the recursion and rewrite $P_{\text{plre}}(w_i|w_{i-1})$ as:

$$P_{\text{plre}}(w_i|w_{i-1}) = \sum_{j=0}^{\eta+1} \gamma_{0:j-1}(w_{i-1})\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1})$$

where $\gamma_{0:j-1}(w_{i-1}) = \prod_{h=0}^{j-1} \gamma_h(w_{i-1})$ and $\gamma_{0:-1}(w_{i-1}) = 1$. Note that $P_{\text{pwr}}(w_i|w_{i-1})$ can be written in the same way.

$$P_{pwr}(w_i|w_{i-1}) = \sum_{j=0}^{\eta} \gamma_{0:j}(w_{i-1})\boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j)}(w_i|w_{i-1}) \tag{9.25}$$

Note that $P_{\text{pwr}}(w_i|w_{i-1})$ already satisfies the marginal constraint i.e.

$$\hat{P}(w) = \sum_{w_{i-1}} P_{\text{pwr}}(w_i|w_{i-1})\hat{P}(w_{i-1}) \tag{9.26}$$

because the discounts were chosen such that $P_{\text{pwr}}(w_i|w_{i-1}) = \hat{P}(w_i|w_{i-1})$

Thus it suffices to show that for all $j = 0, \ldots, \eta + 1$:

$$\sum_{w_{i-1}} \hat{P}(w_{i-1})\gamma_{0:j-1}(w_{i-1})\boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j)}(w_i|w_{i-1}) = \sum_{w_{i-1}} \hat{P}(w_{i-1})\gamma_{0:j-1}(w_{i-1})\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1}) \tag{9.27}$$

The statement above is trivially true when $j = 0$. For all other cases, note that due to the way we have set the discounts, $\gamma_{0:j-1}$ takes a special form:

$$\begin{aligned}\prod_{h=0}^{j-1} \gamma_h(w_{i-1}) &= \frac{d_* \sum_i c_{i,i-1}^{\rho_1}}{\sum_i c_{i,i-1}^{\rho_0}} \frac{d_* \sum_i c_{i,i-1}^{\rho_2}}{\sum_i c_{i,i-1}^{\rho_1}} \ldots \frac{d_* \sum_i c_{i,i-1}^{\rho_j}}{\sum_i c_{i,i-1}^{\rho_{j-1}}} \\ &= \frac{(d_*)^j \sum_i c_{i,i-1}^{\rho_j}}{\sum_i c_{i,i-1}}\end{aligned} \tag{9.28}$$

Using this form in Eq. 5 and simplifying yields:

$$\textstyle\sum_{w_{i-1}} \left(\sum_i c_{i,i-1}^{\rho_j}\right) \boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j)}(w_i|w_{i-1}) = \sum_{w_{i-1}} \left(\sum_i c_{i,i-1}^{\rho_j}\right) \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1})$$

which is equivalent to requiring that

$$\sum_{w_{i-1}} \boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j)}(w_i, w_{i-1}) = \sum_{w_{i-1}} \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i, w_{i-1}) \tag{9.29}$$

which holds because rank minimization under gKL preserves row and column sums. $\square$

9.9.2 Generalization to $n > 2$

Theorem 5. *Let $P_{plre}(w_i|w_{i-n+1}^{i-1})$ indicate the PLRE smoothed conditional probability and $\hat{P}(w)$ indicate the MLE probability of w. Then,*

$$\hat{P}(w) = \sum_{w_{i-n+1}^{i-1}} P_{plre}(w_i|w_{i-n+1}^{i-1})\hat{P}(w_{i-n+1}^{i-1}) \tag{9.30}$$

Proof. Recall that,

$$\begin{aligned} P_{\text{plre}}(w_i|w_{i-n+1}^{i-1}) &= P_{\boldsymbol{D}_0}^{\text{alt}}(w_i|w_{i-n+1}^{i-1}) \\ &+ \gamma_0(w_{i-n+1}^{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_1}^{(\rho_1,\kappa_1)}(w_i|w_{i-n+1}^{i-1}) + \ldots.. \\ &+ \gamma_{\eta-1}(w_{i-n+1}^{i-1})\Big(\boldsymbol{Z}_{\boldsymbol{D}_\eta}^{(\rho_\eta,\kappa_\eta)}(w_i|w_{i-n+1}^{i-1}) \\ &+ \gamma_\eta(w_{i-n+1}^{i-1})\Big(P_{\text{plre}}(w_i|w_{i-n+2}^{i-1})\Big)\Big)\ldots\Big) \end{aligned} \tag{9.31}$$

Define,

$$\begin{aligned} P_{\text{pwr}}(w_i|w_{i-n+1}^{i-1}) &= P_{\boldsymbol{D}_0}^{\text{alt}}(w_i|w_{i-n+1}^{i-1}) \\ &+ \gamma_0(w_{i-n+1}^{i-1})\Big(\boldsymbol{Y}_{\boldsymbol{D}_1}^{(\rho_1,\kappa_1)}(w_i|w_{i-n+1}^{i-1}) + \ldots.. \\ &+ \gamma_{\eta-1}(w_{i-n+1}^{i-1})\Big(\boldsymbol{Y}_{\boldsymbol{D}_\eta}^{(\rho_\eta,\kappa_\eta)}(w_i|w_{i-n+1}^{i-1}) \\ &+ \gamma_\eta(w_{i-n+1}^{i-1})\Big(P_{\text{pwr}}(w_i|w_{i-n+2}^{i-1})\Big)\Big)\ldots\Big) \end{aligned} \tag{9.32}$$

where with a little abuse of notation

$$\boldsymbol{Y}_{\boldsymbol{D}_j}^{\rho_j}(w_i|w_{i-n'+1}^{i-1}) = \frac{\tilde{c}(w_i, w_{i-n'+1}^{i-1})^{\rho_j} - \boldsymbol{D}_j(w_i, w_{i-n'+1}^{i-1})}{\sum_{w_i} \tilde{c}(w_i, w_{i-n'+1}^{i-1})^{\rho_j}} \tag{9.33}$$

and

$$\tilde{c}(w_i, w_{i-n'+1}^{i-1}) = \begin{cases} c(w_i, w_{i-n'+1}^{i-1}), \text{ if } n' = n \\ \\ N_-(w_{i-n'+1}^{i}) \quad \text{if } n' < n \end{cases}$$

