12 Interior-Point Methods in Machine Learning

Jacek Gondzio J.Gondzio@ed.ac.uk
School of Mathematics and Maxwell Institute for Mathematical Sciences
The University of Edinburgh, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom

Interior-point methods for linear and (convex) quadratic programming display several features which make them particularly attractive for very large-scale optimization. They have an impressive low-degree polynomial worst-case complexity. In practice, they display an unrivalled ability to deliver optimal solutions in an almost constant number of iterations which depends very little, if at all, on the problem's dimension. Since many problems in machine learning can be recast as linear or quadratic optimization problems and it is common for them to have large or huge sizes, interior-point methods are natural candidates to be applied in this context.
In this chapter we will discuss several major issues related to interior point methods, including the worst-case complexity result, the features responsible for their ability to solve very large problems, and their existing and potential applications in machine learning.

12.1 Introduction

Soon after Karmarkar (1984) had published his seminal paper, interior-point methods (IPMs) were claimed to have unequalled efficiency when applied to large-scale problems. Karmarkar's first worst-case complexity proof was based on the use of projective geometry and cleverly chosen potential function, but was rather complicated. It generated huge interest in the optimization community and soon led to improvements and clarifications of the theory. A major step in this direction was made by Gill et al. (1986), who drew the community's attention to a close relation between Karmarkar's projective method and the projected Newton barrier method. The impressive effort of Lustig, Marsten, Shanno, and their collaborators in the late 1980s provided a better understanding of the computational aspects of IPMs for linear programming, including the central role played by the logarithmic barrier functions in the theory (Marsten et al., 1990). In the early 1990s sufficient evidence was already gathered to justify claims of the spectacular efficiency of IPMs for very large-scale linear programming (Lustig et al., 1994) and their ability to compete with a much older rival, the simplex method (Dantzig, 1963).

The simplex method has also gone through major developments over the last 25 years (Forrest and Goldfarb, 1992; Maros, 2003; Hall and McKinnon, 2005). It remains competitive for solving linear optimization problems and certainly provides a healthy pressure for further development of IPMs. It is widely accepted nowadays that there exist classes of problems for which one method may significantly outperform the other. The large size of the problem generally seems to favor interior-point methods. However, the structure of the problem, and in particular the sparsity pattern of the constraint matrix which determines the cost of linear algebra operations, may occasionally render one of the approaches impractical. The simplex method exploits well the hypersparsity of the problem (Hall and McKinnon, 2005). On the other hand, interior-point methods have a well-understood ability to take advantage of any block-matrix structure in the linear algebra operations, and therefore are significantly easier to parallelize (Gondzio and Grothey, 2006).

Many machine learning (ML) applications are formulated as optimization problems. Although, the vast majority of them lead to (easy) unconstrained optimization problems, certain classes of ML applications require dealing with linear constraints or variable nonnegativity constraints. interior-point methods are well suited to solve such problems because of their ability to handle inequality constraints very efficiently by using the logarithmic barrier functions.

The support vector machine training problems form an important class of ML applications which lead to constrained optimization formulations, and therefore can take a full advantage of IPMs. The early attempts to apply IPMs in the support vector machine training context (Ferris and Munson, 2003; Fine and Scheinberg, 2002; Goldfarb and Scheinberg, 2004) were very successful and generated further interest among the optimization community, stimulating several new developments (Gertz and Griffin, 2009; Jung et al., 2008; Woodsend and Gondzio, 2009, 2010). They relied on the ability of IPMs at taking advantage of the problem's special structure to reduce the cost of linear algebra operations. In this chapter we will concentrate on the support vector machine training problems and will use them to demonstrate the main computational features of interior-point methods.

The chapter is organized as follows. In Section 12.2 we will introduce the quadratic optimization problem and define the notation used. In Section 12.3 we will comment on the worst-case complexity result of a particular interior-point algorithm for convex quadratic programming, the feasible algorithm operating in a small neighborhood of the central path induced by the 2-norm. In Section 12.4 we will discuss several applications of interior-point methods which have been developed since about 2000 for solving different constrained optimization problems arising in support vector machine training. In Section 12.5 we will discuss existing and potential techniques which may accelerate the performance of interior-point methods in this context. Finally, in Section 12.6 we will give our conclusions and comment on possible further developments of interior-point methods.

12.2 Interior-Point Methods: Background

Consider the primal-dual pair of convex quadratic programming (QP) problems

$$\begin{array}{rlcrl} \multicolumn{2}{l}{\text{Primal}} & & \multicolumn{2}{l}{\text{Dual}} \\ \min & c^T x + \frac{1}{2}x^T Q x & \qquad & \max & b^T y - \frac{1}{2}x^T Q x \\ \text{s.t.} & Ax = b, & & \text{s.t.} & A^T y + s - Qx = c, \\ & x \geq 0; & & & y \text{ free},\;\; s \geq 0, \end{array} \tag{12.1}$$

where $A \in \mathcal{R}^{m \times n}$ has full row rank $m \leq n$, $Q \in \mathcal{R}^{n \times n}$ is a positive semidefinite matrix, $x, s, c \in \mathcal{R}^n$, and $y, b \in \mathcal{R}^m$. Using Lagrangian duality theory (see Bertsekas, 1995), the first-order optimality conditions for these problems can be written as

$$\begin{array}{rcl} Ax & = & b \\ A^T y + s - Qx & = & c \\ XSe & = & 0 \\ (x, s) & \geq & 0, \end{array} \tag{12.2}$$

where X and S are diagonal matrices in $\mathcal{R}^{n \times n}$ with elements of vectors x and s spread across the diagonal, respectively, and $e \in \mathcal{R}^n$ is the vector of ones. The third equation, $XSe = 0$, called the complementarity condition, can be rewritten as $x_j s_j = 0$, $\forall j = \{1, 2, \ldots, n\}$ and implies that at least one of the two variables x_j and s_j has to be zero at the optimum. The complementarity condition is often a source of difficulty when solving optimization problems, and the optimization approaches differ essentially in the way they deal with this condition.

