# The Conjugate Gradient Method

Lecture 5, Continuous Optimisation

Oxford University Computing Laboratory, HT 2006
Notes by Dr Raphael Hauser (hauser@comlab.ox.ac.uk)

The notion of complexity (per iteration) of an algorithm we used so far is simplistic:

- We counted the number of "basic computer operations", without taking into account that some operations are less costly than others.

- We did not take into account the memory requirements of an algorithm and the time a computer spends shifting data between different levels of the memory hierarchy.

**Memory requirement:**

- Quasi-Newton methods create need to keep a $n \times n$ matrix $H_k$ (the inverse of the approximate Hessian $B_k$) or $L_k$ (the Cholesky factor of $B_k$) in the computer memory, i.e., $O(n^2)$ data units.

- The steepest descent method only occupies $O(n)$ memory at any given time, by storing $x_k$ and $\nabla f(x_k)$ and overwriting registers with new data. Can cope with much larger $n$ than q.-N..

The *conjugate gradient method* has

- $O(n)$ memory requirement,

- $O(n)$ complexity per iteration,

- but converges much faster than steepest descent.

This method can be used when the memory requirement of quasi-Newton methods exceeds the active memory of the CPU, or alternatively, to solve positive definite systems of linear equations.

Let $B \succ 0$ be symmetric positive definite and consider

$$\text{(P)} \qquad \min_{x \in \mathbb{R}^n} f(x) = x^\top Bx + b^\top x + a.$$

Since $f$ is convex, $\nabla f(x) = 0$ is a sufficient optimality condition, i.e., (P) is equivalent to solving the positive definite linear system $2Bx = -b$ with solution

$$x^* = -(1/2)B^{-1}b.$$

Let $A \in \mathbb{R}^{n \times n}$ be real symmetric and recall:

- $A$ has real eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ and there exists $Q$ orthogonal such that $A = Q\,\mathrm{Diag}(\lambda)Q^\top$.
- $A^{-1} = QD^{-1}Q^\top$, i.e., $A$ is nonsingular iff $\lambda_i \neq 0\,\forall i$,
- $A$ is positive definite iff $\lambda_i > 0\,\forall i$, and then $A^{1/2} := Q\,\mathrm{Diag}(\lambda^{1/2})Q^\top$ is *unique* symmetric positive definite s.t. $A^{1/2}A^{1/2} = A$.

**Geometric motivation of CG:** Adding a constant to the objective function of

$$\text{(P)} \qquad \min_{x \in \mathbb{R}^n} f(x) = x^\top Bx + b^\top x + a$$

does not change the global minimiser $x^* = -(1/2)B^{-1}b$.

Therefore, it is equivalent to solve

$$\text{(P')} \qquad \min f(x) = (x - x^*)^\top B(x - x^*) = y^\top y = g(y),$$

where $y = B^{1/2}(x - x^*)$.

Thus, the objective function of our minimisation problem looks particularly simple in the transformed variables $y$. Use these to understand the geometry of the method.

We aim to construct an iterative sequence $(x_k)_{k \in \mathbb{N}}$ such that the corresponding sequence of $y_k = B^{1/2}(x_k - x^*)$ behaves sensibly.

Let the current iterate be $x_k$ and apply an exact line search $\alpha_k = \arg\min_\alpha f(x_k + \alpha d_k)$ to $x_k$ in the search direction $d_k$.

Translated into $y$-coordinates,

$$\alpha_k = \arg\min_\alpha g(y_k + \alpha p_k) = \arg\min_\alpha \|y_k\|^2 + 2\alpha p_k^\top y_k + \alpha^2 \|p_k\|^2.$$

where $p_k = B^{\frac{1}{2}} d_k$, and

$$\alpha_k = -\frac{p_k^\top y_k}{\|p_k\|^2}.$$

If we set $y_{k+1} = y_k + \alpha_k p_k$, then we find

$$y_{k+1}^\top p_k = \left( y - \frac{p_k^\top y_k}{\|p_k\|^2} p_k \right)^\top p_k = y_k^\top p_k - y_k^\top p_k = 0. \qquad (1)$$

*Key observation:* (1) holds independently of the location of $x_k$. Applying an exact line search

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}} f(x + \alpha d),$$

to an arbitrary point $x$ in the search direction $d = \pm d_k$, the point $x_+ = x + \alpha^* d$ ends up lying in the affine hyper-plane

