1 Conditional Graphical Models

Fernando Pérez-Cruz
Gatsby Computational Neuroscience Unit. Queen Square, London WC1N 3AR, UK.
DTSC. University Carlos III. Leganés (Madrid) SPAIN. fernando@tsc.uc3m.es

Zoubin Ghahramani
Engineering Department. Cambridge University. ...
zoubin@eng.cam.ac.uk

Massimiliano Pontil
Department of Computer Science, UCL. Gower Street London WC1E 6BT, UK.
m.pontil@cs.ucl.ac.uk

In this chapter we propose a modification of CRF-like algorithms that allows for solving large-scale structured classification problems. Our approach consists in upper bounding the CRF functional in order to decompose its training into independent optimisation problems per clique. Furthermore we show that each sub-problem corresponds to solving a multiclass learning task in each clique, which enlarges the applicability of these tools for large-scale structural learning problems. Before presenting the Conditional Graphical Model (CGM), as we refer to this procedure, we review the family of CRF algorithms. We concentrate on the best known procedures and standard generalisations of CRFs. The objective of this introduction is analysing from the same viewpoint the proposed solutions in the literature to tackle this problem, which allows comparing their different features. We complete the chapter with a case study, in which we show the possibility to work with large-scale problems using CGM and that the obtained performance is comparable to the result with CRF-like algorithms.

1.1 Introduction

In the last decade machine learning tools have moved from heuristic based approaches to more theoretical based ones. There has been an explosion of work on

theoretical and algorithmic developments, as well as potential applications to real-world problems. In particular, in the pattern recognition field, the appearance of the Support Vector Machines (SVMs) (Boser et al. (1992)) blossomed the research into new machine learning algorithms and they bought the kernel concept into the machine learning community (Schölkopf and Smola (2001)). Nevertheless, most real-life applications of pattern recognition cannot be readily cast as a binary (or multiclass) learning problem, because they present an inherent structure that cannot be exploited by general classification algorithms. Some of these applications such as speech or object recognition have developed their own research field. They use a mixture of machine learning tools with specific knowledge about each application to provide meaningful solutions to these relevant problems. But there are many others that can benefit from a machine learning approach, if we are capable of embedding their structure into a generic pattern recognition tool.

Conditional Random Fields (CRFs) address this general problem as an extension of logistic regression for multiclass problems. In their seminal work Lafferty et al. (2001) assume the output for each sample can be expressed as a set of interdependent labels and that this dependency can be captured by an undirected graphical model. They exploit the graphical model structure to avoid the exponential growth of the complexity with the number of labels in each output. There are many different machine learning problems that can be represented by this setting (Dietterich, 2002), such as optical character recognition, part-of-speech tagging, collective web page classification, mobile fraud detection, or pitch accent prediction. There has been several extensions to CRFs using kernels (Altun et al., 2004a,b; Lafferty et al., 2004), Bayesian learning (Qi et al., 2005), maximum margin solution (Altun et al., 2003; Taskar et al., 2004) and 2D CRFs (Kumar and Hebert, 2004).

Although the use of an undirected graphical model makes these procedures tractable, they are difficult to train, needing custom-made optimization tools, and they cannot solve large-scale problems. In this chapter, we present (kernel) Conditional Graphical Models (CGMs), which simplifies the training phase of CRF-like algorithms to solve large-scale structured classification problems. This algorithmic proposal is based on the same principle used for solving the binary classification problem, in which the 0–1 loss is replaced by a convex upper bound to ensure the optimal solution can be easily computed. In our case, we replace the CRF-like algorithm loss by another convex loss-function that decouples the training of each clique in the undirected graph. CGM optimisation is solved independently per clique using any general multi-classification algorithm. CGM complexity depends on the selected multiclass learning tool, so it opens the possibility of solving large-scale problems as there are inexpensive multiclass tools, such as SVMs.

Outline of the Chapter This chapter is divided in two main sections. In the first one, we review CRF-like algorithms and present them using the same notation. Studying the different approaches for solving this fascinating machine learning problem from the same perspective allows a broad comparison between these algorithms. This knowledge helps us understand which algorithm is most

suitable for each structural learning problem. We have also made an effort to simplify the notation so the algorithms are readily understood. In the second part, we introduce Conditional Graphical Models. We first show how a simple transformation of the general CRF-algorithm loss-function can reduce enormously the complexity of its training procedure and then examine in detail the hinge-loss case, helping us understand how the proposed procedure works. We complete the chapter with a case study, showing the advantages of CGMs for solving large-scale problems and using more complex structures.

1.2 A Unifying Review

The objective of this section is two fold; first present all CRF-like algorithms in a unifying framework in order to compare them and understand their different properties; and second introduce the proposed notation. For these algorithms notation is an issue of its own, as it can be complicated and might be difficult to follow.

We address the general supervised classification problem, given a labelled training data set $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, predict the label $\mathbf{y}_*$ for a new given input $\mathbf{x}_*$, where the input $\mathbf{x}_n \in \mathcal{X}$ and the label $\mathbf{y}_n \in \mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_L$ in which each $\mathcal{Y}_\ell = \{1, 2, \ldots, q\}$. L might depend on each training example (L_n) and q might be different for each element in $\mathcal{Y}$ (q_ℓ), but to keep the notation simple we use a unique L for all the samples and a unique q for all the labels. This simplification is exclusive used for presentation clarity and does not limit the applicability of any of the presented procedures for the more general case. This problem can be seen as a huge multi-class learning problem with q^L possible labels, and it is general enough to contain as special cases standard multi-class classification ($L = 1$) and binary classification ($L = 1$ and $q = 2$).

We start from a probabilistic model that enables us to compute the posterior probability for all the possible labellings: $p(\mathbf{y}_*|\mathbf{x}_*, \mathcal{D})$. We are interested in finding the labelling with highest probability: $\max_{\mathbf{y}_*}\{p(\mathbf{y}_*|\mathbf{x}_*, \mathcal{D})\}$. From the Maximum a Posteriori (MAP) approach to learning models, we make the connection with regularised risk minimization, and solutions to the problem using non-probabilistic discriminative methods, such as Support Vector Machines (SVMs).

