Spectral Probabilistic Modeling and Applications to Natural Language Processing

Doctoral Thesis Proposal

Ankur Parikh
Machine Learning Department
Carnegie Mellon University

Thesis Committee:

Eric Xing (chair)
Geoff Gordon
Noah Smith
Le Song (GeorgiaTech)
Ben Taskar (University of Washington)

September 24, 2013

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 modeling power, the key algorithmic problems of learning and inference for these models can become substantially more complicated. Most existing solutions formulate the task 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, parameter learning, and inference using tools such as matrix/tensor decompositions, inversion, and ergodicity coefficients. These new tools enable us to develop novel solutions to learning and inference in latent variable models with theoretical and practical advantages. 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 for language modeling and unsupervised parsing.

Chapter 1

Introduction

Probabilistic graphical models have become an indispensable framework in artificial intelligence [Murphy, 2012, Koller and Friedman, 2009]. 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 [Koller and Friedman, 2009, Wainwright and Jordan, 2008].

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.

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.

Unfortunately, while latent variables provide advantages for model design, they pose substantial challenges for learning and inference. Structure learning, parameter learning, and some types of inference

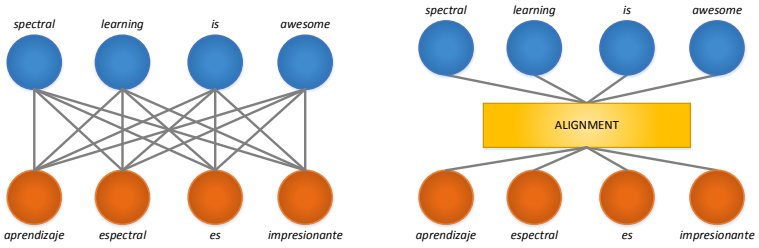


Figure 1.1: Translation example of how latent variables can provide simpler solutions to problems

that are easy for fully observed models are substantially more difficult for latent variable models. For example, maximum likelihood estimation with latent variables for structure/parameter learning is generally NP-hard. In some cases, inference may remain tractable, but there exist problems such as *marginal MAP* [Liu and Ihler, 2013, Jiang et al., 2011] that are NP-hard even when an efficient solution exists for the analogous problem with only observed variables.

As a result, the vast majority of 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.

1.1 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., 2010].

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, ergodicity coefficients, 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, inference, and modeling with latent variables from the linear algebra point of view.

1.2 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 language modeling and unsupervised

parsing, two probabilistic modeling problems which we believe can benefit substantially from the linear algebra perspective.

1.2.1 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 $\mathbb{P}(w_1, \ldots, w_m)$ is first factored and then approximated (with the Markov assumption) as:

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

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 propose to 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. We hope that our approach can take incorporate semantic relatedness among words to improve performance while at the same time be easily scalable to large datasets.

1.2.2 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., 2010, 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 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 learning, parameter learning, and inference using tools such as matrix/tensor decompositions, inversion, and ergodicity coefficients. 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 new insights aid us in designing new models and algorithms for language modeling and unsupervised parsing in Natural Language Processing.

Key Contributions

1. **Chapter 3: Spectral Parameter Learning for Latent Graphical Models** - We leverage tensor algebra to derive provably correct learning methods for latent trees and low-treewidth graphical models via junction trees (*Key theme: parameter learning*)
2. **Chapter 4: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*)
3. **Chapter 5: Spectral Approximations for Inference** - We plan to develop a theoretical framework studying how the spectral properties of the parameters in a latent model determine the dependencies among the observed variables. By leveraging this parameter-specific dependence we hope to design more efficient solutions for inference problems such as parallel collapsed sampling and marginal MAP (*Key theme: inference*)
4. **Chapter 6: A Conditional Latent Tree Model for Unsupervised Parsing** - 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. We plan to theoretically quantify the sample complexity of our approach as well as provide rigorous empirical evaluation (*Key themes: structure learning and modeling*)
5. **Chapter 7: Language Modeling via Power Low Rank Ensembles** - We propose to 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 proposed approach would allow n-grams of non-integer n i.e. 2.5-grams, 1.5-grams etc. that can take advantage of the semantic relatedness among words (*Key theme: modeling*)

1.4 Related Work

As mentioned previously, the basis for this dissertation comes from the work in 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., 2010].

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 approached spectral learning methods for other problems such as predictive state representations [Boots et al., 2010], topic models [Anandkumar et al., 2012,

Anandkumar et al., 2013], latent probabilistic context free grammars [Cohen et al., 2012, Cohen and Collins, 2012] and other models [Luque et al., 2012, Bailly et al., 2010, Balle et al., 2011, Dhillon et al., 2012].

In particular, the works of [Cohen et al., 2012, Cohen and Collins, 2012, Dhillon et al., 2012] leverage the tensor formulation we developed in this thesis.

1.5 Outline

Before describing the primary contributions of this thesis in more detail, we first present a linear algebra perspective on latent variable models in Chapter 2 which serves as a foundation for our work. Chapters 3, 4, 5, 6, and 7 then present the main proposed contributions of the thesis. Chapter 8 presents a tentative timeline.

Chapter 2

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.

2.1 General Notation

First we provide some notation conventions.

Probability: We denote random variables by capital letters e.g. X, Y, Z, and their instantiations by lower case letters x, y, z (scalars are also denoted with lower case letters). $\mathbb{P}(x)$ denotes the probability that $X = x$, and $p(X)$ represents the probability mass function / probability density function associated with X. In general, empirical estimates will be denoted as $\widehat{\mathbb{P}}(x), \widehat{p}(X)$ etc. In general we will assume that all our random variables are discrete unless specified otherwise.

Tensors: An Nth order tensor is a multiway array with N "modes", *i.e.*, N indices $\{i_1, i_2, \ldots, i_N\}$ 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, $\boldsymbol{\mathcal{A}}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N)$ are all elements in the nth mode of a tensor $\boldsymbol{\mathcal{A}}$ with indices from the other $N-1$ modes fixed to $\{i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_N\}$ respectively. Furthermore, we also use the shorthand $\boldsymbol{i}_{p:q} = \{i_p, i_{p+1}, \ldots, i_{q-1}, i_q\}$ for consecutive indices, *e.g.*, $\boldsymbol{\mathcal{A}}(i_1, \ldots, i_{n-1}, :, i_{n+1}, \ldots, i_N) = \boldsymbol{\mathcal{A}}(\boldsymbol{i}_{1:n-1}, :, \boldsymbol{i}_{n+1:N})$.

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. $\boldsymbol{\mathcal{P}}(X)_i = \mathbb{P}(X = i)$. Similarly $\boldsymbol{\mathcal{P}}(X, Y|Z)$ encodes the conditional probability tensor of X, Y given Z i.e. $\boldsymbol{\mathcal{P}}(X, Y|Z)_{i,j,k} = \mathbb{P}(X = i, Y = j|Z = k)$.

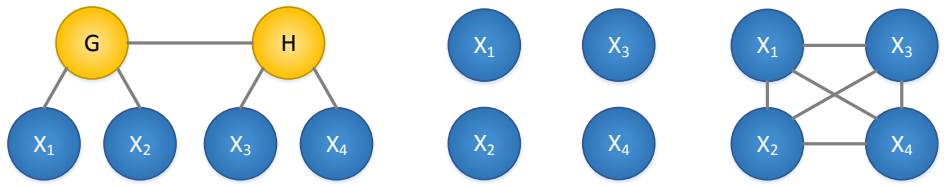
Mode-specific diagonal matrices/tensors. We use $\boldsymbol{\delta}$ to denote an N-way relation: its entry $\boldsymbol{\delta}(\boldsymbol{i}_{1:N})$ at position $\boldsymbol{i}_{1:N}$ equals 1 when all indexes are the same ($i_1 = i_2 = \ldots = i_N$), and 0 otherwise. We will use $\oslash_d$ to 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 $\mathbb{P}(Y) = \sum_X \mathbb{P}(Y|X)\mathbb{P}(X)$.

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 $\mathbb{P}(Y, X) = \mathbb{P}(Y|X)\mathbb{P}(X)$.

Symbol	*Definition*
upper case letters (e.g. X, Y, Z)	random variables
lower case letters (e.g. x, y, z)	constants or instantiations of random variables
$\mathbb{P}(\cdot)$	probability
$p(\cdot)$	probability mass/density function
$\boldsymbol{\mathcal{P}}(X_1, ..., X_n)$	probability vector/matrix/tensor
$\boldsymbol{\mathcal{P}}(\{X_1, X_2, X_4\}, \{X_3, X_5\})$	matricization of probability tensor
$\oslash_i$	diagonal on i modes

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



(a) (b) (c)

Figure 2.2: Different ways of modeling 4 observed variables X_1, X_2, X_3, X_4. G and H are latent variables.

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 $\boldsymbol{\mathcal{P}}(\{X_1\}, \{X_2, X_3, X_4\})$.

Further notation/operations such as tensor multiplication and tensor inversion will be introduced as needed. Figure 2.1 gives provides a quick reference for the notation conventions.

2.2 The Spectral View

Consider Figure 2.2(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 $S_H = 1$. In this case, X_1, X_2, X_3, and X_4 are independent as shown in Figure 2.2(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 2.2(c) which has $O(S_O^4)$ parameters. In general, if we had n observed variables all connected to one latent variable, then setting $S_H = S_O^n$ would be equivalent to n-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 $1 < S_H < m^3$. From the optimization point of view these values of S_H lead to nonconvex objectives and therefore are difficult. However, how much harder is $S_H = m^3 - 1$ 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 2.2(c) there is in general nothing we can say is special about the matrix $\boldsymbol{\mathcal{P}}(X_1, X_2)$. However, in the case of Figure 2.2(b), $\boldsymbol{\mathcal{P}}(X_1, X_2) = \boldsymbol{\mathcal{P}}(X_1)\boldsymbol{\mathcal{P}}(X_2)^\top$. Therefore $\boldsymbol{\mathcal{P}}(X_1, X_2)$ is rank one.

More generally, for Figure 2.2(a), $\boldsymbol{\mathcal{P}}(X_1, X_2) = \boldsymbol{\mathcal{P}}(X_1|G)\boldsymbol{\mathcal{P}}(\oslash_2 G)\boldsymbol{\mathcal{P}}(X_2|G)^\top$. Assuming all the matrices on the right hand side are full rank, this implies that $\boldsymbol{\mathcal{P}}(X_1, X_2)$ has rank S_H, implying that our factorization is a *low rank* factorization.

Lets now extend this logic to all 4 variables in Figure 2.2(a). Let $\boldsymbol{\mathcal{P}}(\{X_1, X_2\}, \{X_3, X_4\})$ 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:

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

Moreover, different matricizations lead to different factorizations i.e.

$$\boldsymbol{\mathcal{P}}(\{X_1\}, \{X_2, X_3, X_4\}) = \boldsymbol{\mathcal{P}}(X_1|G)\boldsymbol{\mathcal{P}}(\oslash_2 G)\boldsymbol{\mathcal{P}}(\{X_2, X_3, X_4\}|G)^\top$$
$$\boldsymbol{\mathcal{P}}(\{X_1, X_2, X_3\}, \{X_4\}) = \boldsymbol{\mathcal{P}}(\{X_1, X_2, X_3|H)\boldsymbol{\mathcal{P}}(\oslash_2 H)\boldsymbol{\mathcal{P}}(X_4|H)^\top$$

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 $\boldsymbol{M} = \boldsymbol{LR}$, $\boldsymbol{M} = \boldsymbol{LS}^{-1}\boldsymbol{SR}$ 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 $\boldsymbol{\mathcal{P}}(\{X_1, X_2\}, \{X_3, X_4\})$ is low rank but $\boldsymbol{\mathcal{P}}(\{X_1, X_3\}, \{X_2, X_4\})$ 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].

- **Inference**: Intuitively we see that X_1 and X_2 are closer together (separated by only one latent variable) than X_1 and X_3, which are separated by two latent variables. Does this mean that X_1 and X_2 are more dependent than X_1 and X_3? In the worst case, no, since G could be a deterministic function of H. However, if the relationship between G and H is non-deterministic, than we can prove that X_1 and X_3 are less dependent than X_1 and X_2 using the concept of the *coefficient of ergodicity*. Can this parameter-specific dependence be used to design more efficient inference algorithms that not only use the structure of the graphical model but also leverage the low rank structure?

- **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.?

Chapter 3

Spectral Parameter Learning for Latent Graphical Models

Using the insights from the previous chapter, we first tackle parameter learning in latent tree graphical models. In particular, we don't focus on learning the original parameters explicitly, but rather an alternate parameterization that can still be used for inference among the observed variables. This is useful in many applications such as NLP and structured prediction where the goal is not to recover the latent variables explicitly but rather to use the model for prediction.

First, we show that for latent trees, the original conditional probability table (CPT) parameterization can be viewed as a tensor factorization of the joint distribution of the observed variables. By noting that this factorization is not unique, we then derive another factorization that is only a function of the observed variables (called the observable factorization). This enables a provably consistent and local-optima-free learning algorithm with sample complexity analysis.

