RC23773 (W0511-025) November 4, 2005
Mathematics

IBM Research Report

Solving Structured Convex Quadratic Programs by Interior Point Methods with Application to Support Vector Machines and Portfolio Optimization

D. Goldfarb
Department of IEOR
Columbia University
New York, NY

K. Scheinberg
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Solving Structured Convex Quadratic Programs by Interior Point Methods with Application to Support Vector Machines and Portfolio Optimization[*]

D. Goldfarb
Columbia University, Dept. of IEOR, New York, NY
K. Scheinberg
IBM T.J. Watson Research Center, Yorktown Heights, NY

October 19, 2005

1 Introduction

In this paper we consider the application of an interior point method (IPM) to the convex quadratic programming (QP) problem with upper bounds on the variables:

$$(P) \qquad \begin{array}{ll} \min & \frac{1}{2}x^T Qx + c^T x \\ \text{s.t.} & Ax = b, \\ & 0 \le x \le u, \end{array}$$

where the symmetric positive semidefinite matrix $Q \in \mathbf{R}^{n\times n}$, the rank m matrix $A \in \mathbf{R}^{m\times n}$, $c \in \mathbf{R}^n$, $u \in \mathbf{R}^n$ and $b \in \mathbf{R}^m$ are the data for the problem, and $x \in \mathbf{R}^n$ is the vector of variables. Three aspects of this topic are explored.

The first concerns a property of the Cholesky factorization of the normal equation matrix that arises at each iteration of an IPM applied to problem (P). In [11] it was shown that when linear programming (LP) problems are solved by an IPM, the unit lower triangular matrix L in the Cholesky factorization $L\Lambda L^T$ of the normal equations matrix remains uniformly bounded (in norm) as the iterates converge to the solution. This result holds even if the condition number of the normal equations matrix becomes unbounded. In [12] this result was extended to the case of second-order cone (SOCP) programming under the assumption that the IPM iterates stay in a neighborhood of the central path. Here we extend the LP result (without any assumptions) to the case of convex QPs.

[*]Research supported in part by NSF Grant DMS 01-04282, ONR Grant N00014-03-0514 and DOE Grant GE-FG01-92ER-25126

The second part of the paper presents efficient methods for implementing the linear algebra at each IPM iteration to exploit structure in the Hessian matrix Q. IPMs have an advantage over active set methods in that they usually obtain an (approximate) solution to a problem after a small number of iterations (often less than 50). However the cost of each iteration is significantly higher than that of an active set method. The major part of this cost comes from the need to solve at each iteration two symmetric positive definite systems of equations $(D^2 + Q)u_1 = w_1$ and $A(D^2 + Q)^{-1}A^T u_2 = w_2$, one n-by-n and the other m by m (D^2 is a diagonal matrix). If n is large, then at least one of these systems is large-scaled. When the matrices Q and A are sparse, solving each system is relatively inexpensive and IPM iterations are efficient. The problem arises when one of the systems is dense. This can happen either because Q or Q^{-1} is dense or because A contains dense columns. The latter case is similar to the case of the LP problems with dense columns. In [11] this was extensively studied and an efficient method for handling dense columns was proposed.

There are a number of applications of convex quadratic programming problems where the density of the systems that are solved comes from Q, and not from A. Often, however, the matrix Q can be represented as the sum of a diagonal matrix and a low rank dense matrix. We will show how one can use this special structure of Q to compute the factorization of the matrices D^2+Q and $A(D^2+Q)^{-1}A^T$ efficiently, effectively extending the results in [11] to such QPs..

There are also cases where the matrix Q is not a low rank matrix, but can be approximated by such a matrix. In this case one might choose to solve the approximate problem, since it can be solved much faster. We give a bound on the error in the objective function when such an approximation is used. We also discuss the possibility of using an active set method to "correct" the approximate solution.

Finally, the last part of the paper addresses two applications where Q has the special structure described above. One such application is in an area of machine learning called support vector machines [?], ... Support vector machines (SVM) is an approach for solving classification problems, whose essence lies in solving a convex quadratic programming problem. The problem (in its dual form) has bound constraints and only one equality constraint. The dimension is the same as the size of the data that has already been classified. Clearly being able to solve large instances results in better classification results. However, the matrix Q is a totally dense matrix, and it becomes prohibitively expensive to solve problems with $n > 1000$ by IPMs without using special linear algebraic techniques. As we will explain in Section 6, even though Q is fully dense, it often has low rank. Therefore, the methods described in Section 4 apply naturally in the SVM setting.

Another application of convex QP where the Hessian is specially structured is portfolio optimization. Portfolio optimization involves selecting a portfolio of assets so as to maximize the return of the portfolio while minimizing the risk. For standard ways of measuring return and risk, several models for treating these conflicting objectives give rise to convex QPs. Structured Hessians of the form considered in Section 4 result from the assumption that the individual asset returns are primarily driven by underlying market factors.

The paper is organized as follows. In Section 2 we discuss interior point methods for the QP problem (P) and two approaches for computing the step direction at each iteration: one based on solving the so-called "augmented system" of linear equations and one based on the "normal equations". In Section 3 we extend the stability results of [11] to the Cholesky factors of the normal equations matrices for the QP case. In Section 4 we introduce methods for solving these systems via low-rank updates. Section 5 contains a discussion of approximating the Hessian matrix Q in problem (P) by a low rank matrix and derives a bound on the error in the optimal value that results from such an approximation. Sections 6 and 7 contain descriptions of two applications of the results in the previous sections, support vectors machines and portfolio optimization, respectively.

2 Interior Point Methods for Convex QPs.

The Karush-Kuhn-Tucker (KKT) necessary and sufficient optimality conditions for Problem (P) are:

$$\begin{aligned}
&A^T y - Qx + s - \xi = c,\\
&Ax = b,\\
&Xs = 0,\\
&(U - X)\xi = 0,\\
&0 \le x \le u,\ \xi \ge 0,\ s \ge 0.
\end{aligned}$$

Here, $y \in \mathbf{R}^m$ and $\xi \in \mathbf{R}^n$ are the vectors of dual variables corresponding to the equality constraints and upper bound constraints in (P), respectively, and $s \in \mathbf{R}^n$ is the vector of dual slack variables. By X, S, U and Ξ we denote the diagonal matrices whose diagonal entries are the elements of the vectors x, s, u and ξ, respectively. We shall use I to denote the identity matrix and e to denote a vector of all ones.

The solutions of the perturbed KKT optimality conditions below define the so-called central path $\{x(\mu), y(\mu), s(\mu)\}$:

$$(CP_\mu) \qquad \begin{aligned}
&A^T y - Qx + s - \xi = c,\\
&Ax = b,\\
&Xs = \mu e,\\
&(U - X)\xi = \mu e,\\
&0 \le x \le u,\ \xi \ge 0,\ s \ge 0.
\end{aligned}$$

These equations have a unique solution for any positive value of the parameter μ.

An iteration of a typical primal-dual interior point method consists of a Newton step towards the solution of the system of nonlinear equations (CP_μ) for a particular value of μ. Thus, one has to compute $(\Delta x, \Delta y, \Delta s)$, satisfying

$$\begin{bmatrix} -Q & A^T & I & -I \\ A & 0 & 0 & 0 \\ S & 0 & X & 0 \\ -\Xi & 0 & 0 & U-X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \\ \Delta \xi \end{bmatrix} \begin{bmatrix} -r_c \\ -r_b \\ -r_{xs} \\ -r_\xi \end{bmatrix}, \tag{1}$$

where $r_b = Ax - b$, $r_c = A^T y - Qx + s - c$, $r_{xs} = Xs - \mu e$ and $r_\xi = (U - X)\xi - \mu e$. In “feasible” IPMs r_b and r_c are zero.

By eliminating Δs and $\Delta \xi$ from the above system we obtain a linear system, usually referred to as the *augmented system*, whose coefficient matrix is

$$T \equiv \begin{bmatrix} -(Q + D^2) & A^T \\ A & 0 \end{bmatrix}, \tag{2}$$

where $D^2 = X^{-1}S + (U - X)^{-1}\Xi$. In the *normal equations* approach to solving this system, one reduces it further by eliminating the Δx variables (forming the Schur complement of $Q + D^2$ in the matrix (2)) to yield a system of equations with a symmetric positive definite coefficient matrix

$$M \equiv A(Q + D^2)^{-1}A^T, \tag{3}$$

that can be solved by forming its Cholesky factorization.

In purely primal and purely dual interior point methods, one has to solve similar systems; in these cases the diagonal matrix D^2 equals $\mu(X^{-2} + (U - X)^{-2})$ and $\frac{1}{\mu}(S^2 + \Xi^2)$, respectively.

3 Uniform Boundedness of the Cholesky Factor L

We study in this section a new uniform boundedness property of the Cholesky factorization $L\Lambda L^T$ of $M \equiv AF^{-1}A^T$, where $F \equiv Q + D^2$. This property is crucial for proving that the product-form Cholesky factorization of M introduced in Section 4 below is numerically stable.

In [11] the following theorem is proved.

