Support Vector Machine Classification with Indefinite Kernels

Ronny Luss* Alexandre d'Aspremont†

March 29, 2008

Abstract

We propose a method for support vector machine classification using indefinite kernels. Instead of directly minimizing or stabilizing a nonconvex loss function, our algorithm simultaneously computes support vectors and a proxy kernel matrix used in forming the loss. This can be interpreted as a penalized kernel learning problem where indefinite kernel matrices are treated as a noisy observations of a true Mercer kernel. Our formulation keeps the problem convex and relatively large problems can be solved efficiently using the projected gradient or analytic center cutting plane methods. We compare the performance of our technique with other methods on several classic data sets.

1 Introduction

A critical step in support vector machine (SVM) classification is choosing a suitable kernel. The kernel measures similarity between data points and must be positive semidefinite, i.e. satisfy Mercer's condition, because it is formed as the Gram matrix of data points in a reproducing kernel Hilbert space. This condition makes the classification problem convex and preserves strong duality between the support vector machine and maximum margin classification problems. Here, we present an algorithm for SVM classification using indefinite kernels, i.e. kernels which do not satisfy Mercer's positive semidefiniteness condition.

Our interest in indefinite kernels is motivated by several observations. First, certain similarity measures take advantage of application-specific structure in the data and often display excellent empirical classification performance. Unlike popular kernels used in support vector machine classification, these similarity matrices are often indefinite, so do not necessarily correspond to a reproducing kernel Hilbert space (see [1] for a discussion).

In particular, an application of classification with indefinite kernels to image classification using Earth Mover's Distance was discussed in [2]. Similarity measures for protein sequences such as the Smith-Waterman and BLAST scores are indefinite yet have provided hints for constructing useful positive semidefinite kernels such as those decribed in [3] or have been transformed into positive semidefinite kernels with good empirical performance (see [4] for example). Tangent distance similarity measures, as described in [5] or [6], are invariant to various simple image transformations and have also shown excellent performance in optical character recognition. Finally, it is sometimes impossible to prove that some kernels satisfy Mercer's condition or the numerical complexity of

*ORFE Department, Princeton University, Princeton, NJ 08544. rluss@princeton.edu
†ORFE Department, Princeton University, Princeton, NJ 08544. aspremon@princeton.edu

evaluating the exact positive kernel is too high and a proxy (and not necessarily positive semidefinite) kernel has to be used instead (see [7] for example). In both cases, our method allows us to bypass these limitations. Our objective here is to directly use these indefinite similarity measures for classification.

Our work closely follows in spirit recent results on kernel learning (see [8] or [9]), where the kernel matrix is learned as a linear combination of given kernels, and the result is explicitly constrained to be positive semidefinite. While this problem is numerically challenging, [10] adapted the SMO algorithm to solve the case where the kernel is written as a positively weighted combination of other kernels. Here, we never *numerically* optimize the kernel matrix because this part of the problem can be solved explicitly, which means that the complexity of our method is substantially lower than that of classical kernel learning algorithms and closer in practice to the algorithm used in [11], who formulate the multiple kernel learning problem of [10] as a semi-infinite linear program and solve it with a column generation technique similar to the analytic center cutting plane method we use here.

1.1 Current results

Several methods have been proposed for dealing with indefinite kernels in SVMs. A first direction embeds data in a pseudo-Euclidean (pE) space: [12], for example, formulates the classification problem with an indefinite kernel as that of minimizing the distance between convex hulls formed from the two categories of data embedded in the pE space. The nonseparable case is handled in the same manner using reduced convex hulls (see [13] for a discussion on geometric interpretations in SVM).

Another direction applies direct spectral transformations to indefinite kernels: flipping the negative eigenvalues or shifting the eigenvalues and reconstructing the kernel with the original eigenvectors in order to produce a positive semidefinite kernel (see [14] and [2] for example). Yet another option is to reformulate either the maximum margin problem or its dual in order to use the indefinite kernel in a convex optimization problem. One reformulation suggested in [15] replaces the indefinite kernel by the identity matrix and maintains separation using linear constraints. This method achieves good performance, but the convexification procedure is hard to interpret. Directly solving the nonconvex problem sometimes gives good results as well (see [16] and [12]) but offers no guarantees on performance.

1.2 Contributions

In this work, instead of directly transforming the indefinite kernel, we simultaneously learn the support vector weights and a proxy Mercer kernel matrix by penalizing the distance between this proxy kernel and the original, indefinite one. Our main result is that the kernel learning part of that problem can be solved explicitly, meaning that the classification problem with indefinite kernels can simply be formulated as a perturbation of the positive semidefinite case.

Our formulation can be interpreted as a penalized kernel learning problem with uncertainty on the input kernel matrix. In that sense, indefinite similarity matrices are seen as noisy observations of a true positive semidefinite kernel and we learn a kernel that increases the generalization performance. From a complexity standpoint, while the original SVM classification problem with indefinite kernel is nonconvex, the penalization we detail here results in a convex problem, hence can be solved efficiently with guaranteed complexity bounds. This paper builds on the earlier con-

ference version [17], providing new results on learning with Mercer kernels, more efficient algorithm formulations and experiments on a broader set of experiments on similarity measures.

The paper is organized as follows. In Section 2 we formulate our main classification result and detail its interpretation as a penalized kernel learning problem. In Section 3 we describe two algorithms for solving this problem. Finally, in Section 4, we test the numerical performance of these methods on various data sets.

Notation
We write $\mathbf{S}^n$ ($\mathbf{S}^n_+$) the set of symmetric (positive-semidefinite) matrices of size n. The vector e is the n-vector of ones. Given a matrix X, $\lambda_i(X)$ denotes the i^{th} eigenvalue of X. X_+ is the positive part of the matrix X, i.e. $X_+ = \sum_i \max(0, \lambda_i) v_i v_i^T$ where λ_i and v_i are the i^{th} eigenvalue and eigenvector of the matrix X.

