Gradient Techniques for Unconstrained Optimization

Gradient techniques can be used to minimize a function \( F(x) \) with respect to the \( n \) by 1 vector \( x \), when the gradient is available or easily estimated. These techniques are usually used when (1) a set of linear equations in \( x \) can’t be obtained, or (2) directly solving the equations for \( x \) requires too much storage.

I. Quadratic Functions

A. Definitions

The Hessian matrix of a function \( F(x) \), denoted as \( \nabla^2 F(x) \) or \( G(x) \), is defined as

\[
\nabla^2 F(x) = G(x) = [g(i,j)],
\]

where

\[
g(i,j) = \frac{\partial^2 F}{\partial x(i)\partial x(j)}
\]

If \( G(x) \) is constant, \( F(x) \) is a quadratic function and can be expressed as

\[
F(x) = (1/2)x^T G x + c^T x + \alpha
\]

(1)

The derivatives of \( F(x) \) are

\[
\nabla F(x) = G x + c = g(x),
\]

(2)

\[
\nabla^2 F(x) = G
\]

(3)
The first three terms of the Taylor series expansion of a general function \( F(x) \) are

\[
F(x+h\mathbf{p}) = F(x) + hg(x)^T \mathbf{p} + (1/2)h^2 \mathbf{p}^T G(x) \mathbf{p} + O(h^3)
\]  (4)

where \( g(x) \) is the gradient of \( F \) defined in (2), \( \mathbf{p} \) is a direction vector, and \( h \) is a small constant. Comparing (1) and (4), we note that a general function can be approximated as a quadratic.

**B. Properties of Quadratic Functions**

Let \( F(x) \) be a quadratic function defined as

\[
F(x) = c^T x + (1/2)x^T G x
\]  (5)

From (2) and (4), \( F(x) \) can be expressed as

\[
F(x+h\mathbf{p}) = F(x) + h\mathbf{p}^T(Gx+c) + (1/2)h^2 \mathbf{p}^T G \mathbf{p}
\]  (6)

If the function \( F \) has a stationary point \( x^* \) where the gradient vector \( g(x) \) vanishes,

\[
\nabla F(x^*) = Gx^* + c = 0
\]  (7)

Therefore the critical point \( x^* \) satisfies the following system of linear equations,

\[
Gx^* = -c
\]  (8)

If (8) has no solution, then \( F(x) \) has no local or global minima, maxima, or points of inflection. \( F(x) \) has as many critical points as there solutions to (8). If \( G \) is nonsingular, \( F(x) \) has a unique critical point. \( F(x) \) has a minimum at \( x^* \) iff

1. \( \|g(x^*)\| = 0 \)
2. \( G \) is positive semidefinite
**II. Steepest Descent Method**

The classical steepest descent approach can be used to minimize a general function $F(x)$ with respect to the $n$ by 1 vector $x$. Here, we start with $k=0$ and obtain an initial guess, $x(0)$. We then go through the following steps.

1. Calculate $g(x(k))$
2. Minimize $F(x(k)+B_2g(k))$ with respect to $B_2$.
3. Iterate $x(k)$ as
   
   $$x(k+1) = x(k) + B_2g(k)$$

4. Increase $k$ by 1 and go to (1), or stop after $F$ no longer decreases.

Step (2) is called the line search sub-problem. These steps are summarized in the flowchart. The steepest descent approach can find a minimum of $F(x)$, but it usually requires a number of iterations much larger than the number of unknowns $n$.

**III. Conjugate Gradient Method**

**A. Assumptions**

Our basic goal here is to develop the conjugate gradient technique for minimizing a function $F(x)$, as shown in the flowchart. The following assumptions are made.

(A1) The function $F(x)$ is accurately modelled as a quadratic as in equation (5). Here $x$ is an $n$ by 1 vector.

(A2) $G$ is the Hessian of $F(x)$. It is assumed positive definite, so that a global minimum of $F(x)$ exists.

(A3) $x(k)$ is the estimate of the minimizing point of $F(x)$, at the $k$th iteration.

(A4) $g(k)$ is $n$ by 1 the gradient vector of $F(x(k))$, which is defined using (7) as

$$\nabla F(x(k)) = Gx(k) + c$$
(A5) \( p(k) \) is a \( n \) by 1 direction vector for the \( k \)th iteration. Its relation to \( g(k) \) will be derived later.

