Efficient SVM Training Using Low-Rank Kernel Representations

Shai Fine FSHAI@US.IBM.COM
Human Language Technologies Department
IBM T. J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598, USA

Katya Scheinberg KATYAS@US.IBM.COM
Mathematical Science Department
IBM T. J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598, USA

Editors: Nello Cristianini, John Shawe-Taylor, Bob Williamson

Abstract

SVM training is a convex optimization problem which scales with the training set size rather than the feature space dimension. While this is usually considered to be a desired quality, in large scale problems it may cause training to be impractical. The common techniques to handle this difficulty basically build a solution by solving a sequence of small scale subproblems. Our current effort is concentrated on the rank of the kernel matrix as a source for further enhancement of the training procedure. We first show that for a low rank kernel matrix it is possible to design a better interior point method (IPM) in terms of storage requirements as well as computational complexity. We then suggest an efficient use of a known factorization technique to approximate a given kernel matrix by a low rank matrix, which in turn will be used to feed the optimizer. Finally, we derive an upper bound on the change in the objective function value based on the approximation error and the number of active constraints (support vectors). This bound is general in the sense that it holds regardless of the approximation method.

Keywords: Support Vector Machine, Interior Point Method, Cholesky Product Form, Cholesky Factorization, Approximate Solution

1. Introduction

In the core of the SVM training problem lies a convex optimization problem which scales with the training set size rather than the feature space dimension (Boser et al., 1992, Vapnik, 1995). While this is usually considered to be a desired quality, since it circumvents the well known "curse of dimensionality", in large scale problems (which are so common in real world application such as speech, document classification, OCR, etc.) it may actually raise a new concern: Although the training problem is, in principle, solvable, in practice it is intractable by the conventional optimization techniques; e.g., each iteration of a general interior point method (IPM) scales cubically with the size of the training set. Several approaches to handle this problem have been suggested in the past few years which basically build a solution by solving a sequence of small scale subproblems. To mention a few: stochastic gradient ascent algorithms such as the Kernel-Adatron (Friess et al., 1998) and the SMO

(Platt, 1999), which sequentially update one or two (resp.) Lagrange multipliers at every training step, and active set methods, such as Chunking (Boser et al., 1992), Decomposition (Osuna et al., 1997) and Shrinking (Joachims, 1999), which gradually build the (hopefully) small set of active constraints by feeding a generic optimizer (usually an interior point algorithm) with small scale subproblems.

Our current effort is concentrated on the numerical rank of the kernel matrix as a source for further enhancement of the performance of the IPM. To this end we make the following three key observations:

1. If the kernel matrix has *low rank*, then it is possible to design an efficient interior point method that scales *linearly* with the size of data set.

2. If the kernel matrix has *full rank* (or almost so), it may be possible to approximate it by a low rank positive semidefinite matrix (which in turn can be used to feed the optimizer).

3. By using an approximation matrix instead of the original kernel matrix we obtain a perturbed optimization problem, which requires less computational effort to solve. In general, the optimal value of the objective function of the perturbed problem is different than the optimal value of the original problem. However, the size of the difference can be explicitely controlled by the quality of the approximation.

We assume that the positive semidefinite kernel matrix, $Q_{n\times n}$, is totally dense and too large to handle, however its rank, k, is significantly smaller than n ("significantly" may be defined according to the context). This means that Q can be represented as $Q = VV^T$, where V is a matrix with k columns and n rows. The most expensive step at every iteration of an IPM (applied to an SVM problem) is inverting or factorizing a matrix of the form $D + Q$, where D is a diagonal (positive) matrix and Q is as described above. In general, this operation requires $O(n^2)$ storage space and takes $O(n^3)$ arithmetic operations, but by handling the linear algebra in a special way we can reduce the complexity to $O(nk^2)$ and the storage to $O(nk)$.

There are two possible approaches that achieve the same complexity. The first approach is based on the Sherman-Morrison-Woodbury formula for a low rank update of an inverse of a matrix. The Sherman-Morrison-Woodbury formula has been used widely in the context of interior point methods for linear programming (Choi et al., 1990, Marxen, 1989). In that context, however, the method often runs into numerical difficulties. The second approach is based on product form Cholesky factorization and is described in Section 3. This method is somewhat more expensive than the first: its workload is about twice and the storage requirement is three times the storage needed for the first method. On the other hand, the second method was shown to be numerically stable (experimentally and theoretically) in the context of interior point methods for LP and quadratically constrained quadratic problems (Goldfarb and Scheinberg, 1999). Some examples that we present in Section 3 suggest that the first method may also have numerical difficulties in the context of our QP problems. The second method works well numerically for all problems that we tried.

In many applications that we considered, a low rank representation $Q = VV^T$ may not be available a priori (even if it exists). It may also happen that even though the rank of Q is close to (or equal to) n, Q can be approximated by a low rank symmetric positive semidefinite matrix $\tilde{Q}$. In this case one can apply incomplete Cholesky factorization method with symmetric pivoting to obtain the desired representation of Q or its approximation. This procedure requires $O(nk^2)$ operations, where k is the number of nonzero pivots in the procedure, which is the same as the rank of matrix Q or of its low rank approximation. The storage requirements reduces to $O(nk)$. It is possible to trace the quality of the approximation and apply an appropriate stopping criteria without an extra effort. This and the overall complexity makes this method compare favorably with other recently suggested approximation techniques (Smola and Schölkopf, 2000, Williams and Seeger, 2001).

Finally, the last observation leads us to derive a bound on the quality of the solution obtained by solving the perturbed problem using a low rank approximation to Q. We show that if the approximation error $Q - \tilde{Q}$ is bounded (in some sense) by ϵ, then the difference between the optimal objective function values of the original problem and the perturbed problems is bounded by $lc^2\epsilon$, where c is the penalty (cost) parameter of the training error and l is the number of active points in the training set of the perturbed problem.

The rest of this paper is organized as follows: In the next section we present the convex QP that we are interested in solving, and the interior point method that we use. In Section 3 we describe the Sherman-Morrison-Woodbury method and the product form Cholesky factorization method, as means for low rank updates. Section 4 is devoted to issues related to approximating the kernel matrix by a low-rank positive semidefinite matrix. We present the algorithm to find such an approximation and a theorem in which we derive the above mentioned bound. Experiments, which demonstrate the utility of the suggested method, are presented in Section 5. We wrap things up in Section 6 with some concluding remarks and leading directions for further study.

1.1 Notation

In general we denote scalars and vectors in lower case letters (we make an explicit distinction when it is not clear from the context), and we use upper case letters to denote matrices. We use subscript indices to indicate elements of a vector or a matrix and we use superscript to label a whole matrix of a vector with an index.

The description in this paper follows the traditional notation used in the optimization literature, which is slightly different from the one used in most SVM papers. The main difference is in the identification of the primal and dual problems (which switch roles) and the convention used to denote the unknowns with x, y, while in the SVM literature x and y are used to denote the input vectors and their labels, resp. Hence, the SVM training problem with 1-norm soft margin is

$$
(SVM) \qquad \begin{array}{ll} \min_{\xi,\mathrm{w},\mathrm{b}} & \frac{1}{2}\langle w, w\rangle + c\sum_{i=1}^{n}\xi_i \\ \text{s.t.} & a_i\left(\langle w, v_i\rangle + b\right) \geq 1 - \xi_i, \quad i = 1,\ldots,n \\ & \xi_i \geq 0, \qquad i = 1,\ldots,n \end{array}
$$

where labeled examples are pairs of the form (v, a) such that v is a finite or infinite dimension feature vector, $a \in \{-1, +1\}$ is a label, and $\langle\cdot,\cdot\rangle$ is an inner product. We often use Q as a shorthand for the labeled kernel matrix, i.e.

$$Q_{i,j} = a_i a_j K(v_i, v_j)$$

where $K(\cdot,\cdot)$ is a (Mercer) kernel function representing dot-products in the feature (RKHS) space.

We will use an equivalent form of the above problem (with somewhat different notation) which we will refer to as the *dual* problem (see problem (D) in the next section). The dual of the (SVM) problem is thus called the *primal* problem (see problem (P) in the next section).

