# A Note on Structural Extensions of SVMs

Mark Schmidt

March 29, 2009

## 1 Introduction

This document is intended as an introduction to structural extensions of support vector machines for those who are familiar with logistic regression (binary and multinomial) and discrete-state probabilistic graphical models (in particular, conditional random fields). No prior knowledge about support vector machines is assumed. The outline is as follows

- §2 motivates and outlines binary support vector machines. The contents of this section are standard, and the reader is referred to [Vapnik, 1995] for more details. However, we will follow a non-standard presentation; instead of motivating support vector machines from the point of view of optimal separating hyper-planes, we focus on the relationship between logistic regression and support vector machines through likelihood ratios, a viewpoint due to SVN Vishwanthan.

- §3 discusses multi-class generalizations of binary suppport vector machines, focusing on the $NK$-slack formulation of [Weston and Watkins, 1999], and the $N$-slack formulation of [Crammer and Singer, 2001].

- §4 discusses extensions of the $N$-slack multi-class support vector machine that can model data with structured output spaces. In particular, this section focuses on hidden Markov support vector machines [Altun et al., 2003, Joachims, 2003], max-margin Markov networks [Taskar et al., 2003], and structural support vector machines [Tsochantaridis et al., 2004].

- §5 (in progress) will discuss solving the optimization problems in §4. There are four main approaches:
  (1) sub-gradient methods
  [Collins, 2002, Altun et al., 2003, Zhang, 2004, Shalev-Shwartz et al., 2007],
  (2) cutting plane and bundle methods
  [Tsochantaridis et al., 2004, Joachims, 2006, Teo et al., 2007, Smola et al., 2008, Joachims et al., 2009],
  (3) polynomial-sized reformulations [Taskar et al., 2003, Bartlett et al., 2005, Collins et al., 2008], and
  (4) min-max formulations [Taskar et al., 2004, Taskar et al., 2006b, Taskar et al., 2006a].

Some of the things that will not be covered in this documentation are a discussion of optimal separating hyper-planes, deriving the Wolfe dual formulations, the kernel trick, and computational learning theory. [Lacoste-Julien, 2003] is an accessible introduction to max-margin Markov networks that discusses these topics, and also contains introductory material on graphical models and conditional random fields.

## 2 Support Vector Machines

We first look at binary classification with an $N$ by $P$ design matrix $X$, and an $N$ by 1 vector of class labels $y$ with $y_i \in \{-1, 1\}$. We will ignore the bias term to simplify presentation, and consider probabilities for the class labels that are proportional to the exponential of a linear function of the data,

$$p(y_i = 1|w, x_i) \propto \exp(w^T x_i),$$

$$p(y_i = -1|w, x_i) \propto \exp(-w^T x_i),$$

Using these class probabilities is equivalent to using a logistic regression model, and we can estimate the parameters $w$ by maximizing the likelihood of the training data, or equivalently minimizing the negative log-likelihood

$$\min_w -\sum_i \log p(y_i|w, x_i).$$

The solution to this problem is not necessarily unique (and the optimal parameters may be unbounded). To yield a unique solution (and avoid over-fitting), we typically add a penalty on the $\ell_2$-norm of the parameter vector and compute a penalized maximum likelihood estimate

$$\min_w -\sum_i \log p(y_i|w, x_i) + \lambda||w||_2^2,$$

where the scalar $\lambda$ controls the strength of the regularizer. With an estimate of the parameter $w$, we can classify a data point $x_i$ using

$$\hat{y} = \begin{cases} 1 & \text{if } p(y_i = 1|w, x_i) > p(y_i = -1|w, x_i) \\ -1 & \text{if } p(y_i = 1|w, x_i) < p(y_i = -1|w, x_i) \end{cases}.$$

But what if our goal isn't to have a good model $p(y_i|w, x_i)$, but rather to make the right decision for all our training set points? We can express this in terms of the following condition on the likelihood ratios

$$\forall_i \ \frac{p(y_i|w, x_i)}{p(-y_i|w, x_i)} \geq c,$$

where $c > 1$. The exact choice of $c$ is arbitrary, since if we can satisfy this for some $c > 1$, then we can also satisfy it for any $c' > 1$ by re-scaling $w$. Taking logarithms, we can re-write this condition as

$$\forall_i \ \log p(y_i|w, x_i) - \log p(-y_i|w, x_i) \geq \log c,$$

and plugging in the definitions of $p(y_i|w, x_i)$ we can write this as

$$\forall_i \ 2y_i w^T x_i \geq \log c.$$

Since $c$ is an arbitrary constant greater than 1, we will pick $c$ so that $(1/2)\log c = 1$, so that our conditions can be written in a very simple form

$$\forall_i \ y_i w^T x_i \geq 1.$$

This is a linear feasibility problem, and it can be solved using techniques from linear programming. However, one of two things will go wrong: (i) the solution may not be unique, or (ii) there may be no solution. As before, we can make the parameters identifiable by using an $\ell_2$-norm regularizer, which leads to a quadratic program

$$\min_w \lambda||w||_2^2,$$

$$s.t. \ \forall_i \ y_i w^T x_i \geq 1,$$

In this case, the choice of $\lambda$ is arbitrary since solving this quadratic program will yield the solution with smallest $\ell_2$ norm for any $\lambda > 0$. To address the case where the linear feasibility problem has no solution, we will introduce a non-negative slack variable $\xi_i$ for each training case that allows the constraint to be violated for that instance, but we will penalize the $\ell_1$-norm of $\xi$ to try and minimize the violation of the constraints. This yields the quadratic program

$$\min_{w,\xi} \sum_i \xi + \lambda||w||_2^2,$$

$$s.t. \ \forall_i \ y_i w^T x_i \geq 1 - \xi_i, \ \ \forall_i \ \xi_i \geq 0,$$

which has $P + N$ variables and is the primal form of support vector machines. Specifically, this is what is known as a soft-margin support vector machine [Vapnik, 1995]. The support vectors are those data points where the non-negativity constraint holds with equality (to keep the presentation simple, we will not discuss the dual form or the 'kernel trick').