Furthermore, by leveraging higher order tensor operations such as multi-mode multiplication and tensor inversion we develop a spectral algorithm for latent junction trees to handle low treewidth loopy models, such as higher order HMMs, and coupled HMMs.

Our work generalizes the spectral algorithms of [Hsu et al., 2009, Bailly et al., 2009] for HMMs to more complex graphical models. Later works such as [Cohen et al., 2012] and [Dhillon et al., 2012] have leveraged our tensor formulation for supervised parsing with latent variables.

3.1 Intuition

Recall that in Chapter 2 we had established that latent variables introduce low rank factorizations, i.e. $\boldsymbol{M} = \boldsymbol{LR}$, over the marginal probability of the observed variables. For the example in Figure 2, we can set,

$$\boldsymbol{M} := \boldsymbol{\mathcal{P}}(\{X_1, X_2\}, \{X_3, X_4\}) \tag{3.1}$$

$$\boldsymbol{L} := \boldsymbol{\mathcal{P}}(\{X_1, X_2\}|H)\boldsymbol{\mathcal{P}}(\oslash_2 H) = \boldsymbol{\mathcal{P}}(\{X_1, X_2\}, H) \tag{3.2}$$

$$\boldsymbol{R} := \boldsymbol{\mathcal{P}}(\{X_3, X_4\}|H) \tag{3.3}$$

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

$$\boldsymbol{M} = \underbrace{\boldsymbol{LS}}_{\widetilde{\boldsymbol{L}}} \underbrace{\boldsymbol{S}^{-1}\boldsymbol{R}}_{\widetilde{\boldsymbol{R}}} \tag{3.4}$$

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).

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:

$$\boldsymbol{\mathcal{P}}(\{X_1, X_2\}, X_3) = \boldsymbol{\mathcal{P}}(\{X_1, X_2\}|H)\boldsymbol{\mathcal{P}}(\oslash_2 H)\boldsymbol{\mathcal{P}}(X_3|H)^\top \tag{3.5}$$

$$\boldsymbol{\mathcal{P}}(X_2, \{X_3, X_4\}) = \boldsymbol{\mathcal{P}}(X_2|H)\boldsymbol{\mathcal{P}}(\oslash_2 H)\boldsymbol{\mathcal{P}}(\{X_3, X_4\}|H)^\top \tag{3.6}$$

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

$$\boldsymbol{\mathcal{P}}(\{X_1, X_2\}, \{X_3, X_4\}) = \boldsymbol{\mathcal{P}}(\{X_1, X_2\}, X_3)\boldsymbol{\mathcal{P}}(X_2, X_3)^{-1}\boldsymbol{\mathcal{P}}(X_2, \{X_3, X_4\}) \tag{3.7}$$

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 $\boldsymbol{S} = \boldsymbol{\mathcal{P}}(X_3|H)$. Then we can prove that

$$\boldsymbol{L}\boldsymbol{S} = \boldsymbol{\mathcal{P}}(\{X_1, X_2\}, X_3) \tag{3.8}$$

$$\boldsymbol{S}^{-1}\boldsymbol{R} = \boldsymbol{\mathcal{P}}(X_2, X_3)^{-1}\boldsymbol{\mathcal{P}}(X_2, \{X_3, X_4\}) \tag{3.9}$$

3.2 A Spectral Algorithm for Latent Tree Graphical Models

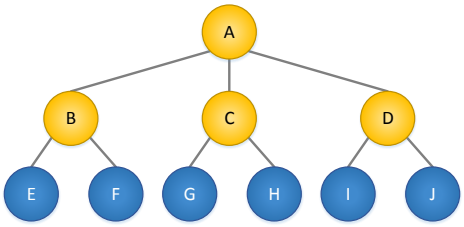


Figure 3.1: Example of a latent tree model with six observed nodes

We now generalize this intuition to 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 (observed or latent) 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 $\mathbb{P}[X_i|X_{\pi_i}]$ are sufficient to characterize the joint distribution (the root node X_r has no parent, *i.e.*, $\mathbb{P}[X_r|X_{\pi_r}] = \mathbb{P}[X_r]$),

$$\mathbb{P}[x_1, \ldots, x_{O+H}] = \prod_{i=1}^{O+H} \mathbb{P}[x_i|x_{\pi_i}]. \tag{3.10}$$

Compared to tree models which are defined solely on observed variables (*e.g.*, models obtained from the [Chow and Liu, 1968] 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:

$$\mathbb{P}[x_1, \ldots, x_O] = \sum_{x_{O+1}} \cdots \sum_{x_{O+H}} \prod_{i=1}^{O+H} \mathbb{P}[x_i | x_{\pi_i}]. \tag{3.11}$$

For simplicity, assume that all leaves are observed variables and all internal nodes are latent. For further notation, define $T(X_i)$ be the set of all observed leaves in the subtree rooted at X_i and let X_{i^*} be some leaf in $T(X_i)$. Let $c_j(i)$ denote the j^{th} child of node X_i.

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 third order tensors.
2. Inserting the invertible transformations $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ to define an alternate low rank factorization
3. Setting $\boldsymbol{F}$ such that the factors in this alternate factorization only depend on observed variables.

For simplicity of exposition, we assume all internal nodes have exactly three neighbors. However, we have developed an alternative tensor representation that demonstrates that only third order tensors are required regardless of the number of neighbors[Parikh et al., 2011].

3.2.1 Further Tensor Notation

We first introduce a bit more notation.

Labeling tensor modes with variables. In contrast to the conventional tensor notation such as the one described in [Kolda and Bader, 2009], the ordering of the modes of a tensor will not be essential. We will use random variables to *label* the modes of a tensor: each mode will correspond to a random variable and what is important is to keep track of this correspondence. Therefore, we think two 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

$$\boldsymbol{A} \cong \boldsymbol{A}^\top \tag{3.12}$$

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 tensors together, we will always carry out the operation along (a set of) modes with matching labels.

Tensor multiplication with mode labels. 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

$$\boldsymbol{C} = \boldsymbol{\mathcal{A}} \times_X \boldsymbol{\mathcal{B}} \quad \in \quad \mathbb{R}^{I_2 \times \cdots \times I_N \times J_2 \times \cdots \times J_M}, \tag{3.13}$$

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

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

3.2.2 Factorizing the Marginal Probability Tensor

Consider Figure 3.1. First let us exploit the low rank structure implied by the root A. We can get the following factorizations:

$$
\begin{aligned}
\boldsymbol{\mathcal{P}}(A,B,C,D,E,F,G,H,I,J) &= \boldsymbol{\mathcal{P}}(B,E,F|A)\boldsymbol{\mathcal{P}}(\oslash_2 A)\boldsymbol{\mathcal{P}}(C,D,G,H,I,J|A)^\top \\
\boldsymbol{\mathcal{P}}(A,B,C,D,E,F,G,H,I,J) &= \boldsymbol{\mathcal{P}}(B,C,E,F,G,H|A)\boldsymbol{\mathcal{P}}(\oslash_2 A)\boldsymbol{\mathcal{P}}(D,I,J|A)^\top \\
\boldsymbol{\mathcal{P}}(A,B,C,D,E,F,G,H,I,J) &= \boldsymbol{\mathcal{P}}(B,D,E,F,I,J|A)\boldsymbol{\mathcal{P}}(\oslash_2 A)\boldsymbol{\mathcal{P}}(C,G,H|A)^\top
\end{aligned}
$$

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

$$
\boldsymbol{\mathcal{P}}(A,B,C,D,E,F,G,H,I,J) = \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{\mathcal{P}}(B,E,F|A) \times_A \boldsymbol{\mathcal{P}}(C,G,H|A) \times_A \boldsymbol{\mathcal{P}}(D,I,J|A) \tag{3.14}
$$

We then proceed to factorize recursively e.g. $\boldsymbol{\mathcal{P}}(B,E,F|A) = \boldsymbol{\mathcal{P}}(\oslash_2 B|A) \times_B \boldsymbol{\mathcal{P}}(E|B) \times_B \boldsymbol{\mathcal{P}}(F|B)$.
In general, the joint probability of all the observed leaves in subtree rooted at X_i can be expressed as:

- If X_i is the *root*: $\boldsymbol{\mathcal{P}}(T(X_i)) = \boldsymbol{\mathcal{P}}(\oslash_3 X_i) \times_{X_i} \boldsymbol{\mathcal{P}}(T(X_{c_1(i)})|X_i) \times_{X_i} \boldsymbol{\mathcal{P}}(T(X_{c_2(i)})|X_i) \times_{X_i} \boldsymbol{\mathcal{P}}(T(X_{c_3(i)})|X_i)$
- If X_i is an *internal node*: $\boldsymbol{\mathcal{P}}(T(X_i)|X_{\pi(i)}) = \boldsymbol{\mathcal{P}}(\oslash_2 X_i|X_{\pi(i)}) \times_{X_i} \boldsymbol{\mathcal{P}}(T(X_{c_1(i)})|X_i) \times_{X_i} \boldsymbol{\mathcal{P}}(T(X_{c_2(i)})|X_i)$
- If X_i is a *leaf*: $\boldsymbol{\mathcal{P}}(T(X_i)|X_{\pi(i)}) = \boldsymbol{\mathcal{P}}(X_i|X_{\pi(i)})$

As one can see the tensor factorization we have derived is essentially the original CPTs embedded into higher-order tensors.

3.2.3 Transformed Representation

To derive our alternate factorization, we now insert invertible transformations $\boldsymbol{F}$. Continuing the running example in Figure 3.1,

$$
\begin{aligned}
\boldsymbol{\mathcal{P}}(A,B,C,D,E,F,G,H,I,J) &= \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{\mathcal{P}}(B,E,F|A) \times_A \boldsymbol{\mathcal{P}}(C,G,H|A) \times_A \boldsymbol{\mathcal{P}}(C,I,J|A) \\
&= \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{I} \times_A \boldsymbol{\mathcal{P}}(B,E,F|A) \times_A \boldsymbol{I} \times_A \boldsymbol{\mathcal{P}}(C,G,H|A) \times_A \boldsymbol{I} \times_A \boldsymbol{\mathcal{P}}(C,I,J|A)
\end{aligned}
$$

where $\boldsymbol{I}$ is the $S_H \times S_H$ identity matrix with mode labels $\{A, A\}$.
Next, expand $\boldsymbol{I}$ as a matrix inversion pair $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ and regroup the terms:

$$
\begin{aligned}
\boldsymbol{\mathcal{P}}(A,B,C,D,E,F,G,H,I,J) &= (\boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{F}_{c_1(A)} \times_A \boldsymbol{F}_{c_2(A)} \times_A \boldsymbol{F}_{c_3(A)}) \\
&\quad \times_{\omega_{c_1(A)}} (\boldsymbol{\mathcal{P}}(B,E,F|A) \times_A \boldsymbol{F}^{-1}_{c_1(A)}) \\
&\quad \times_{\omega_{c_2(A)}} (\boldsymbol{\mathcal{P}}(C,G,H|A) \times_A \boldsymbol{F}^{-1}_{c_2(A)}) \\
&\quad \times_{\omega_{c_3(A)}} (\boldsymbol{\mathcal{P}}(D,I,J|A) \times_A \boldsymbol{F}^{-1}_{c_3(A)})
\end{aligned}
$$

where $\omega_{c_1(A)}$, $\omega_{c_2(A)}$, $\omega_{c_3(A)}$ are mode labels to be defined in the next section.
This proceeds recursively e.g.

$$
\boldsymbol{\mathcal{P}}(D,I,J|A) = (\boldsymbol{\mathcal{P}}(\oslash_2 D|A) \times_D \boldsymbol{F}_{c_1(D)} \times_D \boldsymbol{F}_{c_2(D)}) \times_{\omega_{c_1(D)}} (\boldsymbol{\mathcal{P}}(I|D) \times_D \boldsymbol{F}^{-1}_{c_1(D)}) \times_{\omega_{c_2(D)}} (\boldsymbol{\mathcal{P}}(J|D) \times_D \boldsymbol{F}^{-1}_{c_2(D)})
$$

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

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

3.2.4 Observable Representation

All that remains is to set the $\boldsymbol{F}_i$ so that the alternate factorization is only a function of observed variables. Generalizing our intuition from Section 3.1, we set $\boldsymbol{F}_i = \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})$ (recall that X_{i^*} is an observed leaf in the subtree rooted at X_i). Below is the derivation for internal nodes (the root and leaf are just special cases). First note that,

$$\begin{aligned}
\widetilde{\boldsymbol{\mathcal{T}}}_i &= \boldsymbol{\mathcal{P}}(\oslash_2 X_i|X_{\pi(i)}) \times_{X_{\pi(i)}} \boldsymbol{F}_i^{-1} \times_{X_i} \boldsymbol{F}_{c_1(i)} \times_{X_i} \boldsymbol{F}_{c_2(i)} \\
&= \boldsymbol{\mathcal{P}}(\oslash_2 X_i|X_{\pi(i)}) \times_{X_{\pi(i)}} \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})^{-1} \times_{X_i} \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}|X_i) \times_{X_i} \boldsymbol{\mathcal{P}}(X_{c_2(i)^*}|X_i) \\
&= \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*}|X_{\pi(i)}) \times_{X_{\pi(i)}} \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})^{-1} \qquad (3.15)
\end{aligned}$$

We now prove the following relation:

$$\widetilde{\boldsymbol{\mathcal{T}}}_i \times_{X_{i^*}} \boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*}) = \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \qquad (3.16)$$

where X_{-i^*} is an observed leaf that is not a descendant of X_i. Note that $\boldsymbol{\mathcal{P}}(X_{i^*}, X_{-i^*}) = \boldsymbol{\mathcal{P}}(X_{i^*}|X_{\pi(i)})\boldsymbol{\mathcal{P}}(\oslash X_{\pi(i)})\boldsymbol{\mathcal{P}}(X_{-i^*}|X_{\pi(i)})^\top$.