2 SVM with indefinite kernels

In this section, we modify the SVM kernel learning problem and formulate a penalized kernel learning problem on indefinite kernels. We also detail how our framework applies to kernels that satisfy Mercer's condition.

2.1 Kernel learning

Let $K \in \mathbf{S}^n$ be a given kernel matrix and $y \in \mathbf{R}^n$ be the vector of labels, with $Y = \mathbf{diag}(y)$ the matrix with diagonal y. We formulate the kernel learning problem as in [8] where the authors minimize an upper bound on the misclassification probability when using SVM with a given kernel K. This upper bound is the generalized performance measure:

$$\omega_C(K) = \max_{\{0 \leq \alpha \leq C, \alpha^T y = 0\}} \alpha^T e - \mathbf{Tr}(K(Y\alpha)(Y\alpha)^T)/2 \tag{1}$$

where $\alpha \in \mathbf{R}^n$ and C is the SVM misclassification penalty. This is also the classic 1-norm soft margin SVM problem. They show that $\omega_C(K)$ is convex in K and solve problems of the form:

$$\min_{K \in \mathcal{K}} \omega_C(K) \tag{2}$$

in order to learn an optimal kernel K^* that achieves good generalization performance. When $\mathcal{K}$ is restricted to convex subsets of S^n_+ with constant trace, they show that problem (2) can be reformulated as a convex program. Further restrictions to $\mathcal{K}$ reduce (2) to more tractable optimization problems such as semidefinite and quadratically constrained quadratic programs. Our goal is to solve a problem similar to (2) by restricting the distance between a proxy kernel used in classification and the original indefinite similarity measure.

2.2 Learning from indefinite kernels

The performance measure in (1) is the dual of the SVM classification problem with hinge loss and quadratic penalty. When K is positive semidefinite, this problem is a convex quadratic program. Suppose now that we are given an indefinite kernel matrix $K_0 \in \mathbf{S}^n$. We formulate a new instance of

problem (2) by restricting K to be a positive semidefinite kernel matrix in some given neighborhood of the original (indefinite) kernel matrix K_0 and solve:

$$\min_{\{K\succeq 0,\ \|K-K_0\|_F^2\leq\beta\}} \max_{\{\alpha^T y=0,\ 0\leq\alpha\leq C\}} \alpha^T e - \frac{1}{2}\,\mathbf{Tr}(K(Y\alpha)(Y\alpha)^T)$$

in the variables $K \in \mathbf{S}^n$ and $\alpha \in \mathbf{R}^n$, where the parameter $\beta > 0$ controls the distance between the original matrix K_0 and the proxy kernel K. This is the kernel learning problem (2) with $\mathcal{K} = \{K \succeq 0,\ \|K - K_0\|_F^2 \leq \beta\}$. The above problem is infeasible for small values of β, so we replace here the hard constraint on K by a penalty on the distance between the proxy kernel and the original indefinite similarity matrix and solve instead:

$$\min_{\{K\succeq 0\}} \max_{\{\alpha^T y=0,\ 0\leq\alpha\leq C\}} \alpha^T e - \frac{1}{2}\,\mathbf{Tr}(K(Y\alpha)(Y\alpha)^T) + \rho\|K - K_0\|_F^2 \tag{3}$$

Because (3) is convex-concave and has a compact feasible set, we can switch the max and min to form the dual:

$$\max_{\{\alpha^T y=0,0\leq\alpha\leq C\}} \min_{\{K\succeq 0\}} \alpha^T e - \frac{1}{2}\,\mathbf{Tr}(K(Y\alpha)(Y\alpha)^T) + \rho\|K - K_0\|_F^2 \tag{4}$$

in the variables $K \in \mathbf{S}^n$ and $\alpha \in \mathbf{R}^n$, where the parameter $\rho > 0$ controls the magnitude of the penalty on the distance between K and K_0.

We first note that problem (4) is a convex optimization problem. The inner minimization problem is a convex conic program on K. Also, as the pointwise minimum of a family of concave quadratic functions of α, the solution to the inner problem is a concave function of α, hence the outer optimization problem is also convex (see [18] for further details). Thus, (4) is a concave maximization problem subject to linear constraints and is therefore a convex problem in α. Our key result here is that the inner kernel learning optimization problem in (4) can be solved in closed form.

Theorem 1 *Given a similarity matrix $K_0 \in S^n$, a vector $\alpha \in \mathbf{R}^n$ of support vector coefficients and the label matrix $Y = \mathbf{diag}(y)$, the optimal kernel in problem (4) can be computed explicitly as:*

$$K^* = (K_0 + (1/4\rho)(Y\alpha)(Y\alpha)^T)_+ \tag{5}$$

where $\rho \geq 0$ controls the penalty.