By re-arranging the constraints, we have that for all $i$, $\xi_i \geq 0$ and $\xi_i \geq 1 - y_i w^T x_i$, and therefore we know that $\xi_i \geq \max\{0, 1 - y_i w^T x_i\}$. Further, since we are directly minimizing a linear function of $\xi_i$, we can show that the following minimization

$$\min_w \sum_i (1 - y_i w^T x_i)^+ + \lambda ||w||_2^2,$$

where we use $(z)^+$ to denote $\max\{0, z\}$, has the same solution as the primal form of support vector machines. This is an unconstrained but non-differentiable problem in $P$ variables[1]. The term $\sum_i (1 - y_i w^T x_i)^+$ is known as the *hinge loss*, and is an upper bound on the number of classification errors.

To summarize, $\ell_2$-regularized logistic regression and support vector machines are closely related. They employ a very similar model and use the same regularization, but differ in that logistic regression seeks to maximize the (penalized) likelihood (a log-concave smooth approximation to the number of classification errors), while support vector machines seek to minimize the (penalized) constraint violations in a set of conditions on the likelihood ratios (a convex piecewise-linear upper bound on the number of classification errors).

# 3 Multi-Class Support Vector Machines

Viewed from the optimal separating hyper-plane perspective, it is not immediately clear how to extend binary support vector machines to the case where we consider $K$ possible class labels. Most attempts at developing multi-class support vector machines concentrate on producing a multi-class classifier by combining the decisions made by a set of independent binary classifiers, a popular approach of this type is to use error-correcting output codes [Dietterich and Bakiri, 1995]. However, the likelihood ratio perspective of support vector machines suggests an obvious generalization to the multi-class scenario by analogy to the extensions of logistic regression to the multi-class scenario.

A natural generalization of the binary logistic regression classifier to the case of a $K$-class problem is to use a weight vector $w_k$ for each class

$$p(y_i = k | w_k, x_i) \propto \exp(w_k^T x_i).$$

Parameter estimation is performed as in the binary case, and we are lead to the decision rule

$$\hat{y}_i = \max_k p(y_i = k | w_k, x_i).$$

Viewed in terms of likelihood ratios, we would like our decision rule to satisfy the constraint

$$\forall_i \forall_{k \neq y_i}, \ \frac{p(y_i | w, x_i)}{p(y_i = k | w_k, x_i)} \geq c$$

Following the same steps as before, we arrive at a set of linear constraints of the form

$$\forall_i \forall_{k \neq y_i}, \ w_{y_i}^T x_i - w_k^T x_i \geq 1.$$

We will again consider the minimizing $\ell_2$-norm of $w$ (concatenating all $K$ of the $w_k$ vectors into one large vector), and introduce slack variables to allow for constraint violation. However, there are different ways

[1] If we instead considered penalizing the squared $\ell_2$-norm of the slack variables, then we can re-write the problem as a differentiable unconstrained optimization. This *smooth support vector machine* is due to [Lee and Mangasarian, 2001]

to introduce the slack variables. The original method of [Weston and Watkins, 1999] introduces one slack variable for each of these constraints;

$$\min_{w,\xi} \sum_i \sum_{k \neq y_i} \xi_{i,k} + \lambda ||w||_2^2,$$

$$\forall_i \forall_{k \neq y_i},\ w_{y_i}^T x_i - w_k^T x_i \geq 1 - \xi_{i,k},\ \ \forall_i \forall_{k \neq y_i} \xi_{i,k} \geq 0,$$

a quadratic program with $NK+PK$ variables. We will refer to this formulation as the $NK$-slack formulation. An equivalent unconstrained optimization problem where we eliminate the slack variables is

$$\min_w \sum_i \sum_{k \neq y_i} (1 - w_{y_i}^T x_i + w_k^T x_i)^+ + \lambda ||w||_2^2,$$

an unconstrained non-differentiable problem in $PK$ variables[2].

Later, [Crammer and Singer, 2001] proposed what we will call the $N$-slack multi-class formulation, which can be motivated by re-writing the likelihood ratio in the equivalent form

$$\forall_i \ \frac{p(y_i|w, x_i)}{\max_{k \neq y_i} p(y_i = k|w_k, x_i)} \geq c.$$

Although this leads an equivalent set of constraints, the formulations differ when we introduce slack variables on this set of constraints, since each training instance is only associated with 1 slack variable;

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$\forall_i \forall_{k \neq y_i},\ w_{y_i}^T x_i - w_k^T x_i \geq 1 - \xi_i,\ \ \forall_i \xi_i \geq 0,$$

a quadratic program with $N + PK$ variables. We will refer to this as the $N$-slack formulation. An equivalent unconstrained optimization problem where we eliminate the slack variables is

$$\min_w \sum_i \max_{k \neq y_i} (1 - w_{y_i}^T x_i + w_k^T x_i)^+ + \lambda ||w||_2^2,$$

an unconstrained non-differentiable problem in $PK$ variables.

Comparing these two formulations, we see that the $NK$-slack formulation is a much more punishing penalty, it punishes violation of the constraints for any competing hypothesis. In contrast, the $N$-slack formulation only punishes based on the violation of the most likely competing hypothesis, even if there are many likely competing hypothesis. However, the advantage of the $N$-slack formulation is that it leads to efficient structural extensions.

