Minimization Using Descent Information

- we will consider the minimization of unconstrained functions of several variables where we now assume we have some derivative information such as the gradient vector or the Hessian matrix.

- Recall that Powell's method used the powerful concept of conjugate directions and performed a series of line searches.

- We will see how these conjugate directions are related to the gradient directions and we will introduce a very powerful method called the ***conjugate-gradient*** technique.

- Recall Taylor's expansion uses such information:

$$\boxed{f(\boldsymbol{x} + \Delta\boldsymbol{x}) \cong f(\boldsymbol{x}) + \Delta\boldsymbol{x}^T \nabla f(\boldsymbol{x}) + \frac{1}{2}\Delta\boldsymbol{x}^T H(\boldsymbol{x})\Delta\boldsymbol{x}}$$

- Methods using only first derivatives are called *first-order methods*

- Methods using second order derivatives are called *second-order methods*.

The Gradient Vector Re-examined

- Recall that the gradient vector of a function $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^n$:

$$g(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \ldots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

- Consider a differential length

$$d\boldsymbol{x} = \begin{bmatrix} dx_1 \\ dx_2 \\ \ldots \\ dx_n \end{bmatrix} = \boldsymbol{u}\, ds$$

where $\|\boldsymbol{u}\| = 1$ and $\boldsymbol{u}$ holds the directional information of the differential.

- Now a change in the function $f(\boldsymbol{x})$ along $d\boldsymbol{x}$ is given by

$$df = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} dx_i = (\nabla f(\boldsymbol{x}), d\boldsymbol{x}) = (\nabla f(\boldsymbol{x}), \boldsymbol{u}\, ds) = ds(\nabla f(\boldsymbol{x}), \boldsymbol{u})$$

the rate of change of $f(\boldsymbol{x})$ along the arbitrary direction $\boldsymbol{u}$ is given by

$$\boxed{\frac{df}{ds} = (\nabla f(\boldsymbol{x}), \boldsymbol{u})}$$

- On the other hand, along a direction $\boldsymbol{u}$, the function $f(\boldsymbol{x})$ is described as $f(\boldsymbol{x} + \alpha\boldsymbol{u})$ and thus the rate of change along this direction is also written as

$$\boxed{\frac{df}{ds} = (\nabla f(\boldsymbol{x}), \boldsymbol{u}) = \frac{d}{d\alpha} f(\boldsymbol{x} + \alpha\boldsymbol{u})} \tag{1}$$

- We can examine (1) to see which direction gives the maximum rate of increase.

- A well known inequality in functional analysis is called the *Cauchy-Schwarz* inequality which says

$$\boxed{(\boldsymbol{a}, \boldsymbol{b}) \le \|\boldsymbol{a}\| \|\boldsymbol{b}\|} \tag{2}$$

  Applying this to (1) we find

$$(\nabla f(\boldsymbol{x}), \boldsymbol{u}) \le \|\nabla f(\boldsymbol{x})\| \|\boldsymbol{u}\| = \|\nabla f(\boldsymbol{x})\| \tag{3}$$

  and letting $\boldsymbol{u} = \dfrac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|}$ we have

$$\left(\nabla f(\boldsymbol{x}), \frac{\nabla f(\boldsymbol{x})}{\|\nabla f(\boldsymbol{x})\|}\right) = \frac{\|\nabla f(\boldsymbol{x})\|^2}{\|\nabla f(\boldsymbol{x})\|} = \|\nabla f(\boldsymbol{x})\| \tag{4}$$

  which shows that for this choice of direction, the directional derivative

$$(\nabla f(\boldsymbol{x}), \boldsymbol{u})$$

  reaches its maximum value.

- Therefore we say that $\nabla f(\boldsymbol{x})$ is the **direction of maximum increase** and

- $-\nabla f(\boldsymbol{x})$ is the direction of maximum decrease (also: **steepest ascent and steepest descent**).