(A6) An inner product can be defined using the Hessian, as

\[
\langle p(i), p(j) \rangle = p(i)^T G p(j)
\]

Two vectors \( p(i) \) and \( p(j) \) are mutually conjugate (orthogonal) with respect to the matrix \( G \) if

\[
\langle p(i), p(j) \rangle = p(i)^T G p(j) = 0
\]

for \( i \) not equal to \( j \).

B. Basic Approach

Given (5) our basic approach is to minimize

\[
F(x(k)+P(k)w)
\]

with respect to the \((k+1)\) by 1 vector \( w \), where \( P(k) \) is an \( n \) by \((k+1)\) matrix with columns \( p(i), i=0 \) to \( k \). Initially, we assume that the \( p(k) \) direction vectors are linearly independent. It will turn out later that the \( p(i)'s \) are mutually conjugate with respect to \( G \) and that they can be easily found.

C. Development of the Method

First, substituting \( x(k)+P(k)w \) for \( x \) in (5), we get

\[
\ldots \ldots . \quad F(x(k)+P(k)w) = c^T(x(k)+P(k)w) + (1/2)((x(k)+P(k)w)^T G (x(k)+P(k)w) \quad (9)
\]

Collecting only those terms having \( w \) in (9) yields

\[
Fw(x(k)+P(k)w) = c^T P(k)w + (1/2)[w^T P(k)^T G x(k) + x(k)^T G w P(k) + w^T P(k)^T G P(k) w]
\]

\[
= w^T P(k)^T (c + g x(k)) + (1/2) w^T P(k)^T G P(k) w
\]

\[
\ldots \ldots \ldots \quad = w^T P(k)^T g(k) + (1/2) w^T P(k)^T G P(k) w \quad (10)
\]

using (A4). If \( \nabla \) has elements \( \partial / \partial w(i) \), then
\[ \nabla F w(x(k)+P(k)w) = P(k)^T g(k) + P(k)^T GP(k)w \quad (11) \]

Equating this to the zero vector and solving for \( w \), we get
\[ w = -[P(k)^T GP(k)]^{-1} P(k)^T g(k) \quad (12) \]

Letting \( x(k+1) = x(k)+P(k)w \) and using (12) we get
\[ x(k+1) = x(k)-P(k)[P(k)^T GP(k)]^{-1} P(k)^T g(k) \quad (13) \]

Thus \( x(k+1) \) is a function of \( x(k) \), the gradient \( g(k) \), and the present and past direction vectors stored in \( P(k) \). (13) needs considerable refinement. At this point we know nothing about how to get the columns \( p(i) \) of \( P(k) \).

First note that \( g(k+1) \) is orthogonal to \( P(k) \), since
\[ P(k)^T g(k+1) = P(k)^T (c+Gx(k+1)) = P(k)^T (c+G(x(k)+P(k)w)) = P(k)^T g(k) - P(k)^T GP(k)(P(k)^T GP(k))^{-1} P(k)^T g(k) = 0 \]

Therefore,
\[ g(k+1)^T p(i) = 0 \quad (14) \]
for \( i = 0 \) to \( k \). Obviously,
\[ g(j)^T p(i) = 0 \quad (15) \]
for \( j > i \). Using (15) to evaluate \( P(k)^T g(k) \) in (13), we get
\[ x(k+1) = x(k) + B_1 P(k)(P(k)^T GP(k))^{-1} e(k) \quad (16) \]

where \( e(k) \) is the \((k+1)\)th column of the \((k+1)\) by \((k+1)\) identity matrix and
\[ B_1 = -g(k)^T p(k) \quad (17) \]

To simplify (16) further we assume that the \((k+1)\) vectors \( p(j) \) are mutually conjugate or orthogonal with respect to \( G \), as defined in (A6). This makes \( P(k)^T GP(k) \) diagonal, and
\[ x(k+1) = x(k) + B_i P(k)[0,0,0,0...,1/(p(k)^T Gp(k))]^T \]

\[ = x(k) + B_i p(k)/p(k)^T Gp(k) \]

\[ = x(k) + a(k)p(k), \quad \text{(18)} \]

\[ a(k) = -g(k)^T p(k)/p(k)^T Gp(k) \quad \text{(19)} \]

At this point we have a search method similar to steepest descent, except that the conjugate directions \( p(k) \) are used instead of the gradients \( g(k) \). Also, (18) is much simpler than (13) since it requires \( p(k) \), rather than \( P(k) \). It remains for us to find an easy way to calculate \( p(k) \) from \( g(k) \).