# 4 Hidden Markov Support Vector Machines, Max-Margin Markov Networks, Structural Support Vector Machines

We now move to the case where we no longer have a single class label $y_i$ for each training instance, but instead have a set of $j$ labels $y_{i,j}$. If these labels are independent, then we can simply use the methods above to fit an independent support vector machine to each label. The more interesting case occurs when the elements of the output vector are dependent. For example, we might have a sequential dependency; the

[2]If we consider penalizing the squared $\ell_2$-norm of the $NK$-slack formulation, then we can write the optimization as a differentiable unconstrained optimization. A demo of this multi-class extension of smooth support vector machines is at `http://www.cs.ubc.ca/~schmidtm/Software/minFunc/minFunc.html`

label $y_{i,j}$ depends on the previous label $y_{i,j-1}$. The structural extension of logistic regression to handle this type of dependency structure is known as a linear-chain conditional random field [Lafferty et al., 2001]. In the case of tied parameters, binary labels, Ising-like potentials, and sequences of labels with the same length $S$ (removing any of these restrictions is trivial, but would require more notation), the probability of a vector of labels $Y_i$ is written as

$$p(Y_i|w, X_i) \propto \exp(\sum_{j=1}^{S} y_{i,j} w_n^T x_{i,j} + \sum_{j=1}^{S-1} y_{i,j} y_{i,j+1} w_e^T x_{i,j,j+1}),$$

where $w_n$ represent the node parameters, $w_e$ represent the edge/transition parameters, and we allow a feature vector $x_{i,j}$ for each node and $x_{i,j,j+1}$ for each transition. Although there are other possibilities, we consider the decision rule

$$\hat{Y}_i = \max_{Y_i} p(Y_i|w, X_i).$$

Computing this maximum by considering all possible labels may no longer be feasible, since there are now $2^S$ possible labels, but the maximum can be found in $\mathcal{O}(S)$ by dynamic programming in the form of the Viterbi decoding algorithm.

Moving to an interpretation in terms of likelihood ratios, we would like our classifier to satisfy the criteria

$$\forall_i \forall_{Y_i' \neq Y_i} \frac{p(Y_i|w, X_i)}{p(Y_i'|w, X_i)} \geq c,$$

leading to the set of constraints

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1. \tag{1}$$

Introducing the $\ell_2$-norm (squared) regularizer and the $N$-slack constraint violation penalty, we obtain the quadratic program

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t.\ \ \forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1 - \xi_i, \ \ \forall_i\ \xi_i \geq 0$$

This is known as a *hidden Markov support vector machine* [Altun et al., 2003, Joachims, 2003].

Because it is based on the $N$-slack formulation, the hidden Markov support vector machine penalizes all alternative hypothesis based on the most promising alternative hypothesis. In particular, the number of label differences between $Y_i$ and an alternative configuration $Y_i'$ is never considered. Rather than selecting the constant to be 1 across all alternative configurations, we could consider setting it to $\Delta(Y_i, Y_i')$, the number of label differences between $Y_i$ and $Y_i'$

$$\forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i'). \tag{2}$$

This encourages the likelihood ratio between the true label and an alternative configuration to be higher if the alternative configuration disagrees with $Y_i$ at many nodes. With our usual procedure we obtain the quadratic program

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t.\ \ \forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i') - \xi_i, \ \ \forall_i \xi_i \geq 0,$$

which is known as a *max-margin Markov network* [Taskar et al., 2003], or a *structural support vector machine* with *margin re-scaling* [Tsochantaridis et al., 2004].

Of course, this isn't the only way to change the constraints based on the agreement between $Y_i$ and $Y_i'$, and [Tsochantaridis et al., 2004] argues that this formulation has the disadvantage that it tries to make very large the difference between competing labels that differ in many positions. An alternative to changing the

constant factor in the likelihood ratio is to change the scaling of the slack variables and solve the quadratic program

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t.\ \ \forall_i \forall_{Y_i' \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq 1 - \xi_i / \Delta(Y_i, Y_i'), \ \ \forall_i \xi_i \geq 0.$$

Here, you need to add more 'slack' for a competing hypothesis that differs in many position. This last variant is known as a *structural support vector machine* with *slack re-scaling* [Tsochantaridis et al., 2004].

Equivalent unconstrained versions of these 3 structural extensions of support vector machines are[3]

$$\text{(HMSVM)} \qquad \min_w \sum_i \max_{Y_i' \neq Y_i} (1 - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))^+ + \lambda ||w||_2^2,$$

$$\text{(MMMN)} \qquad \min_w \sum_i \max_{Y_i' \neq Y_i} (\Delta(Y_i, Y_i') - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))^+ + \lambda ||w||_2^2,$$

$$\text{(SSVM)} \qquad \min_w \sum_i \max_{Y_i' \neq Y_i} (\Delta(Y_i, Y_i')(1 - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i)))^+ + \lambda ||w||_2^2.$$

In the (MMMN) and (SSVM) formulations, we can use $\Delta(Y_i, Y_i) = 0$ to simplify these expressions, giving

$$\text{(MMMN)} \qquad \min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i') + \log p(Y_i'|w, X_i)) - \log p(Y_i|w, X_i) + \lambda ||w||_2^2,$$

$$\text{(SSVM)} \qquad \min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i')(1 - \log p(Y_i|w, X_i) + \log p(Y_i'|w, X_i))) + \lambda ||w||_2^2.$$

N.B., in these latter two expressions we can maximize over all $Y_i'$ instead of over all $Y_i' \neq Y_i$; In these cases, we can compute the objective function using the Viterbi decoding algorithm (with a suitably modified input). In contrast, the HMSVM model requires solving a maximization problem over all configurations except $Y_i$, which be may be much harder to solve[4]. Finally, we note that while the structural hinge losses in all three formulations represent an upper bound on the number of sequence errors (ie. the number of times at least one label in the sequence was wrong), the MMMN and SSVM formulations give an upper bound on the number of label errors while the HMSVM does not.

Although this section has focused on a chain-structured dependency structure, we only used this property in two places: (i) the definition of $p(Y_i|w, X_i)$, and (ii) in efficient methods to compute maximums over the space of possible configurations. We can apply the same methodology to derive MMMNs and SSVMs for a wide variety of alternative models. For example, we can consider the cases where conditional random fields can be applied exactly, such as low tree-width structured dependencies, and probabilistic context free grammars. We can also consider models where we consider higher-order dependencies, rather than simply pairwise dependencies, as long as the graph structure or parameterization allows us to efficiently solve the associated maximization problems.

