ϵ-step steepest descent

Setting: The variable x has n coordinates. $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$.

The function $f(x)$ returns a real value for each x.

Goal: find a value of x that minimizes $f(x)$.

The variants of the steepest descent algorithm that are described here get as input an estimate x for the minimum of f and return as output a better estimate for the minimum. They can be viewed as the following black box:

$$x_{\text{improved}} = \text{OneStep}(x)$$

They can be used to find the minimum as follows:

0. Start with x selected at random.

1. Repeat many times: $x = \text{OneStep}(x)$.

2. Produce the final x as output.

The following steepest descent algorithm implements OneStep when $g(x) = \nabla f(x)$ is known:

$$\text{improved } x = \text{OneStep}(x) = x - \epsilon\, g(x) \tag{1}$$

Observe that Algorithm 1 does not use f. It can be shown that for "well behaved" f the algorithm always converges to a local minimum when ϵ is small enough. The parameter ϵ is called **the learning rate**.

Algorithm 1 is written in vector notation. The same algorithm without vector notation is:

$$\text{improved } x_i = x_i - \epsilon\, \frac{\partial f}{\partial x_i} \tag{2}$$

Observe that Algorithm 2 does not use f but it needs the partial derivatives of f with respect to each one of the coordinates of x.

In many practical cases it is impossible or computationally expensive to compute partial derivatives explicitly. Instead, they can be approximated by finite differences. Let δ be a small number such that $\delta < \epsilon$. Let x^i_+ be the vector obtained by adding δ to the ith coordinate of x, and let x^i_- be the vector obtained by subtracting δ from the ith coordinate of x. They can be used to estimate $\frac{\partial f}{\partial x_i}$ as follows:

$$x^i_+ = \begin{pmatrix} x_1 \\ \vdots \\ x_i + \delta \\ \vdots \\ x_n \end{pmatrix}, \quad x^i_- = \begin{pmatrix} x_1 \\ \vdots \\ x_i - \delta \\ \vdots \\ x_n \end{pmatrix}, \qquad \frac{\partial f}{\partial x_i} \approx \frac{f(x^i_+) - f(x^i_-)}{2\delta} \tag{3}$$

Combining (3) with (2) gives an algorithm that does not need the partial derivatives but requires two calculations of f for each coordinate.