COS 594D: A THEORIST'S TOOLKIT

March 2, 2003

Abstract

These are edited lecture notes from a graduate course at the Computer Science department of Princeton University in Fall 2002. The course was my attempt to teach first year graduate students students many mathematical tools useful in theoretical computer science. Of course, the goal was too ambitious for a course with 12 three hour lectures. I had to relegate some topics to homework; these include online algorithms, Yao's lemma as a way to lowerbound randomized complexity, Madhu Sudan's list decoding algorithm (useful recently in complexity theory and pseudorandomness), and pseudorandom properties of expander graphs. If I had time for another lecture I would have covered basic information theory.

To put the choice of topics in context, I should mention that our theory grads take a two semester course sequence on *Advanced Algorithm Design* and *Complexity Theory* during their first year, and I did not wish to duplicate topics covered in them. Inevitably, the choice of topics —especially the final two chapters— also reflected my own current research interests.

The scribe notes were written by students, and I have attempted to edit them. For this techreport I decided to reshuffle material for coherence, and so the scribe names given with each chapter does not completely reflect who wrote that chapter. So I will list all the scribes here and thank them for their help: Tony Wirth, Satyen Kale, Miroslav Dudik, Paul Chang, Elad Hazan, Elena Nabieva, Nir Ailon, Renato F. Werneck, Loukas Georgiadis, Manoj M.P., and Edith Elkind. I hope the course was as much fun for them as it was for me.

Sanjeev Arora
March 2003

Contents

1 Probabilistic Arguments 2

2 LP Duality and its Uses 8

3 The Dimension Method 16

4 The Lore and Lure of Expanders 25

5 Eigenvalues and Expanders 29

6 Markov Chains and Random Walks 36

7 High Dimensional Geometry 43

8 Discrete Fourier Transform and its Uses 57

9 Relaxations for NP-hard Optimization Problems 66

10 Semidefinite Programming 74

Chapter 1

Probabilistic Arguments

SCRIBE: *Tony Wirth*

1.1 Introduction

Think of the topics in this course as a toolbox to rely upon during your research career. Today's lecture shows the application of simple probabilistic arguments to prove seemingly difficult results. Alon and Spencer's text goes into considerably more detail on this topic, as will Sudakov's course in the Math dept.

A random variable is a mapping from a probability space to **R**. To give an example, the probability space could be that of all possible outcomes of n tosses of a fair coin, and X_i is the random variable that is 1 if the ith toss is a head, and is 0 otherwise.

Let $X_1, X_2, X_3, \ldots, X_n$ be a sequence of random variables. The first observation we make is that of the **Linearity of Expectation**, viz.

$$\mathbf{E}[\sum_i X_i] = \sum_i \mathbf{E}[X_i]$$

It is important to realize that linearity holds *regardless* of the whether or not the random variables are independent.

Can we say something about $\mathbf{E}[X_1 X_2]$? In general, nothing much but if X_1, X_2 are independent events (formally, this means that for all a, b $\mathbf{Pr}[X_1 = a, X_2 = b] = \mathbf{Pr}[X_1 = a]\mathbf{Pr}[X_2 = b]$) then $\mathbf{E}[X_1 X_2] = \mathbf{E}[X_1]\mathbf{E}[X_2]$.

The first of a number of inequalities presented today, **Markov's inequality** says that any *non-negative* random variable X satisfies

$$\mathbf{Pr}\left(X \geq k\mathbf{E}[X]\right) \leq \frac{1}{k}.$$

Note that this is just another way to write the trivial observation that $\mathbf{E}[X] \geq k \cdot \mathbf{Pr}[X \geq k]$.

Sometimes we refer to the application of Markov's inequality as an *averaging argument*.

Can we give any meaningful upperbound on $\mathbf{Pr}[X < c \cdot \mathbf{E}[X]]$ where $c < 1$, in other words the probability that X is a lot less than its expectation? In general we cannot. However, if we know an upperbound on X then we can. For example, if $X \in [0, 1]$ and $\mathbf{E}[X] = \mu$ then for any $c < 1$ we have (simple exercise)

$$\mathbf{Pr}[X \leq c\mu] \leq \frac{1 - \mu}{1 - c\mu}.$$

Sometimes this is also called an averaging argument.

EXAMPLE 1 Suppose you took a lot of exams, each scored from 1 to 100. If your average score was 90 then in at least half the exams you scored at least 80.

1.2 Independent sets in random graphs

We illustrate the power of the averaging argument by studying the size of the largest independent set in random graphs. Recall that an independent set is a collection of nodes between every pair of which there is no edge—an anti-clique, if you like. It is **NP**-hard to determine the size of the largest independent set in a graph in general.

Let $G(n, \frac{1}{2})$ stand for the distribution on graphs with n vertices in which the probability that each edge is included in the graph is $1/2$. In fact, $G(n, \frac{1}{2})$ is the uniform distribution on graphs with n nodes (verify this!). Thus one of the reasons for studying random graphs is to study the performance of some algorithms "on the average graph."

What is the size of the largest independent set in a random graph?

THEOREM 1
The probability that a graph drawn from $G(n, \frac{1}{2})$ has an independent set of size greater than $2\lceil \log n \rceil$ is tiny.

PROOF: For all subsets S of $\{1, 2, \ldots, n\}$, let X_S be an indicator random variable for S being an independent set. Now, let r.v. Y_k be the number of independent sets of size k; that is,

$$Y_k = \sum_{S:|S|=k} X_S.$$

Linearity of expectation tells us that

$$\mathbf{E}[Y_k] = \sum_{S:|S|=k} \mathbf{E}[X_S].$$

The X_S r.v.s are indicators and there are $\binom{|S|}{2}$ potential edges between $|S|$ vertices, hence

$$\mathbf{E}[X_S] = \mathbf{Pr}[X_S = 1] = \frac{1}{2^{\binom{|S|}{2}}},$$

Applying both the approximations

$$\binom{n}{k} \approx \left(\frac{ne}{k}\right)^k \quad \text{and} \quad \binom{n}{2} \approx \frac{n^2}{2}, \tag{1.1}$$

we can show that

$$\begin{aligned} \mathbf{E}[Y_k] &= \binom{n}{k} \frac{1}{2^{\binom{|S|}{2}}} \\ &= \left[\frac{ne}{k}\right]^k \frac{1}{2^{k^2/2}} \\ &= \left[\frac{ne}{k2^{k/2}}\right]^k \\ &= \left[\frac{e}{2\log n}\right]^{2\log n}, \end{aligned} \tag{1.2}$$

substituting $2\log n$ for k. Hence the mean value of Y_k tends to zero very rapidly.

In particular, Markov's inequality tells us that $\mathbf{Pr}[Y_k \geq 1] \leq \mathbf{E}[Y_k]$ and hence that the probability of an independent set of size $2\lceil \log n \rceil$ is *tiny*. □

Since the distribution $G(n, 1/2)$ picks each edge with probability $1/2$, it also leaves out each edge with probability $1/2$. Thus we can simultaneously bound the maximum size of a clique by $2\lceil \log n \rceil$.

Now we also show a lowerbound on the size of the largest independent set/clique. Note that substituting $k = 2\lceil \log n \rceil - \sqrt{\log n}$ in the above calculation for $\mathbf{E}[Y_k]$ shows that the expected number of independent sets of size $2\lceil \log n \rceil - \sqrt{\log n}$ is very large. However, we have to rule out the possibility that this number is large with probability ≈ 0.

For this we need a more powerful inequality, **Chebyshev's inequality**, which says

$$\mathbf{Pr}[|X - \mu| \geq k\sigma] \leq \frac{1}{k^2},$$

where μ and σ^2 are the mean and variance of X. Recall that $\sigma^2 = \mathbf{E}[(X-\mu)^2] = \mathbf{E}[X^2] - \mu^2$. Actually, Chebyshev's inequality is just a special case of Markov's inequality: by definition,

$$\mathbf{E}\left[|X - \mu|^2\right] = \sigma^2,$$

and so,

$$\mathbf{Pr}\left[|X - \mu|^2 \geq k^2\sigma^2\right] \leq \frac{1}{k^2}.$$

THEOREM 2
If G is drawn from $G(n, \frac{1}{2})$ there is almost surely *an independent set of size* $2\lceil \log n \rceil - \sqrt{\log n}$.

PROOF: If $k = 2\log n - \sqrt{\log n}$, then by substituting into formula (1.2) we find that $\mathbf{E}[Y_k]$ is approximately

$$\left(\frac{ne}{2\log n \frac{n}{2^{\sqrt{\log n}/2}}}\right)^k \geq \left(\frac{2^{\sqrt{\log n}/2}}{2\log n}\right)^{\log n}.$$

This quantity tends to infinity, so the mean is *much* greater than one. However, the variance could be so large that the value of the mean tells us nothing.

The calculation of the variance is however quite messy. Let N stand for $\binom{n}{k}$ and p stand for $2^{-\binom{k}{2}}$. The expected value of Y_k is Np. We show that $\mathbf{E}[Y_k^2]$ is $N^2p^2 + \varepsilon$, where $\varepsilon \ll N^2p^2$, so that $\mathbf{Var}[Y_k] = \varepsilon$ is smaller than N^2p^2. Then concentration about the mean follows from Chebyshev's inequality. Observe that

$$\mathbf{E}[Y_k^2] = \mathbf{E}[(\sum_{S:|S|=k} X_s)^2] = \sum_{S,T:|S|=|T|=k} \mathbf{E}[X_S X_T],$$

by linearity of expectation.

Note that if $|S \cap T| \leq 1$ then X_S and X_T are independent events and $\mathbf{E}[X_S X_T] = \mathbf{E}[X_S]\mathbf{E}[X_T] = p^2$. Now we calculate the fraction of S, T that satisfy $|S \cap T| = i > 1$. There are $\binom{n}{k}$ ways to choose S and then $\binom{k}{i}\binom{n-k}{k-i}$ ways to choose T. Then the probability that both are independent sets is $2^{-2\binom{k}{2}}2^{\binom{i}{2}} = p^2 2^{\binom{i}{2}}$. Since $X_S X_T = 1$ iff both S and T are independent sets and 0 otherwise, we have

$$\mathbf{E}[Y_k^2] = \binom{n}{k}p^2\left(\binom{n-k}{k} + \binom{k}{1}\binom{n-k-1}{k-1} + \sum_{i\geq 2}\binom{k}{i}\binom{n-k}{k-i}2^{\binom{i}{2}}\right).$$

This can be shown to be $N^2p^2 + \varepsilon$ for some $\varepsilon \ll N^2p^2$. □

In fact, in 1976 Bollobás and Erdős showed that for every n there is a number $k(n)$ (presumably around $2\log n$) such that the distribution of the size of the maximum independent set is very tightly concentrated on $k(n)$ and $k(n)+1$.

Algorithms: Somebody asked if we can find a clique or independent set of size $2\lceil \log n \rceil$ in polynomial time on such graphs. We describe a simple greedy polynomial time algorithm that finds a clique of size $\lceil \log n \rceil$ in a random graph with high probability. The greedy process is simply this: select a vertex v_1 at random, and then look at its set of neighbours, V_1. Next, Select a vertex v_2 in V_1 and restrict attention to neighbours of v_1 and v_2; call this set V_2. We continue this process until we reach a V_k that is empty. Thus $\{v_1, v_2, \ldots, v_k\}$ is a maximal clique.

Using the fact that in a random graph, all degrees are concentrated in $[\frac{n}{2} - O(\sqrt{n}\log n), \frac{n}{2} + O(\sqrt{n}\log n)]$ (see Chernoff bounds below), we can show that $k \geq \log n - O(1)$ with high probability.

OPEN PROBLEM 1 Is it possible to find a clique of size $(1+\varepsilon)\log n$ in polynomial time? (The best algorithm we know of does exhaustive listing of sets of size $(1+\varepsilon)\log n$, which takes about $n^{(1+\varepsilon)\log n}$ time.

1.3 Graph colouring: local versus global

Imagine that one day you pick up a political map of some world and decide to colour in the countries. To keep the boundaries clear you decide to fill in adjacent countries with different colours. It turns out that no matter what map anyone gives you (a map is just a so-called *planar graph*), four colours will always suffice.

We can extend this notion of colouring to graphs: we assign each vertex a colour so that no two vertices with an edge between them share a colour. More formally, a ***k*-colouring** of a graph $G = (V, E)$ is a family of k independent sets whose union is V. Note that we can ask that these sets form a *partition* of V, but we don't have to.

Following convention, let $\alpha(G)$ stand for the size of the largest independent set in G and let $\chi(G)$ stand for the minimum k for which G admits a k-colouring. The formal definition of colouring tells us that

$$\chi(G) \geq \frac{n}{\alpha(G)}.$$

Can we provide an upper bound for the **chromatic number**, $\chi(G)$, using $\alpha(G)$, for instance $2n/\alpha(G)$? Such a bound is impossible: there exist graphs in which there is a large independent set, but the remainder of the graph is a clique. However, if the vertex degrees are bounded, then so is the chromatic number.

THEOREM 3
If the maximum degree of a graph is d then $\chi(G) \leq d+1$.

PROOF: Proof by induction: The base case is trivially true. For the inductive step, assume all graphs with n nodes satisfy the theorem. Given a graph G with $n+1$ nodes, identify one vertex v. The induced subgraph $G - \{v\}$ can be coloured with $d+1$ colours, by the induction assumption. The node v has links to at most d other nodes, due to the maximum degree constraint, and hence is adjacent to at most d colours. Hence the $d+1$th colour is available for v and so G can be $d+1$-coloured. □

Perhaps being able to colour small subgraphs with few colours might help us to colour larger graphs? Such conjectures can waste a lot of brain-cycles. But a simple probabilistic argument —doable on the back of an envelope— can help us dispose of such conjectures, as

in the following 1962 result by Erdős. It shows that the chromatic number is truly a global property and cannot be deduced from looking locally.

THEOREM 4
For all k there exists a positive ε such that for all sufficiently large n there is a graph G on n vertices with $\chi(G) > k$, but every subgraph of G with at most εn vertices is 3-colourable.

PROOF: Let G be a graph selected from $G(n,p)$ where $p = c/n$. We show not only that there exists such a G that satisfies the theorem statement, but that selected in this way G satisfies it almost surely.

First, we show that with high probability $\alpha(G) < n/k$, which implies $\chi(G) > k$. We can approximate $\binom{n}{an}$ with $2^{H(a)n}$, where $H(a)$ is the *entropy* function

$$a \log\left(\frac{1}{a}\right) + (1-a)\log\left(\frac{1}{1-a}\right).$$

Using the fact that when p is small, $1 - p \approx e^{-p}$, we find that the expected number of independent sets of size n/k is

$$\binom{n}{n/k}(1-p)^{\binom{n/k}{2}} \approx 2^{nH(1/k)} e^{-pn^2/2k^2} = \exp\left[nH\left(\frac{1}{k}\right)(\ln 2) - \frac{cn}{k^2}\right].$$

If c is at least $2k^2 H(1/k)\ln 2$, this expectation drops to zero rapidly.

Second, we show that in every induced subgraph on at most εn nodes, the average degree is less than 3. Note that if there exists a subgraph on εn vertices that is not 3-colourable, then there is a minimal such subgraph. In the minimal such subgraph, every vertex must have degree at least 3 (Proof: Suppose a vertex has degree 2 yet omitting it gives a 3-colorable subgraph. Then putting the vertex back in, we can extend the 3-coloring to that vertex by giving it a color that has not been assigned to its 2 neighbors.) So we have a subgraph on $s \le \varepsilon n$ vertices with at least $3s/2$ edges. The probability of such a subgraph existing is at most

$$\sum_{s \le \varepsilon n} \binom{n}{s} \binom{\binom{s}{2}}{3s/2} \left(\frac{c}{n}\right)^{3s/2},$$

recalling that $p = c/n$. If s is $O(1)$, the terms tend to zero rapidly and the sum is negligible. Otherwise, we can use the approximations presented in line (1.1) to arrive at the quantity

$$\sum_{s \le \varepsilon n} \left[\frac{ne}{s}\left(\frac{se}{3}\right)^{3/2}\left(\frac{c}{n}\right)^{3/2}\right]^s \le \left[e^{5/2} 3^{-3/2} c^{3/2}\sqrt{s/n}\right]^s.$$

If ε is less than $e^{-5}3^3 c^{-3}$, the summation terms are strictly less than 1, since $s \le \varepsilon n$. On the other hand we have eliminated the cases where s is $O(1)$ so the remainder of the terms form a geometric series and so their sum is bounded by a constant times the first term, which is tiny.

□

1.4 Bounding distribution tails

When we toss a coin many times, the expected number of heads is half the number of tosses. How tightly is this distribution concentrated? Should we be very surprised if after 1000 tosses we have 625 heads? We can provide bounds on how likely a sum of Poisson trials is to deviate from its mean. (A sequence of $\{0,1\}$ random variables is Poisson, as against Bernoulli, if the expected values can vary between trials.) The **Chernoff bound** presented here was probably known before Chernoff published it in 1952.

THEOREM 5
Let $X_1, X_2, \ldots, X_n$ be independent Poisson trials and let $p_i = \mathbf{E}[X_i]$, where $0 < p_i < 1$. Then the sum $X = \sum_{i=1}^{n} X_i$, which has mean $\mu = \sum_{i=1}^{n} p_i$, satisfies

$$\mathbf{Pr}[X \geq (1+\delta)\mu] \leq \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}.$$

PROOF: Surprisingly, this inequality also is proved using the Markov inequality.

We introduce a positive dummy variable t and observe that

$$\mathbf{E}[\exp(tX)] = \mathbf{E}[\exp(t\sum_i X_i)] = \mathbf{E}[\prod_i \exp(tX_i)] = \prod_i \mathbf{E}[\exp(tX_i)], \tag{1.3}$$

where the last equality holds because the X_i r.v.s are independent. Now,

$$\mathbf{E}[\exp(tX_i)] = (1-p_i) + p_i e^t,$$

therefore,

$$\begin{aligned}\prod_i \mathbf{E}[\exp(tX_i)] = \prod_i [1 + p_i(e^t - 1)] &\leq \prod_i \exp(p_i(e^t-1)) \\ &= \exp(\sum_i p_i(e^t-1)) = \exp(\mu(e^t-1)),\end{aligned} \tag{1.4}$$

as $1 + x \leq e^x$. Finally, apply Markov's inequality to the random variable $\exp(tX)$, viz.

$$\mathbf{Pr}[X \geq (1+\delta)\mu] = \mathbf{Pr}[\exp(tX) \geq \exp(t(1+\delta)\mu)] \leq \frac{\mathbf{E}[\exp(tX)]}{\exp(t(1+\delta)\mu)} = \frac{\exp((e^t-1)\mu)}{\exp(t(1+\delta)\mu)},$$

using lines (1.3) and (1.4) and the fact that t is positive. Since t is a dummy variable, we can choose any positive value we like for it. The right hand size is minimized if $t = \ln(1+\delta)$—just differentiate—and this leads to the theorem statement. □

A similar technique yields this result for the lower tail of the distribution

$$\mathbf{Pr}[X \leq (1-\delta)\mu] \leq \left[\frac{e^{-\delta}}{(1-\delta)^{(1-\delta)}}\right]^{\mu}$$

By the way, if all n coin tosses are fair (Heads has probability $1/2$) then the the probability of seeing N heads where $|N - n/2| > a\sqrt{n}$ is at most $e^{-a^2/2}$. The chance of seeing at least 625 heads in 1000 tosses of an unbiased coin is less than 5.3×10^{-7}.

§1 Prove that the greedy algorithm finds a clique of size $\log n - O(1)$ with high probability in a random graph.

Chapter 2

LP Duality and its Uses

SCRIBE: *Satyen Kale*

We introduce linear programs and the famous Duality Theorem. Our goal is to showcase linear programs as mathematical tools; their use in algorithms is of course well-known. Our main example is a result by Linial and Nisan on Approximate Inclusion-Exclusion. Another example in the same spirit is the LP bound on density of error-correcting codes; see a book on coding theory such as Van Lint or MacWilliams-Sloane.

2.1 Linear Programming and Farkas' Lemma

A Linear Program involves optimizing a linear cost function with respect to *linear* inequality constraints. They are useful for algorithm design as well as a tool in mathematical proofs. The typical program looks as follows.

GIVEN: vectors $\mathbf{c}, \mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_m \in \mathbf{R}^n$, and real numbers $b_1, b_2, \ldots b_m$.
OBJECTIVE: find $\mathbf{X} \in \mathbf{R}^n$ to minimize $\mathbf{c} \cdot \mathbf{X}$, subject to:

$$\begin{array}{rcl} \mathbf{a}_1 \cdot \mathbf{X} & \geq & b_1 \\ \mathbf{a}_2 \cdot \mathbf{X} & \geq & b_2 \\ & \vdots & \\ \mathbf{a}_m \cdot \mathbf{X} & \geq & b_m \\ \mathbf{X} & \geq & 0 \end{array} \tag{2.1}$$

The notation $\mathbf{X} > \mathbf{Y}$ simply means that $\mathbf{X}$ is componentwise larger than $\mathbf{Y}$. Now we represent the system in (2.1) more compactly using matrix notation. Let

$$A = \begin{pmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_m^T \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

Then the Linear Program (LP for short) can be rewritten as:

$$\begin{array}{rl} \min & \mathbf{c}^T\mathbf{X}: \\ A\mathbf{X} & \geq \mathbf{b} \\ \mathbf{X} & \geq 0 \end{array} \tag{2.2}$$

This form is general enough to represent any possible linear program. For instance, if the linear program involves a linear equality $\mathbf{a} \cdot \mathbf{X} = b$ then we can replace it by two inequalities

$$\mathbf{a} \cdot \mathbf{X} \geq b \quad \text{and} \quad -\mathbf{a} \cdot \mathbf{X} \geq -b.$$

If the variable X_i is unconstrained, then we can replace each occurence by $X_i^+ - X_i^-$ where X_i^+, X_i^- are two new non-negative variables.

The set of conditions in an LP may not be satisfiable, however. Farkas' Lemma tells us when this happens.

LEMMA 6
Farkas' Lemma. *The set of linear inequalities (2.1) is infeasible if and only if using positive linear combinations of the inequalities it is possible to derive* $-1 \geq 0$, *i.e. there exist* $\lambda_1, \lambda_2, \ldots \lambda_m \geq 0$ *such that*

$$\sum_{i=1}^{m} \lambda_i \mathbf{a}_i < 0 \quad \textit{and} \quad \sum_{i=1}^{m} \lambda_i b_i > 0.$$

2.2 The Duality Theorem

With every LP we can associate another LP called its *dual*. The original LP is called the *primal*. If the primal has n variables and m constraints, then the dual has m variables and n constraints.

$$\begin{array}{rll|rll} \multicolumn{3}{c|}{\underline{\text{Primal}}} & \multicolumn{3}{c}{\underline{\text{Dual}}} \\ \min & \mathbf{c}^T\mathbf{X}: & & \max & \mathbf{Y}^T\mathbf{b}: & \\ A\mathbf{X} & \geq \mathbf{b} & & \mathbf{Y}^T A & \leq \mathbf{c}^{\mathbf{T}} & \\ \mathbf{X} & \geq 0 & & \mathbf{Y} & \geq 0 & \end{array} \tag{2.3}$$

(Aside: if the primal contains an equality constraint instead of inequality then the corresponding dual variable is unconstrained.)

It is an easy exercise that the dual of the dual is just the primal.

THEOREM 7
The Duality Theorem. *If both the Primal and the Dual of an LP are feasible, then the two optima coincide.*

PROOF: The proof involves two parts:

1. Primal optimum ≥ Dual optimum.
   This is the easy part. Suppose $\mathbf{X}^*, \mathbf{Y}^*$ are the respective optima. This implies that

   $$A\mathbf{X}^* \geq \mathbf{b}.$$

   Now, since $\mathbf{Y}^* \geq 0$, the product $\mathbf{Y}^* A\mathbf{X}^*$ is a non-negative linear combination of the rows of $A\mathbf{X}^*$, so the inequality

   $$\mathbf{Y}^{*T} A\mathbf{X}^* \geq \mathbf{Y}^{*T}\mathbf{b}$$

   holds. Again, since $\mathbf{X}^* \geq 0$ and $\mathbf{c}^T \geq \mathbf{Y}^{*T} A$, the inequality

   $$\mathbf{c}^T\mathbf{X}^* \geq (\mathbf{Y}^{*T} A)\mathbf{X}^* \geq \mathbf{Y}^{*T}\mathbf{b}$$

   holds, which completes the proof of this part.

2. Dual optimum ≥ Primal optimuml.
   Let k be the optimum value of the primal. Since the primal is a minimization problem, the following set of linear inequalities is infeasible for any $\epsilon > 0$:

$$\begin{aligned} -\mathbf{c}^T\mathbf{X} &\geq -(k-\epsilon) \\ A\mathbf{X} &\geq \mathbf{b} \end{aligned} \tag{2.4}$$

   Here, ϵ is a small positive quantity. Therefore, by Farkas' Lemma, there exist $\lambda_0, \lambda_1, \ldots \lambda_m \geq 0$ such that

$$-\lambda_0\mathbf{c} + \sum_{i=1}^{m} \lambda_i \mathbf{a}_i < 0 \tag{2.5}$$

$$-\lambda_0(k-\epsilon) + \sum_{i=1}^{m} \lambda_i b_i > 0. \tag{2.6}$$

   Note that $\lambda_0 > 0$ omitting the first inequality in (2.4) leaves a feasible system by assumption about the primal. Thus, consider the vector

$$\Lambda = (\frac{\lambda_1}{\lambda_0}, \ldots \frac{\lambda_m}{\lambda_0})^T.$$

   The inequality (2.5) implies that $\Lambda^T A \leq \mathbf{c}^T$. So Λ is a feasible solution to the Dual. The inequality (2.6) implies that $\Lambda^T\mathbf{b} > (k-\epsilon)$, and since the Dual is a maximization problem, this implies that the Dual optimal is bigger than $k-\epsilon$. Letting ϵ go to zero, we get that the Dual optimal ≥ k = Primal optimal. Thus, this part is proved, too. Hence the Duality Theorem is proved.

□

Sanjeev's thoughts on this business: (1) Usually textbooks bundle the case of infeasible systems into the statement of the Duality theorem. He feels that this muddies the issue. Usually all applications of LPs fall into two cases: (a) We either know (for trivial reasons) that the system is feasible, and are only interested in the value of the optimum or (b) We do not know if the system is feasible and that is precisely what we want to determine. Then it is best to just use Farkas' Lemma. (2) The proof of the Duality theorem is interesting. The first part shows that for any dual feasible solution $\mathbf{Y}$ the various Y_i's can be used to obtain a *weighted* sum of primal inequalities, and thus obtain a lowerbound on the primal. The second part shows that this method of taking weighted sums of inequalities is *sufficient* to obtain the best possible lowerbound on the primal: there is no need to do anything fancier (e.g., taking products of inequalities or some such thing).

2.3 Example: Max Flow Min Cut theorem in graphs

The input is a directed graph $G(V,E)$ with one source s and one sink t. Each edge e has a capacity c_e. The flow on any edge must be less than its capacity, and at any node apart from s and t, flow must be conserved: total incoming flow must equal total outgoing flow. We wish to maximize the flow we can send from s to t. The maximum flow problem can be formulated as a Linear Program as follows:

Let $\mathcal{P}$ denote the set of all (directed) paths from s to t. Then the max flow problem becomes:

$$\max \sum_{P \in \mathcal{P}} f_P : \tag{2.7}$$

$$\forall P \in \mathcal{P} : f_P \geq 0 \tag{2.8}$$

$$\forall e \in E : \sum_{P : e \in P} f_P \leq c_e \tag{2.9}$$

Going over to the dual, we get:

$$\min \sum_{e \in E} c_e y_e : \tag{2.10}$$

$$\forall e \in E : y_e \geq 0 \tag{2.11}$$

$$\forall P \in \mathcal{P} : \sum_{e \in P} y_e \geq 1 \tag{2.12}$$

Notice that the dual in fact represents the Fractional min $s-t$ cut problem: think of each edge e being picked up to a fraction y_e. The constraints say that a total weight of 1 must be picked on each path. Thus the usual min cut problem simply involves $0-1$ solutions to the y_e's in the dual.

EXERCISE 1 Prove that the optimum solution does have $y_e \in \{0,1\}$.

Thus, the max- st-flow = (capacity of) min-cut.

2.4 Approximate Inclusion-Exclusion

WARNING: THE PROOF OF APPROXIMATE INCLUSION-EXCLUSION GIVEN HERE IS INCOMPLETE TOWARDS THE END. SPECIFICALLY, I FOUND AFTER GIVING THE LECTURE THAT THE MAIN THEOREM OF THE LINIAL-NISAN PAPER IS INCORRECT AS STATED. I PLAN TO CORRECT THIS BUT HAVEN'T GOTTEN AROUND TO. THE READER CAN STILL GET THE BASIC IDEA FROM THE DESCRIPTION BELOW.

Now we see an interesting application of linear programming and the duality theorem. The Inclusion-Exclusion formula for the cardinality of the union of n finite sets $A_1, A_2, \ldots, A_n$ is given by

$$|A_1 \cup A_2 \cup \ldots \cup A_n| = \sum_i |A_i| - \sum_{i<j} \left| A_i \cap A_j \right| - \cdots + (-1)^{n+1} |A_1 \cap A_2 \cap \ldots \cap A_n| \tag{2.13}$$

The question is, suppose we know only the first k terms of the formula, how good an approximation can we get? The simple idea of truncating the formula to the first k terms doesn't work: for instance, consider the case when all A_i are identical.