Active set methods and their prominent example, the simplex method for linear programming, make an intelligent guess that either $x_j = 0$ or $s_j = 0$. They choose a subset of indices $j \in \mathcal{B} \subset \{1, 2, \ldots, n\}$ such that x_j is allowed to be nonzero and force the corresponding $s_j = 0$, while for the remaining indices $j \in \mathcal{N} = \{1, 2, \ldots, n\} \setminus \mathcal{B}$ they force $x_j = 0$ and allow s_j to take nonzero values. Such a choice simplifies the linear algebra operations which can be reduced (in the LP case) to consider only a submatrix of A induced by columns from set $\mathcal{B}$. The simplex method allows only one index to be swapped between $\mathcal{B}$ and $\mathcal{N}$ at each iteration. (In the more general context of active set methods, only one index of variable and/or active constraint can be exchanged at each iteration.) Hence an inexpensive update is performed to refresh active/inactive matrices, and this is reflected in a very low (almost negligible) cost of a single iteration. However, active set methods may require a huge number of iterations to be performed. This is a consequence of the difficulty in guessing the correct partition of indices into basic-nonbasic (active-inactive) subsets. The simplex method for linear programming is not a polynomial algorithm. Klee and Minty (1972) constructed a problem of dimension n, the solution of which requires 2^n iterations of the simplex method. However, in practice it is very rare for the simplex method to perform more than $m + n$ iterations on its way to an optimal solution (Forrest and Goldfarb, 1992; Maros, 2003; Hall and McKinnon, 2005).

Interior-point methods perturb the complementarity condition and replace $x_j s_j = 0$ with $x_j s_j = \mu$, where the parameter μ is driven to zero. This removes the need to "guess" the partitioning into active and inactive inequality constraints: the algorithm gradually reduces μ, and the partition of vectors x and s into zero and nonzero elements is gradually revealed as the algorithm progresses. Removing the need to "guess" the optimal partition is at the origin of the proof of the polynomial worst-case complexity of the interior-point method. Indeed, the best IPM algorithm known to date finds the ε-accurate solution of an LP or convex QP problem in $\mathcal{O}(\sqrt{n} \log(1/\varepsilon))$ iterations (Renegar, 1988). Again, in practice IPMs perform much better than that and converge in a number of iterations which is almost a constant, independent of the problem dimension (Colombo and Gondzio, 2008). However, one iteration of an IPM may be costly. Unlike the simplex method, which works with a submatrix of A, IPM involves the complete matrix A to compute the Newton direction for the perturbed first-order optimality conditions, and for nontrivial sparsity patterns in A this operation may be expensive and occasionally prohibitive.

The derivation of an interior-point method for optimization relies on three basic ideas:

1. Logarithmic barrier functions are used to "replace" the inequality constraints
2. Duality theory is applied to barrier subproblems to derive the first-order optimality conditions which take the form of a system of nonlinear equations, and
3. Newton's method is employed to solve this system of nonlinear equations.

To avoid the need to guess the activity of inequality constraints $x \geq 0$, interior-point methods employ the logarithmic barrier function of the form $-\mu \sum_{j=1}^{n} \log x_j$ added to the objective of the primal problem in (12.1). The barrier parameter μ weighs the barrier in relation to the QP objective. A large value of μ means that the original objective is less important, and the optimization focuses on minimizing the barrier term. The repelling force of the barrier prevents any of the components x_j from approaching their boundary value of zero. In other words, the presence of the barrier keeps the solution x in the interior of the positive orthant. Reducing the barrier term changes the balance between the original QP objective and the penalty for approaching the boundary. The smaller μ is the stronger the role of the original QP objective is. Much of the theory and practice of IPMs concentrates on clever ways of reducing the barrier term from a large initial value, used to promote centrality at the beginning of the optimization, to small values needed to weaken the barrier and to allow the algorithm to approach an optimal solution. In the linear programming case, the optimal solution lies on the boundary of the feasible region and many components of vector x are zero.

Applying Lagrangian duality theory to the barrier QP subproblem

$$\min \;\; c^T x + \frac{1}{2} x^T Q x - \mu \sum_{j=1}^{n} \log x_j \quad \text{s.t.} \quad Ax = b \tag{12.3}$$

gives the following first-order optimality conditions:

$$\begin{array}{rcl} Ax & = & b \\ A^T y + s - Qx & = & c \\ XSe & = & \mu e \\ (x, s) & \geq & 0. \end{array} \tag{12.4}$$

Comparison of (12.2) and (12.4) reveals that the only difference is a perturbation of the complementarity constraint which for the barrier subproblem requires all complementarity products $x_j s_j$ to take the same value μ. Observe that the perturbed complementarity condition is the only nonlinear equation in (12.4). For any $\mu > 0$, system (12.4) has a unique solution, $(x(\mu), y(\mu), s(\mu))$, $x(\mu) > 0$, $s(\mu) > 0$, which is called a μ-center. A family of these solutions for all positive values of μ determines a (continuous) path $\{(x(\mu), y(\mu), s(\mu)) : \mu > 0\}$ which is called the primal-dual *central path* or *central trajectory.*

interior-point algorithms apply Newton's method to solve the system of nonlinear equations (12.4). There is no need to solve this system to a high degree of accuracy. Recall that (12.4) is only an approximation of (12.2) corresponding to a specific choice of the barrier parameter μ. There is no need to solve it exactly because the barrier μ will have to be reduced anyway. IPMs apply only *one* iteration of the Newton method to this system of nonlinear equations and immediately reduce the barrier. Driving μ to zero is a tool which enforces convergence of (12.4) to (12.2), and takes iterates of IPM toward an optimal solution of (12.1). To perform a step of Newton's method for (12.4), the Newton direction $(\Delta x, \Delta y, \Delta s)$ is computed by solving the following system of linear equations,

