1 Outline

We will discuss solution technique for general nonlinear programming. Most of these methods are search methods. In general, the go through the space: you current estimate, gradient and The objective of these methods is to satisfy the KKT conditions.

- Sequential Quadratic Programming
- Reduced Gradient Method
- Active Set Method
- Penalty Method
- Barrier Method

For details also see the reference listed in the class notes, *Linear and Nonlinear Programming*, by Nash and Sopher.

2 Sequential Quadratic Method

To understand the use of SQP in problems with general constraints, we begin by considering the equality-constrained problem:

\[
\text{minimize } f(x) \\
\text{s.t. } c_i(x) = 0 \quad 1 \leq i \leq m
\]

The idea of SQP is to model this problem at the current point \(x_k\) by a quadratic subproblem and to use the solution of this subproblem to find the new point \(x_{k+1}\). SQP is in a way the application of Newton’s method to the KKT optimality conditions.
The Lagrangian function for this problem is \( L(x, \lambda) = f(x) - \lambda^T c(x) \). We define the transpose of Jacobian of the constraints by
\[
\nabla c(x)^T = [\nabla c_1(x), \ldots, \nabla c_m(x)]
\]
which is an \( n \times m \) matrix. The first order KKT conditions to this problem is
\[
\begin{bmatrix}
\nabla L_x(x, \lambda) \\
c(x)
\end{bmatrix} = 0
\]
where \( \nabla L_x(x, \lambda) = \nabla f(x) - \nabla c(x)^T \lambda \) is the gradient of Lagrangian. This set of nonlinear equations can be solved using iterative methods. With the current guess of \( x_k, \lambda_k \), the updates of Newton’s method satisfy:
\[
\begin{bmatrix}
\nabla^2_{xx} L(x_k, \lambda_k) \\
\nabla c(x_k)
\end{bmatrix} p_k = -\nabla L_x(x, \lambda) - \nabla c(x_k)^T \lambda_k
\]
(1)

Where \( \nabla^2_{xx} L(x_k, \lambda_k) \) is the Hessian of the Lagrangian. And the new guess is:
\[
\begin{bmatrix}
x_{k+1} \\
\lambda_{k+1}
\end{bmatrix} = \begin{bmatrix}
x_k \\
\lambda_k
\end{bmatrix} + \begin{bmatrix}
p_k \\
v_k
\end{bmatrix}
\]

An alternative way of looking at this formulation of the SQP is to define the following quadratic problem at \((x_k, \lambda_k)\)
\[
\text{minimize} \quad \frac{1}{2} p^T \nabla^2_{xx} L(x_k, \lambda_k) p + \nabla L_x(x, \lambda)^T p \\
\text{s.t.} \quad \nabla c_i(x_k)^T p + c_i(x_k) = 0 \quad 1 \leq i \leq m
\]

It is easy to see that, the first order condition of this problem is exact the same as equations (1). Let us demonstrate the method with the following example:

\[
\text{minimize} \quad f(x_1, x_2) = e^{3x_1 + 4x_2} \\
\text{s.t.} \quad c(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0
\]

Note, the actual answer to this problem is stated to be as follows:
\[
x^* = (-\frac{3}{5}, -\frac{4}{5}) \quad \text{with} \quad \lambda^* = -\frac{5}{2} e^{-5} \approx -0.0168
\]

We start with an initial guess of \( v_{x0} = (-0.7, -0.7) \) which apparently came to use through divine intervention or in the very least hand waving. In a similar manner we’ll start with the initial value of \( \lambda_0 = -0.01 \). An iterator of SQP
method with $x_0$, $\lambda_0$ is:

$$\nabla f(x_0) = \begin{pmatrix} 3e^{3x_1+4x_2} \\ 4e^{3x_1+4x_2} \end{pmatrix} = \begin{pmatrix} 0.02234 \\ 0.02979 \end{pmatrix}$$

$$\nabla^2 f(x_0) = \begin{pmatrix} 9e^{3x_1+4x_2} & 12e^{3x_1+4x_2} \\ 12e^{3x_1+4x_2} & 16e^{3x_1+4x_2} \end{pmatrix} = \begin{pmatrix} 0.06702 & 0.08936 \\ 0.08936 & 0.11915 \end{pmatrix}$$

$$c(x_0) = x_1^2 + x_2^2 - 1 = -0.02$$

$$\nabla c(x_0) = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix} = \begin{pmatrix} -1.4 \\ -1.4 \end{pmatrix}$$

$$\nabla^2 c(x_0) = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \nabla_x L = \nabla f - \lambda_0 \nabla c = \begin{pmatrix} 0.008360 \\ 0.015786 \end{pmatrix}$$

$$\nabla^2 x L = \nabla^2 f - \lambda_0 \nabla^2 c = \begin{pmatrix} 0.08702 & 0.08936 \\ 0.08936 & 0.1395 \end{pmatrix}$$