Proof. For a fixed α, the inner minimization problem can be written out as:

$$\min_{\{K\succeq 0\}} \alpha^T e + \rho(\mathbf{Tr}(K^T K) - 2\,\mathbf{Tr}(K^T(K_0 + \frac{1}{4\rho}(Y\alpha)(Y\alpha)^T)) + \mathbf{Tr}(K_0^T K_0))$$

where we have replaced $\|K - K_0\|_F^2 = \mathbf{Tr}((K-K_0)^T(K-K_0))$ and collected similar terms. Adding and subtracting the constant $\rho\,\mathbf{Tr}((K_0+\frac{1}{4\rho}(Y\alpha)(Y\alpha)^T)^T(K_0+\frac{1}{4\rho}(Y\alpha)(Y\alpha)^T))$ shows that the inner minimization problem is equivalent to the following problem:

$$\begin{array}{ll} \text{minimize} & \|K - (K_0 + \frac{1}{4\rho}(Y\alpha)(Y\alpha)^T)\|_F^2 \\ \text{subject to} & K \succeq 0 \end{array}$$

in the variable $K \in \mathbf{S}^n$ where we have dropped remaining constants from the objective. This is the projection of the matrix $K_0 + (1/4\rho)(Y\alpha)(Y\alpha)^T$ on the cone of positive semidefinite matrices which yields the desired result. ■

Plugging the explicit solution for the proxy kernel derived in (5) into the classification problem (4), we get:

$$\max_{\{\alpha^T y=0,\ 0\leq\alpha\leq C\}} \alpha^T e - \frac{1}{2}\,\mathbf{Tr}(K^*(Y\alpha)(Y\alpha)^T) + \rho\|K^* - K_0\|_F^2 \tag{6}$$

in the variable $\alpha \in \mathbf{R}^n$, where $(Y\alpha)(Y\alpha)^T$ is the rank one matrix with coefficients $y_i\alpha_i\alpha_j y_j$. Problem (6) can be cast as an eigenvalue optimization problem in the variable α. Letting the eigenvalue decomposition of $K_0 + (1/4\rho)(Y\alpha)(Y\alpha)^T$ be VDV^T, we get $K^* = VD_+V^T$, and with v_i the i^{th} column of V, we can write:

$$\begin{aligned}\mathbf{Tr}(K^*(Y\alpha)(Y\alpha)^T) &= (Y\alpha)^T V D_+ V^T (Y\alpha)\\ &= \sum_i \max\left(0, \lambda_i\left(K_0 + \frac{1}{4\rho}(Y\alpha)(Y\alpha)^T\right)\right)(\alpha^T Y v_i)^2.\end{aligned}$$

Using the same technique, we can also rewrite the term $\|K^* - K_0|_F^2$ using this eigenvalue decomposition. Our original optimization problem (4) finally becomes:

$$\begin{array}{ll}\text{maximize} & \alpha^T e - \frac{1}{2}\sum_i \max(0, \lambda_i(K_0 + (Y\alpha)(Y\alpha)^T/4\rho))(\alpha^T Y v_i)^2\\ & +\rho\sum_i \left(\max(0, \lambda_i(K_0 + (Y\alpha)(Y\alpha)^T/4\rho))\right)^2\\ & -2\rho\sum_i \mathbf{Tr}((v_i v_i^T)K_0)\max(0, \lambda_i(K_0 + (Y\alpha)(Y\alpha)^T/4\rho)) + \rho\,\mathbf{Tr}(K_0K_0)\\ \text{subject to} & \alpha^T y = 0, 0 \leq \alpha \leq C\end{array} \tag{7}$$

in the variable $\alpha \in \mathbf{R}^n$. By construction, the objective function is concave, hence (7) is a convex optimization problem in α.

2.3 Interpretation

Our explicit solution of the optimal kernel given in (5) is the projection of a penalized rank-one update to the indefinite kernel on the cone of positive semidefinite matrices. As ρ tends to infinity, the rank-one update has less effect and in the limit, the optimal kernel is the kernel given by zeroing out the negative eigenvalues of the indefinite kernel. This means that if the indefinite kernel contains a very small amount of noise, the best positive semidefinite kernel to use with SVM in our framework is the positive part of the indefinite kernel.

This limit as ρ tends to infinity also motivates a heuristic for transforming the kernel on the testing set. Since negative eigenvalues in the training kernel are thresholded to zero in the limit, the same transformation should occur for the test kernel. Hence, to measure generalization performance, we update the entries of the full kernel corresponding to training instances by the rank-one update resulting from the optimal solution to (7) and threshold the negative eigenvalues of the full kernel matrix to zero to produce a Mercer kernel on the test set.

2.4 Dual problem

As discussed above, problems (3) and (4) are dual. The inner maximization in problem (3) is a quadratic program in α, whose dual is another quadratic minimization problem. This allows us to write (3) as a joint minimization problem:

$$\begin{array}{ll} \text{minimize} & \mathbf{Tr}(K^{-1}(Y(e-\lambda+\mu+y\nu))(Y(e-\lambda+\mu+y\nu))^T)/2 + C\mu^T e + \rho\|K-K_0\|_F^2 \\ \text{subject to} & K \succeq 0, \lambda, \mu \geq 0 \end{array} \tag{8}$$

in the variables $K \in \mathbf{S}^n$, $\lambda, \mu \in \mathbf{R}^n$ and $\nu \in \mathbf{R}$. This is a quadratic program in the variables λ, μ (which correspond to the constraints $0 \leq \alpha \leq C$) and ν (which is the dual variable for the constraint $\alpha^T y = 0$). As we have seen earlier, any feasible solution $\alpha \in \mathbf{R}^n$ produces a corresponding proxy kernel in (5). Plugging this kernel into problem (8) allows us to compute an upper bound on the optimum value of problem (4) by solving a simple quadratic program in the variables λ, μ, ν. This result can then be used to bound the duality gap in (7) and track convergence.

2.5 Learning from Mercer kernels

While our central motivation is to use indefinite kernels for SVM classification, one would also like to analyze what happens when a Mercer kernel is used as input in (4). In this case, we learn another kernel that decreases the upper bound on generalization performance and produces perturbed support vectors. We can again interpret the input as a noisy kernel, and as such, one that will achieve suboptimal performance. If the input kernel is the best kernel to use (i.e. is not noisy), we will observe that our framework achieves optimal performance as ρ tends to infinity (through cross validation), otherwise we simply learn a better kernel using a finite ρ.