Let \( y(i) \) denote \( g(i+1) - g(i) \). Using (18) and the conjugacy of the \( p(i) \)'s,

\[ p(j)^T y(i) = p(j)^T G (x(i+1) - x(i)) = p(j)^T a(i) Gp(i) \]

\[ = 0 \quad \text{(20)} \]

for \( i \) not equal to \( j \). We can use the Schmidt process to construct a set of mutually conjugate direction vectors \( p(i) \) from the gradient vectors \( g(i) \) as

\[ p(0) = -g(0), \quad \text{(21)} \]

\[ p(k) = -g(k) + \sum_{j=0}^{k-1} B(k,j)p(j) \quad \text{(22)} \]

Let \( P_k \) denote the space spanned by the columns of \( P(k) \). Since (22) can be solved for \( g(k) \), \( g(i) \) is in \( P_k \) for \( i \leq k \). Therefore (15) yields

\[ g(k)^T g(i) = 0 \quad \text{(23)} \]

for \( i < k \). Pre-multiplying (22) by \( p(i)^T G \) and using (20) and the conjugacy of the \( p(i) \)'s, we obtain for \( i=0 \) to \( k-1 \),

\[ p(i)^T Gp(k) = -p(i)^T Gg(k) + \sum_{j=0}^{k-1} B(k,j)p(i)^T Gp(j) \]

\[ = -(1/a(k))(g(i+1) - g(i))^T g(k) + B(k,i)p(i)^T Gp(i) \quad \text{(24)} \]

Equation (23) implies that the first term on the right hand side of (24) equals 0 for \( i < k-1 \). Thus, to
make \( p(k) \) conjugate to \( p(i) \) for \( i < k-1 \), let \( B(k,i) = 0 \) in (24). Since only \( B(k,k-1) \) is non-zero, replace it with \( B(k-1) \) in (24). (22) is now rewritten as

\[
p(k) = -g(k) + B(k-1)p(k-1)
\]  

(25)

Pre-multiplying (25) by \( y(k-1)^T \) and applying the orthogonality condition of (20), we get

\[
0 = -y(k-1)^Tg(k) + B(k-1)y(k-1)^Tp(k-1),
\]

\[
B(k-1) = y(k-1)^Tg(k)/y(k-1)^Tp(k-1)
\]

(26)

Using the orthogonality of the gradients in (23), the numerator in (26) equals \( \|g(k)\|^2 \). Using the definition of \( p(k) \) in (22), the denominator in (26) becomes \( \|g(k-1)\|^2 \). (26) therefore becomes

\[
B(k-1) = \|g(k)\|^2/\|g(k-1)\|^2
\]

(27)

The conjugate gradient technique is summarized as (18), (21), (25), and (27). For the general, non-quadratic case, \( a(k) \) in (18) is found to minimize \( F(x(k)+a(k)p(k)) \). This is the line search sub-problem. This method is summarized in the flowchart.

IV. Flowchart for Gradient or Steepest Descent Algorithm

The goal is to minimize the nonnegative error function \( I(x) = I(x(1), x(2), x(3), \ldots, x(n)) \) with respect to the \( n \)-dimensional vector \( x = (x(1), x(2), x(3), \ldots, x(n))^T \). Use is made of the gradient vector, \( g = (g(1), g(2), g(3), \ldots, g(n))^T \).
Initialise $i$ and $\mathbf{x}$

Pick $x(j)$ for $j=1$ to $n$

Calculate $I(\mathbf{x})$ and list it.
Find gradient of $I(\mathbf{x})$.