The answer (due to Linial and Nisan, 1988) is that for $k \geq \Omega(\sqrt{n})$, we can get a good approximation, correct upto a multiplicative factor $1 - \mathcal{O}(\exp(\frac{-2k}{n}))$, while for $k \leq \mathcal{O}(\sqrt{n})$, no good approximation is possible.

Our approach is to look at a related question. Let $\mathcal{A} = (A_1, A_2, \ldots, A_n)$ and $\mathcal{B} = (B_1, B_2, \ldots, B_n)$ be two collections of sets. Assume that for any set $S \subset [n]$ with $|S| \leq k$ we have $\left|\bigcap_{i \in S} A_i\right| = \left|\bigcap_{i \in S} B_i\right|$. We want to estimate how far apart $|\bigcup A_i|$ and $|\bigcup B_i|$ can be.

Without loss of generality, we may assume that the A_i and the B_i are events in a probability space, and we will consider a slightly more general question, viz. we would like to estimate

$$E(k,n) = sup(\mathbf{Pr}[\bigcup_{i=1}^{n} A_i] - \mathbf{Pr}[\bigcup_{i=1}^{n} B_i]), \tag{2.14}$$

where $(A_1, A_2, \ldots, A_n)$ and $(B_1, B_2, \ldots, B_n)$ satisfy the condition

$$\mathbf{Pr}[\bigcap_{i \in S} A_i] = \mathbf{Pr}[\bigcap_{i \in S} B_i] \tag{2.15}$$

for every $S \subset [n]$ such that $|S| \leq k$.

Now, we define a j-atom to be an intersection of exactly j of the A_i's and the complements of the remaining $(n - j)$. Note that any other event consisting of intersections of events A_i or their complements can be expressed in terms of these atoms, and that all the atoms are disjoint events. We call a collection of n events *symmetric* if all the j-atoms occur with the same probability.

LEMMA 8
The optimum value of $E_{k,n}$ is attained for some $\mathcal{A}$ and $\mathcal{B}$ that are symmetric.

PROOF: For any $\mathcal{A}$ we construct a *symmetric* collection $\mathcal{A}'$ whose each term in the Inclusion-Exclusion formula is the same. The Lemma then follows.

Obtain $\mathcal{A}'$ by setting the probability of each j-atom of $\mathcal{A}'$ to be the average of the probabilities of the j-atom of $\mathcal{A}$. There is only one n-atom, so $\Pr[A'_1 A'_2 \cdots A'_n] = \Pr[A_1 A_2 \cdots A_n]$. Now consider $\Pr[A'_1 A'_2 \cdots A'_{n-1}]$. It may be expressed as

$$\Pr[A'_1 A'_2 \cdots A'_{n-1} \overline{A_n}'] + \Pr[A'_1 A'_2 \cdots A'_{n-1} A'_n].$$

Doing a similar rewriting for all conjunctions of $(n - 1)$ events, and adding, we see that the $(n - 1)$th term of the inclusion-exclusion for $\mathcal{A}'$ is the same as for $\mathcal{A}$. Proceeding this way, a simple induction shows the equivalence of all terms. □

Now let

$$a_j = \text{sum of all } j\text{-atoms of } A_1, A_2, \ldots, A_n; \tag{2.16}$$
$$b_j = \text{sum of all } j\text{-atoms of } B_1, B_2, \ldots, B_n; \tag{2.17}$$
$$r_j = \text{sum of all } j\text{-intersections of } A_1, A_2, \ldots, A_n; \tag{2.18}$$
$$q_j = \text{sum of all } j\text{-intersections of } B_1, B_2, \ldots, B_n. \tag{2.19}$$

LEMMA 9
The following relation connects the r_j and the a_j:

$$r_j = \sum_{i=j}^{n} \binom{i}{j} a_i.$$

PROOF: Consider a generic term, say $\mathbf{Pr}[A_1 \cap A_2 \cap \ldots A_j]$ contributing to r_j. Expand this out as the summation of atoms:

$$\mathbf{Pr}[A_1 \cap A_2 \cap \ldots A_j] = \sum \mathbf{Pr}[A_1 \cap A_2 \cap \ldots A_j \cap s],$$

where s in the summation runs over all the atoms of $A_{j+1}, \ldots, A_n$.

In this summation, each i-atom is counted exactly $\binom{i}{j}$ times: once for each j-subset of the i uncomplemented events. The lemma follows. □

Now, we will construct an LP that solves (2.14).

LEMMA 10

The solution to (2.14) is given by solving the following LP:

$$max \quad \sum_{i=1}^{n} x_i : \tag{2.20}$$

$$\forall j \le k : \quad \sum_{i=j}^{n} \binom{i}{j} x_i = 0 \tag{2.21}$$

$$\forall S \subseteq [n] : \quad \sum_{i \in S} x_i \le 1 \tag{2.22}$$

$$-\sum_{i \in S} x_i \le 1 \tag{2.23}$$

PROOF: First, we check that $x_i = a_i - b_i$ is feasible: for (2.21),

$$\sum_{i=j}^{n} \binom{i}{j} x_i = r_j - q_j = 0 \quad \forall i \le k,$$

and (2.22, 2.23) follow easily from noting that the a_j and b_j are probabilities of *disjoint* events, and so for any $S \subseteq [n]$ the sum $\sum_{i \in S} x_i$ represents a difference in probabilities and is therefore bounded in absolute value by 1.

For the reverse direction, let $x_1, x_2, \ldots, x_n$ be a feasible solution to the LP. Now define the a_j and b_j by

$$a_j = \begin{cases} x_i \text{ if } x_i > 0 \\ 0 \text{ otherwise} \end{cases} \tag{2.24}$$

$$b_j = \begin{cases} -x_i \text{ if } x_i < 0 \\ 0 \text{ otherwise} \end{cases} \tag{2.25}$$

It is easy to check that this defines proper probabilities using (2.22, 2.23), and that the sequences of events they define satisfy (2.15) because of (2.21). □

Now we look at the dual of the LP. It involves finding $\mu_S, \lambda_S \ge 0$ and y_i's such that

$$\min \quad \sum_{S \subseteq [n]} (\lambda_S + \mu_S) : \tag{2.26}$$

$$\forall 1 \le i \le n : \quad \sum_{S \ni i} (\lambda_S - \mu_S) + \sum_{j \le min(i,k)} \binom{i}{j} y_j \ge 1 \tag{2.27}$$

Clearly, the optimum solution will, for each subset S, never need to make both λ_S and μ_S positive. For example, if $\lambda_S = a > 0$ and $\mu_S = b > a$ then making $\lambda_S = 0$ and $\mu_S = b - a$ still satisfies all constraints while lowering the objective.

For any given $y_1, y_2, \ldots, y_n$, what is the best choice of λ_S, μ_S? For each i let $c_i = 1 - \sum_{j \le min(i,k)} \binom{i}{j} y_j$. Let $I = \{i : c_i > 0\}$ and $J = \left\{j : c_j < 0\right\}$. Let $c_+ = \max\{c_i : i \in I\}$ and $c_- = \min\left\{-c_j : j \in J\right\}$. (If I or J is empty the corresponding max or min is defined to be 0.) Then there is a dual solution of cost $c_+ + c_-$, namely, $\lambda_I = c_+$, $\mu_J = c_-$ and the variables associated with all other subsets are zero. Furthermore, every feasible solution must have *some* set with $\lambda_S \ge c_+$ and some other set with $\mu_S \ge c_-$, and thus have cost at least $c_+ + c_-$. Finally, we claim that at the optimum, the y_i's are such that $J = \emptyset$, and hence $c_- = 0$. Suppose not, and $c_- > 0$. Then divide all y_i by $1 + c_-$; a simple calculation shows that the new c_- is 0 whereas the new c_+ is $\frac{c_+ + c_-}{1 + c_-}$. Thus the objective function has gone down, which contradicts optimality.

We have thus proved:

LEMMA 11
The dual solution is the following optimization problem.

$$\min_{y_1,\ldots,y_n}\left[\max_{1\le i\le n}\left(1-\sum_{j\le min(i,k)}\binom{i}{j}y_j\right)\right]: \tag{2.28}$$

$$\left(1-\sum_{j\le min(i,k)}\binom{i}{j}y_j\right)\ge 0 \qquad \forall i. \tag{2.29}$$

LEMMA 12
The optimum value of the program in Lemma 11 is given by

$$\inf_q\left(\max_{m\in[n]}(1-q(m))\right) \tag{2.30}$$

where the infimum ranges over all polynomials q of degree atmost k with constant term 0 such that $q(m)\le 1$ for all $m\in\{1,\ldots,n\}$.

PROOF: Recall that $\binom{x}{i}=\frac{x(x-1)(x-2)\cdots(x-i+1)}{i!}$, which is a degree i polynomial whose constant term is 0. (That is, at $x=0$ its value is 0.) It is also a polynomial that is 0 at $x=1,2\ldots,(i-1)$. We note that any polynomial of degree at most k and constant term 0 can be written as a linear combination of $\binom{x}{i}$, for $1\le i\le k$. (The proof is by induction. If the polynomial is $cx^i+q(x)$ where $q(x)$ is a polynomial of degree at most $i-1$, then it may be expressed as $ci!\binom{x}{i}+r(x)$ where $r(x)$ has degree at most $i-1$.) Finally, note that if we define the polynomial $q(x)=\sum_{j=1}^k\binom{x}{j}y_j$, then (2.28) becomes simply

$$\inf_q\left(\max_{m\in\{1,\ldots,n\}}(1-q(m))\right)$$

as required. The proof follows. □

To prove an upperbound on the primal, we construct a suitable *feasible* solution to the dual (2.28). We use the *Chebyshev polynomials*. Here are some of their properties:

1. Recall that for each integer $m\ge 0$, $\cos(m\theta)$ is a polynomial in $\cos(\theta)$ of degree m. Thus $\cos(m\cos^{-1}(x))$ is a degree m polynomial in x, called the m'th Chebyshev polynomial $T_m(x)$. It is also given by

$$T_m(x)=\frac{(x+\sqrt{x^2-1})^m+(x-\sqrt{x^2-1})^m}{2}. \tag{2.31}$$

2. For $x\in[-1,1]$, we have $-1\le T_m(x)\le 1$.

Consider the following polynomial of degree k,

$$q_{k,n}(x)=1-\frac{T_k(\frac{2x-(n+1)}{n-1})}{T_k(\frac{-(n+1)}{n-1})} \tag{2.32}$$

Note that $q(0)=0$ (i.e. its constant term is 0) and when $x\in[1,n]$, $T_k(\frac{2x-(n+1)}{n-1})\in[-1,1]$ so $\left|q_{k,n}(x)-1\right|\le 1/D$ where $D=\left|T_k(\frac{-(n+1)}{n-1})\right|$.

Then $p(x)=Dq_{k,n}(x)/(1+D)$ satisfies $\frac{D-1}{1+D}\le p(m)\le 1$ for all $m\in[1,n]$. Thus it is a dual feasible solution and we conclude that $E(k,n)\le 1-\frac{D-1}{1+D}=\frac{2}{1+D}$. Thus the maximum ratio

for $\Pr[\cup_i A_i]$ and $\Pr[\cup_i B_i]$ is $\frac{1}{1-E(k,n)} \leq \frac{D+1}{D-1}$. It only remains to estimate this quantity. Since $D = \left| T_k(-\frac{n+1}{n-1}) \right| = (\lambda^k + \lambda^{-k})/2$ where $\lambda = \frac{\sqrt{n}+1}{\sqrt{n}-1} \approx 1 + \frac{2}{\sqrt{n}}$ for large n, we can upperbound this by

$$\frac{D+1}{D-1} \leq 1 + O(k^2/n).$$

2.5 A note on algorithms

We have emphasized the use the duality theorem as a tool for proving theorems. Of course, the primary use of LPs is to solve optimization problems. Several algorithms exist to solve LPs in polynomial time. We want to mention Khachiyan's *ellipsoid algorithm* in particular because it can solve even exponential size LPs provided they have a polynomial time *separation oracle*. (There is an additional technical condition that we need to know a containing ball for the polytope in question, and the ball should not be too much bigger than the polytope. Usually this condition is satisfied.)

A separation oracle for an LP decides whether a given input $(x_1, x_2, \ldots, x_n)$ is feasible or not, and if it isn't, outputs out one constraint that it violates.

For example, consider the dual of max-flow (viz. fractional min-cut) that was discussed in the previous lecture:

$$\min \sum_{e \in E} c_e y_e : \tag{2.33}$$

$$\forall e \in E : y_e \geq 0 \tag{2.34}$$

$$\forall P \in \mathcal{P} : \sum_{e \in P} y_e \geq 1 \tag{2.35}$$

This can be solved in many ways, but the simplest (if we do not care about efficiency too much) is to use the Ellipsoid method, since we can design a polytime separation oracle for this problem using the *shortest path algorithm*. Suppose the oracle is given as input a vector $(y_e)_{e \in E}$. To decide if it is feasible, the oracle computes the shortest path from s to t with edge weights $= y_e$, and checks if the length of this shortest path is atleast 1. (Of course, before anything else one should check if all the $y_e > 0$.) Clearly, $(y_e)_{e \in E}$ is feasible iff the shortest path has length at least 1, and if it is infeasible then the shortest path constitutes an unsatisfied constraint.

Chapter 3

The Dimension Method

SCRIBE:*Miroslav Dudík*

The "Dimension Method" is Sanjeev's name for elementary linear algebra arguments. This is all the algebra one needs 80% of the time; only occasionally (one example is number theoretic cryptography) does one need anything more powerful than elementary linear algebra.

3.1 Basics: Fields and Vector Spaces

We recall some basic linear algebra. A *field* is a set closed under addition, subtraction, multiplication and division by nonzero elements. By addition and multiplication, we mean commutative and associative operations which obey distributive laws. The additive identity is called zero, the multiplicative identity is called unity. Examples of fields are reals $\mathbf{R}$, rationals $\mathbf{Q}$, and integers modulo a prime p denoted by $\mathbf{Z}/p$. We will be mostly concerned with finite fields. The cardinality of a finite field must be a power of prime and all finite fields with the same number of elements are isomorphic. Thus for each power p^k there is essentially one field $\mathbb{F}$ with $|\mathbb{F}| = p^k$. We shall denote this field by $\mathrm{GF}(p^k)$.

A *vector space* V over a field $\mathbb{F}$ is an additive group closed under (left) multiplication by elements of $\mathbb{F}$. We require that this multiplication be distributive with respect to addition in both V and $\mathbb{F}$, and associative with respect to multiplication in $\mathbb{F}$.

Vectors $\mathbf{v_1}, \ldots, \mathbf{v_k}$ are said to be *linearly independent* if $\sum_{i=1}^{k} \alpha_i \mathbf{v_i} = \mathbf{0}$ implies that $\alpha_1 = \alpha_2 = \cdots = \alpha_k = 0$. A maximal set of vectors $\{\mathbf{v}_i\}_{i \in I}$ whose every finite subset is linearly independent is called a *basis* of V; all such sets have the same cardinality, called the *dimension* of V (denoted $\dim V$). If V has a finite dimension k and $\{\mathbf{v}_i\}_{i=1}^{k}$ is a basis then every vector $v \in V$ can be uniquely represented as

$$\mathbf{v} = \sum_{i=1}^{k} \alpha_i \mathbf{v}_i,$$

where $\alpha_i \in \mathbb{F}$. Thus all finite-dimensional vector spaces are isomorphic to $\mathbb{F}^k$. If $\mathbb{F}$ is finite then $|V| = |\mathbb{F}|^k$. An example of a vector space over a finite field is the field $\mathrm{GF}(p^k)$ when viewed as a vector space over $\mathrm{GF}(p)$.

Let $\mathbf{A}_{m \times n} = \left\{a_{ij}\right\}$ be a matrix over a field $\mathbb{F}$. Rank of $\mathbf{A}$, denoted by $\operatorname{rank} \mathbf{A}$, is the maximum number of linearly independent rows in $\mathbf{A}$. It is equal to the maximum number of linearly independent columns. Hence $\operatorname{rank} \mathbf{A} = \operatorname{rank} \mathbf{A}^T$.

Let $\mathbf{M} = \left\{m_{ij}\right\}$ be an n by n matrix. The determinant of $\mathbf{M}$ is defined as follows:

$$\det \mathbf{M} = \sum_{\sigma \in S_n} (-1)^{\pi(\sigma)} \prod_{i=1}^{n} m_{i\sigma(i)},$$

where S_n is the group of permutations over $[n]$, and $\pi(\sigma)$ is the parity of the permutation σ. The matrix $\mathbf{M}_{n\times n}$ has rank n if and only if $\det \mathbf{M} \neq 0$. We will use this fact to prove the following result, which is our first example of the Dimension Method.

THEOREM 13
Let $\mathbf{M}_{n\times n}$ *be a random matrix over GF*(2)*. Then* $\Pr[\det \mathbf{M} \neq 0] \geq 1/4$.

PROOF: Denote the columns of $\mathbf{M}$ by $\mathbf{M}_i$, where $i = 1, 2, \ldots, n$. It suffices to bound the probability that these columns are linearly independent:

$$\begin{aligned}
&\Pr[\mathbf{M}_1, \ldots, \mathbf{M}_n \text{ linearly independent}] \\
&\quad = \prod_{i=1}^{n} \Pr[\mathbf{M}_1, \ldots, \mathbf{M}_i \text{ linearly independent} \mid \mathbf{M}_1, \ldots, \mathbf{M}_{i-1} \text{ linearly independent}] \\
&\quad = \prod_{i=1}^{n} (1 - \Pr[\mathbf{M}_i \in \operatorname{span}(\mathbf{M}_1, \ldots, \mathbf{M}_{i-1}) \mid \mathbf{M}_1, \ldots, \mathbf{M}_{i-1} \text{ linearly independent}].
\end{aligned}$$

Now, if $\mathbf{M}_1, \ldots, \mathbf{M}_{i-1}$ are independent then their span is of dimension $i-1$ and hence it contains 2^{i-1} vectors. The column $\mathbf{M}_i$ is picked uniformly at random from the space of 2^n vectors, independently of $\mathbf{M}_1, \ldots, \mathbf{M}_{i-1}$. Thus the probability that it will lie in their span is $2^{i-1}/2^n$.

$$\begin{aligned}
&= \prod_{i=1}^{n} (1 - 2^{i-1-n}) \geq \prod_{i=1}^{n} \exp\{-2^{i-1-n} \cdot 2 \ln 2\} \\
&\geq \exp\{-2 \ln 2 \sum_{i=1}^{\infty} 2^{i-1-n}\} = \exp\{-2 \ln 2\} = 1/4.
\end{aligned}$$

□

3.2 Systems of Linear Equations

The system of m linear equations in n unknowns over a field $\mathbb{F}$ can be represented by a matrix $\mathbf{A}_{m\times n}$ and a vector $\mathbf{b}_{m\times 1}$ as

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \qquad (*)$$

where $\mathbf{x}_{n\times 1}$ is the vector of unknowns.

PROPOSITION 14

1. *The system* $(*)$ *is feasible if and only if* $b \in \operatorname{span}(\mathbf{A}_1, \ldots, \mathbf{A}_n)$*, which occurs if and only if* $\operatorname{rank}(\mathbf{A}|\mathbf{b}) = \operatorname{rank} \mathbf{A}$*. (Here* $\mathbf{A}|\mathbf{b}$ *is the matrix whose last column is* $\mathbf{b}$ *and the other columns are from* $\mathbf{A}$*.)*

2. *Suppose* $\mathbb{F}$ *is finite. If* $\operatorname{rank} \mathbf{A} = k$ *then the system* $(*)$ *has either 0 solutions (if infeasible) or* $\mathbb{F}^{n-k}$ *solutions (if feasible). In particular, if* $n = k$ *then the solution is unique if it exists.*

3. *If* $\mathbf{b} = \mathbf{0}$ *then a nontrivial solution exists if and only if* $\operatorname{rank} \mathbf{A} \leq n - 1$*. In particular, if* $n > m$ *then nontrivial solutions always exist.*

EXAMPLE 2 Suppose M is a random matrix over GF(2) and $\mathbf{b}$ is a random $n \times 1$ vector. What is the probability that the system $Mx = \vec{b}$ has a unique solution? By Theorem 13 it is at least $1/4$.

THEOREM 15
A nonzero polynomial of degree d has at most d distinct roots.

PROOF: Suppose $p(x) = \sum_{i=0}^{d} c_i x^i$ has $d+1$ distinct roots $\alpha_1, \ldots, \alpha_{d+1}$ in some field $\mathbb{F}$. Then

$$\sum_{i=0}^{d} \alpha_j^i \cdot c_i = p(\alpha_j) = 0,$$

for $j = 1, \ldots, d+1$. This means that the system $\mathbf{Ay} = \mathbf{0}$ with

$$\mathbf{A} = \begin{pmatrix} 1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^d \\ 1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^d \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & \alpha_{d+1} & \alpha_{d+1}^2 & \ldots & \alpha_{d+1}^d \end{pmatrix}$$

has a solution $\mathbf{y} = \mathbf{c}$. The matrix $\mathbf{A}$ is a *Vandermonde* matrix, hence

$$\det \mathbf{A} = \prod_{i>j} (\alpha_i - \alpha_j),$$

which is nonzero for distinct α_i. Hence $\operatorname{rank} \mathbf{A} = d+1$. The system $\mathbf{Ay} = \mathbf{0}$ has therefore only a trivial solution — a contradiction to $\mathbf{c} \neq \mathbf{0}$. □

THEOREM 16
For any set of pairs $(a_1, b_1), \ldots, (a_{d+1}, b_{d+1})$ there exists a unique polynomial $g(x)$ of degree at most d such that $g(a_i) = b_i$ for all $i = 1, 2, \ldots, d+1$.

PROOF: The requirements are satisfied by *Lagrange Interpolating Polynomial*:

$$\sum_{i=1}^{d+1} b_i \cdot \frac{\prod_{j \neq i}(x - a_j)}{\prod_{j \neq i}(a_i - a_j)}.$$

If two polynomials $g_1(x), g_2(x)$ satisfy the requirements then their difference $p(x) = g_1(x) - g_2(x)$ is of degree at most d, and is zero for $x = a_1, \ldots, a_{d+1}$. Thus, from the previous theorem, polynomial $p(x)$ must be zero and polynomials $g_1(x), g_2(x)$ identical. □

3.3 Dispersal of Information Using Polynomials

Polynomials are often useful in situations where information needs to be dispersed in an error-tolerant way, so that it can be reconstructed even if it is partially corrupted or destroyed.

Suppose we want to encode a message into a finite number of packets to be transmitted through a faulty network. This network can drop up to $1/2$ of packets, but it does not corrupt the contents of the remaining packets. To achieve a successful transmission, we can use polynomial interpolation:

Encoding. Without loss of generality assume the message is a $d+1$-tuple $c_0, c_1, \ldots, c_d \in \mathbb{F}$, where $|\mathbb{F}| > 2d$. Take $2d+1$ distinct points $\alpha_1, \ldots, \alpha_{2d+1} \in \mathbb{F}$ and determine values of the polynomial $p(x) = \sum_{i=0}^{d} c_i x^i$ at α_i. Send packets $(\alpha_1, p(\alpha_1))$,...,$(\alpha_{2d+1}, p(\alpha_{2d+1}))$.

Decoding. Packets describe the polynomial p with sufficient redundancy. Even when d packets are dropped, the polynomial p and hence the original message is uniquely determined by the remaining $d+1$ pairs.

Now suppose that the network can corrupt up to 1/4th of the packets. We will use a strategy developed by Berlekamp and Welch in 1985. In order to transmit a message described by a polynomial $p(x)$ of degree d, we will send $20d$ pairs $(\alpha_i, p(\alpha_i))$. Let the received pairs be (α'_i, β'_i) (for missing packets, we can set $\alpha'_i = \beta'_i = 0$). A pair (α'_i, β'_i) will be considered corrupted if $p(\alpha'_i) \neq \beta'_i$. Then there exists a *nonzero* polynomial $e(x)$ of degree at most $5d$, which is zero at all corrupted values α'_i — this is called an *error locator polynomial*.

LEMMA 17
There exist nonzero polynomials $e(x)$ and $c(x)$ such that

$$\deg e \leq 5d$$
$$\deg c \leq 6d$$

and $c(\alpha'_i) = \beta'_i e(\alpha'_i)$ for $i = 1, 2, \ldots, 20d$.

PROOF: Taking the error locator polynomial $e(x)$ and $c(x) = p(x)e(x)$ we obtain

$$c(\alpha'_i) = p(\alpha'_i)e(\alpha'_i) = \begin{cases} \beta'_i e(\alpha'_i) & \text{if pair } (\alpha'_i, \beta'_i) \text{ is not corrupted} \\ p(\alpha'_i) \cdot 0 = 0 = \beta'_i \cdot 0 & \text{if pair } (\alpha'_i, \beta'_i) \text{ is corrupted.} \end{cases}$$

□

COROLLARY 18
Polynomials $e(x)$ and $c(x)$ that satisfy conditions of previous lemma can be found in time polynomial in d.

PROOF: Equations $c(\alpha'_i) = \beta'_i e(\alpha'_i)$, where coefficients of c and e are unknown, form a system of $20d$ linear equations in $11d + 2$ unknowns. The lemma guarantees its feasibility. We can solve for coefficients by Gaussian elimination. □

THEOREM 19 (BERLEKAMP-WELCH)
If $c(x), e(x)$ satisfy the conditions of the lemma then $e(x) | c(x)$ and $p(x) = c(x)/e(x)$.

PROOF: Consider the polynomial $c(x) - p(x)e(x)$. It has degree at most $6d$, but it has at least $15d$ roots because it is zero on all noncorrupted α'_i's. Therefore, $c(x) - p(x)e(x) \equiv 0$ and $c(x) \equiv p(x)e(x)$. □

REMARK 1 This strategy will work whenever a fixed δ-fraction of packets is corrupted, where $\delta < 1/2$. Somebody asked if a scheme is known that recovers the polynomial even if more than 1/2 the packets are corrupted. The answer is Yes, using Sudan's list decoding algorithm. See the homework.

3.4 Hashing: An introduction

Most schemes for Hashing also rely on a simple dimension argument.

Suppose we want to store n numbers from the set $1, 2, \ldots, q$ with a fast look-up. We will use an array of size p and each element insert into a *bucket* indexed by the *hash function*. Each bucket contains a chained list of elements with the same value of hash function. During a

look-up, it suffices to examine contents of a single bucket. If we can guarantee that the number of elements stored in a bucket (the *bucket size*) is small, the operation will be fast.

We will assume that q and p are prime, $p \approx 2n$ and choose a hash function h at random. We pick $a, b \in \mathrm{GF}(q)$ at random and define h as $x \to (ax + b \mod q) \mod p$. The probability of *collisions*, i.e. events when $h(x) = h(y)$ for $x \neq y$, should be low. We might for example require that the family of hash functions be *2-universal*:

$$(\forall x \neq y) \Pr_h[h(x) = h(y)] \leq \frac{2}{p}.$$

It is often possible to prove a stronger statement:

$$(\forall x \neq y)(\forall u, v) \Pr_h[h(x) = u, h(y) = v] = \frac{1}{p^2}.$$

Families satisfying this condition are called *pairwise independent*.

EXAMPLE 3 Consider a hash function $h : x \mapsto ax + b \mod p$, where a, b are picked randomly from $\mathrm{GF}(p)$. For fixed $x, y, u, v \in \mathrm{GF}(p)$, where $x \neq y$, the system

$$ax + b = u$$
$$ay + b = v$$

has a single solution $a, b \in \mathrm{GF}(p)$. Hence the probability that $h(x) = u, h(y) = v$ is $1/p^2$.

EXAMPLE 4 *Element Distinctiveness Problem.* We want to determine if there are two identical numbers in a given sequence. We can hash all elements and then examine each bucket separately. We could do it by sorting elements in every bucket, but for simplicity assume that we examine every pair in a given bucket. If the number of buckets is $O(n)$ and the expected number of pairs in a bucket is $O(1)$ then the expected runtime will be $O(n)$.

Suppose we use a hash function $h : \mathcal{X} \to \mathcal{U}$, $|\mathcal{U}| = p \approx 2n$, picked at random from a pairwise independent family. Fix a bucket u and consider random variables

$$X_x = \begin{cases} 1 & \text{if } h(x) = u, \\ 0 & \text{otherwise,} \end{cases}$$

where x is an element of the sequence. By pairwise independence, choosing arbitrary $y \in \mathcal{X}$ such that $y \neq x$, we obtain

$$\Pr[h(x) = u] = \sum_{v \in \mathcal{U}} \Pr[h(x) = u, h(y) = v] = 1/p.$$

The size of bucket u is $S = \sum_x X_x$. Calculate the expectation of S^2:

$$\begin{aligned} \mathbf{E}[S^2] &= \sum_{x,y} \mathbf{E}[X_x X_y] = \sum_x \mathbf{E}[X_x^2] + \sum_{x \neq y} \mathbf{E}[X_x X_y] \\ &= \sum_x \Pr[h(x) = u] + \sum_{x \neq y} \Pr[h(x) = u, h(y) = u] \\ &= n/p + n(n-1)/p^2 \approx 1/2 + 1/4 = O(1). \end{aligned}$$

Since the number of pairs in a bucket is $O(S^2)$, we obtain by linearity of expectation that the expected runtime is

$$O(\sum_u \mathbf{E}[S_u^2]) = O(n).$$

(*Aside:* The element distinctness problem is impossible to solve (even using randomness) in linear time in the comparison model, where the algorithm is only allowed to compare two numbers at every step.