We are given a set of features ($\boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y}_n)$) that transform the data into a feature space, in which we can solve the classification problem using a linear combination of these features. The *softmax* function:

$$p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w}) = \frac{\exp\left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y}_n)\right)}{\sum_{\mathbf{y}} \exp\left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y})\right)} \tag{1.1}$$

is a standard likelihood function for linear multi-classification problems. In (1.1), we are using an exponential family model to represent the likelihood function, in

which a set of features are linearly combine to construct the probability density of $\mathbf{y}_n$. The denominator is known as the partition function and it comprises the sum over all q^L possible labelling of $\mathbf{y}$ to ensure that $p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w})$ adds up to one. In this equation we have used $\mathbf{y}$ as a compact running index to represent the sum over all possible labelling, i.e. $\sum_{\mathbf{y}} \exp\left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y})\right) = \sum_{i_1=1}^{q} \sum_{i_2=1}^{q} \cdots \sum_{i_L=1}^{q} \exp\left(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y})\right)$ with $\mathbf{y} = [i_1, i_2, \ldots, i_L]^\top$.

We can compute the posterior over the weights using Bayes rule:

$$p(\mathbf{w}|\mathbf{Y}, \mathbf{X}) = \frac{p(\mathbf{Y}|\mathbf{w}, \mathbf{X})p(\mathbf{w})}{p(\mathbf{Y}|\mathbf{X})} \tag{1.2}$$

where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ and $p(\mathbf{Y}|\mathbf{w}, \mathbf{X}) = \prod_n p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w})$. The prior over $\mathbf{w}$ is usually assumed to be an independent and identical zero-mean Gaussian distribution for each component:

$$p(\mathbf{w}) = \frac{1}{\sqrt{(2\pi C)^H}} \exp\left(\frac{\|\mathbf{w}\|^2}{2C}\right) \tag{1.3}$$

where $\mathbf{w} \in \mathbb{R}^H$.

The prediction for a new data point, integrating out the weights, is:

$$p(\mathbf{y}_*|\mathbf{x}_*, \mathcal{D}) = \int p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{w})p(\mathbf{w}|\mathbf{Y}, \mathbf{X})d\mathbf{w} \tag{1.4}$$

The complexity of this problem grows exponentially in L –the length of the label vector– and we need to simplify it to work with large values of q and/or L. The complexity in (1.4) depends exponentially in L in many ways. First of all, we need to solve this equation for all q^L possible labelling of $\mathbf{y}_*$. Second, the complexity of $p(\mathbf{y}_*|\mathbf{x}_*, \mathbf{w})$ depends on the partition function in (1.1), which is the sum of q^L terms. Finally to compute the posterior term $p(\mathbf{w}|\mathbf{Y}, \mathbf{X})$ we need to evaluate the likelihood of the N training examples and each needs to evaluate a sum of q^L terms. Therefore the total complexity is $O(Nq^L)$. Lafferty et al. (2001) propose to use an undirected acyclic graph over the labels. This graph makes it possible to efficiently compute the denominator in (1.1) using a forward-backward algorithm. Lafferty et al. (2001) define Conditional Random Fields (CRFs) as:

Definition 1 (Conditional Random Field) *Let $G = (V, E)$ be a graph such that $\mathbf{y}_n$ is indexed by the vertices of G, then $(\mathbf{x}_n, \mathbf{y}_n)$ is a* conditional random field *in case, when conditioned on $\mathbf{x}_n$, the random variables $\mathbf{y}_n = [y_{n1}, y_{n2}, \ldots, y_{nL}]^\top$ obey the Markov property with respect to the graph:*

$$p(y_{n\ell}|\mathbf{x}_n, y_{n\ell'}, \forall \ell' \neq \ell) = p(y_{n\ell}|\mathbf{x}_n, y_{n\ell'}, \forall \ell' \overset{n}{\sim} \ell) \tag{1.5}$$

where $\ell' \overset{n}{\sim} \ell$ indicates that node ℓ' is neighbour of node ℓ in the graph.

Therefore by the fundamental theorem of random fields, the distribution of $\mathbf{y}_n$ can be simplified to:

$$p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w}) \propto \exp\left(\sum_{t=1}^{T} \mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt})\right) \tag{1.6}$$

where the sum over t runs over the T maximal cliques in the graph. Note that each feature depends on the clique t, which can be used to provide a different set of features for each clique. For example, a feature can select a part of the input $\mathbf{x}_n$, which is relevant for the labels in the t^{th} clique.

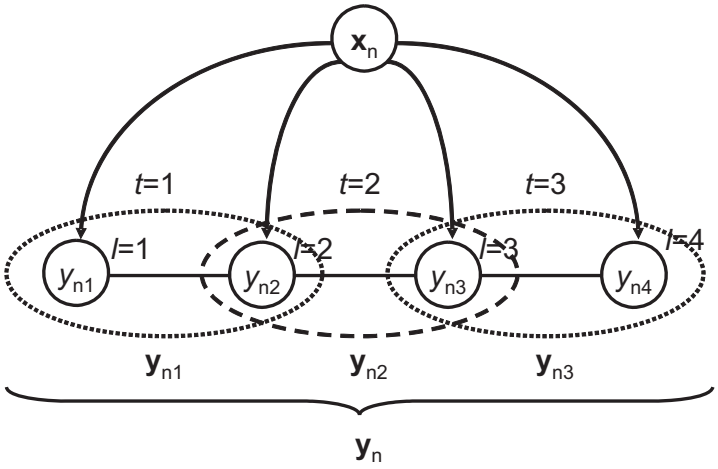

Figure 1.1 We show a chain for a 4D label $\mathbf{y}_n = [y_{n1}, y_{n2}, y_{n3}, y_{n4}]^\top$. We have labelled the nodes ($y_{n\ell}$) and the cliques ($\mathbf{y}_{nt}$) from left to right, i.e. $\mathbf{y}_{n2} = [y_{n2}, y_{n3}]^\top$. Therefore boldfaced $\mathbf{y}_{n2}$ indicates the second clique and italic y_{n2} indicates the second node. We have also introduced $\mathbf{x}_n$ in the graph to indicate that the labels are conditioned on the input.

