Spectral Learning of General Latent-Variable Probabilistic Graphical Models: A Supervised Learning Approach

Borui Wang WBR@STANFORD.EDU
Department of Computer Science, Stanford University, Stanford, CA 94305

Abstract

In this CS 229 project, I designed, proved and tested a new spectral learning algorithm for learning probabilistic graphical models with latent variables by reducing the hard learning problem into a pipeline of supervised learning tasks. This new algorithmic framework can provide us with more learning power by giving us the freedom to plug in all different kinds of supervised learning algorithms into certain slots that can better fit the model. I also demonstrated the effectiveness of this new spectral learning algorithm through experimenting with a learning task in computational biology called *DNA Splicing Sites Classification*.

1. Introduction

Many important problems in modern machine learning and artificial intelligence can be gracefully captured by probabilistic graphical models with latent variables. However, learning such models can often be very difficult, and current approaches often rely on heuristics-based learning algorithms such as Expectation-Maximization (Dempster et al., 1977), which are susceptible to local optima, very slow to converge and have no theoretical guarantees. As an alternative, in recent years there has been a popular family of "spectral learning" algorithms that attempts to employ the method of moments and the matrix decomposition techniques to tackle the problem of learning latent-variable probabilistic graphical model in a more efficient and local-optima-free manner (Hsu et al., 2009). More recently, researchers have generalized the original form of the spectral learning algorithm to learn latent junction trees (Parikh et al., 2012), but their approach is not general enough and has many restrictions on the structure of data as well as the learning procedures. So in this research project, we take a new view of spectral learning through the lens of instrumental variable regression (Wooldridge, 2012) borrowed from modern econometrics (Hefny et al., 2015) and design a new spectral learning framework that reduces the problem of learning latent-variable probabilsitic graphical models down to a pipeline of well-studied supervised learning problems.

Our learning problem can be formalized as the following desired input-output behavior:

Input: the topology of the latent-variable probabilistic graphical model G, a training set of N $i.i.d.$ samples $\{x_1^d, ..., x_{|\mathscr{O}|}^d\}_{d=1}^N$, a set of observed evidence $\{X_i = x_i\}_{i\in\mathcal{E}}$ (where $\mathcal{E}$ denotes the set of index for the set of observable variables that are observed as evidence), and the index Q of the query node X_Q.

Output: Estimated conditional probability distribution $\widehat{P}[X_Q \mid \{X_i = x_i\}_{i\in\mathcal{E}}]$.

In the following sections, I will first present our new spectral learning algorithm, and then theoretically prove its correctness. I will also discuss its performance in the actual learning experiment of DNA splicing sites classification and analyze its advantages and shortcomings in the last section.

2. The New Spectral Learning Algorithm

In this section, I'll present our new spectral learning algorithm for learning general latent-variable probabilistic graphical models. I will focus on a particular form of latent-variable probabilistic graphical models here - namely, the **latent junction trees**. This modeling choice is justified by the fact that any probabilistic graphical model with latent variables can be turned into a corresponding junction tree with latent variables, and the topology as well as many of the nice theoretical properties of the junction tree can offer us great convenience during the learning and inference processes of our spectral algorithm.

Our main learning and inference algorithm for latent-variable probabilistic graphical models is as follow:

Mathematical notations: $\mathscr{O}$ denotes the set of observed variables $\{X_1, ..., X_{|\mathscr{O}|}\}$; $\mathscr{H}$ denotes the set of hidden variables $\{X_{|\mathscr{O}|+1}, ..., X_{|\mathscr{O}|+|\mathscr{H}|}\}$; $\mathbb{N}(\quad)$ denotes the number of elements in a set; $\mathcal{N}(\quad)$ denotes the number of possible values for a discrete random variable; $l(\quad)$ denotes the length of a vector; $\mathcal{I}(a)$ denotes the order index of a certain value a among all the possible realization values of a discrete random variable A; $\gamma(C)$ denotes the set of all separator sets that are connected with a clique node C in a jucntion tree; we denote the indicator tensor for a set D of discrete random variables by $\mathbb{I}^D = \underset{V\in D}{\otimes} e_V$.

