Solution Methods for Quadratic Optimization

Robert M. Freund

April 1, 2004

1 Outline

- Active Set Methods for Quadratic Optimization
- Pivoting Algorithms for Quadratic Optimization
- Four Key Ideas that Motivate Interior Point Methods
- Interior-Point Methods for Quadratic Optimization
- Reduced Gradient Algorithm for Quadratic Optimization
- Some Computational Results

2 Active Set Methods for Quadratic Optimization

In a constrained optimization problem, some constraints will be inactive at the optimal solution, and so can be ignored, and some constraints will be active at the optimal solution. If we knew which constraints were in which category, we could throw away the inactive constraints, and we could reduce the dimension of the problem by maintaining the active constraints at equality.

Of course, in practice, we typically do not know which constraints might be active or inactive at the optimal solution. **Active set methods** are designed to make an intelligent guess of the active set of constraints, and to modify this guess at each iteration.

Herein we describe a relatively simple active-set method that can be used to solve quadratic optimization problems.

Consider a quadratic optimization problem in the format:

$$\begin{array}{lll} \text{QP}: & \text{minimize}_x & \frac{1}{2}x^TQx + c^Tx \\ & \text{s.t.} & Ax = b \\ & & x \geq 0 \ . \end{array} \tag{1}$$

The KKT optimality conditons for QP are as follows:

$$\begin{aligned}
&Ax = b \\
&x \geq 0 \\
&Qx + c - A^T p - s = 0 \\
&s \geq 0 \\
&x_j s_j = 0, \; j = 1, \ldots, n.
\end{aligned}$$

We suppose for this problem that n is very large, but that intuition suggests that very few variables x_j will be non-zero at the optimal solution.

Suppose that we have a feasible solution x, and that we partition the components of x into $x = (x_\beta, x_\eta)$, where $x_\beta \geq 0$ and $x_\eta = 0$. In this partition, the number of elements in β will be small, relative to the value of n.

Using this partition, we can re-write the data for our problem as:

$$Q = \begin{bmatrix} Q_{\beta\beta} & Q_{\beta\eta} \\ Q_{\eta\beta} & Q_{\eta\eta} \end{bmatrix}, \quad A = [A_\beta \; A_\eta], \quad c = \begin{pmatrix} c_\beta \\ c_\eta \end{pmatrix} .$$

Now let us "guess" that the variables $x_j, j \in \eta$ will be zero in the optimal solution. That being the case, we can eliminate the variables x_η from further consideration for the moment. We will then concentrate our efforts on solving the much smaller problem:

$$\begin{aligned}
\mathrm{QP}_\beta : \quad z^*_\beta = \text{minimum}_{x_\beta} \quad & \tfrac{1}{2} x_\beta^T Q_{\beta\beta} x_\beta + c_\beta^T x_\beta \\
\text{s.t.} \quad & A_\beta x_\beta = b \\
& x_\beta \geq 0 \; .
\end{aligned}$$

Let x^*_β be the optimal solution of QP_β, and let p^* be the associated KKT multipliers for the constraints "$A_\beta x_\beta = b$". Also, let s_β denote the associated KKT multipliers on the nonnegativity constraints $x_\beta \geq 0$. Then x^*_β and p^*, s_β will satisfy the KKT optimality conditions for QP$_\beta$, namely:

$$\begin{aligned}
&Q_{\beta\beta}x^*_\beta + c_\beta - A^T_\beta p^* - s_\beta = 0 \\
&A_\beta x^*_\beta = b \\
&x^*_\beta \geq 0, \quad s_\beta \geq 0 \\
&(x^*_\beta)^T s_\beta = 0 \ .
\end{aligned}$$

We can expand x^*_β into the following feasible solution to the original problem:

$$x = (x_\beta, x_\eta) = (x^*_\beta, 0)$$

by setting all of the variables $x_\eta := 0$. This solution will of course be feasible for the original problem, but will it be optimal? The answer lies, of course, in checking the KKT optimality conditions for the original problem using $x = (x^*_\beta, 0)$ and $p = p^*$, which we write as follows:

$$\begin{aligned}
&\begin{bmatrix} Q_{\beta\beta} & Q_{\beta\eta} \\ Q_{\eta\beta} & Q_{\eta\eta} \end{bmatrix} \begin{pmatrix} x^*_\beta \\ 0 \end{pmatrix} + \begin{pmatrix} c_\beta \\ c_\eta \end{pmatrix} - \begin{bmatrix} A^T_\beta \\ A^T_\eta \end{bmatrix} p^* = \begin{pmatrix} s_\beta \\ s_\eta \end{pmatrix} \geq 0 \\
&[A_\beta \ \ A_\eta] \begin{pmatrix} x^*_\beta \\ 0 \end{pmatrix} = b \\
&(x^*_\beta, 0) \geq 0 \\
&(x^*_\beta)^T s_\beta = 0, \quad 0^T s_\eta = 0 \ .
\end{aligned}$$

Here we see that all of these conditions are automatically satisfied except for the condition that $s_\eta \geq 0$. We remark that s_η is defined above to be:

$$s_\eta := Q_{\eta\beta}x^*_\beta + c_\eta - A^T_\eta p^* \ .$$

We therefore have:

Proposition 2.1 *If $s_\eta \ge 0$, then $(x^*_\beta, 0)$ is an optimal solution to (1).*

Proof: Since x^*_β is an optimal solution of QP_β, then the KKT multipliers p^* and s_β satisfy

$$\begin{aligned}
&Q_{\beta\beta}x^*_\beta + c_\beta - A^T_\beta p^* - s_\beta = 0\\
&A_\beta x^*_\beta = b\\
&x^*_\beta \ge 0, \quad s_\beta \ge 0\\
&(x^*_\beta)^T s_\beta = 0 \ .
\end{aligned}$$

If furthermore $0 \le s_\eta := Q_{\eta\beta}x^*_\beta + c_\eta - A^T_\eta p^*$ we have that the vector $(x^*_\beta, 0)$ satisfies

$$\begin{aligned}
&\begin{bmatrix} Q_{\beta\beta} & Q_{\beta\eta} \\ Q_{\eta\beta} & Q_{\eta\eta} \end{bmatrix} \begin{pmatrix} x^*_\beta \\ 0 \end{pmatrix} + \begin{pmatrix} c_\beta \\ c_\eta \end{pmatrix} - \begin{bmatrix} A^T_\beta \\ A^T_\eta \end{bmatrix} p^* = \begin{pmatrix} s_\beta \\ s_\eta \end{pmatrix} \ge 0\\
&[A_\beta \ \ A_\eta] \begin{pmatrix} x^*_\beta \\ 0 \end{pmatrix} = b\\
&(x^*_\beta, 0) \ge 0\\
&(x^*_\beta)^T s_\beta = 0, \quad 0^T s_\eta = 0 \ .
\end{aligned}$$

which are precisely the optimality conditions for (1). ∎

Just as importantly, if $s_j < 0$ for some $j \in \eta$, then the optimal solution to the expanded problem $\text{QP}_{\beta\cup\{j\}}$ will have a strictly better objective function value than that of QP_β.

Proposition 2.2 *If $s_j < 0$ for some $j \in \eta$, then $z_{\beta\cup\{j\}} < z_\beta$.*

We defer the proof of this proposition to the end of this section.

This all suggests the following active-set algorithm for solving our original problem QP:

0. Define some partition of the indices, $\{1, \ldots, n\} = \beta \cup \eta$.

1. Solve the reduced problem

$$\begin{array}{lll} \mathrm{QP}_\beta: & z^*_\beta = \mathrm{minimum}_{x_\beta} & \frac{1}{2} x_\beta^T Q_{\beta\beta} x_\beta + c_\beta^T x_\beta \\ & \text{s.t.} & A_\beta x_\beta = b \\ & & x_\beta \geq 0 \ , \end{array}$$

   obtaining the optimal solution x^*_β and optimal KKT multiplier p^* on the equations "$A_\beta x_\beta = b$".