As discussed in [Taskar et al., 2004], an advantage that MMMNs/SSVMs have over conditional random fields is that we can also consider completely general graph structures (such as lattices or fully connected graphs), if we enforce the additional constraint that the edge weights enforce a sub-modularity condition [Kolmogorov and Zabin, 2004]. In these cases, the exact inference needed for training conditional random fields is intractable, but computing the maximums needed by MMMNs/SSVMs can be formulated as a linear program (that can be efficiently solved using min-cut techniques). [Taskar et al., 2004] proposes *associative max margin Markov networks*, models that take advantage of this property to allow efficient training and decoding in models with an arbitrary graph structure and arbitrary clique sizes. In these models, the

[3] We can also consider penalizing the squared values of the slack variables [Tsochantaridis et al., 2004], but this does not lead to a differentiable minimization problem due to the potential for ties in the max functions.

[4] We could also consider structural extensions based on the $NK$-slack formulation [Altun and Hofmann, 2003]. In this case, the max functions would be replaced by sums, but it is not obvious (to me, anyway) how you would compute these sums efficiently, nor how you would deal with the exponential number of variables and constraints in the quadratic program

potential that all variables in a clique take the state $k$ based on a non-negative feature $x_i$ is modeled with a parameter $\lambda_{k,i} \geq 1$, while the potential for any configuration where they don't take the same state is fixed at 1. For binary variables, the decoding problem can be solved exactly as a linear program for any graph structure, while for multi-class problems the linear program yields a worst-case approximation ratio of $1/c$ (where $c$ is the largest clique size), if an appropriate discretization strategy is used. [Taskar et al., 2006b] also gives the example of weighted bipartite graph matching, where decoding can be computed in polynomial time using linear programming, but the inference needed to train a conditional random field is $\#P$-hard.

For general graphs where we do not want the sub-modularity condition, we can also consider an approximation where we use an approximate decoding method, such as loopy belief propagation or a stochastic local search method [Hutter et al., 2005]). However, theoretical and empirical work in [Finley and Joachims, 2008] shows that using such *under-generating* decoding methods (which may return sub-optimal solutions to the decoding problem) have several disadvantages compared to so-called *over-generating* decoding methods. A typical example of an over-generating decoding method is a linear programming relaxation of an integer programming formulations of decoding, where in the relaxation the objective is optimized exactly over an expanded space [Kumar et al., 2007], hence yielding a fractional decoding that is a strict upper bound on the optimal decoding.

# 5 Training

In this section, we discuss methods for estimating the parameters of structural extensions of SVMs. We will focus on MMMNs since it makes the notation slightly simpler, but most of the techniques can be applied with only minor modifications to the case of SSVMs.

## 5.1 Subgradient Methods

We first consider methods that address the unconstrained non-differentiable formulation of MMMNs by moving along sub-gradients of the objective function

$$f(w) \triangleq \sum_i \max_{Y_i'}(\Delta(Y_i, Y_i') + \log p(Y_i'|w, X_i)) - \log p(Y_i|w, X_i) + \lambda||w||_2^2.$$

If we use $Y_i''$ as an argmax of the max, then a sub-gradient of this objective function is

$$g(w) \triangleq \sum_i \nabla_w \log p(Y_i''|w, X_i) - \nabla_w \log p(Y_i|w, X_i) + 2\lambda w.$$

The function is differentiable and this will be the gradient whenever all of the argmax values are unique. The negative sub-gradient direction is a descent direction at all points where the function is differentiable, which means that $f(w - \eta g(w)) < f(w)$ for sufficiently small $\eta > 0$. In contrast, at non-differentiable points (where at least one argmax is not unique) negative sub-gradient directions may not be descent directions [Bertsekas, 1999, §6.3]. However, we can guarantee that for sufficiently small $\eta$ the distance to the optimal solution is reduced, even if the function value is not [Bertsekas, 1999, §6.3]. We can therefore consider a simple sub-gradient method that uses a sequence of small step sizes $\{\eta_k\}$, where the iterations take of the form

$$w_{k+1} = w_k - \eta_k g(w_k).$$

Several strategies for choosing the step length that lead to convergence of the iterations are discussed in [Bertsekas, 1999, §6.3], with a common choice being a sequence $\{\eta_k\}$ of positive step sizes satisfying

$$\sum_{k=1}^{\infty} \eta_k = \infty, \quad \sum_{k=1}^{\infty} \eta_k^2 < \infty.$$

Convergence of the method can also be shown if each step uses the contribution to the sub-gradient of a suitably chosen individual training example $i$ [Kushner and Yin, 2003]

$$g_i(w) \triangleq \nabla_w \log p(Y_i''|w, X_i) - \nabla_w \log p(Y_i|w, X_i) + (2/N)\lambda w,$$

resulting in steps of the form

$$w_{k+1} = w_k - \eta_k g_i(w_k).$$

This is known as a stochastic sub-gradient descent method, and was presented in [Collins, 2002] (predating the formulation of structural SVMs) for the degenerate case of $\lambda = 0$ and $\forall_{Y_i, Y_i'} \Delta(Y_i, Y_i') = 0$, and where a constant step size ($\eta_k = \eta$) was used instead of a convergent sequence.

An interesting alternative method was also outlined in [Collins, 2002]; In this second method a constant step size was used, and a variant on the procedure was considered where the final estimator is the average of all the iterates;

$$w_{k+1} = w_k - \eta g_i(w_k),$$
$$\tilde{w}_{k+1} = \frac{k-1}{k}\tilde{w}_k + \frac{1}{k}w_{k+1}.$$

Convergence of averaged estimators of the form $\tilde{w}$ is discussed in [Kushner and Yin, 2003], where it is shown that under certain conditions these steps satisfy an asymptotic statistical efficiency property.

