Approximate Learning for Structured Prediction Problems

Alex Kulesza

November 11, 2009

Abstract

Prediction problems such as image segmentation, sentence parsing, and gene prediction involve complex output spaces for which multiple decisions must be coordinated to achieve optimal results. Unfortunately, this means that there are generally an exponential number of possible predictions for every input. Markov random fields can be used to express structure in these output spaces, reducing the number of model parameters to a manageable size; however, the problem of learning those parameters from a training sample remains NP-hard in general. We review some recent results on approximate learning of structured prediction problems. There are two distinct approaches. In the first, results from the well-studied field of approximate inference are adapted to the learning setting. In the second, learning performance is characterized directly, producing bounds even when the underlying inference method does not offer formal approximation guarantees. While the literature on this topic is still sparse, we review the strengths and weaknesses of current results, and discuss issues that remain for future work.

Contents

Chapter 1

Introduction

Standard prediction problems involve making simple, independent decisions, such as determining whether or not a given e-mail message is spam. Many real-world prediction problems, however, involve complex output spaces for which decisions should or must be coordinated to achieve optimal results. Given an image, for example, it may be difficult to independently determine whether each pixel belongs to an object of interest, but if we encourage consistency among neighboring pixels we can expect to do better given that objects tend to be spatially continuous. Similarly, the part-of-speech of the word "green" is ambiguous in general, but given that the following word is a noun we can surmise that "green" is probably an adjective. Thus, knowledge about the *structure* of the output space, defined as the pattern of interactions between individual decisions, can contribute significantly to improved prediction performance. In fact, for some problems output space structure can be mandatory. For example, in natural language parsing the task is often reduced to a set of binary output variables. The values predicted for these variables must be coordinated to ensure that, taken together, they describe a valid parse tree. If they do not, the prediction is effectively nonsensical.

Unfortunately, the introduction of structure typically involves a combinatorial explosion of output possibilities. Naively, we need to consider the complete cross-product of decisions (e.g., all combinations of part-of-speech tags for all words in a sentence) in order to make an optimal prediction. Of course doing so requires exponential time and is generally intractable. As a compromise, we can limit the sort of structure that we allow. A popular approach is to permit only *local* structure: we build a graph in which individual decisions are represented by nodes, and use a sparse set of edges to indicate the presence of interactions between them. This is the basic form of the Markov Random Field (MRF), which we define formally in Chapter 2 and use as our structure representation throughout.

We are interested in *learning* structured models; that is, our goal is to fit parameters of the model using a training sample so that, at test time, we achieve good prediction performance. In this context, the downside of limiting structure is that the model can become less expressive: there may be interactions that would improve our ability to find good parameters but cannot be included due to computational constraints. Thus the primary challenge of introducing structure into learning is to balance effectively between tractability

and expressivity. (There is a third consideration, namely generalization ability, that we do not consider here, although it can be significant in some cases [11].) At one extreme, we can ignore structure and make each decision independently; this approach is fast but potentially suboptimal. At the other end, we can explicitly model every possible combination of decisions, achieving maximum flexibility but with little hope for efficiency.

Local MRFs form a middle ground; however, even there, the tradeoff can be important. Tree-shaped MRFs, for example, can be learned exactly in polynomial time, but the learning task for general MRFs with cycles is NP-hard. Nonetheless, since tree-shaped MRFs are not generally adequate for tasks such as image segmentation, we seek bounds on the performance of efficient but *approximate* learning schemes. Such schemes are used widely in practice [13, 12, 7], are likely to be of continued value as the availability of complex data grows faster than humans experts can build models manually. We would therefore like to understand the structural conditions under which we can provably achieve useful learning results.

While relatively few results of this type are known, we discuss here two general approaches. In the first, the idea is to leverage the existing work on approximations for the task of *inference*, or making predictions given fixed model parameters. Since learning uses inference as a subroutine, it can be shown that under certain conditions formal approximation of the inference problem leads to learning bounds. In the second approach, the performance of a learning scheme is characterized directly, even though the underlying inference algorithm does not offer formal approximation guarantees.

We proceed by defining our notation and the basic structured learning problem in Chapter 2. We discuss some recent results on inference approximations in Chapter 3, and show how they can be turned into bounds on learning performance in Chapter 4. We present the second main result on learning for denoising problems in Chapter 5, and finally conclude in Chapter 6 with a discussion of the larger principles illustrated by the results, as well as challenges for future work.

Chapter 2

Learning structured models

The basic supervised learning problem is as follows. We receive a training sample $S = \{(x^{(t)}, y^{(t)})\}_{t=1}^{N}$ drawn independently from a distribution D over pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is an input space and $\mathcal{Y}$ is the associated output or label space. We are also given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, where $\ell(y', y)$ is the loss incurred predicting y' when the true output is y. For example, we might have an input space consisting of e-mail messages and a label set with the elements "spam" and "not spam". The loss function can be used to specify that it is worse to label a valid message as spam than to label a spam message as valid. The goal of learning is to choose, based on the training sample, a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ such that the test time *risk* $\mathbb{E}_D[\ell(h(x), y)]$ is minimized.

Structured learning problems are those for which the output space $\mathcal{Y}$ is complex, consisting of multiple related decisions. For concreteness, we will assume $\mathcal{Y} = \mathcal{K}^n$ with $\mathcal{K} = \{1, \ldots, k\}$; that is, there are n outputs, each of which must be assigned a label from 1 to k. For example, we may wish to decide whether or not each pixel in an image participates in the depiction of a certain object ($k = 2$), or assign a part-of-speech label to each word in a sentence ($k = 45$ for the Penn Treebank tag set). We use bold variables to denote vectors, so $\boldsymbol{x} \in \mathcal{X}$ is the input (generally vectorial for structured problems, though this is not critical) and $\boldsymbol{y} = (y_1, y_2, \ldots, y_n) \in \mathcal{Y}$ is the vector of labels. We assume use of the Hamming loss $\ell(\boldsymbol{y}', \boldsymbol{y}) = \sum_{i=1}^{n} \mathbb{I}(y'_i \neq y_i)$, which simply counts the number of mislabeled outputs.

The primary challenge of structured learning is to cope with the combinatorial nature of $\mathcal{Y}$. Note that in general, the output space $\mathcal{Y}$ may consist of more complex structures like trees or strings. It may also depend on the input $\boldsymbol{x}$; for example, the number of part-of-speech tags to be assigned depends on the length of the sentence. However, we keep the setting above for simplicity. The results presented here extend naturally to other structured settings.

There are several immediate reductions to the non-structured case. First, we can treat each decision independently, mapping $(\boldsymbol{x}, \boldsymbol{y})$ to $(\boldsymbol{x}, y_i)$ and applying standard methods for predicting y_i from $\boldsymbol{x}$. By doing this n times, once per output, we can predict the entire vector $\boldsymbol{y}$. However, we cannot take into account interactions among the outputs, which can be highly informative or even required if the decisions must be coordinated in order to make sense. Alternatively, we can think of the entire vector $\boldsymbol{y}$ as a single decision taking labels

from a set of size k^n, and again apply standard methods. While this is quite general, we are interested in cases where k^n is too large to fully enumerate all of the label combinations. Furthermore, the number of parameters will scale with k^n, making generalization difficult or impossible.

We therefore seek methods for learning that directly account for output space structure in order to be both efficient and expressive.

2.1 Markov random fields

We focus on the learning of Markov Random Fields (MRFs), also known as Markov networks, which offer a flexible and intuitive means of representing output structure. An MRF is an undirected graphical model defined by a graph $G = (V, E)$ where the nodes $V = \{1, 2, \ldots, n\}$ represent output variables $Y = \{Y_1, Y_2, \cdots, Y_n\}$. As above, each output takes a value from the label set $\mathcal{K}$, and we use y_i to denote an assignment to Y_i. We use bold $\boldsymbol{y}_c$ to denote an assignment to a set of variables Y_c, and $\boldsymbol{y}$ for an assignment to all output variables. The graph edges E encode direct dependence relationships between variables; for example, there might be edges between neighboring pixels or adjacent words.

An MRF defines a joint probability distribution over the output variables that factorizes across the cliques $\mathcal{C}$ of G:

$$P(\boldsymbol{y}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{y}_c). \tag{2.1}$$

Here each ψ_c is a potential function that assigns a non-negative value to every possible assignment $\boldsymbol{y}_c$ of the clique c, and Z is the normalization constant $\sum_{\boldsymbol{y}'} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{y}'_c)$.

The Hammersley-Clifford theorem states that $P(\boldsymbol{y})$ defined as above is always Markov with respect to G; that is, each variable Y_i is conditionally independent of all other variables given its neighbors in G. The converse also holds: any distribution that is Markov with respect to G, as long as it is strictly positive, can be decomposed over the cliques of G as in Equation (2.1) [3]. MRFs therefore offer an intuitive way to model structure. Given domain knowledge about the nature of the ways in which outputs interact, a practitioner can construct a graph that encodes a precise set of conditional independence relations, reducing the modeling problem to one of choosing appropriate potential functions ψ_c for each clique in that graph. (Because a clique c has $k^{|c|}$ unique assignments, computational constraints generally limit us to small cliques.) We will look at ways in which structured learning can be used to automatically estimate these potentials. (Note that *structure* learning is the process of automatically choosing the graph itself, which we do not consider here.)

For simplicity, we will focus on *pairwise* MRFs, where the largest cliques with interesting potential functions are the edges; that is, $\psi_c(\boldsymbol{y}_c) = 1$ for all cliques c where $|c| > 2$. The pairwise distribution is

$$P(\boldsymbol{y}) = \frac{1}{Z} \prod_{i \in V} \psi_i(y_i) \prod_{ij \in E} \psi_{ij}(y_i, y_j). \tag{2.2}$$

We refer to $\psi_i(y_i)$ as node potentials, and $\psi_{ij}(y_i, y_j)$ as edge potentials.

2.1.1 Conditional parameterization

In order to use MRFs for our learning task and predict different outputs $\boldsymbol{y}$ for different inputs $\boldsymbol{x}$, we need our potential functions to depend on $\boldsymbol{x}$. This leads us to a *conditional* MRF. Equation (2.2) becomes

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{i \in V} \psi_i(y_i|\boldsymbol{x}) \prod_{ij \in E} \psi_{ij}(y_i, y_j|\boldsymbol{x}). \tag{2.3}$$

Note that the normalization term $Z(\boldsymbol{x})$ now depends on the input $\boldsymbol{x}$. The value of $\psi_i(y_i|\boldsymbol{x})$ can be seen as a measure of the compatibility between the label y_i and the input $\boldsymbol{x}$. Similarly, $\psi_{ij}(y_i, y_j|\boldsymbol{x})$ is a measure of the three-way compatibility between y_i, y_j, and $\boldsymbol{x}$.

In order to learn the potential functions effectively, we need to parameterize them in some way. (While we could attempt to directly learn values of $\psi_i(y_i|\boldsymbol{x})$ and $\psi_{ij}(y_i, y_j|\boldsymbol{x})$ for every possible $\boldsymbol{x}$, y_i, and y_j, this approach is not feasible in general due to the large number of possible inputs $\boldsymbol{x}$.) We therefore adopt a log-linear feature-based parameterization in which $\psi_i(y_i|\boldsymbol{x}) = \exp(\boldsymbol{w} \cdot \boldsymbol{f}_i(y_i|\boldsymbol{x}))$ and $\psi_{ij}(y_i, y_j|\boldsymbol{x}) = \exp(\boldsymbol{w} \cdot \boldsymbol{f}_{ij}(y_i, y_j|\boldsymbol{x}))$ for a weight vector $\boldsymbol{w}$ and fixed feature representations $\boldsymbol{f}_i(y_i|\boldsymbol{x})$ and $\boldsymbol{f}_{ij}(y_i, y_j|\boldsymbol{x})$, all elements of $\mathbb{R}^d$. We call the elements of $\boldsymbol{f}_i(y_i|\boldsymbol{x})$ node features, and the elements of $\boldsymbol{f}_{ij}(y_i, y_j|\boldsymbol{x})$ edge features.