2. Compute

$$s_\eta := Q_{\eta\beta} x^*_\beta + c_\eta - A_\eta^T p^* \ .$$

   If $s_\eta \geq 0$, STOP. In this case, $(x^*_\beta, 0)$ is optimal for (1). Otherwise if $s_j < 0$ for some $j \in \eta$, then set $\beta \leftarrow \beta \cup \{j\}$, $\eta := \eta \setminus \{j\}$, and go to 1.

In practice we might add more than one index j to the set β at each iteration. Some active set methods add all indices j for which $s_j < 0$.

Also, in order to keep the set β from becoming too large, we might remove indices i from β if x_i has been zero in the optimal solutions of the reduced problems for a large number of previous iterations.

There are also very many other variants of this type of method.

Proof of Proposition 2.2: For ease of notation we will drop the subscript β from the subproblem solved in step 1 of the algorithm. Let us call $\bar{x}$ the optimal solution of this problem, together with KKT multipliers $\bar{p}$ and $\bar{s}$. These all satisfy the KKT conditions:

$$\begin{array}{l} Q\bar{x} + c - A^T \bar{p} - \bar{s} = 0 \\ A\bar{x} = b \\ \bar{x} \geq 0, \quad \bar{s} \geq 0, \quad \bar{x}^T \bar{s} = 0 \ . \end{array}$$

We separate the indices of x into sets such that $\bar{x}_1 > 0$ and $\bar{x}_2 = 0$ and we separate the data for the problem as follows:

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix}, \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad A = [A_1 \ \ A_2] \ .$$

We then write the above system as:

$$Q_{11}\bar{x}_1 + c_1 - A_1^T \bar{p} = 0$$

$$Q_{21}\bar{x}_1 + c_2 - A_2^T \bar{p} - \bar{s}_2 = 0$$

$$A_1 \bar{x}_1 = b$$

$$\bar{x}_1 > 0, \quad \bar{s}_2 \geq 0 \ .$$

We now consider a new variable θ to add to this system. The data associated with this variable is $q, \delta, \hat{c}, a$, such that:

$$Q \to \begin{bmatrix} Q & q \\ q^T & \delta \end{bmatrix}, \quad c \to \quad \begin{pmatrix} c \\ \hat{c} \end{pmatrix}, \quad A \to [A \ \ a]$$

Since this variable by supposition has a "reduced cost" that is negative, then the new data satisfies:

$$q^T \bar{x} + \hat{c} - a^T \bar{p} = -\varepsilon < 0 \ .$$

We will now show that this implies that there exists a feasible descent direction. The objective function of our problem and the formulas for the gradients of the objective function are:

$$\begin{aligned} f(x,\theta) &= \frac{1}{2}(x^T, \theta) \begin{bmatrix} Q & q \\ q^T & \delta \end{bmatrix} \begin{pmatrix} x \\ \theta \end{pmatrix} + (c^T, \hat{c}) \begin{pmatrix} x \\ \theta \end{pmatrix} \\ \nabla f(x,\theta) &= \begin{bmatrix} Q & q \\ q^T & \delta \end{bmatrix} \begin{pmatrix} x \\ \theta \end{pmatrix} + \begin{pmatrix} c \\ \hat{c} \end{pmatrix} \\ \nabla f(\bar{x}, 0) &= \begin{bmatrix} Q\bar{x} + c \\ q^T \bar{x} + \hat{c} \end{bmatrix} = \begin{bmatrix} Q_{11}\bar{x}_1 + c_1 \\ Q_{21}\bar{x}_1 + c_2 \\ q^T \bar{x} + \hat{c} \end{bmatrix} \ . \end{aligned}$$

Invoking a nondegeneracy assumption, we assume that there exists a vector d_1 for which $A_1 d_1 + a = 0$. Then, since $\bar{x}_1 > 0$ we have that $(x_1, x_2, \theta) = (\bar{x}_1 + \alpha d_1, 0, \alpha) \geq 0$, for $\alpha > 0$ and small. Next we compute:

$$\begin{aligned}
\nabla f(\bar{x}, 0)^T (d_1, 0, 1) &= (d_1^T, 0, 1) \begin{bmatrix} Q_{11}\bar{x}_1 + c_1 \\ Q_{21}\bar{x}_1 + c_2 \\ q^T \bar{x} + \hat{c} \end{bmatrix} \\
&= (A_1^T \bar{p})^T d_1 + q^T \bar{x} + \hat{c} \\
&= -a^T \bar{p} + q^T \bar{x} + \hat{c} = -\varepsilon < 0 \ .
\end{aligned}$$

Therefore $(d_1, 0, 1)$ is a feasible descent direction, and so the optimal objective function value of the expanded problem is strictly less than that of the current problem. ■

3 Pivoting Algorithms for Quadratic Optimization

To facilitate the presentation, we will consider the following format for QP:

$$\begin{array}{lll}
\text{QP}: & \text{minimize}_x & \frac{1}{2} x^T Q x + c^T x \\
& \text{s.t.} & Ax \geq b \qquad (\pi) \\
& & x \geq 0 \qquad (y)
\end{array} \tag{2}$$

Let us first look at the KKT optimality conditions for (2):

$$
\begin{array}{ll}
s := Ax - b \geq 0 & \\
x \geq 0 & \\
Qx + c - A^T\pi - y = 0 & \\
\pi \geq 0 & \\
y \geq 0 & \\
\pi_i s_i = 0 & i = 1, \ldots, m \\
y_j x_j = 0 & j = 1, \ldots, n.
\end{array}
$$

We can write the first five conditions in the form of a tableau of linear constraints in the nonnegative variables (y, s, x, π):

y	s	x	π	Relation	RHS
I	0	$-Q$	A^T	$=$	c
0	I	$-A$	0	$=$	$-b$

and keeping in mind the nonnegativity conditions:

$$y \geq 0 \quad s \geq 0 \quad x \geq 0 \quad \pi \geq 0$$

and the complementarity conditions:

$$y_j x_j = 0, \quad j = 1, \ldots, n, \qquad s_i \pi_i = 0, \quad i = 1, \ldots, m.$$

This tableau has exactly $(n + m)$ rows and $2(n + m)$ columns, that is, there are twice as many columns as rows. Therefore a basic feasible solution of this tableau will have one half of all columns in the basis.

We can find a basic feasible solution to this tableau (which has *no* objective function row by the way!) by applying Phase I of the simplex algorithm. However, what we really want to do is to find basic feasible solution that is also "complementary" in the following sense:

- Either y_j is in the basis or x_j is in the basis, but not both, $\quad j = 1, \dots, n$,
- Either s_i is in the basis or π_i is in the basis, but not both, $\quad i = 1, \dots, m$.

Such a basic feasible solution will have the nice property that either $y_j = 0$ or $x_j = 0$, $j = 1, \dots, n$, and also $s_i = 0$ or $\pi_i = 0$, $i = 1, \dots, m$. That is, a complementary basic feasible solution (CBFS) will satisfy the KKT conditions and so will solve QP.

