Conjugate Gradient

Suppose we want to solve the system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

for the vector $\mathbf{x}$, where the known $n \times n$ matrix $\mathbf{A}$ is symmetric (i.e., $\mathbf{A}^T = \mathbf{A}$), positive-definite (i.e. $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all non-zero vectors $\mathbf{x} \in \mathbb{R}^n$), and real, and $\mathbf{b}$ is known as well. We denote the unique solution of this system by $\mathbf{x}^*$.

Solution of the above system of equations is the same as $\operatorname{argmin}_{\mathbf{x}\in\mathbb{R}^n} \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x}$.

Definition 1 (**A**-conjugate directions)**.** *Let* $\mathbf{A}$ *$(n \times n)$ be a symmetric matrix. The vectors* $\{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ *(all* $\in \mathbb{R}^n$*) are called conjugate (or orthogonal) with respect to* $\mathbf{A}$ *if* $\mathbf{d}_i^T\mathbf{A}\mathbf{d}_j = 0$ *for all* $i \neq j$*.*

Lemma 1 (Linear independence)**.** *Let* $\mathbf{A}$ *$(n \times n)$ be positive definite. If the vectors* $\{\mathbf{d}_1, \ldots, \mathbf{d}_k\}$ *(all* $\in \mathbb{R}^n$*) are conjugate (orthogonal) with respect to* $\mathbf{A}$*, then they are linearly independent.*

Proof. Let $\{\mathbf{d}_1, \ldots, \mathbf{d}_k\}$ are linearly dependent and let $\mathbf{d}_k = \sum_{j=1}^{k-1} \alpha_j \mathbf{d}_j$.

By definition of conjugacy, we have $\mathbf{d}_k^T\mathbf{A}\mathbf{d}_j = 0$ for all $j \neq k$. Since, $\mathbf{A}$ is pd, we have $\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k > 0$. On the other hand, $\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k = \sum_{j=1}^{k-1} \alpha_j \mathbf{d}_k^T\mathbf{A}\mathbf{d}_j = 0$ (Contradiction!) □

If $\mathbf{x}^*$ is a solution of the above system of equation, we can represent it as a linear combination of $\{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$, which are conjugate vectors with respect to $\mathbf{A}$.

Let $\mathbf{x}^* = \sum_{j=1}^{n} \alpha_j \mathbf{d}_j$. It is easy to check that $\mathbf{d}_j^T\mathbf{A}\mathbf{x}^* = \alpha_j \mathbf{d}_j^T\mathbf{A}\mathbf{d}_j$. Thus $\alpha_j = \frac{\mathbf{d}_j^T\mathbf{A}\mathbf{x}^*}{\mathbf{d}_j^T\mathbf{A}\mathbf{d}_j}$. But by assumption, $\mathbf{A}\mathbf{x}^* = \mathbf{b}$. Thus $\alpha_j = \frac{\mathbf{d}_j^T\mathbf{b}}{\mathbf{d}_j^T\mathbf{A}\mathbf{d}_j}$. Hence, to know α_j, we do not need to know $\mathbf{x}^*$.

Hence $\mathbf{x}^* = \sum_{j=1}^{n} \frac{\mathbf{d}_j^T\mathbf{b}}{\mathbf{d}_j^T\mathbf{A}\mathbf{d}_j}\mathbf{d}_j$, no matrix inversion or nothing.

In practice, we do not know the conjugate vectors. We thus may follow the Gram–Schmidt type algorithm. We identify the conjugate directions iteratively. Thus, we 'try' to set up the algorithm in a way that $|\alpha_1| \geq |\alpha_2| \geq |\alpha_3| \cdots$ so that we truncate the sum or stop the iteration early and get a reasonable solution. We may not however be able to ensure this in a strict sense, but approximately.

