Max-Margin Markov Networks

Ben Taskar TASKAR@CS.BERKELEY.EDU
Division of Computer Science
University of California
Berkeley, CA 94720, USA

Carlos Guestrin GUESTRIN@CS.CMU.EDU
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Vassil Chatalbashev VASCO@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Daphne Koller KOLLER@CS.STANFORD.EDU
Department of Computer Science
Stanford University
Stanford, CA 94305, USA

Editor: Leslie Pack Kaelbling

Abstract

Standard classification problems involve assigning a single label to an object given its features. We address a more complex setting of *structured classification*, where multiple and interdependent labels must be assigned to objects with sequential, spatial, or relational structure. Probabilistic graphical models, such as Markov networks, have been widely used to represent correlations between labels by exploiting the structure of such tasks. Traditionally, graphical models are estimated by maximizing likelihood or conditional likelihood. An alternative estimation criterion, based on large-margin principle underlying support vector machines, has been recently proposed for sequence models (Collins, 2001; Altun et al., 2003), resulting in an *exponential-size* quadratic programming formulation. We develop *maximum margin Markov (M^3) networks*, which efficiently incorporate kernels and capture label correlations in structured data, using an explicit *polynomial-size* quadratic programming formulation. We present an efficient estimation algorithm that exploits dynamic programming for learning M^3 networks. In an important class of models (binary networks of arbitrary topology with attractive potentials), likelihood-based estimation is believed to be intractable. However, our margin-based estimation provides an exact polynomial-time solution. We provide a novel margin-based generalization bound for structured domains and show experiments on the tasks of handwritten character recognition, web page classification and 3D object segmentation that demonstrate very significant gains over previous approaches.

Keywords: Markov networks, support vector machines, structured classification, kernel methods, large-margin classification

1. Introduction

In standard classification, our goal is to learn how to label an instance using a set of discrete categories. Two successful and popular approaches to this task are margin-based models, such as support vector machines, and conditional probabilistic models, such as logistic regression. Probabilistic models offer good estimates of conditional probability of label given the features and an established set of tools for dealing with background knowledge using priors, as well as partially observed data. Margin-based methods offer sparse, more efficient learning in high-dimensional feature spaces via kernels and strong theoretical generalization guarantees.

In many cases, however, we need to label a set of inter-related instances. For example: optical character recognition (OCR) or part-of-speech tagging both involve labeling an entire sequence of elements into some number of classes; image segmentation involves labeling all of the pixels in an image; and collective webpage classification involves labeling an entire set of interlinked webpages. These problems involve labeling a set of related objects that exhibit *local* correlations: between labels of adjacent letter images in handwriting recognition, neighboring pixels in an image, linked pages in a website. In all of these cases, we need to assign multiple labels simultaneously, leading to a classification problem that has an exponentially large set of joint labels. We refer to this setting as *structured classification.*

Probabilistic graphical models offer a solution in this setting: we can define and learn a joint probabilistic model over the set of label variables. For example, we can learn a hidden Markov model, or a conditional random field (CRF) (Lafferty et al., 2001) over the labels and features of a sequence, and then use a probabilistic inference algorithm (such as the Viterbi algorithm) to classify these instances *collectively*, finding the most likely joint assignment to all of the labels simultaneously.

Markov networks are extensively used to model complex sequential, spatial, and relational interactions in prediction problems arising in many fields (Pearl, 1988; Cowell et al., 1999). They compactly represent complex joint distributions of the label variables by modeling their local interactions. Such models are encoded by a graph, whose nodes represent the different object labels, and whose edges represent and quantify direct dependencies between them.

We address the problem of max-margin estimation the parameters of Markov networks for such structured classification problems. We show a compact convex formulation that seamlessly integrates kernels with graphical models. Recent margin-based formulations for sequence models involve an *exponential-size* quadratic program (Collins, 2001; Altun et al., 2003). In this paper, we show an explicit *polynomial-size* quadratic programming formulation for learning *maximum margin Markov (M^3) networks.* This formulation is exact for networks where inference is tractable (low-treewidth graphs, including sequences, trees, and other common structures). For non-triangulated networks, we provide an approximate formulation based on the relaxation used by belief propagation algorithms (Yedidia et al., 2000). We present an efficient online-style estimation algorithm called structured SMO that exploits dynamic programming and scales to very large datasets.

We also define an important subclass of Markov networks for which margin-based estimation provides an exact solution for arbitrary network topology. This subclass, called *associative Markov networks (AMNs)*, contains networks of discrete variables with K labels and arbitrary-size clique potentials with K parameters that favor the same labels for all variables in the clique. Such positive interactions capture the "guilt by association" pattern of reasoning present in many domains, in which connected ("associated") variables tend to have the same label. AMNs are a natural fit object recognition and segmentation, webpage classification, and many other applications. We show that in binary AMNs, for which likelihood-based estimation is believed to be intractable, our margin-based framework provides a polynomial-time solution. To our knowledge, our method is the first to allow exact training Markov networks of arbitrary connectivity and topology. We present an AMN-based method for object segmentation from 3D range data. By constraining the class of Markov networks to AMNs, our models can be learned efficiently and at run-time, scale up to tens of millions of nodes and edges. The proposed learning formulation effectively and directly learns to exploit a large set of complex surface and volumetric features, while balancing the spatial coherence modeled by the AMN.

We provide a novel margin-based theoretical bound for generalization in structured domains. Unlike previous results (Collins, 2001), our bound grows logarithmically rather than linearly with the number of label variables. We present experiments on the tasks of handwriting recognition, web page classification and 3D terrain segmentation that demonstrate very significant gains over previous approaches.

The rest of this paper is organized as follows. In Sec. 2, we briefly review basic concepts in classification and describe logistic regression and support vector machines, two approaches that highlight differences between likelihood and margin based estimation we explore in this paper. We define structured classification and graphical models in Sec. 3, and review inference and learning in general and associative Markov networks in Sec. 4 and Sec. 5. We present our general framework in Sec. 6 and apply it to general Markov networks (Sec. 7) and AMNs (Sec. 8). In Sec. 9, we provide a novel margin-based generalization bound and describe an efficient structured SMO algorithm for solving the max-margin QP in general Markov networks in Sec. 10. We present experiments in Sec. 11 and conclude with related work (Sec. 12) and discussion (Sec. 13).

2. Classification

In standard classification, we seek a function $h : \mathcal{X} \mapsto \mathcal{Y}$ that maps inputs $\mathbf{x} \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$. The input space $\mathcal{X}$ is an arbitrary set (often $\mathcal{X} = \mathbb{R}^n$), while the output space $\mathcal{Y}$ is a small number of discrete classes. In handwritten character recognition, for example, $\mathcal{X}$ is the set of images of letters and $\mathcal{Y}$ is the alphabet (see Fig. 1). Our input is a set of m i.i.d. (independent and identically distributed) samples $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}$ drawn from a fixed but unknown distribution D over $\mathcal{X} \times \mathcal{Y}$. Our output is a hypothesis h such that $h(\mathbf{x})$ will approximate y on new samples from the distribution $(\mathbf{x}, y) \sim D$.

Learning algorithms can be distinguished among several dimensions, chief among them is the hypothesis class $\mathcal{H}$ of functions h. We will concentrate on the generalized linear model class for several reasons, including accuracy, efficiency, and extensibility to more complex structured classification tasks we consider next. The second crucial dimension of a learning algorithm is the criterion for selection of h from $\mathcal{H}$. We arrive at such a criterion by quantifying what it means for $h(\mathbf{x})$ to approximate y. The risk functional $\mathcal{R}_D^{\ell}[(h)]$ measures the expected error of the approximation:

$$\mathcal{R}_D^\ell[h] = \mathbf{E}_{(\mathbf{x},y)\sim D}[\ell(\mathbf{x}, y, h(\mathbf{x}))], \tag{1}$$

where the loss function $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathrm{I\!R}^+$ measures the penalty for predicting $h(\mathbf{x})$ on the sample $(\mathbf{x}, y)$. In general, we assume that $\ell(\mathbf{x}, y, \hat{y}) = 0$ if $y = \hat{y}$. The standard loss function for classification is 0/1 loss

$$\ell^{0/1}(\mathbf{x}, y, h(\mathbf{x})) \equiv \mathbb{1}(y \neq h(\mathbf{x})),$$

where $\mathbb{1}(\cdot)$ denotes the indicator function, that is, $\mathbb{1}(\text{true}) = 1$ and $\mathbb{1}(\text{false}) = 0$.

Since we do not generally know the distribution D, we estimate the risk of h using its empirical risk $\mathcal{R}_S^\ell$, computed on the training sample S:

$$\mathcal{R}_S^\ell[h] = \frac{1}{m}\sum_{i=1}^{m} \ell(\mathbf{x}^{(i)}, y^{(i)}, h(\mathbf{x}^{(i)})) = \frac{1}{m}\sum_{i=1}^{m} \ell_i(h(\mathbf{x}^{(i)})), \tag{2}$$

where we abbreviate $\ell(\mathbf{x}^{(i)}, y^{(i)}, h(\mathbf{x}^{(i)})) = \ell_i(h(\mathbf{x}^{(i)}))$. For 0/1 loss, $\mathcal{R}_S^\ell[h]$ is simply the proportion of training examples that h misclassifies. $\mathcal{R}_S^\ell[h]$ is often called the training error or training loss.

If our set of hypotheses, $\mathcal{H}$, is large enough, we will be able to find h that has zero or very small empirical risk. However, simply selecting a hypothesis with lowest risk

$$h^* = \arg\min_{h\in\mathcal{H}} \mathcal{R}_S^\ell[h],$$

is generally not a good idea. For example, if $\mathcal{X} = \mathrm{I\!R}, \mathcal{Y} = \mathrm{I\!R}$ and $\mathcal{H}$ includes all polynomials of degree $m-1$, we can always find a polynomial h that passes through all the sample points $(x^{(i)}, y^{(i)})$, $i = (1, ..., m)$ assuming that all the $x^{(i)}$ are unique. This polynomial is very likely to overfit the training data, that is, it will have zero empirical risk, but high actual risk. The key to selecting a good hypothesis is to trade-off complexity of class $\mathcal{H}$ (e.g. the degree of the polynomial) with the error on the training data as measured by empirical risk $\mathcal{R}_S^\ell$. For a vast majority of supervised learning algorithms, this fundamental balance is achieved by minimizing the weighted combination of the two criteria:

$$h^* = \arg\min_{h\in\mathcal{H}} \left(\mathcal{D}[h] + C\mathcal{R}_S^\ell[h]\right), \tag{3}$$

where $\mathcal{D}[h]$ measures the inherent dimension or complexity of h, and $C \geq 0$ is a trade-off parameter. We will not go into derivation of various complexity measures $\mathcal{D}[h]$ here, but simply adopt the standard measures as needed and refer the reader to Vapnik (1995); Devroye et al. (1996); Hastie et al. (2001) for details. The term $\mathcal{D}[h]$ is often called regularization.

Depending on the complexity of the class $\mathcal{H}$, the search for the optimal h^* in Eq. (3) may be a daunting task[1]. For many classes, for example decision trees and multi-layer neural networks, it is intractable (Bishop, 1995; Quinlan, 2001), and we must resort to approximate,

---

[1] For classification, minimizing the objective with the usual 0/1 training error is generally a very difficult problem with multiple maxima for most realistic $\mathcal{H}$. See discussion in the next section about approaches to dealing with 0/1 loss.

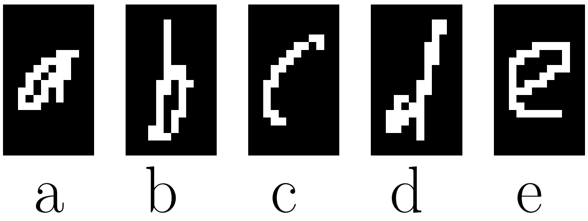


Figure 1: Handwritten character recognition: sample letters from Kassel (1995) data set.

greedy optimization methods. For these intractable classes, the search procedure used by the learning algorithm is crucial. Below however, we will concentrate on models where the optimal h^* can be found efficiently using convex optimization in polynomial time. Hence, the learning algorithms we consider are completely characterized by the hypothesis class $\mathcal{H}$, the loss function ℓ, and the regularization $\mathcal{D}[h]$.

2.1 Generalized linear models

We consider the generalized linear family of hypotheses $\mathcal{H}$. Given n real-valued basis functions $f_j : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, a hypothesis $h_{\mathbf{w}} \in \mathcal{H}$ is defined by a set of n coefficients $w_j \in \mathbb{R}$ such that:

$$h_{\mathbf{w}}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{n} w_j f_j(\mathbf{x}, y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y). \tag{4}$$

We assume that ties in the arg max are broken using some arbitrary but fixed rule. As we discuss below, this class of hypotheses is very rich and includes many standard models. The formulation in Eq. (4) of the hypothesis class in terms of an optimization procedure will become crucial to extending supervised learning techniques to cases where the output space $\mathcal{Y}$ is more complex.

Consider the character recognition example in Fig. 1. Our input $\mathbf{x}$ is a vector of pixel values of the image and y is the alphabet $\{a, \ldots, z\}$. We might have a basis function $f_j(\mathbf{x}, y) = \mathbb{1}(\mathbf{x}_{row,col} = on \wedge y = char)$ for each possible (row, col) and $char \in \mathcal{Y}$, where $\mathbf{x}_{row,col}$ denotes the value of pixel (row, col). Since different letters tend to have different pixels turned on, this very simple model captures enough information to perform reasonably well.

The most common loss for classification is 0/1 loss. Minimizing the 0/1 risk is generally a very difficult problem with multiple maxima for any large class $\mathcal{H}$. The standard solution is minimizing an upper bound on the 0/1 loss, $\overline{\ell}(\mathbf{x}, y, h(\mathbf{x})) \geq \ell(\mathbf{x}, y, h(\mathbf{x}))$. (In addition to computational advantages of this approach, there are statistical benefits of minimizing a *convex* upper bound (Bartlett et al., 2003)). Two of the primary classification methods we consider, logistic regression and support vector machines, differ primarily in their choice of the upper bound on the training 0/1 loss. The regularization $\mathcal{D}[h_{\mathbf{w}}]$ for the linear family is typically the norm of the parameters $||\mathbf{w}||_p$ for $p = 1, 2$. Intuitively, a zero, or small weight w_j implies that the hypothesis $h_{\mathbf{w}}$ does not depend on the value of $f_j(\mathbf{x}, y)$ and hence is simpler than a $h_{\mathbf{w}}$ with a large weight w_j.

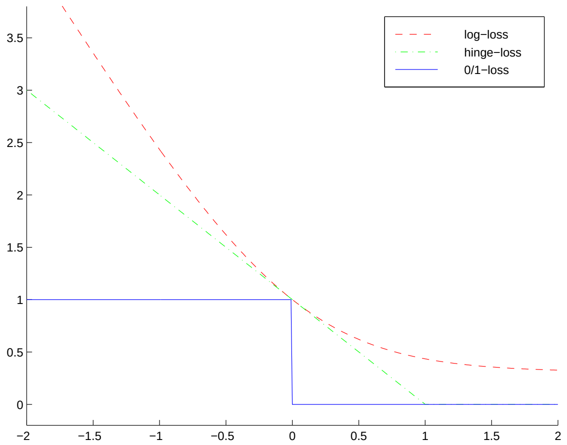


Figure 2: 0/1-loss upper bounded by log-loss and hinge-loss. Horizontal axis shows $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, y) - \max_{y' \neq y} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y')$, where y is the correct label for $\mathbf{x}$, while the vertical axis show the value of the associated loss. The log-loss is shown up to an additive constant for illustration purposes.

2.2 Logistic regression

In logistic regression, we assign a probabilistic interpretation to the hypothesis $h_\mathbf{w}$ as defining a conditional distribution:

$$P_\mathbf{w}(y \mid \mathbf{x}) = \frac{1}{Z_\mathbf{w}(\mathbf{x})} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, y)\}, \tag{5}$$

where $Z_\mathbf{w}(\mathbf{x}) = \sum_{y \in \mathcal{Y}} \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, y)\}$. The optimal weights are selected by maximizing the conditional likelihood of the data (minimizing the log-loss) with some regularization. This approach is called the (regularized) conditional maximum likelihood estimation. Common choices for regularization are 1 or 2-norm regularization on the weights; we use 2-norm below:

$$\min \quad \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \log Z_\mathbf{w}(\mathbf{x}^{(i)}) - \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}), \tag{6}$$

where C is a user-specified constant the determines the trade-off between regularization and likelihood of the data. The log-loss $\log Z_\mathbf{w}(\mathbf{x}) - \mathbf{w}^\top \mathbf{f}(\mathbf{x}, y)$ is an upper bound (up to a constant) on the 0/1 loss $\ell^{0/1}$ (see Fig. 2).

2.3 Logistic dual and maximum entropy

The objective function is convex in the parameters $\mathbf{w}$, so we have an unconstrained (differentiable) convex optimization problem. The gradient with respect to $\mathbf{w}$ is given by:

$$\mathbf{w} + C \sum_i \mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{x}^{(i)}, y)] - \mathbf{f}_i(\mathbf{x}^{(i)}, y^{(i)}) = \mathbf{w} - C \sum_i \mathbf{E}_{i,\mathbf{w}}[\Delta \mathbf{f}_i(y)],$$

where $\mathbf{E}_{i,\mathbf{w}}[f(y)] = \sum_y f(y) P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)})$ is the expectation under the conditional distribution $P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)})$ and $\Delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, y)$. Ignoring the regularization term, the gradient is zero when the basis function expectations are equal to the basis functions evaluated on the labels $y^{(i)}$. It can be shown (Cover and Thomas, 1991) that the dual of the maximum likelihood problem (without regularization) is the maximum entropy problem:

$$\begin{aligned} \max \quad & -\sum_{i,y} P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)}) \log P_{\mathbf{w}}(y \mid \mathbf{x}^{(i)}) \\ \text{s.t.} \quad & \mathbf{E}_{i,\mathbf{w}}[\Delta\mathbf{f}_i(y)] = 0, \quad \forall i. \end{aligned} \tag{7}$$

We can interpret logistic regression as trying to match the empirical basis function expectations while maintaining a high entropy conditional distribution $P_{\mathbf{w}}(y \mid \mathbf{x})$.

2.4 Support vector machines

Support vector machines (Vapnik, 1995) select the weights based on the "margin" of confidence of $h_{\mathbf{w}}$. In the multi-class SVM formulation (Weston and Watkins, 1998; Crammer and Singer, 2001), the margin on example i quantifies by how much the true label "wins" over the wrong ones:

$$\gamma_i = \frac{1}{||\mathbf{w}||} \min_{y \neq y^{(i)}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}) - \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, y) = \frac{1}{||\mathbf{w}||} \min_{y \neq y^{(i)}} \mathbf{w}^\top \Delta\mathbf{f}_i(y),$$

where $\Delta\mathbf{f}_i(y) = \mathbf{f}(\mathbf{x}^{(i)}, y^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, y)$. Maximizing the smallest such margin (and allowing for negative margins) is equivalent to solving the following quadratic program:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\ \text{s.t.} \quad & \mathbf{w}^\top \Delta\mathbf{f}_i(y) \geq \ell^{0/1}(y) - \xi_i, \quad \forall i, \ \forall y \in \mathcal{Y}. \end{aligned} \tag{8}$$

Note that the slack variable ξ_i is constrained to be positive in the above program since $\mathbf{w}^\top \Delta\mathbf{f}_i(y^{(i)}) = 0$ and $\ell^{0/1}(y^{(i)}) = 0$. We can also express ξ_i as $\max_y \ell_i^{0/1}(y) - \mathbf{w}^\top \Delta\mathbf{f}_i(y)$, and the optimization problem Eq. (8) in a form similar to Eq. (6):

$$\min \quad \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \max_y [\ell_i^{0/1}(y) - \mathbf{w}^\top \Delta\mathbf{f}_i(y)]. \tag{9}$$

The hinge-loss $\max_y [\ell_i^{0/1}(y) - \mathbf{w}^\top \Delta\mathbf{f}_i(y)]$ is also an upper bound on the 0/1 loss $\ell^{0/1}$ (see Fig. 2).

2.5 SVM dual and kernels

The form of the dual of Eq. (8) is crucial to efficient solution of SVM and the ability to use a high or even infinite dimensional set of basis functions via kernels.

$$\begin{aligned} \max \quad & \sum_{i,y} \alpha_i(y) \ell_i^{0/1}(y) - \frac{1}{2} \left|\left| \sum_{i,y} \alpha_i(y) \Delta\mathbf{f}_i(y) \right|\right|^2 \\ \text{s.t.} \quad & \sum_y \alpha_i(y) = C, \ \forall i; \qquad \alpha_i(y) \geq 0, \quad \forall i, y. \end{aligned} \tag{10}$$

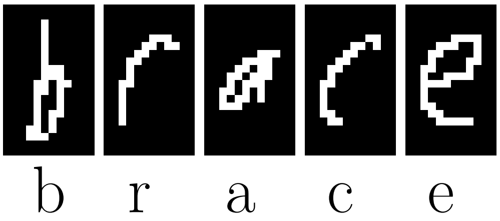


Figure 3: Handwritten word recognition: sample from Kassel (1995) data set.

In the dual, the $\alpha_i(y)$ variables correspond to the $\mathbf{w}^\top \Delta\mathbf{f}_i(y) \geq \ell^{0/1}(y) - \xi_i$ constraints in the primal Eq. (8). The solution to the dual α^* gives the solution to the primal as a weighted combination of basis functions of examples:

$$\mathbf{w}^* = \sum_{i,y} \alpha_i^*(y) \Delta\mathbf{f}_i(y).$$

The pairings of examples and incorrect labels, (i, y), that have non-zero $\alpha_i^*(y)$, are called support vectors.

An important feature of the dual formulation is that the basis functions $\mathbf{f}$ appear only as dot products. Expanding the quadratic term, we have:

$$\left|\left|\sum_{i,y} \alpha_i(y) \Delta\mathbf{f}_i(y)\right|\right|^2 = \sum_{i,y} \sum_{j,\bar{y}} \alpha_i(y)\alpha_j(\bar{y}) \Delta\mathbf{f}_i(y)^\top \Delta\mathbf{f}_j(\bar{y}).$$

Hence, as long as the dot product $\mathbf{f}(\mathbf{x}, y)^\top \mathbf{f}(\bar{\mathbf{x}}, \bar{y})$ can be computed efficiently, we can solve Eq. (10) independently of the actual dimension of $\mathbf{f}$. Note that at classification time, we also do not need to worry about the dimension of $\mathbf{f}$ since:

$$\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \bar{y}) = \sum_{i,y} \alpha_i(y) \Delta\mathbf{f}_i(y)^\top \mathbf{f}(\mathbf{x}, \bar{y}) = \sum_{i,y} \alpha_i(y)[\mathbf{f}(\mathbf{x}^{(i)}, y^{(i)})^\top \mathbf{f}(\mathbf{x}, \bar{y}) - \mathbf{f}(\mathbf{x}^{(i)}, y)^\top \mathbf{f}(\mathbf{x}, \bar{y})].$$

For example, we might have basis functions that are polynomial of degree d in terms of image pixels, $f_j(\mathbf{x}, y) = \mathbb{I}(\mathbf{x}_{row_1, col_1} = on \wedge \ldots \wedge \mathbf{x}_{row_d, col_d} = on \wedge y = char)$ for each possible $(row_1, col_1) \ldots (row_d, col_d)$ and $char \in \mathcal{Y}$. Computing this polynomial kernel can be done independently of the dimension d, even though the number of basis functions grows exponentially with d (Vapnik, 1995).

In fact, logistic regression can also be kernelized. However, the hinge loss formulation usually produces sparse solutions in terms of the number of support vectors, while solutions to the corresponding kernelized log-loss problem are generally non-sparse (all examples are support vectors) and require approximations for even relatively small datasets (Wahba et al., 1993; Zhu and Hastie, 2001).

3. Structured classification

Consider once more the problem of character recognition. In fact, a more natural and useful task is recognizing words and entire sentences. Fig. 3 shows an example handwritten word “brace.” Distinguishing between the second letter and fourth letter (‘r’ and ‘c’) *in isolation* is far from trivial, but in the context of the surrounding letters that together form a word, this task is much less error-prone for humans and should be for computers as well. However, transferring margin-based techniques from standard classification has met with several computational challenges which our work addresses.

We consider prediction problems in which the output is not a single discrete value y, but a set of values $\mathbf{y} = (y_1, \ldots, y_L)$, for example an entire sequence of L characters. We assume that the input $\mathbf{x}$ is pre-segmented into L images corresponding to each letter. The output space $\mathcal{Y}(\mathbf{x}) = \mathcal{Y}_1 \times \ldots \times \mathcal{Y}_L$ is a product of output spaces of single variables. In word recognition, each $\mathcal{Y}_j$ is the alphabet, while $\mathcal{Y}(\mathbf{x})$ is the set of L-letter sequences. The joint spaces we consider have a rich correlation structure. In word recognition, we observe that the letter ‘q’ (almost) never follows ‘z’ in English words, and in general, consecutive letters are highly correlated. In 2D image or 3D range data segmentation, the goal is to break up a set of a set of pixels or points into coherent regions that correspond to object types: for example, trees, buildings, roads (see Fig. 22). Our output space again is the joint space, where each $\mathcal{Y}_j$ is a small number of object types. The correlations in this problem are spatial, since physically adjacent points often belong to the same object type. Similarly, when classifying a set of hyperlinked pages of a website, our output space is the joint space of all page label assignments, and the correlations are strongest between linked pages. We refer to joint spaces with such local correlations as structured. Structured models, like Markov networks we describe in the next section, predict the outputs *jointly*, exploiting such correlations in the output space to make more globally informed decisions.

The range of prediction problems these broad definitions encompass is immense, arising in fields as diverse as natural language analysis, machine vision, and computational biology, to name a few. The class of structured models $\mathcal{H}$ we consider is essentially of the same form as in previous chapter, except that y has been replaced by a vector $\mathbf{y}$:

$$h_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}), \tag{11}$$

where as before $\mathbf{f}(\mathbf{x}, \mathbf{y})$ is a vector of functions $\mathbf{f} : \mathcal{X} \times \mathcal{Y} \mapsto \mathrm{I\!R}^n$. We assume that the output space $\mathcal{Y}(\mathbf{x})$, the set of label variables to be predicted and their possible assignments, is given by some deterministic procedure. This formulation is very general. Clearly, for many models $\mathbf{f}$, finding the optimal $\mathbf{y}$ is intractable. For the most part, we will restrict our attention to models where this inference problem can be solved in polynomial time. This includes, Markov networks with special structure (low-treewidth or binary associative). In other cases, we use an *approximate* polynomial time inference procedure.

3.1 Probabilistic models: generative and conditional

Probabilistic models can be subdivided into generative and conditional with respect to the prediction task. A generative model assigns a normalized joint distribution $P(\mathbf{x}, \mathbf{y})$, or more generally, a density $p(\mathbf{x}, \mathbf{y})$, to the input and output space $\mathcal{X} \times \mathcal{Y}$ with

$$p(\mathbf{x}, \mathbf{y}) \geq 0, \qquad \int_{\mathbf{x} \in \mathcal{X}} \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} p(\mathbf{x}, \mathbf{y}) = 1.$$

A conditional model assigns a normalized distribution $p(\mathbf{y} \mid \mathbf{x})$ only over the output space $\mathcal{Y}$ with

$$P(\mathbf{y} \mid \mathbf{x}) \geq 0, \qquad \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} P(\mathbf{y} \mid \mathbf{x}) = 1 \qquad \forall \mathbf{x} \in \mathcal{X}.$$

Probabilistic interpretation of the model offers well-understood semantics and an immense toolbox of methods for inference and learning. It also provides an intuitive measure of confidence in the predictions of a model in terms of conditional probabilities. In addition, generative models are typically structured to allow very efficient maximum likelihood learning. A very common class of generative models is the exponential family:

$$p(\mathbf{x}, \mathbf{y}) \propto \exp\{\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})\}.$$

For exponential families, the maximum likelihood parameters $\mathbf{w}$ with respect to the joint distribution can be computed in closed form using the empirical basis function expectations $\mathbf{E}_S[\mathbf{f}(\mathbf{x}, \mathbf{y})]$ (DeGroot, 1970; Hastie et al., 2001).

Of course, this efficiency comes at a price. Any model is an approximation to the true distribution underlying the data. A generative model must make simplifying assumptions (more precisely, independence assumptions) about the entire $p(\mathbf{x}, \mathbf{y})$, while a conditional model makes many fewer assumption by focusing on $P(\mathbf{y} \mid \mathbf{x})$. Because of this, by optimizing the model to fit the joint distribution $p(\mathbf{x}, \mathbf{y})$, we may be tuning the approximation away from optimal conditional distribution $P(\mathbf{y} \mid \mathbf{x})$, which we use to make the predictions. Given sufficient data, the conditional model will learn the best approximation to $P(\mathbf{y} \mid \mathbf{x})$ possible using $\mathbf{w}$, while the generative model $p(\mathbf{x}, \mathbf{y})$ will not necessarily do so. Typically, however, generative models actually need fewer samples to converge to a good estimate of the joint distribution than conditional models need to accurately represent the conditional distribution. In a regime with very few training samples (relative to the number of parameters $\mathbf{w}$), generative models may actually outperform conditional models (Ng and Jordan, 2001).

3.2 Prediction models: normalized and unnormalized

Probabilistic semantics are certainly not necessary for a good predictive model if we are simply interested in the optimal prediction (the arg max in Eq. (11)). As we discussed in the previous section, support vector machines, which do not represent a conditional distribution, typically perform as well or better than logistic regression (Vapnik, 1995; Cristianini and Shawe-Taylor, 2000).

In general, we can often achieve higher accuracy models when we do not learn a normalized distribution over the outputs, but concentrate on the margin or decision boundary, the difference between the optimal $\mathbf{y}$ and the rest. Even more importantly, in the case of associative Markov networks we discuss below, normalizing the model, which requires summing over the entire $\mathcal{Y}$, is (believed to be) intractable. This fact makes standard maximum likelihood estimation infeasible. The learning method we advocate in this paper circumvents this problem by requiring only the maximization problem to be tractable. We still heavily rely on the representation and inference tools familiar from probabilistic models for the construction of and prediction in unnormalized models, but largely dispense with the probabilistic interpretation when needed.

