18.409 An Algorithmist's Toolkit October 6, 2009

Lecture 8

Lecturer: Jonathan Kelner *Scribe: Alessandro Chiesa (2009)*

1 Administrivia

You should probably know that

- the first problem set (due October 15) is posted on the class website, and
- its hints are also posted there.

Also, today in class there was a majority vote for posting problem sets earlier. Professor Kelner will post the problem sets from two years ago, but he reserves the right to add new problems once a problem set has already been posted.

Questions from last time.

- *What is a level set?* The level set of a function corresponding to a (fixed) constant c is the set of points in the function's domain whose image equals c.
- *What is a good reference on applications of expander graphs?* A course taught by Nathan Linial and Avi Wigderson [3].

Plan for today. We use what we proved last time to obtain a local clustering algorithm from a random walk scheme. Then, noting that similar results to the ones proved last time also hold for PageRank, we obtain a second scheme that yields a second, better local clustering algorithm. Finally, we briefly motivate the technique of sparsification, which we will discuss next time.

2 Local and Almost Linear-Time Clustering and Partitioning

2.1 Review of Local Clustering

Let us briefly review local clustering, which we introduced last time. Given a vertex v in some graph G, we would like know if v is contained in a cluster, i.e. a subset of vertices that defines a cut with low conductance. However, we want the running time of our algorithm to depend on the *cluster size*, and not on the size of the graph. Last time we mentioned that a good example of a problem of this sort is trying to find a cluster of web pages around `mit.edu`; we surely do not want the running time of this task to depend on the number of sites created on the other side of the world. Let us make our goal a little bit more precise: in this lecture we will describe an algorithm that, after running for time almost linear in K, outputs a cluster of size at least $K/2$ around the starting vertex, if such a cluster exists.

2.2 General Strategy

We observe that if we run a random walk starting from some vertex v contained in a cluster, then low-conductance cuts will be an obstacle to mixing; i.e., the random walk has trouble leaving the cluster. Hence, a good guess for the cluster is the set of vertices with the highest probability masses after a given number of steps (of a random walk that started at v). Last time we showed that this makes sense by proving the Lovász-Simonovits theorem [4].

Therefore, a good primitive to construct an almost linear-time global algorithm is the following. Run a random walk starting from v, and, at each step, for every vertex w, approximate the probability that the random walk is at w; then take the vertices with the k largest probability masses as a possible cut. Repeat this until you get a good cut or you reach a predetermined limit.

2.3 Obstacles

We need a bound that says that our general strategy works, and that is why we proved the Lovász-Simonovits theorem. However, the bound we have is *global*, i.e., it involves the conductance $\phi(G)$ and we do not have the time to compute λ_2 for the whole graph to bound the conductance. Moreover, if we exactly compute all the probabilities of the random walk, it will take too long. Finally, even if we approximate the probabilities, we would need a stronger bound, and the goodness of the approximation depends on the cluster size, which we do not know in advance.

2.4 One Solution

A reasonable solution goes as follows. We recall that the proof of the Lovász-Simonovits theorem that we discussed last time used cuts on level sets of ρ^t. This implies that if a walk does not mix too quickly, we know that one of the cuts had bad conductance. Therefore, obtain the following corollary from the Lovász-Simonovits theorem.

Corollary 1 *Let $G = (V, E)$ be a connected, undirected graph with m edges and let $\pi(x)$ be its stationary distribution $\frac{d_x}{\sum_{v \in V} d_v}$. For every subset of vertices $W \subset V$ and and every time t, if $x \equiv \sum_{w \in W} d_w$ and $\varphi(W)$ is the conductance of the cut $(W, \overline{W})$, then the following inequality holds:*

$$\left| \sum_{w \in W} p^t(w) - \pi(w) \right| \leq \min\left(\sqrt{x}, \sqrt{2m - x}\right) \left(1 - \frac{1}{2}\phi(W)^2\right)^t .$$

Note that in the last lecture we stated a slightly weaker form of the theorem, where the conductance $\varphi(W)$ of the cut $(W, \overline{W})$ was replaced by the conductance $\phi(G)$ of *the whole graph*. Nevertheless, we did actually prove the stronger version stated above.