Theorem 3.1 *Let A be a fixed $m \times n$ matrix, D be a positive diagonal matrix whose diagonal elements depend on μ and $L\Lambda L^T$ be the Cholesky factorization of $M = AD^2A^T$. Then the elements of L are uniformly bounded as $\mu \to 0$.*

One can restate this theorem in the following more compact way.

$$\sup\{\|L\| : \ L\Lambda L^T = AD^2A^T, \forall D\} < \infty.$$

First we note that the result trivially extends to the Cholesky factorization of $F = Q + D^2$, where Q is a fixed positive semidefinite matrix,

$$\sup\{\|L\| : \ L\Lambda L^T = Q + D^2, \forall D\} < \infty. \tag{4}$$

This follows from the fact that

$$F \equiv Q + D^2 = VV^T + D^2 = [I, V] \begin{bmatrix} D^2 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I \\ V^T \end{bmatrix}. \tag{5}$$

We now extend this result to the Cholesky factorization of $M \equiv AF^{-1}A^T$; i.e., we show below that

$$\sup\{\|L\| : \; L\Lambda L^T = A(Q + D^2)^{-1}A^T, \forall D\} < \infty.$$

To do this, we will show how L can be represented in terms of another lower triangular matrix with a unit diagonal, $\hat{L}$, which, in turn, is a Cholesky factor of a matrix of the form $\hat{A}\hat{D}\hat{A}^T$, where $\hat{A}$ is a fixed matrix and $\hat{D}$ is a diagonal matrix depending on μ. Then we will apply Theorem 3.1 to show that the elements of $\hat{L}$ are uniformly bounded as $\mu \to 0$ and conclude that so are the elements of L.

Let $E \in \mathbf{R}^{n-m\times n}$ be any matrix such that the matrix $K^T \equiv [A^T, E^T]$ is nonsingular and let $\bar{M} \equiv KF^{-1}K^T$. For example, if $A = [B, N]$ where B is a nonsingular $m \times m$ matrix, we can let $E = [0, I]$, which gives

$$K \equiv \begin{bmatrix} B & N \\ 0 & I \end{bmatrix}. \tag{6}$$

Also, let

$$P \equiv \begin{bmatrix} & & & 1 \\ & & \cdot & \\ & \cdot & & \\ \cdot & & & \\ 1 & & & \end{bmatrix} \tag{7}$$

be the "reverse order" permutation matrix and let $\hat{L}\hat{\Lambda}\hat{L}^T$ be the Cholesky factorization of $P\bar{M}^{-1}P$. Then

$$\bar{M} = P\hat{L}^{-T}\hat{\Lambda}^{-1}\hat{L}^{-1}P = P\hat{L}^{-T}PP\hat{\Lambda}^{-1}PP\hat{L}^{-1}P = \bar{L}\bar{\Lambda}\bar{L}^T \tag{8}$$

is the Cholesky factorization of $\bar{M}$, where $\bar{L} \equiv P\hat{L}^{-T}P$ and $\bar{\Lambda} \equiv P\hat{\Lambda}^{-1}P$. Since $M = [I, 0]\bar{M}[I, 0]^T$, the Cholesky factorization of M is

$$M = L\Lambda L^T, \quad \text{where} \quad \bar{L} = \begin{bmatrix} L & 0 \\ L' & L'' \end{bmatrix} \quad \text{and} \quad \bar{\Lambda} = \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda'' \end{bmatrix}. \tag{9}$$

Furthermore, if we let $H \equiv PK^{-T}$, then it follows from (5) that

$$P\bar{M}^{-1}P = H(VV^T + D^2)H^T = [H, HV] \begin{bmatrix} D^2 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} H^T \\ V^T H^T \end{bmatrix}.$$

Now, after defining $\hat{A} \equiv [H, HV]$ and $\hat{D} \equiv \begin{bmatrix} D^2 & 0 \\ 0 & I \end{bmatrix}$, we can apply Theorem 3.1 to the Cholesky factorization $\hat{L}\hat{\Lambda}\hat{L}^T$ of $P\bar{M}^{-1}P \equiv \hat{A}\hat{D}\hat{A}^T$ and conclude that the elements of $\hat{L}$ are uniformly bounded for all D. In [11]) the following lemma is proved.

Lemma 3.2 *If L is a lower triangular matrix with ones on the diagonal and subdiagonal elements that depend on a parameter μ and are uniformly bounded as $\mu \to 0$, then the elements of L^{-1} are also uniformly bounded as $\mu \to 0$.*

By this lemma the elements of $\bar{L} = P\hat{L}^{-T}P$ are uniformly bounded for all D, and from (9) so are those of L. Hence, we have the following theorem for QP.

Theorem 3.3 *Let A be a fixed $m \times n$ matrix, Q be a fixed $n \times n$ symmetric positive semidefinite matrix, D be a positive diagonal matrix whose diagonal elements depend on μ and $L\Lambda L^T$ be the Cholesky factorization of $M = A(Q + D^2)^{-1}A^T$. Then the elements of L are uniformly bounded as $\mu \to 0$.*

4 Low rank updates

Solving the normal equation is the most computationally intensive part of each interior point iteration. There are two potentially expensive steps: one is the factorization of the matrix $Q + D^2$ and the other is the factorization of $M = A(D^2 + Q)^{-1}A^T$. If m, the number of linear constraints, is small then it is inexpensive to factorize M. If m is large, then often A is sparse. If Q is also sparse, solving the normal equations involves the factorization of two sparse matrices. However, if Q is dense, then both matrices $Q + D^2$ and M are dense. The factorization of these matrices then takes $O(n^3)$ and $O(m^3)$ operations, respectively. It also takes $O(mn^2)$ operations to form M. As we will show later in the paper, there are number of applications where Q is large scale and completely dense. However, often in these applications Q has low rank; that is, it can be represented as VV^T, where $V \in \mathbf{R}^{n\times k}$, and $k \ll n$. In this case $M = A(VV^T + D^2)^{-1}A^T$ also has a special structure; for instance, if A is sparse it can be represented as a sum of a sparse matrix and a low rank dense matrix (see the next subsection for details). Note that if Q can be represented as a sum of a low rank matrix and a diagonal matrix, then the special structure of the normal equation is essentially the same as when Q is a low rank matrix. It is important to take advantage of this special structure of Q to decrease the computational workload of each interior point iteration. In this section we consider several alternative methods for exploiting this structure.

We first consider two approaches that efficiently solve the systems $(D^2 + VV^T)u_1 = w_1$ and $A(D^2 + VV^T)^{-1}A^T u_2 = w_2$ by taking into consideration the special form of the matrix $(D^2 + VV^T$ and sparsity in A. Both of these methods when applied to the system $(D^2 + VV^T)u_1 = w_1$ require $O(nk)$ storage and $O(nk^2)$ arithmetic operations, where k is the number of columns in V. Thus, if $k << n$ the two methods provide significant savings in terms of workload and storage. The first method is somewhat less expensive than the second. Its workload is about half of that for the second method and its storage requirement is about one third of that for the second method. On the other hand, the first method may experience numerical instability.

A comparison of the the workloads for the two methods in their application to the system $A(D^2 + VV^T)^{-1}A^T u_2 = w_2$ is more complicated and is discussed at the end of Subsection 4.3. In Subsection **??**, we consider two approaches for solving the augmented system.

4.1 Sherman-Morrison-Woodbury update

We first consider the Sherman-Morrison-Woodbury (SMW) formula for updating the inverse of a matrix. Using this update in an interior point algorithm for solving linear SVM problems was recently proposed in [7]. Here we first apply it to the matrix $D^2 + VV^T$; specifically

$$(D^2 + VV^T)^{-1} = D^{-2} - D^{-2}V(I + V^T D^{-2}V)^{-1}V^T D^{-2}. \tag{10}$$

Since we would like to avoid storing the dense matrix $D^2 + VV^T$ computing its inverse, we can apply the above formula to solve the system $(D^2 + VV^T)u_1 = w_1$ as follows: compute

$$\begin{aligned} &z := D^{-2}w_1 \\ &t :\ (I + V^T D^{-2}V)t = V^T z \\ &u_1 := z - D^{-2}Vt. \end{aligned} \tag{11}$$

The linear system of equations in the second step involves the $k \times k$ symmetric positive definite matrix $(I + V^T D^{-2}V)$; hence it can be solved by computing the Cholesky factorization $L_v\Lambda_v L_v^T$ of that matrix. The work required to compute such a factorization is $O(k^3)$, which is small if k is small with respect to n. Computing the matrix $(I + V^T D^{-2}V)$ $nk^2/2 + O(nk)$ takes operations. Thus, the overall work required to solve $(D^2 + VV^T)u = w$ is $nk^2/2$ plus smaller order terms.

Next, to solve the system $A(D^2 + VV^T)^{-1}A^T u_2 = w_2$ we need to apply the SMW formula twice. First, using (10) above, we have that

$$A(D^2 + VV^T)^{-1}A^T = AD^{-2}A^T - AD^{-2}V(I + V^T D^{-2}V)^{-1}V^T D^{-2}A^T.$$

Now $AD^{-2}V(I + V^T D^{-2}V)^{-1}V^T D^{-2}A^T$ can be represented as $\hat{V}\Lambda_v^{-1}\hat{V}^T$, where $\hat{V} = AD^{-2}VL_v^{-T}$, and $L_v\Lambda_v L_v^T$ is the Cholesky factorization of $(I + V^T D^{-2}V)$. Therefore, after computing the Cholesky factorization $L\Lambda L^T = AD^{-2}A^T$, we can solve the system $A(D^2 + VV^T)^{-1}A^T u_2 = w_2$ by the following algorithm:

$$\begin{aligned} \hat{V} :&\quad L_v\hat{V}^T = V^T D^{-2}A^T \\ \bar{V} :&\quad L\bar{V} = \hat{V} \\ z :&\quad Lz = w_2 \\ t :&\quad (\Lambda_v - \bar{V}^T\Lambda^{-1}\bar{V})t = \bar{V}^T\Lambda^{-1}z \\ u_2 :&\quad L^T u_2 := \Lambda^{-1}(z + \bar{V}t). \end{aligned} \tag{12}$$

Assuming that $AD^{-2}A^T$ and its Cholesky factor L are sparse and that obtaining L and solving systems involving it requires $O(m^2)$ and $O(m)$ operations, respectively, the total operations count for

(12) is $O(m^2) + k^2m + O(km) + O(k^3)$ plus lower order terms. The Sherman-Morrison-Woodbury update has been used widely in the context of interior point methods for linear programming, [5], [17]. In that context, however, the method often runs into numerical difficulties. While modifications to the method have been proposed to deal with its numerical instability [1], [20], these modifications do not always help, and they increase the computational cost.

