Optimization Techniques

Laboratory 3

Derivative-based optimization

UNIVERSITY OF TRENTO - Italy
Information Engineering
and Computer Science Department

Gradient Descent

Hiker algorithm that wants to climb down a hill

Each step is determined by:

- Steepness of the mountain (**gradient**)
- Leg length of the hiker (**learning rate**)

$$\nabla_x f(x) \in \mathbb{R}^n = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

$$x \leftarrow x - \alpha \nabla_x f(x)$$

Given:

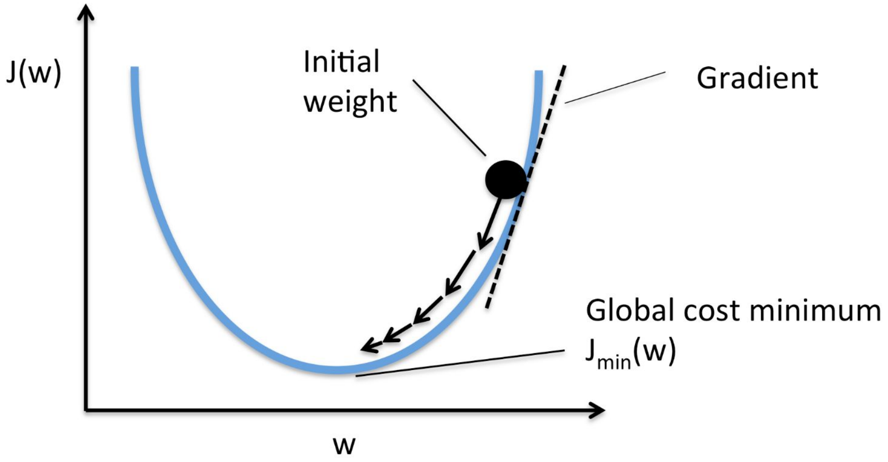
Function f, initial point x_0, step size $\alpha > 0$

Initialize:

$x \leftarrow x_0$

Repeat until convergence:

$x \leftarrow x - \alpha \nabla_x f(x)$



Newton Method

Second-order method ➛ approximates the right step-size

Uses the **steepness** and **curvature** to determine the next step

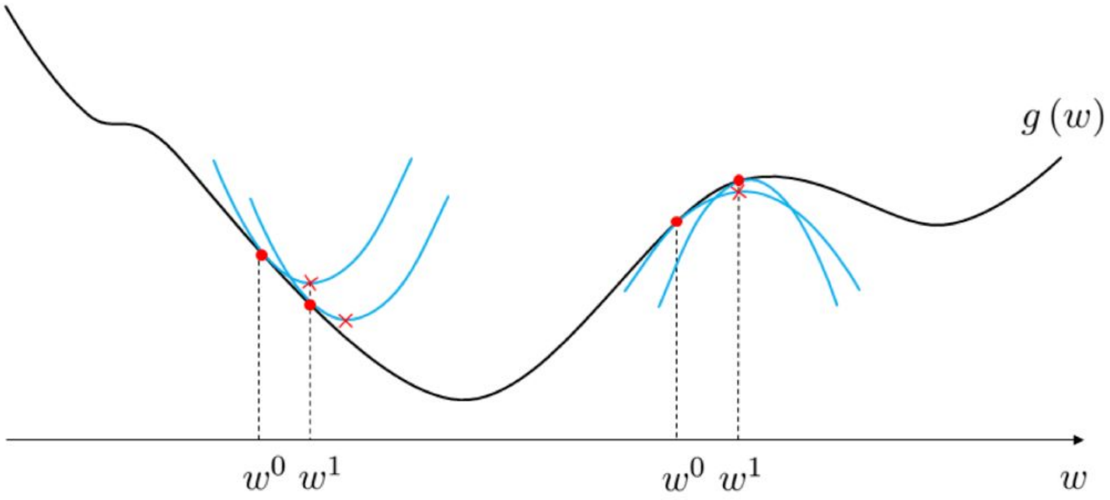
$$\boldsymbol{x}^* = \boldsymbol{x}^{(0)} - H(f)(\boldsymbol{x}^{(0)})^{-1}\nabla_{\boldsymbol{x}} f(\boldsymbol{x}^{(0)})$$

Pros:

- When *f* is quadratic, it jumps to the minimum straight away

Cons:

- Computational burden
- Attracts saddle points



Broyden–Fletcher–Goldfarb–Shanno (BFGS)

Limitation of Newton's method is that it requires the calculation of the inverse of the Hessian matrix

Quasi-Newton method ➛ **approximates the inverse of the Hessian matrix** using the gradient

BFGS uses a line search in the chosen direction to determine how far to move in that direction

$k = 0$ — Initialize iteration counter

$\alpha_{\text{init}} = 1$ — Initial step length for line search

while $\|\nabla f_k\|_\infty > \tau$ **do** — Optimality condition

  if $k = 0$ or reset = $\texttt{true}$ **then**

    $\tilde{V}_k = \frac{1}{\|\nabla f\|} I$

  else

    $s = x_k - x_{k-1}$ — Last step

    $y = \nabla f_k - \nabla f_{k-1}$ — Curvature along last step

    $\sigma = \frac{1}{s^\intercal y}$

    $\tilde{V}_k = \left(I - \sigma s y^\intercal\right) \tilde{V}_{k-1} \left(I - \sigma y s^\intercal\right) + \sigma s s^\intercal$ — Quasi-Newton update

  end if

  $p = -\tilde{V}_k \nabla f_k$ — Compute quasi-Newton step

  $\alpha = \texttt{linesearch}\left(p, \alpha_{\text{init}}\right)$ — Should satisfy the strong Wolfe conditions

  $x_{k+1} = x_k + \alpha p$ — Update design variables

  $k = k + 1$ — Increment iteration index

end while