The bound above has nothing to do with global properties of the graph. Therefore, we can use Corollary 1 for local clustering in the following way. If after $O\left(\left(\frac{\log m}{\phi}\right)^2\right)$ steps a set of vertices contains a constant factor more than what it would have under the stationary distribution, then we can get a cut C such that $\varphi(C) \leq \phi$. (The cut can be obtained by mapping the probabilities to the real line and cut like we did with v_2 a few lectures ago).

A problem with this approach is that computing all the probabilities will be too slow. In particular, after only a few steps we will have too many nonzero values to keep track of. Lovász and Simonovits proposed to simply zero out the smaller probabilities and then prove that it does not hurt much to do so. However, the analysis is really messy. Instead, Andersen, Chung, and Lang [1] propose an approach that, instead of using the probability vector of a lazy random walk, uses a slightly different vector called PageRank; we discuss this approach in the following section.

(Note that for all of this to work we still need to prove a partial converse. Indeed, one can show that if there exists a cut C of conductance ϕ^2, then at least $|C|/2$ of its vertices will give a cut of conductance ϕ, otherwise the random walk would mix too quickly.)

3 PageRank

3.1 Definition

Consider an undirected[1] connected graph $G = (V, E)$. Recall that a *simple random walk* on G is a walk that, starting at some initial vertex, at each step moves from the current vertex to a randomly chosen neighbor of the vertex; a *lazy random walk* on G is a walk that, starting at some initial vertex, at each step with 0.5 probability stays on the current vertex and with 0.5 probability moves from the current vertex to a randomly chosen neighbor of the vertex.

We now consider a new Markov process that is a modification of a lazy random walk on a graph. Fix some distribution s over the vertices V of G and fix a parameter $\alpha \in (0, 1)$ (called the *teleport probability*). Starting from some initial vertex, at each step of the process we do the following: with probability $1 - \alpha$ we take a step of a lazy random walk on G, and with probability α we "teleport" to a vertex drawn from s. For simplicity, we will take s to be a single vertex, i.e., all the probability mass is concentrated on one vertex.

The process converges to a stationary distribution (because it corresponds to an aperiodic, irreducible Markov chain). For consistency with [1], we denote this stationary distribution (which depends on the parameters s and α) by $\mathsf{pr}_\alpha(s)$ and call it the *PageRank vector*; note that $\mathsf{pr}_\alpha(s)$ is a vector in $\mathbb{R}^n$, where $n = |V|$. Moreover, it is easy to see that the stationary distribution $\mathsf{pr}_\alpha(s)$ is the unique solution to the following equation:

$$\mathsf{pr}_\alpha(s) = \alpha s + (1 - \alpha) W \mathsf{pr}_\alpha(s) \ , \tag{1}$$

where W is the transition matrix corresponding to a lazy random walk on G.

The point is that one can show that the Lovász-Simonovits theorem and its corollary hold for the PageRank vector $\mathsf{pr}_\alpha(s)$, where s corresponds to the starting vertex and α corresponds to the number of time steps. Hence, rephrasing the discussion in Section 2.4, we know that if a subset of vertices S contains more than a constant factor more probability under $\mathsf{pr}_\alpha(s)$ than under the stationary distribution, then we can find a cut with conductance $O(\sqrt{\alpha \log \sum_{v \in S} d_v})$. Moreover, approximating the PageRank vector $\mathsf{pr}_\alpha(s)$ is *robust under small errors*, because it is the solution of an equation rather than being the result of many successive computations each with approximations.

Next, we prove some properties about the PageRank vector and then show how to approximate it.

(Note that, just like before, we still need to prove a partial converse. Indeed, one can show that if there exists a cut C of conductance α, then at least $|C|/2$ of its vertices will give a cut of conductance $O(\sqrt{\alpha})$).

3.2 Properties

We now prove three properties about the stationary distribution pr_α.