Before presenting the family of CRF algorithms, let us explicitly state the notation that is being used throughout the paper. The running indices n, ℓ and t represent, respectively, the training samples, the nodes in the graph (the components in each $\mathbf{y}_n = [y_{n1}, y_{n2}, \ldots, y_{n\ell}]^\top$) the cliques in the graph. Therefore $y_{n\ell}$ is the ℓ-th entry in the n training sample and $\mathbf{y}_{nt}$ represents the labels of the n training sample associated with the t-th clique, i.e. $\mathbf{y}_{nt} = [y_{n\ell_1}, y_{n\ell_2}, \ldots]^\top$ and its length depends on the number of variables in each clique (typically 2). We also use as running indices y_ℓ, $\mathbf{y}_t$ and $\mathbf{y}$ to denote that a sum runs, respectively, over all possible configurations in node ℓ, all possible configurations in t-th clique and all possible labelling in the graph. Hence, for a node: $\sum_{y_\ell}(\cdot) = \sum_{i=1}^{q}(\cdot)$; for a clique

with two nodes: $\sum_{\mathbf{y}_t}(\cdot) = \sum_{i_1=1}^{q}\sum_{i_2=1}^{q}(\cdot)$ with $\mathbf{y}_t = [i_1, i_2]^\top$; and for the whole graph: $\sum_{\mathbf{y}}(\cdot) = \sum_{i_1=1}^{q}\sum_{i_2=1}^{q}\cdots\sum_{i_L=1}^{q}(\cdot)$ with $\mathbf{y} = [i_1, i_2, \ldots i_L]^\top$. The subindex in $y/\mathbf{y}$ tells us what are we summing over, the configuration of nodes (y_ℓ), cliques ($\mathbf{y}_t$) or graphs ($\mathbf{y}$). We use boldface for $\mathbf{y}$ and $\mathbf{y}_t$ as they represent a vector of labels and italic for y_ℓ as it represents a scalar value. Although using the index to carry extra information might be misleading at first, it greatly simplifies the notation in the paper. For example, it allows to understand the differences between $\mathbf{y}_n$ (the output of the n training example) and $\mathbf{y}_t$ (a running index over all possible labels in t-th clique) without explicitly mentioning it. Also, we use the standard matrix notation with bold upper cases for matrices, bold lower cases for column vectors and italic for scalars. We have depicted in Figure 1.1 a simple acyclic graph over $\mathbf{y}$ to show a type of graphical model supported by CRFs and to clarify the notation. In this graph each $\mathbf{y}_n$ has four labels $\mathbf{y}_n = [y_{n1}, y_{n2}, y_{n3}, y_{n4}]^\top$ and each clique has two labels $\mathbf{y}_{n1} = [y_{n1}, y_{n2}]^\top$ ..., $\mathbf{y}_{n3} = [y_{n3}, y_{n4}]^\top$. We use throughout the paper boldface for cliques and graphs and italic for nodes.

Lafferty et al. (2001) propose to train the parameters of the CRF model by maximum likelihood (ML). We present the maximum a posteriori (MAP) version, because it is easier to relate to similar approaches and because we can readily obtain the ML solution from the MAP functional by eliminating the prior term.

$$\begin{aligned}\mathbf{w}_{MAP} &= \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{w}|\mathbf{Y}, \mathbf{X}) = \operatorname*{argmin}_{\mathbf{w}} \{-\log(p(\mathbf{w})) - \log(p(\mathbf{Y}|\mathbf{w}, \mathbf{X}))\} \\ &= \operatorname*{argmin}_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\left[\log\left(\sum_{\mathbf{y}}\exp(o(\mathbf{x}_n, \mathbf{y}))\right) - o(\mathbf{x}_n, \mathbf{y}_n)\right]\right\}\end{aligned} \tag{1.7}$$

where

$$o(\mathbf{x}_n, \mathbf{y}) = \sum_{t=1}^{T}\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_t), \tag{1.8}$$

and the sum over $\mathbf{y}$ runs over the q^L possible labels. This sum can be efficiently computed using a forward-backward algorithm if the proposed graph for the CRF has no cycles, see Lafferty et al. (2001) for further details.

In (1.7) we readily see that the loss-function compares the true output $o(\mathbf{x}_n, \mathbf{y}_n)$ with all the other outputs $\log\left(\sum_{\mathbf{y}}\exp(o(\mathbf{x}_n, \mathbf{y}))\right)$. This loss-function is always nonnegative $\log\left(\sum_{\mathbf{y}}\exp(o(\mathbf{x}_n, \mathbf{y}))\right) > \log(\exp(o(\mathbf{x}_n, \mathbf{y}_n))) = o(\mathbf{x}_n, \mathbf{y}_n)$ and it is close to zero iff $o(\mathbf{x}_n, \mathbf{y}_n) \gg o(\mathbf{x}_n, \mathbf{y})\ \forall \mathbf{y} \neq \mathbf{y}_n$. The norm of $\mathbf{w}$ is a regulariser to avoid overfitting. This functional is convex and can be solved using different techniques (Wallach (2002)). The inference phase can be done using a Viterbi-like algorithm over the labels in the graph.

Bayesian Conditional Random Fields (B-CRFs) (Qi et al. (2005)) have been recently proposed, in which the posterior over the weight (1.2) is approximated by a Gaussian using the Power Expectation Propagation (EP) algorithm (Minka and Lafferty (2002)). Once the posterior has been estimated, predictions can be also made using an EP algorithm that considers an independent distribution for each label.

