The Method of Steepest Descent

This is the quadratic function from $\mathbb{R}^n$ to $\mathbb{R}$ that is constructed to have a minimum at the $\mathbf{x}$ that solves the system $A\,\mathbf{x} = \mathbf{b}$:

$$g(\mathbf{x}) = \langle \mathbf{x}, A\mathbf{x} \rangle - 2\langle \mathbf{x}, \mathbf{b} \rangle$$

In the method of steepest descent, we pick a starting point $\mathbf{x}_0$ that we think is close to the solution. We start our search for the solution by heading in the direction of steepest descent. This direction is the negative of the gradient of $g(\mathbf{x})$ at the starting point $\mathbf{x}_0$.

$$\mathbf{v} = -\nabla g(\mathbf{x}_0)$$

Because $g(\mathbf{x})$ has a particularly simple form, we can compute its gradient explicitly and notice that something interesting happens when we do:

$$\nabla g(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial g}{\partial x_1}(\mathbf{x}_0) \\ \frac{\partial g}{\partial x_2}(\mathbf{x}_0) \\ \vdots \\ \frac{\partial g}{\partial x_n}(\mathbf{x}_0) \end{bmatrix}$$

If we work out the details in the partial derivatives, we get a pleasing result.

$$\frac{\partial g}{\partial x_k}(\mathbf{x}) = 2 \sum_{i=1}^{n} a_{k,i}\, x_i - 2\, b_k$$

This leads to

$$\nabla g(\mathbf{x}_0) = \begin{bmatrix} \frac{\partial g}{\partial x_1}(\mathbf{x}_0) \\ \frac{\partial g}{\partial x_2}(\mathbf{x}_0) \\ \vdots \\ \frac{\partial g}{\partial x_n}(\mathbf{x}_0) \end{bmatrix} = 2\,(A\,\mathbf{x}_0 - \mathbf{b}) = -2\,\mathbf{r}_0$$

where $\mathbf{r}_0$ is the residual associated with $\mathbf{x}_0$.

Now that we have a search direction, we can construct a ray starting at $\mathbf{x}_0$ and heading in the direction of $\mathbf{v} = -2\,\mathbf{r}_0$. We want the point along that ray that minimizes the quadratic form $g(\mathbf{x})$.

$$h(t) = g(\mathbf{x} + t\,\mathbf{v})$$

This function has its minimum at the point where $h'(t) = 0$:

$$h'(t) = -2 \langle \mathbf{v}, \mathbf{b}-A\mathbf{x} \rangle + 2\, t \langle \mathbf{v}, A\mathbf{v} \rangle = 0$$

$$t = \frac{\langle \mathbf{v}, \mathbf{b}-A\mathbf{x} \rangle}{\langle \mathbf{v}, A\mathbf{v} \rangle} = \frac{\langle \mathbf{v}, \mathbf{r} \rangle}{\langle \mathbf{v}, A\mathbf{v} \rangle}$$

Using that value of t to compute $\mathbf{x} + t\ \mathbf{v}$ gives us our next point. We then compute a residual at that point to establish our search direction and repeat the process.

The big problem with the method of steepest descent is that it typically takes too many iterations of the method to converge to the solution of the system. In fact, it takes so many iterations that we end up doing more work than if we had just solved the system by Gauss elimination in the first place.

The method does have two big advantages. The first is that the method can be modified to converge more quickly, as we shall see below. The second is that a version of this method works for nonlinear systems. There is no equivalent to Gauss elimination for nonlinear systems of equations, so steepest descent is a primary method for dealing with nonlinear systems.

The Conjugate Direction and Conjugate Gradient Methods

The problem with the method of steepest descent is that the search directions it uses

$$\mathbf{v} = -2\ \mathbf{r}$$

lead to bad convergence behavior. We can fix this problem by picking better search directions. The *conjugate direction method* solves an n by n system by using a set of n search vectors $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$, …, $\mathbf{v}^{(n)}$ that have a special property. The vectors $\mathbf{v}^{(k)}$ are selected be *A orthogonal*. Vectors $\mathbf{v}^{(j)}$ and $\mathbf{v}^{(k)}$ for $j \neq k$ are A orthogonal if

$$\langle \mathbf{v}^{(j)}, A\ \mathbf{v}^{(k)} \rangle = 0$$

The problem with the conjugate direction method is that we typically won't have a set of n A orthogonal vectors just lying around. We need some scheme to generate them.

The *conjugate gradient method* uses a clever scheme to generate the A orthogonal set of vectors.

