Unsupervised and Semi-supervised Multi-class Support Vector Machines

Linli Xu[*]
School of Computer Science
University of Waterloo

Dale Schuurmans
Department of Computing Science
University of Alberta

Abstract

We present new unsupervised and semi-supervised training algorithms for multi-class support vector machines based on semidefinite programming. Although support vector machines (SVMs) have been a dominant machine learning technique for the past decade, they have generally been applied to *supervised* learning problems. Developing unsupervised extensions to SVMs has in fact proved to be difficult. In this paper, we present a principled approach to unsupervised SVM training by formulating convex relaxations of the natural training criterion: find a labeling that would yield an optimal SVM classifier on the resulting training data. The problem is hard, but semidefinite relaxations can approximate this objective surprisingly well. While previous work has concentrated on the two-class case, we present a general, *multi-class* formulation that can be applied to a wider range of natural data sets. The resulting training procedures are computationally intensive, but produce high quality generalization results.

Introduction

Efficient convex optimization techniques have had a profound impact on the field of machine learning. Most of their use to date, however, has been in applying *quadratic* programming techniques to support vector machine (SVM) and kernel machine training (Schoelkopf & Smola 2002). Nevertheless, one lesson from the success of SVMs is that effective algorithmic approaches can lead to new progress in generalization, estimation and modeling issues, even when these are not directly algorithmic questions. The convenience of effective algorithms allows researchers to freely explore generalization ideas in the context of complex models and large data collections.

Currently, new opportunities for developing novel machine learning techniques are offered by the field of semidefinite programming (SDP). Semidefinite programming, and convex programming more generally, significantly extend the toolbox of optimization methods used in machine learning, beyond the current unconstrained, linear and quadratic programming techniques. Recent progress in semidefinite programming has yielded a viable technology that has efficiency characteristics similar to quadratic programming (Boyd & Vandenberghe 2004), including polynomial runtime guarantees (Nesterov & Nimirovskii 1994). These techniques offer a new avenue for solving problems of relevance to machine learning.

In fact, semidefinite programming has already started to prove its utility in machine learning research. Lanckreit *et al*. (2004) show how semidefinite programming can be used to optimize the kernel matrix for a supervised SVM. Xu *et al*. (2004) and De Bie & Cristanini (2003) develop new unsupervised and semi-supervised training techniques for SVMs based on semidefinite programming. Similar techniques are used in the correlation clustering approach of (Bansal, Blum, & Chawla 2002). Graepel and Herbrich (2003) have shown how transformational invariants can be encoded in SVMs. Even though the number of such results remains modest, the initial achievements are impressive.

In this paper we provide a brief introduction to the application of semidefinite programming in machine learning and contribute further progress by extending the work of (Xu *et al.* 2004; De Bie & Cristianini 2003) to *multi-class* SVMs. Multi-class SVMs are similar to their two-class predecessors in that they can be trained by quadratic programming in the *supervised* case (Crammer & Singer 2001). However, developing *unsupervised* and *semi-supervised* multi-class SVMs requires a semidefinite programming formulation that significantly extends the earlier two-class approaches.

After reviewing some of the relevant background, we present new semidefinite programming algorithms for unsupervised training and semi-supervised training of SVMs in the multi-class case. We then present experiments that show semidefinite training can obtain state of the art generalization performance, albeit at a greater computational cost than previous techniques. Further algorithmic developments in convex programming methods will hopefully continue to alleviate some of these costs in the future.

Semidefinite programming

Semidefinite programming is a recent extension of linear and quadratic programming which was significantly advanced by (Nesterov & Nimirovskii 1994), who provided effective algorithmic foundations as well as polynomial runtime guarantees for solution techniques based on interior

[*] Work performed at the Alberta Ingenuity Centre for Machine Learning, University of Alberta. {linli,dale}@cs.ualberta.ca

point methods. Semidefinite programming has since had a profound impact on the field of combinatorial optimization (Goemans & Williamson 1995), and, as mentioned, has opened up exciting new avenues of research in machine learning (Lanckriet *et al.* 2004). Before explaining the direct relevance of these formulations to generalized SVM training, we first briefly review the basic concepts.

The statement of a semidefinite programming problem, as well as the description of the interior point solution approach, are both very simple. A semidefinite programming problem is a *convex* constrained optimization problem where one optimizes a symmetric $n \times n$ matrix of variables X

$$\min_X \langle C, X \rangle \quad \text{subject to} \quad \langle A_i, X \rangle = b_i, \;\; i = 1...m$$
$$X \succeq 0 \qquad (1)$$

where $\langle \cdot, \cdot \rangle$ denotes the simple matrix inner product, $\langle C, X \rangle = \sum_{ij} c_{ij} x_{ij}$, and $X \succeq 0$ denotes the constraint that X must remain positive semidefinite (Helmberg 2000; Boyd & Vandenberghe 2004). Here, the objective is linear in X, and the semidefinite constraint simply means that X must satisfy $\mathbf{z}^\top X \mathbf{z} \geq 0$ for any vector $\mathbf{z} \neq \mathbf{0}$. For a symmetric X, this is true if and only if X has only nonnegative eigenvalues (Strang 1998). Since $\mathbf{z}^\top X \mathbf{z} = \langle X, \mathbf{z}\mathbf{z}^\top \rangle$, one can see that a semidefinite program is a generalized form of *linear* program on X with infinitely many linear constraints

$$\min_X \langle C, X \rangle \quad \text{subject to} \quad \langle A_i, X \rangle = b_i, \;\; i = 1...m$$
$$\langle X, \mathbf{z}\mathbf{z}^\top \rangle \geq 0, \;\; \forall \mathbf{z} \neq \mathbf{0}$$

