THE METHOD OF STEEPEST DESCENT

http://www.phys.uconn.edu/~rozman/Courses/m3511_19s/

Last modified: February 26, 2019

Abstract

The *Steepest Descent* is an iterative method for solving sparse systems of linear equations. The presentation of the method follows Sec. 1–4 of the article "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain" by J. R. Shewchuk (1994).

1 Introduction

Steepest Descent (SD) is an elegant iterative method for solving large systems of linear equations. SD is effective for systems

$$A\mathbf{x} = \mathbf{b}, \tag{1}$$

where $\mathbf{x}$ is an unknown vector, $\mathbf{b}$ is a known vector, and A is a known, $n \times n$ square, symmetric, positive-definite matrix.

Written out fully, Eq. (1) is

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1n} \\ A_{21} & A_{22} & \dots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \dots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}. \tag{2}$$

The scalar product of two vectors is written $\mathbf{x}^t\mathbf{y}$, and represents the following sum:

$$\mathbf{x}^t\mathbf{y} \equiv \sum_{i=1}^{n} x_i y_i. \tag{3}$$

Note, that $\mathbf{x}^t\mathbf{y} = \mathbf{y}^t\mathbf{x}$. We say that the vectors $\mathbf{x}$ and $\mathbf{y}$ are *orthogonal* if $\mathbf{x}^t\mathbf{y} = 0$.

A matrix A is *positive-definite* if, for every nonzero vector $\mathbf{x}$

$$\mathbf{x}^t A\mathbf{x} > 0. \tag{4}$$

2 The quadratic form

A quadratic form is a scalar quadratic function of a vector.

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^t A\mathbf{x} - \mathbf{b}^t\mathbf{x} + c, \tag{5}$$

where A is a matrix, $\mathbf{x}$ and $\mathbf{b}$ are vectors, and c is a scalar constant. We'll show shortly that if A is symmetric and positive-definite, $f(\mathbf{x})$ is minimized by the solution to $A\mathbf{x} = \mathbf{b}$.

The gradient of a quadratic form is defined to be

$$f'(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(\mathbf{x}) \\ \frac{\partial}{\partial x_2} f(\mathbf{x}) \\ \vdots \\ \frac{\partial}{\partial x_n} f(\mathbf{x}) \end{bmatrix}. \tag{6}$$

The gradient is a vector that, for a given point $\mathbf{x}$, points in the direction of greatest increase of $f(\mathbf{x})$. At the bottom of the paraboloid bowl, the gradient is zero. One can minimize $f(\mathbf{x})$ by setting $f'(\mathbf{x})$ equal to zero.

$$f'(\mathbf{x}) = A\mathbf{x} - \mathbf{b}. \tag{7}$$

3 The method of steepest descent

In the method of Steepest Descent, we start at an arbitrary point $\mathbf{x}^{(0)}$ and slide down to the bottom of the paraboloid. We take a series of steps $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots,$ until we are satisfied that we are close enough to the solution $\mathbf{x}$.

When we take a step, we choose the direction in which f decreases most quickly, which is the direction opposite $f'\left(\mathbf{x}^{(i)}\right)$.

According to Eq. (7), this direction is $-f'\left(\mathbf{x}^{(i)}\right) = \mathbf{b} - A\,\mathbf{x}^{(i)}$.

Let's introduce the following definitions.

The error

$$\mathbf{e}^{(i)} = \mathbf{x}^{(i)} - \mathbf{x} \tag{8}$$

is a vector that indicates how far we are from the solution.

The residual

$$\mathbf{r}^{(i)} = \mathbf{b} - A\mathbf{x}^{(i)} \tag{9}$$

indicates how far we are from the correct value of **b**.

It is easy to see that

$$\mathbf{r}^{(i)} = -A\mathbf{e}^{(i)}, \tag{10}$$

and you should think of the residual as being the error transformed by A into the same space as **b**. More importantly,

$$\mathbf{r}^{(i)} = -f'\left(\mathbf{x}^{(i)}\right) \tag{11}$$

and you should also think of the residual as the direction of steepest descent. Whenever you read "residual", think "direction of steepest descent".

Suppose we start at $x^{(0)}$. Our first step, along the direction of steepest descent. In other words, we will choose a point

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha\mathbf{r}^{(0)}. \tag{12}$$

The question is, how big a step should we take?