2. An Interior Point Method

We consider the following convex quadratic programming problem.

$$(P) \qquad \begin{array}{ll} \min_{\mathrm{x}} & \frac{1}{2}x^T Q x - e^T x \\ \text{s.t.} & a^T x = 0, \\ & 0 \leq x \leq c, \end{array}$$

where $x \in \mathbf{R^n}$ is the vector of primal variables, e is the vector of all 1's of length n and a is a vector of labels, 1's and -1's. c is the penalty parameter associated with the training error. In general it may be desirable to use different penalty parameters for different data points. In this case c should be treated as an n-dimensional vector. However, for the sake of simplicity we assume that c is a positive scalar. All our analysis and results extend easily to the more general case. Similarly, one might want to solve a problem with a slightly more general objective function $\frac{1}{2}x^T Qx + q^T x$, where q is some n-dimensional vector (e.g., a small scaled optimization problem arising within a chunking-like meta algorithms). Our analysis and results extend easily to this case as well.

The dual to the above problem is

$$(D) \qquad \begin{array}{ll} \max_{\mathrm{y,s}} & -\frac{1}{2}x^T Q x - c\sum_{i=1}^{n} \xi_i \\ \text{s.t.} & -Qx + ay + s - \xi = -e, \\ & s \geq 0,\ \xi \geq 0, \end{array}$$

Here y is the scalar dual variable (it is equal to the negative bias) and s and ξ are the n-dimensional vectors of dual variables.

Both (P) and (D) have finite optimal solutions and any optimal primal-dual pair of solutions satisfies the Karush-Kuhn-Tucker (KKT) necessary and sufficient optimality conditions:

$$\begin{aligned} &Xs = 0, \\ &(C - X)\xi = 0 \end{aligned}$$

$$
\begin{aligned}
&a^T x = 0,\\
&-Qx + ay + s - \xi = -e,\\
&0 \le x \le c,\ \ s \ge 0,\ \ \xi \ge 0.
\end{aligned}
$$

By X (S, $C - X$) we denote the diagonal matrix whose diagonal entries are the elements of the vector x (s, $c - x$). For optimality conditions of QP and details of applying IPMs to QP (see Wright, 1997) and references therein.

The perturbed KKT optimality conditions

$$
(CP_\mu) \qquad
\begin{aligned}
&Xs = \sigma\mu e,\\
&(C - X)\xi = \sigma\mu e\\
&a^T x = 0,\\
&-Qx + ay + s - \xi = -e,\\
&0 \le x \le c,\ \ s \ge 0,\ \ \xi \ge 0
\end{aligned}
$$

have a unique solution for any positive values of the parameter μ and σ. The trajectory of solutions to the above system of equations for all values of $\mu \in (0, \infty)$ with some fixed positive value of σ is called the *central path*. The central path converges to an optimal solution. A path-following primal-dual interior point method tries to follow the central path towards the optimal solution by iteratively approximating solutions on the path by applying the Newton method to the above system of nonlinear equations, while reducing the value of parameter μ on each iteration.

Any typical interior point method solves a linearization of the above system of nonlinear equations. We use the so-called Mehrotra predictor-corrector algorithm (Mehrotra, 1992), which is considered to be one of the most efficient in practice, however the ideas presented in section 3 on reducing the per-iteration complexity apply to any other interior point method. At every iteration there are two basic steps - the predictor step and the corrector step. The idea behind this algorithm is to start with "predicting" the best reduction in the duality gap, by evaluating a step directly towards the optimality. Such step, however, is not taken, since it may ruin the "centrality" of the iterates. A corrector step is computed instead. The corrector step enforces the centrality and also takes into account the approximate curvature of the central path predicted by the predictor step.

Predictor step is a Newton step towards the solution of the system with $\sigma = 0$ (which is the same as the system of the KKT conditions). Then, using this step, a value for σ is chosen and a better step (corrector step) towards solution of the above system with this value of σ is taken.

For the predictor step we compute $(\Delta x, \Delta y, \Delta s, \Delta\xi)$, satisfying

$$
\begin{bmatrix}
-Q & a & I & -I\\
a^T & 0 & 0 & 0\\
S & 0 & X & 0\\
-\Xi & 0 & 0 & (C - X)
\end{bmatrix}
\begin{bmatrix}
\Delta x\\ \Delta y\\ \Delta s\\ \Delta\xi
\end{bmatrix}
=
\begin{bmatrix}
r_c\\ r_b\\ -Xs\\ -(C - X)\xi
\end{bmatrix}
$$

where $r_b = -a^T x$, $r_c = -e + Qx - ay - s + \xi$. If the current solution is primal and dual feasible, then r_b and r_c are zero (within the numerical accuracy).

By eliminating Δs, $\Delta\xi$ and consequently Δx from the system we obtain an expression for Δy: $a^T(Q+D)^{-1}a\Delta y = r$, where $D = SX^{-1} + \Xi(C-X)^{-1}$ and r is a right hand side that depends on $(Q+D)^{-1}$ and other parameters of the system. Solving for Δy we substitute it back to find Δx, Δs and $\Delta\xi$. The maximum possible step length is determined by computing $\alpha \leq 1$ such that $0 \leq x + \alpha\Delta x \leq c$, $0 \leq s + \alpha\Delta s$ and $0 \leq \xi + \alpha\Delta\xi$.

To compute the corrector step, we set $dx = \Delta x$, $ds = \Delta s$ and $d\xi = \Delta\xi$ and $\hat{\mu} = (x+dx)^T(s+ds) + (c-x-dx)^T(\xi+d\xi)$ (from the predictor step) and compute new $(\Delta x, \Delta y, \Delta s, \Delta\xi)$, satisfying

$$\begin{bmatrix} -Q & a & I & -I \\ a^T & 0 & 0 & 0 \\ S & 0 & X & 0 \\ -\Xi & 0 & 0 & (C-X) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \\ \Delta\xi \end{bmatrix} = \begin{bmatrix} r_c \\ r_b \\ \sigma\mu - dXds - Xs \\ \sigma\mu + dXd\xi - (C-X)\xi \end{bmatrix}$$

where σ is defined by

$$\sigma = \left(\frac{\hat{\mu}}{\mu}\right)^3.$$

We solve this system similarly to the system for the predictor step, compute the largest possible step (but actually take 99% of that step to avoid hitting the boundaries of the feasible region or else the new solution would not be in the interior), and update the current solution.

The stopping criteria is formed by checking the duality gap and the primal and dual feasibility against a predetermined tolerance. If these conditions are still not met, we compute a new value for the parameter μ and repeat the iteration. This algorithm converges to an optimal solutions from any strictly interior starting point (a point that lies inside all inequality constraints). In theory, it takes $O(n\ln(1/\epsilon))$ iterations to converge, where ϵ is the relative accuracy. However, in practice, an interior point method rarely takes more than 50 iterations, regardless of the problem's size.

The most time consuming operation on every step is obtaining a vector of the form $u = (Q+D)^{-1}w$ (e.g., during the computation of the predictor step such operation is done twice - once during computation of the right hand side vector r and once during computation of $a^T(Q+D)^{-1}a$). Since $Q+D$ is a dense $n \times n$, symmetric positive definite matrix, then in general obtaining the vector u requires $O(n^3)$ operations. Typically this can be done by either computing the inverse or factorizing[1] $Q+D$.

The main point of the next section is to show that if Q can be represented as $Q = VV^T$, where V is $n \times k$ matrix (where $k << n$), then vector u can be obtained in $O(k^2 n)$ operations.

3. Low-rank updates

There are two methods which efficiently solve the system $(D + VV^T)u = w$ by taking into consideration the special form of the matrix of the system (i.e., the fact that it is a diagonal matrix plus a low-rank symmetric positive semidefinite matrix). Both of these methods require $O(nk)$ storage and $O(nk^2)$ arithmetic operations, where k is the number of columns

1. Note that since D and Q are the same for both the predictor and the corrector steps, we need to compute the inverse or factorize $D+Q$ only once per every iteration.

in V. Thus, if $k << n$ then the two methods provide significant savings in terms of workload and storage.