4. Markov networks

Markov networks provide a rich framework for modeling the structure of correlations in many domains (Pearl, 1988; Cowell et al., 1999). The models treat the inputs and outputs as random variables $\mathbf{X}$ with domain $\mathcal{X}$ and $\mathbf{Y}$ with domain $\mathcal{Y}$ and compactly define a conditional density $p(\mathbf{Y} \mid \mathbf{X})$ or distribution $P(\mathbf{Y} \mid \mathbf{X})$ (we concentrate here on the conditional Markov networks or CRFs (Lafferty et al., 2001)). The advantage of a *graphical* framework is that it can compactly exploit sparseness in the correlations between outputs Y. The graphical structure of the models encodes the *qualitative* aspects of the distribution: direct dependencies as well as conditional independencies. The *quantitative* aspect of the model is defined by the *potentials* that are associated with nodes and cliques of the graph. Before a formal definition, consider a first-order Markov chain a model for the word recognition task. In Fig. 4, the nodes are associated with output variables Y_i and the edges correspond to direct dependencies or correlations. We do not explicitly represent the inputs $\mathbf{X}$ in the figure. For example, the model encodes that Y_j is conditionally independent of the rest of the variables given Y_{j-1}, Y_{j+1}. Intuitively, adjacent letters in a word are highly correlated, but the first-order model is making the assertion (which is certainly an approximation) that once the value of a letter Y_j is known, the correlation between a letter Y_b before j and a letter Y_a after j is negligible. More precisely, we use a model where

$$P(Y_b \mid Y_j, Y_a, \mathbf{x}) = P(Y_b \mid Y_j, \mathbf{x}), \quad P(Y_a \mid Y_j, Y_b, \mathbf{x}) = P(Y_a \mid Y_j, \mathbf{x}), \quad b < j < a.$$

For the purposes of finding the most likely $\mathbf{y}$, this conditional independence property means that the optimization problem is decomposable: given that $Y_j = y_j$, it suffices to *separately* find the optimal subsequence from 1 to j ending with y_j, and the optimal subsequence starting with y_j from j to L.

4.1 Representation

The structure of a Markov network is defined by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes are associated with variables $\mathcal{V} = \{Y_1, \ldots, Y_L\}$. A clique is a set of nodes $c \subseteq \mathcal{V}$ that form a fully connected subgraph (every two nodes are connected by an edge). Note that each subclique of a clique is also a clique, and we consider each node a singleton clique. In the chain network in Fig. 4, the cliques are simply the nodes and the edges: $\mathcal{C}(\mathcal{G}) = \{\{Y_1\}, \ldots, \{Y_5\}, \{Y_1, Y_2\}, \ldots, \{Y_4, Y_5\}\}$. We denote the set of variables in a clique c as $\mathbf{Y}_c$, an assignment of variables in the clique as $\mathbf{y}_c$ and the space of all assignments to the clique as $\mathcal{Y}_c$. We focus on discrete output spaces $\mathcal{Y}$ below, but many of the same representation and inference concepts translate to continuous domains. No assumption is made about $\mathcal{X}$.

Definition 4.1 *A Markov network is defined by an undirected graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *and a set of potentials* $\Phi = \{\phi_c\}$*. The nodes are associated with variables* $\mathcal{V} = \{Y_1, \ldots, Y_L\}$*. Each clique* $c \in \mathcal{C}(\mathcal{G})$ *is associated with a potential* $\phi_c(\mathbf{x}, \mathbf{y}_c)$ *with* $\phi_c : \mathcal{X} \times \mathcal{Y}_c \mapsto \mathbb{R}^+$*, which specifies a non-negative value for each assignment* $\mathbf{y}_c$ *to variables in* $\mathbf{Y}_c$ *and any input* $\mathbf{x}$*. The Markov network* $(\mathcal{G}, \Phi)$ *defines a conditional distribution:*

$$P(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}(\mathcal{G})} \phi_c(\mathbf{x}, \mathbf{y}_c),$$

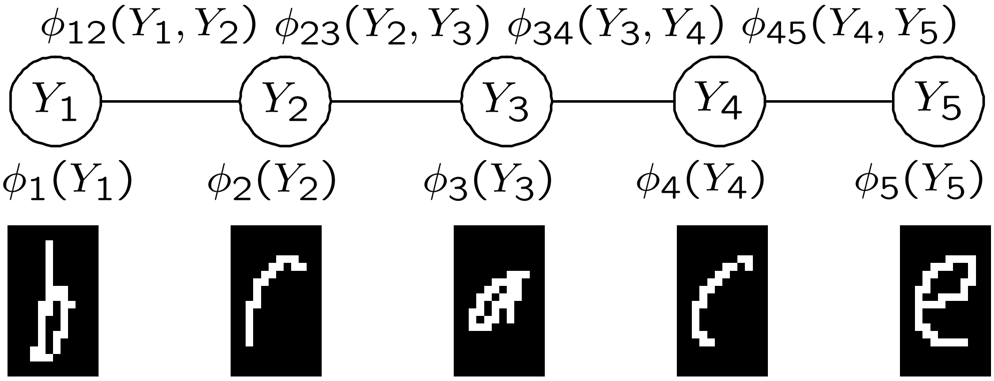


Figure 4: First-order Markov chain: $\phi_i(Y_i)$ are node potentials, $\phi_{i,i+1}(Y_i, Y_{i+1})$ are edge potentials (dependence on $\mathbf{x}$ is not shown).

where $\mathcal{C}(\mathcal{G})$ *is the set of all the cliques of the graph and* $Z(\mathbf{x})$ *is the partition function given by* $Z(\mathbf{x}) = \sum_{\mathbf{y}\in\mathcal{Y}} \prod_{c\in\mathcal{C}(\mathcal{G})} \phi_c(\mathbf{x}, \mathbf{y}_c)$.

In our example Fig. 4, we have node and edge potentials. Intuitively, the node potentials quantify the correlation between the input $\mathbf{x}$ and the value of the node, while the edge potentials quantify the correlation between the pair of adjacent output variables as well as the input $\mathbf{x}$. Potentials do not have a *local* probabilistic interpretation, but can be thought of as defining an unnormalized score for each assignment in the clique. Conditioned on the image input, appropriate node potentials in our network should give high scores to the correct letters (‘b’,‘r’,‘a’,‘c’,‘e’), though perhaps there would be some ambiguity with the second and fourth letter. For simplicity, assume that the edge potentials would not depend on the images, but simply should give high scores to pairs of letters that tend to appear often consecutively. Multiplied together, these scores should favor the correct output “brace”.

In fact, a Markov network is a generalized log-linear model, since the potentials $\phi_c(\mathbf{x}_c, \mathbf{y}_c)$ could be represented (in log-space) as a sum of basis functions over $\mathbf{x}, \mathbf{y}_c$:

$$\phi_c(\mathbf{x}_c, \mathbf{y}_c) = \exp\left[\sum_{k=1}^{n_c} w_{c,k} f_{c,k}(\mathbf{x}, \mathbf{y}_c)\right] = \exp\left[\mathbf{w}_c^\top \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c)\right]$$

where n_c is the number of basis functions for the clique c. Hence the log of the conditional probability is given by:

$$\log P(\mathbf{y} \mid \mathbf{x}) = \sum_{c\in\mathcal{C}(\mathcal{G})} \mathbf{w}_c^\top \mathbf{f}_c(\mathbf{x}, \mathbf{y}_c) - \log Z_{\mathbf{w}}(\mathbf{x}).$$

In case of node potentials for word recognition, we could use the same basis functions as for individual character recognition: $f_{j,k}(\mathbf{x}, y_j) = \mathbb{1}(\mathbf{x}_{j,row,col} = on \wedge y_j = char)$ for each possible (row, col) in $\mathbf{x}_j$, the window of the image that corresponds to letter j and each $char \in \mathcal{Y}_j$ (we assume the input has been segmented into images $\mathbf{x}_j$ that correspond to letters). In general, we condition a clique only on a portion of the input $\mathbf{x}$, which we denote as $\mathbf{x}_c$. For the edge potentials, we can define basis functions for each combination of letters (assume for simplicity no dependence on $\mathbf{x}$) : $f_{j,j+1,k}(\mathbf{x}, y_j, y_{j+1}) = \mathbb{I}(y_j = char_1 \wedge y_{j+1} = char_2)$ for each $char_1 \in \mathcal{Y}_j$ and $char_2 \in \mathcal{Y}_{j+1}$. In this problem (as well as many others), we are likely to "tie" or "share" the parameters of the model $\mathbf{w}_c$ across cliques. Usually, all single node potentials would share the same weights and basis functions (albeit the relevant portion of the input $\mathbf{x}_c$ is different) and similarly for the pairwise cliques, no matter in what position they appear in the sequence.[2]

With slight abuse of notation, we stack all basis functions into one vector $\mathbf{f}$. For the sequence model, $\mathbf{f}$ has node functions and edge functions, so when c is a node, the edge functions in $\mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)$ are defined to evaluate to zero. Similarly, when c is an edge, the node functions in $\mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)$ are also defined to evaluate to zero. Now we can write:

$$\mathbf{f}(\mathbf{x}, \mathbf{y}) = \sum_{c \in \mathcal{C}(\mathcal{G})} \mathbf{f}(\mathbf{x}_c, \mathbf{y}_c).$$

We stack the weights in the corresponding manner, so the most likely assignment according to the model is given by:

$$\arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \log P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}),$$

in the same form as Eq. (11).

4.2 Inference

There are several important questions that can be answered by probabilistic models. The task of finding the most likely assignment, known as maximum a-posteriori (MAP) or most likely explanation (MPE), is just one of such questions, but most relevant to our discussion. The Viterbi dynamic programming algorithm solves this problem for chain networks in $\mathcal{O}(L)$ time. Let the highest score of any subsequence from 1 to $k > 1$ ending with value y_k be defined as

$$\phi^*_k(y_k) = \max_{\mathbf{y}_{1..k-1}} \prod_j \phi_j(\mathbf{x}, y_j)\phi_j(\mathbf{x}, y_{j-1}, y_j).$$

The algorithm computes the highest scores recursively:

$$\begin{aligned} &\phi^*_1(y_1) = \phi_1(\mathbf{x}, y_1), \quad \forall y_1 \in \mathcal{Y}_1; \\ &\phi^*_k(y_k) = \max_{y_{k-1} \in \mathcal{Y}_{k-1}} \phi^*_{k-1}(y_{k-1})\phi_j(\mathbf{x}, y_k)\phi_j(\mathbf{x}, y_{k-1}, y_k), \qquad 1 < k \le L,\ \forall y_k \in \mathcal{Y}_k. \end{aligned}$$

The highest scoring sequence has score $\max_{y_L} \phi^*_L(y_L)$. Using the $\arg\max$'s of the max's in the computation of ϕ^*, we can back-trace the highest scoring sequence itself. We assume that score ties are broken in a predetermined way, say according to some lexicographic order of the symbols.

In general Markov networks, MAP inference is NP-hard (Cowell et al., 1999). However, there are several important subclasses of networks that allow polynomial time inference.

---

[2] Sometimes we might actually want some dependence on the position in the sequence, which can be accomplished by adding more basis functions that condition on the position of the clique.

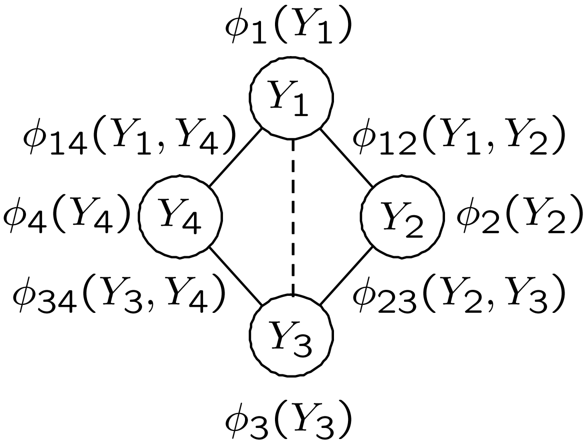


Figure 5: Diamond Markov network (added triangulation edge is dashed).

The most important of these is the class of networks with *low tree-width.* We need the concept of triangulation (or chordality) to formally define tree-width. Recall that a *cycle* of length l in an undirected graph $\mathcal{G}$ is a sequence of nodes $(v_0, v_1, \ldots, v_l)$, distinct except that $v_0 = v_l$, which are connected by edges $(v_i, v_{i+1}) \in \mathcal{G}$. A *chord* of this cycle is an edge $(v_i, v_j) \in \mathcal{G}$ between non-consecutive nodes.

Definition 4.2 (Triangulated graph) *An undirected graph $\mathcal{G}$ is* triangulated *if every one of its cycles of length ≥ 4 possesses a chord.*

Singly-connected graphs, like chains and trees, are triangulated since they contain no cycles. The simplest untriangulated network is the diamond in Fig. 5. To triangulate it, we can add the edge (Y_1, Y_3) or (Y_2, Y_4). In general, there are many possible sets of edges that can be added to triangulate a graph. The inference procedure creates a tree of cliques using the graph augmented by triangulation. The critical property of a triangulation for the inference procedure is the size of the largest clique.

Definition 4.3 (Tree-width of a graph) *The* tree-width *of a triangulated graph $\mathcal{G}$ is the size of its largest clique minus* 1*. The tree-width of an untriangulated graph $\mathcal{G}$ is the minimum tree-width of all triangulations of $\mathcal{G}$.*

The tree-width of a chain or a tree is 1 and the tree-width of Fig. 5 is 2. Finding the minimum tree-width triangulation of a general graph is NP-hard, but good heuristic algorithms exist (Cowell et al., 1999).

The inference procedure is based on a data structure called *junction tree* that can be constructed for a triangulated graph. The junction tree is an alternative representation of the same distribution that allows simple dynamic programming inference similar to the Viterbi algorithm for chains.

Definition 4.4 (Junction tree) *A* junction tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ *for a triangulated graph $\mathcal{G}$ is a tree in which the nodes are a subset of the cliques of the graph, $\mathcal{V} \subseteq \mathcal{C}(\mathcal{G})$ and the edges $\mathcal{E}$*

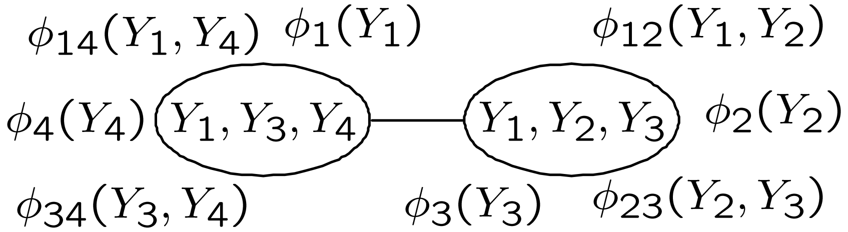


Figure 6: Diamond network junction tree. Each of the original potentials is associated with a node in the tree.

satisfy the running intersection property*: for any two cliques c and c', the variables in the intersection $c \cap c'$ are contained in the clique of every node of the tree on the (unique) path between c and c'.*

Fig. 6 shows a junction tree for the diamond network. Each of the original clique potentials must associated with exactly one node in the junction tree. For example, the potentials for the $\{Y_1, Y_3, Y_4\}$ and $\{Y_1, Y_3, Y_4\}$ nodes are the product of the associated clique potentials:

$$\begin{aligned}\phi_{134}(Y_1,Y_3,Y_4) &= \phi_1(Y_1)\phi_4(Y_4)\phi_{14}(Y_1,Y_4)\phi_{34}(Y_3,Y_4),\\ \phi_{123}(Y_1,Y_2,Y_3) &= \phi_2(Y_2)\phi_3(Y_3)\phi_{12}(Y_1,Y_2)\phi_{23}(Y_2,Y_3).\end{aligned}$$

Algorithms for constructing junction trees from triangulated graphs are described in detail in Cowell et al. (1999).

The Viterbi algorithm for junction trees picks an arbitrary root r for the tree $\mathcal{T}$ and proceeds recursively from the leaves to compute the highest scoring subtree at a node by combining the subtrees with highest score from its children. We denote the leaves of the tree as $Lv(\mathcal{T})$ and the children of node c (relative to the root r) as $Ch_r(c)$:

$$\begin{aligned}&\phi_l^*(\mathbf{y}_l) = \phi_l(\mathbf{x},\mathbf{y}_l), \quad \forall l \in Lv(\mathcal{T}),\ \forall \mathbf{y}_l \in \mathcal{Y}_l;\\ &\phi_c^*(\mathbf{y}_c) = \phi_c(\mathbf{x},\mathbf{y}_c) \prod_{c' \in Ch_r(c)} \max_{\mathbf{y}_{c'} \sim \mathbf{y}_c} \phi_{c'}^*(\mathbf{y}_{c'}), \qquad \forall c \in \mathcal{V}(\mathcal{T}) \setminus Lv(\mathcal{T}),\ \forall \mathbf{y}_c \in \mathcal{Y}_c,\end{aligned}$$

where $\mathbf{y}_{c'} \sim \mathbf{y}_c$ denotes whether the partial assignment $\mathbf{y}_c$ is consistent with the partial assignment $\mathbf{y}_{c'}$ on the variables in the intersection of c and c'. The highest score is given by $\max_{\mathbf{y}_r} \phi_r^*(\mathbf{y}_r)$. Using the arg max's of the max's in the computation of ϕ^*, we can backtrace the highest scoring assignment itself. Note that this algorithm is exponential in the tree-width, the size of the largest clique. Similar type of computations using the junction tree can be used to compute the partition function $Z_{\mathbf{w}}(\mathbf{x})$ (by simply replacing max by $\sum$) as well as marginal probabilities $P(\mathbf{y}_c|\mathbf{x})$ for the cliques of the graph (Cowell et al., 1999).

4.3 Linear programming MAP inference

In this section, we present an alternative inference method based on linear programming. Although solving the MAP inference using a general LP solver is less efficient than the

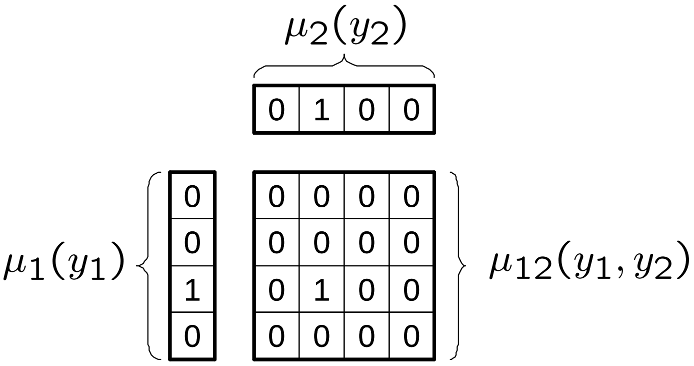


Figure 7: Example of marginal agreement: row sums of $\mu_{12}(y_1, y_2)$ agree with $\mu_1(y_1)$, column sums agree with $\mu_2(y_2)$.

dynamic programming algorithms above, this formulation is crucial in viewing Markov networks in a unified framework of the structured models we consider and to our development of common estimation methods in later chapters. Let us begin with a linear integer program to compute the optimal assignment $\mathbf{y}$. We represent an assignment as a set binary variables $\mu_c(\mathbf{y}_c)$, one for each clique c and each value of the clique $\mathbf{y}_c$, that denotes whether the assignment has that value, such that:

$$\log \prod_c \phi_c(\mathbf{x}, \mathbf{y}_c) = \sum_{c,\mathbf{y}_c} \mu_c(\mathbf{y}_c) \log \phi_c(\mathbf{x}, \mathbf{y}_c).$$

We call these variables marginals, as they correspond to the marginals of a distribution that has all of its mass centered on the MAP instantiation (assuming it is unique). There are several elementary constraints that such marginals satisfy. First, they must sum to one for each clique. Second, the marginals for cliques that share variables are consistent. For any clique $c \in \mathcal{C}$ and a subclique $s \subset c$, the assignment of the subclique, $\mu_s(\mathbf{y}_s)$, must be consistent with the assignment of the clique, $\mu_c(\mathbf{y}_c)$. Together, these constraints define a linear integer program:

$$\begin{aligned} \max \quad & \sum_{c,\mathbf{y}_c} \mu_c(\mathbf{y}_c) \log \phi_c(\mathbf{x}, \mathbf{y}_c) && (12) \\ \text{s.t.} \quad & \sum_{\mathbf{y}_c} \mu_c(\mathbf{y}_c) = 1, \quad \forall c \in \mathcal{C}; \qquad \mu_c(\mathbf{y}_c) \in \{0, 1\}, \quad \forall c \in \mathcal{C},\ \forall \mathbf{y}_c; \\ & \mu_s(\mathbf{y}_s) = \sum_{\mathbf{y}'_c \sim \mathbf{y}_s} \mu_c(\mathbf{y}'_c), \qquad \forall s, c \in \mathcal{C}, \ \ s \subset c, \ \ \forall \mathbf{y}_s. \end{aligned}$$

For example, in case the network is a chain or a tree, we will have node and edge marginals that sum to 1 and agree with each other as in Fig. 7.

Clearly, for any assignment $\mathbf{y}'$, we can define $\mu_c(\mathbf{y}_c)$ variables that satisfy the above constraints by setting $\mu_c(\mathbf{y}_c) = 1\!\!1(\mathbf{y}'_c = \mathbf{y}_c)$. We can also show that converse is true: any valid setting of $\mu_c(\mathbf{y}_c)$ corresponds to a valid assignment $\mathbf{y}$. In fact,

Lemma 4.5 *For a triangulated network with unique MAP assignment, the integrality constraint in the integer program in Eq. (12) can be relaxed and the resulting LP is guaranteed to have integer solutions.*

A proof of this lemma appears in Wainwright et al. (2002). Intuitively, the constraints force the marginals $\mu_c(\mathbf{y}_c)$ to correspond to some valid joint distribution over the assignments. The optimal distribution with the respect to the objective puts all its mass on the MAP assignment. If the MAP assignment is not unique, the value of the LP is the same as the value of the integer program, and any linear combination of the MAP assignments maximizes the LP.

in case the network is not triangulated, the set of marginals is not guaranteed to represent a valid distribution. Consider, for example, the diamond network in Fig. 5 with binary variables, with the following edge marginals that are consistent with the constraints:

$$\begin{aligned} &\mu_{12}(0,0) = \mu_{12}(1,1) = 0.5, && \mu_{12}(1,0) = \mu_{12}(0,1) = 0; \\ &\mu_{23}(0,0) = \mu_{23}(1,1) = 0.5, && \mu_{23}(1,0) = \mu_{23}(0,1) = 0; \\ &\mu_{34}(0,0) = \mu_{34}(1,1) = 0.5, && \mu_{34}(1,0) = \mu_{34}(0,1) = 0; \\ &\mu_{14}(0,0) = \mu_{34}(1,1) = 0, && \mu_{14}(1,0) = \mu_{14}(0,1) = 0.5. \end{aligned}$$

The corresponding node marginals must all be set to 0.5. Note that the edge marginals for $(1,2),(2,3),(3,4)$ disallow any assignment other than 0000 or 1111, but the edge marginal for $(1,4)$ disallows any assignment that has $Y_1 = Y_4$. Hence this set of marginals disallows all assignments. If we triangulate the graph and add the cliques $\{Y_1, Y_2, Y_3\}$ and $\{Y_1, Y_3, Y_4\}$ with their corresponding constraints, the above marginals will be disallowed.

In graphs where triangulation produces very large cliques, exact inference is intractable. We can resort to the above LP *without* triangulation as an approximate inference procedure (augmented with some scheme for rounding possibly fractional solutions). In Sec. 5, we discuss another subclass of networks where MAP inference using LPs is tractable for any network topology, but with a restricted type of potentials.

4.4 Maximum conditional likelihood estimation

A standard discriminative estimation method for Markov networks is based on likelihood. The regularized maximum conditional likelihood approach of learning the weights $\mathbf{w}$ of a Markov network is similar to logistic regression we described in Sec. 2.2. The objective is to minimize the training conditional log-loss with an additional regularization term, usually the squared-norm of the weights $\mathbf{w}$ (Lafferty et al., 2001):

$$\frac{1}{2}||\mathbf{w}||^2 - C\sum_i \log P_{\mathbf{w}}(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \log Z_{\mathbf{w}}(\mathbf{x}^{(i)}) - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}),$$

where $\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$. This objective function is convex in the parameters $\mathbf{w}$, so we have an unconstrained convex optimization problem. The gradient with respect to $\mathbf{w}$ is given by:

$$\mathbf{w} + C\sum_i \left[\mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})] - \mathbf{f}_i(\mathbf{y}^{(i)})\right] = \mathbf{w} - C\sum_i \mathbf{E}_{i,\mathbf{w}}[\Delta\mathbf{f}_i(\mathbf{y})],$$

where $\mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})] = \sum_{\mathbf{y}\in\mathcal{Y}} \mathbf{f}_i(\mathbf{y}) P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}^{(i)})$ is the expectation under the conditional distribution $P_{\mathbf{w}}(\mathbf{y} \mid \mathbf{x}^{(i)})$ and $\Delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$, as before. To compute the expectations, we can use inference in the Markov network to calculate marginals $P_{\mathbf{w}}(\mathbf{y}_c \mid \mathbf{x}^{(i)})$ for each clique c in the network Sec. 4.2. Since the basis functions decompose over the cliques of the network, the expectation decomposes as well:

$$\mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})] = \sum_{c\in\mathcal{C}^{(i)}} \sum_{\mathbf{y}_c\in\mathcal{Y}_c^{(i)}} \mathbf{f}_{i,c}(\mathbf{y}_c) P_{\mathbf{w}}(\mathbf{y}_c \mid \mathbf{x}^{(i)}).$$

As long as the network has low-treewidth, computing these expectations is tractable.

Second order methods for solving unconstrained convex optimization problems, such as Newton's method, require the second derivatives as well as the gradient. Let $\delta\mathbf{f}_i(\mathbf{y}) = \mathbf{f}_i(\mathbf{y}) - \mathbf{E}_{i,\mathbf{w}}[\mathbf{f}_i(\mathbf{y})]$. The Hessian of the objective depends on the covariances of the basis functions:

$$I + C\sum_i \mathbf{E}_{i,\mathbf{w}}\left[\delta\mathbf{f}_i(\mathbf{y})\delta\mathbf{f}_i(\mathbf{y})^\top\right],$$

where I is a $n \times n$ identity matrix. Computing the Hessian is more expensive than the gradient, since we need to calculate joint marginals of every pair of cliques c and c', $P_{\mathbf{w}}(\mathbf{y}_{c\cup c'} \mid \mathbf{x}_i)$ as well as covariances of all basis functions, which is quadratic in the number of cliques and the number of functions. A standard approach is to use an approximate second order method that does not need to compute the Hessian, but uses only the gradient information (Nocedal and Wright, 1999; Boyd and Vandenberghe, 2004). Conjugate Gradients or L-BFGS methods have been shown to work very well on large estimation problems (Sha and Pereira, 2003; Pinto et al., 2003), even with millions of parameters $\mathbf{w}$.

5. Associative networks

Associative interactions arise naturally in the context of image processing, where nearby pixels are likely to have the same label (Besag, 1986; Boykov et al., 1999b). In this setting, a common approach is to use a *generalized Potts model* (Potts, 1952), which penalizes assignments that do not have the same label across the edge: $\phi_{ij}(k,l) = \lambda_{ij}$, $\forall k \neq l$ and $\phi_{ij}(k,k) = 1$, where $\lambda_{ij} \leq 1$.

For binary-valued Potts models, Greig et al. (1989) show that the MAP problem can be formulated as a min-cut in an appropriately constructed graph. Thus, the MAP problem can be solved exactly for this class of models in polynomial time. For $L > 2$, the MAP problem is NP-hard, but a procedure based on a relaxed linear program guarantees a factor 2 approximation of the optimal solution (Boykov et al., 1999b; Kleinberg and Tardos, 1999). Our associative potentials extend the Potts model in several ways. Importantly, AMNs allow different labels to have different attraction strength: $\phi_{ij}(k,k) = \lambda_{ij}(k)$, where $\lambda_{ij}(k) \geq 1$, and $\phi_{ij}(k,l) = 1$, $\forall k \neq l$. This additional flexibility is important in many domains, as different labels can have very diverse affinities. For example, foreground pixels tend to have locally coherent values while background is much more varied.

In a second important extension, AMNs admit non-pairwise interactions between variables, with potentials over cliques involving m variables $\phi(\mu_{i1}, \ldots, \mu_{im})$. In this case, the clique potentials are constrained to have the same type of structure as the edge potentials:

There are K parameters $\phi_c(k,\ldots,k) = \lambda_c(k) \geq 1$ and the rest of the entries are set to 1. In particular, using this additional expressive power, AMNs allow us to encode the pattern of (soft) transitivity present in many domains. For example, consider the problem of predicting whether two proteins interact (Vazquez et al., 2003); this probability may increase if they *both* interact with another protein. This type of transitivity could be modeled by a ternary clique that has high λ for the assignment with all interactions present.

More formally, we define *associative* functions and potentials as follows.

Definition 5.1 *A function $g : \mathcal{Y} \mapsto I\!R$ is* associative for a graph $\mathcal{G}$ over K-ary variables *if it can be written as:*

$$g(\mathbf{y}) = \sum_{v\in\mathcal{V}}\sum_{k=1}^{K} g_v(k)\,1\!\!I(y_v = k) \;+ \sum_{c\in\mathcal{C}\setminus\mathcal{V}}\sum_{k=1}^{K} g_c(k)\,1\!\!I(\mathbf{y}_c = k,\ldots,k); \qquad g_c(k) \geq 0,\;\; \forall c \in \mathcal{C}\setminus\mathcal{V},$$

where $\mathcal{V}$ are the nodes and $\mathcal{C}$ are the cliques of the graph $\mathcal{G}$. A set of potentials $\phi(\mathbf{y})$ is associative if $\phi(\mathbf{y}) = e^{g(\mathbf{y})}$ and $g(\mathbf{y})$ is associative.

5.1 LP Inference

We can write an integer linear program for the problem of finding the maximum of an associative function $g(\mathbf{y})$, where we have a "marginal" variable $\mu_v(k)$ for each node $v \in \mathcal{V}$ and each label k, which indicates whether node v has value k, and $\mu_c(k)$ for each clique c (containing more than one variable) and label k, which represents the event that all nodes in the clique c have label k:

$$\begin{aligned} \max \quad & \sum_{v\in\mathcal{V}}\sum_{k=1}^{K}\mu_v(k)g_v(k) + \sum_{c\in\mathcal{C}\setminus\mathcal{V}}\sum_{k=1}^{K}\mu_c(k)g_c(k) \\ \text{s.t.} \quad & \mu_c(k) \in \{0,1\}, \quad \forall c \in \mathcal{C},\; k; \quad \sum_{k=1}^{K}\mu_v(k) = 1, \quad \forall v \in \mathcal{V}; \\ & \mu_c(k) \leq \mu_v(k), \quad \forall c \in \mathcal{C}\setminus\mathcal{V},\; v \in c,\; k. \end{aligned} \tag{13}$$

Note that we substitute the constraint $\mu_c(k) = \bigwedge_{v\in c}\mu_v(k)$ by linear inequality constraints $\mu_c(k) \leq \mu_v(k)$. This works because the coefficient $g_c(k)$ is non-negative and we are maximizing the objective function. Hence at the optimum, $\mu_c(k) = \min_v \mu_v(k)$, which is equivalent to $\mu_c(k) = \bigwedge_{v\in c}\mu_v(k)$, when $\mu_v(k)$ are binary.

It can be shown that in the binary case, the linear relaxation of Eq. (13), (where the constraints $\mu_c(k) \in \{0,1\}$ are replaced by $\mu_c(k) \geq 0$), is guaranteed to produce an integer solution when a unique solution exists.