3.5 Pairwise and k-wise Independent Sampling

Consider a randomized algorithm that uses n random bits and gives a Yes/No answer. Suppose we know that one of the answer happens with probability at least $2/3$ but we do not know which. We can determine that answer with high probability by running the algorithm m times with independent random bits and taking the majority answer; by the Chernoff bounds the error in this estimation will by $\exp(-m)$. Can we do this estimation using fewer than mn random bits? Intuitively speaking, the algorithm converts n random bits into a single random bit (Yes/No) so it has thrown away a lot of randomness. Can we perhaps "reuse" some of it? Later in the course we will see some powerful techniques to do so; here we use more elementary ideas. Here we see a technique that uses $2n$ random bits and its error probability is $1/m$. (We need $m < 2^n$.) The idea is to use random strings that are pairwise independent and use Chebyshev's inequality.

A sequence of random variables $z_1, z_2, z_3, \ldots$ is *pairwise independent* if every pair is independent.

We can construct a sequence of m pairwise independent strings $\{z_i\}, z_i \in \mathrm{GF}(q)$, $q \approx 2^n$ using $2\log q$ random bits. Let $\{x_i\}, x_i \in \mathrm{GF}(q)$ be any fixed sequence. Pick $a, b \in \mathrm{GF}(q)$ at random and set $z_i = ax_i + b$. Running the algorithm on $z_1, \ldots, z_m$ will guarantee that answers are pairwise independent.

Analogously, we can construct k-wise independent sequences by picking $a_0, \ldots, a_{k-1}$ at random and applying the map $x \mapsto \sum_{j=0}^{k-1} a_j x^j$ to an arbitrary sequence $\{x_i\}, x_i \in \mathrm{GF}(q)$. Chebyshev inequality generalizes to higher moments:

$$\Pr\left\{ |X - \mathbf{E}[X]| > \gamma \left(\mathbf{E}[|X - \mathbf{E}[X]|^k]\right)^{1/k} \right\} < \gamma^{-k}.$$

This uses $k \log q$ random bits but the error in the estimation goes down as $1/m^k$.

EXAMPLE 5 *Secret Sharing (A. Shamir,* How to share a secret, *Comm. ACM 1979).* We want to design a scheme for sharing a secret a_0 among m people so that $k+1$ people can recover the secret, but k or fewer people cannot.

If $a_0, \ldots, a_k$ are picked randomly and person i receives the pair $(\alpha_i, p(\alpha_i))$ where $p(x) = \sum a_i x^i$ then any set of k people will receive a random k-tuple of strings, whereas $k+1$ people will be able to recover the polynomial $p(x)$ by interpolation.

3.6 Madhu Sudan's List Decoding Algorithm

Sudan's algorithm gives a way to recover a polynomial from its values, even when most (an overwhelming majority) of the values are corrupted. See Question 4 on HW 2 for a self-guided tour of this algorithm.

Strictly speaking, this algorithm doesn't use "just" the dimension method: it also uses Berlekamp (or Kaltofen's) algorithm for factoring polynomials over finite fields.

3.7 The Razborov-Smolensky Circuit Lower Bound

Scribe: Paul Chang

We describe the Razborov-Smolensky method for circuit lowerbounds as another example of the dimension method.

We are very far from separating **NP** from **P**; the turing machine model seems too complex to reason about. It is natural to try to prove an exponential lower bound for a restricted model of

computation. The Razborov-Smolensky lower bound applies to one such model. The theorem presented in this section was originally proved by Razborov using his *Method of Approximations* (he won the Nevanlinna prize in 1990 for this work). Smolensky later extended this work and clarified this method for the circuit class considered here.

DEFINITION 1 *A circuit C is a directed acyclic graph with n input nodes, labeled x_1 to x_n, and one output node. All other nodes are labeled with a boolean operation. When we label the input nodes with n bits and let the other nodes compute in the obvious manner (computing the boolean operation on their incoming edges and placing the 1-bit result on the outgoing edges) we get a boolean value on the output node, denoted $C(x)$. The size of a circuit is the number of nodes. Note that the circuit nodes may have unbounded degree.*

DEFINITION 2 *A family of circuits is a collection of circuits $\{C_n\}_{n\geq 1}$, such that the nth circuit has n input nodes. We say that a circuit family computes a function $f : \{0,1\}^* \to \{0,1\}$ if for all x, $C_{|x|}(x) = f(x)$.*

REMARK 2 A circuit family can compute undecidable languages - we may hardwire into a circuit any of 2^n functions on n inputs, and may use different circuits for each input length. (For example, there exists a circuit family which computes $\{1^{<M,w>} | \text{M halts on w}\}$).

We are interested only in a subclass of circuits, the **MOD$_3$** circuits.

DEFINITION 3 *A* **MOD$_3$** *circuit of depth d on n inputs is one whose depth is bounded by a constant d. Each boolean gate performs any of the four operation AND ($\wedge$), OR ($\vee$), NOT ($\neg$), and sum modulo three (MOD_3). The MOD_3 gate outputs zero if the sum of its inputs is zero modulo three, and one otherwise.*

The inclusion of the MOD_3 gates gives the circuit some power. Nevertheless we show that it cannot compute the MOD_2 function, namely, parity of n bits.

THEOREM 20
The Razborov/Smolensky Circuit Lower Bound. *Computing MOD_2 with MOD_3 circuits of depth d requires a circuit of size* $\exp(\Omega(n^{\frac{1}{2d}}))$.

In the rest of the section we prove this theorem. The proof proceeds in two steps.

Step 1. In the first step, we show (using induction on h) that for any depth h MOD_3 circuit on n inputs and size S, there is a polynomial of degree $(2l)^h$ which agrees with the circuit on $1 - S/2^l$ fraction of the inputs. If our circuit C has depth d then we set $2l = n^{1/2d}$ to obtain a degree $\sqrt{n}$ polynomial that agrees with C on $1 - S/2^{n^{1/2d}/2}$ fraction of inputs.

Step 2 We show that no polynomial of degree $\sqrt{n}$ agrees with MOD_2 on more than $49/50$ fraction of inputs.

Together, the two steps imply that $S > 2^{n^{1/2d}/2}/50$ for any depth d circuit computing MOD_2, thus proving the theorem. Now we give details.
Step 1. Consider a node i in the circuit at a depth h . (The input is assumed to have depth 0.) If $f_i(x_i, \cdots, x_n)$ is the function computed at this node, we desire a polynomial $\tilde{f}_i(x_i, \cdots, x_n)$ over $GF(3)$ with degree $(2l)^h$, such that on any input in $\{0,1\}^n \subseteq GF(3)$, polynomial $\tilde{f}_i$ produces an output in $\{0,1\}$. (In other words, even though $\tilde{f}_i$ is a polynomial over $GF(3)$, for $0/1$ inputs it takes values in $\{0,1\} \subseteq GF(3)$.) Furthermore, we desire

$$\mathbf{Pr}_{x\in\{0,1\}^n}[f_i(x) \neq \tilde{f}_i(x)] \leq \left(\frac{\text{circuit size}}{2^l}\right) \tag{3.1}$$

We construct the approximating polynomial by induction. The case $h = 0$ is trivial. Suppose we have replaced the output of all nodes up to height h with polynomials of appropriately bounded degree and error. We wish to approximate the output of a node g of height $h+1$ with immediate inputs $f_1, \cdots, f_k$.

1. If g is a NOT gate, we may write $g = \neg f_i$. Then, $\tilde{g} = 1 - \tilde{f}_i$ is an approximating polynomial with the same degree and error as that of f_i.

2. If g is a MOD_3 gate, we use the approximation $\tilde{g} = (\sum_i \tilde{f}_i)^2$. The degree increases by at most a factor of 2, and there is no increase in the error rate.

3. If g is an AND or an OR gate, the required approximation is slightly harder to produce. Suppose $g = \wedge_{i \in I} f_i$. The naive approach approximates an AND node g with a polynomial $\Pi_{i \in I} \tilde{f}_i$. If $g = \vee_{i \in I} f_i$ we use De Morgan's law and similarly obtain the naive approximator $1 - \prod_{i \in I}(1 - \tilde{f}_i)$. Unfortunately, both of these increase the degree of the polynomial by a factor of $|I|$, which could be much larger than the allowed $2l$.

   Let us give the correct solution for OR, leaving the case of AND to yet another application of De Morgan's laws.

   If g is an OR gate, then $g = 1$ if and only if at least one of the $f_i = 1$. We observe that if any of the $f_i = 1$, the sum (over $GF(3)$) of a random subset of $\{f_i\}$ is one with probability at least $1/2$.

   Pick l subsets $S_1, \cdots, S_l$ of $\{1, \cdots, k\}$ randomly. We compute the l polynomials $\sum_{j \in S_i} (\tilde{f}_j)^2$, each of which has degree at most twice that of the largest input polynomial. We then compute the OR of these l terms using the naive approach. The result is a polynomial with degree at most $2l$ times that of the largest input polynomial. For any x, the probability over the choice of subsets that this polynomial differs from $OR(\tilde{f}_1, \cdots, \tilde{f}_k)$ is at most $\frac{1}{2^l}$. So, by the expectation argument, there exists a choice for the l subsets such that the probability over the choice of x that this polynomial differs from $OR(\tilde{f}_1, \cdots, \tilde{f}_k)$ is at most $\frac{1}{2^l}$.

(There was a question in class about how the errors at different gates affect each other. An approximator may introduce some error, but another approximator higher up may introduce another error which cancels this one. The answer is that we are ignoring this issue, and using just the union bound to *upperbound* the probablity that any of the approximator polynomials anywhere in the circuit miscompute.)

This completes the inductive construction of the proper polynomial.

Step 2. Suppose that a polynomial f agrees with the MOD_2 function for all inputs in a set $G' \subseteq 0, 1^n$. If the degree of f is bounded by $\sqrt{n}$, then we show $|G'| < \left(\frac{49}{50}\right) 2^n$.

Consider the change of variables $y_i = 1 + x_i \pmod 3$. (Thus $0 \to 1$ and $1 \to -1$.) Then, G' becomes some subset of $\{-1, 1\}^n$, and f becomes some other polynomial, say $g(y_1, y_2, \ldots, y_n)$, which still has degree $\sqrt{n}$. Moreover,

$$MOD_2(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \Rightarrow \Pi_{i=1}^n y_i = -1 \\ 0 & \Rightarrow \Pi_{i=1}^n y_i = 1 \end{cases} \tag{3.2}$$

Thus $g(y_1, y_2, \ldots, y_n)$, a degree $\sqrt{n}$ polynomial, agrees with $\Pi_{i=1}^n y_i$ on G. This is decidedly odd, and we show that any such G must be small. Specifically, let F_G be the set of all functions $S : G \to \{0, 1, -1\}$. Clearly, $|F_G| = 3^{|G|}$, and we will show $|F_G| \leq 3^{\left(\frac{49}{50}\right) 2^n}$, whence Step 2 follows.

LEMMA 21
For every $S \in F_G$, there exists a polynomial g_S which is a sum of monomials $a_I \prod_{i \in I} y_i$ where $|I| \leq \frac{n}{2} + \sqrt{n}$ such that $g_S(x) = S(x)$ for all $x \in G$.

PROOF: Let $\hat{S} : GF(3)^n \to GF(3)$ be any function which agrees with S on G. Then $\hat{S}$ can be written as a polynomial in the variables y_i. However, we are only interested in its values on $(y_1, y_2, \ldots, y_n) \in \{-1, 1\}^n$, when $y_i^2 = 1$ and so every monomial $a\Pi_{i\in I} y_i^{r_i}$ wlog has $r_i \leq 1$. Thus wlog, $\hat{S}$ is a polynomial of degree at most n. Now consider any of its monomial terms $\Pi_{i\in I} y_i$ of degree $|I| > n/2$. We can rewrite it as

$$\Pi_{i\in I} y_i = \Pi_{i=1}^n y_i \Pi_{i\in \bar{I}} y_i, \tag{3.3}$$

which takes the same values as $g(y_1, y_2, \ldots, y_n)\Pi_{i\in \bar{I}} y_i$ over $\{-1, 1\}^n$. Thus wlog every monomial in $\hat{S}$ has degree at most $\frac{n}{2} + \sqrt{n}$. □

To conclude, we bound the number of polynomials whose every monomial with a degree at most $\frac{n}{2} + \sqrt{n}$. Clearly this number is #polynomials $\leq 3^{\#\text{monomials}}$, and

$$\#\text{monomials} \leq \left| \{N \subseteq \{1 \cdots n\} \mid |N| \leq \frac{n}{2} + \sqrt{n} \right| \tag{3.4}$$

$$\leq \sum_{i \leq \frac{n}{2} + \sqrt{n}} \binom{n}{i} \tag{3.5}$$

Using knowledge of the tails of a binomial distribution,

$$\leq \frac{49}{50} 2^n \tag{3.6}$$

Chapter 4

The Lore and Lure of Expanders

SCRIBE:*Paul Chang*

Now we move to the study of Expander graphs. Their numerous applications to a areas such as error correcting codes, routing networks, sorting networks, derandomization, and *PCP* reductions make them indispensable in any theorist's toolkit.

DEFINITION 4 *An* (α, β) *expander family is a sequence of* d*-regular bipartite graphs on* $2n$ *nodes, (the family has one graph for each* $n \geq 1$*) such that for all* $\alpha\beta < 1$*, each set* S *of nodes on the left where* $|S| \leq \alpha n$ *has at least* $\beta |S|$ *neighbors.*

REMARK 3 If d is a constant independent of n an alternative definition —equivalent, as we will show later using the Alon-Cheeger inequality— is that there is a gap between the largest eigenvalue (namely d) and the second largest eigenvalue of the adjacency matrix.

As shown in an earlier problem set, for every α and β there exists a d such that almost all d-regular bipartite graphs on n nodes are (α, β) expanders. Today we show one example of such graphs for telephone switching networks.

4.1 Example: Lockdown routing/circuit switching.

This example is partially motivated by the study of massively parallel computers and partly by telephone networks. We model a communications network as a graph with N inputs and N outputs. We wish to route calls from the input users to the output users through routers represented by internal nodes. Each call occupies the routers it uses, so separate calls must take vertex disjoint paths. In particular, we would like to be able to route any permutation matching inputs to outputs with decentralized control. In this section we construct a network in which all permutations can be routed. In the next lecture, we will decentralize the control.

REMARK 4 Any network which can route all of the permutations has size $\Omega(NlogN)$, since

$$N! \leq \#states \leq d^{\#nodes} \tag{4.1}$$

DEFINITION 5 *A* Delta network *on* N *inputs is defined recursively: it consists of the first level, which contains the* N *input nodes, and two subnetworks that are Delta networks on* $\frac{N}{2}$ *inputs, and are called the* up *and* down *networks respectively. Each input node has edges to inputs in both up and down networks. Outputs whose address has the MSB (most significant bit) 1 lie in the up subnetwork and the remaining outputs are in the down subnetwork. Clearly, any call to*

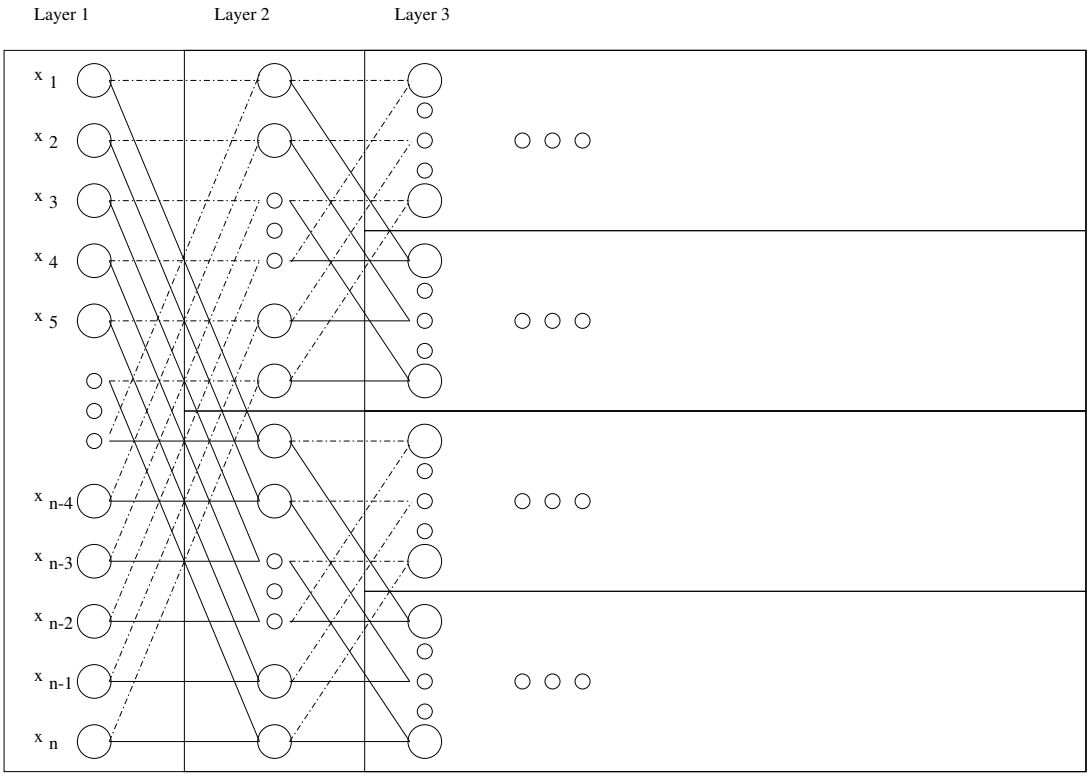

Figure 4.1: An example delta network. Each layer contains a dotted set of edges up and a solid set of edges down. In the multibutterfly, each of these sets forms an expander graph.

a destination whose MSB is 1 *must go from the input node to some node in the up subnetwork. Reasoning similarly for all the levels, we see that the delta network is a layered network in which the up/down decision at level i depends on the ith bit of the destination node. We may route to any of the input nodes of the smaller networks.*

An example of the Delta network is the butterfly*, where each input node* $(b_0 b_1 \ldots b_{\log N})$ *has exactly two outgoing edges, one each to the up and down subnetwork. The edges go to nodes whose label in the that subnetwork is* $(b_1 b_2, \ldots, b_{\log N})$*.*

Butterfly networks cannot route all permutations with vertex disjoint paths. For instance, they function poorly on the transpose permutation. Instead, we present the multibutterfly network of Upfal '88. In the butterfly network, each of N nodes in the input layer is connected to one of the $\frac{N}{2}$ top nodes of the next layer. In the multibutterfly, the degree of that node is d and the subgraph of edges from each of N nodes to the $\frac{N}{2}$ top nodes of the next layer is an (α, β) expander for $\alpha\beta < \frac{1}{2}$.

We assume that the number of inputs (also, number of outputs) is $2\alpha N$ and each is connected to $1/2\alpha$ successive inputs (resp., outputs) of the multibutterfly. (Alternatively, we may still use N for the number of inputs and outputs and then connect them using multibutterflies with $N/2\alpha$ inputs.)

We claim that any permutation on these $2\alpha N$ inputs/outputs can be routed using vertex disjoint paths. We do this layer by layer. At most $\alpha N/2$ calls need to be routed to the top (or bottom) subnetwork. Let S be the set of nodes at which these calls originate. We can route these calls in a vertex disjoint manner if there exists a matching between this subset S of input

nodes and any subset S' of nodes in the top network. To see that this can be done, we apply Hall's theorem, noting that the expander property guarantees that S has at least $|S|$ neighbors.

THEOREM 22
Hall's Theorem. *If $G = (V_1, V_2, E)$ is a bipartite graph, there exists a perfect matching involving V_1 if and only if for all subsets $S \subseteq V_1$, $|\Gamma(S)| \geq |S|$.*

PROOF: Apply the max-flow min-cut theorem. □

Repeating for all levels we obtain our vertex disjoint paths.

4.2 Circuit switching networks - wrap up

scribe: Elad Hazan

Last time we saw a network which enables the routing of the input set $\{1, ..., N\}$ into any permutation on the inputs using vertex disjoint paths. We only showed the *existence* of these paths. Now we describe a distributed algorithm that finds these paths in $O(\log^2 N)$ time.

Recall that in the network there were $\log N$ "layers", each one consisting of N nodes, that together with either the consecutive or previous layer constitute a β-expander (see definition previous lecture, or below) for some $\beta > \frac{d}{2}$ (where d is the degree of each node regarding edges between two consecutive layers). Also, recall that the network was "lightly loaded," and only one out of every $1/2\alpha$ successive inputs/outputs are involved in a call.

Suppose the number of calls is $2\alpha N$. At the first layer, αN wish to enter the "up" subnetwork and αN wish to enter the "down" subnetwork. Wlog let us concentrate on the αN calls that wish to go up, and describe how to set up in $O(\log N)$ time a perfect matching between those nodes and some set of αN nodes in the up subnetwork. Repeating this matching algorithm at each subsequent level gives a total running time of $O(\log^2 N)$ (note that the matching in the up and down networks do not interfere with each other and can go on in parallel).

Here is the matching algorithm. Let $G = (L, R, E)$ be a bipartite d-regular graph that is an (α, β) expander, and let $S \subseteq L$ be a set of at most αN nodes that wish to match to nodes in R. Let $S_1 = S$. The algorithm proceeds in phases; the following is phase i.

1. Every node in S_i sends a "proposal" to every neighbor in R.
2. Every node on R that gets a single proposal accepts.
3. Every node on L that recieves an acceptance, matches to any one of the nodes in R that accepted it, and then drops out. All remaining nodes constitute S_{i+1}.
4. Repeat till all vertices of L are matched.

The next claim shows that $|S_i|$ decreases by a constant factor after every phase, and so in $O(\log N)$ phases it falls below 1, that is, it becomes 0.
Claim: $\frac{|S_{i+1}|}{|S_i|} \leq 2(1 - \beta/d)$.

In phase i, let n_j be the number of vertices from R that get exactly i proposals. Since G is a d-regular graph, we have:

$$|S_i| \cdot d \geq n_1 + 2n_2 + 3n_3 + ... = \sum_{k=1}^{\infty} k \cdot n_k \geq n_1 + 2\sum_{k=2}^{\infty} n_k$$

Since G is an (α, β)-expander:

$$|\Gamma(S)| = n_1 + n_2 + ... = \sum_{k=1}^{\infty} n_k \geq \beta|S|$$

Combining both:

$$n_1 = 2[\sum_{k=1}^{\infty} n_k] - [n_1 + 2\sum_{k=2}^{\infty} n_k] \geq (2\beta - d)|S|.$$

This is the number of nodes in R that send acceptances. Any node in S_i can receive at most d acceptances, so the number that drop out is at least n_1/d. Thus $|S_{i+1}| \leq |S_i| - n_1/d$ and the claim follows.

REMARK 5 This simple algorithm only scratches the surface of what is possible. One can improve the algorithm to run in $O(\log N)$ time, and furthermore, route calls in a *nonblocking* fashion. This means that callers can make calls and hang up any number of times and in any (adversarially determined) order, but still every unused input can call any unused output and the call is placed within $O(\log N)$ steps using local control. The main idea in proving the nonblocking is to treat busy nodes in the circuit —those currently used by other paths— as faulty, and to show that the remaining graph/circuit still has high expansion. See the paper by Arora, Leighton, Maggs and an improvement by Pippenger that requires expansion much less than $d/2$.

Chapter 5

Eigenvalues and Expanders

SCRIBE: *Elad Hazan*

5.1 Spectral properties of graphs and expanders

5.1.1 Basic facts from linear algebra

We begin by stating several definitions and results from linear algebra:
Let $M \in \Re^{n \times n}$ be a square symmetric matrix of n rows and columns.

DEFINITION 6 *An **eigenvalue** of M is a scalar $\lambda \in \Re$ such that exists a vector $x \in \Re^n$ for which $M \cdot x = \lambda \cdot x$. The vector x is called the **eigenvector** corresponding to the eigenvalue λ. (The multiset of eigenvalues is called the* spectrum*.)*

Facts about eigenvalues and eigenvectors of symmetric matrices over $\Re$:

1. M has n real eigenvalues denoted $\lambda_1 \leq ... \leq \lambda_n$. The eigenvectors associated with these eigenvalues form an orthogonal basis for the vector space $\Re^n$ (for any two such vectors the inner product is zero and all vectors are linear independent).

2. The smallest eigenvalue satisfies:

$$\lambda_1 = \min_{x \in R^n, x \neq 0} \frac{x^T M x}{x^T x}$$

   Denote the eigenvector corresponding to λ_i as x_i. Denote the vector space of all vectors in R^n that are orthogonal to x_1 as: $W(x_1) := \Re^n \setminus span\{x_1\}$. Then the second smallest eigenvalues satisfies:

$$\lambda_2 = \min_{x \in W(x_1)} \frac{x^T M x}{x^T x}$$

   If we denote $W(x_1, ..., x_{k-1}) := \Re^n \setminus span\{x_1, ..., x_{k-1}\}$, then the k'th smallest eigenvalue is:

$$\lambda_k = \min_{x \in W(x_1, ..., x_{k-1})} \frac{x^T M x}{x^T x}$$

   This characterization of the spectrum is called the *Courant Fisher Theorem*.

3. Denote by $Spec(M)$ the spectrum of matrix M, that is the multi-set of its eigenvalues. Then for a block diagonal matrix M, that is, a matrix of the form:
$$M = \begin{bmatrix} A & 0 \\ 0 & B \end{bmatrix}$$
The following holds: $Spec(M) = Spec(A) \cup Spec(B)$

4. Eigenvalues of a matrix can be computed in polynomial time. (Eigenvalues are the roots of the characteristic polynomial of a matrix).

5. The Interlacing theorem:
A matrix B is denoted a **principal minor** of matrix M if it can be obtained from M by deleting $k < n$ columns and k rows.
Let $A \in \Re^{(n-1)\times(n-1)}$ be a principal minor of the matrix M. Let:
$$Spec(A) = \{\mu_1 \leq ... \leq \mu_{n-1}\}$$
Then:
$$\lambda_1 \leq \mu_1 \leq \lambda_2 \leq \mu_2 \leq ... \leq \mu_{n-1} \leq \lambda_n$$

5.1.2 Matrices of Graphs

The most common matrix associated with graphs in literature is the **adjacency matrix**. For a graph $G = (V, E)$, the adjacency matrix $A = A_G$ is defined as:
$$A_{i,j} = \begin{cases} 1 & (i,j) \in E \\ 0 & \text{otherwise} \end{cases}$$

Another common matrix is the **Laplacian** of a graph, denoted $\mathcal{L}_G = \mathcal{L}$, and defined as:
$$\mathcal{L}_{i,j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \text{ and } (i,j) \notin E \\ -\frac{1}{\sqrt{d_i d_j}} & i \neq j \text{ and } (i,j) \in E \end{cases}$$
(where d_i is the degree of the node $v_i \in V$)
Notice that if the graph G is d-regular, then its matrices satisfy $A_G = d(I - \mathcal{L}_G)$. Denote by $\{\lambda_i\}$ the eigenvalues of A and by $\{\mu_i\}$ the eigenvalues of $\mathcal{L}$. Then the previous relation implies: $\lambda_i = d(1 - \mu_i)$.
Fact: for any graph, the laplacian is a positive semi-definite matrix, that is, for any vector $y \in R^n$:
$$\forall y \in R^n \; . \; y^T \mathcal{L} y \geq 0$$
(or equivalently, all eigenvalues are nonnegative).

Example - The cycle

Consider a cycle on n nodes. The laplacian is:
$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & -\frac{1}{2} \\ \vdots & & \ddots & \vdots \\ 0 & -\frac{1}{2} & \cdots & 1 \end{bmatrix}$$

This matrix has 1's on the diagonal, and 0's or $-\frac{1}{2}$ elsewhere, depending on whether the indices correspond to an edge. Since a cycle is 2-regular, in each row and column there are exactly two entries with $-\frac{1}{2}$.

CLAIM 1 The all ones vector $\vec{1}$ is an eigenvector of the laplacian of the n-node cycle, corresponding to eigenvalue 0.

PROOF: Observe that since every row has exactly two $-\frac{1}{2}$ then:

$$\mathcal{L} \cdot \vec{1} = 0 = 0 \cdot \vec{1}$$

□ In fact, we can characterize all eigenvalues of the cycle:

CLAIM 2 The eigenvalues of the laplacian of the n-node cycle are:

$$\{1 - \cos\frac{2\pi k}{n} \ , \ k = 0, 1, ..., n-1\}$$

PROOF: Observe that if the vertices are named consecutively, then each vertex i is connected to $i-1, i+1 \mod n$. Therefore, a value λ is an eigenvalue with an eigenvector $\vec{x}$ if and only if for every index of $\vec{x}$:

$$x_i - \frac{1}{2}(x_{i-1} + x_{i+1}) = \lambda \cdot x_i$$

(where sums are modulo n)
It remain to show the eigenvectors. For eigenvalue $\lambda_k = 1 - \cos\frac{2\pi k}{n}$ we associate the eigenvector $\vec{x^k}$ with coordinates:

$$x_i^k = \cos\frac{2\pi ik}{n}$$

And indeed (recall the identity $\cos x + \cos y = 2\cos\frac{x+y}{2}\cos\frac{x-y}{2}$):

$$\begin{aligned} x_i^k - \frac{1}{2}(x_{i-1}^k + x_{i+1}^k) &= \cos\frac{2\pi ik}{n} - \frac{1}{2}\left(\cos\frac{2\pi(i+1)k}{n} + \cos\frac{2\pi(i-1)k}{n}\right) \\ &= \cos\frac{2\pi ik}{n} - \frac{1}{2}\left(2\cos\frac{2\pi ik}{n}\cos\frac{2\pi k}{n}\right) \\ &= \cos\frac{2\pi ik}{n}\left(1 - \cos\frac{2\pi k}{n}\right) \\ &= x_i^k \cdot \lambda_k \end{aligned}$$

□

5.1.3 Expansion and spectral properties

In this section we state the connection between expansion of a graph (defined below) and the eigenvalues of its characteristic matrices.