When the similarity measure K_0 is positive semidefinite (i.e. satisfies Mercer's condition), the proxy kernel K^* in Theorem 1 simplifies to a rank-one update of K_0:

$$K^* = K_0 + (1/4\rho)(Y\alpha^*)(Y\alpha^*)^T \tag{9}$$

whereas, for indefinite K_0, the solution was to project this matrix on the cone of positive semidefinite matrices. Plugging (9) into problem (4) gives:

$$\max_{\{\alpha^T y=0,\ 0\leq\alpha\leq C\}} \ \alpha^T e - \frac{1}{2}\,\mathbf{Tr}(K_0(Y\alpha)(Y\alpha)^T) - \frac{1}{16\rho}\sum_{i,j}(\alpha_i\alpha_j)^2 \tag{10}$$

which is the classic SVM problem given in (1) with a fourth order penalty on the support vectors. For testing in this framework, we do not need to transform the kernel, only the support vectors are perturbed. In this case, computing the gradient no longer requires eigenvalue decompositions at each iteration. Experimental results are shown in Section 4.

2.6 Componentwise penalties

Indefinite SVM can be generalized further with componentwise penalties on the distance between the proxy kernel and the indefinite kernel K_0. We generalize problem (4) to:

$$\max_{\{\alpha^T y=0, 0\leq\alpha\leq C\}} \ \min_{\{K\succeq 0\}} \ \alpha^T e - \frac{1}{2}\,\mathbf{Tr}(K(Y\alpha)(Y\alpha)^T) + \sum_{i,j} H_{ij}(K_{ij}-K_{0ij})^2 \tag{11}$$

where H is now a matrix of varying penalties on the componentwise distances. For a specific class of penalties, the optimal kernel K^* can be derived explicitly as follows.

Theorem 2 *Given a similarity matrix $K_0 \in S^n$, a vector $\alpha \in \mathbf{R}^n$ of support vector coefficients and the label matrix $Y = \mathbf{diag}(y)$, when H is rank-one with $H_{ij} = h_i h_j$, the optimal kernel in problem (11) has the explicit form:*

$$K^* = W^{-1/2}((W^{1/2}(K_0 + \frac{1}{4}(W^{-1/2}Y\alpha^*)(W^{-1/2}Y\alpha^*)^T)W^{1/2})_+)W^{-1/2} \tag{12}$$

where W is the diagonal matrix with $W_{ii} = h_i$.

Proof. The inner minimization problem to problem (11) can be written out as:

$$\min_{\{K \succeq 0\}} \ \sum_{i,j} H_{ij}(K_{ij}^2 - 2K_{ij}K_{0ij} + K_{0ij}^2) - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j K_{i,j}$$

Adding and subtracting $\sum_{i,j} H_{ij}(K_{0ij} + \frac{1}{4H_{ij}} y_i y_j \alpha_i \alpha_j)^2$, combining similar terms, and removing remaining constants leaves the following:

$$\begin{array}{ll} \text{minimize} & \|H^{1/2} \circ (K - (K_0 + \frac{1}{4H} \circ (Y\alpha)(Y\alpha)^T))\|_F^2 \\ \text{subject to} & K \succeq 0 \end{array}$$

where $\circ$ denotes the Hardamard product, $(A \circ B)_{ij} = a_{ij}b_{ij}$, and $(H^{1/2})_{ij} = H_{ij}^{1/2}$. This is a weighted projection problem where H_{ij} is the penalty on $(K_{ij} - K_{0ij})^2$. Since H is rank-one, the result follows from Theorem 3.2 of [19]. ■

Notice that Theorem 2 is a generalization of Theorem 1 where we had $H = ee^T$. In constructing a rank-one penalty matrix H, we simply assign penalties to each training point. The componentwise penalty formulation can also be extended to true kernels. If $K_0 \succeq 0$, then K^* in Theorem 2 simplifies to a rank-one update of K_0:

$$K^* = K_0 + \frac{1}{4}(W^{-1/2}Y\alpha)(W^{-1/2}Y\alpha)^T. \tag{13}$$

where no projection is required here.

3 Algorithms

We now detail two algorithms that can be used to solve problem (7), which maximizes a nondifferentiable concave function subject to convex constraints. An optimal point always exists since the feasibile set is bounded and nonempty. For numerical stability, in both algorithms, we quadratically smooth our objective to compute a gradient. We first describe a simple projected gradient method which has numerically cheap iterations but no fully explicit complexity bound. We then show how to apply the analytic center cutting plane method whose iterations are numerically more complex but which converges linearly.

Smoothing Our objective contains terms of the form $\max\{0, f(x)\}$ for some function $f(x)$, which are not differentiable (described in the section below). These functions are easily smoothed out by a Moreau-Yosida regularization technique (see [20] for example). We replace the max by a continuously differentiable $\frac{\epsilon}{2}$-approximation as follows:

$$\varphi_\epsilon(f(x)) = \max_{0 \le u \le 1} (uf(x) - \frac{\epsilon}{2}u^2).$$

The gradient is then given by $\nabla\varphi_\epsilon(f(x)) = u^*(x)\nabla f(x)$ where $u^*(x) = \operatorname{argmax} \varphi_\epsilon(f(x))$.

Gradient Calculating the gradient of the objective function in (7) requires computing the eigenvalue decomposition of a matrix of the form $X(\alpha) = K + \rho\alpha\alpha^T$. Given a matrix $X(\alpha)$, the derivative of the i^{th} eigenvalue with respect to α is then given by:

$$\frac{\partial\lambda_i(X(\alpha))}{\partial\alpha} = v_i^T \frac{\partial X(\alpha)}{\partial\alpha} v_i \tag{14}$$

where v_i is the i^{th} eigenvector of $X(\alpha)$. We can then combine this expression with the smooth approximation above to get the gradient.