Of course the infinitely many linear constraints in fact define a non-linear constraint on X (a semi-infinite program). Nevertheless, the semidefinite constraint is still a *convex* constraint on X. That is, for any two positive semidefinite matrices X and Y, any convex combination will be positive semidefinite, since $\mathbf{z}^\top(\rho X + (1-\rho)Y)\mathbf{z} = \rho \mathbf{z}^\top X \mathbf{z} + (1-\rho)\mathbf{z}^\top Y \mathbf{z} \geq 0$ for $0 \leq \rho \leq 1$. The dual form of (1) is

$$\max_{\mathbf{y}, Z} \mathbf{b}^\top \mathbf{y} \quad \text{subject to} \quad Z + \textstyle\sum_i y_i A_i = C, \;\; Z \succeq 0$$

(Helmberg 2000; Boyd & Vandenberghe 2004). Below we will see how this type of problem can naturally arise when one attempts to generalize SVM training.

Convex optimization problems generally admit effective algorithmic approaches (Nesterov & Nimirovskii 1994). It has become apparent that interior point (or "barrier") methods are one of the most effective approaches for solving these problems (Boyd & Vandenberghe 2004; Helmberg 2000; Vanderbei 1996). An appealing aspect of popular barrier methods, beyond their effectiveness, is that they are also very intuitive: To handle inequality constraints, one simply replaces them with a convex *barrier* function that ensures the constraint is satisfied. Doing so removes the inequality constraints from the problem and replaces them with convex terms in the optimization objective, yielding a problem that can be easily solved—for example by using Newton's method. In the case of a semidefinite constraint $X \succeq 0$ there is a particularly elegant barrier function: $-\log(\det(X))$. To see how this works, note that for a symmetric matrix X, $\det(X) = \prod_{i=1}^n \lambda_i$, the product of eigenvalues (Strang 1998). Thus, $-\log(\det(X)) = \sum_{i=1}^n \log(1/\lambda_i)$. So as *any* eigenvalue approaches 0 the barrier function goes to infinity and prevents the constraint $X \succeq 0$ from being violated. Crucially, the barrier function $-\log(\det(X))$ is also a convex function of X.

Therefore, an overall interior point (path following) method for solving semidefinite programs is constructed roughly as follows. First, a tightness parameter μ is set to ensure numerical stability. One then solves the barrier formulation of (1), which is a fully convex optimization problem

$$\min_X \langle C, X \rangle - \mu \log(\det(X)) \quad \text{subject to}$$
$$\langle A_i, X \rangle = b_i, \;\; i = 1...m \qquad (2)$$

For a given μ, this can be efficiently solved using Newton's method, in space linear in the size of the original problem (Helmberg 2000). Then μ is reduced by a fixed multiplier, typically $\mu^{(k+1)} = \mu^{(k)}/10$, and the process is repeated. It can be shown that as $\mu \to 0$ the solution to (2) approaches (1) (Boyd & Vandenberghe 2004). In practice, the iteration is stopped at a small value of μ.

In fact, there already exist many software packages on the Web for solving semidefinite programming problems using barrier methods, including SeDuMi, SDPT3 and CSDP (see www.optimization-online.org). Although these techniques are not as well developed as methods for solving quadratic programs, progress is continuing and the current tools are starting to become adequate for solving practical problems.

Two-class SVM training algorithms

We first briefly review the results that have been obtained for generalized two-class SVM training. First, to establish the background ideas from SVMs as well as establish the notation we will use, we consider the *supervised* case.

Assume we are given labeled training examples $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$ where each example is assigned a binary label $y^i \in \{-1, +1\}$. The goal of an SVM of course is to find the linear discriminant $f_{\mathbf{w},b}(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}) + b$ that maximizes the minimum misclassification margin

$$\gamma^* \;=\; \max_{\mathbf{w}, b, \gamma} \gamma \quad \text{subject to} \quad y^i(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^i) + b) \geq \gamma \;\; \forall i$$
$$\|\mathbf{w}\|_2 = 1 \qquad (3)$$

Here the Euclidean normalization constraint on $\mathbf{w}$ ensures that the Euclidean distance between the data and the separating hyperplane (in $\boldsymbol{\phi}(\mathbf{x})$ space) determined by $\mathbf{w}^*, b^*$ is maximized. It is easy to show that this same $\mathbf{w}^*, b^*$ is a solution to the quadratic program

$$\gamma^{*-2} = \min_{\mathbf{w}, b} \tfrac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to} \quad y^i(\mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}^i) + b) \geq 1 \;\forall i$$

Importantly, the minimum value of this quadratic program, γ^{*-2}, is just the inverse square of the optimal solution value γ^* to (3) (Lanckriet *et al.* 2004).