Intuition suggests that numerical problems occur when the matrix $D^2 + VV^T$, or just D^2, is not well conditioned. Consider the following small example:

Example 4.1 Let

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

where $\epsilon^2 << 1$. Applying the SMW procedure (11) to solve $D^2 + VV^T)u = w$, we obtain

$$z = D^{-2}w = \begin{pmatrix} \frac{w_1}{\epsilon^2} \\ w_2 \\ w_3 \end{pmatrix}$$

$$(V^T D^{-2}V + I)t = \begin{bmatrix} 3 + \frac{1}{\epsilon^2} & \frac{1}{\epsilon^2} \\ \frac{1}{\epsilon^2} & 3 + \frac{1}{\epsilon^2} \end{bmatrix} t \quad \text{and} \quad V^T z = \begin{bmatrix} \frac{w_1}{\epsilon^2} - w_2 + w_3 \\ \frac{w_1}{\epsilon^2} + w_2 + w_3 \end{bmatrix}.$$

$V^T D^{-2}V + I$ is badly conditioned and solving the system for t will produce a significant computational error due to round-off. In particular the size of the error depends on the size of ϵ^2: if ϵ^2 is less than machine precision, say 10^{-16}, then the computed t is totally inaccurate; if $\epsilon^2 \sim 10^{-10}$, then t can be computed correctly up to 5 digits. Clearly, the complete system $(D^2 + VV^T)u = w$ is well conditioned and, hence, can be solved with an accuracy up to the machine precision, if solved directly.

In Section 6 we show how such a situation may arise in the support vector machine setting. In the next subsection we present an alternative method that avoids many of the numerical difficulties of the SMW update.

4.2 Product-Form Cholesky Factorization

An alternative efficient method for solving the system $(D^2 + VV^T)u_1 = w_1$ is based on updating the Cholesky factorization of the matrix D^2 to accommodate the low rank term VV^T, rather than using low rank updates to the inverse of D^2 via the SMW formula. (See [2], [9] and [10] for various versions of this approach). Product-form versions of these methods were proposed in [11] for handling dense columns in the context of IPMs for LP and for efficiently solving SOCPs by IPMs in [12], and both of these applications of PFCF have been implemented in stat-of-the-art software such as SeDuMi [23]. The PFCF approach was also applied to the solutions of SVM problems by IPMs in [8]. Below, for completeness, we present the main formulas for one of the methods. For further details, see e.g., [11] and [8].

A product-form Cholesky factorization of F has the general form $L\tilde{L}^1\tilde{L}^2\cdots\tilde{L}^k\tilde{\Lambda}^k(\tilde{L}^k)^T\cdots(\tilde{L}^2)^T(\tilde{L}^1)^T L^T$, where L and each $\tilde{L}^i$ are unit lower triangular matrices and $\tilde{\Lambda}^k$ is a nonnegative diagonal matrix. Moreover, each $\tilde{L}^i$ has a special structure. To solve a system of linear equations $Fu = w$, one has to solve the following sequence of linear systems: $Lu^0 = w$, $\tilde{L}^1u^1 = u^0$, $\tilde{L}^2u^2 = u^1$, $\ldots$, $\tilde{L}^ku^k = u^{k-1}$, $(\tilde{L}^k)^T u^{k+1} = (\Lambda^k)^{-1}u^k$, $\ldots$, $(\tilde{L}^1)^T u = u^{2k-1}$. In general, solving an $n\times n$ triangular system takes $O(n^2)$ operations. However, as we will see below, each $\tilde{L}^i$ has a special structure that enables systems involving $\tilde{L}^i$ to be solved in only $O(n)$ operations.

Let us assume that we have the Cholesky factorization of a matrix F: $F = L\Lambda L^T$. For simplicity, we shall ignore any symmetric permutations that may have been applied to F so as to increase the sparsity of L. We can obtain a product form factorization of $F + vv^T$, where $v \in \mathbf{R}^n$ is a vector (hence, vv^T is a rank-one $n \times n$ matrix) as follows:

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

where p is the solution of $Lp = v$ and $\tilde{L}\tilde{\Lambda}\tilde{L}^T$ is the Cholesky factorization of $\Lambda + pp^T$. As shown in [9], [10], [11] and [12], $\tilde{L}$ has the special form

$$\tilde{L} = \begin{bmatrix} 1 & & & & & \\ p_2\beta_1 & 1 & & & & \\ p_3\beta_1 & p_3\beta_2 & 1 & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ p_{n-1}\beta_1 & p_{n-1}\beta_2 & p_{n-1}\beta_3 & \cdots & 1 & \\ p_n\beta_1 & p_n\beta_2 & p_n\beta_3 & \cdots & p_n\beta_{n-1} & 1 \end{bmatrix}, \tag{13}$$

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

```
$t_0 := 1,$
for $j = 1, 2, \ldots, n$
    $t_j := t_{j-1} + p_j^2/\lambda_j,$                (14)
    $\tilde{\lambda}_j := \lambda_j t_j / t_{j-1},$
    $\beta_j := p_j/(\lambda_j t_j).$
```

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 form Cholesky factorization

$$L\tilde{L}^1\tilde{L}^2\cdots\tilde{L}^k\tilde{\Lambda}^k(\tilde{L}^k)^T\cdots(\tilde{L}^2)^T(\tilde{L}^1)^T L^T. \tag{15}$$

It is important to note that there is no need to form and store the k matrices $\tilde{L}^1, \ldots, \tilde{L}^k$; rather only the pairs of vectors $(p^1, \beta^1), \ldots, (p^k, \beta^k)$ that are needed to define these matrices are stored.

For our first system $(D^2 + VV^T)u_1 = w_1$ the PFCF method is directly applicable. Since D^2 is diagonal, $L = I$ and the computational workload is $k^2n + 8kn + o(kn)$, where $o(kn)$ denotes terms that are of smaller order than km. Thus for k much smaller than n, this computation is very inexpensive (i.e., $O(n)$) even though it is roughly twice as expensive as the SMW computation (11). Alternative product-form methods for updating the Cholesky factorization of a matrix when a rank-k matrix of the form VSV^T is added to it, where $V \in \mathbf{R}^{n\times k}$ and is of full column rank and S is a $k \times k$ nonsingular diagonal matrix are presented in [12]. In the present context, $S = I$, and any of these methods could be used in place of the one given above.

4.3 Product-Form Cholesky Factorization of $A(D^2 + VV^T)^{-1}A^T$.

To apply the PFCF method to the matrix $M = A(D^2 + VV^T)^{-1}A^T$ we can write this matrix as the sum of a sparse matrix and a low rank matrix via the SMW updating formula, as we have shown in the previous section; but this would defeat the purpose of using PFCF, because of the numerical instability of the SMW update.

We now show how PFCF can be used to compute a factorization of $A(D^2 + VV^T)^{-1}A^T$ in a numerically stable manner in roughly $O(n^2) + k^2n$ time. Notice that even though the matrix is $m \times m$, the computational cost depends on n rather than m. This due to the fact that our method embeds this $m \times m$ matrix in a larger $n \times n$ matrix and applies the PFCF procedure to it.

Given the Cholesky factorization $AF^{-1}A^T = L_{11}\Lambda_{11}L_{11}^T$, we first show how we can compute the factorization $L_{11}\tilde{L}_{11}\tilde{\Lambda}_{11}\tilde{L}_{11}^T L_{11}^T = A(F + vv^T)^{-1}A^T$, for some $v \in \mathbf{R}^n$, where $\tilde{L}_{11}$, an $m \times m$ submatrix of an $n \times n$ matrix $\tilde{L}$, has a special form that requires only $O(m)$ operations to do forward and back-solves.