The feature representations are fixed in advance by the practitioner; for learning to be effective, they should contain information thought to be relevant to the compatibility amongst labels and between labels and input. For example, there might be an edge feature that is 1 whenever two neighboring pixels have the same label (and zero otherwise). Or there might be a node feature for each part-of-speech tag that can be assigned to each word.

The set of MRFs that can be encoded with this parameterization form an exponential family:

$$\begin{aligned} P_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x}) &= \frac{1}{Z(\boldsymbol{w}, \boldsymbol{x})} \prod_{i \in V} \exp\left(\boldsymbol{w} \cdot \boldsymbol{f}_i(y_i|\boldsymbol{x})\right) \prod_{ij \in E} \exp\left(\boldsymbol{w} \cdot \boldsymbol{f}_{ij}(y_i, y_j|\boldsymbol{x})\right) \\ &= \frac{1}{Z(\boldsymbol{w}, \boldsymbol{x})} \exp\left(\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x})\right), \end{aligned} \tag{2.4}$$

where $\boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{i \in V} \boldsymbol{f}_i(y_i|\boldsymbol{x}) + \sum_{ij \in E} \boldsymbol{f}_{ij}(y_i, y_j|\boldsymbol{x})$ is the global feature function. Note that the normalization term now depends on both $\boldsymbol{w}$ and $\boldsymbol{x}$.

2.2 Learning

Learning entails choosing the parameters $\boldsymbol{w}$ of the MRF model based on a training sample $S = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^{N}$. We outline two basic approaches.

2.2.1 Maximum likelihood

Since the MRF is probabilistic, we can choose parameters $\boldsymbol{w}$ to maximize the conditional log-likelihood of the observed data:

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, S) &= \log \prod_{t=1}^{N} P_{\boldsymbol{w}}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) \\ &= \sum_{t=1}^{N} \log P_{\boldsymbol{w}}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) \\ &= \sum_{t=1}^{N} \left[\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) - \log Z(\boldsymbol{w}, \boldsymbol{x}^{(t)})\right] \\ &= \boldsymbol{w} \cdot \left[\sum_{t=1}^{N} \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})\right] - \sum_{t=1}^{N} \log Z(\boldsymbol{w}, \boldsymbol{x}^{(t)})\end{aligned} \tag{2.5}$$

This approach has the advantage of calibrating the MRF to produce reasonable probability estimates, but it ignores the loss function ℓ and may not produce optimal performance. (It can be thought of as optimizing the logistic loss instead.)

To optimize the log-likelihood, we can use standard algorithms such as gradient descent or L-BFGS [9]. However, each iteration requires computing the gradients of the normalization terms $Z(\boldsymbol{w}, \boldsymbol{x}^{(t)})$, which is NP-hard for general graphs G. Thus, we need to resort to approximations, which we discuss later.

2.2.2 Maximum margin

An alternative approach is to largely ignore the probabilistic nature of the MRF, and instead treat it simply as a scoring mechanism that assigns values $\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x})$ to outputs $\boldsymbol{y}$ given input $\boldsymbol{x}$. While these scores are unnormalized, and thus cannot be treated as probabilities, they are still adequate to choose the most likely output,

$$\boldsymbol{y}^*(\boldsymbol{x}; \boldsymbol{w}) = \arg\max_{\boldsymbol{y}} P_{\boldsymbol{w}}(\boldsymbol{y}|\boldsymbol{x}) = \arg\max_{\boldsymbol{y}} [\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x})]. \tag{2.6}$$

$\boldsymbol{y}^*$ is known as the maximum *a posteriori* (MAP) labeling.

In this context, we can try to find parameters $\boldsymbol{w}$ that maximize the loss-scaled margin

$$\min_t \min_{\hat{\boldsymbol{y}} \neq \boldsymbol{y}^{(t)}} \frac{\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) - \boldsymbol{w} \cdot \boldsymbol{f}(\hat{\boldsymbol{y}}|\boldsymbol{x}^{(t)})}{\|\boldsymbol{w}\| \ell(\hat{\boldsymbol{y}}, \boldsymbol{y}^{(t)})}. \tag{2.7}$$

Intuitively, we try to ensure that the gap in score between the correct output $\boldsymbol{y}^{(t)}$ and every incorrect output $\hat{\boldsymbol{y}}$ is large compared to the loss $\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}^{(t)})$; that is, for "more wrong" outputs we demand a larger gap.

It is convenient to rewrite this margin maximization problem as a minimization over the norm of $\boldsymbol{w}$ subject to constant margin constraints. The equivalent optimization is:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) \geq \boldsymbol{w} \cdot \boldsymbol{f}(\hat{\boldsymbol{y}}|\boldsymbol{x}^{(t)}) + \ell(\hat{\boldsymbol{y}}, \boldsymbol{y}^{(t)}) \quad \forall\, t, \hat{\boldsymbol{y}} \neq \boldsymbol{y}^{(t)}. \end{aligned} \tag{2.8}$$

Since there are exponentially many outputs $\hat{\boldsymbol{y}}$, this optimization has an intractable number of constraints. We simplify it by converting the exponential constraint set for each example into a single nonlinear constraint:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) \geq \max_{\boldsymbol{y} \in \mathcal{Y}} \left[\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}^{(t)}) + \ell(\boldsymbol{y}, \boldsymbol{y}^{(t)})\right] \quad \forall\, t. \end{aligned} \tag{2.9}$$

Since $\ell(\boldsymbol{y}^{(t)}, \boldsymbol{y}^{(t)})$ is defined to be zero, we do not need to exclude $\boldsymbol{y}^{(t)}$ from the maximization. The maximization sub-problem in Equation (2.9) is called the loss-augmented maximum (LAM). Naturally, it is intractable for general graphs. However, we will see ways of approximating it for certain classes of MRFs.

Equation (2.9) is a convex problem and can be solved in various ways; we will see one approach in Chapter 4.

2.3 Approximate learning

As suggested above, learning formulations for structured problems are generally NP-hard to solve exactly. Hence, we seek bounds on the performance of efficient approximations. In order to have a practical result, we concern ourselves with the quality of *predictions* obtained from a learned model, rather than, for example, the degree to which we can approximate the parameters $\boldsymbol{w}^*$ that would be obtained by exact learning. (The parameters themselves are of little consequence if we cannot make good predictions from them.) We can think of an end-to-end learning *scheme* as consisting of two parts:

1. A learning algorithm for choosing parameters $\boldsymbol{w}$ based on supervised data.
2. A prediction or *inference* algorithm for predicting $\boldsymbol{y}$ given a new input $\boldsymbol{x}$ and learned model parameters $\boldsymbol{w}$.

Both tasks require approximation for general structured models, and in the following chapters we will discuss some ways in which the two approximations can interact. We seek to bound the *risk gap*; that is, the difference between the expected loss of predictions from an efficient scheme and from exact, intractable methods.

Chapter 3

Structured inference

Before getting to approximate structured learning results, we first discuss approximation results for inference, the task of extracting predictions from a fixed model. In a learning setting, inference is required at test time, after the parameters $\boldsymbol{w}$ have been learned, for making predictions on unseen data. Thus inference is an important part of an end-to-end learning scheme. Furthermore, we will see that the intractable steps in learning algorithms themselves can generally be expressed as inference; thus solving hard inference problems can be seen as the central challenge of approximate learning.

Inference given fixed model parameters $\boldsymbol{w}$ is a relatively well-studied problem, and there exist a number of approximate methods, including some with formal approximation guarantees. In this section we review some recent results on using linear programming relaxations to compute MAP solutions for certain classes of MRFs. These will turn out to be especially useful in developing learning bounds.

3.1 MAP inference

We take as our canonical inference problem the computation of the maximum *a posteriori* (MAP) labeling given an input $\boldsymbol{x}$ and parameters $\boldsymbol{w}$. (We discuss the related task of computing marginals in Chapter 5.) Recall that for pairwise MRFs the MAP labeling is

$$\begin{aligned} \boldsymbol{y}^*(\boldsymbol{x}; \boldsymbol{w}) &= \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{y}} \log P(\boldsymbol{y}|\boldsymbol{x}) \\ &= \arg\max_{\boldsymbol{y}} [\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x})], \end{aligned} \tag{3.1}$$

where $\boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}) = \sum_{i \in V} \boldsymbol{f}_i(y_i|\boldsymbol{x}) + \sum_{ij \in E} \boldsymbol{f}_{ij}(y_i, y_j|\boldsymbol{x})$.

It is not obvious from Equation (3.1) how to optimize $\boldsymbol{y}$ other than iterating over all (exponentially many) possible values. However, we can take advantage of the structure of the MRF to rewrite Equation (3.1) as a compact integer linear program (ILP). We define a

binary representation $\boldsymbol{z}$ of $\boldsymbol{y}$ as follows.

$$\begin{aligned} z_i(r) &= \mathbb{I}(y_i = r) \quad \forall\, i \in V, r \in \mathcal{K} \\ z_{ij}(r,s) &= \mathbb{I}(y_i = r \wedge y_j = s) \quad \forall\, ij \in E, r, s \in \mathcal{K} \end{aligned} \tag{3.2}$$

Note that $\boldsymbol{z}$ is a vector of dimension $nk + mk^2$ where $m = |E|$; we use functional notation as a convenient way to refer to specific elements of this vector. There is a one-to-one mapping between outputs $\boldsymbol{y}$ and representations $\boldsymbol{z}$ under the following conditions:

$$\begin{aligned} \sum_r z_i(r) &= 1 \quad \forall\, i \in V \\ \sum_s z_{ij}(r,s) &= z_i(r) \quad \forall\, ij \in E, r \in \mathcal{K} \\ \boldsymbol{z} &\in \{0,1\}^{(nk+mk^2)}. \end{aligned} \tag{3.3}$$

The first constraint ensures that each node receives exactly one label, and the second ensures that the label pairs assigned to edges agree with the node labels. (Implicitly, each edge appears twice: once as ij and once as ji.) The third constraint requires that $\boldsymbol{z}$ is binary. The inference problem can now be written as

$$\begin{aligned} \boldsymbol{z}^*(\boldsymbol{x};\boldsymbol{w}) = \arg\max_{\boldsymbol{z}} \quad & \sum_{i \in V} \sum_{r \in \mathcal{K}} z_i(r)[\boldsymbol{w} \cdot \boldsymbol{f}_i(r|\boldsymbol{x})] + \sum_{ij \in E} \sum_{r,s \in \mathcal{K}} z_{ij}(r,s)[\boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r,s|\boldsymbol{x})] \\ \text{s.t.} \quad & \sum_r z_i(r) = 1 \quad \forall\, i \in V \\ & \sum_s z_{ij}(r,s) = z_i(r) \quad \forall\, ij \in E, r \in \mathcal{K} \\ & \boldsymbol{z} \in \{0,1\}^{(nk+mk^2)}, \end{aligned} \tag{3.4}$$

where $\boldsymbol{z}^*(\boldsymbol{x};\boldsymbol{w})$ denotes our binary representation of the MAP labeling.

