Math. Prog. Comp. (2009) 1:97–118
DOI 10.1007/s12532-009-0005-5

FULL LENGTH PAPER

Support vector machine classification with indefinite kernels

Ronny Luss · Alexandre d'Aspremont

Received: 26 March 2009 / Accepted: 30 July 2009 / Published online: 25 August 2009

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 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 standard data sets.

Mathematics Subject Classification (2000) Primary 62H30; Secondary 90C25 · 68T05

1 Introduction

Support vector machines (SVM) have become a central tool for solving binary classification problems. A critical step in support vector machine classification is choosing a suitable kernel matrix, which measures similarity between data points and must be positive semidefinite because it is formed as the Gram matrix of data points in a reproducing kernel Hilbert space. This positive semidefinite condition on kernel matrices is also known as Mercer's condition in the machine learning literature. The classification problem then becomes a linearly constrained quadratic program. Here, we present an

R. Luss (✉) · A. d'Aspremont
ORFE Department, Princeton University, Princeton, NJ 08544, USA
e-mail: rluss@alumni.princeton.edu

A. d'Aspremont
e-mail: aspremon@princeton.edu

algorithm for SVM classification using indefinite kernels,[1] i.e. kernel matrices formed using similarity measures which are not positive semidefinite.

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 [25] for a discussion.)

In particular, an application of classification with indefinite kernels to image classification using Earth Mover's Distance was discussed in Zamolotskikh and Cunningham [35]. 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 Saigo et al. [29] or have been transformed into positive semidefinite kernels with good empirical performance (see [19], for example). Tangent distance similarity measures, as described in Simard et al. [31] or Haasdonk and Keysers [13], 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 evaluating the exact positive kernel is too high and a proxy (and not necessarily positive semidefinite) kernel has to be used instead (see [9], for example). In both cases, our method allows us to bypass these limitations. Our objective here is to derive efficient algorithms to directly use these indefinite similarity measures for classification.

Our work closely follows, in spirit, recent results on kernel learning (see [20] or [26]), 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, Bach et al. [2] adapted the SMO algorithm to solve the case where the kernel is written as a positively weighted combination of other kernels. In our setting 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 Sonnenberg et al. [32], who formulate the multiple kernel learning problem of Bach et al. [2] as a semi-infinite linear program and solve it with a column generation technique similar to the analytic center cutting plane method (ACCPM) 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: Haasdonk [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

[1] A preliminary version of this paper appeared in the proceedings of the Neural Information Processing Systems (NIPS) 2007 conference and is available at http://books.nips.cc/nips20.html.

embedded in the pE space. The nonseparable case is handled in the same manner using reduced convex hulls. (See [3] 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 [34,35] 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 Lin and Lin [22] 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 [12,33]) 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, and hence can be solved efficiently with guaranteed complexity bounds.

The paper is organized as follows. In Sect. 2 we formulate our main classification result and detail its interpretation as a penalized kernel learning problem. In Sect. 3 we describe three algorithms for solving this problem. Section 4 discusses several extensions of our main results. Finally, in Sect. 5, we test the numerical performance of these methods on various data sets.

Notation We write $\mathbf{S}^n$ ($\mathbf{S}^n_+$) to denote 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. Given a vector x, $\|x\|_1 = \sum |x_i|$.

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 let $y \in \mathbf{R}^n$ be the vector of labels, with $Y = \mathbf{diag}(y)$, the matrix with diagonal y. We formulate the kernel learning problem as in Lanckriet et al. [20], 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

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

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

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

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 ρ on the distance between the proxy kernel and the original indefinite similarity matrix and

solve instead

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

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

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

in the variables $K \in \mathbf{S}^n$ and $\alpha \in \mathbf{R}^n$.

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 [5] 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:*

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

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

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

$$\min_{\{K\succeq 0\}} \alpha^T e + \rho\left(\mathbf{Tr}(K^TK) - 2\,\mathbf{Tr}\left(K^T\left(K_0 + \frac{1}{4\rho}(Y\alpha)(Y\alpha)^T\right)\right) + \mathbf{Tr}(K_0^TK_0)\right)$$

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

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

in the variable $K \in \mathbf{S}^n$, where we have dropped the remaining constants from the objective. This is the projection of the matrix $K_0 + (Y\alpha)(Y\alpha)^T/(4\rho)$ 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

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

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 $K_0 + (Y\alpha)(Y\alpha)^T/(4\rho)$ be VDV^T, we get $K^* = VD_+V^T$, and with v_i the i^{th} column of V, we can write

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

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

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

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

A reformulation of problem (4) appears in Chen and Ye [7] where the authors move the inner minimization problem to the constraints and get the following semi-infinite quadratically constrained linear program (SIQCLP):

$$\begin{array}{ll}\text{maximize} & t\\ \text{subject to} & \alpha^T y = 0,\ 0 \le \alpha \le C\\ & t \le \alpha^T e - \frac{1}{2}\,\mathbf{Tr}\left(K(Y\alpha)(Y\alpha)^T\right) + \rho\|K - K_0\|_F^2 \quad \forall K \succeq 0.\end{array} \tag{8}$$

In Sect. 3, we describe algorithms to solve our eigenvalue optimization problem in (7), as well as an algorithm from Chen and Ye [7] that solves the different formulation in (8), for completeness.

