COMS 6998: Randomness in Computing — Nov 19, 2012

Expander Graphs & The Monte Carlo Method

Instructor: *Xi Chen* — Scribe: *Raanan Cohen*

1 Review

Definition 1. *An (n, d, c)-expander is a d-regular bipartite multigraph $G(X, Y, E)$ with $|X| = |Y| = n/2$ such that for any $S \subseteq X$, $|\Delta(S)| \geq (1 + c(1 - 2|S|/n))|S|$*

Definition 2. *For some (n, d, c)-expander, G. Let $P = Adj/d$ be the probability matrix, and $P_{i,j} = Adj_{i,j}/d$. Let $Q = (I + P)/2$.*

Observation 3. *Q is doubly stochastic and has the uniform distribution as its stationary distribution.*

We saw in the previous lecture that the state probability vector will approach the uniform distribution with error decreasing at an expoonential rate on the number of steps taken in a random walk.

We will return to random walks later, at which point (n, d, c)-expander graphs will be used for probability amplification.

2 Probability amplification

The goal is to amplify the probability to be exponentially small in the number of trials, while using a small number of random bits, which can be very expensive to generate.

2.1 Introduction

Naive approach

Modified (and equivalent) definition of **BBP** that has a lower probability of failure:

Definition 4. BBP *is the class of languages L that have a poly time algorithm A that for any string x, and a long random string r,*

- $x \in L \Rightarrow Pr[A(x, r) rejects] \leq 1/100$
- $x \notin L \Rightarrow Pr[A(x, r) accepts] \leq 1/100$

Observation 5. *By repeating this algorithm k times with random strings $r_1 ... r_k$, the probability that the majority of the outcomes is incorrect is $1/2^{\Omega(k)}$ (proved this with Chernoff bound on a problem set). This uses $k * r$ random bits.*

The goal of this probability amplification exercise is to get a similar exponentially small error, while using many fewer bits. Our target is to use $O(k) + n$ random bits.

Walks on expander graphs

The general idea is to construct an expander graph with N vertices, where $N = 2^n$. Each vertex is associated with a unique bit-string from $\{0,1\}^n$. After some number of random steps, the probability vector is almost uniform, and we will be able to generate length n random bit strings by taking a small number of random steps and returning the label of the current vertex.

Definition 6. *A (d,c)-family of expanders is a sequence of (n,d,c)-expander with $n = 1...\infty$.*

Gabber-Galil expanders

Gabber and Galil gave an explicit construction for creating a (n,d,c)-expander, where $n = 2m^2$ for some $m \in \mathbb{Z}$.

In this construction, $|X| = |Y| = m^2$. Assign each vertiex in X a unique label (x,y), such that $x, y \in \mathbb{Z}_m$. Do the same for Y. Since this is an expander family with degree $d = 5$, for each vertex $v = (x,y) \in X$ add an edge from v to

1. $(x,y) \in Y$
2. $(x, x+y) \in Y$
3. $(x, x+y+1) \in Y$
4. $(x+y, y) \in Y$
5. $(x+y+1, y) \in Y$

Note that all of the additions above are mod n. This graph is 5-regular because each of elements in the above list define a permulation on X, which is a perfect matching from X to Y.

The expansion of this graph can be proven to be $c = (2-\sqrt{3})/4$.

2.2 Walks on expander graphs

Definition 7. *Let G be an (N,d,c)-expander, where $N = 2^n$ and G is created with the Gabber-Galil construction. Let A be the adjacency matrix of this graph. Its eigenvalues are: $5 = d = \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_N = -d = -5$. The eigenvectors are $e_1, e_2, ..., e_N$, and form an orthonormal basis. Note that there is a constant gap between λ_1 and λ_2 (shown in the previous lecture).*

Definition 8. *$P = Adj/d$. Self loops are added to make the Markov process non-periodic, so $Q = (I+P)/2$ is the new probability matrix. The i^{th} eigenvalue of Q is $\lambda_i^{`} = (1+\lambda_i/d)/2$. Therefore $1 = \lambda_1^{`} \geq \lambda_2^{`} \geq ... \geq \lambda_N^{`} = 0$. The eigenvectors $e_1^{`}, e_2^{`}, ..., e_N^{`}$ form an orthonormal basis.*

Select a constant β such that $\lambda_2 \beta < 1/100$. β is a number of fresh bits we will need.