We will use the same technique that was used in the discussion that preceded Lemma 3.2. Suppose $A = [B, N]$, where B is nonsingular, and define

$$K = \begin{bmatrix} B & N \\ 0 & I \end{bmatrix} = \begin{bmatrix} A \\ E \end{bmatrix}, \text{ and } \bar{M} = KF^{-1}K^T. \tag{16}$$

We further assume that we have the Cholesky factorization of $\bar{M}$; i.e., we have

$$\bar{M} = \begin{bmatrix} AF^{-1}A^T & AF^{-1}E^T \\ EF^{-1}A^T & EF^{-1}E^T \end{bmatrix} = L\Lambda L^T$$

where

$$L = \begin{bmatrix} L_{11} & 0 \\ L_{12} & L_{22} \end{bmatrix} \text{and } \bar{\Lambda} = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}.$$

Clearly

$$AF^{-1}A^T = L_{11}\Lambda_1 L_{11}^T$$

Let us define another Cholesky factorization

$$\hat{L}\hat{\Lambda}\hat{L}^T = P\bar{M}^{-1}P = PK^{-T}FK^{-1}P,$$

where P is the reverse order permutation matrix (7). We have

$$\hat{L}^{-T}\hat{\Lambda}^{-1}\hat{L}^{-1} = P\bar{M}P,$$

and, hence,

$$\bar{M} = P\hat{L}^{-T}\hat{\Lambda}^{-1}\hat{L}^{-1}P = \underbrace{P\hat{L}^{-T}P}_{L}\underbrace{P\hat{\Lambda}^{-1}P}_{\Lambda}\underbrace{P\hat{L}^{-1}P}_{L^T} = L\Lambda L^T.$$

Let $z = PK^{-T}v$ and $\hat{L}p = z$ (note that $p = PL^TK^{-T}v$). Then we have that

$$PK^{-T}(F+vv^T)K^{-1}P = PK^{-T}FK^{-1}P + zz^T$$

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

where $\tilde{\hat{L}}\tilde{\hat{\Lambda}}\tilde{\hat{L}}^T$ is the Cholesky factorization of $(\hat{\Lambda}+pp^T)$ and $\tilde{\hat{L}}$ is of the special form (13).

We can then write

$$K(F+vv^T)^{-1}K^T = \underbrace{P\hat{L}^{-T}P}_{L}\underbrace{P\tilde{\hat{L}}^{-T}P}_{\tilde{L}}\underbrace{P\tilde{\hat{\Lambda}}^{-1}P}_{\tilde{\Lambda}}\underbrace{P\tilde{\hat{L}}^{-1}P}_{\tilde{L}^T}\underbrace{P\hat{L}^{-1}P}_{L^T};$$

i.e.,

$$\begin{bmatrix} A(F+vv^T)^{-1}A^T & A(F+vv^T)^{-1}E^T \\ E(F+vv^T)^{-1}A^T & E(F+vv^T)^{-1}E^T \end{bmatrix} = L\tilde{L}\tilde{\Lambda}\tilde{L}^TL^T.$$

Hence

$$A(F+vv^T)^{-1}A^T = L_{11}\tilde{L}_{11}\tilde{\Lambda}_1\tilde{L}_{11}^TL_{11}^T,$$

where

$$\tilde{L} = P\tilde{\hat{L}}^{-T}P = \begin{bmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{12} & \tilde{L}_{22} \end{bmatrix}, \text{ and } \tilde{\Lambda} = P\tilde{\hat{\Lambda}}^{-1}P = \begin{bmatrix} \tilde{\Lambda}_1 & 0 \\ 0 & \tilde{\Lambda}_2 \end{bmatrix}$$

Let

$$\tilde{\hat{L}} = \begin{bmatrix} \tilde{\hat{L}}_{11} & 0 \\ \tilde{\hat{L}}_{12} & \tilde{\hat{L}}_{22} \end{bmatrix}$$

Then

$$\tilde{L} = \begin{bmatrix} P\tilde{\hat{L}}_{22}^{-T}P & 0 \\ -P\tilde{\hat{L}}_{11}^{-T}\tilde{\hat{L}}_{12}^T\tilde{\hat{L}}_{22}^{-T}P & P\tilde{\hat{L}}_{11}^{-T}P \end{bmatrix}.$$

Finally, we have the expression

$$\tilde{L}_{11} = P\tilde{\hat{L}}_{22}^{-T}P.$$

Notice that the product-form factor $\tilde{L}_{11}$ does not have the special structure (13), but $\tilde{\tilde{L}}_{22}$ does, hence performing back and forward solves with $\tilde{L}_{11}$ is the same as multiplying by $\tilde{\tilde{L}}_{22}$ or its transpose. To obtain $\tilde{\tilde{L}}_{22}$ we need to apply the PFCF procedure to the $n \times n$ matrix $P\bar{M}^{-1}P$ and compute the entire lower triangular factor $\tilde{\tilde{L}}$. Moreover, $\tilde{\tilde{L}}$ is needed if further rank-one updates are performed. Here we have shown how to apply a rank-one update to $AF^{-1}A^T$. To use it efficiently, for $F = VV^T + D^2$, where $V \in \mathbf{R}^{n\times k}$, we first factorize $KD^{-2}K^T$ and then apply k rank-one updates iteratively to obtain the PFCF .

$$L\tilde{L}^1\tilde{L}^2\cdots\tilde{L}^k\tilde{\Lambda}^k(\tilde{L}^k)^T\cdots(\tilde{L}^2)^T(\tilde{L}^1)^T L^T. \tag{17}$$

As in Section 4.2, there is no need to form and store the k matrices $\tilde{L}^1, \ldots, \tilde{L}^k$; rather only the pair of vectors p^j and β^j that is needed to define each $\tilde{L}^j$ is stored.

To compute the factorization (17) we need to first compute the LU factorization of the matrix B and compute $\hat{V} = K^{-T}V$. Assuming that A is suitably sparse, this can be done in $O(m^2) + O(kn)$ operations, and needs to be done only once on the first interior point iteration. On every iteration, one needs to compute (i) the Cholesky factorization $L\Lambda L^T$ of $KD^{-2}K^T$, ($O(n^2)$ operations), (ii) $L^T\hat{V}$ ($O(kn)$ operations), (iii) $k(k-1)/2$ products of the form $(\tilde{L}^j)^T q^i \equiv P\tilde{\tilde{L}}^{-1} Pq^i$ ($k(k-1)n$ operations), and (iv) the elements of the k $\tilde{L}^j$ and $\tilde{\Lambda}^j$ by recurrence formulas (14) ($5kn$ operations). Thus the total number of operations needed at each iteration to compute the factorization (17) is $O(n^2) + k^2n$ plus smaller order terms. To solve a linear system of the form $AF^{-1}A^T u = w$ requires an additional $O(m) + 4km$ operations.

Choices for the matrix K other than the one defined by (16) are possible. For example, one can compute the QR factorization of A^T; i.e.,

$$A^T = QR = [Q_1\, Q_2]\begin{bmatrix} R_1 \\ 0 \end{bmatrix}$$

and set $K = \begin{bmatrix} A \\ E \end{bmatrix}$, where $E = Q_2^T$. However, such a choice is generally much more expensive to implement than (16).

One is most likely to use the above updates when n and m are similar in size, so the loss in computational efficiency, caused by factorizing $\bar{M}$ instead of $M = A(Q + D^2)^{-1}A^T$ is not significant. When $m \ll n$ or $AD^{-2}A^T$ is a dense matrix, it is preferable to factorize M directly.

The product-form Cholesky factorization method is numerically more stable than the SMW method both in theory and in practice. To illustrate this, it is easy to check that PFCF produces a correct result when applied to the small example in the previous subsection. In Section 6.2 we present some examples for which the implementation of an interior point method using the SMW update fails numerically. The implementation of the same interior point algorithm using the product-form Cholesky factorization works well for all these examples. PFCF has been implemented for linear programming and second-order cone programming problems, where it has worked well in practice. In [11] an analysis of the stability of PFCF is given for the LP case. The core of that analysis is the proof of the uniform boundedness of the factors L and $\tilde{L}$. We have shown that L is uniformly bounded for the QP case in Theorem 3.3. The uniform boundedness of $\tilde{L}$ is shown by the following theorem.

Theorem 4.2 *Let v be a fixed vector, and let $M = A(Q+D^2)^{-1}A^T$ and $A(Q+D^2+vv^T)^{-1}A^T$ have factorizations $L\Lambda L^T$ and $\bar{L}\bar{\Lambda}\bar{L}^T = L\tilde{L}\tilde{\Lambda}\tilde{L}^T L^T$, respectively. Then the elements of $\tilde{L}$ are uniformly bounded as $\mu \to 0$.*

Proof. By Theorem 3.1 the elements of $\bar{L}$ and L are uniformly bounded as $\mu \to 0$. But we can represent $\bar{L}$ as a product of factors $L\tilde{L}$. By Lemma 4.2 in [11] the elements of L^{-1} are also uniformly bounded. Since the factorization $\bar{L}\bar{\Lambda}\bar{L}^T$ is unique, $\tilde{L} = L^{-1}\bar{L}$ and hence, the elements of $\tilde{L}$ are uniformly bounded as $\mu \to 0$. □

4.4 Augmented System Factorizations

In some implementations of interior point methods, the symmetric indefinite augmented systems matrix T defined by (2) is factorized rather than the positive definite normal equations matrix (3). One way to do this was proposed by Fourer and Mehrotra [7] (see also [6]). Their approach applies a version of Bunch-Parlett [4] symmetric indefinite factorization to T; i.e., it computes

$$PTP^T \equiv L\Lambda L^T, \tag{18}$$

where P is a permutation matrix, L is a unit lower triangular matrix and D is a block diagonal matrix with 1×1 and 2×2 diagonal blocks. Unlike in the positive definite case, the permutation P cannot be determined prior to the numerical phase of the factorization. The Fourer-Mehrotra implementation of the Bunch-Parlett factorization determines P and the block structure of Λ during the numerical factorization by choosing a pivot sequence so that the elements of L remain bounded to ensure numerical stability, while at the same time controlling the growth of fill-in as much as possible. The permutation determined on the first interior point iteration is reused on subsequent iterations until numerical difficulties are encountered, since this speeds up the algorithm considerably. If such difficulties are detected, the full sparse Bunch-Parlett procedure is reapplied, usually resulting in a denser factorization. Computational experience has shown that LPs can usually be solved using only one or two pivot orders. However, step computation time for this approach (again in the LP case) is usually 50% - 100% more than that required by applying Cholesky factorization to (3) [29].