Furthermore, define

$$\begin{aligned} P^{\text{terms}}_{\text{pwr}}(w_i|w^{i-1}_{i-n'+1}) &= P^{\text{alt}}_{\boldsymbol{D}_0}(w_i|w^{i-1}_{i-n'+1}) \\ &+ \gamma_0(w^{i-1}_{i-n'+1})\Big(\boldsymbol{Y}^{(\rho_1,\kappa_1)}_{\boldsymbol{D}_1}(w_i|w^{i-1}_{i-n'+1}) + ... \\ &+ \gamma_{\eta-1}(w^{i-1}_{i-n'+1})\Big(\boldsymbol{Y}^{(\rho_\eta,\kappa_\eta)}(w_i|w^{i-1}_{i-n'+1})\Big)...\Big) \end{aligned} \tag{9.34}$$

Note that because of the way the discounts are computed in Algorithm 19,

$$P^{\text{terms}}_{\text{pwr}}(w_i|w^{i-1}_{i-n'+1}) = P^{\text{alt}}(w_i|w^{i-1}_{i-n'+1}) \tag{9.35}$$

for all $n' \leq n$.

As a result, (for some choice of discount)

$$P_{\text{pwr}}(w_i|w^{i-1}_{i-n+1}) = P_{\text{kn}}(w_i|w^{i-1}_{i-n+1}) \tag{9.36}$$

Since, we know that Kneser Ney satisfies the marginal constraint (Chen and Goodman, 1999) this implies that,

$$\hat{P}(w) = \sum_{w^{i-1}_{i-n+1}} P_{\text{pwr}}(w_i|w^{i-1}_{i-n+1})\hat{P}(w^{i-1}_{i-n+1}) \tag{9.37}$$

Thus, all we have to do is prove that

$$\sum_{w^{i-1}_{i-n+1}} P_{\text{pwr}}(w_i|w^{i-1}_{i-n+1})\hat{P}(w^{i-1}_{i-n+1}) = \sum_{w^{i-1}_{i-n+1}} P_{\text{plre}}(w_i|w^{i-1}_{i-n+1})\hat{P}(w^{i-1}_{i-n+1}) \tag{9.38}$$

Now, we follow the same argument as with $n = 2$ (i.e. unrolling the recursion and applying the fact that gKL preserves row/column sums).

For notational simplicity assume that $n = 3$. Then, we can write $P_{\text{pwr}}(w_i|w^{i-1}_{i-n+1})$ as:

$$P_{\text{pwr}}(w_i|w^{i-1}_{i-2}) = \sum_{j=0}^{\eta} \gamma_{0:j-1}(w^{i-1}_{i-2})\boldsymbol{Y}^{(\rho_j,\kappa_j)}_{\boldsymbol{D}_j}(w_i|w^{i-1}_{i-2}) + \sum_{j=0}^{\eta+1} \gamma_{0:\eta}(w^{i-1}_{i-2})\gamma_{0:j-1}(w_{i-1})\boldsymbol{Y}^{(\rho_j,\kappa_j)}_{\boldsymbol{D}_j}(w_i|w_{i-1}) \tag{9.39}$$

where $\gamma_{0:-1}(w_{i-2}^{i-1}) = 1$ and

$$\gamma_{0:j-1}(w_{i-2}^{i-1}) = \prod_{h=0}^{j-1} \gamma_h(w_{i-2}^{i-1}) = \frac{d_* \sum_i \tilde{c}_{i,i-1,i-2}^{\rho_1}}{\sum_i \tilde{c}_{i,i-1,i-2}^{\rho_0}} \frac{d_* \sum_i \tilde{c}_{i,i-1,i-2}^{\rho_2}}{\sum_i \tilde{c}_{i,i-1,i-2}^{\rho_1}} \dots \frac{d_* \sum_i \tilde{c}_{i,i-1,i-2}^{\rho_j}}{\sum_i \tilde{c}_{i,i-1,i-2}^{\rho_{j-1}}}$$
$$= \frac{(d_*)^j \sum_i \tilde{c}_{i,i-1,i-2}^{\rho_j}}{\sum_i \tilde{c}_{i,i-1,i-2}} \quad (9.40)$$

Here $\tilde{c}_{i,i-1,i-2}$ is shorthand for $\tilde{c}(w_i, w_{i-2}^{i-1})$.