Thus,

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

From here, one can conclude that

$$\widetilde{\boldsymbol{\mathcal{T}}}_i = \boldsymbol{\mathcal{P}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{X_{-i^*}} \boldsymbol{\mathcal{P}}(X_{-i^*}, X_{i^*})^{-1} \qquad (3.17)$$

Note that based on the above we can set the mode labels $\omega_i = X_{i^*}$.

3.2.5 What is $S_H \neq S_O$?

Based on our intuition in Chapter 2, 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 right singular vectors of $\boldsymbol{\mathcal{P}}(X_i^*, X_{-i^*})$. This gives Algorithm 1.

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. [Siddiqi et al., 2010, Cohen et al., 2012]

Algorithm 1 Spectral algorithm for latent tree graphical model

In: Junction tree topology and N *i.i.d.* samples $\left\{x_1^s, \ldots, x_{|O|}^s\right\}_{s=1}^N$

Out: Estimated observable root, internal, and leaf parameters, $\widehat{\boldsymbol{\mathcal{R}}}$, $\widehat{\boldsymbol{\mathcal{T}}}_i$, $\widehat{\boldsymbol{\mathcal{L}}}_i$,

1:

$$\begin{aligned}
\widehat{\boldsymbol{\mathcal{R}}} &= \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(r)^*}, X_{c_2(r)^*}, X_{c_3(r)^*}) \times_{X_{c_1(r)^*}} \widehat{\boldsymbol{U}}_{c_1(r)} \times_{X_{c_2(r)^*}} \widehat{\boldsymbol{U}}_{c_2(r)} \times_{X_{c_3(r)^*}} \widehat{\boldsymbol{U}}_{c_3(r)} \\
\widehat{\boldsymbol{\mathcal{T}}}_i &= \widehat{\boldsymbol{\mathcal{P}}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{X_{-i^*}} (\widehat{\boldsymbol{\mathcal{P}}}(X_{-i^*}, X_{i^*}) \times_{X_{i^*}} \widehat{\boldsymbol{U}}_i)^\dagger \times_{X_{c_1(i)^*}} \widehat{\boldsymbol{U}}_{c_1(i)} \times_{X_{c_2(i)^*}} \widehat{\boldsymbol{U}}_{c_2(i)} \\
\widehat{\boldsymbol{\mathcal{L}}}_i &= \widehat{\boldsymbol{\mathcal{P}}}(X_i, X_{-i^*}) \times_{X_{-i^*}} (\widehat{\boldsymbol{\mathcal{P}}}(X_{-i^*}, X_{i^*}) \times_{X_{i^*}} \widehat{\boldsymbol{U}}_i)^\dagger
\end{aligned}$$

where $\widehat{\boldsymbol{U}}_i$ is the top S_H right singular vectors of $\widehat{\boldsymbol{\mathcal{P}}}(X_{-i^*}, X_{i^*})$ and $\dagger$ indicates pseudoinverse.

proposed solutions to this problem by constructing features out of groups of observations to handle this scenario. In our case, this would mean constructing features out of $T(X_i)$ instead of using just one descendant X_i^*.

3.2.6 Inference in the Observable Representation

So far we have shown that $\boldsymbol{\mathcal{P}}(X_1, ..., X_O)$ can be factorized into smaller tensors. But what about if we given a set of evidence $(\bar{x}_1, ..., \bar{x}_O)$ and want to recover the probability $\mathbb{P}(\bar{x}_1, ..., \bar{x}_O)$?

Algebraically, this can be written as $\mathbb{P}(\bar{x}_1, ..., \bar{x}_O) = \boldsymbol{\mathcal{P}}(X_1, ...X_O) \times_{X_1} \boldsymbol{\delta}_{\bar{x}_1} ... \times_{X_O} \times \boldsymbol{\delta}_{\bar{x}_O}$ where $\boldsymbol{\delta}(\bar{x}_i)$ is an S_O-dimensional delta vector i.e. $\boldsymbol{\delta}(\bar{x}_i)_j = \mathbb{I}[j = \bar{x}_i]$.

Given the fact that tensor multiplication is associative, we don't need to compute the whole tensor $\boldsymbol{\mathcal{P}}(X_1, ..., X_O)$ before selecting the appropriate element. Rather it possible to proceed from the leaves up, and multiply by the delta vectors as we go. Note that this is effectively a message passing algorithm in tensor form where the tensor-matrix multiplication implements "sum-product".

3.2.7 Sample Complexity Analysis

We analyze the sample complexity of Algorithm 1 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

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

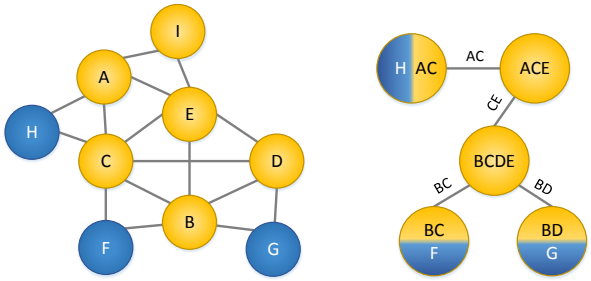
Then with probability $1 - \delta$,
$\sum_{x_1,...,x_O} \left|\widehat{\mathbb{P}}_{spectral}(x_1, \ldots, x_O) - \mathbb{P}(x_1, \ldots, x_O)\right| \leq \epsilon$, where $\widehat{\mathbb{P}}_{spectral}$ indicates the probability returned by the spectral algorithm.

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].

3.3 Spectral Learning of Non-Tree Models via Junction Trees

A natural question to ask is does this approach extend to non-tree models? 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.

In this thesis, we take one step toward this goal by developing a spectral algorithm for junction trees. The basis 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.



(a) (b)

Figure 3.2: Example of a low-treewidth graph and corresponding junction tree

The key difference over the latent tree case is that the separator sets in the junction tree can contain more than one variable, thus requiring higher order tensor operations such as multi-mode multiplication and tensor inversion. Consider multi-mode multiplication:

Multi-mode multiplication: Let $\sigma = \{X_1, \ldots, X_k\}$ 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 $\mathcal{A}$ and $\mathcal{B}$ along σ results in

$$\mathcal{D} = \mathcal{A} \times_\sigma \mathcal{B} \quad \in \quad \mathbb{R}^{I_{k+1} \times \ldots \times I_N \times J_{k+1} \times \ldots \times J_M}, \tag{3.18}$$

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 σ modes of $\mathcal{A}$ and $\mathcal{B}$ into a single mode and doing single-mode tensor multiplication.

We give a brief intuition of how our method works. A full derivation will be included in the thesis. Consider the example in Figure 3.2. The tensor factorization for the conditional probability tensor $\mathcal{P}(F, G|C, E)$ is:

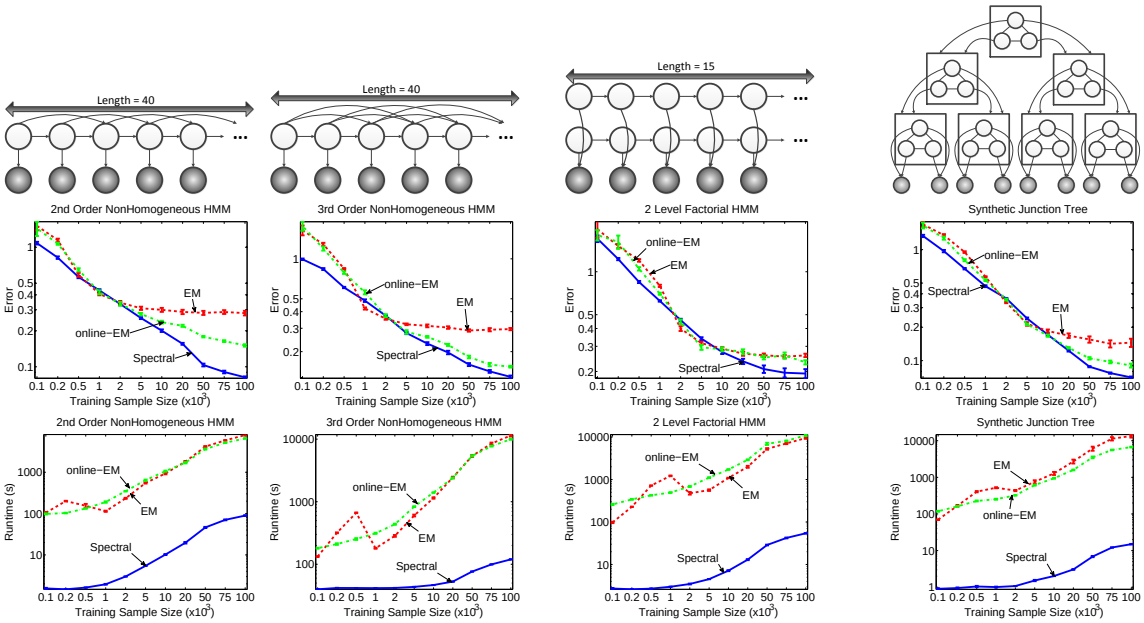
$$\mathcal{P}(F, G|C, E) = \mathcal{P}(\oslash_2 B, \oslash_2 C, D, E) \times_{BC} \mathcal{P}(F|B, C) \times_{BD} \mathcal{P}(G|B, D) \tag{3.19}$$

Here the conditional probability table for the clique $BCDE$ is embedded into the higher order tensor $\mathcal{P}(\oslash_2 B, \oslash_2 C, D, E)$ because B exists in both the child cliques while C exists in one child clique and one parent clique.

Instead of invertible transform matrices $\boldsymbol{F}$, we now require invertible transformation tensors $\mathcal{F}$ to derive the transformed representation e.g.:

$$\begin{aligned} \mathcal{P}(F, G|C, E) = \mathcal{P}(\oslash_2 B, \oslash_2 C, D, E) &\times_{BC} \mathcal{F}_1 \times_{BD} \mathcal{F}_2) \\ &\times_{\omega_1} (\mathcal{P}(F|B, C) \times_{BC} \mathcal{F}_1^{-1}) \\ &\times_{\omega_2} (\mathcal{P}(G|B, D) \times_{BD} \mathcal{F}_2^{-1}) \end{aligned} \tag{3.20}$$

where the inverse is a "mode-specific tensor inversion" that we propose. Deriving the observable representation can be done using a similar technique as before and the details will be included in the final thesis.



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

Figure 3.3: Comparison of our spectral algorithm (blue) to EM (red) and online EM (green) for various latent structures. Both errors and runtimes in log scale.

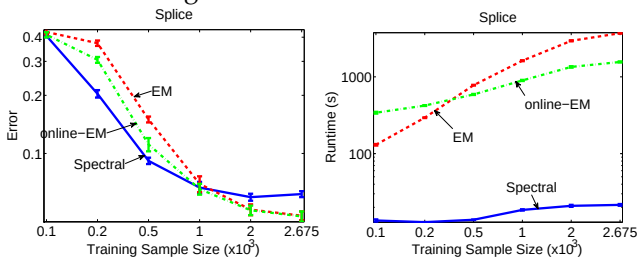


Figure 3.4: Results on Splice dataset

3.3.1 Empirical Results

For conciseness we give a summary of our results for junction trees. Results for the latent tree are similar and will be included in the final thesis. We 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 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.

We first perform a synthetic evaluation. 4 different latent structures are used (see Figure 3.3): 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 $error = \frac{|\widehat{\mathbb{P}}(x_1,\ldots,x_O) - \mathbb{P}(x_1,\ldots,x_O)|}{\mathbb{P}(x_1,\ldots,x_O)}$. We also measure the training time of both methods.

Figure 3.3 shows the results. 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. In terms of speed, 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.

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.

Figure 3.4, 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 the spectral algorithm is much faster for all the sample sizes.

3.4 Summary of Contributions

This work for this chapter is largely complete. The main contributions are:

- Spectral algorithm for latent trees - new algorithm, sample complexity analysis, and synthetic/real data results.
- Spectral algorithm for latent junction trees - new algorithm, sample complexity analysis and synthetic/real data results.

