Unconstrained Optimization

Multivariable function minimization in the absence of any restrictions.

Victor Ivamoto

May, 2020

Contents

Chapter 1

Introduction

Function minimization is a very common problem in machine learning algorithms and several mathematical methods were developed to solve this problem. This report describes the implementation of some methods and compares the performance and features of each of them.

Consider the optimization problem to minimize the function $f(x_1, x_2)$ given by:

$$\text{minimize f}(x_1, x_2) = x_1^2 + 2x_2^2 - 2x_1x_2 - 2x_2$$

We solve this problem using unconstrained optimization algorithms. The contour map shows the range of $f(x_1, x_2)$ in color and each line has the same value, where dark blue indicates the lowest levels.

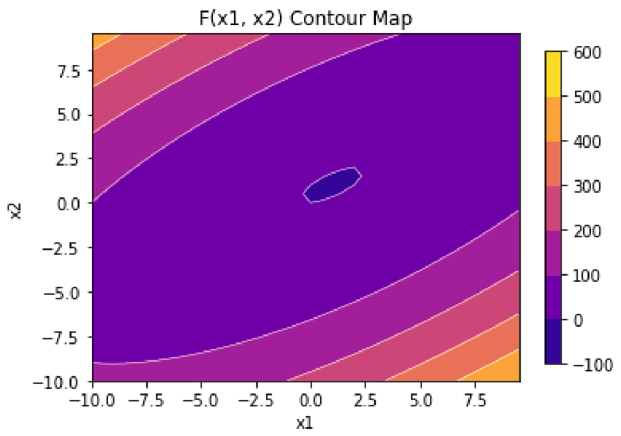

Figure 1.1: Function contour map.

In general, all methods start with arbitrary initial values for x_1 and x_2 and with an estimation of the step size and the direction of the minimum. Then, the new position is calculated, moving towards the minimum value until a certain criteria is achieved. Hence, the general formula is given by:

$$x_{i+1} = x_i + \alpha_i \delta_i \tag{1.1}$$

where x_i is the value of x_1 and x_2 in iteration i, α_i is the step size and δ_i is the direction.

A function $f(x_i)$ is at minimum or maximum when $\|\bigtriangledown f(x_i)\| = 0$ and $x_i =$ arg min$f(x)$. Finding the exact values of x_i that satisfies this condition may require many iterations, so the stop criteria may be the number of iterations or the gradient size becomes smaller than a threshold, $\|\bigtriangledown f(x_i)\| < \epsilon$.

In order to compare the performance of different methods, we use standard initial and stop values, which are $(x_1, x_2) = (-6.16961099, 2.44217542)$, $\epsilon = 10^{-3}$ and maximum number of iterations $itmax = 100$.

The number of iterations may increase significantly with lower gradient size chosen as stop criteria. The algorithm moves around the minimum, varying the gradient size until the minimum ϵ is achieved.

Chapter 2

Gradient Descent

The gradient of a function, $\bigtriangledown f(x)$, points to the direction of maximum value and $-\bigtriangledown f(x)$ to the minimum. This is the simplest method, but may require many iterations as the direction may change from point to point.

Algorithm:

```
1. Define a small value for $\epsilon > 0$ as tolerance.
2. Set arbitrary values for $x_0$
3. Set initial value $i = 0$
4. **While** $\|\bigtriangledown f(x_i)\| > \epsilon$ **do**
5.     Let $\alpha_i$ be the solution to minimize $f(x_i - \alpha_i \delta_i)$ subject to $\alpha_i > 0$
6.     Compute $x_{i+1} = x_i - \alpha_i \bigtriangledown f(x_i)$
7.     Let $i = i + 1$
```

Gradient descent used 14 iterations to find the minimum with resolution better than $\|\bigtriangledown f(x)\| < 10^{-3}$. Notice that the gradient size increased in iterations 12 and 13, as highlighted in Table 2.1.

The chart shows the direction changes and the largest minimization in the first four iterations.

