Conjugate Gradient

Contents

Introduction

Conjugate gradient (CG) [1, 2] in its basic form is an iterative scheme to solve symmetric positive definite linear systems. CG can be seen as an iterative scheme to minimize strictly convex quadratic functions. It can be extended to non quadratic cost functions.

Let:

$$f(x) = \frac{1}{2}x^T Ax - bx + c, \, x \in \mathbb{R}^n$$

with A symmetric positive definite, then a stationary point of f is given by

$$\nabla f(x) = Ax - b = 0$$

which is equivalent to solving the linear system $Ax = b$.

Contrary to standard gradient descent, which uses at each iteration the "steepest" direction, without any use of previous iterations, CG is a multistep approach in the sense that the next direction is informed by the previous ones. This avoids the zig-zag of gradient descent with optimal step size, and is in practice often faster for ill-conditioned problems.

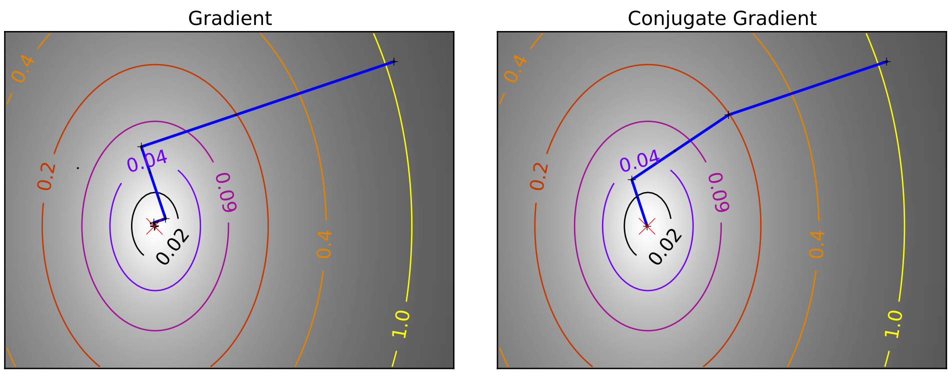

Figure 1: Gradient Descent and conjugate gradient on well conditioned problem.

1 Conjugate gradient for linear systems

Let $A \in \mathbb{S}^n_{++}$ a symmetric positive definite matrix. The scalar product associated with A is defined as:

$$\langle x, y \rangle_A = \langle Ax, y \rangle = x^T Ay$$

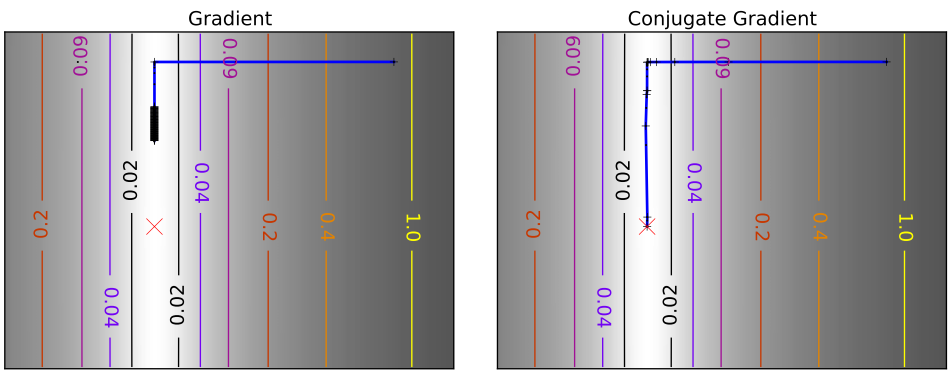

Figure 2: Gradient Descent and conjugate gradient on badly conditioned problem.