Similarly, $\gamma_{0:-1}(w_{i-1}) = 1$ and

$$\gamma_{0:j-1}(w_{i-1}) = \prod_{h=0}^{j-1} \gamma_h(w_{i-1}) = \frac{d_* \sum_i \tilde{c}_{i,i-1}^{\rho_1}}{\sum_i \tilde{c}_{i,i-1}^{\rho_0}} \frac{d_* \sum_i \tilde{c}_{i,i-1}^{\rho_2}}{\sum_i \tilde{c}_{i,i-1}^{\rho_1}} \dots \frac{d_* \sum_i \tilde{c}_{i,i-1}^{\rho_j}}{\sum_i \tilde{c}_{i,i-1}^{\rho_{j-1}}}$$
$$= \frac{(d_*)^j \sum_i \tilde{c}_{i,i-1}^{\rho_j}}{\sum_i \tilde{c}_{i,i-1}} \quad (9.41)$$

(Again, it is assumed that $1 \geq \rho_0 \geq \dots \rho_{\eta+1} = 0$.)

Analogously,

$$P_{\text{plre}}(w_i|w_{i-2}^{i-1}) = \sum_{j=0}^{\eta} \gamma_{0:j-1}(w_{i-2}^{i-1}) \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-2}^{i-1}) + \sum_{j=0}^{\eta+1} \gamma_{0:\eta}(w_{i-2}^{i-1}) \gamma_{0:j-1}(w_{i-1}) \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_h,\kappa_j)}(w_i|w_{i-1}) \quad (9.42)$$

Now for any trigram term we prove that

$$\sum_{w_{i-2}^{i-1}} \gamma_{0:j-1}(w_{i-2}^{i-1}) \boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-2}^{i-1}) \hat{P}(w_{i-2}^{i-1}) = \sum_{w_{i-2}^{i-1}} \gamma_{0:j-1}(w_{i-2}^{i-1}) \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-2}^{i-1}) \hat{P}(w_{i-2}^{i-1}) \quad (9.43)$$

Plugging in the definition of $\gamma_{0:j-1}$ and simplifying gives

$$\sum_{w_{i-2}^{i-1}} (\sum_i \tilde{c}_{i,i-1,i-2}^{\rho_j}) \boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-2}^{i-1}) = \sum_{w_{i-2}^{i-1}} (\sum_i \tilde{c}_{i,i-1,i-2}^{\rho_j}) \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-2}^{i-1}) \quad (9.44)$$

which is equivalent to

$$\sum_{w_{i-2}^{i-1}} \boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i, w_{i-2}^{i-1}) = \sum_{w_{i-2}^{i-1}} \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i, w_{i-2}^{i-1}) \quad (9.45)$$

which holds because of the definition of $\boldsymbol{Z}$ and the fact that rank minimization under *gKL* preserves row/column sums.

Now consider any bigram term. We seek to show that:

$$\sum_{w_{i-2}^{i-1}} \gamma_{0:\eta}(w_{i-2}^{i-1})\gamma_{0:j-1}(w_{i-1})\boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1})\hat{P}(w_{i-2}^{i-1})$$
$$= \sum_{w_{i-2}^{i-1}} \gamma_{0:\eta}(w_{i-2}^{i-1})\gamma_{0:j-1}(w_{i-1})\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1})\hat{P}(w_{i-2}^{i-1}) \tag{9.46}$$

Substituting definition of $\gamma_{0:\eta}(w_{i-2}^{i-1})$ gives

$$\sum_{w_{i-2}^{i-1}} \frac{(d_*)^{\eta+1}\sum_i \tilde{c}_{i,i-1,i-2}^{\rho_{\eta+1}}}{\sum_i \tilde{c}_{i,i-1,i-2}}\gamma_{0:j-1}(w_{i-1})\boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1})\hat{P}(w_{i-2}^{i-1})$$
$$= \sum_{w_{i-2}^{i-1}} \frac{(d_*)^{\eta+1}\sum_i \tilde{c}_{i,i-1,i-2}^{\rho_{\eta+1}}}{\sum_i \tilde{c}_{i,i-1,i-2}}\gamma_{0:j-1}(w_{i-1})\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1})\hat{P}(w_{i-2}^{i-1}) \tag{9.47}$$

Simplifying and pushing in the sum over w_{i-2} gives,

$$\sum_{w_{i-1}}(\sum_{i,i-2} \tilde{c}_{i,i-1,i-2}^{\rho_{\eta+1}})\gamma_{0:j-1}(w_{i-1})\boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1}) = \sum_{w_{i-1}}(\sum_{i,i-2} \tilde{c}_{i,i-1,i-2}^{\rho_{\eta+1}})\gamma_{0:j-1}(w_{i-1})\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1}) \tag{9.48}$$

Note that since $\rho_{\eta+1} = 0$, $\sum_{i,i-2} \tilde{c}_{i,i-1,i-2}^{\rho_{\eta+1}=0} = \sum_i \tilde{c}_{i,i-1}$ (by definition of $\tilde{c}$).

Using this fact and substituting definition of $\gamma_{0:j-1}(w_{i-1})$ gives

$$\sum_{w_{i-1}}(\sum_{i} \tilde{c}_{i,i-1})\frac{(d_*)^j\sum_i \tilde{c}_{i,i-1}^{\rho_j}}{\sum_i \tilde{c}_{i,i-1}}\boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1}) = \sum_{w_{i-1}}(\sum_{i} \tilde{c}_{i,i-1})\frac{(d_*)^j\sum_i \tilde{c}_{i,i-1}^{\rho_j}}{\sum_i \tilde{c}_{i,i-1}}\boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i|w_{i-1}) \tag{9.49}$$

Simplifying gives,

$$\sum_{w_{i-1}} \boldsymbol{Y}_{\boldsymbol{D}_j}^{(\rho_j)}(w_i, w_{i-1}) = \sum_{w_{i-1}} \boldsymbol{Z}_{\boldsymbol{D}_j}^{(\rho_j,\kappa_j)}(w_i, w_{i-1}) \tag{9.50}$$

which holds because rank minimization under KL divergence preserves row and column sums.
□

Chapter 10

Conclusion

In this thesis, we have approached probabilistic modeling through the lens of linear and tensor algebra. Viewing latent variable models from this perspective allowed us to leverage tools such as matrix/tensor decomposition, inversion, and additive metrics to propose many novel solutions to parameter and structure learning as well as modeling with latent variables.