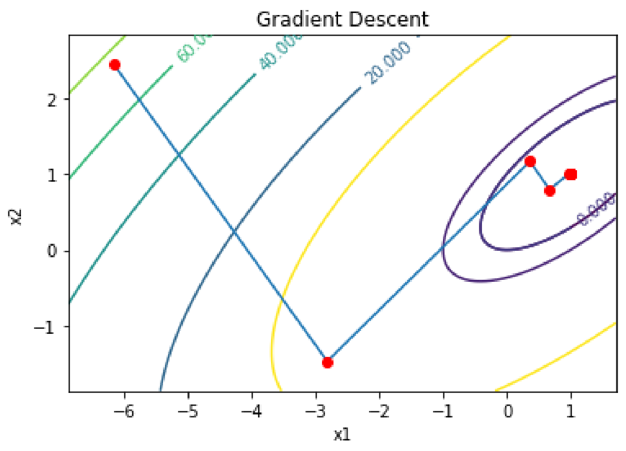

Figure 2.1: Gradient descent

Table 2.1: Gradient Descent Iteration Result.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	26.47603
1	-2.82980	-1.45693	75.24274	26.47603
2	0.51620	1.18517	6.92130	3.49855
3	0.77154	0.85915	-0.51819	2.16987
4	1.01052	1.00434	-0.97249	0.20504
5	1.00685	1.00543	-0.99994	0.01290
6	1.00620	1.00360	-0.99997	0.00852
7	1.00125	1.00170	-0.99998	0.00557
8	1.00140	1.00100	-1.00000	0.00440
9	1.00109	1.00053	-1.00000	0.00145
10	1.00058	1.00056	-1.00000	0.00112
11	1.00056	1.00007	-1.00000	0.00106
12	1.00013	1.00044	-1.00000	0.00129
13	1.00021	1.00025	-1.00000	0.00162
14	1.00022	1.00015	-1.00000	0.00059

Chapter 3

Bisection

The bisection method is often used in combination with other methods to optimize the step size α and speed up the convergence. The problem of minimization of $f(x)$ is equivalent to minimize

$$h(\alpha) = f(x + \alpha\delta) \tag{3.1}$$

where δ is the direction calculated by other methods. The first order derivative of $h(\alpha)$ is

$$h'(\alpha) = \bigtriangledown f(x + \alpha\delta)^{\top}\delta \tag{3.2}$$

The value of α that minimizes (3.1) lies in the interval $[\alpha_l, \alpha_u]$, such that $h'(\alpha_l) < 0$, $h'(\alpha_u) > 0$ and $h'(\alpha) = 0$. We can gradually adjust this interval until we find α

$$\begin{cases} \alpha = \bar{\alpha} & \text{if } h'(\bar{\alpha}) = 0 \\ \alpha \in [\alpha_l, \bar{\alpha}] & \text{if } h'(\bar{\alpha}) > 0 \\ \alpha \in [\bar{\alpha}, \alpha_u] & \text{if } h'(\bar{\alpha}) < 0 \end{cases} \tag{3.3}$$

where

$$\bar{\alpha} = \frac{\alpha_u + \alpha_l}{2} \tag{3.4}$$

Since δ is pointing to min$f(x)$, we know that $h'(0) < 0$ and select $\alpha_l = 0$. A simple way to choose α_u is select any starting positive value and double until $h'(\alpha_u) > 0$.

The maximum number of iterations to find $|\alpha - \bar{\alpha}| < \epsilon \approx 0$ is given by:

$$\left\lceil log_2\left(\frac{\bar{\alpha}}{\epsilon}\right)\right\rceil$$

Algorithm:

1. Define $\epsilon \approx 0$ such that $|\alpha - \bar{\alpha}| < \epsilon$.
2. Define $h'(\alpha)_{min} \approx 0$ as tolerance.
3. Choose a small random value for $\alpha_u > 0$
4. **while** $\|h'(\alpha_u)\| < 0$ **do**

```
 5.     Compute $\alpha_u = 2\alpha_u$
 6. Calculate maximum iterations $it_{max} = \lceil log_2\left(\frac{\bar{\alpha}}{\epsilon}\right)\rceil$
 7. Let $i = 0$
 8. **while** $|h'(\bar{\alpha})| > h'(\alpha)_{min}$ **and** $i < it_{max}$ **do**
 9.     **if** $h'(\bar{\alpha}) = 0$ **stop**
10.     **if** $h'(\bar{\alpha}) > 0$ **do** $\alpha_u = \bar{\alpha}$
11.     **if** $h'(\bar{\alpha}) < 0$ **do** $\alpha_l = \bar{\alpha}$
12.     Let $i = i + 1$
```

Chapter 4

Newton

Newton's method uses the Hessian, or second order derivative, to find the minimum. It requires less iterations than gradient descent.

