Support vector machine training using matrix completion techniques

Martin S. Andersen*† Lieven Vandenberghe*

Abstract

We combine interior-point methods and results from matrix completion theory in an approximate method for the large dense quadratic programming problems that arise in support vector machine training. The basic idea is to replace the dense kernel matrix with the maximum determinant positive definite completion of a subset of the entries of the kernel matrix. The resulting approximate kernel matrix has a sparse inverse and this property can be exploited to dramatically improve the efficiency of interior-point methods. If the sparsity pattern is chordal, the sparse inverse and its Cholesky factors are easily computed by efficient recursive algorithms. Numerical experiments with training sets of size up to 60000 show that the approximate interior-point approach is competitive with the LIBSVM package in terms of error rate and speed. The completion technique can also be applied to other dense convex optimization problems arising in machine learning, for example, in Gaussian process classification.

Keywords: support vector machines, classification, kernel methods, convex optimization, interior-point methods

1 Introduction

Kernel methods typically require the solution of large, dense optimization problems. The best known example is the quadratic program (QP) that arises in the dual formulation of the *support vector machine* (SVM) training problem:

$$\begin{array}{ll} \text{maximize} & -(1/2)z^T Q z + d^T z \\ \text{subject to} & 0 \preceq \mathbf{diag}(d) z \preceq \gamma \mathbf{1} \\ & \mathbf{1}^T z = 0. \end{array} \tag{1}$$

The variable in this problem is $z \in \mathbf{R}^m$, where m is the number of training examples. The vector $d \in \{-1, 1\}^m$ contains the labels of the training examples, $\mathbf{1}$ is the m-vector with elements equal to one, and $\gamma > 0$ is a positive parameter. The inequalities denote componentwise inequality. The matrix Q in the objective is called the *kernel matrix* and is defined as

$$Q_{ij} = h(x_i, x_j), \quad i, j = 1, \ldots, m,$$

*Electrical Engineering Department, University of California, Los Angeles. Research supported in part by NSF grant ECCS-0824003.

†Corresponding author. Address: Electrical Engineering Department, 56-125B Engineering IV Building, Box 951594, Los Angeles, CA 90095-1594. Email: msa@ee.ucla.edu.

where h is a positive semidefinite kernel function and $x_1, \ldots, x_m$ are the training examples. Important kernel functions on $\mathbf{R}^n \times \mathbf{R}^n$ are the *linear* kernel function $h(u,v) = u^T v$, the *radial basis function* kernel $h(u,v) = \exp(-\|u-v\|_2^2/(2\sigma))$, and the *polynomial* kernel $h(u,v) = (u^T v)^\delta$.

The QP (1) is a convex optimization problem and can be solved by standard methods for convex optimization, such as interior-point methods [NW06, ch.16]. This requires knowledge of the entire kernel matrix and, at each iteration, a factorization of the sum of Q and a positive diagonal matrix. Forming the (generally dense) kernel matrix Q requires $O(m^2 n)$ time for the three standard kernel functions mentioned above and $O(m^2)$ storage, and factorizing it costs $O(m^3)$ operations. When the number of training vectors is large (say, greater than 10000), the QP therefore becomes prohibitively expensive to solve by general-purpose QP interior-point solvers.

Efforts to improve the efficiency of large-scale SVM algorithms have been most successful for the linear kernel $h(u,v) = u^T v$. If the linear kernel is used, $Q = XX^T$ if X is the $m \times n$-matrix with rows x_i^T. Therefore $\mathbf{rank}\, Q \leq n$ if $m \geq n$. This property can be exploited to reduce the complexity of an interior-point method from $O(m^3)$ to $O(nm^2)$ per iteration [FS02, FM03]. Research on fast first-order algorithms, such as projected gradient or cutting-plane algorithms, has also largely focused on the linear kernel [HCL$^+$08, FCH$^+$08, Joa06, JY09], by taking advantage of the fact that the gradient of the objective $-Qz + d$ can be evaluated in $O(mn)$ operations if Q has rank n.

If a nonlinear kernel is used, the matrix Q is generally dense and full rank, and this complicates the implementation of quadratic programming algorithms. When m is large, it therefore makes sense to approximate Q by a simpler matrix which is easy to compute, requires less storage, and makes the QP easier to solve. Examples are low-rank [FS02] or diagonal-plus-low-rank [FM03] approximations. If the rank is much less than m the cost per iteration of an interior-point method can be reduced to $O(mn^2)$ by using the Sherman-Morrison-Woodbury formula or the product form Cholesky factorization algorithm. Low-rank or diagonal-plus-low-rank approximations also simplify the implementation of first-order methods because matrix-vector products are simplified. Finding a suitable low-rank approximation, however, is a nontrivial task when m is large, since optimal approximations based on an eigenvalue decomposition are very expensive. Fine and Scheinberg [FS02] discuss an incomplete Cholesky factorization algorithm for computing a low-rank approximation of a kernel matrix. The method requires calculating (but not storing) all the entries in the kernel matrix. In related work, Smola and Schölkopf [SS00] discuss greedy algorithms for low-complexity approximations of the kernel matrix.

In this paper we explore an idea similar to methods based on low-rank kernel matrix approximations. However, instead of making a low-rank approximation, we approximate the kernel matrix by a matrix with a sparse inverse. The approximation is obtained by computing the maximum determinant positive definite completion of a partial kernel matrix. The approximated kernel matrix is dense and full rank, but has the property that its inverse is sparse, making the QP very easy to solve. An added advantage is that only a subset of the entries of the kernel matrix are needed to compute the approximation. We will see that the Cholesky factors of the inverse of the completion can be computed very efficiently if the positions of the specified entries in the partial kernel matrix, which are also the nonzero positions in its inverse, form a *chordal* pattern, for example, a band pattern. The purpose of the paper is to evaluate the performance of an SVM training algorithm based on this sparse inverse kernel approximation. We focus on interior-point methods, but the approximation should be useful in first-order methods as well. It can also be combined with chunking, decomposition, and active set methods that are based on solving a sequence of lower-dimensional subproblems [OFG97, Pla99, Joa99, CL01].

Outline The paper is organized as follows. In section 2 we define the maximum determinant positive definite matrix completion problem and give its main properties. We also discuss in detail how to compute the Cholesky factors of the sparse inverse of the completion of a band matrix. Section 3 describes how matrix completion techniques can be exploited in an interior-point method for solving an approximate SVM training problem. The approximate training problem can be interpreted in two ways, and these lead to two different classifiers. In section 4 we consider some numerical experiments, including a comparison of the two classifiers. We discuss some other applications of the completion techniques in machine learning in section 5, and finally, we conclude the paper in section 6.

Notation We denote with $\mathbf{S}^m$ the symmetric matrices of order m, $\mathbf{S}^m_+$ is the cone of positive semidefinite matrices in $\mathbf{S}^m$, and $\mathbf{S}^m_{++}$ is the set of positive definite matrices in $\mathbf{S}^m$. The inner product of two symmetric matrices A and B is defined as $\mathbf{tr}(AB) = \sum_{ij} A_{ij}B_{ij}$. The inequalities $A \succeq 0$ and $A \succ 0$ denote matrix inequality, *i.e.*, A is positive semidefinite, respectively, positive definite.