In the special case of binary support vector machines (though easily extended to structural support vector machines), [Shalev-Shwartz et al., 2007] considers iterations of the form

$$w_{k+1} = \pi(w_k - \eta_k g(w_k)),$$

where $\pi$ is a projection onto an $\ell_2$-ball that is guaranteed to contain the solution[5]. They showed that the number of iterations to obtain a solution with accuracy $\epsilon$ using this method is $\mathcal{O}(1/\epsilon)$ (ignoring poly-logarithmic factors), which was an improvement over the $\mathcal{O}(1/\epsilon^2)$ number of iterations provided by the previous analysis of the averaged stochastic gradient method [Zhang, 2004] and cutting plane methods [Joachims, 2006]. [Shalev-Shwartz et al., 2007] also consider the case of a projected stochastic gradient update, and show that the iterations achieve an accuracy of $\epsilon$ with probability $1 - \delta$ in $\mathcal{O}(1/(\delta\epsilon))$ iterations. One could also consider combining averaging and projections, whose asymptotic statistical efficiency is considered in [Kushner and Yin, 2003]. Finally, [Ratliff et al., 2007] show that the (deterministic) subgradient method with a sufficiently small constant step size will converge linearly (ie. with the faster rate of $\mathcal{O}(\log(1/\epsilon))$) to a region containing the minimum whose radius is proportional to $C/\lambda$ (where $C$ is a bound on the sub-differential).

## 5.2 Cutting Plane and Bundle Methods

We now consider methods for solving the quadratic programming formulation of MMMNs

$$\min_{w,\xi} \sum_i \xi_i + \lambda||w||_2^2,$$

$$s.t.\ \ \forall_i \forall_{Y_i'} \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i') - \xi_i, \ \ \forall_i \xi_i \geq 0.$$

This formulation is problematic since we have an exponential number of constraints. However, it is conceivable that we do not need to enforce all of these constraints at the optimal solution. In particular, [Tsochantaridis et al., 2004] showed that there always exists a polynomial-sized subset of the constraints that can satisfy all constraints with accuracy at least $\epsilon$. This makes cutting-plane methods for quadratic programming an appealing approach for solving this type of problem.

Cutting plane methods have a very simple structure; We first find the unconstrained minimizer. If the minimizer does not satisfy all the constraints, we find a constraint that is violated and use it to construct our

[5]The projection of a vector $w$ onto a set $\mathcal{W}$ is defined as $\min_{w^* \in W} ||w - w^*||_2$

initial working set $\mathcal{W}$. We then solve the problem with the constraints contained in $\mathcal{W}$, and if this solution still does not satisfy all the constraints we add another violating constraint to $\mathcal{W}$. This continues until we have accumulated enough constraints in $\mathcal{W}$ so that all constraints are satisfied in the solution.

[Tsochantaridis et al., 2004] suggests the following implementation of a cutting plane algorithm for MMMNs. We iterate over the training examples, and use Viterbi decoding to find the maximum structural hinge loss. If no constraint is violated by more than $\epsilon$ then we move on to the next training example. Otherwise, we add the constraint corresponding to this alternative configuration, and optimize the objective function subject to our current working set of constraints. The algorithm terminates when we complete a full pass through the training data without adding any constraints.

The quadratic programs can be efficiently solved using the Wolfe dual formulation (see the next section), which is a relatively small problem since it has only one variable for each working constraint. Further, the sequence of quadratic programs can be solve efficiently since each quadratic program differs from the previous one by only a single constraint. The main concern with using this type of approach is that the number of constraints needed might be so large that the intermediate quadratic programs may be impractical to solve. [Tsochantaridis et al., 2004] show that at most $\mathcal{O}(1/\epsilon^2)$ constraints are required[6].

This cutting plane algorithm for structural SVMs later inspired an efficient algorithm for binary support vector machines [Joachims, 2006]. In this method, the binary support vector machine problem is written in a so-called 1-slack formulation

$$\min_{w,\xi} N\xi + \lambda||w||_2^2,$$

$$s.t.\ \forall_{c\in\{0,1\}^N}\ \sum_i c_i y_i w^T x_i \geq \sum_i c_i - N\xi,\ \ \xi \geq 0.$$

Here, we have an exponential number of constraints but only 1 slack variable. [Joachims, 2006] shows that this formulation has the same optimal parameters (with $\xi = (1/N)\sum_i \xi_i$), and outlines a cutting plane algorithm for binary SVMs that is implemented in the SVMperf software[7]. This method was extended (back) to structural SVMs in [Joachims et al., 2009], where it was shown that at most $\mathcal{O}(1/\epsilon)$ constraints are required. This is the algorithm implemented in the current version of the SVMstruct software[8].

An alternative to applying cutting plane methods for quadratic programming to a constrained formulation of the MMMN problem is to apply the related cutting plane methods for non-smooth optimization to the unconstrained non-differentiable MMMN problem. This is explored in [Teo et al., 2007, Smola et al., 2008].

In cutting plane methods for optimization of (non-smooth) convex functions, we use the definition of a sub-gradient at a point $w_0$ to yield a global lower bound on the function

$$f(w) \geq f(w_0) + (w - w_0)^T g(w_0),$$

that is exact at $w_0$. Cutting plane methods for non-smooth optimization proceed by collecting a set of these lower bounding hyper-planes, and at each iteration generate a new trial point by finding the minimum of these lower bounds. The function value and sub-gradient at the new trial point are then added to the working set, yielding a more accurate lower bound.

A disadvantage of this procedure is that the minimum of the set of lower bounds may suggest a trial point that is far away from the previous point, even if the previous point was close to the optimal solution. This is known as instability [Bertsekas, 1999, §6.3]. A common way to address this is to minimize the lower bound subject to a regularizer $||w_{k+1} - w_k||_2^2$ on the difference between subsequent trial values. This is known as a *bundle method*. The method of [Teo et al., 2007, Smola et al., 2008] is closely related to bundle methods, but differs in the regularizer. Rather than building a lower bound for the full objective function and using the regularizer $||w_{k+1} - w_k||_2^2$, they build a sequence of lower bounds for the hinge loss and use the exact regularizer $\lambda||w_|||_2^2$ already present in the objective function. In [Smola et al., 2008], it is shown that

[6] A closely related method and analysis for HMSVMs is in [Joachims, 2003]

[7] `http://svmlight.joachims.org/svm_perf.html`

[8] `http://svmlight.joachims.org/svm_struct.html`

the method requires $\mathcal{O}(1/\epsilon)$ iterations to converge[9], and that the method offers a speed advantage over the method of [Shalev-Shwartz et al., 2007].