To cope with potentially inseparable data, one normally introduces slack variables to reduce the dependence on noisy examples. This leads to the so called soft margin SVM (and its dual) which is controlled by a tradeoff parameter β

$$\begin{aligned}\gamma^{*-2} \;&=\; \min_{\mathbf{w},b,\boldsymbol{\xi}} \;\tfrac{\beta}{2}\|\mathbf{w}\|^2 + \boldsymbol{\xi}^\top\mathbf{e} \quad \text{subject to} \quad \boldsymbol{\xi} \geq 0, \\ &\qquad\qquad y^i(\mathbf{w}^\top\boldsymbol{\phi}(\mathbf{x}^i) + b) \geq 1 - \xi_i \;\; \forall i \\ &=\; \max_{\boldsymbol{\lambda}} \;\boldsymbol{\lambda}^\top\mathbf{e} - \tfrac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top\rangle \quad \text{subject to} \\ &\qquad\qquad 0 \leq \boldsymbol{\lambda} \leq 1,\; \boldsymbol{\lambda}^\top y = 0 \end{aligned} \tag{4}$$

The notation we use in this second (dual) formulation requires some explanation, since we will use it below: Here K denotes the $n \times n$ kernel matrix formed from the inner products of feature vectors $\Phi = [\boldsymbol{\phi}(\mathbf{x}^1), ..., \boldsymbol{\phi}(\mathbf{x}^n)]$ such that $K = \Phi^\top\Phi$. Thus $K_{ij} = \boldsymbol{\phi}(\mathbf{x}^i)^\top\boldsymbol{\phi}(\mathbf{x}^j)$. The vector $\mathbf{e}$ denotes the vector of all 1 entries. We let $A \circ B$ denote componentwise matrix multiplication. Note that (4) is derived from the standard dual SVM by using the fact that $\boldsymbol{\lambda}^\top(K \circ \mathbf{y}\mathbf{y}^\top)\boldsymbol{\lambda} = \langle K \circ \mathbf{y}\mathbf{y}^\top, \boldsymbol{\lambda}\boldsymbol{\lambda}^\top\rangle = \langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top\rangle$.

Thus for supervised SVM training, one takes a given set of labeled training data $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$, forms the kernel matrix K on data inputs, forms the kernel matrix $\mathbf{y}\mathbf{y}^\top$ on target outputs, sets the slack parameter β, and solves the quadratic program (4) to obtain the dual solution $\boldsymbol{\lambda}^*$ and the inverse square maximum margin value γ^{*-2}. Once these are obtained, one can then recover a classifier directly from $\boldsymbol{\lambda}^*$ (Schoelkopf & Smola 2002).

Unsupervised two-class SVMs

Recently it has been observed that semidefinite programming can be used for *unsupervised* training of two-class SVMs (Xu *et al.* 2004; De Bie & Cristianini 2003). The goal in this case is not to find a large margin classifier given labels on the data, but instead to find a *labeling* that results in a large margin classifier. This amounts to an intuitive two-class clustering principle: find a labeling so that if one were to subsequently run an SVM, the margin obtained would be maximal over all possible labellings (Joachims 1999; Bennett & Demiriz 1998). Unsurprisingly, this is a hard computational problem. However, with some reformulation it can be approximated by a semidefinite program that can efficiently compute a good solution (Xu *et al.* 2004; De Bie & Cristianini 2003).

Suppose one was given *unlabeled* data $\mathbf{x}^1, .., \mathbf{x}^n$, and wished to solve for a labeling $\mathbf{y} \in \{-1, +1\}^n$ that leads to a maximum (soft) margin. Straightforwardly, one could attempt to tackle this optimization problem directly

$$\min_{\mathbf{y}\in\{-1,+1\}^n} \quad \gamma^{*-2}(\mathbf{y}) \quad \text{subject to} \quad -\epsilon \leq \mathbf{e}^\top\mathbf{y} \leq \epsilon \tag{5}$$

$$\text{where} \qquad \gamma^{*-2}(\mathbf{y}) \;=\; \max_{\boldsymbol{\lambda}} \;\boldsymbol{\lambda}^\top\mathbf{e} - \frac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, \mathbf{y}\mathbf{y}^\top\rangle \quad \text{subject to } 0 \leq \boldsymbol{\lambda} \leq 1$$

Unfortunately, $\gamma^{*-2}(\mathbf{y})$ is not a convex function of $\mathbf{y}$, and this formulation does not lead to an effective algorithmic approach. To obtain an efficient technique for solving this problem one first re-expresses the optimization, not directly in terms of the cluster labels $\mathbf{y}$, but instead in terms of the label kernel matrix $M = \mathbf{y}\mathbf{y}^\top$. The main advantage of doing so is that the inverse soft margin γ^{*-2} is in fact a convex function of M (Lanckriet *et al.* 2004)

$$\gamma^{*-2}(M) \;=\; \max_{\boldsymbol{\lambda}} \;\boldsymbol{\lambda}^\top\mathbf{e} - \tfrac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, M\rangle \quad \text{subject to} \quad 0 \leq \boldsymbol{\lambda} \leq 1$$

The convexity of γ^{*-2} with respect to M is easy to establish since this quantity is just a maximum over linear functions of M (Boyd & Vandenberghe 2004).