Solving ILPs is NP-hard in general, so the inference formulation in Equation (3.4) is not practical. The primary difficulty is the integrality constraint $\boldsymbol{z} \in \{0,1\}^{(nk+mk^2)}$. We can obtain a linear programming (LP) relaxation by replacing it with the box constraint $\boldsymbol{z} \in [0,1]^{(nk+mk^2)}$:

$$\begin{aligned} \hat{\boldsymbol{z}}(\boldsymbol{x};\boldsymbol{w}) = \arg\max_{\boldsymbol{z}} \quad & \sum_{i \in V} \sum_{r \in \mathcal{K}} z_i(r)[\boldsymbol{w} \cdot \boldsymbol{f}_i(r|\boldsymbol{x})] + \sum_{ij \in E} \sum_{r,s \in \mathcal{K}} z_{ij}(r,s)[\boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r,s|\boldsymbol{x})] \\ \text{s.t.} \quad & \sum_r z_i(r) = 1 \quad \forall\, i \in V \\ & \sum_s z_{ij}(r,s) = z_i(r) \quad \forall\, ij \in E, r \in \mathcal{K} \\ & \boldsymbol{z} \in [0,1]^{(nk+mk^2)}, \end{aligned} \tag{3.5}$$

where $\hat{\boldsymbol{z}}(\boldsymbol{x};\boldsymbol{w})$ is a possibly fractional representation. We can obtain $\hat{\boldsymbol{z}}(\boldsymbol{x};\boldsymbol{w})$ efficiently using standard algorithms for solving LPs. $\hat{\boldsymbol{z}}(\boldsymbol{x};\boldsymbol{w}) \neq \boldsymbol{z}^*(\boldsymbol{x};\boldsymbol{w})$ in general, although the equivalence holds for graphs G that are trees. However, we are interested here in structured problems for which tree MRFs are insufficient.

3.2 Associative MRFs

In this section we present approximation results for the LP relaxation in Equation (3.5) for MRFs that are *associative* [13]. A MRF is called associative if the clique potentials are at least 1 whenever the nodes in the clique share a common label, and exactly 1 otherwise. Intuitively, the model always encourages nodes in the same clique to have the same label. Formally, a pairwise MRF is associative whenever the objective of Equation (3.4) takes the form

$$\sum_{i \in V} \sum_{r \in \mathcal{K}} \theta_i(r) z_i(r) + \sum_{ij \in E} \sum_{r \in \mathcal{K}} \theta_{ij}(r,r) z_{ij}(r,r) \tag{3.6}$$

where $\theta_i(r) \in \mathbb{R}$ and $\theta_{ij}(r,r) \geq 0$. For MRFs parameterized by joint feature vectors, associativity is equivalent to

$$\begin{aligned} \boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r,r|\boldsymbol{x}) &\geq 0 \quad \forall\, ij \in E, r \in \mathcal{K} \\ \boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r,s|\boldsymbol{x}) &= 0 \quad \forall\, ij \in E, r \neq s. \end{aligned} \tag{3.7}$$

Given these constraints, we can rewrite Equation (3.5) as

$$\begin{aligned} \hat{\boldsymbol{z}}(\boldsymbol{x};\boldsymbol{w}) = \arg\max_{\boldsymbol{z}} \quad & \sum_{i \in V} \sum_{r \in \mathcal{K}} z_i(r)[\boldsymbol{w} \cdot \boldsymbol{f}_i(r|\boldsymbol{x})] + \sum_{ij \in E} \sum_{r \in \mathcal{K}} z_{ij}(r,r)[\boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r,r|\boldsymbol{x})] \\ \text{s.t.} \quad & \sum_r z_i(r) = 1 \quad \forall\, i \in V \\ & z_{ij}(r,r) \leq z_i(r) \quad \forall\, ij \in E, r \in \mathcal{K} \\ & \boldsymbol{z} \in [0,1]^{(nk+mk^2)}, \end{aligned} \tag{3.8}$$

where we have removed unnecessary variables $z_{ij}(r,s)$ for $r \neq s$ and replaced the agreement constraint $\sum_s z_{ij}(r,s) = z_i(r)$ with $z_{ij}(r,r) \leq z_j(r)$ using the fact that the objective coefficients on $z_{ij}(r,r)$ are non-negative, ensuring that they always take their maximum feasible values.

Theorem 1. *For associative MRFs with $k = 2$, an optimum $\hat{\boldsymbol{z}}$ of Equation (3.8) is equal to the optimum $\boldsymbol{z}^*$ of Equation (3.4) (assuming we set $\hat{\boldsymbol{z}}_{ij}(r,s) = \min\{\hat{\boldsymbol{z}}_i(r), \hat{\boldsymbol{z}}_j(s)\}$ for $r \neq s$).*

Proof. It suffices to show that $\hat{\boldsymbol{z}}$ is integral, since the constraints of Equation (3.8) form an outer bound on those of Equation (3.4). (The two problems are equivalent except for the integrality/box constraints.) Thus, if the optimum of Equation (3.8) is integral, it must also be optimal for Equation (3.4).

Suppose that fractional $\hat{\boldsymbol{z}}$ is optimal for Equation (3.8). We construct $\hat{\boldsymbol{z}}'$ with fewer fractional elements that has equal or greater objective value. Let $\boldsymbol{b}$ be a binary vector indicating whether each element of $\hat{\boldsymbol{z}}$ is fractional; that is, $b_i(r) = \mathbb{I}(0 < \hat{z}_i(r) < 1)$ and $b_{ij}(r,r) = \mathbb{I}(0 < \hat{z}_{ij}(r,r) < 1)$. Consider $\hat{\boldsymbol{z}}'$ defined as follows:

$$\begin{aligned} \hat{z}'_i(1) &= \hat{z}_i(1) + b_i(1)C & \hat{z}'_{ij}(1,1) &= \hat{z}_{ij}(1,1) + b_{ij}(1,1)C \\ \hat{z}'_i(2) &= \hat{z}_i(2) - b_i(2)C & \hat{z}'_{ij}(2,2) &= \hat{z}_{ij}(2,2) - b_{ij}(2,2)C, \end{aligned}$$

for some constant C.

Without loss of generality, we can assume that $\hat{z}_{ij}(r,r) = \min\{\hat{z}_i(r), \hat{z}_j(r)\}$, since its objective coefficient is non-negative. Thus if any element of $\hat{\boldsymbol{z}}$ is fractional, some node element must be fractional; that is, there exists i such that $\hat{z}_i(1)$ and $\hat{z}_i(2)$ are both positive. We can therefore define positive $\lambda_1 = \min_{i:\hat{z}_i(1)>0} \hat{z}_i(1)$ and $\lambda_2 = \min_{i:\hat{z}_i(2)>0} \hat{z}_i(2)$. Letting $C = -\lambda_1$ or $C = \lambda_2$ guarantees that $0 \leq \hat{z}'_i(r) \leq 1$ for all i, r. Since $b_i(1) = b_i(2)$, we also have that $\hat{z}'_i(1) + \hat{z}'_i(2) = \hat{z}_i(1) + \hat{z}_i(2) = 1$.

It remains to show that $\hat{z}'_{ij}(r,r) \leq \hat{z}'_i(r)$. If $\hat{z}_{ij}(r,r) = 0$, then $\hat{z}'_{ij}(r,r) = 0 \leq \hat{z}'_i(r)$. Otherwise, assume without loss of generality that $0 < \hat{z}_i(r) \leq \hat{z}_j(r)$ for an edge ij. Then $b_i(r) = b_{ij}(r,r)$, and we have $\hat{z}'_{ij}(1,1) = \hat{z}_{ij}(1,1) + b_{ij}(1,1)C \leq \hat{z}_i(1) + b_i(1)C = \hat{z}'_i(1) \leq \hat{z}'_j(1)$, and similarly for $\hat{z}'_{ij}(2,2)$. Thus $\hat{\boldsymbol{z}}'$ is feasible.

The difference in objective value between $\hat{\boldsymbol{z}}'$ and $\hat{\boldsymbol{z}}$ is

$$\sum_{i \in V} \sum_{r \in \{1,2\}} \boldsymbol{w} \cdot \boldsymbol{f}_i(r|\boldsymbol{x})(\hat{z}'_i(r) - \hat{z}_i(r)) + \sum_{ij \in E} \sum_{r \in \{1,2\}} \boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r,r|\boldsymbol{x})(\hat{z}'_{ij}(r,r) - \hat{z}_{ij}(r,r)) = Cp \tag{3.9}$$

for some constant p. Since we can choose $C = -\lambda_1 < 0$ or $C = \lambda_2 > 0$, we can always find $\hat{\boldsymbol{z}}'$ such that $Cp \geq 0$. □

Since the output $\boldsymbol{y}$ is uniquely determined by the representation $\boldsymbol{z}$, we have that for associative binary MRFs we can efficiently obtain the MAP solution. For more general k, we must consider the possibility that $\hat{\boldsymbol{z}}$ is fractional. In this case, we obtain the output $\hat{\boldsymbol{y}}$ using the following rounding procedure:

- Begin with all outputs unassigned.
- While there exists an unassigned output:
  - Choose a label r uniformly at random from $\mathcal{K}$.
  - Choose a threshold β uniformly at random from $[0, 1]$.
  - For each unassigned output i, set $\hat{y}_i = r$ if $\hat{z}_i(r) > \beta$.
- Return $\hat{\boldsymbol{y}}$.

Theorem 2. *$\hat{\boldsymbol{y}}$ obtained by rounding $\hat{\boldsymbol{z}}(\boldsymbol{x}; \boldsymbol{w})$ is a 2-approximation; that is, it has expected score $\boldsymbol{w} \cdot \boldsymbol{f}(\hat{\boldsymbol{y}}|\boldsymbol{x}) \geq \frac{1}{2} \max_{\boldsymbol{y}} \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x})$.*

Proof. The theorem follows directly from the following two lemmas, given that the objective value of $\hat{\boldsymbol{z}}(\boldsymbol{x}; \boldsymbol{w})$ is always at least the objective value of $\boldsymbol{z}^*(\boldsymbol{x}; \boldsymbol{w})$. □

Lemma 1. *The probability that $\hat{y}_i = r$ is $\hat{z}_i(r)$.*

Proof. On each iteration, $\hat{y}_i$ is set to r with probability $\frac{1}{k}\hat{z}_i(r)$. Since each iteration is independent of the last, the probability that $\hat{y}_i$ is eventually set to r is

$$\frac{\frac{1}{k}\hat{z}_i(r)}{\sum_{r'}\frac{1}{k}\hat{z}_i(r')} = \hat{z}_i(r). \tag{3.10}$$

□

Lemma 2. *The probability that $\hat{y}_i = r$ and $\hat{y}_j = r$ is at least $\frac{1}{2}\hat{z}_{ij}(r,r)$.*

Proof. If neither $\hat{y}_i$ nor $\hat{y}_j$ is assigned at the beginning of an iteration, the iteration sets them both to r with probability $p_1 = \frac{1}{k}\min\{\hat{z}_i(r), \hat{z}_j(r)\} = \frac{1}{k}\hat{z}_{ij}(r,r)$. On the other hand, neither $\hat{y}_i$ nor $\hat{y}_j$ is set during the iteration with probability $p_2 = 1 - \frac{1}{k}\sum_{r'}\max\{\hat{z}_i(r'), \hat{z}_j(r')\}$.

The probability that $\hat{y}_i$ and $\hat{y}_j$ are both eventually set to r is at least the probability that they are both set to r on a single iteration after remaining unassigned though all previous iterations. That probability is

$$\begin{aligned}\sum_{i=1}^{\infty} p_1 p_2^{(i-1)} &= \frac{p_1}{1-p_2} = \frac{\hat{z}_{ij}(r,r)}{\sum_{r'}\max\{\hat{z}_i(r'), \hat{z}_j(r')\}} \\ &\geq \frac{\hat{z}_{ij}(r,r)}{\sum_{r'}[\hat{z}_i(r') + \hat{z}_j(r')]} = \frac{1}{2}\hat{z}_{ij}(r,r).\end{aligned} \tag{3.11}$$

□

In general, the argument in Lemma 2 can be used to show an approximation ratio equal to the size of the largest clique in G.