Theorem 5.2 *If $K = 2$, for any associative function g, the linear relaxation of Eq. (13) has an integral optimal solution.*

See Appendix A.1 for the proof. This result states that the MAP problem in binary AMNs is tractable, regardless of network topology or clique size. In the non-binary case ($L > 2$), these LPs can produce fractional solutions and we use a rounding procedure to get an integral solution.

Theorem 5.3 *If $K > 2$, for any associative function g, the linear relaxation of Eq. (13) has a solution that is larger than the solution of the integer program by at most the number of variables in the largest clique.*

In the appendix, we also show that the approximation ratio of the rounding procedure is the inverse of the size of the largest clique (e.g., $\frac{1}{2}$ for pairwise networks). Although artificial examples with fractional solutions can be easily constructed by using symmetry, it seems that in real data such symmetries are often broken. In fact, in all our experiments with $L > 2$ on real data, we never encountered fractional solutions.

We can also use efficient min-cut algorithms to perform exact inference on the learned models for $K = 2$ and approximate inference for $K > 2$ (Boykov et al., 1999a). See appendix for the details of the reduction.

However, for associative Markov networks, maximum conditional likelihood requires computing the partition function $Z_{\mathbf{w}}(\mathbf{x})$ (or clique marginals), which is #P-Complete. The marginals can be approximated using MCMC methods (Jerrum and Sinclair, 1993) or loopy belief propagation Yedidia et al. (2000).

6. Maximum margin estimation

We finally turn to the estimation method we propose in this paper. Given a sample $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$, we aim to find parameters $\mathbf{w}$ such that:

$$\arg\max_{\mathbf{y}\in\mathcal{Y}^{(i)}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}) \;\approx\; \mathbf{y}^{(i)}, \quad \forall i,$$

where $\mathcal{Y}^{(i)} = \mathcal{Y}(\mathbf{x}^{(i)})$. We develop a compact convex formulation for finding such parameters $\mathbf{w}$. There are several reasons to derive compact convex formulations. First and foremost, we can find globally optimal parameters (with fixed precision) in polynomial time. Second, we can use standard optimization software to solve the problem. Although special-purpose algorithms that exploit the structure of a particular problem are often much faster (see Sec. 10), the availability of off-the-shelf software is very important for quick development and testing of such models. Third, we can analyze the generalization performance of the framework without worrying about the actual algorithms used to carry out the optimization and the associated woes of intractable optimization problems: local minima, greedy and heuristic methods, etc.

Throughout, we will adopt the hinge upper bound $\bar{\ell}_i(h(\mathbf{x}^{(i)}))$ on the loss function for structured classification inspired by max-margin criterion:

$$\bar{\ell}_i(h(\mathbf{x}^{(i)})) = \max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})] - \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) \geq \ell_i(h(\mathbf{x}^{(i)})),$$

where as before, $\bar{\ell}_i(h(\mathbf{x}^{(i)})) = \bar{\ell}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, h(\mathbf{x}^{(i)}))$, $\ell_i(h(\mathbf{x}^{(i)})) = \ell(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, h(\mathbf{x}^{(i)}))$, and $\mathbf{f}_i(\mathbf{y}) = \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$. With this upper bound, the min-max formulation for structured classification problem is analogous to multi-class SVM formulation in Eq. (8) and Eq. (9):

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \qquad (14) \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \max_{\mathbf{y}\in\mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \quad \forall i. \end{aligned}$$

The above formulation is a convex quadratic program in $\mathbf{w}$, since $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top\mathbf{f}_i(\mathbf{y})+\ell_i(\mathbf{y})]$ is convex in $\mathbf{w}$ (maximum of affine functions is a convex function).

The problem with Eq. (14) is that the constraints have a very unwieldy form. Another way to express this problem is using $\sum_i|\mathcal{Y}^{(i)}|$ linear constraints, which is generally exponential in L_i, the number of variables in $\mathbf{y}_i$.

$$\begin{aligned}\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i && (15)\\ \text{s.t.} \quad & \mathbf{w}^\top\mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \mathbf{w}^\top\mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \ \ \forall \mathbf{y}\in\mathcal{Y}^{(i)}.\end{aligned}$$

This form reveals the "maximum margin" nature of the formulation. We can interpret $\frac{1}{||\mathbf{w}||}\mathbf{w}^\top[\mathbf{f}_i(\mathbf{y}^{(i)})-\mathbf{f}_i(\mathbf{y})]$ as the *margin* of $\mathbf{y}^{(i)}$ over another $\mathbf{y}\in\mathcal{Y}^{(i)}$. Assuming ξ_i are all zero (say because C is very large), the constraints enforce

$$\mathbf{w}^\top\mathbf{f}_i(\mathbf{y}^{(i)}) - \mathbf{w}^\top\mathbf{f}_i(\mathbf{y}) \geq \ell_i(\mathbf{y}),$$

so minimizing $||\mathbf{w}||$ maximizes the smallest such margin, scaled by the loss $\ell_i(\mathbf{y})$. The slack variables ξ_i allow for violations of the constraints at a cost $C\xi_i$.

As in the univariate prediction, we measure the error of approximation using a loss function ℓ. In structured problems, where we are jointly predicting multiple variables, the loss is often not just the simple 0-1 loss. For structured classification, a natural loss function is a kind of Hamming distance between $\mathbf{y}^{(i)}$ and $h(\mathbf{x}^{(i)})$: the number of variables predicted incorrectly. If the loss function is not uniform over all the mistakes $\mathbf{y}\neq\mathbf{y}^{(i)}$, then the constraints make costly mistakes (those with high $\ell_i(\mathbf{y})$) less likely. In Sec. 9 we analyze the effect of non-uniform loss function (Hamming distance type loss) on generalization, and show a strong connection between the loss-scaled margin and expected risk of the learned model.

The formulation in Eq. (15) is a standard QP with linear constraints, but its exponential size is in general prohibitive. We now return to Eq. (14) and transform it to a a more manageable problem. The key to solving Eq. (14) efficiently is the **loss-augmented** inference

$$\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top\mathbf{f}_i(\mathbf{y})+\ell_i(\mathbf{y})]. \tag{16}$$

Even if $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}\mathbf{w}^\top\mathbf{f}_i(\mathbf{y})$ can be solved in polynomial time using convex optimization, the form of the loss term $\ell_i(\mathbf{y})$ is crucial for the loss-augmented inference to remain tractable. The range of tractable losses will depend strongly on the problem itself ($\mathbf{f}$ and $\mathcal{Y}$). Even within the range of tractable losses, some are more efficiently computable than others. We focus on decomposable loss functions (like Hamming loss), which we will define more formally below.

Assume we can use the linear programming formulation of inference in general Markov networks (Eq. (12)) and AMNs (Eq. (13)) to compute this loss-augmented maximum:

$$\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top\mathbf{f}_i(\mathbf{y})+\ell_i(\mathbf{y})] \quad = \quad d_i + \max_{\substack{\mu_i\geq 0\\ \mathbf{A}_i\mu_i\leq\mathbf{b}_i}} \quad (\mathbf{F}_i\mathbf{w}+\mathbf{l}_i)^\top\mu_i, \tag{17}$$

where $\mathbf{A}_i, \mathbf{b}_i$ depend only on $\mathbf{x}^{(i)}$ and $\mathbf{f}(\mathbf{x},\mathbf{y})$ and $d_i, \mathbf{F}_i, \mathbf{l}_i$ additionally depend on $\mathbf{y}^{(i)}$, $\mathbf{w}$ and ℓ. We will specify the exact dependence below. Such formulation is compact if the number of variables μ_i and constraints $\mathbf{A}_i\mu_i \le \mathbf{b}_i$ is polynomial in L_i, the number of variables in $\mathbf{y}^{(i)}$.

We can assume that the LP on the right-hand-side of Eq. (17) is feasible and bounded if Eq. (16) is, so the its LP dual is feasible and bounded as well. By strong duality, we have

$$d_i + \max_{\substack{\mu_i \ge 0 \\ \mathbf{A}_i\mu_i \le \mathbf{b}_i}} (\mathbf{F}_i\mathbf{w} + \mathbf{l}_i)^\top \mu_i = d_i + \min_{\substack{\lambda_i \ge 0 \\ \mathbf{A}_i^\top \lambda_i \ge \mathbf{F}_i\mathbf{w}+\mathbf{l}_i}} \mathbf{b}_i^\top \lambda_i, \tag{18}$$

where λ_i is a vector of dual variables (Lagrange multipliers). Plugging Eq. (18) into Eq. (14), we get

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i & (19)\\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \ge d_i + \min_{\substack{\lambda_i \ge 0 \\ \mathbf{A}_i^\top \lambda_i \ge \mathbf{F}_i\mathbf{w}+\mathbf{l}_i}} \mathbf{b}_i^\top \lambda_i, \quad \forall i. \end{aligned}$$

Moreover, we can combine the minimization over λ with minimization over $\{\mathbf{w}, \xi\}$. The reason for this is that if the right hand side is not at the minimum, the constraint is tighter than necessary, leading to a suboptimal solution $\mathbf{w}$. Optimizing jointly over λ as well will produce a solution to $\{\mathbf{w}, \xi\}$ that is optimal.

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i & (20)\\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \ge d_i + \mathbf{b}_i^\top \lambda_i, \quad \forall i; \\ & \mathbf{A}_i^\top \lambda_i \ge \mathbf{F}_i\mathbf{w} + \mathbf{l}_i, \quad \forall i; \\ & \lambda_i \ge 0, \quad \forall i. \end{aligned}$$

Hence we have a joint and compact convex optimization program for estimating $\mathbf{w}$. We investigate the exact form of this program for general Markov networks and AMNs in sections below. Before moving on to the particulars however, we consider general methods of approximating the estimation problem.

6.1 Approximations: upper and lower bounds

We have described several classes of Markov networks for which we can not solve the inference problem efficiently. Often, we cannot compute $\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ exactly or explicitly, but can only upper or lower bound it. Fig. 8 shows schematically how approximating of the max subproblem reduces or extends the feasible space of $\mathbf{w}$ and ξ and leads to approximate solutions. The nature of these lower and upper bounds depends on the problem, but we consider two general cases below.

6.1.1 CONSTRAINT GENERATION

When compact inference formulation as an LP is infeasible, but the maximization problem can be solved or approximated by a combinatorial algorithm, we can resort to *constraint generation* or *cutting plane* methods. Consider Eq. (15), where we have an exponential

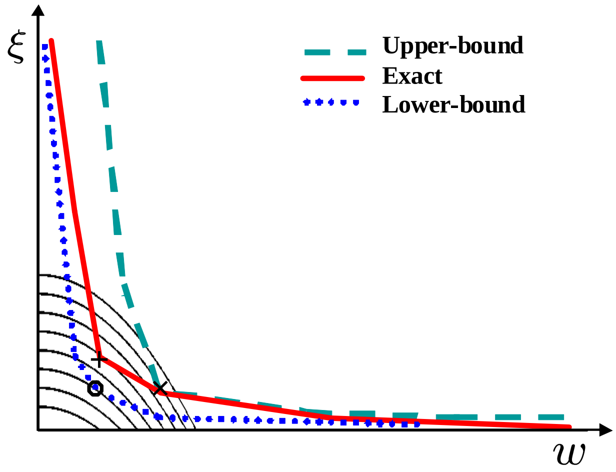


Figure 8: Exact and approximate constraints on the max-margin quadratic program. The solid red line represents the constraints imposed by the assignments $\mathbf{y} \in \mathbf{Y}^{(i)}$, whereas the dashed and dotted lines represent approximate constraints. The approximate constraints may coincide with the exact constraints in some cases, and be more stringent or relaxed in others. The parabolic contours represent the value of the objective function and '+', 'x' and 'o' mark the different optima.

number of linear constraints, one for each i and $\mathbf{y} \in \mathcal{Y}^{(i)}$. Only a subset of those constraints will be active at the optimal solution $\mathbf{w}$. In fact, not more than the number of parameters n plus the number of examples m can be active in general, since that is the number of variables. If we can identify a small number of constraints that are critical to the solution, we do not have to include all of them. Of course, identifying these constraints is in general as difficult as solving the problem, but a greedy approach of adding the most violated constraints often achieves good approximate solutions after adding a small (polynomial) number of constraints. If we continue adding constraints until there are no more violated ones, the resulting solution is optimal.

We assume that we have an algorithm that produces $\mathbf{y} = \arg\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. The algorithm is described in Fig. 9. We maintain, for each example i, a small but growing set of assignments $\widetilde{\mathcal{Y}}^{(i)} \subset \mathcal{Y}^{(i)}$. At each iteration, we solve the problem with a subset of constraints:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \qquad (21) \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \;\; \forall \mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}. \end{aligned}$$

The only difference between Eq. (15) and Eq. (21) is that $\mathcal{Y}^{(i)}$ has been replaced by $\widetilde{\mathcal{Y}}^{(i)}$. We then compute $\mathbf{y} = \arg\max_{\mathbf{y}\in\mathcal{Y}^{(i)}}[\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ for each i and check whether the constraint $\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i + \epsilon \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$, is violated, where ϵ is a user defined precision parameter. If it is violated, we set $\widetilde{\mathcal{Y}}^{(i)} = \widetilde{\mathcal{Y}}^{(i)} \cup \mathbf{y}$. The algorithm terminates

```
Input: precision parameter $\epsilon$.
    1. Initialize: $\widetilde{\mathcal{Y}}^{(i)} = \{\}, \ \forall i$.
    2. Set violation = 0 and solve for $\mathbf{w}$ and $\xi$ by optimizing
           min    $\frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i$
           s.t.   $\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y}), \quad \forall i, \ \forall \mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}$.
    3. For each $i$,
           Compute $\mathbf{y} = \arg\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$,
           if   $\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i + \epsilon \leq \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$,
           then   set $\widetilde{\mathcal{Y}}^{(i)} = \widetilde{\mathcal{Y}}^{(i)} \cup \mathbf{y}$ and violation = 1
    4. if violation = 1 goto 2.
Return $\mathbf{w}$.
```

Figure 9: A constraint generation algorithm.

when no constraints are violated. In Fig. 8, the lower-bound on the constraints provided by $\widetilde{\mathcal{Y}}^{(i)} \cup \mathbf{y}$ keeps tightening with each iteration, terminating when the desired precision ϵ is reached. We note that if the algorithm that produces $\mathbf{y} = \arg\max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$ is suboptimal, the approximation error of the solution we achieve might be much greater than ϵ. The number of constraints that must be added before the algorithm terminates depends on the precision ϵ and problem specific characteristics. See (Bertsimas and Tsitsiklis, 1997; Boyd and Vandenberghe, 2004) for a more in-depth discussion of cutting planes methods. This approach may also be computationally faster in providing a very good approximation in practice if the explicit convex programming formulation is polynomial in size, but very large, while the maximization algorithm is comparatively fast.

6.1.2 Constraint strengthening

In many problems, the maximization problem we are interested in may be very expensive or intractable, as in MAP inference in large tree-width Markov networks or multi-class AMNs. Many such problems can be written as *integer* programs. Relaxations of such integer programs into LPs, QPs or SDPs often provide excellent approximation algorithms (Hochbaum, 1997; Nemhauser and Wolsey, 1999). The relaxation usually defines a larger feasible space $\widetilde{\mathcal{Y}}^{(i)} \supset \mathcal{Y}^{(i)}$ over which the maximization is done, where $\mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}$ may correspond to a "fractional" assignment. For example, a solution to the MAP LP in Eq. (12) for an untriangulated network may not correspond to any valid assignment. In such a case, the approximation is an over-estimate of the constraints:

$$\max_{\mathbf{y} \in \widetilde{\mathcal{Y}}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})] \geq \max_{\mathbf{y} \in \mathcal{Y}^{(i)}} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})].$$

Hence the constraint set is tightened with such invalid assignments. Fig. 8 shows how the over-estimate reduces the feasible space of $\mathbf{w}$ and ξ.

Note that for every setting of the weights $\mathbf{w}$ that produces fractional solutions for the relaxation, the approximate constraints are tightened because of the additional invalid assignments. In this case, the approximate MAP solution has higher value than any integer solution, including the true assignment $\mathbf{y}^{(i)}$, thereby driving up the corresponding slack ξ_i. By contrast, for weights $\mathbf{w}$ for which the MAP approximation is integer-valued, the margin has the standard interpretation as the difference between the score of $\mathbf{y}^{(i)}$ and the MAP $\mathbf{y}$ (according to $\mathbf{w}$). As the objective includes a penalty for the slack variable, intuitively, minimizing the objective tends to drive the weights $\mathbf{w}$ away from the regions where the solutions to the approximation are fractional. In essence, the estimation algorithm is finding weights that are not necessarily optimal for an *exact* maximization algorithm, but (close to) optimal for the particular *approximate* maximization algorithm used. In practice, we will show experimentally that such approximations often work very well.

7. Maximum margin Markov networks

We now address the problem of max margin estimation in general Markov networks. We are given a labeled training sample $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m$, drawn from a fixed distribution D over $\mathcal{X} \times \mathcal{Y}$. We assume the structure of the network is given: we have a mapping from an input $\mathbf{x}$ to the corresponding Markov network graph $\mathcal{G}(\mathbf{x}) = \{\mathcal{V}, \mathcal{E}\}$ where the nodes $\mathcal{V}$ map to the variables in $\mathbf{y}$. We abbreviate $\mathcal{G}(\mathbf{x}^{(i)})$ as $\mathcal{G}^{(i)}$ below. In handwriting recognition, this mapping depends on the segmentation algorithm that determines how many letters the sample image contains and splits the image into individual images for each letter. It also depends on the basis functions we use to model the dependencies of the problem, for example, first-order Markov chain or a higher-order models. Note that the topology and size of the graph $\mathcal{G}^{(i)}$, might be different for each example i. For instance, the training sequences might have different lengths.

We know from Sec. 4.3 how to express $\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})$ as an LP, but the important difference is the loss function ℓ_i. The simplest loss is the 0/1 loss $\ell_i(\mathbf{y}) \equiv \mathbb{I}(\mathbf{y}^{(i)} \neq \mathbf{y})$. In fact this loss for sequence models was used by Collins (2001) and Altun et al. (2003). However, in structured problems, where we are predicting multiple labels, the loss is often not just the simple 0/1 loss, but may depend on the number of labels and type of labels predicted incorrectly or perhaps the number of cliques of labels predicted incorrectly. In general, we assume that the loss, like the basis functions, decomposes over the cliques of labels.

Assumption 7.1 *The loss function $\ell_i(\mathbf{y})$ is decomposable:*

$$\ell_i(\mathbf{y}) = \sum_{c \in \mathcal{C}(\mathcal{G}^{(i)})} \ell(\mathbf{x}_c^{(i)}, \mathbf{y}_c^{(i)}, \mathbf{y}_c) = \sum_{c \in \mathcal{C}(\mathcal{G}^{(i)})} \ell_{i,c}(\mathbf{y}_c).$$

We will focus on decomposable loss functions below. A natural choice that we use in our experiments is the Hamming distance:

$$\ell^H(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{y}) = \sum_{v \in \mathcal{V}^{(i)}} \mathbb{I}(y_v^{(i)} \neq y_v).$$

With this assumption, we can express this inference problem for a triangulated graph as a linear program for each example i as in Sec. 4.3:

$$\begin{aligned} \max \quad & \sum_{c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)[\mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c)] && (22)\\ \text{s.t.} \quad & \sum_{\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c) = 1, \quad \forall i, \ \forall c \in \mathcal{C}^{(i)}; \qquad \mu_{i,c}(\mathbf{y}_c) \geq 0, \ \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c; \\ & \mu_{i,s}(\mathbf{y}_s) = \sum_{\mathbf{y}'_c \sim \mathbf{y}_s} \mu_{i,c}(\mathbf{y}'_c), \qquad \forall s, c \in \mathcal{C}^{(i)}, \ \ s \subset c, \ \ \forall \mathbf{y}_s, \end{aligned}$$

where $\mathcal{C}^{(i)} = \mathcal{C}(\mathcal{G}^{(i)})$ are the cliques of the Markov network for example i.

As we showed before, the constraints ensure that the μ_i's form a proper distribution. If the most likely assignment is unique, then the distribution that maximizes the objective puts all its weight on that assignment. (If the arg max is not unique, any convex combination of the assignments is a valid solution). The dual of Eq. (22) is given by:

$$\begin{aligned} \min & \sum_c \lambda_{i,c} && (23)\\ \text{s.t.} \ \ & \lambda_{i,c} + \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c) - \sum_{s \subset c, \ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c), \ \ \forall c \in \mathcal{C}^{(i)}, \forall \mathbf{y}_c. \end{aligned}$$

In this dual, the $\lambda_{i,c}$ variables correspond to the normalization constraints, while $m_{i,c,s}(\mathbf{y}_c)$ variables correspond to the agreement constraints in the primal in Eq. (22).

Plugging the dual into Eq. (14) for each example i and maximizing jointly over all the variables ($\mathbf{w}, \xi, \lambda$ and $\mathbf{m}$), we have:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i && (24)\\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \sum_{i,c} \lambda_{i,c}, \quad \forall i; \\ & \lambda_{i,c} + \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c) - \sum_{s \subset c, \ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c), \ \ \forall c \in \mathcal{C}^{(i)}, \forall \mathbf{y}_c. \end{aligned}$$

In order to gain some intuition about this formulation, we make a change of variables from $\lambda_{i,c}$ to $\xi_{i,c}$:

$$\lambda_{i,c} = \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c}, \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}.$$

The reason for naming the new variables using the letter ξ will be clear in the following. For readability, we also introduce variables that capture the effect of all the agreement variables $\mathbf{m}$:

$$M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c, \ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.$$

With these new variables, we have:

$$\min \qquad \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \tag{25}$$

$$
\begin{aligned}
\text{s.t.} \quad & \xi_i \geq \sum_c \xi_{i,c}, \quad \forall i; \\
& \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c} \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c) + M_{i,c}(\mathbf{y}_c), \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c; \\
& M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c, \ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.
\end{aligned}
$$

Note that $\xi_i = \sum_c \xi_{i,c}$ at the optimum, since the slack variable ξ_i only appears only in the constraint $\xi_i \geq \sum_c \xi_{i,c}$ and the objective minimizes $C\xi_i$. Hence we can simply eliminate this set of variables:

$$
\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i,c} \xi_{i,c} && (26) \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c} \geq \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c) + M_{i,c}(\mathbf{y}_c), \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c; \\
& M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c, \ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.
\end{aligned}
$$

Finally, we can write this in a form that resembles our original formulation Eq. (14), but defined at a local level, for each clique:

$$
\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i,c} \xi_{i,c} && (27) \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) + \xi_{i,c} \geq \max_{\mathbf{y}_c} \, [\mathbf{w}^\top \mathbf{f}_{i,c}(\mathbf{y}_c) + \ell_{i,c}(\mathbf{y}_c) + M_{i,c}(\mathbf{y}_c)], \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}; \\
& M_{i,c}(\mathbf{y}_c) = \sum_{s \subset c, \ \mathbf{y}'_s \sim \mathbf{y}_c} m_{i,c,s}(\mathbf{y}'_s) - \sum_{s \supset c} m_{i,s,c}(\mathbf{y}_c), \ \ \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c.
\end{aligned}
$$

Note that without $M_{i,c}$ and $m_{i,c,s}$ variables, we essentially treat each clique as an independent classification problem: for each clique we have a hinge upper-bound on the local loss, or a margin requirement. The $m_{i,c,s}(\mathbf{y}_s)$ variables correspond to a certain kind of messages between cliques that distribute "credit" to cliques to fulfill this margin requirement from other cliques which have sufficient margin.

As an example, consider the first-order Markov chain in Fig. 10. The set of cliques consists of the five nodes and the four edges. Suppose for the sake of this example that our training data consists of only one training sample. The figure shows the local slack variables ξ and messages m between cliques for this sample. For brevity of notion in this example, we drop the dependence on the sample index i in the indexing of the variables (we also used $y_j^{(*)}$ instead of $y_j^{(i)}$ below). For concreteness, below we use the Hamming loss ℓ^H, which decomposes into local terms $\ell_j(y_j) = 1\!\!1(y_j \neq y_j^{(*)})$ for each node and is zero for the edges.

The constraints associated with the node cliques in this sequence are:

$$
\begin{aligned}
\mathbf{w}^\top \mathbf{f}_1(y_1^{(*)}) + \xi_1 \quad & \geq \quad \mathbf{w}^\top \mathbf{f}_1(y_1) + 1\!\!1(y_1 \neq y_1^{(*)}) - m_{1,12}(y_1), \ \ \forall y_1; \\
\mathbf{w}^\top \mathbf{f}_2(y_2^{(*)}) + \xi_2 \quad & \geq \quad \mathbf{w}^\top \mathbf{f}_2(y_2) + 1\!\!1(y_2 \neq y_2^{(*)}) - m_{2,12}(y_2) - m_{2,23}(y_2), \ \ \forall y_2; \\
\mathbf{w}^\top \mathbf{f}_3(y_3^{(*)}) + \xi_3 \quad & \geq \quad \mathbf{w}^\top \mathbf{f}_3(y_3) + 1\!\!1(y_3 \neq y_3^{(*)}) - m_{3,23}(y_3) - m_{3,34}(y_3), \ \ \forall y_3;
\end{aligned}
$$

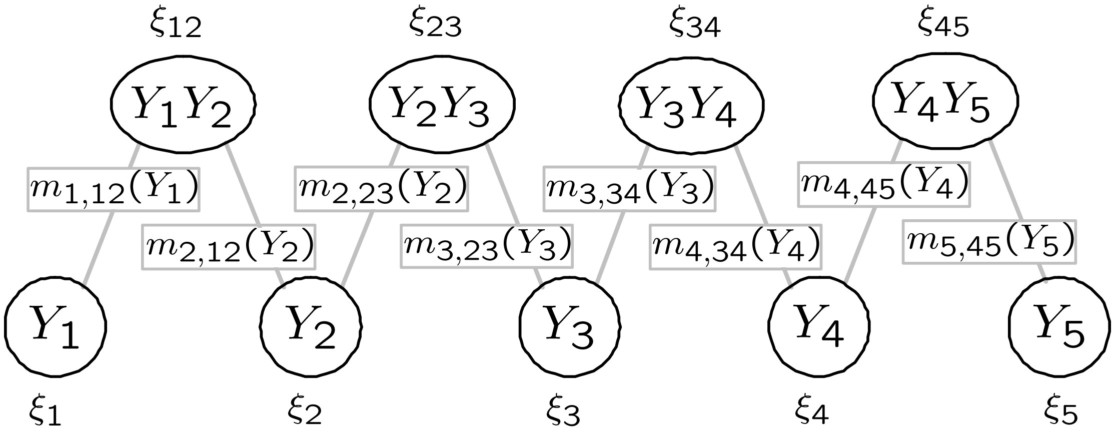


Figure 10: First-order chain shown as a set of cliques (nodes and edges). Also shown are the corresponding local slack variables ξ for each clique and messages m between cliques.

$$\begin{aligned}
\mathbf{w}^\top \mathbf{f}_4(y_4^{(*)}) + \xi_4 &\geq \mathbf{w}^\top \mathbf{f}_4(y_4) + 1\!\!1(y_4 \neq y_4^{(*)}) - m_{4,34}(y_4) - m_{4,45}(y_4), \;\; \forall y_4; \\
\mathbf{w}^\top \mathbf{f}_5(y_5^{(*)}) + \xi_5 &\geq \mathbf{w}^\top \mathbf{f}_5(y_5) + 1\!\!1(y_5 \neq y_5^{(*)}) - m_{5,45}(y_5), \;\; \forall y_5.
\end{aligned}$$

The edge constraints are:

$$\begin{aligned}
\mathbf{w}^\top \mathbf{f}_{12}(y_1^{(*)}, y_2^{(*)}) + \xi_{12} &\geq \mathbf{w}^\top \mathbf{f}_{12}(y_1, y_2) + m_{1,12}(y_1) + m_{2,12}(y_2), \;\; \forall y_1, y_2; \\
\mathbf{w}^\top \mathbf{f}_{23}(y_2^{(*)}, y_3^{(*)}) + \xi_{23} &\geq \mathbf{w}^\top \mathbf{f}_{23}(y_2, y_3) + m_{2,23}(y_2) + m_{3,23}(y_3), \;\; \forall y_2, y_3; \\
\mathbf{w}^\top \mathbf{f}_{34}(y_3^{(*)}, y_4^{(*)}) + \xi_{34} &\geq \mathbf{w}^\top \mathbf{f}_{34}(y_3, y_4) + m_{3,34}(y_3) + m_{4,34}(y_4), \;\; \forall y_3, y_4; \\
\mathbf{w}^\top \mathbf{f}_{45}(y_4^{(*)}, y_5^{(*)}) + \xi_{45} &\geq \mathbf{w}^\top \mathbf{f}_{45}(y_4, y_5) + m_{4,45}(y_4) + m_{5,45}(y_5), \;\; \forall y_4, y_5.
\end{aligned}$$

7.1 M^{3}N dual and kernels

In the previous section, we showed a derivation of a compact formulation based on LP inference. In this section, we develop an alternative dual derivation that provides a very interesting interpretation of the problem and is a departure for special-purpose algorithms we develop. We begin with the formulation as in Eq. (15):

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i \\
\text{s.t.} \quad & \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}) \geq \ell_i(\mathbf{y}) - \xi_i, \;\; \forall i, \mathbf{y},
\end{aligned} \tag{28}$$

where $\Delta\mathbf{f}_i(y) \equiv \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) - \mathbf{f}(\mathbf{x}^{(i)}, \mathbf{y})$. The dual is given by:

$$\max \quad \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\ell_i(\mathbf{y}) - \frac{1}{2}\left|\left|\sum_{i,\mathbf{y}} \alpha_i(\mathbf{y})\Delta\mathbf{f}_i(\mathbf{y})\right|\right|^2 \tag{29}$$

$$\text{s.t.} \qquad \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = C, \ \forall i; \qquad \alpha_i(\mathbf{y}) \geq 0, \ \ \forall i, \mathbf{y}.$$

In the dual, the exponential number of $\alpha_i(\mathbf{y})$ variables correspond to the exponential number of constraints in the primal. We make two small transformations to the dual that do not change the problem: we normalize α's by C (by letting $\alpha_i(\mathbf{y}) = C\alpha'_i(\mathbf{y})$), so that they sum to 1 and divide the objective by C. The resulting dual is given by:

$$\begin{aligned} \max \qquad & \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \ell_i(\mathbf{y}) - \frac{1}{2} C \left|\left| \sum_{i,\mathbf{y}} \alpha_i(\mathbf{y}) \Delta \mathbf{f}_i(\mathbf{y}) \right|\right|^2 \\ \text{s.t.} \qquad & \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = 1, \ \forall i; \qquad \alpha_i(\mathbf{y}) \geq 0, \ \ \forall i, \mathbf{y}. \end{aligned} \tag{30}$$

As in multi-class SVMs, the solution to the dual α gives the solution to the primal as a weighted combination: $\mathbf{w}^* = C \sum_{i,\mathbf{y}} \alpha_i^*(\mathbf{y}) \Delta \mathbf{f}_i(\mathbf{y})$.