Proposition 2 (Uniqueness) *$\mathsf{pr}_\alpha(s)$ is unique.*

Proof We must show that Equation (1) has a unique solution. Rewrite the equation as $(I - (1 - \alpha)W)\mathsf{pr}_\alpha(s) = \alpha s$. The matrix $I - (1 - \alpha)W$ is strictly diagonally dominant[2] because the off-diagonal elements in each column add up to $1/2$, while each diagonal element is $1 - (1 - \alpha)(1/2)$. By the Gershgorin circle theorem [2], it must be nonsingular, so that the equation has a unique solution. ■

Proposition 2 allows us to extend the definition of PageRank: given any vector $s \in \mathbb{R}^n$, *not necessarily* a probability distribution over the vertices of the graph, we define $\mathsf{pr}_\alpha(s)$ as the unique solution of Equation (1).

Proposition 3 (Linearity) $\mathsf{pr}_\alpha(cv + dw) = c \cdot \mathsf{pr}_\alpha(v) + d \cdot \mathsf{pr}_\alpha(w)$.

[1] Google uses the *directed* version, because hyperlinks "go only one way".
[2] A matrix is *strictly diagonally dominant* if $a_{ii} > \sum_{j \neq i} |a_{ji}|$ for all i.

Proof By definition, the vector $x \equiv \mathsf{pr}_\alpha(cv + dw)$ satisfies the following equation

$$x = \alpha(cv + dw) + (1 - \alpha)Wx \ .$$

Let us verify that $x' \equiv c\mathsf{pr}_\alpha(v) + d\mathsf{pr}_\alpha(w)$ satisfies the same equation:

$$\begin{aligned}
\alpha(cv + dw) + (1 - \alpha)Wx' &= \alpha(cv + dw) + (1 - \alpha)W(c\mathsf{pr}_\alpha(v) + d\mathsf{pr}_\alpha(w)) \\
&= \alpha cv + (1 - \alpha)Wc\mathsf{pr}_\alpha(v) + \alpha dw + (1 - \alpha)Wd\mathsf{pr}_\alpha(w) \\
&= c\mathsf{pr}_\alpha(v) + d\mathsf{pr}_\alpha(w) \\
&= x' \ .
\end{aligned}$$

By Proposition 2, the equation has a unique solution, so that $x = x'$ and the result follows. ■

Proposition 4 (Commutativity with W) $\mathsf{pr}_\alpha(Ws) = W\mathsf{pr}_\alpha(s)$.

Proof By definition, the vector $x \equiv \mathsf{pr}_\alpha(Ws)$ satisfies the following equation

$$x = \alpha(cv + dw) + (1 - \alpha)Wx \ .$$

Let us verify that $x' \equiv W\mathsf{pr}_\alpha(s)$ satisfies the same equation:

$$\begin{aligned}
\alpha(cv + dw) + (1 - \alpha)Wx' &= \alpha Ws + (1 - \alpha)W^2\mathsf{pr}_\alpha(s) \\
&= W(\alpha s + (1 - \alpha)W\mathsf{pr}_\alpha(s)) \\
&= W\mathsf{pr}_\alpha(s) \\
&= x' \ .
\end{aligned}$$

By Proposition 2, the equation has a unique solution, so that $x = x'$ and the result follows. ■

As a corollary of Propositions 2 and 4, we deduce that $\mathsf{pr}_\alpha(s)$ is the unique solution to

$$\mathsf{pr}_\alpha(s) = \alpha s + (1 - \alpha)\mathsf{pr}_\alpha(Ws) \ . \tag{2}$$

3.3 Approximating PageRank

We would like to come up with a fast way to find an approximation to the unique solution $\mathsf{pr}_\alpha(s)$ of Equation (1). We now describe an iterative procedure that does that.

