Lecture 7: More on Graph Eigenvalues, and the Power Method

Aditya Bhaskara[*]

CS 5968/6968: Techniques in Algorithms and Approximation

February 2nd, 2016

Abstract

We will discuss a few basic facts about the *distribution* of eigenvalues of the adjacency matrix, and some applications. Then we discuss the question of computing the eigenvalues of a symmetric matrix.

1 Eigenvalue distribution

Let us consider a d-regular graph G on n vertices. Its adjacency matrix A_G is an $n \times n$ symmetric matrix, with all of its eigenvalues lying in $[-d, d]$.

How are the eigenvalues *distributed* in the interval $[-d, d]$? Are there always many negative eigenvalues? What is the typical magnitude of the eigenvalues? The key to answering these questions is the simple fact that the trace of a matrix is the sum of its eigenvalues. Since all the diagonal entries of A_G are 0 (the graph has no self loops), we have that

The trace, denoted $Tr(\cdot)$, is defined to be the sum of the diagonal entries of a matrix.

$$Tr(A) = \sum_i \lambda_i = 0.$$

This means that the *average* of the eigenvalues is 0. Since we know that one of the eigenvalues is d, there have to exist eigenvalues that are < 0.

What is the typical *magnitude* of the eigenvalues? One way to measure this is to look at the average value of λ_i^2, i.e., $(1/n)\sum_i |\lambda_i|^2$. To compute this, the idea is to come up with a matrix whose eigenvalues are λ_i^2, for $i = 1, \ldots, n$, and compute its trace. We note that A_G^2 is such a matrix.

This is a special case of a more general phenomenon – for any polynomial $p()$, the eigenvalues of $P(A)$ are $p(\lambda_i)$, where λ_i are the eigenvalues of A.

What is the trace of A_G^2? Let us consider the (i,i)th entry. It is precisely $\langle A_i, A_i \rangle$, where A_i is the ith row (or column) of A_G. For any i, this inner product is equal to $\sum_j A_{ij}^2 = d$, since precisely d of the entries are 1 and the rest are zero. Thus the trace is the sum over i of this quantity, which is nd. Thus, we have

$$\frac{1}{n}\sum_i \lambda_i^2 = d$$

Thus, we expect the *typical* eigenvalue to have magnitude roughly $\sqrt{d}$. While this is not true of arbitrary graphs (see HW), it turns out that for *random* graphs of degree d, all the eigenvalues except the top one (which is d) turn out to lie between $-2\sqrt{d}$ and $2\sqrt{d}$. In fact, they are *distributed* in a very nice way. See: *https://en.wikipedia.org/wiki/Wigner_semicircle_distribution*.

*bhaskara@cs.utah.edu

1.1 exercise. Show that we need not have eigenvalues that are $\sim \sqrt{d}$. We could have n/d eigenvalues that are $\sim d$, roughly.

1.1 *Higher powers of the eigenvalues and walks*

A nice combinatorial connection exists between powers of the adjacency matrix and the graph. Let us consider A_G^3, for concreteness.

What is the i,j'th entry of A_G^3? If we write $A = A_G$, and $B = A_G^2$, we the quantity we are interested in, is

$$AB(i,j) = \sum_k A(i,k)B(k,j) = \sum_{k,l} A(i,k)A(k,l)A(l,j).$$

The sum is over all the possible choices of k and l. The term in the summation is non-zero precisely when ik, kl, lj are all edges in the graph. Thus the i,j'th entry of A_G^3 measures exactly the number of *walks* of length 3 between i and j in the graph.

A walk is different from a simple path in that vertices can be repeated. For instance, we could have picked $i-k-i-j$, and that is a valid walk.

One consequence of this, is the fact that $\mathrm{Tr}(A_G^3)$ is three times the number of *triangles* in the graph! Why? From the above, we know that the i,i'th entry of A_G^3 is the number of walks of length-3 between i and itself. A length-3 walk between i is exactly the number of triangles with i as one of the vertices (note that there is no way we can have repeated vertices in a walk of length 3 from i to itself). Every traingle is counted three times when we take the trace – once for each of its end-points. Thus $\mathrm{Tr}(A_G^3)$ is three times the number of triangles.

The walk interpretation of the adjacency matrix is useful – it lets us use properties of the graph to infer things about the distribution of eigenvalues and vice-versa.

2 Computing Eigenvalues

We have so far defined eigenvalues as the roots of the characteristic polynomial (the values λ such that $\det(A - \lambda I) = 0$), and we iteratively defined λ_i as minimizers of the quadratic form $x^T Ax$ over unit vectors x.

How do we efficiently compute eigenvalues efficiently, given a matrix A. Suppose for now that A is an $n \times n$ real, symmetric matrix, which implies it has n real eigenvalues. Call them $\lambda_1, \ldots, \lambda_n$, and let $v_1, v_2, \ldots, v_n$ be the corresponding eigenvectors. Then, we saw that v_i form an orthonormal basis for $\mathbb{R}^n$. Furthermore, we saw that we can write

$$A = \sum_i \lambda_i v_i v_i^T. \tag{1}$$

2.1 *Power Iteration*

Suppose we start with some vector $x \in \mathbb{R}^n$, and compute

$$Ax, A^2x, A^3x, \ldots$$

Can we analyze what happens? It turns out that the right way to see what is going on is by writing x in terms of the eigenvectors. Suppose $x = \sum_i \alpha_i v_i$, for some α_i (since the v_i form an orthonormal basis, there is a unique representation of x in this manner).

