The Conjugate gradient algorithm is Summarized below:

1. Set $k := 0$; select the initial point $x^{(0)}$.
2. $g^{(0)} = \nabla f(x^{(0)})$. If $g^{(0)} = 0$, stop; else $d^{(0)} = -g^{(0)}$.
3. $$\alpha_k = \frac{-g^{(k)T} d^{(k)}}{d^{(k)T} Q d^{(k)}}$$
4. $x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$.
5. $g^{(k+1)} = \nabla f(x^{(k+1)})$. If $g^{(k+1)} = 0$ stop.
6. $$\beta_k = \frac{g^{(k+1)} Q d^{(k)}}{d^{(k)T} Q d^{(k)}}$$
7. $d^{(k+1)} = -g^{(k+1)} + \beta_k d^{(k)}$.
8. Set $k := k+1$ go to step 3.

→ Another explanation of Conjugate Gradient

Gradient descent can perform poorly in narrow vallys. The conjugate gradient method overcomes this issue by borrowing inspiration from methods for optimizing quadratic functions:

$$\min_x f(x) = \frac{1}{2} x^T A x + b^T x + c.$$

where (A) is symmetric and positive definite, & thus f has a unique local minimum.

- The conjugate gradient method can optimize n-dimensional quadratic functions in n-steps. Its directions are mutually conjugate with w.r.t A.

$$d^{(i)T} A d^{(j)} = 0 \quad \text{for } i \neq j.$$

The mutually conjugate vectors are the basis vectors of A. They are generally not orthogonal to one another.

The successive conjugate directions are computed using gradient information and the previous descent direction. The algorithm starts with the direction of the steepest descent:

$$d^{(1)} = -g^{(1)}$$

We then use line search to find the next design point. For quadratic functions, the step factor α can be computed exactly. The update is then:

$$x^{(2)} = x^{(1)} + \alpha^{(1)} d^{(1)}$$

- Suppose we want to derive the optimal step factor for a line search on a quadratic function:

$$\text{Min} \quad f(x+\alpha d).$$

- We can compute the derivative with respect to α.

$$\frac{\partial f(x+\alpha d)}{\partial \alpha} = \frac{\partial}{\partial \alpha}\left[\frac{1}{2}(x+\alpha d)^T A (x+\alpha d) + b^T(x+\alpha d) + c\right]$$

$$= d^T A(x+\alpha d) + d^T b$$

$$= d^T (Ax+b) + \alpha d^T A d$$

Setting:

$$\frac{\partial f(x+\alpha d)}{\partial \alpha} = 0 \quad \text{results in:}$$

$$\boxed{\alpha = -\frac{d^T(Ax+b)}{d^T A d}}$$

Subsequent iterations choose $d^{(k+1)}$ based on the next gradient and a contribution from the current descent direction:

$$d^{(k+1)} = -g^{(k+1)} + \beta^{(k)} d^{(k)}$$

for scalar parameter β. Larger values of β indicate that the previous descent directions contributes more strongly.

- We can derive the best value for β for a known A, using the fact that $d^{(k+1)}$ is conjugate to $d^{(k)}$:

$$d^{(k+1)} A \, d^{(k)} = 0$$

$$\Rightarrow \left(-g^{(k+1)} + \beta^{(k)} d^{(k)}\right) A \, d^{(k)} = 0$$

$$\Rightarrow -g^{(k+1)} A \, d^{(k)} + \beta^{(k)} d^{(k)T} A \, d^{(k)} = 0$$

$$\Rightarrow \boxed{\beta^{(k)} = \frac{g^{(k+1)} A \, d^{(k)}}{d^{(k)T} A \, d^{(k)}}}$$

The conjugate gradient method can be applied to non-quadratic functions as well. Smooth, continuous functions behave like quadratic functions close to a local minimum, and the conjugate method will converge very quickly in such regions.

→ Unfortunately, we do not know the value of A that best approximates f around $x^{(k)}$. Instead several choices of $\beta^{(k)}$ tend to work well:

Fletcher-Reeves : $\beta^{(k)} = \dfrac{g^{(k)T} g^{(k)}}{g^{(k-1)T} g^{(k-1)}}$

Polak-Ribiere $\beta^{(k)} = \dfrac{g^{(k)T}(g^{(k)} - g^{(k-1)})}{g^{(k-1)T} g^{(k-1)}}$

Convergence for the Polak Ribiere method can be guaranteed if we modify it to allow for automatic Resets:

$\beta \leftarrow \max(\beta, 0)$