Handout Probabilistic Graphical Models: Derivation of Conjugate Gradients Algorithm

Volkan Cevher, Matthias Seeger

Abstract

The idea behind the conjugate gradients algorithm, as shown in the lecture, is to maintain conjugate search directions, so that improvements realized in some iteration are not lost later on. This idea alone leads to the famous algorithm, but the derivation is too convolved to present in a lecture. I provide it here. I'd probably not have written this note, were there a readable, yet still concise derivation out there. I took the motivation of CG from [1], but filled in the gaps.

Recall the setting. $\boldsymbol{A}$ is symmetric positive definite. CG minimizes the quadratic

$$q(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{x}, \quad \boldsymbol{g}(\boldsymbol{x}) = \nabla q(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}, \quad \boldsymbol{x} \in \mathbb{R}^n.$$

We start with some $\boldsymbol{x}_0$. In iteration k, starting from $\boldsymbol{x}_{k-1}$, we pick a search direction $\boldsymbol{d}_k$, possibly using the gradient $\boldsymbol{g}_{k-1} = \boldsymbol{g}(\boldsymbol{x}_{k-1})$, then search along that line for the minimum point: $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k\boldsymbol{d}_k$. Be careful not to confuse search *directions* and *gradients*. They are the same only in the method of steepest descent, which CG is certainly different from (except for $\boldsymbol{A} = \boldsymbol{I}$). Annoyingly for students, CG is called *conjugate gradients*, it should be called *conjugate directions*.

Remember why steepest descent is not a good idea. Since

$$\frac{dq(\boldsymbol{x}_{k-1} + \alpha\boldsymbol{d}_k)}{d\alpha} = \boldsymbol{g}(\boldsymbol{x}_{k-1} + \alpha\boldsymbol{d}_k)^T\boldsymbol{d}_k = 0$$

at $\alpha = \alpha_k$ (line minimum), then $\boldsymbol{g}_k^T\boldsymbol{d}_k = 0$. Now, if $\boldsymbol{d}_k = -\boldsymbol{g}_{k-1}$, this means that subsequent search directions are orthogonal. And this is a really bad idea for minimizing $q(\boldsymbol{x})$ if $\boldsymbol{A}$ has eigenvalues of widely different sizes (remember the zig-zagging problem from the lecture). A much better idea is to select $\boldsymbol{d}_k$ in a way that renders *new* gradients, anywhere along the line searched, orthogonal to *old* directions:

$$0 = \boldsymbol{g}(\boldsymbol{x}_k + \alpha\boldsymbol{d}_{k+1})^T\boldsymbol{d}_k = \boldsymbol{g}_k^T\boldsymbol{d}_k + \alpha\boldsymbol{d}_{k+1}^T\boldsymbol{A}\boldsymbol{d}_k.$$

Since $\boldsymbol{g}_k^T\boldsymbol{d}_k = 0$ in any case (line minimization condition), we therefore need *conjugate* (rather than orthogonal) directions: $\boldsymbol{d}_j^T\boldsymbol{A}\boldsymbol{d}_k = 0$ for $j \neq k$.

If somebody gave us a conjugate set of directions, starting with $\boldsymbol{d}_1 = -\boldsymbol{g}_0$, we were done. However, at least to me, it is not at all obvious how to obtain such a basis in a feasible way. The conjugate gradients algorithm is a neat method for doing just that, with *a single matrix-vector multiplication* (MVM) *with* $\boldsymbol{A}$ *per iteration*, requiring $O(n)$ storage only. And all that is needed in order to derive it, is stated above. It is still a bit of magic, so please read on. By the way, CG is really a cornerstone of numerical mathematics. If its three-term recurrence is interpreted differently, the Lanczos algorithm is obtained, the most important method for computing eigenvectors of very large matrices.

