1 Efficient algorithms for max-margin structured classification

Juho Rousu
Department of Computer Science
University of Helsinki, Finland
juho.rousu@cs.helsinki.fi

Craig Saunders, Sandor Szedmak and John Shawe-Taylor
Electronics and Computer Science
{cjs,ss03v,jst}@ecs.soton.ac.uk

Abstract

We present a general and efficient optimization methodology for max-margin structured classification tasks. The efficiency of the method relies on the interplay of several techniques: formulation of the structured support vector machine or max-margin Markov problem as an optimization problem; marginalization of the dual of the optimization; partial decomposition via a gradient formulation; and finally tight coupling of a maximum likelihood inference algorithm into the optimization algorithm, as opposed to using inference as a working set maintenance mechanism only. The tight coupling also allows fast approximate inference to be used effectively in the learning.

The generality of the method follows from the fact that changing the output structure in essence only changes the inference algorithm, that is, the method can to a large extent be used in a 'plug and play' fashion.

1.1 Introduction

Structured classification methods based on the Conditional Random Field (CRF) model (Lafferty et al., 2001) are proving themselves in various application fields. Recently, techniques inspired by support vector machines for learning the parame-

ters of CRFs (Taskar et al., 2004b; Tsochantaridis et al., 2004; Lafferty et al., 2004) or related graphical models (Altun et al., 2003; Bartlett et al., 2004) have emerged.

In this chapter, we present a general and efficient approach for max-margin learning of CRF parameters when the CRF takes the form of a hypergraph. The method benefits from many of the above works and also from the related research on exponential families and their inference methods (Wainwright and Jordan, 2003; Wainwright et al., 2003).

The main contribution of this chapter is to show that the max-margin optimization relying on the marginal dual formulation (c.f. Taskar et al. 2004b) can be made efficient without simplifying the problem or settling for approximations. Key ingredients are feature and loss representations that adhere to the hypergraph structure, a partial decomposition via gradient-based optimization and, finally, tight coupling of the inference algorithm to conditional gradient optimization which avoids an explicit description of the constraints.

Outline of the Chapter

The structure of this chapter is the following. In Section 1.2 we present the classification framework, review loss functions and derive a quadratic optimization problem for finding the maximum margin model parameters. In Section 1.3 we present an efficient learning algorithm relying on a decomposition of the problem into single training example subproblems, and then conducting iterative conditional gradient ascent in marginal dual variable subspaces corresponding to single training examples. We show that search directions of the conditional gradient method can be efficiently found by solving an inference problem on the hypergraph. We demonstrate the algorithm's behaviour in Section 1.4 in a hierarchical classification task. We conclude the chapter with a discussion in Section 1.5.

Notation

For a cartesian product $S = S_1 \times \cdots \times S_k$ of sets, and $p = \{p_1, \ldots, p_l\} \subset \{1, \ldots, k\}$ an index set, we use the shorthand $S_p = S_{p_1} \times \cdots \times S_{p_l}$ to denote the restriction of S to the index set p. Similarly, we use $s = (s_1, \ldots, s_k) \in S$ and $s_p = (s_{p_1}, \ldots, s_{p_l})$ to denote the members of the set and their restrictions to p, respectively. When p is clear from the context, we sometimes drop the subscripts and write s instead of s_p.

In this chapter we need to refer extensively to vectors with a nested block structure; for example a marginal dual vector $\mu = (\mu_i)_{i=1}^m$, where $\mu_i = (\mu_{ie})_{e \in E}$ (E will be a set of hyperedges introduced below with $\mathcal{Y}_e$ the set of values over the hyperedge node e) and $\mu_{ie} = (\mu_{ie}(\mathbf{u}))_{\mathbf{u} \in \mathcal{Y}_e}$. To denote individual items in these vectors we may use multi-indices $\mu(i, e, \mathbf{u}) = \mu_i(e, \mathbf{u}) = \mu_{ie}(\mathbf{u})$. With no confusion we sometimes transpose the vectors so that $\mu_e = (\mu_{ie})_{i=1}^m$; the subscript will make clear which vector is meant.

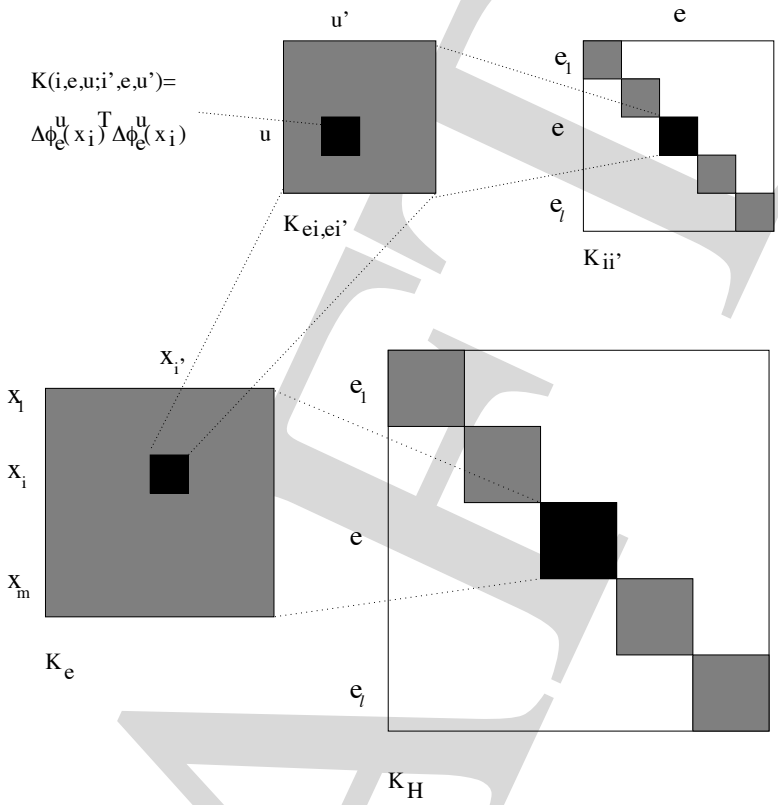

Figure 1.1 Illustration of the nested block structure of the kernels applied in this chapter. The kernel block $K_{ie,i'e'}$ (top, left) occurs as a block both in the edge-kernel K_e (top, right) and the $K_{ii'}$ kernel (bottom, left). The full marginalized kernel K_H (bottom, right) is composed of K_e blocks on the diagonal. Grey color indicates potentially non-zero values.

For matrices with nested block structure similar conventions are used (Figure 1.1): the notation $K = (K_{ii'})_{i,i'=1}^m$, $K_{ii'} = (K_{ie,i'e'})_{e,e' \in E}$ and

$$K_{ie,i'e'} = (K_{ie,i'e'}(\mathbf{u}, \mathbf{u}'))_{\mathbf{u} \in \mathcal{Y}_e, \mathbf{u}' \in \mathcal{Y}_{e'}}$$

is used to refer to the blocks, and multi-indices

$$K(i, e, \mathbf{u}; i', e', \mathbf{u}') = K_{ii'}(e, \mathbf{u}; e', \mathbf{u}') = K_{ie,i'e'}(\mathbf{u}, \mathbf{u}')$$

are used to access the individual items. Furthermore, different permutations will be used $K_{ee'} = (K_{ie,ie'})_{i=1}^m$ that will again be clear from the context. For diagonal blocks the double subscript is sometimes replaced with a single one $K_i = K_{ii}, K_e = K_{ee}$.

When referring to elements of the training set, when no confusion arises we sometimes use the shorthand i in place of $\mathbf{x}_i$ or $\mathbf{y}_i$, e.g. $\alpha(i, \mathbf{y})$ instead of $\alpha(\mathbf{x}_i, \mathbf{y})$, and $\ell_e(i, \mathbf{u}_e)$ in place of $\ell_e(y_{ie}, \mathbf{u}_e)$.

1.2 Structured Classification Model

In this chapter we use the conditional random field (CRF) model for structured classification. We first briefly introduce the setting.

We consider data from a domain $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{X}$ is a set and $\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_k$ is a cartesian product of finite sets $\mathcal{Y}_j = \{1, \ldots, |\mathcal{Y}_j|\}, j = 1, \ldots, k$. A vector $\mathbf{y} = (y_1, \ldots, y_k) \in \mathcal{Y}$ is called the *multilabel* and the components y_j are called *microlabels*.