2.1. The Learning Algorithm

- **Model Construction:** Convert the latent-variable graphical model G into an appropriate corresponding latent-variable junction tree T such that each observable variable in the junction tree T can be associated with one leaf clique in T (See Figure 1 for an example).

- **Model Specification:** For each separator set S in the junc-

Course Project Report of *CS 229: Machine Learning* at Stanford University, Fall Term 2016.

tion tree T, pick features for the inside subtree $\phi^S = f^S[\alpha(S)]$ and the outside subtree $\psi^S = h^S[\beta(S)]$, where $\alpha(S) = \{A_1, A_2, ..., A_{\mathbb{N}[\alpha(S)]}\}$ and $\beta(S) = \{B_1, B_2, ..., B_{\mathbb{N}[\beta(S)]}\}$ indicate the critical sets of variables that are involved in the feature function of the inside and outside trees, respectively. Here we require that ϕ^S must be a sufficient statistic for $\mathbb{P}[\alpha(S) \mid \beta(S)]$. For each separator set S_l that is right above a leaf clique node C_l, we require that $\alpha(S_l)$ must be the set of observale variables that are associated with C_l.

- **Stage 1A Regression (S1A):** At each non-leaf separator set S in the latent junction tree T, learn a (possibly non-linear) regression model to estimate $\bar{\phi^S} = \mathbb{E}[\phi^S \mid \psi^S]$. The training data for this regression model are $\left\{\left(f^S(\alpha(S)^d), g^S(\beta(S)^d)\right)\right\}_{d=1}^N$ across all N $i.i.d.$ training samples.

- **Stage 1B Regression (S1B):** At each non-leaf separator set S in the latent junction tree T, where S is connected with K "childen" separator sets $\{S_1, S_2, ..., S_K\}$ below it (see Figure 1), learn a (possibly non-linear) regression model to estimate $\bar{\bigotimes_{k=1}^K \phi^{S_k}} = \mathbb{E}[\bigotimes_{k=1}^K \phi^{S_k} \mid \psi^S]$. The training data for this regression model are $\left\{\left(\bigotimes_{k=1}^K f^{S_k}(\alpha(S_k)^d), g^S(\beta(S)^d)\right)\right\}_{d=1}^N$ across all N $i.i.d.$ training samples.

- **Stage 2 Regression (S2):** At each non-leaf separator set S in the latent junction tree T, use the feature expectations estimated in S1A and S1B to train a linear regression model to predict $\bar{\bigotimes_{k=1}^K \phi^{S_k}} = \mathcal{W}^{\mathcal{S}} \times_S \bar{\phi^S}$, where $\mathcal{W}^{\mathcal{S}}$ is a linear operator associated with S. The training data for this model are estimates of $\left(\bar{\bigotimes_{k=1}^K \phi^{S_k}}, \bar{\phi^S}\right)$ obtained from S1A and S1B across all possible configurations of $\beta(S)$ in the outside subtree of S.

- **Root Covariance Estimation:** At the root clique C_r of the latent junction tree T, estimate the expectation of the outer product of the inside feature vectors of all adjecent separator sets that are connected with C_r by taking average across all N $i.i.d.$ training samples:

$$\mathcal{T}^{C_r} = \widehat{\mathbb{E}}[\bigotimes_{S \in \gamma(C_r)} f^S(\alpha(S))] \approx \frac{\sum_{d=1}^N \bigotimes_{S \in \gamma(C_r)} f^S(\alpha(S)^d)}{N}$$

2.2. The Inference Algorithm

- **Design Tensor Construction:** For each leaf separator set S of the latent junction tree T, construct an inside feature design tensor with modes ϕ^S, A_1, A_2, ..., $A_{\mathbb{N}[\alpha(S)]}$, and dimensions $l(\phi^S) \times \mathcal{N}(A_1) \times \mathcal{N}(A_2) \times ... \times \mathcal{N}(A_{\mathbb{N}[\alpha(S)]})$. Each fiber of Φ^S along mode ϕ^S would be the realization vector of the inside feature function f^S at the corresponding values for variables $A_1, A_2, ..., A_{\mathbb{N}[\alpha(S)]}$, i.e.