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 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 the quadratic minimization problem

$$\begin{array}{l} \text{minimize } \frac{1}{2}(e-\delta+\mu+y\nu)^T(YKY)^{-1}(e-\delta+\mu+y\nu)+C\mu^T e \\ \text{subject to } \delta,\ \mu \geq 0. \end{array} \tag{9}$$

Substituting (9) for the inner maximization in problem (3) allows us to write a joint minimization problem

$$\begin{array}{l} \text{minimize } \mathbf{Tr}\left(K^{-1}(Y^{-1}(e-\delta+\mu+y\nu))(Y^{-1}(e-\delta+\mu+y\nu))^T\right)/2 \\ \qquad\qquad +C\mu^T e+\rho\|K-K_0\|_F^2 \\ \text{subject to } K \succeq 0,\ \delta,\ \mu \geq 0 \end{array} \tag{10}$$

in the variables $K \in \mathbf{S}^n$, $\delta, \mu \in \mathbf{R}^n$ and $\nu \in \mathbf{R}$. This is a quadratic program in the variables δ, 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 (10) allows us to compute an upper bound on the optimum value of problem (4) by solving a simple quadratic program in the variables δ, μ, ν. This result can then be used to bound the duality gap in (7) and track convergence.

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 feasible 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 less predictable performance in practice. We then show how to apply ACCPM, whose iterations are numerically more complex but which converges linearly. For completeness, we also describe an exchange method from Chen and Ye [7] used to

solve problem (8), where the numerical bottleneck is a quadratically constrained linear program solved at each iteration.

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 [16], for example). We replace the max by a continuously differentiable $\frac{\epsilon}{2}$-approximation as follows:

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

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

$$\frac{\partial\lambda_i(X(\alpha))}{\partial\alpha} = v_i^T \frac{\partial X(\alpha)}{\partial\alpha} v_i \tag{11}$$

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

3.1 Computing proxy kernels

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

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

we now briefly recall how v_i and $\lambda_i(X(\alpha))$ can be computed efficiently in this case (see [10] for further details). We refer the reader to Kulis et al. [18] 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 [27] for a discussion), but a subgradient can be used instead of the gradient in all the algorithms detailed here. Lewis [21] shows how to compute an approximate subdifferential of the k-th largest eigenvalue of a symmetric matrix. This can then be used to form a regular subgradient of the objective function in (7) which is concave by construction.

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 [4], for example). We choose an initial point $\alpha_0 \in \mathbf{R}^n$ and the algorithm proceeds as in Algorithm 1.

Algorithm 1 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. Experiments below use a stepsize of $5/k$ for IndefiniteSVM and $10/k$ for PerturbSVM (described in Sect. 4.3) where k is the iteration number. A good stepsize is crucial to performance, and must be chosen separately for each data set as there is no rule of thumb. 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 Sect. 2.4 where

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

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 Nesterov [24] for a complete discussion.

3.3 Analytic center cutting plane method

The analytic center cutting plane method 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 $\mathcal{L}_0 = \{x \in \mathbf{R}^n \mid x^T y = 0, 0 \leq x \leq C\}$, which we can write as $\{x \in \mathbf{R}^n \mid A_0 x \leq b_0\}$, to be our first localization set for the optimal solution. The method is described in Algorithm 2 (see [4] for a more complete treatment of cutting plane methods).

Algorithm 2 Analytic center cutting plane method

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

$$x_{i+1} = \underset{x \in \mathbf{R}^n}{\operatorname{argmin}} \; -\sum_{i=1}^{m} \log\left(b_i - a_i^T x\right)$$

where a_i^T represents the i^{th} row of coefficients from the left-hand side of $\{x \in \mathbf{R}^n \mid A_i x \leq b_0\}$.

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

$$\mathcal{L}_{i+1} = \mathcal{L}_i \cap \left\{\nabla f(x_{i+1})(x - x_{i+1}) \geq 0\right\}$$

where f is objective in problem (7).

3: If $m \geq 3n$, reduce the number of constraints to $3n$.