7.1.1 Dual as a distribution

Our main insight is that the variables $\alpha_i(\mathbf{y})$ in the dual formulation Eq. (30) can be interpreted as a kind of *distribution* over $\mathbf{y}$, since they lie in the simplex

$$\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) = 1; \quad \alpha_i(\mathbf{y}) \geq 0, \ \forall \mathbf{y}.$$

This dual distribution does not represent the probability that the model assigns to an instantiation, but the importance of the constraint associated with the instantiation to the solution. The dual objective is a function of expectations of $\ell_i(\mathbf{y})$ and $\Delta \mathbf{f}_i(\mathbf{y})$ with respect to $\alpha_i(\mathbf{y})$. Since $\ell_i(\mathbf{y}) = \sum_c \ell_{i,c}(\mathbf{y}_c)$ and $\Delta \mathbf{f}_i(\mathbf{y}) = \sum_c \Delta \mathbf{f}_{i,c}(\mathbf{y}_c)$ decompose over the cliques of the Markov network, we only need clique marginals of the distribution $\alpha_i(\mathbf{y})$ to compute their expectations. We define the marginal dual variables as follows:

$$\mu_{i,c}(\mathbf{y}_c) = \sum_{\mathbf{y}' \sim \mathbf{y}_c} \alpha_i(\mathbf{y}'), \qquad \forall i, \quad \forall c \in \mathcal{C}^{(i)}, \quad \forall \mathbf{y}_c, \tag{31}$$

where $\mathbf{y}' \sim \mathbf{y}_c$ denotes whether the partial assignment $\mathbf{y}_c$ is consistent with the full assignment $\mathbf{y}'$. Note that the number of $\mu_{i,c}(\mathbf{y}_c)$ variables is small (polynomial) compared to the number of $\alpha_i(\mathbf{y})$ variables (exponential) if the size of the largest clique is constant with respect to the size of the network.

Now we can reformulate our entire QP (30) in terms of these marginal dual variables. Consider, for example, the first term in the objective function (fixing a particular i):

$$\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) \ell_i(\mathbf{y}) = \sum_{\mathbf{y}} \alpha_i(\mathbf{y}) \sum_c \ell_{i,c}(\mathbf{y}_c) = \sum_{c,\mathbf{y}_c} \ell_{i,c}(\mathbf{y}_c) \sum_{\mathbf{y}' \sim \mathbf{y}_c} \alpha_i(\mathbf{y}') = \sum_{c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c) \ell_{i,c}(\mathbf{y}_c).$$

The decomposition of the second term in the objective is analogous.

$$\sum_{\mathbf{y}} \alpha_i(\mathbf{y}) \Delta \mathbf{f}_i(\mathbf{y}) = \sum_{c,\mathbf{y}_c} \Delta \mathbf{f}_{i,c}(\mathbf{y}_c) \sum_{\mathbf{y}' \sim \mathbf{y}_c} \alpha_i(\mathbf{y}') = \sum_{c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c) \Delta \mathbf{f}_{i,c}(\mathbf{y}_c).$$

Let us denote the the objective of Eq. (30) as $\mathcal{Q}(\alpha)$. Note that it only depends on $\alpha_i(\mathbf{y})$ through its marginals $\mu_{i,c}(\mathbf{y}_c)$, that is, $\mathcal{Q}(\alpha) = \mathcal{Q}'(\mathcal{M}(\alpha))$, where $\mathcal{M}$ denotes the marginalization operator defined by Eq. (31) . The domain of this operator, $\mathcal{D}[\mathcal{M}]$, is the product of simplices for all the m examples. What is its range, $\mathcal{R}[\mathcal{M}]$, the set of legal marginals? Characterizing this set (also known as *marginal polytope*) compactly will allow us to work in the space of μ's:

$$\max_{\alpha\in\mathcal{D}[\mathcal{M}]} \mathcal{Q}(\alpha) \Leftrightarrow \max_{\mu\in\mathcal{R}[\mathcal{M}]} \mathcal{Q}'(\mu).$$

Hence we must ensure that μ_i corresponds to *some* distribution α_i, which is exactly what the constraints in the LP for MAP inference enforce (see discussion of Lemma 4.5). Therefore, when all $\mathcal{G}^{(i)}$ are triangulated, the following *structured* dual QP has the same primal solution ($\mathbf{w}^*$) as the original *exponential* dual QP in Eq. (30):

$$\begin{aligned} \max \quad & \sum_{i,c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\ell_{i,c}(\mathbf{y}_c) - \frac{1}{2}C\left|\left|\sum_{i,c,\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c)\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)\right|\right|^2 \qquad (32) \\ \text{s.t.} \quad & \sum_{\mathbf{y}_c} \mu_{i,c}(\mathbf{y}_c) = 1, \quad \forall i, \ \forall c \in \mathcal{C}^{(i)}; \qquad \mu_{i,c}(\mathbf{y}_c) \geq 0, \ \forall i, \ \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c; \\ & \mu_{i,s}(\mathbf{y}_s) = \sum_{\mathbf{y}'_c \sim \mathbf{y}_s} \mu_{i,c}(\mathbf{y}'_c), \quad \forall i, \ \forall s, c \in \mathcal{C}^{(i)}, \ s \subset c, \ \forall \mathbf{y}_s. \end{aligned}$$

The solution to the structured dual μ^* gives us the primal solution:

$$\mathbf{w}^* = C \sum_{i,c,\mathbf{y}_c} \mu^*_{i,c}(\mathbf{y}_c)\Delta\mathbf{f}_{i,c}(\mathbf{y}_c).$$

In this structured dual, we only enforce that there exists an α_i consistent with μ_i, but do not make a commitment about what it is. In general, the α distribution is not unique, but there is a continuum of distributions consistent with a set of marginals. The objective of the QP Eq. (30) does not distinguish between these distributions, since it only depends on their marginals. The maximum-entropy distribution α_i consistent with a set of marginals μ_i, however, is unique for a triangulated model and can be computed using the junction tree $\mathcal{T}^{(i)}$ for the network (Cowell et al., 1999).

Specifically, associated with each edge (c, c') in the tree $\mathcal{T}^{(i)}$ is a set of variables called the separator $s = c \cap c'$. Note that each separator s and complement of a separator $c \setminus s$ is also a clique of the original graph, since it is a subclique of a larger clique. We denote the set of separators as $\mathcal{S}^{(i)}$. Now we can define the maximum-entropy distribution $\alpha_i(\mathbf{y})$ as follows:

$$\alpha_i(\mathbf{y}) = \frac{\prod_{c\in\mathcal{T}^{(i)}} \mu_{i,c}(\mathbf{y}_c)}{\prod_{s\in\mathcal{S}^{(i)}} \mu_{i,s}(\mathbf{y}_s)}. \qquad (33)$$

Again, by convention $0/0 \equiv 0$.

7.1.2 Kernels

Note that the solution is a weighted combination of local basis functions and the objective of Eq. (32) can be expressed in terms of dot products between local basis functions

$$\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)^\top \Delta\mathbf{f}_{j,\bar{c}}(\mathbf{y}_{\bar{c}}) = [\mathbf{f}(\mathbf{x}_c^{(i)}, \mathbf{y}_c^{(i)}) - \mathbf{f}(\mathbf{x}_c^{(i)}, \mathbf{y}_c)]^\top [\mathbf{f}(\mathbf{x}_{\bar{c}}^{(j)}, \mathbf{y}_{\bar{c}}^{(j)}) - \mathbf{f}(\mathbf{x}_{\bar{c}}^{(j)}, \mathbf{y}_{\bar{c}})].$$

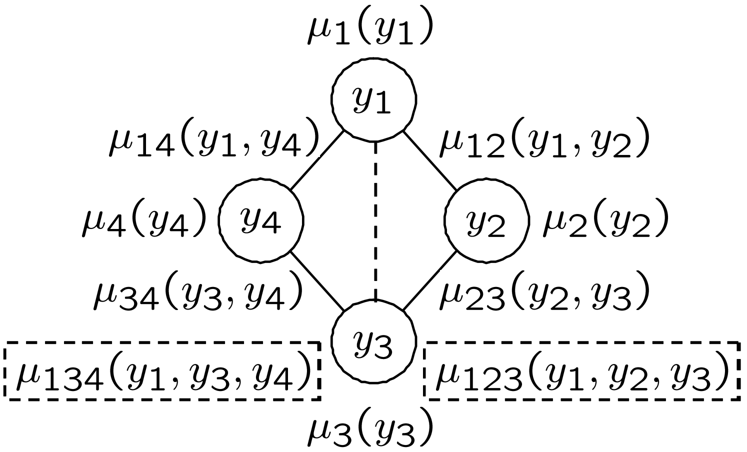


Figure 11: Diamond Markov network (added triangulation edge is dashed and three-node marginals are in dashed rectangles).

Hence, we can locally kernelize our models and solve Eq. (32) efficiently. Kernels are typically defined on the input, e.g. $k(\mathbf{x}_c^{(i)}, \mathbf{x}_{\bar{c}}^{(j)})$. In our handwriting example, we use a polynomial kernel on the pixel values for the node cliques. We usually extend the kernel over the input space to the joint input and output space by simply defining

$$\mathbf{f}(\mathbf{x}_c, \mathbf{y}_c)^\top \mathbf{f}(\mathbf{x}_{\bar{c}}, \mathbf{y}_{\bar{c}}) \equiv \mathbb{1}(\mathbf{y}_c = \mathbf{y}_{\bar{c}}) k(\mathbf{x}_c, \mathbf{x}_{\bar{c}}).$$

Of course, other definitions are possible and may be useful when the assignments in each clique $\mathbf{y}_c$ have interesting structure. In Sec. 11.1 we experiment with several kernels for the handwriting example. As in SVMs, the solutions to the max-margin QP are typically sparse in the μ variables. Hence, each log-potential in the network "remembers" only a small proportion of the relevant training data inputs.

7.2 Untriangulated models

If the underlying Markov net is not chordal, we must address the problem by triangulating the graph, that is, adding fill-in edges to ensure triangulation. For example, if our graph is a 4-cycle Y_1—Y_2—Y_3—Y_4—Y_1 as in Fig. 11, we can triangulate the graph by adding an arc Y_1—Y_3. This will introduce new cliques Y_1, Y_2, Y_3 and Y_1, Y_3, Y_4 and the corresponding marginals, $\mu_{123}(y_1, y_2, y_3)$ and $\mu_{134}(y_1, y_3, y_4)$. We can then use this new graph to produce the constraints on the marginals:

$$\sum_{y_1} \mu_{123}(y_1, y_2, y_3) = \mu_{23}(y_2, y_3), \qquad \forall y_2, y_3;$$

$$\sum_{y_3} \mu_{123}(y_1, y_2, y_3) = \mu_{12}(y_1, y_2), \qquad \forall y_1, y_2;$$

$$\sum_{y_1} \mu_{134}(y_1, y_3, y_4) = \mu_{34}(y_3, y_4), \qquad \forall y_3, y_4;$$

$$\sum_{y_3} \mu_{134}(y_1, y_3, y_4) = \mu_{13}(y_1, y_3), \qquad \forall y_1, y_3.$$

The new marginal variables appear only in the constraints; they do not add any new basis functions nor change the objective function.

In general, the number of constraints introduced is exponential in the number of variables in the new cliques — the tree-width of the graph. Unfortunately, even sparsely connected networks, for example 2D grids often used in image analysis, have large tree-width. However, we can still solve the QP in the structured primal Eq. (26) or the structured dual Eq. (32) defined by an untriangulated graph. Such a formulation, which enforces only local consistency of marginals, optimizes our objective only over a relaxation of the marginal polytope. However, the learned parameters produce very accurate approximate models in practice, as experiments in Sec. 11.2 demonstrate.

Note that we could also strengthen the untriangulated relaxation without introducing an exponential number of constraints. For example, we can add positive semidefinite constraints on the marginals μ used by Wainwright and Jordan (2003), which tend to improve the approximation of the marginal polytope. Although this and other more complex relaxations are a very interesting area of future development, they are often much more expensive.

The approximate QP does not guarantee that the learned model using *exact* inference minimizes the true objective: (upper-bound on) empirical risk plus regularization. But do we really need these optimal parameters if we cannot perform exact inference? A more useful goal is to make sure that training error is minimized using the *approximate* inference procedure via the untriangulated LP. We conjecture that the parameters learned by the approximate QP in fact do that to some degree. For instance, consider the separable case, where 100% accuracy is achievable on the training data by some parameter setting $\mathbf{w}$ such that approximate inference (using the untriangulated LP) produces integral solutions. Solving the problem as $C \rightarrow \infty$ will find this solution even though it may not be optimal (in terms of the norm of the $\mathbf{w}$) using exact inference. For C in intermediate range, the formulation trades off fractionality of the untriangulated LP solutions with complexity of the weights $||\mathbf{w}||^2$.

8. Associative Markov networks

In the previous section, we considered low-treewidth Markov networks, which allow exact inference and learning. The chief computational bottleneck in applying Markov networks for other large-scale prediction problems is inference, which is NP-hard in general networks suitable in a broad range of practical Markov network structures, including grid-topology networks (Besag, 1986). In this section, we show that in binary AMNs, for which likelihood-based estimation is believed to be intractable, our margin-based framework provides a polynomial-time solution. For the non-binary case, we provide and and approximation that empirical results suggest works well in practice.

The potentials of the AMN are restricted to be associative (i.e. arbitrary node potentials and attractive clique potentials as described in Definition 5.1). Once again we use log-linear combinations of basis functions to represent these potentials. We will need the following assumption to ensure that $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ is associative:

Assumption 8.1 *Basis functions* $\mathbf{f}$ *are component-wise associative for* $\mathcal{G}(\mathbf{x})$ *for any* $(\mathbf{x}, \mathbf{y})$.

Recall that this implies that for cliques larger than one, all basis functions evaluate to 0 for assignments where the values of the nodes are not equal and are non-negative for the assignments where the values of the nodes are equal. To ensure that $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ is associative, it is useful to separate the basis functions with support only on nodes from those with support on larger cliques.

Definition 8.2 *Let* $\dot{\mathbf{f}}$ *be the subset of basis functions* $\mathbf{f}$ *with support only on singleton cliques:*

$$\dot{\mathbf{f}} = \{f \in \mathbf{f} : \forall \mathbf{x} \in \mathcal{X},\ \mathbf{y} \in \mathcal{Y},\ c \in \mathcal{C}(\mathcal{G}(\mathbf{x})),\ |c| > 1,\ f_c(\mathbf{x}_c, \mathbf{y}_c) = 0\}.$$

Let $\ddot{\mathbf{f}} = \mathbf{f} \setminus \dot{\mathbf{f}}$ *be the rest of the basis functions. Let* $\{\dot{\mathbf{w}}, \ddot{\mathbf{w}}\} = \mathbf{w}$ *be the corresponding subsets of parameters.*

It is easy to verify that any non-negative combination of associative functions is associative, and any combination of basis functions with support only on singleton cliques is also associative, so we have:

Lemma 8.3 $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$ *is associative for* $\mathcal{G}(\mathbf{x})$ *for any* $(\mathbf{x}, \mathbf{y})$ *whenever Assumption 8.1 holds and* $\ddot{\mathbf{w}} \geq 0$.

We must make similar associative assumption on the loss function in order to guarantee that the LP inference can handle it.

Assumption 8.4 *The loss function* $\ell(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, \mathbf{y})$ *is associative for* $\mathcal{G}^{(i)}$ *for all* i.

In practice, this restriction is fairly mild, and the Hamming loss, which we use in generalization bounds and experiments, is associative.

Using the above Assumptions 8.1 and 8.4 and some algebra (see Appendix A.4 for derivation), we have the following max-margin QP for AMNs:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i, v \in \mathcal{V}^{(i)}} \xi_{i,v} \\
\text{s.t.} \quad & \mathbf{w}^\top \Delta \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& \ddot{\mathbf{w}}^\top \Delta \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m_{i,c,v}(k) \geq -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k; \\
& \ddot{\mathbf{w}} \geq 0;
\end{aligned} \tag{34}$$

where $\mathbf{f}_{i,c}(k) = \mathbf{f}_{i,c}(k, \ldots, k)$ and $\ell_{i,c}(k) = \ell_{i,c}(k, \ldots, k)$.

While this primal is more complex than the regular M^{3}N factored primal in Eq. (24), the basic structure of the first two sets of constraints remains the same: we have local margin requirements and "credit" passed around through messages $m_{i,c,v}(k)$. The extra constraints are due to the associativity constraints on the resulting model.

The dual of Eq. (34) (see derivation in Sec. A.4) is given by:

$$\max \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\ell_{i,c}(k) - \frac{C}{2}\left|\left|\sum_{i,v\in\mathcal{V}^{(i)},\,k}\mu_{i,v}(k)\Delta\dot{\mathbf{f}}_{i,v}(k)\right|\right|^2 - \frac{C}{2}\left|\left|\ddot{\nu} + \sum_{i,c\in\mathcal{C}^{(i)},\,k}\mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k)\right|\right|^2$$

$$\text{s.t.}\quad \mu_{i,c}(k)\geq 0,\quad \forall i,\ \forall c\in\mathcal{C}^{(i)},\ k; \qquad \sum_{k=1}^{K}\mu_{i,v}(k)=1,\quad \forall i,\ \forall v\in\mathcal{V}^{(i)};$$

$$\mu_{i,c}(k)\leq\mu_{i,v}(k),\quad \forall i,\ \forall c\in\mathcal{C}^{(i)}\setminus\mathcal{V}^{(i)},\ v\in c,\ k;$$

$$\ddot{\nu}\geq 0.$$

In the dual, there are marginals μ for each node and clique, for each value k, similar to Eq. (32). However, the constraints are different, and not surprisingly, are essentially the constraints from the inference LP relaxation in Eq. (13).

The dual and primal solutions are related by

$$\dot{\mathbf{w}} = \sum_{i,v\in\mathcal{V}^{(i)},\,k}\mu_{i,v}(k)\Delta\dot{\mathbf{f}}_{i,v}(k); \qquad \ddot{\mathbf{w}} = \ddot{\nu} + \sum_{i,c\in\mathcal{C}^{(i)},\,k}\mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k).$$

The $\ddot{\nu}$ variables simply ensure that $\ddot{\mathbf{w}}$ are positive (if any component $\sum_{i,c\in\mathcal{C}^{(i)},\,k}\mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k)$ is negative, maximizing the objective will force the corresponding component of $\ddot{\nu}$ to cancel it out). Note that the objective can be written in terms of dot products of node basis functions $\Delta\dot{\mathbf{f}}_{i,v}(k)^\top\Delta\dot{\mathbf{f}}_{j,\bar{v}}(\bar{k})$, so they can be kernelized. Unfortunately, the edge basis functions cannot be kernelized because of the non-negativity constraint.

For $K = 2$, the LP inference is exact, so that Eq. (34) learns *exact* max-margin weights for Markov networks of *arbitrary* topology. For $K > 2$, the linear relaxation leads to a strengthening of the constraints on $\mathbf{w}$ by potentially adding constraints corresponding to fractional assignments as in the case of untriangualated networks. Thus, the optimal choice $\mathbf{w}, \xi$ for the original QP may no longer be feasible, leading to a different choice of weights. However, as our experiments show, these weights tend to do well in practice.

9. Generalization bound

In this section, we show a generalization bound for the task of structured classification that allows us to relate the error rate on the training set to the generalization error. To the best of our knowledge, this bound is the first to deal with structured error, such as the Hamming distance. Our analysis of Hamming loss allows to prove a significantly stronger result than previous bounds for the 0/1 loss, as we detail below.

Our goal in structured classification is often to minimize the number of misclassified labels, or the Hamming distance between $\mathbf{y}$ and $h(\mathbf{x})$. An appropriate error function is the *average per-label loss*

$$\mathcal{L}(\mathbf{w},\mathbf{x},\mathbf{y}) = \frac{1}{L}\ell^H(\mathbf{y}, \arg\max_{\mathbf{y}'}\mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y}')),$$

where L is the number of label variables in $\mathbf{y}$. As in other generalization bounds for margin-based classifiers, we relate the generalization error to the margin of the classifier. Consider an upper bound on the above loss:

$$\mathcal{L}(\mathbf{w},\mathbf{x},\mathbf{y}) \leq \overline{\mathcal{L}}(\mathbf{w},\mathbf{x},\mathbf{y}) = \max_{\mathbf{y}':\ \mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}')} \frac{1}{L}\ell^H(\mathbf{y},\mathbf{y}').$$

This upper bound is tight if $\mathbf{y} = \arg\max_{\mathbf{y}'} \mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}')$, Otherwise, it is adversarial: it picks from all $\mathbf{y}'$ which are better $(\mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}'))$, one that maximizes the Hamming distance from $\mathbf{y}$. We can now define a *γ-margin per-label loss*:

$$\mathcal{L}(\mathbf{w},\mathbf{x},\mathbf{y}) \leq \overline{\mathcal{L}}(\mathbf{w},\mathbf{x},\mathbf{y}) \leq \mathcal{L}^\gamma(\mathbf{w},\mathbf{x},\mathbf{y}) = \max_{\mathbf{y}':\ \mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}') + \gamma \ell^H(\mathbf{y},\mathbf{y}')} \frac{1}{L}\ell^H(\mathbf{y},\mathbf{y}').$$

This upper bound is even more adversarial: it is tight if $\mathbf{y} = \arg\max_{\mathbf{y}'}[\mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}') + \ell^H(\mathbf{y},\mathbf{y}')]$, otherwise, it picks from all $\mathbf{y}'$ which are better *when helped by* $\gamma\ell^H(\mathbf{y},\mathbf{y}')$, one that maximizes the Hamming distance from $\mathbf{y}$. Note that the loss we minimize in the max-margin formulation is very closely related (although not identical to) this upper bound.

We can now prove that the generalization accuracy of any hypothesis $\mathbf{w}$ is bounded by its empirical γ-margin per-label loss, plus a term that grows inversely with the margin.To state the bound, we need to define several other factors it depends upon. Let N_c be the maximum number of cliques in $\mathcal{G}(\mathbf{x})$, V_c be the maximum number of values in a clique $|\mathbf{Y}_c|$, q be the maximum number of cliques that have a variable in common, and R_c be an upper-bound on the 2-norm of clique basis functions. Consider a first-order sequence model as an example, with L as the maximum length, and V the number of values a variable takes. Then $N_c = 2L - 1$ since we have L node cliques and $L - 1$ edge cliques; $V_c = V^2$ because of the edge cliques; and $q = 3$ since nodes in the middle of the sequence participate in 3 cliques: previous-current edge clique, node clique, and current-next edge clique.

Theorem 9.1 *For the family of hypotheses parameterized by* $\mathbf{w}$*, and any* $\delta > 0$*, there exists a constant* K *such that for any* $\gamma > 0$ *per-label margin, and* $m > 1$ *samples, the expected per-label loss is bounded by:*

$$\mathbf{E}_D[\mathcal{L}(\mathbf{w},\mathbf{x},\mathbf{y})] \quad \leq \quad \mathbf{E}_S[\mathcal{L}^\gamma(\mathbf{w},\mathbf{x},\mathbf{y})] + \sqrt{\frac{K}{m}\left[\frac{R_c^2||\mathbf{w}||^2 q^2}{\gamma^2}[\ln m + \ln N_c + \ln V_c] + \ln\frac{1}{\delta}\right]} \ ,$$

with probability at least $1 - \delta$. ▮

Proof: See Appendix B for the proof details and the exact value of the constant K. ▮

The first term upper bounds the training error of $\mathbf{w}$. Low loss $\mathbf{E}_S[\mathcal{L}^\gamma(\mathbf{w},\mathbf{x},\mathbf{y})]$ at high margin γ quantifies the confidence of the prediction model. The second term depends on $||\mathbf{w}||/\gamma$, which corresponds to the complexity of the classifier (normalized by the margin level). Thus, the result provides a bound to the generalization error that trades off the *effective* complexity of the hypothesis space with the training error.

The proof uses a covering number argument analogous to previous results in SVMs (Zhang, 2002). However we propose a novel method for covering the space of structured prediction models by using a cover of the individual clique basis function differences $\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)$. This new type of cover is polynomial in the number of cliques, yielding significant improvements in the bound. Specifically, our bound has a logarithmic dependence on the number of cliques ($\ln N_c$) and depends only on the 2-norm of the basis functions per-clique (R_c). This is a significant gain over the previous result of Collins (2001) for 0/1 loss, which has linear dependence (inside the square root) on the number of nodes (L), and depends on the joint 2-norm of all of the basis functions for an example (which is $\sim N_c R_c$). Such a result was, until now, an open problem for margin-based sequence classification (Collins, 2001). Finally, for sequences, note that if $\frac{L}{m} = \mathcal{O}(1)$ (for example, in OCR, if the number of instances is at least a constant times the length of a word), then our bound is independent of the number of labels L.

10. Solving the M^{3}N QP

In this section, we describe an efficient algorithm for estimation of low-treewidth networks. Although the number of variables and constraints in the structured dual in Eq. (32) is polynomial in the size of the data, unfortunately, for standard QP solvers, the problem is often too large even for small training sets. Instead, we use a coordinate dual ascent method analogous to the sequential minimal optimization (SMO) used for SVMs Platt (1999).

Let us begin by considering the primal and dual QPs for multi-class SVMs:

$$\begin{aligned} &\min\ \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i && \max\ \sum_{i,y}\alpha_i(y)\ell_i(y) - \frac{1}{2}C\left|\left|\sum_{i,y}\alpha_i(y)\Delta\mathbf{f}_i(y)\right|\right|^2 \\ &\text{s.t.}\ \ \mathbf{w}^\top\Delta\mathbf{f}_i(y) \geq \ell(y) - \xi_i,\ \ \forall i, y. && \text{s.t.}\ \ \sum_y \alpha_i(y) = 1,\ \forall i;\ \ \alpha_i(y) \geq 0,\ \forall i, y. \end{aligned}$$

The KKT conditions Bertsekas (1999); Boyd and Vandenberghe (2004) provide sufficient and necessary criteria for optimality of a dual solution α. As we describe below, these conditions have certain locality with respect to each example i, which allows us to perform the search for optimal α by repeatedly considering one example at a time.

A feasible dual solution α and a primal solution defined by:

$$\begin{aligned} \mathbf{w} &= C\sum_{i,y}\alpha_i(y)\Delta\mathbf{f}_i(y) \qquad (35) \\ \xi_i &= \max_y\,[\ell_i(y) - \mathbf{w}\Delta\mathbf{f}_i(y)] = \max_y\,[\ell_i(y) + \mathbf{w}^\top\mathbf{f}_i(y)] - \mathbf{w}^\top\mathbf{f}_i(y^{(i)}), \end{aligned}$$

are optimal if they satisfy the following two types of constraints:

$$\begin{aligned} \alpha_i(y) = 0 \ &\Rightarrow\ \mathbf{w}^\top\Delta\mathbf{f}_i(y) > \ell_i(y) - \xi_i; && \text{(KKT1)} \\ \alpha_i(y) > 0 \ &\Rightarrow\ \mathbf{w}^\top\Delta\mathbf{f}_i(y) = \ell_i(y) - \xi_i. && \text{(KKT2)}, \end{aligned}$$

We can express these conditions as

$$\begin{aligned} \alpha_i(y) = 0 \ &\Rightarrow\ \mathbf{w}^\top\mathbf{f}_i(y) + \ell_i(y) < \max_{y'}\,[\mathbf{w}^\top\mathbf{f}_i(y') + \ell_i(y')]; && \text{(KKT1)} \\ \alpha_i(y) > 0 \ &\Rightarrow\ \mathbf{w}^\top\mathbf{f}_i(y) + \ell_i(y) = \max_{y'}\,[\mathbf{w}^\top\mathbf{f}_i(y') + \ell_i(y')]. && \text{(KKT2)} \end{aligned}$$

To simplify the notation, we define

$$v_i(y) = \mathbf{w}^\top\mathbf{f}_i(y) + \ell_i(y); \quad v_i(\overline{y}) = \max_{y' \neq y}[\mathbf{w}^\top\mathbf{f}_i(y') + \ell_i(y')].$$

```
1. Initialize: $\alpha_i(y) = \mathbb{1}(y = y^{(i)}),\ \ \forall\, i, y$.
2. Set $violation = 0$,
3. For each $i$,
4.   If $\alpha_i$ violates (KKT1) or (KKT2),
5.     Set $violation = 1$,
6.     Find feasible $\alpha'_i$ such that $\mathcal{Q}(\alpha'_i, \alpha_{-i}) > \mathcal{Q}(\alpha_i, \alpha_{-i})$ and set $\alpha_i = \alpha'_i$.
7. If $violation = 1$ goto 2.
```

Figure 12: Block-coordinate dual ascent.

With these definitions, we have

$$\alpha_i(y) = 0 \Rightarrow v_i(y) < v_i(\overline{y}); \quad \text{(KKT1)} \qquad \alpha_i(y) > 0 \Rightarrow v_i(y) \geq v_i(\overline{y}); \quad \text{(KKT2)}.$$

In practice, however, we will enforce KKT conditions up to a given tolerance $0 < \epsilon \ll 1$.

$$\alpha_i(y) = 0 \Rightarrow v_i(y) \leq v_i(\overline{y}) + \epsilon; \qquad \alpha_i(y) > 0 \Rightarrow v_i(y) \geq v_i(\overline{y}) - \epsilon. \tag{36}$$

Essentially, $\alpha_i(y)$ can be *zero* only if $v_i(y)$ is at most ϵ *larger than the all others*. Conversely, $\alpha_i(y)$ can be *non-zero* only if $v_i(y)$ is at most ϵ *smaller than the all others*.

Note that the normalization constraints on the dual variables α are local to each example i. This allows us to perform dual block-coordinate ascent where a block corresponds to the vector of dual variables α_i for a single example i. The general form of block-coordinate ascent algorithm as shown in Fig. 12 is essentially coordinate ascent on blocks α_i, maintaining the feasibility of the dual. When optimizing with respect to a single block i, the objective function can be split into two terms:

$$\mathcal{Q}(\alpha) = \mathcal{Q}(\alpha_{-i}) + \mathcal{Q}(\alpha_i, \alpha_{-i}),$$

where α_{-i} denotes all dual α_k variables for k other than i. Only the second part of the objective $\mathcal{Q}(\alpha_i, \alpha_{-i})$ matters for optimizing with respect to α_i. The algorithm starts with a feasible dual solution α and improves the objective block-wise until all KKT conditions are satisfied. Checking the constraints requires computing $\mathbf{w}$ and ξ from α according to Eq. (35).

As long as the local ascent step over α_i is guaranteed to improve the objective when KKT conditions are violated, the algorithm will converge to the global maximum in a finite number of steps (within the precision). This allows us to focus on efficient updates to a single block of α_i at a time.

