CS-621 Theory Gems October 18, 2012

Lecture 10

Lecturer: Aleksander Mądry *Scribes: Mani Bastani Parizi and Christos Kalaitzis*

1 Introduction

In this lecture, we will see how one can use random walks to tackle the web search problem, i.e., we will describe the PageRank, a concept that is one of the key ideas behind Google's success.
Later, we introduce the fundamental object of spectral graph theory: the Laplacian matrix. We conclude by proving the Courant-Fisher theorem that provides a general technique for deriving upper bounds on the eigenvalues of symmetric matrices.

2 PageRank

Consider the web search problem, in which one has an index of a large number of web pages (think: Google's web crawling archives) and wants to present to the user web pages that will be most relevant to the query that he/she entered.

One can view this task as consisting of three steps:

1. The user inputs his/her query;
2. The engine is finding all the pages in its index that contain this query;
3. These pages are sorted according to their "rank" – that should reflect relevance – and the results are presented to the user in that order.[1]

Clearly, the first two task are fairly simple (at least conceptually), and the most crucial and non-trivial step is the last one. After all, the number of pages found in the second step will likely be extremely large, so the users will be able to look only through a tiny fraction of them and it is important that the ranking ensures that the most relevant ones are presented first.

In the ancient times of Internet, the existing solutions for the ranking problem were very unsatisfactory, and only the emergence of Google with its completely new approach to this task dramatically improved the situation (and led to Google to its success).

In this approach, one models the Web as a gigantic directed graph $G = (V, E)$ – so-called *web graph* – with nodes corresponding to individual pages and arcs reflecting the links between these pages, i.e., there is an arc (u, v) in the graph G if page u has a link to page v. (Note that this graph will be rather sparse, i.e., the number of its arcs will be not much larger than the number of nodes, as usually pages have only limited number of outgoing links.)

Now, we want to come up with a ranking of all the pages. We will view ranking as a vector $rank \in \mathbb{R}^V$ that associates with each vertex v a real number reflecting its rank. The general – and somewhat counterintuitive – idea here is to c completely ignore the actual content of the pages when inferring this ranking. Instead, we will compute the ranking of pages relaying purely on the structure of the web graph itself.

More precisely, we will treat the links between different pages as a form of "recommendation", i.e., if page u links to page v, we interpret it as page u "recommending" page v. (This way, instead of trying to make the computer "understand" the content of pages and thus their relevance – which would be an extremely hard task – we leverage the fact that if a given page is really containing useful information (as judged by humans) it is more likely to have many links pointing to it.)

[1] Note that the ranking here is assumed to be static, i.e., independent of the actual query. So, we are trying here to capture some intrinsic relevance of web pages – we believe that such intrinsically relevant pages that contain the query that the user supplied are indeed most relevant to that query.

As we will see shortly, there exists a very elegant way of aggregating all these recommendations into a ranking – so-called *PageRank* – that turns out to be indeed very good in practice and outperforms the previous approaches to this task that were based on analyzing the web page content. Following [1], we will develop PageRank as a sequence of natural refinements of a the basic idea we outlined above.

First Idea. As already mentioned, we expect the number of links pointing to a given page to be an indication of its relevance. So, the first idea – let us denote the resulting ranking as *countrank* – is to simply count, for each vertex u, the number of incoming arcs, i.e.,

$$countrank_u = \sum_{v \in V} A_{v,u}.$$

where A is the adjacency matrix of the graph that we introduced in Lecture 9, defined as

$$A_{u,v} = \begin{cases} 1 & \text{if } (u,v) \in E \\ 0 & \text{otherwise.} \end{cases}$$

So, we can write the definition of *countrank* compactly as

$$countrank = A\mathbf{1} \tag{1}$$

where $\mathbf{1}$ is all-ones vector.

Although very natural, *countrank* does not perform too well. To understand why, consider the following situation. There are two pages describing good restaurants in a city. The first one lists 100 restaurants, while the second one only 3. In *countrank*, recommendations of both pages carry the same weight while, in reality, the recommendation of the latter page should be worth more than that of the former one (after all, being in "top 3" carries more weight than being in "top 100").