The CG method is descent method where the descent direction d_k is not equal to the gradient $g_k = Ax_k - b$, but the gradient g_k "corrected" such that all the directions d_k obtained are orthogonal, a.k.a., conjugate, for the dot product $\langle \cdot, \cdot \rangle_A$. More precisely:

$$d_k = g_k + \alpha_k d_{k-1},$$

where $\alpha_k \in \mathbb{R}$ is such that:

$$\langle d_k, d_{k-1} \rangle_A = 0$$

The conjugate gradient algorithm reads:

Algorithm 1: Conjugate gradient

```
Require: $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$
 1: $x_0 \in \mathbb{R}^n$, $g_0 = Ax_0 - b$
 2: for $k = 0$ to $n$ do
 3:   if $g_k = 0$ then
 4:     break
 5:   end if
 6:   if $k = 0$ then
 7:     $d_k = g_0$
 8:   else
 9:     $\alpha_k = -\frac{\langle g_k, Ad_{k-1} \rangle}{\langle d_{k-1}, Ad_{k-1} \rangle}$
10:     $d_k = g_k + \alpha_k d_{k-1}$
11:   end if
12:   $\rho_k = \frac{\langle g_k, d_k \rangle}{\langle d_k, Ad_k \rangle}$
13:   $x_{k+1} = x_k - \rho_k d_k$
14:   $g_{k+1} = Ax_{k+1} - b$
15: end for
16: return $x_{k+1}$
```

Theorem 1. *The conjugate gradient algorithm converges to an optimal solution of a quadratic function f, with $A \in \mathbb{R}^{n \times n}$ a symmetric definite positive matrix, in at most n iterations.*

PROOF. If $g_k = 0$, then $x_k = x^*$ is solution of the linear system $Ax = b$. For $k = 1$, we have $d_0 = g_0$, so:

$$\langle g_1, d_0 \rangle = \langle Ax_1 - b, d_0 \rangle = \langle Ax_0 - b, d_0 \rangle - \rho_0 \langle Ad_0, d_0 \rangle = \langle g_0, d_0 \rangle - \rho_0 \langle Ad_0, d_0 \rangle = 0 \quad (1)$$

by definition of ρ_0. This leads to

$$\langle g_1, g_0 \rangle = \langle g_1, d_0 \rangle = 0$$

and

$$\langle d_1, Ad_0 \rangle = \langle g_1, Ad_0 \rangle + \alpha_0 \langle d_0, Ad_0 \rangle = 0$$

by definition of α_0. One can prove the result by recurrence assuming that:

$$\begin{aligned}\langle g_k, g_j\rangle &= 0 \text{ for } 0 \le j < k\\ \langle g_k, d_j\rangle &= 0 \text{ for } 0 \le j < k\\ \langle d_k, Ad_j\rangle &= 0 \text{ for } 0 \le j < k\end{aligned} \tag{2}$$

If $g_k \neq 0$, the algorithm computes x_{k+1}, g_{k+1} and d_{k+1}.

- By construction one has $\langle g_{k+1}, d_k\rangle = 0$ (cf. (1)).
- For $j < k$:

  $$\langle g_{k+1}, d_j\rangle = \langle g_{k+1}, d_j\rangle - \langle g_k, d_j\rangle = \langle g_{k+1} - g_k, d_j\rangle = -\rho_k\langle Ad_k, d_j\rangle = 0 \text{ (recurrence hypothesis)}$$

- For $j \le k$:

  $$\langle g_{k+1}, g_j\rangle = \langle g_{k+1}, d_j\rangle - \alpha_j\langle g_{k+1}, d_{j-1}\rangle = 0 \ ,$$

  since $g_j = d_j - \alpha_j d_{j-1}$.
- Now: $d_{k+1} = g_{k+1} + \alpha_{k+1} d_k$. For $j < k$

  $$\langle d_{k+1}, Ad_j\rangle = \langle g_{k+1}, Ad_j\rangle + \alpha_{k+1}\langle d_k, Ad_j\rangle = \langle g_{k+1}, Ad_j\rangle \ .$$

  As $g_{j+1} = g_j - \rho_j Ad_j$, one obtains

  $$\langle g_{k+1}, Ad_j\rangle = \frac{1}{\rho_j}\langle g_{k+1}, g_j - g_{j+1}\rangle = 0 \text{ if } \rho_j \neq 0.$$

  This implies that if $\rho_j \neq 0$, $\langle d_{k+1}, Ad_j\rangle = 0$ for $j < k$.
- Furthermore one has $\langle d_{k+1}, Ad_k\rangle = 0$. So $\langle d_{k+1}, Ad_j\rangle = 0$ for $j < k + 1$.

This completes the proof for $\rho_j \neq 0$ and $g_j \neq 0$. However one has that

$$\langle g_k, d_k\rangle = \langle g_k, g_k\rangle + \alpha_k\langle g_k, d_{k-1}\rangle = \|g_k\|^2 \ ,$$

and $\rho_k = \frac{\langle g_k, d_k\rangle}{\langle Ad_k, d_k\rangle}$. So ρ_k can only be 0 if $g_k = 0$, which would imply that $x_k = x^*$.