Chapter 4

Nonparametric Latent Trees with Kernel Embeddings

In the previous chapter we proposed solutions to parameter learning in latent tree graphical models where the variables are discrete. However, in many real world scenarios, variables take on continuous distributions. Moreover, these distributions may not easily be approximated by Gaussians due to skewness and multimodality. One example is demographics where variables like income, crime rate, and age rarely follow bell curves.

In many ways, this problem is much more challenging than the discrete case. Often, it is no longer possible to efficiently run non-convex optimization like EM, since the distributions no longer easily fit into a parametric family.

However, we will see that approaching the problem from a linear algebra point of view enables an elegant solution via Hilbert Space Embeddings [Smola et al., 2007, Song et al., 2009, Song et al., 2010]. In particular, we are able to generalize our method from Chapter 3 to the continuous variable case.

Below we give a very brief introduction to Hilbert Space Embeddings and then describe the intuition behind our algorithm.In addition to parameter learning, we also propose a structure learning algorithm for latent trees with continuous variables.

4.1 Hilbert Space Embeddings

The central motivation behind Hilbert Space Embeddings is to create a sufficient statistic μ_X for an arbitrary continuous distribution $p(X)$. Let us consider the following motivation.

One simple statistic would be just to use the mean i.e. $\mu_X = \mathbb{E}[X]$. The problem is that many different distributions map to the same mean. Therefore this statistic is not sufficient. A more complex statistic would be $\mu_X = (\mathbb{E}[X], \mathbb{E}[X^2])$ i.e. the first two moments. This is a sufficient statistic for Gaussians, since the mean and variance completely characterize a Gaussian distribution. However, for non-Gaussian distributions it is entirely possible that two different distributions may have the same mean and variance. Similarly, we could use third order moments $\mu_X = (\mathbb{E}[X], \mathbb{E}[X^2], \mathbb{E}[X^3])$ but the same problem exists.

The intuition behind Hilbert Space Embeddings is to create an infinite dimensional sufficient statistic. Moreover, the statistic has a special form: it is a function in a Reproducing Kernel Hilbert Space (RKHS). Therefore, although it is infinite dimensional, it can be computed using the "kernel trick". More formally, let us define an RKHS [Schölkopf and Smola, 2002]:

Definition 1. *A reproducing kernel Hilbert Space (RKHS) $\mathcal{F}$ on $\mathcal{X}$ with kernel k is a Hilbert Space of functions $f: \mathcal{X} \to \mathbb{R}$. Its dot product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ satisfies the reproducing property:*

$$\begin{aligned} &\langle f(\cdot), k(x, \cdot) \rangle_{\mathcal{F}} = f(x), \textit{and thus} \\ &\quad \langle k(x, \cdot), k(x', \cdot) \rangle_{\mathcal{F}} = k(x, x') \end{aligned} \tag{4.1}$$

We can view $k(x, \cdot)$ as a function of one variable which we denote as the feature map ϕ_x.

An RKHS is special in that the feature map ϕ_x is the evaluation functional i.e. it serves to evaluate any function $f \in \mathcal{F}$ at point x. Then each element of the kernel matrix $k(x, x')$ can be thought of as the evaluation of the feature map ϕ_x at x' which is equal to $\phi_{x'}$ at x. The intuition behind the kernel trick is that although the functions are continuous, they only need to be evaluated on the points in the dataset. Thus, the kernel matrix is all that is needed.

The Hilbert Space Embedding (mean map) of $p(X)$ is defined to be [Smola et al., 2007]:

$$\mu_X := \mathbb{E}_X[\phi_X] \tag{4.2}$$

It can be proven that if the kernel function k is *characteristic*, then the mean map is *injective* i.e. two different distributions do not have the same embedding. The Gaussian RBF and Laplace kernels are examples of *characteristic* kernels.

4.1.1 Covariance and Conditional Embedding Operators

In addition to simply embedding distributions of one variable, Hilbert Space Embeddings also allow you to embed distributions of multiple variables. Let $\otimes$ denote outer-product. We can define the following operators:

- $\boldsymbol{C}(X, Y) = \mathbb{E}[\phi_X \otimes \phi_Y]$ is the embedding of $P(X, Y)$. In general $\boldsymbol{C}(X_1, ..., X_n) = \mathbb{E}[\phi_{X_1} \otimes .....\phi_{X_n}]$ is the embedding of $P(X_1, ..., X_n)$.
- $\boldsymbol{C}(\oslash_2 X) = \mathbb{E}[\phi_X \otimes \phi_X]$ is the analog of $\boldsymbol{\mathcal{P}}(\oslash_2 X)$ for discrete variables.
- $\boldsymbol{C}(Y|X) = \boldsymbol{C}(Y, X)\boldsymbol{C}(\oslash_2 X)^{-1}$ is the embedding of the conditional distribution $P(Y|X)$

Interestingly, the sum/chain rule apply with RKHS operators too! [Song et al., 2009].

- **Sum Rule**: $\mu_Y = \boldsymbol{C}(Y|X)\mu_X$. This is the RKHS analog of the matrix sum rule: $\boldsymbol{\mathcal{P}}(Y) = \boldsymbol{\mathcal{P}}(Y|X)\boldsymbol{\mathcal{P}}(X)$.
- **Chain Rule**: $\boldsymbol{C}(Y, X) = \boldsymbol{C}(Y|X)\boldsymbol{C}(\oslash_2 X)$. This is the RKHS analog of the matrix chain rule: $\boldsymbol{\mathcal{P}}(Y, X) = \boldsymbol{\mathcal{P}}(Y|X)\boldsymbol{\mathcal{P}}(\oslash_2 X)$.

4.2 A Kernel Spectral Algorithm for Latent Tree Graphical Models

The similarity of the sum/chain rule above to matrix multiplication enable us to generalize our spectral algorithm from Chapter 3. We note that although the results look very similar, the derivation is more complicated due to the presence of infinite dimensional operators. Below we just give some intuition and state the final result (the notation has been chosen so that the analogy to the discrete case is clear). More details will be included in the full thesis.

Consider Figure 3.1 again. Recall that with discrete variables we had the following factorization at the root.

$$\boldsymbol{\mathcal{P}}(A, B, C, D, E, F, G, H, I, J) = \boldsymbol{\mathcal{P}}(\oslash_3 A) \times_A \boldsymbol{\mathcal{P}}(B, E, F|A) \times_A \boldsymbol{\mathcal{P}}(C, G, H|A) \times_A \boldsymbol{\mathcal{P}}(D, I, J|A) \tag{4.3}$$

the analog in the continuous case via Hilbert Space Embeddings will be:

$$\boldsymbol{C}(A, B, C, D, E, F, G, H, I, J) = \boldsymbol{C}(\oslash_3 A) \times_A \boldsymbol{C}(B, E, F|A) \times_A \boldsymbol{C}(C, G, H|A) \times_A \boldsymbol{C}(D, I, J|A) \tag{4.4}$$

Here we have informally used $\times_A$ which is a bit odd for an infinite dimensional operator (but it is possible to give a rigorous definition). Going through the derivation gives us the following observable parameterization:

$$\widehat{\mathcal{R}} = \widehat{\boldsymbol{C}}(X_{c_1(r)^*}, X_{c_2(r)^*}, X_{c_3(r)^*}) \times_{X_{c_1(r)^*}} \widehat{\boldsymbol{U}}_{c_1(r)} \times_{X_{c_2(r)^*}} \widehat{\boldsymbol{U}}_{c_2(r)} \times_{X_{c_3(r)^*}} \widehat{\boldsymbol{U}}_{c_3(r)}$$

$$\widehat{\mathcal{T}}_i = \widehat{\boldsymbol{C}}(X_{c_1(i)^*}, X_{c_2(i)^*}, X_{-i^*}) \times_{X_{-i^*}} (\widehat{\boldsymbol{C}}(X_{-i^*}, X_{i^*}) \times_{X_{i^*}} \widehat{\boldsymbol{U}}_i)^\dagger \times_{X_{c_1(i)}} \widehat{\boldsymbol{U}}_{c_1(i)} \times_{X_{c_2(i)}} \widehat{\boldsymbol{U}}_{c_2(i)}$$

$$\widehat{\mathcal{L}}_i = \widehat{\boldsymbol{C}}(X_{i^*}, X_{-i^*}) \times_{X_{-i^*}} (\widehat{\boldsymbol{C}}(X_{-i^*}, X_{i^*}) \times_{X_{i^*}} \widehat{\boldsymbol{U}}_i)^\dagger$$

where $\widehat{\boldsymbol{U}}_i$ is the top S_H right singular vectors of $\widehat{\boldsymbol{C}}(X_{-i^*}, X_{i^*})$ and † indicates pseudoinverse.

Empirically estimating these quantities is a bit tricky since they are infinite dimensional. However, it is possible using the "kernel trick" for SVMs (i.e. some of the operators can be grouped together so these new grouped terms are finite).

4.3 Structure Learning of Nonparametric Latent Trees

While many methods have developed for learning the structure of a latent tree where the variables are discrete or Gaussian [Saitou and Nei, 1987, Lake, 1994, Choi et al., 2011, Anandkumar et al., 2011, Ishteva et al., 2012], it is unclear how to learn the structure of a nonparametric latent tree from data. We propose a method for constructing nonparametric latent trees based on additive tree metrics that has provably guarantees on structure recovery.

The key concept behind many previous approaches for discrete/Gaussian variables is an additive tree metric: a distance function that maps pairs of nodes in the graphical model to a real-valued distance.

Definition 2. *A function $d_{\mathcal{T}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is an* additive tree metric *if $\forall X_i, X_j \in \mathcal{X}$ the following relation holds:*

$$d_{\mathcal{T}}(i, j) = \sum_{(a,b) \in path_{\mathcal{T}}(i,j)} d(a, b) \tag{4.5}$$

where $path_{\mathcal{T}}(i, j)$ is the set of all the edges in the path from X_i to X_j in the tree $\mathcal{T}$.

For example, for discrete latent trees where $S_H = S_O$ the following is an additive tree metric [Lake, 1994]:

$$d_{\mathcal{T}}^{discrete}(i, j) = -\tfrac{1}{2} \log \det(\boldsymbol{\mathcal{P}}(X_i, X_j)) + \tfrac{1}{4} \log \det(\boldsymbol{\mathcal{P}}(\oslash_2 X_i)) + \tfrac{1}{4} \log \det(\boldsymbol{\mathcal{P}}(\oslash_2 X_j)) \tag{4.6}$$

Many meta-algorithms, such as neighbor joining [Saitou and Nei, 1987] or the recursive grouping algorithm [Choi et al., 2011] take as input a distance matrix among all the observed variables and output a latent tree structure. Interestingly, although many of these approaches are greedy, they are provably consistent if the distance matrix satisfies the additive metric assumption.

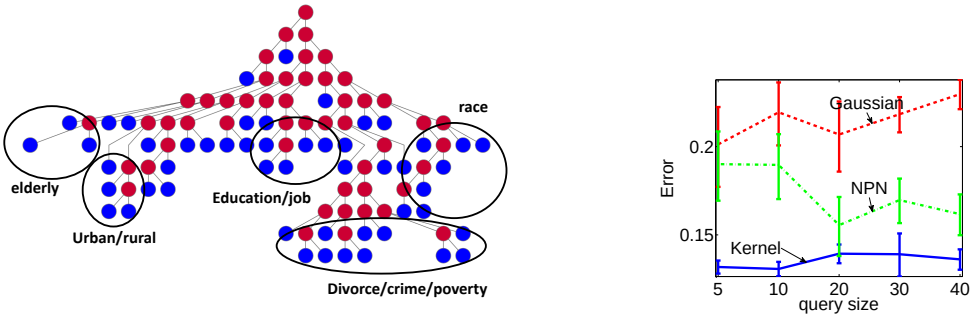
However this metric for discrete latent trees doesn't directly generalize to the infinite dimensional case, since the eigenvalues of the covariance operator decay to zero and thus $\det(\boldsymbol{C}(X, Y)) = 0$. Instead we use the notion of pseudo-determinant to propose the following additive tree metric for non-Gaussian, continuous variables:

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 two continuous non-Gaussian variables X_i and X_j as

$$d_{\mathcal{T}}^{kernel}(i, j) = -\tfrac{1}{2} \log \left| \boldsymbol{C}(X_i, X_j) \boldsymbol{C}(X_i, X_j)^\top \right|_\star + \tfrac{1}{4} \log |\boldsymbol{C}(\oslash_2 X_i) \boldsymbol{C}(\oslash_2 X_i)^\top|_\star + \tfrac{1}{4} \log |\boldsymbol{C}(\oslash_2 X_j) \boldsymbol{C}(\oslash_2 X_j)^\top|_\star. \tag{4.7}$$

4.4 Experiments

We give some results on the performance of our method on a communities and crime dataset from the UCI repository [Asuncion and Newman, 2007, Redmond and Baveja, 2002]. More experiments are contained in [Song et al., 2011] and will be provided for the final thesis. In this dataset, several real valued attributes are collected for several communities, such as ethnicity proportions, income, poverty rate, divorce rate etc.,



(a) (b)

Figure 4.1: Qualitative and quantitative results on crime and communities dataset

and the goal is to predict the number of violent crimes (proportional to the size of the 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 the neighbor joining algorithm [Saitou and Nei, 1987]. We depict the tree in Figure 4.1(a) 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 captures sensible relationships.