2 Maximum entropy kernel completion

In this section we review the basic properties of the maximum determinant positive definite matrix completion problem [Dem72, GJSW84, Lau01]. We also make connections with positive definite kernels and Gaussian processes [SS02, RW06].

2.1 Maximum determinant positive definite matrix completion

The positive definite matrix completion problem is a fundamental problem in matrix algebra [GJSW84, Lau01]. We are given certain entries of a symmetric matrix and are asked to choose the remaining entries so that the matrix is positive definite with maximum determinant. If we denote the given entries by Q_{ij}, $(i,j) \in V$, then the problem can be expressed as an optimization problem

$$\begin{array}{ll} \text{maximize} & \log\det X \\ \text{subject to} & X_{ij} = Q_{ij} \quad \forall (i,j) \in V \end{array} \tag{2}$$

with variable $X \in \mathbf{S}^m$. (To simplify the notation we take as the domain of $\log\det X$ the set of positive definite matrices $\mathbf{S}^m_{++}$, *i.e.*, the constraint $X \succ 0$ is made implicit in the definition of the objective function.) Since $\log\det X$ is concave on $\mathbf{S}^m_{++}$, the problem (2) is a convex optimization problem.

The matrix completion problem has an interesting statistical interpretation. Suppose f is a random vector with known second moments $Q_{ij} = \mathbf{E}\, f_i f_j$ for $(i,j) \in V$. Then the distribution with maximum entropy that matches these moments is the zero-mean normal distribution $\mathcal{N}(0, X)$ where X is the optimal solution of (2) [CT91, p.270]. For this reason the solution is also known as the maximum entropy completion.

In the context of kernel methods, positive definite kernel functions h are often interpreted as covariance functions of Gaussian processes f, *i.e.*, $h(x,y) = \mathbf{E}(f(x)f(y))$ [RW06], [SS02, Ch.16]. The solution of the completion problem with $Q_{ij} = h(x_i, x_j)$ for $(i,j) \in V$ then provides the maximum entropy extension of the samples $\mathbf{E}(f(x_i)f(x_j)) = Q_{ij}$, $(i,j) \in V$ of the covariance function.

2.2 Duality and optimality conditions

We assume that the diagonal entries are included in V, so problem (2) cannot be unbounded. However, depending on V and Q_{ij} it may be infeasible. If it is feasible, the optimal solution of (2) satisfies the optimality conditions

$$X \succ 0, \qquad X_{ij} = Q_{ij} \quad \forall (i,j) \in V, \qquad (X^{-1})_{ij} = 0 \quad \forall (i,j) \notin V. \tag{3}$$

Therefore the inverse of the optimal completion has zeros in the positions of the unspecified entries of X.

Additional insight in the problem can be gained from the dual of (2), which is

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(QZ) - \log\det Z - m \\ \text{subject to} & Z_{ij} = 0 \quad \forall (i,j) \notin V \end{array} \tag{4}$$

with variable $Z \in \mathbf{S}^m$. (Note that since $Z_{ij} = 0$ for $(i,j) \notin V$, the values of Q_{ij} outside V are irrelevant in the objective.) The optimal solutions of (2) and (4) are related as $Z = X^{-1}$. Therefore the dual optimal Z is the sparse positive definite matrix whose inverse has values $(Z^{-1})_{ij} = Q_{ij}$ in positions $(i,j) \in V$.

The dual problem also has an interesting statistical meaning. Recall that the relative entropy (or Kullback-Leibler divergence) between two normal distributions $\mathcal{N}(0,Q)$ and $\mathcal{N}(0,X)$ is defined as

$$\frac{1}{2}\left(\mathbf{tr}(QX^{-1}) - \log\det(QX^{-1}) - m\right) \tag{5}$$

[Kul97, p.189]. Except for the factor $1/2$ this is identical to the dual objective function of (4) evaluated at $Z = X^{-1}$. The maximum determinant completion X can therefore be interpreted as the covariance of the normal distribution $\mathcal{N}(0,X)$ that minimizes the relative entropy with $\mathcal{N}(0,Q)$, subject to a sparsity constraint on X^{-1}. The solution is independent of the values of Q_{ij} outside V.

2.3 Solution for band patterns

It is clear from (2) and (4) that only the entries Q_{ij} for $(i,j) \in V$ are needed to compute the optimal completion X and its inverse Z. For a general sparsity pattern V the problems must be solved iteratively by a numerical optimization algorithm, for example, Newton's method. However when the sparsity pattern V is *chordal*, the problem (2) can be solved much more easily using a simple recursive algorithm [DVR08, GJSW84]. In this section we give the details for a special case of a chordal pattern, a band pattern with bandwidth $2w+1$, *i.e.*, we take

$$V = \{(i,j) \mid |i-j| \leq w\}.$$

We focus on band patterns because they will be used in the numerical experiments in section 4. Matrix completion with general chordal patterns is discussed in section 2.4.

A necessary condition for existence of a positive definite completion is that the principal submatrices of order $w+1$,

$$\begin{bmatrix} Q_{kk} & Q_{k,k+1} & \cdots & Q_{k,k+w+1} \\ Q_{k+1,k} & Q_{k+1,k+1} & \cdots & Q_{k+1,k+w+1} \\ \vdots & \vdots & \ddots & \vdots \\ Q_{k+w+1,k} & Q_{k+w+1,k+1} & \cdots & Q_{k+w+1,k+w+1} \end{bmatrix}, \quad k = 1, \ldots, m-w-1, \tag{6}$$

are positive definite. We will assume that this is the case and prove constructively that the condition is also sufficient. To derive the optimal completion algorithm we consider the optimality conditions (3). The conditions show that X^{-1} is banded with bandwidth $2w+1$. Therefore the optimal X can be factored as $X^{-1} = RR^T$ with R upper triangular and banded with bandwidth $w+1$. To determine R we examine the equation

$$XR = R^{-T}, \tag{7}$$

and use the fact that the entries X_{ij} for $(i,j) \in V$ are given and equal to Q_{ij}, and that R^{-T} is lower triangular with diagonal entries $1/R_{kk}$.

The leading block of R of order $w+1$ follows from

$$\begin{bmatrix} X_1 & \times \\ \times & \times \end{bmatrix} \begin{bmatrix} R_1 & \times \\ 0 & \times \end{bmatrix} = \begin{bmatrix} R_1^{-T} & 0 \\ \times & \times \end{bmatrix}.$$

Here X_1 and R_1 denote the leading principal submatrices of order $w+1$ and the '$\times$' entries are blocks of the matrix that are irrelevant in the present discussion. Since all positions in X_1 are in V, the matrix X_1 is entirely specified and positive definite, so R_1 can be computed from a dense Cholesky factorization $X_1 = R_1^{-1}R_1^{-T}$.