4: 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.
Step 3. This step requires ordering the constraints according to their relevance in the localization set. One relevance measure for the j^{th} constraint at iteration i is

$$\frac{a_j^T \nabla^2 f(x_i)^{-1} a_j}{(a_j^T x_i - b_j)^2} \tag{12}$$

where f is the objective function of the analytic center problem. Computing the hessian is easy: it requires matrix multiplication of the form $A^T D A$ where A is $m \times n$ (matrix multiplication is kept inexpensive in this step by pruning redundant constraints) and D is diagonal. Restricting the number of constraints to $3n$ is a rule of thumb; raising this limit increases the per iteration complexity while decreasing it increases the required number of iterations.

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 Goffin and Vial [11] using (cheaper) approximate centers, for example.

3.4 Exchange method for SIQCLP

The algorithm considered in Chen and Ye [7] in order to solve problem (8) falls under a class of algorithms called exchange methods (as defined in [14]). These methods iteratively solve problems constrained by a finite subset of the infinitely many constraints, where the solution at each iterate gives an improved lower bound to the maximization problem. The subproblem solved at each iteration here is

$$\begin{aligned} &\text{maximize } t \\ &\text{subject to } \alpha^T y = 0, 0 \le \alpha \le C \\ &\qquad t \le \alpha^T e - \tfrac{1}{2}\,\mathbf{Tr}\left(K_i(Y\alpha)(Y\alpha)^T\right) + \rho\|K_i - K_0\|_F^2 \quad i = 1, \ldots, p \end{aligned} \tag{13}$$

where p is the number of constraints used to approximate the infinitely many constraints of problem (8). Let (t_1, α_1) be an initial solution found by solving problem (13) with $p = 1$ and $K_1 = (K_0)_+$, where K_0 is the input indefinite kernel. The algorithm proceeds as in Algorithm 3 below.

Algorithm 3 Exchange method

1: Compute K_{i+1} by solving the inner minimization problem of (4) as a function of α_i.
2: Stop if

$$\alpha_i^T e - \frac{1}{2}\,\mathbf{Tr}\left(K_{i+1}(Y\alpha_i)(Y\alpha_i)^T\right) + \rho\|K_{i+1} - K_0\|_F^2 \ge t_i.$$

3: Solve problem (13) with an additional constraint using K_{i+1} to get (t_{i+1}, α_{i+1}) and go back to step 1.

The complexity of each iteration breaks down as follows:

Step 1. This step can be solved analytically using Theorem 1. An efficient calculation of K_{i+1} can be made as in the other algorithms above using an $O(n^2)$ procedure plus one matrix multiplication.
Step 2 (Stopping Criterion). The previous point (t_i, α_i) is optimal if it is feasible with respect to the new constraint, in which case it is feasible for the infinitely many constraints of the original problem (8) and hence also optimal.
Step 3. This step requires solving a QCLP with a number of quadratic constraints equivalent to the number of iterations. As shown in Chen and Ye [7], the QCLP can be written as a regularized version of the multiple kernel learning (MKL) problem

from Lanckriet et al. [20], where the number of constraints here is equivalent to the number of kernels in MKL. Efficient methods to solve MKL with many kernels is an active area of research, most recently in Rakotomamonjy et al. [28]. There, the authors use a gradient method to solve a reformulation of problem (13) as a smooth maximization problem. Each objective value and gradient computation requires computing a support vector machine, hence each iteration requires several SVM computations which can be speeded up using warm-starting. Furthermore, Chen and Ye [7] prune inactive constraints at each iteration in order to decrease the number of constraints in the QCLP.

Complexity No rate of convergence is known for this algorithm, but the duality gap given in Chen and Ye [7] is shown to monotonically decrease.

3.5 Matlab implementation

The first 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 the LIBSVM code of Chang and Lin [6] to produce suboptimality bounds and track convergence. A Matlab implementation of the exchange method (due to the authors of [7]) that uses MOSEK [23] to solve problem (13) is compared against the projected gradient method in Sect. 5.

4 Extensions

In this section, we extend our results to other kernel methods, namely support vector regressions and one-class support vector machines. In addition, we apply our method to using Mercer kernels and show how to use more general penalties in our formulation.

4.1 SVR with indefinite kernels

The practicality of indefinite kernels in SVM classification similarly motivates using indefinite kernels in support vector regression (SVR). We here extend the formulations in Sect. 2 to SVR with linear ϵ-insensitive loss

$$\omega_C(K) = \max_{\{-C\leq\alpha\leq C, \alpha^T e=0\}} \alpha^T y - \epsilon|\alpha| - \mathbf{Tr}(K\alpha\alpha^T)/2 \tag{14}$$

where $\alpha \in \mathbf{R}^n$ and C is the SVR penalty parameter. The indefinite SVR formulation follows directly as in Sect. 2.2 and the optimal kernel is learned by solving

$$\max_{\{\alpha^T e=0, -C\leq\alpha\leq C\}} \min_{\{K\succeq 0\}} \alpha^T y - \epsilon|\alpha| - \frac{1}{2}\mathbf{Tr}\left(K\alpha\alpha^T\right) + \rho\|K - K_0\|_F^2 \tag{15}$$