Working with the matrix M turns out to be particularly convenient here because of the following useful fact: if $M_{ij} = y_i y_j$, then $M_{ij} = 1$ if and only if $y_i = y_j$, and $M_{ij} = -1$ otherwise. That is, for $M \in \{-1, +1\}^{n\times n}$, $M = \mathbf{y}\mathbf{y}^\top$ for some $\mathbf{y}$ if and only if M is an *equivalence relation* matrix. Therefore the optimization problem (5) can be cast as working directly with equivalence relations M instead of example labellings $\mathbf{y}$. (We will need to exploit this observation below for the multi-class case.)

Conveniently, the property of being an equivalence relation matrix can be enforced with a simple semidefinite constraint, since $M \in \{-1, +1\}^{n\times n}$ encodes an equivalence relation if and only if $\text{diag}(M) = \mathbf{e}$ and $M \succeq 0$ (Helmberg 2000; Laurent & Poljak 1995). However, in addition, one clearly needs to impose some sort of constraint on the class balance, since otherwise one could simply assign all the data points to the same class and obtain an unbounded margin. This can be enforced with an additional linear constraint $-\epsilon\mathbf{e} \leq M\mathbf{e} \leq \epsilon\mathbf{e}$.

Therefore, putting the pieces together, one is left with an optimization problem on M that has a convex objective and convex constraints, except for the integer constraint $M \in \{-1, +1\}^{n\times n}$. Dropping the integer constraint yields the convex relaxation of the maximum margin labeling problem:

$$\begin{aligned}&\min_{M} \;\max_{\boldsymbol{\lambda}} \;\boldsymbol{\lambda}^\top\mathbf{e} - \tfrac{1}{2\beta}\langle K \circ \boldsymbol{\lambda}\boldsymbol{\lambda}^\top, M\rangle \text{ subject to} \\ &0 \leq \boldsymbol{\lambda} \leq 1, \;\; \text{diag}(M) = \mathbf{e}, \;\; M \succeq 0, \;\; -\epsilon\mathbf{e} \leq M\mathbf{e} \leq \epsilon\mathbf{e}\end{aligned} \tag{6}$$

This can be turned into an equivalent semidefinite program (Xu *et al.* 2004)

$$\begin{aligned}&\min_{M,g,\boldsymbol{\mu},\boldsymbol{\nu}} \; g \quad \text{subject to} \\ &\qquad \begin{bmatrix} M \circ K & \mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu} \\ (\mathbf{e} + \boldsymbol{\mu} - \boldsymbol{\nu})^\top & g - \frac{2}{\beta}\boldsymbol{\nu}^\top\mathbf{e} \end{bmatrix} \succeq 0 \\ &\qquad \text{diag}(M) = \mathbf{e}, \;\; M \succeq 0, \;\; -\epsilon\mathbf{e} \leq M\mathbf{e} \leq \epsilon\mathbf{e}\end{aligned} \tag{7}$$

Given a solution M^* to (7) one can recover a soft clustering $\mathbf{y}$ by setting $\mathbf{y} = \sqrt{\lambda_1}\mathbf{v}_1$, where $\lambda_1, \mathbf{v}_1$ are the maximum eigenvalue and corresponding eigenvector of M^*.

The results in (Xu *et al.* 2004) show that this method obtains clustering performance that often exceeds spectral clustering (Ng, Jordan, & Weiss 2001). However, a significant drawback of the method is the restriction to two-classes.

Semi-supervised two-class SVMs

Before considering multi-class extensions to this SVM training technique, we first note that these clustering algorithms can easily be extended to semi-supervised SVM training.

For semi-supervised training, one assumes that a small labeled training set $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$ given as well as an unlabeled training set $\mathbf{x}^{n+1}, ..., \mathbf{x}^N$. The goal in this case is to combine the information in these two data sets to produce a more accurate classifier.

It turns out to be very straightforward to extend the previous equivalence relation based clustering procedures to semi-supervised training. One simply adds constraints on the matrix M to force it to respect the observed equivalence relations among the *labeled* training data: $M_{ij} = y_i y_j$ for *labeled* examples $i, j \in \{1, ..., n\}$. Note that the observed training labels y_i for $i \in \{1, ..., n\}$ are *constants*, and therefore the new constraints are still linear in the parameters of M that are being optimized. The resulting method, although a simple extension, appears to obtain superior performance to previous semi-supervised training procedures for SVMs (Joachims 1999; Bennett & Demiriz 1998).