In particular, Part I proposed spectral learning algorithms for latent tree graphical models (Chapter 4 and 7) and latent junction trees (Chapter 5). Unlike traditional methods, our algorithms are provably consistent, local-optima-free, and 1-2 orders of magnitude faster than EM for large sample sizes. They also easily generalize to the continuous, nonparametric setting (Chapter 6).

In Part II, we used these insights to approach latent variable modeling for NLP from a linear algebra perspective. Our method for unsupervised parsing (Chapter 8) is the first algorithm that has both theoretical guarantees and is also practical, performing favorably to the CCM method of Klein and Manning. We also developed power low rank ensembles (Chapter 9), a framework for language modeling that generalizes existing n-gram techniques to non-integer n. It consistently outperforms state-of-the-art Kneser Ney baselines and can train on billion-word datasets in a few hours.

Finally, this thesis also leaves open many directions for future work, two of which are described below.

Discriminative spectral learning methods: Spectral learning methods have been developed primarily for generative latent variable modeling. However, in many cases, discriminative techniques lead to better predictive power. A natural question to ask is can spectral learning be integrated with discriminative training?

This is an exciting problem that has recently been of interest in the spectral learning community (Chaganty and Liang, 2013; Quattoni et al., 2014) and is challenging since most discriminative methods (even for fully observed models) rely on optimization that can be difficult to integrate with spectral estimation. One promising idea is to discriminatively learn the projection matrix $\boldsymbol{U}_i$ that would learn a subspace that is more optimal for the prediction problem (an idea suggested by Quattoni et al. (2014) for sequence models). There has been work along these lines in context of oriented PCA (Platt et al., 2010) for translingual document representations.

Connecting Neural Networks and Latent Variable Models: This thesis has sought to develop connections between low rank decomposition and latent variable modeling, two different formulations of a similar intuition: that there exists an underlying "simpler" explanation of the data that is unobserved. However, neural networks are another popular technique that exploit a similar insight. Can these approaches be related to either linear algebra methods and/or latent variable models?

For example, let us consider recurrent neural networks for language modeling (Mikolov et al., 2010) that model sequences of words. Let $\boldsymbol{w}_t$ denote the $V \times 1$ indicator vector representation for w_t (the t^{th} word in the sequence) and $\boldsymbol{w}_t(j)$ indicate its j^{th} index. Let $\boldsymbol{s}_t$ denote the $K \times 1$ dimensional state vector at position t and $\boldsymbol{y}_t$ be the $V \times 1$ dimensional output vector that will represent the estimated probability $\widehat{P}(w_{t+1}|w_1, ..., w_t)$. Then, a recurrent neural network is defined by the following equations:

$$\boldsymbol{s}_t(j) = f\left(\sum_{i=1}^{V} \boldsymbol{w}_t(i)\mu_{ji} + \sum_{l=1}^{K} \boldsymbol{s}_{t-1}(l)\kappa_{jl}\right) \tag{10.1}$$

$$\boldsymbol{y}_t(m) = g\left(\sum_{j=1}^{K} \boldsymbol{s}_t(j)\nu_{mj}\right) \tag{10.2}$$

where $\mu_{ji}, \kappa_{jl}, \nu_{mj}$ are weights to be learned. Typically $\boldsymbol{f}$ and $\boldsymbol{g}$ are set to:

$$f(z) = \frac{1}{1 + \exp(-z)} \tag{10.3}$$

$$g(z_m) = \frac{\exp(z_m)}{\sum_k \exp(z_k)} \tag{10.4}$$

Therefore in a recurrent neural net, $\boldsymbol{y}_t$ only depends on the state vector $\boldsymbol{s}_t$. $\boldsymbol{s}_{t+1}$ depends on $\boldsymbol{w}_t$ and $\boldsymbol{s}_t$. As a result, the RNN requires $O(K^2 + KV)$ parameters where K is the size of the latent dimension and V is the vocabulary size.

Note that the dependencies in an RNN are different from a traditional hidden Markov model (HMM) in one aspect, namely that $\boldsymbol{s}_{t+1}$ depends on both $\boldsymbol{s}_t$ and $\boldsymbol{w}_t$. This is different from a traditional HMM where the state at time t only depends on the state $t-1$ via the transition matrix.

However, it is easy to alter the HMM so that it can also have the same dependences as the RNN with the same number of parameters. In particular we can have define the following model parameters for the prior, transition, and observation probabilities respectively:

$$\pi(i) := P(h_t = i) \tag{10.5}$$

$$\boldsymbol{T}(i, j, m) := P(h_{t+1} = i|h_t = j, w_t = m) := \frac{\exp(\alpha_{i,j} + \beta_{i,m})}{\sum_n \exp(\alpha_{nj} + \beta_{nm})} \tag{10.6}$$

$$\boldsymbol{O}(m, i) := P(w_t = m|h_t = i) \tag{10.7}$$

Note that here we have denoted the state by h instead of $\boldsymbol{s}$ as with the RNN since both models don't have the same definition of state. A state in an HMM is a discrete valued variable (although the probability distribution over it is a real-valued vector) while the RNN state $\boldsymbol{s}$ is itself a real-valued (non-probabilistic) vector. Moreover the transition tensor $\boldsymbol{T}$ is now parameterized using log-linear models so that the number of parameters is still $O(K^2 + KV)$.

