Fall 2014 Randomized Algorithms Nov 19, 2014

Lecture 9

Prof. Friedrich Eisenbrand *Scribes: Ramazanli Ilqar*

Rapidly mixing expander random walks

Consider the simple random walk on an (n, d, c)-expander G. Since we permit multigraphs in the definition of expanders, it is necessary to generalize the definition of the random walk, as follows : at each step, the random walk proceeds along a ramdomly chosen edge among those incident on the current vertex v; thus, if there are k edges from v to w, then the probability that the random walk goes from v to w is $k/d(v)$. For an (n, d, c)-expander G, this corresponds to a Markov chain with the probability transition matrix P=A(G)/d.

Simple algebra shows that the eigenvalues of P are given by λ_i/d, and the corresponding eigenvectors remain unchanged. Notice that now all the eigenvalues lie between 1 and -1, and the gap between the first and second eigenvalue is reduced by a factor of d. A technical problem is that the random walk on such a bipartite graph results in a periodic Markov chain. We use a standard trick to get around this problem: reduce all transition probabilities by a factor of 2, and add a self-loop of probability 1/2 at each vertex. Observe that the new Markov chain still has G as its underlying graph, but the transition probability matrix $Q = (I + P)/2$ now has a stationary distribution.

Let the eigenvalues of Q be $\lambda'_1, \lambda'_2, ..., \lambda'_n$. Since the identity matrix has all its eigenvalues equal to 1, it can be verified that the eigenvalues of Q are given by

$$\lambda'_i = \frac{1+\lambda_i/d}{2}.$$

Thus, $1{=}\lambda'_1 \geq \lambda'_2 \geq ... \geq \lambda' n = 0$ and, assuming that $\lambda_2 = d - \epsilon$, we have that $\lambda'_2 = 1 - \epsilon/2d$. The eigenvectors of Q can be chosen to form an orthonormal basis since it is a symmetric matrix. In fact, the first eigenvector e'_1 is the same as that of A, i.e. $\frac{1}{\sqrt{n}}(1, 1, ..., 1)$.

We show that the Markov chain defined by Q is "rapidly mixing" in the following sense. Starting from any initial distribution, the Markov chain converges to its stationary distribution in a small number of steps. To make this notion more precise, we first define measure of convergence to the stationary distribution.

Definition 1 *Let q^t denote the state probability vector of a Markov chain defined by Q at $t \geq 0$, given any initial distribution $q^{(0)}$. Let π denote the station-*

ary distribution of Q. The relative pointwise distance (r.p.d.) of the Markov chain at time t is a measure of deviation from the limit and is defined as

$$\triangle(t) = \max_i \frac{|q^{(t)} - \pi_i|}{\pi_i}.$$

Theorem 2 *Let Q be the transition matrix of the a periodic random walk on a (n,d,c)-expander G, with $\lambda_2 \leq d - \epsilon$. Then, for any initial distribution $q^{(0)}$, the relative pointwise distance is bounded as follows :*

$$\triangle(t) \leq n^{1.5}(\lambda_2')^t \leq n^{1.5}(1 - \tfrac{\epsilon}{2d})^t.$$

Proof We know that the distribution of the Markov chain at time t is given by the following equation:

$$q^{(t)} = q^{(0)}Q^t. \tag{1}$$

Now the eigenvectors of Q are chosen to form an orthonormal basis for R^n. This implies that we can write $q^{(0)}$ as a linear combination of those vectors, as follows :

$$q^{(0)} = \sum_{i=1}^{n} c_i e_i \tag{2}$$

Combining (1) and (2), we obtain

$$q^{(t)} = \sum_{i=1}^{n} c_i e_i Q^t = \sum_{i=1}^{n} c_i (\lambda_i')^t e_i.$$

Let $\mathcal{L} \subset R^n$ be the vector space spanned by the first eigenvector e_1. This space contains all scalar multiples of the all-ones vector; the orthogonal space $\mathcal{L}^\perp$ contains all linear combinations of the remaining n-1 eigenvectors. Then $q^{(0)} = x + y$ for some $x \in \mathcal{L}$ and $y \in \mathcal{L}^\perp$; in fact, $x = c_1 e_1$ and $y = \sum_{i=2}^{n} c_i e_i$. Since x and y are orthogonal, the Pythagoras Inequality implies that $\|x\| \leq \left\|q^{(0)}\right\|$ and $\|y\| \leq \left\|q^{(0)}\right\|$.