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. The following corollary to Theorem 1 provides the solution to the inner minimization problem in (15)

Corollary 2 *Given a similarity matrix $K_0 \in S^n$ and a vector $\alpha \in \mathbf{R}^n$ of support vector coefficients, the optimal kernel in problem (15) can be computed explicitly as*

$$K^* = \left(K_0 + \alpha\alpha^T/(4\rho)\right)_+ \tag{16}$$

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

The proof follows directly as in Theorem 1; the slight difference is that the vector of labels y does not appear in the optimal kernel. Plugging in (16) into (15), the resulting formulation can be rewritten as the convex eigenvalue optimization problem

$$\begin{array}{l} \text{maximize } \alpha^T y - \epsilon|\alpha| - \frac{1}{2}\sum_i \max\left(0, \lambda_i\left(K_0 + \alpha\alpha^T/(4\rho)\right)\right)\left(\alpha^T v_i\right)^2 \\ \qquad +\rho\sum_i \left(\max\left(0, \lambda_i(K_0 + \alpha\alpha^T/4\rho)\right)\right)^2 \\ \qquad -2\rho\sum_i \mathbf{Tr}\left((v_i v_i^T)K_0\right)\max\left(0, \lambda_i\left(K_0 + \alpha\alpha^T/(4\rho)\right)\right) + \rho\,\mathbf{Tr}(K_0 K_0) \\ \text{subject to } \alpha^T e = 0, -C \leq \alpha \leq C \end{array} \tag{17}$$

in the variable $\alpha \in \mathbf{R}^n$. Again, a proxy kernel given by (16) can be produced from any feasible solution $\alpha \in \mathbf{R}^n$. Plugging the proxy kernel into problem (15) allows us to compute an upper bound on the optimum value of problem (15) by solving a SVR problem.

4.2 One-class SVM with indefinite kernels

The same reformulation can also be applied to one-class support vector machines which have the formulation (see [30])

$$\omega_\nu(K) = \max_{\{0\leq\alpha\leq\frac{1}{\nu l}, \alpha^T e = 1\}} -\mathbf{Tr}\left(K\alpha\alpha^T\right)/2 \tag{18}$$

where $\alpha \in \mathbf{R}^n$, ν is the one-class SVM parameter, and l is the number of training points. The indefinite one-class SVM formulation follows again as done for binary SVM and SVR; the optimal kernel is learned by solving

$$\max_{\{\alpha^T e = 1, 0\leq\alpha\leq\frac{1}{\nu l}\}} \min_{\{K\succeq 0\}} -\frac{1}{2}\mathbf{Tr}(K\alpha\alpha^T) + \rho\|K - K_0\|_F^2 \tag{19}$$

in the variables $K \in \mathbf{S}^n$ and $\alpha \in \mathbf{R}^n$. The inner minimization problem is identical to that of indefinite SVR and the optimal kernel has the same form as given in Corollary 2. Plugging (16) into (19) gives another convex eigenvalue optimization problem

$$\begin{array}{l}\text{maximize } -\frac{1}{2}\sum_i \max\left(0, \lambda_i\left(K_0+\alpha\alpha^T/4\rho\right)\right)\left(\alpha^T v_i\right)^2 \\ \qquad +\rho\sum_i\left(\max\left(0,\lambda_i\left(K_0+\alpha\alpha^T/(4\rho)\right)\right)\right)^2 \\ \qquad -2\rho\sum_i \mathbf{Tr}\left(\left(v_i v_i^T\right)K_0\right)\max\left(0,\lambda_i\left(K_0+\alpha\alpha^T/(4\rho)\right)\right)+\rho\,\mathbf{Tr}(K_0K_0) \\ \text{subject to } \alpha^T e=1, 0\le\alpha\le\frac{1}{vl}\end{array} \tag{20}$$

in the variable $\alpha \in \mathbf{R}^n$, which is identical to (17) without the first two terms in the objective and slightly different constraints. The algorithm follows almost directly the same as above for the indefinite SVR formulation.

4.3 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, the proxy kernel K^* in Theorem 1 simplifies to a rank-one update of K_0

$$K^* = K_0 + (Y\alpha^*)(Y\alpha^*)^T/(4\rho) \tag{21}$$

whereas, for indefinite K_0, the solution was to project this matrix on the cone of positive semidefinite matrices. Plugging (21) into problem (4) gives:

$$\max_{\{\alpha^T y=0,\ 0\le\alpha\le C\}} \alpha^T e - \frac{1}{2}\mathbf{Tr}\left(K_0(Y\alpha)(Y\alpha)^T\right) - \frac{1}{16\rho}\sum_{i,j}(\alpha_i\alpha_j)^2, \tag{22}$$

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 Sect. 5.