The second order Taylor series approximation of $f(x - x_i)$ at point x_i is given by

$$f(x) \approx h(x) = f(x_i) + \bigtriangledown f(x_i)^\top (x - x_i) + \frac{1}{2}(x - x_i)^\top H(x_i)(x - x_i) \tag{4.1}$$

The minimum value of (4.1) is when the gradient equals to zero:

$$\bigtriangledown h(x) = \bigtriangledown f(x_i) + H(x_i)(x - x_i) = 0 \tag{4.2}$$

Solving this equation for x, we find the next value of x_{i+1}

$$x_{i+1} = x_i - H(x_i)^{-1} \bigtriangledown f(x_i) \tag{4.3}$$

The similarity with (1.1) is evident, however note that $\delta_i = -H(x_i)^{-1} \bigtriangledown f(x_i)$ may not be a descent direction. Moreover, (4.3) assumes the Hessian matrix is invertible and positive definite and $f(x)$ is continuously twice differentiable. In general, the Newton method may not converge if $H(x)$ is not invertible.

Algorithm:

```
1. Define a small value for $\epsilon > 0$ as tolerance.
2. Set arbitrary values for $x_0$
3. Set initial value $i = 0$, $\alpha = 1$
4. **While** $\|\bigtriangledown f(x)\| > \epsilon$ **do**
5.     Compute $\delta_i = -H(x_i)^{-1} \bigtriangledown f(x_i)$
6.     (optional) let $\alpha_i$ be the solution to minimize $f(x_i + \alpha_i \delta_i)$ subject to $\alpha_i > 0$
7.     Compute $x_{i+1} = x_i + \alpha \delta_i$
8.     Let $i = i + 1$
```

Newton's method converges in just two iterations and the plot shows the algorithm goes direct to the minimum.

Table 4.1: Newton Method.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	7.31322
1	1.00000	1.00000	75.24274	7.31322
2	1.00000	1.00000	-1.00000	0.00000

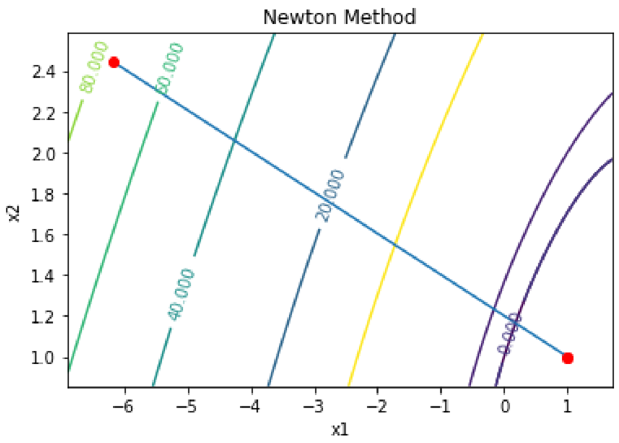

Figure 4.1: Newton method

4.1 Modified Newton

In some cases the Hessian matrix isn't positive definite and the Newton's method doesn't apply. In this case, we can replace $H(x)$ to

$$\begin{cases} M(x_i) = H(x_i) & \text{if } \lambda_{i,min} > 0 \\ M(x_i) = H(x_i) + (\epsilon - \lambda_{i,min}) & \text{if } \lambda_{i,min} \leq 0 \end{cases}$$

where $\lambda_{i,min}$ is the minimum eigenvalue of $H_i(x)$.

In our original problem to minimize (1.1), the minimum eigenvalue λ_{min} of $H(x_1, x_2)$ is positive, and the result is identical to Newton's method.

Table 4.2: Newton Modified Iteration Result.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	7.31322
1	1.00000	1.00000	75.24274	7.31322
2	1.00000	1.00000	-1.00000	0.00000

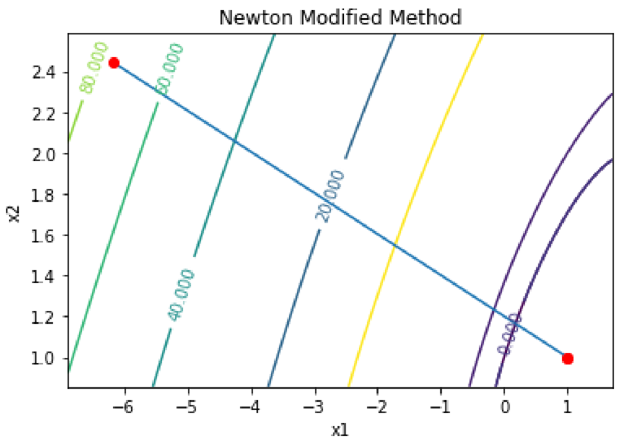

Figure 4.2: Newton modified

Chapter 5

Levenberg-Marquardt

Newton method assumes the Hessian matrix is invertible and the complexity of inverting $O(n^3)$ may become less efficient than gradient descent.