CRFs were initially proposed to solve the multi-label problem using a known set of features ($\boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y}_n)$). Its functional in (1.7) fulfils the conditions of the Representer theorem in Schölkopf and Smola (2001) (which is a generalization of the Representer theorem originally proposed by Kimeldorf and Wahba (1971)). Therefore we can represent the optimal solution as a linear combination of the training examples:

$$\mathbf{w}_t = \sum_{n'=1}^{N} \sum_{\mathbf{y}'_t} \beta^t_{n',\mathbf{y}'_t} \boldsymbol{\phi}_t(\mathbf{x}_{n'}, \mathbf{y}'_t) \qquad \forall t \tag{1.9}$$

If we define the kernel for each feature in each clique as: $\kappa_t(\mathbf{x}_{n'}, \mathbf{y}_{n't}, \mathbf{x}_n, \mathbf{y}_{nt}) = \boldsymbol{\phi}_t^\top(\mathbf{x}_{n'}, \mathbf{y}_{n't})\boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt})$, we can still obtain the MAP estimate in terms of β's by solving (1.7). In this case the output of the classifier can be written as:

$$o(\mathbf{x}_n, \mathbf{y}) = \sum_{t=1}^{T} \sum_{n'=1}^{N} \sum_{\mathbf{y}'_t} \beta^t_{n',\mathbf{y}'_t} \kappa_t(\mathbf{x}_{n'}, \mathbf{y}'_t, \mathbf{x}_n, \mathbf{y}_t) \tag{1.10}$$

which is the standard kernel formulation for multi-class problems. $o(\mathbf{x}_n, \mathbf{y})$ is computed as a linear combination of the kernels involving all the inputs in the training set with every possible labelling of the outputs.

The number of β grows as $N\prod_{t=1}^{T} q^{|\mathbf{y}_t|}$, where $|\mathbf{y}_t|$ indicates the number of nodes in the t-th clique. For a chain (or tree-like structure) the number of β is NTq^2, which is linear in the number of training samples and cliques and quadratic in the possible labellings in each node. If we had not used the graph to simplify the dependencies between the labels, the output would be $o(\mathbf{x}_n, \mathbf{y}) = \sum_{n'=1}^{N} \sum_{\mathbf{y}'} \beta_{n',\mathbf{y}'} \kappa(\mathbf{x}_{n'}, \mathbf{y}', \mathbf{x}_n, \mathbf{y})$ and the number of β's would grow as Nq^L, which increases exponentially with the length of the label vector.

Using the Representer theorem and the kernel trick to solve CRFs was independently proposed by Lafferty et al. (2004) and Altun et al. (2004a). We refer to the general approach as Kernel Conditional Random Fields (K-CRFs). In both of these papers the authors propose to simplify the solution by forcing (in a smart and controlled way) that some β should be zero at the solution. The runtime complexity to infer the label of a new input sequence does not grow with the number of training samples. Otherwise all β are nonzero due to the applied loss-function (logistic regression). Another option to get a sparse solution in terms of the β is to change the loss-function by a hinge-loss, which is presented in the following section.

1.2.1 Support Vector Machines

Once we have described the optimization of CRFs as the minimization in (1.7), the comparison with SVMs is straightforward, as we can substitute the logistic regression loss-function by any other, such as the hinge-loss used by SVMs.

There are two alternative formulation for multi-class SVMs: Weston and Watkins (1998) and Crammer and Singer (2001). The difference lies in how they penalised the training errors. In Weston and Watkins (1998), the M-SVM penalises any possible labelling that provides a larger output than the true labelling. While in Crammer and Singer (2001), the M-SVM only penalises the largest incorrect labelling, if it is greater than the true labelling. For this problem, in which the number of possible labellings grows exponentially, the formulation by Weston and Watkins (1998) can result in an exponential growth in the number of nonzero support vectors and therefore it is more advisable to use the formulation by Crammer and Singer (2001).

The M-SVM formulation by Crammer and Singer (2001) can be represented as an unconstrained optimisation problem:

$$\min_{\mathbf{w}} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \left[\max_{\mathbf{y}} \left(M_{\mathbf{y}_n,\mathbf{y}} + o(\mathbf{x}_n, \mathbf{y}) - o(\mathbf{x}_n, \mathbf{y}_n) \right) \right]_+ \right\}$$

where $M_{\mathbf{y}_n,\mathbf{y}}$ is the margin that depends on the true labelling and the labelling that it is being compared against and $[u]_+ = \max(0, u)$ is the hinge-loss. This functional is equivalent to (1.7) replacing the logistic regression loss-function by the SVM's hinge-loss.

This formulation can be expressed in the more standard constrained optimization setting, in which we need to optimise:

$$\min_{\mathbf{w},\xi_n} \left\{ \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{n=1}^{N} \xi_n \right\} \tag{1.11}$$

subject to:

$$o(\mathbf{x}_n, \mathbf{y}_n) - o(\mathbf{x}_n, \mathbf{y}) \geq M_{\mathbf{y}_n,\mathbf{y}} - \xi_n \qquad \forall n, \forall \mathbf{y} \tag{1.12}$$

where $M_{\mathbf{y}_n,\mathbf{y}_n} = 0$ to ensure that $\xi_n \geq 0$. The number of constraints grows exponentially in L (q^L) but as there is only one ξ_n per training sample there are only few active constraints for each sample, typically none or two. Thus the growth of the complexity (nonzero Lagrange multipliers) are not exponential in L. This formulation is equivalent to the Hidden Markov Support Vector Machine (HM-SVM) proposed in Altun et al. (2003) with small variations on how the margin is imposed and the slack variable ξ_n is penalised.