Second Idea. To alleviate the above shortcoming, we modify our ranking method to make it weight recommendation of each page v by the inverse of the total number of recommendations it makes (i.e., the number of its outgoing links $d(v)$). This gives rise to a new ranking that we will call *weightrank* defined as

$$weightrank_u = \sum_{v \in V} A_{v,u} \frac{1}{d(v)}.$$

Again, in matrix form, we will have

$$weightrank = AD^{-1}\mathbf{1} = W\mathbf{1} \tag{2}$$

where W is the random walk matrix introduced in the Lecture 9, and D is an *out-degree matrix*[2] defined as

$$D^{-1}_{u,v} = \begin{cases} \frac{1}{d(v)} & \text{if } u = v \\ 0 & \text{otherwise.} \end{cases}$$

(To ensure that matrix D^{-1} is well-defined even when some vertex v does not have any outgoing links and thus $d(v) = 0$, we just add a self-loop to it.[3])

In the light of (2), after appropriate normalization, one can view *weightrank* as a ranking that reflects the probability that we will end up at some particular page u, after choosing first a page v uniformly at random and then taking one step of (directed) random walk in G.

[2]Note that in Lecture 9 we also defined a degree matrix D, but the difference is that here we only count the outgoing links.

[3]Doing this transformation may distort the ranking, as each such v has now an additional (self-)recommendation that is very strong (as it comes from a vertex with out-degree 1). However, as we will see shortly, our further refinement of the ranking method will take care of this problem.

Third Idea. Although the *weightrank* ranking performs better, there is still some room for improvement. In particular, one idea is to weight each recommendation of a page v not only based on the total number of recommendations made by v, but also on the ranking of the page v itself. That is, recommendation of a page that itself is well-recommended should have more impact.

Formally, we capture this refinement by defining a *recrank* ranking of a vertex u to be

$$recrank_u = \sum_{v \in V} A_{v,u} \frac{1}{d(v)} recrank_v,$$

which in matrix form can be expressed as

$$recrank = W recrank. \tag{3}$$

Observe that if we insist that *recrank* a probability distribution then (3) would make *recrank* be a stationary distribution of the directed random walk in our web graph.

Note, however, that since the web graph is directed, it is not clear how the corresponding stationary distribution would look like or even whether it at all exists. (The explicit description of the stationary distribution that we stated in Lecture 9, is valid only for undirected graphs.)

Fortunately, a stationary distribution exists in every directed graph, as long as, each vertex has its out-degree non-zero (which we ensured is true for our web graph).

Claim 1 *For any directed graph $G = (V, E)$ such that the out-degree $d(v)$ of every vertex $v \in V$ is positive, there exists a non-zero solution to the Equation* (3)*.*

Proof Observe that, as all vertices have non-zero out-degree, entries of each column of the walk matrix W sum up to one. Therefore, entries of each column of the matrix $I - W$ sum up to zero.

Now, if we take all the rows of the matrix $I - W$ but the last one, and add them to the last row then this row becomes an all-zeros row. This means that the matrix $I - W$ is *not* full-rank.

From basic linear algebra we know that the null-space of a rank-deficient matrix is non-trivial. In other words, there exists a vector x such that $(I - W)x = \mathbf{0}$ and $x \neq \mathbf{0}$. This shows that (3) has indeed a non-zero solution. ■

Now, to be able to conclude that a stationary distribution exists, we would need to also show that there exists a solution to (3) that is not only non-zero, but also has all its entries non-negative. This can be shown to be indeed the case, but we omit the proof.

Unfortunately, even though we know now that the stationary distribution exists, it does not mean that the vector *recrank* is uniquely defined, as in principle a directed graph can have many different stationary distributions. To see this, note that for any vertex v whose only outgoing arc is a self-loop, a distribution that assigns 1 to v and 0 to all the other vertices, is a stationary distribution. (One can show that a necessary and sufficient condition for a directed graph to have a unique stationary distribution is that the graph is *strongly connected*,i.e., that for any two vertices v and u there has to be a directed path from u to v and from v to u.)