The first method is somewhat cheaper than the second. Its workload is about half of that of the second method and the storage requirement is one third of the storage of the second method. On the other hand, the first method may experience numerical instability.

3.1 Sherman-Morrison-Woodbury update

We consider a classical method for a low-rank update of an inverse of a matrix. This update is usually called the Sherman-Morrison-Woodbury (SMW) update or formula:

$$(D + VV^T)^{-1} = D^{-1} - D^{-1}V(I + V^T D^{-1}V)^{-1}V^T D^{-1}.$$

Since we would like to avoid storing the matrix $D + VV^T$, we also would like to avoid storing its inverse. We can apply the above formula to solve the system $(D + VV^T)u = w$ as follows:

$$u = D^{-1}w - D^{-1}V(I + V^T D^{-1}V)^{-1}V^T D^{-1}w.$$

Hence, compute

$$\begin{aligned} &z := D^{-1}w \\ &t:\ (I + V^T D^{-1}V)t = V^T z \\ &u := z - D^{-1}Vt. \end{aligned}$$

The system in the second step is $k \times k$ symmetric positive definite and can be solved by computing Cholesky factorization of $(I + V^T D^{-1}V)$. The work required to compute such factorization is $O(k^3)$, which is small if k is small with respect to n. Computing the matrix of that system takes $nk^2/2 + O(nk)$ multiplications (and the same number of additions). The overall work, in terms of the number of multiplications, required to solve $(D + VV^T)u = w$ is $nk^2/2$ plus smaller order terms.

The Sherman-Morrison-Woodbury has been used widely in the context of interior point methods for linear programming, (Choi et al., 1990, Marxen, 1989). In that context, however, the method often runs into numerical difficulties. There are few modifications of the method that have been proposed to deal with the numerical instability (Andersen, 1996, Scheinberg and Wright, 2000). However, these modifications do not always help, and they increase the computational cost.

It is not easy to point out rigorously the situations for which the SMW formula is numerically unstable. The intuition suggests that it happens when matrix $D + VV^T$ or just D is not well conditioned. Consider the following small example:

Example 1 *Let*

$$D = \begin{bmatrix} \epsilon^2 & \\ & 1 \end{bmatrix} \quad \text{and} \quad V = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Then assuming that ϵ^2 is smaller than the relative machine precision - i.e., $1 \pm \epsilon^2$ is computed as 1 *- we have that*

$$D + VV^T \approx \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}. \tag{1}$$

Now consider solving $(D + VV^T)u = w$, *where* $w = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}$ *and* w_1 *and* w_2 *are* $\Theta(1)$. *Clearly*

$$u \approx \begin{pmatrix} 2w_1 + w_2 \\ w_1 + w_2 \end{pmatrix}. \tag{2}$$

Let us apply the Sherman-Morrison-Woodbury formula to this example.

$$z = \begin{pmatrix} \frac{w_1}{\epsilon^2} \\ w_2 \end{pmatrix}$$

$$(V^T D^{-1} V + I)t = (2 + \frac{1}{\epsilon^2})t \quad \text{and} \quad V^T z = \frac{w_1}{\epsilon^2} - w_2.$$

Because of round-off errors $V^T D^{-1} V + I$ *is computed as* $1/\epsilon^2$ *and* $V^T z$ *as* w_1/ϵ^2. *Hence* $t = w_1$,

$$u = z - D^{-1}Vt = \begin{pmatrix} \frac{w_1}{\epsilon^2} - \frac{w_1}{\epsilon^2} \\ w_2 + w_1 \end{pmatrix} = \begin{pmatrix} 0 \\ w_2 + w_1 \end{pmatrix}.$$

Clearly, the computed solution is not even close to the approximately correct solution (2).

A tuation similar to the above example happens often in linear programming. For our QP problems such a situation is less natural, but still possible. The diagonal elements of D are $s_i/x_i + \xi_i/(c - x_i)$. By complementarity, as μ approaches zero, either $s_i \to 0$ and $x_i \to x_i^* > 0$ or $x_i \to 0$ and $s_i \to s_i^* > 0$ (analogously, either $\xi_i \to 0$ and $x_i \to x_i^* < c$ or $x_i \to c$ and $\xi_i \to \xi_i^* > 0$). Moreover, whenever a variable converges to zero, it converges with the same rate as the parameter μ. Thus, whenever s_i and ξ_i both converge to zero (which happens for the points that are in the active optimal set), then D_i converges to zero with the same rate as μ. Otherwise, D_i converges to infinity with the same rate as $1/\mu$. If $\mu = 10^{-8}$, then some elements of D are of the order 10^{-8} and the others are of the order 10^8. If we scale D and Q by 10^{-8} then D looks similar to the diagonal matrix in the example; i.e., some of the elements are of the order of 1 and some are of the order of 10^{-16} (which often is the order of the machine precisions). To have matrix Q and V consistent with the example, we need the elements of scaled matrix $10^{-8}Q$ to be of the order of 1. This will happen if the elements of V, e.g. the data points, are of the order 10^4. In section 5 we briefly discuss some examples of practical SVM problems for which our implementation of the interior point method with the SMW update failed to converge.

In the next subsection we present an alternative method, which avoids many of the numerical difficulties of the SMW update.

3.2 Product-Form Cholesky Factorization

An alternative efficient method for solving the system $(D + VV^T)u = w$ is based on a low rank update of the Cholesky factorization of the matrix, rather than using low rank updates to the inverse of the matrix. This is the reason for good numerical properties of this approach. We will use a, so called, product form Cholesky factorization of $D + VV^T$.

Here a Cholesky factorization of a symmetric positive semidefinite matrix $M \in \mathbf{R^{n\times n}}$ is defined as $M = L\Lambda L^T$, where L is a lower triangular matrix with ones on the diagonal, and Λ is a nonnegative diagonal matrix. A product form Cholesky factorization, in general, is of the form $\tilde{L}^1\tilde{L}^2\ldots\tilde{L}^k\tilde{\Lambda}^k(\tilde{L}^k)^T\ldots(\tilde{L}^2)^T(\tilde{L}^1)^T$, where each $\tilde{L}^i$ is a lower triangular matrix and $\tilde{\Lambda}^k$ is a nonnegative diagonal matrix.

To solve a system of linear equations $Mu = w$, one has to solve the following sequence of linear systems: $\tilde{L}^1u^1 = w$, $\tilde{L}^2u^2 = u^1$, $\ldots$, $\tilde{L}^ku^k = u^{k-1}$, $(\tilde{L}^k)^T u^{k+1} = (\Lambda^k)^{-1}u^k$, $\ldots$, $(\tilde{L}^1)^T u = u^{2k-1}$. Each such system has a lower triangular or an upper triangular matrix, and, in a general dense case, solving each such system takes $O(n^2)$ operations. However, if each $\tilde{L}^i$ has a special structure (such as we will see below) then solving each system takes only $O(n)$ operations.

Let us assume that we have a Cholesky factorization of a matrix M: $M = L\Lambda L^T$. We would like to obtain a product form factorization of $M + vv^T$, where $v \in \mathbf{R^n}$ is a vector (hence, vv^T is a rank-one $n \times n$ matrix):

$$L\Lambda L^T + vv^T = L(\Lambda + pp^T)L^T = L\tilde{L}\tilde{\Lambda}\tilde{L}^T L^T,$$

where p is the solution of the equations $Lp = v$ and $\tilde{L}\tilde{\Lambda}\tilde{L}^T$ is the Cholesky factorization of $\Lambda + pp^T$.

$\tilde{L}$ has a special form

$$\tilde{L} = \begin{bmatrix} 1 & & & & & \\ p_2\beta_1 & 1 & & & & \\ p_3\beta_1 & p_3\beta_2 & 1 & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ p_{n-1}\beta_1 & p_{n-1}\beta_2 & p_{n-2}\beta_3 & \cdots & 1 & \\ p_n\beta_1 & p_n\beta_2 & p_n\beta_3 & \cdots & p_n\beta_{n-1} & 1 \end{bmatrix}, \qquad (3)$$

where $\beta \in \mathbf{R^n}$ and $\tilde{\Lambda} = \operatorname{diag}\{\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n\}$ can be computed from the following recurrence relations:

$$\begin{aligned} & t_0 := 1, \\ & \text{for } j = 1, 2, \ldots, n \\ & \quad t_j := t_{j-1} + p_j^2/\lambda_j, \\ & \quad \tilde{\lambda}_j := \lambda_j t_j / t_{j-1}, \\ & \quad \beta_j := p_j/(\lambda_j t_j). \end{aligned}$$

Various versions of this method for updating Cholesky factorization by a rank-one term were proposed by Bennet (1965), Fletcher and Powell (1974) and Gill et al. (1975).