Finally, there has been an independent formulation of this solution, known as Max Margin Markov networks (M^3 nets) by Taskar et al. (2004), in which the above formulation is simplified, not needing exponentially many Lagrange multipliers

to solve it. We believe the easiest way to present M^3 nets is working from the Lagrangian of (1.11)

$$\begin{aligned} L(\mathbf{w}, \xi_n, \alpha_{n,\mathbf{y}}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ - \sum_{n=1}^{N}\sum_{\mathbf{y}}\alpha_{n,\mathbf{y}}\Big(\sum_{t=1}^{T}\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt}) - \sum_{t=1}^{T}\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_t) \\ - \sum_{\ell=1}^{L}[1 - \delta(y_{n\ell}, y_\ell)] + \xi_n\Big) \end{aligned} \tag{1.13}$$

in which we have substituted $o(\mathbf{x}_n, \mathbf{y})$ by its definition in (1.8) and we have defined the margin per node to be the Hamming distance between the true and compared labels as proposed by Taskar et al. (2004). Now for each training sample and each configuration in every clique, we define:

$$\beta^t_{n,\mathbf{y}_t} = \sum_{\mathbf{y}\sim\mathbf{y}_t}\alpha_{n,\mathbf{y}} \qquad \forall n, \forall t, \forall \mathbf{y}_t \tag{1.14}$$

where the sum runs over all possible labelling of $\mathbf{y}$ with the labels in the t-th clique is fixed at $\mathbf{y}_t$. And for each configuration in every node, we define:

$$\beta^\ell_{n,y_\ell} = \sum_{\mathbf{y}\sim y_\ell}\alpha_{n,\mathbf{y}} \qquad \forall n, \forall \ell, \forall y_\ell \tag{1.15}$$

Thus we can rewrite the Lagrange functional (1.13) in terms of $\beta^t_{n,\mathbf{y}_t}$ and β^ℓ_{n,y_ℓ} as follows:

$$\begin{aligned} L(\mathbf{w}, \xi_n, \beta^t_{n,\mathbf{y}_t}, \beta^\ell_{n,y_\ell}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n \\ - \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{\mathbf{y}_t}\beta^t_{n,\mathbf{y}_t}\left(\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt}) - \mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_t) + \xi_n\right) \\ + \sum_{n=1}^{N}\sum_{\ell=1}^{L}\sum_{y_\ell}\beta^\ell_{n,y_\ell}[1 - \delta(y_{n\ell}, y_\ell)] \end{aligned} \tag{1.16}$$

in which we can see that the sum over $\mathbf{y}$ (q^L terms) has been replaced by sums over $\mathbf{y}_t$ (q^2 terms) and over y_ℓ (q terms). We have as many $\beta^t_{n,\mathbf{y}_t}$ as we did for the K-CRFs and the β^ℓ_{n,y_ℓ} are significantly fewer, reducing the exponentially many Lagrange multipliers ($\alpha_{n,\mathbf{y}}$) in (1.13).

Note that the β are not independent from each other and when solving the Lagrange functional we have to impose an additional constraint, besides the β being positive.

$$\beta^\ell_{n,y_\ell} = \sum_{\mathbf{y}_t\sim y_\ell}\beta^t_{n,\mathbf{y}_t} \tag{1.17}$$

This constraint is necessary to ensure that the α can be recovered from the β and that we are obtaining the same solution as in the HM-SVM. This constraint must hold for all the samples, for all the nodes, for all the cliques that contain the node ℓ and for all possible labellings in the node.

1.2.2 Summary

In this section we have presented the family of CRF algorithms. We have discussed CRF (Lafferty et al. (2001)), B-CRF (Qi et al. (2005)), K-CRF (Altun et al. (2004a,b); Lafferty et al. (2004)), HM-SVM (Altun et al. (2003)) and M^3 nets (Taskar et al. (2004)). In all these algorithms an exponentially growing structured multiclass learning problem is simplified by imposing a graphical model over the output space. This simplification allows solving the multi-classification tractably both in the needed computational power and in the training sample size. In Table 1.1, we classify all this procedures according to their relevant properties. The main difference HM-SVM and M^3 nets lies within the optimization procedure, as explained in the previous section (it does not show up in the table).

Table 1.1 Comparisons of CFR-like algorithms.

	Probabilistic output	Kernels	loss-function
CRF	No	No	Logistic regression
B-CRF	Yes	No	Logistic regression
K-CRF	No	Yes	Logistic regression
HM-SVM	No	Yes	Hinge-loss
M^3 nets	No	Yes	Hinge-loss

1.3 Conditional Graphical Models

The CRF-like algorithms can be represented in a general form by the following convex optimization problem:

$$\min_{\mathbf{w}} \frac{1}{2} \sum_{t=1}^{T} ||\mathbf{w}_t||^2 + C \sum_{n=1}^{N} L\left(\sum_{t=1}^{T} \mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_t) \right) \tag{1.18}$$

where $L(\cdot)$ represent the loss-function and we have replaced $\mathbf{w}$ by $\mathbf{w} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_T]^\top$. The used of a logistic regression loss-function leads us to (1.7) and the use of the hinge-loss to (1.11). Any other loss-function gives rise to other CRF-like algorithm, i.e. LS-CRF with a quadratic loss-function.

In our proposal, (Kernel) Conditional Graphical Models (CGM), we upper bound the CRF loss-function to obtain an optimization functional that it is significantly

easier to optimise and can be addressed using any multiclass learning tool. The idea behind CGM is identical to the one used to solve binary classification problems. In binary classification, the 0–1 loss, which is non-convex and non-differentiable, is replaced by a convex loss-function (square loss, hinge loss, logistic regression, ...) that upper bounds the 0–1 loss, to ensure that the paid penalty is at least as large as the 0–1 loss. The change of the loss-function allows solving a simpler optimization problem and we can concentrate on other tasks, as defining nonlinear classifiers or obtaining the relevant features for classification.

We propose to upper bound (1.18), using Jensen's inequality over the loss-function, to build the CGM optimization functional:

$$\min_{\mathbf{w}} \sum_{t=1}^{T} \left\{ \frac{1}{2}||\mathbf{w}_t||^2 + C \sum_{n=1}^{N} L\left(\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_t)\right) \right\} \tag{1.19}$$

Interchanging the loss-function and the sum over the cliques, we obtain an upper bound to (1.18), if the loss-function is convex. Therefore, we are penalising errors at least as much as we did in the original formulation.