To fix the above problem, we just change the definition of our ranking slightly and obtain the desired PageRank vector.

Definition 2 (Page Rank) *For a graph $G = (V, E)$, the PageRank vector pr is defined as solution of the following equation*

$$pr = (1 - \alpha) W pr + \frac{\alpha}{n} \mathbf{1}, \tag{4}$$

where $n = |V|$ and $0 < \alpha < 1$ is a parameter of choice.

One can view the PageRank vector as corresponding to a stationary distribution of a randomized process – sometimes called "random surfer" model – in which in every step, with probability $(1-\alpha)$ one performs a step of random walk and, with probability α, one teleports himself/herself to a random node of the graph.

It can be shown that PageRank vector exists and is uniquely defined for any directed graph with all out-degrees being non-zero.[4] To prove that in the special case of undirected graphs, one just needs to note that

$$pr = (I - (1-\alpha)W)^{-1} \frac{\alpha}{n}\mathbf{1}.$$

(As we know that for undirected graphs the eigenvalues of W lie in the interval $[-1,1]$, the eigenvalues of $I-(1-\alpha)W$ are strictly positive, for any $\alpha > 0$, which makes the matrix $I-(1-\alpha)W$ invertible.)

Applying the identity $(I-M)^{-1} = \sum_{t\geq 0} M^t$ to the above equation allows us to express the PageRank vector as

$$pr = \sum_{t\geq 0} \alpha(1-\alpha)^t W^t \frac{1}{n}\mathbf{1}. \tag{5}$$

This shows that the parameter α is controlling how quickly the "random surfer gets bored". To see that, note that each of the terms in the above series corresponds to an event that the surfer starts from a random page ($\frac{1}{n}\mathbf{1}$) and then continues the random walk for t steps (W^t). The probability of this event is $\alpha(1-\alpha)^t$, so the smaller the α the less likely such long runs are.

3 The Laplacian Matrix

In this section, we introduce a fundamental object of spectral graph theory: the Laplacian matrix of a graph. Similarly to the case of walk matrix, we constrain our discussion here to undirected graphs. Also, it will be convenient for us to work with weighted versions of graphs.

To this end, let us consider a weighted and undirected graph $G = (V, E, w)$, where w is a $|E|$-dimensional vectors assigning non-negative weights to edges. As already briefly mentioned in Lecture 9, we can generalize the definition of adjacency and degree matrices to weighted graphs as follows

$$A_{u,v} := \begin{cases} w_e & \text{if } e = (u,v) \in V \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

$$D_{u,v} := \begin{cases} 0 & \text{if } u \neq v \\ d_w(u) & \text{if } u = v \end{cases} \tag{7}$$

where $d_w(u)$ is the *weighted degree* of vertex u given by

$$d_w(u) := \sum_{e=(u,v)\in E} w_e$$

Now, the *Laplacian* matrix of the graph G is defined as

$$L := D - A. \tag{8}$$

One can verify that

$$L_{u,v} = \begin{cases} \sum_{e=(u,v')} w_e & \text{if } u = v \\ -w_e & \text{if } e = (u,v) \in E \text{ and } u \neq v \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

[4] To handle vertices without any outgoing arcs, one usually just adds to the graph arcs connecting this vertex to every other vertex. In this way, in the "random surfer" model, whenever such vertex is reached, one teleports to a uniformly random vertex with probability 1 (instead of usual α).

It will be sometimes convenient to view the Laplacian L of the graph G as a sum of $m = |E|$ Laplacians that correspond to each of its edges. Formally, for a given edge $e = (v_1, v_2) \in E$, let us define L^e to be the Laplacian of a graph on vertex set V that has only one edge e. That is,

$$L^e_{u,v} = \begin{cases} w_e & \text{if } u = v = v_1 \text{ or } u = v = v_2 \\ -w_e & \text{if } (u,v) = (v_1, v_2) \text{ or } (u,v) = (v_2, v_1) \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

We now have that

$$L = \sum_{e \in E} L^e. \tag{11}$$

We will be often interested in the quadratic form of Laplacian L given by

$$x^T L x.$$

Using (11) and (10) one can see that

$$x^T L x = \sum_{e=(u,v) \in E} w_e (x_u - x_v)^2. \tag{12}$$

It turns out that this quadratic form allows us to understand some important properties of the spectrum of the Laplacian.