If we allow some λ_j to be zero, then it is easy to verify that the above recurrence relations become:

It is often important for the purpose of numerical stability to consider λ_j as equal to zero, even if it is not precisely zero, but is very small (relatively to the other elements of Λ). The above recursions can be used (with slight modifications) in the approximate sense as well.

$$
\begin{array}{l}
t_0 := 1, \\
\textbf{for } j = 1, 2, \ldots, n \\
\quad \textbf{if } t_{j-1} \neq \infty \\
\quad\quad \textbf{if } \lambda_j \neq 0 \\
\quad\quad\quad t_j := t_{j-1} + p_j^2/\lambda_j, \\
\quad\quad\quad \tilde{\lambda}_j := \lambda_j t_j / t_{j-1}, \\
\quad\quad\quad \beta_j := p_j/(\lambda_j t_j). \\
\quad\quad \textbf{else} \\
\quad\quad\quad \textbf{if } p_j \neq 0 \\
\quad\quad\quad\quad t_j := \infty \\
\quad\quad\quad\quad \tilde{\lambda}_j := p_j^2/t_{j-1}, \\
\quad\quad\quad\quad \beta_j := 1/p_j. \\
\quad\quad\quad \textbf{else} \\
\quad\quad\quad\quad t_j := t_{j-1} \\
\quad\quad\quad\quad \tilde{\lambda}_j := 0 \\
\quad\quad\quad\quad \beta_j := 0 \\
\quad \textbf{else} \\
\quad\quad t_j := \infty \\
\quad\quad \tilde{\lambda}_j := \lambda_j, \\
\quad\quad \beta_j := 0.
\end{array}
$$

Figure 1: Rank-One update for Product-Form Cholesky Factorization

Now assume that we have a factorization $L\Lambda L^T$ and we want to obtain a product form factorization of $L\Lambda L^T + v^1(v^1)^T + v^2(v^2)^T + \ldots + v^k(v^k)^T$. By repeating the above procedure k times, once for each rank-one term $v^i(v^i)^T$, we can obtain the product from Cholesky factorization $L\tilde{L}^1\tilde{L}^2\ldots,\tilde{L}^k\tilde{\Lambda}^k(\tilde{L}^k)^T\ldots(\tilde{L}^2)^T(\tilde{L}^1)^T L^T$ from the following:

Procedure 1.

1. Set $\tilde{\Lambda}^0 := \Lambda$,

2. For $i = 1, \ldots, k$

- **(i)** Solve $Lq^{0,i} = v^i$ for $q^{0,i}$;
- **(ii)** For $j = 1, \ldots, i-1$, solve $\tilde{L}^j q^{j,i} = q^{j-1,i}$ for $q^{j,i}$;
- **(iii)** Set $p := q^{i-1,i}$; compute β and $\tilde{\Lambda}$ by recurrence relations in Fig. 1;
- **(iv)** Store $p^i := p$ and $\beta^i := \beta$; replace $\tilde{\Lambda}^{i-1}$ by $\tilde{\Lambda}^i$.

In the case of our QP problems, we need to compute product form Cholesky factorization of a matrix $M = D+VV^T$, where D is a diagonal matrix in $\mathbf{R^{n\times n}}$, V is a rectangular matrix in $\mathbf{R^{n\times k}}$ (assuming $k << n$), and $VV^T = \sum_{i=1}^k v^i(v^i)^T$, where v^i is the i-th column of V. Thus, we can apply the above Procedure 1 by setting initial Λ to equal D and L to I_n, an $n \times n$ identity matrix.

To store information about the product form Cholesky factorization of $(D + VV^T)$ we need to store the matrix V, the diagonal elements of Λ^k, k vectors p^i and k vectors β^i, $i = 1,\ldots k$. Overall, our memory requirement for the factorization is $3kn$ plus smaller order terms.

To compute the vectors p^i and β^i, $i = 1,\ldots,k$, we need to solve $(k-1)(k-2)/2$ systems of the form $\tilde{L}q = r$ (by $\tilde{L}$, without an index, we mean any matrix of the form of Eq. (3)). Due to the special structure of $\tilde{L}$ such solution can be obtained by the following procedure:

Procedure 2.

Given p, β and r;

1. Set $q := r$ and $\sigma := q_1\beta_1$;

2. For $j = 2,\ldots,n$

Set $q_j := q_j - p_j\sigma$,

$\sigma := \sigma + q_j\beta_j$

This procedure requires $2n$ multiplications (and the same number of additions). Recursions in Fig. 1 require $O(n)$ operations for each $i = 1,\ldots,k$. Hence, overall, the computation of the product from Cholesky factorization of $D + VV^T$ requires k^2n (plus smaller order term) multiplications.

Clearly, solving the system of equations of the form $\tilde{L}^T q = r$, and performing matrix multiplication $q = \tilde{L}r$ or $q = \tilde{L}^T r$ each can be done by a procedure which takes into account the special structure of $\tilde{L}$, similarly to Procedure 2, and requires $O(n)$ operations. Thus, once the factorization of $D + VV^T$ is computed, solving $(D + VV^T)u = w$ requires only $O(kn)$ operations. The overall complexity in terms of the number of multiplications is then k^2n plus smaller order term. This is about twice as much as the complexity of solving this system using the SMW update. However, if $k << n$ this complexity is still a very significant improvement over $O(n^3)$ operations required to solve $(D + VV^T)u = w$ directly.

The product form Cholesky factorization method is numerically more stable than the SMW method both in theory and in practice. First, it is easy to check that it produces a correct result when applied to the small example in the previous subsection. Second, in section 5 we show some examples of QP for which the implementation of an interior point method using the product form Cholesky factorization works well, while the implementation of the same interior point method with SMW update fails numerically. Finally, in Goldfarb and Scheinberg (1999) it is shown that the product form Cholesky factorization is numerically stable when used in the context of interior point method for linear programming (under the assumption that the initial data is well conditioned). This result is supported by extensive computational evidence. The case of QP for SVM is somewhat different from the case of linear programming. The proof of numerical stability does not apply directly, and extending it is tedious and beyond interest of this paper; however, similar intuition applies. Computational experiments show that the product form Cholesky factorization method avoids numerical difficulties even when the initial data is not well conditioned.

4. Approximating the Kernel Matrix

It is often the case that the exact low-rank representation of the kernel matrix $Q = VV^T$ is not given, or even does not exist, i.e. the rank of Q is large. In these cases we may hope that Q can at least be approximated[2] by a low rank matrix $\tilde{Q}$. However, finding a good approximation while keeping its rank low is not trivial. In this section we address the following problem: given a symmetric positive semidefinite matrix, Q, construct a matrix $\tilde{Q}$ such that its rank k and the bound on the error $\Delta Q = Q - \tilde{Q}$ are acceptably small.

Recently, Smola and Schölkopf (2000) suggested a method for approximating the data set in the feature space by a set in a low-dimensional subspace. The authors suggested to select (via random sampling) a small subset of data points to form the basis of the approximating subspace[3] . All other data points are then approximated by linear combinations of the elements of the basis. The basis is build iteratively, each new candidate element is chosen by a greedy method to reduce the bound on the approximation error $\Delta Q = Q - \tilde{Q}$ as much as possible. The authors use the value $\mathrm{tr}(\Delta Q)$ as the bound on the norm of ΔQ, and hence, as a stopping criteria: the algorithm stops when $\mathrm{tr}(\Delta Q)$ is below some tolerance, say ϵ_{tol}.