We assume that a training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$ has been given, consisting of training examples $(\mathbf{x}_i, \mathbf{y}_i)$ of a training pattern $\mathbf{x}_i$ and multilabel $\mathbf{y}_i$. A pair $(\mathbf{x}_i, \mathbf{y})$, where $\mathbf{x}_i$ is a training pattern and $\mathbf{y} \in \mathcal{Y}$ is arbitrary, is called a *pseudo-example* in order to denote the fact that the output may or may not have been generated by the distribution generating the training examples.

The multilabels conform to a given fixed hypergraph structure $H = (V, E)$ that consists of nodes $V = \{1, \ldots, k\}$ and hyperedges $E = \{e_1, \ldots, e_r\}$, with $e_h \subset V$ for $h = 1, \ldots, r$. For some or all nodes $v \in V$, the singleton hyperedge $\{v\}$ may be contained in E. The existence of a hyperedge $e = \{v_1, \ldots, v_l\} \in E$ indicates that there is a potential statistical dependency between the microlabels $y_{v_1}, \ldots, y_{v_l}$.

In the conditional random field defined on H, the probability of a multilabel $\mathbf{y}$ given a training pattern $\mathbf{x}$ is determined by the product of hyperedge potentials

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \prod_{e \in E} \varphi_e(\mathbf{x}, \mathbf{y}_e, \mathbf{w})$$

where $Z(\mathbf{x}, \mathbf{w}) = \sum_{\mathbf{u} \in \mathcal{Y}} \prod_{e \in E} \varphi(\mathbf{x}, \mathbf{u}_e, \mathbf{w})$ is the normalization factor also referred to as the partition function. We concentrate on the case where the potentials are given by an exponential family

$$\varphi_e(\mathbf{x}, \mathbf{y}_e, \mathbf{w}) = \exp\left(\mathbf{w}_e^T \phi_e(\mathbf{x}, \mathbf{y})\right),$$

where $\mathbf{w}_e$ is the vector of the appropriate block entries for that hyperedge. This choice gives us a log-linear model

$$\log P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \phi(\mathbf{x}, \mathbf{y}) - \log Z(\mathbf{x}, \mathbf{w}). \tag{1.1}$$

1.2.1 Max margin learning

Typically in learning probabilistic models one aims to learn maximum likelihood parameters, which in the exponential CRF amounts to solving

$$\operatorname*{argmax}_{\mathbf{w}} \log \left(\prod_{i=1}^m P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) \right) = \operatorname*{argmax}_{\mathbf{w}} \sum_{i=1}^m \left[\mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) - \log Z(\mathbf{x}_i, \mathbf{w}) \right].$$

This estimation problem is hampered by the need to compute the logarithm of the partition function Z. For a general graph this problem is hard to solve. Approximation methods for its computation are the subject of active research (c.f.

Wainwright et al. 2005). Also in the absence of regularization the max-likelihood model is likely to suffer from overfitting

An alternative formulation (c.f Altun et al. 2003; Taskar et al. 2004b), inspired by support vector machines, is to estimate parameters that in some sense maximize the ratio

$$\frac{P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})}{P(\mathbf{y}|\mathbf{x}_i, \mathbf{w})}$$

between the probability of the correct labelling $\mathbf{y}_i$ and the closest competing incorrect labelling $\mathbf{y}$. With the exponential family, the problem translates to the problem of maximizing the minimum linear margin

$$\mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^T \phi(\mathbf{x}_i, \mathbf{y})$$

in the log-space.

In classical SVM learning the required margin between an example $(\mathbf{x}, y)$ and an (incorrect) pseudo-example $(x, -y)$ is taken to be constant. In structured classification, it is proposed to grade the required margin based on the (structured) loss $\ell(\mathbf{y}, \mathbf{y}_i)$, so that the margin requirement is a nondecreasing function $\gamma(\mathbf{y}, \mathbf{y}_i)$ of the loss of corresponding pseudo-examples. We consider the detailed loss functions and the margin scaling below.

Using the canonical hyperplane representation (c.f. Cristianini and Shawe-Taylor 2000) the problem can be exactly stated as the following minimization problem:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}}\ & \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t. } & \mathbf{w}^T \Delta\phi(\mathbf{x}_i, \mathbf{y}) \geq \gamma(\mathbf{y}_i, \mathbf{y}), \text{ for all } i \text{ and } \mathbf{y}, \end{aligned} \tag{1.2}$$

where $\Delta\phi(\mathbf{x}_i, \mathbf{y}) = \phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \mathbf{y})$ and $\gamma(\mathbf{y}_i, \mathbf{y})$ is the margin required from the pseudo-example $(\mathbf{x}_i, \mathbf{y})$. Note that in the problem (1.2) the need to compute the log-partition function has been avoided. Also, margin-maximization provides resistance against overfitting.

As with SVMs we are not usually able to find a model satisfying margin constraints exactly, and so it is necessary to add slack variables ξ_i to allow examples to deviate from the margin boundary. Altogether this results in the following optimization problem

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}}\ & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i \\ \text{s.t. } & \mathbf{w}^T \Delta\phi(\mathbf{x}_i, \mathbf{y}) \geq \gamma(\mathbf{y}_i, \mathbf{y}) - \xi_i, \text{ for all } i \text{ and } \mathbf{y}. \end{aligned} \tag{1.3}$$

For many feature representations such as for example strings, images or graphs the problem can be very high-dimensional making it advisable to seek a dual representation:

$$\underset{\alpha \geq 0}{\text{maximize}} \;\; \alpha^T \gamma - \frac{1}{2} \alpha^T K \alpha, \text{ s.t.} \sum_{\mathbf{y}} \alpha(i, \mathbf{y}) \leq C, \forall i, \mathbf{y}, \tag{1.4}$$

where $K = \Delta\Phi^T \Delta\Phi$ is the *joint* kernel matrix for *pseudo-examples* $(\mathbf{x}_i, \mathbf{y})$ and $\gamma = (\gamma(\mathbf{y}_i, \mathbf{y}))_{i,\mathbf{y}}$ encodes the margin requirements for each $(\mathbf{x}_i, \mathbf{y})$.

This approach makes it possible to avoid working with explicit feature vectors. However, in the dual problem there are exponentially many dual variables $\alpha(i, \mathbf{y})$, one for each pseudo-example. There are a few main routes by which the exponential complexity can be circumvented:

- Dual working set methods where the constraint set is grown incrementally by adding the worst margin violator

$$\underset{i,\mathbf{y}}{\operatorname{argmin}} \, \mathbf{w}^T \Delta\phi(\mathbf{x}_i, \mathbf{y}) - \gamma(\mathbf{y}_i, \mathbf{y})$$

to the dual problem. One can guarantee an approximate solution with a polynomial number of support vectors using this approach (Altun et al., 2003; Tsochantaridis et al., 2004).

- Primal methods where the solution of the above inference problem is integrated into the primal optimization problem, hence avoiding the need to write down the exponential-sized constraint set (Taskar et al., 2004a).

- Marginal dual methods, in which the problem is translated to a polynomially-sized form by considering the marginals of the dual variables (Taskar et al., 2004b; Bartlett et al., 2004).

The methodology presented in this chapter belongs to the third category.

1.2.2 Loss functions

We assume that associated with the set $\mathcal{Y}$ is a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_+$ that associates for each pair $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$ a non-negative loss $\ell(\mathbf{y}, \mathbf{y}')$. There are many ways to define loss functions for a multilabel classification setting, and it will depend on the application which loss function is the most suitable. Nonetheless a few general guidelines can be set. The loss function should obviously fulfil some basic conditions: $\ell(\mathbf{y}, \mathbf{y}') = 0$ if and only if $\mathbf{y} = \mathbf{y}'$, $\ell(\mathbf{y}, \mathbf{y}')$ is maximal when $y_j \neq y'_j$ for every $1 \leq j \leq k$, and ℓ should be monotonically non-decreasing with respect to inclusion of the sets of incorrect microlabels. These conditions are, for example, satisfied by the *zero-one* loss

$$\ell_{0/1}(\mathbf{y}, \mathbf{y}') = [\mathbf{y} \neq \mathbf{y}'].$$

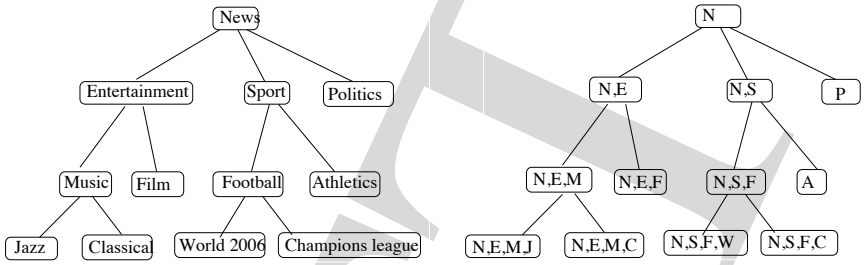

Figure 1.2 A classification hierarchy represented as a tree (left) and a hypergraph consisting of partial paths of the tree as hyperedges (right).