4.4 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

$$\max_{\{\alpha^T y=0, 0\le\alpha\le C\}} \min_{\{K\succeq 0\}} \alpha^T e - \frac{1}{2}\mathbf{Tr}\left(K(Y\alpha)(Y\alpha)^T\right) + \sum_{i,j} H_{ij}(K_{ij}-K_{0ij})^2 \quad (23)$$

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 3 *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 (23) has the explicit form*

$$K^* = W^{-1/2}\left(\left(W^{1/2}\left(K_0 + \frac{1}{4}(W^{-1}Y\alpha^*)(W^{-1}Y\alpha^*)^T\right)W^{1/2}\right)_+\right)W^{-1/2} \quad (24)$$

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

Proof The inner minimization problem to problem (23) can be written out as

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

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

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

where $\circ$ denotes the Hardamard product, $(A \circ B)_{ij} = a_{ij}b_{ij}$, $(H^{1/2})_{ij} = H_{ij}^{1/2}$, and $\left(\frac{1}{4H}\right)_{ij} = \frac{1}{4H_{ij}}$. 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 Higham [15]. □

Notice that Theorem 3 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 3 simplifies to a rank-one update of K_0:

$$K^* = K_0 + \frac{1}{4}\left(W^{-1/2}Y\alpha\right)\left(W^{-1/2}Y\alpha\right)^T \quad (25)$$

where no projection is required.

5 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.

5.1 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 Wu et al. [34] for further details. We also implemented an SVM modification (denoted *Mod SVM*) suggested in [22] 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 not guaranteed to be optimal).

We first experiment on data from the USPS handwritten digits database Hull [17] 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 Haasdonk and Keysers [13] 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 [1] using the indefinite sigmoid kernel.

Table 1 Summary statistics for the various data sets used in our experiments

Data set	# Train	# Test	$\lambda_{\min}$	$\lambda_{\max}$
USPS-3-5-SS	767	773	−70.00	903.94
USPS-3-5-TD1	767	773	−0.31	764.72
USPS-4-6-SS	829	857	−74.38	819.36
USPS-4-6-TD1	829	857	−0.72	771.07
Diabetes-sig	384	384	−0.65	211.62
German-sig	500	500	−928.10	8.50
A1a-sig	803	802	−0.01	84.44

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 kernels are mostly highly indefinite while the one-sided tangent distance kernel is nearly positive semidefinite. The sigmoid kernel is highly indefinite depending on the parametrization. Statistics for sigmoid kernels refer to the optimal kernel parameterized under cross validation with Indefinite SVM. Spectrums are based on the full kernel, i.e. combining training and testing data

Table 2 Indefinite SVM performs favorably for the highly indefinite Simpson kernels

Data set	Measure	Denoise	Flip	Shift	Mod SVM	SVM	Indefinite SVM
USPS-3-5-SS	Accuracy	95.47	95.21	93.27	**96.12**	69.47	95.73
	Recall	94.50	94.50	94.98	96.17	67.94	**97.13**
	Average	94.98	94.86	94.12	96.15	68.71	**96.43**
USPS-3-5-TD1	Accuracy	**98.58**	98.45	**98.58**	98.19	**98.58**	98.45
	Recall	**98.56**	98.33	**98.56**	97.85	**98.56**	98.33
	Average	**98.57**	98.39	**98.57**	98.02	**98.57**	98.39
USPS-4-6-SS	Accuracy	**98.60**	98.25	96.73	**98.60**	84.36	98.25
	Recall	99.32	99.32	96.61	99.32	81.72	**99.77**
	Average	98.96	98.79	96.67	98.96	83.04	**99.01**
USPS-4-6-TD1	Accuracy	**99.30**	**99.30**	99.18	99.18	**99.30**	**99.30**
	Recall	**99.77**	**99.77**	99.55	99.55	**99.77**	**99.77**
	Average	**99.54**	**99.54**	99.37	99.37	**99.54**	**99.54**
Diabetes-sig	Accuracy	74.48	74.74	76.56	76.04	73.70	**77.08**
	Recall	78.40	76.80	**89.60**	78.40	76.40	89.20
	Average	76.44	75.77	83.08	77.22	75.05	**83.14**
German-sig	Accuracy	70.40	70.40	**75.60**	72.60	69.40	62.80
	Recall	78.00	78.00	46.67	66.00	80.00	**85.33**
	Average	74.20	74.20	61.13	69.30	**74.70**	74.07
A1a-sig	Accuracy	74.06	76.18	75.69	78.55	75.69	**82.92**
	Recall	87.31	87.82	87.31	**89.34**	87.82	81.73
	Average	80.69	82.00	81.50	**83.95**	81.75	82.32

Performance is comparable for the nearly positive semidefinite one-sided tangent distance kernel. Comparable performance with sigmoid kernels is more consistent with indefinite SVM across data sets. The performance measures are: Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$, Recall $= \frac{TP}{TP+FN}$, and Average $=$ (Accuracy $+$ Recall)/2. Bold values indicate the best performance across all measures

The data is randomly divided into training and testing data. We apply fivefold cross validation and use an average of the accuracy and recall measures (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.