Let us assume that k points are already in the basis. To compute the reduction of the approximation bound for each of the remaining $n-k$ points extra $O(nk(n-k))$ operations are required. This is clearly too expensive. The authors suggest at each iteration to randomly select N candidates for the basis and apply their greedy approach only to those candidates. Thus, the overall complexity is $O(Nnk^2)$.

Here we present a way of constructing $\tilde{Q}$ by directly approximating the Cholesky factorization of Q. The proposed algorithm has complexity $O(nk^2)$ and takes the full advantage of the greedy approach for the best reduction in the approximation bound $\mathrm{tr}(\Delta Q)$.

Any positive definite matrix Q can be represented by its Cholesky factorization $Q = GG^T$, where G is a lower triangular matrix[4] (see Golub and Van Loan, 1996, Ch. 4). If Q is positive semidefinite and singular, then it is still possible to compute an "incomplete" Cholesky factorization GG^T, where some columns of G are zero. Such procedure requires $O(k^2 n)$ operations, where k is the number of nonzero pivots in the procedure, which is the same as the rank of matrix Q. This procedure can also be applied to *almost* singular matrices by skipping pivots that are below a certain threshold (see Golub and Van Loan, 1996). If the eigenvalues of Q are distinctly separated into a group of very small eigenvalues and relatively large eigenvalues then by choosing an appropriate value of the threshold such procedure will produce both: very small numerical error and a good approximation

2. Indeed in the context of SVM it was already been noted that typically Q has rapidly decaying eigenvalues (Williamson et al., 1998, Smola and Schölkopf, 2000) although there exist kernels for which their eigenspectrum is flat (Oliver et al., 2000).
3. Also see Williams and Seeger (2001) for another random sampling technique for low rank approximation.
4. This is the traditional definition of Cholesky factorization. It relates to the definition presented at Section 3, $Q = L\Lambda L^T$, by setting $G = L\Lambda^{1/2}$.

(see Wright, 1996). If, however, the eigenvalues of Q have a more complicated structure (varying from very small to relatively large), then a symmetric permutation of rows and columns of Q may be necessary during the Cholesky factorization procedure to stabilize the computations and guarantee a good approximation. Symmetric permutations (sometimes referred to as "symmetric pivoting", cf. 4.2.9 of Golub and Van Loan 1996) are also crucial in case when one is trying to find the best possible approximation while keeping the rank of the approximating matrix below a prescribed bound. Figure 2 present an algorithm which computes an approximation $\tilde{Q} = GG^T$ of Q using incomplete Cholesky factorization with symmetric permutations. Note that this procedure generates the *Cholesky triangle* G column by column, while keeping in memory only the diagonal elements of Q. All other entries of Q are needed only once and thus can be computed on demand. Hence it is easy to derive a "kernelized" version of the suggested procedure, assuming an access to the input vectors and the kernel function.

for $i = 1 : n$
  for $j = i : n$
    $G_{jj} := Q_{jj}$
    for $k = 1 : i-1$
      $G_{jj} := G_{jj} - G_{jk}G_{jk}$
    end
  end
  if $\sum_{j=i}^{n} G_{jj} > \epsilon_{tol}$
    find $j^* : \; G_{j^*j^*} = \max_{j=i:n} G_{jj}$
    $G_{i:n,i} := Q_{j^*:n,j^*}$
    $G_{i,1:i} \leftrightarrow G_{j^*,1:i}$
    for $j = 1 : i-1$
      $G_{i+1:n,i} := G_{i+1:n,i} - G_{i+1:n,j}G_{i,j}$
    end
    $G_{ii} := \sqrt{G_{ii}}$
    $G_{i+1:n,i} := G_{i+1:n,i}/G_{ii}$
  else
    $k := i-1$
    Stop
  endif
end

Figure 2: Column-wise Cholesky Factorization with Symmetric Pivoting

The computational complexity of this procedure is $O(nk^2)$, since the only extra work is in computing diagonal elements G_{jj}, which requires $O(nk)$ operations at each iteration. The storage requirement is $O(nk)$. Symmetric permutations provide a greedy way of building the approximation of Q. After i iterations of the Cholesky factorization procedure the matrix $G^i = G_{1:n,1:i}$ is such that $G^i(G^i)^T = \tilde{Q}^i$, where $\tilde{Q}^i$ is an approximation of Q subject to a symmetric permutation of rows and columns. For simplicity, let us assume that the rows and columns of Q are ordered "correctly" and no permutation is necessary (this assumption does not affect our analysis). Let $\Delta Q^i = Q - \tilde{Q}^i$. Let $G^i_{1:i}$ be the matrix composed of the first i rows of G^i and $G^i_{i+1:n}$ in the matrix composed of the last $n-i$ columns. Then

$$\Delta Q^i = \begin{bmatrix} 0 & 0 \\ 0 & Q_{i+1:n,i+1:n} - G^i_{i+1:n}(G^i_{1:i})^{-1}Q_{1:i,i+1:n} \end{bmatrix}.$$

From the properties of Cholesky factorization, ΔQ^i is positive semidefinite and it is easy to see that after updating G_{jj} for all $j = i+1, \ldots, n$ at the i-th iteration, $\sum_{j=i}^n G_{jj} = \mathrm{tr}\,\Delta Q^i$. Thus, when the above algorithm stops (after k iteration), we have $\mathrm{tr}\,\Delta Q \leq \epsilon_{tol}$.

4.1 Bound on the error in the optimal objective value

Consider the perturbed optimization problem $(\tilde{P})$, which is constructed by replacing the kernel matrix Q by a low-rank approximation $\tilde{Q}$, i.e.

$$(\tilde{P}) \qquad \begin{array}{ll} \min_x & \frac{1}{2}x^T\tilde{Q}x - e^T x \\ \text{s.t.} & a^T x = 0, \\ & 0 \leq x \leq c. \end{array}$$

A natural question to ask is: How close is the solution of the perturbed problem to the solution of the original problem.

Let x^* denote an optimal solution of the original problem (P) and let $\tilde{x}^*$ denote an optimal solution of the perturbed problem $(\tilde{P})$. Also let f denote the objective function of (P), and $\tilde{f}$ be the objective function of $\tilde{P}$. We would like to estimate $|f(x^*) - \tilde{f}(\tilde{x}^*)|$. The feasible sets of P and $\tilde{P}$ are the same, hence a feasible solution of one problem is feasible for the other.

From optimality of $\tilde{x}^*$ $\tilde{f}(\tilde{x}^*) \leq \tilde{f}(x^*) = f(x^*) + \frac{1}{2}(x^*)^T\Delta Q x^*$. We know that $\Delta Q = Q - \tilde{Q}$ is positive semidefinite, thus $\tilde{f}(\tilde{x}^*) \leq \tilde{f}(x^*) \leq f(x^*)$; i.e. optimal objective value of the perturbed problem is always smaller than the optimal objective value of the original problem[5].

On the other hand, optimality of x^* $f(x^*) \leq f(\tilde{x}^*)$, hence $f(x^*) - \tilde{f}(\tilde{x}^*) \leq f(\tilde{x}^*) - \tilde{f}(\tilde{x}^*) = \frac{1}{2}(\tilde{x}^*)^T\Delta Q\tilde{x}^* \leq \frac{1}{2}\lambda_1(\Delta Q)||\tilde{x}^*||^2 \leq \frac{1}{2}\mathrm{tr}\,(Q - \tilde{Q})||\tilde{x}^*||^2$. Let l be the number of positive components of $\tilde{x}^*$. From the feasibility constraints $0 \leq x \leq c$ these components are bounded from above by c. Let ΔQ_l be the principal sub-matrix of ΔQ that corresponds to

5. Recall that our definition of the objective function is the negative of the typical objective function definition used in SVM literature. Thus, for the more traditional objective function its optimal value for the original problem is always *smaller* than the optimal value for the perturbed problem.

positive components of $\tilde{x}^*$. The above yields the bound

$$0 \leq f(x^*) - \tilde{f}(\tilde{x}^*) \leq \frac{1}{2} x(0)^T \Delta Q x(0) \leq \frac{c^2 l \epsilon}{2}.$$

Hence we have the following theorem:

Theorem 1 *If* $(Q - \tilde{Q})$ *is positive semidefinite and* $\mathrm{tr}\,(Q - \tilde{Q}) \leq \epsilon$ *then the optimal objective value of the original problem is larger than the optimal objective value of the perturbed problem and their difference is bounded by* $c^2 l \epsilon / 2$*, where* l *is the number of active constraints (support vectors) in the perturbed problem.*

The above error bound does not require approximation matrix $\tilde{Q}$ to be obtained by Cholesky factorization with permutations. Theorem 1 holds for any $\tilde{Q}$ such that $Q - \tilde{Q}$ is positive semidefinite and it's trace is bounded by ϵ. If $Q - \tilde{Q}$ is not positive semidefinite, then it is still possible to provide a bound on the change in the optimal value function through a bound on the norm of ΔQ (see Fine and Scheinberg 2001 for more details).