The Levenberg and Marquardt method provides fast convergence using first order derivative when $f(x_i)$ with $i = 1, .., n$ can be written in the form:

$$f(x) = \frac{1}{2}\sum_{k=1}^{m} r_k^2(x), \begin{cases} m \geq n, \\ r_k(x) : \mathbb{R}^n \to \mathbb{R} \end{cases}$$

Using calculus, the Hessian matrix of this function can be approximated to

$$H(x) \approx \bigtriangledown r(x)^\top \bigtriangledown r(x)$$

and the gradient of $f(x)$ is

$$\bigtriangledown f(x) = \bigtriangledown r(x)^\top r(x)$$

Hence, the new value of x_i is given by:

$$x_{i+1} = x_i - [\bigtriangledown r(x_i)^\top \bigtriangledown r(x_i)]^{-1} \bigtriangledown r(x_i)^\top \bigtriangledown r(x_i)$$

In our case, we can rewrite (1.1) as

$$\begin{aligned} f(x_1, x_2) &= x_1^2 + 2x_2^2 - 2x_1x_2 - 2x_2 \\ &= (x_1 - x_2)^2 + (x_2 - 1)^2 \\ &\approx r_1^2 + r_2^2 \end{aligned}$$

Where

$$\begin{cases} r_1 &= x_1 - x_2 \\ r_2 &= x_2 - 1 \end{cases}$$

Algorithm:

```
1. Define a small value for $\epsilon > 0$ as tolerance.
2. Set arbitrary values for $x_0$
3. Define small arbitrary value for $\mu > 0$
4. Set initial value $i = 0$
5. **While** $\|\bigtriangledown f(x)\| > \epsilon$ **do**
6.     Compute $\delta_i = -(\bigtriangledown r(x_i)^\top \bigtriangledown r(x_i) + \mu I)^{-1} \bigtriangledown r(x_i)^\top r(x_i)$
7.     Let $\alpha_i$ be the solution to minimize $f(x_i + \alpha_i \delta_i)$ subject to $\alpha_i > 0$
8.     Calculate $x_{i+1} = x_i + \alpha_i \delta_i$
9.     Set $i = i + 1$
```

The algorithm converges in three iterations.

Table 5.1: Levenberg-Marquardt Method Iteration Result.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	7.31321
1	1.00992	0.99800	-0.99985	7.31321
2	1.00097	0.99980	-1.00000	0.01012
3	1.00029	0.99994	-1.00000	0.00099
4	1.00008	0.99998	-1.00000	0.00029

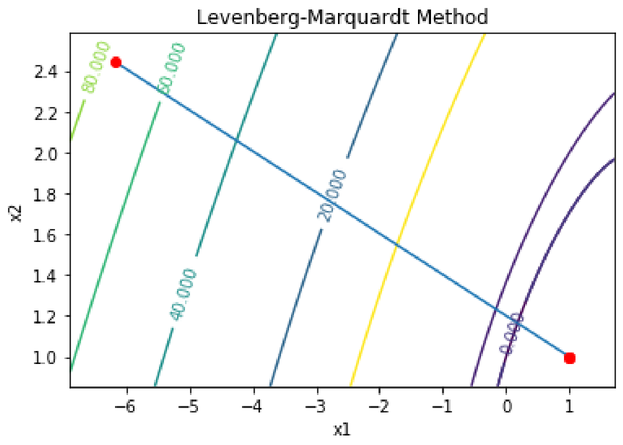

Figure 5.1: Levenberg-Marquardt

Chapter 6

Quasi-Newton Methods

These methods approximates the inverse of the Hessian on each iteration, in an attempt to overcome Newton's method weaknesses.

$$\lim_{i \to \text{inf}} M_i = [H(x)]^{-1}$$

Davidson-Fletcher-Powell - DFP

$$M_{i+1} = M_i + \frac{p_p i_i^\top}{p_i^\top q_i} - \frac{M_i q_i q_i^\top M_i}{q_i^\top M_i q_i}, \quad i = 0, 1, ..., n \tag{6.1}$$

Broyden-Fletcher-Goldfarb-Shanno - BFGS

$$M_{i+1} = M_i + \frac{p_i p_i^\top}{p_i^\top q_i} \left[1 + \frac{q_i^\top M_i q_i}{p_i^\top q_i}\right] - \frac{M_i q_i p_i^\top + p_i q_i^\top M_i}{p_i^\top q_i}, \quad i = 0, 1, ..., n \tag{6.2}$$

where:

$$p_i = \alpha_i d_i = y_{i+1} - y_i \tag{6.3}$$

$$q_i = \bigtriangledown f(y_{i+1}) - \bigtriangledown f(y_i) \tag{6.4}$$

Algorithm:

```
1. Define a small value for $\epsilon > 0$ as tolerance.
2. Set arbitrary values for $x_0$
3. Define initial symmetric positive definite matrix $M$.
4. Let $y_0 = x_0$, $k = j = 1, i = 0$
5. while $\|\bigtriangledown f(y)\| > \epsilon$ do
6.     $\delta_i = -M \bigtriangledown f(y_i)$
7.     let $\alpha_i$ be the solution to minimize $f(y_i + \alpha_i \delta_i)$ subject to $\alpha_i > 0$
8.     calculate $y_{i+1} = y_i + \alpha_i \delta_i$
9.     if $j < n$ do
```

```
10.         $p_i = \alpha_i \delta_i$
11.         $q_i = \bigtriangledown f(y_{i+1}) - \bigtriangledown f(y_i)$
12.         Use (6.1) or (6.2) to calculate $M_{i+1}$
13.         Let $j = j + 1$
14.     **else if** $j = n$ **do**
15.         $x_i = y_{i+1}$, $j = 1$, $k = k + 1$, $M = I$, where $I$ is the identity matrix.
16.     set $i = i + 1$
```

DFP converged in 7 iterations and BFGS converged in 3 iterations.

Table 6.1: Davidon-Fletcher-Powell.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	26.47603
1	-6.16961	2.44218	75.24274	26.47603
2	1.01009	0.99164	6.92376	3.48940
3	1.01009	0.99164	-0.99959	0.06507
4	1.00117	1.00180	-0.99998	0.00342
5	1.00117	1.00180	-1.00000	0.00503
6	1.00067	1.00047	-1.00000	0.00136
7	1.00067	1.00047	-1.00000	0.00068

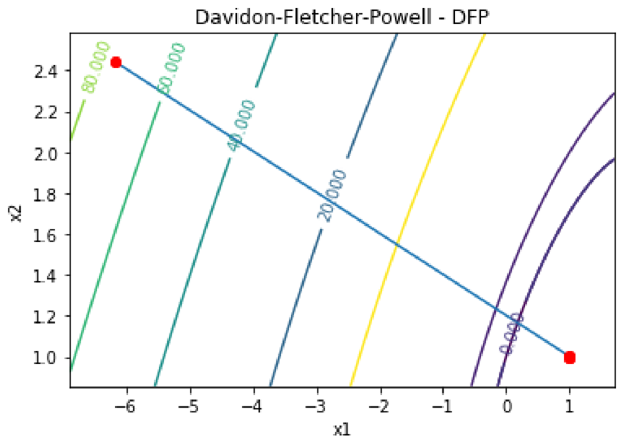

Figure 6.1: Davidon-Fletcher-Powell

Table 6.2: Broyden-Fletcher-Goldfarb-Shanno.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	26.47603
1	-6.16961	2.44218	75.24274	26.47603
2	1.00019	0.99906	6.91942	3.54554
3	1.00019	0.99906	-1.00000	0.00472

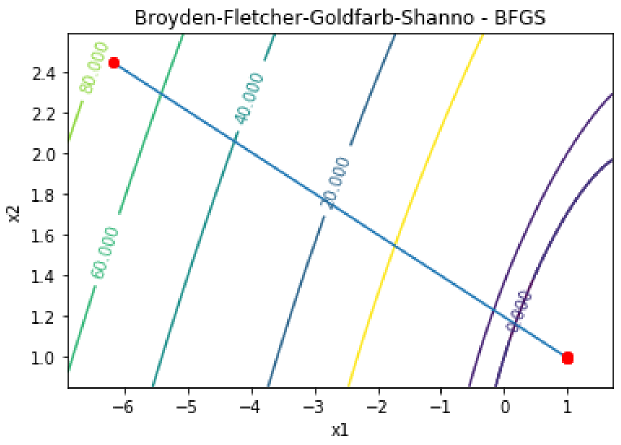

Figure 6.2: Broyden-Fletcher-Goldfarb-Shanno

Chapter 7

One Step Secant

Since the quasi-Newton algorithms require more storage and computation in each iteration than the conjugate gradient algorithms (explained later), there is need for a secant approximation with smaller storage and computation requirements. The one step secant (OSS) method is an attempt to bridge the gap between the conjugate gradient algorithms and the quasi-Newton (secant) algorithms. This algorithm does not store the complete Hessian matrix; it assumes that at each iteration, the previous Hessian was the identity matrix. This has the additional advantage that the new search direction can be calculated without computing a matrix inverse.[1]

$$A_i = -\left[1 + \frac{q_i^\top q_i}{s_i^\top q_i}\right] \frac{s_i^\top \bigtriangledown f(x_i)}{s^\top q_i} + \frac{q_i^\top \bigtriangledown f(x_i)}{s_i^\top q_i} \tag{7.1}$$

$$B_i = \frac{s_i^\top \bigtriangledown f(x_i)}{s_i^\top q_i} \tag{7.2}$$

$$\delta_{i+1} = -\bigtriangledown f(x_i) + A_i s_i + B_i q_i \tag{7.3}$$

where:

$$s_i = x_{i+1} - x_i = p_i, q_i = \bigtriangledown f(x_{i+1}) - \bigtriangledown f(x_i), p_i = \alpha_i \delta_i \tag{7.4}$$