3.1 Computing proxy kernels

Because the proxy kernel in (5) only requires a rank one update of a (fixed) eigenvalue decomposition:

$$K^* = (K_0 + (1/4\rho)(Y\alpha)(Y\alpha)^T)_+$$

we now briefly recall how v_i and $\lambda_i(X(\alpha))$ can be computed efficiently in this case (see [21] for further details). We refer the reader to [22] for another kernel learning example using this method. Given the eigenvalue decomposition $X = VDV^T$, by changing basis this problem can be reduced to the decomposition of the diagonal plus rank-one matrix, $D + \rho uu^T$, where $u = V^T\alpha$. First, the updated eigenvalues are determined by solving the secular equations:

$$\det(D + \rho uu^T - \lambda I) = 0,$$

which can be done in $O(n^2)$. While there is an explicit solution for the eigenvectors corresponding to these eigenvalues, they are not stable because the eigenvalues are approximated. This instability is circumvented by computing a vector $\hat{u}$ such that approximate eigenvalues λ are the exact eigenvalues of the matrix $D + \rho\hat{u}\hat{u}^T$, then computing its stable eigenvectors explicitly, where both steps can be done in $O(n^2)$ time. The key is that $D + \rho\hat{u}\hat{u}^T$ is close enough to our original matrix so that the eigenvalues and eigenvectors are stable approximations of the true values. Finally, the eigenvectors of our original matrix are computed as VW, with W as the stable eigenvectors of $D + \rho\hat{u}\hat{u}^T$. Updating the eigenvalue decomposition is reduced to an $O(n^2)$ procedure plus one matrix multiplication, which is then the complexity of one gradient computation.

We note that eigenvalues of symmetric matrices are not differentiable when some of them have multiplicities greater than one (see [23] for a discussion). Most kernels tested here were of full rank with distinct eigenvalues however.

3.2 Projected gradient method

The projected gradient method takes a steepest descent step, then projects the new point back onto the feasible region (see [24] for example). We choose an initial point $\alpha_0 \in \mathbf{R}^n$ and the algorithm proceeds as follows:

Projected gradient method

1. Compute $\alpha_{i+1} = \alpha_i + t\nabla f(\alpha_i)$.
2. Set $\alpha_{i+1} = p_A(\alpha_{i+1})$.
3. If gap $\leq \epsilon$ stop, otherwise go back to step 1.

Here, we have assumed that the objective function is differentiable (after smoothing). The method is only efficient if the projection step is numerically cheap. The complexity of each iteration then breaks down as follows:
Step 1. This requires an eigenvalue decomposition that is computed in $O(n^2)$ plus one matrix multiplication as described above. We note that a line search would be costly here because it would require multiple eigenvalue decompositions to recalculate the objective multiple times.
Step 2. This is a projection onto the region $A = \{\alpha^T y = 0,\ 0 \leq \alpha \leq C\}$ and can be solved explicitly by sorting the vector of entries, with cost $O(n \log n)$.
Stopping Criterion. We can compute a duality gap using the results of §2.4 where:

$$K_i = (K_0 + (Y\alpha_i)(Y\alpha_i)^T/4\rho)_+$$

is the candidate kernel at iteration i and we solve problem (1), which simply means solving a SVM problem with the positive semidefinite kernel K_i, and produces an upper bound on (7), hence a bound on the suboptimality of the current solution.
Complexity. The number of iterations required by this method to reach a target precision of ϵ grows as $O(1/\epsilon^2)$. See [25] for a complete discussion.

3.3 Analytic center cutting plane method

The analytic center cutting plane method (ACCPM) reduces the feasible region at each iteration using a new *cut* computed by evaluating a subgradient of the objective function at the analytic center of the current feasible set, until the volume of the reduced region converges to the target precision. This method does not require differentiability. We set $\mathcal{A}_0 = \{\alpha^T y = 0, 0 \leq \alpha \leq C\}$ which we can write as $\{A_0 \leq b_0\}$ to be our first localization set for the optimal solution. The method then works as follows (see [24] for a more complete treatment of cutting plane methods):

Analytic center cutting plane method

1. Compute α_i as the analytic center of $\mathcal{A}_i$ by solving:

$$\alpha_{i+1} = \underset{y \in \mathbf{R}^n}{\operatorname{argmin}} \ -\sum_{i=1}^{m} \log(b_i - a_i^T y)$$

where a_i^T represents the i^{th} row of coefficients from the left-hand side of $\{A_0 \leq b_0\}$.

2. Compute $\nabla f(x)$ at the center α_{i+1} and update the (polyhedral) localization set:

$$\mathcal{A}_{i+1} = \mathcal{A}_i \cup \{\nabla f(\alpha_{i+1})(\alpha - \alpha_{i+1}) \geq 0\}$$

3. If gap $\leq \epsilon$ stop, otherwise go back to step 1.

The complexity of each iteration breaks down as follows:
Step 1. This step computes the analytic center of a polyhedron and can be solved in $O(n^3)$ operations using interior point methods for example.
Step 2. This simply updates the polyhedral description. It includes the gradient computation which again is $O(n^2)$ plus one matrix multiplication.
Stopping Criterion. An upper bound is computed by maximizing a first order Taylor approximation of $f(\alpha)$ at α_i over all points in an ellipsoid that covers $\mathcal{A}_i$, which can be computed explicitly.
Complexity. ACCPM is provably convergent in $O(n \log(1/\epsilon)^2)$ iterations when using cut elimination which keeps the complexity of the localization set bounded. Other schemes are available with slightly different complexities: a bound of $O(n^2/\epsilon^2)$ is achieved in [26] using (cheaper) approximate centers for example.