5. Experiments

In this section we compare performances of the suggested method to state-of-the-art SVM training procedures, construct a simple toy example to demonstrate the numerical instability of the Sherman-Morrison-Woodbury update, and examined the impact of approximating the SVM solution from both the optimization problem and the classification problem perspectives.

5.1 Cholesky Product Form QP vs. SMO

We've modeled 150 different Speaker Id binary problems[6], in the following way: For each speaker we trained a mixture of 4 Gaussians, 25 dimensions[7], diagonal covariance. We then applied Fisher kernel methodology (Jaakkola and Haussler, 1999), which resulted in transforming the original data vectors to a 204 dimensional space. We further applied linear normalization transformation to each dimension to balance the transformed vectors. This caused the transformed vectors to be fully dense. The size of the training set in a typical problem ranges from 6500 to 7000 vectors, roughly equally divided to positive and negative example. We chose a dot-product kernel and compared our technique with the SMO algorithm to find a maximal margin linear separator. Our choice was motivated by former evidences which support the claim that most of the work in separating the positive and negative points is carried out through the transformation represented by the Fisher kernel. In the current setting the choice of a dot-product kernel served another purpose - to obtain results of the SMO at the peek of its performance. To this end we used a special designed variant of SMO which takes advantage of the fact that the kernel operations are just dot-products (for further discussion see Platt, 1999). The comparison in CPU time (depicted at Fig. 3) clearly favors our method. Note also that we have gained roughly a

6. This is part of a much larger multi-class Speaker ID system (Fine, Navrátil, and Gopinath, 2001).
7. 13 dimension cepstrum, augmented with first derivatives ('delta') and finally discarding the energy coefficient.

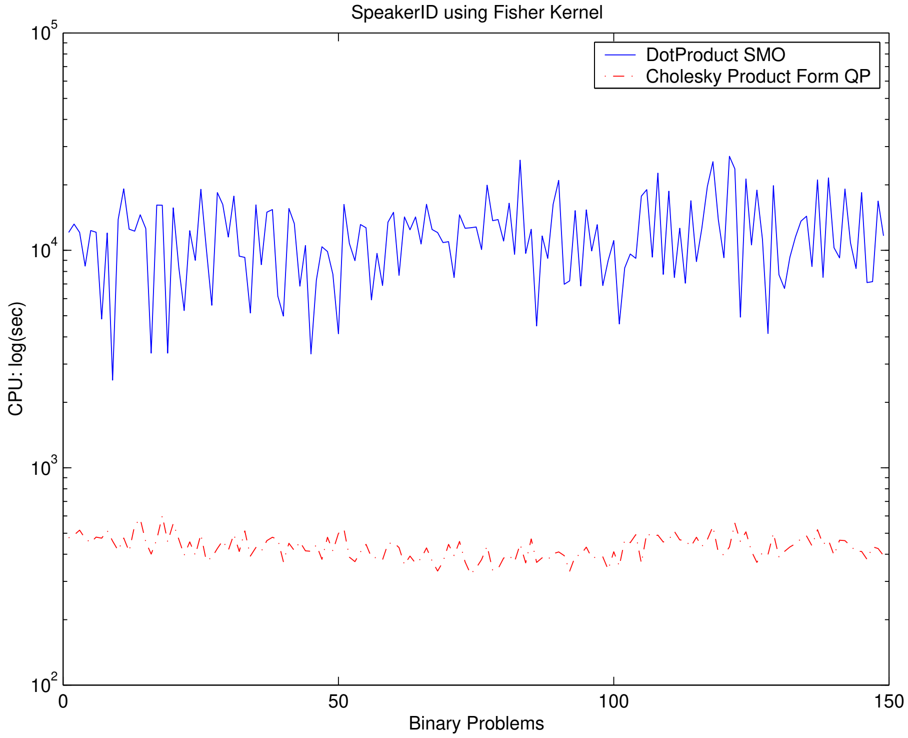

Figure 3: Cholesky Product Form QP vs. SMO

factor of 1100 in computational complexity and a factor of 33 in the storage requirements over traditional IPMs.

5.2 Cholesky Product Form QP vs. SVM^{light}

We next turn to compare performances with Joachims's SVM^{light} package, which is an active set method applying the "shrinking" approach to select the subproblem to be solved at each iteration (Joachims, 1999). As the QP subproblem solver we used *loqo*, which is Smola's implementation of an IPM solver provided with the SVM^{light} package. We examined performances on a moderate size problem, the Abalone dataset from the UCI Repository (Blake and Merz, 1998). Since, at this point, we were not interested in evaluating generalization performances, we used the whole set (of size 4177) for training. The gender encoding (male/female/infant) was mapped into $\{(1,0,0),(0,1,0),(0,0,1)\}$. Then data was rescaled to lie in the [-1,1] interval. Similarly to the SMO experiment, we restricted ourselves to dot-product kernel, while controlling the difficulty of the problem with the choice of the penalty (cost) term c. We examined performances for 3 levels of difficulty, while gradually increasing the allowed size of the subproblems the SVM^{light} algorithm. Note that there's no similar tuning to be done for our method since the whole QP problem fits into memory. The results are depicted at Figure 4. This figure highlights the fact that SVM^{light} is quite sensitive to the choice of the subproblem (chunk) size, as well as the difficulty of the overall

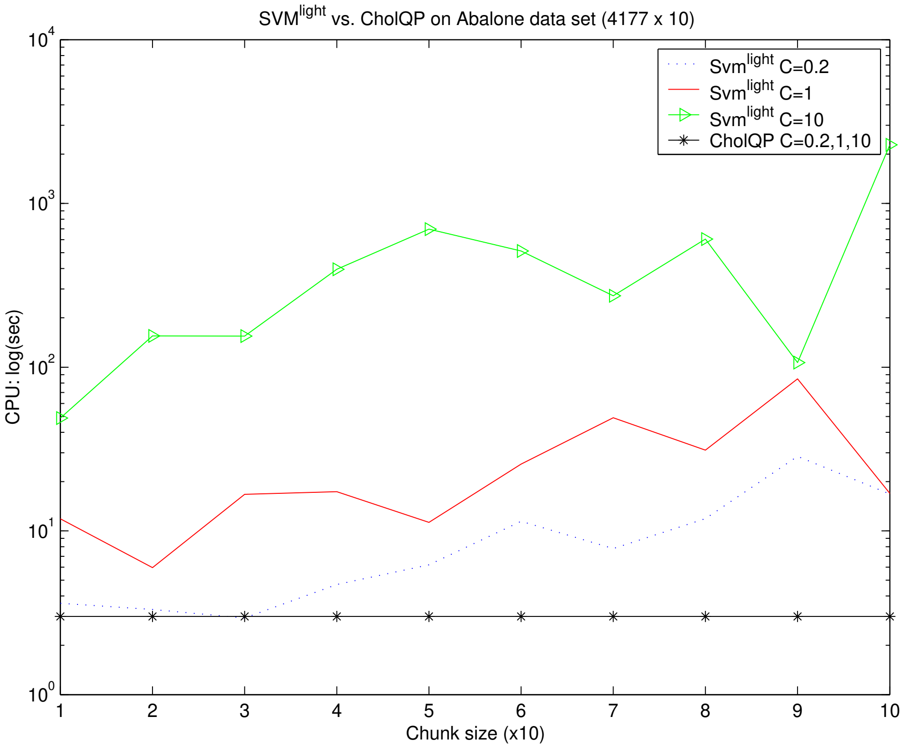

Figure 4: Cholesky Product Form QP vs. SVM^{light}

problem, while our method (denoted "CholQP" it the figure) is stable for all levels of difficulty.