To this end, let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ with $n = |V|$, be all the eigenvalues of the Laplacian (indexed in non-decreasing order) and let $v^1, \ldots, v^n$ be the corresponding eigenvectors. (As we recall from Lecture 9, since Laplacian is a symmetric matrix, it has n real eigenvalues and the corresponding eigenvectors can be chosen to be normalized and orthogonal to each other.)

Theorem 3 *We have $0 = \lambda_1 \leq \ldots \leq \lambda_n$ and if the graph G is connected then $\lambda_2 > 0$.*

Proof To see that $0 \leq \lambda_1$, we note that by (12)

$$(v^1)^T L v^1 = \sum_{e=(u,v) \in E} w_e (v^1_u - v^1_v)^2 \geq 0,$$

as each term in that sum is always non-negative. On the other hand, we have that

$$(v^1)^T L v^1 = \lambda_1 (v^1)^T v^1 = \lambda_1.$$

So, indeed $\lambda_1 \geq 0$.

To see that $\lambda_1 = 0$, consider an all-ones vector $\mathbf{1}$, we see that (12) implies

$$\mathbf{1}^T L \mathbf{1} = \sum_{e=(u,v) \in E} w_e (1-1)^2 = 0,$$

so indeed 0 is always one of the eigenvalues of the Laplacian.

Finally, to prove that $\lambda_2 > 0$ when G is connected, note that by (12) if a vector z is an eigenvector corresponding to eigenvalue 0 then

$$0 = z^T L z = \sum_{e=(u,v) \in E} w_e (z^2_u - z^2_v)^2.$$

However, this expression can be zero only if z is constant everywhere (that's were we use the connectedness of G). This in turn implies that z has to be co-linear with all-ones vector $\mathbf{1}$ and thus the dimension of the eigenspace corresponding to eigenvalue 0 has to be one. So, $\lambda_2 > 0$, as desired. ■

Note that one of the consequences of the above theorem is that

$$L \succcurlyeq 0.$$

That is, the Laplacian matrix is always positive semi-definite. (Recall that a symmetric matrix A is *positive semi-definite*, denoted as $A \succcurlyeq 0$, if for any vector x, $x^T A x \geq 0$.)

3.1 Normalized Laplacian

Another important matrix that is closely related to the Laplacian is the *normalized Laplacian* matrix defined as

$$\widehat{L} := D^{-\frac{1}{2}} L D^{-\frac{1}{2}}. \tag{13}$$

By Theorem 3, we can immediately conclude that if $\hat{\lambda}_j$ is the j-th smallest eigenvalue of $\widehat{L}$ and G is connected then

$$0 = \hat{\lambda}_1 < \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_n.$$

It turns out that there is a direct connection between the eigenvalues of $\widehat{L}$ and the eigenvalues of the walk matrix W.

Claim 4 *For any i, if ω_i is the i-th largest eigenvalue of the walk matrix W then*

$$\hat{\lambda}_i = 1 - \omega_i. \tag{14}$$

Proof

Note that by definition of $\widehat{L}$ and definition of the Laplacian L (cf. (8))

$$\begin{aligned}
\widehat{L} &= D^{-\frac{1}{2}} (D - A) D^{-\frac{1}{2}} \\
&= I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \\
&= I - D^{-\frac{1}{2}} W D^{\frac{1}{2}}
\end{aligned}$$

In Lecture 9, we showed that the eigenvalues of the matrix $S = D^{-\frac{1}{2}} W D^{\frac{1}{2}}$ are the same as those of W. This proves the claim. ■

4 Courant-Fisher Theorem

The goal of spectral graph theory is to connect various properties of a graph to the eigenvalues (spectrum) of its Laplacian. So, it is important that we are be able to estimate the values of eigenvalues of graphs to exploit these connections. Unfrotunately, due to the delicate nature of eigenvalues, it is rather unusual that we are able to exactly determine their value for a given graph Laplacian. Therefore, we tend to settle on getting just their loose estimates.