For a more quantitative evaluation, we condition on a set $\mathcal{E}$ of evidence variables and predict the violent crimes class label. We experiment with a varying number of sizes of evidence sets from 5 to 40 and repeat for 40 randomly chosen evidence sets of a fixed size. Since the crime variable is a number between 0 and 1, our error measure is simply $err(\hat{c}) = |\hat{c} - c^*|$ (where $\hat{c}$ is the predicted value and c^* is the true value. As one can see in Figure 4.1(b) our method outperforms both the Gaussian and the nonparanormal [Liu et al., 2009] for the range of query sizes. Thus, in this case our method is better able to capture the skewed distributions of the variables than the other methods.

4.5 Summary of Contributions

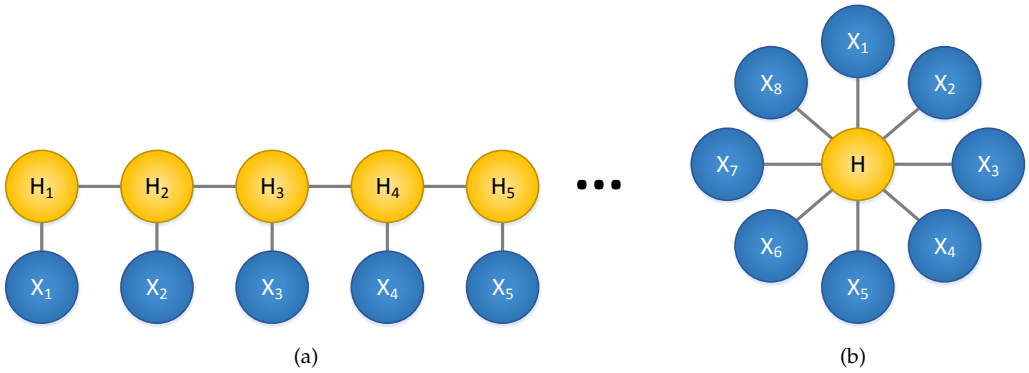
This work is mostly complete. The contributions are:

1. **Structure Learning for Nonparametric Latent Trees** - new algorithm, evaluation on synthetic and real data.

2. **Parameter Learning for Nonparametric Latent Trees** - new algorithm, evaluation on real data.

Chapter 5

Spectral Approximations for Inference



(a) (b)

Figure 5.1: (a) Example of an HMM (b) Example of 8 observed nodes connected by a latent variable. In both cases marginalizing out the latent variables creates a clique.

In this chapter, we examine inference in graphical models from the spectral perspective. Commonly, methods for inference in graphical models rely on the *structure* of the model to design and analyze inference algorithms [Koller and Friedman, 2009, Wainwright and Jordan, 2008]. For example, the difficulty of inference is primarily characterized by the *treewidth* of the model.

However, this paradigm can be quite limiting, especially in parallel settings. For example, collapsed samplers do not structurally have conditional independence statements. However, many recent works have shown that simply letting the sampler run "hogwild" (i.e. asynchronously) can often produce good results in some cases [Smyth et al., 2008].

In this thesis we explore how the spectral view of latent variable models can lead to a different theoretical perspective and new solutions for inference in graphical models. We give two examples below.

Marginal MAP:

A common task in graphical models is to compute the MPE (most probable explanation), assignment: given a set of evidence variables, to compute the most likely assignment of all the other variables (This is also commonly referred to as the MAP assignment). This problem has proved useful in a number of real world problems such as protein side chain prediction, and many methods have been developed to tackle it [Wainwright and Jordan, 2008, Sontag et al., 2012].

However, in many applications, we do not want the most likely assignment of *all* the other variables, but rather only a particular *subset* of them (called *marginal MAP*). This is especially the case in latent variable structured prediction models where the goal is to predict a structured output while marginalizing out the latent variables. One example is parsing in NLP, where latent variable models are used, but the end goal is to only to find the most likely parse for a given word (while integrating out the latent

variables) [Petrov et al., 2006, Matsuzaki et al., 2005, Smith, 2011].

Interestingly, Marginal MAP is substantially more difficult than MPE / inference problems and is difficult even for trees because the sum and max operators do not commute [Park, 2002, Koller and Friedman, 2009]. For example, consider the structure shown in Figure 5.1(a). Summing out the X variables and then finding the most likely assignment to the H variables is tractable (i.e. Viterbi algorithm). However, summing out the H variables while finding the most likely assignment of the X variables is generally intractable, since after summing out the H variables, the X nodes become connected in a clique.

Until recently, most methods for tackling marginal MAP were either simply running sum product, max product, or Expectation Maximization where the max/sum steps were alternatively performed. Recently Jiang et al. [Jiang et al., 2011] and Liu and Ihler [Liu and Ihler, 2013] proposed hybrid message passing schemes based on the general variational formalism [Wainwright and Jordan, 2008].

Collapsed Sampling: In many graphical models that require approximate inference such as topic/admixture models [Blei et al., 2003, Hoff et al., 2002], collapsed sampling [Griffiths and Steyvers, 2004, Porteous et al., 2008] has emerged as a popular sampling method in which some of the latent variables are marginalized out. For example in Figure 5.1(b), a traditional gibbs sampler would iteratively sample the values of $H, X_1, ..., X_8$. However, a collapsed sampler would marginalize out H and only sample the $X_1, ..., X_8$.

While collapsed sampling presents many advantages for mixing and Markov chain convergence, it presents challenges for parallelization. The uncollapsed sampler can be easily parallelized since conditioned on a value of H, the X_i are all conditionally independent and can easily be sampled in parallel. However, once H has been marginalized out, as in the collapsed sampler, the $X_1, .., X_8$ have no conditional independencies and thus parallelization becomes nontrivial.

5.1 Goals

Recall in Chapter 2 that our notion of a *complexity* of a model was not simply the structure but also the *rank* of the latent variables. For example, none of the X variables in Figure 5.1(b) are conditionally independent of one another once H is summed out. However, they only share a low rank dependence providing H does not take on too many states. In other words, the cardinality of the latent space and the *rank* of the related conditional probability tables determine the complexity of the model.

Our goal is to use this insight to derive new methods for inference that can leverage low rank structure. To help reach this goal, in this proposal we simply describe the following two key intuitions. For the first we provide a formal proof.

1. **Distance Intuition:** Variables farther apart in the graphical model are more approximately independent than those nearby.

2. **Clique Intuition:** Consider a set of observed variables $\mathcal{X}$ that are connected by one low rank variable e.g. Figure 2 5.1(b). Then $p(X_i|\mathcal{Z}) \approx p(X_i|\{\mathcal{X} \setminus X_i\})$ where $\mathcal{Z}$ is a "large enough" subset of $\{\mathcal{X} \setminus X_i\}$.

In some sense, our work is inspired by recent work in optimization such as parallel coordinate descent for the LASSO [Bradley et al., 2011] where approximate independence among covariates can be leveraged for parallelism even if the coordinate descent is technically a sequential algorithm.

5.2 Distance Intuition

To help formalize our distance intuition, consider the model in Figure 5.1(a). Recall from Chapter 2 that $\boldsymbol{\mathcal{P}}(H_n|H_1)$ is rank one if and only if $X_n \perp X_1$. We seek to show that as n increases (i.e. the nodes become farther apart), $\boldsymbol{\mathcal{P}}(H_n|H_1)$ becomes closer and closer to being rank one.

Note that

$$\boldsymbol{\mathcal{P}}(H_n|H_1) = \prod_{t=1}^{n-1} \boldsymbol{\mathcal{P}}(H_{t+1}|H_t) \tag{5.1}$$

First assume the HMM is homogeneous i.e. $\boldsymbol{\mathcal{P}}(X_{t+1}|X_t)$ equals a fixed stochastic matrix T $\forall t$. Let $\lambda_2(T)$ denote the second eigenvalue of T. Note that since T is stochastic the largest eigenvalue is equal to 1 and therefore $\lambda_2(T) \leq 1$.

Then, $\boldsymbol{\mathcal{P}}(H_n|H_1) = T^{n-1}$ and thus $\lambda_2(\boldsymbol{\mathcal{P}}(H_n|H_1)) = \lambda_2(T)^{n-1}$. Assuming that $\lambda_2(T) < 1$ (i.e. basically that T is not deterministic) this implies that $\lambda_2(\boldsymbol{\mathcal{P}}(H_n|H_1)) \to 0$ as $n \to \infty$.

However, when the HMM is not homogeneous then we cannot apply the argument above because $\lambda_2(\boldsymbol{\mathcal{P}}(H_n|H_1)) \neq \lambda_2(T)^{n-1}$. Thus, we need another measure of the rank-deficiency of a matrix that is multiplicative (or atleast submultiplicative) in the non-homogeneous case.

Seneta [Seneta, 1979] developed such a measure called the **coefficient of ergodicity**.[1]

Definition 3. *A **coefficient of ergodicity** is a scalar function $\tau(\cdot)$ continuous on the set of $(n \times n)$ column stochastic matrices and satisfying $0 \leq \tau(\boldsymbol{A}) \leq 1$. It is then said to be **proper coefficient of ergodicity** if*

$$\tau(\boldsymbol{A}) = 0 \iff \boldsymbol{A} = \boldsymbol{v}\mathbf{1}^\top \tag{5.2}$$

where $\boldsymbol{v}$ is any probability vector.

In particular the coefficient of ergodicity we will consider is [Seneta, 1979, Ipsen and Selee, 2011]

$$\tau_1(\boldsymbol{A}) = \max_{\|z\|_1=1, z^\top\mathbf{1}=0} \|\boldsymbol{A}z\|_1 \tag{5.3}$$

In particular, $\tau_1(\cdot)$ is submultiplicative i.e. $\tau_1(\boldsymbol{A}_1\boldsymbol{A}_2) \leq \tau_1(\boldsymbol{A}_1)\tau_2(\boldsymbol{A}_2)$.

Using this fact (and a few other basic properties) it is simple to prove the following lemma:

Lemma 1. *Assume that $\tau_1(\boldsymbol{\mathcal{P}}(H_{j+1}|H_j)) < \gamma \quad \forall j$. Then,*

$$\|\boldsymbol{\mathcal{P}}(H_n|H_1) - \boldsymbol{M}_{H_n}\|_1 \leq 2S_H^2\gamma^{n-1} \tag{5.4}$$

where H_n and H_1 both take on S_H states, $\boldsymbol{M}_{H_n}$ is the $S_H \times S_H$ matrix where every column contains the marginal probability vector of H_n, and $\|\cdot\|_1$ is the elementwise one-norm.

5.3 Clique Intuition

Formalizing this intuition is still a work in progress. Ideally, we would like to show that $p(X_i|\mathcal{Z}) \approx p(X_i|\{\mathcal{X} \setminus X_i\})$ where $\mathcal{Z}$ is a "large enough" subset of $\{\mathcal{X} \setminus X_i\}$. The challenge lies in the fact that we are now trying to prove bounds on the posterior distribution which is more challenging than the conditional.

5.4 Summary of Contributions

This project is mostly future work and much remains left to be done. The goal is to use the intuition explained above to design new inference techniques for problems such as marginal MAP and approximately parallel collapsed sampling.

[1] We present it using the formulation of [Ipsen and Selee, 2011]

Chapter 6

A Conditional Latent Tree Model for Unsupervised Parsing

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.

Instead of attempting to directly propose a spectral algorithm for learning an existing model, we propose a new approach to unsupervised parsing that revolves around structure learning (as opposed to existing approaches which center around parameter learning). Our goal is to develop a method has comes with strong theoretical guarantees on latent structure recovery and also works well empirically.

In essence, our approach reduces to learning the structure of a "conditional" latent tree graphical model, where the structure of the latent tree varies across each sentence. We propose to generalize existing spectral approaches for learning a fixed latent tree [Saitou and Nei, 1987, Lake, 1994, Choi et al., 2011, Anandkumar et al., 2011, Ishteva et al., 2012] that we briefly discussed in Chapter 4 to this varying structure scenario. In particular, we leverage kernel smoothing techniques from the statistics community [Zhou et al., 2010, Kolar et al., 2010b, Kolar et al., 2010a] to deal with the ensuing data sparsity problem.

