CS 591 O1: Iterative Methods for Graph Algorithms and Network Analysis Spring 2015

Lecture 8: Power Method for Linear Systems, Condition Number, Gradient Descent

Instructor: Lorenzo Orecchia Scribe: Qiaobin Fu & Baichuan Zhou

1 Review of Lecture 7

Recall from the last lecture some things we discussed:

a. The Second Eigenvector of Normalized Laplacian: $Lx = \lambda_2 Dx$

b. Voltages/Linear Systems: $x = L^+ b$, where $b \perp \overrightarrow{1} = 0$. Using Gaussian Elimination to get the exact solution x takes $O(n^3)$.

2 Introduction to Iterative Methods

We think of an iterative algorithms as producing a sequence of candidate solutions

$$x^{(0)}, x^{(1)}, \ldots, x^{(t)}, \ldots \xrightarrow{t\to\infty} x^*$$

, where x^* is the optimal solution we are trying to compute. In most cases, convergence to the optimum will be realized only at infinity. Our iterative algorithm will instead stop after a certan number of iterations T and produce an approximate solution $x^{(T)}$. The quality of our approximation will be measured by some notion of distance from optimum and will be parametrized by ϵ. For instance, we may want to guarantee that the output satisfies: $\|x^{(T)} - x^*\| \leq \epsilon$.

In this lecture, we will focus on the problem of solving a psd linear system $Ax = b$. In the process, we will start to look at gradient descent as an iterative algorithm to optimize strongly convex functions.

Definition 1. For a matrix A, the sparsity $\text{nnz}(A)$ is the number of non-zero entries of A. Notice that if the matrix is the Laplacian L of a graph $\text{L} = O(m)$, where m is the number of edges of the graph.

The iterative methods will focus on in the next few lectures peform a number of iterations, where the running time of each iteration is dominated by a matrix-vector multiplication involving matrix A. This operation can be performed in $nnz(A)$-time. Hence, from here on, our running time analyisis will focus on the number of iterations required.

3 Iterative Algorithms for PD Linear Systems

3.1 Optimization Characterization

Consider a full-rank square linear system $Ax = b$, where A is positive definite (i.e., symmetric and all eigenvalues are positive). We can cast the task of solving for x as a convex optimization problem

as follows. Define the function $f : \mathbb{R}^n \to \mathbb{R}$:

$$f(x) = \frac{1}{2}x^T Ax - b^T x.$$

Exercise 1. *Check that $f(x)$ is convex by computing the Hessian $\nabla^2 f(x)$.*

The minimum of f is achieved when $\nabla f(x) = 0$. As $\nabla f(x) = Ax - b$, we have $x^* = A^{-1}b$. There are two other useful ways of thinking about $f(x)$ by completing the square.

$$f(x) + b^T A^{-1} b = (Ax - b)^T A^{-1}(Ax - b) = (x - A^{-1}b)^T A(x - A^{-1}b) = (x - x^*)^T A(x - x^*).$$

This motivates us to consider the A-norm $\|y\|_A \stackrel{\text{def}}{=} \sqrt{y^T Ay}$. In particular, our approximation guarantee will be:

$$f(x^{(T)}) - f(x^*) \leq \epsilon \leftrightarrow \frac{1}{2}\|x^{(t)} - x^*\|_A^2 \leq \epsilon$$

3.2 Gradient Descent Algorithm

Our iterative algorithm has a simple description. At every iteration t, we set:

$$x^{(t+1)} = x^{(t)} + \delta_t \varphi^{(t)}$$

We call δ_t the *step length*, $\varphi^{(t)}$ the *descent direction*. We want to choose δ_t and $\varphi^{(t)}$ to get a good improvement in the objective function. To this end, it makes sense to choose $\varphi^{(t)} = -\nabla f(x^{(t)})$, as this is the direction in which the function decreases the fastest.

Now the goal is $f(x^{t+1}) = (f(x^{(t)}) - \delta_t \nabla f(x^t)) << f(x^{(t)})$. The issue is how to pick the *step length* δ_t to have this happen.. There are two methods:

A. Fixed choice, i.e., $\delta_t = \delta$.

B. Adaptive choice, i.e., $\delta_t(x_t)$.

Today we are going to analyze a fixed choice of step length. This allows us to get a simple convergence bound for any system $Ax = b$.

Definition 2. Residual: For $\nabla f(x) = Ax - b$, we call $-\nabla f(x) = b - Ax = r_x$ the residual.