CGM function decomposes per clique, so each $\mathbf{w}_t$ is trained independently using the features and outputs corresponding to the t-th clique. This is a major advantage as we do not need to keep track of what it is happening in the rest of the graph lo learn the parameters of each clique. Furthermore the optimisation in each clique:

$$\min_{\mathbf{w}_t} \frac{1}{2}||\mathbf{w}_t||^2 + C \sum_{n=1}^{N} L\left(\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_t)\right) \tag{1.20}$$

is equivalent to a regularised multiclass problem with q^2 labels[1]. We can apply any multi-classification tool to train the model in each clique without needing a custom-made procedure for defining the parameters of the model. CGM opens up the range of problems in which structured machine learning can be applied, as it has a simple training procedure and can be trained for large-scale problems. For example, if we use a standard tool as LibSVM (Lin, 1999), we could train the CGM with up to several thousands of training samples.

To infer the label of new inputs CGMs work as CRF algorithms do. For this phase there is no difference between both procedures and the solution provided by CGMs cannot be read independently for each clique. To infer an output a Viterbi-like algorithm has to be run-over the assumed graphical model to find the most likely output sequence. Because the outputs of each clique have to agree on the labels they assign to each node. Therefore, we cannot compute the output independently for each clique as a given node (shared among different cliques) can present a different labelling for each clique.

1. if each clique only contains 2 labels.

1.3.1 Support Vector Machines

In this section, we compare HM-SVMs and M^3 nets with CGMs with hinge-loss. This comparison helps us draw some useful conclusions about the validity of the proposed algorithm and how the two approaches penalise errors in training. The HM-SVM/M^3 net solves the following constrained optimization problem:

$$\min_{\mathbf{w},\xi_n} \frac{1}{2}\sum_{t=1}^{T}\|\mathbf{w}_t\|^2 + C\sum_{n=1}^{N}\xi_n \tag{1.21}$$

subject to:

$$\sum_{t=1}^{T}\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n,\mathbf{y}_{nt}) - \sum_{t=1}^{T}\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n,\mathbf{y}_t) \geq \sum_{t=1}^{T} M_{\mathbf{y}_n,\mathbf{y}_t} - \xi_n \qquad \forall n, \forall \mathbf{y} \tag{1.22}$$

and CGM with hinge-loss solves:

$$\min_{\mathbf{w},\xi_n} \sum_{t=1}^{T}\left\{\frac{1}{2}\|\mathbf{w}_t\|^2 + C\sum_{n=1}^{N}\xi_{nt}\right\} \tag{1.23}$$

subject to:

$$\mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n,\mathbf{y}_{nt}) - \mathbf{w}_t^\top \boldsymbol{\phi}_t(\mathbf{x}_n,\mathbf{y}_t) \geq M_{\mathbf{y}_{nt},\mathbf{y}_t} - \xi_{nt} \qquad \forall n, \forall t, \forall \mathbf{y}_t \tag{1.24}$$

Both optimization functional (1.21) and (1.23) are identical with the definition of the slacks variables being the only difference. The main difference lies in the linear constraints that are responsible for the obtained solution for both methods.

Comparing constraints (1.24) and (1.22), we can notice the reduction in the number of constraints and therefore the needed runtime complexity during training. Initially it might seem that when using (1.24), we are multiplying the number of constraints by T, the number of cliques, as it divides (1.22) into T different constraints (one per clique). But each one of the constraints in (1.24) only needs to distinguish between $q^{|\mathbf{y}_t|}$ labels in each clique instead the q^L different labelling of each sequence in (1.22). We end up having a significant reduction in the number of constraints[2] in our optimisation formulation for CGMs in (1.23)-(1.24).

As we commented in the previous paragraph the constraint in (1.22) is the sum over all cliques of the constraint in (1.24). Thus the constraints in (1.24) is more restrictive, because it enforces the margin constraint clique by clique instead of over the whole label sequence. This is due to the modification of the loss-function in (1.19), where we changed the loss in CRF-like algorithms by a convex upper bound. This constraint allows us to be more restrictive and more precise, as we only penalise the cliques that are in error. Let us illustrate these properties with two examples.

2. For a tree-like structure the number of constraints drop from Nq^L to NTq^2.

Suppose a labelling $\mathbf{y}$ fulfills the margin requirement for the n training example, but individually one of its cliques does not. In the original M^3 net formulation, this labelling is not penalised and the discriminative information about the clique, which does not fulfil the margin requirement, is lost. But the formulation in (1.23)-(1.24) enforces the margin per clique, so it uses the information in the erroneously classified clique to build the classifier, incorporating its discriminative information into $\mathbf{w}_t$.

The complementary example is also relevant. Suppose a labelling $\mathbf{y}$ does not fulfil the margin requirement because one of its cliques is completely flawed, although all the other cliques are correctly classified with enough margin. In M^3 network and the other CRF methods, this flawed clique forces the whole sequence to be a support vector and it is incorporated into the construction of the classifier of every clique, although it only presents discriminative information for one clique. In the CGM solution, this flawed clique is considered an outlier and is incorporated into the solution of the classifier in that clique. But it does not affect the classifiers in the remaining cliques and it does not force the whole sequence to be a support vector in every clique.

In a way we can see the formulation per clique to be *more restrictive* than the formulation in (1.22), as we can learn locally in a clique from sequences that are globally correctly classified. At the same time it is *more precise*, as it only needs to learn from the cliques in which the errors occur and does not need to incorporate the whole sequence if it does not bring discriminative information for every clique in the graph.

To sum up, we started the motivation for the algorithm by arguing that solving the optimization per clique would provide a huge computational cost reduction and that we might be able to trade-off some accuracy for getting this complexity reduction. We have finally shown that we do not only get this computational cost reduction, but also a more sensible learning procedure that only incorporates those cliques that bring discriminative information for the whole sequence and not complete sequences that might only provide local discriminative information. We believe this approach can provide higher discriminative power than the previous proposed methods with a much simpler learning mechanism and with a significant reduction in the computational complexity. We test these claims experimentally in the next section.