$$\Phi^S\big(:, \mathcal{I}(a_1), \mathcal{I}(a_2), ..., \mathcal{I}(a_{\mathbb{N}[\alpha(S)]})\big) = f^S(a_1, a_2, ..., a_{\mathbb{N}[\alpha(S)]}).$$

- **Initial Leaf Message Generation:** At each leaf clique node C of the latent junction tree T, let $\delta(C)$ indicate the set of observable variables that are contained in C, and let S indicate the separator set right above C. We define a function $\zeta(X)$ of an observale variable X that evaluates to an all-one vector if X is not observed in the evidence and evaluates to a value-indicator vector if X is observed in the evidence. That is to say:

$$\zeta(X) = \begin{cases} \vec{1}, & \text{if } X \text{ is not observed} \\ e_x, & \text{if } X \text{ is observed to have value } x \end{cases}$$

Then the upward message that we send from C to its parent clique node C_p can be calculated as:

$$m_{C \to C_p} = \Phi^{S^\dagger} \times_{\{\delta(C)\}} \big[\bigotimes_{X \in \delta(C)} \zeta(X)\big]$$

- **Message Passing:**

☐ *From leaf to root (the upward phase):* For each non-root parent clique node C in T where it is separated by a separator set S from its parent node C_p, once it has received the messages from all of its children nodes $C_1, C_2, ..., C_K$, which are separated from it by the separator sets $S_1, S_2, ..., S_K$ repectively, compute the upward message:

$$\begin{aligned} m_{C \to C_p} &= \mathcal{W}^{\mathcal{S}} \times_{S_1} m_{C_1 \to C} \times_{S_2} m_{C_2 \to C} \times_{S_3} ... \\ &\quad \times_{S_K} m_{C_K \to C} \\ &= \mathcal{W}^{\mathcal{S}} \times_{\mathrm{S_j}}_{j \in \{1,2,...,K\}} m_{C_j \to C} \end{aligned}$$

and then send this message to its parent node C_p.

☐ *At the Root:* At the root clique node C_r in T where it is surrounded by all of its children node $C_1, ..., C_K$ (each separated by the separator sets $S_1, ..., S_K$ respectively), for each $k \in \{1, 2, ..., K\}$, once C_r has received all the $K-1$ upward messages from all of its other $K-1$ children cliques except for C_k, compute the downward message:

$$\begin{aligned} m_{C_r \to C_k} &= \mathcal{T}^{C_r} \times_{S_1} m_{C_1 \to C_r} \times_{S_2} ... \times_{S_{k-1}} m_{C_{k-1} \to C_r} \\ &\quad \times_{S_{k+1}} m_{C_{k+1} \to C_r} \times_{S_{k+2}} ... \times_{S_K} m_{C_K \to C_r} \\ &= \mathcal{T}^{C_r} \times_{\mathrm{S_j}}_{j \in \{1,2,...,K\} \setminus k} m_{C_j \to C_r} \end{aligned}$$

and then send this message to its childen node C_k.

☐ *From root to leaf (the downward phase):* For each non-root parent clique node C in T where it is separated from its parent node C_p by a separator set S and separated from its children nodes $C_1, C_2, ..., C_K$ by the separator sets $S_1, S_2, ..., S_K$ respectively, once it has received the downward message $m_{C_p \to C}$ from its parent node C_p, for each $k \in \{1, 2, ..., K\}$, compute the downward message:

$$\begin{aligned} m_{C \to C_k} &= \mathcal{W}^{\mathcal{S}} \times_S m_{C_p \to C} \times_{S_1} m_{C_1 \to C} \times_{S_2} ... \\ &\quad \times_{S_{k-1}} m_{C_{k-1} \to C} \times_{S_{k+1}} m_{C_{k+1} \to C} \\ &\quad \times_{S_{k+2}} ... \times_{S_K} m_{C_K \to C} \\ &= \mathcal{W}^{\mathcal{S}} \times_S m_{C_p \to C} \times_{\mathrm{S_j}}_{j \in \{1,2,...,K\} \setminus k} m_{C_j \to C} \end{aligned}$$

and then send this message to its children node C_k.