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I_n \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_p \\ \xi_d \\ \xi_\mu \end{bmatrix} = \begin{bmatrix} b - Ax \\ c + Qx - A^T y - s \\ \mu e - XSe \end{bmatrix} \tag{12.5}$$

where I_n denotes the identity matrix of dimension n.

The theory of interior-point methods requires careful control of the error in the perturbed complementarity condition $XSe \approx \mu e$. Take an arbitrary $\theta \in (0, 1)$, compute $\mu = x^T s/n$, and define

$$N_2(\theta) = \{(x, y, s) \in \mathcal{F}^0 \mid \|XSe - \mu e\| \leq \theta\mu\}, \tag{12.6}$$

where $\mathcal{F}^0 = \{(x, y, s) \mid Ax = b,\ A^T y + s - Qx = c,\ (x, s) > 0\}$ denotes the primal-dual strictly feasible set. (Unless explicitly stated otherwise, the vector norm $\|\cdot\|$ will always denote the Euclidean norm.) Observe that all points in $N_2(\theta)$ exactly satisfy the first two (linear) equations in (12.4) and approximately satisfy the third (nonlinear) equation. In fact, $N_2(\theta)$ defines a neighborhood of the central path. Interestingly, the size of this neighborhood reduces with the barrier parameter μ. The theory of IPMs requires all the iterates to stay in this neighborhood. This explains why an alternative name to IPMs is path-following methods: indeed, these algorithms follow the central path on their way to optimality.

In the next section we will comment on an impressive feature of the interior-point method: it is possible to prove that an algorithm operating in the $N_2(\theta)$ neighborhood that is applied to a convex QP converges to an ε-accurate solution in $\mathcal{O}(\sqrt{n}\log(1/\varepsilon))$ iterations.

12.3 Polynomial Complexity Result

A detailed proof of the complexity result is beyond the scope of this chapter. The reader interested in the proof may consult an excellent textbook on IPMs by Wright (1997) in which a proof for the linear programming case is given. An extension to an IPM for quadratic programming requires some extra effort, and care has to be taken of terms which result from the quadratic objective.

The proof heavily uses the fact that all iterates belong to an $N_2(\theta)$ neighborhood (12.6) of the central path. Consequently, all iterates are strictly primal-dual feasible which simplifies the right-hand-side vector in the linear system defining the Newton direction (12.5):

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I_n \\ S & 0 & X \end{bmatrix} \cdot \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu e - XSe \end{bmatrix}. \tag{12.7}$$

A systematic (though very slow) reduction of the complementarity gap is imposed by forcing a decrease of the barrier term in each iteration l. The required reduction of μ may seem very small: $\mu^{l+1} = \sigma\mu^l$, where $\sigma = 1 - \beta/\sqrt{n}$ for some $\beta \in (0,1)$. However, after a sufficiently large number of iterations, proportional to $\sqrt{n}$, the achieved reduction is already noticeable because

$$\frac{\mu^l}{\mu^0} = (1 - \beta/\sqrt{n})^{\sqrt{n}} \approx e^{-\beta}.$$

After $C \cdot \sqrt{n}$ iterations, the reduction achieves $e^{-C\beta}$. For a sufficiently large constant C the reduction can thus be arbitrarily large (i.e., the complementarity gap can become arbitrarily small). In other words, after a number of iterations proportional to $\sqrt{n}$, the algorithm gets arbitrarily close to a solution. In the parlance of complexity theory, the algorithm converges in $\mathcal{O}(\sqrt{n})$ iterations. We state the complexity result but omit the proof.

Theorem 12.3.1. *Given $\varepsilon > 0$, suppose that a feasible starting point $(x^0, y^0, s^0) \in N_2(0.1)$ satisfies $(x^0)^T s^0 = n\mu^0$,* where *$\mu^0 \leq 1/\varepsilon^\kappa$, for some positive constant κ. Then there exists an index L with $L = \mathcal{O}(\sqrt{n}\ln(1/\varepsilon))$, such that $\mu^l \leq \varepsilon, \quad \forall l \geq L$.*

The very good worst-case complexity result of IPM for quadratic programming is beyond any competition in the field of optimization. Two features in particular are unprecedented. First, the number of iterations is bounded by the square root of the problem dimension. The computational experience of Colombo and Gondzio (2008) shows a much better practical iteration complexity which displays a logarithmic dependence on the problem dimension. Second, the complexity result reveals the dependence $\mathcal{O}(\ln(1/\varepsilon))$ on the required precision ε. Unlike IPMs, gradient methods (Nesterov, 2005) can provide only complexity results of $\mathcal{O}(1/\varepsilon)$ or $\mathcal{O}(1/\varepsilon^2)$. If one solves problems to merely 1- or 2-digit exact solution ($\varepsilon = 10^{-1}$ or $\varepsilon = 10^{-2}$), the terms $1/\varepsilon$ or $1/\varepsilon^2$ in the complexity result may seem acceptable. However, for a higher accuracy, say, $\varepsilon = 10^{-3}$ or smaller, the superiority of IPMs becomes obvious. (In the author's opinion, this outstanding feature of IPMs is not appreciated enough by the machine learning community.)