However, despite their similarity, there have been no theoretical or empirical comparisons between these two models. Such a comparison could shed light into the differences in representation power between these two classes of approaches.

These are just two possible future directions of this thesis. Spectral probabilistic modeling is still a nascent field with many open questions. Its strong mathematical foundation make it a natural framework for studying theoretical questions, and its natural scalability make it very practical for large scale data analysis. Thus, it has the potential to continue to have an impact on machine learning and natural language processing for many years to come.

Bibliography

Akritas, A. G., Akritas, E. K., and Malaschonok, G. I. (1996). Various proofs of sylvester's (determinant) identity. *Mathematics and Computers in Simulation*, 42(4):585–593.

Anandkumar, A., Chaudhuri, K., Hsu, D., Kakade, S. M., Song, L., and Zhang, T. (2011). Spectral methods for learning multivariate latent tree structure. *arXiv preprint arXiv:1107.1283*.

Anandkumar, A., Foster, D. P., Hsu, D., Kakade, S. M., and Liu, Y.-K. (2012). Two svds suffice: Spectral decompositions for probabilistic topic modeling and latent dirichlet allocation. *arXiv preprint arXiv:1204.6703*.

Anandkumar, A., Ge, R., Hsu, D., and Kakade, S. M. (2013). A tensor spectral approach to learning mixed membership community models. *arXiv preprint arXiv:1302.2684*.

Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., Wu, Y., and Zhu, M. (2012a). A practical algorithm for topic modeling with provable guarantees. *arXiv preprint arXiv:1212.4777*.

Arora, S., Ge, R., and Moitra, A. (2012b). Learning topic models–going beyond svd. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10. IEEE.

Asuncion, A. and Newman, D. (2007). Uci machine learning repository.

Atteson, K. (1997). The performance of neighbor-joining algorithms of phylogeny reconstruction. In *Computing and Combinatorics*, pages 101–110. Springer.

Bach, F. R. and Jordan, M. I. (2003). Kernel independent component analysis. *The Journal of Machine Learning Research*, 3:1–48.

Bailly, R., Carreras, X., Luque, F. M., and Quattoni, A. (2013). Unsupervised spectral learning of WCFG as low-rank matrix completion. In *Proceedings of EMNLP*.

Bailly, R., Denis, F., and Ralaivola, L. (2009). Grammatical inference as a principal component analysis problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 33–40. ACM.

Bailly, R., Habrard, A., and Denis, F. (2010). A spectral approach for probabilistic grammatical inference on trees. In *Algorithmic Learning Theory*, pages 74–88. Springer.

Balakrishnan, S., Wainwright, M. J., and Yu, B. (2014). Statistical guarantees for the em algorithm: From population to sample-based analysis. *arXiv preprint arXiv:1408.2156*.

Baldi, P. et al. (2001). *Bioinformatics: the machine learning approach*. The MIT Press.

Balle, B., Quattoni, A., and Carreras, X. (2011). A spectral learning algorithm for finite state transducers. In *Machine Learning and Knowledge Discovery in Databases*, pages 156–171. Springer.

Balle, B., Quattoni, A., and Carreras, X. (2012). Local loss optimization in operator models: A new insight into spectral learning. *arXiv preprint arXiv:1206.6393*.

Bellegarda, J. R. (2000). Large vocabulary speech recognition with multispan statistical language models. *IEEE Transactions on Speech and Audio Processing*, 8(1):76–84.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bickson, D. (2008). Gaussian belief propagation: Theory and application. *Arxiv preprint arXiv:0811.2518*.

Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013). Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Boots, B. and Gordon, G. (2013). A spectral learning approach to range-only slam. In *Proc. 30th Intl. Conf. on Machine Learning (ICML)*.

Boots, B., Siddiqi, S. M., and Gordon, G. J. (2010). Closing the learning-planning loop with predictive state representations. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1369–1370. International Foundation for Autonomous Agents and Multiagent Systems.

Brown, P., Desouza, P., Mercer, R., Pietra, V., and Lai, J. (1992). Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Buneman, O. P. (1971). The recovery of trees from measures of dissimilarity. *Mathematics in the archaeological and historical sciences*.

Buneman, P. (1974). A note on the metric properties of trees. *Journal of Combinatorial Theory, Series B*, 17(1):48–50.

Cai, J.-F., Candès, E. J., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982.

Candès, E. J. and Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717–772.

Casella, G. and Berger, R. L. (1990). *Statistical inference*, volume 70. Duxbury Press Belmont, CA.

Casella, G. and Berger, R. L. (2002). *Statistical inference*, volume 2. Duxbury Pacific Grove, CA.

Chaganty, A. T. and Liang, P. (2013). Spectral experts for estimating mixtures of linear regressions. *arXiv preprint arXiv:1306.3729*.

Chaganty, A. T. and Liang, P. (2014). Estimating latent-variable graphical models using moments and likelihoods. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1872–1880.

Chen, S. F. (2009). Shrinking exponential language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 468–476, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

Chen, S. F. and Rosenfeld, R. (2000). A survey of smoothing techniques for me models. *Speech and Audio Processing, IEEE Transactions on*, 8(1):37–50.

Chiang, D. (2007). Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 224–233. Association for Computational Linguistics.

Choi, M. J., Tan, V. Y., Anandkumar, A., and Willsky, A. S. (2010). Learning latent tree graphical models. In *arXiv:1009.2722v1*.