- **Computing Query Result:** For the query node X_Q, locate the leaf clique node that X_Q is associated with and call it C_Q. Call C_Q's parent node C_p and the separator set between them S_Q. We first use the design tensor Φ^{S_Q} and its Moore-Penrose psuedoinverse $\Phi^{S_Q^\dagger}$ to transform the downward incoming message $m_{C_p \to C_Q}$ and the upward outgoing message $m_{C_Q \to C_p}$, respectively, and then compute the Hadamard product of these transformed versions of the two messages to obtain an estimate of the unnormalized conditional probability of $\delta(C_Q)$ given all the evidence $\{X_i = x_i\}_{i \in \mathcal{E}}$:

$$\widehat{\mathbb{P}}[\delta(C_Q) \mid \{X_i = x_i\}_{i \in \mathcal{E}}] \propto \left(m_{C_p \to C_Q} \times_{S_Q} \Phi^{S_Q^\dagger}\right) \circ \left(\Phi^{S_Q} \times_{S_Q} m_{C_Q \to C_p}\right)$$

We can then marginalize out the variables in $\delta(C_Q) \backslash X_Q$ and renormalize to obtain the final query result - the estimate of the conditional probability distribution of the query variable X_Q given all the evidence: $\widehat{\mathbb{P}}[X_Q \mid \{X_i = x_i\}_{i \in \mathcal{E}}]$.

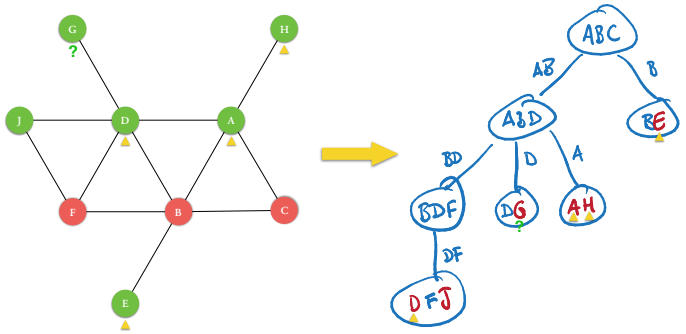

Figure 1. Example of turning a latent-variable probabilistic graphical model into a corresponding latent junction tree

3. Proof of the Algorithm

In this section, we prove the correctness of our main algorithm in three steps:

3.1. Stage 2 Regression

In the first step, let's take a closer look at what the stage 2 regression really does in our learning algorithm. For a typical internal clique node C in the junction tree that is neither a leaf clique nor the root clique, we could draw a generic picture to illustrate what is happening during the stage 2 regression (see Figure 2).

As we could see in the picture, clique C has K children cliques, each separated by a different separator set S_i. Here we denote the subtree under a separator set S (including S) by $T(S)$, and we denote the set of visible variables that are involved in the inside feature function ϕ^S and outside feature function ψ^S of S by $\alpha(S) \subseteq T(S)$ and $\beta(S) \subseteq T \setminus T(S)$, respectively.

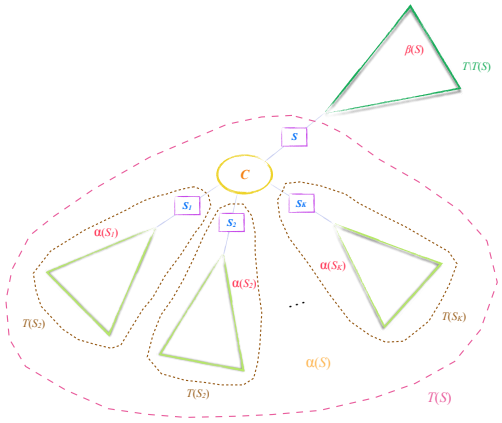

Figure 2. Illustration of a typical internal clique node C in a junction tree

As we described in our main algorithm, at each internal clique node C, the stage 2 regression is a regression from $\mathbb{E}[\phi^S \mid \psi^S]$ to $\mathbb{E}[\phi^{S_1} \otimes \phi^{S_2} \otimes ... \otimes \phi^{S_K} \mid \psi^S]$. Here we assume that all the variables in our graphical model are discrete-valued. For the clarity of presentation, let's analyze the stage 2 regression in two steps below, first in the simple scenario and then in the general scenario.