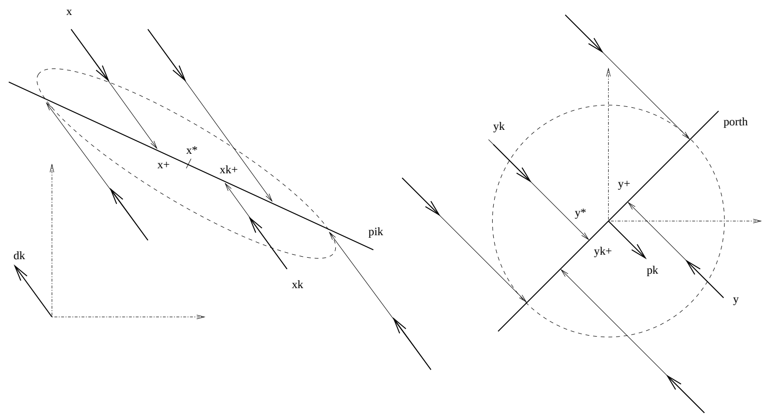
$$\pi_k := x^* + B^{-1/2} p_k^\perp.$$

The requirement that all subsequent line searches are to be conducted within $\pi_k$ amounts to the condition $p_j \perp p_k$ for all $j > k$, or equivalently expressed in $x$-coordinates,

$$d_k^\top B d_j = 0 \qquad \forall j \geq k+1. \qquad (2)$$

If this relation holds, we say that $d_k$ and $d_j$ are $B$-conjugate (which is the same as orthogonality with respect to the Euclidean inner product defined by $B$).

In subsequent searches, it therefore never makes sense to leave $\pi_k$ again!




**Observations:**

- $f|_{\pi_k}$ is a strictly convex quadratic function on $\pi_k$. Choosing $d_{k+1}$ satisfying (2), we can thus repeat our argument and find that $x_{k+2}$ will lie in an affine hyper-plane $\pi_{k+1}$ of $\pi_k$ to which any future line-search must be restricted.

- Arguing iteratively, the dimension of the search space $\pi_k$ is decreased by 1 in every iteration, thus termination occurs in $n$ iterations.

- Thus will have chosen mutually $B$-conjugate search directions

$$d_i^\top B d_j = 0 \qquad \forall\, i \neq j.$$

**Theorem 1.** Let $f(x) := x^\top Bx + b^\top x + a$, where $B \succ 0$. For $k = 0, \dots, n-1$ let $d_k$ be chosen such that

$$d_i^\top B d_j = 0 \qquad \forall\, i \neq j.$$

Let $x_0 \in \mathbb{R}^n$ be arbitrary and

$$x_{k+1} = x_k + \alpha_k d_k \qquad (k = 0, \dots, n-1),$$

where $\alpha_k = arg\min_{\alpha \in \mathbb{R}} f(x_k + \alpha d_k)$.

Then $x_n$ is the global minimiser of $f$.

*Proof:* Induction over $k$.

- For $k = 0$ there is nothing to prove.
- Assume that $d_i^\top B d_j = 0$ for all $i, j \in \{0, \dots, k-1\}$, $i \neq j$.
- For $i < k$,
$$d_i^\top B d_k = d_i^\top B v_k - \sum_{j=0}^{k-1} \frac{d_j^\top B v_k}{d_j^\top B d_j} d_i^\top B d_j = d_i^\top B v_k - d_i^\top B v_k = 0.$$
- The linear independence of the $v_j$ guarantees that none of the $d_j$ is zero, and hence $d_j^\top B d_j > 0$ for all $j$.

**How to choose $B$-conjugate search directions?**

**Lemma 1: Gram-Schmidt orthogonalisation.** Let $v_0, \dots, v_{n-1} \in \mathbb{R}^n$ be linearly independent vectors, and let $d_0, \dots, d_{n-1}$ be recursively defined as follows,

$$d_k = v_k - \sum_{j=0}^{k-1} \frac{d_j^\top B v_k}{d_j^\top B d_j} d_j. \tag{3}$$

Then $d_i^\top B d_k = 0$ for all $i \neq k$.

Unfortunately, this procedure would require that we hold the vectors $d_j$ $(j < k)$ in the computer memory. Thus, as $k$ approaches $n$ the method would require $O(n^2)$ memory.

A second key observation shows that we can get away with $O(n)$ storage if we choose the steepest descent direction as $v_k$:

**Lemma 2: Orthogonality.** Choose $d_0 = -\nabla f(x_0)$ and for $k = 1, \dots, n-1$ let $d_k$ be computed via

$$d_k = -\nabla f(x_k) - \sum_{j=0}^{k-1} \frac{d_j^\top B(-\nabla f(x_k))}{d_j^\top B d_j} d_j. \tag{4}$$

Then $\nabla f(x_j)^\top \nabla f(x_k) = 0$ and $d_j^\top \nabla f(x_k) = 0$ for $j < k$.