Theorem 1 and Theorem 2 are closely related to results for the metric labeling problem [4, 2]. However, associative MRFs can express different affinities for different labels; that is, $\boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r, r|\boldsymbol{x})$ can depend on r, whereas the metric condition requires that the cost of an edge between identical labels is always zero. This additional flexibility can be useful in practice.

3.3 Discussion

Since the optimum of a linear program is found at a vertex, the results in Section 3.2 show that for associative MRFs the polytope defined by the constraints in Equation (3.5) has vertices that do not lie too far from integral solutions. It turns out that the exact, integral polytope can also be characterized as an LP, albeit one with exponentially many constraints. Hence these results can be seen as bounding the gap between two polytopes. The exact polytope is sometimes referred to as MARG(G), and the polytope defined by Equation (3.5) is called LOCAL(G). We will return to this characterization of approximate inference in Chapter 5.

Since we are interested primarily in using inference to make predictions from learned parameters $\boldsymbol{w}$, we must ensure associativity of the learned model (Equation (3.7)) in order

for the results presented here to be useful. This can be ensured by choosing non-negative edge features for matching configurations, choosing zero edge features for non-matching configurations, and restricting the parameters $\boldsymbol{w} \geq \mathbf{0}$. Doing this does not materially change the learning problem.

There is, however, another important connection between inference and learning. Recall that the central challenge of max-margin learning is the loss-augmented maximum

$$\max_{\boldsymbol{y} \in \mathcal{Y}} \left[\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}^{(t)}) + \ell(\boldsymbol{y}, \boldsymbol{y}^{(t)}) \right]. \tag{3.12}$$

Recall as well that $\ell(\boldsymbol{y}, \boldsymbol{y}^{(t)})$ is the Hamming loss $\sum_{i=1}^{n} \mathbb{I}(y_i \neq y_i^{(t)})$, which we can write in terms of the binary representation as $n - \sum_{i=1}^{n} \sum_r z_i(r) z_i^{(t)}(r)$. By letting $\theta_i(r) = \boldsymbol{w} \cdot \boldsymbol{f}_i(r|\boldsymbol{x}) - z_i^{(t)}(r)$ and $\theta_{ij}(r,r) = \boldsymbol{w} \cdot \boldsymbol{f}_{ij}(r,r|\boldsymbol{x})$ we can see that the LAM is itself an associative MAP inference problem whenever the underlying MRF is associative.

Thus, MAP inference is useful not only for making test time predictions, but also for learning the parameters of the model itself. In general, learning algorithms of all types rely on inference subroutines. We will see later that for maximum likelihood learning, the central problem of computing the gradient of the normalization term is essentially equivalent to the test time inference problem of computing the marginals of $\boldsymbol{y}$.

Given this dependence on inference at all stages, it is natural to ask whether the inference approximation bounds shown here (and other similar results) lead directly to bounds on the overall performance of a learning scheme. Unfortunately, they generally do not, as we showed in previous work [5]. It turns out that even formally approximate inference algorithms can lead to arbitrarily bad learning results, particularly when the learning and inference algorithms are not appropriately matched. We will need to make additional assumptions in order to obtain end-to-end learning scheme guarantees.

Chapter 4

Learning bounds from approximate inference

In this chapter we describe a recent result due to Martins, Smith, and Xing [6] showing that under certain conditions approximate inference can be used as the basis for an end-to-end learning scheme that provably achieves low additional risk compared with exact methods. We begin by describing an online subgradient algorithm for optimizing the max-margin objective in Equation (2.9). We show how to bound its performance in general and under certain approximation assumptions. Unfortunately, the results from Chapter 3 do not meet these assumptions. However, we describe additional conditions that, combined with inference approximations, lead to an overall risk bound for approximate learning.

4.1 Subgradient optimization

We begin with the max-margin formulation

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{s.t.} \quad & \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) \geq \max_{\boldsymbol{y} \in \mathcal{Y}} \left[\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}^{(t)}) + \ell(\boldsymbol{y}, \boldsymbol{y}^{(t)})\right] \quad \forall\, t. \end{aligned} \tag{4.1}$$

In practice, the data may not be separable; that is, we may not be able to achieve positive margin, making the constraints infeasible. The standard solution is to add slack terms ζ to cover any negative margin, and penalize them in the objective:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{N}\sum_{t=1}^{N} \zeta_t \\ \text{s.t.} \quad & \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)}) \geq \max_{\boldsymbol{y} \in \mathcal{Y}} \left[\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}^{(t)}) + \ell(\boldsymbol{y}, \boldsymbol{y}^{(t)})\right] - \zeta_t \quad \forall\, t, \end{aligned} \tag{4.2}$$

where λ is a tradeoff parameter that controls the strength of the penalty for margin violations.

Given that the slack variables are always tight at the optimum, we can push the margin constraints directly into the objective:

$$\min_{\boldsymbol{w}\in\mathcal{W}} \quad \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{1}{N}\sum_{t=1}^{N} r_t(\boldsymbol{w}), \tag{4.3}$$

where $r_t(\boldsymbol{w}) = \max_{\boldsymbol{y}\in\mathcal{Y}} \left[\boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}^{(t)}) + \ell(\boldsymbol{y},\boldsymbol{y}^{(t)})\right] - \boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})$. Note that $r_t(\boldsymbol{w}) \geq \ell(\boldsymbol{y}^*(\boldsymbol{x}^{(t)};\boldsymbol{w}),\boldsymbol{y}^{(t)})$ since by definition $\boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{y}^*(\boldsymbol{x}^{(t)};\boldsymbol{w})|\boldsymbol{x}^{(t)}) \geq \boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})$. There may be additional constraints, e.g., the non-negativity constraint on $\boldsymbol{w}$ required for learning associative MRFs; we assume these constraints are included in the convex feasible region $\mathcal{W}$. The objective in Equation (4.3) is convex but not generally differentiable due to the maximum operator. In order to optimize it, we can apply a subgradient descent algorithm [10]:

- Initialize $\boldsymbol{w}_1 = 0$, and fix a learning rate sequence $\langle\eta_t\rangle_{t=1}^N$.
- For $t = 1, 2, \ldots, N$:
  - Compute the LAM $\boldsymbol{y}'_t = \arg\max_{\boldsymbol{y}\in\mathcal{Y}} \left[\boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}^{(t)}) + \ell(\boldsymbol{y},\boldsymbol{y}^{(t)})\right]$.
  - Compute the subgradient $\boldsymbol{g}_t = \lambda\boldsymbol{w}_t + \boldsymbol{f}(\boldsymbol{y}'_t|\boldsymbol{x}^{(t)}) - \boldsymbol{f}(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})$.
  - Generate new parameters $\boldsymbol{w}_{t+1} = \mathrm{Proj}_{\mathcal{W}}(\boldsymbol{w}_t - \eta_t\boldsymbol{g}_t)$.
- Return the averaged parameters $\frac{1}{N}\sum_{t=1}^N \boldsymbol{w}_t$.

4.2 Regret bounds

Ratliff et al. [10] prove the following bound on the regret of the subgradient algorithm.

Theorem 3. *Let G bound the norm of $\boldsymbol{g}_t$. (G is finite if feature vectors and parameters $\boldsymbol{w}$ have bounded norm. The latter can be enforced through $\mathcal{W}$.) Fix $\eta_t = 1/(\lambda t)$. Then*

$$\begin{aligned}\frac{1}{N}\sum_{t=1}^{N} \ell(\boldsymbol{y}^*(\boldsymbol{x}^{(t)};\boldsymbol{w}_t),\boldsymbol{y}^{(t)}) &\leq \frac{1}{N}\sum_{t=1}^{N} r_t(\boldsymbol{w}_t) \\ &\leq \frac{1}{N}\sum_{t=1}^{N} r_t(\boldsymbol{w}^*) + \frac{\lambda}{2}\|\boldsymbol{w}^*\|^2 + \frac{G^2(1+\log N)}{2\lambda N},\end{aligned} \tag{4.4}$$

where $\boldsymbol{w}^*$ is an arbitrary reference weight vector, e.g., the one that minimizes the bound.

If we choose $\lambda = \sqrt{\frac{G^2(1+\log N)}{\|\boldsymbol{w}^*\|^2 N}}$, then we get

$$\frac{1}{N}\sum_{t=1}^{N} \ell(\boldsymbol{y}^*(\boldsymbol{x}^{(t)};\boldsymbol{w}_t),\boldsymbol{y}^{(t)}) \leq \frac{1}{N}\sum_{t=1}^{N} r_t(\boldsymbol{w}^*) + \|\boldsymbol{w}^*\|\sqrt{\frac{G^2(1+\log N)}{N}}. \tag{4.5}$$

This says that the average loss (regret) experienced during training, if predictions are made at each iteration using the current parameter vector $\boldsymbol{w}_t$, is bounded by the regret of the optimal parameter vector $\boldsymbol{w}^*$ plus a quantity that diminishes with N.

The bound also applies for approximations of the subgradient. We begin by writing the LAM in a slightly different form. Let $F(\boldsymbol{x})$ be a matrix with columns (using our functional indices) given by $F_i(r|\boldsymbol{x}) = \boldsymbol{f}_i(r|\boldsymbol{x})$ and $F_{ij}(r,s|\boldsymbol{x}) = \boldsymbol{f}_{ij}(r,s|\boldsymbol{x})$. We treat $\boldsymbol{w}$ and $\boldsymbol{z}$ as column vectors. Using the technique from Section 3.3 to incorporate the loss we have

$$\max_{\boldsymbol{y}\in\mathcal{Y}} \left[\boldsymbol{w}\cdot\boldsymbol{f}(\boldsymbol{y}|\boldsymbol{x}^{(t)}) + \ell(\boldsymbol{y},\boldsymbol{y}^{(t)})\right] = \max_{\boldsymbol{z}} \left(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}^{(t)}_{\text{node}})^\top\right)\boldsymbol{z} + n, \tag{4.6}$$

where $\boldsymbol{z}_{\text{node}}$ denotes $\boldsymbol{z}$ with edge indicators $z_{ij}(r,s)$ zeroed out, and the maximization over $\boldsymbol{z}$ is implicitly subject to the ILP constraints in Equation (3.4). With this notation, we have

$$r_t(\boldsymbol{w}) = \max_{\boldsymbol{z}} \left(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}^{(t)}_{\text{node}})^\top\right)\boldsymbol{z} + n - \boldsymbol{w}^\top F(\boldsymbol{x}^{(t)})\boldsymbol{z}^{(t)}. \tag{4.7}$$

When we maximize with respect to the LP-relaxed constraint set, we use the approximate operator $\widehat{\max}$ in place of max.

Now consider computing the LAM at each iteration with $\widehat{\max}$; that is, performing approximate inference using the formulation in Equation (3.5). Then Theorem 3 holds with $r_t(\boldsymbol{w})$ replaced by $\hat{r}_t(\boldsymbol{w}) = \widehat{\max}_{\boldsymbol{z}}(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}^{(t)}_{\text{node}})^\top)\boldsymbol{z} + n - \boldsymbol{w}^\top F(\boldsymbol{x}^{(t)})\boldsymbol{z}^{(t)}$. Unfortunately, the approximation bounds in Chapter 3 are inadequate to guarantee that $\hat{r}_t(\boldsymbol{w})$ approximates $r_t(\boldsymbol{w})$. This is because they guarantee an approximation ratio for $\widehat{\max}_{\boldsymbol{z}}(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}^{(t)}_{\text{node}})^\top)\boldsymbol{z}$ but do not account for the terms $n - \boldsymbol{w}^\top F(\boldsymbol{x}^{(t)})\boldsymbol{z}^{(t)}$. Hence we need further assumptions to bound learning performance.

