# 1 Quadratic Objectives and Quasi-Newton (Section 11.4)

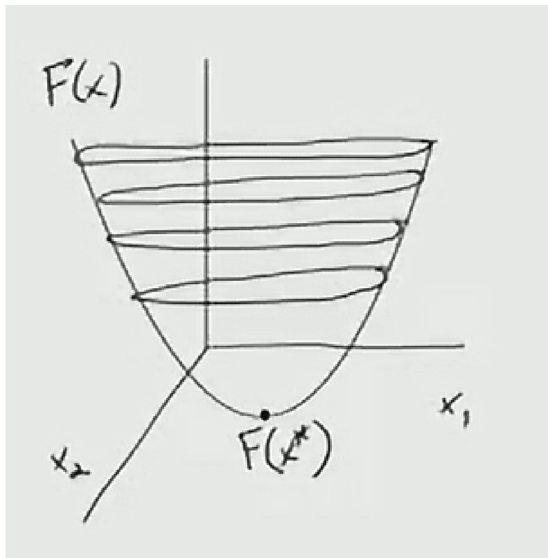
We now want to consider quadratic functions in $m$-dimensions,

$$a \in \mathbb{R}, \qquad b \in \mathbb{R}^m, \qquad A \in \mathbb{R}^{m\times m} \quad \text{(Symmetric.)}$$

The function of interest is

$$F(x) = a - b^T x + \frac{1}{2} x^T A x, \qquad x \in \mathbb{R}^m.$$

If $A$ is positive definite, then $F(x)$ is convex and it has a minimum $x^*$.




Note that the constant $a$ does not change the location of the minimum. In other words[1],

$$x^* = \arg\min_{x\in\mathbb{R}^m} F(x) = \arg\min_{x\in\mathbb{R}^m}\left(-b^T x + \frac{1}{2}x^T A x\right).$$

The given derivative is then

$$F(x) = a - \sum_{i=1}^m b_i x_i + \frac{1}{2} \underbrace{\sum_{i=1}^m \sum_{j=1}^m x_i A_{ij} x_j}_{\text{Corresponds to } x^T(Ax)} .$$

Its gradient is given by

$$\begin{aligned}
\frac{\partial F}{\partial x_k} &= \frac{d}{dx_k}\left(a - \sum_{i=1}^m b_i x_i + \frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m x_i A_{ij} x_j\right) \\
&= -b_k + \frac{1}{2}\sum_{j=1}^m A_{kj}x_j + \frac{1}{2}\sum_{i=1}^m x_i A_{ik} \\
&= -b_k + 2\left(\frac{1}{2}\right)\sum_{j=1}^m A_{kj}x_j \\
&= -b_k + \left(\frac{1}{2}\right)\sum_{j=1}^m A_{kj}x_j \qquad (1 \le k \le m).
\end{aligned}$$

[1] $\arg\min_{x\in\mathbb{R}^m} F(x)$ returns the $x$ value corresponding to the smallest $F(x)$.

Note that the above two sums basically produce the same results, but with different indexing. So, we were able to combine them.

$$\nabla F = \begin{bmatrix} \frac{\partial F}{\partial x_1} \\ \vdots \\ \frac{\partial F}{\partial x_m} \end{bmatrix} = \begin{bmatrix} -b_1 + \sum_{j=1}^m A_{ij} x_j \\ \vdots \\ -b_m + \sum_{j=1}^m A_{mj} x_j \end{bmatrix} = -b + Ax = g(x).$$

For $1 \leq i \leq m$ and $1 \leq k \leq m$, the Hessian matrix is given by

$$\frac{\partial^2 F}{\partial x_k \partial x_i} = \frac{d}{dx_i} \left( -b_k + \sum_{j=1}^m A_{kj} x_j \right) = A_{ki}.$$

Thus, we can write $\nabla^2 F = A = H(x)$, the entire Hessian matrix.

