Conjugate Gradient Methodology

The previous two unconstrained optimization algorithms, Newton's method and steepest descent, only utilize local characteristics about the point under test x_k. Conjugate gradient descent uses the previous search direction in its pursuit to find the new direction (i.e. it has memory!). The differences between the 'Big Three' derivative based techniques are summarized below.

Newton's Method

- **Idea:** Uses both the slope (gradient) and curvature (Hessian) of the function to “jump” directly toward a stationary point.
- **Analogy:** Like standing on a hill and using not just the steepness but also the bend of the ground to guess where the valley is.
- **Strength:** Can converge very quickly near the optimum.
- **Limitation:** Needs the Hessian (can be expensive to compute) and only uses *local curvature information*, which may mislead if the surface is complicated.

Steepest (Gradient) Descent

- **Idea:** Always steps in the direction of the steepest downward slope (negative gradient).
- **Analogy:** Like a blindfolded hiker always walking straight downhill from where they’re standing.
- **Strength:** Simple, only requires the gradient.
- **Limitation:** Can zig-zag and be slow, because “steepest downhill” locally is not always the fastest way to the bottom globally.

Conjugate Gradient Descent

- **Idea:** Improves on steepest descent by remembering the *previous search direction* and adjusting the next step to avoid undoing progress.
- **Analogy:** Like hiking downhill but keeping track of where you just came from, so you don’t keep walking back and forth across the same valley.
- **Strength:** Efficient for large problems, especially quadratic ones, and uses both local slope and history.
- **Limitation:** More bookkeeping than steepest descent, but avoids a lot of wasted zig-zagging.

Conjugate Gradient Step Derivation

Conjugate gradient method - Wikipedia

Derivation of the conjugate gradient method - Wikipedia

Conjugate Gradient Descent Algorithm

1. Start with the steepest descent: pick an initialization $\mathbf{x}_0$ and set the initial step to $\mathbf{s}_0 = -\nabla f(\mathbf{x}_0)$.
2. Compute $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{s}_k$ by choosing α which minimizes f in the direction $\mathbf{s}_k$.

3. Compute $f(\mathbf{x}_{k+1})$ and $\nabla f(\mathbf{x}_{k+1})$.
4. Find $\mathbf{s}_{k+1}$ using $\mathbf{s}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \mathbf{s}_k \dfrac{\nabla^T f(\mathbf{x}_{k+1}) \nabla f(\mathbf{x}_{k+1})}{\nabla^T f(\mathbf{x}_k) \nabla f(\mathbf{x}_k)}$
5. Test convergence via $||\nabla f(\mathbf{x}_k)| < \epsilon$.