One strategy for finding a CBFS is to add a nonnegative parameter θ and solve the following tableau:

y	s	x	π	Relation	RHS
I	0	$-Q$	A^T	$=$	$c + \theta e$
0	I	$-A$	0	$=$	$-b + \theta e$

(Here e is the vector of ones of any appropriate dimension.) Now notice that for θ sufficiently large, that the RHS is nonnegative and so the complementary basic solution:

$$\begin{pmatrix} y \\ s \\ x \\ \pi \end{pmatrix} = \begin{pmatrix} c + \theta e \\ -b + \theta e \\ 0 \\ 0 \end{pmatrix}$$

is a complementary basic feasible solution. The strategy then is try to start from this solution, and to drive θ to zero by a sequence of pivots that maintain a complementary basic feasible solution all the while. As it turns out, it is easy to derive pivot rules for such a strategy. The resulting algorithm is called the "complementary pivot algorithm". It was originally developed by George Dantzig and Richard Cottle in the 1960s, and analyzed and extended by Carl Lemke, also in the 1960s.

We will not go over the specific pivot rules of the complementary pivot algorithm. They are simple but technical, and are easy to implement. What can we say about convergence of the complementary pivot algorithm? Just like the simplex method for linear optimzation, one can prove that the complementary pivot algorithm must converge in a finite number of pivots (whenever the matrix Q is SPSD).

Regarding computational performance, notice that the tableau generally has twice the number of constraints and twice the number of variables as a regular linear optimization problem would have. This is because the primal and the dual variables and constraints are written down simultaneously in the tableau. This leads to the general rule that a complementary pivot algorithm will take roughly eight times as long to solve a quadratic program via pivoting as it would take the simplex method to solve a linear optimization problem of the same size as the primal problem alone.

Up until recently, most software for quadratic optimization was based on the complementary pivot algorithm. However, nowadays most software for quadratic optimization uses interior-point methods, which we will soon discuss.

4 Four Key Ideas that Motivate Interior Point Methods

We will work in this section with quadratic optimization in the following format:

$$\begin{array}{lll} \mathrm{QP}: & \mathrm{minimize}_x & \frac{1}{2}x^TQx + c^Tx \\ & \mathrm{s.t.} & Ax = b \\ & & x \geq 0\ . \end{array} \tag{3}$$

4.1 A barrier function

The feasible region of (QP) is the set

$$\mathcal{F} := \{x \mid Ax = b, x \geq 0\}\ .$$

The boundary of $\mathcal{F}$ will be where one or more of the components of x is equal to zero. The boundary is composed of linear pieces, and is generally arranged in an exponentially complex manner. The simplex method takes explicit advantage of the linearity of the boundary, but can take a long time to solve a problem if there are exponentially many extreme points and edges to traverse. For this reason, we will take the curious step of introducing a *barrier function* $B(x)$. A barrier function is any (convex) function defined on the interior of $\mathcal{F}$ and whose values go to infinity as the boundary is approached. That is:

- $B(x) < +\infty$ for all $x \in \text{int}\mathcal{F}$, and
- $B(x) \to \infty$ as $x_j \to 0$.

The idea in a barrier method is to dissuade points x from ever approaching the boundary of the feasible region. For a fixed value of $\mu > 0$, we consider solving:

$$\begin{array}{llc} \text{QP}_\mu: & \text{minimize}_x & \frac{1}{2}x^TQx + c^Tx + \mu B(x) \\ & \text{s.t.} & Ax = b \\ & & x > 0\ . \end{array} \tag{4}$$

Let $x(\mu)$ denote the optimal solution of QP_μ. For large values of μ, the optimal solution $x(\mu)$ will be more concerned about staying away from the boundary than in optimizing the original quadratic objective function. However, for small values of μ, the optimal solution $x(\mu)$ will be ever more concerned about optimizing the original quadratic objective function while simultaneously not getting too close to the boundary.

Interior-point methods use the barrier function:

$$B(x) = -\sum_{j=1}^{n} \ln(x_j)\ .$$

This function is convex, and approaches $+\infty$ as any x_j approaches zero.

4.2 Staying Centered relative to the Objective Function Value

Suppose that $\bar{x}$ is a the current iterate of an algorithm for solving (QP). If $\bar{x}$ is very far from the boundary of the feasible region $\mathcal{F}$, then no matter what direction we wish to go at the next iteration, we will have plenty of room to take a step, see Figure 1. However, since the optimal solution will generally lie on the boundary of the feasible region, we will want to keep the algorithm iterates as centered as possible for their objective function values. In the figure, point a is close to the boundary, point b is far from the boundary, and points c and d are far from the boundary relative to their objective function level sets.

Now consider the barrier version of our quadratic problem with the logarithmic barrier:

$$\begin{array}{lll} \mathrm{BP}_\mu: & \text{minimize}_x & \frac{1}{2}x^TQx + c^Tx - \mu \sum_{j=1}^{n} \ln(x_j) \\ & \text{s.t.} & Ax = b \\ & & x > 0\ . \end{array}$$

The objective function here will minimize the barrier term while also accounting for the original quadratic objective function. In this way, the solutions $x(\mu)$ of BP_μ will be nicely centered relative to other feasible values $x \in \mathcal{F}$ with as good or better objective function values. Therefore the solutions $x(\mu)$ will be relatively far from the boundary for their objective function values.

4.3 Treating the KKT Conditions of the Barrier Problem as a Nonlinear Equation System

For a general barrier function $B(x)$, our parameterized barrier problem is:

$$\begin{array}{lll} \mathrm{QP}_\mu: & \text{minimize}_x & \frac{1}{2}x^TQx + c^Tx + \mu B(x) \\ & \text{s.t.} & Ax = b \\ & & x > 0\ . \end{array} \tag{5}$$

Because the positivity conditions $x > 0$ will never be binding, the optimality conditions for this problem are:

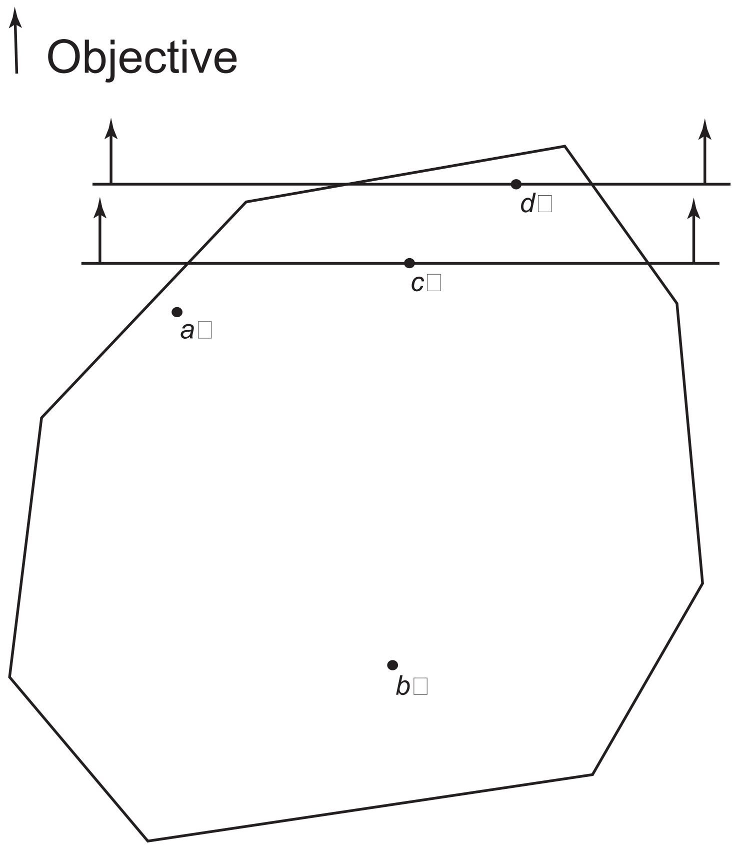

Figure 1: Point a is close to the boundary, point b is far from the boundary. Points c and d are far from the boundary relative to their objective function level sets.