A line search is a procedure that chooses α to minimize f along a line $\mathbf{x}^{(0)} + \alpha\mathbf{r}^{(0)}$. α minimizes f when the derivative $\frac{\mathrm{d}}{\mathrm{d}\alpha} f\left(\mathbf{x}^{(1)}\right)$ is equal to zero. By the chain rule,

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} f\left(\mathbf{x}^{(1)}\right) = \frac{\mathrm{d}f(\mathbf{x}^{(1)})}{\mathrm{d}\mathbf{x}^{(1)}}\,\frac{\mathrm{d}\mathbf{x}^{(1)}}{\mathrm{d}\alpha} = -f'\left(\mathbf{x}^{(1)}\right)\mathbf{r}^{(0)} \tag{13}$$

To determine α, note that $f'\left(\mathbf{x}^{(i)}\right) = -\mathbf{r}^{(i)}$, and we have:

$$\begin{aligned}
\left(\mathbf{r}^{(1)}\right)^t \mathbf{r}^{(0)} &= 0, & (14)\\
\left(\mathbf{b} - A\mathbf{x}^{(1)}\right)^t \mathbf{r}^{(0)} &= 0, & (15)\\
\left(\mathbf{b} - A\left(\mathbf{x}^{(0)} + \alpha\mathbf{r}^{(0)}\right)\right)^t \mathbf{r}^{(0)} &= 0, & (16)\\
\left(\mathbf{b} - A\mathbf{x}^{(0)}\right)^t \mathbf{r}^{(0)} - \alpha\left(A\mathbf{r}^{(0)}\right)^t \mathbf{r}^{(0)} &= 0, & (17)\\
\left(\mathbf{b} - A\mathbf{x}^{(0)}\right)^t \mathbf{r}^{(0)} &= \alpha\left(A\mathbf{r}^{(0)}\right)^t \mathbf{r}^{(0)}, & (18)\\
\left(\mathbf{r}^{(0)}\right)^t \mathbf{r}^{(0)} &= \alpha\left(\mathbf{r}^{(0)}\right)^t \left(A\mathbf{r}^{(0)}\right), & (19)
\end{aligned}$$

$$\alpha = \frac{\left(\mathbf{r}^{(0)}\right)^t \mathbf{r}^{(0)}}{\left(\mathbf{r}^{(0)}\right)^t A\mathbf{r}^{(0)}} \qquad (20)$$

Putting it all together, the method of Steepest Descent is:

$$\begin{aligned}
\mathbf{r}^{(i)} &= \mathbf{b} - A\mathbf{x}^{(i)} & (21)\\
\alpha_i &= \frac{\left(\mathbf{r}^{(i)}\right)^t \mathbf{r}^{(i)}}{\left(\mathbf{r}^{(i)}\right)^t A\,\mathbf{r}^{(i)}} & (22)\\
\mathbf{x}^{(i+1)} &= \mathbf{x}^{(i)} + \alpha_i \mathbf{r}^{(i)}. & (23)
\end{aligned}$$

The algorithm, as written above, requires two matrix-vector multiplications per iteration. The computational cost of Steepest Descent is dominated by matrix-vector products; fortunately, one can be eliminated. By premultiplying both sides of Eq. (23) by $-A$ and adding $\mathbf{b}$, we have

$$\mathbf{r}^{(i+1)} = \mathbf{r}^{(i)} - \alpha_i A\mathbf{r}^{(i)}. \qquad (24)$$

Although Eq. (21) is still needed to compute $\mathbf{r}^{(0)}$, Eq. (24) can be used for every iteration thereafter. The product $A\mathbf{r}$, which occurs in both Eqs. (22) and (24), need only be computed once. The disadvantage of using this recurrence is that the sequence defined by Eq. (22) is generated without any feedback from the value of $\mathbf{x}^{(i)}$, so that accumulation of floating point roundoff error may cause $\mathbf{x}^{(i)}$ to converge to some point near $\mathbf{x}$. This effect can be avoided by periodically using Eq. (21) to recompute the correct residua.

4 Algorithm implementation

When SD reaches the minimum point, the residual becomes zero, and if Eq. (22) is evaluated an iteration later, a division by zero will result. It seems, then, that one must stop immediately when the residual is zero. Usually, however, one wishes to stop before convergence is complete. Because the error term is not available, it is customary to stop when the norm of the residual falls below a specified value; often, this value is some small fraction of the initial residual.

Given the inputs A, $\mathbf{b}$, a starting value $\mathbf{x}$, a maximum number of iterations `maxiter`, and an error tolerance `tol`, matlab code for SD is shown in Listing 1.

```
function [x, conv] = mysteepest(A, b, x, tol, maxiter)
% MYSTEEPEST - solve A*x = b using steepest descent algorithm
%              returns the solution and the convergence information
    iter = 1;
    r = b - A*x;
    delta = r'*r;
    conv = delta;
    delta0 = delta;
    while (delta > tol*delta0) && (iter < maxiter)
        q = A*r;
        alpha = delta/(q'*r);
        x = x + alpha*r;
        if mod(iter,50) == 0
          r = b - A*x;       % once in a while recalculate r
        else
          r = r - alpha*q;
        end
        delta = r'*r;
        conv = [conv, delta];
        iter = iter + 1;
    end
end
```

Listing 1: MATLAB implementation of Steepest Descent algorithm