6.1 Motivation

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 data required for supervised training, which

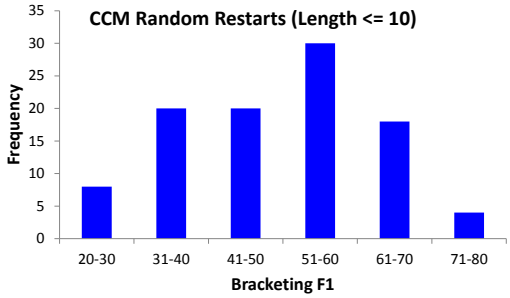


Figure 6.1: Histogram of performance over 100 random restarts of CCM[Klein and Manning, 2004], a common unsupervised constituent parsing algorithm, on WSJ10 [Marcus et al., 1993]. As one can see performance varies widely.

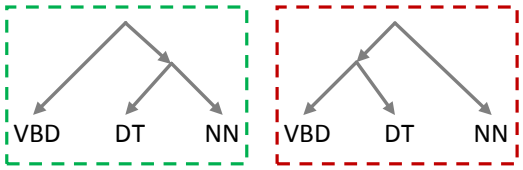


Figure 6.2: Candidate constituent parses for $\boldsymbol{x} = (VBD, DT, NN)$ (left-correct, right -incorrect)

requires manual syntactic annotation.

Most of the solutions suggested treat the problem of unsupervised parsing by assuming a parametric model. Learning the model then corresponds to optimizing a set of parameters θ that achieve 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., 2010, 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 (For example, see Figure 6.1).

Recently, spectral techniques have provided insight into unsupervised learning of probabilistic context free grammars [Hsu et al., 2012] showing that they are in general non-identifiable, requiring substantial restrictions to become identifiable. Even when identifiable, deriving an actual learning algorithm is challenging and unlikely to perform well on real world data.

In this thesis, we propose a model that allows for a learning algorithm with theoretical guarantees on latent structure recovery. Notably, we do not explicitly learn a grammar. Instead, we treat unsupervised parsing as a structure learning problem, where we recover a latent structure (an undirected latent tree) for each sentence. We then apply a *bias mapping* to direct this latent tree to generate the final syntactic structure (either a dependency or a constituent parse).

Our approach is inspired by methods for recovering latent tree structure in the machine learning and phylogeneitics community [Saitou and Nei, 1987, Lake, 1994, Choi et al., 2011, Anandkumar et al., 2011, Ishteva et al., 2012]. These approaches leverage the spectral properties of the relevant covariance matrices to consistently learn latent trees.

One key challenge in this approach is being able to robustly compute these covariance matrices since different sentences may be associated with different intermediate structures. To handle this issue, we present an "anchoring" strategy that is inspired by ideas from kernel smoothing in the Statistics community [Zhou et al., 2010, Kolar et al., 2010b, Kolar et al., 2010a]. This allows principled sharing of samples from different but similar underlying distributions.

In the following sections, we motivate the rationale behind our approach and then present the core proposed algorithm.

6.2 Unsupervised Parsing as a Structure Learning Problem

Let $\boldsymbol{W} = (W_1, ..., W_\ell)$ be a vector of words, where in general, each W_i is a real valued vector representing some embedding/feature vector of word i in the sentence. Let $\boldsymbol{X} = (X_1, ..., X_\ell)$ be the corresponding vector of part-of-speech (POS) tags (i.e. X_i is the POS tag of W_i)[1].

For intuition, consider the tag sequence $\boldsymbol{x} = (VBD, DT, NN)$.

Some candidate constituent parse structures are shown in Figure 6.2 and the correct one is circled in green (the other is boxed in red).

Of course without supervision, it would be impossible to determine the correct dependency structure without extra information. In our scenario, we are also given word phrases $\boldsymbol{W} = (W_1, W_2, W_3)$ that have this tag sequence e.g. (*hit the ball*, *ate an apple*, *baked the cake*).

One can see that based on these examples, the determiner is rather independent of the verb i.e. the value of W_2 is largely independent of W_1. However, W_2 is largely correlated with W_3 i.e. if W_2 = "*a*" then we know W_3 must be singular and start with a consonant.

[1] W_i and X_i are capitalized since they are random variables. Their instantiations will be lower-case.

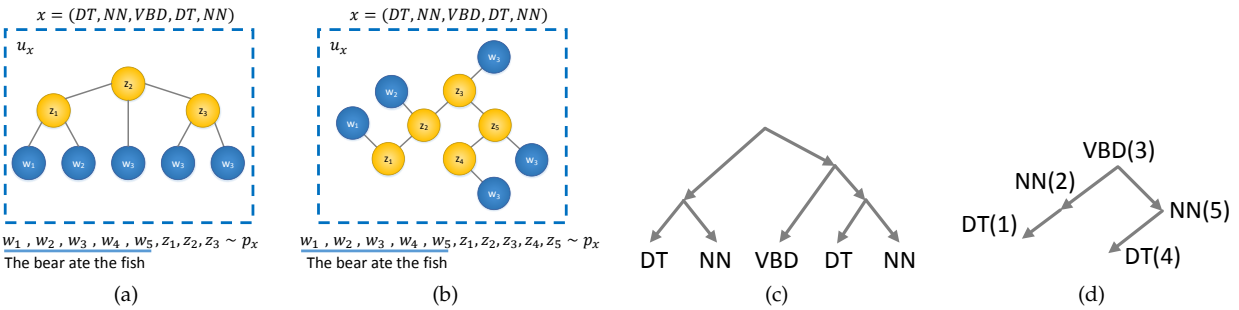


Figure 6.3: Example of undirected latent trees for (a) constituent and (b) dependency parsing. And (c) the actual constituent parse (d) the actual dependency parse. See text for details

Thus, we can deduce that W_2 is more associated with W_3 than W_1, giving us the correct structure.

Our contention is that the relationship among the lexical forms (conditioned on the tag sequences) exposes the underlying parse structure. Formalizing this intuition, we assume the conditional model:

$$\boldsymbol{W}|\boldsymbol{x} \sim P_{u_x,\theta_x} \tag{6.1}$$

Here, $u_x \in \mathcal{U}$ is the undirected graph structure that characterizes the conditional independencies among the W_i (conditioned on a tag sequence $\boldsymbol{x}$). θ_x is the corresponding set of parameters.

Note that $\mathcal{U}$ is not a set of parse trees, but rather a family of intermediate structures (described later). To obtain the final parse structure, we propose a *bias mapping* $h : \mathcal{U} \to \mathcal{T}$ that deterministically maps undirected graph structures to parse trees (h is assumed to be known since it represents domain knowledge). As a result, our parsing problem reduces to recovering the latent structure u_x for each $\boldsymbol{x}$.

The exact choice of $\mathcal{U}$ depends on whether we desire a dependency or a constituent parse. However, in all cases, we use a subset of the family of *latent tree graphical models* as described in Chapter 3. Briefly, latent trees are a tree structure over the observed variables ($\boldsymbol{W}$ in our case), as well as an additional set of latent variables that we denote as $\boldsymbol{Z}$. The addition of the latent variables allows for longer range dependencies compared to a fully observed tree over just the $\boldsymbol{W}$. In particular for any choice of the root, the joint distribution then factorizes as

$$\mathbb{P}(\boldsymbol{w}, \boldsymbol{z}|\boldsymbol{x}) = \prod_{i=1}^{\ell(\boldsymbol{x})} \mathbb{P}(z_i|z_{\pi(z_i)}, \boldsymbol{x}) \prod_{i=1}^{\ell(\boldsymbol{x})} \mathbb{P}(w_i|z_{\pi(w_i)}, \boldsymbol{x}) \tag{6.2}$$

where $\pi(\cdot)$ returns the parent node of the argument (We have assumed all the observed nodes W_i are leaves) and $\ell(\boldsymbol{x})$ is the length of tag sequence $\boldsymbol{x}$. Recall that each W_i is a real valued vector that represents an embedding/feature for word i.

For constituent and dependency parsing we make restrictions on the trees that we detail below:

- **Constituent parsing:** We choose the sub-family of latent trees where all the observed variables (words) are leaves and each internal node is a latent node with exactly three neighbors. An example is shown in Figure 6.3(a).
- **Dependency parsing:** We choose the sub-family of latent trees where each observed node (word) is a leaf and each hidden node has exactly one observed child (but it can have arbitrary latent children). An example is shown in Figure 6.3(b). Note that another option would be simply a tree consisting of only the words. However, this model is more expressive since it allows longer range dependencies via the latent variables.

6.2.1 Bias Mappings

After obtaining the undirected latent tree $u \in \mathcal{U}$, we need to convert it to a constituent/dependency tree via the *bias mapping* h:

- **Dependency parsing**: We choose the first non-participle verb in the sentence and pick the latent node associated with that verb to be the root. The dependency tree is simply the latent tree with the observed nodes removed (see Figure 6.3(d)).
- **Constituent parsing**: For undirected constituent trees, note that all the internal nodes have 3 neighbors. However, in a (binary) constituent parse, the root has only two neighbors. Our bias mapping selects one edge in the tree and inserts a new node there that will be the root (the tree is then directed accordingly). For example in Figure 6.3(a), the new root node should be placed between z_1 and z_2 (see Figure 6.3(c)). In general, we pick the edge that separates the noun phrase from the verb phrase, which can be heuristically determined.

6.2.2 Projectivity

In general latent tree graphical models, the order of nodes is irrelevant. However, this is not the case in parsing, where the parse tree must respect the order of the words of the sentence. This requirement is called projectivity[2]. We give a rigorous definition of projectivity for constituent parse trees below (a similar definition exists for dependency trees).

Definition 4. *For any pair of sentence positions i, j (where $i < j$) let a_{ij} be the closest common ancestor and let $desc(a_{ij})$ be the set of descendants of a_{ij}. A constituent parse tree is **projective** if and only if for all position pairs i, j s.t. $i < j$, $k \in desc(a_{ij})$ if and only if $i \leq k \leq j$.*

6.3 Learning the Projective Latent Tree Structure

Assume N examples of the form $\mathcal{D} = [(\boldsymbol{w}^{(i)}, \boldsymbol{x}^{(i)})]_{i=1}^{N}$ are observed and we would like to recover $u_{\boldsymbol{x}}$ for tag sequence $\boldsymbol{x}$. 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. Consider constituent parsing for simplicity (e.g. Figure 6.3(a)).

Assume first that that $|\mathcal{D}(x)|$ is large (we address the data sparsity problem in the next section). In this case, we can leverage additive tree metrics and associated algorithms [Saitou and Nei, 1987, Lake, 1994, Choi et al., 2011, Anandkumar et al., 2011] discussed in Chapter 4. The only difference is that our tree metric is now conditioned on $\boldsymbol{x}$, since we would like to recover a different latent tree for each tag sequence. In particular, since each W_i is a vector, rather than a scalar, we propose the following additive tree metric based on [Anandkumar et al., 2011]:

$$d_{\boldsymbol{x}}(i,j) = -\tfrac{1}{2}\log\Lambda_k(\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j)) + \tfrac{1}{4}\log\Lambda_k(\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,i)) + \tfrac{1}{4}\log\Lambda_k(\boldsymbol{\Sigma}_{\boldsymbol{x}}(j,j)) \tag{6.3}$$

where $\Lambda_k(\boldsymbol{A})$ denotes the product of the top k singular values of $\boldsymbol{A}$ and $\boldsymbol{\Sigma}_{\boldsymbol{x}}(i,j) := \mathbb{E}[W_i W_j^\top | \boldsymbol{x}]$.

As mentioned in Chapter 4, we can then use the meta-algorithms of [Saitou and Nei, 1987, Choi et al., 2011, Anandkumar et al., 2011] to recover $u_{\boldsymbol{x}}$. It is possible to restrict many of these meta-algorithms to return only projective trees.

6.3.1 A More Optimal Tree via Minimum Balanced Evolution

The above technique is provably consistent: if the modeling assumptions hold, then as the sample size increases, the empirical distance matrix will exactly satisfy the additive tree metric property, and the correct latent tree will be recovered. However, in practice it is unlikely that the empirical distance matrix exactly satisfies an additive tree metric (due to both estimation error and modeling error). In this case, we would like an algorithm that returns the best tree according to some criterion. The most natural criterion would be the likelihood, but this will most likely be intractable.

Instead we suspect it is possible to find the optimal latent tree according to the *balanced minimum evolution* (BME) criterion [Desper and Gascuel, 2005], which is commonly used in phylogenetics. As the

[2]There is also interest in non-projective parsing, but we do not focus on it here

name suggests, the goal is to find the latent tree that has the smallest sum of edge weights (where the edge weights are set according to a certain formula that leverages the additive tree metric assumption).

For general trees, finding the BME-optimal latent tree is NP-hard. However, we suspect that it can actually be solved efficiently for projective latent trees using a variant of the CKY dynamic programming algorithm [Manning and Schütze, 1999].

6.3.2 Anchoring for Dealing With Data Sparsity

We now address the data sparsify problem, in particular that $\mathcal{D}(\boldsymbol{x})$ can be very small. For example, the Penn treebank [Marcus et al., 1993] has a total number of 43,498 sentences, with 42,246 *unique* part-of-speech tag sequences, averaging $|\mathcal{D}(\boldsymbol{x})|$ to be 1.04.