DEFINITION 7 *Let $G = (V, E)$ be a graph. For any subset of vertices $S \subseteq V$ define its **volume** to be:*

$$Vol(S) := \sum_{i \in S} d_i$$

For a subset of vertices $S \subseteq V$ denote by $E(S, \overline{S})$ the set of edges crossing the cut defined by S. Using the above definition we can define edge expansion of a graph:

DEFINITION 8 *The **Cheeger constant** of G is:*

$$h_G := \min_{S \subseteq V} \frac{|E(S,\overline{S})|}{\min\{Vol(S), Vol(\overline{S})\}} = \min_{S \subseteq V, Vol(S) \leq |E|} \frac{|E(S,\overline{S})|}{Vol(S)}.$$

(N.B. This number is closely related to the conductance *of the graph.)*

The vertex expansion is defined as:

DEFINITION 9 *A graph $G = (V,E)$ is a c-**expander** if for every subset of vertices of cardinality less then half of the vertices, the number of vertices in the set of neighbors is at least c times the original set. Formally:*

$$\forall S \subseteq V, |S| < \frac{1}{2}|V| \;.\; |\Gamma(S)| \geq c \cdot |S|$$

Denote by c_G the maximal constant c for which the graph G is a c-expander.

Observe that for a d-regular graph G (or if d denotes the average degree in *any* graph):

$$h_G \leq c_G \leq d \cdot h_G$$

We now arrive at the main theorem of the lesson, describing the connection between eigenvalues of the Laplacian and the edge expansion:

THEOREM 23 (CHEEGER-ALON)
If λ is the second smallest eigenvalue of the Laplacian of the graph G, then:

$$\frac{h_G^2}{2} \leq \lambda \leq 2h_G$$

PROOF:[first part] We first prove that $\lambda \leq 2h_G$.
We begin with a small claim regarding the smallest eigenvalue:

CLAIM 3 *G has an eigenvalue 0, corresponding to the eigenvector $\vec{w}$ with coordinates:*

$$w_i = \sqrt{d_i}$$

PROOF: By straightforward calculation. Denote by $\vec{b}$ the vector:

$$\vec{b} := \mathcal{L} \cdot w$$

The i'th coordinate of $\vec{b}$ is:

$$b_i = w_i - \sum_{j \in \Gamma(i)} \frac{w_j}{\sqrt{d_i d_j}} = \sqrt{d_i} - \sum_{j \in \Gamma(i)} \frac{1}{\sqrt{d_i}} = 0$$

Hence $\vec{b}$ is the zero vector, and thus 0 is an eigenvalue corresponding to the above eigenvector.
□

Using the previous claim, we can write an explicit expression for the second smallest eigenvalue:

$$\lambda = \min_{x : \Sigma_i \sqrt{d_i} x_i = 0} \frac{x^T \mathcal{L} x}{x^T x} \tag{5.1.1}$$

Using the identity: $z^T M x = \sum_{i,j} z_i M_{i,j} x_j$:

$$= \min_{x:\Sigma_i \sqrt{d_i} x_i = 0} \frac{1}{\sum_i x_i^2} \left[\sum_i \left(x_i^2 - \sum_{j \in \Gamma(i)} \frac{x_i x_j}{\sqrt{d_i d_j}} \right) \right] \tag{5.1.2}$$

Now substitute $y_i := \frac{x_i}{\sqrt{d_i}}$:

$$= \min_{\Sigma_i d_i y_i = 0} \frac{\left(\sum_i d_i y_i^2 - \sum_{j \in \Gamma(i)} y_i y_j \right)}{\sum_i d_i y_i^2} \tag{5.1.3}$$

$$= \min_{\Sigma_i d_i y_i = 0} \frac{\sum_{(i,j) \in E} \left(y_i - y_j \right)^2}{\sum_i d_i y_i^2} \tag{5.1.4}$$

(Aside: This characterization of the second eigenvalue is worth keeping in mind.)

Now let $S \subseteq V$ so that $Vol(S) \leq |E|$ (note that $Vol(V) = 2|E|$). Fix $\vec{a}$ to be with coordinates:

$$a_i = \begin{cases} \frac{1}{Vol(S)} & i \in S \\ -\frac{1}{Vol(\overline{S})} & i \notin S \end{cases}$$

Notice that $\vec{a}$ is legal as:

$$\sum_i d_i a_i = \sum_{i \in S} \frac{d_i}{Vol(S)} - \sum_{i \notin S} \frac{d_i}{Vol(\overline{S})} = \frac{Vol(S)}{Vol(S)} - \frac{Vol(\overline{S})}{Vol(\overline{S})} = 0$$

Now, according to the last expression obtained for λ we get:

$$\begin{aligned} \lambda &\leq \frac{\sum_{(i,j) \in E} \left(a_i - a_j \right)^2}{\sum_i d_i a_i^2} \\ &= \frac{\left(\frac{1}{Vol(S)} + \frac{1}{Vol(\overline{S})} \right)^2 \cdot E(S, \overline{S})}{\sum_{i \in S} \frac{d_i}{Vol(S)^2} + \sum_{i \notin S} \frac{d_i}{Vol(\overline{S})^2}} \\ &= \left(\frac{1}{Vol(S)} + \frac{1}{Vol(\overline{S})} \right) \cdot E(S, \overline{S}) \\ &\leq \frac{2}{Vol(S)} \cdot E(S, \overline{S}) \end{aligned}$$

And since this holds for any $S \subseteq V$, it specifically holds for the set that minimizes the quantity in Cheeger's constant, and we get:

$$\lambda \leq 2 h_G$$

□

Before we proceed to the more difficult part of the theorem, we recall the Cauchy-Schwartz inequality: if $a_1, ..., a_n \in R$; $b_1, ..., b_n \in R$, then:

$$\sum_{i=1}^{n} a_i b_i \leq \left(\sum_i a_i^2 \right)^{\frac{1}{2}} \cdot \left(\sum_i b_i^2 \right)^{\frac{1}{2}} \tag{5.1.5}$$

PROOF:[second part] Let $\vec{y}$ be the vector so that:

$$\lambda = \frac{\sum_{(i,j)\in E}\left(y_i - y_j\right)^2}{\sum_i d_i y_i^2}$$

Define two vectors $\vec{u}, \vec{v}$ with coordinates:

$$u_i = \begin{cases} -y_i & y_i < 0 \\ 0 & \text{otherwise} \end{cases}$$

$$v_i = \begin{cases} y_i & y_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Observe that:

$$(y_i - y_j)^2 \geq (u_i - u_j)^2 + (v_i - v_j)^2$$

And hence:

$$\lambda \geq \frac{\sum_{(i,j)\in E}[(u_i - u_j)^2 + (v_i - v_j)^2]}{\sum_i d_i(u_i^2 + v_i^2)}$$

Since $\frac{a+b}{c+d} \geq \min\{\frac{a}{c}, \frac{b}{d}\}$ it suffices to show that:

$$\frac{\sum_{(i,j)\in E}(u_i - u_j)^2}{\sum_i d_i u_i^2} \geq \frac{h_G^2}{2}$$

Now comes the mysterious part (at least to Sanjeev): multiply and divide by the same quantity.

$$\begin{aligned} \frac{\sum_{(i,j)\in E}(u_i - u_j)^2}{\sum_i d_i u_i^2} &= \frac{\sum_{(i,j)\in E}(u_i - u_j)^2}{\sum_i d_i u_i^2} \times \frac{\sum_{(i,j)\in E}(u_i + u_j)^2}{\sum_{(i,j)\in E}(u_i + u_j)^2} \\ &\geq \frac{[\sum_{(i,j)\in E}(u_i - u_j)^2][\sum_{(i,j)\in E}(u_i + u_j)^2]}{\sum_i d_i u_i^2 \cdot 2\sum_{(i,j)\in E}(u_i^2 + u_j^2)} \end{aligned}$$

Where the last inequality comes from $(a+b)^2 \leq 2(a^2+b^2)$. Now using Cauchy-Schwartz in the numerator:

$$\begin{aligned} &\geq \frac{[\sum_{(i,j)\in E}(u_i - u_j)(u_i + u_j)]^2}{2(\sum_i d_i u_i^2)^2} \\ &= \frac{[\sum_{(i,j)\in E}(u_i^2 - u_j^2)]^2}{2(\sum_i d_i u_i^2)^2} \end{aligned}$$

Now denote by $S_k = \{v_1, ..., v_k\} \subseteq V$ the set of the first k vertices. Denote by C_k the size of the cut induced by S_k:

$$C_k := |E(S_k, \overline{S_k})|$$

Now, since $u_i^2 - u_j^2 = u_i^2 - u_{i+1}^2 + u_{i+1}^2 - u_{i+1}^2 ... + u_{j-1}^2 - u_i^2$, we can write:

$$\sum_{(i,j)\in E}(u_i^2 - u_j^2) = \sum_k (u_k^2 - u_{k+1}^2) \cdot C_k$$

And thus, returning to the chain of inequalities we get:

$$\begin{aligned}\frac{\sum_{(i,j)\in E}(u_i-u_j)^2}{\sum_i d_i u_i^2} &\geq \frac{[\sum_{(i,j)\in E}(u_i^2-u_j^2)]^2}{2(\sum_i d_i u_i^2)^2} \\ &= \frac{[\sum_k (u_k^2-u_{k+1}^2)\cdot C_k]^2}{2(\sum_i d_i u_i^2)^2}\end{aligned}$$

According to the definition of h_G we know that $C_k \geq h_G \cdot (\sum_{i\leq k} d_i)$ (as h_G is the minimum of a set of expressions containing these). Hence:

$$\begin{aligned}&\geq \frac{[\sum_k (u_k^2-u_{k+1}^2)\cdot h_G\cdot(\sum_{i\leq k} d_i)]^2}{2(\sum_i d_i u_i^2)^2} \\ &= h_G^2\cdot\frac{[\sum_k (u_k^2-u_{k+1}^2)\cdot(\sum_{i\leq k} d_i)]^2}{2(\sum_i d_i u_i^2)^2} \\ &= h_G^2\cdot\frac{(\sum_k d_k u_k^2)^2}{2(\sum_i d_i u_i^2)^2} \\ &= \frac{h_G^2}{2}\end{aligned}$$

And this concludes the second part of the theorem. □

REMARK 6 Note that we proved the stronger result that actually one of the cuts C_k satisfies $\frac{E(S_k,\overline{S_k})}{\min Vol(S_k),Vol(\overline{S_k})} \leq \sqrt{2\lambda}$. Namely, the algorithm to find a sparse cut is to take the eigenvector $(y_1, y_2, \ldots, y_n)$ corresponding to λ, and check all the n cuts of the type $S_k = \{i : x_i \leq x_k\}$.

Chapter 6

Markov Chains and Random Walks

SCRIBE:*Elena Nabieva*

6.1 Basics

A *Markov chain* is a discrete-time stochastic process on n states defined in terms of a transition probability matrix (M) with rows i and columns j.

$$\mathbf{M} = \left(P_{ij}\right)$$

A transition probability P_{ij} corresponds to the probability that the state at time step $t+1$ will be j, given that the state at time t is i. Therefore, each row in the matrix $\mathbf{M}$ is a distribution and $\forall i, j \in S.P_{ij} \geq 0$ and $\sum_j P_{ij} = 1$.

We stress that the evolution of a Markov chain is *memoryless*: the transition probability P_{ij} depends only on the state i and not on the time t or the sequence of transititions taken before this time.

One way to think about Markov chains is of a certain amount fluid sitting at each node (corresponding to the initial distribution). At every step, the fluid sitting at node i distributes to its neighbors, such that P_{ij} fraction goes to j.

Let the initial distribution be given by the row vector $\mathbf{x} \in \mathbf{R}^n$, $x_i \geq 0$ and $\sum_i x_i = 1$. After one step, the new distribution is $\mathbf{xM}$. After t steps, the distribution is $\mathbf{xM}^t$.

DEFINITION 10 *A distribution π for the Markov chain $\mathbf{M}$ is a* stationary distribution *if $\pi\mathbf{M} = \pi$.*

DEFINITION 11 *A Markov chain $\mathbf{M}$ is* erdodic *if there exists a unique stationary distribution π and for every (initial) distribution $\mathbf{x}$ the limit $\lim_{t\to\infty} \mathbf{xM}^t = \pi$.*

THEOREM 24
The following are necessary and sufficient conditions for ergodicity:

1. *connectivity: $\forall i, j.\mathbf{M}^t(i,j) > 0$ for some t.*
2. *aperiodicity: $\forall i.gcd\{t : \mathbf{M}^t(i,j) > 0\} = 1$.*

REMARK 7 Clearly, these conditions are necessary. If the Markov chain is disconnected it cannot have a unique stationary distribution. Similarly, a bipartite graph does not have a unique distribution (if the initial distribution places all probability on one side of the bipartite graph, then the distribution at time t oscillates between the two sides depending on whether t is odd

or even). Note that in a bipartite graph $gcd\{t : \mathbf{M}^t(i,j) > 0\} \geq 2$. The sufficiency of these conditions is proved using eigenvalue techniques (for inspiration see the analysis of mixing time later on).

Both conditions are easily satisfied in practice. In particular, any Markov chain can be made aperiodic by adding self-loops assigned probability $1/2$.

DEFINITION 12 *An erdodic Markov chain is* reversible *if the stationary distribution π satisfies for all i, j, $\pi_i \mathbf{P}_{ij} = \pi_j \mathbf{P}_{ji}$.*

6.2 Mixing Times

Informally, the *mixing time* of a Markov chain is the time it takes to reach "nearly uniform" distribution from any arbitrary starting distribution.

DEFINITION 13 *The mixing time of an ergodic Markov chain M is t if for every starting distribution x, the distribution xM^t satisfies $\left| xM^t - \pi \right|_1 \leq 1/4$. (Here $|\cdot|_1$ denotes the ℓ_1 norm and the constant "$1/4$" is arbitrary.)*

The next exercise clarifies why we are interested in ℓ_1 norm.

EXERCISE 2 For any distribution π on $\{1,2,\ldots,N\}$, and $S \subseteq \{1,2,\ldots,N\}$ let $\pi(S) = \sum_{i \in S} \pi_i$. Show that for any two distributions π, π',

$$\left| \pi - \pi' \right|_1 = 2 \max_{S \subseteq \{1,\ldots,N\}} \left| \pi(S) - \pi'(S) \right|. \tag{6.2.1}$$

Here is another way to restate the property in (6.2.1). Suppose A is some deterministic algorithm (we place no bounds on its complexity) that, given any number $i \in \{1,2,\ldots,N\}$, outputs Yes or No. If $|\pi - \pi'|_1 \leq \epsilon$ then the probability that A outputs Yes on a random input drawn according to π cannot be too different from the probability it outputs Yes on an input drawn according to π'. For this reason, ℓ_1 distance is also called *statistical difference*.

We are interested in analysing the mixing time so that we can draw a sample from the stationary distribution.

EXAMPLE 6 (Mixing time of a cycle) Consider an *n-cycle*, i.e., a Markov chain with n states where, at each state, $\mathbf{Pr}(left) = \mathbf{Pr}(right) = \mathbf{Pr}(stay) = 1/3$.

Suppose the initial distribution concentrates all probability at state 0. Then t steps correspond to about $2t/3$ random coin tosses and the index of the final state is

$$(\#(\text{Heads}) - \#(\text{Tails})) \pmod{n}.$$

Clearly, it takes $\Omega(n^2)$ steps for the walk to reach the other half of the circle with any reasonable probability, and the mixing time is $\Omega(n^2)$. We will later see that this lowerbound is fairly tight.

6.2.1 A Motivation: Approximate Counting and Sampling

Markov chains allow one to sample from non-trivial sets. As a motivating example consider the following problem. Given $a_1 \ldots a_n, b \in \mathbf{Z}^+$, compute the number of vectors $(x_i,\ldots,x_n)$ s.t. $\sum a_i x_i \leq b$. In other words, count the number of feasible solutions to the 0-1 knapsack problem.

Remark: this problem is in #**P** (the complexity class corresponding to counting the number of solutions to an **NP** problem). In fact, we have.

THEOREM 25 (VALIANT, 1970S)
If there exist a polynomial-time algorithm for solving the knapsack counting problem, then $\mathbf{P} = \mathbf{NP}$ *(in fact,* $\mathbf{P} = \mathbf{P}^{\#\mathbf{P}}$*).*

Valiant's Theorem does not rule out finding good approximations to this problem.

DEFINITION 14 *A* Fully Polynomial Randomized Approximation Scheme *(FPRAS) is a randomized algorithm, which for any* ϵ *finds an answer in time polynomial in* $(\frac{n}{\epsilon}\log\frac{1}{\delta})$ *that is correct within a multiplicative factor (1+*ϵ*) with probability (1-*δ*).*

THEOREM 26 (JERRUM, VALIANT, VAZIRANI, 1984)
If we can sample almost uniformly, in polynomial time, from $A = \{(x_1, \ldots, x_n) \colon \sum a_i x_i \leq b\}$*, then we can design an FPRAS for the knapsack counting problem.*

REMARK 8 By "sampling almost uniformly" we mean having a sampling algorithm whose output distribution has ℓ_1 distance $\exp(-n^2)$ (say) from the uniform distribution. For ease of exposition, we think of this as a uniform sample.

PROOF: Suppose we have a sampling algorithm for knapsack. We know that either $\mathbf{Pr}_{\mathbf{x}\in A}(x_1 = 1)$ or $\mathbf{Pr}_{\mathbf{x}\in A}(x_1 = 0)$ is at least $1/2$. After generating a few samples we can determine say that the former holds, and say $\mathbf{Pr}_{\mathbf{x}\in A}(x_1 = 1) \in [p_1(1+\gamma)^{-1}, p_1(1+\gamma)]$.

Let $A_1 = \{(x_2, \ldots, x_n) \colon \sum_{i=2}^{n} x_i a_i \leq b - a_1\}$.

Then

$$p_1(1+\gamma) \geq \frac{|A_1|}{|A|} \geq p_1(1+\gamma)^{-1}.$$

Note that A_1 is also the set of solutions to some knapsack problem, so we can use our algorithm on it and determine say that $\mathbf{Pr}_{\mathbf{x}\in A_1}(x_2 = 0) \in [p_2(1+\gamma)^{-1}, p_2(1+\gamma)]$. Thus if $A_2 = \{(x_3, \ldots, x_n) \colon \sum_{i=3}^{n} x_i a_i \leq b - a_1\}$ we have shown that

$$p_2(1+\gamma) \geq \frac{|A_2|}{|A_1|} \geq p_2(1+\gamma)^{-1}.$$

Proceeding this way we can estimate $|A_i| \,/\, |A_{i+1}|$ for each i and thus have an approximation to

$$\prod_{i=0}^{n-1} \frac{|A_{i+1}|}{|A_i|} = \frac{|A_n|}{|A_0|}$$

(where $A_0 = A$) that is correct to within a factor $(1+\gamma)^n \approx (1+n\gamma)$. Since $|A_n|$ is easily computable, we have a good approximation to the size of $A_0 = A$.

Therefore, by choosing ϵ appropriately and using the Chernoff bound, we can achieve the desired bound on the error in polynomial time. (Remark: This is not exactly accurate; there is a little bit more work involved and the details are not trivial.)

This proves one direction. The other direction is similar. If we have an approximate counter, then we can output a random sample in A bit by bit. We can estimate p_1 and output $x_1 = 1$ with probability p_1, and proceed inductively. Details are left as an exercise.

□

Thus to count approximately, it suffices to sample from the uniform distribution. We define a Markov chain M on A whose stationary distribution is uniform. Then we show that its mixing time is poly(n).

The Markov chain is as follows. If the current node is $(x_1, \ldots, x_n)$ (note $a_1x_1 + a_2x_2 + \ldots = a_nx_n \leq b$) then

1. with probability $1/2$ remain at the same node

2. else pick $i \in \{1, \ldots, n\}$.

   Let $\mathbf{y} = (x_1, \ldots, x_{i-1}, 1 - x_i, x_{i+1}, \ldots, x_n)$. If $\mathbf{y} \in A$, go there. Else stay put.

Note that M is

1. aperiodic because of self-loops
2. connected because every sequence can be turned into the zero vector in a finite number of transformations, i.e., every node is connected to $\vec{0}$.

Therefore, M is ergodic, i.e., has a unique stationary distribution. Since the uniform distribution is stationary, it follows that the stationary distribution of M is uniform.

Now the question is: how fast does M converge to the uniform distribution? If M mixes fast, we can get an efficient approximation algorithm for the knapsack counting: we get the solution by running M for the mixing time and sampling from the resulting distribution after the mixing time has elapsed.

THEOREM 27
(Morris-Sinclair, 1999): The mixing time for M is $O(n^8)$.

Fact (see our remark later in our analysis of mixing time): running the M for a bit longer than the mixing time results in a distribution that is extremely close to uniform.

Thus, we get the following sampling algorithm:

1. Start with the zero vector as the initial distribution of M.
2. Run M for $O(n^9)$ time.
3. output the node at which the algorithm stops.

This results in a uniform sampling from A.

Thus Markov chains are useful for sampling from a distribution. Often, we are unable to prove any useful bounds on the mixing time (this is the case for many Markov chains used in simulated annealing and the Metropolis algorithm of statistical physics) but nevertheless in practice the chains are found to mix rapidly. Thus they are useful even though we do not have a proof that they work.

6.3 Bounding the mixing time

For simplicity we restrict attention to regular graphs.

Let M be a Markov chain on a d-regular undirected graph with an adjacency matrix A. Assume that M is ergodic and that d includes any self-loops.

Then, clearly $M = \frac{1}{d}A$.

Since M is ergodic, and since $\frac{1}{n}\vec{1}$ is a stationary distribution, then $\frac{1}{n}\vec{1}$ is the unique stationary distribution for M.

The question is how fast does M convege to $\frac{1}{n}\vec{1}$? Note that if $\mathbf{x}$ is a distribution, $\mathbf{x}$ can be written as

$$\mathbf{x} = \vec{1}\frac{1}{n} + \sum_{i=2}^{n} \alpha_i \mathbf{e_i}$$

where $\mathbf{e_i}$ are the eigenvectors of M which form an orthogonal basis and $\mathbf{1}$ is the first eigenvector with eigenvalue 1. (Clearly, $\mathbf{x}$ can be written as a combination of the eigenvectors; the

observation here is that the coefficient in front of the first eigenvector $\vec{1}$ is $\vec{1} \cdot x / \left|\vec{1}\right|_2^2$ which is $\frac{1}{n}\sum_i x_i = \frac{1}{n}$.)

$$\begin{aligned} M^t\mathbf{x} &= M^{t-1}(M\mathbf{x}) \\ &= M^{t-1}(\frac{1}{n}\vec{1} + \sum_{i=2}^{n} \alpha_i\lambda_i\mathbf{e_i}) \\ &= M^{t-2}(M(\frac{1}{n}\vec{1} + \sum_{i=2}^{n} \alpha_i\lambda_i\mathbf{e_i})) \\ &\cdots \\ &= \frac{1}{n}\vec{1} + \sum_{i=2}^{n} \alpha_i\lambda_i^t\mathbf{e_i} \end{aligned}$$

Also

$$\| \sum_{i=2}^{n} \alpha_i\lambda_i^t\mathbf{e_i}\|_2 \leq \lambda_{max}^t$$

where λ_{max} is the second largest eigenvalue of M. (Note that we are using the fact that the total ℓ_2 norm of any distribution is $\sum_i x_i^2 \leq \sum_i x_i = 1$.)

Thus we have proved $\left|M^t\mathbf{x} - \frac{1}{n}\mathbf{1}\right|_2 \leq \lambda_{max}^t$. Mixing times were defined using ℓ_1 distance, but Cauchy Schwartz inequality relates the ℓ_2 and ℓ_1 distances: $|p|_1 \leq \sqrt{n}\,|p|_2$. So we have proved:

THEOREM 28
The mixing time is at most $O(\frac{\log n}{\lambda_{\max}})$.

Note also that if we let the Markov chain run for $O(k\log n/\lambda_{\max})$ steps then the distance to uniform distribution drops to $\exp(-k)$. This is why we were not very fussy about the constant $1/4$ in the definition of the mixing time earlier.

Finally, we recall from the last lecture: for $S \subset V$, $Vol(S) = \sum_{i\in S} d_i$, where d_i is the degree of node i, the *Cheeger Constant* is

$$h_G = \min_{S\subset V, vol(S)\leq \frac{Vol(V)}{2}} \frac{\left|E(S,\overline{S}\right|}{Vol(S)}$$

If μ is the smallest nonzero eigenvalue of the Laplacian L of M, then

$$2h_G \geq \mu \geq \frac{h_G^2}{2}$$

The Laplacian for our graph is

$$L = I - M$$

Therefore,

$$spec(L) = \{0 = \mu_1 \leq \mu_2 \leq \ldots \leq \mu_n\}$$

and

$$spec(M) = \{1 = 1 - \mu_1 \geq 1 - \mu_2 \geq \ldots \geq 1 - \mu_n\}$$

Note that $\lambda_{max} = (1 - \mu_2)^t$.

Therefore,

$$\| \sum_{i=2}^{n} \alpha_i \lambda_i^t \mathbf{e_i} \|_2 \leq (1 - \frac{h_G^2}{2})^t \sqrt{n}$$

.

and we obtain the Jerrum-Sinclair inequality:
$\|M^t \mathbf{x} - \frac{1}{n}\vec{1}\|_2 \leq (1 - \frac{h_G^2}{2})^t \sqrt{n}$.
Examples:

1. For n-cycle: $\lambda_{max} = (1 - \frac{c}{n^2})^t$, mixing time $\approx O(n^2 \log n)$ (c is some constant).
2. For a hypercube on 2^n nodes (with self-loops added), $\lambda_{max} = (1 - \frac{c}{n})$ (this was a homework problem), so mixing time $\approx O(n \log n)$ (c is some constant).

Observe that the mixing time is much smaller than the number of nodes, i.e., the random walk does not visit all nodes.

Finally, we note that random walks also give a randomized way to check $s - t$ connectivity (for undirected graphs) in logarithmic space, a surprising result since the usual method of checking $s - t$ connectivity, namely, breadth-first-search, seems to inherently require linear space.

The main idea is that a random walk on a connected graph on n nodes mixes in $O(n^4)$ time (the Cheeger constant must be at least $1/n^2$) and so a logarithmic space algorithm can just do a random walk for $O(n^2 \log n)$ steps (note that space $O(\log n)$ is required is just to store the current node and a counter for the number of steps) starting from s, and if it never sees t, it can output reject. This answer will be correct with high probability. This application of random walks by Alleliunas, Karp, Karmarkar, Lipton and Lovasz 1979 was probably one of the first in theoretical computer science and it has been the subject of much further work recently.

6.4 Analysis of Mixing Time for General Markov Chains

Thanks to Satyen Kale for providing this additional note

In the class we only analysed random walks on d-regular graphs and showed that they converge exponentially fast with rate given by the second largest eigenvalue of the transition matrix. Here, we prove the same fact for general ergodic Markov chains. We need a lemma first.

LEMMA 29
Let M be the transition matrix of an ergodic Markov chain with stationary distribution π and eigenvalues $\lambda_1 (= 1) \geq \lambda_2 \geq \ldots \geq \lambda_n$, corresponding to eigenvectors $v_1 (= \pi), v_2, \ldots v_n$. Then for any $k \geq 2$,

$$v_k \vec{1} = 0.$$

PROOF: We have $v_k M = \lambda_k v_k$. Mulitplying by $\vec{1}$ and noting that $M\vec{1} = \vec{1}$, we get

$$v_k \vec{1} = \lambda_k v_k \vec{1}.$$

Since the Markov chain is ergodic, $\lambda_k \neq 1$, so $v_k \vec{1} = 0$ as required. □

We are now ready to prove the main result concerning the exponentially fast convergence of a general ergodic Markov chain:

THEOREM 30
In the setup of the lemma above, let $\lambda = max\{|\lambda_2|, |\lambda_n|\}$. Then for any initial distribution x, we have

$$||xM^t - \pi||_2 \leq \lambda^t ||x||_2.$$

PROOF: Write x in terms of $v_1, v_2, \ldots, v_n$ as

$$x = \alpha_1 \pi + \sum_{i=2}^{n} \alpha_i v_i.$$

Multiplying the above equation by $\vec{1}$, we get $\alpha_1 = 1$ (since $x\vec{1} = \pi\vec{1} = 1$). Therefore $xM^t = \pi + \sum_{i=2}^{n} \alpha_i \lambda_i^t v_i$, and hence

$$\begin{aligned} ||xM^t - \pi||_2 &\leq ||\sum_{i=2}^{n} \alpha_i \lambda_i^t v_i||_2 && (6.4.1) \\ &\leq \lambda^t \sqrt{\alpha_2^2 + \cdots + \alpha_n^2} && (6.4.2) \\ &\leq \lambda^t ||x||_2, && (6.4.3) \end{aligned}$$

as needed. □

6.5 Example of use of expanders: Recycling Random Bits

Scribe: Nir Ailon

Now we describe one more application of expander graphs: recycling random bits.

