STA 141C - Big Data & High Performance Statistical Computing **Spring 2022**

Week 6-1: Iterative method for solving linear equations

Lecturer: Bo Y.-C. Ning *May 03, 2022*

Announcement

- Next week's lab, the TA will teach conducting simulation studies

Last time

- Parallel computing

Today

- Google PageRank problem
- Iterative method for solving linear equations

1 Introduction: Google PageRank problem

So far, we have studied several direct methods for solving linear equations: LU, Cholesky, QR, sweep operator, and soon we will learn the singular value decomposition. Those methods are good for dense, small or moderate sized matrices, and most of the time, they are unstructured.

When dealing with large and sparse linear systems, iterative methods are more efficient.

Consider the PageRank problem. The goal is to measure the importance of website pages. A famous example is Google Search, which needs to rank web pages in their search engine results. The name *PageRank* is named after both the term "web page" and co-founder Larry Page, who is one of the co-founders of Google.

The problem is described as follows, let $A \in \{0,1\}^{n\times n}$ be the connectivity matrix with entries

$$a_{ij} = \begin{cases} 1 & \text{if page } i \text{ links to page } j \\ 0 & \text{otherwise} \end{cases}$$

The size of A, n can be very large, e.g., $\approx 10^9$. A is known as the *Google matrix*.

Define $r_i = \sum_j a_{ij}$ the out-degree of page i and $s_i = \sum_i a_{ij}$ the in-degree of page i, image a random surfer wandering on internet according to following rules:

- From a page i with $r_i > 0$:
  - with probability p, she randomly chooses a link on page i uniformly and follows that link to the next page
  - with probability $1-p$, she randomly chooses one page from the set of all n uniformly and proceeds to that page
- From a page i with $r_i = 0$ (known as a dangling page), she randomly chooses one page from the set of all n pages uniformly and proceeds to that page

This process defines a Markov chain on the space of n pages.

Definition: A Markov chain or Markov process is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.

The probability p is called the *teleportation parameter* and is usually set at 0.85. The transition probability from page i to page j is given as

$$p_{ij} = \begin{cases} pa_{ij}/r_i + (1-p)/n & \text{if } r_i > 0 \\ 1/n & \text{if } r_i = 0 \end{cases}$$

Using the matrix-vector notation, we write

$$P = \text{diag}(r^+)A + z\mathbb{1}_n',$$

where $r^+ \in \mathbb{R}^n$ has entries

$$r_i^+ = \begin{cases} p/r_i & \text{if } r_i > 0 \\ 0 & \text{if } r_i = 0. \end{cases}$$

and $z \in \mathbb{R}^n$ has entries

$$z_i = \begin{cases} (1-p)/n & \text{if } r_i > 0 \\ 1/n & \text{if } r_i = 0. \end{cases}$$

Since the Markov chain is irreducible (meaning that we can reach page j from page i for any i and j), there exists a unique stationary distribution $x \in \mathbb{R}^n$, which is the left eigenvector of the transition matrix corresponding to eigenvalue 1. That is $P'x = x$, equivalently, x is a solution to the linear system

$$(I_n - P')x = 0_n.$$

Note that:

- The column sums of the matrix $I_n - P'$ are all 0s (why?). This implying this matrix is singular. We thus replace the first equation with the constraint $\mathbb{1}_n'x = 1$ when calling the `lu` function (do not need to do this for qr and iterative method).
- Note that P has a "sparse + low rank" structure, one may want to take advantage of it in computing. In fact, A is not only sparse, all of its nonzero entries are 1.

So far, we showed that the problem can be considered as a solving linear system problem. Consider in real world, $n = 10^9$. How long does it take to solve x? Using GE and LU, on a tera-flop supercomputer, it takes $2 \times (10^9)^3/3/10^{12} \approx 6.66 \times 10^{14}$ seconds, which is about 10^7 years!

2 Iterative methods

We introduce three iterative methods: Jacobi method, Gauss-Seidel method, and Successive over-relaxation (SOR) method.

2.1 Jacobi method

Let $Ax = b$, where A be a square matrix, decompose A into a diagonal matrix D, a lower triangular matrix L and an upper triangular matrix U, such that

$$A = D + L + U.$$

The Jacobi method is then iterating

$$x^{(k+1)} = D^{-1}b - D^{-1}(L + U)x^{(k)}.$$

(Do we need to solve D to obtain D^{-1}?) One can check that this is the same as solving $x^{(k+1)}$ from

$$Dx^{(k+1)} = -(L + U)x^{(k)} + b.$$

It turns out for each x_i, it can be solved by iterating

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}}.$$

The total flops for Jacobi is about $O(n^2)$ for unstructured A for each iteration, it is faster than GE/LU if the algorithm converges in less than n iteration. For sparse matrix A, the saving is huge.

A few things to consider: How to choose the starting point for Jacobi? (Random guess; start at $1/n$ for all x_i; solution for small matrix?) When to stop the algorithm?

2.2 Gauss-Seidel method

The Gauss-Seidel method solve $x^{(k+1)}$ from

$$(D + L)x^{(k+1)} = -Ux^{(k)} + b,$$

this implies that $x^{(k+1)} = -(D + L)^{-1}Ux^{(t)} + (D + L)^{-1}b$. Thus, for each $x_i^{(k+1)}$, its solution is

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}}.$$

The Gauss-Seidel method converges for any starting point if A is symmetric and positive definite. The default convergence rate for this method is chosen to be the spectral radius of the matrix $(D + L)^{-1}U$. For Jacobi method, the default convergence rate is chosen to be the spectral radius of the matrix $D^{-1}(L + U)$. Note that comparing Jacobi with Gauss-Seidel, Jacobi is particularly attractive for parallel computing.

2.3 Successive over-relaxation (SOR)

When the Gauss-Seidel method has a slow convergence rate, a way to improve the convergence rate is the SOR method, which solve $x_i^{(k+1)}$ by solving

$$x_i^{(k+1)} = w\frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}} + (1-w)x_i^{(k)},$$

where one need to pick a value for $w \in [0, 1]$ beforehand.

2.4 Conjugate gradient method

One can solve $Ax = b$ by minimizing the quadratic function $\frac{1}{2}x'A'x - b'x$. We will study this method later when we introducing the gradient descent method. For now, note that conjugate gradient and its variants are the top-notch iterative methods for solving huge, structured linear systems.

Table 1. Kershaw's results for a fusion problem.

Method	Number of iterations
Gauss Seidel	208,000
Block successive overrelaxation methods	765
Incomplete Cholesky conjugate gradients	25