Cauchy's Method (Steepest Descent)

- A logical minimization strategy is to use the direction of steepest descent and perform a line search in that direction.

- Assume we are at a point $\boldsymbol{x}_k$ and that we have calculated the gradient at this point, $\boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{x}_k) = \nabla f(\boldsymbol{x}_k)$.

- Then we can minimize along $\boldsymbol{g}_k$ starting from $\boldsymbol{x}_k$:

$$\boxed{\alpha_k = \underset{\alpha}{\arg\min}\ f(\boldsymbol{x}_k + \alpha \boldsymbol{g}_k)} \tag{5}$$

and arrive at the new point

$$\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{g}_k}. \tag{6}$$

Algorithm: Cauchy's Method of Steepest Descent

1. input: $f(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$, $\boldsymbol{x}_0$, g_{tol}, k_{max}
2. set: $\boldsymbol{x} = \boldsymbol{x}_0$, $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x})$
3. while $k < k_{max}$ and $\|\boldsymbol{g}\| > g_{tol}$
4. set: $\alpha = \underset{\alpha}{\arg\min}\ f(\boldsymbol{x} + \alpha \boldsymbol{g})$
5. set: $\boldsymbol{x} = \boldsymbol{x} + \alpha \boldsymbol{g}$
6. set: $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{x})$, $k = k + 1$
7. end

- Note that we don't bother to take the negative of $\boldsymbol{g}$ in the line search since this is automatically accomplished by allowing negative values of α.

- The iterations end when either a maximum number of iterations have been reached or the norm of the gradient at the current point is less than a user defined value g_{tol} which is a value close to zero.

- The successive directions of minimization in the method of steepest descent are orthogonal to each other.

- This can be shown as follows: assume that we are at a point $\boldsymbol{x}_k$ and we need to find

$$\alpha_k = \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{g}_k) \tag{7}$$

in order to arrive at the new point

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{g}_k . \tag{8}$$

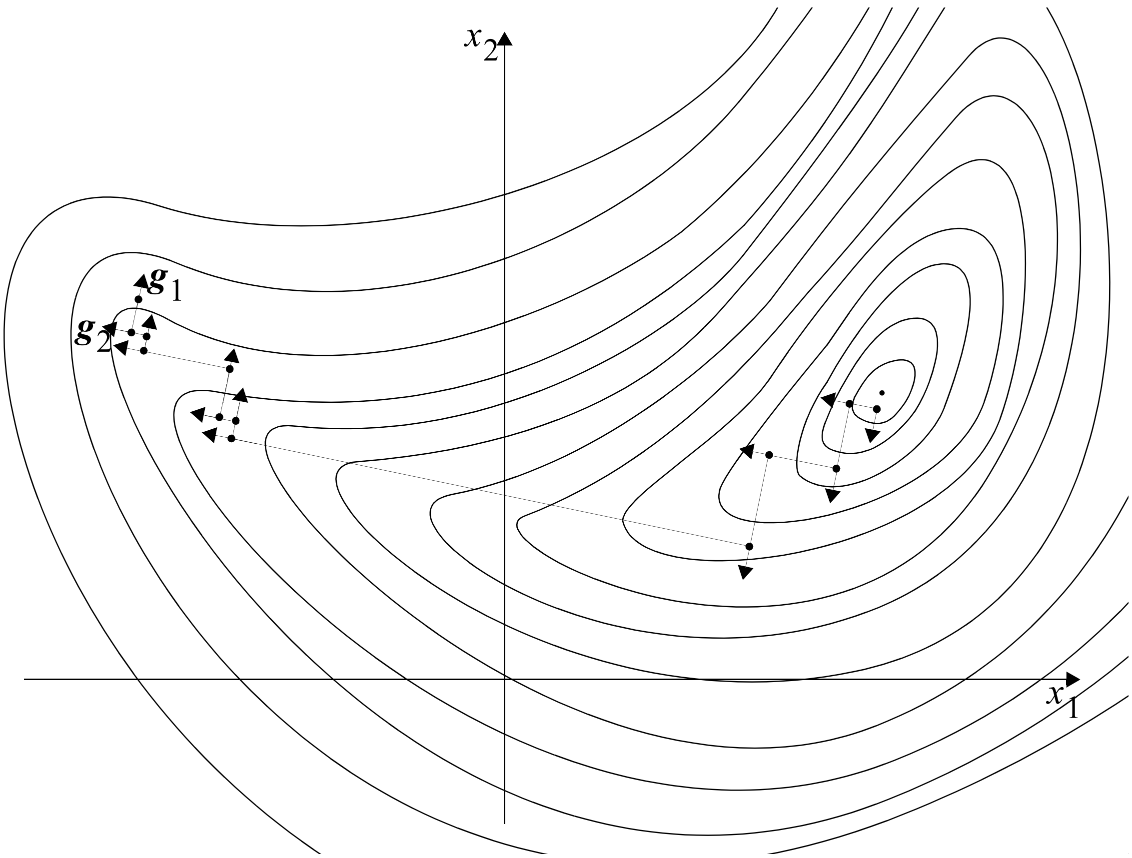
- We can find α_k by setting the derivative of $F(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{g}_k)$ with respect to α equal to zero, which from (1) we can write:

$$\frac{d}{d\alpha} F(\alpha) = (\nabla f(\boldsymbol{x}_{k+1}), \boldsymbol{g}_k) = (\boldsymbol{g}_{k+1}, \boldsymbol{g}_k) = 0 \quad . \tag{9}$$

this immediately shows the orthogonality between the successive directions $\boldsymbol{g}_{k+1}$ and $\boldsymbol{g}_k$.

- Now, although it may seem like a good idea to minimize along the direction of steepest descent, it turns out that this method is not very efficient.

- For relatively complicated functions, this method will tend to zig-zag towards the minimum.



Zig-zaging effect of Cauchy's method.

- Note that since successive directions of descent are always orthogonal, in two dimensions the algorithm searches in only two directions.

- This is what makes the algorithm slow to converge to the minimum and is generally not recommended.

The Conjugate-Gradient Method

- It turns out that if we have access to the gradient of the objective function, then we can determine conjugate directions relatively efficiently.

- Recall that Powell's conjugate direction method requires n single variable minimizations per iteration in order to determine one new conjugate direction at the end of the iteration.

  This results in approximately n^2 line minimizations to find the minimum of a quadratic function.

- In the conjugate-gradient algorithm, access to the gradient of $f(\boldsymbol{x})$, that is $\boldsymbol{g}(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$, allows us to set up a new conjugate direction after every line minimization.

- Consider the quadratic function

$$Q(\boldsymbol{x}) = a + \boldsymbol{b}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T C\boldsymbol{x} \tag{10}$$

  where $\boldsymbol{x} \in \mathbb{R}^n$.

- We want to perform successive line minimizations along conjugate directions, say $\boldsymbol{s}(\boldsymbol{x}_k)$, where $\boldsymbol{x}_k$ is the current search point and we minimize to the next point as

$$x_{k+1} = \boldsymbol{x}_k + \lambda_k \boldsymbol{s}(\boldsymbol{x}_k). \tag{11}$$

- Now, how do we find these conjugate directions

$$\boldsymbol{s}_k = \boldsymbol{s}(\boldsymbol{x}_k)$$

  given previous information?

- Expand the search directions in terms of the gradient at the current point

$$\boldsymbol{g}_k = \nabla f(\boldsymbol{x}_k)$$

and a linear combination of the previous search directions:

$$\boldsymbol{s}_k = -\boldsymbol{g}_k + \sum_{i=0}^{k-1} \gamma_i \boldsymbol{s}_i \tag{12}$$

where we start with the initial search direction as the steepest descent direction

$$\boldsymbol{s}_0 = -\boldsymbol{g}_0 .$$

- The coefficients of the expansion, γ_i, $i = 1 \ldots k-1$, are to be chosen so that the $\boldsymbol{s}_k$ are ***C*-conjugate**.