3.4 Matlab Implementation

The two algorithms discussed here were implemented in Matlab for the cases of indefinite (IndefiniteSVM) and positive semidefinite (PerturbSVM) kernels and can be downloaded from the authors' webpages in a package called IndefiniteSVM. The ρ penalty parameter is one-dimensional in the implementation. This package makes use of LIBSVM [27] to produce suboptimality bounds and track convergence.

4 Experiments

In this section we compare the generalization performance of our technique to other methods applying SVM classification to indefinite similarity measures. We also examine classification performance using Mercer kernels. We conclude with experiments showing convergence of our algorithms. All experiments on Mercer kernels use the LIBSVM library.

4.1 Generalization with indefinite kernels

We compare our method for SVM classification with indefinite kernels to several kernel preprocessing techniques discussed earlier. The first three techniques perform spectral transformations on the indefinite kernel. The first, called *denoise* here, thresholds the negative eigenvalues to zero. The second transformation, called *flip*, takes the absolute value of all eigenvalues. The last transformation, *shift*, adds a constant to each eigenvalue making them all positive. See [14] for further details. We also implemented an SVM modification (denoted *Mod SVM*) suggested in [15] where a nonconvex quadratic objective function is made convex by replacing the indefinite kernel with the identity matrix. The kernel only appears in linear inequality constraints that separate the data. Finally, we compare our results with a direct use of SVM classification on the original indefinite

kernel (SVM converges but the solution is only a stationary point and is not guaranteed to be optimal).

We first experiment on data from the USPS handwritten digits database [28] using the indefinite simpson score and the one-sided tangent distance kernel to compare two digits. The tangent distance is a transformation invariant measure - it assigns high similarity between an image and slightly rotated or shifted instances - and is known to perform very well on this data set. Our experiments symmetrize the one-sided tangent distance using the square of the mean tangent distance defined in [6] and make it a similarity measure by negative exponentiation. We also consider the Simpson score for this task which is much cheaper to compute (a ratio comparing binary pixels). We finally analyze three data sets (diabetes, german and ala) from the UCI repository [29] using the indefinite sigmoid kernel.

The data is randomly divided into training and testing data. We apply 5-fold cross validation and use an accuracy measure (described below) to determine the optimal parameters C, ρ, and any kernel inputs. We then train a model with the full training set and optimal parameters and test on the independent test set.

Data Set	# Train	# Test	λ_{min}	λ_{max}
USPS-3-5-SS	767	773	-70.00	903.94
USPS-4-6-SS	829	857	-74.38	819.36
USPS-3-5-TD1	767	773	-0.03	32.70
USPS-4-6-TD1	829	857	-0.56	75.62
diabetes-sig	384	384	-584.64	6.88
german-sig	500	500	-0.32	24.99
a1a-sig	803	802	-1582.3	9.63

Table 1: Summary statistics for the various data sets used in our experiments. The USPS data comes from the USPS handwritten digits database, the other data sets are taken from the UCI repository. *SS* refers to the simpson kernel, *TD1* to the one-sided tangent distance kernel, and *sig* to the sigmoid kernel. Training and testing sets were divided randomly. Notice that the simpson and sigmoid kernels are mostly highly indefinite while the one-sided tangent distance kernel is nearly positive semidefinite. Statistics for sigmoid kernels refer to the optimal kernel parameterized under cross validation with Indefinite SVM. Spectrums are based on full kernel, i.e. combining training and testing data.

Table 1 provides summary statistics for these data sets, including the minimum and maximum eigenvalues of the training similarity matrices. We observe that the simpson and sigmoid kernels are highly indefinite while the the one-sided tangent distance kernel is nearly positive semidefinite. The spectrum of sigmoid kernels varies greatly across examples because it is very sensitive to the sigmoid kernel parameters. Table 2 compares accuracy and recall for denoise, flip, shift, modified SVM, direct SVM and the indefinite SVM algorithm described in this work.

We observe that indefinite SVM performs favorably using highly indefinite similarity measures (aside from recall for *a1a*), while, somewhat predictably, the performance is equal across all methods when the kernels are not highly indefinite (the proxy kernel is very close to the original one). As expected, classification using the tangent distance outperforms classification with the simpson

score but, as mentioned above, the simpson score is cheaper to compute. We also note that other documented classification results on this USPS data set perform multi-classification, while here we only perform binary classification.

4.2 Generalization with Mercer kernels

Using this time linear and gaussian (Mercer) kernels on the USPS data set, we now compare classification performance using regular SVM and the penalized kernel learning problem (10) of Section 2.5, which we call PerturbSVM here. We also test these two techniques on Mercer kernels formed using noisy data sets (created by adding uniformly distributed noise in the USPS data set), in which case PerturbSVM can be seen as optimally denoised support vector classification. We again cross-validate on a training set and test on the same independent group of examples used in the experiments above. The results are summarized in Table 3.

These results show that PerturbSVM performs at least as well in almost all cases. As expected, noise decreased generalization performance in all experiments. When performance is comparable, ρ is chosen to be very high implying no perturbation is necessary. Except in the *USPS-3-5-gaussian* example, the value of ρ selected was not the highest possible for each test where PerturbSVM outperforms SVM in at least one measure; this implies that the support vectors were perturbed to improve classification. Overall, when zero or moderate noise is present, PerturbSVM does improve performance over regular SVM. When too much noise is present however (pixel data with range in $[-1, 1]$ was modified with uniform noise in $[-2, 2]$ before the data was normalized to [0,1]), the performance of both techniques is comparable.

4.3 Convergence

We ran our two algorithms on data sets created by randomly perturbing the four USPS data sets used above. Average results and standard deviation are displayed in Figure 1 in semilog scale (note that the codes were not stopped here to show convergence, but duality gap improvement targets are usually much smaller than 10^{-8}). As expected, ACCPM converges much faster (in fact linearly) to a higher precision while each iteration requires solving a linear program of size n. The gradient projection method converges faster in the beginning but stalls at higher precision, however each iteration only requires a rank one update on an eigenvalue decomposition.