Let $\alpha'_i(y) = \alpha_i(y) + \lambda(y)$. Note that $\sum_y \lambda(y) = 0$ and $\alpha_i(y) + \lambda(y) \geq 0$ so that α'_i is feasible. We can write the objective $\mathcal{Q}(\alpha_{-i}) + \mathcal{Q}(\alpha'_i, \alpha_{-i})$ in terms of λ and α:

$$\sum_{j,y} \alpha_j(y)\ell_j(y) + \sum_y \lambda(y)\ell_i(y) - \frac{1}{2}C \left|\left| \sum_y \lambda(y)\Delta\mathbf{f}_i(y) + \sum_{j,y} \alpha_j(y)\Delta\mathbf{f}_j(y) \right|\right|^2.$$

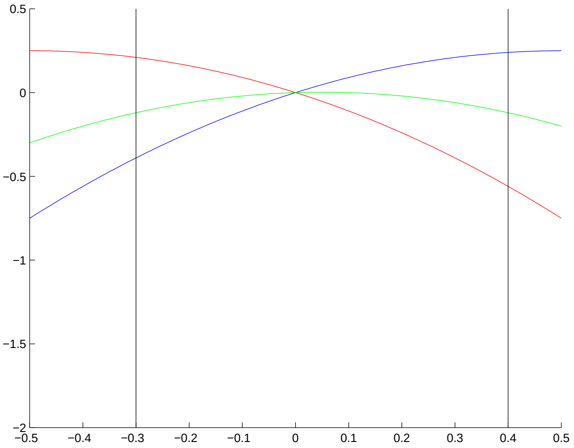


Figure 13: Representative examples of the SMO subproblem. Horizonal axis represents δ with two vertical lines depicting the upper and lower bounds c and d. Vertical axis represents the objective. Optimum either occurs at the maximum of the parabola if it is feasible or the upper or lower bound otherwise.

By dropping all terms that do not involve λ, and making the substitution $\mathbf{w} = C\sum_{j,y} \alpha_j(y)\Delta\mathbf{f}_j(y)$, we get:

$$\sum_y \lambda(y)\ell_i(y) - \mathbf{w}^\top\left(\sum_y \lambda(y)\Delta\mathbf{f}_i(y)\right) - \frac{1}{2}C\left|\left|\sum_y \lambda(y)\Delta\mathbf{f}_i(y)\right|\right|^2.$$

Since $\sum_y \lambda(y) = 0$,

$$\sum_y \lambda(y)\Delta\mathbf{f}_i(y) = \sum_y \lambda(y)\mathbf{f}_i(y^{(i)}) - \sum_y \lambda(y)\mathbf{f}_i(y) = -\sum_y \lambda(y)\mathbf{f}_i(y).$$

Below we also make the substitution $v_i(y) = \mathbf{w}^\top\mathbf{f}_i(y) + \ell_i(y)$ to get the optimization problem for λ:

$$\begin{aligned} \max \quad & \sum_y \lambda(y)v_i(y) - \frac{1}{2}C\left|\left|\sum_y \lambda(y)\mathbf{f}_i(y)\right|\right|^2 \\ \text{s.t.} \quad & \sum_y \lambda(y) = 0; \quad \alpha_i(y) + \lambda(y) \geq 0, \ \forall y. \end{aligned}$$

10.1 SMO

We do not need to solve the optimization subproblem above at each pass through the data. All that is required is an ascent step, not a full optimization. Sequential Minimal Optimization (SMO) approach takes an ascent step that modifies the least number of variables. In our case, we have the simplex constraint, so we must change at least two variables in order to respect the normalization constraint (by moving weight from one dual variable to another). We address a strategy for selecting the two variables in the next section, but for now assume we have picked $\lambda(y')$ and $\lambda(y'')$. Then we have $\delta = \lambda(y') = -\lambda(y'')$ in order to sum to 1. The optimization problem becomes a single variable quadratic program in δ:

$$\begin{aligned} \max \quad & [v_i(y') - v_i(y'')]\delta - \frac{1}{2}C||\mathbf{f}_i(y') - \mathbf{f}_i(y'')||^2\delta^2 \\ \text{s.t.} \quad & \alpha_i(y') + \delta \geq 0; \quad \alpha_i(y'') - \delta \geq 0. \end{aligned} \tag{37}$$

With $a = v_i(y') - v_i(y'')$, $b = C||\mathbf{f}_i(y') - \mathbf{f}_i(y'')||^2$, $c = -\alpha_i(y')$, $d = \alpha_i(y'')$, we have:

$$\max\,[a\delta - \frac{b}{2}\delta^2] \;\text{ s.t. }\; c \leq \delta \leq d, \tag{38}$$

where the optimum is achieved at the maximum of the parabola a/b if $c \leq a/b \leq d$ or at the boundary c or d (see Fig. 10.1). Hence the solution is given by simply clipping a/b:

$$\delta^* = \max(c, \min(d, a/b)).$$

The key advantage of SMO is the simplicity of this update. Computing the coefficients involves dot products (or kernel evaluations) to compute $\mathbf{w}^\top \mathbf{f}_i(y')$ and $\mathbf{w}^\top \mathbf{f}_i(y'')$ as well as $(\mathbf{f}_i(y') - \mathbf{f}_i(y''))^\top(\mathbf{f}_i(y') - \mathbf{f}_i(y''))$.

10.2 Selecting SMO pairs

How do we actually select such a pair to guarantee that we make progress in optimizing the objective? Note that at least one of the assignments y must violate (KKT1) or (KKT2), because otherwise α_i is optimal with respect to the current α_{-i}. The selection algorithm is outlined in Fig. 14.

The first variable in the pair, y', corresponds to a violated condition, while the second variable, y'', is chosen to guarantee that solving Eq. (37) will result in improving the objective. There are two cases, corresponding to violation of KKT1 and violation of KKT2.

Case KKT1. $\alpha_i(y') = 0$ but $v_i(y') > v_i(\overline{y'}) + \epsilon$. This is the case where i, y' is a not support vector but should be. We would like to increase $\alpha_i(y')$, so we need $\alpha_i(y'') > 0$ to borrow from. There will always be a such a y'' since $\sum_y \alpha_i(y) = 1$ and $\alpha_i(y') = 0$. Since $v_i(y') > v_i(\overline{y'}) + \epsilon$, $v_i(y') > v_i(y'') + \epsilon$, so the linear coefficient in Eq. (38) is $a = v_i(y') - v_i(y'') > \epsilon$. Hence the unconstrained maximum is positive $a/b > 0$. Since the upper-bound $d = \alpha_i(y'') > 0$, we have enough freedom to improve the objective.

Case KKT2. $\alpha_i(y') > 0$ but $v_i(y') < v_i(\overline{y'}) - \epsilon$. This is the case where i, y' is a support vector but should not be. We would like to decrease $\alpha_i(y')$, so we need $v_i(y'') > v_i(y')$ so that $a/b < 0$. There will always be a such a y'' since $v_i(y') < v_i(\overline{y'}) - \epsilon$. Since the lower-bound $c = -\alpha_i(y') < 0$, again we have enough freedom to improve the objective.

Since at each iteration we are guaranteed to improve the objective if the KKT conditions are violated and the objective is bounded, we can use the SMO in the block-coordinate ascent algorithm to converge in a finite number of steps. To the best of our knowledge, there are no upper bounds on the speed of convergence of SMO, but experimental evidence has shown it a very effective algorithm for SVMs Platt (1999). Of course, we can improve the speed of convergence by adding heuristics in the selection of the pair, as long as we guarantee that improvement is possible when KKT conditions are violated.

```
1.  Set $violation = 0$.
2.  For each $y$,
3.    KKT1: If $\alpha_i(y) = 0$ and $v_i(y) > v_i(\overline{y}) + \epsilon$,
4.      Set $y' = y$ and $violation = 1$ and goto 7.
5.    KKT2: If $\alpha_i(y) > 0$ and $v_i(y) < v_i(\overline{y}) - \epsilon$,
6.      Set $y' = y$ and $violation = 2$ and goto 7.
7.  If $violation > 0$,
8.    For each $y \neq y'$,
9.      If $violation = 1$ and $\alpha_i(y) > 0$,
10.       Set $y'' = y$ and goto 13.
11.     If $violation = 2$ and $v_i(y) > v_i(y')$,
12.       Set $y'' = y$ and goto 13.
13. Return $y'$ and $y''$.
```

Figure 14: SMO pair selection.

10.3 Structured SMO

Clearly, we cannot perform the above SMO updates in the space of α directly for the structured problems, since the number of α variables is exponential. The constraints on μ variables are much more complicated, since each μ participates not only in non-negativity and normalization constraints, but also clique-agreement constraints. We cannot limit our ascent steps to changing only two μ variables at a time, because in order to make a change in one clique and stay feasible, we need to modify variables in overlapping cliques. Fortunately, we can perform SMO updates on α variables implicitly in terms of the marginal dual variables μ.

The diagram in Fig. 10.3 shows the abstract outline of the algorithm. The key steps in the SMO algorithm are checking for violations of the KKT conditions, selecting the pair $\mathbf{y}'$ and $\mathbf{y}''$, computing the corresponding coefficients a, b, c, d and updating the dual. We will show how to do these operations by doing all the hard work in terms of the polynomially many marginal μ_i variables and auxiliary "max-marginals" variables.

10.3.1 Structured KKT conditions

As before, we define $v_i(\mathbf{y}) = \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$. The KKT conditions are, for all $\mathbf{y}$:

$$\alpha_i(\mathbf{y}) = 0 \Rightarrow v_i(\mathbf{y}) \leq v_i(\overline{\mathbf{y}}); \qquad \alpha_i(\mathbf{y}) > 0 \Rightarrow v_i(\mathbf{y}) \geq v_i(\overline{\mathbf{y}}). \tag{39}$$

Of course, we cannot check these explicitly. Instead, we define max-marginals for each clique in the junction tree $c \in \mathcal{T}^{(i)}$ and its values $\mathbf{y}_c$, as:

$$\widehat{v}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y} \sim \mathbf{y}_c} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})], \qquad \widehat{\alpha}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y}).$$

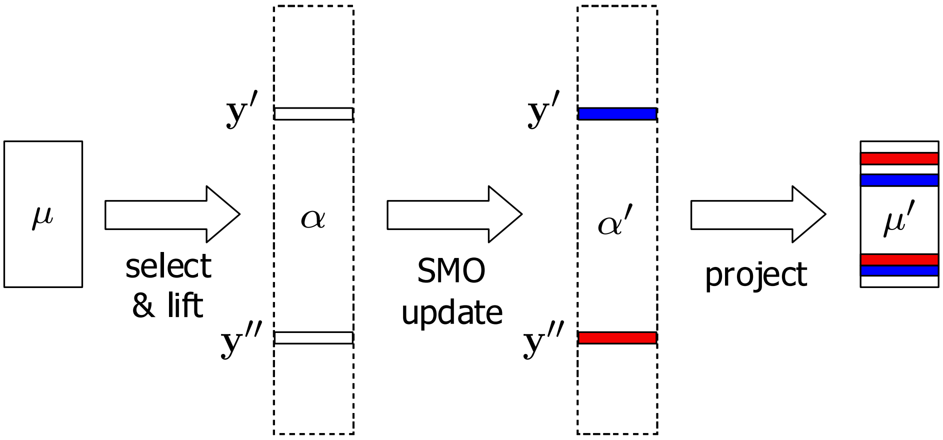


Figure 15: Structured SMO diagram. We use marginals μ to select an appropriate pair of instantiations $\mathbf{y}'$ and $\mathbf{y}''$ and reconstruct their α values. We then perform the simple SMO update and project the result back onto the marginals.

We also define $\widehat{v}_{i,c}(\overline{\mathbf{y}_c}) = \max_{\mathbf{y}'_c \neq \mathbf{y}_c} \widehat{v}_{i,c}(\mathbf{y}'_c) = \max_{\mathbf{y} \not\sim \mathbf{y}_c} [\mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})]$. Note that we do not explicitly represent $\alpha_i(\mathbf{y})$, but we can reconstruct the maximum-entropy one from the marginals μ_i by using Eq. (33). Both $\widehat{v}_{i,c}(\mathbf{y}_c)$ and $\widehat{\alpha}_{i,c}(\mathbf{y}_c)$ can be computed by using the Viterbi algorithm (one pass propagation towards the root and one outwards from the root Cowell et al. (1999)). We can now express the KKT conditions in terms of the max-marginals for each clique $c \in \mathcal{T}^{(i)}$ and its values $\mathbf{y}_c$:

$$\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0 \Rightarrow \widehat{v}_{i,c}(\mathbf{y}_c) \leq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}); \qquad \widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0 \Rightarrow \widehat{v}_{i,c}(\mathbf{y}_c) \geq \widehat{v}_{i,c}(\overline{\mathbf{y}_c}). \tag{40}$$

Theorem 10.1 *The KKT conditions in Eq. (39) and Eq. (40) are equivalent.*

Proof: See Appendix C.

10.3.2 Structured SMO pair selection and update

As in multi-class problems, we will select the first variable in the pair, $\mathbf{y}'$, corresponding to a violated condition, while the second variable, $\mathbf{y}''$, to guarantee that solving Eq. (37) will result in improving the objective. Having selected $\mathbf{y}'$ and $\mathbf{y}''$, the coefficients for the one-variable QP in Eq. (38) are $a = v_i(\mathbf{y}') - v_i(\mathbf{y}'')$, $b = C||\mathbf{f}_i(\mathbf{y}') - \mathbf{f}_i(\mathbf{y}'')||^2$, $c = -\alpha_i(\mathbf{y}')$, $d = \alpha_i(\mathbf{y}'')$. As before, we enforce approximate KKT conditions in the algorithm in Fig. 16. We have two cases, corresponding to violation of KKT1 and violation of KKT2.

Case KKT1. $\widehat{\alpha}_{i,c}(\mathbf{y}'_c) = 0$ but $\widehat{v}_{i,c}(\mathbf{y}'_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) + \epsilon$. We have set $\mathbf{y}' = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} v_i(\mathbf{y})$, so $v_i(\mathbf{y}') = \widehat{v}_{i,c}(\mathbf{y}'_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) + \epsilon > v_i(\overline{\mathbf{y}'}) + \epsilon$ and $\alpha_i(\mathbf{y}') = 0$. This is the case where $i, \mathbf{y}'$ is a not support vector but should be. We would like to increase $\alpha_i(\mathbf{y}')$, so we need $\alpha_i(\mathbf{y}'') > 0$ to borrow from. There will always be a such a y'' (with $\mathbf{y}''_c \neq \mathbf{y}'_c$) since $\sum_y \alpha_i(\mathbf{y}) = 1$ and $\alpha_i(\mathbf{y}') = 0$. We can find one by choosing $\mathbf{y}_c$ for which $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, which guarantees that for $\mathbf{y}''_c = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})$, $\alpha_i(\mathbf{y}'') > 0$. Since $v_i(\mathbf{y}') \geq v_i(\overline{\mathbf{y}'}) + \epsilon$, $v_i(\mathbf{y}') \geq v_i(\mathbf{y}'') + \epsilon$, so the linear coefficient in Eq. (38) is $a = v_i(\mathbf{y}') - v_i(\mathbf{y}'') > \epsilon$. Hence the unconstrained maximum

```
1.  Set $violation = 0$.
2.  For each $c \in \mathcal{T}^{(i)}$, $\mathbf{y}_c$
3.    KKT1: If $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0$, and $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c}) + \epsilon$,
4.      Set $\mathbf{y}'_c = \mathbf{y}_c$, $\mathbf{y}' = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} v_i(\mathbf{y})$ and $violation = 1$ and goto 7.
5.    KKT2: If $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, and $\widehat{v}_{i,c}(\mathbf{y}_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}_c}) - \epsilon$,
6.      Set $\mathbf{y}'_c = \mathbf{y}_c$, $\mathbf{y}' = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})$ and $violation = 2$ and goto 7.
7.  If $violation > 0$,
8.    For each $\mathbf{y}_c \neq \mathbf{y}'_c$,
9.      If $violation = 1$ and $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$,
10.       Set $\mathbf{y}''_c = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})$ and goto 13.
11.     If $violation = 2$ and $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\mathbf{y}'_c)$,
12.       Set $\mathbf{y}'' = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} v_i(\mathbf{y})$ and goto 13.
13. Return $\mathbf{y}'$ and $\mathbf{y}''$.
```

Figure 16: Structured SMO pair selection.

is positive $a/b > 0$. Since the upper-bound $d = \alpha_i(\mathbf{y}'') > 0$, we have enough freedom to improve the objective.

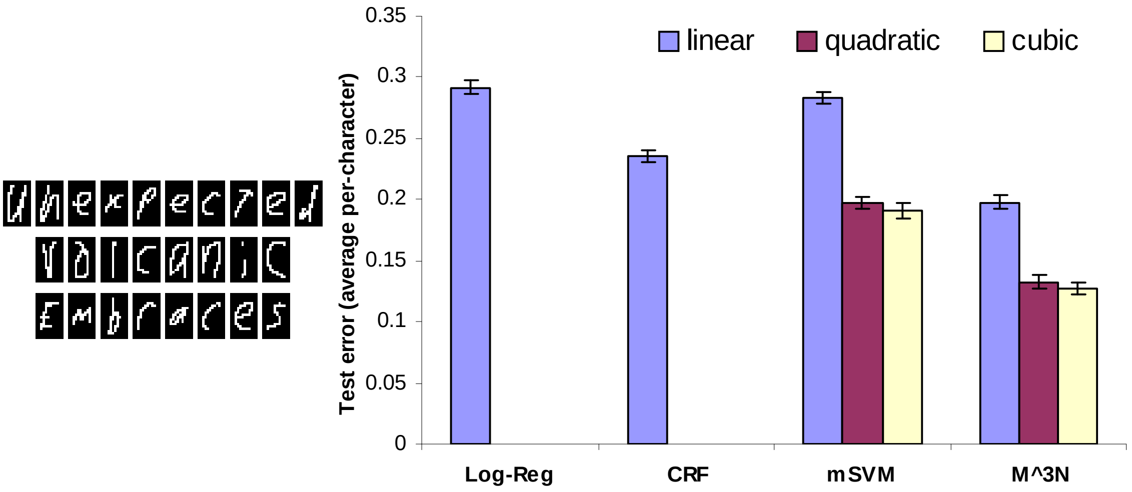
Case KKT2. $\widehat{\alpha}_{i,c}(\mathbf{y}'_c) > 0$ but $\widehat{v}_{i,c}(\mathbf{y}'_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) - \epsilon$. We have set $\mathbf{y}' = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y})$, so $\alpha_i(\mathbf{y}') = \widehat{\alpha}_{i,c}(\mathbf{y}'_c) > 0$ and $v_i(\mathbf{y}') < \widehat{v}_{i,c}(\mathbf{y}'_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}) - \epsilon < v_i(\overline{\mathbf{y}'}) - \epsilon$. This is the case where $i, \mathbf{y}'$ is a support vector but should not be. We would like to decrease $\alpha_i(\mathbf{y}')$, so we need $v_i(\mathbf{y}'') > v_i(\mathbf{y}')$ so that $a/b < 0$. There will always be a such a $\mathbf{y}''$ since $v_i(\mathbf{y}') < v_i(\overline{y'}) - \epsilon$. We can find one by choosing $\mathbf{y}_c$ for which $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\mathbf{y}_c) - \epsilon$, which guarantees that for $\mathbf{y}''_c = \arg\max_{\mathbf{y} \sim \mathbf{y}_c} v_i(\mathbf{y})$, $v_i(\mathbf{y}'') > v_i(\mathbf{y}') - \epsilon$, Since the lower-bound $c = -\alpha_i(\mathbf{y}') < 0$, again we have enough freedom to improve the objective.

Having computed new values $\alpha'_i(\mathbf{y}') = \alpha_i(\mathbf{y}') + \delta$ and $\alpha'_i(\mathbf{y}'') = \alpha_i(\mathbf{y}') - \delta$, we need to project this change onto the marginal dual variables μ_i. The only marginal affected are the ones consistent with $\mathbf{y}'$ and/or $\mathbf{y}''$, and the change is very simple:

$$\mu'_{i,c}(\mathbf{y}_c) = \mu_{i,c}(\mathbf{y}_c) + \delta \mathbb{1}(\mathbf{y}_c \sim \mathbf{y}') - \delta \mathbb{1}(\mathbf{y}_c \sim \mathbf{y}'').$$

11. Experiments

We evaluated our framework on the tasks of handwriting recognition, 3D terrain segmentation and web page classification. These tasks range over many types of networks: sequences for handwriting recognition, 3D-grid type AMNs for terrain segmentation and arbitrary topology AMNs and general MNs for web page classification.



(a)

(b)

Figure 17: (a) 3 example words from the OCR data set; (b) OCR: Average per-character test error for logistic regression, CRFs, multiclass SVMs, and M^{3}Ns, using linear, quadratic, and cubic kernels.

11.1 Handwriting recognition

We selected a subset of $\sim$ 6100 handwritten words, with average length of $\sim$ 8 characters, from 150 human subjects, from the data set collected by Kassel (1995). Each word was divided into characters, each character was rasterized into an image of 16 by 8 binary pixels. (See Fig. 17(a).) In our framework, the image for each word corresponds to $\mathbf{x}$, a label of an individual character to $\mathcal{Y}_j$, and a labeling for a complete word to $\mathcal{Y}$. Each label $\mathcal{Y}_j$ takes values from one of 26 classes $\{a, \ldots, z\}$.

The data set is divided into 10 folds of $\sim$ 600 training and $\sim$ 5500 testing examples. The accuracy results, summarized in Fig. 17(b), are averages over the 10 folds. We implemented a selection of state-of-the-art classification algorithms: *independent label approaches*, which do not consider the correlation between neighboring characters — logistic regression, multi-class SVMs as described in Eq. (8), and one-against-all SVMs (whose performance was slightly lower than multi-class SVMs); and *sequence approaches* — CRFs, and our proposed M^3 networks. Logistic regression and CRFs are both trained by maximizing the conditional likelihood of the labels given the features, using a zero-mean diagonal Gaussian prior over the parameters, with a standard deviation between 0.1 and 1. The other methods are trained by margin maximization. Our features for each label $\mathcal{Y}_j$ are the corresponding image of ith character. For the sequence approaches (CRFs and M^3), we used an indicator basis function to represent the correlation between $\mathcal{Y}_j$ and $\mathcal{Y}_{i+1}$. For margin-based methods (SVMs and M^3), we were able to use kernels (both quadratic and cubic were evaluated) to increase the dimensionality of the feature space. We used the structured SMO algorithm with about 30-40 iterations through the data. Using these high-dimensional feature spaces in CRFs is not feasible because of the enormous number of parameters.

Fig. 17(b) shows two types of gains in accuracy: First, by using kernels, margin-based methods achieve a very significant gain over the respective likelihood maximizing methods. Second, by using sequences, we obtain another significant gain in accuracy. Interestingly, the error rate of our method using linear features is 16% lower than that of CRFs, and about the same as multi-class SVMs with cubic kernels. Once we use cubic kernels our error rate is 45% lower than CRFs and about 33% lower than the best previous approach. For comparison, the previously published results, although using a different setup (e.g., a larger training set), are about comparable to those of multiclass SVMs.

11.2 Hypertext classification

We also tried out our framework on the *WebKB* dataset (Craven et al., 1998). The data set contains webpages from four different Computer Science departments: Cornell, Texas, Washington and Wisconsin. Each page is classified as one of *course, faculty, student, project* or *other*. The data set is problematic in that the category *other* is a grab-bag of pages of many different types. The number of pages classified as *other* is quite large, so that a baseline algorithm that simply always selected *other* as the label would get an average accuracy of 75%. We could restrict attention to just the pages with the four other labels, but in a structured classification setting, the deleted webpages might be useful in terms of their interactions with other webpages. Hence, we compromised by eliminating all *other* pages with fewer than three outlinks, making the number of *other* pages commensurate with the other categories. The resulting category distribution is: course (237), faculty (148), other (332), research-project (82) and student (542). The number of remaining pages for each school are: Cornell (280), Texas (292), Washington (315) and Wisconsin (454). The number of links for each school are: Cornell (574), Texas (574), Washington (728) and Wisconsin (1614).

For each page, we have access to the entire html of the page and the links to other pages. Our goal is to collectively classify webpages into one of these five categories. In all of our experiments, we learn a model from three schools and test the performance of the learned model on the remaining school, thus evaluating the generalization performance of the different models. We used $C \in [0.1, 10]$ and took the best setting for all models.

Unfortunately, we cannot directly compare our accuracy results with previous work because different papers use different subsets of the data and different training/test splits. However, we compare to standard text classifiers such as Naive Bayes, Logistic Regression, and Support Vector Machines, which have been demonstrated to be successful on this data set (Joachims, 1999).

11.2.1 Flat models

The baseline approach we tried predicts the categories based on just the text content on the webpage. The text of the webpage is represented using a set of binary attributes that indicate the presence of different words on the page. We found that stemming and feature selection did not provide much benefit and simply pruned words that appeared in fewer than three documents in each of the three schools in the training data. We also experimented

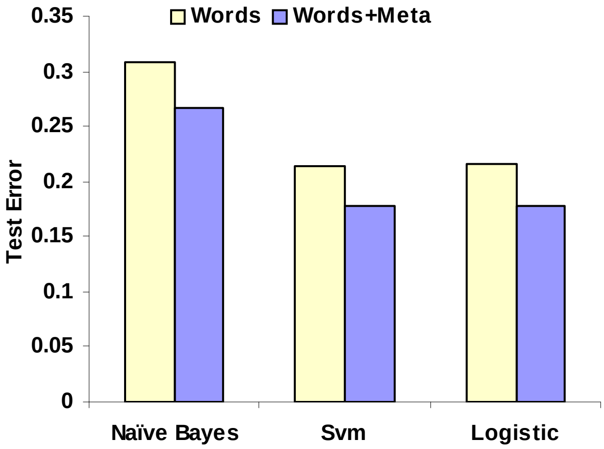


Figure 18: Comparison of Naive Bayes, Svm, and Logistic on WebKB, with and without meta-data features. (Only averages over the 4 schools are shown here.)

with incorporating meta-data: words appearing in the title of the page, in anchors of links to the page and in the last header before a link to the page (Yang et al., 2002). Note that meta-data, although mostly originating from pages linking into the considered page, are easily incorporated as features, i.e. the resulting classification task is still flat feature-based classification. Our first experimental setup compares three well-known text classifiers — Naive Bayes, linear support vector machines (Svm), and logistic regression (Logistic) — using words and meta-words. The results, shown in Fig. 18, show that the two discriminative approaches outperform Naive Bayes. Logistic and Svm give very similar results. The average error over the 4 schools was reduced by around 4% by introducing the meta-data attributes.

Incorporating meta-data gives a significant improvement, but we can take additional advantage of the correlation in labels of related pages by classifying them collectively. We want to capture these correlations in our model and use them for transmitting information between linked pages to provide more accurate classification. We experimented with several models that captured such correlations.

11.2.2 Link model

Our first model captures direct correlations between labels of linked pages. These correlations are very common in our data: courses and research projects almost never link to each other; faculty rarely link to each other; students have links to all categories but mostly courses. The Link model captures this correlation through links: in addition to the local bag of words and meta-data attributes, we introduce a pairwise clique between the labels of any two pages that are linked.We train this model using maximum conditional likelihood (labels given the words and the links) and maximum margin.

We also compare to a directed graphical model to contrast discriminative and generative models of relational structure. The Exists-ML model is a (partially) generative model

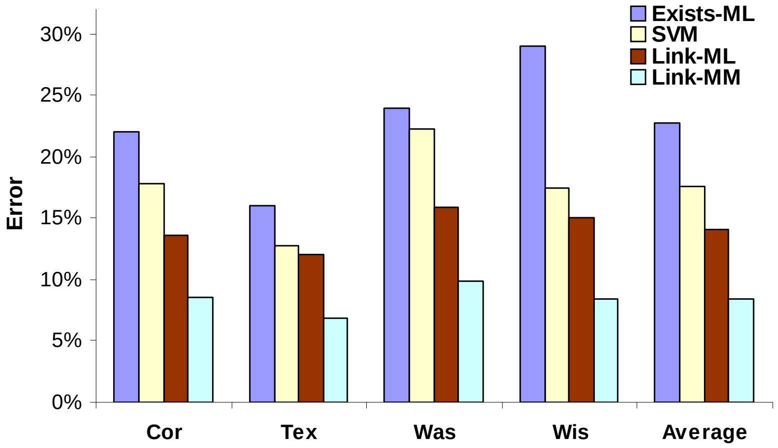


Figure 19: Comparison of flat versus collective classification on WebKB: SVM, Exists model with logistic regression and the Link model estimated using the maximum likelihood (ML) and the maximum margin (MM) criteria.

proposed by Getoor et al. (2001). For each page, a logistic regression model predicts the page label given the words and meta-features. Then a simple generative model specifies a probability distribution over the existence of links between pages conditioned on both pages' labels. Concretely, we learn the probability of existence of a link between two pages given their labels. Note that this model does not require inference during learning. Maximum likelihood estimation (with regularization) of the generative component is closed form given appropriate co-occurrence counts of linked pages' labels. However, the prediction phase is much more expensive, since the resulting graphical model includes edges not only for the existing hyperlinks, but also those that do not exist. Intuitively, observing the link structure directly correlates all page labels in a website, linked or not. By contrast, the Link model avoids this problem by only modeling the conditional distribution given the existing links.

Fig. 19 shows a gain in accuracy from SVMs to the Link model by using the correlations between labels of linked web pages. There is also very significant additional gain by using maximum margin training: the error rate of Link-MM is 40% lower than that of Link-ML, and 51% lower than multi-class SVMs. The Exists model doesn't perform very well in comparison. This can be attributed to the simplicity of the generative model and the difficulty of the resulting inference problem.

11.2.3 Cocite model

The second structured model uses the insight that a webpage often has internal structure that allows it to be broken up into *sections*. For example, a faculty webpage might have one section that discusses research, with a list of links to all of the projects of the faculty member, a second section might contain links to the courses taught by the faculty member, and a third to his advisees. We can view a section of a webpage as a fine-grained version of

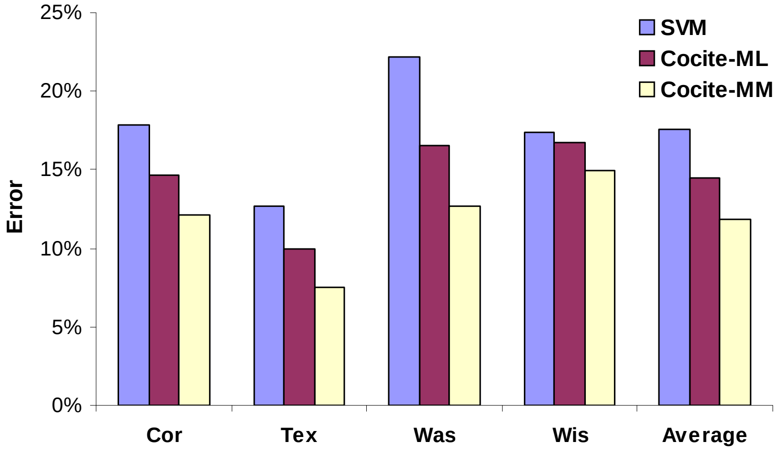


Figure 20: Comparison of Naive Bayes, Svm, and Logistic on WebKB, with and without meta-data features. (Only averages over the 4 schools are shown here.)