## 1.1 Computing a Minimum

Suppose the function is convex. If we have $A$ and we can solve for it, then to compute the minimum, we can set the gradient equal to zero:

$$0 = g(x) = -b + Ax \implies Ax = b \implies A^{-1} b = x.$$

But, in reality, $A$ might be too big, in which case we might consider iteratively solving the minimization problem. In any case, the line-search for quadratic functions can be computed explicitly, by fixing $x^{(k)} \in \mathbb{R}^m$ and $P \in \mathbb{R}^m$, and doing

$$\min_\alpha F(x^{(k)} + \alpha P).$$

We want to compute the directional derivative, which can be done by solving

$$\begin{aligned} 0 &= \frac{d}{d\alpha} F(x^{(k)} + \alpha P) \\ &= P^T \nabla F(x^{(k)} + \alpha P) \\ &= P^T \left( A(x^{(k)} + \alpha P) - b \right) \\ &= P^T (Ax^{(k)} - b) + \alpha P^T A P - P^T (Ax^{(k)} - b) \\ &= \alpha P^T A P. \end{aligned}$$

or just

$$\alpha_k = \frac{P^T (b - Ax^{(k)})}{P^T a P}.$$

$\alpha_k$ is the solution to the line-search problem. The "residue" is given by $r^{(k)} = b - Ax^{(k)}$.

(Example.) The steepest descent for quadratics is given by

$$P^{(k)} = -g(x^{(k)}) = -(Ax^{(k)} - b).$$

A very brief algorithm is shown below, which takes

- $x^{(0)}$, the starting point, and
- $M$, the number of iterations,

as the arguments.

**Algorithm 1** Steepest Descent

**function** STEEPEST($x^{(0)}, M$)<br>
    $k \leftarrow 0$<br>
    **while** $k < M$ **do**<br>
        $x^{(k+1)} = x^{(k)} + \alpha_k P^{(k)}$<br>
    **end while**<br>
**end function**

## 1.2 Quasi-Newton Methods

An effective class of methods to now consider is the **Quasi-Newton**. For general nonlinear $F : \mathbb{R}^m \to \mathbb{R}$, we want to only use the gradient function $g(x)$. In other words, we don't need the Hessian matrix here, as computing a Hessian matrix can be quite expensive. So, we can either *approximate* the Hessian matrix[2], $H(x_k) \approx B_k$, or approximate the inverse of the Hessian matrix $(H(x_k))^{-1} \approx \hat{H}_k$. These approximations satisfy the secant condition,

$$s_k = x_{k+1} - x_k \in \mathbb{R}^m.$$

$$y_j = g_{j+1} - g_j \in \mathbb{R}^m.$$

We can make use of an algorithm that makes use of this concept. This algorithm, known as the Davidon-Fletcher-Powell Method, takes the following arguments:

- $x_0 \in \mathbb{R}^m$, the starting point;
- $g$, the gradient of $F$;
- $F$, the function itself;
- $M$, the maximum number of iterations; and
- $\epsilon$, the tolerance.

The algorithm is as follows:

**Algorithm 2** Davidon-Fletcher-Powell Method

**function** DFP($x_0, g, F, M, \epsilon$)<br>
    $k \leftarrow 0$<br>
    $H_k \leftarrow I$<br>
    **while** $||g(x_k)||_2 > \epsilon$ and $k \leq M$ **do**<br>
        $P_k \leftarrow -H_k g(x_k)$<br>
        $\alpha_k \leftarrow \min_\alpha F(x_k + \alpha P_k)$ ▷ Line-Search Algorithm<br>
        $s_k \leftarrow \alpha_k P_k$<br>
        $x_{k+1} \leftarrow x_k + s_k$<br>
        $y_k \leftarrow g(x_{k+1}) - g(x_k)$<br>
        $H_k \leftarrow H_k + \frac{s_k s_k^T}{y_k^T s_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}$<br>
        $k \leftarrow k + 1$<br>
    **end while**<br>
**end function**

[2] $H(x_k)$ just means that we're evaluating the Hessian matrix at point $x_k$.