We finally examine the computing time of Indefinite SVM using the projected gradient method. Figure 2 shows total runtime (left) and average iteration runtime (right) for varying problem dimensions on an example from the USPS data with simpson kernel. Note that the number of iterations required varies widely (between 200 and 1000 iterations in this experiment) as a function of ρ, C, the chosen kernel and the stepsize.

5 Conclusion

We have proposed a technique for support vector machine classification with indefinite kernels, using a proxy kernel which can be computed explicitly. We also show how this framework can be used to improve generalization performance with potentially noisy Mercer kernels. We give two provably convergent algorithms for solving this problem on relatively large data sets. Our initial experiments show that our method fares quite favorably compared to other techniques handling

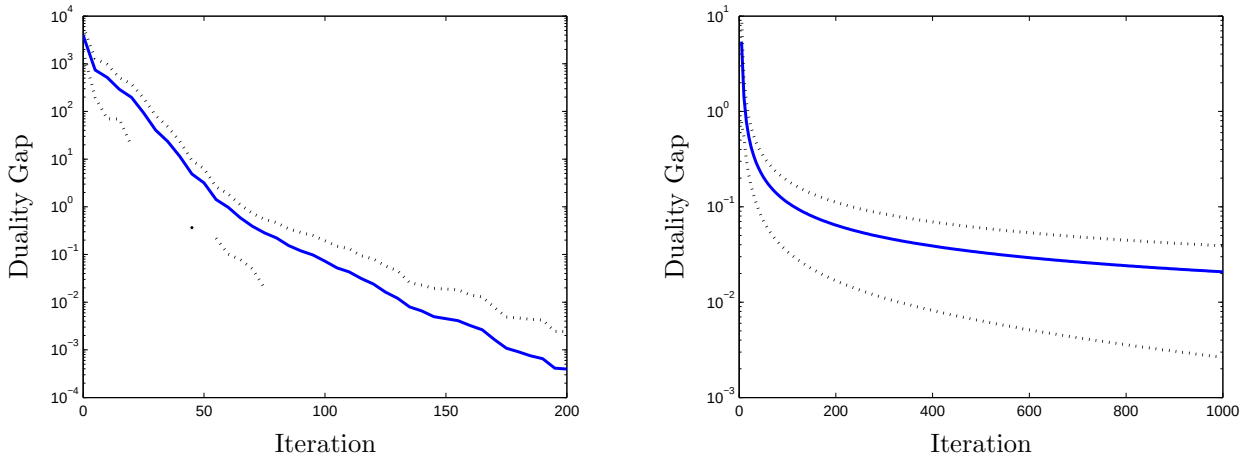

Figure 1: Convergence plots for ACCPM (left) & projected gradient method (right) on randomly perturbed USPS data sets (average gap versus iteration number, dashed lines at plus and minus one standard deviation). ACCPM converges linearly to a higher precision while the gradient projection method converges faster in the beginning but stalls at a higher precision.

indefinite kernels in the SVM framework and, in the limit, provides a clear interpretation for some of these heuristics.

Acknowledgments

We are very grateful to Mátyás Sustik for his rank-one update eigenvalue decomposition code. The authors would also like to acknowledge support from NSF grant DMS-0625352, ONR grant number N00014-07-1-0150, a Peek junior faculty fellowship and a gift from Google, Inc.

References

[1] C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[2] A. Zamolotskikh and P. Cunningham. An assessment of alternative strategies for constructing emd-based kernel functions for use in an svm for image classification. *Technical Report UCD-CSI-2007-3*, 2004.

[3] H. Saigo, J. P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[4] G. R. G. Lanckriet, N. Cristianini, M. I. Jordan, and W. S. Noble. Kernel-based integration of genomic data using semidefinite programming. 2003. citeseer.ist.psu.edu/648978.html.

[5] P. Y. Simard, Y. A. Le Cun, J. S. Denker, and B. Victorri. Transformation invariance in pattern recognition-tangent distance and tangent propogation. *Lecture Notes in Computer Science*, 1524:239–274, 1998.

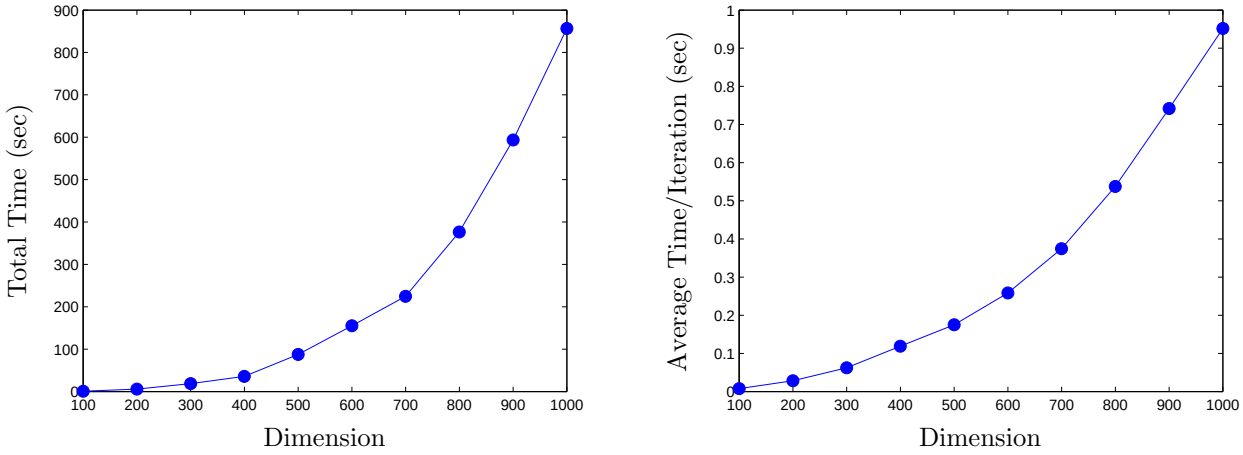

Figure 2: Total time versus dimension (left) & average time per iteration versus dimension (right) both using projected gradient Indefinite SVM. The number of iterations for convergence varies from 200 for the smallest dimension to 1000 for the largest dimension in this example which uses a simpson kernel on USPS data.