The rest of R is determined column by column. To compute a column $k > w+1$ of R, consider column k of the two sides of the equation (7):

$$\begin{bmatrix} \times & \times & \times & \times \\ \times & X_k & \hat{X}_k & \times \\ \times & \hat{X}_k^T & X_{kk} & \times \\ \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} 0 \\ \hat{R}_k \\ R_{kk} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1/R_{kk} \\ \times \end{bmatrix}.$$

Here X_k has size $w \times w$, and $\hat{X}_k$ and $\hat{R}_k$ are vectors of length w. This gives two equations from which we can determine $\hat{R}_k$ and R_{kk}:

$$X_k\hat{R}_k + \hat{X}_k R_{kk} = 0, \qquad \hat{X}_k^T R_k + X_{kk}R_{kk} = \frac{1}{R_{kk}}.$$

The equations are solvable because the submatrix

$$\begin{bmatrix} X_k & \hat{X}_k \\ \hat{X}_k^T & X_{kk} \end{bmatrix}$$

is completely specified (and equal to the matrix (6)) and positive definite. The solution is

$$R_{kk} = \left(X_{kk} - \hat{X}_k^T X_k^{-1} \hat{X}_k\right)^{-1/2}, \qquad \hat{R}_k = -R_{kk}X_k^{-1}\hat{X}_k.$$

We can compute R_{kk} and $\hat{R}_k$ using a Cholesky factorization of $X_k = L_k L_k^T$ in three steps:

$$y := L_k^{-1}\hat{X}_k, \qquad R_{kk} := (X_{kk} - y^T y)^{-1/2}, \qquad \hat{R}_k := -R_{kk}L_k^{-T}y.$$

If the columns of R are computed in the natural order, we can reuse part of the Cholesky factorization of X_k from one step to the next, by noting that X_{k+1} is obtained from X_k by deleting the

first row and column and by adding a new last row and column. Its Cholesky factorization can therefore be updated from the factorization of X_k in $O(w^2)$ operations. This reduces the cost of computing the maximum determinant completion to $O(w^2m)$ if $m \gg w$.

In summary, the maximum determinant positive definite completion of the values Q_{ij}, $|i-j| \leq w$, in a diagonal band of width $2w+1$ exists if and only if the submatrices (6) are positive definite. The inverse of the optimal completion is banded and its Cholesky factors can be computed from the specified values Q_{ij} in $O(w^2m)$ steps.

2.4 Solution for general chordal patterns

An undirected graph is called *chordal* if every cycle of length greater than three has a chord, *i.e.*, an edge connecting non-adjacent nodes. A symmetric sparsity pattern V is chordal if the graph with edges $(i,j) \in V$ is chordal. Chordal sparse matrices have many important properties, including three that extend the properties of band matrices used above. First, every positive definite matrix Z with a chordal sparsity pattern has a zero-fill Cholesky factorization, *i.e.*, there exists a permutation P such that $P^TZP = RR^T$ where R is triangular with the same sparsity pattern as P^TZP (*i.e.*, $R + R^T$ has the sparsity pattern of P^TZP.) Second, the maximum determinant positive definite completion of a subset of entries Q_{ij}, $(i,j) \in V$, with a chordal sparsity pattern V exists if and only if all the completely specified principal submatrices are positive definite. These principal submatrices correspond to cliques in the associated graph. Finally, there exist explicit formulas and finite recursive algorithms for the maximum determinant positive definite completion of a matrix with a chordal sparsity pattern. For example, as for the band patterns, one can compute the Cholesky factors of the inverse of the optimal completion directly from Q_{ij}, $(i,j) \in V$. We refer the reader to [DVR08] for more details.

3 SVM training by matrix completion

The matrix completion theory in the previous section can be applied to compute optimal (*i.e.*, having minimum relative entropy) approximation of a kernel matrix by a positive definite matrix with a sparse inverse. In this section we examine the computational advantages of a sparse inverse kernel approximations when solving the SVM training problem (1).

Suppose $\tilde{Q}$ is the maximum determinant extension of $Q_{ij} = h(x_i, x_j)$, $(i,j) \in V$, where V is a chordal pattern. (In our experiments we will select V randomly, by choosing a band pattern after applying a random permutation of the training vectors.) We will substitute $\tilde{Q}$ for Q in the training problem (1) and consider

$$\begin{array}{ll} \text{maximize} & -(1/2)z^T\tilde{Q}z + d^Tz \\ \text{subject to} & 0 \preceq \mathbf{diag}(d)z \preceq \gamma\mathbf{1} \\ & \mathbf{1}^Tz = 0. \end{array} \tag{8}$$

In this section we first describe in detail how one can take advantage of the sparsity of $\tilde{Q}^{-1}$ in an interior-point method for the QP (8) (see sections 3.1 and 3.2). We then discuss two possible ways of using the solution of (8). First, we can interpret the optimal z as the *exact* dual solution of an SVM training problem for an unknown kernel function $\tilde{h}$ with values $\tilde{h}(x_i, x_j) = \tilde{Q}_{ij}$ on the training set. We will discuss two choices for $\tilde{h}$ in section 3.3 and compare the performance of the resulting classifiers in section 4. Second, we can view the optimal z as an *approximation* of the solution of

the QP (1) associated with the original kernel h and use the values of z to select a subset of training vectors for which we then solve a smaller dense QP exactly. This idea is investigated in section 4.

3.1 Quadratic programming duality

We first review the quadratic programming duality theory for the SVM training problem (1). The dual of the problem can be expressed as

$$\begin{array}{ll} \text{minimize} & (1/2)y^T Q^\dagger y + \gamma \mathbf{1}^T v \\ \text{subject to} & y \in \text{range}(Q) \\ & \mathbf{diag}(d)(y + b\mathbf{1}) + v \succeq \mathbf{1} \\ & v \succeq 0, \end{array} \tag{9}$$

with variables $y \in \mathbf{R}^m$, $b \in \mathbf{R}$, and $v \in \mathbf{R}^m$. The notation $Q^\dagger$ denotes the pseudo-inverse of Q (for now we do not make any assumptions on the invertibility of Q). The vector b is the multiplier associated with the equality constraint in (1), the vector v is the multiplier for the inequality $\mathbf{diag}(d)z \preceq \gamma\mathbf{1}$. The multiplier for the constraint $\mathbf{diag}(d)z \succeq 0$ has been eliminated from (9) and is equal to $\mathbf{diag}(d)(y + b\mathbf{1}) + v - \mathbf{1}$. If we make a substitution $y = Qu$ we can simplify the dual problem as

$$\begin{array}{ll} \text{minimize} & (1/2)u^T Q u + \gamma \mathbf{1}^T v \\ \text{subject to} & \mathbf{diag}(d)(Qu + b\mathbf{1}) + v \succeq \mathbf{1} \\ & v \succeq 0 \end{array} \tag{10}$$

with variables u, b. It can be shown that if z is optimal in (1), then $y = Qz$ is optimal in (9) and hence $u = z$ is optimal in (10).