Assume we have a randomized polynomial time algorithm deciding on a language L, that given an input x answers as follows:

$$\begin{cases} Pr(\text{Answer "yes"}) \geq \frac{1}{2} & \text{if } x \in L \\ Pr(\text{Answer "yes"}) = 0 & \text{if } x \notin L \end{cases} \quad (6.5.1)$$

(Note: the class of such languages is called *co-RP*). We would like to get an amplification of the success probability by repeating the algorithm k times. This would replace the probability $\frac{1}{2}$ with probability $1 - \frac{1}{2^k}$. Assuming that the algorithm requires r random bits for input x, a naive approach would require rk random bits. A result by [AKS],[CW],[IZ] (87-89) says that we can get this using only $O(r + k)$ bits. This seems surprising at first, but from an information theoretic point of view there is a simple intuition: we pay r bits and get 1 "yes"/"no" bit back, so we can pocket the change. The idea is as follows:

Take, say, a degree 5 expander on 2^r nodes (note: there are explicit such constructions that allow computation of $\Gamma(v)$ in poly(r) time for any node v). Recall that one of the equivalent definitions of an expander graph is that there is an eigenvalue gap in the adjacency matrix of the graph. Start with a random node (requires r random bits) and perform a random walk of length k, each time taking the bits corresponding to the visited node as the fresh r random bits. In each step, we need $O(1)$ random bits, and there are a total of k steps, resulting in a total of $O(r + k)$ random bits.

To prove that this works, assume $x \in L$ and denote by C the set of random strings of length r that would cause the algorithm to answer the correct answer, "yes". So $\frac{|C|}{2^r} \geq \frac{1}{2}$. You will show in the Homework 4 that a random walk of length k avoids C with probability at most $O(e^{-O(k)})$.

Chapter 7

High Dimensional Geometry

SCRIBE:*Nir Ailon, Renato F. Werneck*

Now we move on to a study of high-dimensional geometry. By this we mean the study of algorithms for n points in $\Re^d$, where d is large (say $\log n$ or even n). This has become increasingly important recently, both theoretically and practically.

7.1 High Dimensional Geometry: Introduction

Some useful generalizations of geometric objects to higher dimensional geometry:

- The *n-cube* in $\mathbf{R}^n$: $\{(x_1...x_n) : 0 \leq x_i \leq 1\}$. To visualize this in $\Re^4$, think of yourself as looking at one of the faces, say $x_1 = 1$. This is a cube in $\Re^3$ and if you were able to look in the fourth dimension you would see a parallel cube at $x_1 = 0$. The visualization in $\Re^n$ is similar.

  The volume of the n-cube is 1.

- The unit *n-ball* in $\mathbf{R}^n$: $B_n := \{(x_1...x_n) : \sum x_i^2 \leq 1\}$. Again, to visualize the ball in $\Re^4$, imagine you have sliced through it with a hyperplane, say $x_1 = 1/2$. This slice is a ball in $\Re^3$ of radius $\sqrt{1 - 1/2^2} = \sqrt{3}2$. Every parallel slice also gives a ball.

  The volume of B_n is $\frac{\pi^{n/2}}{(n/2)!}$ (assume n even if the previous expression bothers you), which is $\frac{1}{n^{\Theta(n)}}$.

7.1.1 An approximate way to think about B_n

A good approximation to picking a random point on the surface of B_n is by choosing random $x_i \in \{-1, 1\}$ independently for $i = 1..n$ and normalizing to get $\frac{1}{\sqrt{n}}(x_1, ..., x_n)$. To get a point inside the ball, it is necessary to pick the distance from $\tilde{0}$ randomly. Note that the distance is not distributed uniformly: the density at radius r is proportional to r^{n-1}.

Remark: An exact way to pick a random point on the surface of B^n is to choose x_i from the normal distribution for $i = 1..n$, and to normalize: $\frac{1}{l}(x_1, ..., x_n)$, where $l = (\sum_i x_i^2)^{1/2}$.

7.1.2 Funny facts

1. Scribe's contribution: The volume of the unit n-ball tends to 0 as the dimension tends to ∞.

2. For any $c > 1$, a $(1 - \frac{1}{c})$ - fraction of the volume of the n-ball lies in a strip of width $\mathcal{O}(\sqrt{\frac{c \log n}{n}})$. A strip of width a is B_n intersected with $\{(x_1, ..., x_n) | x_1 \in [-\frac{a}{2}, \frac{a}{2}]\}$.

3. If you pick 2 vectors on the surface of B_n independently, then with probability $> 1 - \frac{1}{n}$,

$$|\cos(\Theta_{x,y})| = \mathcal{O}(\frac{\sqrt{\log n}}{n}),$$

where $\Theta_{x,y}$ is the angle between x and y. In other words, the 2 vectors are almost orthogonal w.h.p. To prove this, we use the following lemma:

LEMMA 31

Suppose a is a unit vector in $\mathbf{R}^n$. Let $x = (x_1, ..., x_n) \in R^n$ be chosen from the surface of B_n by choosing each coordinate at random from $\{1, -1\}$ and normalizing by factor $\frac{1}{\sqrt{n}}$. Denote by X the random variable $a \cdot x = \sum a_i x_i$. Then:

$$Pr(|X| > t) < e^{-nt^2}$$

PROOF: We have:

$$\mu = E(X) = E(\sum a_i x_i) = 0$$

$$\sigma^2 = E[(\sum a_i x_i)^2] = E[\sum a_i a_j x_i x_j] = \sum a_i a_j E[x_i x_j] = \sum \frac{a_i^2}{n} = \frac{1}{n}.$$

Using the Chernoff bound, we see that,

$$Pr(|X| > t) < e^{-(\frac{t}{\sigma})^2} = e^{-nt^2}$$

□

COROLLARY 32

If two unit vectors x, y are chosen at random from $\mathbf{R}^n$, then

$$Pr\left(|cos(\theta_{x,y})| > \sqrt{\frac{-\log \epsilon}{n}}\right) < \epsilon$$

Now, to get fact (3), put $\epsilon = \frac{1}{n}$.

7.2 Random Walks in Convex Bodies

We can apply our earlier study of random walks to geometric random walks, and derive bounds on the mixing time using geometric ideas.

Definition of a convex body: A set $K \subset \mathbf{R}^n$ is convex if $\forall x, y \in K, \lambda \in [0, 1]$,

$$\lambda x + (1 - \lambda) y \in K.$$

In other words, for any two points in K, the line segment connecting them is in the set.

examples: Balls, cubes, polytopes (=set of feasible solutions to an LP), ellipsoides etc.

Convex bodies are very important in theory of algorithms. Many optimization problems can be formulated as optimizing convex functions in convex domains. A special case is linear programming.

Today's Goal: To generate a random point in a convex body. This can be used to approximate the volume of the body and to solve convex programs (we saw how to do the latter in Vempala's talk at the theory lunch last week). We emphasize that the description below is missing many details, though they are easy to figure out.

First we need to specify how the convex body is represented:

- $0 \in K$.
- K is contained in a cube of side R with center at 0.
- A unit cube is contained in K.
- $R \leq n^2$.
- there exists a "separation oracle" that given any point $x \in \mathbf{R}^n$ returns "yes" if $x \in K$ and if $x \notin K$ then returns a separating hyperplane $\{y | a^T y = b\}$ s.t. x and K are on opposite sides (i.e. $a^T x < b, a^T z > b \ \forall z \in K$). Note 1: a (closed) convex body can actually be characterized by the property that it can be separated from any external point with a separating hyperplane. Note 2: In the linear program case, giving a separating hyperplane is equivalent to specifying a violated inequality.

The idea is to approximate the convex body K with a fine grid of scale $\delta < \frac{1}{n^2}$. The volume of K is approximately proportional to the number of grid points. There is some error because we are approximating the convex body using a union of small cubes, and the error term is like

$$\text{(number of grid cubes that touch the surface of } K) \times \text{(volume of small cube)},$$

which is $\ll$ volume of K since the minimum crosssection of K (at least 1) is much larger than the dimensions of the grid cube.

Consider the graph G_K whose vertices are grid notes contained in K, and there is an edge between every pair of grid neighbors. To a first approximation, the graph is regular: almost every node has degree $2n$. The exceptions are grid points that lie close to the surface of K, since not their $2n$ neighbors may lie inside K. But as noted, such grid points form a negligible fraction. Thus generating a random point of G_K is a close approximation to generating a random point of K.

However, we only know one node in G_K, namely, the origin 0. Thus one could try the following idea. Start a random walk from 0. At any step, if you're on a grid point x, stay at x with probability $1/2$, else randomly pick a neighbor y of x. If $y \in K$, move to y, else stay at x. If this walk is ergodic, then the stationary distribution will be close to uniform on G_K, as noted. The hope is that this random walk is rapidly mixing, so running it for poly(n) time does yield a fairly unbiased sample from the stationary distribution.

Unfortunately, the walk may not be ergodic since G_K may not be connected (see figure 7.1).

To solve this problem, we smoothen K, namely, we replace K with a ρ-neighborhood of K defined as

$$K' = \bigcup_{x \in k} B_n(x, \rho),$$

where $B_n(x, \rho)$ is a closed ball centered at x with radius ρ, and $\rho \approx \frac{1}{n}$ (see figure 7.2). It can be checked that this negligibly increases the volume while ensuring the following two facts:

- K' is convex, and a separation oracle for K' can be easily built from the separation oracle for K, and,
- Any 2 grid points in K are connected via a path of grid points in K'.

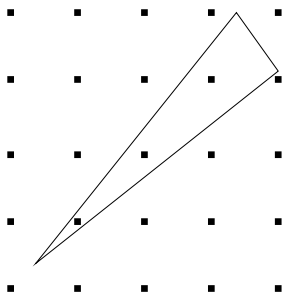

Figure 7.1: The bottom-left most grid point in the triangle is isolated.

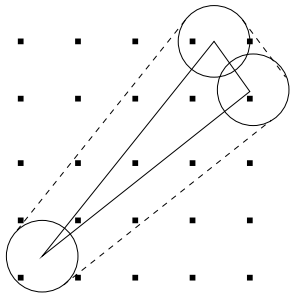

Figure 7.2: The dashed convex body is obtained by smoothening the solid triangle.

Now let $G_{K'}$ be the graph on grid nodes contained in K'. It is connected, and almost all its nodes are in K (since the volume increase in going from K to K' was minimal).

We go back to estimating the mixing time. Recall that the "mixing time" is the time t s.t. starting from any distribution x_0, after t steps the distribution x_t satisfies

$$\|x_t - \pi\|_2 \leq \frac{1}{4}.$$

Using the Jerrum-Sinclair inequality from the previous lecture, we have

$$\|x_t - \pi\|_2 \leq (1 - \frac{\phi^2}{2})^t \sqrt{N},$$

where ϕ is the Cheeger number attached to the graph and N is the size of the graph (number of grid points in K'). So we reduce to the problem of estimating the Cheeger constant.

[Aside: In case you ever go through the literature, you will see that many papers use the "conductance" of the graph instead of the "Cheeger" number. The conductance of a graph G is

$$\min_{S \subset V} \frac{|E(S, \bar{S})| \mathrm{Vol}(V)}{\mathrm{Vol}(S)\mathrm{Vol}(\bar{S})}.$$

Easy exercise: the conductance is within factor 2 of the Cheeger number.]

To estimate the Cheeger number of the grid graph, we use a result by Lovasz and Simonovits. Let U be a subset of nodes of $G_{K'}$. Consider the union of the corresponding set of grid cubes; this is a measurable subset (actually it is even better behaved; it is a finite union of convex subsets) of K. Call this set S. Let ∂S be the boundary of S. Then it is not hard to see that:

$$\mathrm{Vol}_{n-1}(\partial S) \leq E(S, \bar{S}) \leq 2n \mathrm{Vol}_{n-1}(\partial S),$$

where $E(S, \bar{S})$ is the number of grid edges crossing between S and $\bar{S}$. The [LS] result is:

THEOREM 33
If K is a convex body containing a unit cube, then for any measurable $S \subset K$,

$$Vol_{n-1}(\partial S) \geq \frac{2}{D} \min\{Vol_n(S), Vol_n(\bar{S})\},$$

where D is the diameter of K.

Recall that the diameter of our body is poly(n) since we started with a body whose diameter was n^2 and then we placed a grid of size $1/n^2$ or so. Combining all the above information, we get that

$$\phi \geq \frac{1}{\text{poly}(n)}.$$

Therefore, the mixing time is $\mathcal{O}(\frac{1}{\phi^2} \log N) = \mathcal{O}(\text{poly}(n))$.

7.3 Dimension Reduction

Now we describe a central result of high-dimensional geometry (at least when distances are measured in the ℓ_2 norm). Problem: Given n points $z^1, z^2, ..., z^n$ in $\Re^n$, we would like to find n points $u^1, u^2, ..., u^n$ in $\Re^m$ where m is of low dimension (compared to n) and the metric restricted to the points is almost preserved, namely:

$$\|z^i - z^j\|_2 \leq \|u^i - u^j\|_2 \leq (1+\epsilon)\|z^j - z^j\|_2 \ \forall i, j. \tag{7.3.1}$$

The following main result is by Johnson & Lindenstrauss :

THEOREM 34
In order to ensure (7.3.1), $m = \mathcal{O}(\frac{\log n}{\epsilon^2})$ suffices.

Note: In class, we used the notation of k vectors $z^1 ... z^k$ in $\Re^n$, but we can always embed the k vectors in a k-dimensional space, so here I assume that $n = k$ and use only n.
PROOF:

Choose m vectors $x^1, ..., x^m \in \Re^n$ at random by choosing each coordinate randomly from $\{\sqrt{\frac{1+\epsilon}{m}}, -\sqrt{\frac{1+\epsilon}{m}}\}$. Then consider the mapping from $\Re^n$ to $\Re^m$ given by

$$z \longrightarrow (z \cdot x^1, z \cdot x^2, \ldots, z \cdot x^m).$$

In other words $u^i = (z^i \cdot x^1, z^i \cdot x^2, ..., z^i \cdot x^m)$ for $i = 1, \ldots, k$. We want to show that with positive probability, $u^1, ..., u^k$ has the desired properties. This would mean that there exists at least one choice of $u^1, ..., u^k$ satisfying inequality 7.3.1. To show this, first we write the expression $\|u^i - u^j\|$ explicitly:

$$\|u^i - u^j\|^2 = \sum_{k=1}^{m} \left(\sum_{l=1}^{n} (z_l^i - z_l^j) x_l^k \right)^2.$$

Denote by z the vector $z^i - z^j$, and by u the vector $u^i - u^j$. So we get:

$$\|u\|^2 = \|u^i - u^j\|^2 = \sum_{k=1}^{m} \left(\sum_{l=1}^{n} z_l x_l^k \right)^2.$$

Let X_k be the random variable $(\sum_{l=1}^{n} z_l x_l^k)^2$. Its expectation is $\mu = \frac{1+\epsilon}{m}\|z\|^2$ (can be seen similarly to the proof of lemma 31). Therefore, the expectation of $\|u\|^2$ is $(1+\epsilon)\|z\|^2$. If we show that $\|u\|^2$ is concentrated enough around its mean, then it would prove the theorem. More formally, we state the following Chernoff bound lemma:

LEMMA 35
There exist constants $c_1 > 0$ and $c_2 > 0$ such that:

1. $Pr[\|u\|^2 > (1+\beta)\mu] < e^{-c_1\beta^2 m}$
2. $Pr[\|u\|^2 < (1-\beta)\mu] < e^{-c_2\beta^2 m}$

Therefore there is a constant c such that the probability of a "bad" case is bounded by:

$$Pr[(\|u\|^2 > (1+\beta)\mu) \vee (\|u\|^2 < (1-\beta)\mu)] < e^{-c\beta^2 m}$$

Now, we have $\binom{n}{2}$ random variables of the type $\|u_i - u_j\|^2$. Choose $\beta = \frac{\epsilon}{2}$. Using the union bound, we get that the probability that any of these random variables is not within $(1 \pm \frac{\epsilon}{2})$ of their expected value is bounded by

$$\binom{n}{2} e^{-c\frac{\epsilon^2}{4}m}.$$

So if we choose $m > \frac{8(\log n + \log c)}{\epsilon^2}$, we get that with positive probability, all the variables are close to their expectation within factor $(1 \pm \frac{\epsilon}{2})$. This means that for all i,j:

$$(1-\frac{\epsilon}{2})(1+\epsilon)\|z^i - z^j\|^2 \le \|u^i - u^j\|^2 \le (1+\frac{\epsilon}{2})(1+\epsilon)\|z^i - z^j\|^2$$

Therefore,

$$\|z_i - z_j\|^2 \le \|u^i - u^j\|^2 \le (1+\epsilon)^2\|z^i - z^j\|^2,$$

and taking square root:

$$\|z^i - z^j\| \le \|u^i - u^j\| \le (1+\epsilon)\|z^i - z^j\|,$$

as required.

It remains to prove lemma 7.3. We prove the first part. Let $\alpha^2 = \frac{1+\epsilon}{m}$, so $\mu = \alpha^2 m\|z\|^2$ and we get the following equation:

$$\begin{aligned}
P := Pr[\|u\|^2 > (1+\beta)\mu] &= Pr[\|u\|^2 > (1+\beta)\alpha^2 m\|z\|^2] \\
&= Pr[\|u\|^2 - (1+\beta)\alpha^2 m\|z\|^2 > 0] \\
&= Pr[t(\|u\|^2 - (1+\beta)\alpha^2 m\|z\|^2) > 0] \quad \forall t > 0 \\
&= Pr[\exp(t(\|u\|^2 - (1+\beta)\alpha^2 m\|z\|^2)) > 1] \\
&\le E[\exp(t(\|u\|^2 - (1+\beta)\alpha^2 m\|z\|^2))] \quad \text{(Markov)}
\end{aligned} \tag{7.3.2}$$

We calculate the last expectation:

$$\begin{aligned}
P &\le E[\exp(t(\|u\|^2))]\exp(-t(1+\beta)\alpha^2 m\|z\|^2) \quad \text{(constant goes out)} \\
&= E[\exp(t(\sum_{k=1}^{m}(\sum_{l=1}^{n} z_l x_l^k)^2))]\exp(-t(1+\beta)\alpha^2 m\|z\|^2) \\
&= E[\exp(t(\sum_k(\sum_l z_l^2 (x_l^j)^2)) + t(\sum_k(\sum_{l\neq h} z_l z_h x_l^k x_h^k)))]\exp(-t(1+\beta)\alpha^2 m\|z\|^2) \\
&= E[\exp(t\alpha^2 m\|z\|^2 + t(\sum_k(\sum_{l\neq h} z_l z_h x_l^k x_h^k)))]\exp(-t(1+\beta)\alpha^2 m\|z\|^2)
\end{aligned} \tag{7.3.3}$$

The last step used the fact that $(x_l^k)^2 = \alpha^2$ and $\sum z_l^2 = \|z\|^2$. So continuing, we get:

$$P \le E[\exp(t(\sum_k(\sum_{l\neq h} z_l z_h x_l^k x_h^k)))]\exp(-t\beta\alpha^2 m\|z\|^2) \tag{7.3.4}$$

The set of variables $\{x_l^k x_h^k\}_{l\neq h}$ are pairwise independent. Therefore the above expectation can be rewritten as a product of expectations:

$$P \leq \left(\prod_k \prod_{l\neq h} E[\exp(t z_l z_h x_l^k x_h^k)] \right) \exp(-t\beta\alpha^2 m\|z\|^2) \tag{7.3.5}$$

we notice that

$$E[\exp(t z_l z_h x_l^k x_h^k)] = \frac{1}{2}\exp(t z_l z_h \alpha^2) + \frac{1}{2}\exp(-t z_l z_h \alpha^2) < \exp(t^2 z_l^2 z_h^2 \alpha^4)$$

(the last inequality is easily obtained by Taylor expanding the exponent function). Plugging that in (7.3.5), we get:

$$\begin{aligned} P &< \left(\prod_k \prod_{l\neq h} \exp\left(t^2 z_l^2 z_h^2 \alpha^4\right) \right) \exp\left(-t\beta\alpha^2 m\|z\|^2\right) \\ &= \left(\prod_{l\neq h} \exp\left(t^2 z_l^2 z_h^2 \alpha^4\right) \right)^m \exp\left(-t\beta\alpha^2 m\|z\|^2\right) \\ &= \exp\left(m t^2 \sum_{l\neq h} z_l^2 z_h^2 \alpha^4 - t\beta\alpha^2 m\|z\|^2\right) \end{aligned} \tag{7.3.6}$$

Using simple analysis of quadratic function we see that the last expression obtains its minimum when

$$t = \frac{\beta\|z\|^2}{2\alpha^2 \sum_{l\neq h} z_l^2 z_h^2}.$$

Substituting for t, we get:

$$P < \exp\left(-\beta^2 m \frac{\|z\|^4}{4\sum_{l\neq h} z_l^2 z_h^2}\right) \tag{7.3.7}$$

Finally, the expression

$$\delta(z) = \left(\frac{\|z\|^4}{4\sum_{l\neq h} z_l^2 z_h^2} \right)$$

is bounded below by a constant c_1. To prove this, first note that $\delta(z) = \delta(\gamma z)$ for any $\gamma \neq 0$. So it is enough to consider the case $\|z\| = 1$. Then, using Lagrange multipliers technique, for example, we get that $\delta(z)$ obtains its minimum when $z_l = \frac{1}{\sqrt{n}}$ for each $l = 1..n$. Plugging this in the expression for $\delta(z)$ we see that it is bounded above by a constant c_1 that does not depend on n. This completes the proof. □

7.4 VC Dimension

scribe: Renato F. Werneck

We continue our study of high dimensional geometry with the concept of *VC-dimension*, named after its inventors Vapnik and Chervonenkis. This useful tool allows us to reason about situations in which the trivial probabilistic method (i.e., the union bound) cannot be applied. Typical situations are those in which we have to deal with uncountably many objects, as in the following example.

EXAMPLE 7 Suppose we pick set of m points randomly from the unit square in $\Re^2$. Let C be any triangle inside the unit square of area $1/4$. The expected number of samplepoints that lie inside it is $m/4$, and Chernoff bounds show that the probability that the this number is not in $m/4[1 \pm \epsilon]$ is $\exp(-\epsilon^2 m)$.

Now suppose we have a collection of $2^{\sqrt{m}}$ triangles of area $1/4$. The union bound shows that whp the sample is such that *all* triangles in our collection have $m/4[1 \pm \epsilon]$ samplepoints.

But what if we want the preceding statement to hold for all (*uncountably many*!) triangles of area $1/4$ in the unit square? The union bound fails badly! Nevertheless the following statement is true.

THEOREM 36
Let $\epsilon > 0$ and m be sufficiently large. Let S be a random sample of m points in the unit sphere. Then the probability is at least $1 - o(1)$ that every *triangle of area $1/4$ has between $m/4[1 \pm \epsilon]$ samplepoints.*

How can we prove such a result? The intuition is that even though the set of such triangles is uncountable, the number of "vastly different" triangles is not too large. Thus there is a small set of typical representatives, and we only need to argue about those.

7.4.1 Definition

VC-dimension is used to formalize the above intuition. First, some preliminary definitions:

DEFINITION 15 *A* Range Space *is a pair (X, R), where X is a set and R is a family of subsets of X ($R \subseteq 2^X$).*[1]

DEFINITION 16 *For any $A \subseteq X$, we define the $P_R(A)$, the projection of R on A, to be $\{r \cap A : r \in R\}$.*

DEFINITION 17 *We say that A is* shattered *if $P_R(A) = 2^A$, i.e., if the projection of R on A includes all possible subsets of A.*

With these definitions, we can finally define the *VC-dimension* of a range space:

DEFINITION 18 *The* VC-dimension *of a range space (X, R) is the cardinality of the largest set A that it shatters: VC-dim $= \sup\{|A| : A \text{ is shattered}\}$. It may be infinite.*

Example 1. Consider the case where $X = \mathbf{R}^2$ (the plane) and R is the set of all axis-aligned squares. The sets A we consider in this case are points on the plane. Let's consider different values of $|A|$:

- $|A| = 1$: With a single point on the plane, there are only two subsets A, the empty set and A itself. Since there for every point there are axis-aligned squares that contain it and others that don't, A is shattered.
- $|A| = 2$: If we have two points, it is easy to find axis-aligned squares that cover both of them, none of them, and each of them separately.
- $|A| = 3$: It is possible to come up with a set of three points such that are shattered; the vertices of an equilateral triangle, for example: there are axis-aligned squares that contain none of the points, each point individually, each possible pair of points, and all 3 points.
- $|A| = 4$: In that case, it is impossible to come up with four points on the plane that are shattered by the set of axis-parallel squares. There will always be some pair of points that cannot be covered by a square.

Therefore, the VC-dimension of this range space if 3.

[1] 2^X denotes the power set of X

Example 2. Let $X = \mathcal{F}^n$ (where $\mathcal{F}$ is a finite field) and R be the set of all linear subspaces of dimension d. (Note that these cannot possibly shatter a set of $d+1$ linearly independent vectors.)

First assume $d \leq n/2$. We claim that any set $A = \{v_1, v_2, \ldots, v_d\}$ of d linearly independent vectors is shattered by X. Take any subset I of the vectors. We can build a basis of a d-dimension subspace as follows: first, take the $|I|$ vectors of this set (which are all linearly independent); then, complete the basis with $d-|I|$ linearly independent vectors z_j in $\mathcal{F}^n$ that do not belong to the (d-dimensional) subspace determined by $v_1, v_2, \ldots, v_d$. This basis determines the subspace

$$\text{span}\{\{v_i\}_{i\in I}, z_{d+1}, z_{d+2}, \ldots, z_{d+d-|I|}\},$$

which is d-dimensional, as desired.

But note that this construction works as long as there are $d - |I|$ linearly independent vectors "available" to complete the basis. This happens when $d \leq n/2$. If $d > n/2$, we can pick at most $n - d$ vectors outside the set. Therefore, the VC-dimension of the range space analized is $\min\{d, n-d\}$.

We now study some properties of range spaces related to VC-dimension. The next theorem gives a bound; it is tight (exercise: prove this using the ideas in the previous example).

THEOREM 37
If (X, R) has VC-dimension d and $A \subseteq x$ has n elements then $|P_R(A)| \leq g(d,n)$ where $g(d,n) = \sum_{i\leq d} \binom{n}{i}$.

To prove this theorem, all we need is the following result:

LEMMA 38
If (X, R) has VC-dimension d and $|X| = n$ then $|R| \leq g(d,n)$.

This lemma clearly implies the theorem above: just apply the lemma with A instead of X, and look at the range space $(A, P_R(A))$. We now proceed to the proof of the lemma.

PROOF: We prove Lemma 38 by induction on $d+n$. Let $S = (X, R)$, and consider some element $x \in X$. By definition, we have that

$$S - x = (X - \{x\}, \overbrace{\{r - \{x\} : r \in R\}}^{R_1})$$

and

$$S \setminus x = (X - \{x\}, \overbrace{\{x \in R : x \notin r \text{ but } r \cup \{x\} \in R\}}^{R_2})$$

Note that every element of R that does not contain x is contained in R_1; and every element of R that does contain x is in R_2 (with x removed). Therefore,

$$|R| = |R_1| + |R_2|.$$

A well-known property of binomial coefficients states that

$$\binom{n}{i} = \binom{n-i}{i} + \binom{n-i}{i-1},$$

which implies that

$$g(d,n) = g(d,n-1) + g(d-1,n-1),$$

according to our definition of $g(d,n)$.

In the subproblems defined above, $|X - \{x\}| = n - 1$. We claim that there exists an x such that $S \setminus x$ has VC-dimension at most $d-1$. That being the case, the inductive hypothesis immediately applies:

$$|R| = |R_1| + |R_2| \leq g(d, n-1) + g(d-1, n-1) = g(d,n).$$

So once we prove the claim, we will be done. Let $A \subseteq S$ be shattered in S with $|A| = d$. We want to prove that in $S \setminus x$ no set $B \subseteq X - \{x\}$ can be shattered if $|B| = d$. But this is straighforward: if B is shattered in $S \setminus x$, then $B \cup \{x\}$ is shattered in S. This completes the proof of Lemma 38. (Incidentally, the bound given by the lemma is tight.) □

The following result is due to Haussler and Welzl (1987), and also appears in Vapnik and Chervonenkis's paper.

THEOREM 39
Let (X,R) be a range space with VC-dimension d, and let $A \subseteq X$ have size n. Suppose S is a random sample of size m drawn (with replacement) from A, and let m be such that

$$m \geq \max\left\{\frac{4}{\epsilon}\log\frac{2}{\delta}, \frac{8d}{\epsilon}\log\frac{8d}{\epsilon}\right\}.$$

Then, with probability at least $1-\delta$, S is such that for all $r \in R$ such that $|r \cap A| \geq \epsilon n$, we have $|r \cap S| \neq \emptyset$.

PROOF: Define r to be "heavy" if $|r \cap A| \geq \epsilon n$. We will now analyze a single experiment from two different points of view.

1. From the first point of view, we pick *two random samples* N and T, each of size exactly m (note that they both have the same characteristics as the set S mentioned in the statement of the theorem).