Choi, M. J., Tan, V. Y., Anandkumar, A., and Willsky, A. S. (2011). Learning latent tree graphical models. *The Journal of Machine Learning Research*, 12:1771–1812.

Chow, C. and Liu, C. (1968a). Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467.

Chow, C. and Liu, C. (1968b). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

Chow, C. K. and Liu, C. N. (1968c). Approximating Discrete Probability Distributions With Dependence Trees. *IEEE Transactions on Information Theory*, IT-14:462–467.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181.

Cohen, S. and Collins, M. (2012). Tensor decomposition for fast parsing with latent-variable pcfgs. In *Advances in Neural Information Processing Systems 25*, pages 2528–2536.

Cohen, S., Stratos, K., Collins, M., Foster, D., and Ungar, L. (2012). Spectral learning of latent-variable pcfgs. In *Association of Computational Linguistics (ACL)*, volume 50.

Cohen, S. B. and Collins, M. (2014). A provably correct learning algorithm for latent-variable pcfgs.

Cohen, S. B. and Smith, N. A. (2009). Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of HLT-NAACL*.

Cohen, S. B. and Smith, N. A. (2010). Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization. In *Proceedings of ACL*.

Cohen, S. B. and Smith, N. A. (2012). Empirical risk minimization for probabilistic grammars: Sample complexity and hardness of learning. *Computational Linguistics*, 38(3):479–526.

Cowell, R., Dawid, A., Lauritzen, S., and Spiegelhalter, D. (1999). *Probabilistic Networks and Expert Sytems*. Springer, New York.

Dasgupta, S. (1999). Learning mixtures of gaussians. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 634–644. IEEE.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–22.

Desper, R. and Gascuel, O. (2005). The minimum evolution distance-based approach to phylogenetic inference. *Mathematics of evolution and phylogeny*, pages 1–32.

Dhillon, P. S., Rodu, J., Collins, M., Foster, D. P., and Ungar, L. H. (2012a). Spectral dependency parsing with latent variables. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 205–213. Association for Computational Linguistics.

Dhillon, P. S., Rodu, J., Foster, D. P., and Ungar, L. H. (2012b). Two step cca: A new spectral method for estimating vector models of words. In *Proceedings of the 29th International Conference on Machine learning*, ICML'12.

Ding, W., Rohban, M. H., Ishwar, P., and Saligrama, V. (2013). Topic discovery through data dependent and random projections. *arXiv preprint arXiv:1303.3664*.

Dyer, C., Weese, J., Setiawan, H., Lopez, A., Ture, F., Eidelman, V., Ganitkevitch, J., Blunsom, P., and Resnik, P. (2010). cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12. Association for Computational Linguistics.

Edmonds, J. (1967). Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(4):233–240.

Eisner, J. and Satta, G. (1999). Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of ACL*.

Erdõs, P. L., Steel, M. A., Székely, L., and Warnow, T. J. (1999). A few logs suffice to build (almost) all trees: Part ii. *Theoretical Computer Science*, 221(1):77–118.

Fine, S. and Scheinberg, K. (2002). Efficient svm training using low-rank kernel representations. *The Journal of Machine Learning Research*, 2:243–264.

Ghahramani, Z. and Jordan, M. (1997). Factorial hidden Markov models. *Machine learning*, 29(2):245–273.

Gillenwater, J., Ganchev, K., Graça, J., Pereira, F., and Taskar, B. (2010). Sparsity in dependency grammar induction. In *Proceedings of ACL*.

Gimpel, K. and Smith, N. (2012). Concavity and initialization for unsupervised dependency parsing. In *Proceedings of NAACL*.

Goldwater, S., Griffiths, T., and Johnson, M. (2006). Interpolating between types and tokens by estimating power-law generators. In *Advances in Neural Information Processing Systems*, volume 18.

Golland, D. and DeNero, J. (2012). A feature-rich constituent context model for grammar induction. In *Proceedings of ACL*.

Goodman, J. (2001). Classes for fast maximum entropy training. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 1, pages 561–564. IEEE.

Gormley, M. and Eisner, J. (2013). Nonconvex global optimization for latent-variable models. In *Proceedings of ACL*.

Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

Gutmann, M. and Hyvärinen, A. (2012). Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13:307–361.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Harmeling, S. and Williams, C. K. (2011). Greedy learning of binary latent trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(6):1087–1097.

Harshman, R. A. (1970). Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer Verlag.

Headden, W. P., Johnson, M., and McClosky, D. (2009). Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proc. of NAACL-HLT*.

Heafield, K. (2011). KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, United Kingdom.

Heller, K. A. and Ghahramani, Z. (2005). Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pages 297–304. ACM.

Hitchcock, F. L. (1927). *The expression of a tensor or a polyadic as a sum of products*. sn.

Ho, N.-D. and Van Dooren, P. (2008). Non-negative matrix factorization with fixed row and column sums. *Linear Algebra and its Applications*, 429(5):1020–1025.

Horn, R. and Johnson, C. (1990). *Matrix analysis*. Cambridge Univ Pr.

Hsu, D., Kakade, S., and Zhang, T. (2009). A spectral algorithm for learning hidden Markov models. In *Proc. Annual Conf. Computational Learning Theory*.

Hsu, D., Kakade, S. M., and Liang, P. (2012). Identifiability and unmixing of latent parse trees. In *Advances in NIPS*.

Hutchinson, B., Ostendorf, M., and Fazel, M. (2011). Low rank language models for small training sets. *Signal Processing Letters, IEEE*, 18(9):489–492.

Ishteva, M., Park, H., and Song, L. (2012). Unfolding latent tree structures using 4th order tensors. *arXiv preprint arXiv:1210.1258*.