### 5.3 Polynomial-Size Reformulations

In this section, we examine an alternative to cutting plane methods for dealing with the exponential sized quadratic programs. In particular, we consider methods that take advantage of the sparse dependency structure in the underlying distribution to rewrite the exponential-sized quadratic program into a polynomial-sized problem. In this section, we will largely follow [Taskar et al., 2003] and will work with a dual formulation to the MMMN quadratic program. In this dual formulation, we will have a variable $\alpha_i(Y_i')$ for each possible configuration $Y_i'$ of training example $i$. We will also find it convenient to use the feature representation of the probability functions

$$p(Y_i|w, X_i) \propto w^T F(X_i, Y_i),$$

and use the notation

$$\Delta F_i(Y_i') \triangleq F(X_i, Y_i) - F(X_i, Y_i').$$

Using this notation, [Taskar et al., 2003] shows that the following problem is dual to the MMMN quadratic program

$$\max_{\alpha} \sum_i \sum_{Y_i'} \alpha_i(Y_i')\Delta(Y_i, Y_i') - \frac{1}{2}\sum_i \sum_{Y_i'} \sum_j \sum_{Y_j'} \alpha_i(Y_i')\alpha_j(Y_j')\Delta F_i(Y_i')^T \Delta F_j(Y_j'),$$

$$s.t.\ \forall_i \sum_{Y_i'} \alpha_i(Y_i') = \frac{1}{2\lambda},\ \ \forall_i \forall_{Y_i'} \alpha_i(Y_i') \geq 0.$$

Although this dual formulation has an exponential number of constraints *and* an exponential number of variables, note the simple form of the constraints. If $\lambda = 1/2$, then the constraints simply enforce that $\alpha_i(Y_i')$ is a valid probability distribution over all values of $Y_i'$ (for each training example $i$). For other values of $\lambda$, it enforces that $\alpha_i(Y_i')$ is an unnormalized distribution (with normalizing constant $1/2\lambda$). If our dependency structure factorizes over nodes and edges in a graph, then we can consider a parameterization in terms of expectations with respect to this distribution

$$\mu_i(y_{ij}) = \sum_{Y_i' \sim [y_{ij}]} \alpha_i(Y_i'),$$

$$\mu_i(y_{ij}, y_{ik}) = \sum_{Y_i' \sim [y_{ij}, y_{ik}]} \alpha_i(Y_i'),$$

where the notation $Y_i' \sim [y_{ij}]$ is used to denote that a configuration $Y_i'$ is consistent with the label $y_{ij}$. When we consider using these variables, we will require that the node parameters satisfy the constraints of the original problem

$$\forall_i \forall_j\ \mu_i(y_{ij}) \geq 0,\ \ \forall_i\ \sum_j \mu_i(y_{ij}) = \frac{1}{2\lambda}.$$

To ensure that the edge variables form a legal density, they must belong to the marginal polytope. In the case of tree-structured graphs, this is equivalent to the requirement that [Taskar et al., 2003]

$$\forall_i \forall_{(j,k)\in E} \sum_{y_i j} \mu_i(y_{ij}, y_{ik}) = \mu_i(y_{ik}),\ \ \forall_i \forall_{(j,k)\in E}\ \mu_i(y_{ij}, y_{ik}) \geq 0.$$

Using these expectations, we can re-write the set of first terms in the dual formulation as

$$\sum_{Y_i'} \alpha_i(Y_i')\Delta(Y_i, Y_i') = \sum_{Y_i'} \sum_j \alpha_i(Y_i')\Delta_j(y_{ij}, y_{ij}') = \sum_j \sum_{y_{ij}'} \Delta_j(y_{ij}, y_{ij}')\mu_i(y_{ij}').$$

[9] [Smola et al., 2008] also show that applying their method to train conditional random fields with $\ell_2$-regularization achieves the faster rate of $\mathcal{O}(\log(1/\epsilon))$.

This is now a sum over individual label assignments rather than over joint assignments. We can similarly re-write the second term in the dual in terms of edge expectations. This leads to what [Taskar et al., 2003] calls the factored dual

$$\max_{\mu} \sum_i \sum_j \sum_{y'_{ij}} \Delta_j(y_{ij}, y'_{ij}) \mu_i(y'_{ij}) -$$

$$\frac{1}{2} \sum_i \sum_{i'} \sum_{(j,k) \in E} \sum_{(j',k') \in E} \sum_{y'_{ij}, y'_{ik}} \sum_{y'_{i'j'}, y'_{i'k'}} \mu_i(y'_{ij}, y'_{ik}) \mu_{i'}(y'_{i'j'}, y'_{i'k'}) F_i(X_i, y'_{ij}, y'_{ik})^T F_i(X_i, y'_{i'j'}, y'_{i'k'}),$$

$$s.t. \;\; \forall_i \forall_j \; \mu_i(y_{ij}) \geq 0, \;\; \forall_i \; \sum_j \mu_i(y_{ij}) = \frac{1}{2\lambda},$$

$$\forall_i \forall_{(j,k) \in E} \sum_{y_i j} \mu_i(y_{ij}, y_{ik}) = \mu_i(y_{ik}), \;\; \forall_i \forall_{(j,k) \in E} \; \mu_i(y_{ij}, y_{ik}) \geq 0.$$

While this formulation is now polynomial-size, the quadratic form has a very large number of terms (ie. it is quadratic in the number of training examples and instances). [Taskar et al., 2003] proposes a method, known as the structural SMO method, for solving this optimization problem. In this method, violation of the constrained optimality conditions is used to select the two most violating variables, and their optimal value given all the other variables is computed in closed form to update the variables.