   Consider the following events:

   - E_1: there exists a heavy r such that $r \cap N = \emptyset$ (this is a "bad event"; we must prove that it happens with probability at most δ);
   - E_2: there exists a heavy r such that $r \cap N = \emptyset$, but $|r \cap T| \geq \epsilon m/2$.

   We claim that, if E_1 happens with high probability, then so does E_2. Specifically, we prove that $Pr(E_2|E_1) \geq 1/2$. Suppose E_1 has happened and that some specific r is the culprit. Now, pick T. Consider the probability that this r satisfies $|r \cap T| \geq \frac{\epsilon m}{2}$ (this is a lower bound for $Pr(E_2|E_1)$). We know that $|r \cap T|$ is a binomial random variable; its expectation is ϵm and its variance is $\epsilon(1-\epsilon)m$ (and therefore strictly smaller than ϵm. Using Chebyschev's inequality, we get:

   $$\Pr\left(|r \cap T| < \frac{\epsilon m}{2}\right) \leq \frac{\epsilon m}{(\epsilon m/2)^2} = \frac{4}{\epsilon m} \tag{7.4.1}$$

   Because the statement of the theorem assumes that $m > 8/\epsilon$, we conclude that indeed $Pr(E_2|E_1) \geq 1/2$.

2. Now consider the same experiment from a different point of view: we pick a *single sample* of size $2m$, which is then partitioned randomly between N and T. Consider the following event:

   E_r: $r \cap N = \emptyset$, but $|r \cap T| \geq \epsilon m/2$.

It is clear that

$$E_2 = \bigcup_{r:\text{heavy}} E_r.$$

Given $N \cup T$, the number of distinct events E_r is no greater than the number of different sets in the projection $P_{N\cup T}(R)$. From Lemma 38, the fact that the VC-dimension of X is d implies that $P_{N\cup T} \leq g(d, 2m)$. (Note: This is the crucial step that uses the VC dimension.)

Given a choice of $N \cup T$, we now estimate the probability of E_r happening when we select N and T:

$$P' = \Pr\left(r \cap N = \varnothing \mid |r \cap (N \cup T)| > \frac{\epsilon m}{2}\right).$$

Let $p = |r \cap (N \cup T)|$. Then, the conditional probability above is

$$\begin{aligned} P' &= \frac{(2m-p)(2m-p-1)\cdots(m-p+1)}{2m(2m-1)\cdots(m+1)} \\ &= \frac{m(m-1)\cdots(m-p+1)}{2m(2m-1)\ldots(2m-p+1)} \leq 2^{-p} \leq 2^{-\frac{\epsilon m}{2}} \end{aligned}$$

There are no more than $g(d, 2m)$ distinct events E_r. Applying the union bound, we see that the probability that at least one of the occurs for a fixed $N \cup T$ is at most $g(d, 2m) \cdot 2^{-\epsilon m/2}$. But this holds for any choice of $N \cup T$, so

$$\Pr(E_2) \geq g(d, 2m) \cdot 2^{-\epsilon m/2}. \tag{7.4.2}$$

Together, Facts 7.4.1 and 7.4.2 imply that

$$\Pr(E_1) \leq 2g(d, 2m) \cdot 2^{-\epsilon m/2}.$$

It remains to be proven that the RHS of the above expression is a lower bound on δ. The hard case is $d \geq 2$. The definition of $g(d, 2m)$ ensures that $g(d, 2m) \leq (2m)^d$, so it's enough to prove that

$$2 \cdot (2m)^d 2^{-\epsilon m/2} < \delta. \tag{7.4.3}$$

Rearranging this expression and taking the (base-two) logarithm of both sides,

$$\frac{\epsilon m}{2} \geq d \log(2m) + \log\frac{2}{\delta} \Longrightarrow \frac{\epsilon m}{4} + \frac{\epsilon m}{4} \geq d \log(2m) + \log\frac{2}{\delta}$$

The constraints on m defined in the statement of the theorem take care of the second "half" of this inequality:

$$m \geq \frac{4}{\epsilon}\log\frac{2}{\delta} \Longrightarrow \frac{\epsilon m}{4} \geq \frac{2}{\delta}.$$

It remains to be proven that

$$\frac{\epsilon m}{4} \geq d \log(2m).$$

This follows from a simple calculation (omitted).

□

7.4.2 VC dimension of intersections of range spaces

Let (X, R) be a range space of VC dimension d. Consider the range space $(X, R_{\cap h})$ where $R_{\cap h}$ consists of h-wise intersections of sets in R. We claim that the VC-dimension is at most $2dh\log(dh)$.

For, suppose A is some subset of R that is shattered in $(X, R_{\cap h})$, and let $k = |A|$. Then $|P_{R_{\cap h}}(A)| = 2^k$, but we know that it is at most $\binom{m}{h}$ where $m = |P_R(A)| \leq g(k,d)$. Hence

$$2^k \leq \binom{g(k,d)}{h} \leq k^{(d+1)h},$$

whence it follows that $k \leq 2dh\log(dh)$.

Exercise: Generalize to range spaces in which the subsets are boolean combinations of the sets of R.

7.4.3 Applications to Learning Theory

VC-dimension has a close connection to Valiant's PAC (Probably Approximately Correct) model, introduced in 1984. Specifically, VC-dimension gives conditions under which one can learn a concept in *principle* (using possibly a very tedious nonpolynomial time computation). Valiant's notion requires learning algorithms be polynomial. Nevertheless, in many cases one can obtain a polynomial-time algorithm using VC-dimension arguments, as in the following simple example.

Suppose we are trying to learn an unknown rectangle C contained in $[0;1]\times[0;1]$. Let μ be an unknown measure on $[0;1]\times[0;1]$ — one can think of if as concetrated on grid points of a fine grid. Somebody picks in samples according to μ and labels them + or − depending on whether C contains them or not. Our task is to guess what C is.

Our algorithm is the obvious one: find some rectangle C' that separates + from − (note: one exists, namely, C) and output that as your guess. Intuitively this makes sense, but how well does it approximate the actual rectangle C we are looking for?

Let's look at the symmetric difference between C and C', defined as follows.

$$C\Delta C' = (C \setminus C') \cup (C' \setminus C).$$

(Observation: the range space defined by the difference of rectangles has constant VC-dimension, as shown by Section 7.4.2.) This is the set of all points that are wrongly classified (points that belong to C and not to C' or vice-versa). Suppose that the probability of error is high, i.e., that $Pr_\mu[C\Delta C']$ is large. With high probability the sample is such that for every C_1, C_2 such that $\mu(C_1\Delta C_2)$ is large, $C_1\Delta C_2$ contains a sample point. Therefore, with high probability (over the choice of the sample), C' is such that $\mu(C'\Delta C)$ is small.

7.4.4 Geometric Algorithms

VC dimension arguments are invaluable for designing (randomized) geometric algorithms. We now consider a problem that has attracted great interest recently: nearest neighbor searching in $\mathbf{R}^d$, where d is large. Given n points in $\mathbf{R}^d$, our goal is to build a data structure using which one can, given any $y \in \mathbf{R}^d$, return the closest neighbor in the set. (A number of norms can be used to define "closeness": ℓ_1, ℓ_2, and ℓ_∞ are among those commonly used.)

A typical application is as follows. Assume you have a class of objects (images, for example, each of which is a vector of pixel values). Each object can be modeled as a point in $\mathbf{R}^d$. This would allow you to make similarity queries: given a new point (image), find the existing point that is closest (the image in the database that is similar to the new one, for example).Of course,

this assumes that the modeling of objects as vectors is such that similar objects map to nearby points, which happens to be true in many applications.

As a warmup, let's consider the 2-dimensional case. This is a well-solved problem. First, we build the *Voronoi diagram* of the original points: this diagram partitions the space into separate regions, one for each point in the original set. The region associated with point p represents the set of points that are closer to q than to any other point in the space (the way ties are handled depends on the application). This diagram can be built in $O(n \log n)$ time.

The original nearest neighbor searching problem then reduces to point-location in the Voronoi diagram. One way to do that is to divide the space into *vertical slabs* separated by vertical lines — one passing through each vertex of the Voronoi diagram. There are at most $O(n)$ slabs and each is divided into up to $O(n)$ subregions (from the Voronoi diagram). These slabs can be maintained in a straightforward way in a structure of size $O(n^2)$ (more sophisticated strucutures can reduce this significantly). Queries can be thought of as two binary searches (one to find the correct slab and another within the slab), and require $O(\log n)$ time.

Higher dimensions: Meiser's Algorithm The notion of Voronoi diagrams can be generalized in a straighforward way to arbitrary dimensions. For every two points p and q, the points that are equidistant from both define a set H_{pq} given by:

$$H_{pq} = \{x : d(x,p) = d(x,q)\}.$$

Considering a d-dimensional space with the ℓ_2 norm, we get that

$$\begin{aligned} d(x,p) &= d(x,q) \\ \sum_{i=1}^{d} (x_i - p_i)^2 &= \sum_{i=1}^{d} (x_i - q_i)^2 \\ 2\sum_{i=1}^{d} x_i(p_i - q_i) &= d(q_i^2 - p_i^2). \end{aligned}$$

Since p_i, q_i, and d are constants, H_{pq} must be a hyperplane.

The number of hyperplanes is $m = \binom{n}{2}$, quadratic in the number of points. However, the number of cells in the Voronoi diagram could be $n^{O(d)}$. How can we store this diagram so as to quickly answer nearest neighbor queries? Here we describe Meiser's algorithm (1986), which does so in $O(f(d)\log n)$ time, where $f(d)$ is some exponential function of d.

First, take a random sample S cointaining $O(d^2 \log^2 d)$ points. Then, construct a data structure for doing point location within S. Triangulate each cell of the arrangement of S. (In a plane, triangulation means dividing the cell into triangles; in higher dimensions one gets simplices. Recall that a simplex in $\Re^d$ is an intersection of $d+1$ halfspaces.) For each cell C in this triangulated arrangement, recursively construct a data structure for point location with respect to the hyperplanes that intersect C.

CLAIM: With high probability, S is such that for every cell C, the number of hyperplanes that intersect C is at most $m/2$.

Assuming the claim is true, it is easy to see that $Q(m)$, the query time for a structure of size m, is

$$Q(m) \le f(d^2 \log^2 d) + Q(m/2) \approx f(d^2 \log^2 d) \log m,$$

where f is the time required to perform point location within S. Moreover, the total space $P(m)$ necessary to process instances with m hyperplanes is

$$P(m) \le g(d^2 \log^d) \cdot S(m/2) \approx [g(d^2 \log^2 d)]^{\log m},$$

where g determines the size of the structure for point location within the sample. We don't compute f, g because that is not the point of this simplified description. (Using some care, the running time is $d^{O(d)} \log n$ and storage is $n^{O(d)}$.) We now proceed to prove the claim:

PROOF: Consider the range space (X, R), where X is the set of all m hyperplanes of the Voronoi diagram and R contains for each simplex D a subset $R_D \subseteq X$ containing all hyperplanes that intersect the interior of D. From Section 7.4.2 we know that the VC-dimension of this range space is $O(d^2 \log d)$. Then with high probability the sample must be such that for every simplex D that is intersected by at least $m/2$ hyperplanes is intersected by at least one hyperplane in the sample. This means that D cannot be a cell in the arrangement of the sample. □

Because the RHS grows faster than the LHS, we only have to worry about the smaller possible value of m, which is $m_0 = (8d/\epsilon)\log(8d/\epsilon)$ according to the statement of the theorem:

$$\begin{aligned}
\frac{\epsilon m_0}{4} &\geq d \log(2m_0) \\
\frac{\epsilon}{4}\left(\frac{8d}{\epsilon}\log\frac{8d}{\epsilon}\right) &\geq d \log\left(2 \cdot \frac{8d}{\epsilon}\log\frac{8d}{\epsilon}\right) \\
2d \log\left(\frac{8d}{\epsilon}\right) &\geq d \log\left(\frac{16d}{\epsilon}\log\frac{8d}{\epsilon}\right) \\
\log\left(\frac{8d}{\epsilon}\right)^2 &\geq \log\left(\frac{16d}{\epsilon}\log\frac{8d}{\epsilon}\right) \\
\left(\frac{8d}{\epsilon}\right)^2 &\geq \frac{16d}{\epsilon}\log\frac{8d}{\epsilon} \\
\frac{8d}{\epsilon} &\geq 2\log\frac{8d}{\epsilon} \\
\frac{4d}{\epsilon} &\geq \log\frac{8d}{\epsilon}.
\end{aligned}$$

The last inequality is obviously true, so we are done.

Chapter 8

Discrete Fourier Transform and its Uses

SCRIBE:*Loukas Georgiadis*

8.1 Introduction

Sanjeev admits that he used to find fourier transforms intimidating as a student. His fear vanished once he realized that it is a rewording of the following trivial idea.
If $u_1, u_2, \ldots, u_n$ is an orthonormal basis of $\Re^n$ then every vector v can be expressed as $\sum_i \alpha_i u_i$ where $\alpha_i =< v, u_i >$ and $\sum_i \alpha_i^2 = |v|_2^2$.

Whenever the word "fourier transform" is used one is usually thinking about vectors as functions. In the above example, one can think of a vector in $\Re^n$ as a function from $\{0, 1, \ldots, n-1\}$ to $\Re$. Furthermore, the basis $\{u_i\}$ is defined in some nice way. Then the α_i's are called *fourier coefficients* and the simple fact mentioned about their ℓ_2 norm is called *Parseval's Identity*.

The classic fourier transform, using the basis functions $\cos(2\pi nx)$ and $\sin(2\pi nx)$, is analogous except we are talking about differentiable (or at least continuous) functions from $[-1, 1]$ to $\Re$, and the definition of "inner product" uses integrals instead of sums (the vectors are in an infinite dimensional space).

EXAMPLE 8 (Fast fourier transform) The FFT studied in undergrad algorithms uses the following orthonormal set $u_j = \left(1 \quad \omega_N^j \quad \ldots \quad \omega_N^{(N-1)j}\right)^T$ for $j = 0, 1, \ldots, N-1$. Here $\omega_N = e^{-2\pi i/N}$ is the N^{th} root of 1.

Consider a function $f : [1, N] \mapsto \mathbf{R}$, i.e. a vector in $\mathbf{R}^N$. The Discrete Fourier Transform (DFT) of f, denoted by $\hat{f}$, is defined as

$$\hat{f}_k = \sum_{x=1}^{N} f(x)\ \omega_N^{(k-1)(x-1)},\ k \in [1, N] \tag{8.1.1}$$

The inverse transform reconstructs the initial vector f by

$$f(x) = \frac{1}{N} \sum_{k=1}^{N} \hat{f}_k\ \omega_N^{-(k-1)(x-1)}. \tag{8.1.2}$$

In other words f is written as a linear combination of the basis vectors $u_j = \begin{pmatrix} 1 & \omega_N^j & \dots & \omega_N^{(N-1)j} \end{pmatrix}^T$, for j in $[1, N]$. Another way to express this is by the matrix-vector product $\hat{f} = Mf$, where

$$M = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega_N^{1\cdot 1} & \omega_N^{1\cdot 2} & \dots & \omega_N^{1\cdot (N-1)} \\ 1 & \omega_N^{2\cdot 1} & \omega_N^{2\cdot 2} & \dots & \omega_N^{2\cdot (N-1)} \\ & & & \dots & \\ 1 & \omega_N^{(N-1)\cdot 1} & \omega_N^{(N-1)\cdot 2} & \dots & \omega_N^{(N-1)\cdot (N-1)} \end{pmatrix} \tag{8.1.3}$$

is the DFT matrix.

Fast Fourier Transform (FFT): We can exploit the special structure of M and divide the problem of computing the matrix-vector product Mf into two subproblems of size $N/2$. This gives an $O(N \log N)$ algorithm for computing the Fourier Transform. Using a simple property of the Discrete Fourier Transform and by applying the FFT algorithm to compute the DFT we can multiply n-bit numbers in $O(n \log n)$ operations.

8.2 Discrete Fourier Transform

In general we can use any orthonormal basis $u_1, u_2, \dots, u_N$. Then we can write

$$f = \sum_{i=1}^{N} \hat{f}_i u_i, \tag{8.2.1}$$

and the coefficients $\hat{f}_i$ are due to orthonormality equal to the inner product of f and of the corresponding basis vector, that is $\hat{f}_i = < f, u_i >$.

DEFINITION 19 *The length of f is the L_2-norm*

$$\|f\|_2 = \sqrt{\sum_i f^2(i)}. \tag{8.2.2}$$

Parseval's Identity: The transform preserves the length of f, i.e.

$$\|f\|_2^2 = \sum_i \hat{f}_i^2. \tag{8.2.3}$$

8.2.1 Functions on the boolean hypercube

Now we consider real-valued functions whose domain is the boolean hypercube, i.e., $f : \{-1, 1\}^n \mapsto \mathbf{R}$. The inner product of two functions f and g is defined to be

$$< f, g >= \frac{1}{2^n} \sum_{\mathbf{x} \in \{-1,1\}^n} f(\mathbf{x}) g(\mathbf{x}). \tag{8.2.4}$$

Now we describe a particular orthonormal basis. For all the subsets S of $\{1, \dots, n\}$ we define the functions

$$\chi_S(\mathbf{x}) = \prod_{i \in S} x_i. \tag{8.2.5}$$

where $\mathbf{x} = \begin{pmatrix} x_1 & \dots & x_n \end{pmatrix}^T$ is a vector in $\{-1,1\}^n$. The χ_S's form an orthonormal basis. Moreover, they are the eigenvectors of the adjacency matrix of the hypercube.

EXAMPLE 9 Note that

$$\chi_\emptyset(\mathbf{x}) = 1$$

and

$$\chi_{\{i\}}(\mathbf{x}) = x_i.$$

Therefore,

$$< \chi_{\{i\}}, \chi_\emptyset >= \frac{1}{2^n} \sum_{\mathbf{x}\in\{-1,1\}^n} x_i = 0.$$

REMARK 9 Let $x_i = 1$ for all i in $I \subseteq \{1,\dots,n\}$. Then $\chi_S(\mathbf{x}) = 1$ iff $|\bar{I} \cap S|$ is even.

REMARK 10 Now we show our basis is an orthonormal basis. Consider two subsets S and S' of $\{1,\dots,n\}$. Then,

$$\begin{aligned} < \chi_S, \chi_{S'} > &= \frac{1}{2^n} \sum_{\mathbf{x}\in\{-1,1\}^n} \chi_S(\mathbf{x})\chi_{S'}(\mathbf{x}) = \frac{1}{2^n} \sum_{\mathbf{x}\in\{-1,1\}^n} \prod_{i\in S} x_i \prod_{j\in S'} x_j \\ &= \frac{1}{2^n} \sum_{\mathbf{x}\in\{-1,1\}^n} \prod_{i\in S\triangle S'} x_i =< \chi_{S\triangle S'}, \chi_\emptyset >= 0 \text{ unless } S = S'. \end{aligned}$$

Here $S \triangle S' = (S - S') \cup (S' - S)$ is the symmetric difference of S and S'.

Since the χ_S's form a basis every f can be written as

$$f = \sum_{S\subseteq\{1,\dots,n\}} \hat{f}_S \chi_S, \tag{8.2.6}$$

where the coefficients are

$$\hat{f}_S =< f, \chi_S >= \frac{1}{2^n}\Big(\sum_{\mathbf{x}:\chi_S(\mathbf{x})=1} f(\mathbf{x}) - \sum_{\mathbf{x}:\chi_S(\mathbf{x})=-1} f(\mathbf{x})\Big). \tag{8.2.7}$$

REMARK 11 The basis functions are just the eigenvectors of the adjacency matrix of the boolean hypercube, which you calculated in an earlier homework. This observation —and its extension to Cayley graphs of groups— forms the basis of analysis of random walks on Cayley graphs using fourier transforms.

8.3 Applications of the Fourier Transform

In this section we describe two applications of the version of the Discrete Fourier Transform that we presented in section 8.2.1. The Fourier Transform is useful for understanding the properties of functions. The applications that we discuss here are: (i) PCP's and (ii) constructing small-bias probability spaces with a low number of random bits.

8.3.1 DFT and PCP's

We consider assignments of a boolean formula. These can be encoded in a way that enables them to be probabilistically checked by examining only a constant number of bits. If the encoded assignment satisfies the formula then the verifier accepts with probability 1. If no satisfying assignment exists then the verifier rejects with high probability.

DEFINITION 20 *The function $f : GF(2)^n \mapsto GF(2)$ is linear if there exist $a_1, \ldots, a_n$ such that $f(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i$ for all $\mathbf{x} \in GF(2)^n$.*

DEFINITION 21 *The function $g : GF(2)^n \mapsto GF(2)$ is δ-close if there exists a linear function f such that $\mathbf{Pr}_{x \in GF(2)^n}[f(\mathbf{x}) = g(\mathbf{x})] \geq 1 - \delta$.*

We think of g as an adversarily constructed function. The adversary attempts to deceive the verifier to accept g.

The functions that we considered are defined in $GF(2)$ but we can change the domain to be an appropriate group for the DFT of section 8.2.1.

$GF(2)$	$\{-1, 1\} \subseteq R$
$0 + 0 = 0$	$1 \cdot 1 = 1$
$0 + 1 = 1$	$1 \cdot (-1) = -1$
$1 + 1 = 0$	$(-1) \cdot (-1) = 1$

Therefore if we use the mapping $0 \mapsto 1$ and $1 \mapsto -1$ we have that $\sum_{i \in S} x_i \equiv \prod_{i \in S} x_i$. This implies that the linear functions become the Fourier Functions χ_S. The verifier uses the following test in order to determine if g is linear.

LINEAR TEST
Pick $\mathbf{x}, \mathbf{y} \in_R GF(2)^n$
Accept iff $g(\mathbf{x}) + g(\mathbf{y}) = g(\mathbf{x} + \mathbf{y})$

It is clear that a linear function is indeed accepted with probability one. So we only need to consider what happens if g is accepted with probability $1 - \delta$.

THEOREM 40
If g is accepted with probability $1 - \delta$ then g is δ-close.

PROOF: We change our view from $GF(2)$ to $\{-1, 1\}$. Notice now that

$$''\mathbf{x} + \mathbf{y}'' = \begin{pmatrix} x_1 y_1 & x_2 y_2 & \ldots & x_n y_n \end{pmatrix}$$

and that if ρ is the fraction of the points in $\{-1, 1\}^n$ where g and χ_S agree then by definition

$$\hat{g}_S = < g, \chi_S > = \rho - (1 - \rho) = 2\rho - 1.$$

Now we need to show that if the test accepts g with probability $1 - \delta$ it implies that there exists a subset S such that $\hat{g}_S \geq 1 - 2\delta$. The test accepts when for the given choice of $\mathbf{x}$

and $\mathbf{y}$ we have over $GF(2)$ that $g(\mathbf{x}) + g(\mathbf{y}) = g(\mathbf{x}+\mathbf{y})$. So the equivalent test in $\{-1,1\}$ is $g(\mathbf{x}) \cdot g(\mathbf{y}) \cdot g(\mathbf{x}+\mathbf{y}) = 1$ because both sides must have the same sign.

Therefore if the test accepts with probability $1-\delta$ then

$$\mu = \mathbf{E}_{\mathbf{x},\mathbf{y}}[g(\mathbf{x}) \cdot g(\mathbf{y}) \cdot g(\mathbf{x}+\mathbf{y})] = (1-\delta) - \delta = 1 - 2\delta. \tag{8.3.1}$$

We replace g in (8.3.1) with its Fourier Transform and get

$$\begin{aligned}
\mu &= \mathbf{E}_{\mathbf{x},\mathbf{y}}\Big[\Big(\sum_{S_1\subseteq\{1,\ldots,n\}} \hat{g}_{S_1}\chi_{S_1}(\mathbf{x})\Big)\cdot\Big(\sum_{S_2\subseteq\{1,\ldots,n\}} \hat{g}_{S_2}\chi_{S_2}(\mathbf{y})\Big)\cdot\Big(\sum_{S_3\subseteq\{1,\ldots,n\}} \hat{g}_{S_3}\chi_{S_3}(\mathbf{x}+\mathbf{y})\Big)\Big] \\
&= \mathbf{E}_{\mathbf{x},\mathbf{y}}\Big[\sum_{S_1,S_2,S_3\subseteq\{1,\ldots,n\}} \hat{g}_{S_1}\ \hat{g}_{S_2}\ \hat{g}_{S_3}\ \ \chi_{S_1}(\mathbf{x})\ \chi_{S_2}(\mathbf{y})\ \chi_{S_3}(\mathbf{x}+\mathbf{y})\Big] \\
&= \sum_{S_1,S_2,S_3\subseteq\{1,\ldots,n\}} \hat{g}_{S_1}\ \hat{g}_{S_2}\ \hat{g}_{S_3}\mathbf{E}_{\mathbf{x},\mathbf{y}}\Big[\chi_{S_1}(\mathbf{x})\ \chi_{S_2}(\mathbf{y})\ \chi_{S_3}(\mathbf{x}+\mathbf{y})\Big] \\
&= \sum_{S_1,S_2,S_3\subseteq\{1,\ldots,n\}} \hat{g}_{S_1}\ \hat{g}_{S_2}\ \hat{g}_{S_3}\mathbf{E}_{\mathbf{x},\mathbf{y}}\Big[\chi_{S_1}(\mathbf{x})\ \chi_{S_3}(\mathbf{x})\ \chi_{S_2}(\mathbf{y})\ \chi_{S_3}(\mathbf{y})\Big].
\end{aligned}$$

However

$$\mathbf{E}_{\mathbf{x}}\Big[\chi_{S_1}(\mathbf{x})\chi_{S_3}(\mathbf{x})\Big] = <\chi_{S_1},\chi_{S_3}>$$

which is non-zero only if $S_1 = S_3$. Similarly for fixed $\mathbf{x}$ we conclude that it must be $S_2 = S_3$. Therefore a non-zero expectation implies $S_1 = S_2 = S_3$ and we have

$$\mu = \sum_{S\subseteq\{1,\ldots,n\}} \hat{g}_S^3 \le \max_S |\hat{g}_S| \sum_{S\subseteq\{1,\ldots,n\}} \hat{g}_S^2 = \max_S |\hat{g}_S|.$$

because Parseval's identity gives $\sum_S \hat{g}_S^2 = 1$. We conclude that $\max_S |\hat{g}_S| \ge 1-2\delta$.
□

8.3.2 Long Code

DEFINITION 22 *The function $f : GF(2)^n \mapsto GF(2)$ is a coordinate function if there is an $i \in \{1,\ldots,n\}$ such that $f(\mathbf{x}) = x_i$.*

Obviously a coordinate function is also linear.

DEFINITION 23 *The Long Code for $[1\ldots W]$ encodes $w \in [1,\ldots,W]$ by the function $L_w(\mathbf{x}) = \mathbf{x}_w$.*

REMARK 12 A $\log W$ bit message is being encoded by a function from $GF(2)^W$ to $GF(2)$, i.e. with 2^W bits. So the Long Code is extremely redundant but is very useful in the context of PCP's.

Now the verifier is given a function and has to check if it is "close" to a long code. We give Hastad's test.

LONG CODE TEST
Pick $\mathbf{x},\mathbf{y} \in_R GF(2)^W$
Pick a "noise" vector $\mathbf{z}$ such that $\mathbf{Pr}[z_i = 1] = \epsilon$
Accept iff $g(\mathbf{x}) + g(\mathbf{y}) = g(\mathbf{x}+\mathbf{y}+\mathbf{z})$

Suppose that $g(\mathbf{x}) = x_W$ for all $\mathbf{x}$. Then $g(\mathbf{x}) + g(\mathbf{y}) = x_W + y_W$, while $g(\mathbf{x} + \mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y} + \mathbf{z})_W = x_W + y_W + x_W$. So the probability of acceptance in this case is $1 - \epsilon$.

THEOREM 41 (HÅSTAD)
If the Long Word Test accepts with probability $1/2 + \delta$ *then*

$$\sum_S \hat{f}_S^3 (1 - 2\epsilon)^{|S|} \geq 2\delta.$$

The proof is similar to that of the Theorem 40 and is left as an exercise. The interpretation of Theorem 41 is that if f passes the test with must depend on a "few" coordinates. This is not the same as being close to a coordinate function but it suffices for the PCP application.

REMARK 13 Hastad's analysis belongs to a field called *influence of boolean variables*, that has been used recently in demonstration of sharp thresholds for many random graph properties (Friedgut and others) and in learning theory.

8.3.3 Reducing the random bits in small-bias probability spaces

We consider n random variables $x_1, \ldots, x_n$, where each x_i is in $\{0, 1\}$ and their joint probability distribution is D. Let U be the uniform distribution. The distance of D from the uniform distribution is

$$\|D - U\| = \sum_{\omega \in \{0,1\}^n} |U(\omega) - D(\omega)|.$$

DEFINITION 24 *The bias of a subset* $S \subseteq \{1, \ldots, n\}$ *for a distribution* D *is*

$$bias_D(S) = \Big| \mathbf{Pr}_D\Big[\sum_{i \in S} x_i = 1 \Big] - \mathbf{Pr}_D\Big[\sum_{i \in S} x_i = 0 \Big] \Big|. \tag{8.3.2}$$

Here we are concerned with the construction of (small) ϵ-bias probability spaces that are k-wise independent, where k is a function of ϵ. Such spaces have several applications. For example they can be used to reduce the amount of randomness that is necessary in several randomized algorithms and for derandomizations.