For structured classification, another useful property is that the loss decomposes so that it can be expressed as a combination of the losses of the hyperedges. This is beneficial for algorithmic efficiency and it is not a significant restriction: the need to express the loss of some set of variables $g \subset V$ implies a statistical dependency between those variables. If this dependency is not preempted by the dependencies of the hyperedges that intersect with g, then g really should be a hyperedge in H. We therefore restrict ourselves to losses that are defined as weighted combinations of hyperedge losses

$$\ell(\mathbf{y}, \mathbf{y}') = \sum_{e \in E} \ell_e(\mathbf{y}_e, \mathbf{y}'_e).$$

The simplest way of defining a loss of this type is to take $\ell_e(\mathbf{y}_e, \mathbf{y}'_e) = [\mathbf{y}_e \neq \mathbf{y}'_e]$, in which case the overall loss is the number of incorrectly predicted hyperedges. If all singleton hyperedges $\{v\}, v \in V$ are in E, defining $\ell_e(\mathbf{y}_e, \mathbf{y}'_e) = 0$ for all non-singleton hyperedges ($|e| > 1$) gives us the Hamming loss:

$$\ell_\Delta(\mathbf{y}, \mathbf{y}') = \sum_{\{v\} \in E} [y_v \neq y'_v],$$

which penalizes the errors made in vertices individually, but does not take into account the structure implied by the non-singleton hyperedges; this is also referred to as the microlabel loss.

Example 1 *For particular structures, one can define more elaborate losses. For example, for hierarchical classification (c.f. Figure 1.2), predicting the parent microlabel correctly is typically more important than predicting the child correctly, as the child may deal with some detailed concept that the user may not be interested in; for example whether a document was about* CHAMPIONS LEAGUE *football or not may not be relevant to a person who is interested in* FOOTBALL *in general. Also, from the learner's point of view, if the parent class has already been predicted incorrectly, we don't want to penalize the mistake at the child. Loss functions with these kinds of properties can be defined in more than one way. If one represents the*

classification hierarchy as a set of nodes and directed edges $i \mapsto j$, one may define an edge *loss*

$$\ell_{\tilde{H}}(\mathbf{y}, \mathbf{y}') = \sum_{e=\{i \mapsto j\} \in E} c_e [y_j \neq y'_j \ \& \ y_i = y'_i],$$

that penalizes a mistake in a child only if the label of the parent was correct. If, on the other hand, the hierarchy is represented as a hypertree with the hyperedges given by the partial paths $p = (v_1, \ldots, v_k)$ where v_i is the parent of v_{i+1}, and v_k is either a leaf or an internal node, one can define a path *loss*

$$\ell_H(\mathbf{y}, \mathbf{y}') = \sum_{p=(v_1,\ldots,v_k) \in E} c_p [y_k \neq y'_k \ \& \ \ (y_h = y'_h \forall h \in anc(k))\,],$$

where $anc(j)$ denotes the set of ancestors h of node j.

1.2.3 Scaling the margin requirement

As discussed above it is useful to enforce larger margins for pseudo-examples with high loss and vice versa. A natural way to incorporate this is to define the required margin to be a function $\gamma(y_i, y)$ that is monotonically increasing with respect to the loss. Examples of margin scaling include:

- Linear scaling (Taskar et al., 2004b,a): $\gamma(\mathbf{y}_i, \mathbf{y}) = \ell(\mathbf{y}_i, \mathbf{y})$. The benefit of linear scaling is that any decomposability properties of the loss function are translated to the margin requirement. The potential drawback is the fact that some capacity of the learning machine is wasted in tuning the margins of high-loss pseudo-examples; to some degree the classification problem is turned into an ordinal regression problem.
- Inverse scaling (Tsochantaridis et al., 2004): $\gamma(\mathbf{y}_i, \mathbf{y}) - \xi_i = 1 - \xi_i / \ell(\mathbf{y}_i, \mathbf{y})$. Here the slack is downscaled so high loss examples receive smaller slacks. Inverse scaling is strictly concave with respect to the loss, which makes the margin requirement loss-sensitive in the low loss regime but less sensitive in the high-loss regime. However, the margin requirement is in general not decomposable even if the loss function ℓ is.

Avoiding the tradeoff between retaining decomposability and the apparent waste of capacity in enforcing high margins in the high loss regime seems difficult. In this chapter, we follow the first approach, as we prefer to retain the possibility of decomposing the learning problem, hence making it possible to tackle larger structures and training sets.

1.2.4 Feature representation

In a learning problem, there are two general types of features that can be distinguished:

Global features are given by the feature map $\phi^x : \mathcal{X} \mapsto \mathcal{F}_x$. They are not tied to a particular vertex or hyperedge but represent the structured object as a whole. For example, the bag-of-words of a document is not tied to a single class of documents in a hierarchy, but a given word can relate to different classes with different importance.

Local features, are given by a feature map $\phi_e^x : \mathcal{X} \mapsto \mathcal{F}_{xe}$ tied to a particular vertex or hyperedge of the structure. For example, for sequence annotation based on a Hidden Markov Model each position in the sequence is tied to a set of attributes, e.g. the type and biochemical properties of the nucleotide or amino acid, location specific sequence homology, and so on.

When the features are used in structured classification on a hypergraph H, the features need to be associated with the labellings of the hypergraph. This is done via constructing a joint feature map $\phi : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{F}_{xy}$. There are important design choices to be made in how the hypergraph structure should be reflected in the feature representation.

Orthogonal feature representation is defined as

$$\phi(\mathbf{x}, \mathbf{y}) = (\phi_e(\mathbf{x}, \mathbf{y}_e))_{e \in E},$$

so that there is a block for each hyperedge, that in turn is divided into blocks for specific hyperedge-labelling pairs $(e, \mathbf{u}_e)$, i.e. $\phi_e(\mathbf{x}, \mathbf{y}_e) = (\phi_e^{\mathbf{u}_e}(\mathbf{x}, \mathbf{y}_e))_{\mathbf{u}_e \in \mathcal{Y}_e}$.

The map ϕ_e^u should both incorporate the x-features relevant to the hyperedge and encode the dependency on the labelling of the hyperedge. A simple choice is to define

$$\phi_e^{\mathbf{u}_e}(\mathbf{x}, \mathbf{y}_e) = [\mathbf{u}_e = \mathbf{y}_e] \left(\phi^x(\mathbf{x}), \phi_e^x(\mathbf{x})\right)^T$$

that incorporates both the global and local features if the hyperedge is labeled $\mathbf{y}_e = \mathbf{u}_e$, and a zero vector otherwise. Intuitively, the features are turned 'on' only for the particular labelling of the hyperedge that is consistent with $\mathbf{y}$.

Note that in this representation, global features get weighted in a context-dependent manner: some features may be more important in labelling one hyperedge than another. Thus, the global features will be 'localized' by the learning algorithm. The size of the feature vectors grows linearly in the number of hyperedges, which requires careful implementation if solving the primal optimization problem (1.3) rather than the dual.

The kernel induced by the above feature map decomposes as

$$K(\mathbf{x}, \mathbf{y}; \mathbf{x}', \mathbf{y}') = \sum_{e \in E} \phi_e(\mathbf{x}, \mathbf{y}_e)^T \phi_e(\mathbf{x}', \mathbf{y}'_e) = \sum_{e \in E} K_e(\mathbf{x}, \mathbf{y}_e; \mathbf{x}', \mathbf{y}'_e), \tag{1.5}$$

which means that there is no crosstalk between the hyperedges:

$$\phi_e(\mathbf{x}, \mathbf{y}_e)^T \phi_{e'}(\mathbf{x}, y_{e'}) = 0$$

if $e \neq e'$, hence the name 'orthogonal'. The number of terms in the sum when calculating the kernel obviously scales linearly in the number of hyperedges.

Additive feature representation is defined as

$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{e \in E} \sum_{u \in \mathcal{Y}_e} [\mathbf{y}_e = u] \left(\phi^x(\mathbf{x}), \phi^x_e(\mathbf{x})\right)^T,$$

thus the features of the hyperedges are added together.