[Taskar et al., 2003] discusses extensions of the above method to graph structures that are not forests. For triangulated graphs, additional constraints can be imposed to ensure that the distribution remains in the marginal polytope, although the number of additional constraints is exponential in the worst case. If we have an untriangulated graph or do not want to triangulate and add these extra constraints, the factored dual can be solved as an approximation (analogous to loopy belief propagation). It is also possible to derive factored duals when the original distribution factorizes over higher-order cliques. Finally, [Taskar et al., 2003] discusses a factored version of the primal problem, where we have an extended set of slack variables that mimic the dependency structure in the graph and constrain dual variables enforcing constraints on the marginal polytope.

An alternative method for optimizing the dual formulation using a polynomial-sized number of variables is outlined in [Bartlett et al., 2005, Collins et al., 2008]. In this method, the exponentiated gradient algorithm is used to (indirectly) optimize the variables $\alpha$ in the dual formulation. The exponentiated gradient algorithm [Kivinen and Warmuth, 1997] is a method to optimize a function $f(x)$, where $x$ lies on the probability simplex[10]

$$\forall_i x_i \geq 0, \;\; \sum_i x_i = 1.$$

The exponentiated gradient algorithm takes steps of the form

$$x_i = \frac{x_i \exp(-\eta \nabla_i f(x))}{\sum_{i'} x_{i'} \exp(-\eta \nabla_{i'} f(x))},$$

where as before $\eta$ is a learning rate, and we can consider applying the algorithm using the contribution to the gradient of one training example at a time instead of all the training examples at once.

[Bartlett et al., 2005, Collins et al., 2008] work with a slightly modified dual formulation where the constraints take the form of a probability simplex (instead of an unnormalized version), and apply the exponentiated gradient algorithm to this problem. To avoid dealing with the exponential number of variables $\alpha$, they use an implicit representation for $\alpha$ in terms of variables that are analogous to the $\mu$ variables above. Similar to the methods from previous sections, they show that the method requires $\mathcal{O}(1/\epsilon)$ iterations to reach an accuracy of $\epsilon$, and that this applies whether the full gradient or the gradient contribution from an individual

[10] [Kivinen and Warmuth, 1997] also outline an extension that can handle non-negative variables subject to the $\ell_1$-norm of the vector being 1, and extensions where the constant 1 is replaced by another known constant $C$.

training example is used[11]. A disadvantage of this optimization procedure is that using the implicit representation of $\alpha$ requires inference in the underlying probabilistic model. For example, running belief propagation to compute marginals in tree-sturctured graphs. Since the inference problem is strictly harder than the decoding problem needed by sub-gradient/cutting-plane methods, the exponentiated gradient method seems to only apply to a subset of the problems where these other methods apply.

## 5.4 Min-Max Formulations

Finally, the last set of methods that we consider deals with the exponential number of constraints in the quadratic program by replacing them with one non-linear constraint for each training example.

$$\min_{w,\xi} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t.\ \ \forall_i \log p(Y_i|w, X_i) + \xi_i \geq \max_{Y_i'} \log p(Y_i'|w, X_i) + \Delta(Y_i, Y_i'),\ \ \forall_i \xi_i \geq 0.$$

Above, we re-arranged the constraint so that the right-hand side of the inequality is the solution to the modified decoding problem.

We first consider a method to solve this *min-max* formulation that is due to [Taskar et al., 2004], that applies in the special case where the optimal decoding can be formulated as a linear program. In this method, we write the linear program that solves the problem $\max_{Y_i'} \log p(Y_i'|w, X_i) + \Delta(Y_i, Y_i')$ in standard form

$$\max_z \ \ wBz \ \ s.t.\ \ z \geq 0,\ \ Az \leq b,$$

for suitably defined values of $A$, $B$, and $b$. We obtain an equivalent minimization problem by working with the dual of this linear program

$$\min_z \ \ b^T z \ \ s.t.\ \ z \geq 0,\ \ A^T z \geq (wB)^T.$$

We then substitute this dual problem into our original problem to eliminate the non-linear constraint, yielding the quadratic program

$$\min_{w,\xi_z} \sum_i \xi_i + \lambda ||w||_2^2,$$

$$s.t.\ \ \forall_i \log p(Y_i|w, X_i) + \xi_i \geq b^T z,\ \ \forall_i \xi_i \geq 0,\ \ z \geq 0,\ \ A^T z \geq (wB)^T.$$

Subsequently, we can solve this (reasonably-sized) quadratic program to exactly solve the original quadratic program whenever the optimal decoding can be formulated as a linear program. In cases where the linear programming relaxation is not exact, we can still use this formulation to compute an approximate solution, where the slack variables will be increased for any training examples where a fractional decoding yields a better configuration than the best integer decoding.

An alternative min-max formulation was explored in [Taskar et al., 2006b]. In this formulation, we plug a linear programming problem that solves the decoding problem into the unconstrained version of the MMMN problem, giving the saddle-point problem

$$\min_{w \in \mathcal{W}} \max_{z \in \mathcal{Z}} \ \ \lambda ||w||_2^2 + \sum_i w^T F_i z_i + c_i^T z_i - w^T F(X_i, Y_i),$$

for appropriately defined $\mathcal{Z}$, $F_i$ and $c_i$. Subsequently, the extragradient algorithm is applied to find a value of $w$ and $z$ that minimizes over $w$ the maximizing value of $z$. Using $L(w, z)$ to denote the objective function in

[11] [Collins et al., 2008] also shows that the faster of rate of $\mathcal{O}(\log(1/\epsilon))$ is achieved when the method is applied to optimizing a dual formulation of training conditional random fields.

the saddle point problem, the iterations of the extragradient algorithm take the form of alternating projected gradient steps in $w$ and $z$;

$$\begin{aligned} w^p &= \pi_{\mathcal{W}}(w - \eta \nabla_w L(w, z)), \\ z_i^p &= \pi_{\mathcal{Z}}(z_i + \eta \nabla_{z_i} L(w, z)), \\ w^c &= \pi_{\mathcal{W}}(w - \eta \nabla_w L(w^p, z^p)), \\ z_i^c &= \pi_{\mathcal{Z}}(z_i + \eta \nabla_{z_i} L(w^p, z^p)). \end{aligned}$$

The step size $\eta$ is selected by a backtracking line search until a suitable condition on $w^p$ and $z^p$ is satisfied [Taskar et al., 2006b]. The iterations converge linearly to the solution, which corresponds to a rate of $\mathcal{O}(\log(1/\epsilon))$. Since $w$ is unconstrained, the projection $\pi_{\mathcal{W}}$ is simply the identity function (for associative max-margin Markov networks, we project the edge parameters onto the non-negative orthant), while the projection $\pi_{\mathcal{Z}}$ is the solution of a min-cost quadratic flow problem [Taskar et al., 2006b]. In [Taskar et al., 2006a], a dual extragradient method is considered, along with an extension that uses proximal steps with respect to a Bregman divergence rather than Euclidean projections.