The practical implementation of IPMs differs in several points from the algorithm which possesses the best theoretical worst-case complexity. First, the most efficient primal-dual method is the *infeasible* algorithm. Indeed, there is no reason to force the algorithm to stay within the primal-dual strictly feasible set $\mathcal{F}^0$ and unnecessarily limit its space to maneuver. IPMs deal easily with any infeasibility in the primal and dual equality constraints by taking them into account in the Newton system (12.5). Second, there is no reason to restrict the iterates to the (very small) $N_2(\theta)$ neighborhood of the central path. Practical algorithms (Colombo and Gondzio, 2008) use a symmetric neighborhood $N_S(\gamma) = \{(x, y, s) \in \mathcal{F}^0 \,|\, \gamma\mu \le x_j s_j \le 1/\gamma\mu, \forall j\}$, where $\gamma \in (0, 1)$ or a so-called infinity neighborhood $N_\infty(\gamma) = \{(x, y, s) \in \mathcal{F}^0 \,|\, \gamma\mu \le x_j s_j, \forall j\}$, in which only too-small complementarity products are forbidden. Third, there is no reason to be overcautious in reducing the complementarity gap by a term $\sigma = 1 - \beta/\sqrt{n}$ which is so close to 1. Practical algorithms allow σ to be any number from the interval $(0, 1]$ and, indeed, the author's experience (Colombo and Gondzio, 2008) shows that the average reduction of the complementarity gap achieved in each IPM iteration $\sigma_{average}$ is usually in the interval $(0.1, 0.5)$. Deviation from the (close to 1) value of σ allowed by the theory requires the extra safeguards to make sure x and s remain nonnegative. This means that Newton steps have to be damped and stepsize α takes values smaller than 1.

12.4 Interior-Point Methods for Machine Learning

The main difficulty and the main computational effort in IPM algorithms is the solution of the Newton equation system: either (12.7) if we use a feasible algorithm of theoretical interest, or (12.5) if we use a practical infeasible algorithm. A common approach is to eliminate $\Delta s = X^{-1}(\xi_\mu - S\Delta x)$ and get the following symmetric but indefinite augmented system,

$$\begin{bmatrix} -Q-\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} f \\ h \end{bmatrix} = \begin{bmatrix} \xi_d - X^{-1}\xi_\mu \\ \xi_p \end{bmatrix}, \tag{12.8}$$

where $\Theta = XS^{-1}$, or make one more elimination step $\Delta x = (Q + \Theta^{-1})^{-1}(A^T\Delta y - f)$ and get the symmetric and positive definite normal equations system

$$(A(Q+\Theta^{-1})^{-1}A^T)\Delta y \;=\; g = A(Q+\Theta^{-1})^{-1}f + h. \tag{12.9}$$

For linear optimization problems (when $Q = 0$) the normal equations system (12.9) is usually the preferred (and default) option. For quadratic optimization problems with nontrivial matrix Q, an augmented system (12.8) is the best option. Indeed, the inversion of $(Q + \Theta^{-1})$ might completely destroy the sparsity in (12.9) and make the solution of this system very inefficient. There exists an important class of *separable* quadratic optimization problems in which Q is a diagonal matrix, and therefore the operation $(Q + \Theta^{-1})^{-1}$ produces a diagonal matrix and allows for the reduction to normal equations.

Several well-known reformulation tricks allow the extension of the class of separable problems and the conversion of certain nonseparable problems into separable ones (see Vanderbei, 1997). This is possible, for example, when matrix Q can be represented as $Q = Q_0 + VDV^T$, where Q_0 is easily invertible (say, diagonal) and $V \in \mathcal{R}^{n\times k}$, $D \in \mathcal{R}^{k\times k}$ with $k \ll n$ defining a low-rank correction. By introducing an extra variable $u = V^T x$, the quadratic objective term in problem (12.1) can be rewritten as $x^TQx = x^TQ_0x + u^TDu$ and the following quadratic optimization problem equivalent to (12.1) is obtained:

$$\begin{array}{rl} \min & c^Tx + \frac{1}{2}x^TQ_0x + u^TDu \\ \text{s.t.} & Ax = b, \\ & V^Tx - u = 0, \\ & x \geq 0,\ u \text{ free.} \end{array} \tag{12.10}$$

Although this new problem has more constraints ($m + k$ as opposed to m in (12.1)) and has $n + k$ variables, while (12.1) had only n, it is significantly easier to solve because its quadratic form $\begin{bmatrix} Q_0 & 0 \\ 0 & D \end{bmatrix}$ is easily invertible (diagonal) and allows for the use of the normal equations formulation in the computation of Newton direction.