Furthermore

$$\|d_k\|^2 = \|g_k\|^2 + \alpha_k^2\|d_{k-1}\|^2 \ .$$

So if $g_k \neq 0$ then $d_k \neq 0$. Consequently, if the vectors g_0, g_1, ..., g_k are all non-zero, the vectors d_0, d_1, ..., d_k are also non-zero. These vectors are an orthogonal basis for the dot product $\langle\cdot,\cdot\rangle_A$ and the $k+1$ directions g_0, g_1, ..., g_k are an orthogonal basis for the dot product $\langle\cdot,\cdot\rangle$. These directions are therefore independent. As a consequence, if g_0, g_1, ..., g_{n-1} are all non-zero, one has that $d_n = g_n = 0$, which demonstrates that algorithm has converged after n iterations at the most.

Remark 1. *In practice due to numerical precision issues, the test $g_k = 0$ is replaced by $\|g_k\| < \varepsilon$, where ε is a tolerance parameter.*

2 Conjugate gradient for general functions

The conjugate gradient algorithm can be extended to differentiable functions, non necessarily quadratic (See Algorithm 2). This method is also known as the method of Fletcher and Reeves.

This algorithm is motivated by the fact that in the quadratic case

$$\alpha_k = -\frac{\langle g_k, Ad_{k-1}\rangle}{\langle Ad_{k-1}, d_{k-1}\rangle} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}$$

Indeed, $Ad_{k-1} = \frac{g_{k-1} - g_k}{\rho_{k-1}}$ so that $\langle g_k, Ad_{k-1}\rangle = -\frac{\|g_k\|^2}{\rho_{k-1}}$. The same way:

$$\langle d_{k-1}, Ad_{k-1}\rangle = \frac{\langle d_{k-1}, g_{k-1}\rangle}{\rho_{k-1}} = \frac{\langle g_{k-1} + \alpha_{k-1} d_{k-2}, g_{k-1}}{\rho_{k-1}} = \frac{\|g_{k-1}\|^2}{\rho_{k-1}} \ .$$

See Figure 3 for an example on the Rosenbrock function.

Remark 2. *Conjugate gradient can be applied to non-quadratic problems yet it is generally prefered to use a quasi-newton method such as L-BFGS (cf. notes on Newton and quasi-Newton methods).*

Algorithm 2: Conjugate gradient

```
Require: $\varepsilon > 0$ (tolerance), $K$ (maximum number of iterations)
 1: $x_0 \in \mathbb{R}^n$, $g_0 = \nabla f(x_0)$
 2: for $k = 0$ to $K$ do
 3:     if $\|g_k\| < \varepsilon$ then
 4:         break
 5:     end if
 6:     if $k = 0$ then
 7:         $d_k = g_0$
 8:     else
 9:         $\alpha_k = -\frac{\|g_k\|^2}{\|g_{k-1}\|^2}$
10:         $d_k = g_k + \alpha_k d_{k-1}$
11:     end if
12:     if $\langle d_k, g_k \rangle > 0$ then
13:         $d_k = g_k$ (steepest descent)
14:     end if
15:     Optimize the step size $\rho_k$ so that it minimizes $f(x_k - \rho_k d_k)$ i.e.
            $$\langle \nabla f(x_k - \rho d_k), d_k \rangle = 0$$
16:     $x_{k+1} = x_k - \rho_k d_k$
17: end for
18: return $x_{k+1}$
```

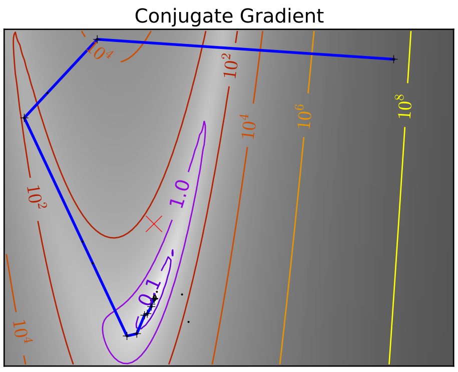

Figure 3: Conjugate gradient on non-quadratic problem (the Rosenbrock function).

References

[1] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 6, dec 1952.

[2] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006.