The Fourer-Mehrotra procedure can be applied to QP, resulting in sparse and numerically stable factorizations if both Q and A are suitably sparse, but it is not able to directly take advantage of the special structure of Q considered in this paper. However, it is possible to develop a product-form symmetric indefinite factorization analogously to the PFCF. Specifically, given the factorization (18), where T is the matrix in (2), we can write

$$P\begin{bmatrix} -(Q+vv^T+D^2) & A^T \\ A & 0 \end{bmatrix} P^T \begin{array}{l} = L(\Lambda - pp^T)L^T, \\ = L\tilde{P}^T\tilde{L}\tilde{\Lambda}\tilde{L}^T\tilde{P}L^T, \end{array} \tag{19}$$

where p is the solution of

$$Lp = \begin{pmatrix} (v \\ 0 \end{pmatrix} \tag{20}$$

and $\tilde{L}\tilde{\Lambda}\tilde{L}^T$ is the symmetric indefinite factorization of $\tilde{P}(\Lambda - pp^T)\tilde{P}^T$. Let $\hat{p} \equiv \tilde{P}p$ and $\hat{\Lambda} \equiv \tilde{P}\Lambda\tilde{P}^T$, and let $\hat{p}$ be partitioned into 1 and 2 element subvectors $\hat{p}_j$ according to the block structure of $\hat{\Lambda}$; i.e., assuming that $\hat{\Lambda} \equiv \text{diag}\{\hat{\Lambda}_1, \hat{\Lambda}_2, \ldots, \hat{\Lambda}_{\mathrm{k}}\}$, $\hat{p}$ can be written as $\hat{p}^T \equiv (\hat{p}_1^T, \hat{p}_2^T, \ldots, \hat{p}_k^T)$. We note that , as in the Fourer-Mehrotra and Bunch-Parlett cases, the permutation $\tilde{P}$ and the block structure of $\hat{\Lambda}$ can only be determined during the numerical factorization phase - i.e., during the computation of the recurrence formulas (22) below. It is easy to verify that $\tilde{L}$ has a special form, namely

$$\tilde{L} = \begin{bmatrix} I & & & & & \\ \hat{p}_2\beta_1^T & I & & & & \\ \hat{p}_3\beta_1^T & \hat{p}_3\beta_2^T & I & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ \hat{p}_{n-1}\beta_1^T & \hat{p}_{n-1}\beta_2^T & \hat{p}_{n-1}\beta_3^T & \cdots & I & \\ \hat{p}_n\beta_1^T & \hat{p}_n\beta_2^T & \hat{p}_n\beta_3^T & \cdots & \hat{p}_n\beta_{n-1}^T & I \end{bmatrix}, \tag{21}$$

where $\beta^T \equiv (\beta_1^T, \beta_2^T, \ldots, \beta_k^T)$ and $\tilde{\Lambda} \equiv \text{diag}\{\tilde{\Lambda}_1, \tilde{\Lambda}_2, \ldots, \tilde{\Lambda}_{\mathrm{k}}\}$ are partitioned to conform with the partitions of $\hat{p}$ and $\hat{\Lambda}$ and can be computed from the following recurrence relations:

$$\begin{array}{l} \sigma_0 := 1, \\ \text{for } j = 1, 2, \ldots, k \\ \quad \tilde{\Lambda}_j := \hat{\Lambda}_j + \sigma_{j-1}\hat{p}_j\hat{p}_j/t, \\ \quad \beta_j := \sigma_{j-1}\tilde{\Lambda}_j^{-1}\hat{p}_j, \\ \quad \sigma_j := \sigma_{j-1} - \beta_j^T\tilde{\Lambda}_j\beta_j. \end{array} \tag{22}$$

The above recursion is a specialization of a method first proposed by Bennett [2]. We have used it rather than an analogous version of the recurrence formulas (14) first proposed by Fletcher and Powell [9] and Gill, Murray and Saunders [10], since we no longer need to guarantee positive semi-definiteness of $\tilde{\Lambda}$ in the presence of round-off errors, as required by the PFCF approach. Since testing for growth in the elements in the factorization and determining the permutation $\tilde{P}$ must be done while computing the factorization, and this is expensive, one can reuse the same permutation $\tilde{P}$ on subsequent interior point iterations as in the Fourer-Mehrotra procedure until numerical difficulties are encountered.

A second approach for computing a factorization of T of the form (18) has been proposed by Vanderbei and Carpenter [25]. As above, P is a permutation matrix, but now Λ is a nonsingular diagonal matrix (i.e., it has strictly positive and negative diagonal elements but no 2×2 diagonal blocks). Such a factorization is guaranteed to exist as long as the first set of pivots is chosen from the diagonals of $-(Q + D^2)$ and enough of them are chosen so that the corresponding columns of A span the column space of A. The remaining pivots (i.e., the symmetric permutation P) can then be chosen in any order from the remaining diagonals since the matrix that results after the first set of pivots is *quasidefinite* [24]. Like the Fourer-Mehrotra procedure, this approach is able to take advantage of sparsity in Q and A, but cannot directly take advantage of the special structure of Q considered in this paper.

Again it is possible to develop a product-form version of this "indefinite" Cholesky factorization, but we do not advocate this since the factorization is numerically unstable like the SMW approach. In fact, it can be shown that the indefinite Cholescky factorization of a quasidefinite matrix is equivalent to a nested sequence of SMW updates.

5 Approximating Q by a low rank matrix

In some applications, Q is not a low rank matrix, but it can be approximated well by one. If Q is large-scaled, then it is desirable to compute and to use such a representation without computing and storing the whole matrix Q. One method for doing this is to compute an *incomplete* or a *truncated* Cholesky factorization VV^T of Q as proposed in [8]. This method is basically a Cholesky factorization with symmetric pivoting, where the k-th column of Q is computed only on the k-th step of the algorithm, when the k-th column of the Cholesky factor V is computed. Initially, only the diagonal elements of Q are needed. After forming V_k, the first k columns of the Cholesky factor, the trace $\mathrm{tr}(\Delta Q)$ of the residual matrix $\Delta Q \equiv Q - V_k V_k^T$ is computed. If the trace $\mathrm{tr}(\Delta Q)$ is below some predefined threshold, then the approximation $V_k V_k^T$ of Q is considered sufficiently good and the algorithm terminates. One can also decide to terminate the algorithm when a given threshold on the number of columns has been reached, and obtain a bound on the quality of the approximation achieved.

The proposed algorithm has complexity $O(nk^2)$ and takes the full advantage of a greedy approach for the best reduction in the approximation bound $\mathrm{tr}(\Delta Q)$.

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

$$(\tilde{P}) \qquad \begin{array}{ll} \min_{\mathrm{x}} & \frac{1}{2} x^T \tilde{Q} x - e^T x \\ \text{s.t.} & Ax = b, \\ & 0 \le x \le u. \end{array}$$

A natural question to ask is: how close is the solution of the perturbed problem to the solution of the original problem. Given a bound on the approximation error matrix, $\Delta Q = Q - \tilde{Q}$, we provide a bound on the difference between the optimal objective values of the original and the perturbed problems.

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.

Consider the following parametric QP problem

$$(P_\theta) \qquad \begin{array}{ll} \min_{\mathrm{x}} & \frac{1}{2}x^T(\tilde{Q} + \theta\Delta Q)x - e^T x \\ \text{s.t.} & Ax = b, \\ & 0 \le x \le u, \end{array}$$

where θ is a scalar parameter and $0 \le \theta \le 1$. Clearly, for $\theta = 1$ we have the original problem (P) and for $\theta = 0$ we have the perturbed problem $(\tilde{P})$. Let $x(\theta)$ denote an optimal solution of (P_θ), $X(\theta)$ denote the set of all optimal solutions of (P_θ), $f_\theta(x)$ denote the objective function of (P_θ), and $\phi(\theta) = f_\theta(x(\theta))$ be the optimal objective value as a function of θ.

The following lemma is a special case of a more general result on convex parametric optimization described in [19].

Lemma 5.1 *$\phi(\theta)$ is a concave, piecewise differentiable function. The right derivative of $\phi(\theta)$ is*

$$d\phi/d\theta_+ = \min\{\frac{1}{2}x^T\Delta Qx |\, x \in X(\theta)\}$$

and the left derivative is

$$d\phi/d\theta_- = \max\{\frac{1}{2}x^T\Delta Qx |\, x \in X(\theta)\}.$$

From the concavity of $\phi(\theta)$ and the expression for the derivatives it is easy to demonstrate that $|\phi(1) - \phi(0)| \le \frac{1}{2}u_{max}l_{max}\epsilon$, where l_{max} is the maximum of the number of positive components of $x(1)$ and $x(0)$ and u_{max} is the maximum upper bound.

In [8] a similar bound was presented for ΔQ positive semidefinite. The bound that we have shown here applied to any symmetric ΔQ, as long as $\tilde{Q} + \theta\Delta Q$ is positive semidefinite for $\theta \in [0, 1]$.