Multi-class formulation

Our main technical contribution in this paper is to extend the previous two-class methods to general multi-class SVM training algorithms. Although our strategy is similar to that employed above, there are some significant complications to deriving an effective training procedure.

As before, we start with the supervised case. Assume we are given labeled training examples $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$ where each example is assigned a label from a fixed finite set $y^i \in \{1, ..., \ell\}$. Here, we need to extend our feature functions $\boldsymbol{\phi}(\mathbf{x}, y)$ to include the y-labels explicitly, which provides a separate weight vector $\mathbf{w}_k$ for each class k. Once a complete weight vector has been learned, subsequent test examples $\mathbf{x}$ are classified according to $y^* = \arg\max_y \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}, y)$. The dominant multi-class training procedure for SVMs is due to (Crammer & Singer 2001), which, including slack variables, is formulated as

$$\omega \quad = \quad \min_{\mathbf{w},\boldsymbol{\xi}} \ \tfrac{\beta}{2}\|\mathbf{w}\|^2 + \boldsymbol{\xi}^\top \mathbf{e} \quad \text{subject to}$$

$$\mathbf{w}^\top(\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, k)) \geq \delta(y^i, k) - \xi_i \ \ \forall_{i,k} \quad (8)$$

where $\delta(y^i, k) = 1_{(y^i \neq k)}$, and ω is the multi-class analog of the inverse squared margin γ^{*-2}. Our main result depends crucially on being able to reformulate the quadratic program (8) so that it can be expressed entirely in terms of equivalence relation matrices instead of individual y-labels. To achieve this, we need to derive a different formulation of the dual from that given in (Crammer & Singer 2001).

Unsupervised multi-class SVMs

With some work, one can show that the following quadratic program is equivalent to the dual of (8).

Proposition 1 *The dual quadratic program of (8) can be reformulated as*

$$\omega(M, D) = \max_{\boldsymbol{\Lambda}} \ Q(\Lambda, M, D) \text{ subject to } \Lambda \geq 0, \ \Lambda\mathbf{e} = \mathbf{e}$$

$$\textit{where} \quad Q(\Lambda, M, D) = \ t - \langle D, \Lambda\rangle - \tfrac{1}{2\beta}\langle K, M\rangle + \tfrac{1}{\beta}\langle KD, \Lambda\rangle - \tfrac{1}{2\beta}\langle \Lambda\Lambda^\top, K\rangle \quad (9)$$

Here D and Λ are $n \times \ell$ matrices, M and K are $n \times n$ matrices, and we define the equivalence relation indicator matrices M and D such that $M_{ij} = 1_{(y_i = y_j)}$ and $D_{ik} = 1_{(y_i = k)}$ respectively. An important consequence of these definitions, which we exploit below, is that $M = DD^\top$.

This is not the same formulation of the dual to (8) given in (Crammer & Singer 2001). Importantly, this reformulation of the dual expresses the problem explicitly in terms of the equivalence relations M and D—which gives us a necessary advantage over the formulation of the dual given in (Crammer & Singer 2001).

We can now establish our main result. We would like to compute a solution to the problem

$$\min_{M,D} \ \omega(M, D) \qquad \text{subject to} \quad M = DD^\top$$

$$M \in \{0,1\}^{n\times n}, \ D \in \{0,1\}^{n\times\ell}$$

A solution to this problem will minimize the inverse square margin criterion ω and thus maximize the original margin criterion. However, just as in the two-class case, we need to worry about class balance, because a large margin value can always be achieved by eliminating classes. To impose class balance in the multi-class case we add the constraint $(\frac{1}{\ell} - \epsilon)n\mathbf{e} \leq M\mathbf{e} \leq (\frac{1}{\ell} + \epsilon)n\mathbf{e}$ for some ϵ.

The overall formulation, to this point, enjoys the advantage that the objective $\omega(M, D)$ is jointly convex in M and D. Unfortunately, neither the integer constraints nor the nonlinear constraint $M = DD^\top$ are convex. Therefore, as before, we need to derive a convex relaxation to this problem that preserves as much of the structure of the problem as possible. To do so, we first relax the constraint that M and D be $\{0,1\}$-valued, and instead allow them to take values in $[0, 1]$. However, we also have to cope with the nonlinear equality constraint $M = DD^\top$. To remove the non-convexity implicit in the nonlinear equality, we replace it with the convex inequalities $M \succeq DD^\top$ and diag$(M) = \mathbf{e}$. The resulting problem is a convex optimization over M and D.

$$\min_{M,D} \ \omega(M, D) \ \text{ subject to } \quad 0 \leq M \leq 1, \ \ 0 \leq D \leq 1$$

$$\text{diag}(M) = \mathbf{e}, \ \ M \succeq DD^\top$$

$$(\frac{1}{\ell} - \epsilon)n\mathbf{e} \leq M\mathbf{e} \leq (\frac{1}{\ell} + \epsilon)n\mathbf{e} \qquad (10)$$

This in fact yields the convex optimization problem we wish to solve.

To tackle this problem in practice, it is convenient to first reformulate it as a semidefinite program.