We maintain two vectors p, the *approximation vector*, and r, the *error vector*, that satisfy the following invariant

$$p = \mathsf{pr}_\alpha(s - r) \ .$$

Starting with initial values $p = 0$ and $r = s$, in each iteration, we pick a vertex u, and update the two vectors p and r to the new vectors p' and r' defined as follows:

$$\begin{aligned}
p' &= p + \alpha r(u)\chi_u \ , \\
r' &= r - r(u)\chi_u + (1 - \alpha)r(u)W\chi_u \ .
\end{aligned}$$

The vector χ_u is the *characteristic vector* of u, i.e., the vector with a 1 in the coordinate corresponding to vertex u and 0 elsewhere. Given a fixed $\epsilon > 0$, we keep iterating as long as there exists some vertex u such that $r(u) \geq \epsilon d(u)$.

First, we prove that each iteration of the algorithm preserves the invariant $p = \mathsf{pr}_\alpha(s - r)$.

Proposition 5 $p' = \mathsf{pr}_\alpha(s - r')$.

Proof By Proposition 3, it suffices to show that $p' + \mathsf{pr}_\alpha(r') = p + \mathsf{pr}_\alpha(r)$. So let us verify that:

$$\begin{aligned} p + \mathsf{pr}_\alpha(r) &= p + \mathsf{pr}_\alpha(r - r(u)\chi_u) + \mathsf{pr}_\alpha(r(u)\chi_u) \\ &= p + \mathsf{pr}_\alpha(r - r(u)\chi_u) + \alpha r(u)\chi_u + (1-\alpha)\mathsf{pr}_\alpha(Wr(u)\chi_u) \\ &= (p + \alpha r(u)\chi_u) + \mathsf{pr}_\alpha(r - r(u)\chi_u + (1-\alpha)r(u)W\chi_u) \\ &= p' + \mathsf{pr}_\alpha(r') \ . \end{aligned}$$

where the third equation resulted from an application of Equation (2). ■

Next, we prove a bound on the error vector.

Proposition 6 $||r'||_1 \leq ||r||_1 - \alpha r(u)$.

Proof Using the triangle inequality,

$$||r'||_1 = ||r - r(u)\chi_u + (1-\alpha)r(u)W\chi_u||_1 \leq ||r - r(u)\chi_u||_1 + (1-\alpha)r(u)||W\chi_u||_1 \ .$$

However, $||W\chi_u||_1 \leq 1$. Indeed, the ith element of $W\chi_u$ is $\frac{1}{2d(u)}$ when $i \neq u$ and $\frac{1}{2}$ when $i = u$. Therefore,

$$||r'||_1 \leq ||r||_1 - r(u) + (1-\alpha)r(u) = ||r||_1 - \alpha r(u) \ ,$$

as desired. ■

Finally, we prove that the iterative procedure works.

Theorem 7 *Fix $\epsilon > 0$. Suppose that in each iteration we pick a vertex u with the property that $r(u) \geq \epsilon d(u)$. Then the process terminates in $O\left(\frac{1}{\epsilon\alpha}\right)$ iterations with vectors p and r that satisfy the following properties:*

1. $\max_v \frac{r(v)}{d(v)} \leq \epsilon$.
2. $\mathsf{vol}(\mathsf{supp}(p)) \leq \frac{1}{\epsilon\alpha}$, *where* $\mathsf{supp}(p)$ *is the set of vertices for which p is nonzero and* $\mathsf{vol}(S) \equiv \sum_{x \in S} d_x$.

Proof Initially, $||r||_1 = 1$. By Proposition 6, $||r||_1$ decreases at each iteration by $\alpha r(u)$, which by assumption is at least $\alpha\epsilon d(u)$. Therefore, since the degree of each vertex is at least 1, $||r||_1$ decreases at each iteration by at least $\alpha\epsilon$. We deduce that the algorithm must terminate in at most $O\left(\frac{1}{\epsilon\alpha}\right)$ iterations.

Next, by definition, the process terminates when there are no more vertices u such that $r(u) \geq \epsilon d(u)$. Therefore, condition (1) is automatically satisfied.