Kleinberg's hub (Kleinberg, 1999) (a page that contains a lot of links to pages of particular category). Intuitively, if two pages are *cocited*, or linked to from the same section, they are likely to be on similar topics. Note that we expect the correlation between the labels in this case to be positive, so we can use AMN-type potentials in the max-margin estimation. The Cocite model captures this type of correlation. We defined a section as a sequence of three or more links that have the same path to the root in the html parse tree. We then connected the all the label variables of the pages in the section using pairwise cliques.

We compared the performance of SVM, Cocite-ML and Cocite-MM. The results, shown in Fig. 20, also demonstrate significant improvements of the Markov network models over the SVM. The improvement is present when testing on each of the schools. Again, maximum likelihood trained model Cocite-ML achieves a worse test error than maximum margin Cocite-MM model, which shows a 30% relative reduction in test error over SVM. We note that, in our experiments, the learned Cocite-MM weights never produced fractional solutions when used for inference, which suggests that the optimization successfully avoided problematic parameterizations of the network, even in the case of the non-optimal multi-class relaxation.

11.3 Terrain segmentation

We also applied associative Markov networks to the task of terrain classification, which is very useful in real-world environments for path planning, target detection, and as a preprocessing step for other perceptual tasks. The Stanford Segbot Project[3] has provided us with a laser range maps of the Stanford campus collected by a moving robot equipped with SICK2 laser sensors (Fig. 21). The data consists of around 35 million points, represented

---

[3] Many thanks to Michael Montemerlo and Sebastian Thrun for sharing the data.

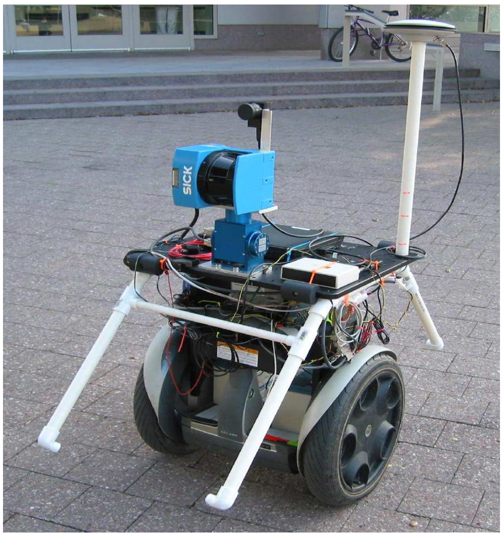


Figure 21: Segbot: roving robot equipped with SICK2 laser sensors.

as 3D coordinates in an absolute frame of reference (Fig. 22). Thus, the only available information is the location of points. Each reading was a point in 3D space, represented by its (x, y, z) coordinates in an absolute frame of reference. Thus, the only available information is the location of points, which was fairly noisy because of localization errors.

Our task is to classify the laser range points into four classes: ground, building, tree, and shrubbery. Since classifying ground points is trivial given their absolute z-coordinate (height), we classify them deterministically by thresholding the z coordinate at a value close to 0. After we do that, we are left with approximately 20 million non-ground points. Each point is represented simply as a location in an absolute 3D coordinate system. The features we use require pre-processing to infer properties of the local neighborhood of a point, such as how planar the neighborhood is, or how much of the neighbors are close to the ground. The features we use are invariant to rotation in the x-y plane, as well as the density of the range scan, since scans tend to be sparser in regions farther from the robot.

Our first type of feature is based on the principal plane around it. For each point we sample 100 points in a cube of radius 0.5 meters. We run PCA on these points to get the plane of maximum variance (spanned by the first two principal components). We then partition the cube into $3 \times 3 \times 3$ bins around the point, oriented with respect to the principal plane, and compute the percentage of points lying in the various sub-cubes. We use a number of features derived from the cube such as the percentage of points in the central column, the outside corners, the central plane, etc. These features capture the local distribution well and are especially useful in finding planes. Our second type of feature is based on a column around each point. We take a cylinder of radius 0.25 meters, which extends vertically to include all the points in a "column". We then compute what percentage of the points lie in various segments of this vertical column (e.g., between 2m and 2.5m). Finally, we also use an indicator feature of whether or not a point lies within $2m$ of the ground. This feature is especially useful in classifying shrubbery.

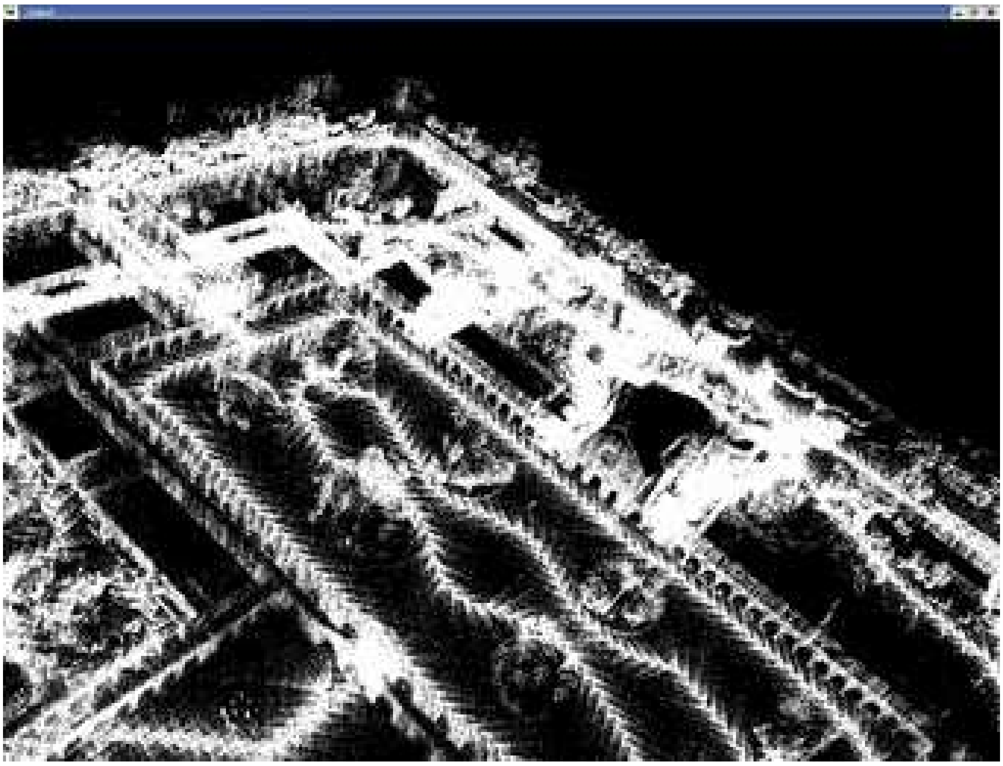


Figure 22: 3D laser scan range map of the Stanford Quad.

For training we select roughly 30 thousand points that represent the classes well: a segment of a wall, a tree, some bushes. We considered three different models: SVM, Voted-SVM and AMNs. All methods use the same set of features, augmented with a quadratic kernel.

The first model is a multi-class SVM with a quadratic kernel over the above features. This model (Fig. 23, right panel and Fig. 25, top panel) achieves reasonable performance in many places, but fails to enforce local consistency of the classification predictions. For example arches on buildings and other less planar regions are consistently confused for trees, even though they are surrounded entirely by buildings.

We improved upon the SVM by smoothing its predictions using voting. For each point we took its local neighborhood (we varied the radius to get the best possible results) and assigned the point the label of the majority of its 100 neighbors. The Voted-SVM model (Fig. 23, middle panel and Fig. 25, middle panel) performs slightly better than SVM: for example, it smooths out trees and some parts of the buildings. Yet it still fails in areas like arches of buildings where the SVM classifier has a locally consistent wrong prediction.

The final model is a pairwise AMN over laser scan points, with associative potentials to ensure smoothness. Each point is connected to 6 of its neighbors: 3 of them are sampled randomly from the local neighborhood in a sphere of radius 0.5m, and the other 3 are sampled at random from the vertical cylinder column of radius 0.25m. It is important to ensure vertical consistency since the SVM classifier is wrong in areas that are higher off the

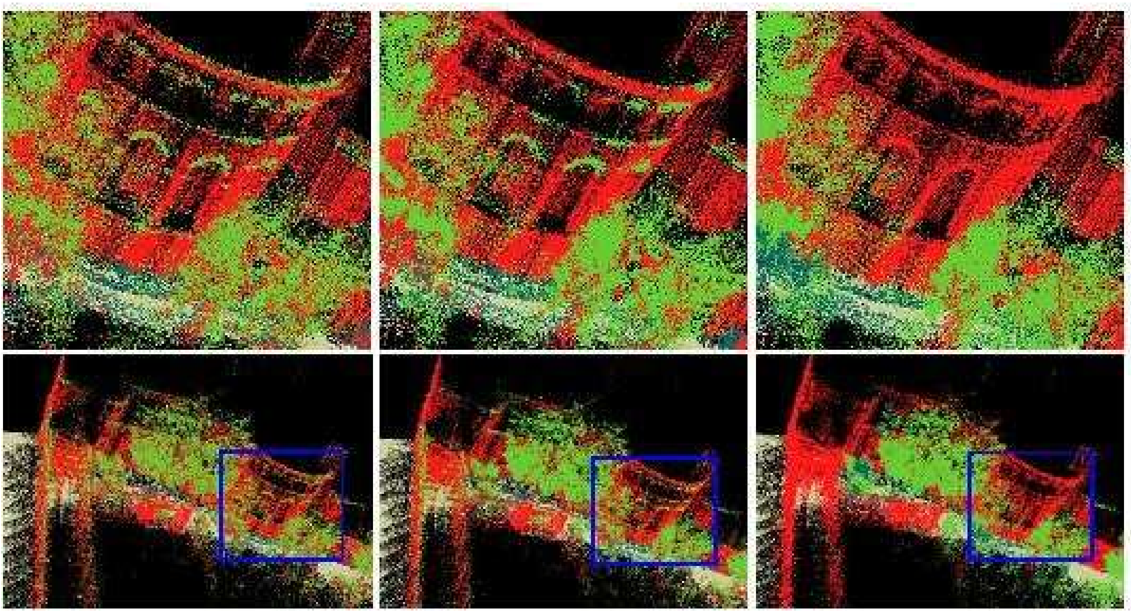


Figure 23: Terrain classification results showing Stanford Memorial Church obtained with SVM, Voted-SVM and AMN models. (Color legend: buildings/red, trees/green, shrubs/blue, ground/gray).

ground (due to the decrease in point density) or because objects tend to look different as we vary their z-coordinate (for example, tree trunks and tree crowns look different). While we experimented with a variety of edge features including various distances between points, we found that even using only a constant feature performs well.

We trained the AMN model using CPLEX to solve the quadratic program; the training took about an hour on a Pentium 3 desktop. The inference over each segment was performed using min-cut with α-expansion moves as described above. We used a publicly available implementation of the min-cut algorithm, which uses bidirectional search trees for augmenting paths (see Boykov and Kolmogorov (2004)). The implementation is largely dominated by I/O time, with the actual min-cut taking less than two minutes even for the largest segment. The performance is summarized in Fig. 24, and as we can see, it is roughly linear in the size of the problem (number of nodes and number of edges).

We can see that the predictions of the AMN (Fig. 23, left panel and Fig. 25, bottom panel) are much smoother: for example building arches and tree trunks are predicted correctly. We also hand-labeled around 180 thousand points of the test set (Fig. 26) and computed accuracies of the predictions shown in Fig. 27 (excluding ground, which was classified by pre-processing). The differences are dramatic: SVM: 68%, Voted-SVM: 73% and AMN: 93%. See more results, including a fly-through movie of the data, at `http://ai.stanford.edu/~btaskar/3Dmap/`.

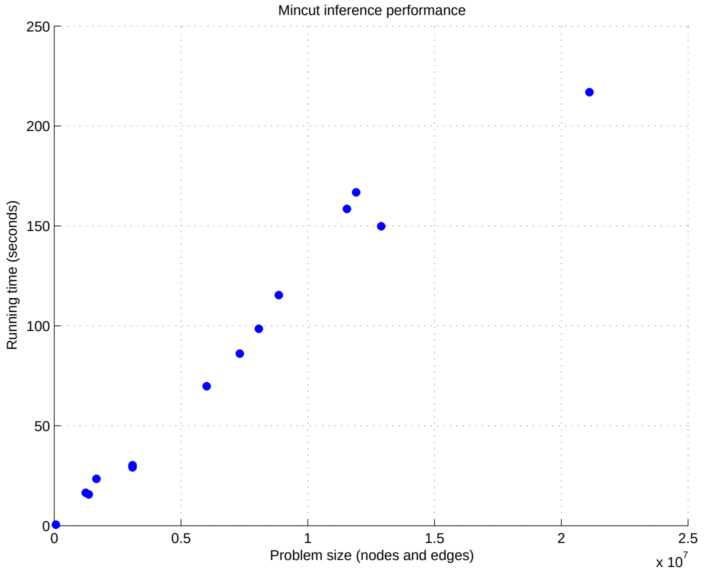


Figure 24: The running time (in seconds) of the min-cut-based inference algorithm for different problem sizes. The problem size is the sum of the number of nodes and the number of edges. Note the near linear performance of the algorithm and its efficiency even for large models.

12. Related Work

There is a large body of material related to the work we have presented. We organize it below in several subsections.

12.1 Structured maximum margin estimation

Our max-margin formulation is related to a body of work called inverse combinatorial and convex optimization (Burton and Toint, 1992; Zhang and Ma, 1996; Ahuja and Orlin, 2001; Heuberger, 2004). An *inverse optimization problem* is defined by an instance of an optimization problem $\max_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^\top \mathbf{f}(\mathbf{y})$, a set of nominal weights $\mathbf{w}^0$, and a target solution $\mathbf{y}^t$. The goal is to find the weights $\mathbf{w}$ closest to the nominal $\mathbf{w}^0$ in some norm, which make the target solution optimal:

$$\begin{aligned} \min \quad & ||\mathbf{w} - \mathbf{w}^0||_p \\ \text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}(\mathbf{y}^t) \geq \mathbf{w}^\top \mathbf{f}(\mathbf{y}), \quad \forall \mathbf{y} \in \mathcal{Y}. \end{aligned}$$

Most of the attention has been on L_1 and L_∞ norms, but L_2 norm is also used.

The study of inverse problems began with geophysical scientists (see (Tarantola, 1987) for in-depth discussion of a wide range of applications). Modeling a complex physical system often involves a large number of parameters which scientists find hard or impossible to set correctly. Provided educated guesses for the parameters $\mathbf{w}^0$ and the behavior of the system as a target, the inverse optimization problem attempts to match the behavior while not perturbing the "guesstimate" too much.

Although there is a strong connection between inverse optimization problems and our formulations, the goals are very different than ours. In our framework, we are learning a parameterized objective function that depends on the input $\mathbf{x}$ and will generalize well in prediction on new instances. Moreover, we do not assume as given a nominal set of weights. Note that if we set $\mathbf{w}^0 = 0$, then $\mathbf{w} = 0$ is trivially the optimal solution. The solution $\mathbf{w}$ depends critically on the choice of nominal weights, which is not appropriate in the learning setting.

The inverse reinforcement learning problem (Ng and Russell, 2000; Abbeel and Ng, 2004) is much closer to our setting. The goal is to learn a reward function that will cause a rational agent to act similar to the observed behavior of an expert. A full description of the problem is beyond our scope, but we briefly describe the Markov decision process (MDP) model commonly used for sequential decision making problems where an agent interacts with its environment. The environment is modeled as a system that can be in one of a set of discrete states. At every time step, the agent chooses an action from a discrete set of actions and the system transitions to a next state with a probability that depends on the current state and the action taken. The agent collects a reward at each step, which generally depends on the on the current and the next state and the action taken. A rational agent executes a policy (essentially, a state to action mapping) that maximizes its expected reward. To map this problem (approximately) to our setting, note that a policy roughly corresponds to the labels $\mathbf{y}$, the state sequence correspond to the input $\mathbf{x}$ and the reward for a state/action sequence is assumed to be $\mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y})$ for some basis functions $\mathbf{w}^\top\mathbf{f}(\mathbf{x},\mathbf{y})$. The goal is to learn $\mathbf{w}$ from a set of state/action sequences $(\mathbf{x}^{(i)},\mathbf{y}^{(i)})$ of the expert such that the maximizing the expected reward according to the system model makes the agent imitate the expert. This and related problems are formulated as a convex program in Ng and Russell (2000) and Abbeel and Ng (2004).

12.2 M3Ns

The application of margin-based estimation methods to parsing and sequence modeling was pioneered by Collins (2001) using the Voted-Perceptron algorithm (Freund and Schapire, 1998). He provides generalization guarantees (for 0/1 loss) that hold for separable case and depend on the number of mistakes the perceptron makes before convergence. Remarkably, the bound does not explicitly depend on the length of the sequence, although undoubtedly the number of mistakes does.

Collins (2004) also suggested an SVM-like formulation (with exponentially many constraints) and a constraint generation method for solving it. His generalization bound (for 0/1 loss) based on the SVM-like margin, however, has linear dependence (inside the square root) on the number of nodes (L). It also depends on the joint 2-norm of all of the basis functions for an example (which is $\sim N_c R_c$). By considering the more natural Hamming loss, we achieve a much tighter analysis.

Altun et al. (2003) have applied the exponential-size formulation with constraint generation we described in Sec. 6.1.1 to problems natural language processing. In a follow-up paper, Tsochantaridis et al. (2004) show that only a polynomial number of constraints are needed to be generated to guarantee a fixed level of precision of the solution. However, the number of constraints in many important cases is several orders higher (in L) than in the the approach we present. In addition, the corresponding problem needs to be resolved (or at least approximately resolved) after each additional constraint is added, which is prohibitively expensive for large number of examples and label variables.

The work of Guestrin et al. (2003) presents LP decompositions based on graphical model structure for the value function approximation problem in factored MDPs (Markov decision processes with structure). Describing the exact setting is beyond our scope, but it suffices to say that our original decomposition of the max-margin QP was inspired by the proposed technique to transform an exponential set of constraints into a polynomial one using a triangulated graph.

There has been a recent explosion of work in maximum conditional likelihood estimation of Markov networks. The work of Lafferty et al. (2001) has inspired many applications in natural language, computational biology, computer vision and relational modeling (Sha and Pereira, 2003; Pinto et al., 2003; Kumar and Hebert, 2003; Sutton et al., 2004; Taskar et al., 2002, 2003). As in the case of logistic regression, maximum conditional likelihood estimation for Markov networks can also be kernelized (Altun et al., 2004; Lafferty et al., 2004). However, the solutions are non-sparse and the proposed algorithms are forced to use greedy selection of support vectors or heuristic pruning methods.

12.3 AMNs

Several authors have considered extensions to the Potts model. Kleinberg and Tardos (1999) extend the multi-class Potts model to have more general edge potentials, under the constraints that negative log of the edge potentials form a metric on the set of labels. They also provide a solution based on a relaxed LP that has certain approximation guarantees.

More recently, Kolmogorov and Zabih (2002) showed how to optimize energy functions containing binary and ternary interactions using graph cuts, as long as the parameters satisfy a certain regularity condition. Our definition of associative potentials below also satisfies the Kolmogorov and Zabih regularity condition for $K = 2$. However, the structure of our potentials is simpler to describe and extend for the multi-class case. In fact, we can extend our max-margin framework to estimate their more general potentials by expressing inference as a linear program.

Our terrain classification approach is most closely related to work in vision applying conditional random fields (CRFs) to 2D images. Kumar and Hebert (2003) train CRFs using a pseudo-likelihood approximation to the distribution $P(\mathbf{Y} \mid \mathbf{X})$ since estimating the true conditional distribution is intractable. Unlike their work, our learning formulation provides an exact and tractable optimization algorithm, as well as formal guarantees for binary classification problems. Moreover, unlike their work, our approach can also handle multi-class problems in a straightforward manner.

12.4 Structured SMO

The kernel-adatron (Friess et al., 1998) and voted-perceptron algorithms (Freund and Schapire, 1998) for large-margin classifiers have a similar online optimization scheme. Collins (2001) have applied voted-perceptron to structured problems in natural language. Although head-to-head comparisons have not been performed, it seems that, empirically, less passes (about 30-40) are needed for our algorithm than in the perceptron literature.

Recently, the Exponentiated Gradient (Kivinen and Warmuth, 1997) algorithm has been adopted to solve our structured QP for max-margin estimation (Bartlett et al., 2004). Although the EG algorithm has attractive convergence properties, it has yet to be shown to learn faster than Structured SMO, particularly in the early iterations through the dataset.

13. Discussion

We present a discriminative framework for labeling and segmentation of structured data such as sequences, images, etc. Our approach seamlessly integrates state-of-the-art kernel methods developed for classification of independent instances with the rich language of graphical models that can exploit the structure of complex data. In our experiments with handwriting recognition, for example, our sequence model significantly outperforms other approaches by incorporating high-dimensional decision boundaries of polynomial kernels over character images while capturing correlations between consecutive characters.

Our formulation for max-margin estimation of Markov networks uses a compact convex optimization problem. We exploit graph decomposition to derive an exact, compact, convex max-margin formulation for Markov networks with sequence and other low-treewidth structure. The formulation avoids the exponential blow-up in the number of constraints in the max-margin QP that plagued previous approaches. We also use approximate graph decomposition to derive a compact approximate formulation for Markov networks in which inference is intractable. We show that our approximation performs well in our experiments on structured hypertext classification.

Although the number of variables and constraints of our QP formulation is polynomial in the example size (e.g., sequence length), we also address its quadratic growth using an effective optimization procedure inspired by SMO. This simple procedure uses inference in the model and analytic updates to solve very large instances without the help of QP solver software.

We provide an algorithm for max-margin training of associative Markov networks, a subclass of Markov networks that allows only positive interactions between related variables, but makes no restriction on the topology (e.g., low-treewidth). Because our approach only relies using the MAP in the model for prediction, and does not require a normalized distribution $P(\mathbf{y} \mid \mathbf{x})$ over all outputs, maximum margin estimation is tractable, although maximum likelihood is not. Our method is guaranteed to find the optimal (margin-maximizing) solution for any binary-valued AMN, regardless of the clique size or the connectivity. To our knowledge, this algorithm is the first to provide an effective learning procedure for Markov networks of such general structure. In the non-binary case, we are not guaranteed exact solutions, but show that our approximation works well in experiments with hypertext classification and 3D image segmentation. We present large-scale experiments with terrain segmentation and classification from 3D range data involving AMNs with tens of millions of nodes and edges.

We provide theoretical guarantees on the average *per-label* generalization error of our models in terms of the training set margin. Our generalization bound significantly tightens previous results of Collins (2001) and suggests possibilities for analyzing per-label generalization properties of graphical models.

Our experiments on the tasks of handwriting recognition, web page classification and 3D terrain segmentation demonstrate very significant gains over previous approaches. Overall, we believe that M^3 networks will significantly further the applicability of high accuracy margin-based methods to real-world structured data.

Acknowledgments

This work was supported by ONR Contract F3060-01-2-0564-P00002 under DARPA's EELD program.

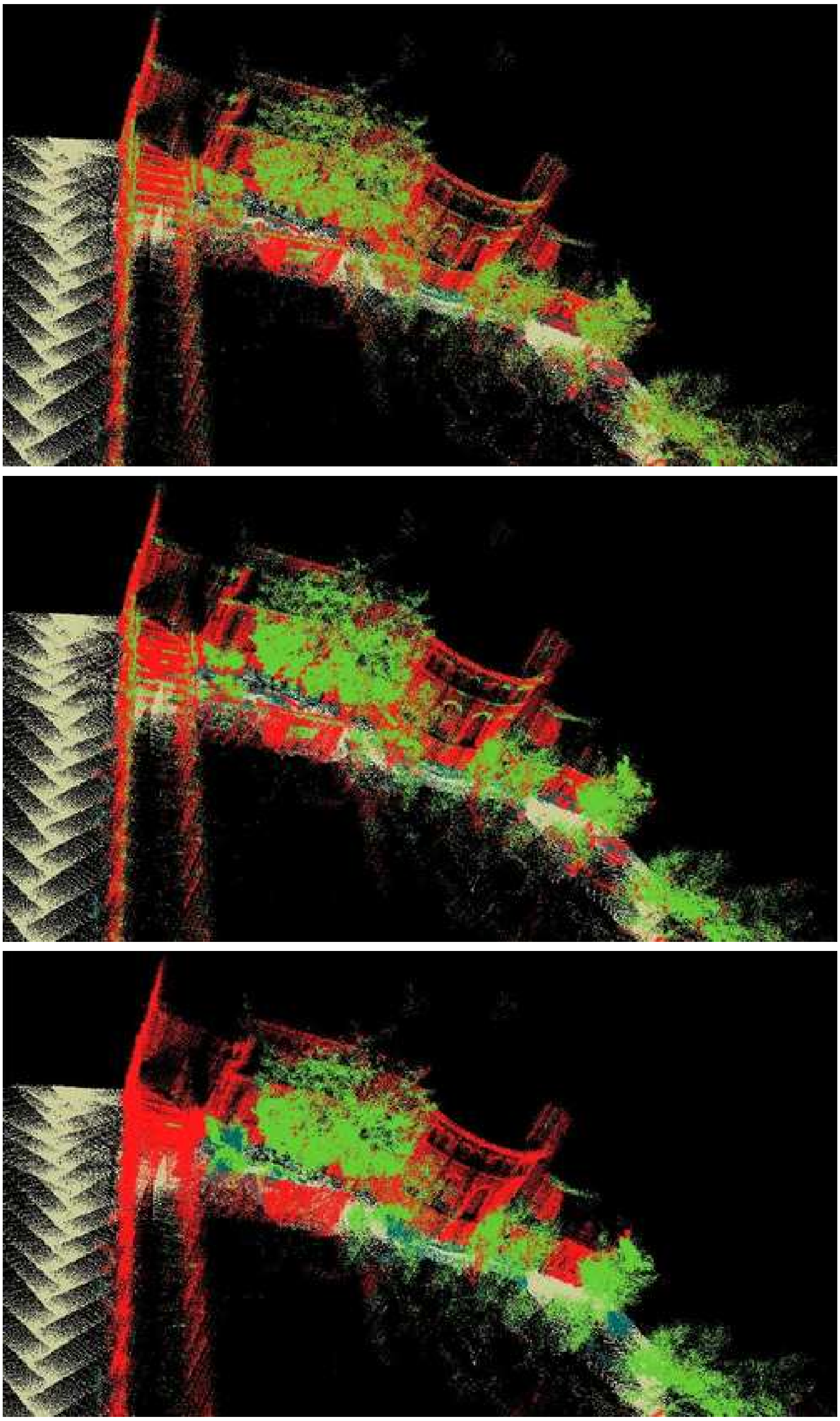


Figure 25: Results from the SVM, Voted-SVM and AMN models.

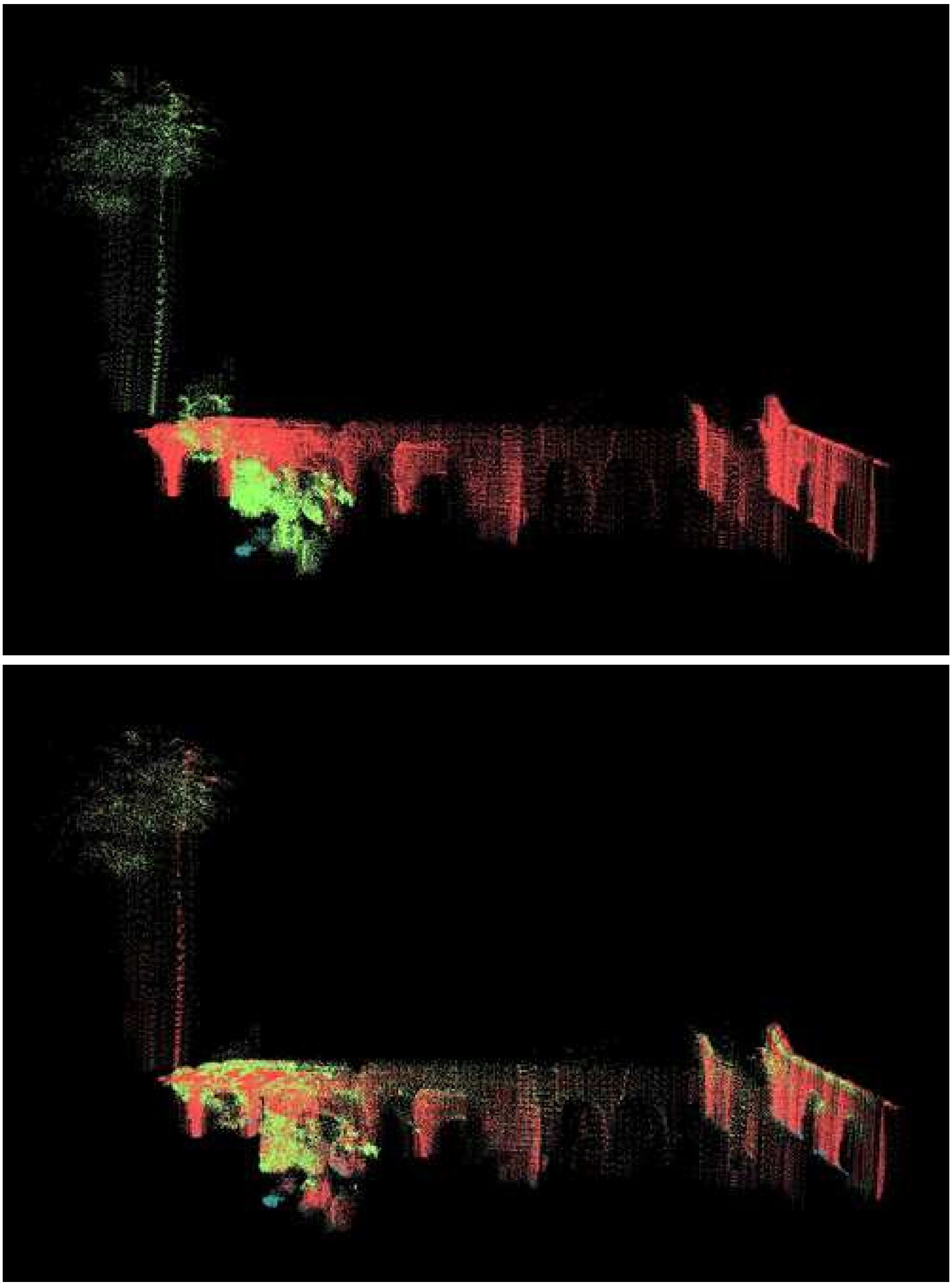


Figure 26: Labeled part of the test set: ground truth (top) and SVM predictions (bottom).