Proposition 2 *The convex optimization problem (10) is equivalent to*

$$\min_{M,D,V,\boldsymbol{\alpha},g} \ g \qquad \textit{subject to}$$

$$\begin{bmatrix} I \otimes K & \mathbf{c} \\ \mathbf{c}^\top & g + \frac{1}{\beta^2}\langle K, M\rangle + \frac{2}{\beta}\boldsymbol{\alpha}^\top\mathbf{e} \end{bmatrix} \succeq 0 \qquad (11)$$

$$\begin{bmatrix} I & D^\top \\ D & M \end{bmatrix} \succeq 0$$

$$\textit{diag}(M) = \mathbf{e}, \ \ 0 \leq M \leq 1, \ \ 0 \leq D \leq 1, \ \ V \geq 0$$

$$(\tfrac{1}{\ell} - \epsilon)n\mathbf{e} \leq M\mathbf{e} \leq (\tfrac{1}{\ell} + \epsilon)n\mathbf{e}$$

where $\mathbf{c} = vec(\frac{1}{\beta}KD - D + V + \boldsymbol{\alpha}\mathbf{e}^\top)$.[1]

This formulation re-expresses the problem in a form that can be solved by conventional semidefinite programming techniques (Helmberg 2000; Boyd & Vandenberghe 2004).

Semi-supervised multi-class SVMs

Again, before presenting experimental results, we briefly note that the above formulation of *unsupervised* multi-class SVM training can easily be extended to *semi-supervised* learning. The extension of the unsupervised training procedure to the semi-supervised case proceeds similarly to the two-class case. Here we simply add the constraints in the observed labels to the semidefinite program: $M_{ij} = 1_{(y_i=y_j)}$ for all labeled training pairs i, j, and $D_{ik} = 1_{(y_i=k)}$ for each labeled training pair and class label i, k. The remainder of the program is equivalent to (11), and thus still defines a semidefinite programming problem that can be solved by standard methods.

Experimental results

We implemented the generalized multi-class SVM training procedures based on (11) using the semidefinite programming package SDPT3 available at www.optimization-online.org. Results for the unsupervised and semi-supervised formulations respectively are reported below.

Unsupervised results

First, for unsupervised training, we compared the performance of our semidefinite multi-class clustering technique to the spectral clustering method of (Ng, Jordan, & Weiss 2001) and also straightforward k-means clustering. Both semidefinite clustering and spectral clustering were run with the same radial basis function kernel and matching width parameters. In fact, in each case, we chose the best width parameter *for spectral clustering* by searching over a small set of five widths related to the scale of the problem. In addition, the slack parameter for maximum margin clustering was simply set to an arbitrary value.[2] To assess clustering performance we first took a set of labeled data, removed the labels, ran the clustering algorithms, labeled each of the resulting clusters with the majority class according to the original training labels, and finally measured the number of misclassifications made by each clustering.

Our first experiments were conducted on the synthetic data sets depicted in Figure 1. Table 1 shows that for the first four sets of data (AAAI, Circle&Balls, 3JoinedCircles, Squiggles) semidefinite and spectral clustering obtained identical small error rates, which were in turn significantly smaller than those obtained by k-means (except for the AAAI case).

We also conducted clustering experiments on two real data sets consisting of images of hand-written digits; see Figures 2 and 3. The first data set, DigitsA, consists of black and white images of 20×16 pixels, as shown in Figure 2. The second data set, DigitsB, consists of grey scale images of 16×16 pixels, as shown in Figure 3. Table 1 shows that semidefinite clustering demonstrates a significant advantage over k-means clustering on these data sets, but also a systematic advantage over spectral clustering. (See the Table 1 caption for details of the experimental set up.)

Semi-supervised results

We tested our approach to semi-supervised learning on various data sets from the UCI repository. To the best of our knowledge, there are no other semi-supervised training procedures available for multi-class SVMs that apply the large margin approach of (Joachims 1999; Bennett & Demiriz 1998) or the semidefinite approach we develop above. Therefore, we compare to standard *supervised* multi-class SVMs (Crammer & Singer 2001) but also to a general semi-supervised approach based on clustering the data using spectral clustering, and then using the labeled data to assign minimum error labels to the classes. In each case, we evaluated the techniques transductively. That is, we split the data into a labeled and unlabeled part, held out the labels of the unlabeled portion, trained the semi-supervised techniques, reclassified the unlabeled examples using the learned results, and measured the misclassification error on the held out labels.

In these experiments, we used two UCI data sets (Balance-scale and Cars), one of the hand-written digits data sets (DigitsA), and a data set of face images. Here we see that the semi-supervised semidefinite approach tends to outperform the other methods, particularly on the digits data. Generally, semi-supervised training shows an advantage over strictly supervised training which ignores the extra unlabeled data. Table 2 shows that the semidefinite SVM method is effective at exploiting unlabeled data to improve the prediction of held out labels. In every case (except 5faces), it significantly reduces the error of standard multi-class SVM, and obtains the best overall performance of the semi-supervised learning techniques we have investigated. (See the Table 2 caption for details of the experimental set up.)