4.3 Approximating the subgradient

Theorem 4. *Assume that the maximum 2-norm of any column of $F(\boldsymbol{x})$ (over all $\boldsymbol{x}\in\mathcal{X}$) is R. (For example, if we have binary features, then R is the square root of maximum number of simultaneous non-zero features at any single node or edge.) Suppose that the $\widehat{\max}$ operator is an α-approximation for the LAM objective. Then for any $\boldsymbol{w}$ and t,*

$$r_t(\boldsymbol{w}) \le \hat{r}_t(\boldsymbol{w}) \le r_t(\boldsymbol{w}) + (\alpha-1)(n + \|\boldsymbol{w}\|_2 R(n+m)) \tag{4.8}$$

Proof. Since the LP-relaxed constraints define an outer bound on the ILP constraints, we have that $\hat{r}_t(\boldsymbol{w}) \ge r_t(\boldsymbol{w})$. For the second inequality, we have

$$\begin{aligned}
\hat{r}_t(\boldsymbol{w}) &= \widehat{\max_{\boldsymbol{z}}} \left(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}^{(t)}_{\text{node}})^\top\right)\boldsymbol{z} + n - \boldsymbol{w}^\top F(\boldsymbol{x}^{(t)})\boldsymbol{z}^{(t)} \\
&\le \alpha \max_{\boldsymbol{z}} \left(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}^{(t)}_{\text{node}})^\top\right)\boldsymbol{z} + n - \boldsymbol{w}^\top F(\boldsymbol{x}^{(t)})\boldsymbol{z}^{(t)} \\
&= r_t(\boldsymbol{w}) + (\alpha-1)\max_{\boldsymbol{z}} \left(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}^{(t)}_{\text{node}})^\top\right)\boldsymbol{z}.
\end{aligned} \tag{4.9}$$

Using Hölder's inequality,

$$\begin{aligned}
&\left(\boldsymbol{w}^\top F(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}_{\text{node}}^{(t)})^\top\right)\boldsymbol{z} \\
&= \left(\boldsymbol{w}^\top F_{\text{node}}(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}_{\text{node}}^{(t)})^\top\right)\boldsymbol{z}_{\text{node}} + \boldsymbol{w}^\top F_{\text{edge}}(\boldsymbol{x}^{(t)})\boldsymbol{z}_{\text{edge}} \\
&\leq \|\boldsymbol{w}^\top F_{\text{node}}(\boldsymbol{x}^{(t)}) - (\boldsymbol{z}_{\text{node}}^{(t)})^\top\|_\infty \|\boldsymbol{z}_{\text{node}}\|_1 + \|\boldsymbol{w}^\top F_{\text{edge}}(\boldsymbol{x}^{(t)})\|_\infty \|\boldsymbol{z}_{\text{edge}}\|_1 \\
&\leq (\|w\|_2 R + 1)n + \|w\|_2 Rm.
\end{aligned} \tag{4.10}$$

The bound follows. □

Martins et al. prove a slightly different version of the above bound, assuming an alternate type of LAM approximation.

Theorem 5. *Suppose that for every $\hat{\boldsymbol{z}}$ within the LP-relaxed polytope there exists a vertex $\boldsymbol{z}$ of the ILP polytope such that $\ell(\hat{\boldsymbol{z}}, \boldsymbol{z}) \doteq n - \boldsymbol{z}^\top \hat{\boldsymbol{z}} \leq L$ for some constant L. Then*

$$r_t(\boldsymbol{w}) \leq \hat{r}_t(\boldsymbol{w}) \leq r_t(\boldsymbol{w}) + L(1 + \|\boldsymbol{w}\|_2 R). \tag{4.11}$$

Now that we have a bound on the difference between $\hat{r}_t(\boldsymbol{w})$ and $r_t(\boldsymbol{w})$, we can apply Theorem 3 to obtain an approximate learning bound. We use the following concentration result to convert the regret bound to a risk bound.

Lemma 3 (from [1])**.** *Let B be a constant that upper bounds $|\hat{r}_t(\boldsymbol{w})|$. Then for any $\delta > 0$, with probability at least $1 - \delta$,*

$$\frac{1}{N}\sum_{t=1}^{N} \mathbb{E}[\hat{r}_t(\boldsymbol{w}_t)] \leq \frac{1}{N}\sum_{t=1}^{N} \hat{r}_t(\boldsymbol{w}_t) + B\sqrt{\frac{2}{N}\log\frac{1}{\delta}}. \tag{4.12}$$

Theorem 6. *Let $\hat{\boldsymbol{w}}$ be the averaged weight vector produced by the subgradient algorithm using α-approximate LAM inference with learning rate $\eta_t = 1/(\lambda t)$ and $\lambda = \sqrt{\frac{G^2(1+\log N)}{\|\boldsymbol{w}^*\|^2 N}}$. Let $\hat{\boldsymbol{y}}(\boldsymbol{x}; \boldsymbol{w})$ be the result of rounding $\hat{\boldsymbol{z}}(\boldsymbol{x}; \boldsymbol{w})$ as described in Section 3.2. Then for any $\boldsymbol{w}^*$ and $\delta > 0$, with probability at least $1 - \delta$,*

$$\begin{aligned}
&\mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell(\hat{\boldsymbol{y}}(\boldsymbol{x}; \hat{\boldsymbol{w}}), \boldsymbol{y})] \\
&\leq \frac{1}{N}\sum_{t=1}^{N} r_t(\boldsymbol{w}^*) + \|\boldsymbol{w}^*\|\sqrt{\frac{G^2(1+\log N)}{N}} + (\alpha - 1)(n + \|\boldsymbol{w}^*\|_2 R(n+m)) + B\sqrt{\frac{2}{N}\log\frac{1}{\delta}}.
\end{aligned} \tag{4.13}$$

Proof. Since $\hat{r}(\boldsymbol{w})$ is convex, by Jensen's inequality we have

$$\mathop{\mathbb{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim D}[\ell(\hat{\boldsymbol{y}}(\boldsymbol{x}; \hat{\boldsymbol{w}}), \boldsymbol{y})] \leq \mathbb{E}[\hat{r}(\hat{\boldsymbol{w}})] \leq \frac{1}{N}\sum_{t=1}^{N} \mathbb{E}[\hat{r}_t(\boldsymbol{w}_t)]. \tag{4.14}$$

Combining with Theorem 4, Theorem 3, and Lemma 3 yields the bound. □

4.4 Discussion

The bound in Theorem 6 has four terms. The first two are the standard regret terms for the subgradient algorithm, and the fourth is the penalty for converting the regret bound (over the empirical distribution S) to a risk bound (over the true distribution D). The third term, due to the approximation, is most relevant to our discussion. It says that the additional risk incurred due to the use of α-approximate inference for learning and prediction is bounded by $(\alpha-1)(n+\|\boldsymbol{w}^*\|_2 R(n+m))$. Thus, as our approximation ratio approaches 1, the risk penalty shrinks to zero. It also says that the risk penalty scales with the norm of the optimal weight vector $\boldsymbol{w}^*$ and the maximum norm of any single node or edge feature vector R. This can be seen as analogous to the standard Perceptron mistake bound, where the number of mistakes made by the algorithm on linearly separable data is known to be $(\|\boldsymbol{w}^*\|_2 R)^2$, where $\boldsymbol{w}^*$ in this case is the smallest weight vector achieving a margin of 1. However, the Perceptron bound is on the total number of mistakes, while the bound in Theorem 6 is a penalty on the risk; hence as $N \to \infty$ the Perceptron yields zero average regret (for separable data), while the approximation term in the subgradient bound does not shrink with N.

Martins et al. note that we can modify the approximate subgradient algorithm by sometimes performing approximate LAM inference, and sometimes performing exact LAM inference. Though the latter is generally intractable, for some problems it can be solved in a reasonable amount of time using standard software packages such as CPLEX. They show that the more often we perform exact inference, the lower the risk penalty. In particular, if we flip a biased coin on each iteration, doing approximate inference with probability $1/\sqrt{t}$ and exact inference otherwise, then the risk penalty goes to zero as $N \to \infty$. However, this does not seem to be significantly more practical than performing exact inference on every iteration, which we have assumed is too expensive to begin with.

Theorem 6 is, to our knowledge, the first end-to-end approximate learning bound for MRFs in this general setting. However, it is limited in a few important ways. First, noting that $\|\boldsymbol{w}^*\|_2 R(n+m) \geq 0$, the bound is trivial whenever $\alpha \geq 2$ since n is the maximum risk. Recall that the best known approximation ratio for general associative MRFs is 2. In general nontriviality requires

$$\alpha < \frac{2n + \|\boldsymbol{w}^*\|_2 R(n+m)}{n + \|\boldsymbol{w}^*\|_2 R(n+m)}. \tag{4.15}$$

For many problems, e.g., NLP tasks, the number of features can be extremely large, hence the ratio above may be very close to 1. The bound therefore does not offer much practical comfort. The alternate version given in Theorem 5 is interesting (here the risk penalty due to approximation is $L(1+\|\boldsymbol{w}^*\|_2 R)$); however, it is not clear in what situations we can expect an approximation bound of this type to hold for LP-relaxed inference. A counter-example showing that in general L can be $O(n)$ is given in our previous work [5].

A second concern is that the bound is in terms of exact $r_t(\boldsymbol{w})$, which only upper bounds the exact loss. Thus we cannot technically conclude that the approximate setting leads to a bound on the risk gap, only the gap between approximate risk and exact $r_t(\boldsymbol{w})$. Nonetheless, this is a relatively minor issue, since risk bounds for exact methods typically rely on the same

upper bound $r_t(\boldsymbol{w})$. Thus, Theorem 6 says that the risk of the approximate scheme is at most a certain amount more than the best known *bound* on the exact risk.

Chapter 5

Approximate learning of denoising problems

In this chapter we show a result due to Wainwright [14] that bounds the performance penalty incurred in a particular setting when approximate structured learning and inference are used together in place of intractable exact methods. Though the result does not extend easily to some of the problems for which approximate structured methods are commonly used, it provides a useful perspective on the effects of approximate learning under noisy conditions.

The result is also unique in that it does not rely on inference approximation bounds; in fact, the inference method used here does not offer general approximation guarantees for the setting in which it is applied. It is interesting, then, that one can obtain a bound on the performance of an approximate inference method when using an appropriately *learned* model, even though an approximation bound does not exist for general models. This suggests that more learning results might be available even where inference bounds are not, in contrast to the approach discussed in the previous chapter.

5.1 Setting

Suppose we wish to learn a simple denoising task over a set of structured outputs. As before, we have a set of n discrete variables Y, each taking one of k labels, that we will model with a pairwise MRF. However, here each label $r \in \mathcal{K}$ corresponds to a Gaussian distribution $\mathcal{N}(\nu_r, \sigma_r^2)$, and our prediction target is an instance vector $\boldsymbol{z} \in \mathbb{R}^d$ drawn from a Gaussian mixture model where the value of y_i determines the component from which z_i is sampled. The input $\boldsymbol{x} \in \mathbb{R}^d$ is a noisy version of $\boldsymbol{z}$ where independent zero-mean Gaussian noise with unit variance has been added to every component; that is, $x_i = \alpha z_i + \sqrt{1-\alpha^2}\epsilon_i$ for some parameter α, where $\epsilon_i \sim \mathcal{N}(0,1)$. To summarize, examples $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$ are generated according to the following procedure:

1. Sample a vector of labels $\boldsymbol{y} \in \mathcal{Y}$ from the fixed distribution D.
2. Sample an instance vector $\boldsymbol{z}$ according to $z_i \sim \mathcal{N}(\nu_{y_i}, \sigma_{y_i}^2)$.

3. Compute input $\boldsymbol{x} = \alpha \boldsymbol{z} + \sqrt{1-\alpha^2}\boldsymbol{\epsilon}$, where $\epsilon_i \sim \mathcal{N}(0,1)$.