Solving equation (1), we have $p_0 = (0.14196, -0.15624), v_0 = -0.004802$

### 3 Reduced Gradient Method

Reduced gradient method is a method to solve a problem with nonlinear target function and linear equality constraints. It is based on projecting the search direction into the null space of the constraints. So it is also called as gradient projection method. We have the following problem:

$$\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad Ax = b
\end{align*}$$

Where $x$ and $b$ are vector of length $n$ and $A$ is a $m \times n$ matrix. If we have a current value of $x_k$ and some direction $p_k$ to decrease $f(x)$, we need to reduce $p_k$ into another direction $u_k$ to make sure that the update $x_{k+1} = x_k + u_k$ still satisfies the constraints. Since the constraints are linear, this requirement can be easily expressed as $Au_k = 0$. Aiming at reducing direction the least, we try to find a direction $u_k$ satisfying

$$\begin{align*}
\text{minimize} & \quad u_k^T p_k \\
\text{subject to} & \quad Au_k = 0 \\
& \quad u_k^T u_k = 1
\end{align*}$$

The constraints is equivalent with that $u_k$ belongs to the null space of $A$. Let $Z$ be an orthogonal basis of the null space of $A$ and express $u_k$ as $Zv_k$. With $Z^T Z = I$, the former problem can be restated as finding $v_k$:

$$\begin{align*}
\text{minimize} & \quad v_k^T Z^T Z^T p_k \\
\text{subject to} & \quad v_k^T v_k = 1
\end{align*}$$
Noticing that $ZZ^T p_k$ is in subspace $Z$, this problem is essentially an unconstrained inner product minimization with respect to $ZZ^T p_k$, the solution of which is unit length vector in the same direction. In fact, $ZZ^T$ is the orthogonal projector onto $Z$ and can be calculated as $I - A^T (AA^T)^{-1} A$. This result can be also directly obtained by a Lagrange treatment of the original problem of $u_k[1]$. An extended version of the algorithm called generalized reduced method is applied to general nonlinear problems with linear approximation of constraints and manipulation of basic and nonbasic variables[1, 2].

4 Active Set Method

The SQP and Reduced Gradient method both work on equality constraints. In contrast, the Active Set Method can be used with both equality and inequality constraints. Recall that the premise of the Active Set Method is to select a set of constraints to be binding and update the set each iteration.

1. The algorithm starts with the following optimality test:

$$Z^T \nabla f(x_k) = 0$$

If no constraints are active than we have a local minimum and we can stop. Otherwise we must compute the Lagrange multiplier as follows:

$$\bar{\lambda} = \bar{A}_r \nabla f(x_k)$$

where $A_r$ is the right inverse of $A$, satisfying $A_r A = I$. If $\lambda = 0$ we can stop as we have an optimal solution.

2. Search in the descent direction staying feasible

3. Step $\alpha$ such that $f(x_k + \alpha p_k) < f(x_k)$ and $x_{vk+1}$ stays feasible.

4. Update $x_{k+1} = x_k + \alpha p_k$. If a new constraint is hit then add it to the active set and goto step 1. To illustrate the Active Set Method consider the following example problem:

$$\text{minimize} \quad f(x) = \frac{1}{2} (x_1 - 3)^3 + (x_2 - 2)^2$$

s.t.  \quad \begin{align*}
2x_1 - x_2 & \geq 0 \\
-x_1 - x_2 & \geq -4 \\
x_2 & \geq 0
\end{align*} \quad (2)$$