Moreover, if we let T denote the number of iterations that the algorithm takes to terminate and let d_i denote the degree of the vertex picked in the ith step of the algorithm, then $\alpha\epsilon \sum_{i=1}^{T} d_i \leq 1$, so that $\sum_{i=1}^{T} d_i \leq \frac{1}{\epsilon\alpha}$. Now note that every vertex in $\mathsf{supp}(p)$ must have been picked at least once during the execution of the algorithm, so that

$$\mathsf{vol}(\mathsf{supp}(p)) \leq \sum_{i=1}^{T} d_i \leq \frac{1}{\epsilon\alpha} \ ,$$

thus showing (2), and completing the proof of the theorem. ■

The theorem we just proved gives the approximation to the PageRank vector that we need, and we finally get a *local* clustering algorithm. Note that to find a cut C we need $\epsilon = O(1/\mathsf{vol}(C))$, so that the running time of the process is proportional to $\frac{\mathsf{vol}(C)}{\alpha}$.

In order to obtain from this an almost-linear *global* partitioning algorithm, we do as follows. Let us suppose that $\phi(G)$ is $\mathsf{polylog}(n)$. If we pick a random vertex v in a cluster of vertices C with conductance ϕ^2, we will find with probability at least 0.5 a set with volume at least $\mathsf{vol}(C)/2$. However, this holds only if we use "appropriate" parameters α and ϵ, which we do not know! The fix is to binary search over the

possibilities, incurring an additional cost that is only a logarithmic multiplicative factor. In conclusion, we can find a globally optimal ϕ (up to the usual squaring error times some log factors) by cutting off chunks of the graph and repeating. The total running time is almost linear because the running time on each chunk is almost linear in its volume.

Caveat. In a random walk scheme, we need to take $1/\phi$ steps in order to get a cut of conductance $1/\sqrt{\phi}$; hence, that takes time that is about (size of chunk) $\cdot$ $\mathsf{poly}(1/\phi)$. Similarly, in a PageRank scheme, we need to take $1/\alpha$ steps in order to get a cut of conductance $1/\sqrt{\alpha}$; again, that takes time that is about (size of chunk) $\cdot$ $\mathsf{poly}(1/\phi)$. As a consequence, the algorithm will run in time that is almost linear times some $\mathsf{poly}(1/\phi)$, which is almost linear only if ϕ is at least $\mathsf{polylog}(n)$. Improving this for smaller conductances is still an open problem.

4 Intro to Sparsification

Sparsification is a technique used in dynamic graph algorithms to reduce the dependence of an algorithm's time on the number of edges in a graph. We briefly motivate this technique now, and will discuss it next time.

Suppose that we have a graph $G = (V, E)$ with $m = \Theta(n^2)$ edges. We would like to solve some cut problem (e.g., sparsest cut, min cut, s-t min cut). Most algorithms that solve these kinds of problems have running times that typically grow with m, the number of edges in the graph. As a consequence, such algorithms are much slower for dense graphs than for sparse graphs.

It would be really nice if we could somehow throw out a lot of edges from G and still get an approximate answer, because the running time of the algorithm for the resulting graph will be close to that for a sparse graph. More precisely, is there any way to "approximate" our graph G with a *sparse* graph G' that has the property that all of its cuts have more or less the same size as the original graph G?

To answer this question, next time we will introduce the idea of *randomized sampling*. It is not a spectral technique, but we will discuss spectral techniques that improve it.

References

[1] Reid Andersen, Fan Chung, and Kevin Lang. *Local Graph Partitioning using PageRank Vectors*. In FOCS '06: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pages 475–486, Washington, DC, USA, 2006. IEEE Computer Society. Full version available at http://www.math.ucsd.edu/~fan/wp/localpartfull.pdf. 8-2, 8-3

[2] Gershgorin circle theorem. http://en.wikipedia.org/wiki/Gershgorin_circle_theorem 8-3

[3] Nathan Linial and Avi Wigderson. *Expander Graphs And Their Applications*. http://www.math.ias.edu/~boaz/ExpanderCourse/ 8-1

[4] László Lovász and Miklós Simonovits. *The mixing rate of Markov chains, an isoperimetric inequality, and computing the volume*. In FOCS '90: Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science, pages 346–354, Washington, DC, USA, 1990. IEEE Computer Society. 8-1

18.409 Topics in Theoretical Computer Science: An Algorithmist's Toolkit
Fall 2009