Jelinek, F., Lafferty, J. D., and Mercer, R. L. (1992). *Basic methods of probabilistic context free grammars*. Springer.

Jelinek, F. and Mercer, R. (1980). Interpolated estimation of markov source parameters from sparse data. *Pattern recognition in practice*.

Jiang, J., Rai, P., and Iii, H. D. (2011). Message-passing for approximate map inference with latent variables. In *Advances in Neural Information Processing Systems*, pages 1197–1205.

Katayama, T. (2005). *Subspace methods for system identification*. Springer.

Klein, D. and Manning, C. D. (2002). A generative constituent-context model for improved grammar induction. In *Proceedings of ACL*.

Klein, D. and Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184. IEEE.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Kolar, M., Parikh, A. P., and Xing, E. P. (2010a). On sparse nonparametric conditional covariance selection. In *Proceedings of ICML*.

Kolar, M., Song, L., Ahmed, A., and Xing, E. P. (2010b). Estimating time-varying networks. *The Annals of Applied Statistics*, 4(1):94–123.

Kolda, T. and Bader, B. (2009a). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Kolda, T. G. and Bader, B. W. (2009b). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. The MIT Press.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Kundu, A., He, Y., and Bahl, P. (1989). Recognition of handwritten word: first and second order hidden Markov model based approach. *Pattern recognition*, 22(3):283–297.

Lake, J. A. (1994). Reconstructing evolutionary trees from dna and protein sequences: paralinear distances. *Proceedings of the National Academy of Sciences*, 91(4):1455–1459.

Lee, D. D. and Seung, H. S. (2001). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562.

Liang, P. and Klein, D. (2009). Online em for unsupervised models. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 611–619. Association for Computational Linguistics.

Liu, H., Lafferty, J., and Wasserman, L. (2009). The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *The Journal of Machine Learning Research*, 10:2295–2328.

Liu, Q. and Ihler, A. (2013). Variational algorithms for marginal map. *arXiv preprint arXiv:1302.6584*.

Luque, F. M., Quattoni, A., Balle, B., and Carreras, X. (2012). Spectral learning for non-deterministic dependency parsing. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 409–419. Association for Computational Linguistics.

Mackey, L., Talwalkar, A., and Jordan, M. I. (2011). Divide-and-conquer matrix factorization. *arXiv preprint arXiv:1107.0789*.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19:313–330.

Melnyk, I. and Banerjee, A. (2014). A spectral algorithm for inference in hidden semi-markov models. *arXiv preprint arXiv:1407.3422*.

Miikkulainen, R. and Dyer, M. G. (1991). Natural language processing with modular pdp networks and distributed lexicon. *Cognitive Science*, 15:343–399.

Mikolov, T. (2012). Rnnlm toolkit.

Mikolov, T., Karafit, M., Burget, L., ?ernock?, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, volume 2010, pages 1045–1048. International Speech Communication Association.

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., and Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Mnih, A. and Hinton, G. (2007). Three new graphical models for statistical language modelling. In *Proceedings of the 24th international conference on Machine learning*, pages 641–648. ACM.

Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference on Machine Learning*.

Mossel, E. and Roch, S. (2005). Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 366–375. ACM.

Mossel, E. and Roch, S. (2006). Learning nonsingular phylogenies and hidden Markov models. *AOAP*, 16(2):583–614.

Murphy, K. (2002). *Dynamic bayesian networks: representation, inference and learning*. PhD thesis, University of California.

Murphy, K. (2005). Hidden Markov model (HMM) toolbox for matlab http://www.cs.ubc.ca/murphyk/software/.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. The MIT Press.

Nelakanti, A. K., Archambeau, C., Mairal, J., Bach, F., and Bouchard, G. (2013). Structured penalties for log-linear language models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 233–243, Seattle, Washington, USA. Association for Computational Linguistics.

Ney, H., Essen, U., and Kneser, R. (1994). On Structuring Probabilistic Dependencies in Stochastic Language Modelling. *Computer Speech and Language*, 8:1–38.

Nguyen, N., Drineas, P., and Tran, T. (2010). Tensor sparsification via a bound on the spectral norm of random tensors. *Arxiv preprint arXiv:1005.4732*.

Nguyen, T., Hu, Y., and Boyd-Graber, J. (2014). Anchors regularized: Adding robustness and extensibility to scalable topic-modeling algorithms. In *Association for Computational Linguistics*.

Och, F. J. (1995). Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung. Bachelor's thesis (Studienarbeit), University of Erlangen.

Papineni, K., Roukos, S., Ward, T., and jing Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Parikh, A., Song, L., and Xing, E. (2011). A spectral algorithm for latent tree graphical models. In *Proceedings of the 28th International Conference on Machine Learning*, pages 1065–1072. ACM.

Parikh, A. P., Cohen, S. B., and Xing, E. P. (2014a). Spectral unsupervised parsing with additive tree metrics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Long Papers. Association for Computational Linguistics*, volume 2, page 1.

Parikh, A. P., Cohen, S. B., and Xing, E. P. (2014b). Spectral unsupervised parsing with additive tree metrics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Long Papers. Association for Computational Linguistics*, volume 2, page 1.

Parikh, A. P., Song, L., Ishteva, M., Teodoru, G., and Xing, E. P. (2012). A spectral algorithm for latent junction trees. *arXiv preprint arXiv:1210.4884*.

Pearl, J. (1988). Probabilistic inference in intelligent systems.

Petrov, S., Das, D., and McDonald, R. (2011). A universal part-of-speech tagset. *ArXiv:1104.2086*.