We start with the first and third constraints as binding so we have the following:

$$x_0 = (0, 0)$$

This provides us with the initial binding bases as follows:

$$\bar{A} = \begin{pmatrix} 2 & -1 \\ 0 & 1 \end{pmatrix}$$
\(Z\) is empty so \(Z^T \nabla f(x) = 0\)
\[
\bar{\lambda} = \bar{A}_r^T \nabla f(x_k)
\]
\[
= \begin{pmatrix}
\frac{1}{2} & 0 \\
\frac{1}{2} & 1 \\
\end{pmatrix}
\begin{pmatrix}
-3 \\
-4 \\
\end{pmatrix}
\]
\[
= \begin{pmatrix}
-3/2 \\
-11/2 \\
\end{pmatrix}
\]
Note in equation 4 that both \(\bar{\lambda}_1\) and \(\bar{\lambda}_2\) are negative. This means that the constraints are not helping the objective and thus we remove the third constraint because it is the most negative. This gives us the following:
\[
\bar{A} = \begin{pmatrix}
2 & -1 \\
\end{pmatrix}
\]
\[
Z = \begin{pmatrix}
\frac{1}{2} \\
1 \\
\end{pmatrix}
\]
\[
\bar{A}_r = \begin{pmatrix}
\frac{1}{2} \\
0 \\
\end{pmatrix}
\]
Using the reduced gradient we get the following:
\[
Z^T \nabla f = \frac{11}{2} \neq 0
\]
Thus we can calculate the search direction, \(p\), as follows where the resulting value tells us the multiples of the direction to move in.
\[
p = -Z(Z^T \nabla^2 f(x))^{-1} \nabla f
\]
\[
= \begin{pmatrix}
11/9 \\
22/9 \\
\end{pmatrix}
\]
Thus we will step in following way:
\[
x_1 = x_0 + \alpha \begin{pmatrix}
11/9 \\
22/9 \\
\end{pmatrix}
\text{ with } x_0 = \begin{pmatrix}
0 \\
0 \\
\end{pmatrix}
\]
Step until a constraint is violated. Often times \(\alpha = 1\) is chosen. After the stepping is complete check for optimality.

5 Penalty Methods

Penalty methods incorporate the constraints into an extra term in the objective function. The reformulation is generically expressed as
\[
\text{minimize } f(x) + p\phi(x)
\]
a sum of the original objective function and the penalty function $\phi$ weighted by $p$. This form permits the feasibility constraints to be violated; the function $\phi(x)$ imposes a greater penalty the farther $x$ is from the feasible region. This encourages solutions to hit the boundary, and thus is well-suited for nonlinear programs with equality constraints. The choice of $\phi$ depends on the structure of the problem. Two common forms of $\phi$ for the equality constraints are

$$
\phi_{eq1}(x) = \frac{1}{\gamma} \sum_{i=1}^{m} |c_i(x)|^\gamma \text{ for } \gamma \geq 1
$$

$$
\phi_{eq2}(x) = \frac{1}{2} \sum_{i=1}^{m} c_i(x)^2 = \frac{1}{2} c^T(x)c(x)
$$

For the first form, an increase in $\gamma$ reflects the amount by which the base penalty is increased given the distance of $x$ from the feasible region. The $p$ parameter is initially chosen to be high, reflecting a great penalty for leaving the feasible boundary, as is illustrated in Figure 1(a). In this way, the penalty method discourages infeasible solutions, pushing them toward the boundary.

Inequality constraints cannot be easily incorporated into a penalty functions; indicator functions are sometimes used,

$$
\phi(x) = I(c_i(x) \geq 0) = \begin{cases} 
1 & c_i(x) < 0 \\
0 & c_i(x) \geq 0
\end{cases}
$$

The trouble with the indicator form is that it may not necessarily preserve convexity so any previous guarantees of finding a global minimum are now lost. A closely related technique to the penalty method, called the barrier method, avoids this problem.
6 Barrier Methods

Barrier methods approach the problem slightly differently. All trial solutions must lie within the feasible region. The closer a point is to the boundary, the greater the penalty imposed. In general, a point can never hit the boundary because the penalty approaches infinity as the trial solutions approach the feasible boundary. This effect is illustrated in Figure 1(b). The form of the reformulated objective for the barrier method is exactly the same as the penalty method,

\[ \text{minimize} \quad f(x) + \mu \phi(x) \]

except \( \mu \) is used instead to weight \( \phi \). To achieve the effect just described, two forms of \( \phi \) may work,

\[ \phi_{in1}(x) = \sum_{i=1}^{m} \log c_i(x), \]
\[ \phi_{in2}(x) = \sum_{i=1}^{m} \frac{1}{c_i(x)}. \]

As \( c_i(x) \) approaches zero, its log and multiplicative inverse approach infinity.