The goal at test time is to reconstruct, given $\boldsymbol{x}$, the instance values $\boldsymbol{z}$. (Note that we are not attempting to predict $\boldsymbol{y}$ itself, a point we will return to later.) In order to take into account structured information, we will learn the parameters of a pairwise MRF over Y that captures interactions between the adjacent variables.

To take a concrete example, suppose that we receive a sequence of images depicting only sky, grass, and earth. We can think of $\boldsymbol{y}$ as specifying which of the three is visible at each pixel. Sky tends to be blue, but is not uniformly blue, thus the true color z of a particular sky pixel is drawn from some Gaussian distribution centered in blue. Likewise, grass pixel colors are drawn from a Gaussian centered in green, and earth pixel colors from a Gaussian centered in brown. In addition, the camera used to create the images is imperfect, so the final image $\boldsymbol{x}$ that we receive has random noise added to the color of every pixel. Our goal is to deduce the "true" image $\boldsymbol{z}$ using our noisy input and knowledge about $\boldsymbol{y}$; for example, sky pixels tend to be next to other sky pixels, and tend to appear near the top of an image.

5.2 Parameterization

Because the underlying distribution over Y is not conditioned on any input, we do not need a conditional MRF with a parameterization based on a joint feature function. Instead, we define the functional parameters $w_i(r)$ for each node i and label r, letting the potential $\psi_i(r) = \exp(w_i(r))$. Similarly, define $w_{ij}(r,s)$ for each edge ij and label pair (r,s), letting $\psi_{ij}(r,s) = \exp(w_{ij}(r,s))$. Let $\boldsymbol{w}$ be the concatenation of all such parameters. Then the probability distribution defined by our MRF is given by

$$P_{\boldsymbol{w}}(\boldsymbol{y}) = \frac{1}{Z(\boldsymbol{w})} \prod_{i \in V} \exp(w_i(y_i)) \prod_{ij \in E} \exp(w_{ij}(y_i, y_j)), \tag{5.1}$$

and the log-probability of a labeling $\boldsymbol{y}$ is

$$\log P_{\boldsymbol{w}}(\boldsymbol{y}) = \sum_{i \in V} w_i(y_i) + \sum_{ij \in E} w_{ij}(y_i, y_j) - \log Z(\boldsymbol{w}). \tag{5.2}$$

We can think of this setup as utilizing a trivial feature representation $\phi(\boldsymbol{y})$ which is the concatenation of the indicators $\mathbb{I}(y_i = r)$ for all nodes i and labels r and $\mathbb{I}(y_i = r \wedge y_j = s)$ for all edges ij and label pairs (r,s). The above can then be rewritten as

$$\log P_{\boldsymbol{w}}(\boldsymbol{y}) = \boldsymbol{w} \cdot \phi(\boldsymbol{y}) - \log Z(\boldsymbol{w}). \tag{5.3}$$

The dimension of this parameterization is $d = nk + mk^2$.

5.3 Variational inference

Recall that for maximum likelihood learning we need to compute the gradient of the logarithm of the normalization term $Z(\boldsymbol{w}) = \sum_{\boldsymbol{y}} P_{\boldsymbol{w}}(\boldsymbol{y})$, which is in general NP-hard. However,

$\log Z(\boldsymbol{w})$ is convex (it is the log-sum-exp of quantities linear in $\boldsymbol{w}$), so we will use convex duality to write it in a variational form.

First, we note that the gradient of $\log Z(\boldsymbol{w})$ is simply the marginal distribution of $\boldsymbol{y}$ under $P_{\boldsymbol{w}}(\boldsymbol{y})$:

$$\begin{aligned} \nabla \log Z(\boldsymbol{w}) &= \frac{1}{Z(\boldsymbol{w})} \sum_{\boldsymbol{y}} \exp(\boldsymbol{w} \cdot \phi(\boldsymbol{y}))\phi(\boldsymbol{y}) \\ &= \mathop{\mathbb{E}}_{\boldsymbol{y} \sim P_{\boldsymbol{w}}} [\phi(\boldsymbol{y})]. \end{aligned} \tag{5.4}$$

Next, consider the conjugate dual function $A^*(\boldsymbol{\mu})$ of $A(\boldsymbol{w}) = \log Z(\boldsymbol{w})$, defined as follows:

$$A^*(\boldsymbol{\mu}) = \sup_{\boldsymbol{w} \in \mathbb{R}^d} \left[\boldsymbol{\mu} \cdot \boldsymbol{w} - A(\boldsymbol{w})\right]. \tag{5.5}$$

To solve the supremum, we take the gradient.

$$\begin{aligned} \nabla \left[\boldsymbol{\mu} \cdot \boldsymbol{w} - A(\boldsymbol{w})\right] &= \boldsymbol{\mu} - \nabla \log Z(\boldsymbol{w}) \\ &= \boldsymbol{\mu} - \mathop{\mathbb{E}}_{\boldsymbol{y} \sim P_{\boldsymbol{w}}} [\phi(\boldsymbol{y})]. \end{aligned} \tag{5.6}$$

Thus, the supremum is attained at $\boldsymbol{w}$ for which $\boldsymbol{\mu}$ represents the marginals of $P_{\boldsymbol{w}}(\boldsymbol{y})$. If there is no such $\boldsymbol{w}$, then $A^*(\boldsymbol{\mu})$ is infinite.

Call the set of $\boldsymbol{\mu}$ for which $A^*(\boldsymbol{\mu})$ is finite $\mathrm{MARG}(G)$, where G is the MRF graph. This set consists of all $\boldsymbol{\mu}$ that represent valid (globally consistent) marginals for G. Let $\boldsymbol{w}(\boldsymbol{\mu})$ be the optimum $\boldsymbol{w}$, i.e., the parameters that generate marginals $\boldsymbol{\mu}$. We omit the details, but it is possible to show that $A^*(\boldsymbol{\mu}) = -H(P_{\boldsymbol{w}(\boldsymbol{\mu})})$ whenever it is finite, where H is the entropy function $H(P) = -\sum_{\boldsymbol{y}} P(\boldsymbol{y}) \log P(\boldsymbol{y})$.

We can now write $\log Z(\boldsymbol{w})$ in terms of its conjugate dual:

$$\begin{aligned} \log Z(\boldsymbol{w}) = A(\boldsymbol{w}) &= \sup_{\boldsymbol{\mu} \in \mathbb{R}^d} \left[\boldsymbol{\mu} \cdot \boldsymbol{w} - A^*(\boldsymbol{\mu})\right] \\ &= \max_{\boldsymbol{\mu} \in \mathrm{MARG}(G)} \left[\boldsymbol{\mu} \cdot \boldsymbol{w} + H(P_{\boldsymbol{w}(\boldsymbol{\mu})})\right]. \end{aligned} \tag{5.7}$$

By Danskin's theorem, we have that have that

$$\nabla \log Z(\boldsymbol{w}) = \mathop{\arg\max}_{\boldsymbol{\mu} \in \mathrm{MARG}(G)} \left[\boldsymbol{\mu} \cdot \boldsymbol{w} + H(P_{\boldsymbol{w}(\boldsymbol{\mu})})\right]. \tag{5.8}$$

Of course, solving this optimization is not necessarily any easier than computing $\nabla \log Z(\boldsymbol{w})$ by exponential sum. First, the set $\mathrm{MARG}(G)$ is difficult to characterize. Second, we do not have a closed form for $H(P_{\boldsymbol{w}(\boldsymbol{\mu})})$. However, we can provide approximations for both of these, leading to a tractable estimate of $\nabla \log Z(\boldsymbol{w})$.

5.3.1 Convex approximation

We approximate Equation (5.8) by specifying a convex outer bound for $\mathrm{MARG}(G)$ and a convex approximation for $-H(P_{\boldsymbol{w}(\boldsymbol{\mu})})$. Observe that any $\boldsymbol{\mu}$ forming the marginal distribution for some choice of parameters $\boldsymbol{w}$ must at least be locally consistent. We define the following constraints on pseudomarginals $\boldsymbol{\tau}$.

$$\begin{aligned} \sum_{r=1}^{k} \tau_i(r) &= 1 \quad \forall\ i \\ \sum_{s=1}^{k} \tau_{ij}(r,s) &= \tau_i(r) \quad \forall\ ij \in E, r \in \mathcal{K} \end{aligned} \tag{5.9}$$

We call the set of $\boldsymbol{\tau}$ satisfying these constraints $\mathrm{LOCAL}(G)$, where, by the above logic, $\mathrm{LOCAL}(G) \supseteq \mathrm{MARG}(G)$. Note that these are the same constraints as those for the linear program in Equation (3.5).

We next turn to approximating the entropy. Since we want to avoid the difficult task of actually locating an appropriate $\boldsymbol{w}$ (which may not exist for pseudomarginals $\boldsymbol{\tau}$ anyway), we compute a local entropy approximation using the fact that $\boldsymbol{\tau}$ are pseudomarginals. In order to ensure that the result is convex, we take a convex combination of tree entropies, for which the local approximation is always exact. (Convexification can be done in other ways as well, see [8] for some recent methods.)

Let $\mathcal{T}$ be a set of trees over the nodes of G, and let $\rho(T)$ be a distribution over the trees $T \in \mathcal{T}$. The choice of trees is arbitrary, though for a good approximation at least one tree should cover each edge in G. We assume $\mathcal{T}$ has been fixed in advance. We define the convex negative entropy approximation

$$B^*(\boldsymbol{\tau}) = -\sum_{T \in \mathcal{T}} \rho(T) \left[\sum_{i \in V} H(\boldsymbol{\tau}_i) - \sum_{ij \in E} I(\boldsymbol{\tau}_{ij}) \right], \tag{5.10}$$

where H is again the entropy function, this time for the marginal distribution of a single variable, and I is the mutual information between two adjacent variables. In addition to being convex, it can be shown that B^* is a lower bound on the true negative entropy.

Combining our entropy approximation with the outer bound $\mathrm{LOCAL}(G)$, we can now state a convex variational approximation of $\log Z(\boldsymbol{w})$:

$$\log Z(\boldsymbol{w}) \approx B(\boldsymbol{w}) = \max_{\boldsymbol{\tau} \in \mathrm{LOCAL}(G)} \left[\boldsymbol{\tau} \cdot \boldsymbol{w} - B^*(\boldsymbol{\tau}) \right]. \tag{5.11}$$

Similarly, we can approximate the gradient as

$$\nabla \log Z(\boldsymbol{w}) \approx \operatorname*{arg\,max}_{\boldsymbol{\tau} \in \mathrm{LOCAL}(G)} \left[\boldsymbol{\tau} \cdot \boldsymbol{w} - B^*(\boldsymbol{\tau}) \right]. \tag{5.12}$$

This approximate formulation is efficiently solvable since the objective is concave and $\mathrm{LOCAL}(G)$ is described by a polynomial number of linear constraints. A standard convex

optimization package could be used to compute the solution, although the tree re-weighted (TRW) message passing algorithm is designed for this purpose and generally more efficient [15].