☐ *The Simple Case:*

In the simple case, if we choose to use indicator functions to build our inside feature function ϕ^S for each non-leaf separator set S as: $\phi^S = \bigotimes_{V \in \alpha(S)} e_V$, while the outside feature function ψ^S could be any legitimate function $f(\beta(S))$ of $\beta(S)$, then we have:

$$\mathbb{E}[\bigotimes_{V \in \bigcup_{i=1}^{K} \alpha(S_i)} e_V \mid f(\beta(S))] = \mathcal{W}^{\mathcal{S}} \times_{\{\alpha(S)\}} \mathbb{E}[\bigotimes_{U \in \alpha(S)} e_U \mid f(\beta(S))],$$

and thus

$$\mathbb{P}[\bigcup_{i=1}^{K} \alpha(S_i) \mid f(\beta(S))] = \mathcal{W}^{\mathcal{S}} \times_{\{\alpha(S)\}} \mathbb{P}[\alpha(S) \mid f(\beta(S))].$$

Therefore,

$$\mathcal{W}^{\mathcal{S}} = \mathbb{P}[\bigcup_{i=1}^{K} \alpha(S_i) \mid \alpha(S), f(\beta(S))].$$

However, according to the running intersection property of the junction tree, the variables contained in a separator set S must completely separate the two sets of variables in the inside tree $T(S)$ and outside tree $T \setminus T(S)$. So we have $T(S) \perp\!\!\!\perp \beta(S) \mid S$. Moreover, since we assume that $\alpha(S)$ serves as a critical set of variables whose joint distribution completely determines the distribution of the whole subtree $T(S)$ (i.e. the so-called "predictive state representation"), knowing the distribution of $\alpha(S)$ would determine the distribution of S. Therefore, $T(S) \perp\!\!\!\perp \beta(S) \mid \alpha(S)$, and thus $\bigcup_{i=1}^{K} \alpha(S_i) \perp\!\!\!\perp f(\beta(S)) \mid \alpha(S)$. As a result, we have:

$$\mathcal{W}^{\mathcal{S}} = \mathbb{P}[\bigcup_{i=1}^{K} \alpha(S_i) \mid \alpha(S), f(\beta(S))] = \mathbb{P}[\bigcup_{i=1}^{K} \alpha(S_i) \mid \alpha(S)].$$

Now we see that in the simplest case where the inside features are the outer products of indicator functions for individual observable variables, the linear operator tensor $\mathcal{W}^{\mathcal{S}}$ that we recover from

the stage 2 regression is actually the conditional probability table $\mathcal{P}[\bigcup_{i=1}^{K} \alpha(S_i) \mid \alpha(S)]$.

□ *The General Case:*

In the general case, we don't put any restriction on the inside and outside feature functions for the separator sets in the junction tree T at all. Therefore, for each separator set S in T, ϕ^S and ψ^S could be any legitimate functions of $\alpha(S)$ and $\beta(S)$. During the model specification step in the learning stage of our main algorithm we've already picked the inside feature vector $\phi^S = f^S[\alpha(S)]$ and the outside feature vector $\psi^S = h^S[\beta(S)]$ for each separator set S in our junction tree T. Now we can construct an inside feature design tensor Φ^S for each S with modes ϕ^S, A_1, A_2, ..., $A_{\mathbb{N}[\alpha(S)]}$, and dimensions $l(\phi^S) \times \mathcal{N}(A_1) \times \mathcal{N}(A_2) \times ... \times \mathcal{N}(A_{\mathbb{N}[\alpha(S)]})$. Each fiber of Φ^S along mode ϕ^S would be the realization vector of the inside feature function f^S at the corresponding values for variables $A_1, A_2, ..., A_{\mathbb{N}[\alpha(S)]}$, i.e.,

$$\Phi^S\big(:, \mathcal{I}(a_1), \mathcal{I}(a_2), ..., \mathcal{I}(a_{\mathbb{N}[\alpha(S)]})\big) = f^S(a_1, a_2, ..., a_{\mathbb{N}[\alpha(S)]}).$$