Our approach to solve this data sparsity issue with $\mathcal{D}(x)$ is to use a method we call *anchoring*. Consider estimating the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{x}}(1,2)$ for the tag sequence $\boldsymbol{x} = (\texttt{DT}_1, \texttt{NN}_2, \texttt{VBD}_3, \texttt{DT}_4, \texttt{NN}_5)$ shown in Figure 6.3. $\mathcal{D}(\boldsymbol{x})$ may be insufficient for an accurate empirical estimate. However, consider another sequence $\boldsymbol{x}' = (\texttt{RB}_1, \texttt{DT}_2, \texttt{NN}_3, \texttt{VBD}_4, \texttt{DT}_5, \texttt{ADV}_6, \texttt{NN}_7)$. 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 anchoring. 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 part-of-speech tags $\boldsymbol{x}$ to a local context $G(j, x)$. The anchor we use is $G(j, x) = (j, x_j)$. 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 $G(j, x)$.

More formally, let $C_{j',k'|i'} = w_{j'}^{(i')}(w_{k'}^{(i')})^\top$. Then,

$$\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{x}^{(i)}}(j,k) = \frac{\sum_{i'=1}^{N}\sum_{j'=1}^{\ell(x^{(i')})}\sum_{k'=1}^{\ell(x^{(i')})} K(j,k,j',k'|\boldsymbol{x}^{(i)},\boldsymbol{x}^{(i')})C_{j',k'|i'}}{\sum_{i'=1}^{N}\sum_{j'=1}^{\ell(x^{(i')})}\sum_{k'=1}^{\ell(x^{(i')})} K_\gamma(j,k,j',k'|\boldsymbol{x}^{(i)},\boldsymbol{x}^{(i')})} \tag{6.4}$$

where $K_\gamma(j,k,j',k'|\boldsymbol{x}^{(i)},\boldsymbol{x}^{(i')})$ is the kernel smoothing function. γ is the bandwidth parameter that trades off the bias and variance of the estimate.

6.4 Summary of Contributions

This project is a work in progress. Currently we are exploring the anchoring strategy but with Chow Liu (fully) observed trees. After getting this to perform well in practice I will work on the latent tree case.

1. Make sure anchoring strategy works with chow liu tree. Examine if dependencies among lexical forms are sufficient, or if some prior over tree structures is required.

2. Extend to latent tree case and determine if possible to achieve optimal tree according to balanced minimum evolution criterion.

3. Prove a sample complexity bound.

4. Evaluate in comparison to DMV and CCM (Klein and Manning 2004) on English and non-English languages.

Chapter 7

Language Modeling via Power Low Rank Ensembles

Finally, we propose to use the linear algebra point of view of latent variables to develop a novel approach for n-gram language modeling, where we believe low rank approaches have the potential to present an attractive solution to the data sparsity problem. We present a preliminary model called **power low rank ensembles**, a low rank framework that includes existing language models, such as absolute discounting and Kneser Ney, as special cases.

7.1 Motivation

Language modeling is the task of evaluating the probability of a sequence of words in a language, based on the patterns previously observed in corpora of that language. It is an important component in, among other applications, modern speech recognition [Rabiner and Juang, 1993] and machine translation decoders [Koehn, 2010], by scoring various hypotheses produced by such systems. The language model plays a key role in discarding unlikely word sequences by assigning low probabilities to them, and promoting hypotheses that contain more likely word sequences.

The predominant approach to language modeling is the n-gram language model, wherein the probability of a word sequence $\mathbb{P}(w_1, \ldots, w_\ell)$ is first factored and then approximated (with the Markov assumption) as:

$$\begin{aligned}\mathbb{P}(w_1, \ldots, w_\ell) &= \prod_{i=1}^{m} \mathbb{P}(w_i | w_1, \ldots, w_{i-1}) \\ &\approx \prod_{i=1}^{\ell} \mathbb{P}(w_i | w_{i-n+1}^{i-1})\end{aligned}$$

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: the vocabulary can often be huge, and thus the large number of parameters can often be inaccurately estimated from the data. For example, due to the power-law nature of language, naive maximum likelihood approaches will be able to estimate some common subsequences very accurately while impractically assigning zero probability to rare, but legitimate word sequences that are not observed in the training 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 reassign probability mass, especially at the lower end of the frequency curve, to take into account such unseen word sequences, and to *interpolate* with or *back off* to lower order n-gram models as the need arises [Chen and Goodman, 1999]. Techniques such as Kneser-Ney smoothing [Kneser and Ney, 1995] and variants have often been the state of the art for more than a decade, by making intelligent choices on how back-off weights are computed and what distribution to back off to.

However, in some cases the semantic relatedness among words has the potential to alleviate the data sparsity problem. For example, the word *jubiliant* appears much less than the word *happy*. However, since both are synonyms, they tend to precede/follow similar words. How can the knowledge of the distribution of words that precedes/follows *happy* be used to make more robust estimate of the distribution of words that precedes/follows *jubiliant*?

This has led many to speculate that there exists an underlying (lower dimensional) latent space that can help alleviate the data sparsity problem in language modeling. A simple latent model for bigrams could be just the two observed nodes w_i and w_{i-1} separated by a latent variable [Saul and Pereira, 1997]. As described in Chapter 2, this would simply correspond to a low rank matrix where the rank corresponds to the number of states that the latent variable takes. Computing this low rank approximation can be done either by SVD, EM or Nonnegative Matrix Factorization depending on what characteristics are desired in the low rank matrix. This idea an be extended to higher order n-grams via tensor factorizations.

While the intuition is quite compelling, very few low rank-based approaches exist for language modeling. There are a few works (e.g. [Saul and Pereira, 1997, Hutchinson et al., 2011]) which mostly look at taking low rank approximations of the bigram or trigram maximum likelihood estimate (MLE) to leverage an underlying latent space. However, these approaches tend to only be effective in very restricted settings and in general are non-competitive with Kneser Ney.

In this proposal, we contend that low-rank approaches do have potential, and that what is missing is a general framework that can give theoretical guidance on how to develop and integrate these low rank models into existing n-gram approaches. To address this problem, we propose a method called **power low rank ensembles**. We hope that our method, by including Absolute Discounting and Kneser Ney as special cases, can provide a natural way to incorporate low rank structure in language models.

7.2 Absolute Discounting and Kneser Ney Smoothing

To motivate our approach, we first briefly give an overview of two common n-gram smoothing methods, Absolute Discounting[Ney et al., 1994] and Kneser Ney[Kneser and Ney, 1995] . Absolute Discounting is a particularly simple method while Kneser Ney is widely considered to be the state-of-the-art n-gram approach (and is based on Absolute Discounting). We focus on bigram models for simplicity, but these methods can easily be extended to higher order models.

Consider first the following definitions. Let $\widehat{\mathbb{P}}(w_i)$ refers to the maximum likelihood estimate (MLE) of the probability of w_i (and similarly for $\widehat{\mathbb{P}}(w_i|w_{i-1})$) $c(w)$ is count of word w (similarly for $c(w, w_{i-1})$). $N_+(w_{i-1}) := |\{w : c(w, w_{i-1}) > 0\}|$ (the number of distinct words that appear after w_{i-1}). $N_-(w_i) := |\{w : c(w_i, w) > 0\}|$ (the number of distinct words that appear before w_i)

7.2.1 Absolute Discounting

The general intuition behind absolute discounting is to interpolate the higher order n-gram models (which are very sparse) which lower-order n-gram models (that are more dense). However, in order to do this, some probability 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, absolute discounting uses the following equation[1]:

$$\mathbb{P}_{absdisc}(w_i|w_{i-1}) = \frac{\max(c(w_i, w_{i-1}) - D, 0)}{c(w-1)} + \gamma(w_{i-1})\frac{c(w_i)}{\sum_w c(w)} \tag{7.1}$$

where D is the discount that creates the "leftover probability" and $\gamma(w_{i-1})$ is chosen so that the probability sums to one:

$$\gamma(w_{i-1}) = \frac{D \times N_+(w_{i-1})}{\sum_w c(w, w_{i-1})} \quad \text{(if } D \text{ is less than 1)} \tag{7.2}$$

[1]We only present the interpolated version for simplicity.

7.2.2 Kneser Ney Smoothing

The weakness of absolute discounting is that it uses the original unigram probability as the lower order distribution. This is intuitively suboptimal, since the lower order distribution is only given weight if we are unsure about the higher order distribution (i.e. when $\gamma(w_{i-1})$ is large). Thus, intuitively, the lower order distribution should be altered to condition on this fact.

This is the inspiration behind Kneser Ney, an elegant algorithm that is widely considered the state-of-the-art in n-gram language modeling. It proposes the following adjusted lower order estimate:

$$\widehat{\mathbb{P}}_{kn-uni}(w_i) = \frac{N_-(w_i)}{\sum_{w_i} N_-(w_i)} \tag{7.3}$$

Intuitively $P_{kney}(w_i)$ 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.

Thus, the overall bigram Kneser Ney model looks like

$$\widehat{\mathbb{P}}_{kney}(w_i|w_{i-1}) = \frac{\max(c(w_i, w_{i-1}) - D, 0)}{c(w-1)} + \gamma(w_{i-1})\widehat{\mathbb{P}}_{kn-uni}(w_i) \tag{7.4}$$

where $\gamma(w_{i-1})$ is the leftover weight that we obtained from the discounting of the MLE bigram as shown in Eq. 7.2.

7.3 Power Low Rank Ensembles

We now introduce our low rank framework. Our intuition is that just like the Kneser Ney lower order models, our low rank models should also not be low rank approximations of the original bigrams/trigrams, but rather alternate quantities. For simplicity, we only consider second-order models, but plan to generalize the approach to higher order n-grams.

Definition 5. *Define a **power low rank ensemble** to be a tuple $(\boldsymbol{B}, \eta, \boldsymbol{\rho}, \boldsymbol{\kappa}, \boldsymbol{\mathcal{D}}, Z)$ where*

1. *$\boldsymbol{B}$ is the bigram count matrix: $\boldsymbol{B}(w_i, w_{i-1}) = c(w_i, w_{i-1})$.*
2. *η is the number of matrices in the ensemble.*
3. *$\boldsymbol{\rho} = (\rho_1, ..., \rho_\eta)$ is a vector that specifies the power of each matrix in the ensemble.*
4. *$\boldsymbol{\kappa} = (\kappa_1, \kappa_2, ..., \kappa_\eta)$ is a vector that specifies the rank of each matrix in the ensemble.*
5. *$\boldsymbol{\mathcal{D}}(\cdot,\cdot,\cdot)$ is a tensor such that $\boldsymbol{D}(i, j, m)$ is the discount for position (i, j) in the m^{th} matrix in the ensemble.*
6. *Z is the divergence/norm that determines what divergence/norm the low rank approximation is respect to.*

Let $\boldsymbol{B}^{\circ\rho}$ refer to $\boldsymbol{B}$ taken to the elementwise-power ρ. Furthermore define $\boldsymbol{B}_Z^{(\rho_m,\kappa_m)}$ to be the best non-negative rank κ_m approximation of $\boldsymbol{B}^{\circ\rho}$ under the Z divergence.

$$\boldsymbol{B}_Z^{(\rho_m,\kappa_m)} := min_{\boldsymbol{M}\geq 0: rank(\boldsymbol{M})=\kappa_m} \|\boldsymbol{B}^{\circ\rho} - \boldsymbol{M}\|_Z \tag{7.5}$$

We then define the output probability as

$$\widehat{\mathbb{P}}_{plre}(w_i|w_{i-1}) = \sum_{m=1}^{\eta} \left(\prod_{m'=0}^{m-1} \lambda_{m'}(w_{i-1}) \right) \frac{\boldsymbol{B}_Z^{(\rho_m,\kappa_m)}(w_i, w_{i-1}) - \boldsymbol{\mathcal{D}}(w_i, w_{i-1}, m)}{\sum_w \boldsymbol{B}_Z^{(\rho_m,\kappa_m)}(w, w_{i-1})} \tag{7.6}$$

where $\lambda_{m'}(w_{i-1})$ is the leftover probability from discounting the m'^{th} matrix (let $\lambda_0(w_{i-1}) = 1$).

To shed some light onto our model, we first show how the Absolute Discounting/Kneser Ney components are subcases of our framework. First we set Z to be the generalized KL divergence (KL), which is defined to be $KL(\boldsymbol{A}||\boldsymbol{B}) = \sum_{ij}\left(\boldsymbol{A}_{ij}\log(\frac{\boldsymbol{A}_{ij}}{\boldsymbol{B}_{ij}}) - \boldsymbol{A}_{ij} + \boldsymbol{B}_{ij})\right)$. Now consider the following lemma.