3.3 Convergence Proof

We need to find the relationship between $\|x^{(t+1)} - x^*\|_A^2$ and $\|x^{(t)} - x^*\|_A^2$. We can deduce it as follows:

$$\begin{aligned} \|x^{(t+1)} - x^*\|_A^2 = \|(I - \delta A)x^{(t)} + \delta b - x^*\|_A^2 = \\ \|(I - \delta A)x^{(t)} - (I - \delta A)x^*\|_A^2 = \|(I - \delta A)(x^{(t)} - x^*)\|_A^2 = \\ \|(I - \delta A)\|^2 \|(x^{(t)} - x^*)\|_A^2 \end{aligned}$$

Recall, $\nabla f(x) = Ax - b$, we have $-\nabla f(x) = b - Ax$. We can rewrite that $x^{(t+1)} = x^{(t)} + \delta(b - Ax^{(t)}) = (I - \delta A)x^{(t)} + \delta b$. Hence, we should pick δ such that $\|I - \delta A\| \leq 1$. Let us review the notion of matrix norm for a positive definite matrix.

Lemma 1. *For a psd matrix $B \in \mathbb{R}^{n\times n}$ with eigenvalues $0 < \beta_1 \leq \beta_2 \leq \ldots \leq \beta_n$.*

$$\|B\|^2 = \max_x \frac{x^T BBx}{x^T x} = \max_x \frac{\|Bx\|^2}{\|x\|^2} = \beta_n.$$

Let A have eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n > 0$. By the lemma above, we know that:

$$\|I - \delta A\| = \max(|1 - \delta\lambda_1|, |1 - \delta\lambda_n|).$$

To ensure that this is less than 1, we can pick

$$\delta = \frac{2}{\lambda_1 + \lambda_n} = \left(\frac{\lambda_1 + \lambda_n}{2}\right)^{-1}.$$

This choice yields $|1 - \delta\lambda_1| = \dfrac{|\lambda_n - \lambda_1|}{\lambda_1 + \lambda_n}$ and $|1 - \delta\lambda_n| = \dfrac{|\lambda_1 - \lambda_n|}{\lambda_1 + \lambda_n}$. Therefore,

$$\|I - \delta A\| \leq \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n} = 1 - \frac{2\lambda_n}{\lambda_1 + \lambda_n}$$

under the condition that $\delta = \dfrac{2}{\lambda_1 + \lambda_n}$.

We can now derive the relationship between $\|x^{(t+1)} - x^*\|_A^2$ and $\|x^{(t)} - x^*\|_A^2$ as follows:

$$\|x^{(t+1)} - x^*\|_A^2 \leq \left(1 - \frac{2\lambda_n}{\lambda_1 + \lambda_n}\right)^2 \|x^{(t)} - x^*\|_A^2$$

$$\leq \left(1 - \frac{2\lambda_n}{\lambda_1 + \lambda_n}\right)^{2t} \|x^{(0)} - x^*\|_A^2 \leq \exp\left\{-\frac{4t\lambda_n}{\lambda_1 + \lambda_n}\right\} \|x^{(0)} - x^*\|_A^2$$

To guarantee the approximation, we can set

$$t = \frac{\lambda_1 + \lambda_n}{4\lambda_n} \cdot \log\left(\frac{\|x^{(0)} - x^*\|_A^2}{\epsilon}\right) = \left(\frac{1}{4} + \frac{1}{4} \cdot \frac{\lambda_1}{\lambda_n}\right) \log\left(\frac{\|x^{(0)} - x^*\|_A^2}{\epsilon}\right) =$$

$$\left(\frac{1}{4} + \frac{K(A)}{4}\right) \log\left(\frac{\|x^{(0)} - x^*\|_A^2}{\epsilon}\right),$$

where the condition number is $\dfrac{\lambda_1}{\lambda_n} = K(A)$. For larger K, we need more steps to approach the optimal solutions.

3.4 Running Time

We know that $x^{(t+1)} = (I - \delta A)x^{(t)} + \delta b = x^{(t)} - \delta A x^{(t)} + \delta b$, so the total running time is $O\left(m \cdot K \cdot \log\left(\dfrac{\|x^{(0)} - x^*\|_A^2}{\epsilon}\right)\right)$, where $O(m)$ is the sparsity of A, and K is the condition number.

3.5 Examples

Consider the case when A is a normalized Laplacian:

$$A = \mathscr{L} = \left(I - D^{-1/2}AD^{-1/2}\right) = D^{-1/2}(I - W)D^{-1/2}.$$

The algorithm above iterates the following update:

$$x^{(t+1)} = (I - \delta\mathscr{L})x^{(t)} + \delta b = \left((1-\delta)I + \delta D^{-1/2}AD^{-1/2}\right)x^{(t)} + \delta b = D^{-1/2}((1-\delta)I + \delta W)D^{1/2}x^{(t)} + \delta b.$$

This is equivalent to:

$$D^{1/2}x^{(t+1)} = [(1-\delta)I + \delta W]D^{1/2}x^{(t)} + \delta D^{1/2}b.$$

This shows that at every iteration a lazy random walk takes place and a bit of b is added to the solution.