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 are highly indefinite, while 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, recall, and their average for denoise, flip, shift, modified SVM, direct SVM and the indefinite SVM algorithm described in this work.

Based on the interpretation from Sect. 2.3, Indefinite SVM should be expected to perform at least as well as denoise; if denoise were a good transformation, then cross-validation over ρ should choose a high penalty that makes Indefinite SVM and denoise nearly equivalent. The rank-one update provides more flexibility for the transformation and similarities concerning data points x_i that are easily classified ($\alpha_i = 0$) are

not modified by the rank-one update. Further interpretation for the specific rank-one update is not currently known. However, Chen et al. [8] recently proposed spectrum modifications in a similar manner to Indefinite SVM. Rather than perturb the entire indefinite similarity matrix, they perturb the spectrum directly allowing improvements over the denoise as well as flip transformations. They also note that Indefinite SVM might perform better on sparse kernels because the rank-one update may then allow inference of hidden relationships.

We observe that Indefinite SVM performs comparably on all USPS examples (slightly better for the highly indefinite Simpson kernels), which are relatively easy classification problems. 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. Classification of the UCI data sets with sigmoid kernels is more difficult (as demonstrated by lower performance measures). Indefinite SVM here is the only technique that outperforms in at least one of the measures across all three data sets.

5.2 Generalization with Mercer kernels

Using this time linear and gaussian (both positive semidefinite, i.e. Mercer) kernels on the USPS data set, we now compare classification performance using regular SVM and the penalized kernel learning problem (22) of Sect. 4.3, which we call PerturbSVM here. We also test these two techniques on positive semidefinite kernels formed using noisy USPS data sets (created by adding uniformly distributed noise in $[-1,1]$ to each pixel before normalizing to [0,1]), in which case PerturbSVM can be seen as optimally denoised support vector machine classification. We again cross-validate on a training set and test on the same independent group of examples used in the experiments above. Optimal parameters from classification of unperturbed data were used to train classifiers for perturbed data. 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. Except in the *USPS-4-6-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 as shown. When too much noise is present, however, (for example, pixel data with range in $[-1, 1]$ was modified with uniform noise in $[-2, 2]$ before being normalized to [0, 1]), the performance of both techniques is comparable.

5.3 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 Fig. 1 in semilog scale (note that the codes were not stopped here and that the target duality gap improvement is usually much smaller than 10^{-8}). As expected, ACCPM converges

Table 3 Performance measures for USPS data using linear and gaussian kernels

Data Set	Measure	Unperturbed		Noisy	
		SVM	Perturb SVM	SVM	Perturb SVM
USPS-3-5-linear	Accuracy	**96.25**	96.12	90.27	**93.16**
	Recall	95.69	**95.93**	90.00	**92.87**
	Average	95.97	**96.03**	90.14	**93.01**
USPS-4-6-linear	Accuracy	**99.07**	**99.07**	97.39	**97.97**
	Recall	99.10	**99.32**	97.34	**98.13**
	Average	99.08	**99.19**	97.36	**98.05**
USPS-3-5-gaussian	Accuracy	**97.67**	97.54	92.11	**93.57**
	Recall	**98.09**	97.37	91.27	**92.89**
	Average	**97.88**	97.46	91.69	**93.23**
USPS-4-6-gaussian	Accuracy	99.18	**99.30**	**98.00**	97.99
	Recall	**99.55**	**99.55**	98.15	**98.19**
	Average	99.37	**99.42**	98.08	**98.09**

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. Bold values indicate the best performance across all measures

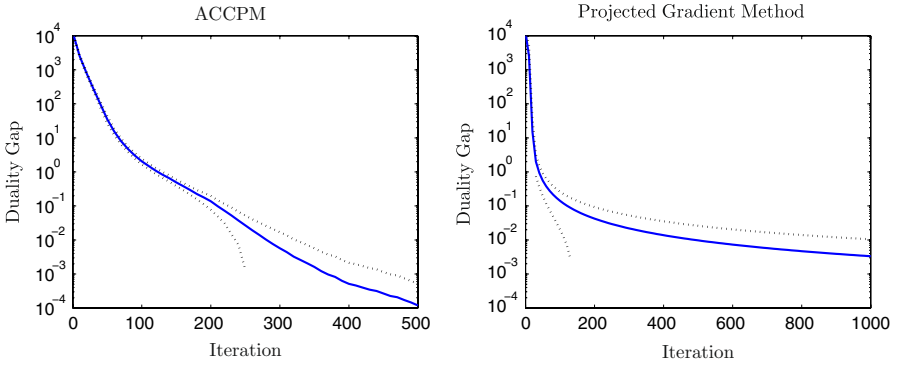

Fig. 1 Convergence plots for ACCPM (*left*) and projected gradient method (*right*) on random subsets of the USPS-SS-3-5 data set (average gap versus iteration number, *dashed lines* at plus and minus one standard deviation). ACCPM converges linearly to a higher precision while the gradient projection method converges faster in the beginning but stalls at a higher precision