Since $\lambda_1' = 1$, $xQ = x$ and we can write

$$q^{(t)} = q^{(0)}Q^t = (x + y)Q^t = x + \sum_{i=2}^{n} c_i (\lambda_i')^t e_i.$$

We now obtain the following bounds on the L_1-norm of $q^{(t)} - x$.

$$\left\|q^{(t)} - x\right\|_1 \leq \sqrt{n} \left\|q^{(t)} - x\right\|$$

$$= \sqrt{n} \left\| \sum_{i=2}^{n} c_i (\lambda_i)^t e_i \right\|$$

$$= \sqrt{n}\sqrt{\sum_{i=2}^{n} c_i^2(\lambda_2)^{2t}}$$

$$= \sqrt{n}\sqrt{\sum_{i=2}^{n} c_i^2(\lambda_2')^{2t}} \qquad (3)$$

$$= \sqrt{n}(\lambda_2')^t\sqrt{\sum_{i=2}^{n} c_i^2}$$

$$\leq \sqrt{n}(\lambda_2')^t \|y\| \qquad (4)$$

$$\leq \sqrt{n}(\lambda_2')^t \left\|q^{(0)}\right\| \qquad (5)$$

The inequality (3) relies on the fact that λ_2' has the second largest absolute value; the inequality (4) follows from the fact that $y = \sum_{i=2}^{n} c_i e_i$; the inequality (5) is a consequence of the Pythagoras Inequality. Since $q^{(0)}$ is a probability distribution, its components are all non-negative and sum to 1; thus, by Proposition $\left\|q^{(0)}\right\| \leq \left\|q^{(0)}\right\|_1 = 1$. We obtain that

$$\left\|q^{(t)} - x\right\|_1 \leq \sqrt{n}(\lambda_2')^t.$$

It can be proven that for any doubly stochastic matrix, the stationary distribution π must be uniform. Since $\lambda_2' < 1$, we know that as t increases, $\left\|q^{(t)} - x\right\|$ goes to 0 and $q^{(t)}$ converges to x. We conclude that $x = \pi$, and that

$$\left\|q^{(t)} - \pi\right\|_1 \leq \sqrt{n}(\lambda_2')^t.$$

The relative pointwise distance can now be bounded as follows.

$$\triangle(t) = \max_i \frac{|q_i^t - \pi_i|}{\pi_i}$$

$$= n \text{ x } \max_i |q_i^t - \pi_i|$$

$$\leq n \text{ x } \left\|q^{(t)} - \pi\right\|_1$$

$$\leq n \text{ x } \sqrt{n}(\lambda_2')^t$$

$$= n^{1.5}(\lambda_2')^t. \qquad \blacksquare$$

Approximate Counting, $\#P - complete$ problems

In this section we apply randomization to hard counting problems. After defining the class $\#P$, we present a $\#P$-complete problem. We present a (randomized) polynomial time approximation scheme for the problem of counting

the number if satisfying truth assignments for a DNF formula.

We say that a decision problem Π is in NP if for any YES-instance I of Π, there exists a proof that I is a YES-instance that can be verified in polynomial time. The class of counting problems associated with NP decision problems is denoted by $\#P$. Intuitively, the class $\#P$ consists of all counting problems associated with the decision problems in NP. We say that Π is $\#P - complete$ if for any problem Π' in $\#P, \Pi'$ can be reduced to Π by a polynomial time Turing machine.

While there are "easy" problems in $\#P$ such as counting spanning trees (where polynomial time algorithms are known), a large number of such counting problems to be intractable. Quite clearly, a $\#P - complete$ problem can be solved in polynomial time only if $P = NP$, implying that it is quite unlikely that we can efficiently solve such problems. Unfortunately, we do not know of a good deterministic approximation algorithms for any $\#P-$complete problem. However, the situation changes appreciably if we permit ourselves the use of randomization in the approximation algorithm,

Definition 3 *A polynomial approximation scheme (PAS) for a counting problem* Π *is a deterministic algorithm* $\mathcal{A}$ *that takes an input instance I and a real number* $\epsilon > 0$*, and in time polynomial in* $n = |I|$ *produces an output A(I) such that*

$$(1 - \epsilon)\#(I) \leq A(I) \leq (1 + \epsilon)\#(I).$$

A fully polynomial approximation scheme (FPAS) is a polynomial approximation scheme whose running time is polynomial bounded in both n *and* $1/\epsilon$*.*