Then we see that this more general feature vector ϕ^S is connected with the simple indicator function feature tensor $\bigotimes_{V \in \alpha(S)} e_V$ by this inside feature design tensor through the following equation:

$$\mathbb{E}[\phi^S] = \Phi^S \times_{\{A_1, A_2, ..., A_{\mathbb{N}[\alpha(S)]}\}} \mathbb{E}[\bigotimes_{V \in \alpha(S)} e_V].$$

If the matricized version of the tensor Φ^S with dimensions $l(\phi^S) \times [\mathcal{N}(A_1) \cdot \mathcal{N}(A_2) \cdot ... \cdot \mathcal{N}(A_{\mathbb{N}[\alpha(S)]})]$ has full column rank, we can easily retrieve the value of $\mathbb{E}[\bigotimes_{V \in \alpha(S)} e_V]$ from $\mathbb{E}[\phi^S]$ through the mode-specific Moore-Penrose pseudoinverse of Φ^S:

$$\mathbb{E}[\bigotimes_{V \in \alpha(S)} e_V] = \Phi^{S^\dagger} \times_{\{\phi^S\}} \mathbb{E}[\phi^S].$$

This time during the stage 2 regression, at each internal separator set S we are running a regression from $\mathbb{E}[\phi^S \mid \psi^S]$ to $\mathbb{E}[\phi^{S_1} \otimes \phi^{S_2} \otimes ... \otimes \phi^{S_K} \mid \psi^S]$, which is now from

$$\mathbb{E}[\Phi^S \times_{\{\alpha(S)\}} \mathbb{I}^{\alpha(S)} \mid \psi^S]$$

$$\Downarrow \text{ to}$$

$$\mathbb{E}[(\Phi^{S_1} \times_{\{\alpha(S_1)\}} \mathbb{I}^{\alpha(S_1)}) \otimes (\Phi^{S_2} \times_{\{\alpha(S_2)\}} \mathbb{I}^{\alpha(S_2)}) \otimes ... \otimes (\Phi^{S_K} \times_{\{\alpha(S_K)\}} \mathbb{I}^{\alpha(S_K)}) \mid \psi^S]$$

So the new linear operator that we learn from the stage 2 regression would be:

$$\mathcal{W}^S = \mathcal{P}[\bigcup_{i=1}^{K} \alpha(S_i) \mid \alpha(S)] \times_{\{\alpha(S_1)\}} \Phi^{S_1} \times_{\{\alpha(S_2)\}} \Phi^{S_2} \times_{\{\alpha(S_3)\}} ... \times_{\{\alpha(S_K)\}} \Phi^{S_K} \times_{\{\alpha(S)\}} \Phi^{S^\dagger}$$

Now we see that the result $\mathcal{W}^S = \mathbb{P}[\bigcup_{i=1}^{K} \alpha(S_i) \mid \alpha(S)]$ for the simple case above can actually be included here in the general case as a special case where the design tensors Φ^{S_1}, Φ^{S_2}, ..., Φ^{S_k} are all mode-specific identity tensors $\{\mathcal{I}_{\{\alpha(S_i)\}}\}_{i=1}^{K}$.

3.2. Root Covariance Estimation

In the second step, we investigate what we obtain in the root covariance estimation step of our learning algorithm. In the general case, we don't put any restriction on the inside and outside feature functions for the separator sets in the junction tree T at all. Therefore, for each separator set S in T, ϕ^S and ψ^S could be any legitimate functions of $\alpha(S)$ and $\beta(S)$. Following the analysis in the previous section, we have:

$$\begin{aligned}\mathcal{T}^{C_r} &= \widehat{\mathbb{E}}[\bigotimes_{S \in \gamma(C_r)} f^S(\alpha(S))] \\ &= \widehat{\mathbb{E}}\Big[\bigotimes_{S \in \gamma(C_r)} \Big(\Phi^S \times_{\{A_1, A_2, ..., A_{\mathbb{N}[\alpha(S)]}\}} \big(\bigotimes_{V \in \alpha(S)} e_V\big)\Big)\Big] \\ &= \widehat{\mathcal{P}}[\bigcup_{S \in \gamma(C_r)} \alpha(S)] \times_{\substack{\{\alpha(S)\} \\ S \in \gamma(C_r)}} \Phi^S\end{aligned}$$

So we see that in the general case, the covariance tensor $\mathcal{T}^{C_r}$ is actually the design-tensor-transformed version of the empirical estimate of the joint probability table $\widehat{\mathcal{P}}[\bigcup_{S \in \gamma(C_r)} \alpha(S)]$.