$$Ax = b$$

$$Qx + c - \mu \nabla B(x) - A^T p = 0$$

$$x > 0 \ .$$

This is a nonlinear equation system with $n + m$ equations in the $n + m$ variables (x, p). A generic algorithm for this system can be described as follows:

0. Start with some $\bar{x} > 0$ and some $\bar{p}$.
1. If $\|A\bar{x} - b\| \leq \epsilon$ and $\|Q\bar{x} + c - \mu \nabla B(\bar{x}) - A^T \bar{p}\| \leq \epsilon$ for some small tolerance $\epsilon > 0$, STOP. Otherwise go to Step 2.
2. Use some method, perhaps Newton's method, to generate a desired step $d = (d_x, d_p)$.
3. Compute a step-size that will maintain the positivity of the iterate value of x. Set $r = 0.99$, and compute:

$$\theta \quad = \quad \min \left\{ 1, \ r \min_{\Delta x_j < 0} \left\{ \frac{\bar{x}_j}{-\Delta x_j} \right\} \right\} \ .$$

4. Re-set $(\bar{x}, \bar{p}) \leftarrow (\bar{x} + \theta d_x, \bar{p} + \theta d_p)$, and go to Step 1.

In this method, we check to see if we satisfy our nonlinear equations (to within some tolerance) in Step 1. We generate a presumably good direction $d = (d_x, d_p)$ in Step 2. Although we would like to re-set $(\bar{x}, \bar{p}) \leftarrow (\bar{x} + d_x, \bar{p} + d_p)$, we must ensure that our iterate values of x remain strictly positive. This is accomplished in Step 3. Note that the value $r = 0.99$ could be set to any positive value strictly less than 1.0 .

4.4 Newton's Method

The last motivating idea in interior-point methods is Newton's method. In its various forms, Newton's method is a method for solving nonlinear equations by linearizing the equations at each iteration. Suppose that we wish to solve the equation system:

$$\begin{array}{rcl} h_1(x) & = & 0 \\ \vdots & \vdots & \vdots \\ h_n(x) & = & 0 \end{array} \tag{6}$$

where $x = (x_1, \ldots, x_n)$. We can write this succinctly as:

$$h(x) = 0$$

where $h(x) = (h_1(x), \ldots, h_n(x))$. Let $\bar{x}$ denote the current iterate value. Then from Taylor's theorem, we have:

$$h(\bar{x} + \Delta x) \approx \tilde{h}(\bar{x} + \Delta x) := h(\bar{x}) + [H(\bar{x})]\, \Delta x \ , \tag{7}$$

where $H(x)$ is the Jacobian matrix:

$$H_{ij}(x) := \frac{\partial h_i(x)}{\partial x_j} \ .$$

Here the function $\tilde{h}(\bar{x} + \Delta x)$ is the linear approximation of $h(x)$ around the point $x = \bar{x}$. We can try to solve the system $h(\bar{x} + \Delta x) = 0$ by instead solving $\tilde{h}(\bar{x} + \Delta x) = 0$, which from (7) is:

$$h(\bar{x}) + [H(\bar{x})]\, \Delta x = 0 \ . \tag{8}$$

Solving (8) for Δx, we obtain:

$$d := \Delta x = - [H(\bar{x})]^{-1}\, h(\bar{x}) \ . \tag{9}$$

The direction d is called the *Newton direction* or the *Newton step*. In a pure version of Newton's method, we would set the next iteration value to be

$$x_{\text{new}} = \bar{x} + d \ .$$

Newton's method has remarkably fast convergence properties if the iterates lie near to the solution of the system. However, if the iterates are very far away from the solution of the system, Newton's method typically does not have particularly good convergence properties.

5 Interior-Point Methods for Quadratic Optimization

5.1 The Central Path Equation System

For this section it will be most convenient to work with quadratic optimization in the following format:

$$\begin{array}{lll} \mathrm{QP}: & \mathrm{minimize}_x & \frac{1}{2}x^TQx + c^Tx \\ & \mathrm{s.t.} & Ax = b \\ & & x \geq 0 \ . \end{array} \tag{10}$$

Using the usual Lagrange duality constructions, we can derive the following dual problem:

$$\begin{array}{lll} \mathrm{DP}: & \mathrm{maximize}_{p,s,x} & b^Tp - \frac{1}{2}x^TQx \\ & \mathrm{s.t.} & A^Tp + s - Qx = c \\ & & s \geq 0 \ . \end{array}$$

The following weak duality result follows by direct algebraic manipulation of the feasibility conditions:

Proposition 5.1 *If x if feasible for (QP) and (p, s, x) is feasible for (DP), then the duality gap is:*

$$\frac{1}{2}x^TQx + c^Tx - \left(b^Tp - \frac{1}{2}x^TQx\right) = s^Tx \ .$$

The KKT optimality conditons for QP are as follows:

$$Ax = b$$

$$x \geq 0$$

$$A^T p + s - Qx = c$$

$$s \geq 0$$

$$x_j s_j = 0, \; j = 1, \ldots, n.$$

The strong duality result is as follows:

Proposition 5.2 *If x if feasible for (QP) and (p, s, x) is feasible for (DP), then x if optimal for (QP) and (p, s, x) is optimal for (DP) if and only if:*

$$s^T x = 0 \; .$$

Before we move on, let us introduce some notation that is used in interior-point methods. For vectors x and s, we denote:

$$X = \begin{pmatrix} x_1 & & 0 \\ & \ddots & \\ 0 & & x_n \end{pmatrix} \quad , \quad S = \begin{pmatrix} s_1 & & 0 \\ & \ddots & \\ 0 & & s_n \end{pmatrix} \quad , \quad e = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \; .$$

Therefore, for example,

$$Xe = x, \quad Se = s, \quad XSe = \begin{pmatrix} x_1 s_1 \\ \vdots \\ x_n s_n \end{pmatrix} \; .$$

Now let us relax the KKT conditions above, by picking a value of $\mu > 0$ and instead solve:

$$Ax = b, \quad x > 0$$

$$A^T p + s - Qx = c, \quad s > 0$$

$$x_j s_j = \mu, \; j = 1, \ldots, n.$$

With our interior-point notation we can write this system as:

$$Ax = b, \quad x > 0$$

$$A^T p + s - Qx = c, \quad s > 0$$

$$XSe = \mu e \ .$$

A solution $(x, p, s) = (x_\mu, p_\mu, s_\mu)$ to this system for some value μ is called a point on the *central path* or *central trajectory* of the problem.

Proposition 5.3 *If (x, p, s) is on the central path for some value $\mu > 0$, then the duality gap is*

$$\frac{1}{2}x^T Qx + c^T x - \left(b^T p - \frac{1}{2}x^T Qx\right) = s^T x = n\mu \ .$$

The proof of this proposition is a straightforward extension of the algebra of the system. The value of this proposition is that it shows that if we solve the system for smaller values of μ, then our duality gap will go to zero, and we will approximately solve our original problem.

We can also derive the central path by considering the following logarithmic barrier problem associated with QP for a given value of μ:

$$\begin{array}{lll} \mathrm{BP}_\mu : & \text{minimize}_x & \frac{1}{2}x^T Qx + c^T x - \mu \sum\limits_{j=1}^{n} \ln(x_j) \\ & \text{s.t.} & Ax = b \\ & & x > 0 \ . \end{array}$$

The KKT conditions for this logarithmic barrier problem are:

$$Ax = b, \quad x > 0$$

$$Qx + c - \mu X^{-1}e - A^T p = 0 \ .$$

If we then define $s := \mu X^{-1}e > 0$ and rearrange the above system, we obtain:

$$\begin{aligned}
&Ax = b, \quad x > 0\\
&A^T p + s - Qx = c\\
&XSe = \mu e\ .
\end{aligned}$$

Since $x > 0$ implies that $s > 0$, this is exactly the same system of equations we had derived earlier that define the central path.