*Proof:* Note that $\nabla f(x_k) = 2Bx_k + b$ for all $k$.

By induction over $k$ we prove $d_j^\top \nabla f(x_k) = 0$ for all $j < k$.

- Okay for $k = 0$. Assume it holds for $k$. Then

$$\begin{aligned} d_j^\top \nabla f(x_{k+1}) &= d_j^\top \left(2B(x_k + \alpha_k d_k) + b\right) \\ &= d_j^\top \nabla f(x_k) + 2\alpha_k d_j^\top B d_k \\ &= 0, \qquad (j = 0, \ldots, k-1). \end{aligned}$$

- Furthermore, $d_k^\top \nabla f(x_{k+1}) = 0$ is the first order optimality condition for the line search $\min_\alpha f(x_k + \alpha d_k)$ defining $x_{k+1}$.

Next, (4) implies that for all $k$,

$$\mathrm{span}(d_0, \ldots, d_k) = \mathrm{span}(\nabla f(x_0), \ldots, \nabla f(x_k)).$$

For $j < k$ there exist therefore $\lambda_1, \ldots, \lambda_j$ such that $\nabla f(x_j) = \sum_{i=0}^{j} \lambda_i d_i$, and we have

$$\nabla f(x_j)^\top \nabla f(x_k) = \sum_{i=1}^{j} \lambda d_i^\top \nabla f(x_k) = 0. \qquad \square$$

**Putting the pieces together:** Recall (4),

$$d_k = -\nabla f(x_k) - \sum_{j=0}^{k-1} \frac{d_j^\top B(-\nabla f(x_k))}{d_j^\top B d_j} d_j.$$

Substituting $\nabla f(x_{j+1}) - \nabla f(x_j) = 2\alpha_j B d_j$ into (4),

$$d_k = -\nabla f(x_k) + \sum_{j=0}^{k-1} \frac{\nabla f(x_{j+1})^\top \nabla f(x_k) - \nabla f(x_j)^\top \nabla f(x_k)}{\nabla f(x_{j+1})^\top d_j - \nabla f(x_j)^\top d_j} d_j.$$

Lemma 2 implies that all but the last summand in the the right hand side expression are zero,

$$d_k = -\nabla f(x_k) - \frac{\nabla f(x_k)^\top \nabla f(x_k)}{\nabla f(x_{k-1})^\top d_{k-1}} d_{k-1}. \tag{5}$$

Multiplying (4) by $\nabla f(x_k)^\top$ and then replacing $k$ by $k-1$, Lemma 2 implies

$$d_{k-1}^\top \nabla f(x_{k-1}) = -\|\nabla f(x_{k-1})\|^2.$$

Substituting into (5),

$$d_k = -\nabla f(x_k) + \frac{\|\nabla f(x_k)\|^2}{\|\nabla f(x_{k-1})\|^2} d_{k-1}.$$

This is the *conjugate gradient* rule for updating the search direction.

- In the computation of $d_k$ we only need to keep two vectors and one number stored in the main memory: $d_{k-1}$, $x_k$, and $\|\nabla f(x_{k-1})\|^2$.

- The registers occupied by these data can be overwritten during the computation of the new data $d_k$, $x_{k+1}$, and $\|\nabla f(x_k)\|^2$.

- The method terminates in at most $n$ iterations.

- Furthermore, in general $x_k$ approximates $x^*$ closely after very few iterations, and the remaining iterations are used for fine-tuning the result.

**The Fletcher-Reeves Method:**

Algorithm 1 can be adapted for the minimisation of an arbitrary $C^1$ objective function $f$ and is then called *Fletcher-Reeves method*. The main differences are the following:

- Exact line-searches have to be replaced by practical line-searches.

- A termination criterion $\|\nabla f(x_k)\| < \epsilon$ has to be used to guarantee that the algorithm terminates in finite time.

- Since Lemma 2 only holds for quadratic functions, the conjugacy of $d_k$ is only be achieved approximately. To overcome this problem, reset $d_k$ to $-\nabla f(x_k)$ periodically.

**Algorithm 1: Conjugate Gradients.** $x_0 \in \mathbb{R}^n$, $d_0 := -\nabla f(x_0)$.

For $k = 0, 1, \ldots, n-1$ repeat

**S1** Compute $\alpha_k = \arg\min_\alpha f(x_k + \alpha d_k)$ and set $x_{k+1} = x_k + \alpha_k d_k$.

**S2** If $k < n-1$, compute

$$d_{k+1} = -\nabla f(x_{k+1}) + \frac{\|\nabla f(x_{k+1})\|^2}{\|\nabla f(x_k)\|^2} d_k.$$

Return $x^* = x_n$.

**Reading Assignment:** Lecture-Note 5.