Algorithm:

```
1. Define a small value for $\epsilon > 0$ as tolerance.
2. Set arbitrary values for $x_0$
3. Let $\delta_0 = -\bigtriangledown f(x_0), i = 0$
4. **while** $\|\bigtriangledown f(x_i)\| > \epsilon$ **do**
5.     Use (7.3) to compute $\delta_{i+1}$
6.     **if** $mod(i, P) = 0$ **then** set $\delta_i = \bigtriangledown f(x_i)$
7.     Let $\alpha_i$ be the solution to minimize $f(y_i + \alpha_i \delta_i)$ subject to $\alpha_i > 0$
8.     $x_{i+1} = x_i + \alpha_i \delta_i$
9.     $s_i = \alpha_i \delta_i$
```

[1] http://matlab.izmiran.ru/help/toolbox/nnet/backpr11.html

```
10.   $q_i = \bigtriangledown f(x_{i+1}) - \bigtriangledown f(x_i)$
11.   Let $i = i + 1$
```

One step secant converged in 12 iterations, and the gradient norm increased in 2.

Table 7.1: One Step Secant.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	26.47603
1	-2.79278	-1.50015	6.92166	26.47603
2	-0.64741	0.86199	1.29732	3.58300
3	-0.65418	0.86814	1.33484	4.07871
4	0.87175	0.87246	-0.98373	1.46568
5	0.87175	0.87193	-0.98360	0.25365
6	1.06418	1.06355	-0.99596	0.18202
7	1.06418	1.06372	-0.99594	0.12583
8	0.99212	0.99254	-0.99994	0.09114
9	0.99188	0.98854	-0.99986	0.01411
10	0.99704	0.99554	-0.99998	0.01453
11	0.99839	0.99017	-0.99984	0.01229
12	0.99936	0.99944	-1.00000	0.00954

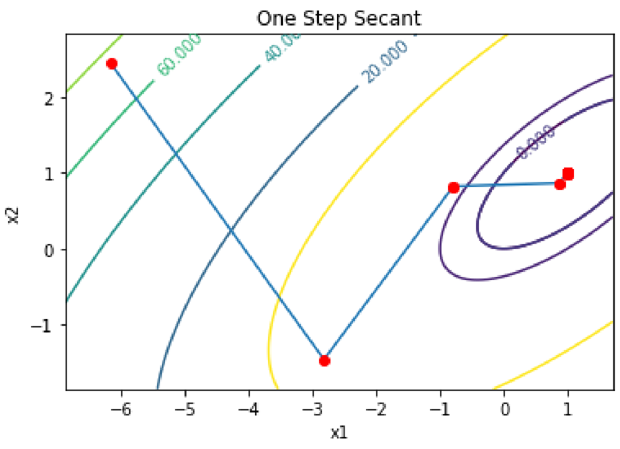

Figure 7.1: One step secant

Chapter 8

Conjugate Gradient Methods

These methods are less efficient than quasi-Newton's, however the lower storage requirement compared to the Hessian matrix make them attractive for large problems.[1]

They converge in at most n iterations for unconstrained quadratic problems in R^n. Non quadratic functions can be approximated to quadratic using Taylor series, benefiting from these methods as well.

The basic approach is to create a sequence of y_i, such that

$$y_{i+1} = y_i + \alpha_i \delta_i$$

where

$$\delta_{i+1} = -\bigtriangledown f(y_{i+1}) + \lambda_i \delta_i$$

There are three methods to calculate λ, the Hestenes-Stiefel (HS), Polak-Ribiere (PR) and Fletcher-Reeves (FR) equations:

$$\lambda_i^{HS} = \frac{\bigtriangledown f(x_{i+1})^\top q_i}{\delta_i^\top q_i} \tag{8.1}$$

$$\lambda_i^{PR} = \frac{\bigtriangledown f(x_{i+1})^\top q_i}{\|\bigtriangledown f(x_i)\|^2} \tag{8.2}$$

$$\lambda_i^{FR} = \frac{\|\bigtriangledown f(x_{i+1})\|^2}{\|\bigtriangledown f(x_i)\|^2} \tag{8.3}$$