We can eliminate v and write the dual QP (10) as an unconstrained nondifferentiable problem

$$\text{minimize} \quad \frac{1}{2}u^T Q u + \gamma \sum_{i=1}^{m} \max\{0, 1 - d_i(\sum_{j=1}^{m} Q_{ij}u_j + b)\}. \tag{11}$$

Since $Q_{ij} = h(x_i, x_j)$ the second term is

$$\sum_{i=1}^{m} \max\{0, 1 - d_i(\sum_{j=1}^{m} h(x_i, x_j)u_j + b)\},$$

i.e., the familiar hinge-loss penalty for the classifier

$$g(x) = \mathbf{sign}(\sum_{j=1}^{m} h(x, x_j)u_j + b) \tag{12}$$

evaluated on the training set defined by x_i, d_i, $i = 1, \ldots, m$. If the linear kernel is used ($h(x_i, x_j) = x_i^T x_j$ and $Q = X^T X$, where X is the matrix with rows x_i^T), then (11) can be written as

$$\text{minimize} \quad \frac{1}{2}w^T w + \gamma \sum_{i=1}^{m} \max\left\{0, 1 - d_i(w^T x_i + b)\right\}$$

with $w = X^T u$. The classifier (12) reduces to $g(x) = \mathbf{sign}(w^T x + b)$.

3.2 Interior-point method

We now discuss the implementation of interior-point algorithms for solving the QP (8) and its dual, when $\tilde{Q}$ is large and dense, but also invertible with a sparse inverse. For invertible $\tilde{Q}$ the dual reduces to

$$\begin{array}{ll} \text{minimize} & (1/2)y^T\tilde{Q}^{-1}y + \gamma\mathbf{1}^Tv \\ \text{subject to} & \mathbf{diag}(d)(y + b\mathbf{1}) + v \succeq \mathbf{1} \\ & v \succeq 0. \end{array} \tag{13}$$

The main step in an iteration of an interior-point method applied to (8) and (13) is the solution of a linear equation of the form

$$\begin{bmatrix} \tilde{Q}^{-1} & 0 & 0 & -\mathbf{diag}(d) & 0 \\ 0 & 0 & 0 & -d^T & 0 \\ 0 & 0 & 0 & -I & -I \\ -\mathbf{diag}(d) & -d & -I & -D_1 & 0 \\ 0 & 0 & -I & 0 & -D_2 \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta b \\ \Delta v \\ \Delta z \\ \Delta w \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix}, \tag{14}$$

where D_1, D_2 are positive diagonal matrices with values that change at each iteration. We will refer to (14) as the Newton system because it can be interpreted as the linearization of the nonlinear equations that define the primal-dual central path.

Interior-point methods are known to reach a high accuracy after a small number of iterations (in the range 10–30), almost independent of problem size, so to get an idea of the overall complexity it is fair to focus on the cost of solving one Newton system (14). (Some common interior-point algorithms solve multiple Newton systems (usually two or three) per iteration, with different righthand sides but with the same values for D_1 and D_2. The cost of solving the multiple systems is therefore essentially the same as the cost of solving one Newton system [Wri97, NW06].) We now describe an efficient algorithm for solving (14) by taking advantage of the sparsity of $\tilde{Q}^{-1}$.

We start by eliminating Δv, Δz, Δw. From the bottom three block rows in (14) and the identity

$$\begin{bmatrix} 0 & I & I \\ I & D_1 & 0 \\ I & 0 & D_2 \end{bmatrix}^{-1} = \begin{bmatrix} -D_1D_2 & D_2 & D_1 \\ D_2 & I & -I \\ D_1 & -I & I \end{bmatrix} \begin{bmatrix} (D_1+D_2)^{-1} & 0 & 0 \\ 0 & (D_1+D_2)^{-1} & 0 \\ 0 & 0 & (D_1+D_2)^{-1} \end{bmatrix}$$

we can express Δv, Δz, Δw, as a function of Δy, Δb:

$$\begin{aligned} \begin{bmatrix} \Delta v \\ \Delta z \\ \Delta w \end{bmatrix} &= -\begin{bmatrix} 0 & I & I \\ I & D_1 & 0 \\ I & 0 & D_2 \end{bmatrix}^{-1} \left(\begin{bmatrix} 0 & 0 \\ \mathbf{diag}(d) & d \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta b \end{bmatrix} + \begin{bmatrix} r_3 \\ r_4 \\ r_5 \end{bmatrix} \right) \\ &= -\begin{bmatrix} D_2 \\ I \\ -I \end{bmatrix} \mathbf{diag}(d)(D_1+D_2)^{-1} \begin{bmatrix} I & \mathbf{1} \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta b \end{bmatrix} \\ &\quad - \begin{bmatrix} -D_1D_2 & D_2 & D_1 \\ D_2 & I & -I \\ D_1 & -I & I \end{bmatrix} \begin{bmatrix} (D_1+D_2)^{-1}r_3 \\ (D_1+D_2)^{-1}r_4 \\ (D_1+D_2)^{-1}r_5 \end{bmatrix}. \end{aligned} \tag{15}$$

Substituting this in (14) gives a smaller equation in Δy, Δb:

$$\begin{bmatrix} \tilde{Q}^{-1} + \widehat{D} & \widehat{D}\mathbf{1} \\ \mathbf{1}^T\widehat{D} & \mathbf{1}^T\widehat{D}\mathbf{1} \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta b \end{bmatrix} = \begin{bmatrix} \hat{r}_1 \\ \hat{r}_2 \end{bmatrix} \tag{16}$$

where $\widehat{D} = (D_1 + D_2)^{-1}$ is positive diagonal and

$$\begin{bmatrix} \hat{r}_1 \\ \hat{r}_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} - \begin{bmatrix} \mathbf{diag}(d) \\ d^T \end{bmatrix} \widehat{D}(D_2 r_3 + r_4 - r_5).$$

The system (16) is a positive definite equation because $\widehat{D} - (\mathbf{1}^T\widehat{D}\mathbf{1})^{-1}\widehat{D}\mathbf{1}\mathbf{1}^T\widehat{D} \succeq 0$ for positive diagonal $\widehat{D}$. To solve (16) we can solve two equations

$$(\tilde{Q}^{-1} + \widehat{D})y^{(1)} = \hat{r}_1, \qquad (\tilde{Q}^{-1} + \widehat{D})y^{(2)} = -\widehat{D}\mathbf{1}, \tag{17}$$

and make a linear combination $\Delta y = y^{(1)} + \Delta b\, y^{(2)}$, with Δb chosen to satisfy the last equation in (16), *i.e.*,

$$\Delta b = \frac{\hat{r}_2 - \mathbf{1}^T\widehat{D}y^{(1)}}{\mathbf{1}^T\widehat{D}(y^{(2)} + \mathbf{1})}. \tag{18}$$

In summary, we can solve (14) by first solving the two equations (17), then computing Δb and $\Delta y = y^{(1)} + \Delta b\, y^{(2)}$, and then Δv, Δz, Δw from (15). If $\tilde{Q}^{-1}$ is sparse then the matrix $\tilde{Q}^{-1} + \widehat{D}$ is sparse, with the same sparsity pattern as $\tilde{Q}^{-1}$. In particular, as we saw in section 2, if the sparsity pattern of $\tilde{Q}^{-1}$ is chordal, then we can factor $\tilde{Q}^{-1} + \widehat{D}$ with a zero-fill Cholesky factorization. For a band pattern with constant bandwidth, for example, the cost of solving (14), and hence the cost of the interior-point algorithm itself, is *linear* in m.