Probability amplification walk on (n, d, c)-expander

Definition 9. *Algorithm:*

Select a vertex p uniformly at random from the set of N vertices (using n random bits);
for $i \leftarrow 1$ **to** k **do**
walk β steps;
v_i = current vertex;
results[i] = $A(x, labelOf(v_i))$;
if *count1's*$(results) > k/2$ **then**
return 1; //assume that k is odd, so there are no ties
end
return 0;
end

Observation 10. *This algorithm runs in polynomial time. The graph has exponential size, but there is no need to explicitly construct the graph, because given a vertex, its neighbors can be computed in polynomial time.*

Each step of the random walk selects from 6 choices, only a constant number of random bits are needed per step. Therefore, the total number of bits used by this algorithm is $n + O(\beta k) = n + O(k)$.

Work with BBP as a class with witnesses.

Definition 11. *Let $W = \{r \in \{0,1\}^n | A(x,r) = 1\}$ and $\overline{W} = \{r \in \{0,1\}^n | A(x,r) = 0\}$ From the definition of BPP above, $|W| = 99N/100$ and $|\overline{W}| = N/100$.*

Definition 12. *Let $p^{(i)}$ be the distribution for the i^{th} step (which is also r_i).*
Let M be a diagonal matrix, where $M_{i,i} = 1$ if $i \in W$ and $M_{i,i} = 0$ otherwise.
Let $\overline{M}$ be a diagonal matrix where $\overline{M}_{i,i} = 1$ if $i \in \overline{W}$ and $m_{i,i} = 0$ otherwise. $\overline{M} = I - M$.

Since this is a Markov process, $p^{(i)} = p^{(0)}B^i$.
$||p^{(i)}M||_1 = \Pr[r_i$ is a witness].
$||p^{(i)}\overline{M}||_1 = \Pr[r_i$ is not a witness].
Let $B = Q^\beta$.
$||p^{(0)}BM||_1 = \Pr$[first sample is a witness]. because B is encapsulates β steps.
Select $S = (S_1, ...S_k)$ where $S_i \in \{M, \overline{M}\}$, such that $s_i = m$ iff $r_i \in W$ and $s_i = \overline{M}$ iff $r_i \notin W$.

Lemma 13. *$Pr[S$ occurs$] = ||p^{(0)}(BS_1)(BS_2)...(BS_{k-1})(BS_k)||_1$.*

Lemma 14. *for any distribution p:*

1. $||pBM|| \leq ||p||$
2. $||pB\overline{M}|| \leq (1/5)||p||$

Claim 15. *The probability of error for this algorithm is $\approx 1/2^k$.*

Proof. The algorithm fails when more than half of the samples fail. Let κ be the number of S_i terms in S, not equal to $\overline{M}$. The algorithm fails when $\kappa > k/2$.
$Pr[S$ occurs$] = ||p^{(0)}(BS_1)(BS_2)...(BS_{k-1})(BS_k)||_1$.

$\leq \sqrt{N}||p^{(0)}(BS_1)(BS_2)...(BS_{k-1})(BS_k)||$
$\leq \sqrt{N}(1/5)^{\kappa}||p^{(0)}||$ - this is by repeated application of Lemma 14.
$\leq \sqrt{N}(1/5)^{k/2}||p^{(0)}||$.

Now we estimate how many events S there are, such that the algorithm fails. An overestimate on this number is 2^k. Thus, Pr[majority of samples fail] $\leq 2^k\sqrt{N}(1/5)^{k/2}||p^{(0)}||$. Since $p^{(0)}$ is selected from the uniform distribution, its norm is $1/\sqrt{N}$.

Therefore Pr[algorithm fails]$\leq 2^k/5^{k/2}$. □

Proof of Lemma 14

1.