Algorithm:

1. Define a small value for $\epsilon > 0$ as tolerance.
2. Set arbitrary values for x_0
3. Set $y_0 = x_0$, $k = j = 1$, $i = 0$, $\delta_0 = -\bigtriangledown f(y_0)$

[1]"Nonlinear Programming"

```
4.  while $\|\bigtriangledown f(y_i)\| > \epsilon$ do
5.      Let $\alpha_i$ be the solution to minimize $f(y_i + \alpha_i \delta_i)$ subject to $\alpha_i > 0$
6.      Calculate $y_{i+1} = y_i + \alpha_i \delta_i$
7.      if $j < n$ do
8.          Compute $q_i = \bigtriangledown f(y_{i+1}) - \bigtriangledown f(y_i)$
9.          Use (8.1), (8.2) or (8.3) to calculate $\lambda_i$
10.         Compute $\delta_{i+1} = -\bigtriangledown f(y_{i+1}) + \lambda_i \delta_i$
11.         Let $j = j + 1$
12.     else
13.         Compute $x_k = y_{i+1}$, $\delta = -\bigtriangledown f(y_i)$, $j = 1$, $k = k + 1$
14.     Let $i = i + 1$
```

Conjugate gradient has more iterations with increase in gradient size compared to the previous methods.

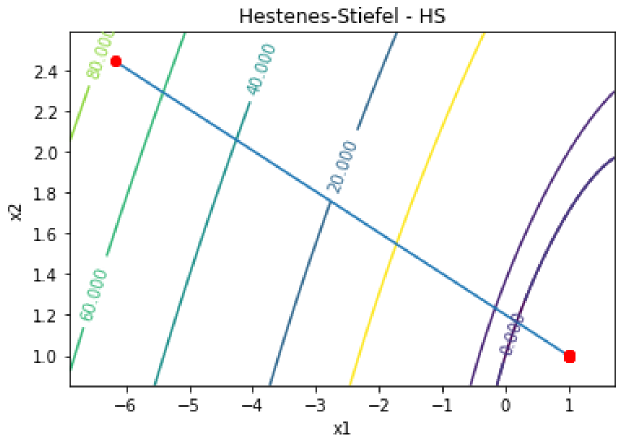

Figure 8.1: Hestenes-Stiefel

Table 8.1: Hestenes-Stiefel.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	26.47603
1	-6.16961	2.44218	75.24274	3.52095
2	0.97382	1.02457	-0.99682	3.49500
3	0.97382	1.02457	-0.99682	0.21293
4	1.01784	1.01453	-0.99978	0.18643
5	1.01784	1.01453	-0.99978	0.01744
6	1.00232	1.00299	-0.99999	0.01875
7	1.00232	1.00299	-0.99999	0.00223
8	1.00186	1.00115	-1.00000	0.00215
9	1.00186	1.00115	-1.00000	0.00293
10	1.00063	0.99974	-1.00000	0.00372
11	1.00063	0.99974	-1.00000	0.00022

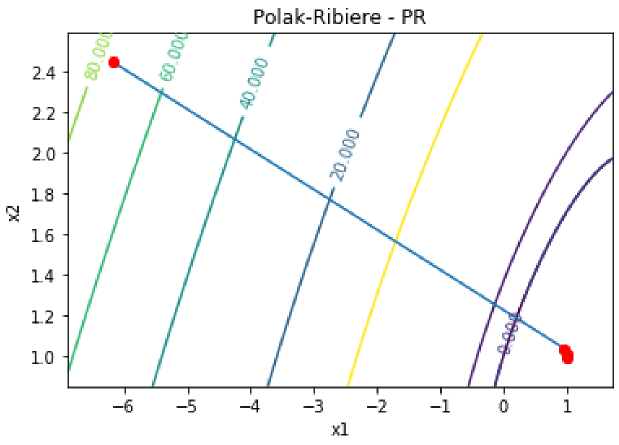

Figure 8.2: Polak-Ribiere

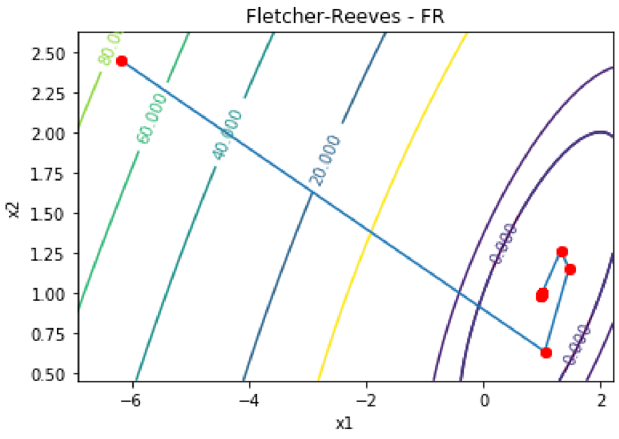

Figure 8.3: Fletcher-Reeves

Table 8.2: Polak-Ribiere.