3.3. Message-Passing Inference

In the final step, we present an induction-based proof that our message-passing inference procedure in the main algorithm indeed correctly yields the posterior conditional distribution of our query variable given the linear operator tensors $\mathcal{W}^{\mathcal{S}}$ that we learned from Stage 1 and Stage 2 regressions. We can use mathematical induction to prove the following two lemmas:

Lemma 1. *All the upward messages that each non-leaf non-root clique node C sends to its parent node C_p (which is separated from C by the separator set S) is of the form:*

$$m_{C \to C_p} = \widehat{\mathcal{P}}[\{X_i = x_i\}_{X_i \in \eta(C)} \mid \alpha(S)] \times_{\{\alpha(S)\}} \Phi^{S^\dagger}$$

Lemma 2. *All the downward messages that each non-leaf clique node C sends to any of its children node C_i (which is separated from C by the separator set S_i) is of the form:*

$$m_{C \to C_i} = \widehat{\mathcal{P}}[\{X_j = x_j\}_{X_j \in \mu(C_i)}, \alpha(S_i)] \times_{\{\alpha(S_i)\}} \Phi^{S_i}$$

(The actual proofs for this two lemmas are quite long and technical, and because of the tight length limit of this project report, I don't have enough space to include the proofs here in the 5 pages. The interested readers are welcome to contact the author for a copy of the theoretical proofs. Thank you!)

4. Experiment and Analysis

To test the performance of our new spectral learning algorithm in real-world learning tasks, I implemented and experimented with a version of this learning and inference algorithm using Matlab on the task of **DNA Splicing Sites Classification**. This task is a computational biology one of classifying DNA splicing sites using a generative approach through learning higher-order hidden Markov models. Here we make the modeling assumption that the DNA sequences of different splicing site types are generated from different second-order hidden Markov models with different parameters. The **inputs** of this learning task are DNA sequences that are each 60-bases long, and the **outputs** are one of the three possible classification labels of the input DNA sequence - exon/intron boundaries (referred to as *EI* sites), intron/exon boundaries (refered to as *IE* sites), or

neither (denoted N). A concrete example in the training dataset is: 'CCAGCTGCATCACAGGAGGCCAGCGAGCAGGTCTGTTCCAAGGGCCTTCGAGCCAGTCTG' $\rightarrow EI$. In this experiment, I am using the public DNA splicing dataset hosted by the UC Irvine machine learning repository that is available online at: https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences).

This dataset contains a total of 3190 labeled instances of DNA splicing sites, of which 767 are EI sites (25%), 768 are IE sites (25%), and 1655 are Neither (50%). For each individual DNA splicing site sequence of length 60, we construct a length-60 second-order Hidden Markov Model to model its generative process, as shown in Figure 3, and we then turn this second-order HMM into a corresponding latent-variable junction tree as shown in Figure 4.

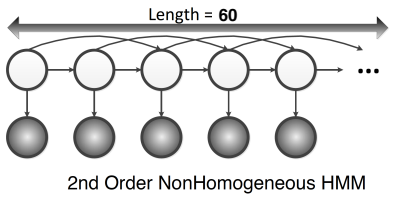

Figure 3. A length-60 second-order Hidden Markov Model (Parikh et al., 2012)