Conclusion

We have proposed a general, unified principle for clustering and semi-supervised learning based on the maximum margin principle popularized by supervised SVMs. Our work generalizes previous approaches to the multi-class case. Our results on both unsupervised and semi-supervised learning are competitive with, and sometimes exceed the state of the art. Overall, semidefinite programming appears to be an effective approach for unifying and generalizing SVM training techniques, and applying SVMs to a wider range of circumstances, like unsupervised and semi-supervised learning.

We plan to extend these results to the multivariate case where there are multiple, correlated y labels associated with the input data observations. Recently, substantial progress has been made on learning classifiers that make dependent

[1] The notation $\text{vec}(X)$ means turning X into a vector by concatenating its columns. $A \otimes B$ denotes the Kronecker product.

[2] It turns out that the slack parameter β did not have a significant effect on any of our preliminary investigations, so we just set it to $\beta = 0.01$ for all of the experiments reported here.

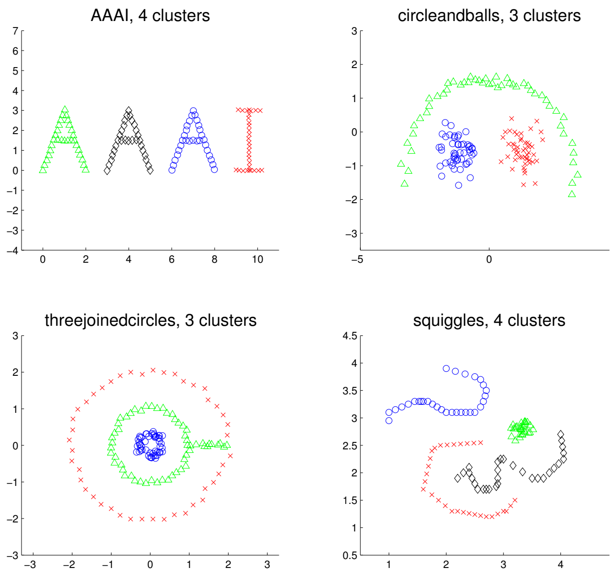

Figure 1: Four artificial data sets used in the clustering experiments.

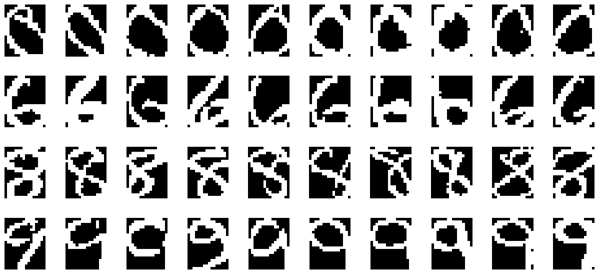

Figure 2: A sampling of handwritten digit images from the first data set, DigitsA (zeros, sixes, eights and nines).

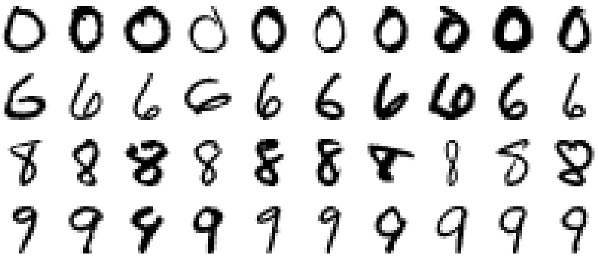

Figure 3: A sampling of handwritten digit images from the second data set, DigitsB (zeros, sixes, eights and nines).

Clustering:	Semidefinite	Spectral	Kmeans
AAAI	0	0	0
Circle&Balls	0	0	18.7
3Circles	4.0	4.0	47.3
Squiggles	0	0	28.3
DigitsA689*	3.4	12.0	12.0
DigitsA689	7.2 ±1.3	12.3 ±2.3	14.0 ±4.4
DigitsB689	13.3 ±2.8	15.1 ±3.9	18.5 ±4.6
DigitsA0689*	7.5	9.2	38.3
DigitsA0689	11.6 ±1.8	20.4 ±2.4	24.1 ±4.1
DigitsB0689	21.1 ±2.3	25.8 ±2.6	27.7 ±4.4

Table 1: Clustering results: Percentage misclassification errors of the various clustering algorithms on the various data sets (± one standard deviation). DigitsA are the results on the black and white digit data set, and DigitsB are the results on the grey scale digit data set. The digits included in the data sample are explicitly indicated. DigitsA* reports one run using all 39 examples of each digit. DigitsA reports 10 repeats of subsampling 20 out of 39 examples of each digit. DigitsB reports 10 repeats of subsampling 30 out of 1100 examples of each digit.