Numerous classification problems in support vector machine training applications benefit from the above transformation. They include, for example, 1- or 2-norm classification, universum classification, and ordinal and ε-insensitive regressions. To demonstrate how the technique works, we will consider a 2-norm classification with support vector machines using the simplest linear kernel. Let a training set of n points $p_j \in \mathcal{R}^k$, $j = 1, 2, ..., n$ with binary labels $r_j \in \{-1, 1\}$, $j = 1, 2, ..., n$ be given. We look for a hyperplane $w^T p + w_0 = 0$ which best separates the points with different labels, namely, it maximizes the separation margin and minimizes the overall 1-norm error of misclassifications. The corresponding quadratic optimization problem and its dual have the following forms:

$$\begin{array}{llcll} & \text{Primal} & & & \text{Dual} \\ \min & \frac{1}{2}w^T w + \tau e^T \xi & & \max & e^T z - \frac{1}{2} z^T (RP^T PR) z \\ \text{s.t.} & R(P^T w + w_0 e) \geq e - \xi & & \text{s.t.} & r^T z = 0 \\ & \xi \geq 0; & & & 0 \leq z \leq \tau e, \end{array} \tag{12.11}$$

where $P \in \mathcal{R}^{k \times n}$ is a matrix the columns of which are formed by the points $p_j \in \mathcal{R}^k$, $R \in \mathcal{R}^{n \times n}$ is a diagonal matrix with labels r_j on the diagonal, $\xi \in \mathcal{R}^n$ are errors of misclassification, and τ is a positive parameter measuring the penalty of misclassifications.

Direct application of IPM to any of these problems would be challenging because of the expected very large size of the data set n. The primal problem has an easy, separable quadratic objective but a large number of linear constraints. The dual problem, on the other hand, has only a single equality constraint but its Hessian matrix $RP^T PR \in \mathcal{R}^{n \times n}$ is completely dense. The dual form is preferred by the ML community because it can easily accommodate any general kernel K. (The dual problem in (12.11) corresponds to a linear kernel $K = P^T P$.)

To provide a better understanding of where the difficulty is hidden, we give forms of augmented equation systems which would be obtained if an IPM was applied directly to the primal or to the dual in (12.11):

$$\begin{bmatrix} -I_k & 0 & PR \\ 0 & -\Theta_\xi^{-1} & I_n \\ RP^T & I_n & 0 \end{bmatrix} \begin{bmatrix} \Delta w \\ \Delta \xi \\ \Delta y \end{bmatrix} = \begin{bmatrix} f_w \\ f_\xi \\ h \end{bmatrix} \tag{12.12}$$

and

$$\begin{bmatrix} -(RP^T PR + \Theta_z^{-1}) & r \\ r^T & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta y \end{bmatrix} = \begin{bmatrix} f_z \\ h \end{bmatrix}. \tag{12.13}$$

To simplify the discussion, we keep using the notation of (12.8) and always denote Lagrange multipliers associated with the linear constraints as y and the right-hand-side vectors in these equations as (f, h). The dimensions of these vectors have to be derived from the formulations of the primal and dual problems in (12.11). For example, for the primal problem and equation

(12.12), $\Delta y \in \mathcal{R}^n$, $f_w \in \mathcal{R}^k$, $f_\xi \in \mathcal{R}^n$, and $h \in \mathcal{R}^n$; for the dual problem and equation (12.13), $\Delta y \in \mathcal{R}$, $f_z \in \mathcal{R}^n$, and $h \in \mathcal{R}$. It is easy to verify that the elimination of diagonal block diag$\{I_k, \Theta_\xi^{-1}\}$ in (12.12) (which corresponds to the elimination of Δw and $\Delta\xi$) would create a dense normal equations matrix of form $RP^TPR + \Theta_\xi$, producing a dense linear equation system with difficulty comparable to that of (12.13).

Although the matrix $RP^TPR + \Theta_\xi \in \mathcal{R}^{n\times n}$ (or $RP^TPR + \Theta_z^{-1} \in \mathcal{R}^{n\times n}$ in (12.13)) is completely dense and is expected to be large, its inversion can be computed efficiently using the Sherman-Morrison-Woodbury (SMW) formula, which exploits the low-rank representation of this matrix. Indeed, since $PR \in \mathcal{R}^{k\times n}$ and $\Theta_\xi \in \mathcal{R}^{n\times n}$ is invertible, we can write

$$(RP^TPR + \Theta_\xi)^{-1} = \Theta_\xi^{-1} - \Theta_\xi^{-1}RP^T(I_k + PR\Theta_\xi^{-1}RP^T)PR\Theta_\xi^{-1} \quad (12.14)$$

and then replace equation $(RP^TPR + \Theta_\xi)\Delta y = g$ with a sequence of operations:
Step 1: calculate $t_1 = PR\Theta_\xi^{-1}g$,
Step 2: solve $(I_k + PR\Theta_\xi^{-1}RP^T)t_2 = t_1$,
Step 3: calculate $\Delta y = \Theta_\xi^{-1}(g - RP^Tt_2)$.
Since we expect $k \ll n$, the application of the SMW formula offers a major improvement over a direct inversion of the large and dense matrix $RP^TPR + \Theta_\xi$. Indeed, SMW requires several matrix-vector multiplications with $PR \in \mathcal{R}^{k\times n}$ which involve only kn flops, and building and inversion of the Schur complement matrix

$$S = I_k + PR\Theta_\xi^{-1}RP^T, \quad (12.15)$$

which needs $\mathcal{O}(k^2n + k^3)$ flops. In contrast, building and inversion of $RP^TPR + \Theta_\xi$ would require $\mathcal{O}(kn^2 + n^3)$ flops. An additional and very important advantage of the SMW algorithm is its storage efficiency: the matrix $RP^TPR + \Theta_\xi$ does not have to be formulated and stored; we only need to store original data $PR \in \mathcal{R}^{k\times n}$ and the $k \times k$ Schur complement matrix (12.15).

Ferris and Munson (2003) considered a variety of formulations of linear support vector machines and applied interior-point methods to solve them. They used the OOQP solver of Gertz and Wright (2003) as a basic tool for their developments. The Newton equation systems were solved using the SMW formula. The results of their efforts very clearly demonstrated the IPM's ability to deal with problems in which the number of data points n was large, reaching millions. Their test examples had a moderate number of features $k = 34$.

