Iterative Algorithms

Zvi Rosen
Department of Mathematics

October 23, 2016

Idea

We want to solve $Ax = b$ for large matrices A. Naive Gaussian Elimination takes $\frac{2}{3}n^3 + O(n^2)$ floating-point operations to solve the system.

What happens if we trade precision for iterative approximation?

Description of the Algorithm

The Gauss-Seidel algorithm is a type of fixed-point iteration. Using the equation $Ax = b$, we take $A = L_* + U$ where L_* is lower-triangular with nonzero diagonal, and U is strictly upper-triangular with zero diagonal. Then,

$$L_* x + Ux = b \Rightarrow L_* x = b - Ux.$$

Given the value of the i-th iteration $\mathbf{x}^{(\mathbf{i})}$, we solve for $x_1^{(i+1)}$ using $L_* x_1^{(i+1)} = b - Ux^{(i)}$ and then substitute forward to solve for $x_k^{(i+1)}$ in terms of $x_1^{(i+1)}, \dots, x_{k-1}^{(i+1)}$ and $x_{k+1}^{(i)}, \dots, x_n^{(i)}$.

Description of the Algorithm

The Gauss-Seidel algorithm is a type of fixed-point iteration. Using the equation $Ax = b$, we take $A = L_* + U$ where L_* is lower-triangular with nonzero diagonal, and U is strictly upper-triangular with zero diagonal. Then,

$$L_* x + Ux = b \Rightarrow L_* x = b - Ux.$$

Given the value of the i-th iteration $\mathbf{x^{(i)}}$, we solve for $x_1^{(i+1)}$ using $L_* x_1^{(i+1)} = b - Ux^{(i)}$ and then substitute forward to solve for $x_k^{(i+1)}$ in terms of $x_1^{(i+1)}, \ldots, x_{k-1}^{(i+1)}$ and $x_{k+1}^{(i)}, \ldots, x_n^{(i)}$.

Algorithm - part 2

In particular, we can use the following formula to compute $x_k^{(i+1)}$ using the following formula:

$$x_k^{(i+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{k-1} a_{kj} x_j^{(i+1)} - \sum_{j=k+1}^{n} a_{kj} x_j^{(i)} \right).$$

When does this converge?

The Gauss-Seidel algorithm converges in two cases:

1. Diagonally dominant matrices, whose diagonal elements $|a_{ii}|$ are larger than the sum of the rest of the row $\sum_{j \neq i} |a_{ij}|$.
2. Symmetric positive-definite matrices.

It may also converge in other cases, but convergence is not guaranteed.

Gauss-Seidel Example

The following matrix is a diagonally dominant 4×4 matrix.

$$\begin{bmatrix} 4.0855 & 0.9289 & 0.2373 & 0.5211 \\ 0.2625 & 4.7303 & 0.4588 & 0.2316 \\ 0.8010 & 0.4886 & 4.9631 & 0.4889 \\ 0.0292 & 0.5785 & 0.5468 & 4.6241 \end{bmatrix}$$

The first three iterations of Gauss-Seidel yield:

$$(0.1662, 0.0744, 0.0399, 0.1986)$$

$$\rightarrow (0.1217, 0.0633, 0.0286, 0.2016)$$

$$\rightarrow (0.1245, 0.0641, 0.0278, 0.2016)$$

Actual performance

Gauss-Seidel in practice tends to be pretty slow.

For *very* large matrices, it's a useful way of sharpening a good initial guess.

Relaxations

At each stage of the algorithm, we get a new value for each x_k using the Gauss-Seidel method.

Sometimes, it is advantageous to use a weighted average of the previous value and the new value. In particular, we can take:

$$x_k^{(i+1)} = \lambda x_k^{(i+1)} + (1 - \lambda) x_k^{(i)}.$$

The parameter λ is allowed to take values in the interval $(0, 2)$. If $\lambda < 1$ our algorithm is called underrelaxed, and if $\lambda > 1$, it is overrelaxed.