Express p as a linear combination of eigenvectors, $p = \sum_{i=1}^{N} c_i e_i$.
$pB = \sum_{i=1}^{N} c_i e_i B = \sum_{i=1}^{N} c_i \lambda_i^{\beta} e_i$, where λ_i^{β} is in the interval $[0, 1]$.
Since W is a diagonal 0-1 matrix, it can only reduce the norm of a vector that it is applied to. Therefore $||pBW|| \leq ||pB||$.
Rewrite as $||\sum_{i=1}^{N} c_i \lambda_i^{\beta} e_i|| = \sqrt{\sum_{i=1}^{N} c_i^2 \lambda_i^{2\beta}} \leq \sqrt{\sum_{i=1}^{N} c_i^2}$. this last step is because the eigenvalue is between 0 and 1.
This last term, $\sqrt{\sum_{i=1}^{N} c_i^2} = ||p||$ □

2.

Decompose p into x and y components: $p = \sum_{i=1}^{N} c_i e_i = x + y = c_1 e_1 + \sum_{i=2}^{N} c_i e_i$.
By the Pythagorean Inequality, $||x|| \leq ||p||$ and $||y|| \leq ||p||$.

Focusing on x,

since $\lambda_1 = 1$, $xB = x\lambda_1 c_1 = x$.

$||x\overline{W}|| \leq ||x||/10$, beacuse $\overline{W}$ will zero out at least 99/100 of the entries. which will reduce the number of nonzero enties to at most 1/100 of the original amount, which will reduce the norm by at least $\sqrt{1/100} = 1/10$.

Focusing on y,

$yB = \sum_{i=2}^{N} c_i e_i B = \sum_{i=2}^{N} c_i \lambda_i^{\beta} e_i$.

$||yB\overline{W}|| \leq ||yB||$ because the diagonal matrix can only zero out elements.

Since $\lambda_2^{\beta} \leq 1/10$ is the second largest eigenvalue,

$||yB\overline{W}|| \leq \sqrt{\sum_{i=2}^{N} c_i^2 \lambda_i^{2\beta}} \leq \lambda_2^{\beta}\sqrt{\sum_{i=2}^{N} c_i^2} \leq ||y||/10$.

Combining the parts with triangle inequality: $||pB\overline{W}|| \leq ||xB\overline{W}|| + ||yB\overline{W}|| \leq (1/10)(||x|| + ||y||)$.
Since $||x|| \leq ||p||$ and $||y|| \leq ||p||$, $(1/10)(||x|| + ||y||) < 2||p||/10 = ||p||/5$ □

3 Monte Carlo method

Monte carlo method, (and later, the Markov Chain Monte Carlo) estimates values with random sampling.

Start with a simple example for estimating π and move onto to try estimating the number of satisfying assignments of a DNF formula, which will raise some problems with the naive sampling used in the first example.

Approximating π

Example 16. *Approximating π.*

Use a unit circle with center at the origin embedded in a square with sides of length 2.

Sample (x, y) pairs, n times, such that x, y are in the interval $[-1, 1]$ and are chosen uniformly at random.

Define indicator random variable Z_i as follows:

- *$Z_i = 1$ if $x_i^2 + y_i^2 \leq 1$ which is the case that the i^{th} sample falls in the circle.*
- *$Z_i = 0$ in all other cases.*

$Pr[Z_i = 1]$ = area of circle/area of square $= \pi/4$.

Let $Z = \sum_{i=1}^{n} Z_i$.

$E[Z] = \sum_{i=1}^{n} E[Z_i] = n\pi/4$.

Let $Z' = 4Z/n$.

$E[Z'] = \pi$.