The efficiency of an SMW-based IPM implementation is determined by the linear algebra operation of solving (12.12) (or (12.13)). This approach is very easy to parallelize (Ferris and Munson, 2003) because the bulk of the work lies in the matrix-vector multiplications operating on PR and its transpose. Indeed, significant speedups may be achieved simply by splitting the storage of this matrix between different processors and reducing the number n_i of points stored on a given processor $i = 1, 2, ..., p$, $(\sum_{i=1}^{p} n_i = n)$ to improve data locality.

The Schur complement approach has an inherent weakness that is difficult to overcome. Its numerical accuracy critically depends on the stability of the easily invertible matrix (Θ_ξ in (12.14)) and the scaling of columns in the low-rank corrector (PR in (12.14)). It is actually a general weakness of SMW that is unrelated to IPM applications. In our case, when SMW is applied in the interior-point method for support vector machines, only one of these two potential weaknesses can be remedied. It is possible to scale the original problem data P and improve the properties of the low-rank corrector PR. However, to the best of the author's knowledge, there is no easy way to control the behavior of matrix Θ_ξ. The entries of this matrix display a disastrous difference in magnitude: as IPM approaches optimality, elements in one subset go to infinity while elements in the other subset go to zero. Consequently, the inversion of Θ_ξ is very unstable and always adversely affects the accuracy of the solution which can be obtained using the SMW formula (12.14).

Goldfarb and Scheinberg (2008) constructed a small artificial dataset on which a Schur complement-based IPM implementation ran into numerical difficulties and could not attain the required accuracy of solution. The product-form Cholesky factorization (PFCF) approach of Goldfarb and Scheinberg (2004) can handle such cases in a stable way. Instead of computing an explicit Cholesky decomposition, their approach builds the Cholesky matrix through a sequence of updates of an initial factor. The approach is well suited to dealing with matrices of the form $Q = Q_0 + VV^T$, such as the matrix $\Theta_\xi + RP^TPR$ in (12.14). It starts from a decomposition of Q_0 and updates it after adding every rank-1 corrector $V_iV_i^T$ from the matrix VV^T. The approach has been implemented in two solvers, SVM-QP and SVM-QP-presolve (Goldfarb and Scheinberg, 2008), and when applied to medium-scale problems it has demonstrated numerical stability in practice. It is not clear whether the PFCF can be implemented in parallel and this seems to question its applicability to large-scale machine learning problems (see Woodsend and Gondzio, 2009).

Bearing in mind the need to develop parallel implementation to tackle very large problems, Woodsend and Gondzio (2010) have exploited the separable

QP formulations of several support vector machine problems and solved them directly with an interior-point method. Their approach avoids the use of the SMW formula, which could introduce instability but still relies on parallelism-friendly block-matrix operations. We will illustrate the key idea by considering the dual in (12.11).

As we have already observed, the matrix of the quadratic form in this problem, RP^TPR, is dense. However, it is a low-rank matrix and we will exploit its known decomposition. Namely, we define $u = PRz$ and observe that $z^TRP^TPRz = u^Tu$, so the problem can be reformulated as

$$\begin{array}{ll} \min & -e^Tz + \frac{1}{2}u^Tu \\ \text{s.t.} & r^Tz = 0 \\ & PRz - u = 0 \\ & 0 \le z \le \tau e,\ u \text{ free.} \end{array} \tag{12.16}$$

Unlike the dual in (12.11), which had n variables and only one constraint, the new problem has $n+k$ variables and $k+1$ constraints. It is slightly larger than (12.11) but is separable, and the linear equation system to compute the Newton direction

$$\begin{bmatrix} -\Theta_z^{-1} & 0 & r & RP^T \\ 0 & -I_k & 0 & -I_k \\ r^T & 0 & 0 & 0 \\ PR & -I_k & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta u \\ \Delta y_1 \\ \Delta y_2 \end{bmatrix} = \begin{bmatrix} f_z \\ f_u \\ h_1 \\ h_2 \end{bmatrix}, \tag{12.17}$$

has an easy-to-invert $(n+k)\times(n+k)$ diagonal block at position $(1,1)$. After the elimination of this leading block (which corresponds to the elimination of Δz and Δu), we obtain the normal equations

$$\left(\begin{bmatrix} r^T & 0 \\ PR & -I_k \end{bmatrix} \begin{bmatrix} \Theta_z & 0 \\ 0 & I_k \end{bmatrix} \begin{bmatrix} r & RP^T \\ 0 & -I_k \end{bmatrix} \right) \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}, \tag{12.18}$$

which form a system of only $k+1$ linear equations with $k+1$ unknowns. Forming the matrix involved in this system can easily be parallelized. It suffices to split the matrix $P \in \mathcal{R}^{k\times n}$ into blocks $P_i \in \mathcal{R}^{k\times n_i}$, $i = 1, 2, ..., p$ with $\sum_{i=1}^{p} n_i = n$ and gather the partial summation results in the operation

$$PR\Theta_z RP^T = \sum_{i=1}^{p} P_i R_i \Theta_{zi} R_i P_i^T, \tag{12.19}$$

executed on p independent blocks. The separability-exploiting IPM approach of Woodsend and Gondzio (2009) described above has been implemented using OOPS (Gondzio and Grothey, 2006) and tested on very large-scale problems from the PASCAL Challenge,

`http://largescale.first.fraunhofer.de/`.
The implementation is available for research use from
`http://www.maths.ed.ac.uk/ERGO/software.html`.