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 IndefiniteSVM using the projected gradient method and ACCPM and compare them with the SIQCLP method of Chen and Ye [7]. 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.

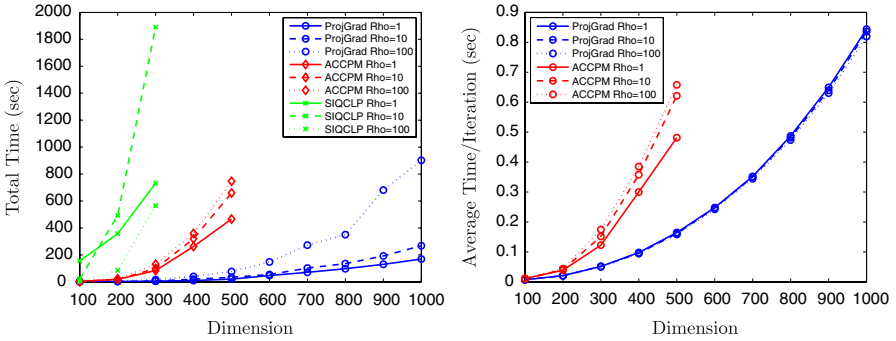

Fig. 2 Total time versus dimension (*left*) and average time per iteration versus dimension (*right*) using projected gradient and ACCPM IndefiniteSVM and SIQCLP (only for total time). The number of iterations for convergence varies from 100 for the smallest dimension to 2,000 for the largest dimension in this example which uses a Simpson kernel on the USPS 3-5 data

Experiments are averaged over 10 random data subsets and we fix $C = 10$ with a tolerance of .1 for the duality gap. For the projected gradient method, increasing ρ increases the number of iterations to converge; notice that the average time per iteration does not vary over ρ. SIQCLP also requires more iterations to converge for higher ρ, however, the average iteration time seems to be less for higher ρ, so no clear pattern is seen when varying ρ. Note that the number of iterations required varies widely (between 100 and 2,000 iterations in this experiment) as a function of ρ, C, the chosen kernel and the stepsize.

Results for ACCPM and SIQCLP are shown only up to dimensions 500 and 300, respectively, because this sufficiently demonstrates that the projected gradient method is more efficient. ACCPM clearly suffers from the complexity of the analytic center problem each iteration. However, improvements can be made in the SIQCLP implementation such as using a regularized version of an efficient MKL solver (e.g. [28]) to solve problem (13) rather than MOSEK. SIQCLP is also useful because it makes a connection between the indefinite SVM formulation and multiple kernel learning. We observed from experiments that the duality gap found from SIQCLP is tighter than the upper bound on the duality gap used for the projected gradient method. This could potentially be used to create a better stopping condition, however, the complexity to derive the tighter duality gap (solving regularized MKL) is much higher than that to compute our current gap (solving a single SVM).

6 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, as well as extend it to other kernel methods such as SVR and one-class support vector machines. 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 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 and to Jianhui Chen and Jieping Ye for their SIQCLP Matlab code. We would also like to acknowledge support from NSF grant DMS-0625352, NSF CDI grant SES-0835550, a NSF CAREER award, a Peek junior faculty fellowship and a Howard B. Wentz Jr. junior faculty award.