5.2 The Newton Equation System

Suppose that we have a current "point" (actually a triplet) $(\bar{x}, \bar{p}, \bar{s})$ for which $\bar{x} > 0$ and $\bar{s} > 0$, and we are given a value of $\mu > 0$. We want to compute a direction $(\Delta x, \Delta p, \Delta s)$ so that $(\bar{x}+\Delta x, \bar{p}+\Delta p, \bar{s}+\Delta s)$ approximately solves the central path equations. We set up the system:

$$\begin{aligned}
&(1) \quad A(\bar{x} + \Delta x) = b\\
&(2) \quad -Q(\bar{x} + \Delta x) + A^T(\bar{p} + \Delta p) + (\bar{s} + \Delta s) = c\\
&(3) \quad (\bar{X} + \Delta X)(\bar{S} + \Delta S)e = \mu e\ .
\end{aligned}$$

If we linearize this system of equations and rearrange the terms, we obtain the Newton equations for the current point $(\bar{x}, \bar{p}, \bar{s})$:

$$\begin{aligned}
&(1) \quad A\Delta x = b - A\bar{x} =: r_1\\
&(2) \quad -Q\Delta x + A^T\Delta p + \Delta s = c + Q\bar{x} - A^T\bar{p} - \bar{s} =: r_2\\
&(3) \quad \bar{S}\Delta x + \bar{X}\Delta s = \mu e - \bar{X}\bar{S}e =: r_3
\end{aligned}$$

We refer to the solution $(\Delta x, \Delta p, \Delta s)$ to the above system as the *Newton direction* at the point $(\bar{x}, \bar{p}, \bar{s})$.

5.3 The Step-sizes

Given our current point $(\bar{x}, \bar{p}, \bar{s})$ and a given value of μ, we compute the Newton direction $(\Delta x, \Delta p, \Delta s)$ and we update our variables by choosing a step-size θ, to obtain new values:

$$(\tilde{x}, \tilde{p}, \tilde{s}) \leftarrow (\bar{x}, \bar{p}, \bar{s}) + \theta(\Delta x, \Delta p, \Delta s) \ .$$

In order to ensure that $\tilde{x} > 0$ and $\tilde{s} > 0$, we choose a value of r satisfying $0 < r < 1$ ($r = 0.99$ is a common value in practice), and determine θ as follows:

$$\begin{aligned} \theta_P &= \min\left\{1,\ r \min_{\Delta x_j < 0} \left\{ \frac{\bar{x}_j}{-\Delta x_j} \right\}\right\} \\ \theta_D &= \min\left\{1,\ r \min_{\Delta s_j < 0} \left\{ \frac{\bar{s}_j}{-\Delta s_j} \right\}\right\} \ . \end{aligned}$$

We then set the step-size to be:

$$\theta := \min\{\theta_P, \theta_D\} \ .$$

5.4 Decreasing the Path Parameter μ

We also want to shrink μ at each iteration, in order to (hopefully) shrink the duality gap. The current iterate is $(\bar{x}, \bar{p}, \bar{s})$, and the current values satisfy:

$$\mu \approx \frac{\bar{x}^T \bar{s}}{n}$$

We then re-set μ to

$$\mu \leftarrow \left(\frac{1}{10}\right)\left(\frac{\bar{x}^T \bar{s}}{n}\right) \ ,$$

where “10” is user-specified.

5.5 The Stopping Criterion

We typically declare the problem solved when it is "almost" primal feasible, "almost" dual feasible, and there is "almost" no duality gap. We set our tolerance ϵ to be a small positive number, typically $\epsilon = 10^{-8}$, for example, and we stop when:

$$\begin{aligned} &(1) \quad \|A\bar{x} - b\| \leq \epsilon \\ &(2) \quad \|-Q\bar{x} + A^T\bar{p} + \bar{s} - c\| \leq \epsilon \\ &(3) \quad \bar{s}^T\bar{x} \leq \epsilon \end{aligned}$$

5.6 The Full Interior-Point Algorithm

a) Given (x^0, p^0, s^0) satisfying $x^0 > 0$, $s^0 > 0$, and $\mu^0 > 0$, and r satisfying $0 < r < 1$, and $\epsilon > 0$. Set $k \leftarrow 0$.

b) Test stopping criterion. Check if:

$$\begin{aligned} &(1) \quad \|Ax^k - b\| \leq \epsilon \\ &(2) \quad \|-Qx^k + A^Tp^k + s^k - c\| \leq \epsilon \\ &(3) \quad (s^k)^Tx^k \leq \epsilon \ . \end{aligned}$$

If so, STOP. If not, proceed.

c) Set $\mu \leftarrow \left(\frac{1}{10}\right)\left(\frac{(x^k)^T(s^k)}{n}\right)$

d) Solve the Newton equation system:

$$\begin{aligned} &(1) \quad A\Delta x = b - Ax^k =: r_1 \\ &(2) \quad -Q\Delta x + A^T\Delta p + \Delta s = c + Qx^k - A^Tp^k - s^k =: r_2 \\ &(3) \quad S^k\Delta x + X^k\Delta s = \mu e - X^kS^ke =: r_3 \end{aligned}$$

e) Determine the step-size:

$$\begin{aligned}
\theta_P &= \min\left\{1,\ r \min_{\Delta x_j < 0}\left\{\frac{x_j^k}{-\Delta x_j}\right\}\right\} \\
\theta_D &= \min\left\{1,\ r \min_{\Delta s_j < 0}\left\{\frac{s_j^k}{-\Delta s_j}\right\}\right\} \\
\theta &= \min\{\theta_P, \theta_D\}\ .
\end{aligned}$$

f) Update values:

$$(x^{k+1}, p^{k+1}, s^{k+1}) \quad \leftarrow \quad (x^k, p^k, s^k) + \theta(\Delta x, \Delta p, \Delta s)\ .$$

Re-set $k \leftarrow k+1$ and return to (b).

5.7 Solving the Newton Equations

We now show how to manipulate the Newton equations to yield some formulas for the solution of the Newton equation system. Given our current point $(\bar{x}, \bar{p}, \bar{s})$ for which $\bar{x} > 0$ and $\bar{s} > 0$, and a value of $\mu > 0$, we wish to solve the Newton equations below for $(\Delta x, \Delta p, \Delta s)$:

(1) $A\Delta x = b - A\bar{x} =: r_1$

(2) $-Q\Delta x + A^T\Delta p + \Delta s = c + Q\bar{x} - A^T\bar{p} - \bar{s} =: r_2$

(3) $\bar{S}\Delta x + \bar{X}\Delta s = \mu e - \bar{X}\bar{S}e =: r_3$

Pre-multiplying equation (2) by $\bar{X}$, we obtain the following string of equations that derive from one another:

$$-\bar{X}Q\Delta x + \bar{X}A^T\Delta p + \bar{X}\Delta s = \bar{X}r_2$$

$$-\bar{X}Q\Delta x + \bar{X}A^T\Delta p - \bar{S}\Delta x + r_3 = \bar{X}r_2$$

$$-\left(\bar{X}Q + \bar{S}\right)\Delta x + \bar{X}A^T\Delta p = \bar{X}r_2 - r_3$$

$$-\Delta x + \left(\bar{X}Q + \bar{S}\right)^{-1}\bar{X}A^T\Delta p = \left(\bar{X}Q + \bar{S}\right)^{-1}\left(\bar{X}r_2 - r_3\right)$$

$$-r_1 + A\left(\bar{X}Q + \bar{S}\right)^{-1}\bar{X}A^T\Delta p = A\left(\bar{X}Q + \bar{S}\right)^{-1}\left(\bar{X}r_2 - r_3\right)$$

$$A\left(\bar{X}Q + \bar{S}\right)^{-1}\bar{X}A^T\Delta p = A\left(\bar{X}Q + \bar{S}\right)^{-1}\left(\bar{X}r_2 - r_3\right) + r_1$$