5.3 Example of failure of the Sherman-Morrison-Woodbury update

We constructed a small example in the spirit of the one described in Section 3 to demonstrate possible numerical instability of the Sherman-Morrison-Woodbury update. Figure 5 illustrates the data in the example. There are two crucial features: the set of active support vectors is redundant (so the problem is degenerate) and the support vectors are scaled by a large number (10^4). On this problem an interior point method using SMW update failed to converge after achieving only 2 digits of accuracy. Whereas the same IPM with the product form Cholesky update converged to 8 digits of accuracy.

To demonstrate that such failures can happen in practice we applied the Sherman-Morrison-Woodbury version of the code to an approximate problem arising from Abalone data set using polynomial kernel (as described in the next subsection). The algorithm stalled after achieving only 5 digits of accuracy, whereas the product form Cholesky version converged to 12 digits of accuracy.

5.4 Incomplete Cholesky Factorization (ICF)

To demonstrate the utility of the proposed ICF method and the impact of its approximation on the SVM solution, we conducted an experiment similar to the one described

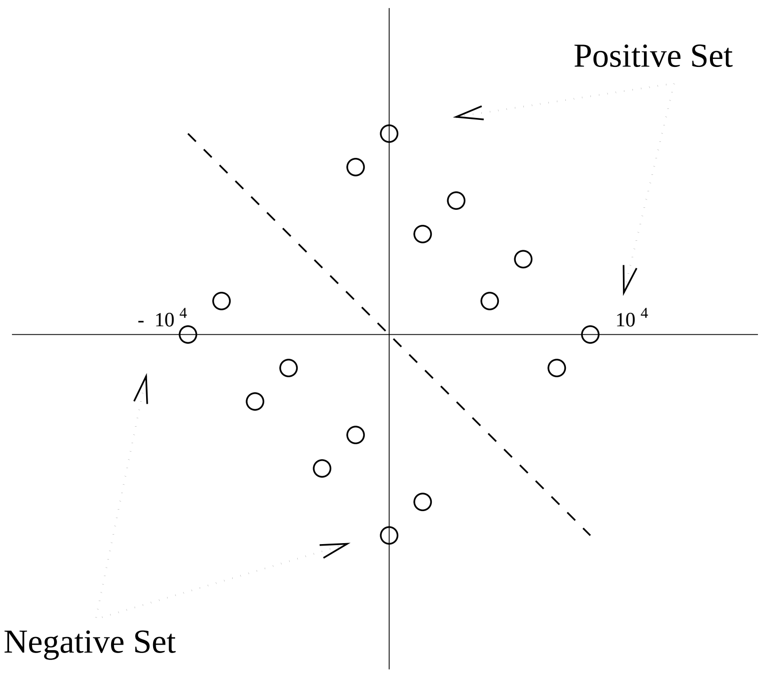

Figure 5: Example for the failure of SMW update

at (Smola and Schölkopf, 2000) on the Abalone data set[8] using polynomial kernels , i.e. $k(x,y) = (\langle x,y \rangle + const)^d$. Thus we actually used the "kernelized" version of the factorization algorithm (cf. section 4). We examined the resulted factorization from two stand points: Figure 6 demonstrates the impact of the low-rank approximation on the solution of the optimization problem as a function of the rank, i.e. the optimal value of the objective function[9] and the norm of the separating hyperplane. Figure 7 shows the impact of the low-rank approximation from the classification performances perspective, i.e. training and testing errors. Both figures are scaled with a graph which measures the quality of the approximation in Frobenius norm (the square root of the sum of squares of the matrix elements) with respect to the original kernel matrix.

6. Concluding Remarks

Our experiments show that an IPM (with the proposed approach to linear algebra) can be more efficient than other state-of-the-art methods for QP problems arising in the course of training support vector machines, if the kernel matrix has a low rank (compared to the size of the data) or if it can be approximated by a low-rank positive semidefinite matrix. We considered two possible ways of handling linear algebra. We showed that Sherman-Morrison-Woodbury update may be numerically unstable. However, if accuracy is not a

8. The data set was preprocessed as described in section 5.2, and we used the first 3000 points for training and the remaining 1177 points for testing.
9. This is actually a "negative" plot of the optimal values (due to our definition of the objective function) to ease the comparison with similar plots published in the SVM literature.

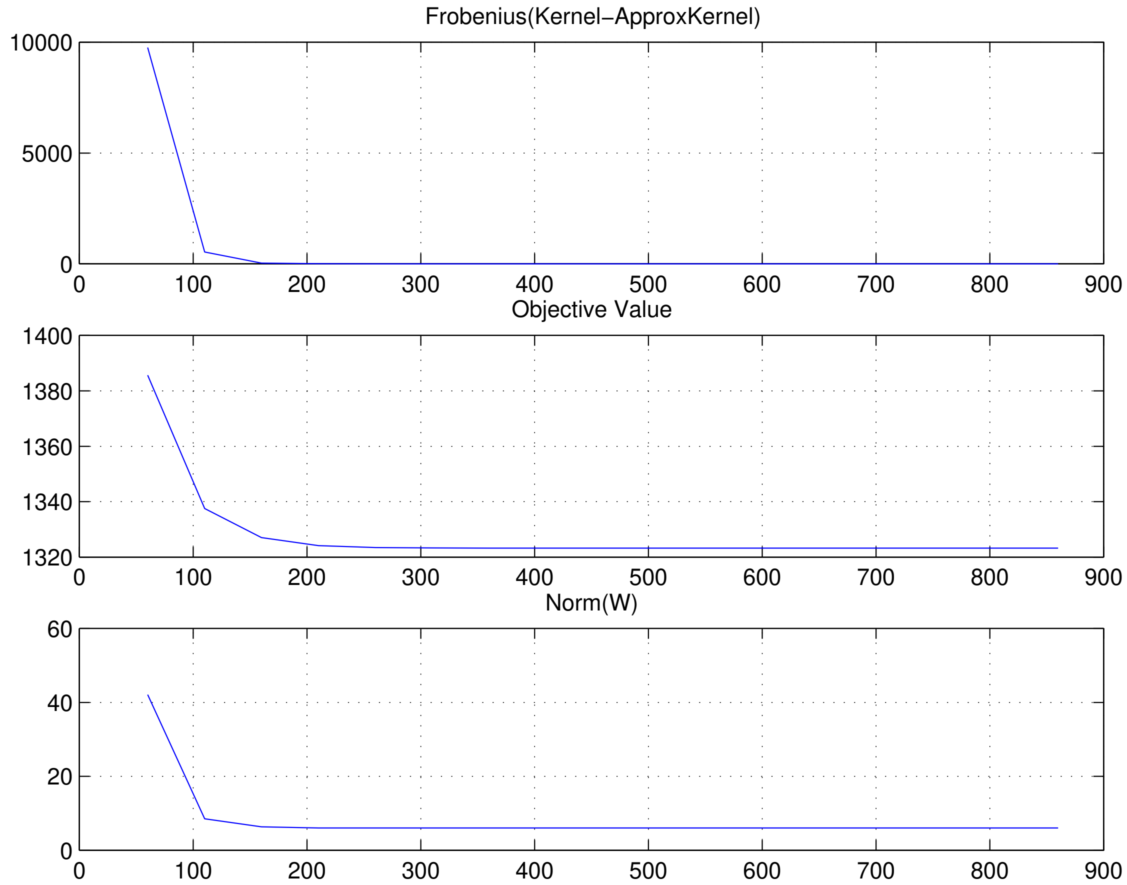

Figure 6: Incomplete Cholesky Factorization for Poly Kernel (input dim = 10, poly degree = 5) on the Abalone data set - **The optimization problem perspective:** Optimal value of the objective function and norm of the separating hyperplane. X axis is the rank (note that the feature space dim ≈ 3000).

high priority one might prefer to use the SMW method over the product form Cholesky factorization, since it is faster and requires less storage.