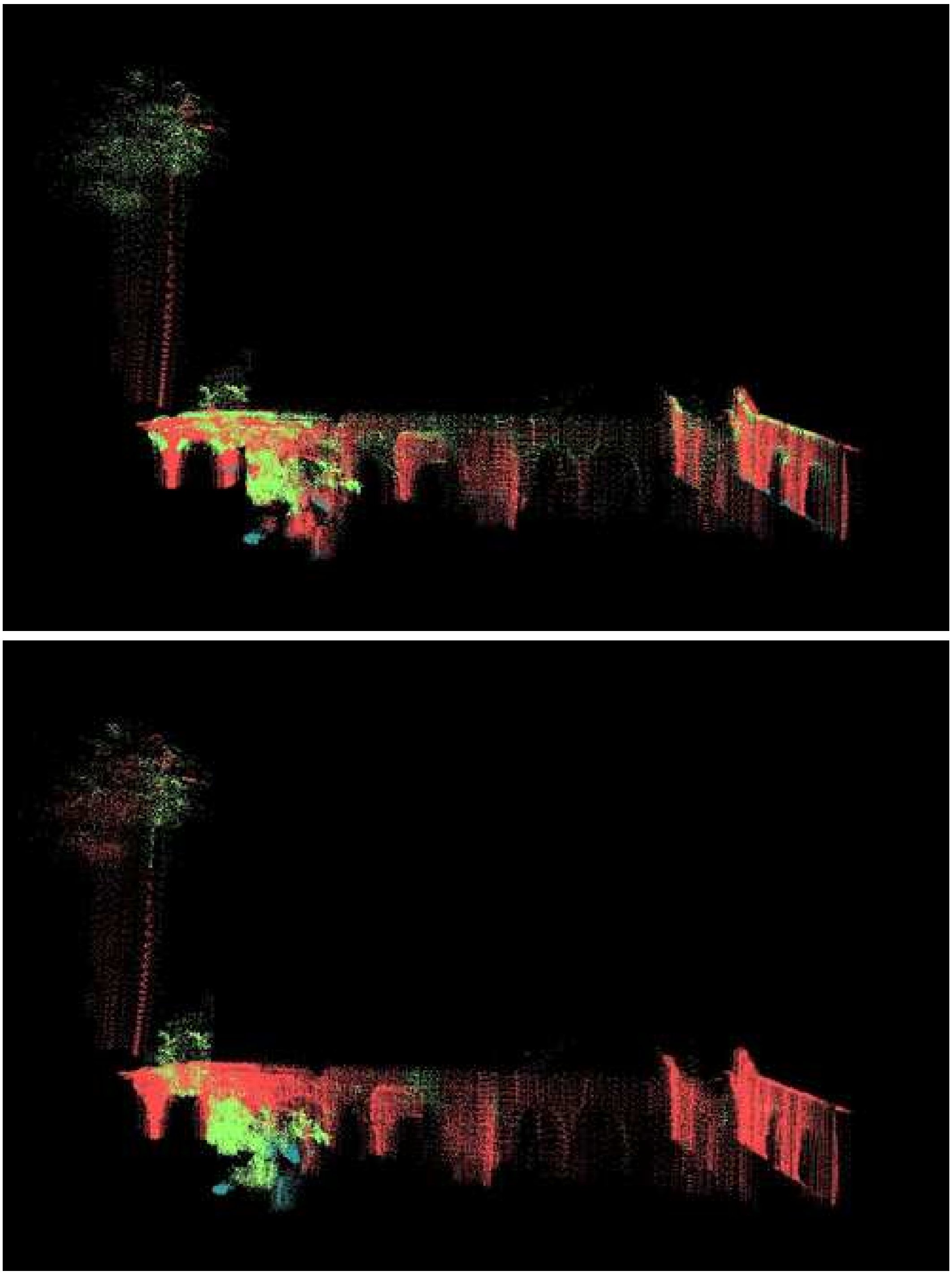


Figure 27: Predictions of the Voted-SVM (top) and AMN (bottom) models.

Appendix A. AMN proofs and derivations

In this appendix, we present proofs of the LP inference properties and derivations of the factored primal and dual max-margin formulation from Ch. 8. Recall that the LP relaxation for finding the optimal $\max_{\mathbf{y}} g(\mathbf{y})$ is:

$$\begin{aligned}
\max \quad & \sum_{v \in \mathcal{V}} \sum_{k=1}^{K} \mu_v(k) g_v(k) + \sum_{c \in \mathcal{C} \setminus \mathcal{V}} \sum_{k=1}^{K} \mu_c(k) g_c(k) && (41)\\
\text{s.t.} \quad & \mu_c(k) \geq 0, \quad \forall c \in \mathcal{C},\ k; \quad \sum_{k=1}^{K} \mu_v(k) = 1, \quad \forall v \in \mathcal{V};\\
& \mu_c(k) \leq \mu_v(k), \quad \forall c \in \mathcal{C} \setminus \mathcal{V},\ v \in c,\ k.
\end{aligned}$$

A.1 Binary AMNs

Proof (For Theorem 5.2) Consider any fractional, feasible μ. We show that we can construct a new feasible assignment μ' which increases the objective (or leaves it unchanged) and furthermore has fewer fractional entries.

Since $g_c(k) \geq 0$, we can assume that $\mu_c(k) = \min_{v \in c} \mu_v(k)$; otherwise we could increase the objective by increasing $\mu_c(k)$. We construct an assignment μ' from μ by leaving integral values unchanged and uniformly shifting fractional values by λ:

$$\begin{aligned}
\mu'_v(1) &= \mu_v(1) - \lambda \mathbb{1}(0 < \mu_v(1) < 1), & \mu'_v(2) &= \mu_v(2) + \lambda \mathbb{1}(0 < \mu_v(2) < 1),\\
\mu'_c(1) &= \mu_c(1) - \lambda \mathbb{1}(0 < \mu_c(1) < 1), & \mu'_c(2) &= \mu_c(2) + \lambda \mathbb{1}(0 < \mu_c(2) < 1).
\end{aligned}$$

Now consider the smallest fractional $\mu_v(k)$, $\lambda(k) = \min_{v\,:\,\mu_v(k)>0} y_v(k)$ for $k = 1, 2$. Note that if $\lambda = \lambda(1)$ or $\lambda = -\lambda(2)$, μ' will have at least one more integral $\mu'_v(k)$ than μ. Thus if we can show that the update results in a feasible and better scoring assignment, we can apply it repeatedly to get an optimal integer solution. To show that μ' is feasible, we need $\mu'_v(1) + \mu'_v(2) = 1$, $\mu'_v(k) \geq 0$ and $\mu'_c(k) = \min_{i \in c} \mu'_v(k)$.

First, we show that $\mu'_v(1) + \mu'_v(2) = 1$.

$$\begin{aligned}
\mu'_v(1) + \mu'_v(2) &= \mu_v(1) - \lambda \mathbb{1}(0 < \mu_v(1) < 1) + \mu_v(2) + \lambda \mathbb{1}(0 < \mu_v(2) < 1)\\
&= \mu_v(1) + \mu_v(2) = 1.
\end{aligned}$$

Above we used the fact that if $\mu_v(1)$ is fractional, so is $\mu_v(2)$, since $\mu_v(1) + \mu_v(2) = 1$.

To show that $\mu'_v(k) \geq 0$, we prove $\min_v \mu'_v(k) = 0$.

$$\begin{aligned}
\min_v \mu'_v(k) &= \min_v \left[\mu_v(k) - \big(\min_{i:\mu_v(k)>0} \mu_v(k)\big) \mathbb{1}(0 < \mu_v(k) < 1) \right]\\
&= \min \left(\min_i \mu_v(k), \min_{i:\mu_v(k)>0} \left[\mu_v(k) - \min_{i:\mu_v(k)>0} \mu_v(k) \right] \right) = 0.
\end{aligned}$$

Lastly, we show $\mu'_c(k) = \min_{i \in c} \mu'_v(k)$.

$$\mu'_c(1) \quad = \quad \mu_c(1) - \lambda \mathbb{1}(0 < \mu_c(1) < 1)$$

$$
\begin{aligned}
&= (\min_{i\in c} \mu_v(1)) - \lambda 1\!\!1(0 < \min_{i\in c} \mu_v(1) < 1) = \min_{i\in c} \mu'_v(1);\\
\mu'_c(2) &= \mu_c(2) + \lambda 1\!\!1(0 < \mu_c(1) < 1)\\
&= (\min_{i\in c} \mu_v(2)) + \lambda 1\!\!1(0 < \min_{i\in c} \mu_v(2) < 1) = \min_{i\in c} \mu'_v(2).
\end{aligned}
$$

We have established that the new μ' are feasible, and it remains to show that we can improve the objective. We can show that the change in the objective is always λD for some constant D that depends only on μ and g. This implies that one of the two cases, $\lambda = \lambda(1)$ or $\lambda = -\lambda(2)$, will necessarily increase the objective (or leave it unchanged). The change in the objective is:

$$
\begin{aligned}
&\sum_{v\in\mathcal{V}} \sum_{k=1,2} [\mu'_v(k) - \mu_v(k)] g_v(k) + \sum_{c\in\mathcal{C}\setminus\mathcal{V}} \sum_{k=1,2} [\mu'_c(k) - \mu_c(k)] g_c(k)\\
&= \lambda \left[\sum_{v\in\mathcal{V}} [D_v(1) - D_v(2)] + \sum_{c\in\mathcal{C}\setminus\mathcal{V}} [D_c(1) - D_c(2)] \right] = \lambda D\\
&D_v(k) \;=\; g_v(k) 1\!\!1(0 < \mu_v(k) < 1), \quad D_c(k) \;=\; g_c(k) 1\!\!1(0 < \mu_c(k) < 1).
\end{aligned}
$$

Hence the new assignment μ' is feasible, does not decrease the objective function, and has strictly fewer fractional entries. ▮

A.2 Multi-class AMNs

For $K > 2$, we use the randomized rounding procedure of Kleinberg and Tardos (1999) to produce an integer solution for the linear relaxation, losing at most a factor of $m = \max_{c\in\mathcal{C}} |c|$ in the objective function. The basic idea of the rounding procedure is to treat $\mu_v(k)$ as probabilities and assign labels according to these probabilities in phases. In each phase, we pick a label k, uniformly at random, and a threshold $\alpha \in [0,1]$ uniformly at random. For each node i which has not yet been assigned a label, we assign the label k if $\mu_v(k) \geq \alpha$. The procedure terminates when all nodes have been assigned a label. Our analysis closely follows that of Kleinberg and Tardos (1999).

Lemma A.1 *The probability that a node i is assigned label k by the randomized procedure is $\mu_v(k)$.*

Proof The probability that an unassigned node is assigned label k during one phase is $\frac{1}{K}\mu_v(k)$, which is proportional to $\mu_v(k)$. By symmetry, the probability that a node is assigned label k over all phases is exactly $\mu_v(k)$. ▮

Lemma A.2 *The probability that all nodes in a clique c are assigned label k by the procedure is at least $\frac{1}{|c|}\mu_c(k)$.*

Proof For a single phase, the probability that all nodes in a clique c are assigned label k if none of the nodes were previously assigned is $\frac{1}{K}\min_{i\in c}\mu_v(k) = \frac{1}{K}\mu_c(k)$. The probability that *at least one* of the nodes will be assigned label k in a phase is $\frac{1}{K}(\max_{i\in c}\mu_v(k))$. The probability that *none* of the nodes in the clique will be assigned *any* label in one phase is $1 - \frac{1}{K}\sum_{k=1}^{K}\max_{i\in c}\mu_v(k)$.

Nodes in the clique c will be assigned label k by the procedure if they are assigned label k in one phase. (They can also be assigned label k as a result of several phases, but we can ignore this possibility for the purposes of the lower bound.) The probability that all the nodes in c will be assigned label k by the procedure in a single phase is:

$$\begin{aligned}\sum_{j=1}^{\infty}\frac{1}{K}\mu_c(k)\left(1-\frac{1}{K}\sum_{k=1}^{K}\max_{i\in c}\mu_v(k)\right)^{j-1} &= \frac{\mu_c(k)}{\sum_{k=1}^{K}\max_{i\in c}\mu_v(k)}\\ \geq \frac{\mu_c(k)}{\sum_{k=1}^{K}\sum_{i\in c}\mu_v(k)} = \frac{\mu_c(k)}{\sum_{i\in c}\sum_{k=1}^{K}\mu_v(k)} &= \frac{\mu_c(k)}{|c|}.\end{aligned}$$

Above, we first used the fact that for $d < 1$, $\sum_{i=0}^{\infty} d^i = \frac{1}{1-d}$, and then upper-bounded the max of the set of positive $\mu_v(k)$'s by their sum. ∎

Theorem A.3 *The expected cost of the assignment found by the randomized procedure given a solution* μ *to the linear program in Eq. (41) is at least* $\sum_{v\in\mathcal{V}}\sum_{k=1}^{K} g_v(k)\mu_v(k) + \sum_{c\in\mathcal{C}\setminus\mathcal{V}}\frac{1}{|c|}\sum_{k=1}^{K} g_c(k)\mu_c^k$.

Proof This is immediate from the previous two lemmas.

The only difference between the expected cost of the rounded solution and the (non-integer) optimal solution is the $\frac{1}{|c|}$ factor in the second term. By picking $m = \max_{c\in\mathcal{C}}|c|$, we have that the rounded solution is at most m times worse than the optimal solution produced by the LP of Eq. (41). ∎

We can also derandomize this procedure to get a deterministic algorithm with the same guarantees, using the method of conditional probabilities, similar in spirit to the approach of Kleinberg and Tardos (1999).

Note that the approximation factor of m applies, in fact, only to the clique potentials. Thus, if we compare the log-probability of the optimal MAP solution and the log-probability of the assignment produced by this randomized rounding procedure, the terms corresponding to the log-partition-function and the node potentials are identical. We obtain an additive error (in log-probability space) only for the clique potentials. As node potentials are often larger in magnitude than clique potentials, the fact that we incur no loss proportional to node potentials is likely to lead to smaller errors in practice. Along similar lines, we note that the constant factor approximation is smaller for smaller cliques; again, we observe, the potentials associated with large cliques are typically smaller in magnitude, reducing further the actual error in practice.

A.3 Min-cut inference

We can also use efficient min-cut algorithms to perform exact inference on the learned models for $K = 2$ and approximate inference for $K > 2$. For simplicity, we focus on the pairwise AMN case. We first consider the case of binary AMNs, and later show how to use

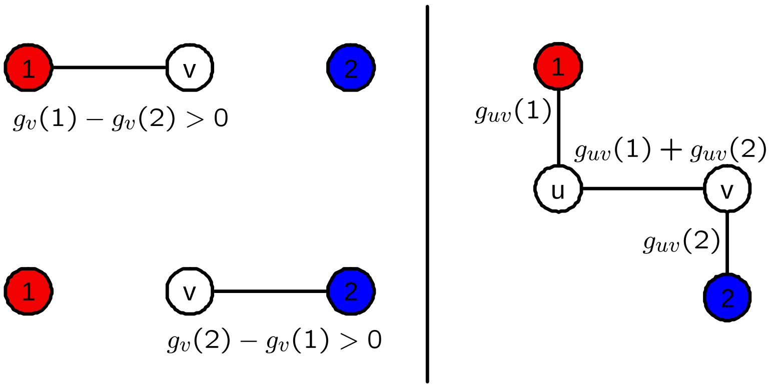


Figure 28: Min-cut graph construction of node (left) and edge (right) terms.

the local search algorithm developed by Boykov et al. (1999a) to perform (approximate) inference in the general multi-class case. For pairwise, binary AMNs, the objective of the integer program in Eq. (13) is:

$$\max \sum_{v \in \mathcal{V}} [\mu_v(1) g_v(1) + \mu_v(2) g_v(2)] + \sum_{uv \in \mathcal{E}} [\mu_{uv}(1) g_{uv}(1) + \mu_{uv}(2) g_{uv}(2)]. \tag{42}$$

Graph construction

We construct a graph in which the *min-cut* will correspond to the optimal MAP labeling for the above objective. First, we recast the objective as minimization by simply reversing the signs on the value of each θ.

$$\min \; -\sum_{v \in \mathcal{V}} [\mu_v(1) g_v(1) + \mu_v(2) g_v(2)] - \sum_{uv \in \mathcal{E}} [\mu_{uv}(1) g_{uv}(1) + \mu_{uv}(2) g_{uv}(2)]. \tag{43}$$

The graph will consist of a vertex for each node in the AMN, along with the 1 and 2 terminals. In the final $(\mathcal{V}_1, \mathcal{V}_2)$ cut, the $\mathcal{V}_1$ set will correspond to label 1, and the $\mathcal{V}_2$ set will correspond to label 2. We will show how to deal with the node terms (those depending only on a single variable) and the edge terms (those depending on a pair of variables), and then how to combine the two.

Node terms

Consider a node term $-\mu_v(1) g_v(1) - \mu_v(2) g_v(2)$. Such a term corresponds to the node potential contribution to our objective function for node v. For each node term corresponding to node v we add a vertex v to the min-cut graph. We then look at $\Delta_v = g_v(1) - g_v(2)$, and create an edge of weight $|\Delta_v|$ from v to either 1 or 2, depending on the sign of Δ_v. The reason for that is that the final min-cut graph must consist of only positive weights. An example is presented in Fig. A.3.

From Fig. A.3, we see that if the AMN consisted of only node potentials, the graph construction above would add an edge from each node to its more likely label. Thus if we run min-cut, we would simply get a cut with cost 0, since for each introduced vertex we have only one edge of positive weight to either 1 or 2, and we would always choose not to cut any edges.

Edge terms

Now consider an edge term of the form $-\mu_{uv}(1)g_{uv}(1) - \mu_{uv}(2)g_{uv}(2)$. To construct a min-cut graph for the edge term we will introduce two vertices u and v. We will connect vertex u to 1 with an edge of weight $g_{uv}(1)$, connect v to 2 with an edge of weight $g_{uv}(2)$ and connect u to v with an edge of weight $g_{uv}(1) + g_{uv}(2)$. Fig. A.3 shows an example. Observe what happens when both nodes are on the $\mathcal{V}_2$ side of the cut: the value of the min-cut is $g_{uv}(1)$, which must be less than $g_{uv}(2)$ or the min-cut would have placed them both on the 1 side. When looking at edge terms in isolation, a cut that places each node in different sets will not occur, but when we combine the graphs for node terms and edge terms, such cuts will be possible.

We can take the individual graphs we created for node and edge terms and merge them by adding edge weights together (and treating missing edges as edges with weight 0). It can be shown that the resulting graph will represent the same objective (in the sense that running min-cut on it will optimize the same objective) as the sum of the objectives of each graph. Since our MAP-inference objective is simply a sum of node and edge terms, merging the node and edge term graphs will result in a graph in which min-cut will correspond to the MAP labeling.

Multi-class case

The graph construction above finds the best MAP labeling for the binary case, but in practice we would often like to handle multiple classes in AMNs. One of the most effective algorithms for minimizing energy functions like ours is the α-expansion algorithm proposed by Boykov et al. (1999a). The algorithm performs a series of “expansion” moves each of which involves optimization over two labels, and it can be shown that it converges to within a factor of 2 of the global minimum.

Expansion Algorithm

Consider a current labeling μ and a particular label $k \in 1, \ldots, K$. Another labeling μ' is called an “α-expansion” move (following Boykov et al. (1999a)) from μ if $\mu'_v \neq k$ implies $\mu'_v = \mu_v$ (where μ_v is the label of the node v in the AMN.) In other words, a k-expansion from a current labeling allows each label to either stay the same, or change to k.

The α-expansion algorithm cycles through all labels k in either a fixed or random order, and finds the new labeling whose objective has the lowest value. It terminates when there is no α-expansion move for any label k that has a lower objective than the current labeling (Fig. 29).

The key part of the algorithm is computing the best α-expansion labeling for a fixed k and a fixed current labeling μ. The min-cut construction from earlier allows us to do exactly that since an α-expansion move essentially minimizes a MAP-objective over two labels: it either allows a node to retain its current label, or switch to the label α. In this new binary problem we will let label 1 represent a node keeping its current label and label 2 will denote a node taking on the new label k. In order to construct the right coefficients

```
1. Begin with arbitrary labeling $\mu$
2. Set $success := 0$
3. For each label $k \in \{1, \dots K\}$
    3.1 Compute $\hat{\mu} = \arg\min -g(\mu')$ among $\mu'$ within one $\alpha$-expansion of $\mu$.
    3.2 If $E(\hat{\mu}) < E(\mu)$, set $\mu := \hat{\mu}$ and success $:= 1$
4. If $success = 1$ goto 2.
5. Return $\mu$
```

Figure 29: α-expansion algorithm

for the new binary objective we need to consider several factors. Below, let $\theta_i'^k$ and $\theta_{ij}'^{k,k}$ denote the node and edge coefficients associated with the new binary objective:

- **Node potentials** For each node i in the current labeling whose current label is not α, we let $\theta_i'^0 = \theta_i^{y_i}$, and $\theta_i'^1 = \theta_i^{\alpha}$, where y_i denotes the current label of node i, and θ^{y_i} denotes the coefficient in the multiclass AMN MAP objective. Note that we ignore nodes with label α altogether since an α-expansion move cannot change their label.
- **Edge potentials** For each edge $(i, j) \in E$ whose nodes have labels different from α, we add a new edge potential, with weights $\theta_{ij}'^1 = \theta_{ij}^{\alpha,\alpha}$. If the two nodes of the edge currently have the same label, we set $\theta_{ij}'^0 = \theta_{ij}^{y_i,y_j}$, and if the two nodes currently have different labels we let $\theta_{ij}'^0 = 0$. For each edge $(i, j) \in E$ in which exactly one of the nodes has label α in the current labeling, we add $\theta_{ij}^{\alpha,\alpha}$, to the node potential $\theta_i'^1$ of the node whose label is different from α.

After we have constructed the new binary MAP objective as above, we can apply the min-cut construction from before to get the optimal labeling within one α-expansion from the current one. Veksler (1999) shows that the α-expansion algorithm converges in $O(N)$ iterations where N is the number of nodes. As noted in Boykov et al. (1999a) and as we have observed in our experiments, the algorithm terminates only after a few iterations with most of the improvement occurring in the first 2-3 expansion moves.

A.4 Derivation of the factored primal and dual max-margin QP

Using Assumptions 8.1 and 8.4, we have the dual of the LP used to represent the interior max subproblem $\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) + \ell_i(\mathbf{y})$ in Eq. (12):

$$\begin{aligned} \min \quad & \sum_{v \in \mathcal{V}} \xi_{i,v} && (44) \\ \text{s.t.} \quad & -\mathbf{w}^\top \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \end{aligned}$$

$$\begin{aligned}
& -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m_{i,c,v}(k) \geq 0, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k;
\end{aligned}$$

where $\mathbf{f}_{i,c}(k) = \mathbf{f}_{i,c}(k, \ldots, k)$ and $\ell_{i,c}(k) = \ell_{i,c}(k, \ldots, k)$. In the dual, we have a variable $\xi_{i,v}$ for each normalization constraint in Eq. (13) and variables $m_{i,c,v}(k)$ for each of the inequality constraints.

Substituting this dual into Eq. (14), we obtain:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\
\text{s.t.} \quad & \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}^{(i)}) + \xi_i \geq \sum_{v \in \mathcal{V}^{(i)}} \xi_{i,v}, \quad \forall i; \\
& -\mathbf{w}^\top \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m_{i,c,v}(k) \geq 0, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k; \\
& \ddot{\mathbf{w}} \geq 0.
\end{aligned} \tag{45}$$

Now let $\xi_{i,v} = \xi'_{i,v} + \mathbf{w}^\top \mathbf{f}_{i,v}(\mathbf{y}_v^{(i)}) + \sum_{c \supset v} \ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|$ and $m_{i,c,v}(k) = m'_{i,c,v}(k) + \ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|$. Re-expressing the above QP in terms of these new variables, we get:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_i \xi_i \\
\text{s.t.} \quad & \xi_i \geq \sum_{v \in \mathcal{V}^{(i)}} \xi'_{i,v}, \quad \forall i; \\
& \mathbf{w}^\top \Delta \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m'_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi'_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& \ddot{\mathbf{w}}^\top \Delta \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m'_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k; \\
& m'_{i,c,v}(k) \geq -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k; \\
& \ddot{\mathbf{w}} \geq 0.
\end{aligned} \tag{46}$$

Since $\xi_i = \sum_{i,v \in \mathcal{V}^{(i)}} \xi'_{i,v}$ at the optimum, we can eliminate ξ_i and the corresponding set of constraints to get the formulation in Eq. (34), repeated here for reference:

$$\begin{aligned}
\min \quad & \frac{1}{2}||\mathbf{w}||^2 + C \sum_{i,v \in \mathcal{V}^{(i)}} \xi_{i,v} \\
\text{s.t.} \quad & \mathbf{w}^\top \Delta \mathbf{f}_{i,v}(k) - \sum_{c \supset v} m_{i,c,v}(k) \geq \ell_{i,v}(k) - \xi_{i,v}, \quad \forall i, v \in \mathcal{V}^{(i)}, k; \\
& \ddot{\mathbf{w}}^\top \Delta \ddot{\mathbf{f}}_{i,c}(k) + \sum_{v \in c} m_{i,c,v}(k) \geq \ell_{i,c}(k), \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, k;
\end{aligned} \tag{47}$$

$$
\begin{aligned}
& m_{i,c,v}(k) \geq -\ddot{\mathbf{w}}^\top \ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|, \quad \forall i, c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)}, v \in c, k; \\
& \ddot{\mathbf{w}} \geq 0.
\end{aligned}
$$

Now the dual of Eq. (47) is given by:

$$
\begin{aligned}
\max \quad & \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\ell_{i,c}(k) - \frac{1}{2}\left|\left|\sum_{i,v\in\mathcal{V}^{(i)},\,k} \mu_{i,v}(k)\Delta\dot{\mathbf{f}}_{i,v}\right|\right|^2 \qquad (48) \\
& -\frac{1}{2}\left|\left|\ddot{\tau} + \sum_{i,c\in\mathcal{C}^{(i)}\setminus\mathcal{V}^{(i)},v\in c,\,k} \lambda_{i,c,v}(k)\ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c| + \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k)\right|\right|^2 \\
\text{s.t.} \quad & \mu_{i,c}(k) \geq 0, \quad \forall i,\ \forall c \in \mathcal{C}^{(i)},\ k; \qquad \sum_{k=1}^{K} \mu_{i,v}(k) = C, \quad \forall i,\ \forall v \in \mathcal{V}^{(i)}; \\
& \mu_{i,c}(k) - \mu_{i,v}(k) = \lambda_{i,c,v}(k), \quad \forall i,\ \forall c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)},\ v \in c,\ k; \\
& \lambda_{i,c,v}(k) \geq 0 \quad \forall i,\ \forall c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)},\ v \in c,\ k, \\
& \ddot{\tau} \geq 0.
\end{aligned}
$$

In this dual, μ correspond to the first two sets of constraints, while λ and $\ddot{\tau}$ correspond to third and fourth set of constraints. Using the substitution

$$
\ddot{\nu} = \ddot{\tau} + \sum_{i,c\in\mathcal{C}^{(i)}\setminus\mathcal{V}^{(i)},v\in c,\,k} \lambda_{i,c,v}(k)\ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)})/|c|
$$

and the fact that $\lambda_{i,c,v}(k) \geq 0$ and $\ddot{\mathbf{f}}_{i,c}(\mathbf{y}_c^{(i)}) \geq 0$, we can eliminate λ and $\ddot{\tau}$, as well as divide μ's by C, and re-express the above QP as:

$$
\begin{aligned}
\max \quad & \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\ell_{i,c}(k) - \frac{1}{2}C\left|\left|\sum_{i,v\in\mathcal{V}^{(i)},\,k} \mu_{i,v}(k)\Delta\dot{\mathbf{f}}_{i,v}\right|\right|^2 - \frac{1}{2}C\left|\left|\ddot{\nu} + \sum_{i,c\in\mathcal{C}^{(i)},\,k} \mu_{i,c}(k)\Delta\ddot{\mathbf{f}}_{i,c}(k)\right|\right|^2 \\
\text{s.t.} \quad & \mu_{i,c}(k) \geq 0, \quad \forall i,\ \forall c \in \mathcal{C}^{(i)},\ k; \qquad \sum_{k=1}^{K} \mu_{i,v}(k) = 1, \quad \forall i,\ \forall v \in \mathcal{V}^{(i)}; \\
& \mu_{i,c}(k) \leq \mu_{i,v}(k), \quad \forall i,\ \forall c \in \mathcal{C}^{(i)} \setminus \mathcal{V}^{(i)},\ v \in c,\ k; \qquad \ddot{\nu} \geq 0.
\end{aligned}
$$

Appendix B. Generalization bound: Proof of Theorem 9.1

The proof of Theorem 9.1 uses the covering number bounds of Zhang (2002) (in the Data-Dependent Structural Risk Minimization framework (Shawe-Taylor et al., 1998).) Zhang provides generalization guarantees for linear binary classifiers of the form $h_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\mathbf{w}^\top\mathbf{x})$. His analysis is based on the upper bounds on the covering number for the class of linear functions $\mathcal{F}_L(\mathbf{w}, \mathbf{z}) = \mathbf{w}^\top\mathbf{z}$ where the norms of the vectors $\mathbf{w}$ and $\mathbf{z}$ are bounded. We reproduce the relevant definitions and theorems from Zhang (2002) here to highlight the necessary extensions for structured classification.

The covering number is a key quantity in measuring function complexity. Intuitively, the covering number of an infinite class of functions (e.g. parameterized by a set of weights $\mathbf{w}$) is the number of vectors necessary to approximate the values of any function in the class on a sample. Margin-based analysis of generalization error uses the margin achieved by a classifier on the training set to approximate the original function class of the classifier by a finite covering with precision that depends on the margin. Here, we will only define the ∞-norm covering number.

B.1 Binary classification

In binary classification, we are given a sample $S = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{m}$, from distribution D over $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{Y}$ is mapped to ± 1, so we can fold $\mathbf{x}$ and y into $\mathbf{z} = y\mathbf{x}$.

Definition B.1 (Covering Number) *Let $\mathcal{V} = \{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(r)}\}$, where $\mathbf{v}^{(j)} \in \mathbb{R}^m$, be a* covering *of a function class $\mathcal{F}(\mathbf{w}, S)$ with ϵ-precision under the metric ρ, if for all $\mathbf{w}$ there exists a $\mathbf{v}^{(j)}$ such that for each data sample $\mathbf{z}^{(i)} \in S$:*

$$\rho(\mathbf{v}_i^{(j)}, \mathcal{F}(\mathbf{w}, \mathbf{z}^{(i)})) \leq \epsilon.$$

The covering number of a sample S *is the size of the smallest covering: $\mathcal{N}_\infty(\mathcal{F}, \rho, \epsilon, S) = \inf |\mathcal{V}|$ s.t. $\mathcal{V}$ is a covering of $\mathcal{F}(\mathbf{w}, S)$. We also define the* covering number for any sample of size m*: $\mathcal{N}_\infty(\mathcal{F}, \rho, \epsilon, m) = \sup_{S:\ |S|=m} \mathcal{N}_\infty(\mathcal{F}, \rho, \epsilon, S)$.* ▮

When the norms of $\mathbf{w}$ and $\mathbf{z}$ are bounded, we have the following upper bound on the covering number of linear functions under the linear metric $\rho_L(v, v') = |v - v'|$.