Lemma 2. *Let V be the size of the vocabulary i.e. number of possible values of w_i. Then,*

1. $\widehat{\mathbb{P}}(w_i|w_{i-1}) = \frac{\boldsymbol{B}_{KL}^{(1,V)}(w_i,w_{i-1})}{\sum_w \boldsymbol{B}_{KL}^{(1,V)}(w,w_{i-1})}$
2. $\widehat{\mathbb{P}}(w_i) = \frac{\boldsymbol{B}_{KL}^{(1,1)}(w_i,w_{i-1})}{\sum_w \boldsymbol{B}_{KL}^{(1,1)}(w,w_{i-1})}$
3. $\widehat{\mathbb{P}}_{kn-uni}(w_i) = \frac{\boldsymbol{B}_{KL}^{(0,1)}(w_i,w_{i-1})}{\sum_w \boldsymbol{B}_{KL}^{(0,1)}(w,w_{i-1})}$

From this we can make two observations:

- The rank κ is what differentiates bigrams from unigrams. Large κ implies more complex dependence between w_i and w_{i-1} and is thus closer to a bigram. By varying κ we can get something that is between a unigram and a bigram.
- The power ρ is what allows our low rank ensemble to differentiate between Absolute Discounting ($\rho = 1$) and Kneser Ney ($\rho = 0$). Intuitively, the Kneser Ney unigram essentially ignores the frequency of the count (providing its non-zero) and thus downweights very high frequency pairs. The MLE unigram on the other hand is simply the marginal probability and therefore does not do any downweighting.

  The power ρ quantifies this difference. Element-wise powering a matrix with small ρ (close to 0) will relatively downweight high frequency entries. Setting $\rho = 0$ will give a binary matrix (assuming the convention that $0^0 = 0$).

If we set $\mathcal{D}(w_i, w_{i-1}, m) = \min(D, \boldsymbol{B}^{(\rho_m,\kappa_m)}(w_i, w_{i-1}))\mathbb{I}[m < \eta]$ where D is a parameter, we can show Absolute Discounting and Kneser Ney are special cases of our framework. Absolute Discounting can be written as:

$$\widehat{\mathbb{P}}_{absdisc}(w_i|w_{i-1}) = \frac{\boldsymbol{B}_{KL}^{(1,V)}(w_i, w_{i-1}) - \mathcal{D}(w_i, w_{i-1}, 1)}{\sum_w \boldsymbol{B}_{KL}^{(1,V)}(w, w_{i-1})} + \lambda_1(w_{i-1})\frac{\boldsymbol{B}_{KL}^{(1,1)}(w_i, w_{i-1}) - \mathcal{D}(w_i, w_{i-1}, 2)}{\sum_w \boldsymbol{B}_{KL}^{(1,1)}(w, w_{i-1})} \tag{7.7}$$

And Kneser Ney can be written as:

$$\widehat{\mathbb{P}}_{kney}(w_i|w_{i-1}) = \frac{\boldsymbol{B}_{KL}^{(1,V)}(w_i, w_{i-1}) - \mathcal{D}(w_i, w_{i-1}, 1)}{\boldsymbol{B}_{KL}^{(1,V)}(w, w_{i-1})} + \lambda_1(w_{i-1})\frac{\boldsymbol{B}_{KL}^{(0,1)}(w_i, w_{i-1}) - \mathcal{D}(w_i, w_{i-1}, 2)}{\sum_w \boldsymbol{B}_{KL}^{(0,1)}(w, w_{i-1})} \tag{7.8}$$

This formulation asks the natural question, how can other pairs (ρ, κ) be incorporated into language models? How to set the discounts $\mathcal{D}$ in the more general case? Since the vocabulary is so large, the discounts must have some simple structure so all of them do not have to be computed/stored jointly. This is a work in the progress and the main obstacle to implementing the method currently.

7.4 Summary of Contributions

1. Finish developing the method e.g. how to set the discounts, and how to vary ρ and κ.
2. Evaluate and compare to Kneser Ney on perplexity as well as a downstream machine translation task.
3. Develop more scalable approximations to be able to scale to higher order n-grams and vocabulary.

Chapter 8

Proposed Timeline

We provide a tentative timeline for the thesis, showing what has been accomplished and what remains.

Contribution	Status and Schedule
Spectral Learning of Latent Trees	[ICML 2011, ICML 2013], journal version* (Summer 2014)
Spectral Learning of Latent Junction Trees	[UAI 2012], journal version* (Summer 2014)
Kernel Embeddings of Latent Trees	[NIPS 2011]
Power Low Rank Language Models	Fall 2013
Unsupervised Parsing	Fall 2013/Early Spring 2014
Spectral Approximations of Inference	Spring/Summer 2014
Scalable Low Rank Language Models	Fall 2014
Writing Thesis	Fall 2014

Figure 8.1: Timeline for thesis (* = this is one journal paper)

Relevant publications:

- A.P. Parikh, L. Song, and E.P. Xing, **A Spectral Algorithm for Latent Tree Graphical Models**, The 28th International Conference on Machine Learning (ICML 2011)
- L. Song, A.P. Parikh, and E.P. Xing, **Kernel Embeddings of Latent Tree Graphical Models**, Neural Information Processing Systems (NIPS 2011)
- A.P. Parikh, L. Song, M. Ishteva, G. Teodoru and E.P. Xing, **A Spectral Algorithm for Latent Junction Trees**, The 28th Conference on Uncertainty in Artificial Intelligence (UAI 2012)
- L. Song, M. Ishteva, A.P. Parikh, H. Park, and E.P. Xing. **Hierarchical Tensor Decomposition for Latent Tree Graphical Models**. In the 30th International Conference on Machine Learning (ICML 2013)

Bibliography

[Anandkumar et al., 2011] 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 et al., 2012] 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 et al., 2013] 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*.

[Asuncion and Newman, 2007] Asuncion, A. and Newman, D. (2007). UCI machine learning repository.

[Bailly et al., 2009] 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 et al., 2010] 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.

[Baldi et al., 2001] Baldi, P. et al. (2001). *Bioinformatics: the machine learning approach*. The MIT Press.

[Balle et al., 2011] 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.

[Blei et al., 2003] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

[Boots et al., 2010] 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.

[Bradley et al., 2011] Bradley, J. K., Kyrola, A., Bickson, D., and Guestrin, C. (2011). Parallel coordinate descent for l1-regularized loss minimization. *arXiv preprint arXiv:1105.5379*.

[Chen and Goodman, 1999] Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

[Choi et al., 2011] Choi, M. J., Tan, V. Y., Anandkumar, A., and Willsky, A. S. (2011). Learning latent tree graphical models.

[Chow and Liu, 1968] Chow, C. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467.

[Cohen and Collins, 2012] 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 et al., 2012] 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 and Smith, 2009] 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 and Smith, 2010] Cohen, S. B. and Smith, N. A. (2010). Viterbi training for PCFGs: Hardness results and competitiveness of uniform initialization. In *Proceedings of ACL*.

[Dasgupta, 1999] Dasgupta, S. (1999). Learning mixtures of gaussians. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 634–644. IEEE.

[Dempster et al., 1977] 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 and Gascuel, 2005] Desper, R. and Gascuel, O. (2005). The minimum evolution distance-based approach to phylogenetic inference. *Mathematics of evolution and phylogeny*, pages 1–32.

[Dhillon et al., 2012] Dhillon, P. S., Rodu, J., Collins, M., Foster, D. P., and Ungar, L. H. (2012). 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.

[Gillenwater et al., 2010] Gillenwater, J., Ganchev, K., Graça, J., Pereira, F., and Taskar, B. (2010). Sparsity in dependency grammar induction. In *Proceedings of ACL*.

[Golland and DeNero, 2012] Golland, D. and DeNero, J. (2012). A feature-rich constituent context model for grammar induction. In *Proceedings of ACL*.

[Griffiths and Steyvers, 2004] Griffiths, T. L. and Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235.

[Guennebaud et al., 2010] Guennebaud, G., Jacob, B., et al. (2010). Eigen v3. http://eigen.tuxfamily.org.

[Harmeling and Williams, 2011] 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.

[Hastie et al., 2009] 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 et al., 2009] Headden, W. P., Johnson, M., and McClosky, D. (2009). Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proc. of NAACL-HLT*.

[Hoff et al., 2002] Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098.

[Hsu et al., 2009] Hsu, D., Kakade, S., and Zhang, T. (2009). A spectral algorithm for learning hidden Markov models. In *Proc. Annual Conf. Computational Learning Theory*.

[Hsu et al., 2012] Hsu, D., Kakade, S. M., and Liang, P. (2012). Identifiability and unmixing of latent parse trees. In *Advances in NIPS*.

[Hutchinson et al., 2011] Hutchinson, B., Ostendorf, M., and Fazel, M. (2011). Low rank language models for small training sets. *Signal Processing Letters, IEEE*, 18(9):489–492.

[Ipsen and Selee, 2011] Ipsen, I. C. and Selee, T. M. (2011). Ergodicity coefficients defined by vector norms. *SIAM Journal on Matrix Analysis and Applications*, 32(1):153–200.

[Ishteva et al., 2012] Ishteva, M., Park, H., and Song, L. (2012). Unfolding latent tree structures using 4th order tensors. *arXiv preprint arXiv:1210.1258*.

[Jiang et al., 2011] 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, 2005] Katayama, T. (2005). *Subspace methods for system identification*. Springer.

[Klein and Manning, 2004] Klein, D. and Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of ACL*.

[Kneser and Ney, 1995] 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, 2010] Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

[Kolar et al., 2010a] Kolar, M., Parikh, A. P., and Xing, E. P. (2010a). On sparse nonparametric conditional covariance selection. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 559–566.

[Kolar et al., 2010b] 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 and Bader, 2009] Kolda, T. and Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

[Koller and Friedman, 2009] Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. The MIT Press.

[Lake, 1994] 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.

[Liang and Klein, 2009] 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 et al., 2009] 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 and Ihler, 2013] Liu, Q. and Ihler, A. (2013). Variational algorithms for marginal map. *arXiv preprint arXiv:1302.6584*.

[Luque et al., 2012] 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.

[Manning and Schütze, 1999] Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*, volume 999. MIT Press.

[Marcus et al., 1993] Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

[Matsuzaki et al., 2005] Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005). Probabilistic cfg with latent annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 75–82. Association for Computational Linguistics.

[Mossel and Roch, 2005] 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.

[Murphy, 2012] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. The MIT Press.

[Ney et al., 1994] Ney, H., Essen, U., and Kneser, R. (1994). On Structuring Probabilistic Dependencies in Stochastic Language Modelling. *Computer Speech and Language*, 8:1–38.

[Parikh et al., 2011] 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.

[Park, 2002] Park, J. D. (2002). Map complexity results and approximation methods. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 388–396. Morgan Kaufmann Publishers Inc.

[Petrov et al., 2006] Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

[Porteous et al., 2008] Porteous, I., Newman, D., Ihler, A., Asuncion, A., Smyth, P., and Welling, M. (2008). Fast collapsed gibbs sampling for latent dirichlet allocation. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577. ACM.

[Rabiner and Juang, 1993] Rabiner, L. and Juang, B.-H. (1993). Fundamentals of speech recognition.

[Redmond and Baveja, 2002] 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.

[Saitou and Nei, 1987] 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.

[Saul and Pereira, 1997] 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.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels*. The MIT Press.

[Seneta, 1979] Seneta, E. (1979). Coefficients of ergodicity: structure and applications. *Advances in applied probability*, pages 576–590.

[Siddiqi et al., 2010] 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, 2011] Smith, N. A. (2011). Linguistic structure prediction. *Synthesis Lectures on Human Language Technologies*, 4(2):1–274.

[Smola et al., 2007] Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A hilbert space embedding for distributions. In *Algorithmic Learning Theory*, pages 13–31. Springer.

[Smyth et al., 2008] Smyth, P., Welling, M., and Asuncion, A. U. (2008). Asynchronous distributed learning of topic models. In *Advances in Neural Information Processing Systems*, pages 81–88.

[Song et al., 2010] Song, L., Boots, B., Siddiqi, S., Gordon, G., and Smola, A. (2010). Hilbert space embeddings of hidden Markov models. In *Proceedings of the 27th International Conference on Machine Learning*, pages 991–998. ACM.

[Song et al., 2009] 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 et al., 2011] Song, L., Parikh, A., and Xing, E. (2011). Kernel embeddings of latent tree graphical models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 24, pages 2708–2716.

[Sontag et al., 2012] Sontag, D., Meltzer, T., Globerson, A., Jaakkola, T. S., and Weiss, Y. (2012). Tightening lp relaxations for map using message passing. *arXiv preprint arXiv:1206.3288*.

[Spitkovsky et al., 2010] Spitkovsky, V. I., Alshawi, H., Jurafsky, D., and Manning, C. D. (2010). Viterbi training improves unsupervised dependency parsing. In *Proceedings of CoNLL*.

[Stewart and Sun, 1990] Stewart, G. and Sun, J. (1990). *Matrix perturbation theory*, volume 175. Academic press New York.

[Wainwright and Jordan, 2008] 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.

[Zhou et al., 2010] Zhou, S., Lafferty, J., and Wasserman, L. (2010). Time varying undirected graphs. *Machine Learning*, 80(2-3):295–319.