For massive data sets which do not fit the memory restrictions, an IPM may still be the most efficient approach to solve smaller subproblems. This motivates an effort to embed our approximation technique and QP solver in a Chunking/Shrinking meta algorithm. Such a scheme is expected to enhance performance in terms of storage requirement and computational complexity and, thus, it will enable to efficiently handle larger chunks.

Finally, if the approximation is not too rough, then the set of active constraints (Support Vectors) may be identical to the set which correspond to the original optimization problem, though their values should obviously differ (due to the approximation). Hence, one may use the approximation problem just to identify the set of active constraints (assuming this set is not too large), and then resolve the reduced problem which constructed solely by the corresponding original (non approximated) SV, and thus obtain the optimal solution.

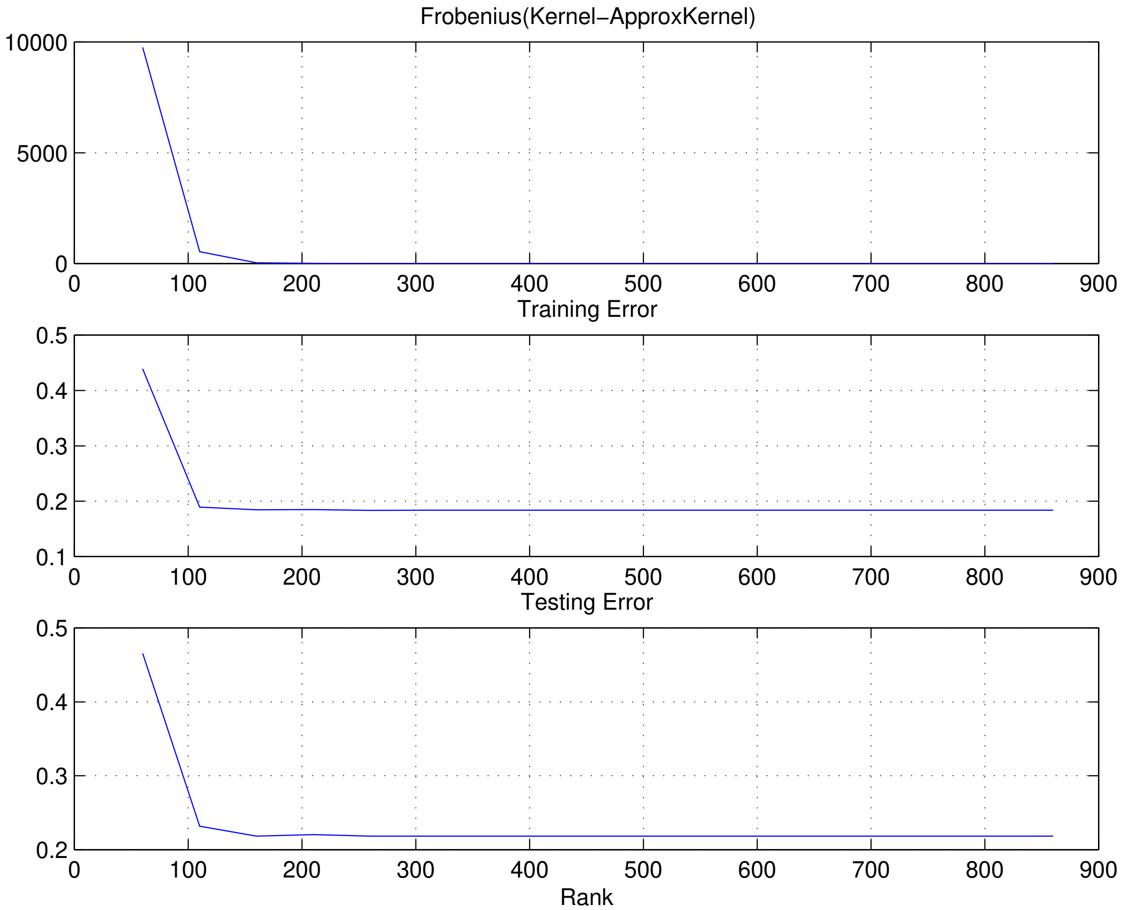

Figure 7: Incomplete Cholesky Factorization for Poly Kernel (input dim = 10, poly degree = 5) on the Abalone data set - **the classification problem perspective:** Training and testing errors. X axis is the rank (note that the feature space dim ≈ 3000).

Working out a bound for the approximation error that will ensure the identification of the active set is a subject for future research.

Acknowledgments

The authors wishes to thank Ramesh Gopinath for pointing out a shorter proof of Theorem 1, and to the anonymous referees for their useful comments and suggestions.

References

K. D. Andersen. A modifies schur complement method for handling dense columns in interior point methods for linear programming. *ACM Transactions on Mathematical Software*, 22 (3):348–356, 1996.

J. M. Bennet. Triangular factors of modified matrices. *Numerisches Mathematik*, 7:217–221, 1965.

C. L. Blake and C. J Merz. UCI repository of machine learning databases, 1998.

B. Boser, I. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

I. C. Choi, C. L. Monma, and D. F. Shanno. Further development of primal-dual interior point methods. *ORSA J. on Computing*, 2(4):304–311, 1990.

S. Fine, J. Navrátil, and R. A. Gopinath. A hubrid gmm/svm approach to speaker identification. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

S. Fine and K. Scheinberg. Efficient application of interior point methods for quadratic problems arising in support vector machines using low-rank kernel representation. Submitted to Mathematical Programming, 2001.

R. Fletcher and M. J. D. Powell. On the modification of ldl^T factorization. *Mathematics of Computation*, 28(128):1067–1087, 1974.

T. T. Friess, N. Cristianini, and C. Campbell. The kernel-adaraton algorithm: A fast simple learning procedure for support vector machines. In *Proceedings of the 15th International Conference on Machine Learning*, pages 188–196. Morgan Kaufman, 1998.

Ph. E. Gill, W. Murray, and M. A. Saunders. Methods for computing and modifying the ldv factors of a matrix. *Mathematics of Computation*, 29:1051–10, 1975.

D. Goldfarb and K. Scheinberg. A product-form cholesky factorization method for handling dense columns in interior point methods for linear programming. Submitted, 1999.

G. H. Golub and Ch. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore and London, 3 edition, 1996.

T. S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11. MIT Press, 1999.

T. Joachims. Making large-scale support vector machine learning practical. In B. Schölkopf, C. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, chapter 12, pages 169–184. MIT Press, 1999.

A. Marxen. Primal barrier methods for linear programming. Technical report, Dept. of Operations Research, Stanford University, Stanford, CA, 1989.

S. Mehrotra. On implementation of a primal-dual interior point method. *SIAM J. on Optimization*, 2(4):575–601, 1992.

N. Oliver, B. Schölkopf, and A. J. Smola. Natural regularization from generative models. In A. J. Smola, B. Schölkopf, P. L. Bartlett, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, chapter 4, pages 51–60. MIT Press, 2000.

E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for support vector machines. In *Proceedings of the IEEE Neural Networks for signal Processing VII Workshop*, pages 276–285. IEEE, 1997.

J. C. Platt. Fast trining support vector machines using sequential mininal optimization. In B. Schölkopf, C. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods*, chapter 12, pages 185–208. MIT Press, 1999.

K. Scheinberg and S. Wright. A note on modified cholesky and schur complement in interior point methods for linear programming. Manuscript, 2000.

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, Stanford University, CA, 2000. Morgan Kaufmann Publishers.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.

C. Williams and M. Seeger. Using the nyström method to speed up kernel machines. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

R. C. Williamson, A. J. Smola, and B. Schölkopf. Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. NeuroCOLT NC-TR-98-019, Royal Holloway College, University of London, UK, 1998.

S. Wright. Modified cholesky factorizations in interior point algorithms for linear programming. Preprint anl/mcs-p600-0596, Argonne National Laboratory, Argonne, IL, 1996.

S. Wright. *Primal-Dual Interior Point Methods*. SIAM, Philadelphia, 1997.