This feature representation differs from the orthogonal one in a few important respects. First, the dimension of the feature vector is independent of the size of the hypergraph, thus optimization in the primal representation (1.3) is more feasible for large structures. Second, as there are no hyperedge specific feature weights, the existence of local features is mandatory in this approach, otherwise the hypergraph structure is not reflected in the computed output. Third, the kernel

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}; \mathbf{x}', \mathbf{y}') &= \left(\sum_e \phi_e(\mathbf{x}, \mathbf{y})\right)^T \left(\sum_e \phi_e(\mathbf{x}', \mathbf{y}')\right) \\ &= \sum_{e,e'} \phi_e(\mathbf{x}, \mathbf{y}_e)^T \phi_{e'}(\mathbf{x}, \mathbf{y}'_e) = \sum_{e,e'} K_{ee'}(\mathbf{x}, \mathbf{y}_e; \mathbf{x}', \mathbf{y}'_e) \end{aligned}$$

induced by this representation typically has non-zero blocks $K_{ee'} \neq 0$, for $e \neq e'$, reflecting cross-talk between hyperedges. There are two consequences of this fact. First, the kernel does not exhibit the sparsity that is implied by the hypergraph, thus it creates the possibility of overfitting. Second, the complexity of the kernel will grow quadratically in the size of the hypergraph rather than linearly as is the case of orthogonal features. This is another reason why a primal optimization approach for this representation might be more justified than a dual approach.

In the sequel, we describe a method that relies on the orthogonal feature representation that will give us a dual formulation with complexity growing linearly in the number of hyperedges in H. The kernel defined by the feature vectors, denoted by

$$K^x(\mathbf{x}, \mathbf{x}') = \phi^x(\mathbf{x})^T \phi^x(\mathbf{x}'),$$

is referred to as the x-kernel, while $K(\mathbf{x}, \mathbf{y}; \mathbf{x}, \mathbf{y}')$ is referred to as the *joint kernel*.

1.2.5 Marginal dual polytope

The feasible set of the dual problem (1.4) is a cartesian product $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$ of identical closed polytopes

$$\mathcal{A}_i = \{\alpha_i \in \mathbb{R}^{|\mathcal{Y}|} \mid \alpha_i \geq 0, ||\alpha_i||_1 \leq C\}, \tag{1.6}$$

with a vertex set $V_i = \{0, Ce_1, \ldots, Ce_{|\mathcal{Y}|}\} \subset \mathbb{R}^{|\mathcal{Y}|}$ consisting of the zero vector and the unit vectors of $\mathbb{R}^{|\mathcal{Y}|}$, scaled by C. The vertex set of $\mathcal{A}$ is the cartesian product $V_1 \times \cdots \times V_m$.

The dimension of the set $\mathcal{A}$, $d_{\mathcal{A}} = m|\mathcal{Y}|$ is exponential in the length of the multilabel vectors. This means that optimizing directly over the the set $\mathcal{A}$ is not tractable. Fortunately by utilizing the structure of H, the set $\mathcal{A}$ can be mapped to a set $\mathcal{M}$ of polynomial dimension, called the marginal polytope of H, where optimization becomes more tractable (c.f Taskar et al. 2004b).

Given a subset $p \subset \{1, \ldots, k\}$ of vertices, and an associated labelling y_p, the marginal of $\alpha(i, \mathbf{y})$ for the pair (p, y_p) is given by

$$\mu(i, p, y_p) = \sum_{\mathbf{u} \in \mathcal{Y}} [y_p = u_p] \alpha(i, \mathbf{u}), \tag{1.7}$$

where the sum picks up those dual variables $\alpha(i, \mathbf{u})$ that have equal value $u_p = y_p$ on the subset $p \subset \{1, \ldots, k\}$.

For the hypergraph H, the marginal dual vector containing the hyperedge marginals of the example $\mathbf{x}_i$ is given by

$$\mu_i = (\mu(i, e, \mathbf{u}_e))_{e \in E, \mathbf{u}_e \in \mathcal{Y}_e}.$$

The marginal vector of the whole training set is the concatenation of the single example marginal dual vectors $\mu = (\mu_i)_{i=1}^m$. The vector has dimension $d_{\mathcal{M}} = m \sum_{e \in E} |\mathcal{Y}_e| = O(m|E| \max_e |\mathcal{Y}_e|)$. Thus the dimension is linear in the number of examples, hyperedges and the maximum cardinality of the set of labellings of a single hyperedge.

The indicator functions in the definitions (1.7) of all relevant marginals can be collectively represented by the matrix M_H, $M_H(e, \mathbf{u}_e; \mathbf{y}) = [\mathbf{u}_e = \mathbf{y}_e]$, and the relationship between a dual vector α and the corresponding marginal vector μ is given by the linear map $M_H \alpha_i = \mu_i$ and $\mu = (M_H \alpha_i)_{i=1}^m$. The image of the set $\mathcal{A}_i$, defined by

$$\mathcal{M}_i = \{\mu_i | \ \exists \alpha_i \in \mathcal{A}_i : M_H \alpha_i = \mu_i\}$$

is called the *marginal polytope* of α_i on H.

The following properties of the set $\mathcal{M}_i$ are immediate:

Theorem 1 *Let $\mathcal{A}_i$ be the polytope of (1.6) and let $\mathcal{M}_i$ be the corresponding marginal polytope. Then*

- *the vertex set of $\mathcal{M}_i$ is the image of the vertex set of $\mathcal{A}_i$:*

$$V_i^{\mu} = \{\mu | \ \exists \alpha \in V_i^{\alpha} : M_{\mathcal{H}} \alpha = \mu\}.$$

- *As an image of a convex polytope $\mathcal{A}_i$ under the linear map M_H, $\mathcal{M}_i$ is a convex polytope.*

These properties underlie the efficient solution of the dual problem on the marginal polytope.

1.2.6 Marginal dual problem

The exponential size of the dual problem (1.4) can be tackled via the relationship between its feasible set $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_m$ and the marginal polytopes $\mathcal{M}_i$ of each $\mathcal{A}_i$.

Given a decomposable loss function

$$\ell(\mathbf{y}_i, \mathbf{y}) = \sum_{e \in E} \ell_e(i, \mathbf{y}_e)$$

and linear margin scaling $\gamma(\mathbf{y}', \mathbf{y}) = \ell(\mathbf{y}_i, \mathbf{y})$, the linear part of the objective satisfies

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{\mathbf{y} \in \mathcal{Y}} \alpha(i, \mathbf{y}) \ell(i, \mathbf{y}) &= \sum_{i=1}^{m} \sum_{\mathbf{y}} \alpha(i, \mathbf{y}) \sum_{e} \ell_e(i, \mathbf{y}_e) \\
&= \sum_{i=1}^{m} \sum_{e \in E} \sum_{u \in \mathcal{Y}_e} \sum_{\mathbf{y}: \mathbf{y}_e = u} \alpha(i, \mathbf{y}) \ell_e(i, u) = \sum_{i=1}^{m} \sum_{e \in E} \sum_{u \in \mathcal{Y}_e} \mu(e, u) \ell_e(i, u) \\
&= \sum_{i=1}^{m} \mu_i^T \ell_{\mathcal{M}_i} = \mu^T \ell_{\mathcal{M}}, \qquad (1.8)
\end{aligned}$$

where $\ell_H = (\ell_e(i, u))_{i=1, e \in E, u \in \mathcal{Y}_e}^{m}$ is the marginal loss vector.

Given an orthogonal feature representation inducing a decomposable kernel (1.5), the quadratic part of the objective becomes

$$\begin{aligned}
\alpha K \alpha = & \\
&= \sum_{e} \sum_{i, i'} \sum_{\mathbf{y}, \mathbf{y}'} \alpha(i, \mathbf{y}) K_e(i, \mathbf{y}_e; i', \mathbf{y}'_e) \alpha(i', \mathbf{y}') \\
&= \sum_{e} \sum_{i, i'} \sum_{\mathbf{u}, \mathbf{u}'} K_e(i, \mathbf{u}; i', \mathbf{u}') \sum_{\mathbf{y}: \mathbf{y}_e = \mathbf{u}} \sum_{\mathbf{y}': \mathbf{y}'_e = \mathbf{u}'} \alpha(i, \mathbf{y}) \alpha(i', \mathbf{y}') \\
&= \sum_{e} \sum_{i, i'} \sum_{\mathbf{u}, \mathbf{u}'} \mu_e(i, \mathbf{u}) K_e(i, \mathbf{u}; i', \mathbf{u}') \mu_e(i, \mathbf{u}') \\
&= \mu^T K_H \mu, \qquad (1.9)
\end{aligned}$$

where $K_H = \operatorname{diag}(K_e, e \in E)$ is a block diagonal matrix with hyperedge-specific kernel blocks K_e.