### 5.5 Comments

As the discussion above indicates, efficiently solving the structural support vector machine optimization remains a challenging problem. Currently, the sub-gradient descent methods are one appealing choice because of their simplicity, while the cutting plane methods are another appealing option due to the availability of efficient implementations of these methods. However, the $\mathcal{O}(1/\epsilon)$ (sub-linear) worst-case convergence rates of these methods is somewhat unappealing. While the $\mathcal{O}(\log(1/\epsilon))$ (linear) convergence speed of the extragradient method is more appealing, it does not approach the $\mathcal{O}(\log(\log(1/\epsilon)))$ convergence rates of standard methods for solving quadratic programs. We could consider applying interior-point methods (see [Boyd and Vandenberghe, 2004], for example) to one of the polynomial-sized reformulations or one of the min-max formulations, but the number of variables/constraints in these problems would likely make iterations of the method impractical.

## References


[Altun and Hofmann, 2003] Altun, Y. and Hofmann, T. (2003). Large margin methods for label sequence learning. In *Eighth European Conference on Speech Communication and Technology*. ISCA.

[Altun et al., 2003] Altun, Y., Tsochantaridis, I., Hofmann, T., et al. (2003). Hidden markov support vector machines. In *ICML*, volume 20, page 3.

[Bartlett et al., 2005] Bartlett, P., Collins, M., Taskar, B., and McAllester, D. (2005). Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems 17: Proceedings Of The 2004 Conference*, page 113. MIT Press.

[Bertsekas, 1999] Bertsekas, D. (1999). Nonlinear programming. *Athena Scientific*.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). Convex Optimization. 2004.

[Collins, 2002] Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics Morristown, NJ, USA.

[Collins et al., 2008] Collins, M., Globerson, A., Koo, T., Carreras, X., and Bartlett, P. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *The Journal of Machine Learning Research*, 9:1775–1822.

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

[Dietterich and Bakiri, 1995] Dietterich, T. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Arxiv preprint cs.AI/9501101*.

[Finley and Joachims, 2008] Finley, T. and Joachims, T. (2008). Training structural SVMs when exact inference is intractable. In *Proceedings of the 25th international conference on Machine learning*, pages 304–311. ACM New York, NY, USA.

[Hutter et al., 2005] Hutter, F., Hoos, H., and Stutzle, T. (2005). Efficient stochastic local search for MPE solving. In *INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 19, page 169. LAWRENCE ERLBAUM ASSOCIATES LTD.

[Joachims, 2003] Joachims, T. (2003). Learning to align sequences: A maximum-margin approach. online manuscript.

[Joachims, 2006] Joachims, T. (2006). Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM New York, NY, USA.

[Joachims et al., 2009] Joachims, T., Finley, T., and Yu, C. (2009). Cutting plane training of structural SVMs. *Machine Learning*.

[Kivinen and Warmuth, 1997] Kivinen, J. and Warmuth, M. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*.

[Kolmogorov and Zabin, 2004] Kolmogorov, V. and Zabin, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159.

[Kumar et al., 2007] Kumar, M., Kolmogorov, V., and Torr, P. (2007). An analysis of convex relaxations for MAP estimation. *Advances in Neural Information Processing Systems*, 20:1041–1048.

[Kushner and Yin, 2003] Kushner, H. and Yin, G. (2003). *Stochastic approximation and recursive algorithms and applications*. Springer.

[Lacoste-Julien, 2003] Lacoste-Julien, S. (2003). Combining SVM with graphical models for supervised classification: an introduction to Max-Margin Markov Network.

[Lafferty et al., 2001] Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289.

[Lee and Mangasarian, 2001] Lee, Y. and Mangasarian, O. (2001). SSVM: A smooth support vector machine for classification. *Computational optimization and Applications*, 20(1):5–22.

[Ratliff et al., 2007] Ratliff, N., Bagnell, J. A. D., and Zinkevich, M. (2007). (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*.

[Shalev-Shwartz et al., 2007] Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM New York, NY, USA.

[Smola et al., 2008] Smola, A., Vishwanathan, S., and Le, Q. (2008). Bundle methods for machine learning. *Advances in Neural Information Processing Systems*, 20.

[Taskar et al., 2004] Taskar, B., Chatalbashev, V., and Koller, D. (2004). Learning associative Markov networks. In *Proceedings of the twenty-first international conference on Machine learning*. ACM New York, NY, USA.

[Taskar et al., 2003] Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin Markov networks. *Advances in neural information processing systems*, 16:51.

[Taskar et al., 2006a] Taskar, B., Lacoste-Julien, S., and Jordan, M. (2006a). Structured prediction, dual extragradient and Bregman projections. *The Journal of Machine Learning Research*, 7:1627–1653.

[Taskar et al., 2006b] Taskar, B., Lacoste-Julien, S., and Jordan, M. (2006b). Structured prediction via the extragradient method. *Advances in neural information processing systems*, 18:1345.

[Teo et al., 2007] Teo, C., Smola, A., Vishwanathan, S., and Le, Q. (2007). A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM New York, NY, USA.

[Tsochantaridis et al., 2004] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*. ACM New York, NY, USA.

[Vapnik, 1995] Vapnik, V. (1995). The nature of statistical learning theory.

[Weston and Watkins, 1999] Weston, J. and Watkins, C. (1999). Support vector machines for multi-class pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, volume 4.

[Zhang, 2004] Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*. ACM New York, NY, USA.