Algorithm 1: Conjugate gradient

(i) Start setting $\mathbf{r}_1 = \mathbf{b} - \mathbf{A}\mathbf{x}_1$ with some starting value $\mathbf{x}_1$.
(ii) If $\mathbf{r}_0$ is small enough, return $\mathbf{x} = \mathbf{x}_1$ as solution.
(iii) Otherwise, set $\mathbf{d}_1 = \mathbf{r}_1$ and $k = 1$.
(iv) Start loop

- $\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k}$
- $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
- $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{d}_k$
- If $\mathbf{r}_{k+1}$ is small enough, return $\mathbf{x} = \mathbf{x}_{k+1}$ as solution.
- Otherwise, $\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$, $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$, move $k = k + 1$ and Repeat

In the above algorithm, how the term 'gradient' fits in? Say $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x} - \mathbf{b}^T \mathbf{x}$ and we are trying to solve $\min_{\mathbf{x}} f(\mathbf{x})$. Following the steps in gradient descent, the updates should look like $\mathbf{x}_{k+1} = \mathbf{x}_k - t_k \bigtriangledown f(\mathbf{x}_k)$. For our function, we have $\bigtriangledown f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$. Thus, we need to move towards $\mathbf{b} - \mathbf{A}\mathbf{x}$. The other vectors in the basis will be made conjugate to this gradient vector. Thus, the term 'conjugate gradient' comes in.

To see above, note that $\beta_k = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k} = -\frac{\mathbf{r}_{k+1} \mathbf{A}\mathbf{d}_k}{\mathbf{d}_k^T \mathbf{A}\mathbf{d}_k}$. Then, $\mathbf{d}_{k+1}^T \mathbf{A}\mathbf{d}_k = 0$

If the condition number of $\mathbf{A}$ is large, the convergence is slow as it leads to slower improvement. Hence, to ensure faster convergence in this case, **Preconditioning conjugate gradient** is used. [Condition number of a matrix $\mathbf{A}$ is $\kappa(\mathbf{A}) = \frac{\sigma_{\max}(\mathbf{A})}{\sigma_{\min}(\mathbf{A})}$, where $\sigma_{\max}(\mathbf{A})$ and $\sigma_{\min}(\mathbf{A})$ are maximal and minimal singular values of $\mathbf{A}$ respectively. In our case, the singular values and eigenvalues are the same as $\mathbf{A}$ is symmetric]

Do check whether $|\alpha_1| \geq |\alpha_2| \geq |\alpha_3| \cdots$ is satisfied or not in the example codes for clarity. Also, you can see how the condition number of $\mathbf{A}$ affects this.

Algorithm 2: Pre-conditioned conjugate gradient

$$\begin{aligned}\mathbf{r}_0 &= \mathbf{b} - \mathbf{A}\mathbf{x}_0 \\ \mathbf{z}_0 &= \mathbf{M}^{-1}\mathbf{r}_0 \\ \mathbf{p}_0 &= \mathbf{z}_0 \\ k &= 0\end{aligned}$$

repeat

$$\begin{aligned}\alpha_k &= \frac{\mathbf{r}_k^\mathsf{T}\mathbf{z}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k} \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k\mathbf{p}_k \\ \mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k\end{aligned}$$

if r_{k+1} is sufficiently small, then exit loop end if

$$\begin{aligned}\mathbf{z}_{k+1} &= \mathbf{M}^{-1}\mathbf{r}_{k+1} \\ \beta_k &= \frac{\mathbf{r}_{k+1}^T\mathbf{z}_{k+1}}{\mathbf{r}_k^T\mathbf{z}_k} \\ \mathbf{p}_{k+1} &= \mathbf{z}_{k+1} + \beta_k\mathbf{p}_k \\ k &= k+1,\end{aligned}$$

end repeat;
The result is x_{k+1}

In the above algorithm $\mathbf{M}$ is fixed. We often set $\mathbf{M}$ as $\mathbf{L}\mathbf{L}^T$ where $\mathbf{L}$ is an incomplete Cholesky decomposition of $\mathbf{A}$ or sometimes the diagonal entries in $\mathbf{A}$ (Jacobi preconditioner). Or, we often set $\mathbf{M}^{-1}$= some approximate inverse of $\mathbf{A}$. The approximate inverse computation would depend on the shape and structure of $\mathbf{A}$. For clarity, we are now applying the basic CG steps for solving 1) $\mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^{-1})^T\mathbf{y} = \mathbf{L}^{-1}\mathbf{b}$ and subsequently get the solution as $\mathbf{x} = \mathbf{L}^T\mathbf{y}$ such that the condition number of $\mathbf{L}^{-1}\mathbf{A}(\mathbf{L}^{-1})^T$ is small.