This invites the question: how will the barrier method find an optimal solution if it lies close to the boundary? While the algorithm can never hit the boundary, the solution can venture outward from the inner part of the feasible volume as the barrier weight \( \mu \) is lowered.

Recall the difficulty we had expressing inequality constraints in the penalty method reformulation. Barrier methods have the opposite problem: equality constraints do not easily integrate into the penalty function. One alternative is to replace each equality constraint \( c_j(x) = \epsilon \) with two inequality constraints

\[ c_j(x) \leq \epsilon \]
\[ c_j(x) \geq -\epsilon. \]

but this can be problematic since the solution can never hit the boundary defined by \( \epsilon \).

7 SUMT

We have seen that Barrier methods are strictly feasible methods.

\[ P(x, r) = f(x) - r \cdot B(x) \]  \hspace{1cm} (8)

where, \( f \) is a maximizing function and \( B \) is a barrier function.

Consider the following:

Maximize \( f(x) \) subject to \( g_i(x) \leq b_i, \quad x_i \geq 0 \)
\[ B(x) = \sum_{i=1}^{m} \frac{1}{b_i - g_i(x)} + \sum_{j=1}^{n} \frac{1}{x_j} \]

Note: if \( b_i \to g_i(x) \), then \( B(x) \to \infty \). Hence the function does not hit the barrier.

Idea of SUMT: Solve the unconstrained \( \min P(x,r) \) over a sequence of times. Each sequence reduces \( r \) i.e. allows us to move closer to boundary. Usually go until some error tolerance is reached. If \( x_1 = \max(P(x,r), x^* \max f(x)) \), then \( f(x_1) \leq f(x^*) \leq f(x_1) + r.B(x_1) \). Step : \( r_{\text{new}} = \theta r \), say \( \theta = 0.1 \) where the stopping condition is : \( r.B(x) \leq \epsilon \).

### 7.1 Algorithm for SUMT

1. Let \( x = \text{initial solution (not on boundary)} \) \( k = 1, \theta = 0.01 \) and \( r = 1 \)

2. At \( k\)-th iteration,
   \( x_{k-1} = \) starting point, we can use multivariate unconstrained technique e.g. \( \text{gradient search method} \) to find the local maximum/minimum of
   \[ P(x,r) = f(x) - r.\left( \sum_{i=1}^{m} \frac{1}{b_i - g_i(x)} + \sum_{j=1}^{n} \frac{1}{x_j} \right) \]  
   (9)

3. Stop if \( |x_k - x_{k-1}| \leq \epsilon \).

4. else \( k = k + 1 \) and \( r = \theta r \); go to step 2

Note: if we had equality constraints i.e. \( g_i(x_i) = b_i \), then instead of \( \frac{-r}{b_i - g_i(x)} \) we would use \( \frac{-r}{\sqrt{r(g - g_i(x))^2}} \).

### 7.2 Example of SUMT

Maximize \( f(x) = 5x_1 - x_1^2 + 8x_2 - 2x_2^2 \) subject to

\[ \begin{align*}
3x_1 + 2x_2 & \leq 6 \\
x_1 & \geq 0, \quad x_2 \geq 0
\end{align*} \]

Let us construct the barrier function for the problem.

\[ B(x) = \frac{1}{6 - 3x_1 - 2x_2} + \frac{1}{x_1} + \frac{1}{x_2} \]  
(10)

and we have to maximize \( P(x,r) = f(x) - r.B(x) \). Let us start with \( r = 1 \) and \( \theta = 0.1 \). Initial trial solution \( x_1 = 1, x_2 = 1 \).

After the first iteration by gradient search we obtain \( x = (x_1, x_2) = (0.851, 1.329) \)
In the second iteration, we take \( r = 0.1 \) and in the third and fourth iteration we value of \( r \) as 0.01 and 0.001 respectively. After fourth iteration, we obtain \( x^* = (1,1.5) \) which is close enough to barrier, so that we can stop the iteration.
References