References

1. Asuncion, A., Newman, D.: UCI machine learning repository, University of California, Irvine, School of Information and Computer Sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html (2007). Accessed 10 Jan 2008
2. Bach, F.R., Lanckriet, G.R.G., Jordan, M.I.: Multiple kernel learning, conic duality, and the SMO algorithm. Proceedings of the 21st International Conference on Machine Learning. 8 pp (2004)
3. Bennet, K.P., Bredensteiner, E.J.: Duality and geometry in svm classifiers. Proceedings of the 17th International conference on Machine Learning pp. 57–64 (2000)
4. Bertsekas, D.: Nonlinear Programming, 2nd edn. Athena Scientific (1999)
5. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, USA (2004)
6. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. Accessed 1 April 2008
7. Chen, J., Ye, J.: Training SVM with indefinite kernels. Proceedings of the 25th International Conference on Machine Learning. 8 pp (2008)
8. Chen, Y., Gupta, M.R., Recht, B.: Learning kernels from indefinite similarities. Proceedings of the 26th International Conference on Machine Learning. 8 pp (2009)
9. Cuturi, M.: Permanents, transport polytopes and positive definite kernels on histograms. Proceedings of the Twentieth International Joint Conference on Artificial Intelligence pp. 732–737 (2007)
10. Demmel, J.W.: Applied Numerical Linear Algebra. SIAM (1997)
11. Goffin, J.-L., Vial, J.-P.: Convex nondifferentiable optimization: a survey focused on the analytic center cutting plane method. Optim. Methods Softw. **17**(5), 805–867 (2002)
12. Haasdonk, B.: Feature space interpretation of SVMs with indefinite kernels. IEEE Trans. Pattern Anal. Mach. Intel. **27**(4), 482–492 (2005)
13. Haasdonk, B., Keysers, D.: Tangent distance kernels for support vector machines Proceedings of the 16th International Conference on Pattern Recognition **2**, 864–868 (2002)
14. Hettich, R., Kortanek, K.O.: Semi-infinite programming: theory, methods, and applications. SIAM Rev. **35**(3), 380–429 (1993)
15. Higham, N.: Computing the nearest correlation matrix—a problem from finance. IMA J. Numer. Anal. **22**, 329–343 (2002)
16. Hiriart-Urruty, J.-B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms. Springer, Berlin (1993)
17. Hull, J.J.: A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell. **16**(5), 550–554 (1994)
18. Kulis, B., Sustik, M., Dhillon, I.: Learning low-rank kernel matrices. Proceedings of the 23rd International Conference on Machine Learning. pp. 505–512 (2006)
19. Lanckriet, G.R.G., Cristianini, N., Jordan, M.I., Noble, W.S.: Kernel-based integration of genomic data using semidefinite programming. In: Kernel Methods in Computational Biology. MIT Press, Cambridge (2003)
20. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., Ghaoui, L.E., Jordan, M.I.: Learning the kernel matrix with semidefinite programming. J. Mach. Learn. Res. **5**, 27–72 (2004)
21. Lewis, A.: Nonsmooth analysis of eigenvalues. Math. Program. **84**, 1–24 (1999)
22. Lin, H.-T., Lin, C.-J.: A Study on Sigmoid Kernel for SVM and the Training of Non-PSD Kernels by SMO-type Methods. National Taiwan University, Department of Computer Science and Information Engineering, Taipei, Taiwan (2003)
23. MOSEK ApS: The MOSEK optimization tools manual. Version 5.0, revision 105. Software available at http://www.mosek.com. (2008). Accessed 1 Dec 2008

24. Nesterov, Y.: Introductory Lectures on Convex Optimization. Springer, Berlin (2003)
25. Ong, C.S., Mary, X., Canu, S. Smola, A.J.: Learning with non-positive kernels. Proceedings of the 21st International Conference on Machine Learning. 8 pp (2004)
26. Ong, C.S., Smola, A.J., Williamson, R.C.: Learning the kernel with hyperkernels'. J. Mach. Learn. Res. **6**, 1043–1071 (2005)
27. Overton, M.: Large-scale optimization of eigenvalues. SIAM J. Optim. **2**(1), 88–120 (1992)
28. Rakotomamonjy, A., Bach, F., Canu, S., Grandvalet, Y.: SimpleMKL. J. Mach. Learn. Res. **9**, 2491–2521 (2008)
29. Saigo, H., Vert, J.P., Ueda, N., Akutsu, T.: Protein homology detection using string alignment kernels. Bioinformatics **20**(11), 1682–1689 (2004)
30. Schölkopf, B., Smola, A.: Learning with Kernels. The MIT Press, Cambridge (2002)
31. Simard, P.Y., Cun, Y.A.L., Denker, J.S., Victorri, B.: Transformation invariance in pattern recognition-tangent distance and tangent propogation. Lect. Notes Comp. Sci. **1524**, 239–274 (1998)
32. Sonnenberg, S., Rätsch, G., Schäfer, C., Schölkopf, B.: Large scale multiple kernel learning. J. Mach. Learn. Res. **7**, 1531–1565 (2006)
33. Woźnica, A., Kalousis, A. Hilario, M.: Distances and (indefinite) kernels for set of objects. Proceedings of the 6th International Conference on Data Mining pp. 1151–1156 (2006)
34. Wu, G., Chang, E.Y., Zhang, Z.: An analysis of transformation on non-positive semidefinite similarity matrix for kernel machines. Proceedings of the 22nd International Conference on Machine Learning. 8 pp (2005)
35. Zamolotskikh, A., Cunningham, P.: An Assessment of Alternative Strategies for Constructing EMD-Based Kernel Functions for Use in an SVM for Image Classification. Proceedings of the International Workshop on CBMI 2007, 11–17 (2007)