1.4 Experiments

We test the CGM with a handwritten-word recognition task. The dataset was collected by Kassel (1995) and contains around 6877 words of varying length (4 to 15 letters) written by 150 different subjects. Each word is divided into 16×8 binary images for each letter and its corresponding label. Our inputs will be a $16 \times 8L$ binary image and its corresponding label vector will contain L elements

out of q^L possibilities. This data set was preprocessed by Taskar et al. (2004) to test the M^3 network. The dataset was divided in 10 groups for cross-validation.

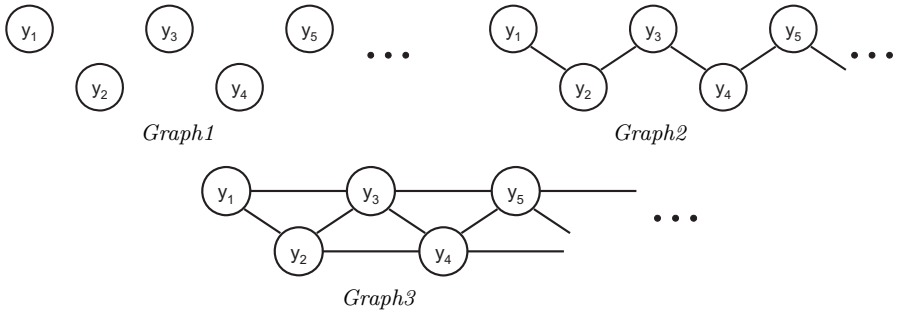

Figure 1.2 We represent the 3 graphical models that will be used in the experiments to test the CGM.

We test the CGM using the 3 graphical models shown in Figure 1.2. The first graph contains no edges and in it we will be training a multi-class model for each letter independently. The second one is a chain, in which we have pairwise interactions between letters, and the third takes into account three-letter interactions. For this experiment, we will use the following feature vector:

$$\boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt}) = [\mathbf{0}, \ldots, \boldsymbol{\psi}(\mathbf{x}_{nt}), \mathbf{0}, \ldots]^\top \tag{1.25}$$

where $\boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt})$ contains $q^{|\mathbf{y}_t|}$ terms and the nonzero element is indexed by the labelling $\mathbf{y}_{nt}$. In this feature, we only consider as inputs the images of the letters in each clique. The kernel of $\boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt})$ is:

$$\kappa_t(\mathbf{x}_{n'}, \mathbf{y}_{n't}, \mathbf{x}_n, \mathbf{y}_{nt}) = \delta(\mathbf{y}_{nt} = \mathbf{y}_{n't}) k(\mathbf{x}_{nt}, \mathbf{x}_{n't})$$

where we have defined $k(\mathbf{x}_{nt}, \mathbf{x}_{n't}) = \boldsymbol{\psi}^\top(\mathbf{x}_{nt})\boldsymbol{\psi}(\mathbf{x}_{n't}) = k(\mathbf{x}_{nt}, \mathbf{x}_{n't}) = \exp(\|\mathbf{x}_{nt} - \mathbf{x}_{n't}\|^2/2/\sigma^2)$. To solve each one of the multiclass SVM we have used the LibSVM code Lin (1999).

We have first solved this experiment using 1 group for training and the other 9 for validation, as done in Taskar et al. (2004), and we have repeated it over the 10 different sets. We have set $C = 5$, $M_{\mathbf{y}_{nt},\mathbf{y}_t} = 1$ and $\sigma = \sqrt{d/16}$, where d is the dimension of $\mathbf{x}_{nt}$. We have reported the letter and word probability of error in Table 1.2.

Table 1.2 We show the mean and standard deviation for 10-fold cross-validation for the 3 graphs with 1 set for training and 9 for validation.

	Graph1	*Graph2*	*Graph3*
Word	0.795±.0012	0.369±.0013	0.195±.0009
Letter	0.271±.0009	0.125±.0011	0.058±.0005

In Taskar et al. (2004) the authors reported an error rate of around 13% in the letter recognition task, for the same partition of the data used in this experiment. This result is directly comparable to the letter recognition for *Graph2*, as they used a chain in their experiments. The proposed CGM can be used with higher connectivity graphs, such as *Graph3*, because we can perform the training independently per clique and the error rate is reduced by over 50%.

We have also computed the cross-validation error in the standard setting, in which 9 groups are used for training and one for validation, to show that we can work with larger training sets. The mean probability of error of letter and word recognition are reported in Table 1.3. For this setting *Graph3* is still better than

Table 1.3 We show the mean and standard deviation for 10-fold cross-validation for the 3 graphs with 9 set for training and 1 for validation.

	Graph1	*Graph2*	*Graph3*
Word	0.528±.0037	0.133±.0015	0.128±.0019
Letter	0.126±.0009	0.031±.0003	0.027±.0004

Graph2, but there the difference is not as significant as it was for shorter training sequences.

The performance of *Graph2* and *Graph3* in the above experiments were significantly higher than those of *Graph1*, because they incorporate some error correcting capabilities, as not all the transitions in the graph are allowed and the different cliques have to agree on the predicted label in the nodes they share. But if we look at the performance clique by clique (individual decisions), *Graph1* presents a lower error rate as its learning problem is much simpler. It only needs to distinguish between q labels, instead of q^2 or q^3, and the training set has the same number of entries with lower input dimension. We propose the following feature to incorporate the individual decisions in *Graph1* with the error correcting capabilities provided by graphs with higher connectivity among its nodes. We will train *Graph2* using the following feature vector:

$$\boldsymbol{\phi}_t(\mathbf{x}_n, \mathbf{y}_{nt}) = [\mathbf{0}, \ldots, \boldsymbol{\psi}(\mathbf{x}_{nt_1}), \ldots \mid \mathbf{0}, \ldots, \boldsymbol{\psi}(\mathbf{x}_{nt_2}), \ldots]^\top$$

which has $2q^2$ elements, twice as many entries as the feature defined in (1.25). The positions of $\boldsymbol{\psi}(\mathbf{x}_{nt_1})$ and $\boldsymbol{\psi}(\mathbf{x}_{nt_2})$ are indexed by $\mathbf{y}_{nt}$. The vectors $\mathbf{x}_{nt_1}$ and $\mathbf{x}_{nt_2}$ are, respectively, the images of the first and second letter in the t^{th} clique.