DEFINITION 25 *The variables* $x_1, \ldots, x_n$ *are* ϵ*-biased if for all subsets* $S \subseteq \{1, \ldots, n\}$, $bias_D(S) \leq \epsilon$. *They are* k*-wise* ϵ*-biased if for all subsets* $S \subseteq \{1, \ldots, n\}$ *such that* $|S| \leq k$, $bias_D(S) \leq \epsilon$.

DEFINITION 26 *The variables* $x_1, \ldots, x_n$ *are* k*-wise* δ*-dependent if for all subsets* $S \subseteq \{1, \ldots, n\}$ *such that* $|S| \leq k$, $\|U(S) - D(S)\| \leq \delta$. *(*$D(S)$ *and* $U(S)$ *are the distributions* D *and* U *restricted to* S*.)*

REMARK 14 If $\delta = 0$ this is just the notion of k-wise independence studied briefly in an earlier lecture.

The following two conditions are equivalent:

1. All x_i are independent and $\mathbf{Pr}[x_i = 0] = \mathbf{Pr}[x_i = 1]$.
2. For every subset $S \subseteq \{1, \ldots, n\}$ we have $\mathbf{Pr}[\sum_{i \in S} x_i = 0] = \mathbf{Pr}[\sum_{i \in S} x_i = 1]$, that is the parity of the subset is equally likely to be 0 or 1.

Here we try to reduce the size of the sample space by relaxing the second condition, that is we consider biased spaces.

We will use the following theorem.

THEOREM 42 (DIACONIS AND SHAHSHAHANI '88)

$$\|D - U\| \leq 2^n \Big(\sum_{S \subseteq \{1,\ldots,n\}} \hat{D}_S^2 \Big)^{1/2} = \Big(\sum_{S \subseteq \{1,\ldots,n\}} bias_D(S)^2 \Big)^{1/2}. \tag{8.3.3}$$

PROOF: We write the Fourier expansion of D as $D = \sum_S \hat{D}_S \chi_S$. Then,

$$D = U + \sum_{S \neq \emptyset} \hat{D}_S \chi_S. \tag{8.3.4}$$

Now observe that

$$\| \sum_{S \neq \emptyset} \hat{D}_S \chi_S \|_2^2 = 2^n \Big\langle \sum_{S \neq \emptyset} \hat{D}_S \chi_S, \sum_{S \neq \emptyset} \hat{D}_S \chi_S \Big\rangle = 2^n \sum_{S \neq \emptyset} \hat{D}_S^2 \leq 2^n \sum_S \hat{D}_S^2. \tag{8.3.5}$$

Also by the Cauchy-Schwartz inequality

$$\|D - U\| = \|D - U\|_1 \leq 2^{n/2} \|D - U\|_2. \tag{8.3.6}$$

Thus

$$\|D - U\| \leq 2^n \Big(\sum_{S \subseteq \{1,\ldots,n\}} \hat{D}_S^2 \Big)^{1/2} = \Big(\sum_{S \subseteq \{1,\ldots,n\}} bias_D(S)^2 \Big)^{1/2}. \tag{8.3.7}$$

The second equality holds from equation (8.2.7). □

If we restrict our attention to subsets of size at most k then Theorem 8.3.3 implies:

COROLLARY 43
If D is ϵ-biased then it is also k-wise δ-independent for $\delta = 2^{k/2}\epsilon$.

The main theorem that we want to show is

THEOREM 44 (J. NAOR AND M. NAOR)
We can construct a probability space of ϵ-biased $\{0,1\}$ random variables $x_1, \ldots, x_n$ using $O(\log n + \log \frac{1}{\epsilon})$ random bits.

PROOF: The construction consists of three stages:

STAGE 1: Construct a family $\mathcal{F} \subseteq \{0,1\}^n$ such that for $\mathbf{r} \in_R \mathcal{F}$ and for all subsets $S \subseteq \{1, \ldots, n\}$,

$$\mathbf{Pr}[\chi'_S(\mathbf{r}) = 1] \geq \beta,$$

where

$$\chi'_S(\mathbf{x}) = \sum_{i \in S} x_i \ (\mathrm{mod}\ 2).$$

and β is some constant. We discuss this stage in section 8.3.3.

STAGE 2: Sample $\ell = O(\log(1/\epsilon))$ vectors $\mathbf{r}^1, \ldots, \mathbf{r}^\ell$ from $\mathcal{F}$ such that for all subsets S

$$\mathbf{Pr}_{\mathbf{r}^1,\ldots,\mathbf{r}^\ell}\Big[\forall i \in \{1,\ldots,\ell\}, \chi'_S(\mathbf{r}^i) = 0\Big] \leq \epsilon.$$

Note that if we just sample uniformly from $\mathcal{F}$ with $\log(1/\epsilon)$ samples then we need $\log n \cdot \log(1/\epsilon)$ random bits. We can do better by using a random walk on an expander. This will reduce the number of necessary random bits to $O(\log n + \log(1/\epsilon))$.

STAGE 3: Let $\mathbf{a}$ be a vector in $\{0,1\}^\ell$ chosen uniformly at random. The assignment to the random variables $x_1, \ldots, x_n$ is

$$\begin{pmatrix} x_1 & x_2 & \ldots & x_n \end{pmatrix}^T = \sum_{i=1}^{\ell} a_i \mathbf{r}^i$$

With probability at least $1 - \epsilon$ not all $\mathbf{r}^i$'s are such that for all S, $\chi'_S(\mathbf{r}^i) = 0$. But if there exists a vector $\mathbf{r}^j$ in the sample with $\chi'_S(\mathbf{r}^j) = 1$ then we can argue that $\mathbf{Pr}_D[\chi'_S(x_1, \ldots, x_n) = 1] = \mathbf{Pr}_D[\chi'_S(x_1, \ldots, x_n) = 0]$. Consequently $bias_D(S) \leq \epsilon$. □

Construction of the family $\mathcal{F}$

We construct $\log n$ families $\mathcal{F}_1, \ldots, \mathcal{F}_{\log n}$. For a vector $\mathbf{v} \in \{0,1\}^n$ let $|\mathbf{v}|$ be the number of ones in $\mathbf{v}$. For a vector $\mathbf{r}$ chosen uniformly at random from $\mathcal{F}_i$ and for each vector $\mathbf{v}$ in $\{0,1\}^n$ with $k \leq |\mathbf{v}| \leq 2k-1$ we require that

$$\mathbf{Pr}_{\mathbf{r}}[< \mathbf{r}, \mathbf{v} >= 1] \geq \beta.$$

Assume that we have random variables that are uniform on $\{1, \ldots, n\}$ and c-wise independent (using only $O(c \log n)$ random bits - see Lecture 4). We define three random vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{r}$ such that

1. The u_i's are c-wise independent and $\mathbf{Pr}[u_i = 0] = \mathbf{Pr}[u_i = 1] = 1/2$.
2. The w_i's are pairwise independent and $\mathbf{Pr}[w_i = 1] = 2/k$.
3. The elements of $\mathbf{r}$ are

$$r_i = \begin{cases} 1 & \text{if } u_i = 1 \text{ and } w_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

We claim that $\mathbf{Pr}[< \mathbf{r}, \mathbf{v} >= 1] \geq 1/4$ for all $\mathbf{v}$ with $k \leq |\mathbf{v}| \leq 2k-1$. In order to prove this we define the vector $\mathbf{v}'$ with coordinates:

$$v'_i = \begin{cases} 1 & \text{if } v_i = 1 \text{ and } w_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

By the above definition we have $< \mathbf{v}', \mathbf{u} >=< \mathbf{r}, \mathbf{v} >$. We wish to show that

$$\mathbf{Pr}[1 \leq |\mathbf{v}'| \leq c] \leq \frac{1}{2}. \tag{8.3.8}$$

Because the u_i's are c-wise independent (8.3.8) sufficies to guarantee that $\mathbf{Pr}[< \mathbf{v}', \mathbf{u} >= 1] \geq 1/4$. We view $|\mathbf{v}'|$ as a binomial random variable h with probability $p = 2/k$ for each non-zero

entry. Therefore, $\mathbf{E}[h] = pl$ and $\mathbf{Var}[h] = p(1-p)l$, where $l \in [k, 2k-1]$ is the number of potential non-zero entries. By Chebyshev's inequality at the endpoints of $[k, 2k-1]$ we have

$$\mathbf{Pr}[|h - pk| \geq 2] \leq \frac{pk(1-p)}{4} \leq \frac{1}{2}$$

and

$$\mathbf{Pr}[|h - 2pk| \geq 3] \leq \frac{2pk(1-p)}{9} \leq \frac{4}{9}.$$

Thus, for $c = 2kp + 3 = 7$ we get success probability at least $1/2$.

So if $|\mathbf{v}|$ is approximately known we can get high probability of success with $O(\log n)$ bits. Since we don't know $|\mathbf{v}|$ we chose $a \in_R \{0,1\}^{\log n}$ and construct

$$\mathbf{r} = \sum_{i=1}^{\log n} a_i \mathbf{r}^i, \text{ where } \mathbf{r}^i \in_R \mathcal{F}_i.$$

Each family $\mathcal{F}_i$ corresponds to the case that $|\mathbf{v}| \in [2^{i-1}, 2^i - 1]$. Now for any $\mathbf{v}$ we can show that $\mathbf{Pr}[< \mathbf{v}, \mathbf{r} >= 1] \geq \frac{1}{8}$ (so $\beta = 1/8$).

Chapter 9

Relaxations for NP-hard Optimization Problems

SCRIBE:*Manoj M P*

9.1 Introduction

In this lecture (and the next one) we will look at methods of finding approximate solutions to NP-hard optimization problems. We concentrate on algorithms derived as follows: the problem is formulated as an *Integer* Linear Programming (or integer semi-definite programming, in the next lecture) problem; then it is relaxed to an ordinary Linear Programming problem (or semi-definite programming problem, respectively) and solved. This solution may be fractional and thus not correspond to a solution to the original problem. After somehow modifying it ("rounding") we obtain a solution. We then show that the solution thus obtained is a approximately optimum.

The first main example is the Leighton-Rao approximation for "Sparsest Cut" in a graph. Then we switch gears to look at "Lift and Project" methods for designing LP relaxations. These yield a hierarchy of relaxations, with potentially a trade-off between running time and approximation ratio. Understanding this trade-off remains an important open probem.

9.2 Vertex Cover

We start off by introducing the LP relaxation methodology using a simple example. Vertex Cover problem involves a graph $G = (V, E)$, and the aim is to find the smallest $S \subset V$ such that $\forall \{i, j\} \in E$, $i \in S$ or $j \in S$.

An Integer Linear Programming (ILP) formulation for this problem is as follows:

$$\begin{aligned} x_i &\in \{0, 1\} \text{ for each } i \in V \\ x_i + x_j &\geq 1 \text{ for each } \{i, j\} \in E \\ \text{Minimize } &\sum_{i \in V} x_i \end{aligned}$$

The LP relaxation replaces the integrality constraints with

$$0 \leq x_i \leq 1 \text{ for each } i \in V$$

If OPT is the solution to the ILP (i.e., the exact solution to the vertex cover problem), and OPT_f the solution to the LP relaxation (f indicates that this solution involves *fractional* values for x_i), then clearly $\text{OPT} \geq \text{OPT}_f$. Now we show that $\text{OPT} \leq 2 \times \text{OPT}_f$. Consider the following *rounding algorithm* which converts a feasible solution to the LP to a feasible solution to the ILP, and increases the objective function by at most a factor of 2: if $x_i < \frac{1}{2}$, set $x'_i = 0$, else set $x'_i = 1$, where $\{x_i\}$ is the solution to the LP, and $\{x'_i\}$ a solution to the ILP.

We observe that the factor of 2 above is fairly tight. This is illustrated by the problem on a clique κ_n, where $\text{OPT} = n - 1$ and $\text{OPT}_f = n/2$ (by taking $x_i = \frac{1}{2}$ for all i).

What we have shown is that the the *integrality gap*, $\text{OPT}/\text{OPT}_f \leq 2$, and that this is tight.

Note that if the integrality gap of the relaxation is α then solving the LP gives an α-approximation to the optimum *value* (in this case, the size of the smallest vertex cover). Usually we want more from our algorithms, namely, an actual solution (in this case, a vertex cover). However, known techniques for upperbounding the integrality gap tend to be algorithmic in nature and yield the solution as well (though conceivably this may not always hold).

To sum up, the smaller the integrality gap (i.e., the *tighter* the relaxation), the better.

9.3 Sparsest Cut

Our next example is Graph Expansion, or Sparsest-Cut problem. Given a graph $G = (V, E)$ define

$$\beta_G = \min_{S \subset V} \frac{E(S, \bar{S})}{\min\{|S|, |\bar{S}|\}} = \min_{S \subset V : |S| \leq \frac{|V|}{2}} \frac{E(S, \bar{S})}{|S|}$$

where $E(S, \bar{S})$ is the number of edges between S and $\bar{S}$. [1] The Expansion problem is to find a cut S which realizes β_G.

LP formulation

Leighton and Rao (1988) showed how to approximate β_G within a factor of $O(\log n)$ using an LP relaxation method. The ILP formulation is in terms of the *cut-metric* induced by a cut.

DEFINITION 27 *The* cut-metric *on V, induced by a cut $(S, \bar{S})$, is defined as follows: $\Delta_S(i, j) = 0$ if $i, j \in S$ or $i, j \in \bar{S}$, and $\Delta_S(i, j) = 1$ otherwise.*

Note that this metric satisfies the triangle inequality (for three points i, j, k, all three pairwise distances are 0, or two of them are 1 and the other 0; in either case the triangle inequality is satisfied).

ILP formulation and Relaxation: It will be convenient to work with

$$\tilde{\beta}_G = \min_{S \subset V : |S| \leq \frac{|V|}{2}} \frac{E(S, \bar{S})}{|S||\bar{S}|}$$

Since $\frac{n}{2} \leq |\bar{S}| \leq n$, we have $\frac{n}{2}\tilde{\beta}_G \leq \beta_G \leq n\tilde{\beta}_G$, and the factor "2" here will end up being absorbed in the "$O(\log n)$."

[1] Compare this with the "Edge Expansion" problem we encountered in an earlier lecture: the Cheegar constant was defined as

$$h_G = \min_{S \subset V} \frac{E(S, \bar{S})}{\min\{\text{Vol}(S), \text{Vol}(\bar{S})\}}$$

where $\text{Vol}(S) = \sum_{v \in S} \deg(v)$. Note that for a d-regular graph, we have $\text{Vol}(S) = d|S|$ and hence $h_G = \beta_G / d$.

Below, the variables x_{ij} for $i, j \in V$ are intended to represent the distances $\Delta(i, j)$ induced by the cut-metric. Then we have $\sum_{i<j} x_{ij} = |S||\bar{S}|$. Consider the following ILP which fixes $|S||\bar{S}|$ to a value t:

$$\begin{aligned} \sum_{i<j} x_{ij} &= 1 && \text{(scaling by } 1/t\text{)} \\ x_{ij} &= x_{ji} \\ x_{ij} + x_{jk} &\geq x_{ik} \\ x_{ij} &\in \{0, 1/t\} && \text{(integrality constraint)} \\ \text{Minimize } & \sum_{\{i,j\} \in E} x_{ij} \end{aligned}$$

Note that this exactly solves the sparsest cut problem if we could take t corresponding to the optimum cut. This is because, (a) the cut-metric induced by the optimum cut gives a solution to the above ILP with value equal to $\tilde{\beta}_G$, and (b) a solution to the above ILP gives a cut S defined by an equivalence class of points with pair-wise "distance" (values of x_{ij}) equal to 0, so that $\frac{E(S,\bar{S})}{t} \leq \sum_{\{i,j\} \in E} x_{ij}$.

In going to the LP relaxation we substitute the integrality constraint by $0 \leq x_{ij} \leq 1$. This in fact is a relaxation for all possible values of t, and therefore lets us solve a single LP without knowing the correct value of t.

Rounding the LP

Once we formulate the LP as above it can be solved to obtain $\{x_{ij}\}$ which define a metric on the nodes of the graph. But to get a valid cut from this, this needs to be "rounded off" to a cut-metric- i.e., an integral solution of the form $x_{ij} \in \{0, \alpha\}$ has to be obtained (which, as we noted above, can be converted to a cut of cost no more than the cost it gives for the LP). The rounding proceeds in two steps. First the graph is embedded into the ℓ_1-norm metric-space $\mathbf{R}^{O(\log^2 n)}$. This entails some distortion in the metric, and can increase the cost. But then, we can go from this metric to a cut-metric with out any increase in the cost. Below we elaborate the two steps.

LP solution → $\mathbf{R}^{O(\log^2 n)}$ metric: The n vertices of the graph are embedded into the real space using Bourgain's Lemma.

LEMMA 45 (BOURGAIN'S LEMMA, FOLLOWING LLR'94)
There is a constant c such that $\forall n$, for every metric $\Delta(\cdot, \cdot)$ on n points, $\exists$ points $z_i \in \mathbf{R}^{O(\log^2 n)}$ such that for all $i, j \in [n]$,

$$\Delta(i, j) \leq ||z_i - z_j||_1 \leq c \log n \Delta(i, j) \tag{9.3.1}$$

Further such z_i can be found using a randomized polynomial time algorithm.

$\mathbf{R}^{O(\log^2 n)}$ metric → cut-metric: We show that ℓ_1-distances can be expressed as a positive combination of cut metrics.

LEMMA 46
For any n points $z_1, \ldots, z_n \in \mathbf{R}^m$, there exist cuts $S_1, \ldots, S_N \subset [n]$ and $\alpha_1, \ldots, \alpha_N \geq 0$ where $N = m(n-1)$ such that for all $i, j \in [n]$,

$$||z_i - z_j||_1 = \sum_k \alpha_k \Delta_{S_k}(i, j) \tag{9.3.2}$$

PROOF: We consider each of the m co-ordinates separately, and for each co-ordinate give $n-1$ cuts S_k and corresponding α_k. Consider the first co-ordinate for which we produce cuts $\{S_k\}_{k=1}^{n-1}$ as follows: let $\zeta_i^{(1)} \in \mathbf{R}$ be the first co-ordinate of z_i. w.l.o.g assume that the n points are sorted by their first co-ordinate: $\zeta_1^{(1)} \leq \zeta_2^{(1)} \leq \ldots \leq \zeta_n^{(1)}$. Let $S_k = \{z_1, \ldots, z_k\}$ and $\alpha_k = \zeta_{k+1}^{(1)} - \zeta_k^{(1)}$. Then $\sum_{k=1}^{n-1} \alpha_k \Delta_{S_k}(i,j) = |\zeta_i^{(1)} - \zeta_j^{(1)}|$.

Similarly defining S_k, α_k for each co-ordinate we get,

$$\sum_{k=1}^{m(n-1)} \alpha_k \Delta_{S_k}(i,j) = \sum_{\ell=1}^{m} |\zeta_i^{(\ell)} - \zeta_j^{(\ell)}| = ||z_i - z_j||_1$$

□

The cut-metric we choose is given by $S^* = \text{argmin}_{S_k} \frac{E(S_k, \bar{S_k})}{|S_k||\bar{S_k}|}$.

Now we shall argue that S^* provides an $O(\log n)$ approximate solution to the sparsest cut problem. Let OPT_f be the solution of the LP relaxation.

THEOREM 47
$\text{OPT}_f \leq \tilde{\beta}_G \leq c \log n \text{OPT}_f$.

PROOF: As outlined above, the proof proceeds by constructing a cut from the metric obtained as the LP solution, by first embedding it in $\mathbf{R}^{O(\log^2 n)}$ and then expressing it as a positive combination of cut-metrics. Let x_{ij} denote the LP solution. Then $\text{OPT}_f = \frac{\sum_{\{i,j\} \in E} x_{ij}}{\sum_{i<j} x_{ij}}$.

Applying the two inequalities in Equation (9.3.1) to the metric $\Delta(i,j) = x_{ij}$,

$$\frac{c \log n \sum_{\{i,j\} \in E} x_{ij}}{\sum_{i<j} x_{ij}} \geq \frac{\sum_{\{i,j\} \in E} ||z_i - z_j||_1}{\sum_{i<j} ||z_i - z_j||_1}$$

Now by Equation (9.3.2),

$$\begin{aligned}
\frac{\sum_{\{i,j\} \in E} ||z_i - z_j||_1}{\sum_{i<j} ||z_i - z_j||_1} &= \frac{\sum_{\{i,j\} \in E} \sum_k \alpha_k \Delta_{S_k}(i,j)}{\sum_{i<j} \sum_k \alpha_k \Delta_{S_k}(i,j)} = \frac{\sum_k \alpha_k \sum_{\{i,j\} \in E} \Delta_{S_k}(i,j)}{\sum_k \alpha_k \sum_{i<j} \Delta_{S_k}(i,j)} \\
&= \frac{\sum_k \alpha_k E(S_k, \bar{S_k})}{\sum_k \alpha_k |S_k||\bar{S_k}|} \geq \min_k \frac{E(S_k, \bar{S_k})}{|S_k||\bar{S_k}|} \\
&= \frac{E(S^*, \bar{S^*})}{|S^*||\bar{S^*}|}
\end{aligned}$$

where we used the simple observation that for $a_i, b_i \geq 0$, $\frac{\sum_k a_k}{\sum_k b_k} \geq \min_k \frac{a_k}{b_k}$.

Thus, $c \log n \text{OPT}_f \geq \frac{E(S^*, \bar{S^*})}{|S^*||\bar{S^*}|} \geq \tilde{\beta}_G$. On the other hand, we have already seen that the LP is a relaxation of an ILP which solves $\tilde{\beta}_G$ exactly, and therefore $\text{OPT}_f \leq \tilde{\beta}_G$. □

COROLLARY 48
$\frac{n}{2}\text{OPT}_f \leq \beta_G \leq c \log n \frac{n}{2} \text{OPT}_f$.

The $O(\log n)$ integrality gap is tight, as it occurs when the graph is a constant degree (say 3-regular) expander. (An exercise; needs a bit of work.)

9.4 Lift and Project Methods

In both examples we saw thus far, the integrality gap proved was tight. Can we design tighter relaxations for these problems? Researchers have looked at this question in great detail. Next, we consider an more abstract view of the process of writing better relaxation.

The feasible region for the ILP problem is a polytope, namely, the convex hull of the *integer* solutions [2]. We will call this the *integer polytope*. The set of feasible solutions of the relaxation is also a polytope, which contains the integer polytope. We call this the *relaxed polytope*; in the above examples it was of polynomial size but note that it would suffice (thanks to the Ellipsoid algorithm) to just have an implicit description of it using a polynomial-time separation oracle. On the other hand, if **P** $\neq$ **NP** the integer polytope has no such description. The name of the game here is to design a relaxed polytope that as close to the integer polytope as possible. *Lift-and-project* methods give, starting from any relaxation of our choice, a hierarchy of relaxations where the final relaxation gives the integer polytope. Of course, solving this final relaxation takes exponential time. In-between relaxations may take somewhere between polynomial and exponential time to solve, and it is an open question in most cases to determine their integrality gap.

Basic idea

The main idea in the *Lift and Project* methods is to try to simulate non-linear programming using linear programming. Recall that nonlinear constraints are very powerful: to restrict a variable x to be in $\{0,1\}$ we simply add the quadratic constraint $x(1-x)=0$. Of course, this means nonlinear programming is NP-hard in general. In lift-and-project methods we introduce auxiliary variables for the nonlinear terms.

EXAMPLE 10 Here is a quadratic program for the vertex cover problem.

$0 \leq x_i \leq 1$ for each $i \in V$

$(1-x_i)(1-x_j)=0$ for each $\{i,j\} \in E$

To *simulate* this using an LP, we introduce extra variables $y_{ij} \geq 0$, with the intention that y_{ij} represents the product $x_i x_j$. This is the **lift** step, in which we *lift* the problem to a higher dimensional space. To get a solution for the original problem from a solution for the new problem, we simply **project** it onto the variables x_i. Note that this a relaxation of the original integer linear program, in the sense that any solution of that program will still be retained as a solution after the lift and project steps. Since we have no way of ensuring $y_{ij} = x_i x_j$ in every solution of the lifted problem, we still may end up with a relaxed polytope. But note that this relaxation can be no worse than the original LP relaxation (in which we simply dropped the integrality constraints), because $1 - x_i - x_j + y_{ij} = 0$, $y_{ij} \geq 0 \Rightarrow x_i + x_j \geq 1$, and any point in the new relaxation is present in the original one.

(If we insisted that the matrix (y_{ij}) formed a positive semi-definite matrix, it would still be a (tighter) relaxation, and we get a Semi-definite Programming problem. We shall see this in the next lecture.)

9.5 Sherali-Adams Lift and Project Method

Now we describe the method formally. Suppose we are given a polytope $P \subset \mathbf{R}^n$ (via a separation oracle) and we are trying to get a representation for $P_0 \subset P$ defined as the convex hull of $P \cap \{0,1\}^n$. We proceed as follows:

[2] By integer solutions, we refer to solutions with co-ordinates 0 or 1. We assume that the integrality constraints in the ILP correspond to restricting the solution to such points.

The first step is *homogenization*. We change the polytope P into a *cone* K in $\mathbf{R}^{n+1}$. (A *cone* is a set of points that is closed under scaling: if x is in the set, so is $c \cdot x$ for all $c \geq 0$.) If a point $(\alpha_1, \ldots, \alpha_n) \in P$ then $(1, \alpha_1, \ldots, \alpha_n) \in K$. In terms of the linear constraints defining P this amounts to multiplying the constant term in the constraints by a new variable x_0 and thus making it homogeneous: i.e., $\sum_{i=1}^{n} a_i x_i \geq b$ is replaced by $\sum_{i=1}^{n} a_i x_i \geq b x_0$. Let $K_0 \subset K$ be the cone generated by the points $K \cap \{0,1\}^{n+1}$ ($x_0 = 0$ gives the origin; otherwise $x_0 = 1$ and we have the points which define the polytope P_0).

For $r = 1, 2, \ldots, n$ we shall define $SA^r(K)$ to be a cone in $\mathbf{R}^{V_{n+1}(r)}$, where $V_{n+1}(r) = \sum_{i=0}^{r} \binom{n+1}{i}$. Each co-ordinate corresponds to a variable y_s for $s \subset [n+1]$, $|s| \leq r$. The intention is that the variable y_s stands for the homogenuous term $(\prod_{i \in s} x_i) \times x_0^{r-|s|}$. Let $\mathbf{y^{(r)}}$ denote the vector of all the $V_{n+1}(r)$ variables.

DEFINITION 28 *Cone $SA^r(K)$ is defined as follows:*

- *$SA^1(K) = K$, with $y_{\{i\}} = x_i$ and $y_\phi = x_0$.*
- *$SA^r(K)$ The constraints defining $SA^r(K)$ are obtained from the constraints defining $SA^{r-1}(K)$: for each constraint $\mathbf{ay^{(r-1)}} \geq 0$, for each $i \in [n]$, form the following two constraints:*

$$(1 - x_i) * \mathbf{ay^{(r-1)}} \geq 0$$

$$x_i * \mathbf{ay^{(r-1)}} \geq 0$$

where the operator "$*$" distributes over the sum $\mathbf{ay^{(r-1)}} = \sum_{s \subset [n]: |s| \leq r} a_s y_s$ and $x_i * y_s$ is a shorthand for $y_{s \cup \{i\}}$.

Suppose $(1, x_1, \ldots, x_n) \in K \cap \{0,1\}^{n+1}$. Then we note that the cone $SA^r(K)$ contains the points defined by $y_s = \prod_{i \in s} x_i$. This is true for $r = 1$ and is maintained inductively by the constraints we form for each r. Note that if $i \in s$ then $x_i * y_s = y_s$, but we also have $x_i^2 = x_i$ for $x_i \in \{0,1\}$.

To get a relaxation of K we need to come back to $n+1$ dimensions. Next we do this:

DEFINITION 29 *$S^r(K)$ is the cone obtained by projecting the cone $SA^r(K)$ to $n+1$ dimensions as follows: a point $\mathbf{u} \in SA^r(K)$ will be projected to $\mathbf{u}|_{s:|s| \leq 1}$; the variable $\mathbf{u}_\phi$ is mapped to x_0 and for each $i \in [n]$ $\mathbf{u}_{\{i\}}$ to x_i.*

EXAMPLE 11 This example illustrates the above procedure for Vertex Cover, and shows how it can be thought of as a "simulation" of non-linear programming. The constraints for $S^1(K)$ come from the linear programming constraints:

$$\forall j \in V, 0 \leq y_{\{j\}} \leq y_\phi$$

$$\forall \{i,j\} \in E, y_{\{i\}} + y_{\{j\}} - y_\phi \geq 0$$

Among the various constraints for $SA^2(K)$ formed from the above constraints for $SA^1(K)$, we have $(1 - x_i) * (y_\phi - y_{\{j\}}) \geq 0$ and $(1 - x_i) * (y_{\{i\}} + y_{\{j\}} - y_\phi) \geq 0$ The first one expands to $y_\phi - y_{\{i\}} - y_{\{j\}} + y_{\{i,j\}} \geq 0$ and the second one becomes to $y_{\{j\}} - y_{\{i,j\}} - y_\phi - y_{\{i\}} \geq 0$, together enforcing $y_\phi - y_{\{i\}} - y_{\{j\}} + y_{\{i,j\}} = 0$. This is "simulating" the quadratic constraint $1 - x_i - x_j + x_i x_j = 0$ or the more familiar $(1 - x_i)(1 - x_j) = 0$ for each $\{i,j\} \in E$. It can be shown that the defining constraints for the cone $S^2(K)$ are the odd-cyle constraints: "for an odd cycle C, $\sum_{x_i \in C} \geq (|C| + 1)/2$." An exact characterization of $S^(K)r)$ for $r > 2$ is open.