3.3 Completion kernel classifier

Every positive definite matrix can be interpreted as a kernel matrix for some positive definite kernel function [SS02, p.44]. Replacing the kernel matrix Q with a positive definite completion $\tilde{Q}$ of a subset of the entries of Q can therefore be thought of as applying a modified positive definite kernel function $\tilde{h}$. The value of the modified kernel function $\tilde{h}$ is known and equal to $\tilde{Q}_{ij}$ at pairs of training points (x_i, x_j), $i, j = 1, \ldots, m$, but is not uniquely defined for other points. To evaluate the decision function

$$g(x) = \mathbf{sign}(\sum_{i=1}^{m} \tilde{h}(x, x_i)z_i + b), \tag{19}$$

at at test point x we therefore need to assign a value $\tilde{h}(x, x_i)$.

A first choice is to simply use $\tilde{h}(x, x_i) = h(x, x_i)$. While our results below indicate that this works well in some cases (*e.g.*, if the bandwidth is chosen sufficiently large), there is no guarantee that $\tilde{h}$ is a positive definite kernel, as it can happen that the bordered kernel matrix

$$\tilde{Q}' = \begin{bmatrix} \tilde{Q} & \tilde{q} \\ \tilde{q}^T & h(x, x) \end{bmatrix}, \tag{20}$$

is not positive definite if we take $\tilde{q}_i = h(x, x_i)$. We will refer to the classifier (19) with $\tilde{h}(x, x_i) = h(x, x_i)$ as the *standard kernel classifier*.

A second choice is to take the chordal pattern V used to define the completion $\tilde{Q}$ and extend it to a chordal pattern V' for the bordered kernel matrix $\tilde{Q}'$ in (20). We define $\tilde{Q}'_{ij}$ for $(i, j) \in V'$ as

$$\tilde{Q}_{ij} = h(x_i, x_j) \quad \forall (i, j) \in V, \qquad \tilde{q}_i = h(x_i, x) \quad \forall (i, m+1) \in V',$$

and use the maximum determinant completion of these entries to define $\tilde{q}_i = \tilde{h}(x_i, x)$ at the other values of i. More specifically, suppose V is a band pattern of bandwidth $2w+1$ and let V' be a band pattern of the same bandwidth. The $w+1$ nonzero entries ρ and r_i, $i > m-w$, in the last column of the Cholesky factorization of

$$\begin{bmatrix} \tilde{Q} & \tilde{q} \\ \tilde{q}^T & h(x,x) \end{bmatrix}^{-1} = \begin{bmatrix} R & r \\ 0 & \rho \end{bmatrix} \begin{bmatrix} R^T & 0 \\ r^T & \rho \end{bmatrix}$$

can be obtained from R using the algorithm described in section 2.3. This requires $w+1$ kernel evaluations and $O(w^2)$ operations. The factorization $\tilde{Q}' = (R')^{-T}(R')^{-1}$ provides a method for evaluating $\tilde{h}(x_i, x)$, *i.e.*,

$$\tilde{h}(x_i, x) = e_i^T \tilde{Q}' e_{m+1} = u_i \tag{21}$$

where u is obtained by solving $R'(R')^T u = e_{m+1}$. The cost of computing u is $O(wm)$. If $w \ll m$ the cost of evaluating the decision function (19) is therefore $O(wm)$. We will refer to this classifier as the *completion kernel classifier*.

Experiments with the standard kernel and completion kernel classifiers are given in the next section.

4 Numerical experiments

To evaluate the performance and accuracy of SVMs obtained with the completion kernels, we have conducted a series of experiments based on the MNIST database of handwritten digits [LC98]. The training set consists of 60000 patterns while the test set consists of 10000 patterns. We scale the data by 1/256; no other preprocessing is used. All experiments were conducted on a desktop computer with an Intel Core 2 Quad Q6600 CPU (2.4 GHz), 4 GB RAM, and running Ubuntu 9.10 (64 bit). The algorithm was implemented in Python as a custom KKT-solver for CVXOPT 1.1.2 [DV08] and using CHOMPACK 1.1 [DV09] for chordal matrix computations[1]. The software package LIBSVM 2.9 [CL01] was used for comparison.

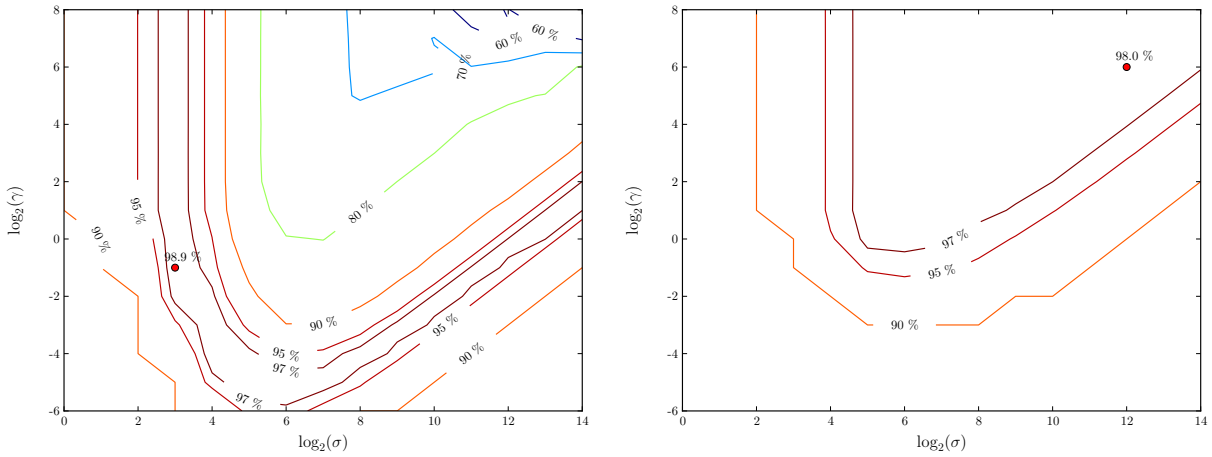
We use the RBF kernel function $h(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma))$. The kernel completions $\tilde{Q}$ are computed for band patterns V with bandwidth $2w+1$, after applying a random permutation of the training examples.