The objective should be maximized with respect to μ whilst ensuring that there exist $\alpha \in \mathcal{A}$ satisfying $M\alpha = \mu$, so that the marginal dual solution represents a feasible solution of the original dual. By Theorem 1 the feasible set of the marginalized problem is the marginal dual polytope, or to be exact the cartesian product of the marginal polytopes of single examples (which are in fact equal):

$$\mathcal{M} = \mathcal{M}_1 \times \cdots \times \mathcal{M}_m$$

In summary, the marginalized optimization problem can be stated in implicit form as

$$\max_{\mu \in \mathcal{M}} \mu^T \ell_H - \frac{1}{2} \mu^T K_H \mu. \tag{1.10}$$

This problem is a quadratic programme with a linear number of variables in the number of training examples and in the number of hyperedges. If the cardinality of hyperedges is bounded by a constant, the number of variables is linear also in the number of microlabels.

For optimization algorithms, an explicit characterization of the feasible set is required. However, characterizing the polytope $\mathcal{M}$ in terms of linear constraints defining the faces of the polytope, is in general infeasible. Singly-connected graphs are an exception: for such a graph $G = (V, E)), E \subset V \times V$, the marginal polytope is exactly reproduced by the box constraints

$$\sum_{\mathbf{u}_e} \mu_e(i, \mathbf{u}_e) \leq C, \forall i, e \in E, \mu_e \geq 0 \tag{1.11}$$

and the local consistency constraints

$$\sum_{y_k} \mu_{kj}(i, (y_k, y_j)) = \mu_j(i, y_j); \sum_{y_j} \mu_{kj}(i, (y_k, y_j)) = \mu_k(i, y_k). \tag{1.12}$$

In this case the size of the resulting constraint set is linear in the number of vertices of the graph. Thus for small singly-connected graphs they can be written down explicitly and the resulting optimization problem has linear size both in the number of examples and the size of the graph. Thus the approach can in principle be made to work, although not with off-the-shelf QP solvers (see sections 1.3 and 1.4).

For general graphs and hypergraphs the situation is more complicated (c.f Wainwright and Jordan 2003): the local consistency of edges or hyperedges is not sufficient to ensure global consistency, that is, there are marginal dual vectors μ for which there exists no $\alpha \in \mathcal{A}$ such that $M_H \alpha = \mu$. For global consistency, one needs to derive the junction tree of the hypergraph and write down the local consistency constraints of the junction tree. Consequently, the size of the constraint set is linear in the size of the junction tree, which, unfortunately, can be much more than the size of the original hypergraph. Thus in general an explicit description of the constraint set is not a tractable approach.

In the following, we derive an approach where we avoid the explicit consideration of the constraints, which in part contributes towards an efficient optimization approach.

1.3 Efficient optimization on the marginal dual polytope

Despite the polynomial number of dual variables, the marginal dual problem is still a challenging one to solve if the hypergraph is large or dense. In the following,

we describe an algorithm that enables us to tackle general graph structures with reasonable computational complexity. The main ingredients are

- Partial decomposition via a gradient-based approaches
- Efficient optimization via the conditional gradient method
- Computation of feasible descent directions via solving an inference problem on the hypergraph.

1.3.1 Decomposition of the learning problem

The size of the optimization problem suggests that we should try to decompose it in some way. However, the marginalized dual problem has a property that defies full decomposition:

- The constraints decompose by the examples, i.e. we can collect all the constraints related to training example $\mathbf{x}_i$ into the linear system $A\mu_i \leq b, C\mu_i = d$. However, decomposition by the hyperedges is not possible due to the consistency constraints between hyperedges.
- The kernel decomposes by the hypergraph structure as $K_H = \text{diag}\,(K_e, e \in E)$ but the interactions between examples (represented by a non-sparse x-kernel K^x) forbid a similar decomposition by the examples.

A partial decomposition becomes possible via gradient-based approaches. The gradient of the objective $obj(\mu) = \ell_H^T\mu - (1/2)\mu^T K_H \mu$,

$$g = \nabla[obj(\mu)] = \ell_H - K_H\mu = (\ell_i - (K_{i1}, \ldots, K_{im})\mu)_{i=1}^m = (g_i)_{i=1}^m$$

can be consulted and updated for each example independently. Thus a general gradient-based iterative approach is possible:

1. For example $\mathbf{x}_i$, using the gradient information g_i find $\Delta\mu_i$ such that $\mu_i + \Delta\mu_i$ is feasible and the objective value is increased, i.e. $g_i^T \Delta\mu_i > 0$
2. If a stopping criterion is satisfied, stop, otherwise move to the next example, and repeat.

In the next section, we describe the application of a conditional gradient algorithm (c.f. Bertsekas 1999), which follows the above general template.

1.3.2 Conditional gradient algorithm

Let us consider optimizing the dual variables $\mu_i = (\mu_e(i, \mathbf{u}))_{e\in E, \mathbf{u}\in \mathcal{Y}_e}$ of example $\mathbf{x}_i$. We denote by $\ell_i = (\ell_e(i, \mathbf{u}_e)_{e\in E}$ the corresponding loss vector and by $K_{ij} = diag(K_{eij}, e \in E)$, where $K_{eij} = (K_e(i, u; j, v)_{u,v\in \mathcal{Y}_e}$, the block of kernel values between examples i and j, on edge e (Note that K_{ij} also is block-diagonal like the full marginalized kernel K_H). Finally we denote by $K_{i\cdot} = (K_{ij})_{j\in\{1,\ldots,m\}}$ the columns of the kernel matrix K_H referring to example i.

Obtaining the gradient for the $\mathbf{x}_i$-subspace requires computing the corresponding part of the gradient of the objective function in (1.10) which is $\mathbf{g}_i = \ell_i - K_{i\cdot}\mu$. However when updating μ_i only, evaluating the change in objective and updating the gradient can be done more cheaply. We have

$$\Delta\mathbf{g}_i = -K_{ii}\Delta\mu_i$$

and

$$\Delta obj = \mathbf{g}_i^T\Delta\mu_i - \frac{1}{2}\Delta\mu_i K_{ii}\Delta\mu_i.$$

Thus local optimization in a subspace of a single training example can be done without consulting the other training examples. On the other hand, we do not want to spend too much time in optimizing a single example: since the dual variables of the other examples are non-optimal, so is the initial gradient $\mathbf{g}_i$. Thus the optimum we would arrive at by optimizing μ_i whilst keeping other examples fixed, would not be the global optimum of the quadratic objective. It makes more sense to optimize all examples more or less in tandem so that the full gradient approaches its optimum as quickly as possible.

In our approach, we have chosen to conduct a few optimization steps for each training example using a conditional gradient ascent (see Algorithm 1.2) before moving on to the next example. The iteration limit for each example is set by using the Karush-Kuhn-Tucker(KKT) conditions as a guideline; the larger contribution to the duality gap by an example, the higher the iteration limit.

The pseudocode of our algorithm is given in Algorithm 1.1. It takes as input the training data, the hypergraph H, and the loss vector $\ell_\mu = (\ell_i)_{i=1}^m$. The algorithm chooses a chunk of examples as the working set, computes the kernel for each $\mathbf{x}_i$ and makes an optimization pass over the chunk. After one pass, the gradient, slacks and the duality gap are computed and a new chunk is picked. The process is iterated until the duality gap falls below a given threshold.

Note in particular, that the joint kernel is not explicitly computed, although evaluating the gradient requires computing the product $K_H\mu$. We are able to take advantage of the special structure of the feature vectors, where the interaction between the labellings and the x-features of a hyperedge is given by a tensor product, to facilitate the computation using the x-kernel and the dual variables only.

1.3.3 Conditional Subspace Gradient Ascent

The optimization algorithm used for a single example is a variant of conditional gradient ascent (or descent) algorithms (Bertsekas, 1999). The algorithms in this family solve a constrained quadratic problem by iteratively stepping to the best feasible direction with respect to the current gradient. It exploits the fact if μ^* is an optimum solution of a maximization problem with objective function f over the

Algorithm 1.1 Maximum margin optimization algorithm for a Conditional Random Field on a hypergraph.