- Therefore, we have

$$\boldsymbol{s}_1 = -\boldsymbol{g}_1 + \gamma_0 \boldsymbol{s}_0 = -\boldsymbol{g}_1 - \gamma_0 \boldsymbol{g}_0 \tag{13}$$

and we require that

$$(\boldsymbol{s}_1, C\boldsymbol{s}_0) = 0 = (-\boldsymbol{g}_1 - \gamma_0 \boldsymbol{g}_0, C\boldsymbol{s}_0) \tag{14}$$

but $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \lambda_0 \boldsymbol{s}_0$ which can be solved for $\boldsymbol{s}_0$ as

$$\frac{\boldsymbol{x}_1 - \boldsymbol{x}_0}{\lambda_0} = \boldsymbol{s}_0 = \frac{\Delta \boldsymbol{x}_0}{\lambda_0} \tag{15}$$

where $\Delta \boldsymbol{x}_0 = \boldsymbol{x}_1 - \boldsymbol{x}_0$ is the forward difference operator. Using this in (14) we have

$$\left(-\boldsymbol{g}_1 - \gamma_0 \boldsymbol{g}_0, C\frac{\Delta \boldsymbol{x}_0}{\lambda_0}\right) = 0 \tag{16}$$

but $\boldsymbol{g}(\boldsymbol{x}) = C\boldsymbol{x} + \boldsymbol{b}$ which means that we can set

$$\Delta \boldsymbol{g}_0 = \boldsymbol{g}_1 - \boldsymbol{g}_0 = C(\boldsymbol{x}_1 - \boldsymbol{x}_0) = C\Delta \boldsymbol{x}_0 \tag{17}$$

which substituting into (16), we have

$$\left(-\boldsymbol{g}_1 - \gamma_0 \boldsymbol{g}_0, \frac{\Delta \boldsymbol{g}_0}{\lambda_0}\right) = 0$$

$$(-\boldsymbol{g}_1, \Delta \boldsymbol{g}_0) = \gamma_0 (\boldsymbol{g}_0, \Delta \boldsymbol{g}_0)$$

and finally

$$\gamma_0 = -\frac{(\Delta \boldsymbol{g}_0, \boldsymbol{g}_1)}{(\Delta \boldsymbol{g}_0, \boldsymbol{g}_0)}. \tag{18}$$

- We can further reduce the numerator as follows:

$$(\Delta \boldsymbol{g}_0, \boldsymbol{g}_1) = (\boldsymbol{g}_1 - \boldsymbol{g}_0, \boldsymbol{g}_1) = (\boldsymbol{g}_1, \boldsymbol{g}_1) - (\boldsymbol{g}_0, \boldsymbol{g}_1) = \|\boldsymbol{g}_1\|^2 \tag{19}$$

since $(\boldsymbol{g}_0, \boldsymbol{g}_1) = 0$ (successive directions of minimization are orthogonal)

- The denominator can also be rewritten as

$$(\Delta \boldsymbol{g}_0, \boldsymbol{g}_0) = (\boldsymbol{g}_1 - \boldsymbol{g}_0, \boldsymbol{g}_0) = (\boldsymbol{g}_1, \boldsymbol{g}_0) - (\boldsymbol{g}_0, \boldsymbol{g}_0) = -\|\boldsymbol{g}_0\|^2 \tag{20}$$

and therefore the coefficient γ_0 can be calculated from

$$\gamma_0 = \frac{\|\boldsymbol{g}_1\|^2}{\|\boldsymbol{g}_0\|^2}. \tag{21}$$

- The conjugate direction, $\boldsymbol{s}_1$, can be calculated as

$$\boldsymbol{s}_1 = -\boldsymbol{g}_1 + \frac{\|\boldsymbol{g}_1\|^2}{\|\boldsymbol{g}_0\|^2} \boldsymbol{s}_0. \tag{22}$$