It is worth mentioning important advantages of the separable QP formulation which distinguish it from two approaches discussed earlier: the one based on the SMW formula (Ferris and Munson, 2003) and the one employing the product form Cholesky factorization (Goldfarb and Scheinberg, 2004, 2008). Unlike the SMW approach which can easily lose accuracy due to multiple inversions of Θ_ξ in (12.14), the separable formulation (12.16) avoids such operations and does not suffer from any instability. In contrast to the PFCF approach, which is inherently sequential, the separable formulation (12.16) allows for an easy parallelization of its major computational tasks.

In summary, interior-point methods provide an attractive alternative to a plethora of other approaches in machine learning. In the context of support vector machines, extensive tests on large instances from the PASCAL Challenge demonstrated (Woodsend and Gondzio, 2009) that IPMs compete very well with the other approaches in terms of CPU time efficiency, and outperform the other approaches in terms of reliability. This is consistent with a general reputation of IPMs as very stable and reliable optimization algorithms.

12.5 Accelerating Interior-Point Methods

Stability and reliability of IPMs have their origin in the use of the Newton method for barrier subproblems and a very "mild" nonlinearity introduced by the logarithmic barrier function. In some applications these features come at too high a price. Numerous optimization problems, including those arising in machine learning, do not have to be solved to a high degree of accuracy. Therefore, fast algorithms are sought which could provide a very rough approximation to the solution of the optimization problem in no time at all. This is one of the reasons why the ML community is so keen on very simple first-order (gradient)-based optimization techniques.

There have been several attempts to improve interior-point methods by reducing the cost of a single iteration. Two of them have been specifically developed for support vector machine applications. They share a common feature and try to guess the activity of inequality constraints, then use only a subset of these constraints when computing Newton directions. Consider again the primal problem in (12.11) and the corresponding Newton equation system in (12.12). The elimination of a large but easily invertible block

$$\begin{bmatrix} -\Theta_\xi^{-1} & I_n \\ I_n & 0 \end{bmatrix},$$

which corresponds to the elimination of $\Delta\xi$ and Δy from the equation system (12.12), produces the small $k \times k$ system

$$(I_k + PR\Theta_\xi RP^T)\Delta w = g_w. \tag{12.20}$$

We have already mentioned that the magnitude of elements of matrix Θ_ξ may vary significantly. Indeed, $\Theta_{\xi j} = \xi_j/\eta_j$, where η_j is the Lagrange multiplier associated with the simple inequality constraint $\xi_j \geq 0$. The complementarity condition requires that $\xi_j\eta_j = 0$ at optimality. IPM uses a perturbed complementarity condition $\xi_j\eta_j = \mu$ and forces ξ_j and η_j to be strictly positive. However, when IPM approaches optimality, one of these variables necessarily has to become very small. Consequently, the ratio ξ_j/η_j goes either to infinity or to zero. Although this might be the source of numerical difficulties when solving systems (12.12) and (12.20), it may also be exploited as a feature to simplify these equations. The matrix in (12.20) can be written in the outer product form

$$M = I_k + \sum_{j=1}^{n} r_j^2 \Theta_{\xi j} p_j p_j^T, \tag{12.21}$$

where $r_j^2 = 1$ (because $r_j \in \{-1, +1\}$) and p_j denotes column j of P, that is, point j in the training set. Since many elements of Θ_ξ are very small, their corresponding outer product contributions to M may be neglected. An approximation of M may be formed as follows:

$$\tilde{M} = I_k + \sum_{\{j:\Theta_{\xi j} \geq \delta\}} \Theta_{\xi j} p_j p_j^T, \tag{12.22}$$

where δ is a prescribed tolerance.

Jung et al. (2008) use information on complementarity products of ξ_j and η_j to determine small elements of $\Theta_{\xi j}$ which may be dropped in the summation. The constraints $r_j(p_j^T w + w_0) \geq 1 - \xi_j$ in the primal problem (12.11), which correspond to indices j associated with small terms $\Theta_{\xi j}$, are likely to be active at optimality. Jung et al. (2008) use the approximation $\tilde{M}$ of M to compute an approximate Newton step. Gertz and Griffin (2009) use the same approximation for a different purpose. They employ a conjugate gradient algorithm to solve (12.20) and use $\tilde{M}$ as a preconditioner of M. In summary, both approaches try to simplify the computations and replace M with its approximation $\tilde{M}$, exploiting obvious savings resulting from replacing the summation over $j \in \{1, 2, ..., n\}$ with the summation over a subset of indices $\{j : \Theta_{\xi j} \geq \delta\}$. However, both approaches have to deal with certain computational overheads: Jung et al. (2008) have to accept a significant increase of the number of iterations resulting from the use of inexact directions, while Gertz and Griffin (2009) need to bear an extra effort of matrix-vector multiplications in the conjugate gradient algorithm.

To conclude the discussion of different acceleration techniques applicable in the IPM context, we need to draw the reader's attention to a recent development of a *matrix-free* variant of the interior-point method (Gondzio, 2010). This approach has been developed with the purpose of solving very large and huge optimization problems for which storage of the problem data alone may already be problematic, and constructing and factoring any of the matrices in the Newton equations (augmented system or normal equations) is expected to be prohibitive. The approach works in a matrix-free regime: Newton equations are never formulated explicitly. Instead, an inexact Newton method (Dembo et al., 1982; Bellavia, 1998) is used, that is, the Newton direction is computed using an iterative approach from the Krylov subspace family. The key feature of the new method which distinguishes it from other matrix-free approaches is that the preconditioner for the iterative method is constructed using the matrix-free regime as well. The method has been described in Gondzio (2010) as a general-purpose one. However, it should be straightforward to specialize it to machine learning problems. We discuss it briefly below.