5.4 Learning

We assume a training sample of label vectors $S = \{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(N)}\}$. Recall that the goal of maximum likelihood training is to maximize the log-likelihood

$$\begin{aligned}\mathcal{L}(\boldsymbol{w}, S) &= \sum_{t=1}^{N} \log P_{\boldsymbol{w}}(\boldsymbol{y}^{(t)}) \\ &= \sum_{t=1}^{N} \left[\boldsymbol{w} \cdot \phi(\boldsymbol{y}^{(t)}) - \log Z(\boldsymbol{w})\right] \\ &= \boldsymbol{w} \cdot \left[\sum_{t=1}^{N} \phi(\boldsymbol{y}^{(t)})\right] - N \log Z(\boldsymbol{w}).\end{aligned} \tag{5.13}$$

We can use standard methods to optimize $\mathcal{L}(\boldsymbol{w}, S)$ by computing the gradient

$$\nabla\mathcal{L}(\boldsymbol{w}, S) = \sum_{t=1}^{N} \phi(\boldsymbol{y}^{(t)}) - N\nabla \log Z(\boldsymbol{w}). \tag{5.14}$$

However, we cannot compute $\nabla \log Z(\boldsymbol{w})$ exactly. Instead, we substitute our variational approximation:

$$\nabla\hat{\mathcal{L}}(\boldsymbol{w}, S) = \sum_{t=1}^{N} \phi(\boldsymbol{y}^{(t)}) - N \max_{\boldsymbol{\tau} \in \mathrm{LOCAL}(G)} \left[\boldsymbol{\tau} \cdot \boldsymbol{w} - B^*(\boldsymbol{\tau})\right]. \tag{5.15}$$

Proposition 1 (moment matching)**.** *The parameters $\hat{\boldsymbol{w}}$ obtained at the optimum of Equation (5.15) yield pseudomarginals $\hat{\boldsymbol{\tau}} = \arg\max_{\boldsymbol{\tau} \in \mathrm{LOCAL}(G)} \left[\boldsymbol{\tau} \cdot \hat{\boldsymbol{w}} - B^*(\boldsymbol{\tau})\right]$ that are equal to the empirical marginals of S. That is,*

$$\begin{aligned}\hat{\tau}_i(r) &= \frac{1}{N}\sum_{t=1}^{N} \mathbb{I}(y_i^{(t)} = r) \quad \forall\ i, r \\ \hat{\tau}_{ij}(r, s) &= \frac{1}{N}\sum_{t=1}^{N} \mathbb{I}(y_i^{(t)} = r \wedge y_j^{(t)} = s) \quad \forall\ ij \in E, r, s.\end{aligned} \tag{5.16}$$

Proof. Immediate from Equation (5.15) and the optimality condition $\nabla\hat{\mathcal{L}}(\boldsymbol{w}, S) = 0$. □

By an analogous argument we can show that $\boldsymbol{w}^*$ optimizing the *exact* log-likelihood yields a distribution whose *exact* marginals $\boldsymbol{\mu}^*$ are equal to the empirical marginals of S.

5.5 Inference

At test time, we need to estimate the instance vector $\boldsymbol{z}$ given an input $\boldsymbol{x}$. The idea will be to first infer from the noisy input a set of marginals $\boldsymbol{\mu}$ for the label variables Y, and then use those marginals to derive an estimate $\boldsymbol{z}(\boldsymbol{x})$ of the instance vector. Again, the exact formulation will be intractable, and we will rely on an efficient approximation.

For concreteness we assume that $k = 2$—that is, there are only two underlying components in the Gaussian mixture model—although the result applies more generally. For $r \in \{1, 2\}$ we write $\tilde{r} = 3 - r$ to indicate the opposite label. Omitting details, it turns out that for a new noisy input $\boldsymbol{x}$ the posterior distribution of $\boldsymbol{y}$ is given by $P_{\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})}$, where $\boldsymbol{w}^*$ is the weight vector that maximizes the log-likelihood of the training data and $\boldsymbol{v}(\boldsymbol{x})$ expresses the additional information contained in $\boldsymbol{x}$. The form of $\boldsymbol{v}(\boldsymbol{x})$ is as follows:

$$\begin{aligned} v_i(r; \boldsymbol{x}) &= \frac{1}{2}\left(\log \frac{\alpha^2\sigma_{\tilde{r}}^2 + (1-\alpha^2)}{\alpha^2\sigma_r^2 + (1-\alpha^2)} - \frac{(x_i - \alpha\nu_r)^2}{\alpha^2\sigma_r^2 + (1-\alpha^2)} + \frac{(x_i - \alpha\nu_{\tilde{r}})^2}{\alpha^2\sigma_{\tilde{r}}^2 + (1-\alpha^2)}\right) \\ v_{ij}(r, s; \boldsymbol{x}) &= 0. \end{aligned} \tag{5.17}$$

Note that $\boldsymbol{x}$ offers information about node values y_i, but not about pairwise interactions.

To compute marginals $\boldsymbol{\mu}(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) \doteq \mathbb{E}_{\boldsymbol{y} \sim P_{(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x}))}}[\phi(y)]$, we recall from Equation (5.4) that

$$\mathop{\mathbb{E}}_{\boldsymbol{y} \sim P_{(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x}))}}[\phi(y)] = \nabla \log Z(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) = \mathop{\arg\max}_{\boldsymbol{\mu} \in \mathrm{MARG}(G)} \left[\boldsymbol{\mu} \cdot (\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) + H(P_{(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x}))})\right]. \tag{5.18}$$

Thus, as in max-margin learning, the same inference problem appears during learning and prediction. We can compute Equation (5.18) using the TRW algorithm.

Given marginals $\boldsymbol{\mu}(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x}))$ of the posterior distribution of $\boldsymbol{y}$, the optimal Bayesian least squares estimator $\boldsymbol{z}^*(\boldsymbol{x})$ is given by

$$z_i^*(x_i) = \sum_{r \in \{1,2\}} \mu_i(r; \boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) g_r(x_i), \tag{5.19}$$

where the Gaussian estimator for mixture component r is

$$g_r(x_i) = \frac{\alpha\sigma_r^2}{\alpha^2\sigma_r^2 + (1-\alpha^2)}(x_i - \alpha\nu_r) + \nu_r. \tag{5.20}$$

Since we cannot compute $\boldsymbol{w}^*$ or $\boldsymbol{\mu}(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x}))$, we resort to a variational approximation and instead compute test time pseudomarginals

$$\boldsymbol{\tau}(\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x})) = \mathop{\arg\max}_{\boldsymbol{\tau} \in \mathrm{LOCAL}(G)} \left[\boldsymbol{\tau} \cdot (\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x})) - B^*(\boldsymbol{\tau})\right], \tag{5.21}$$

where $\hat{\boldsymbol{w}}$ is the weight vector learned using the variational approximation to the maximum likelihood objective. Our approximate estimator $\hat{\boldsymbol{z}}(\boldsymbol{x})$ is given by

$$\hat{z}_i(x_i) = \sum_{r \in \{1,2\}} \tau_i(r; \hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x})) g_r(x_i). \tag{5.22}$$

5.6 Risk bound

We will show that the mean-squared risk gap between the exact estimator and the approximate estimator is bounded. We begin with a lemma showing that small changes in model parameters result in small changes to the marginal predictions.

Lemma 4 (stability)**.** *There exists $L < \infty$ such that for any $\delta \in \mathbb{R}^d$,*

$$\|\boldsymbol{\mu}(\boldsymbol{w} + \delta) - \boldsymbol{\mu}(\boldsymbol{w})\|_2 \leq L\|\delta\|_2. \tag{5.23}$$

Proof. By Equation (5.4) we have that $\boldsymbol{\mu}(\boldsymbol{w}) = \nabla \log Z(\boldsymbol{w}) = \nabla A(\boldsymbol{w})$, thus

$$\begin{aligned} \|\boldsymbol{\mu}(\boldsymbol{w} + \delta) - \boldsymbol{\mu}(\boldsymbol{w})\|_2 &= \|\nabla A(\boldsymbol{w} + \delta) - \nabla A(\boldsymbol{w})\|_2 \\ &= \|\nabla^2 A(\boldsymbol{w} + t\delta)\delta\|_2 \end{aligned} \tag{5.24}$$

for some $t \in [0, 1]$, by the mean value theorem. By properties of the conjugate dual, $\nabla^2 A(\boldsymbol{w}) = [\nabla^2 A^*(\boldsymbol{\mu}(\boldsymbol{w}))]^{-1}$. Since A^* is the negative entropy function, it is strongly convex. (Technically, we need to remove affine dependencies in our representation to ensure strong convexity. Since doing so does not affect the expressivity of the model or contribute any useful intuition, we omit the details.) Thus the singular values of $\nabla^2 A^*$ are uniformly bounded away from zero, and the singular values of $\nabla^2 A$ are uniformly bounded from above. □

Since our variational approximation is based on a strongly convex negative entropy approximation, the same logic allows us to conclude that there exists $R < \infty$ such that $\|\boldsymbol{\tau}(\boldsymbol{w} + \delta) - \boldsymbol{\tau}(\boldsymbol{w})\|_2 < R\|\delta\|_2$ for all $\delta \in \mathbb{R}^d$. We can now state the bound.

Theorem 7. *Let $\mathcal{R}^* = \frac{1}{n}\mathbb{E}\,\|\boldsymbol{z}^*(\boldsymbol{x}) - \boldsymbol{z}\|_2^2$ be the expected per-position mean-squared error of the exact estimator $\boldsymbol{z}^*(\boldsymbol{x})$, and let $\hat{\mathcal{R}} = \frac{1}{n}\mathbb{E}\,\|\hat{\boldsymbol{z}}(\boldsymbol{x}) - \boldsymbol{z}\|_2^2$be the expected per-position mean-squared error of the approximate estimator $\hat{\boldsymbol{z}}(\boldsymbol{x})$. Then the gap $\hat{\mathcal{R}} - \mathcal{R}^*$ is bounded by*

$$\mathbb{E}\left[\min\left(1, (L+R)\frac{\|\boldsymbol{v}(\boldsymbol{x})\|_2}{\sqrt{n}}\right)\sqrt{\frac{1}{n}\sum_{i\in V}|g_1(x_i) - g_2(x_i)|^4}\right]. \tag{5.25}$$

Proof. Since $\boldsymbol{z}^*(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{z}|\boldsymbol{x}}[\boldsymbol{z}]$ is the optimal Bayesian least squares estimator, we have

$$\begin{aligned} \hat{\mathcal{R}} - \mathcal{R}^* &= \frac{1}{n}\mathbb{E}\,\|\hat{\boldsymbol{z}}(\boldsymbol{x}) - \boldsymbol{z}\|_2^2 - \frac{1}{n}\mathbb{E}\,\|\boldsymbol{z}^*(\boldsymbol{x}) - \boldsymbol{z}\|_2^2 \\ &= \frac{1}{n}\mathbb{E}\,\|\boldsymbol{z}^*(\boldsymbol{x}) - \hat{\boldsymbol{z}}(\boldsymbol{x})\|_2^2. \end{aligned} \tag{5.26}$$

Since $k = 2$, $\mu_i(r) = 1 - \mu_i(\tilde{r})$ and similarly for $\boldsymbol{\tau}$, so

$$z_i^*(\boldsymbol{x}) - \hat{z}_i(\boldsymbol{x}) = \left[\mu_i(1; \boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \tau_i(1; \hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x}))\right](g_1(x_i) - g_2(x_i)), \tag{5.27}$$

and thus

$$\begin{aligned}\frac{1}{n}\|\boldsymbol{z}^*(\boldsymbol{x}) - \hat{\boldsymbol{z}}(\boldsymbol{x})\|_2^2 &= \frac{1}{n}\sum_{i\in V}\left[\mu_i(1;\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \tau_i(1;\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x}))\right]^2 \left(g_1(x_i) - g_2(x_i)\right)^2 \\ &\leq \frac{1}{n}\sum_{i\in V}\left|\mu_i(1;\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \tau_i(1;\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x}))\right| \left(g_1(x_i) - g_2(x_i)\right)^2,\end{aligned} \tag{5.28}$$

since $\boldsymbol{\mu}$ and $\boldsymbol{\tau}$ are (pseudo)marginal vectors whose elements are bounded by 1. We can apply the Cauchy-Schwarz inequality to upper bound the last line by

$$\frac{1}{n}\|\boldsymbol{\mu}(1;\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \boldsymbol{\tau}(1;\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x}))\|_2 \sqrt{\sum_{i\in V}|g_1(x_i) - g_2(x_i)|^4}, \tag{5.29}$$

where $\boldsymbol{\mu}(1) \in \mathbb{R}^n$ is the vector of node marginals for the label 1, and likewise for $\boldsymbol{\tau}(1)$. We omit the 1s going forward to lighten notation.