Definition 4 *An* (ϵ, δ) *approximation scheme for a counting problem* Π *is a fully polynomial randomized approximation scheme that takes an input instance I and computes an* ϵ*-approximation to* $\#(I)$ *with probability at least* $1 - \delta$ *in time polynomial in* $n, 1/\epsilon, and log 1/\delta$*.*

Approximate counting is an area in which randomization makes a dramatic difference in our ability to (approximately) solve problems.

The DNF Counting Problem

Let $F(X_1, X_2, ..., X_n)$ be a Boolean formula in disjunctive normal form (DNF) over the n Boolean variables $X_1, X_2, ..., X_n$. In other words, F is disjunction $C_1 \vee C_2 .... \vee C_m$ of clauses C_i, where each clause C_i is a conjunction $L_1 \wedge ... \wedge L_{r_i}$ of r_i literals. Each literal L_j is either a variable X_k or its negation $\overline{X_k}$. We may assume that each variable at most once in any given clause.

A truth assignment $a = (a_1, a_2, ..., a_n)$ is an assignment of value a_i to the variable X_i for each i. A truth assignment **a** is said to satisfy F if $F(a_1, a_2, ..., a_n)$ evaluates tp 1. We denote by $\#F$ the number of distinct satisfying assignments of a given formula F. Clearly $0 < \#F \leq 2^n$.

The DNF counting problem is to compute the value of $\#F$. This problem is known to be $\#P-complete$ and hence it is unlikely to have an exact polynomial time algorithm. We describe (ϵ, δ) scheme for this problem. The input size is at most nm. We desire that the approximation scheme have a running time that is polynomial in $n, m, 1/\epsilon$ and $\log 1/\delta$.

Let $U = \{0,1\}^n$ and S be set of possible values for $(X_1, X_2, ..., X_n)$ which makes boolean function-F TRUE. Our aim is to find approximate value of $|S|$ (in fact $|S|=\#F$). An obvious randomized approach to estimate $|S|$ is to use the Monte Carlo method. This involves choosing N independent samples from U, say $u_1, u_2, ..., u_N$ and using the value of F on these samples to estimate the probability that a random choice will lie in S. More formally define the random variables $Y_1, Y_2, ..., Y_N$ as $Y_i = 1$ if $F(u_i) = 1$ and $Y_i = 0$ otherwise.

By this definition $Y_i = 1$ if and only if $u_i \in S$. Finally, define the estimator random variable

$$Z = |U| \sum_{i=1}^{n} \frac{Y_i}{N}$$

It is easy to verify that $E[Z] = |S|$.

Theorem 5 *Let $\rho = |S|/|U|$. Then the Monte Carlo method yields an ϵ-approximation to $|S|$ with probability at least $1 - \delta$ provided*

$$N \geq \frac{4}{\epsilon^2 \rho} ln \frac{2}{\delta}.$$

Proof Fix some $\epsilon \in (0,1]$ and $\delta \in (0,1]$. Notice that the random variables Y_i have the Bernoulli distribution with parameter ρ. Define $Y = \sum_{i=1}^{N} Y_i$, and observe that this has the binomial distribution with parameters N and ρ. Moreover, the estimator Z=|U|Y/N. By straightforward application of the Chernoff bound we obtain that

$$Pr[(1-\epsilon)|S| \leq Z \leq (1+\epsilon)|S|] =$$

$$= Pr[(1-\epsilon)N\rho \leq Y \leq (1+\epsilon)N\rho] \geq$$

$$\geq 1 - F^+(Np, \epsilon) - F^-(NP, \epsilon) \geq$$

$$\geq 1 - 2e^{-N\rho\epsilon^2/4} \geq$$
$$\geq 1-\delta.$$ ■

The Coverage Algorithm

Now, we can do an approximation scheme for given (ϵ, δ) pairs. And we want to reduce size of sample space so as to ensure that the ratio ρ is relatively large, while ensuring that the set S is still completely represented. Then, we will approach to the problem with a new idea that called the coverage algorithm.