Today, we present one technique that can aid us in obtaining such estimates. The key idea here is realizing that we can view the eigenvalues as a result of certain optimization process. To see this, let us focus first on the smallest eigenvalue of a given symmetric matrix. One can show the following result.

Observation 5 *For an $n \times n$ symmetric matrix M, with $\alpha_1 \leq \ldots \leq \alpha_n$ being its eigenvalues*

$$\alpha_1 = \min_{x \in \mathbb{R}^n} \frac{x^T M x}{x^T x} \tag{15}$$

Proof Consider v^i to be the eigenvector of M that corresponds to eigenvalue α_i. We know that we can choose v^is to make them orthogonal to each other and normalized. Furthermore, as there is n of them, it must be the case that they form a basis in $\mathbb{R}^n$.

Therefore, any $x \in \mathbb{R}^n$ can be expressed as

$$x = \sum_{i=1}^{n} c_i v^i$$

where $c_i = (v^i)^T x$. Furthermore, we must have that $x^T x = \|x\|_2^2 = \|\bar{c}\|_2^2$, where $\bar{c} = (c_1, ..., c_n)$ is the expression of x in basis formed by the vectors v^i.

Now, we have that the value of our objective function on vector x can be written as

$$\frac{x^T M x}{x^T x} = \frac{\sum_{i=1}^{n} \alpha_i c_i^2}{\sum_{i=1}^{n} c_i^2}$$

As our objective is scaling-invariant, without loss of generality, we can assume that $\|\bar{c}\|_2^2 = 1$. Then, the above expression can be simplified as follows

$$\frac{\sum_{i=1}^{n} \alpha_i c_i^2}{\sum_{i=1}^{n} c_i^2} = \sum_{i=1}^{n} \alpha_i c_i^2$$

which means that the task of minimizing our objective function boils down to solving the following minimization problem

$$\min_{c:\|c\|_2^2=1} \sum_{i=1}^{n} \alpha_i c_i^2.$$

However, since α_1 is the smallest eigenvalue, it is not hard to see that the minimizer for this problem is a c^* such that c_1^*=1 and $c_i^* = 0$ for all $i > 1$. The claim follows. ■

Now, we just generalize this approach to make it capture in similar fashion any eigenvalue, not only the smallest one. (Although we state this theorem in the context of Laplacians, it holds for any symmetric matrix.)

Theorem 6 (Courant-Fisher) *Let L be a Laplacian matrix with λ_i being its i-th smallest eigenvalue, and let $\mathcal{S}_k$ denote the set of k-dimensional subspaces of $\mathbb{R}^n$, then*

$$\lambda_k = \min_{S \in \mathcal{S}_k} \max_{y \in S} \frac{y^T L y}{y^T y}. \tag{16}$$

Proof First, we prove that $\lambda_k \geq \min_{S \in \mathcal{S}_k} \max_{y \in S} \frac{y^T L y}{y^T y}$. In order to do that, choose $\bar{S} = \texttt{span}(v^1, \ldots, v^k)$, where v^i is the eigenvector corresponding to eigenvalue λ_i. Clearly, $\bar{S} \in \mathcal{S}_k$.

As $\lambda_i \leq \lambda_k$, for all $i \leq k$, by decomposing any $y \in \bar{S}$ into a linear combination of the k eigenvectors (i.e. by following the approach from the proof of Observation 5), we can see that

$$\max_{y \in \bar{S}} \frac{y^T L y}{y^T y} = \lambda_k,$$

as desired.