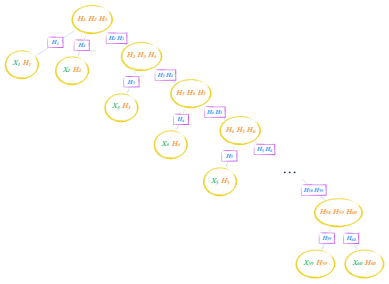

Figure 4. The corresponding latent-variable junction tree

In the experiment, I randomly select out 20% of the training examples for each type of DNA to hold up as the testing data, and then use the other 80% of the training examples to train our learning algorithm. And for the feature selection, I am using the one-hot encoding indicator vectors as the sufficient statistics features for all the bases A,T,C,G in the DNA splicing sequences. As for the choices of supervised learning methods, I am currently using ridge regression for the S1A and S1B stages, and ordinary least squares linear regression for the S2 stage. In the experiment with 5-fold cross-validation, this new learning and inference algorithm implemented with Laplace Smoothing can on average achieve an 81.5% of overall classification accuracy on the testing dataset, with the accuracy breakdown: 79.1% for category EI, 74.5% for category IE, and 85.8% for category N. Figure 5 plots the confusion matrix for the classfication results of our algorithm, and Figure 6 shows the log-likelihoods for three different DNA splicing site categories when predicting test data that are actually labeled with type N.

The current accuracy results of our new sepctral learning algorithm are comparable to the baseline classification accuracy obtained from the classical Expectation-Maximization learning algorithm, while the training time of our new algorithm is significantly shorter than that of the EM algorithm. This is because our new algorithm doesn't rely on the slow heuristics-based optimization procedures and directly captures the essential statistcal information in the training data using method of moments in a much more time-efficient fashion. In the future work, I plan to further extend this new spectral learning framework to the continuous-valued domain through repreducing-kernel Hilbert space embeddings and run more sophisticated experiments on the many challenging structured prediciton tasks in computer vision.

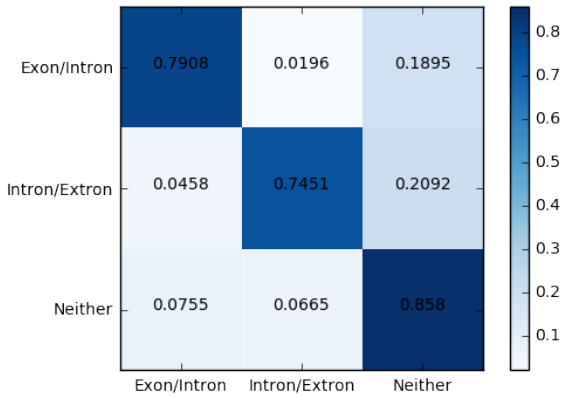

Figure 5. The confusion matrix of the classification result

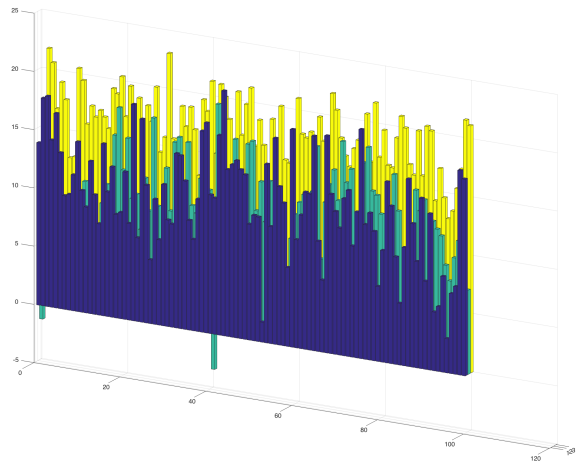

Figure 6. Log-likelihoods for three different DNA splicing site categories when predicting test data that are actually labeled with type N (blue: EI; green: IE; yellow: N)

Acknowledgements

I would like to thank Professor Geoff Gordon for his great advising on this challenging project. I would also like to thank Professor Andrew Ng, Professor John Duchi, my project TA Yin Zi and all the teaching staff of CS 229 for their warm support for my course project and throughout this term's wonderful journey of machine learning. Thank you so much!

References

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 1977.

Hefny, Ahmed, Downey, Carlton, and Gordon, Geoffrey. Supervised learning for dynamical system learning. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

Hsu, Daniel, Kakade, Sham M., and Zhang, Tong. A spectral algorithm for learning hidden markov models. In *COLT*, 2009.

Parikh, Ankur P., Song, Le, Ishteva, Mariya, Teodoru, Gabi, and Xing, Eric P. A spectral algorithm for latent junction trees. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI 2012)*, 2012.

Wooldridge, Jeffrey M. *Introductory Econometrics: A Modern Approach*. Upper Level Economics Titles. South-Western College Pub, 5th edition, 2012.