Now, we have $F = C_1 \vee C_2 \vee ... \vee C_m$ and each of C_i has set of satisfying set is S_i. We can find each of S_i in polynomial time and our aim to find the set $S = S_1 \cup S_2 \cup ... \cup S_m$. The brute force approach to compute S is inefficient when S_i's have the large cardinality. The inclusion-exclusion formula is extremely inefficient for large m, since it requires computing roughly 2^m terms. As a solution we will suggest define the multiset union $T = S_1 \uplus S_2 ... \uplus S_m$. We adopt the convention that the elements of T are ordered pairs of the form (v, i) corresponds to $v \in S_i$.

Observe that $|T| = \sum_{j=1}^{m} |S_i| \geq |S|$.

Definition 6 *For all $v \in S$ the coverage set is defined by :*

$$cov(v) = \{(v, i) | (v, i) \in U\}$$

The size of the coverage set is exactly the number of $S_i's$ containing v, or the multiplicity of v in the multiset version of T. (In the DNF problem, for a truth assignment **a**, the set cov(a) is the set of clauses satisfied by a). The following observations are immediate :

1. The number of coverage sets is exactly $|S|$, and these coverage sets are easy to compute

2. The coverage sets partition T, i.e., $T = \bigcup_{v \in S} cov(v)$
3. $|T|$ is easily computed as $|T| = \sum_{v \in S} |cov(v)|$
4. For all $v \in S$ $|cov(v)| \leq m$

Definition 7 *The function $f : T \to \{0, 1\}$ is defined as follows*

$$f((v, i)) = \begin{cases} 1 & i = min\{j | v \in H_j\} \\ 0 & otherwise \end{cases}$$

We will also define the set G as the inverse image of 1 under f

$$G = \{(v, j) \in T | f((v, j)) = 1\}.$$

Crucial observation here is that there exists a one-to-one corresponding between elements of G and S. So, this implies $|G| = |S|$.

Lemma 8 *In the union of sets problem,*

$$\rho = \frac{|G|}{|T|} \geq \frac{1}{m}$$

Proof The proof relies on the observations made above.

$$\begin{aligned} |T| &= \sum_{v \in S} |cov(v)| \leq \\ &\leq \sum_{v \in S} m \leq \\ &\leq m|S| = m|G| \end{aligned}$$

The lemma follows ■

The following theorem shows that the Monte Carlo sampling technique gives as (ϵ, δ) scheme for $|G|$ and hence also for $|S|$.

Theorem 9 *The Monte Carlo method yields and ϵ-approximation to $|G|$ with probability at least $1 - \delta$ provided*

$$N \geq \frac{4m}{\epsilon^2} ln(\frac{2}{\delta}).$$

The running time is polynomial in N

Proof The sampling procedure and analysis are exactly as in the prvious theorem. We merely have to show that f can be computed in polynomial time and that it is possible to sample uniformly from T.

To compute $f((v, i))$ we check whether the truth assignment v satisfies C_i but none of the clauses C_j for $j < i$. Sampling an element (v, i) uniformly from T is performed in 2 stages. First, choose i such that $1 \leq i \leq m$ and

$$Pr[i] = \frac{S_i}{T} = \frac{|S_i|}{\sum\limits_{i=1}^{m} |S_i|}.$$

Then an element $v \in S_i$ is chosen uniformly at random. It is easy to verify that the resulting pair (v, i) is uniform over T ■

Notice that the lemma implies a polynomial bound on the running time.

So, with the help of coverage algorithm, we can estimate S with smaller sample which will makes algorithm run more faster.

Approximating the Permanent

We turn to the problem of counting the number of perfect matchings in a bipartite graph. The input to this problem is a bipartite graph $G(U, V, E)$ with independent sets of vertices $U = \{u_1, u_2, ..., u_n\}$ and $V = \{v_1, v_2, ..., v_n\}$. The problem of counting the number of perfect matchings in a given bipartite graph is $\#P - complete$.

Definition 10 *Let $Q = (Q_{ij})$ be an nxn matrix. The permanent of the matrix is defined as*

$$per(Q) = \sum_{\pi \in S_n} \prod_{i=1}^{n} Q_{i,\pi(i)}$$

where S_n is the symmetric group of permutations of size n.

One interesting fact about permanents is that if G is a bipartite graph then $per(A(G))$ is equal to number of perfect matchings. Thus computing the permanent of a 0-1 matrix is $\#P - complete$.