Next, we show that $\lambda_k \leq \min_{S \in \mathcal{S}_k} \max_{y \in S} \frac{y^T L y}{y^T y}$. To this end, let us define $T_k = \texttt{span}(v^k, ...v^n)$, i.e., T_k is the subspace spanned by all but first $k-1$ eigenvectors of L. Obviously, the dimension of T_k is equal to $n - k + 1$.

Now, let S^* be the minimizer of $\min_{S \in \mathcal{S}_k} \max_{y \in S} \frac{y^T L y}{y^T y}$. Since S^* has dimension k, a simple dimension argument implies that the intersection of S^* and T_k is non-empty. Therefore, we need to have

$$\max_{y \in S^*} \frac{y^T L y}{y^T y} \geq \max_{y \in S^* \cap T_k} \frac{y^T L y}{y^T y},$$

as $S^* \cap T_k \subseteq S^*$.

Let y^* be the maximizer of $\frac{y^T L y}{y^T y}$ among all $y \in S^* \cap T_k$. If we represent y^* in the basis spanned by the eigenvectors of L, we will have that $y^* = (0, 0..., c_k, ...c_n)$, where $c_i = (v^i)^T y^*$. Therefore

$$\frac{{y^*}^T L y^*}{{y^*}^T y^*} = \frac{\sum\limits_{i \geq k}^{n} \lambda_i c_i^2}{\sum\limits_{i \geq k}^{n} c_i^2} \geq \frac{\lambda_k \sum\limits_{i \geq k}^{n} c_i^2}{\sum\limits_{i \geq k}^{n} c_i^2} = \lambda_k$$

which proves that

$$\lambda_k \leq \max_{y \in S^* \cap T_k} \frac{y^T L y}{y^T y} \leq \max_{y \in S^*} \frac{y^T L y}{y^T y} = \min_{S \in \mathcal{S}_k} \max_{y \in S} \frac{y^T L y}{y^T y}$$

and thus completes the proof of the theorem. ■

Note that for $k = 1$ the above theorem reduces to Observation 5. Also, as we will be mostly interested in λ_2, i.e., the second-smallest eigenvalues of Laplacian, we provide below a more explicit version of the Courant-Fisher theorem for the case of $k = 2$.

Corollary 7

$$\lambda_2 = \min_{x \in \mathbb{R}^n : x \perp \mathbf{1}} \frac{x^T L x}{x^T x}$$

Proof By Theorem 6, we have

$$\lambda_2 = \min_{S \in \mathcal{S}_2} \max_{y \in S} \frac{y^T L y}{y^T y}.$$

Note that from the proof of that theorem we know that without loss of generality we can assume that S has to contain the eigenvector v^1 of L that corresponds to λ_1. So, since S has dimension two, it must be spanned by the vector v^1 and some other vector x that has to be orthogonal to v^1. Furthermore, we know that v^1 is just a scaled version of all-ones vector $\mathbf{1}$. Thus, $x \perp \mathbf{1}$.

So, our optimization problem simplifies to

$$\lambda_2 = \min_{x \in \mathbb{R}^n : x \perp \mathbf{1}} \max_{y \in \mathtt{span}(v^1, x)} \frac{y^T L y}{y^T y}.$$

However, by reasoning similar to the one from the proof of Observation 5, we know that y has to be orthogonal to v^1 (as v^1 corresponds to the smallest eigenvalue of L) and thus (due to scale-invariance of our objective) we can assume it to be equal to x. The corollary follows. ■

A crucial aspect of this corollary (as well as, of the Courant-Fisher theorem) is that to prove an upper bound on the value of λ_2, we do not need to really solve the corresponding optimization problem. As this is a minimization problem, just exhibiting any vector x that is orthogonal to all-ones vector, immediately proves that $\lambda_2 \leq \frac{x^T L x}{x^T x}$. As we will see later, exhibiting such vectors x that provide very good (but not optimal!) upper bounds on λ_2, will be much easier than figuring out the exact value of this eigenvalue.

References

[1] Brian White, *How Google Ranks Web Pages*, `http://math.stanford.edu/~brumfiel/math_51-06/PageRank.pdf`.