Calculate $g(j)=\frac{dI(\mathbf{x})}{dx(j)}$
Reduce $I(\mathbf{x})$

Minimize $I(\mathbf{x}-B_2 \cdot g)$ with respect to $B_2$ such that $I(\mathbf{x}+B_2 \cdot g)/dB_2 = 0$
Update $\mathbf{x}$

$x(j)=x(j)+B_2 \cdot g(j)$ for $j=1$ to $n.$

$i=i+1$

If $I(\mathbf{x})$ is a simple sum of squares function, we can pick $x(j)=0$ initially. Several sequences of $(n+1)$ iterations will usually be necessary.
V. Flowchart for Conjugate Gradient Algorithm

The goal is to minimize the nonnegative error function $I(\mathbf{x}) = I(x(1), x(2), x(3), \ldots, x(n))$ with respect to the $n$-dimensional vector $\mathbf{x} = (x(1), x(2), x(3), \ldots, x(n))'$. Use is made of the gradient vector, $\mathbf{g} = (g(1), g(2), g(3), \ldots, g(n))'$ and the direction vector $\mathbf{p} = (p(1), p(2), p(3), \ldots, p(n))'$.

1. Initialize $i$, $\mathbf{X}_D$, and $\mathbf{x}$.

2. Pick $x(j)$ for $j = 1$ to $n$.

3. Calculate $I(\mathbf{x})$ and list it. Calculate $g(j) = \frac{dI(\mathbf{x})}{dx(j)}$ for $j = 1$ to $n$.

   \[
   \mathbf{X}_N = \sum_{j=1}^{n} g^2(j) 
   \]

4. Find gradient of $I(\mathbf{x})$.

5. Find direction vector $\mathbf{p}$.

6. Minimize $I(\mathbf{x} + B_2 \mathbf{p})$ with respect to $B_2$ such that $\frac{dI(\mathbf{x} + B_2 \mathbf{p})}{dB_2} = 0$.

7. $x(j) = x(j) + B_2 p(j)$ for $j = 1$ to $n$.

8. $i = i + 1$.

9. If $i > n+1$, stop.

10. Otherwise, go to step 2.
VI. Line Search Sub-Problem

If \( I(x) \) is a simple sum of squares function, we can pick \( x(j) = 0 \) initially, and stop after \((n+1)\) iterations, as indicated. If \( I(x) \) is a more complicated function, more than one sequence of \((n+1)\) iterations will usually be necessary. For simple \( I(x) \) functions, closed form expressions for \( B_2 \) can be found. Otherwise, we can pick \( B_2 \) after first expanding \( \partial I(x + B_2 p) / \partial B_2 \) as

\[
\frac{\partial I(x + B_2 p)}{\partial B_2} = B + C \cdot B_2 + D \cdot B_2^2
\]

There are three common approaches.

A. Binomial Expansion (best approach)

Suppose that \( I(x + B_2 p) \) can be written as

\[
I(x + B_2 p) = \sum_{i=1}^{m} [r(i) + B_2 s(i)]^{2L}
\]

where \( L \) is a positive integer. The partial derivative of the binomial expansion of \( I \) yields

\[
B = \sum_{i=1}^{m} 2L r(i)^{2L-1} s(i)
\]

\[
C = \sum_{i=1}^{m} 2L(2L-1)r(i)^{2L-2}s(i)^2
\]

\[
D = \sum_{i=1}^{m} 2L(2L-1)(2L-2)r(i)^{2L-3}s(i)^3 \cdot \frac{3}{2}
\]

B. Taylor Series Approach

Pick a number \( e \) which is small compared to \( \|x\| / \|p\| \). Calculate \( C(k) = I(x + (k-1)e \cdot p) \) for \( k = 1, 2, 3, 4 \). Find constants \( B, C, \) and \( D \) by using the Taylor series or polynomial fit procedures given below.

\[
B = (C(2) - C(1)) \cdot e,
\]

\[
C = (C(3) - 2C(2) + C(1))
\]
\[ D = (C(4) - 3 \cdot C(3) + 3 \cdot C(2) - C(1)) / (2 \cdot e), \]

**C. Polynomial Fit Approach (second best approach)**

\[
B = e \cdot (-11 \cdot C(1) + 18 \cdot C(2) - 9 \cdot C(3) + 2 \cdot C(4)) / 6
\]

\[
C = (6 \cdot C(1) - 15 \cdot C(2) + 12 \cdot C(3) - 3 \cdot C(4)) / 3
\]

\[
D = (C(4) - 3 \cdot C(3) + 3 \cdot C(2) - C(1)) / (2 \cdot e),
\]

**D. Final Calculation**

Now find \( B_2 \) as

\[
B_2 = \frac{-C + \sqrt{C^2 - 4BD}}{2D} \quad \forall \quad B_2 = \frac{-B}{C}
\]

**VII. References**