Therefore Δp is determined by solving the last system above, or equivalently, by the formula:

$$\Delta p = \left(A\left(\bar{X}Q + \bar{S}\right)^{-1}\bar{X}A^T\right)^{-1}\left(A\left(\bar{X}Q + \bar{S}\right)^{-1}\left(\bar{X}r_2 - r_3\right) + r_1\right) \ .$$

Once Δp has been computed, we can compute Δx and Δs as follows:

$$\Delta x = \left(\bar{X}Q + \bar{S}\right)^{-1}\left(r_3 - \bar{X}r_2 + \bar{X}A^T\Delta p\right)$$

$$\Delta s = Q\Delta x + r_2 - A^T\Delta p \ .$$

Note that if Q is very sparse or nicely structured (for example, if Q is diagonal), then the bulk of the work above lies in forming the matrix:

$$A\left(\bar{X}Q + \bar{S}\right)^{-1}\bar{X}A^T$$

and then solving a system of equations with this matrix. In particular when Q is diagonal, then the sparsity pattern of this matrix will be the same as the sparsity pattern of AA^T.

5.8 Remarks on Interior-Point Methods

- The algorithm just described is almost exactly what is used in commercial interior-point method software.
- A typical interior-point code will solve a linear or quadratic optimization problem in 25-80 iterations, regardless of the dimension of the problem.
- There is a very "deep" theory of convergence for interior-point methods, when the constant "10" is replaced by the number

$$\frac{2+4\sqrt{n}}{1+4\sqrt{n}}$$

- Interior-point methods are still an area of very active research.
- These days, interior-point methods have been extended to allow for the solution of problems with convex quadratic constraints:

$$\begin{array}{lll} \mathrm{QCQP}: & \min_x & c^T x + \frac{1}{2}x^T Q x \\ \mathrm{s.t.} & & a_i^T x + \frac{1}{2}x^T Q_i x \le b_i \ , \quad i = 1, \ldots, m \end{array}$$

6 Reduced Gradient Algorithm for Nonlinear Optimization

The reduced gradient algorithm is designed to solve a problem of the form:

$$\begin{array}{lll} \mathrm{NLP}: & \mathrm{minimize}_x & f(x) \\ & \mathrm{s.t.} & Ax = b \\ & & x \ge 0 \ . \end{array} \tag{11}$$

In the case of a suitably formatted quadratic problem,

$$f(x) = \frac{1}{2}x^T Q x + c^T x$$

and

$$\nabla f(x) = Qx + c \ .$$

We will make the following assumptions regarding bases for the system "$Ax = b, x \geq 0$":

- For any basis $\beta \subset \{1, \ldots, n\}$ with $|\beta| = m$, A_β is nonsingular.
- For any basic feasible solution $(x_\beta, x_\eta) = (A_\beta^{-1} b, 0)$ we have $x_\beta > 0$, where here $\eta := \{1, \ldots, n\} \setminus \beta$.

The reduced gradient algorithm tries to mimic certain aspects of the simplex algorithm.

At the beginning of an iteration of the reduced gradient algorithm, we have a feasible solution $\bar{x}$ for our problem NLP. Pick a basis β for which $\bar{x}_\beta > 0$. Remember that such a basis must exist by our assumptions, but that such a basis might not be uniquely determined by the current solution $\bar{x}$. Set $\eta := \{1, \ldots, n\} \setminus \beta$. Now invoke the equations of our system, which state that:

$$A_\beta x_\beta + A_\eta x_\eta = b$$

for any $x = (x_\beta, x_\eta)$, and so we can write:

$$x_\beta = A_\beta^{-1}(b - A_\eta x_\eta) \ .$$

Since the basic variables are uniquely determined by the nonbasic variables above, we can substitute these into the formula for the objective function:

$$f(x) = f(x_\beta, x_\eta) = f(A_\beta^{-1}(b - A_\eta x_\eta), x_\eta) \ .$$

We therefore can define our *reduced objective function* to be:

$$h(x_\eta) = f(A_\beta^{-1}(b - A_\eta x_\eta), x_\eta) \ .$$

The gradient of the reduced objective function is called the *reduced gradient*, and has the following formula:

$$\nabla h(x_\eta) = \nabla f_\eta(A_\beta^{-1}(b - A_\eta x_\eta), x_\eta) - A_\eta^T A_\beta^{-T} \nabla f_\beta(A_\beta^{-1}(b - A_\eta x_\eta), x_\eta) \ ,$$

which evaluates at $\bar{x}$ to be:

$$\nabla h(\bar{x}_\eta) = \nabla f_\eta(\bar{x}) - A_\eta^T A_\beta^{-T} \nabla f_\beta(\bar{x}) \ .$$

Let us re-name the reduced gradient $\bar{r} = \bar{r}_\eta$ for notational convenience. The reduced gradient can then be used to define a new direction d as follows: for $j \in \eta$, we define:

$$d_j := \begin{cases} -\bar{r}_j & \text{if } \bar{x}_j > 0 \\ \\ -\bar{r}_j & \text{if } \bar{r}_j \le 0 \\ \\ 0 & \text{if } \bar{x}_j = 0 \text{ and } \bar{r}_j > 0 \\ \end{cases}$$

For $j \in \beta$, we define:

$$d_\beta := -A_\beta^{-1} A_\eta d_\eta \ .$$

This direction has the following properties, which are fairly easy to prove:

- d is a feasible direction: $Ad = 0$ and $\bar{x} + \alpha d \ge 0$ for all $\alpha > 0$ and sufficiently small.
- If $d = 0$, then $\bar{x}$ is a KKT point, with associated KKT multipliers $\bar{p} = A_\beta^{-T} \nabla f_\beta(\bar{x})$ for the equations "$Ax = b$".
- If $d \ne 0$, then d is a descent direction, that is, $\nabla f(\bar{x})^T d < 0$.

This leads to the following algorithm for solving NLP:

0. The current point is $\bar{x}$. Choose β for which $x_\beta > 0$ and $|\beta| = m$. Define $\eta := \{1, \dots, n\} \setminus \beta$.
1. Compute the reduced gradient:

$$\bar{r}_\eta := \nabla f_\eta(\bar{x}) - A_\eta^T A_\beta^{-T} \nabla f_\beta(\bar{x}) \ .$$

2. Compute the direction d. For $j \in \eta$, define:

$$d_j := \begin{cases} -\bar{r}_j & \text{if } \bar{x}_j > 0 \\ -\bar{r}_j & \text{if } \bar{r}_j \le 0 \\ 0 & \text{if } \bar{x}_j = 0 \text{ and } \bar{r}_j > 0 \end{cases}$$

   For $j \in \beta$, define:

$$d_\beta := -A_\beta^{-1} A_\eta d_\eta \ .$$

3. Compute the step-size. Define the following three quantities:

$$\begin{aligned} \alpha^1 &:= \max\{\alpha | (\bar{x} + \alpha d)_\beta \ge 0\} \\ \alpha^2 &:= \max\{\alpha | (\bar{x} + \alpha d)_\eta \ge 0\} \\ \alpha^3 &:= \arg\min_\alpha \{f(\bar{x} + \alpha d) | 0 \le \alpha \le \min\{\alpha^1, \alpha^2\}\} \end{aligned}$$

4. Update values. If $d = 0$, then STOP ($\bar{x}$ is a KKT point). If $d \ne 0$, then set $\bar{x} \leftarrow \bar{x} + \alpha^3 d$, and go to (0.).