Then, using (1), we observe that

$$
\begin{aligned}
Ax &= \sum_i \alpha_i \lambda_i v_i \\
A^2 x &= \sum_i \alpha_i \lambda_i^2 v_i \\
&\vdots \\
A^r x &= \sum_i \alpha_i \lambda_i^r v_i
\end{aligned}
$$

Thus the coefficients of v_i *evolve* in a very clean way when we repeatedly multiply by A. Let us see a simple example. Suppose we have $n = 3$, and suppose the eigenvalues are $-1, 1, 2$. Then, if we start with an x as above, we have $A^r x = (-1)^r \alpha_1 v_1 + \alpha_2 v_2 + 2^r \alpha_3 v_3$. Now the crucial thing to observe is that the coefficient of v_3 grows at a much faster rate than the coefficients of v_1 and v_2. Suppose we started with all α_i being equal to 1. Then, after 10 steps, the vector we have is $v_1 + v_2 + 1024 v_3$, which when normalized is almost entirely aligned with v_3!

This is a general phenomenon. As long as we have one eigenvalue that is strictly larger than the others in magnitude, the term corresponding to that eigenvalue dominates, for large enough r.

2.1 theorem. *Suppose the eigenvalues of A satisfy* $\max_{i<n} |\lambda_i| < (1-\delta)|\lambda_n|$, *and let $\epsilon \in (0,1)$ be the desired accuracy. Then for any vector x as above, consider $A^r x$, for*

It is important to look at the magnitudes. Note also that sometimes the most negative eigenvalue could be the one with the largest magnitude.

$$
r \geq \frac{\log D}{2\delta}, \qquad \text{where } D = \frac{\sum_{i<n} \alpha_i^2}{\epsilon^2 \alpha_n^2}.
$$

Then we have $\| \frac{A^r x}{\|A^r x\|} - v_n \| < \epsilon$.

Proof. The proof easily follows from what we observed earlier, and straightforward calculation. For any r, we have

$$
A^r x = \sum_i \alpha_i \lambda_i^r v_i = \alpha_n \lambda_n^r \left(v_n + \sum_{i<n} \frac{\alpha_i}{\alpha_n} \left(\frac{\lambda_i}{\lambda_n} \right)^r v_i \right).
$$

Let us consider the norm of the term in the summation. Since the v_i are all orthogonal, the squared-norm is

$$
\sum_{i<n} \frac{\alpha_i^2}{\alpha_n^2} \left(\frac{\lambda_i}{\lambda_n} \right)^{2r} < \frac{\sum_{i<n} \alpha_i^2}{\alpha_n^2} (1-\delta)^{\log D/\delta}.
$$

Using the familiar inequality $1 - \delta \leq e^{-\delta}$ and simplifying, we get that the squared-norm is $< \epsilon^2$. Now, the vector in the summation is orthogonal to v_n, since the v_i's are all orthogonal.

Thus, we have written $A^r x = C(v_n + v_n^{\perp})$, where $v_n^{\perp}$ is orthogonal to v_n and has norm $< \epsilon$. This implies the theorem. (Details left as an exercise.) □

Now consider the following algorithm: (called *power iteration*)

1. start with a random $x \in \mathbb{R}^n$
2. repeat r times: $x \leftarrow (Ax)/\|Ax\|$

What r should we choose? The theorem gives an r that works, but note that we do not know the values α_i without knowing the v_i (which are what we are after in the first place!). Here's where the starting point being *random* comes in. With good probability (at least 99%), if x is chosen randomly (say, from the n dimensional Gaussian distribution), we will have $\sum_i \alpha_i^2/\alpha_n^2 \leq O(n)$, implying that choosing

If you do not see this immediately, it is a good exercise.

$$r = \frac{\log(n/\epsilon)}{\delta}$$

works with good probability.

The main factor determining the running time is $(1/\delta)$, which is often called the *eigenvalue gap*. Power iteration with a random starting point converges quickly if and only if the gap is large. In practice, the power method is a common tool in computing eigenvalues and eigenvectors.

What about matrices in which there is no (or very little) eigenvalue gap? We will see examples of this in the homework.

2.2 *Beyond the top eigenvalue*

What if we are interested in eigenvalues other than the *top* one? There are a couple of ways of extending the power method.

The natural one is to compute λ_n and v_n to a sufficiently high accuracy, and then *subtract it off* from the matrix. Since $A = \sum_i \lambda_i v_i v_i^T$, we will be left with $\sum_{i<n} \lambda_i v_i v_i^T$ (plus a small noise, which we will need to keep track of).

The second way, which is often much better, is what is called the *block power method*. This works well if we are interested in the top-k eigenvalues in magnitude (for a small k).

The idea is to keep an $n \times k$ matrix X (instead of a vector x), and repeatedly compute AX, followed by an orthonormalization step (instead of simply a normalization). It turns out that a similar analysis can be done, and now the convergence depends on the gap between the kth largest, and the $(k+1)$st largest eigenvalues.