We put aside the quantity $\sqrt{\frac{1}{n}\sum_{i\in V}|g_1(x_i) - g_2(x_i)|^4}$ and focus on bounding $\|\boldsymbol{\mu}(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \boldsymbol{\tau}(\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x}))\|_2/\sqrt{n}$. Trivially, this term is at most 1, since $\boldsymbol{\mu}$ and $\boldsymbol{\tau}$ are (pseudo)marginal vectors. When $\boldsymbol{v}(\boldsymbol{x})$ is small, we can do better by applying the moment matching property from Proposition 1, which ensures that $\boldsymbol{\tau}(\hat{\boldsymbol{w}}) = \boldsymbol{\mu}(\boldsymbol{w}^*)$:

$$\begin{aligned}\|\boldsymbol{\mu}(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \boldsymbol{\tau}(\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x}))\|_2 &= \|[\boldsymbol{\mu}(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \boldsymbol{\mu}(\boldsymbol{w}^*)] - [\boldsymbol{\tau}(\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x})) - \boldsymbol{\tau}(\hat{\boldsymbol{w}})]\|_2 \\ &\leq \|\boldsymbol{\mu}(\boldsymbol{w}^* + \boldsymbol{v}(\boldsymbol{x})) - \boldsymbol{\mu}(\boldsymbol{w}^*)\|_2 + \|\boldsymbol{\tau}(\hat{\boldsymbol{w}} + \boldsymbol{v}(\boldsymbol{x})) - \boldsymbol{\tau}(\hat{\boldsymbol{w}})\|_2 \\ &\leq (L + R)\|\boldsymbol{v}(\boldsymbol{x})\|_2,\end{aligned} \tag{5.30}$$

applying Lemma 4 on the last line.

Taking the minimum of these two bounds and the expectation overall yields the bound. □

5.7 Discussion

To get intuition about the bound in Theorem 7, we consider two cases: the high noise regime ($\alpha \to 0$) and the low noise regime ($\alpha \to 1$). In the high noise regime, the amount of information being contributed by the input $\boldsymbol{x}$ is very small; as α goes to zero, so does $\|\boldsymbol{v}(\boldsymbol{x})\|_2$. Thus the bound shrinks to zero. In the particular case of $\alpha = 0$; that is, the input is complete noise, we can do no better than guessing the empirical marginals for $\boldsymbol{y}$ that we observed during training. Because of the moment matching property of our approximate variational inference, our estimates of the pseudomarginals are the exact empirical marginals in the training set. Thus, we make the same predictions that would be made using exact learning and inference. (Note that in fact we do not need clever algorithms at all in this case; we could simply use the empirical marginals directly.) For small $\alpha > 0$, we have used the stability property of our approximation to show that the small amount of information

added by the input $\boldsymbol{x}$ does not cause a dramatic change in the estimated marginals. Thus, our predictions cannot differ from the exact estimates by too much. As α rises, the stability bound becomes weaker, and at some point it is better to simply bound the difference between exact and approximate marginals by 1.

Interestingly, the above discussion of the high noise regime does not hold if our *approximate* inference procedure relies on the parameters $\boldsymbol{w}^*$ found using *exact* learning, because the moment matching property fails. That is, the inference method by itself does not offer any guarantees. It is only when paired with appropriately learned parameters that performance approaches the optimal result. To some extent this mirrors the intuitions in Chapter 4, but there the goal was to adapt known inference bounds to the task of learning. Here, the result arises directly from the connection between learning and inference, which suggests that even inference methods that do not have known guarantees on their own might be useful for learning results.

In the low noise regime, as $\alpha \to 1$, the value of $g_r(x)$ approaches x and thus the term $\sqrt{\frac{1}{n}\sum_{i \in V} |g_1(x_i) - g_2(x_i)|^4}$ tends to zero, as does the bound. Intuitively, when α is large, we do not need to know the underlying labels $\boldsymbol{y}$ since $\boldsymbol{x}$ is a good estimate of $\boldsymbol{z}$. Although the amount of information $\|\boldsymbol{v}(\boldsymbol{x})\|_2$ injected into the computation of marginals might be large and cause a poor approximation $\boldsymbol{\tau}$ of the marginals $\boldsymbol{\mu}$, it hardly matters, since we will mostly ignore them and predict $\boldsymbol{z} \approx \boldsymbol{x}$. For moderate noise levels, the bound is an interpolation between the two extremes. Our marginal approximations $\boldsymbol{\tau}$ are imperfect, but they only partially determine our predictions.

While the bound of Theorem 7 is a compelling result and offers useful intuition about the effects of noise on the performance of approximate learning and inference, the denoising setting is somewhat specific. In general, we would like to understand the behavior of approximate methods on structured classification problems as defined in Chapter 2. Unfortunately, it is not obvious how to extend the result shown here to the more general case for several reasons.

First, Theorem 7 assumes that the noise model is known. Though the bound is instructive in both high and low-noise regimes, we cannot describe the estimators in Equation (5.19) and Equation (5.22) without knowing which regime we are in; that is, without knowing α. If we think of "noise" more generally as a measure of the (inverse) amount of information provided by the input $\boldsymbol{x}$, then in most practical settings we can expect to see some examples that are less noisy, and others that are more noisy, without knowing in advance which is which.

Second, in the low noise regime, which is the most interesting for learning, the bound breaks down when the variances σ_r^2 of the mixture components become small. To see this, observe that if $\sigma_r^2 \ll (1 - \alpha^2)$, $g_r(x)$ is close to ν_r. This means that the bound is nearly as large as the squared difference between ν_1 and ν_2. This is a trivial bound in the sense that we can set every z_i to ν_1 or ν_2 at random and be assured of the same result. If the mixture components are tightly concentrated around the means, i.e., the goal is essentially to predict which of the components each z_i belongs to, the bound does not offer much guidance. Unfortunately, this is exactly the classification setting, where the identity of the label $\boldsymbol{y}$ is

what matters, and there is not in general a continuous semantic relationship between different labels.

Finally, it is not clear how to adapt the bound to a conditional MRF where we use a feature-based parameterization to model $P(\boldsymbol{y}|\boldsymbol{x})$. In the conditional scenario, the moment matching property implies that the learned parameters $\hat{\boldsymbol{w}}$ yield *joint* feature expectations matching the empirical distribution. Given a new input $\boldsymbol{x}$ at test time, this does not guarantee that the *conditional* marginals produced by the exact and approximate methods will agree in the high noise regime, in contrast to the denoising case. Indeed, for conditional models the notion of noise itself is somewhat vague, since we do not model the distribution of $\boldsymbol{y}$ without a reference input $\boldsymbol{x}$. In that sense the input always contains all of the signal; there can be no marginals without $\boldsymbol{x}$.

Chapter 6

Conclusions

We have seen two basic approaches to bounding the performance of efficient structured learning schemes. In the first, the relatively well-studied inference problem was used as a basis for deriving risk bounds on a scheme in which approximate inference is used for both learning the model parameters and making predictions on new inputs. In the second, a bound was derived directly for the learning scheme as a whole; the underlying inference method, used again for both learning and prediction, was not required to satisfy (and generally did not satisfy) an approximation bound. While the first approach allowed us to derive bounds for general learning of MRFs, allowing the use of any applicable inference approximation, the second showed that it may be possible to get bounds even for problems where we do not have inference guarantees. This seems like an especially interesting direction for future work.

The two main results presented in this report share several high-level intuitions. First, both require the use of closely related inference methods during learning and prediction. For the subgradient bound, we used the same LP-relaxation to compute the LAM during training and the MAP solution for testing. For the denoising bound, it was important to use the same variational approximation for estimating both the gradient of the normalization term for maximum likelihood learning and the label marginals for new inputs. The value of this coordination is intuitive: given that an inexact inference algorithm will make mistakes at test time, it seems natural that it would be advantageous to choose the model parameters with an awareness of those mistakes.

Second, both results depend on compatibility between the learning formulation and the type of inference approximation used as a subroutine. In the subgradient algorithm, LAM results are used to compute subgradients of the max-margin objective; thus approximate LAM inference leads directly to approximate subgradients and hence approximate convergence results. For the denoising bound, we relied on the fact that maximum likelihood parameter estimation combined with a convex dual approximation to the normalization term guarantees the moment-matching property. Again, there are strong intuitions for such coordination between learning and inference: the learning method cannot be expected to succeed if it relies on properties of exact inference that are not maintained by the approximation. For example, in previous work we showed that the combination of Perceptron learning with loopy belief

propagation inference can lead to divergence even when inference yields a 1.1-approximation and the data are separable [5].

Finally, the fundamental difference between inference for prediction and inference as a subroutine for learning is that at test time we typically assume the model parameters are fixed and "correct", while during learning they are in flux and do not necessarily yield scores that correlate with the quality of the solution. (Such correlation is really the end goal of the learning process, so almost by definition it cannot exist while learning is ongoing.) To use approximate inference for learning, then, we need some means of controlling the relationship between model scores and solution loss. The results presented here achieve this control in different ways. For the subgradient bound, the connection is a worst-case analysis in terms of the norms of the feature vectors and the optimal weight vector, as well as the number of outputs n. (This interpretation may not be obvious from the proofs shown in Chapter 4, but is made apparent by Lemma 1 in [6].) However, as we showed, this bound can be quite loose. In the denoising case the stability condition ensures that predictions made with similar sets of parameters (and thus for which scores are similar) have low loss. Unfortunately, this relationship breaks down as the mixture distribution because more peaked, as in classification. Thus, a central challenge going forward seems to be improving the understanding of this important relationship between model score and loss during learning.

The literature on approximate learning of structured prediction problems is still sparse, and the bounds shown here are among the earliest known results. Each comes with significant limitations. Nonetheless, collectively they offer some useful insight into the problem and the ways in which learning, inference for learning, and inference for prediction interact. They also highlight the major remaining challenges, and suggest broad theoretical principles that may lead to useful progress in future work.

6.1 Acknowledgements

Many thanks to João Graça for inspiring me by promising to quit smoking when I finished my WPE-II. I would also like to thank Jennann for telling me the number of part-of-speech tags in the Penn Treebank tag set.

Bibliography

[1] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. In *Advances in neural information processing systems*, page 359. MIT Press, 2002.

[2] C. Chekuri, S. Khanna, J. S. Naor, and L. Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 109–118. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2001.

[3] G. R. Grimmett. A theorem about random fields. *Bulletin of the London Mathematical Society*, 5(13):81–84, 1973.

[4] J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.

[5] A. Kulesza and F. Pereira. Structured learning with approximate inference. *Advances in Neural Information Processing Systems*, 20, 2007.

[6] A. F. T. Martins, N. A. Smith, and E. P. Xing. Polyhedral outer approximations with application to natural language parsing. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

[7] A.F.T. Martins, N.A. Smith, and E.P. Xing. Concise integer linear programming formulations for dependency parsing. In *Proc. of Annual Meeting of the Association for Computational Linguistics (ACLIJCNLP09), Singapore*, 2009.

[8] O. Meshi, A. Jaimovich, A. Globerson, and N. Friedman. Convexifying the Bethe Free Energy. In *Conference on Uncertainty in Artifical Intelligence (UAI)*, 2009.

[9] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[10] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich. Subgradient methods for structured prediction. In *AI Statistics*, 2007.

[11] D. Roth and W. Yih. A linear programming formulation for global inference in natural language tasks. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8, 2004.

[12] Charles Sutton and Andrew McCallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, 2004.

[13] Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative Markov networks. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 102, 2004.

[14] Martin J. Wainwright. Estimating the "wrong" graphical model: Benefits in the computation-limited setting. *Journal of Machine Learning Research*, 7:1829–1859, 2006.

[15] M.J. Wainwright, T.S. Jaakkola, and A.S. Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.