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	26.47603
1	-6.16961	2.44218	75.24274	3.58515
2	1.04357	0.94876	-0.98839	3.56476
3	1.04357	0.94876	-0.98839	0.64392
4	0.98845	0.96030	-0.99763	0.26471
5	0.98845	0.96030	-0.99763	0.03591
6	0.99037	0.98388	-0.99970	0.03451
7	0.99037	0.98388	-0.99970	0.00265
8	0.99665	0.99831	-0.99999	0.00477
9	0.99665	0.99831	-0.99999	0.01755
10	1.00397	1.00502	-0.99997	0.00771
11	1.00397	1.00502	-0.99997	0.00362
12	1.00153	0.99898	-0.99999	0.00977
13	1.00153	0.99898	-0.99999	0.00224
14	0.99865	1.00012	-1.00000	0.00317
15	0.99865	1.00012	-1.00000	0.00138
16	1.00024	0.99910	-1.00000	0.00312
17	1.00024	0.99910	-1.00000	0.00007

Table 8.3: Fletcher-Reeves

Iteration	x1	x2	f(x)	Gradient Norm
0	-6.16961	2.44218	75.24274	26.47603
1	-6.16961	2.44218	75.24274	3.56099
2	0.63291	1.14015	-0.72306	3.57247
3	0.63291	1.14015	-0.72306	3.77743
4	1.13759	1.32987	-0.85422	1.65935
5	1.13759	1.32987	-0.85422	0.25106
6	1.17619	1.04292	-0.98040	0.23941
7	1.17619	1.04292	-0.98040	0.47123
8	0.98248	0.98542	-0.99978	0.36251
9	0.98248	0.98542	-0.99978	0.19268
10	0.98044	0.98820	-0.99980	0.01749
11	0.98044	0.98820	-0.99980	0.01105
12	0.99907	0.99757	-0.99999	0.00855
13	0.99907	0.99757	-0.99999	0.04155
14	1.00288	0.99589	-0.99993	0.02204
15	1.00288	0.99589	-0.99993	0.01644
16	0.99989	1.00077	-1.00000	0.01190
17	0.99989	1.00077	-1.00000	0.57070
18	0.99694	1.00531	-0.99990	0.02650
19	0.99694	1.00531	-0.99990	0.00987
20	1.00095	0.99870	-0.99999	0.00816
21	1.00095	0.99870	-0.99999	0.00313
22	1.00133	0.99801	-0.99999	0.01110
23	1.00133	0.99801	-0.99999	0.00327
24	0.99999	1.00004	-1.00000	0.00892

Chapter 9

Conclusion

The number of iterations shall be considered when choosing an optimization method, however more important than this is the computational cost, both in terms of memory usage and processing complexity.

During the search of the minimum, the gradient norm may increase from one iteration to the next, as highlighted in the tables.

Gradient descent is very simple method to minimize a function. It may require many iterations and direction changes to reach the minimum. Variable step size may be used to improve efficiency.

The bisection method can be used to calculate the optimum step size α on each iteration. This provides faster convergence than using fixed step size.

The Newton method is the most efficient, reaching the minimum with the least number of iterations, and it has more restrictions of all methods. Some concerns are the size of the Hessian and the complexity to invert it, the method may not convert and $f(x_{i+1})$ is not necessarily less than $f(x_i)$.

Quasi-Newton provide alternatives to Newton's method if the computational cost of computing the inverse of Hessian is high or doesn't' exist.

Conjugate gradient methods are less efficient and less robust than Newton's and quasi-Newton methods, but are more memory efficient and are preferred in large problems. They converge faster than gradient descent.

One step secant doesn't' store the Hessian matrix and is more computationally efficient than quasi-Newton and conjugate gradient methods.

Reference

Clodoaldo A. M. Lima, Norton Trevisan (2014) - “Aula 03 - Revisao sobre metodos de otimizacao”

M. S. Bazaraa, H. D. Sherali, C. M. Shetty. (2006) - “Nonlinear Programming - Theory and Algorithms” - 3rd edition

1994-2005 The MathWorks, Inc. - Neural Network Toolbox - http://matlab.izmiran.ru/help/toolbox/nnet/backpr11.html