[6] B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. *Proc. of the 16th Int. Conf. on Pattern Recognition*, 2:864–868, 2002.

[7] Marco Cuturi. Permanents, transport polytopes and positive definite kernels on histograms. *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

[8] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[9] C. S. Ong, A. J. Smola, and R. C. Williamson. Learning the kernel with hyperkernels. *Journal of Machine Learning Research*, 6:1043–1071, 2005.

[10] F. R. Bach, G. R. G. Lanckriet, and M. I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[11] S. Sonnenberg, G. Rätsch, C. Schäfer, and B. Schölkopf. Large scale multiple kernel learning. *Journal of Machine Learning Research*, 7:1531–1565, 2006.

[12] B. Haasdonk. Feature space interpretation of svms with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4), 2005.

[13] K. P. Bennet and E. J. Bredensteiner. Duality and geometry in svm classifiers. *Proceedings of the 17th International conference on Machine Learning*, pages 57–64, 2000.

[14] G. Wu, E. Y. Chang, and Z. Zhang. An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. *Proceedings of the 22nd International Conference on Machine Learning*, 2005.

[15] H.-T. Lin and C.-J. Lin. A study on sigmoid kernel for svm and the training of non-psd kernels by smo-type methods. 2003.

[16] A. Woźnica, A. Kalousis, and M. Hilario. Distances and (indefinite) kernels for set of objects. *Proceedings of the 6th International Conference on Data Mining*, pages 1151–1156, 2006.

[17] Ronny Luss and Alexandre d'Aspremont. Support vector machine classification with indefinite kernels. *Neural Information Processing Systems*, 2007.

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

[19] N. Higham. Computing the nearest correlation matrix-a problem from finance. *IMA Journal of Numerical Analysis*, 22:329–343, 2002.

[20] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer, 1993.

[21] J. W. Demmel. *Applied Numerical Linear Algebra*. Siam, 1997.

[22] B. Kulis, M. Sustik, and I. Dhillon. Learning low-rank kernel matrices. *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[23] M. Overton. Large-scale optimization of eigenvalues. *SIAM Journal on Optimization*, 2(1):88–120, 1992.

[24] D. Bertsekas. *Nonlinear Programming, 2nd Edition*. Athena Scientific, 1999.

[25] Y. Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2003.

[26] J.-L. Goffin and J.-P. Vial. Convex nondifferentiable optimization: A survey focused on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[27] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[28] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5), 1994.

[29] A. Asuncion and D.J. Newman. *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Data Set	Measure	Denoise	Flip	Shift	Mod SVM	SVM	Indefinite SVM
USPS-3-5-SS	Accuracy	95.47	95.73	90.43	95.21	69.47	**95.99**
	Recall	94.50	95.46	92.11	95.22	67.94	**96.41**
USPS-3-5-TD1	Accuracy	**97.41**	**97.41**	**97.41**	97.28	**97.41**	**97.41**
	Recall	**96.41**	**96.41**	**96.41**	95.69	**96.41**	**96.41**
USPS-4-6-SS	Accuracy	**98.25**	**98.25**	94.28	97.90	84.36	98.0
	Recall	98.87	**99.32**	93.68	98.65	81.72	**99.32**
USPS-4-6-TD1	Accuracy	**98.60**	**98.60**	**98.60**	98.25	**98.60**	**98.60**
	Recall	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
diabetes-sig	Accuracy	74.22	71.62	**74.74**	72.14	73.44	71.62
	Recall	81.20	83.20	84.00	76.00	79.60	**86.40**
german-sig	Accuracy	71.00	70.80	**76.40**	73.20	71.80	**76.40**
	Recall	70.86	70.29	90.57	75.14	72.29	**90.86**
a1a-sig	Accuracy	81.42	80.92	81.92	80.92	81.42	**82.92**
	Recall	**80.20**	61.93	53.30	69.54	**80.20**	38.07

Table 2: Indefinite SVM performs favorably for the highly indefinite simpson and sigmoidal kernels. Performance is equivalent for the nearly positive semidefinite one-sided tangent distance kernel. The performance measures are: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$ and Recall $= \frac{TP}{TP+FN}$.

		Unperturbed		Noisy	
Data Set	Measure	SVM	Perturb SVM	SVM	Indefinite SVM
USPS-3-5-linear	Accuracy	95.60	**96.51**	89.13	**90.17**
	Recall	94.74	**96.41**	85.89	**90.91**
USPS-4-6-linear	Accuracy	**98.60**	**98.60**	**95.80**	**95.80**
	Recall	**98.87**	**98.87**	**95.71**	**95.71**
USPS-3-5-gaussian	Accuracy	97.41	**97.93**	**88.88**	**88.88**
	Recall	97.13	**98.09**	89.00	**89.47**
USPS-4-6-gaussian	Accuracy	**99.18**	**99.18**	95.45	**95.57**
	Recall	**99.32**	**99.32**	95.71	**95.92**

Table 3: Performance measures for USPS data using linear and gaussian kernels. *Unperturbed* refers to classification of the original data and *Noisy* refers to classification of data that is perturbed by uniform noise. Perturb SVM perturbs the support vectors to improve generalization. However, performance is lower for both techniques in the presence of high noise.