The above approximation bound generates another question: how does the active set at the solution changes when the original problem is replaced by a perturbed one. For example, if the perturbation of the problem does not preserve the value of x^* but preserves the optimal partition of components of x^* into those at their upper bound, lower bound and in between, then it is simple to identify the true solution to the original problem by solving a system of linear equations, whose size is the number of variables not at their bounds (in many applications this is much smaller than n).

Unfortunately deriving a theoretical bound on the error matrix ΔQ for which the active set is preserved is, in general, at least as hard as doing sensitivity analysis of a linear program with respect to the constraint matrix. While one might try to use the particular form of the perturbation ΔQ it still seems unclear if any practical bound can be obtained.

However, one might use this idea in practice to try to update a solution to approximate problem to obtain a solution to the original problem by using an active set method. The hope is that if the optimal partition does not change significantly from the original problem to the perturbed one then the optimal solution can be "corrected" just by a few iterations of an active set method.

In Section 6.2 we present a few experiments which show that such a "correction" of an approximate solution can work very nicely in practice.

6 Support Vector Machines

Support Vector Machines (SVM) is a field of Machine Learning that has received a lot of attention in the last decade, since the seminal work of Vapnik (see [27] and references therein). The core of Support Vector Machines is the following problem: we are given a set of data points in a k dimensional space $\mathcal{V} = \{v_i \in \mathbf{R}^k | \, i = 1, \ldots, n\}$, and a set of labels $\{a_i | \, i = 1, \ldots, n\}$ that partition the set $\mathcal{V}$ into a "positive" and "negative" subsets, $\mathcal{V}^+$ and $\mathcal{V}^-$: if $a_i = 1$ then $v_i \in \mathcal{V}^+$ and if $a_i = -1$ then $v_i \in \mathcal{V}^-$. We would like to find a surface that separates these two sets of points. After such a surface, or *classifier*, is found it is used to classify new unlabeled data points by assigning each new point to $\mathcal{V}^+$ or $\mathcal{V}^-$. The quality of the classifier is usually measured by the *generalization error* - the number of "wrong guesses" during labeling of the new data. For *linear* SVM a classifier is a hyperplane defined by a normal vector w and a bias scalar y, that separates the two sets; in other words w and y have to satisfy $a_i(w^T v_i + y) > 0$, for all $i = 1, \ldots, n$. A hyperplane that satisfies these constraints may not be unique. The "best" hyperplane is selected in such a way that the *margin* $\min_i\{a_i(w^T v_i + y)\}$ is maximized. Intuitively, such a classifier is positioned as far away from both sets as possible, while still separating them. Since the margin can be made as large as desired by scaling up w and y, some constraint on $||w||$ is necessary. For reasons that will be explained later in this section a bound on the Euclidean norm of w is usually used (versus, for example, a bound on the 1-norm of w). This bound introduces a nonlinear constraint in an otherwise linear problem. Instead of introducing such a constraint an equivalent reformulation of this problem was proposed in [26, 27]. In this formulation the objective is to minimize $||w||^2$, while the margin is fixed to equal 1: $a_i(w^T v_i + y) \geq 1$, for all $i = 1, \ldots, n$. This formulation is further extended to allow *outliers*; i.e., points violating the margin constraints. The constraints then become $a_i(w^T v_i + y) \geq 1 - \xi_i$, $\xi_i \geq 0$, for all $i = 1, \ldots, n$. To minimize the number of outliers a penalty term is added to the objective function. Here we consider the penalty term $c \sum_i \xi_i$, with some positive penalty parameter c, which leads to a formulation called the *1-norm soft margin* classification problem. This formulation is more efficient in SVM practice and more common than the *2-norm* formulation [13], which corresponds to adding the penalty term $c \sum_i \xi_i^2$ (not to be confused with the Euclidean norm of w which is used independently of the type of penalty term). The results of this paper easily extend to the 2-norm formulation.

Thus, we consider the following so-called linear SVM problem

$$\begin{array}{llll} & \min_{\xi,\mathrm{w},\mathrm{y}} & \frac{1}{2} w^T w + c \sum_{i=1}^{n} \xi_i & \\ (SVM) & \text{s.t.} & a_i(w^T v_i + y) \geq 1 - \xi_i, & i = 1, \ldots, n \end{array}$$

$$\xi \geq 0, \qquad i = 1, \ldots, n.$$

This is a convex quadratic programming problem. From QP duality, the optimal solution w^* of (SVM) can be expressed as $w^* = \sum_{i=1}^n x_i a_i v_i$, where x_i, $i = 1, \ldots, n$, $0 \leq x_i \leq c$ is an optimal solution to the dual problem; hence problem (SVM) can be written as

$$
(D_{SVM}) \qquad \begin{array}{ll}
\min_{\mathrm{x,y},\xi} & \frac{1}{2} x^T D_a V V^T D_a x + c \sum_{i=1}^n \xi_i \\
\text{s.t.} & D_a V V^T D_a x + a y \geq 1 - \xi, \\
\text{s.t.} & \xi \geq 0, 0 \leq x \leq ce.
\end{array}
$$

Here $V \in \mathbf{R}^{n \times k}$ is a matrix whose rows are the data vectors v_i, $i = 1, \ldots, n$ and D_a is a diagonal matrix with elements of the vector a on the diagonal.

The strength of this formulation, and, hence, of the SVM methodology in general, is that it can be easily extended to the nonlinear case. If the data is not linearly separable in the original space it may be desirable to lift it into a higher dimensional *feature* space, using some nonlinear mapping $\phi(\cdot)$, where the data becomes linearly separable (in a broad sense, this means that some presence of outliers is tolerated). Hence, we need to solve the linear SVM with the data vectors $\phi(v_i)$, $i = 1, \ldots, n$. Unfortunately the dimension of the feature space can be very large, and even infinite. Even storing the data in the feature space may be too costly or impossible. However, if we can somehow compute $\phi(v_i)^T \phi(v_j)$ for any pair i, j, without computing $\phi(v_i)$ and $\phi(v_j)$ then we can obtain all the input data for the second formulation (D_{SVM}). This can be done by means of kernel operation, $K(\cdot, \cdot)$, which is a scalar function of two vectors, satisfying a certain set of properties (that ensure that $K(v_i, v_j) = \phi(v_i)^T \phi(v_j)$ for some $\phi(\cdot)$). In SVM practice the choice of a good kernel operation is an important issue. A kernel corresponds to a class nonlinear separating surfaces in the original space; for example, the kernel defined by $K(u, v) = (u^T v)^2$ generates second order surfaces. It also corresponds to the mapping $\phi(u) = (u_1^2, \sqrt{2} u_1 u_2, \sqrt{2} u_1 u_3, \ldots, \sqrt{2} u_{n-1} u_n, u_n^2) \in \mathbf{R}^{n(n-1)/2}$.

Once the kernel operation is chosen, there is no need to determine $\phi(\cdot)$. The values $K(v_i, v_j)$ are plugged directly into (D_{SVM}). The problem is then solved for x, y and ξ. w is not computed but is expressed implicitly as $\sum_i^n x_i a_i \phi(v_i)$. When a new unlabeled data point v_{new} is classified, $w^T \phi(v_{new})$ is computed as $\sum_i^n x_i a_i \phi(v_i)^T \phi(v_{new}) = \sum_i^n x_i a_i K(v_i, v_{new})$. For SVM problems the number of positive components x_i at the optimal solution (which is the same as the number of data points which are on the margin or are outliers) is typically much smaller than the size of the whole data set, hence most of the elements of the last sum are zero and computing this sum in relatively inexpensive. Notice that if we did not use the Euclidean norm of w in the objective function in the original formulation, then we would not be able to express all the data in terms of inner products of the vectors in feature space, and hence we would not be able to use kernel operations efficiently.

The dual to problem (D_{SVM}) corresponding to the kernel K is the following convex quadratic programming problem.

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

where $x \in \mathbf{R}^n$ is a vector of variables, $a \in \mathbf{R}^n$ is the vector of labels 1 and -1, $Q = D_a K D_a$ and c is the penalty parameter associated with the training error.

For large scale problems (which are common in real world applications such as speech, document classification, optical character recognition, etc.) the training set is typically large and the Hessian of the objective function is a large dense matrix. However, if the Hessian of the objective function has a low rank (which happens when the dimension of the feature space is small) one can significantly improve the performance of an interior point method by using the techniques described in Section 4. In the case of SVM there is only one equality constraint, hence the only work lies in factorizing $Q + D^2$.

When a nonlinear kernel is used the rank of the Hessian matrix, in general, can be as large as the dimension of the feature space, which can be as large as infinity. However, it has been observed [22, **?**, 28] that for many classes of nonlinear SVM problems the eigenvalues of the Hessian matrix rapidly decay to zero. Hence, one can approximate the Hessian by a low-rank positive semidefinite matrix and take advantage of this low-rank representation to speed up an IPM as described in Section 5. The bound on the error in the objective function when such approximation is used is $\frac{1}{2}cl_{max}\epsilon$, where l_{max} is the maximum of the number of support vectors in the original and the perturbed problems.

Low rank approximation of the kernel matrix can be useful in other areas of SVM and was considered by Smola and Schölkopf in [22] in application to training set sampling (also see [**?**] for another low rank approximation). The quality of the approximation and the overall complexity of our method compares favorably with those of the approximation schemes proposed in [22] and [**?**] (also see [8] for details).