The idea behind the derivation of CG is to establish a number of relationships between the gradients $\boldsymbol{g}_k$, the directions $\boldsymbol{d}_k$, and the step sizes α_k (for different k values), starting from $\boldsymbol{g}_k^T\boldsymbol{d}_k = 0$ and $\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k\boldsymbol{d}_k$ only, which imply that all directions are mutually conjugate, and all gradients are mutually orthogonal (yes, to get you completely confused!). There is a single *ansatz*, namely that $\boldsymbol{d}_{k+1} = -\boldsymbol{g}_k + \beta_k\boldsymbol{d}_k$, involving some further scalars β_k. A formally clean way is to use induction over the number of iterations k, but because it is convolved enough, I will rather proceed in the most digestable ordering. I'll also number equations excessively.

$$\boldsymbol{g}_k = \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b} \tag{1}$$
$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \alpha_k\boldsymbol{d}_k \tag{2}$$
$$\boldsymbol{g}_k = \boldsymbol{g}_{k-1} + \alpha_k\boldsymbol{A}\boldsymbol{d}_k \tag{3}$$
$$\boldsymbol{g}_k^T\boldsymbol{d}_k = 0 \tag{4}$$

Here, (3) is just (2) times $\boldsymbol{A}$, minus $\boldsymbol{b}$, and we have discussed (4) above. We have also seen why the following holds:

$$\boldsymbol{g}_{k-1}^T\boldsymbol{d}_j = 0,\ \boldsymbol{d}_k^T\boldsymbol{A}\boldsymbol{d}_j = 0 \quad \Rightarrow \quad \boldsymbol{g}_k^T\boldsymbol{d}_j = 0 \quad (j \le k) \tag{5}$$

In order to obtain a recurrence for the search directions, we make the ansatz

$$\boldsymbol{d}_{k+1} = -\boldsymbol{g}_k + \beta_k\boldsymbol{d}_k. \tag{6}$$

We would obtain steepest descent with $\beta_k = 0$, but in CG, we always have $\beta_k \neq 0$, unless we are done (we will see that below). Amazingly, this simple recurrence suffices to get all the rest. In order to determine what β_k should be, recall that we want conjugate directions.

$$0 = \boldsymbol{d}_{k+1}^T\boldsymbol{A}\boldsymbol{d}_k = (-\boldsymbol{g}_k + \beta_k\boldsymbol{d}_k)^T\boldsymbol{A}\boldsymbol{d}_k \quad \Rightarrow \quad \beta_k = \frac{\boldsymbol{g}_k^T\boldsymbol{A}\boldsymbol{d}_k}{\boldsymbol{d}_k^T\boldsymbol{A}\boldsymbol{d}_k}. \tag{7}$$

We will simplify this below. What about the gradients? If we assume the r.h.s. of (5) to hold, then for $j < k$:

$$\boldsymbol{g}_k^T\boldsymbol{g}_j \overset{(6)}{=} \boldsymbol{g}_k^T(\beta_j\boldsymbol{d}_j - \boldsymbol{d}_{j+1}) \overset{(5)}{=} 0.$$

Not relying on (5), we have

$$\boldsymbol{g}_k^T\boldsymbol{d}_j = \boldsymbol{g}_k^T\boldsymbol{d}_{j-1} = 0 \quad \Rightarrow \quad \boldsymbol{g}_k^T\boldsymbol{g}_j = 0. \tag{8}$$

OK, small break here. In steepest descent, search directions are orthogonal, and that is bad. In that method, search directions are (negative) gradients, so gradients are orthogonal. In CG, improving on steepest descent, directions are not orthogonal, but *conjugate*. However, gradients in CG are still *orthogonal*. And to really get everybody confused, the whole method is called conjugate gradients!