6.1 Some Comments on the Reduced Gradient Algorithm

- The reduced gradient algorithm has been very effective on problems with linear equality and inequality constraints. That is because it is actually mimicking the simplex algorithm as much as possible.
- One very useful modification of the reduced gradient algorithm is to set β to correspond to the index of the set of the m largest components of $\bar{x}$.
- As it turns out, there is no proof of convergence for the algorithm as written above. A slightly modified rule for determining the direction d is as follows, and leads to a straightforward proof of convergence:

For $j \in \eta$, define:

$$d_j := \begin{cases} -\bar{r}_j & \text{if } \bar{x}_j > 0 \text{ and } \bar{r}_j \leq 0 \\ -\bar{r}_j & \text{if } \bar{x}_j = 0 \text{ and } \bar{r}_j \leq 0 \\ -\bar{r}_j \bar{x}_j & \text{if } \bar{x}_j > 0 \text{ and } \bar{r}_j > 0 \\ 0 & \text{if } \bar{x}_j = 0 \text{ and } \bar{r}_j > 0 \end{cases}$$

7 Some Computational Results

In this section we report some computational results on solving quadratic pattern classification problems. The problem under consideration is the dual of the pattern classification quadratic program with penalties for the case when we cannot strictly separate the points, namely D1:

$$\begin{array}{lll} \text{D1}: & \text{maximize}_{\lambda,\gamma} & \sum_{i=1}^{k} \lambda_i + \sum_{j=1}^{k} \gamma_j - \frac{1}{2} \left(\sum_{i=1}^{k} \lambda_i a^i - \sum_{j=1}^{m} \gamma_j b^j \right)^T \left(\sum_{i=1}^{k} \lambda_i a^i - \sum_{j=1}^{m} \gamma_j b^j \right) \\ & \text{s.t.} & \sum_{i=1}^{k} \lambda_i - \sum_{j=1}^{m} \gamma_j = 0 \\ & & \lambda_i \leq C \quad i = 1, \ldots, k \\ & & \gamma_j \leq C \quad j = 1, \ldots, m \\ & & \lambda \geq 0, \quad \gamma \geq 0 \ . \end{array} \tag{12}$$

We refer to this problem as the **REGULAR** model. By defining the extra variables x:

$$x := \left(\sum_{i=1}^{k} \lambda_i a^i - \sum_{j=1}^{m} \gamma_j b^j \right) \ ,$$

we can re-write this model as:

$$\begin{array}{lll}
\text{D2}: & \text{maximum}_{\lambda,\gamma,x} & \sum_{i=1}^{k} \lambda_i + \sum_{j=1}^{m} \gamma_j - \frac{1}{2} x^T x \\
 & \text{s.t.} & x - \left(\sum_{i=1}^{k} \lambda_i a^i - \sum_{j=1}^{m} \gamma_j b^j \right) = 0 \\
 & & \sum_{i=1}^{k} \lambda_i - \sum_{j=1}^{m} \gamma_j = 0 \\
 & & \lambda_i \leq C \quad i = 1, \ldots, k \\
 & & \gamma_j \leq C \quad j = 1, \ldots, m \\
 & & \lambda \geq 0, \gamma \geq 0 \\
 & & \lambda \in \Re^k, \gamma \in \Re^m \ .
\end{array}$$

We refer to this model as the **EXPANDED** model.

We solved ten instances of the pattern classification model that we created, whose main characteristics are as shown in Table 1. In solving these problem instances, we used the penalty value $C = 1,000$.

The two-dimensional problems are illustrated in Figure 2, Figure 3, Figure 4, and Figure 5.

We solved these instances using the following two algorithms:

- **IPM (Interior-Point Method)** We solved the problems using the software code LOQO, which is an interior-point algorithm implementation for linear, quadratic, and nonlinear optimization.

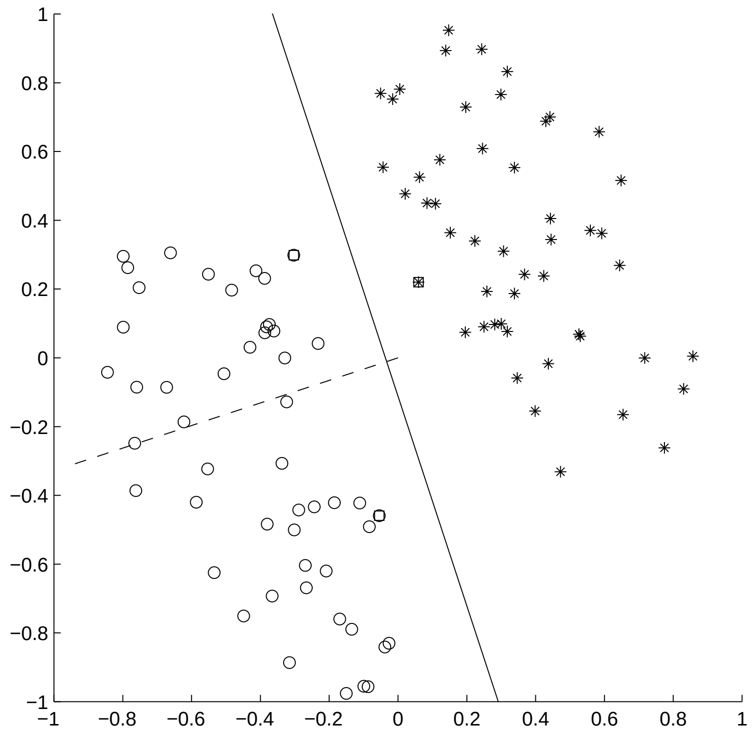

Figure 2: Separable model with 100 points.

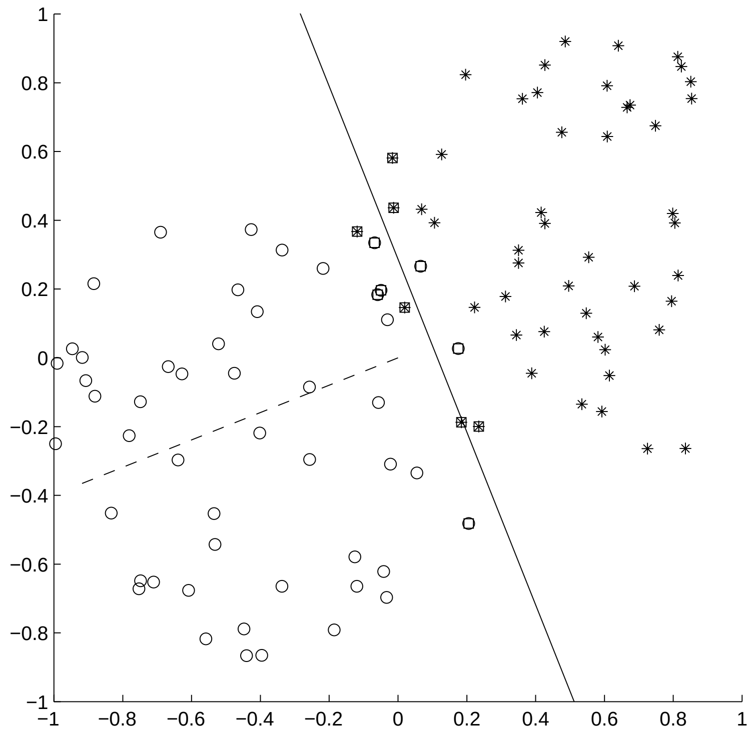

Figure 3: Non-separable model with 100 points.

