CONJUGATE GRADIENT FOR OPTIMIZATION

David F. Gleich

April 25, 2023

The conjugate gradient method is a relative of the gradient-descent method for line search. Like gradient descent, it'll have linear convergence. There are tons of derivations of the CG method from a variety of perspectives. We cover these in CS515 because they are relevant. For optimization, these are less relevant.

The starting point to derive the method is to consider a strongly convex quadratic:

$$\text{minimize} \quad \tfrac{1}{2}\mathbf{x}^T \boldsymbol{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

with solution $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ and $\boldsymbol{A} = \boldsymbol{A}^T$, $\boldsymbol{A} \succ 0$.

The gradient of this function is $\mathbf{g} = \boldsymbol{A}\mathbf{x} - \mathbf{b}$, which is what's called the residual of the linear system.

The CG method can be derived by looking for search directions which are *conjugate* to the previous search direction. Through an emergent property of the algorithm (based on conjugacy), this will result in an expanding search space that always visits a new direction. (That is, it won't oscillate between two directions...)

Since in an n dimensional space there are only n possible different directions, after n steps, for a linear system of equations, the method will complete.

Our point here isn't to look at the derivation of CG, but rather to look at the mechanics of the algorithm for a linear system. After much simplification the algorithm is:

Input: $\boldsymbol{A}, \mathbf{b}$

Let
- $\mathbf{x}_0 = 0$
- $\mathbf{r}_0 = \boldsymbol{A}\mathbf{x}_0 - \mathbf{b} = -\mathbf{b}$
- $\mathbf{p}_0 = -\mathbf{r}_0$ (Neg. gradient!)

While $\|\mathbf{r}_k\| \ge \tau$
- Let α_k be optimal line search **in** direction $\mathbf{p}_k$
  - ($\alpha_k = \mathbf{r}_k^T\mathbf{r}_k / \mathbf{p}_k^T \boldsymbol{A}\mathbf{p}_k$.)
- $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$ (Update step)
- $\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \boldsymbol{A}\mathbf{p}_k$
- $\beta_{k+1} = \mathbf{r}_{k+1}^T\mathbf{r}_k / \mathbf{r}_k^T\mathbf{r}_k$
- $\mathbf{p}_k = -\mathbf{r}_k + \beta_{k+1}\mathbf{p}_k$
- $k = k + 1$

To recap: this does a line search (optimally) and then updates the search direction $\mathbf{p}_k$ based on the gradient $\mathbf{r}_k$. The idea with using it for optimization is that the same algorithm will work where we replace the optimal line search with a line search algorithm (like strong Wolfe) and the residual with the gradient.

The Fletcher-Reeves CG method uses:

$$\mathbf{r}_k \to \mathbf{g}(\mathbf{x}_k)$$
$$\alpha_k \to \text{Strong-Wolfe line search}$$

The Polak-Ribière CG method uses (also):

$$\beta_k = \mathbf{g}_{k+1}^T(\mathbf{g}_{k+1} - \mathbf{g}_k)/\|\mathbf{g}_k\|^2$$

CONVERGENCE

The convergence theory for these methods is simple. Every n iterations, we set $\beta_k = 0$, which gives a gradient descent iteration. Consequently, since we never revisit things based on line search, we are set!

DESCENT

We need a few details to always guarantee that $\mathbf{p}_k$ is a descent direction. Strong Wolfe is enough for Fletcher-Reeves. For Polak-Ribière, this isn't enough and the book discusses additional details. Along with why you might pick one or the other.