6.1 Example of bad numerical behavior of SMW update for SVM problems

In section 4 we gave an example of large numerical error produced by the SMW update. In the context of SVM problems, similar situations may arise. Let us try to provide some intuition. The diagonal elements of D^2 are $s_i/x_i + \xi_i/(c - x_i)$. By complementarity, as μ approaches zero, if $x_i \to x_i^* > 0$ then $s_i \to 0$ and if $s_i \to s_i^* > 0$ then $x_i \to 0$ (analogously, if $x_i \to x_i^* < c$ then $\xi_i \to 0$ and if $\xi_i \to \xi_i^* > 0$ then $x_i \to c$). Moreover, for strictly complementary solutions, whenever a variable converges to zero (or c), it converges with the same rate as the parameter μ. Thus, whenever s_i and ξ_i both converge to zero (which happens when the i-th data point is on the margin), then D_i^2 converges to zero with the same rate as μ. Otherwise, D_i^2 converges to infinity with the same rate as $1/\mu$. If $\mu = 10^{-8}$, then some elements of D^2 are of the order 10^{-8} and the others are of the order 10^8. Depending on the structure of the data V, the matrix $(V^T D^{-2} V + I)$ in (10) may become ill-conditioned and solving the "small" system of equations with this matrix produces significant error.

In our experiments the severe numerical problems of the SMW update could usually be avoided by applying some scaling to the data or using a different kernel matrix. In some other cases the loss of accuracy could not be avoided but was not very significant. However, overall, the SMW update seems to be sensitive to the scaling of the problem and is not robust. As an illustration, in the next subsection we present two small examples of SVM problems for which an SMW update based IPM fails to achieve various levels of accuracy depending on the setting and scaling of the problems.

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

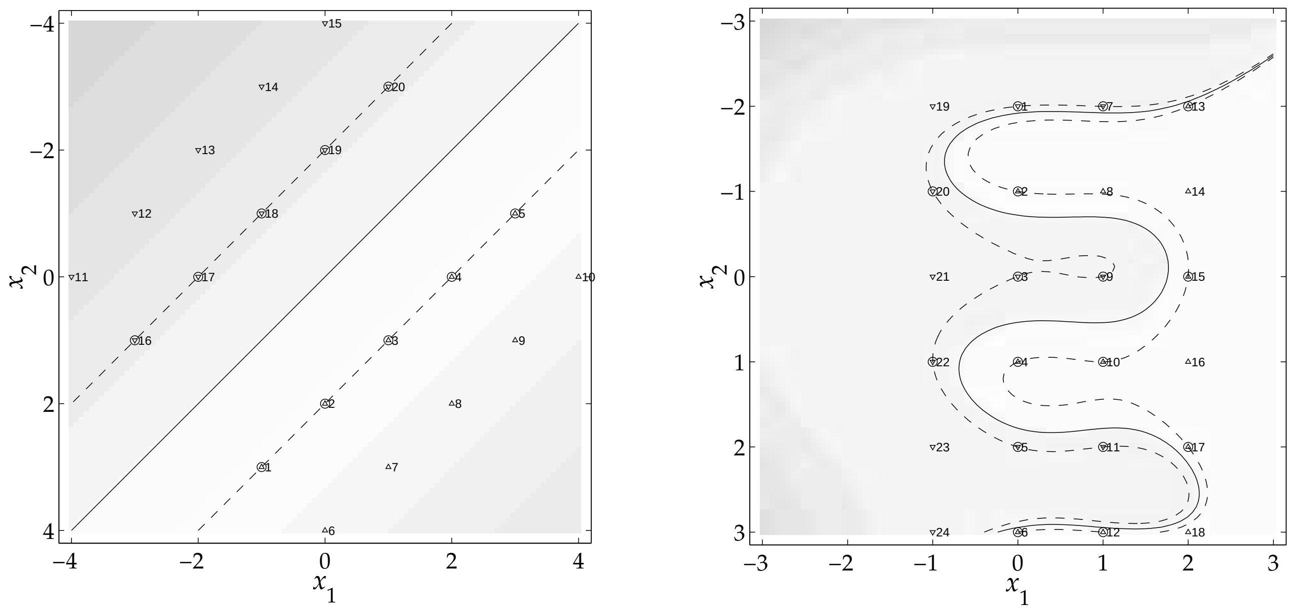

Figure 1: Examples of failure of the SMW update

We constructed a small example in the spirit of the one described in Section 4 to demonstrate possible numerical instability of the Sherman-Morrison-Woodbury update. The first plot in Figure 1 illustrates the first data set. Notice that the set of active support vectors is redundant (so the problem is degenerate). We then scaled this data by some number N. For $N = 1$ the SMW based solver convergds to 10 digits of accuracy but required 5 extra iterations compared to the PFCF based solver (PFCF solver took 13 iterations to converge). For $N = 10$ the SMW based solver stalled after achieving 6 digits of accuracy. For $N = 100$ there was no progress after 3 digits of accuracy and for $N = 1000$ the algorithm failed completely. The same IPM with the product-form Cholesky update converged to 10 digits of accuracy within 20 iterations for all instances.

The second plot in Figure 1 shows the data set of the second example. This data is not linearly separable. It is apparent from the layout that a 6-th degree polynomial is a good separator for this data set. Thus we used a polynomial kernel up to degree 6 $K(i,j) = ((v^i)^T v_j)/N + 1)^6$, where N is a scaling parameter. For $N = 1$ the SMW based solver achieved only 5 digit of accuracy and terminated after 88 iterations. For $N = 5$ the accuracy was 9 digits after 39 iterations and for $N = 10$ the method converged after 22 iterations. For all these instances the PFCF method found more than 10 significant digits in under about 14 iterations. We also shifted the data set by adding 3 to the x_2 component and the SMW based solver failed to achieve any accuracy for that instance with $N = 1$. Other values of N were tried and a similar trend as described above was observed.

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 a polynomial kernel. The algorithm stalled after achieving only 5 digits of accuracy, whereas the product-form Cholesky version converged to 12 digits of accuracy.

For interior point methods the problem not of not being able to obtain sufficient accuracy lies not only in the loss of accuracy itself, but also in the inability of the algorithm to terminate. It sometimes may take $50 - 100\%$ extra iterations for an interior point method to detect lack of progress and terminate.

7 Portfolio optimization

Portfolio optimization refers to a class of problems involving the allocation of capital over a number of available assets in order to maximize the "return" on the investment while minimizing the "risk". Different versions of the portfolio optimization problem arise from the way that these two conflicting objectives are handled. Markowitz is credited with being the first person to formulate mathematical models for this class of problems [15, 16]. In these models, the "return" and "risk" of a portfolio is measured by the expected value and the variance, respectively, of the random portfolio return. Markowitz showed that, given a lower bound on the acceptable return, the optimal (minimum risk) portfolio can be obtained by solving a convex quadratic programming problem. This mean-variance model has had a profound impact on the economic modeling of financial markets and the pricing of assets. The Capital Asset Pricing Model (CAPM) [21], [14] and [18] is based upon Markowitz's model, and Sharpe and Markowitz were awarded the Nobel Memorial Prize in Economic Sciences in 1990 for their work.

We consider the following "factor" model of the market that has n traded assets. The vector of asset returns over a single market period is denoted by $r \in \mathbf{R}^n$, with the interpretation that asset i returns $(1 + r_i)$ dollars for every dollar invested in it. The returns on the assets in different market periods are assumed to be independent. The single period return r is assumed to be a random variable given by

$$r = \mu + V^T f + \epsilon, \tag{23}$$

where $\mu \in \mathbf{R}^n$ is the vector of mean returns, $f \sim \mathcal{N}(0, F) \in \mathbf{R}^m$ is the vector of returns of the factors that drive the market, $V \in \mathbf{R}^{m \times n}$ is the matrix of factor loadings of the n assets and $\epsilon \sim \mathcal{N}(0, D)$

is the vector of residual returns. Here $x \sim \mathcal{N}(\mu, \Sigma)$ denotes that x is a multivariate Normal random variable with mean vector μ and covariance matrix Σ.

In addition, we assume that the vector of residual returns ϵ is independent of the vector of factor returns f, the covariance matrix $F \succ 0$ and the covariance matrix $D = \text{diag}(d) \succeq 0$, i.e. $d_i \geq 0$, $i = 1, \ldots, n$. Thus, the vector of asset returns $r \sim \mathcal{N}(\mu, V^T FV + D)$. Typically, the eigenvalues of the residual covariance matrix D are much smaller than those of the covariance matrix $V^T FV$ implied by the factors, i.e. the $V^T FV$ is a good low rank approximation to the covariance of the asset returns.

The assumption that factor and residual returns are Normally distributed can be relaxed. In fact, the distributions need not be Normal or even known. All that is needed are their first and second moments.

There are many different kinds of factors used in the financial industry. These include major market indices such as the S&P 500, Nasdaq 100, Russell 2000 and 30-year treasury bond indices. Often, a principal component analysis of the covariance matrix of asset returns (using recent historical data) and the eigenvectors corresponding to the leading eigenvalues are used as factors. In other cases, factors along with their covariance matrix F and the factor loadings V are provided by financial consulting firms. In some cases upper bounds on the weighting of each factor in the portfolio are imposed.