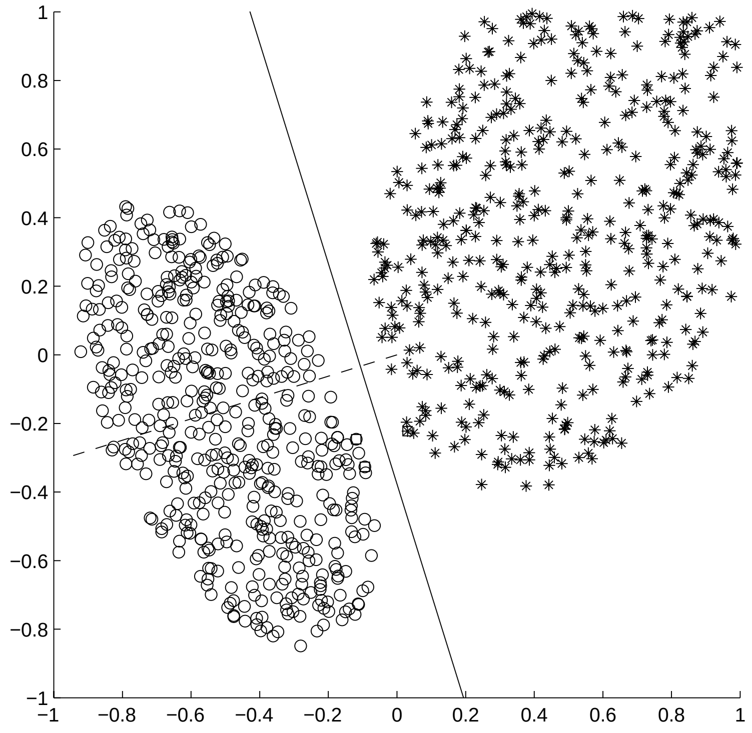

Figure 4: Separable model with 1,000 points.

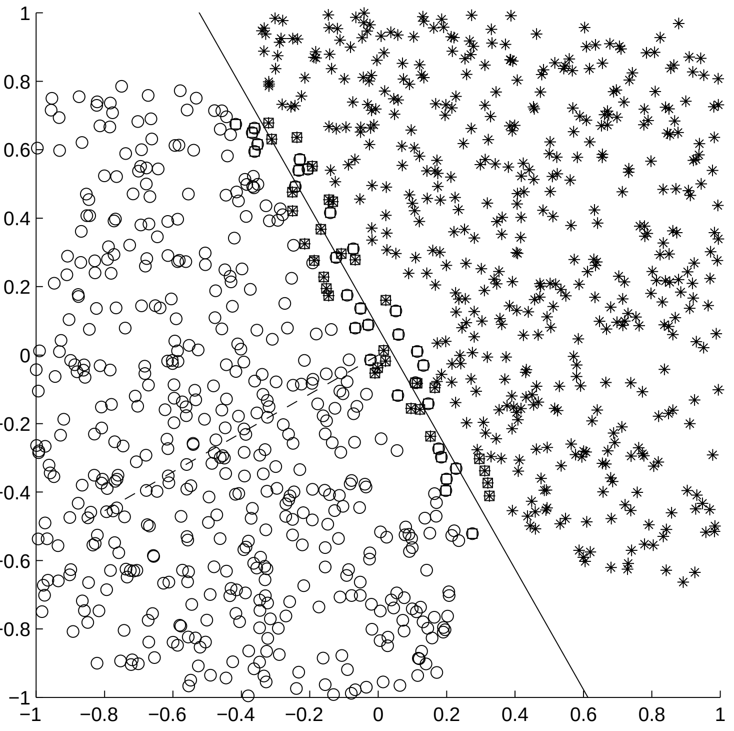

Figure 5: Non-separable model with 1,000 points.

Separable or Not	Dimension n	Number of points m
Separable	2	100
Non-separable	2	100
Separable	2	1,000
Non-separable	2	1,000
Separable	20	300
Non-separable	20	300
Separable	20	1,000
Non-separable	20	1,000
Separable	20	10,000
Non-separable	20	10,000

Table 1: Ten instances of the pattern classification problem.

- **ACTSET (Active Set Method)** We solved the problems using a simple active set method, that adds one new index at each outer iteration, and does not drop any indices from consideration. The method starts with 30 indices in the set β. The interior-point algorithm LOQO was used to solve the subproblems at each iteration.

Note that for this particular model, that active set methods should perform very well. This is because we would expect very few points to be binding in the primal, and so very few of the dual variables λ_i and/or γ_i will be non-zero in the dual problem. And it is the dual problem which we are going to solve.

Table 2 shows the optimal objective function values for each method on each problem. Notice from these tables that both the REGULAR and the EXPANDED models achieve the same optimal objective functions on the different algorithms, when they have enough memory to solve the problem.

Table 3 shows the number of iterations of each method on each problem. Notice the following:

- The non-separable problems take more iterations to solve.

			REGULAR MODEL	EXPANDED MODEL	
Type	n	m	IPM	IPM	ACTSET
Separable	2	100	19.4941	19.4941	19.4941
Non-separable	2	100	992.4144	992.4144	992.4144
Separable	2	1,000		88.8831	88.8831
Non-separable	2	1,000		5,751.2161	5,751.2161
Separable	20	300	1.1398	1.1398	1.1398
Non-separable	20	300	2,101.6155	2,101.6155	2,101.6155
Separable	20	1,000		1.9110	1.9110
Non-separable	20	1,000		7,024.1791	7,024.1791
Separable	20	10,000			4.0165
Non-separable	20	10,000			

Table 2: Optimal solution values reported by each method for each problem. Blanks indicate the method ran out of memory.

- Interior-point methods use very few iterations in general.
- The iteration counts for interior-point methods do not really grow much at all with the size of the problem.
- The exception to these observations is the active set algorithm, for which the total iteration count can be misleading.

Table 4 shows the CPU time for each method and each problem, in seconds. Notice the following:

- The REGULAR model takes more time to solve than does the EXPANDED model.
- As the dimensions of the problem grows, the time to solve the models grows, with one exception.
- The non-separable problem instances take more time to solve.
- The active set method is the clear winner.

This leaves several questions:

			REGULAR MODEL	EXPANDED MODEL	
Type	n	m	IPM	IPM	ACTSET
Separable	2	100	27	26	58
Non-separable	2	100	32	32	194
Separable	2	1,000		28	101
Non-separable	2	1,000		60	1,237
Separable	20	300	30	32	340
Non-separable	20	300	40	39	778
Separable	20	1,000		35	417
Non-separable	20	1,000		54	2,022
Separable	20	10,000			598
Non-separable	20	10,000			$>$ 20,000

Table 3: Iteration count for each problem and each method. Blanks indicate the method ran out of memory.

			REGULAR MODEL	EXPANDED MODEL	
Type	n	m	IPM	IPM	ACTSET
Separable	2	100	2.89	0.20	0.04
Non-separable	2	100	3.23	0.24	0.17
Separable	2	1,000		240.85	0.09
Non-separable	2	1,000		278.56	3.39
Separable	20	300	27.16	8.04	0.99
Non-separable	20	300	41.39	4.30	3.90
Separable	20	1,000		175.60	1.27
Non-separable	20	1,000		289.15	12.00
Separable	20	10,000			1.02
Non-separable	20	10,000			$>$ 58,759

Table 4: CPU times, in seconds. Blanks indicate the method ran out of memory.

- Why does the REGULAR model takes more time to solve than does the EXPANDED model?

- Why do the larger models run out of memory?

The answer to both of these questions probably has to do with the way LOQO handles the numerical linear algebra of forming and then factoring the Newton equation system at each interior-point iteration.