Consider a problem such as the separable reformulation (12.16) of the dual problem (12.11) and assume that the number of rows, $k + 1$, and the number of columns, $n + k$, are large. One might think of the number of features k being one the order 10^4 or larger, and the number of training points n going into millions or larger. The otherwise very efficient separable formulation (12.16) would demonstrate its limitations for such dimensions because the $(k + 1) \times (k + 1)$ normal equation matrix (12.18) would be excessively expensive to form and factor. Following Woodsend and Gondzio (2010), building the matrix would need $\mathcal{O}(nk^2)$ flops, and factoring it would require an additional $\mathcal{O}(k^3)$ flops. The matrix-free approach (Gondzio, 2010) solves (12.18) *without* forming and factoring the normal equation matrix. It uses the conjugate gradient method, which does not require the normal equation matrix

$$H = \bar{A}\bar{D}\bar{A}^T = \begin{bmatrix} r^T & 0 \\ PR & -I_k \end{bmatrix} \begin{bmatrix} \Theta_z & 0 \\ 0 & I_k \end{bmatrix} \begin{bmatrix} r & RP^T \\ 0 & -I_k \end{bmatrix} \tag{12.23}$$

to be explicitly formulated but needs only to perform matrix-vector multiplications with it. These operations can be executed as a sequence of matrix-vector multiplications with the constraint matrix $\bar{A}$, its transpose, and the diagonal scaling matrix $\bar{D}$. Matrix Θ_z in the diagonal part is always very ill-conditioned, and consequently so is H. The conjugate gradient algorithm will never converge unless an appropriate preconditioner is used. The preconditioner proposed by Gondzio (2010) is a low-rank partial Cholesky factorization of H which is also constructed in the matrix-free regime.

12.6 Conclusions

In this chapter we have discussed the main features of interior-point methods which make them attractive for very large-scale optimization and for application in the machine learning context. IPMs offer an unequalled worst-case complexity: they converge to an ε-accurate solution in $\mathcal{O}(\sqrt{n}\log(1/\varepsilon))$ iterations. In practice they perform much better than the worst-case analysis predicts, and solve linear or convex quadratic problems in a number of iterations which very slowly (logarithmically) grows with the problem dimension. Since machine learning applications are usually very large, IPMs offer an attractive solution methodology for them. We have illustrated the use of IPMs in a particular class of ML problems: support vector machine training. IPMs display excellent stability and robustness, which makes them very competitive in this context. A novel matrix-free variant of the interior-point method is a promising approach for solving very large and huge optimization problems arising in machine learning applications.

Acknowledgment

The author is grateful to Marco Colombo, Pedro Munari and Kristian Woodsend for reading a draft of this chapter and offering useful suggestions which led to its improvement.

12.7 References

S. Bellavia. An inexact interior-point method. *Journal of Optimization Theory and Applications*, 96(1):109–121, 1998.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1995.

M. Colombo and J. Gondzio. Further development of multiple centrality correctors for interior point methods. *Computational Optimization and Applications*, 41(3): 277–305, 2008.

G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, N.J., 1963.

R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19:400–408, 1982.

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

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

J. J. H. Forrest and D. Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, 57:341–374, 1992.

E. M. Gertz and J. Griffin. Using an iterative linear solver in an interior-point method for generating support vector machines. *Computational Optimization and Applications*, 47(3):431–453, 2009.

E. M. Gertz and S. J. Wright. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29(1):58–81, 2003.

P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright. On the projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Mathematical Programming*, 36:183–209, 1986.

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):1–34, 2004.

D. Goldfarb and K. Scheinberg. Numerically stable LDL^T factorizations in interior point methods for convex quadratic programming. *IMA Journal of Numerical Analysis*, 28(4):806–826, 2008.

J. Gondzio. Matrix-free interior point method. Technical Report ERGO-2009-012, School of Mathematics, University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK, April 2010.

J. Gondzio and A. Grothey. Direct solution of linear systems of size 10^9 arising in optimization with interior point methods. In R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Wasniewski, editors, *Parallel Processing and Applied Mathematics*, volume 3911 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, Berlin, 2006.

J. A. J. Hall and K. I. M. McKinnon. Hyper-sparsity in the revised simplex method and how to exploit it. *Computational Optimization and Applications*, 32(3):259–283, 2005.

J. H. Jung, D. O'Leary, and A. Tits. Adaptive constraint reduction for training support vector machines. *Elecronic Transactions on Numerical Analysis*, 31: 156–177, 2008.

N. K. Karmarkar. A new polynomial–time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

V. Klee and G. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities-III*, pages 159–175. Academic Press, 1972.

I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing*, 6 (1):1–14, 1994.

I. Maros. *Computational Techniques of the Simplex Method.* Kluwer Academic, Boston, 2003.

R. E. Marsten, R. Subramanian, M. J. Saltzman, I. J. Lustig, and D. F. Shanno. Interior point methods for linear programming: Just call Newton, Lagrange, and

Fiacco and McCormick! *Interfaces*, 20(4):105–116, 1990.

Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming, Series A*, 103:127–152, 2005.

J. Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988.

R. J. Vanderbei. *Linear Programming: Foundations and Extensions.* Kluwer Academic, Boston, 1st edition, 1997.

K. Woodsend and J. Gondzio. Hybrid MPI/OpenMP parallel linear support vector machine training. *Journal of Machine Learning Research*, 10:1937–1953, 2009.

K. Woodsend and J. Gondzio. Exploiting separability in large-scale linear support vector machine training. *Computational Optimization and Applications*, 2010. Published online October 14 2009.

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