- Now although we derived this for $\boldsymbol{s}_1$, we could have derived it for any $\boldsymbol{s}_k$ to get

$$\boxed{\boldsymbol{s}_k = -\boldsymbol{g}_k + \frac{\|\boldsymbol{g}_k\|^2}{\|\boldsymbol{g}_{k-1}\|^2}\boldsymbol{s}_{k-1}} \tag{23}$$

- This represents the *Fletcher-Reeves* scheme (1964) for determining conjugate directions for search from the gradient at the current point, $\boldsymbol{g}_k$, the previous search direction, $\boldsymbol{s}_{k-1}$, and the magnitude of the gradient at the previous point, $\|\boldsymbol{g}_{k-1}\|$.

- Two alternative update equations are the *Hestenes-Stiefel* method (1952):

$$\boxed{\boldsymbol{s}_k = -\boldsymbol{g}_k + \frac{(\Delta\boldsymbol{g}_{k-1}, \boldsymbol{g}_k)}{(\Delta\boldsymbol{g}_{k-1}, \boldsymbol{s}_k)}\boldsymbol{s}_{k-1}} \tag{24}$$

and the *Polak-Ribière* method (1969):

$$\boxed{\boldsymbol{s}_k = -\boldsymbol{g}_k + \frac{(\Delta\boldsymbol{g}_{k-1}, \boldsymbol{g}_k)}{\|\boldsymbol{g}_{k-1}\|^2}\boldsymbol{s}_{k-1}} \tag{25}$$

- All three schemes are identical for quadratic functions but will be different for non-quadratic functions.

- For quadratic functions, these methods will find the exact minimum in n iterations.

- For non-quadratic functions, the search direction is reset to the steepest descent direction every n iterations.

Algorithm: Conjugate Gradient Method (*Fletcher-Reeves*)

1. input: $f(\boldsymbol{x})$, $\boldsymbol{g}(\boldsymbol{x})$, $\boldsymbol{x}_0$, g_{tol},k_{max}, n
2. set: $k = 0$, $\boldsymbol{x} = \boldsymbol{x}_0$
3. set: $\boldsymbol{g}_{old} = \boldsymbol{g}_{new} = \boldsymbol{g}(\boldsymbol{x})$
4. set: $s_{old} = \boldsymbol{s}_{new} = \boldsymbol{g}_{old}$
5. while $\|\boldsymbol{g}_{new}\| > g_{tol}$ and $k < k_{max}$
6. $k = k + 1$
7.   for $i = 1(1)n$
8.   if $i = 1$ then
9.   set: $\boldsymbol{s}_{new} = -\boldsymbol{g}_{new}$
10.   else
11.   set: $\gamma = \|\boldsymbol{g}_{new}\|^2 / \|\boldsymbol{g}_{old}\|^2$
12.   set: $\boldsymbol{s}_{new} = -\boldsymbol{g}_{new} + \gamma\boldsymbol{s}_{old}$
13.   end
14.   set: $\lambda = \arg\min_{\lambda} f(\boldsymbol{x} + \lambda\boldsymbol{s}_{new})$
15.   set: $\boldsymbol{x} = \boldsymbol{x} + \lambda\boldsymbol{s}_{new}$
16.   set: $\boldsymbol{g}_{old} = \boldsymbol{g}_{new}$, $\boldsymbol{g}_{new} = \boldsymbol{g}(\boldsymbol{x})$
17.   set: $\boldsymbol{s}_{old} = \boldsymbol{s}_{new}$
18.   end
19. end

- Notes: line 11 contains the *Fletcher-Reeves* update scheme.

- This can be changed to use the *Hestenes-Stiefel* or the *Polak-Ribière* as desired (*i.e.* (24) or (25) respectively); the algorithm terminates once the norm of the gradient is smaller than a user specified value g_{tol} or the number of iterations grow beyond k_{max}.