We will now close the loop, by showing that orthogonality of gradients implies conjugacy of directions. Assume that for some $j < k$: $\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_j = 0$. Then,

$$\boldsymbol{d}_{k+1}^T \boldsymbol{A} \boldsymbol{d}_j \overset{(6)}{=} (-\boldsymbol{g}_k + \beta_k \boldsymbol{d}_k)^T \boldsymbol{A} \boldsymbol{d}_j = -\boldsymbol{g}_k^T \boldsymbol{A} \boldsymbol{d}_j \overset{(3)}{=} -\boldsymbol{g}_k^T (\boldsymbol{g}_j - \boldsymbol{g}_{j-1})/\alpha_j.$$

But if gradients are orthogonal, the r.h.s. is zero. Therefore,

$$\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_j = 0,\ \boldsymbol{g}_k^T \boldsymbol{g}_j = \boldsymbol{g}_k^T \boldsymbol{g}_{j-1} = 0 \quad \Rightarrow \quad \boldsymbol{d}_{k+1}^T \boldsymbol{A} \boldsymbol{d}_j = 0. \tag{9}$$

The mutual orthogonality of *all* gradients and the conjugacy of *all* directions follows by running the cycle (5) $\Rightarrow$ (8) $\Rightarrow$ (9) $\Rightarrow$ (5) $\dots$ Make sure you understand that this cycle is kickstarted by (4), which holds by line minimization.

We are almost done. Right now, CG would not really be elegant. While we have analytic expressions for α_k and β_k (7), they are clumsy and require more than one MVM with $\boldsymbol{A}$ per iteration. By (4), (3): $0 = \boldsymbol{g}_k^T \boldsymbol{d}_k = (\boldsymbol{g}_{k-1} + \alpha \boldsymbol{A} \boldsymbol{d}_k)^T \boldsymbol{d}_k$, so that

$$\alpha_k = \frac{-\boldsymbol{g}_{k-1}^T \boldsymbol{d}_k}{\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_k} \overset{(6)}{=} \frac{-\boldsymbol{g}_{k-1}^T(-\boldsymbol{g}_{k-1} + \beta_{k-1} \boldsymbol{d}_{k-1})}{\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_k} \overset{(4)}{=} \frac{\|\boldsymbol{g}_{k-1}\|^2}{\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_k}. \tag{10}$$

And now, let's use (almost) everything above:

$$\beta_k \overset{(7)}{=} \frac{\boldsymbol{g}_k^T \boldsymbol{A} \boldsymbol{d}_k}{\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_k} \overset{(10)}{=} \frac{\boldsymbol{g}_k^T (\alpha_k \boldsymbol{A} \boldsymbol{d}_k)}{\|\boldsymbol{g}_{k-1}\|^2} \overset{(3)}{=} \frac{\boldsymbol{g}_k^T (\boldsymbol{g}_k - \boldsymbol{g}_{k-1})}{\|\boldsymbol{g}_{k-1}\|^2} \overset{(8)}{=} \frac{\|\boldsymbol{g}_k\|^2}{\|\boldsymbol{g}_{k-1}\|^2}. \tag{11}$$

From this equation, we see that if $\beta_k = 0$, then $\boldsymbol{g}_k = \boldsymbol{0}$, and we have reached the global minimum point of $q(\boldsymbol{x})$.

That's it. The CG algorithm itself is given as a lecture slide. Finally, note that (5) and the conjugacy of the $\boldsymbol{d}_k$ imply that with at most n iterations, we are done. Namely, any set of conjugate directions is also linearly independent (easy exercise). But then, $\boldsymbol{g}_n$ is orthogonal to all $\boldsymbol{d}_k$, $k \le n$, which is possible only if $\boldsymbol{g}_n = \boldsymbol{0}$. For special matrices $\boldsymbol{A}$, whose characteristic polynomial has rank $< n$, this can happen earlier (a very simple example is $\boldsymbol{A} = \boldsymbol{I}$). In practice, you will probably never see this happening. Also note that (5) (new gradients orthogonal to old directions) directly implies the Krylov subspace minimization characteristic of CG that was discussed in the lecture.

References

[1] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1st edition, 1995.