7.1 Basic portfolio models

A portfolio is a vector $\phi \in \mathbf{R}^n$, where the i-th component ϕ_i represents the fraction of total wealth invested in asset i. If no short selling is allowed, $\phi_i \geq 0$ for all i. The return r_ϕ of the portfolio ϕ is given by

$$r_\phi = r^T \phi = \mu^T \phi + f^T V\phi + \epsilon^T \phi \sim \mathcal{N}(\phi^T \mu, \phi^T (V^T FV + D)\phi).$$

The Markowitz portfolio selection model assumes that the expected value $\mathbf{E}[r]$ of the asset returns and the covariance $\mathbf{Var}\,[\mathbf{r}]$ are known with certainty. In this model investment "return" is the expected value $\mathbf{E}[r_\phi] \equiv \mu^T \phi$ of the portfolio return and the associated "risk" is the variance $\mathbf{Var}\,[\mathbf{r}_\phi] \equiv \phi^T (V^T FV + D)\phi$. The objective is to choose a portfolio ϕ^* that has the minimum variance amongst those that have expected return at least α, i.e. ϕ^* is the optimal solution of the convex quadratic programming problem

$$\begin{array}{rl} \text{minimize} & \mathbf{Var}\,[\mathbf{r}_\phi] \\ \text{subject to} & \mathbf{E}[r_\phi] \geq \alpha, \\ & e^T \phi = 1, \\ & \phi \geq 0. \end{array} \tag{24}$$

An alternative version of the portfolio optimization problem is the *maximum return problem*, in which the expected return is maximized subject to an upper bound on the variance. While this problem is not a convex QP, the following Lagrangian variant of it:

$$\begin{array}{rl} \text{maximize} & \mathbf{E}[r_\phi] - \lambda\, \mathbf{Var}\,[\mathbf{r}_\phi] \\ \text{subject to} & e^T \phi = 1, \\ & \phi \geq 0, \end{array} \tag{25}$$

yields the same solution for an appropriate choice of the Lagrange multiplier $\lambda > 0$ and is a convex QP.

Another variant of problem (**??**) is the *maximum Sharpe ratio problem*. Here the objective is to choose a portfolio that maximizes the ratio of the expected return of the portfolio, in excess of the risk-free rate r_f, to the standard deviation of the return; i.e.,

$$\text{maximize}_{\{\phi : e^T \phi = 1, \phi \geq 0\}} \quad \left\{ \frac{\mathbf{E}[r_\phi] - r_f}{\sqrt{\mathbf{Var}[\mathbf{r}_\phi]}} \right\} \tag{26}$$

Let us assume that the optimal value of this problem is strictly positive, i.e. there exists a portfolio with finite variance whose return is strictly greater than the risk-free rate r_f. Since, in practice, the variance of every asset is bounded, this assumption is equivalent to requiring that the return of at least one asset is greater than r_f.

Since $e^T \phi = 1$, the objective of problem (26),

$$\frac{\mu^T \phi - r_f}{\sqrt{\mathbf{Var}\,[\mathbf{r}_\phi]}} = \frac{(\mu - r_f e)^T \phi}{\sqrt{\mathbf{Var}\,[\mathbf{r}_\phi]}},$$

is a homogeneous function of ϕ. Therefore the normalization condition $e^T \phi = 1$ in (26) can be replaced by the constraint $(\mu - r_f e)^T \phi = 1$. Hence, (26) is equivalent to

$$\begin{array}{rlcl} \text{minimize} & \phi^T (V^T F V + D) \phi & & \\ \text{subject to} & (\mu - r_f e)^T \phi & = & 1, \\ & \phi & \geq & 0. \end{array} \tag{27}$$

It is easy to see that the constraint $(\mu - r_f e)^T \phi = 1$ can be relaxed to $(\mu - r_f e)^T \phi \geq 1$, since the latter will always be tight at an optimal solution. Consequently, the maximum Sharpe ratio problem is equivalent to a minimum variance problem with $\alpha = 1$, μ replaced by $\mu - r_f e$, and ϕ no longer normalized.

The maximum Sharpe ratio problem can be transformed into a convex QP even when there are additional equality and inequality constraints on ϕ. Specifically, suppose the portfolio ϕ is constrained to satisfy $A\phi \geq b$ (equality constraints $\bar{A}\phi = \bar{b}$ are handled in a similar manner). Then, the maximum Sharpe ratio problem,

$$\max_{\{\phi : A\phi \geq b, e^T \phi = 1, \phi \geq 0\}} \left\{ \frac{\mu^T \phi - r_f}{\sqrt{\mathbf{Var}\,[\phi]}} \right\}, \tag{28}$$

is equivalent to the minimum variance problem

$$\begin{array}{rrcl} \text{minimize} & \phi^T(V^TFV+D)\phi & & \\ \text{subject to} & (\mu - r_f e)^T\phi & \geq & 1, \\ & A\phi & \geq & \zeta b, \\ & e^T\phi & = & \zeta, \\ & \phi & \geq & 0, \\ & \zeta & \geq & 0, \end{array} \tag{29}$$

where ζ is an auxiliary variable that has been introduced to homogenize the constraints $A\phi \geq b$ and $e^T\phi = 1$.

7.2 Portfolio models in practice

In practice the basic models presented in the last subsection, all of which are convex QPs, are augmented with additional constraints depending on the specific application. Typically, upper bounds are imposed on the fraction of wealth ϕ_i invested in each asset i. Upper bounds on the total fractions of the portfolio invested in subsets of assets are also commonly specified. For example, restrictions are placed on how much may be invested in telecommunications stocks. One may also have variable upper bound constraints of the form $\phi_i \leq \phi_j$ for various asset pairs (i, j) to capture the preference of a portfolio manager for one asset j over another asset i from the same asset class. Most importantly, in most applications the portfolio optimization problem is solved given an existing portfolio; i.e., the optimal portfolio is obtained by modifying an existing portfolio rather than forming one from scratch and there are transaction costs that are often different for buying and selling a particular asset. In such situations, if $\bar{\phi}_i$ is the current fraction of asset i in the portfolio, then "trading" constraints of the form

$$\begin{array}{l} \phi_i - z_i + y_i = \bar{\phi}_i, \\ 0 \leq z_i \leq \theta_i - \bar{\phi}_i, \\ 0 \leq y_i \leq \bar{\phi}_i, \end{array} \tag{30}$$

are added to the basic model, where z_i and y_i are, respectively, the amounts of asset i added to and removed from the portfolio (in terms of fractions of the total wealth), and θ_i is an upper bound on ϕ_i.

In a typical application, the universe of assets might include the 500 stocks in the S&P500 index and the number of factors used might be approximately 20. There tend to be only a moderate number of asset subset upper bound constraints but these are usually fairly dense as the cardinalities of the subsets tend to be large. The number of trading constraints (30) can be quite large as these are imposed for each asset in the current portfolio. If these are the only constraints in the problem in addition to $e^T\phi = 1$, then the matrix B in (16) is the identity matrix with a row of ones appended to it and both K and K^{-1} are very sparse.

In some applications, disjunctive constraints of the form, either $\phi_i \geq \alpha_i$ or $\phi_i = 0$, and cardinality constraints which limit the number of assets in the portfolio with nonzero ϕ_i's to a specified number are imposed. This leads to a mixed integer quadratic program which can be solved by repeatedly solving convex QPs, to which polyhedral cuts have been added, within a branch-and-cut framework (e.g., see [3]).

References

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

[2] J. M. Bennett. Triangular factors of modified matrices. *Numerisches Mathematik*, 7:217–221, 1965.

[3] D. Bienstock. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming*, 74:121–140, 1996.

[4] J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. on Num. Anal.*, 8:639–655, 1971.

[5] 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.

[6] I. S. Duff, N. I. M. Gould, J. K. Reid, J. A. Scott, and K. Turner. The factorization of sparse indefinite matrices. *IMA J. of Num. Anal.*, 11:181–204, 1991.

[7] M.C. Ferris and T. S. Munson. Interior point methods for massive support vector machines. Technical report, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 2000.

[8] S. Fine and K. Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264, 2001.

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

[10] P. 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.

[11] D. Goldfarb and K. Scheinberg. A product-form cholesky factorization method for handling dense columns in interior point methods for linear programming. *Mathematical Programming*, 99:1–34, 2004.

[12] D. Goldfarb and K. Scheinberg. Product-form cholesky factorization in interior point methods for second-order cone programming. *Mathematical Programming*, 103:153–179, 2005.

[13] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy. A fast iterative nearest point algorithm for support vector machine classifier design. *IEEE Trnasactions on Neural Networks*, 11(1):124–136, 2000.

[14] J. Lintner. Valuation of risky assets adnd the selection of risky investments in stock portfolios and capital budgets. *Review of Economics and Statistics*, 47:13–37, 1965.

[15] H. M. Markowitz. Portfolio selection. *J. Finance*, 7:77–91, 1952.

[16] H. M. Markowitz. *Portfolio Selection.* Wiley, New York, 1959.

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

[18] J. Mossin. Equilibrium in capital asset markets. *Econometrica*, 34(4):768–783, 1966.

[19] R. T. Rockafellar. *Conjugate Duality and Optimization.* SIAM, Philadelphia, 1974.

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

[21] W. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *Journal of Finance*, 19(3):425–442, 1964.

[22] 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.

[23] J. F. Sturm. Sedumi1.03 matlab tool, http://fewcal.kub.nl/sturm/software/sedumi.html. Personal communication.

[24] R. J. Vanderbei. Symmetric quasi-definite matrices. *SIAM J. on Optimization*, 5 pages = 100-113, 1995.

[25] R. J. Vanderbei and T. J. Carpenter. Symmetric indefinite systems for interior point methods. *Mathematical Programming*, 58:1–32, 1993.

[26] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, 1982.

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

[28] 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.

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