For this feature, we can describe the weight vector as:

$$\mathbf{w}_t = [\mathbf{w}_{t1} \mid \mathbf{w}_{t2}] = [\mathbf{w}_{t1_1}, \ldots, \mathbf{w}_{t1_{q^2}} \mid \mathbf{w}_{t2_1}, \ldots, \mathbf{w}_{t2_{q^2}}]$$

In each $\mathbf{w}_{t1}$ there are q^2 terms, but in the corresponding part of the feature vector, we will only deal with the image of a letter (q different values). As the same letter can be placed in different positions by the labelling $\mathbf{y}_{nt}$, we will clamp together the $\mathbf{w}_{t1_s}$ (and $\mathbf{w}_{t2_s}$) that multiply the image of the same letter, so we can have a higher performance as each weight vector will be trained with more training examples

(all the letters in each node). But the feature vector will still keep the information about which couple of letters are allowed in each clique. The kernel for this feature will be: $\kappa_t(\mathbf{x}_{n'}, \mathbf{y}_{n't}, \mathbf{x}_n, \mathbf{y}_{nt}) = \delta(\mathbf{y}_{nt} = \mathbf{y}_{n't})[k(\mathbf{x}_{nt_1}, \mathbf{x}_{n't_1}) + k(\mathbf{x}_{nt_2}, \mathbf{x}_{n't_2})]$, where $k(\cdot, \cdot)$ is defined as above.

We have computed the cross-validation error for this feature using the previously defined sets. When a single set is used for training the mean letter recognition error and standard deviation are: 0.097±.0006 (for words: 0.327±.0012). And when we used 9 sets for training and one for validation, the performance results are: 0.025±.0003 (for words: 0.120±.0019). We can see that these results provide lower error rates than the ones reported in Tables 1.2 and 1.3 for *Graph1* and *Graph2*, because we incorporate the error correcting capabilities of *Graph2* and the more precise performance in each individual decision. This feature vector can be extended to the *Graph3*.

1.5 Conclusions and further work

In this paper, we have presented a unified framework that covers the most relevant proposals to solve the multi-label classification problem using graphical models to reduce the exponentially growing complexity with the label length. We have presented a compact notation that can be used to represent Conditional Random Fields (Lafferty et al., 2001), Bayesian CRFs (Qi et al., 2005), Kernel CRFs (Altun et al., 2004a,b; Lafferty et al., 2004) and Maximum Margin Markov networks (Altun et al., 2003; Taskar et al., 2004). This notation is simpler than most of the notation used in those papers, and allows to compare these models and to understand their similar properties. There is a different approach Weston et al. (2002), which uses a kernel over the labels to deal with the complexity of the addressed problem. Although, we have not studied the connection of our framework with this method, as we end up using a kernel over the inputs and labels (1.10), we believe that connection can be made and it is left as further work.

In the second part of the paper, based on the presented framework, we have proposed a new learning algorithm to solve the multi-label problem. The CGM can be solved independently per clique, which is its main difference with the algorithms proposed in the literature. Therefore the CGM is much simpler to solve, so we can use much larger training datasets and it can be applied over more complex graphs. We have also argued, and shown experimentally, that this training per clique is more precise than the training of the sequences as a whole. Because the classification of a new example is based on the individual decisions of each clique (then combine with the graph to ensure the solution is consistent), so if we use all the discriminative information to train each clique we will be able to provide much more accurate answer, when we predict the labels of new samples. We have left as further work the connection between the Conditional Graphical Models and the probabilistic approaches for solving the multi-label problem.

Acknowledgements

Fernando Pérez-Cruz is Supported by the Spanish Ministry of Education Postdoctoral fellowship EX2004-0698.

References

Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *International Conference on Machine Learning*, Washington, USA, 2003.

Y. Altun, A. Smola, and T. Hofmann. Exponential families for conditional random fields. In *Conference on Uncertainty in Artificial Intelligence*, Banff, Canada, 2004a.

Yasemin Altun, Thomas Hofmann, and Alexander J. Smola. Gaussian process classification for segmenting and annotating sequences. In *International Conference on Machine Learning*, Banff, Canada, 2004b.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT*, pages 144–152, Pittsburgh, PA, 1992. ACM Press. URL `http://www.clopinet.com/isabelle/Papers/colt92.ps`.

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

Thomas G. Dietterich. Machine learning for sequential data: A review. In T. Caelli, editor, *In Structural, Syntactic, and Statistical Pattern Recognition; Lecture Notes in Computer Science*, volume 2396, pages 15–30. Springer-Verlag, 2002.

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

G. S. Kimeldorf and G. Wahba. Some results in tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33:82–95, 1971.

S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, Massachusetts, USA, 2001.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *International Conference on Machine Learning*, Banff, Canada, 2004.

Chih-Jen Lin. Formulations of support vector machines: a note from an optimization point of view. Technical report, National Taiwan University, Dept. of Computer Science, 1999. http://www.csie.ntu.edu.tw/~cjlin/.

Thomas P. Minka and John Lafferty. Expectation propagation for the generative aspect model. In *Conference on Uncertainty in Artificial Intelligence*, 2002.

Yuan Qi, Martin Szummer, and Thomas P. Minka. Bayesian conditional random fields. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

B. Schölkopf and A. Smola. *Learning with kernels*. M.I.T. Press, 2001.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

Hanna Wallach. Efficient training of conditional random fields. Master's thesis, Division of Informatics, University of Edinburgh, 2002.

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, Egham, UK, 1998. URL `http://www.dcs.rhbnc.ac.uk/research/compint/areas/comp\_learn/sv/pub/report98-04.ps`.

J. Weston, O. Chapelle, A Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*. MIT Press, Cambridge, MA, 2002.