Platt, J. C., Toutanova, K., and Yih, W.-t. (2010). Translingual document representations from discriminative projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 251–261. Association for Computational Linguistics.

Poon, L., Zhang, N. L., Chen, T., and Wang, Y. (2010). Variable selection in model-based clustering: To do or to facilitate. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 887–894.

Quattoni, A., Balle, B., Carreras, X., and Globerson, A. (2014). Spectral regularization for max-margin sequence tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1710–1718.

Rabiner, L. and Juang, B.-H. (1993). Fundamentals of speech recognition.

Redmond, M. and Baveja, A. (2002). A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research*, 141(3):660–678.

Roark, B., Allauzen, C., and Riley, M. (2013). Smoothed marginal distribution constraints for language modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 43–52.

Roy, B. (2011). Bounds on the expected entropy and kl-divergence of sampled multinomial distributions.

Rzhetsky, A. and Nei, M. (1993). Theoretical foundation of the minimum-evolution method of phylogenetic inference. *Molecular Biology and Evolution*, 10(5):1073–1095.

Saitou, N. and Nei, M. (1987). The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425.

Salakhutdinov, R. and Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM.

Saluja, A., Dyer, C., and Cohen, S. B. (2014). Latent-variable synchronous cfgs for hierarchical translation. In *Proceedings of EMNLP*.

Saul, L. and Pereira, F. (1997). Aggregate and mixed-order markov models for statistical language processing. In *Proceedings of the second conference on empirical methods in natural language processing*, pages 81–89. Somerset, New Jersey: Association for Computational Linguistics.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of ACL*.

Semple, C. and Steel, M. A. (2003). *Phylogenetics*, volume 24. Oxford University Press.

Shalizi, C. R. (2015). *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press, in progress, *http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/*.

Siddiqi, S., Boots, B., and Gordon, G. J. (2010). Reduced-rank hidden Markov models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010)*.

Smith, N. A. and Eisner, J. (2005). Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*, pages 354–362.

Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A hilbert space embedding for distributions. In *Algorithmic Learning Theory*, pages 13–31. Springer.

Song, L., Boots, B., Siddiqi, S., Gordon, G., and Smola, A. (2010a). Hilbert space embeddings of hidden Markov models. In *Proceedings of the 27th International Conference on Machine Learning*, pages 991–998. ACM.

Song, L., Boots, B., Siddiqi, S. M., Gordon, G. J., and Smola, A. J. (2010b). Hilbert space embeddings of hidden markov models. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 991–998.

Song, L., Gretton, A., Bickson, D., Low, Y., and Guestrin, C. (2011a). Kernel belief propagation. *arXiv preprint arXiv:1105.5592*.

Song, L., Gretton, A., and Guestrin, C. (2010c). Nonparametric tree graphical models via kernel embeddings. In *International Conference on Artificial Intelligence and Statistics*, pages 765–772.

Song, L., Huang, J., Smola, A., and Fukumizu, K. (2009). Hilbert space embeddings of conditional distributions with applications to dynamical systems. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 961–968. ACM.

Song, L., Parikh, A., and Xing, E. (2011b). Kernel embeddings of latent tree graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 2708–2716.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2010a). From baby steps to leapfrog: how less is more in unsupervised dependency parsing. In *Proceedings of NAACL*.

Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2013). Breaking out of local optima with count transforms and model recombination: A study in grammar induction. In *Proceedings of EMNLP*.

Spitkovsky, V. I., Alshawi, H., Jurafsky, D., and Manning, C. D. (2010b). Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*.

Stewart, G. and Sun, J. (1990). *Matrix perturbation theory*, volume 175. Academic press New York.

Stolcke, A. (2002). SRILM - An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference in Spoken Language Processing*.

Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4.

Subakan, C., Traa, J., and Smaragdis, P. (2014). Spectral learning of mixture of hidden markov models. In *Advances in Neural Information Processing Systems*, pages 2249–2257.

Teh, Y. W. (2006). A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311.

Turian, J. P., Ratinov, L.-A., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL*.

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013). Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA. Association for Computational Linguistics.

Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

Wang, Y. and Zhu, J. (2014). Spectral methods for supervised topic models. In *Advances in Neural Information Processing Systems*, pages 1511–1519.

Wood, F., Archambeau, C., Gasthaus, J., James, L., and Teh, Y. W. (2009). A stochastic memoizer for sequence data. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1129–1136. ACM.

Wood, F. and Teh, Y. W. (2009). A hierarchical nonparametric Bayesian approach to statistical language model domain adaptation. In *Artificial Intelligence and Statistics*, pages 607–614.

Wu, J. and Khudanpur, S. (2000). Efficient training methods for maximum entropy language modeling. In *Interspeech*, pages 114–118.

Xu, P., Gunawardana, A., and Khudanpur, S. (2011). Efficient subsampling for training complex language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1128–1136, Stroudsburg, PA, USA. Association for Computational Linguistics.

Zhang, N. L. (2004). Hierarchical latent class models for cluster analysis. *The Journal of Machine Learning Research*, 5:697–723.

Zhang, Y., Chen, X., Zhou, D., and Jordan, M. I. (2014). Spectral methods meet em: A provably optimal algorithm for crowdsourcing. In *Advances in Neural Information Processing Systems*, pages 1260–1268.

Zhou, S., Lafferty, J., and Wasserman, L. (2010). Time varying undirected graphs. *Machine Learning*, 80(2-3):295–319.

Zipf, G. (1949). Human behaviour and the principle of least-effort. Addison-Wesley, Cambridge, MA.

ML

MACHINE LEARNING
DEPARTMENT

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
www.ml.cmu.edu