The process begins by picking a starting point for the search and using the steepest descent direction

$$\mathbf{v}^{(1)} = \mathbf{r}^{(0)}$$

as its first search direction. After conducting the first line search we land at the second point $\mathbf{x}^{(1)}$ and compute a residual $\mathbf{r}^{(1)}$ there. For subsequent iterations, we seek a new search direction that satisfies the relationship

$$\mathbf{v}^{(k)} = \mathbf{r}^{(k\text{-}1)} + s_{k\text{-}1}\ \mathbf{v}^{(k\text{-}1)}$$

The trick is to select $s_{k\text{-}1}$ so that $\mathbf{v}^{(k\text{-}1)}$ and $\mathbf{v}^{(k)}$ are A orthogonal:

$$\langle \mathbf{v}^{(k\text{-}1)}, A\left(\mathbf{r}^{(k\text{-}1)} + s_{k\text{-}1}\ \mathbf{v}^{(k\text{-}1)}\right) \rangle = 0$$

$$\langle \mathbf{v}^{(k-1)}, A\mathbf{r}^{(k-1)} \rangle + s_{k-1} \langle \mathbf{v}^{(k-1)}, A\ \mathbf{v}^{(k-1)} \rangle = 0$$

$$s_{k-1} = - \frac{\langle \mathbf{v}^{(k-1)}, A\ \mathbf{r}^{(k-1)} \rangle}{\langle \mathbf{v}^{(k-1)}, A\ \mathbf{v}^{(k-1)} \rangle}$$

After n iterations of this scheme, we will arrive at the solution of the original system.

Summary of the Method

We have now worked out all the details, but it might be useful to summarize and condense our findings.

$$g(\mathbf{x}) = \langle \mathbf{x}, A\mathbf{x} \rangle - 2\langle \mathbf{x}, \mathbf{b} \rangle$$

$$\mathbf{x}^{(0)} = \text{our starting guess}$$

$$\mathbf{r}^{(k)} = \mathbf{b} - A\ \mathbf{x}^{(k)}$$

$$\mathbf{v}^{(1)} = \mathbf{r}^{(0)}$$

$$t_k = \frac{\langle \mathbf{v}^{(k)}, \mathbf{r}^{(k-1)} \rangle}{\langle \mathbf{v}^{(k)}, A\mathbf{v}^{(k)} \rangle}$$

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + t_k\ \mathbf{v}^{(k)}$$

$$s_k = - \frac{\langle \mathbf{v}^{(k)}, A\ \mathbf{r}^{(k)} \rangle}{\langle \mathbf{v}^{(k)}, A\ \mathbf{v}^{(k)} \rangle}$$

$$\mathbf{v}^{(k)} = \mathbf{r}^{(k-1)} + s_{k-1}\ \mathbf{v}^{(k-1)}$$

Accelerating Convergence

The conjugate gradient method is effective, but we would also like to make it fast. One way to make it faster is to arrange for more rapid convergence of the sequence generated by the method.

It turns out that the main thing that affects the speed of convergence of the sequence is the matrix A and its eigenvalue, eigenvector structure. In an effort to make A behave more nicely, a common technique is to *precondition* the matrix A by doing

$$\tilde{A} = C^{-1}\ A\ \left(C^{-1}\right)^t$$

for some appropriately chosen C. We then use the conjugate gradient method to compute an approximate solution for the system

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

where

$$\tilde{\mathbf{b}} = C^{-1}\,\mathbf{b}$$

$$\tilde{\mathbf{x}} = C^{t}\,\mathbf{x}$$

We use the latter equation to solve for **x** at the end of the process:

$$\mathbf{x} = \left(C^{t}\right)^{-1}\tilde{\mathbf{x}}$$

What are some ways to select preconditioning matrices C? One method is to set

$$C_{i,j} = \begin{cases} \dfrac{1}{\sqrt{A_{i,i}}} & i = j \\ 0 & i \neq j \end{cases}$$

A second method is to do a Cholesky decomposition on A:

$$A = LL^{T}$$

$$C = L$$

In practice, the actual Cholesky decomposition is not used, because it is too expensive to compute. For A with a great many 0 entries, we can compute an "approximate" Cholesky decomposition by doing

1. Force L to have the same pattern of non-zero terms as A.
2. Use the Cholesky formulas to compute only those entries of L that we think should be nonzero.

Since the case of positive definite A with many 0 entries arises frequently in applications, this is a useful approach.