Theorem B.2 (Theorem 4 from Zhang (2002)) *If $\|\mathbf{w}\|_2 \leq a$ and $\|\mathbf{z}\|_2 \leq b$, then $\forall$ $\epsilon > 0$,*

$$\log_2 \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, m) \leq 36 \frac{a^2 b^2}{\epsilon^2} \log_2 \left(2 \lceil 4ab/\epsilon + 2 \rceil m + 1\right). \quad ▮$$

In order to use the classifier's margin to bound its expected loss, the bounds below use a stricter, margin-based loss on the training sample that measures the worst loss achieved by the approximate covering based on this margin. Let $f : \mathbb{R} \mapsto [0, 1]$ be a loss function. In binary classification, we let $f(v) = \mathbb{1}(v \leq 0)$ be the step function, so that 0-1 loss of $\mathrm{sgn}(\mathbf{w}^\top \mathbf{x})$ is $f(\mathcal{F}_L(\mathbf{w}, \mathbf{z}))$. The next theorem bounds the expected f loss in terms of the γ-margin loss, $f^\gamma(v) = \sup_{\rho(v,v')<2\gamma} f(v')$, on the training sample. For 0-1 loss and linear metric ρ_L, the corresponding γ-margin loss is $f^\gamma(v) = \mathbb{1}(v \leq 2\gamma)$.

Theorem B.3 (Corollary 1 from Zhang (2002)) *Let $f : \mathbb{R} \mapsto [0, 1]$ be a loss function and $f^\gamma(v) = \sup_{\rho(v,v')<2\gamma} f(v')$ be the γ-margin loss for a metric ρ. Let $\gamma_1 > \gamma_2 > \ldots$ be a decreasing sequence of parameters, and p_i be a sequence of positive numbers such that $\sum_{i=1}^\infty p_i = 1$, then for all $\delta > 0$, with probability of at least $1 - \delta$ over data:*

$$\mathbf{E}_D[f(\mathcal{F}(\mathbf{w}, \mathbf{z}))] \quad \leq \quad \mathbf{E}_S[f^\gamma(\mathcal{F}(\mathbf{w}, \mathbf{z}))] + \sqrt{\frac{32}{m}\left[\ln 4\mathcal{N}_\infty(\mathcal{F}, \rho, \gamma_i, S) + \ln \frac{1}{p_i \delta}\right]}$$

for all $\mathbf{w}$ and γ, where for each fixed γ, we use i to denote the smallest index s.t. $\gamma_i \leq \gamma$.

B.2 Structured classification

We will extend this framework to bound the average per-label loss $\ell^H(\mathbf{y})/L$ for structured classification by defining an appropriate loss f and a function class $\mathcal{F}$ (as well as a metric ρ) such that $f(\mathcal{F})$ computes average per-label loss and $f^\gamma(\mathcal{F})$ provides a suitable γ-margin loss. We will bound the corresponding covering number by building on the bound in Theorem B.2.

We can no longer simply fold $\mathbf{x}$ and $\mathbf{y}$, since $\mathbf{y}$ is a vector, so we let $\mathbf{z} = (\mathbf{x}, \mathbf{y})$. In order for our loss function to compute average per-label loss, it is convenient to make our function class *vector-valued* (instead of scalar-valued as above). We define a new function class $\mathcal{F}_M(\mathbf{w}, \mathbf{z})$, which is a vector of minimum values of $\mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y})$ for each error level $\ell^H(\mathbf{y})$ from 1 to L as described below.

Definition B.4 (dth-error-level function) *The* dth-error-level function $M_d(\mathbf{w}, \mathbf{z})$ *for* $d \in \{1, \ldots, L\}$ *is given by:*

$$M_d(\mathbf{w}, \mathbf{z}) = \min_{\mathbf{y}: \ell^H(\mathbf{y}) = d} \mathbf{w}^\top \Delta \mathbf{f}_i(\mathbf{y}). \quad \blacksquare$$

Definition B.5 (Multi-error-level function class) *The* multi-error-level function class $\mathcal{F}_M(\mathbf{w}, \mathbf{z})$ *is given by:*

$$\mathcal{F}_M(\mathbf{w}, \mathbf{z}) = (M_1(\mathbf{w}, \mathbf{z}), \ldots, M_d(\mathbf{w}, \mathbf{z}), \ldots, M_L(\mathbf{w}, \mathbf{z}))\,. \quad \blacksquare$$

We can now compute the average per-label loss from $\mathcal{F}_M(\mathbf{w}, \mathbf{z})$ by defining an appropriate loss function f_M.

Definition B.6 (Average per-label loss) *The* average per-label loss $f_M : I\!R^{\,L} \mapsto [0, 1]$ *is given by:*

$$f_M(\mathbf{v}) = \frac{1}{L} \arg \min_{d: v_d \leq 0} v_d,$$

where in case $\forall d, \quad v_d > 0$*, we define* $\arg\min_{d: v_d \leq 0} v_d \equiv 0$. $\blacksquare$

With the above definitions, we have an upper bound on the average per-label loss

$$f_M(\mathcal{F}_M(\mathbf{w}, \mathbf{z})) = \frac{1}{L} \arg \min_{d: M_d(\mathbf{w}, \mathbf{z}) \leq 0} M_d(\mathbf{w}, \mathbf{z}) \geq \frac{1}{L} \ell^H(\arg\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y})).$$

Note that the case $\forall d, \quad M_d(\mathbf{w}, \mathbf{z}) > 0$ corresponds to the classifier making no mistakes: $\arg\max_{\mathbf{y}} \mathbf{w}^\top \mathbf{f}_i(\mathbf{y}) = \mathbf{y}$. This upper bound is tight if $\mathbf{y} = \arg\max_{\mathbf{y}'} \mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}')$, Otherwise, it is adversarial: it picks from all $\mathbf{y}'$ which are better ($\mathbf{w}^\top \mathbf{f}(\mathbf{y}) \leq \mathbf{w}^\top \mathbf{f}(\mathbf{y}')$), one that maximizes the Hamming distance from $\mathbf{y}$.

We now need to define an appropriate metric ρ that in turn defines γ-margin loss for structured classification. Since the margin of the hypothesis grows with the number of mistakes, our metric can become "looser" with the number of mistakes, as there is more room for error.

Definition B.7 (Multi-error-level metric) *Let the* multi-error-level metric $\rho_M : I\!R^L \times I\!R^L \mapsto I\!R$ *for a vector in* $I\!R^{\,L}$ *be given by:*

$$\rho_M(\mathbf{v}, \mathbf{v}') = \max_d \frac{|v_d - v'_d|}{d}. \quad \blacksquare$$

We now define the corresponding γ-margin loss using the new metric:

Definition B.8 (γ-margin average per-label loss) *The* γ-margin average per-label loss $f_M^\gamma : I\!R^{\,L} \mapsto [0,1]$ *is given by:*

$$f_M^\gamma(\mathbf{v}) = \sup_{\rho_M(\mathbf{v},\mathbf{v}') \leq 2\gamma} f_M(\mathbf{v}'). \quad \blacksquare$$

Combining the two definitions, we get:

$$f_M^\gamma(\mathcal{F}_M(\mathbf{w},\mathbf{z})) \quad = \quad \sup_{\mathbf{v}:|v_d - M_d(\mathbf{w},\mathbf{z})| \leq 2d\gamma} \frac{1}{L} \arg\min_{d:v_d \leq 0} v_d.$$

We also define the corresponding covering number for our vector-valued function class:

Definition B.9 (Multi-error-level covering number) *Let* $\mathcal{V} = \{\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(r)}\}$*, where* $\mathbf{V}^{(j)} = (\mathbf{V}_1^{(j)}, \ldots, \mathbf{V}_i^{(j)}, \ldots, \mathbf{V}_m^{(j)})$ *and* $\mathbf{V}_i^{(j)} \in I\!R^L$*, be a* covering *of* $\mathcal{F}_M(\mathbf{w}, S)$*, with* ϵ*-precision under the metric* ρ_M*, if for all* $\mathbf{w}$ *there exists a* $\mathbf{V}^{(j)}$ *such that for each data sample* $\mathbf{z}^{(i)} \in S$*:*

$$\rho_M(\mathbf{V}_i^{(j)}, \mathcal{F}_M(\mathbf{w}, \mathbf{z}^{(i)})) \leq \epsilon.$$

The covering number of a sample S *is the size of the smallest covering:* $\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, S) = \inf |\mathcal{V}|$ *s.t.* $\mathcal{V}$ *is a covering of* $\mathcal{F}_M(\mathbf{w}, S)$*. We also define*

$$\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, m) = \sup_{S:\ |S|=m} \mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, S).$$

We provide a bound on the covering number of our new function class in terms of a covering number for the linear function class. Recall that N_c is the maximum number of cliques in $\mathcal{G}(\mathbf{x})$, V_c is the maximum number of values in a clique $|\mathbf{Y}_c|$, q is the maximum number of cliques that have a variable in common, and R_c is an upper-bound on the 2-norm of clique basis functions. Consider a first-order sequence model as an example, with L as the maximum length, and V the number of values a variable takes. Then $N_c = 2L-1$ since we have L node cliques and $L-1$ edge cliques; $V_c = V^2$ because of the edge cliques; and $q = 3$ since nodes in the middle of the sequence participate in 3 cliques: previous-current edge clique, node clique, and current-next edge clique.

Lemma B.10 (Bound on multi-error-level covering number)

$$\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, m) \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c - 1)).$$

Proof: *We will show that* $\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, S) \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, S')$ *for any sample* S *of size* m*, where we construct the sample* S' *of size* $mN_c(V_c - 1)$ *in order to cover the clique potentials as described below. Note that this is sufficient since* $\mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, S') \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c - 1))$*, by definition, so*

$$\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, m) = \sup_{S:|S|=m} \mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon q, S) \leq \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c - 1)).$$

The construction of S' *below is inspired by the proof technique in Collins (2001), but the key difference is that our construction is linear in the number of cliques* N_c *and exponential* *in the number of label variables per clique, while his is exponential in the total number of label variables per example. This reduction in size comes about because our covering approximates the values of clique potentials* $\mathbf{w}^\top \Delta\mathbf{f}_{i,c}(\mathbf{y}_c)$ *for each clique* c *and clique assignment* $\mathbf{y}_c$ *as opposed to the values of entire assignments* $\mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})$.

For each sample $\mathbf{z} \in S$*, we create* $N_c(V_c - 1)$ *samples* $\Delta\mathbf{f}_{i,c}(\mathbf{y}_c)$*, one for each clique* c *and each assignment* $\mathbf{y}_c \neq \mathbf{y}_c^{(i)}$*. We construct a set of vectors* $\mathcal{v} = \{\mathbf{v}^{(1)}, \ldots, \mathbf{v}^{(r)}\}$*, where* $\mathbf{v}^{(j)} \in I\!R^{mN_c(V_c-1)}$*. The component of* $\mathbf{v}^{(j)}$ *corresponding to the sample* $\mathbf{z}^{(i)}$ *and the assignment* $\mathbf{y}_c$ *to the labels of the clique* c *will be denoted by* $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$*. For convenience, we define* $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c^{(i)}) = 0$ *for correct label assignments, as* $\Delta\mathbf{f}_{i,c}(\mathbf{y}_c^{(i)}) = 0$*. To make* $\mathcal{v}$ *an* ∞*-norm covering of* $\mathcal{F}_L(\mathbf{w}, S')$ *under* ρ_L*, we require that for any* $\mathbf{w}$ *there exists a* $\mathbf{v}^{(j)} \in \mathcal{v}$ *such that for each sample* $\mathbf{z}^{(i)}$*:*

$$|\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c) - \mathbf{w}^\top \Delta\mathbf{f}_{i,c}(\mathbf{y}_c)| \le \epsilon; \quad \forall c \in \mathcal{C}^{(i)}, \ \forall \mathbf{y}_c. \tag{49}$$

By Definition B.1, the number of vectors in $\mathcal{v}$ *is given by* $r = \mathcal{N}_\infty(\mathcal{F}_L, \rho_L, \epsilon, mN_c(V_c - 1))$.

We can now use $\mathcal{v}$ *to construct a covering* $\mathcal{V} = \{\mathbf{V}^{(1)}, \ldots, \mathbf{V}^{(r)}\}$*, where*

$$\mathbf{V}^{(j)} = (\mathbf{V}_1^{(j)}, \ldots, \mathbf{V}_i^{(j)}, \ldots, \mathbf{V}_m^{(j)})$$

and $\mathbf{V}_i^{(j)} \in I\!R^L$*, for our multi-error-level function* $\mathcal{F}_M$*. Let* $\mathbf{v}_i^{(j)}(\mathbf{y}) = \sum_c \mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$*, and* $M_d(\mathbf{v}_i^{(j)}, \mathbf{z}^{(i)}) = \min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{v}_i^{(j)}(\mathbf{y})$*, then*

$$\mathbf{V}_i^{(j)} = (M_1(\mathbf{v}^{(j)}, \mathbf{z}^{(i)}), \ldots, M_d(\mathbf{v}^{(j)}, \mathbf{z}^{(i)}), \ldots, M_L(\mathbf{v}^{(j)}, \mathbf{z}^{(i)})) \ . \tag{50}$$

Note that $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$ *is zero for all cliques* c *for which the assignment is correct:* $\mathbf{y}_c = \mathbf{y}_c^{(i)}$*. Thus for an assignment* $\mathbf{y}$ *with* d *mistakes, at most* dq $\mathbf{v}_{i,c}^{(j)}(\mathbf{y}_c)$ *will be non-zero, as each label can appear in at most* q *cliques. By combining this fact with Eq. (49), we obtain:*

$$\left|\mathbf{v}_i^{(j)}(\mathbf{y}) - \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})\right| \le dq\epsilon, \qquad \forall i, \ \ \forall \mathbf{y} : \ell_i^H(\mathbf{y}) = d. \tag{51}$$

We conclude the proof by showing that $\mathcal{V}$ *is a covering of* $\mathcal{F}_M$ *under* ρ_M*: For each* $\mathbf{w}$*, pick* $\mathbf{V}^{(j)} \in \mathcal{V}$ *such that the corresponding* $\mathbf{v}^{(j)} \in \mathcal{v}$ *satisfies the condition in Eq. (49). We must now bound:*

$$\rho_M(\mathbf{V}_i^{(j)}, \mathcal{F}_M(\mathbf{w}, \mathbf{z}^{(i)})) = \max_d \frac{|\min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{v}_i^{(j)}(\mathbf{y}) - \min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})|}{d}.$$

Fix any i*. Let* $\mathbf{y}_d^{\mathbf{v}} = \arg\min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{v}_i^{(j)}(\mathbf{y})$ *and* $\mathbf{y}_d^{\mathbf{w}} = \arg\min_{\mathbf{y}:\ell_i^H(\mathbf{y})=d} \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y})$*. Consider the case where* $\mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{v}}) \ge \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}_d^{\mathbf{w}})$ *(the reverse case is analogous), we must prove that:*

$$\mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{v}}) - \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}_d^{\mathbf{w}}) \le \mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{w}}) - \mathbf{w}^\top \Delta\mathbf{f}_i(\mathbf{y}_d^{\mathbf{w}}) \le dq\epsilon \ ; \tag{52}$$

where the first step follows from definition of $\mathbf{y}_d^{\mathbf{v}}$*, since* $\mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{v}}) \le \mathbf{v}_i^{(j)}(\mathbf{y}_d^{\mathbf{w}})$*. The last step is a direct consequence of Eq. (51). Hence* $\rho_M(\mathbf{V}_i^{(j)}, \mathcal{F}_M(\mathbf{w}, \mathbf{z}^{(i)})) \le q\epsilon$. ∎

Lemma B.11 (Numeric bound on multi-error-level covering number)

$$\log_2 \mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \epsilon, m) \le 36 \frac{R_c^2 \|\mathbf{w}\|_2^2 q^2}{\epsilon^2} \log_2 \left(1 + 2 \left\lceil 4 \frac{R_c \|\mathbf{w}\|_2 q}{\epsilon} + 2 \right\rceil m N_c (V_c - 1)\right).$$

Proof: *Substitute Theorem B.2 into Lemma B.10.* ∎

Theorem B.12 (Multi-label analog of Theorem B.3) *Let f_M and $f_M^\gamma(v)$ be as defined above. Let $\gamma_1 > \gamma_2 > \dots$ be a decreasing sequence of parameters, and p_i be a sequence of positive numbers such that $\sum_{i=1}^\infty p_i = 1$, then for all $\delta > 0$, with probability of at least $1-\delta$ over data:*

$$E_{\mathbf{z}} f_M(\mathcal{F}_M(\mathbf{w}, \mathbf{z})) \quad \le \quad E_S f_M^\gamma(\mathcal{F}_M(\mathbf{w}, \mathbf{z})) + \sqrt{\frac{32}{m}\left[\ln 4\mathcal{N}_\infty(\mathcal{F}_M, \rho_M, \gamma_i, S) + \ln \frac{1}{p_i \delta}\right]}$$

for all $\mathbf{w}$ and γ, where for each fixed γ, we use i to denote the smallest index s.t. $\gamma_i \le \gamma$.
Proof: *Similar to the proof of Zhang's Theorem 2 and Corollary 1 Zhang (2002) where in Step 3 (derandomization) we substitute the vector-valued $\mathcal{F}_M$ and the metric ρ_M.* ∎

Theorem 9.1 follows from above theorem with $\gamma_i = R_c \|\mathbf{w}\|_2 / 2^i$ and $p_i = 1/2^i$ using an argument identical to the proof of Theorem 6 in Zhang (2002).

Appendix C. Structured SMO: Proof of Theorem 10.1

We repeat the theorem here for convenience:

Theorem C.1 *The KKT conditions in Eq. (39) and Eq. (40) are equivalent.*

Eq. (39) ⇒ Eq. (40). Assume Eq. (39). Suppose, we have a violation of KKT1: for some $c, \mathbf{y}_c$, $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0$, but $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$. Since $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = \max_{\mathbf{y} \sim \mathbf{y}_c} \alpha_i(\mathbf{y}) = 0$, then $\alpha_i(\mathbf{y}) = 0$, $\forall \mathbf{y} \sim \mathbf{y}_c$. Hence, by Eq. (39), $v_i(\mathbf{y}) \le v_i(\overline{\mathbf{y}})$, $\forall \mathbf{y} \sim \mathbf{y}_c$. But $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$ implies the opposite: there exists $\mathbf{y} \sim \mathbf{y}_c$ such that $v_i(\mathbf{y}) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$, which also implies $v_i(\mathbf{y}) > v_i(\overline{\mathbf{y}})$, a contradiction.

Now suppose we have a violation of KKT2: for some $i, \mathbf{y}_c$, $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, but $\widehat{v}_{i,c}(\mathbf{y}_c) < \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$. Then $v_i(\mathbf{y}) < v_i(\overline{\mathbf{y}})$, $\forall \mathbf{y} \sim \mathbf{y}_c$. But $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$ implies there exists $\mathbf{y} \sim \mathbf{y}_c$ such that $\alpha_i(\mathbf{y}) > 0$. For that $\mathbf{y}$, by Eq. (39), $v_i(\mathbf{y}) \ge v_i(\overline{\mathbf{y}})$, a contradiction.

Eq. (40) ⇒ Eq. (39). Assume Eq. (40). Suppose we have a violation of KKT1: for some $\mathbf{y}$, $\alpha_i(\mathbf{y}) = 0$, but $v_i(\mathbf{y}) > v_i(\overline{\mathbf{y}})$. This means that $\mathbf{y}$ is the optimum of $v_i(\cdot)$, hence $\widehat{v}_{i,c}(\mathbf{y}_c) = v_i(\mathbf{y}) > v_i(\overline{\mathbf{y}}) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$, $\forall c \in \mathcal{T}^{(i)}, \mathbf{y}_c \sim \mathbf{y}$. But by Eq. (40), if $\widehat{v}_{i,c}(\mathbf{y}_c) > \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$, then we cannot have $\widehat{\alpha}_{i,c}(\mathbf{y}_c) = 0$. Hence all the $\mathbf{y}$-consistent α_i max-marginals are positive $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, $\forall c \in \mathcal{T}^{(i)}$, and it follows that all the $\mathbf{y}$-consistent marginals μ_i are positive as well $\mu_{i,c}(\mathbf{y}_c) > 0$, $\forall c \in \mathcal{T}^{(i)}$ (since sum upper-bounds max). But $\alpha_i(\mathbf{y}) = \frac{\prod_{c \in \mathcal{T}^{(i)}} \mu_{i,c}(\mathbf{y}_c)}{\prod_{c \in \mathcal{S}^{(i)}} \mu_{i,s}(\mathbf{y}_s)}$, so if all the $\mathbf{y}$-consistent marginals are positive, then $\alpha_i(\mathbf{y}) > 0$, a contradiction.

Now suppose we have a violation of KKT2: for some $\mathbf{y}$, $\alpha_i(\mathbf{y}) > 0$, but $v_i(\mathbf{y}) < v_i(\overline{\mathbf{y}})$. Since $\alpha_i(\mathbf{y}) > 0$, we know that all the $\mathbf{y}$-consistent α_i max-marginals are positive $\widehat{\alpha}_{i,c}(\mathbf{y}_c) > 0$, $\forall c \in \mathcal{T}^{(i)}$. By Eq. (40), $\widehat{v}_{i,c}(\mathbf{y}_c) \ge \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$, $\forall c \in \mathcal{T}^{(i)}$. Note that trivially $\max_{\mathbf{y}'} v_i(\mathbf{y}') = \max(\widehat{v}_{i,c}(\mathbf{y}'_c), \widehat{v}_{i,c}(\overline{\mathbf{y}'_c}))$ for any clique c and clique assignment $\mathbf{y}'_c$. Since

$\widehat{v}_{i,c}(\mathbf{y}_c) \geq \widehat{v}_{i,c}(\overline{\mathbf{y}_c})$, $\forall c \in \mathcal{T}^{(i)}$, then $\max_{\mathbf{y}'} v_i(\mathbf{y}') = \widehat{v}_{i,c}(\mathbf{y}_c),$, $\forall c \in \mathcal{T}^{(i)}$. That is, $\widehat{v}_{i,c}(\mathbf{y}_c)$ is the optimal value. We will show that $v_i(\mathbf{y}) = \widehat{v}_{i,c}(\mathbf{y}_c)$, a contradiction. To show that this, we consider any two adjacent nodes in the tree $\mathcal{T}^{(i)}$, cliques a and b, with a separator s, and show that $\widehat{v}_{i,a\cup b}(\mathbf{y}_{a\cup b}) = \widehat{v}_{i,a}(\mathbf{y}_a) = \widehat{v}_{i,b}(\mathbf{y}_b)$. By chaining this equality from the root of the tree to all the leaves, we get $v_i(\mathbf{y}) = \widehat{v}_{i,c}(\mathbf{y}_c)$ for any c.

We need to introduce some more notation to deal with the two parts of the tree induced by cutting the edge between a and b. Let $\{A, B\}$ be a partition of the nodes $\mathcal{T}^{(i)}$ (cliques of $\mathcal{C}^{(i)}$) resulting from removing the edge between a and b such that $a \in A$ and $b \in B$. We denote the two subsets of an assignment $\mathbf{y}$ as $\mathbf{y}_A$ and $\mathbf{y}_B$ (with overlap at $\mathbf{y}_s$). The value of an assignment $v_i(\mathbf{y})$ can be decomposed into two parts: $v_i(\mathbf{y}) = v_{i,A}(\mathbf{y}_A) + v_{i,B}(\mathbf{y}_B)$, where $v_{i,A}(\mathbf{y}_A)$ and $v_{i,B}(\mathbf{y}_B)$ only count the contributions of their constituent cliques. Take any maximizer, $\mathbf{y}^{(a)} \sim \mathbf{y}_a$ with $v_i(\mathbf{y}^{(a)}) = \widehat{v}_{i,a}(\mathbf{y}_a) \geq \widehat{v}_{i,a}(\overline{\mathbf{y}_a})$ and any maximizer $\mathbf{y}^{(b)} \sim \mathbf{y}_b$ with $v_i(\mathbf{y}^{(b)}) = \widehat{v}_{i,b}(\mathbf{y}_b) \geq \widehat{v}_{i,b}(\overline{\mathbf{y}_b})$, which by definition agree with $\mathbf{y}$ on the intersection s. We decompose the two associated values into the corresponding parts: $v_i(\mathbf{y}^{(a)}) = v_i(\mathbf{y}_A^{(a)}) + v_i(\mathbf{y}_B^{(a)})$ and $v_i(\mathbf{y}^{(b)}) = v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(b)})$. We create a new assignment that combines the best of the two: $\mathbf{y}^{(s)} = \mathbf{y}_A^{(b)} \cup \mathbf{y}_B^{(a)}$. Note that $v_i(\mathbf{y}^{(s)}) = v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(a)}) = \widehat{v}_{i,s}(\mathbf{y}_s)$, since we essentially fixed the intersection s and maximized over the rest of the variables in A and B separately. Now $\widehat{v}_{i,a}(\mathbf{y}_a) = \widehat{v}_{i,b}(\mathbf{y}_b) \geq \widehat{v}_{i,s}(\mathbf{y}_s)$ since they are optimal as we said above. Hence we have $v_i(\mathbf{y}_A^{(a)}) + v_i(\mathbf{y}_B^{(a)}) = v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(b)}) \geq v_i(\mathbf{y}_A^{(b)}) + v_i(\mathbf{y}_B^{(a)})$ which implies that $v_i(\mathbf{y}_A^{(a)}) \geq v_i(\mathbf{y}_A^{(b)})$ and $v_i(\mathbf{y}_B^{(b)}) \geq v_i(\mathbf{y}_B^{(a)})$. Now we create another assignment that clamps the value of both a and b: $\mathbf{y}^{(a\cup b)} = \mathbf{y}_A^{(a)} \cup \mathbf{y}_B^{(b)}$. The value of this assignment is optimal $v_i(\mathbf{y}^{(a\cup b)}) = v_i(\mathbf{y}_A^{(a)}) + v_i(\mathbf{y}_B^{(b)}) = v_i(\mathbf{y}^{(a)}) = v_i(\mathbf{y}^{(b)})$. ∎

References

P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proc. ICML*, 2004.

R.K Ahuja and J.B. Orlin. Inverse optimization, Part I: Linear programming and general problem. *Operations Research*, 35:771–783, 2001.

Y. Altun, A. Smola, and T. Hofmann. Exponential families for conditional random fields. In *Proc. UAI*, 2004.

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *Proc. ICML*, 2003.

P. Bartlett, M. Collins, B. Taskar, and D. McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *NIPS*, 2004.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. Technical Report 638, Department of Statistics, U.C. Berkeley, 2003.

D. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1999.

D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Programming.* Athena Scientific, 1997.

J. E. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48, 1986.

C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, UK, 1995.

S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. In *PAMI*, 2004.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *ICCV*, 1999a.

Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *CVPR*, 1999b.

D. Burton and Ph. L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53:45–61, 1992.

M. Collins. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In *IWPT*, 2001.

M. Collins. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In H. Bunt, J. Carroll, and G. Satta, editors, *New Developments in Parsing Technology.* Kluwer, 2004.

T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, New York, 1991.

R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer, New York, 1999.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(5):265–292, 2001.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proc AAAI98*, pages 509–516, 1998.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, 2000.

M. H. DeGroot. *Optimal Statistical Decisions.* McGraw-Hill, New York, 1970.

L. Devroye, L. Györfi, and G. Lugosi. *Probabilistic theory of pattern recognition.* Springer-Verlag, New York, 1996.

Y. Freund and R.E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learing Theory*, 1998.

T. Friess, N. Cristianini, and C. Campbell. The kernel adatron algorithm: a fast and simple learning procedure for support vector machine. In *Proc. ICML*, 1998.

M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, 1979.

L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classification. In *Proc. IJCAI01 Workshop on Text Learning: Beyond Supervision*, Seattle, Wash., 2001.

D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binar images. *J. R. Statist. Soc. B*, 51, 1989.

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19, 2003.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer-Verlag, New York, 2001.

C. Heuberger. Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, 8, 2004.

Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, 1997.

M. Jerrum and A. Sinclair. Polynomial-time approximation algorithms for the ising model. *SIAM J. Comput.*, 22, 1993.

T. Joachims. Transductive inference for text classification using support vector machines. In *Proc. ICML99*, pages 200–209. Morgan Kaufmann Publishers, San Francisco, US, 1999.

R. Kassel. *A Comparison of Approaches to On-line Handwritten Character Recognition.* PhD thesis, MIT Spoken Language Systems Group, 1995.

J. Kivinen and M. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

J. Kleinberg and E. Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. In *FOCS*, 1999.

J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

V. Kolmogorov and R. Zabih. What energy functions can be minimized using graph cuts? In *PAMI*, 2002.

S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *NIPS*, 2003.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *ICML*, 2004.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* J. Wiley, New York, 1999.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *NIPS*, 2001.

A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Proc. ICML*, 2000.

J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, New York, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Francisco, 1988.

D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. In *Proc. ACM SIGIR*, 2003.

J. Platt. Using sparseness and analytic QP to speed training of support vector machines. In *NIPS*, 1999.

R. B. Potts. Some generalized order-disorder transformations. *Proc. Cambridge Phil. Soc.*, 48, 1952.

J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 2001.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL*, 2003.

J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE Trans. on Information Theory*, 44(5): 1926–1940, 1998.

C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *Proc. ICML*, 2004.

A Tarantola. *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation.* Elsevier, Amsterdam, 1987.

B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, 2002.

B. Taskar, M.F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Proc. NIPS*, 2003.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Twenty-first international conference on Machine learning*, 2004.

L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, New York, New York, 1995.

A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani. Global protein function prediction from protein-protein interaction networks*h*. *Nature Biotechnology*, 6, 2003.

O. Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision.* PhD thesis, Cornell University, 1999.

G. Wahba, C. Gu, Y. Wang, and R. Chappell. Soft classification, a.k.a. risk estimation, via penalized log likelihood and smoothing spline analysis of variance. In *Computational Learning Theory and Natural Learning Systems*, 1993.

M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. In *Proc. Allerton Conference on Communication, Control and Computing*, 2002.

M. Wainwright and M. I. Jordan. Variational inference in graphical models: The view from the marginal polytope. In *Proc. Allerton Conference on Communication, Control and Computing*, 2003.

J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.

Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 18(2), 2002.

J. Yedidia, W. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, 2000.

J. Zhang and Z. Ma. A network flow method for solving inverse combinatorial optimization problems. *Optimization*, 37:59–72, 1996.

T. Zhang. Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2:527–550, 2002.

J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *Proc. NIPS*, 2001.