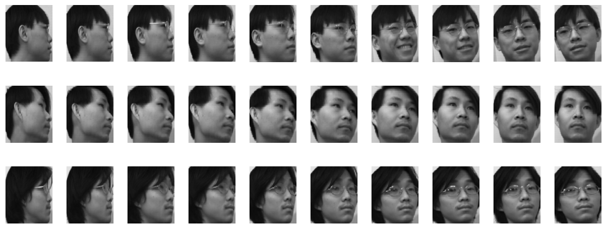

Figure 4: A sampling of the face data (three people).

predictions of test labels that are explicitly related (Taskar, Guestrin, & Koller 2003; Altun, Tsochantaridis, & Hofmann 2003; Tsochantaridis *et al.* 2004). The work of (Taskar, Guestrin, & Koller 2003), in particular, considers maximum margin Markov networks, which are directly a multivariate version of SVMs. Developing convex unsupervised and semi-supervised training algorithms for maximum margin Markov networks remains an important challenge.

Acknowledgments

Research supported by the Alberta Ingenuity Centre for Machine Learning, NSERC, MITACS, CFI, and the Canada Research Chairs programme.

References

Altun, Y.; Tsochantaridis, I.; and Hofmann, T. 2003. Hidden Markov support vector machines. In *Proceedings International Conference on Machine Learning (ICML-03)*.

Bansal, N.; Blum, A.; and Chawla, S. 2002. Correlation clustering. In *Conference on Foundations of Computer Science (FOCS-02)*.

Bennett, K., and Demiriz, A. 1998. Semi-supervised sup-

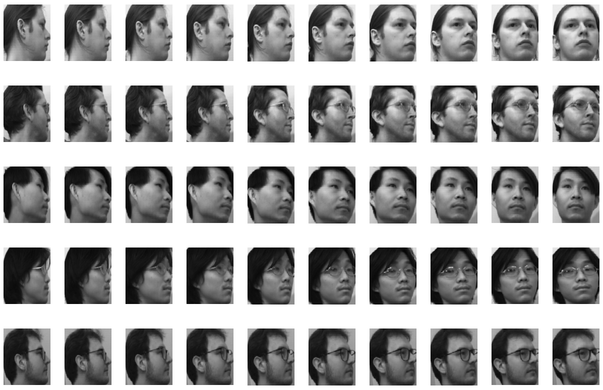

Figure 5: A sampling of the face data (five people).

Semi-sup:	Semidef	supSVM	Spectral
UCI:Balance-scale	24.1 ±1.7	30.3 ±2.4	26.3 ±4.3
UCI:Cars	41.9 ±0.4	43.1 ±1.4	43.2 ±0.4
DigitsA689	5.6 ±0.7	16.3 ±1.6	12.7 ±0.3
DigitsA0689	6.5 ±0.5	20.6 ±3.3	11.8 ±2.7
3faces	0	10.0 ±2.0	0
5faces	5.2 ±1.5	14.4 ±1.7	0

Table 2: Semi-supervised learning results: Percentage misclassification errors of the various semi-supervised learning algorithms on the various data sets (± one standard deviation). 10 repeats, each randomly subsampling 1/10 of the data to be labeled. Data set sizes are: UCI:Balance-scale: 3 classes, 40 examples each. UCI:Cars: 3 classes, 40 examples each. DigitsA689: 3 classes, 39 examples each. DigitsA0689: 4 classes, 39 examples each. 3faces: 3 classes, 32 examples each. 5faces: 5 classes, 30 examples each.

port vector machines. In *Advances in Neural Information Processing Systems 11 (NIPS-98)*.

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

Crammer, K., and Singer, Y. 2001. On the algorithmic interpretation of multiclass kernel-based vector machines. *Journal of Machine Learning Research* 2.

De Bie, T., and Cristianini, N. 2003. Convex methods for transduction. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*.

Goemans, M., and Williamson, D. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *JACM* 42:1115–1145.

Graepel, T., and Herbrich, R. 2003. Invariant pattern recognition by semidefinite programming machines. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*.

Helmberg, C. 2000. Semidefinite programming for combinatorial optimization. Technical Report ZIB-Report ZR-00-34, Konrad-Zuse-Zentrum Berlin.

Joachims, T. 1999. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML-99)*.

Lanckriet, G.; Cristianini, N.; Bartlett, P.; Ghaoui, L.; and Jordan, M. 2004. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research* 5.

Laurent, M., and Poljak, S. 1995. On a positive semidefinite relaxation of the cut polytope. *Linear Algebra and its Applications* 223/224.

Nesterov, Y., and Nimirovskii, A. 1994. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM.

Ng, A.; Jordan, M.; and Weiss, Y. 2001. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems 14 (NIPS-01)*.

Schoelkopf, B., and Smola, A. 2002. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

Strang, G. 1998. *Introduction to Linear Algebra*. Wellesley-Cambridge Press.

Taskar, B.; Guestrin, C.; and Koller, D. 2003. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS-03)*.

Tsochantaridis, I.; Hofmann, T.; Joachims, T.; and Altun, Y. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings International Conference on Machine Learning (ICML-04)*.

Vanderbei, R. 1996. *Linear Programming: Foundations and Extensions*. Kluwer.

Xu, L.; Neufeld, J.; Larson, B.; and Schuurmans, D. 2004. Maximum margin clustering. In *Advances in Neural Information Processing Systems 17 (NIPS-04)*.