4.1 Cross-validation accuracy

In the first experiment we compare the cross-validation accuracy for the two classifiers defined in section 3.3. We use a training set consisting of 10000 randomly chosen examples from the MNIST database (1000 examples of digit 0 and 9000 examples of digits 1–9), and for each pair of parameters $(\gamma = 2^p, \sigma = 2^q)$, where p and q are integers, we compute the 10-fold cross-validation accuracy. The half-bandwidth is $w = 100$. The results are shown in Figure 1. We see that the two classifiers have similar cross-validation accuracies when the best combination of parameters is chosen. The completion kernel classifier appears to be less sensitive to parameter changes.

The optimal values of the parameters obviously depend on the bandwidth used in the approximation problem as well as the number of training vectors. In the next experiment we fix the parameters γ and σ and examine the test error rate as a function of bandwidth w.

[1]The code used in the experiments is available at `www.ee.ucla.edu/~vandenbe/software`.



(a) Standard kernel classifier.

(b) Completion kernel classifier.

Figure 1: 10-fold cross-validation accuracy using the RBF kernel. The dots mark the best parameter pairs. The completion kernel classifier performs well on a large set of parameters, and it appears to be less sensitive to the choice of parameters than the standard kernel classifier.

4.2 Bandwidth versus test error rate

We consider a single binary classification problem (digit 0 versus digits 1–9) and again we use as training set a subset of 10000 training vectors from the MNIST training set (1000 randomly chosen examples of digit 0 and 9000 randomly chosen examples digits 1–9). In addition to the standard kernel classifier and the completion kernel classifier, we also compute a third classifier by training an SVM with the exact RBF kernel function $h(x_i, x_j)$ on the set of support vectors from the approximation problem. We use $\gamma = 4$ and $\sigma = 32$. Since the elements in the completed kernel matrix depend on the permutation of the training set, we repeated the experiment 10 times with different pseudo-randomly generated permutations. The average of these results is shown in Table 1. As the bandwidth increases and the approximation gets better, the CPU time increases rapidly (as w^2), as expected. In this example the completion classifier performs better than the approximation classifier except for the smallest bandwidth ($w = 10$). Notice also that the third classifier (obtained by solving a subproblem with the support vectors from the approximation problem) consistently performs well, and the problem size (and hence also the training time) decreases quite fast with increasing bandwidths. The total training time therefore involves a trade-off between the complexity of the approximation problem and the dense subproblem. In this example the fastest total training time (41 seconds) is obtained for $w = 150$.

In column 2 we notice that the solution time for the approximation problem with $w = 1000$ is roughly the same as the time for the full dense QP with $m = 10000$. We believe that this is due to overhead in the current implementation of CHOMPACK (in which it is assumed that $w \ll m$). In a more sophisticated implementation the cost for the banded case should approach the cost for the dense case only when $w \approx m$.

Extrapolating the values in columns 4 and 5, we also note that the error rate in the completion classifier and the support vectors stop decreasing substantially after a certain value of $w < m$.

	Approx. problem ($m = 10000$)			SV subset			
w	time	SC error (%)	CC error (%)	m	time	#SVs	error (%)
10	2	6.22	8.77	7786	567	654	0.30
20	4	7.95	7.39	5985	244	654	0.30
30	5	10.06	6.55	5100	146	649	0.31
40	5	12.10	5.52	4563	104	645	0.31
50	6	13.53	4.65	4152	79	644	0.32
75	10	15.79	3.84	3456	45	642	0.32
100	15	16.57	3.10	2988	28	640	0.32
150	26	16.18	2.28	2407	15	631	0.32
200	41	14.29	1.89	2057	10	630	0.31
250	59	12.12	1.54	1805	7	627	0.32
300	81	9.86	1.34	1623	5	626	0.31
400	138	6.25	1.01	1382	3	622	0.31
500	218	3.85	0.90	1240	2	616	0.31
750	643	1.39	0.67	1024	1	608	0.32
1000	1193	0.76	0.57	925	1	615	0.31

Table 1: Average training time (in seconds) and test error rates for the approximation problem and the dense subproblem, parameterized by the half bandwidth w and with parameters $\gamma = 4$ and $\sigma = 32$. SC refers to the standard kernel classifier and CC refers to the completion kernel classifier. For the dense subproblem, m is the number of support vectors obtained from the approximation problem. We remark that solving the full QP ($m = 10000$) took 1194 seconds and produced 659 support vectors and a test error of 0.30 %.

4.3 Multistage method

In the previous experiment we were able to obtain a low error rate by solving a smaller, dense SVM QP with a reduced training set consisting of the support vectors from the approximation problem. This suggests using the approximated QP as a heuristic for working set selection. In our next experiment we solve a sequence of approximation problems with increasing bandwidth and decreasing training set size. We consider ten binary classification problems (each digit versus the nine other digits). For each instance, we first solve the approximation QP (8) using the full MNIST data set as training data and with half-bandwidth w_1. We refer to this as stage 1. At a subsequent stage i we solve (8) with half-bandwidth w_i and using as training data the support vectors from stage $i-1$. If the number of support vectors from stage j is below $\bar{m} = 6000$, we proceed to a final stage and solve the SVM QP with the exact kernel matrix Q. At most five stages are used before the final stage, regardless of the number of support vectors in stage 5. If the number of support vectors at stage 5 exceeds the threshold $\bar{m}$, we only keep the $\bar{m}$ training vectors with the largest values of z_i in the solution of the stage 5 problem. We choose $w_1 = 100$ and set $w_i \approx \sqrt{2} w_{i-1}$ at subsequent stages.

Table 2 lists the number of support vectors and the time required to solve the QP in each stage. Table 3 shows the total CPU time for all stages and the error rate for the resulting classifier. We can note that on average the total CPU time for the multiple stages is less than half the CPU time for LIBSVM. Except for digit 9, the test error rate obtained in the final stage is comparable to that of LIBSVM. Furthermore, the final stage produces slightly fewer support vectors than LIBSVM. Whether the smaller number of support vectors is due to the suboptimality of the approximation problem

Digit	Stage 1 ($w = 100$)		Stage 2 ($w = 141$)		Stage 3 ($w = 200$)		Stage 4 ($w = 282$)		Stage 5 ($w = 400$)		Final stage (dense)	
	time	#SVs	time	#SVs	time	#SVs	time	#SVs	time	#SVs	time	#SVs
0	94	15558	35	6290	22	3757	-	-	-	-	50	1584
1	94	8410	19	2688	-	-	-	-	-	-	21	1161
2	100	21417	48	11976	44	8175	47	6392	66	5102	120	2869
3	100	19525	47	10932	41	8097	47	6636	69	5477	150	3149
4	95	19395	44	10553	39	7179	41	5636	-	-	175	2713
5	100	24613	56	14841	55	10311	61	8059	86	6421	192	3192
6	100	17550	39	7564	27	4662	-	-	-	-	92	2002
7	99	16752	40	7354	26	5126	-	-	-	-	122	2511
8	95	23122	52	13367	50	9861	58	8110	86	6789	192	3633
9	100	19293	44	11802	44	8931	52	7585	80	6481	193	3722

Table 2: CPU time (seconds) and number of support vectors at each stage. The parameters $\gamma = 0.667$ and $\sigma = 32$ were used at all stages. The full training set ($m = 60000$) is used at stage 1, and at the following stages, the support vectors from the previous stage are used as training set. We skip to the final stage, where we solve a dense subproblem, when the number of support vectors is less than 6000. If there are more than 6000 support vectors at stage 5, we truncate the set based on the magnitude of the variables z_i. On average 9.2 interior-point iterations were needed to solve a subproblem.