Intuitively, as we increase r we get tighter relaxations of K_0, as we are adding more and more "valid" constraints. Let us prove this formally. First, we note the following characterization of $SA^r(K)$.

LEMMA 49
$\mathbf{u} \in SA^r(K)$ *iff* $\forall i \in [n]$ *we have* $\mathbf{v}^i, \mathbf{w}^i \in SA^{r-1}(K)$, *where* $\forall s \subset [n]$, $|s| \leq r-1$, $\mathbf{v}^i = \mathbf{u}_{s\cup\{i\}}$ *and* $\mathbf{w}^i = \mathbf{u}_s - \mathbf{u}_{s\cup\{i\}}$.

PROOF: To establish the lemma, we make our notation used in defining $SA^r(K)$ from $SA^{r-1}(K)$ explicit. Suppose $SA^{r-1}(K)$ has a constraint $\mathbf{a}\mathbf{y}^{(\mathbf{r-1})} \geq 0$. Recall that from this, for each $i \in [n]$ we form two constraints for $SA^r(K)$, say $\mathbf{a}'\mathbf{y}^{(\mathbf{r})} \geq 0$ and $\mathbf{a}''\mathbf{y}^{(\mathbf{r})} \geq 0$, where $\mathbf{a}'\mathbf{y}^{(\mathbf{r})} \equiv x_i * \mathbf{a}\mathbf{y}^{(\mathbf{r-1})}$ is given by

$$\mathbf{a}'_s = \begin{cases} 0 & \text{if } i \notin s \\ \mathbf{a}_s + \mathbf{a}_{s\setminus\{i\}} & \text{if } i \in s \end{cases} \tag{9.5.1}$$

and $\mathbf{a}''\mathbf{y}^{(\mathbf{r})} \equiv (1 - x_i) * \mathbf{a}\mathbf{y}^{(\mathbf{r-1})}$ by

$$\mathbf{a}''_s = \begin{cases} \mathbf{a}_s & \text{if } i \notin s \\ -\mathbf{a}_{s\setminus\{i\}} & \text{if } i \in s \end{cases} \tag{9.5.2}$$

Then we see that

$$\begin{aligned} \mathbf{a}'\mathbf{u} &= \sum_{s\ni i} \left(\mathbf{a}_s + \mathbf{a}_{s\setminus\{i\}}\right)\mathbf{u}_s \\ &= \sum_{s\ni i} \mathbf{a}_s\mathbf{v}^i_s + \sum_{s\not\ni i} \mathbf{a}_s\mathbf{v}^i_s = \mathbf{a}\mathbf{v}^i \end{aligned}$$

where we used the fact that for $s \ni i$, $\mathbf{u}_s = \mathbf{v}^i_s = \mathbf{v}^i_{s\setminus\{i\}}$. Similarly, noting that for $s \ni i$, $\mathbf{w}^i_s = 0$ and for $s \not\ni i$, $\mathbf{w}^i_s = \mathbf{u}_s - \mathbf{u}_{s\setminus\{i\}}$, we have

$$\begin{aligned} \mathbf{a}''\mathbf{u} &= \sum_{s\ni i} -\mathbf{a}_{s\setminus\{i\}}\mathbf{u}_s + \sum_{s\not\ni i} \mathbf{a}_s\mathbf{u}_s \\ &= \sum_{s\not\ni i} \left(-\mathbf{a}_s\mathbf{u}_{s\cup\{i\}} + \mathbf{a}_s\mathbf{u}_s\right) = \mathbf{a}\mathbf{w}^i \end{aligned}$$

Therefore, $\mathbf{u}$ satisfies the constraints of $SA^r(K)$ iff for each $i \in [n]$ $\mathbf{v}^i$ and $\mathbf{w}^i$ satisfy the constraints of $SA^{r-1}(K)$. □

Now we are ready to show that $S^r(K)$, $1 \leq r \leq n$ form a hierarchy of relaxations of K_0.

THEOREM 50
$K_0 \subset S^r(K) \subset S^{r-1}(K) \subset K$ *for every* r.

PROOF: We have already seen that each integer solution $\overline{x}$ in K_0 gives rise to a corresponding point in $SA^r(K)$: we just take $\mathbf{y}^{()}s = \prod_{i\in s} x_i$. Projecting to a point in $S^r(K)$, we just get $\overline{x}$ back. Thus $K_0 \subset S^r(K)$ for each r.

So it is left to only show that $S^r(K) \subset S^{r-1}(K)$, because $S^1(K) = K$.

Suppose $\mathbf{x} \in S^r(K)$ is obtained as $\mathbf{u}|_{s:|s|\leq 1}$, $\mathbf{u} \in SA^r(K)$. Let $\mathbf{v}^i, \mathbf{w}^i \in SA^{r-1}(K)$ be two vectors (for some $i \in [n]$) as given by Lemma 49. Since $SA^{r-1}(K)$ is a convex cone, $\mathbf{v}^i + \mathbf{w}^i \in SA^{r-1}(K)$. Note that for each $s \subset [n]$, $|s| \leq r-1$, $\mathbf{w}^i_s + \mathbf{v}^i_s = \mathbf{u}_s$. In particular this holds for s, $|s| \leq 1$. So $\mathbf{x} = (\mathbf{v}^i + \mathbf{w}^i)|_{s:|s|\leq 1} \in S^{r-1}(K)$. □

THEOREM 51
$S^n(K) = K_0$

PROOF: Recall from the proof of Theorem 50 that if $\mathbf{x} \in S^r(K)$, then there are two points $\mathbf{v}^i, \mathbf{w}^i \in SA^{r-1}(K)$ such that $\mathbf{x} = \mathbf{v}^i|_{s:|s|\leq 1} + \mathbf{w}^i|_{s:|s|\leq 1}$. It follows from the definition of $\mathbf{v}^i$ and $\mathbf{w}^i$ that $\mathbf{v}^i_\phi = \mathbf{v}^i_{\{i\}}$ and $\mathbf{w}^i_{\{i\}} = 0$. So $\mathbf{v}^i|_{s:|s|\leq 1} \in S^{r-1}(K)|_{y_{\{i\}}=y_\phi}$ and $\mathbf{w}^i|_{s:|s|\leq 1} \in S^{r-1}(K)|_{y_{\{i\}}=0}$. Hence, $S^r(K) \subset S^{r-1}(K)|_{y_{\{i\}}=y_\phi} + S^{r-1}(K)|_{y_{\{i\}}=0}$. Further, this holds for all $i \in [n]$. Thus,

$$S^r(K) \subset \bigcap_{i\in[n]} \left(S^{r-1}(K)|_{y_{\{i\}}=y_\phi} + S^{r-1}(K)|_{y_{\{i\}}=0}\right)$$

Repeating this for $r-1$ and so on, we get

$$S^r(K) \subset \bigcap_{\{i_1,\ldots,i_r\}\subset[n]} \left(\sum_{T\in\{0,y_\phi\}^r} K|_{(y_{\{i_1\}},\ldots,y_{\{i_r\}})=T} \right)$$

For $r = n$ this becomes simply

$$S^n(K) \subset \sum_{T\in\{0,y_\phi\}^n} K|_{(y_{\{1\}},\ldots,y_{\{n\}})=T} = K_0$$

Along with $K_0 \subset S^n(K)$ this gives us that $S^n(K) = K_0$. □

Finally we note that for small r we can solve an optimization problem over $S^r(K)$ relatively efficiently.

THEOREM 52
If there is a polynomial (in n) time separation oracle for K, then there is an $n^{O(r)}$ algorithm for optimizing a linear function over $S^r(K)$.

PROOF: Note that using Lemma 49, a separation oracle for $S^r(K)$ can be formed by calling the separation oracle for $S^{r-1}(K)$ $2n$ times (for $\mathbf{v}^i$ and $\mathbf{w}^i$, $i \in [n]$), and therefore, calling the separation oracle for K $n^{O(r)}$ times. Given access to the separation oracle, the optimization can be carried out using the ellipsoid method in polynomial time. □

Thus, on the one hand a higher value of r gives a tighter relaxation (a smaller integrality gap) and possibly a better approximation (with $r = n$ giving the exact solution), but on the other hand the time taken is exponential in r. So to use this method, it is crucial to understand this trade-off. It has been shown by Arora, Bollobas and Lovasz that in applying a related method by Lovasz and Schriver to the Vertex cover problem, even for $r = \Omega(\sqrt{\log n})$ the integrality gap is as bad as $2 - o(1)$. Sanjeev conjectures that for somewhat higher r the integrality gap may go down significantly. But this problem is open.

Chapter 10

Semidefinite Programming

SCRIBE:*Edith Elkind*

10.1 Introduction

Semidefinite programming (SDP) is an extension of linear programming that can be applied to integer optimization problems even when linear programming itself is of little help.

In this method, in addition to linear constraints, the matrix formed by variables is required to be positive semidefinite. As the feasible solution space becomes smaller, the solution to the semidefinite program might be closer to the solution to the original integer problem, as compared to the corresponding LP solution.

Since SDP is a special case of convex programming, a semidefinite program can be solved to an arbitrary precision in polynomial time using the ellipsoid method or a version of Karmarkar's algorithm.

10.2 Basic definitions

DEFINITION 30 *A symmetric matrix $M \in \mathbb{R}^{n\times n}$ is* positive semidefinite *(note: this is often written as $A \succcurlyeq 0$) if any of the following conditions holds:*

(i) *M can be represented as AA^T for some $A \in \mathbb{R}^{n\times n}$;*

(ii) *M has no negative eigenvalues;*

(iii) *$x^T Mx \geq 0$ for any $x \in \mathbb{R}^n$;*

(iv) *$\exists \mathbf{v}_1, \ldots, \mathbf{v}_n \in \mathbb{R}^n$ such that $M_{ij} = \langle \mathbf{v}_i, \mathbf{v}_j \rangle$.*

THEOREM 53
Conditions (i)–(iv) are equivalent.

PROOF: An easy exercise. □

LEMMA 54
The set $\{M \in \mathbb{R}^{n\times n} \mid M$ is a symmetric positive semidefinite matrix $\}$ is a convex cone.

PROOF: From (ii), it follows that if M is positive semidefinite, then so is λM for any $\lambda > 0$. From (iii), it follows that if M_1, M_2 are positive semidefinite, then so is $M_1 + M_2$. Obviously, symmetry is preserved, too. □

DEFINITION 31 *A* semidefinite program *is formulated as follows:*

$$\begin{aligned} \min & C \cdot Y \\ A_1 \cdot Y &\geq b_1, \\ & \dots \\ A_m \cdot Y &\geq b_m, \\ Y &\succcurlyeq 0 \end{aligned}$$

where Y is an $n \cdot n$-matrix of variables, $C, A_1, \dots, A_m \in \mathbb{R}^{n \cdot n}$, $b_1, \dots, b_m \in \mathbb{R}$ and $X \cdot Y$ is interpreted as $\sum_{i,j=1}^{n} X_{ij} Y_{ij}$.

Such a convex program can be solved to an arbitrary precision (modulo some technical requirements we do not go into) using the Ellipsoid method. As mentioned earlier this method can solve a linear program to an arbitrary degree of precision in polynomial time even if the number of constraints is non-polynomial (or infinite), so long as there is a polynomial-time separation oracle for the set of constraints. In case of the above convex program, we can replace the "$Y \succcurlyeq 0$" by the infinite family of linear constraints $x^T Y x \geq 0$ for every $x \in \Re^n$. Furthermore, given any Y that is not psd, there is a simple algorithm (derived from the Cholesky decomposition) that gives an $x \in \Re^n$ such that $x^T Y x < 0$; this is the separation oracle.

10.3 An SDP for Graph Expansion

Last time, we wrote a linear program for Graph Expansion. Given a graph $G = (V, E)$, for every $i, j \in V$, we introduce a variable x_{ij} and consider the following LP:

$$\begin{aligned} \min & \frac{\sum_{\{i,j\} \in E} x_{ij}}{\sum_{i<j} x_{ij}} \\ x_{ij} + x_{jk} &\leq x_{ik} \text{ for all } i, j, k \text{ (triangle inequality)} \\ x_{ij} &\geq 0 \\ x_{ij} &= x_{ji} \end{aligned}$$

In an effort to tighten the relaxation, we require the following constraint in addition:

$$\exists\, v_1, v_2, \dots, v_n \in \Re^n \text{ such that } x_{ij} = (v_i - v_j) \cdot (v_i - v_j).$$

We show that this constraint is not an unfair one, by shoing that the optimum integer solution (and, more generally, any cut metric) satisfies it: simply pick an arbitrary vector $\mathbf{u}$ and set $\mathbf{v}_i = \mathbf{u}$ if $v_i \in S$ and $\mathbf{v}_i = -\mathbf{u}$ if $v_i \in \bar{S}$, so the optimum integer solution is still feasible.

How can we optimize under this constraint using SDP? Consider the matrix M where $M_{ii} = 1$ and $M_{ij} = 1 - \frac{x_{ij}}{2}$ if $i \neq j$. The constraint amounts to saying this matrix to be psd (see Theorem 30 part (iv)).

We leave it as an exercise to show that this SDP provides a lower bound that is at least as good as the eigenvalue bound for d-regular graphs and no worse than Leighton–Rao bound for the general case. Thus this approach combines the best of both worlds.

It is conjectured that the linearity gap for this program is $O(1)$ (it is known to be $O(\log n)$). This would follow from another conjecture (by Goemans and Linial), namely, that any metric $d(i, j)$ that can be represented as $\|\mathbf{v}_i - \mathbf{v}_j\|^2$ for some $\{\mathbf{v}_i\}_{i=1}^n$, can be embedded into l^1 with $O(1)$ distortion.

10.4 0.878-Approximation for Max Cut

Let us present the celebrated 0.878 approximation algorithm for Max Cut by Goemans and Williamson (1993). Here the goal is to partition the vertices of a graph $G = (V, E)$ into two sets S and $\bar{S}$ so as to *maximize* $E(S, \bar{S})$.

The setup is almost identical to the previous case; the only difference is that now we seek to maximize $\sum_{\{i,j\}\in E} x_{ij} = \sum_{\{i,j\}\in E} 1/4\|\mathbf{v}_i - \mathbf{v}_j\|^2$, where $\|\mathbf{v}_i\| = 1$, $i = 1, \ldots, n$. (The factor $1/4$ is needed for renormalization: if $\mathbf{v}_i = -\mathbf{v}_j$, then $\|\mathbf{v}_i - \mathbf{v}_j\|^2 = 4$.)

Denote by OPT_f the optimal fractional solution; the corresponding vectors are $\mathbf{v}_1, \ldots, \mathbf{v}_n$. Note that for any pair of vectors we can draw a 2-dimensional plane through them, so

$$\|\mathbf{v}_i - \mathbf{v}_j\|^2 = \langle \mathbf{v}_i - \mathbf{v}_j, \mathbf{v}_i - \mathbf{v}_j \rangle = 2 - 2\cos\theta_{ij}, \tag{10.4.1}$$

where θ_{ij} is the angle between these two vectors in the plane.

This suggests the following algorithm: pick a random hyperplane H passing through the origin. It divides the endpoints of $\{\mathbf{v}_i\}_{i=1}^n$ into two sets; assign the vertices of the graph to S or $\bar{S}$ accordingly.

For any $\{i, j\} \in E$, the probability that $\{i, j\}$ crosses the cut is θ_{ij}/π, so the expected value of $E(S, \bar{S})/\mathrm{OPT}_f$ is

$$\frac{4\sum_{\{i,j\}\in E}\theta_{ij}}{\pi \times 2\sum_{\{i,j\}\in E}(1-\cos\theta_{ij})} \geq \min \frac{\theta_{ij}}{1-\cos\theta_{ij}} \times \frac{2}{\pi} = 0.878 \tag{10.4.2}$$

Hence, this algorithm gives a 0.878-approximation.

10.5 Spectral Partitioning of Random Graphs

This part is based on a FOCS 2001 paper by Frank McSherry, which considers the task of finding a solution to an NP-hard graph problem when the input graph is drawn from a distribution over acceptable graphs.

For example, suppose that a graph on n vertices is constructed as follows: include each edge with probability p; then pick k vertices out of n (uniformly over all sets of size k), and complete the clique on these vertices.

Given a graph constructed according to this procedure, can we find the planted clique with high probability (over the coin tosses of the generating algorithm as well as our own random choices)?

More generally, consider the following setup: for a graph on n vertices, there is a mapping $\psi : [n] \mapsto [k]$ that partitions the vertices into k classes, and a matrix $P_{k\times k}$, $0 \leq P_{ij} \leq 1$. To generate a random instance $\hat{G}$ on a vertex set V, $|V| = n$, include an edge between v_i and v_j with probability $P_{\psi(i)\psi(j)}$. We would like to construct an algorithm that given $\hat{G}$ can reconstruct ψ with high probability. (Usually, after that it is possible to (approximately) reconstruct P).

Note that for the clique problem described above, $k = 2$, $P_{00} = 1$, $P_{01} = P_{10} = P_{11} = p$ and the planted clique can be described as $\psi^{-1}(0)$.

Denote the adjacency matrix of $\hat{G}$ by $\hat{M}$. Note that $M = \mathbf{E}\hat{M}$ gives us the corresponding edge probabilities. Hence, for any two vertices that are in the same class with respect to ψ, the corresponding columns are identical, so the rank of M is at most k. Obviously, if we knew M, we could find ψ by grouping the columns; since $\hat{M}$ is obtained from M by randomized rounding, let us try to use clustering.

Let E be the error matrix, $E = \hat{M} - M$. Suppose that P projects the space spanned by the columns of $\hat{M}$ onto k dimensions. If both $\|P(E)\|$ and $\|P(M) - M\|$ are small, then by triangle inequality, $P(\hat{M}) \approx M$. Now, our goal is to find a suitable P.

Since M is a low-rank symmetric matrix, we can use singular value decomposition (SVD).

It is known that an $m \times n$-matrix A of rank r can be represented as $U\Sigma V^T$, where U is an $m \times r$ orthogonal matrix, $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, and V is an $n \times r$ orthogonal matrix.

This decomposition is unique up to a permutation of σ_i's (and respective permutations of rows and columns of U and V), so we can assume $\sigma_1 \geq \ldots \geq \sigma_r$; for a symmetric A, σ_i's are the eigenvalues.

By truncating U to k columns, Σ to k values, and V^T to k rows, we obtain a matrix A_k, which is known to be the best rank k approximation of A in the spectral norm. The columns of this matrix are projections of the corresponding columns of A onto the first k eigenvectors of A. In what follows, we denote the projection $X \mapsto X_k$ by P_{X_k}.

Instead of applying SVD directly, we will use a minor modification of this approach. Namely, consider the following algorithm **Partition**(τ):

- Randomly divide $[n]$ into two parts; let us say that this division splits the columns of $\hat{G}$ as $[\hat{A} \mid \hat{B}]$.
- Compute $P_1 = P_{B_k}$, $P_2 = P_{A_k}$.
- Let $H = [P_1(\hat{A})|P_2(\hat{B})]$.
- While there are unclustered nodes, choose one of them (say, u) and put all unclustered vs such that $|\hat{H_u} - \hat{H_v}| < \tau$ into u's cluster; repaet.
- Return the resulting partition.

Here τ is a parameter (intercluster distance). Intuitively, since our partition is random, any class is likely to be split evenly between A and B, so the projection computed for A should work for B and vice versa. By splitting the matrix into two parts we avoid some tricky interdependency issues.

Suppose that for all u we have

$$|P_1(A_u) - A_u| \leq \gamma_1 \text{ and } |P_1(A_u - \hat{A_u})| \leq \gamma_2$$
$$|P_2(B_u) - B_u| \leq \gamma_1 \text{ and } |P_2(B_u - \hat{B_u})| \leq \gamma_2$$

and $|G_u - G_v| \geq 4(\gamma_1 + \gamma_2)$ whenever $\psi(u) \neq \psi(v)$. Then our algorithm finds ψ correctly if $\tau = 2(\gamma_1 + \gamma_2)$. To see that, suppose that u and v ended up in the same cluster. As $|G_u - \hat{H_u}| \leq \gamma_1 + \gamma_2$, $|G_v - \hat{H_v}| \leq \gamma_1 + \gamma_2$, and $|\hat{H_u} - \hat{H_v}| < 2(\gamma_1 + \gamma_2)$, it must be that $G_u = G_v$. Conversely, if $G_u = G_v$, $\hat{H_u}$ and $\hat{H_v}$ cannot be too far apart.

To make use of this algorithm, we need a bound on γ_1 and γ_2.

THEOREM 55
With probability at least $1 - \delta$,

$$|P_{\hat{A}}(B_u) - B_u| \leq 8\sigma\sqrt{nk/s_u}$$
$$|P_{\hat{A}}(B_u - \hat{B_u})| \leq \sqrt{2k\log(n/\delta)},$$

where σ *is the largest deviation of an entry in* $\hat{A}$ *and* s_u *is the size of the part containing* u*, i.e.,* $s_u = |\{v \mid \psi(v) = \psi(u)\}|$.

For the proof, the reader is referred to the paper.

PRINCETON UNIVERSITY F'02 COS 597D: A THEORIST'S TOOLKIT

Homework 1

Out: *September 16* Due: *September 23*

You can collaborate with your classmates, but be sure to list your collaborators with your answer. If you get help from a published source (book, paper etc.), cite that. Also, limit your answers to one page or less —you just need to give enough detail to convince me. If you suspect a problem is open, just say so and give reasons for your suspicion.

§1 For every integer $k \geq 1$ show that any graph with more than k^k nodes has either a clique or an independent set of size at least k. (*Moral:* A graph cannot be completely disordered: there is always some local order in it.) Can you prove the same statement for graphs with fewer than k^k nodes?

§2 A bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2|$ is said to be an (α, β) *expander* if every subset $S \subseteq V_1$ of size at most $\alpha |V_1|$ has at least $\beta |S|$ neighbors in V_2. Show that for every $\alpha, \beta > 0$ satisfying $\alpha\beta < 1$ there is an integer d such that a d-regular expander exists for every large enough $|V_1|$.

§3 A graph is said to be *nontransitive* if there is no triple of vertices i, j, k such that all of $\{i, j\}, \{j, k\}, \{k, i\}$ are edges. Show that in a nontransitive d-regular graph, there is an independent set of size at least $n \log d / 8d$. (Hint: Suppose we try to pick an independent set S randomly from all independent sets in the graph. For any vertex v, suppose you have picked the portion of S except for v and the neighbors of v. How would you pick the rest of S?)

PRINCETON UNIVERSITY F'02 COS 597D: A THEORIST'S TOOLKIT

Homework 2

Out: *October 7* Due: *October 21*

You can collaborate with your classmates, but be sure to list your collaborators with your answer. If you get help from a published source (book, paper etc.), cite that. Also, limit your answers to one page or less —you just need to give enough detail to convince me. If you suspect a problem is open, just say so and give reasons for your suspicion.

§1 Let $\mathcal{A}$ be a class of deterministic algorithms for a problem and I be the set of possible inputs. Both sets are finite. Let $\text{cost}(A, x)$ denote the cost of running algorithm A on input x. The *distributional complexity* of the problem is

$$\max_{\mathcal{D}} \min_{A \in \mathcal{A}} \mathbf{E}_{x \in \mathcal{D}}[\text{cost}(A, x)],$$

where $\mathcal{D}$ is a probability distribution on I. The *randomized complexity* is

$$\min_{\mathcal{P}} \max_{x \in I} \mathbf{E}_{A \in \mathcal{P}}[\text{cost}(A, x)],$$

where $\mathcal{P}$ is a probability distribution on $\mathcal{A}$. Prove Yao's lemma, which says that the two complexities are equal.

Does this result hold if the class of algorithms and inputs is infinite?

§2 (*The rent-or-buy problem*) Your job requires you to take long trips by car. You can either buy a car ($ 5000) or rent one whenever you need it ($ 200 each time).

If i is the number of trips you end up making, clearly, it makes sense to buy if $i > 25$. (Let us ignore emotional factors, as well as the fact that a used car is still worth something after a few years.) Furthermore the optimum expenditure as a function of i is $C(i) = \min\{5000, 200i\}$.

However, you do not know i ahead of time: in fact, you don't learn about each trip until the day before, at which time you have to rent or buy. (Such problems are studied in a field called *online algorithms*.)

(a) Show that if you make the decision to buy or rent at each step using a deterministic algorithm then there is a strategy whose cost is at most $2C(i)$ for any i. Show that no deterministic algorithm can do much better for some i. (Alas, life is suboptimal??)

(b) Does your answer change if you can use a randomized strategy, and try to minimize your expected cost? (Hint: Lowerbounds can be proved using Yao's Lemma.)

§3 Prove the Schwartz-Zippel Lemma: If $g(x_1, x_2, \ldots, x_m)$ is any nonzero polynomial of total degree d and $S \subseteq F$ is any subset of field elements, then the fraction of $(a_1, a_2, \ldots, a_m) \in S^m$ for which $g(a_1, a_2, \ldots, a_m) = 0$ is at most $d/|S|$.

§4 (*Sudan's list decoding*) Let $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n) \in F^2$ where $F = GF(q)$ and $q \gg n$. We say that a polynomial $p(x)$ *describes* k of these pairs if $p(a_i) = b_i$ for k values of i.

(a) Show that there exists a bivariate polynomial $Q(z, x)$ of degree at most $\lceil \sqrt{n} \rceil + 1$ in z and x such that $Q(b_i, a_i) = 0$ for each $i = 1, \ldots, n$. Show also that there is an efficient (poly(n) time) algorithm to construct such a Q.

(b) Show that if $R(z,x)$ is a bivariate polynomial and $g(x)$ a univariate polynomial then $z - g(x)$ divides $R(z,x)$ iff $R(g(x),x)$ is the 0 polynomial.

(c) Suppose $p(x)$ is a degree d polynomial that describes k of the points. Show that if d is an integer and $k > (d+1)(\lceil\sqrt{n}\rceil + 1)$ then $z - p(x)$ divides the bivariate polynomial $Q(z,x)$ described in part (a). (Aside: Note that this places an upperbound on the number of such polynomials. Can you improve this upperbound by other methods?)

(There is a randomized polynomial time algorithm due to Berlekamp that factors a bivariate polynomial. Using this we can efficiently recover all the polynomials p of the type described in (c). This completes the description of Sudan's algorithm for *list decoding*.)

PRINCETON UNIVERSITY F'02 COS 597D: A THEORIST'S TOOLKIT

Homework 3

Out: *October 21* Due: *November 4*

You can collaborate with your classmates, but be sure to list your collaborators with your answer. If you get help from a published source (book, paper etc.), cite that. Also, limit your answers to one page or less —you just need to give enough detail to convince me. If you suspect a problem is open, just say so and give reasons for your suspicion.

§1 Show that rank A for an $n \times n$ matrix A is the least k such that A can be expressed as the sum of k rank 1 matrices. (This characterization of rank is often useful.)

§2 Compute all eigenvalues and eigenvectors of the Laplacian of the boolean hypercube on $n = 2^k$ nodes.

§3 Suppose $\lambda_1 (= d) \geq \lambda_2 \geq \cdots \geq \lambda_n$ are the eigenvalues of the adjacency matrix of a connected d-regular graph G. Then show that $\alpha(G) \leq -\frac{n(\lambda_n}{d-\lambda_n}$.

§4 Let G be an n-vertex connected graph. Let λ_2 be the second largest eigenvalue of its adjacency matrix and x be the corresponding eigenvector. Then show that the subgraph induced on $S = \{i : x_i \geq 0\}$ is connected.

PRINCETON UNIVERSITY F'02 COS 597D: A THEORIST'S TOOLKIT

Homework 4

Out: *November 11* Due: *November 25*

You can collaborate with your classmates, but be sure to list your collaborators with your answer. If you get help from a published source (book, paper etc.), cite that. Also, limit your answers to one page or less —you just need to give enough detail to convince me. If you suspect a problem is open, just say so and give reasons for your suspicion.

§1 Estimate the mixing time of the random walk on the *lollipop* graph, which consists of a complete graph on n nodes and a path of length n attached to one of those nodes.

§2 Give an efficient algorithm for the following task, which is used for dimension reduction in many *clustering* algorithms. We are given n vectors $x_1, x_2, \ldots x_n \in \Re^n$ and a number k, and we desire the k dimensional subspace S of $\Re^n$ which minimizes the sum of the squared lengths of the projections of $x_1, x_2, \ldots, x_n$ to S. You may assume that eigenvalues and eigenvectors can be efficiently computed.

§3 In this question you will prove that random walks on constant degree expanders mix very rapidly: for any subset of vertices C that is fairly large, the probability that a random walk of length l avoids C is $\exp(-l)$. Let $G = (V, E)$ be an unweighted undirected graph and A be its adjacency matrix.

(a) Show that the number of walks of length l is $g^T A^l g$ where g is the all-1 vector.

(b) Suppose now that G is d-regular and has n vertices. Suppose that each of the eigenvalues of A except the largest (which is d) has magnitude at most λ. Let C be a subset of cn vertices and let A' be the adjacency matrix of the induced graph on $V \setminus C$. Show that every eigenvalue of A' is at most $(1-c)d + c\lambda$ in magnitude.

(c) Conclude that if $\lambda < 0.9d$ (i.e., G is an expander) and $c = 1/2$ then the probability that a random walk of length l in G (starting at a randomly chosen vertex) avoids C is at most $\exp(-l)$.