Require: Training data $S = ((\mathbf{x}_i, \mathbf{y}_i))_{i=1}^m$, hyperedge set E of the hypergraph, a loss vector ℓ, and the feasibility domain $\mathcal{M}$.
Ensure: Dual variable vector μ and objective value $f(\mu)$.
1: Initialize $\mathbf{g} = \ell_\mu$, $\xi = \ell$,$dg = \infty$ and $OBJ = 0$.
2: **while** $dg > dg_{min}$ & $iter < max\_iter$ **do**
3: $[WS, Freq]$ = UpdateWorkingSet$(\mu, \mathbf{g}, \xi)$;
4: Compute x-kernel values $K_{X,WS}$ with respect to the working set;
5: **for** $i \in WS$ **do**
6: Compute joint kernel block K_{ii} and subspace gradient $\mathbf{g}_i$;
7: $[\mu_i, \Delta obj]$ = CSGA$(\mu_i, \mathbf{g}_i, K_{ii}, \mathcal{M}_i, Freq_i)$;
8: **end for**
9: Compute gradient $\mathbf{g}$, slacks ξ and duality gap dg;
10: **end while**

feasibility domain $\mathcal{M}_i$ then it has to satisfy the first order optimality condition, that is the inequality

$$\nabla f(\mu_i)(\mu_i - \mu^*) \geq 0 \tag{1.13}$$

has to hold for any feasible μ_i chosen from $\mathcal{M}_i$.

The pseudocode of our variant CSGA is given in Algorithm 1.2. The algorithm takes as input the current dual variables, gradient, constraints and the kernel block for the example $\mathbf{x}_i$, and an iteration limit. It outputs new values for the dual variables μ_i and the change in objective value. As discussed above, the iteration limit is set very tight so that only a few iterations will be typically conducted.

First we need to find a feasible μ^* which maximizes the first order feasibility condition (1.13) at a fixed μ_i. This problem is an LP

$$\mu^* = \operatorname*{argmax}_{\mathbf{v} \in \mathcal{M}_i} \mathbf{g}_i^T \mathbf{v}. \tag{1.14}$$

The solution gives a direction potentially increasing the value of objective function f. Then we have to choose a step length τ that gives the optimal feasible solution as a stationary point along the line segment $\mu_i(\tau) = \mu_i + \tau \Delta\mu,\ \tau \in (0, 1]$, where $\Delta\mu = \mu^* - \mu_i$, starting from the known feasible solution μ_i.

The stationary point is found by solving the equation

$$\frac{\mathbf{d}}{\mathbf{d}\tau} \mathbf{g}_i^T \mu_i(\tau) - 1/2 \mu_i(\tau)^T K_{ii} \mu_i(\tau) = 0, \tag{1.15}$$

expressing the optimality condition with respect to τ. If $\tau > 1$, the stationary point is infeasible and the feasible maximum is obtained at $\tau = 1$. In our experience, the time taken to compute the stationary point was typically significantly smaller than the time taken to find μ_i^*.

Algorithm 1.2 Conditional subspace gradient ascent optimization step.

CSGA($\mu_i, \mathbf{g}_i, K_{ii}, \mathcal{M}_i, maxiter_i$)

Require: Initial dual variable vector μ_i, gradient $\mathbf{g}_i$, the feasible region $\mathcal{M}_i$, a joint kernel block K_{ii} for the subspace, and an iteration limit $maxiter_i$.

Ensure: New values for dual variables μ_i and change in objective Δobj.

$\Delta obj = 0; iter = 0;$
while $iter < maxiter$ **do**
  % find highest feasible point given $\mathbf{g}_i$
  $\mu^* = \text{argmax}_{\mathbf{v} \in \mathcal{M}_i} \ \mathbf{g}_i^T \mathbf{v}$;
  $\Delta\mu = \mu^* - \mu_i$;
  $q = \mathbf{g}_i^T \Delta\mu, \ r = \Delta\mu^T K_{ii} \Delta\mu$; % taken from the solution of (1.15)
  $\tau = \min(q/r, 1)$; % clip to remain feasible
  if $\tau \leq 0$ **then**
    break; % no progress, stop
  else
    $\mu_i = \mu_i + \tau\Delta\mu$; % update
    $\mathbf{g}_i = \mathbf{g}_i - \tau K_{ii} \Delta\mu$;
    $\Delta obj = \Delta obj + \tau q - \tau^2 r/2$;
  end if
  $iter = iter + 1$;
end while

1.3.4 Efficient computation of the feasible ascent direction

The main difficulty in optimizing the max-margin problem in the marginal dual form arises from the need to ensure marginal consistency: the box constraints are easy to satisfy by many algorithms, including variants of sequential margin optimization, SMO (Platt, 1999), or simple steepest gradient search. For tree-structured graphs the constraints can be written down explicitly, as in such graphs local consistency of adjacent edges is sufficient to ensure global consistency. For general graphs, such a relation does not hold: it is easy to find examples where a locally consistent distribution can be globally inconsistent. In principle, a sufficient constraint set for a general graph can be found via construction of the junction tree of the graph and writing down consistency constraints of the hyperedges. However, this approach suffers from the fact that, for a dense graph, the junction tree and consequently the constraint set, may be very (exponentially) large. Thus, other means need to be used to ensure global consistency of the marginal dual solution.

The basis of our solution is the following relationship, which can also be seen as a consequence of (Wainwright and Jordan, 2003, theorem 4):

Lemma 2 *For any gradient $\mathbf{g}_i$, there is a vertex $\alpha_i^* \in \mathcal{A}_i$ such that $\mu_i^* = M_H \alpha_i^*$ is an optimizer of (1.14).*

Proof Since $\mathcal{M}_i$ is a polyhedron and the objective is linear, it follows that among the optimal solutions of the conditional gradient (1.14) there is a vertex of $\mathcal{M}_i$. Denote this vertex by μ_i^*. Since the vertex set of $\mathcal{M}_i$ is the image of the vertex set of $\mathcal{A}_i$, μ_i^* is the image of some vertex $\alpha_i^* \in \mathcal{A}_i$. ■

Thus, α_i^* corresponding to the conditional gradient is either the zero vector or a unit vector corresponding to some multilabel $\mathbf{y}^*$.

Lemma 3 *If $\mu_i^* \neq 0$, then for all hyperedges $e \in E$ we have $\mu_i^*(e, \mathbf{y}_e^*) = C$, and $\mu_i^*(e, u) = 0$ for all $\mathbf{y}_e^* \neq u$.*

Proof Since $M_H \alpha_i^* = \mu_i^*$ and α_i^* has a single non-zero component, $\alpha^*(i, \mathbf{y}^*) = C$, μ_i^* is the $\mathbf{y}^*$'th column of M_H, multiplied by C. Thus the non-zero values of μ_i^* equal C. Let us now assume, contrary to the claim, that $\mu_i^*(e, \mathbf{u}) = \mu_i^*(e, \mathbf{u}') = C$ for some $\mathbf{u} \neq \mathbf{u}'$. But then by the definition of matrix M_H we must have $[\mathbf{y}_e^* = \mathbf{u}] = 1 = [\mathbf{y}_e^* = \mathbf{u}']$ which is a contradiction, and the claim follows. ∎

Consequently, $\mu_i^* = \mu_i(\mathbf{y}^*)$ is directly defined by the optimal labelling $\mathbf{y}^*$. The lemma also gives a recipe for constructing μ_i^* given $\mathbf{y}^*$.

We can now rewrite (1.14) in terms of the multilabels

$$\mathbf{y}^* = \operatorname*{argmax}_{\mathbf{y}} \mathbf{g}_i^T \mu_i^*(\mathbf{y}) = \operatorname*{argmax}_{\mathbf{y}} \sum_{e \in E} \mathbf{g}_{ie}^T \mu_{ie}^*(\mathbf{y}_e) = \operatorname*{argmax}_{\mathbf{y}} \sum_{e \in E} \mathbf{g}_{ie}(\mathbf{y}_e) C, \quad (1.16)$$

which is readily seen as an inference problem on the hypergraph H: one must find the configuration $\mathbf{y}^*$ that maximizes the sum of the hyperedge gradients $\mathbf{g}_{ie}(\mathbf{y}_e)$.

Thus we have translated our feasible ascent direction problem into an inference problem on the hypergraph. If we can solve the inference problem (1.16) efficiently, the conditional gradient method will be very efficient.

In addition, for our purposes there is no real need to compute the exact optimum, a direction that promises ascent with high likelihood is sufficient. Hence, fast approximate inference algorithms suffice here. Some examples of available methods are the following.