Digit	Multistage approximation			LIBSVM		
	#SVs	time	error (%)	#SVs	time	error (%)
0	1584	203	0.21	1650	560	0.23
1	1161	135	0.24	1254	266	0.20
2	2869	430	0.52	3051	936	0.54
3	3149	456	0.65	3492	1100	0.51
4	2713	395	0.40	2869	804	0.49
5	3192	553	0.38	3412	1066	0.51
6	2002	260	0.32	2015	472	0.33
7	2511	289	0.71	2540	672	0.76
8	3633	536	0.72	4072	1205	0.67
9	3722	515	3.53	4138	1072	0.89

Table 3: Number of support vectors, total CPU time (in seconds), and test error rate for the multistage method as well as LIBSVM. We used the parameters $\gamma = 0.667$ and $\sigma = 32$ for all classifiers. The multistage method yields fewer support vectors in all cases. For digits 0–8, the test error rates obtained with the approximation method are similar or slightly better than those obtained with LIBSVM. The last classifier (9-versus-the rest), however, has an error rate of nearly four times that of LIBSVM. The approximation algorithm required a total of 3777 seconds whereas the total for LIBSVM was 8156 seconds.

m	Stage 1 ($w = 100$)		Stage 2 ($w = 200$)		Stage 3 (dense)		LIBSVM	
	#SVs	error (%)	#SVs	error (%)	#SVs	error (%)	#SVs	error (%)
2000	613	2.89	294	0.88	244	0.55	277	0.56
4000	1221	8.74	525	0.75	380	0.32	405	0.33
8000	2289	4.66	916	0.62	508	0.31	543	0.33
16000	4686	2.34	1687	0.84	712	0.28	764	0.28
32000	8628	1.97	2950	0.71	1075	0.24	1114	0.25
50000	13023	4.76	4188	0.81	1398	0.21	1435	0.23

Table 4: Test error rates and number of support vectors at the three stages and for LIBSVM. Here m is the number of training vectors used at stage 1 and with LIBSVM, and the error rates reported at stage 1 and 2 were obtained using the completion kernel classifier. The parameters $\gamma = 40000/m$ and $\sigma = 32$ were used.

Stage 1		Stage 2		Stage 3		Stage 1+2+3	LIBSVM
m	time	m	time	m	time	time	time
2000	3	613	1	294	0.09	5	2
4000	6	1221	4	525	0.3	11	6
8000	13	2289	8	916	1	23	15
16000	26	4686	18	1687	5	50	41
32000	57	8628	32	2950	25	115	128
50000	85	13023	52	4188	74	212	424

Table 5: CPU time (in seconds) for the three stages and for LIBSVM. The parameters $\gamma = 40000/m$ and $\sigma = 32$ were used. The time at stages 1 and 2 grows linearly with m whereas at stage 3, which involves a dense QP, the time grows faster than linear.

or simply related to thresholding is not clear. On average about nine interior-point iterations were required at each stage and it is possible that this number can be reduced by using a warm-start technique. Finally notice that the number of support vectors for each of the ten classifiers varies by more than a factor of three.

4.4 Number of training vectors versus test error rate

In the last example we consider the test error rate as a function of the number of training vectors m. As in the previous experiment, we use a multi-stage approach where the last stage is a dense QP, but in this experiment we do not use a threshold before the final dense SVM QP. We train a single one-versus-the rest classifier with 0 as class 1 and 1–9 as class 2. The training set consists of m_+ randomly chosen examples from class 1 and $m_- = 9m_+$ randomly chosen examples from class 2. The test set is the full MNIST test set. Table 4 shows the test error rate at each stage as well as the test error rate obtained with LIBSVM. As can be seen, the number of support vectors at stage 1 grows roughly linearly with the number of training vectors. As a result, the overall CPU time grows faster than linearly (more or less quadratically) with the training set size. This growth rate is comparable with LIBSVM.

5 Other applications

Sparse inverse approximations of dense positive definite matrices are useful for a variety of optimization problems. In this section we mention two interesting further examples from machine learning.

5.1 Gaussian process classification

We first consider Gaussian process (GP) classification [RW06, Section 3.3]. In two-class GP classification it is assumed that the probability of observing a binary outcome ± 1 at a point $x \in \mathbf{R}^n$ depends on the value of a *latent variable* $f(x)$ via the formulas

$$\mathbf{prob}\,(\text{outcome at } x \text{ is } 1 \mid f(x)) = \kappa(f(x))$$

and

$$\begin{aligned}\mathbf{prob}\,(\text{outcome at } x \text{ is } -1 \mid f(x)) &= 1 - \kappa(f(x)) \\ &= \kappa(-f(x)),\end{aligned}$$

where $\kappa : \mathbf{R} \to \mathbf{R}$ is a symmetric function (*i.e.*, satisfying $\kappa(-u) = 1 - \kappa(u)$) with values in $[0, 1]$. The latent variable $f(x)$ is assumed to be random with a zero-mean Gaussian process with covariance function $h(x, y) = \mathbf{E}\, f(x)f(y)$ as prior distribution. Common choices for κ are the logistic function $\kappa(u) = 1/(1 + \exp(-u))$ and the probit function (the cumulative density of a zero-mean unit-variance Gaussian). We note that these two functions $\kappa(u)$ are log-concave.

Suppose we have a training set of observed outcomes $d_i \in \{-1, 1\}$ at m training points x_i. To simplify the notation we denote by $F = (f(x_1), \ldots, f(x_m))$ the random vector of latent variables at the training points. The first step in deriving the GP classifier is a maximum a posteriori (MAP) estimation of F given the observed outcomes $d = (d_1, \ldots, d_m)$. The MAP estimate of F is the solution of the optimization problem

$$\text{maximize} \quad L(u) = \sum_{i=1}^{m} \log \kappa(d_i u_i) - \frac{1}{2} u^T Q^{-1} u - \frac{1}{2} \log\det Q \tag{22}$$

with variable $u \in \mathbf{R}^m$. The matrix Q has elements $Q_{ij} = h(x_i, x_j)$ and is the covariance of the prior distribution $\mathcal{N}(0, Q)$ of F. The function $L(u)$ is, up to a constant, the logarithm of the posterior density $p_{F|d}(u)$ of the latent variables F, given the observed outcomes d_i, $i = 1, \ldots, m$. The MAP estimate of the latent variables at the training points is the solution of (22). Problem (22) is a convex optimization problems if the function κ is log-concave. We refer the reader to [RW06, §3.4.2] for the details on how the MAP estimate is applied for classifying test points.