Applying Chernoff bound:

$Pr(|Z' - \pi| \geq \epsilon\pi)$

$= Pr(|Z - (n\pi)/4| \geq (n\epsilon\pi)/4)$

$= Pr(|Z - E[Z]| \geq \epsilon E[Z])$

$\leq 2e^{-n\pi\epsilon^2/12}$

If we want to approximate within a certain error probability (δ) simple algebraic manipulation yields: $2e^{-n\pi\epsilon^2/12} \leq \delta \rightarrow n \geq 12ln(2/\delta)/(\pi\epsilon^2)$

The example above is quite natural and efficient. Note the ratio of events that cause Z_i to succeed and fail: $\pi/4$.

Claim 17. *In general, there must be a dense space to run a polynomial time Monte Carlo simulation, or else interesting objects will most likely not be sampled. The ratio should be $1/p(n)$ for some polynomial $p(n)$.*

#DNF

Definition 18. **#P** *is the class of counting problems for langauges whose decision problems are in* **NP**. *Given an instance of a problem from* **NP** *the analogious* **#P** *solution is the number of solutions that satisfy the NP decision problem. It is easy to see that* **#P** *is at least as hard to solve as the analogoius NP problem, because if there is a known* **#P** *algorithm, the NP algorithm could run the* **#P** *algorithm and return true if the result was not 0.*

Definition 19. **#DNF***: Count the number of satisfying assignments for a given boolean formula which is represented in DNF.*

DNF is the disjuntion (or) of conjuctive (and) clauses. for example: $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_4) \vee (x_1)$ is in DNF.

Observation 20. *Note that the decision problem is easy. Check if any of the clauses are true. The counting problem is harder, and #**P** complete. The formula can be negated and by De Morgan's law, the resulting formula will be in CNF. The relationship between these two counting problems is $a = 2^n - b$, where a is answer to #**DNF**, b is answer to #**CNF**, and n is the number of variables in the formula. Thus, if there an algorithm for the #**DNF** problem, is can be used to compute the answer to #**CNF**. Since #**CNF** is #**P** complete, so is this problem. Therefore, a polynomial algorithm can only exist if* **P** = **NP**.

Since it is unlikely that **P** = **NP**, let us try to find an approximation algorithm that runs on polynomial time.

Approximating #DNF

Try the naive approach, which worked for estimating π.

Example 21. *Let F be a formula in DNF.*
Randomly assign true/false to each of the n variables in F.
Define random variable X_i as follows:

- $X_i = 1$ *if the i^{th} assignment is satisfying.*
- $X_i = 0$ *in all other cases.*

Let $c(F)$ be the number of satisfying assignments.
$Pr[X_i = 1] = c(F)/2^n$.
$E[X_1 + ... + X_m] = mc(F)/2^n$.
Applying a Chernoff bound and algebra we derive that the number of trials for desired error prob is:
$m \geq (3 * 2^n ln(2/\delta))/(\epsilon^2 c(F))$.
Therefore, if $c(F) \approx 2^n$, then only a polynomial number of steps is needed. If $c(F)$ is much smaller than 2^n, then m is exponential.

Observation 22. *When the space we are sampling is dense with "intersting" events, Monte Carlo method takes polynomial samples, else it takes exponential samples.*

Definition 23. *The solution to sparse samples space are Densifiers. The goal is to define a set of assignments S such that:*

1. *S contains all of the satisfying assignments.*
2. *S is not too big compared to $c(F)$, this should be bounded by an inverse polynomial.*
3. *Can uniformly randomly sample S.*

Definitions for next lecture

The lecture closed with some definitions that will be helpful for the next week:

$C_1 ... C_t$ are clauses.

For each i, let SC_i be the set of assignments that satisfy C_i.

If C_i has l literals, then there are 2^{n-l} satisfiying assignments, because the clause only has one satisfying assignment with its variables, and all other variables are free.

$c(F) = \cup_{i=1}^{t} SC_i$

Let $S = (i, a) : 1 \leq i \leq t$ and SC_i

$|S| = \sum_{i=1}^{t} |SC_i|$.

Let $i(a)$ be the smallest i, such that $a \in SC_i$.

$S\text{`} = \{(i(a), a) : \forall$ satisfying assignment$\}$.

$|S\text{`}| = c(F)$.

Next lecture will show how to sample this set in polynomial time for an accurate estimate.