- For sequences and trees, inference can be implemented via dynamic programming and it has generally a linear time-complexity.
- Hypergraphs with low tree-width, can be converted to their junction trees and dynamic programming can be used on the junction tree to find the maximizing configuration. The size of the junction tree depends on the tree-width of the graph.
- Loopy belief propagation (LBP) refers to the use of the message-passing algorithm on a cyclic hypergraph. While this algorithm is not guaranteed to converge on such graphs, it has a successful track record in practical applications. For our purposes, the asymptotic convergence is not a central issue as long as the initial convergence is fast enough to find a configuration $\mathbf{y}^*$ corresponding to a descent direction.
- Tree reparametrization algorithm (TRP) (Wainwright et al., 2003) based on computing a series of spanning trees of the (hyper)graph. The convergence is often faster than that of LBP. Also in the case of TRP, the algorithm can be stopped after a few iterations once a configuration $\mathbf{y}^*$ guaranteeing descent is found.

All of the methods can be viewed as instantiations of message-passing algorithms (Wainwright and Jordan, 2003). In the next section we exemplify the optimization

approach on hierarchical problems, where exact inference can be implemented by dynamic programming.

1.4 Experiments

We tested the presented learning approach on three datasets that have an associated classification hierarchy:

- Reuters Corpus Volume 1, RCV1 (Lewis et al., 2004). 2500 documents were used for training and 5000 for testing. As the label hierarchy we used the 'CCAT' family of categories, which had a total of 34 nodes, organized in a tree with maximum depth 3. The tree is quite unbalanced, half of the nodes residing in depth 1.
- WIPO-alpha patent dataset (WIPO, 2001). The dataset consisted of the 1372 training and 358 testing document comprising the D section of the hierarchy. The number of nodes in the hierarchy was 188, with maximum depth 3. Each document in this dataset belongs to exactly one leaf category, hence it contains no multiple or partial paths.
- ENZYME classification dataset. The training data consisted of 7700 protein sequences with hierarchical classification given by the Enzyme Classification (EC) system. The hierarchy consisted of 236 nodes organized into a tree of depth three. Test data consisted of 1755 sequences.

The two first datasets were processed into bag-of-words representation with TFIDF weighting. No word stemming or stop-word removal was performed. For the ENZYME sequences a length-4 subsequence kernel was used. Note that in the Reuters Corpus multiple-partial paths exist: it is not the case that the correct classification is simply a single path to a leaf node; for a single example multiple paths in the hierarchy may be positively labelled, and it is not necessary that a path ends at a leaf node.

We compared the performance of the presented max-margin conditional random field (MMCRF)learning approach to three algorithms: SVM denotes an SVM trained for each microlabel separately, H-SVM denotes the case where the SVM for a microlabel is trained only with examples for which the ancestor labels are positive.

The SVM and H-SVM were run using the SVM-light package. After pre-computation of the kernel these algorithms are as fast as one could expect, as they just involve solving an SVM for each node in the graph (with the full training set for SVM and usually a much smaller subset for H-SVM).

H-RLS is a batch version of the hierarchical least squares algorithm described in Cesa-Bianchi et al. (2004). It essentially solves for each node i a least squares style problem $\mathbf{w}_i = (I + S_i S_i^T + \mathbf{x}\mathbf{x}^T)^{-1} S_i \mathbf{y}_i$, where S_i is a matrix consisting of all training examples for which the parent of node i was classified as positive, $\mathbf{y}_i$ is a microlabel vector for node i of those examples and I is the identity matrix of

appropriate size. Predictions for a node i for a new example $\mathbf{x}$ is -1 if the parent of the node was classified negatively and $\text{sign}(\mathbf{w}_i^{\mathrm{T}}\mathbf{x})$ otherwise.

H-RLS requires a matrix inversion for each prediction of each example, at each node along a path for which errors have not already been made. No optimization of the algorithm was made, except to use extension approaches to efficiently compute the matrix inverse (for each example an inverted matrix needs to be extended by one row/column, so a straightforward application of the Sherman-Morrison formula to efficiently update the inverse can be used).

The H-RLS and MMCRF algorithms were implemented in MATLAB. The tests were run on a high-end PC. For SVM,H-SVM and MMCRF, the regularization parameter value $C = 1$ was used in all experiments as in initial experiments its value did not seem to have a significant effect.

1.4.1 Obtaining consistent labellings.

As the learning algorithms compared here all decompose the hierarchy for learning, the multilabel composed of naively combining the microlabel predictions may be inconsistent, that is, they may predict a document as part of the child but not as part of the parent. For SVM and H-SVM consistent labellings were produced by post-processing the predicted labellings as follows: start at the root and traverse the tree in a breadth-first fashion. If the label of a node is predicted as -1 then all descendants of that node are also labelled negatively. This post-processing turned out to be crucial to obtain good accuracy, thus we only report results with the post-processed labellings. Note that H-RLS performs essentially the same procedure (see above). For the max-margin CRF models, we computed by dynamic programming the consistent multilabel with maximum likelihood

$$\hat{\mathbf{y}}(x) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}_T} P(\mathbf{y}|x) = \operatorname*{argmax}_{\mathbf{y}} \mathbf{w}^T\phi(x,\mathbf{y}),$$

where $\mathcal{Y}_T$ is the set of multilabels that correspond to unions of partial paths in T. This inference problem can be solved by the same dynamic programming algorithm as the one used for learning, with the exception that the set of multilabels considered is restricted to those consistent with the union of partial paths model.

1.4.2 Efficiency of optimization.

To give an indication of the efficiency of the MMCRF algorithm, Figure 1.3 shows an example learning curve on the WIPO-alpha dataset. The number of marginal dual variables for this training set is just over one million and the marginalized kernel matrix K_H—if computed explicitly—would have approximately 5 billion entries. Note that the solutions for this optimization are not sparse, typically less than 25% of the marginal dual variables are zero. Training and test losses (ℓ_Δ) are all close to their optima within 10 minutes of starting the training, and the objective is within 2 percent of the optimum in 30 minutes.

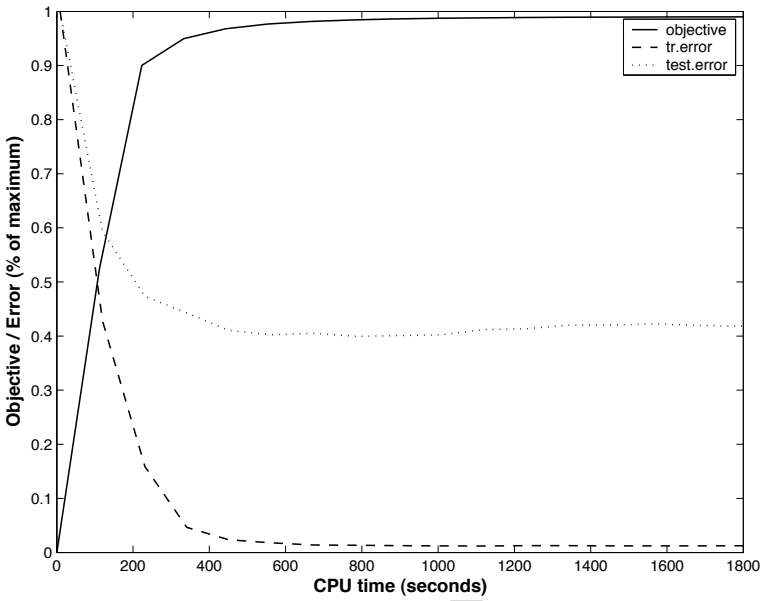

Figure 1.3 The objective function (% of optimum) and ℓ_Δ losses for MMCRF on training and test sets (WIPO-alpha)

To put these results in perspective, for the WIPO data set SVM (SVM-light) takes approximately 50 seconds per node, resulting in a total running time of about 2.5 hours. The running time of H-RLS was slower than the other methods, however this could be due to our non-optimised implementation. It is our expectation that it would be very close to the time taken by H-SVM if coded more efficiently.

Therefore, the methods presented in this paper are very competitive from a computational efficiency point of view to other methods which do not operate in the large feature/output spaces of MMCRF.

Figure 1.4 shows for WIPO-alpha the efficiency of the dynamic programming (DP) based computation of update directions as compared to solving the update directions with MATLAB's linear interior point solver LIPSOL. The DP based updates result in an order of magnitude faster optimization than using LIPSOL.

In addition for DP the effect of the iteration limit for optimization speed is depicted. Setting the iteration limit too low (1) or too high (50) slows down the optimization, for different reasons. A too tight iteration limit makes the overhead in moving from one example to the other dominate the running time. A too high iteration limit makes the the algorithm spend too much time optimizing the dual variables of a single example. Unfortunately, it is not straightforward to suggest a iteration limit that would be universally the best.