The key step in GP classification is the solution of the unconstrained optimization problem (22). If the function κ is log-concave, this problem is convex and can be solved by Newton's method. Each step requires the solution of an equation $\nabla^2 L(u)\Delta u = \nabla L(u)$, or

$$\left(Q^{-1} + D\right) \Delta u = -\nabla L(u) \tag{23}$$

where D is diagonal with diagonal elements

$$D_{ii} = \frac{\kappa'(d_i u_i)^2 - \kappa''(d_i u_i)\kappa(d_i u_i)}{\kappa(d_i u_i)^2}.$$

For most common kernel functions the covariance Q and its inverse Q^{-1} are dense, so the Newton equation may be expensive to solve when n is large. The complexity can be improved by making low-rank or low-rank-plus-diagonal approximations, as described in section 1 (see also [RW06, section 3.4.3]). Clearly, the Newton equation (23) becomes much easier to solve if we can replace Q^{-1} by a sparse approximation $\tilde{Q}^{-1}$, obtained via a maximum determinant positive definite completion $\tilde{Q}$ of $Q_{ij} = h(x_i, x_j)$, $(i,j) \in V$. This is equivalent to replacing the Gaussian prior $\mathcal{N}(0, Q)$ in (22) with the maximum entropy distribution that matches the moments $\mathbf{E}\, f(x_i)f(x_j) = h(x_i, x_j)$, $(i,j) \in V$.

5.2 Kernel PCA

In the two applications discussed so far, we have used sparse (zero-fill) Cholesky factorizations of the inverse completion kernel to reduce the linear algebra cost of interior-point algorithms and Newton's method. Another benefit of the sparse inverse is that it simplifies matrix-vector products. As an example, we discuss how the matrix completion techniques can be exploited in kernel principal component analysis (PCA).

The main computational effort in kernel PCA lies in computing an eigenvalue decomposition of the centered kernel matrix [SSM98]

$$\begin{aligned} Q_c &= (I - (1/m)\mathbf{1}\mathbf{1}^T)Q(I - (1/m)\mathbf{1}\mathbf{1}^T) \\ &= Q - (1/m)\mathbf{1}\mathbf{1}^T Q - (1/m)Q\mathbf{1}\mathbf{1}^T + (1/m^2)\mathbf{1}\mathbf{1}^T Q \mathbf{1}\mathbf{1}^T \end{aligned} \tag{24}$$

where $Q_{ij} = h(x_i, x_j)$. Computing the entire eigenvalue decomposition is clearly not feasible when the number of observations m is large, so in practice some approximation is often used, for example, sparse greedy methods [SS00]. If matrix-vector products with Q_c are cheap, the dominant eigenvalues and eigenvectors can also be computed iteratively, using the Lanczos method [GL96].

Suppose we replace the dense kernel matrix Q with the maximum determinant positive definite completion $\tilde{Q}$ of a partially specified kernel matrix. The matrix-vector product $\tilde{Q}_c v$ can then be evaluated efficiently given the sparse factorization $\tilde{Q}^{-1} = RR^T$, *i.e.*, using (24),

$$\tilde{Q}_c v = R^{-T}(R^{-1}v - z\mathbf{1}^T v) - \mathbf{1}z^T R^{-1}v + z^T z \mathbf{1}\mathbf{1}^T v \tag{25}$$

where z is the solution of $Rz = \mathbf{1}$. The cost of evaluating (25) is *linear* in m for band patterns with fixed bandwidth. Hence the dominant eigenvalues of $\tilde{Q}_c$ and the corresponding eigenvectors can then be computed efficiently using the Lanczos method.

6 Conclusions

Interior-point methods for SVM training provide robust and accurate solutions but are well known to be limited by the high demands of computation time and storage, as a consequence of the density of the kernel matrix. Scaling interior-point methods to training set sizes above 10000 therefore requires approximations of the dense kernel matrix. Examples of such approximations that have been studied in the literature include low-rank, diagonal-plus-low-rank, and sparse approximations. In this paper we investigate the use of sparse inverse approximations. By approximating the positive definite kernel matrix by the maximum determinant positive definite completion of a partially specified kernel matrix with chordal sparsity, we obtain an approximate QP where the inverse of the kernel matrix is sparse. Exploiting the sparsity of the inverse in an interior-point method

leads to a dramatic improvement in solution time and memory requirements. As a consequence, very large problems can be solved on a standard desktop PC using an interior-point method. Numerical results with band sparsity indicate that the method is typically faster than LIBSVM while the test error rates are comparable. However, more experimentation is needed to evaluate the robustness of the method across different kernel functions and data sets. The positive definite matrix completion techniques should also be of interest in other applications that involve large dense convex optimization problems. As two examples in machine learning we have mentioned Gaussian process classification and kernel PCA.

References

[CL01] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[CT91] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[Dem72] A. P. Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

[DV08] J. Dahl and L. Vandenberghe. *CVXOPT: A Python Package for Convex Optimization*. `abel.ee.ucla.edu/cvxopt`, 2008.

[DV09] J. Dahl and L. Vandenberghe. *CHOMPACK: Chordal Matrix Package*. `abel.ee.ucla.edu/chompack`, 2009.

[DVR08] J. Dahl, L. Vandenberghe, and V. Roychowdhury. Covariance selection for non-chordal graphs via chordal embedding. *Optimization Methods and Software*, 23(4):501–520, 2008.

[FCH+08] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[FM03] M. C. Ferris and T. S. Munson. Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804, 2003.

[FS02] S. Fine and K. Scheinberg. Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2002.

[GJSW84] R. Grone, C. R. Johnson, E. M Sá, and H. Wolkowicz. Positive definite completions of partial Hermitian matrices. *Linear Algebra and Appl.*, 58:109–124, 1984.

[GL96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 3rd edition, 1996.

[HCL+08] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 408–415, New York, NY, USA, 2008. ACM.

[Joa99] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

[Joa06] T. Joachims. Training linear SVMs in linear time. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, pages 217–226, 2006.

[JY09] T. Joachims and C.-N. J. Yu. Sparse kernel SVMs via cutting-plane training. *Machine Learning*, 76(2-3):179–193, 2009. European Conference on Machine Learning (ECML) Special Issue.

[Kul97] S. Kullback. *Information Theory and Statistics*. Dover Publications, 1997. Originally published by John Wiley & Sons, 1959.

[Lau01] M. Laurent. Matrix completion problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, volume III, pages 221–229. Kluwer, 2001.

[LC98] Y. LeCun and C. Cortes. The MNIST database of handwritten digits. Available at `http://yann.lecun.com/exdb/mnist/`, 1998.

[NW06] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[OFG97] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the 7th IEEE Workshop on Neural Networks for Signal Processing*, pages 276–285, 1997.

[Pla99] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 185–208. MIT Press, Cambridge, MA, 1999.

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[SS00] A. J. Smola and B. Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th International Conference on Machine Learning*, pages 911–918. Morgan Kaufmann, 2000.

[SS02] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[SSM98] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[Wri97] S. J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, 1997.