Table 1.1 Prediction losses $\ell_{0/1}$ and ℓ_Δ, precision, recall and F1 values obtained using different learning algorithms. All figures are given as percentages. Precision and recall are computed in terms of totals of microlabel predictions in the test set.

REUTERS	$\ell_{0/1}$	ℓ_Δ	P	R	F1
SVM	32.9	0.61	94.6	58.4	72.2
H-SVM	29.8	0.57	92.3	63.4	75.1
H-RLS	28.1	**0.55**	91.5	65.4	**76.3**
MMCRF-ℓ_Δ	**27.1**	0.58	91.0	64.1	75.2
MMCRF-$\ell_{\tilde{H}}$	27.9	0.59	85.4	68.3	75.9
WIPO-alpha	$\ell_{0/1}$	ℓ_Δ	P	R	F1
SVM	87.2	1.84	93.1	58.2	71.6
H-SVM	76.2	1.74	90.3	63.3	74.4
H-RLS	72.1	1.69	88.5	66.4	75.9
MMCRF-ℓ_Δ	70.9	**1.67**	90.3	65.3	75.8
MMCRF-$\ell_{\tilde{H}}$	**65.0**	1.73	84.1	70.6	**76.7**
ENZYME	$\ell_{0/1}$	ℓ_Δ	P	R	F1
SVM	99.7	1.3	99.6	41.1	58.2
H-SVM	98.5	**1.2**	98.9	41.7	58.7
H-RLS	95.6	2.0	51.9	54.7	53.3
MMCRF-ℓ_Δ	95.7	**1.2**	87.0	49.8	**63.3**
MMCRF-$\ell_{\tilde{H}}$	**85.5**	2.5	44.5	66.7	53.4

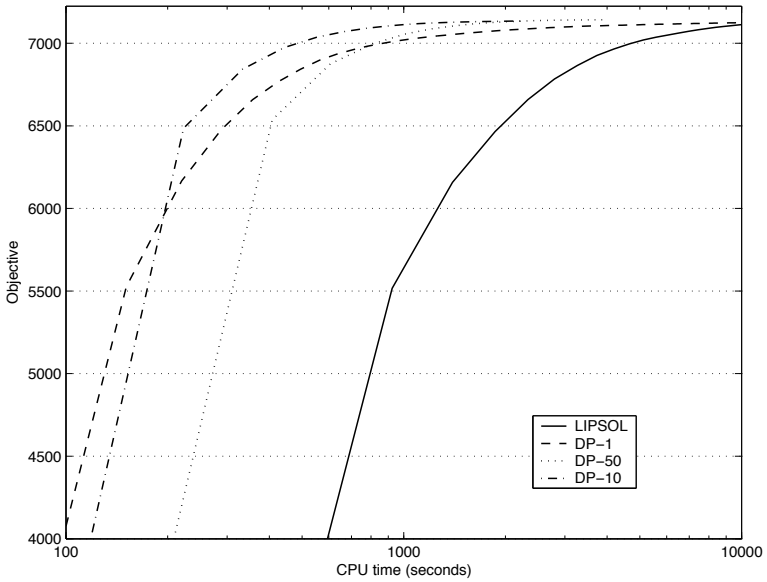

Figure 1.4 Learning curves for MMCRF using LIPSOL and dynamic programming (DP) to compute update directions (WIPO-alpha). Curves with iteration limits 1,10 and 50 are shown for DP. The LIPSOL curve is computed with iteration limit set to 1.

1.4.3 Comparison of predictive accuracies of different algorithms.

In our final test we compare the predictive accuracy of MMCRF to other learning methods. For the MMCRF we include the results for training with ℓ_Δ and $\ell_{\tilde{H}}$ losses (see Section 1.2.2 for a discussion of loss functions). For training SVM and H-SVM, these losses produce the same learned model.

Table 1.1 depicts the different test losses, as well as the standard information retrieval statistics precision (P), recall (R) and F1 statistic ($F1 = 2PR/(P+R)$). Precision and recall were computed over all microlabel predictions in the test set. Flat SVM is expectedly inferior to the competing algorithms with respect to most statistics, as it cannot utilize the dependencies between the microlabels in any way. The two variants of MMCRF are the most efficient in getting the complete tree correct as shown by the lower zero-one loss. With respect to other statistics, the structured methods are quite evenly matched overall.

1.5 Discussion

In this chapter we have introduced a general methodology for efficient optimization of structured classification tasks in the max-margin setting. We discussed how the choice of feature representation and loss function can affect the computational burden imposed by the primal and dual formulations. We have shown that for

the non-restrictive setting where an orthogonal feature representation is used in combination with a loss function that is edge-decomposable, we can efficiently solve the optimization problem using conditional gradient methods by exploiting the block structure of the gradient. The resulting method has been tested on 3 datasets for which the labels are placed within a hierarchical structure. The first two of these were document classification tasks that used the standard TF/IDF feature representation. The third data set focussed on enzyme analysis and used a string kernel as the feature mapping; this task would therefore not be practical in alternative max-margin settings where only the primal objective function is used. In all cases the approach in this chapter achieved high performance and took less computation time.

Our method can be contrasted to the structured Exponentiated Gradient (EG) approach presented in (Bartlett et al., 2004). Both algorithms are iterative gradient-based algorithms but with significant differences. First, the update direction of the MMCRF algorithm is towards the best feasible direction while the structured EG update tends to look for sparse directions of steep ascent. Conditional gradient updates are known to work well in the early stages of optimization but less so in the final stages (Bertsekas, 1999). On the other hand, the emphasis on sparse solutions should benefit EG in the final stages of optimization as the (full) dual optimum typically is sparse. Finally, the tractable formulation of the structured EG (Bartlett et al., 2004) relies on enforcing a Gibbs distribution (with polynomial number of parameters) on the dual variables while our MMCRF does not make any additional assumptions of the distribution of the dual variables. We leave as future work to study the relative merits of these methods.

One advantage of the marginal approach used in this paper is that there is a clear relationship between the complexity of the optimization and the representation of the output structure which is used. For the hierarchical data sets used in this paper, the inference step can be solved exactly and efficiently using dynamic programming; thus ensuring that the computational complexity of the terms in the objective function only grow linearly with the size of the output structure. In the case of more general structures, then the inference step must either be solved by considering junction trees where possible, or by applying approximate inference methods such as loopy belief propagation. It is still an open question as to how the performance of the algorithm would be effected should a technique such as LBP be used on a more complex output structure. Given that the inference is only used to find a suitable direction across which to optimize, one could expect that exact inference is unnecessary and a good approximation is more than sufficient to guide the optimization process. In general though this is an open question and will be the subject of future research.

Acknowledgments

This work was supported in part by the PASCAL Network of Excellence, IST-2002-506778. C. Saunders is supported in part by the EPSRC grant no GR/S22301/01

(Development and Application of String-Type Kernels). The work by J. Rousu was partially funded by a Marie Curie Individual Fellowship HPMF-2002-02110 and undertaken whilst visiting Royal Holloway, University of London.

References

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

P. L. Bartlett, M. Collins, and B. Taskar amd D. McAllester. Exponentiated gradient algorithms for large-margin structured classication. In *Neural Information Processing Systems*, 2004.

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

N. Cesa-Bianchi, C. Gentile, A. Tironi, and L. Zaniboni. Incremental algorithms for hierarchical classification. In *Neural Information Processing Systems*, 2004.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. In *Proc. 21th International Conference on Machine Learning*, pages 504–511, 2004.

J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: probabilistic modeling for segmenting and labeling sequence data. In *18th International Conference on Machine Learning ICML*, 2001.

David D. Lewis, Y. Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, Apr 2004.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods - -Support Vector Learning*, pages 185 – 208, Cambridge, MA, 1999. MIT Press.

B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *Proc. 21th International Conference on Machine Learning*, pages 807–814, 2004a.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Neural Information Processing Systems 2003*, 2004b.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y.n Altun. Support vector machine learning for interdependent and structured output spaces. In *Proc. 21th International Conference on Machine Learning*, pages 823–830, 2004.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-based reparameterization framework for analysis of sum-product and related algorithms. *IEEE Transactions on*

information theory, 49:1120–1146, May 2003.

M. Wainwright, T. Jaakkola, and A. Willsky. Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11):3697–3717, 2005.

M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. Technical Report 649, UC Berkeley, Department of Statistics, September 2003.

WIPO. *World Intellectual Property Organization. http://www.wipo.int/classifications/en.* 2001.