COMS 6998: Randomness in Computing — Oct 22, 2012

The Probabilistic Method

Instructor: *Xi Chen* — Scribes: *David Solimano*

1 Overview

The setup is that we have a finite set of objects and we want to show that at least one object in the set satisfies some property. Sometimes we have a more ambitious goal, in that we want to actually construct the object with the desired property, with either a randomized or deterministic polynomial time algorithm.

In general, we will define a probability distribution over a set A of objects. Then, we will prove that the probability that an object has the desired property is *strictly* greater than 0. The probability distribution can, in general, be any distribution, but we will often consider the uniform distribution as a special case.

A further reference for the probabilistic method, if one is desired, is "The Probabilistic Method" by Alon and Spencer.

2 Proving existence

For example 1, we will use the example of graph coloring to show that a coloring with a given property must exist.

Definition 1 (K_n)**.** *K_n is a complete graph with n vertices*

Theorem 2. *If $\binom{n}{k}2^{-\binom{k}{2}+1} < 1$, then there exists a 2-coloring of the edges of K_n such that it has no monochromatic clique of size k.*

This is theorem 6.1 in PC. We will use the uniform distribution. We can pick a coloring randomly, or equivalently randomly assign colors to edges.

Proof. The number of cliques of size k is $\binom{n}{k}$. Give them an arbitrary ordering $C_1, C_2, \ldots, C_{\binom{n}{k}}$. If A_i is the event that a given C_i is monochromatic,

$$\Pr(A_i) = 2^{-\binom{k}{2}+1}$$

. So, by the union bound,

$$\Pr\left(\cup A_i\right) \leq \sum \Pr\left(A_i\right) = \binom{n}{k} 2^{-\binom{k}{2}+1} < 1$$

Which means that there must be a case where this property isn't satisfied, that is, there is a 2-coloring with no such monochromatic clique of size k.

□

3 Efficient construction

For example 2, we will find and output the maximum cut of a graph that is at least a given size. We will show both a randomized and deterministic algorithm for building the cut.

3.1 Max-Cut existence

Definition 3 (Max-Cut)**.** *Given a graph $G = (V, E)$, we want to output a partition (A, B) to maximze the count $C(A, B)$ of edges between A and B.*

First let's define an indicator random variable that will prove useful.

Definition 4 (X_e)**.** *For each edge $e \in E$,*

$$X_e = \begin{cases} 1 & \text{if edge } e \text{ is between } A \text{ and } B \\ 0 & \text{otherwise} \end{cases}$$

We can use this variable and some properties of expectation to prove that the partition exists.

Theorem 5. *if $m = |E|$, then there exists a partition of V into A, B such that $C(A, B) \geq \frac{m}{2}$*

Proof. We will again use a uniform distribution. For each vertex $v \in V$, put it in A or B with probability $\frac{1}{2}$. Note that

$$E\left[\sum_{e \in E} X_e\right] = \sum_{e \in E} E\left[X_e\right] = \frac{1}{2}m$$

We can see this because each edge has one endpoint in each of A,B with probability $\frac{1}{2}$ and there are m edges. From this, we can tell that

$$\Pr\left[\sum_{e \in E} X_e \geq \frac{m}{2}\right] > 0$$

This is true because for our expectation to be $\frac{m}{2}$, there must be at least one element greater than or equal to and one element less than or equal to the expectation.

□

3.2 Max-Cut randomized construction

So now we know that for any graph, there must be a maximum cut of size at least $\frac{|E|}{2}$. But what if we want to know a partition that gives a cut at least that big? We can use a randomized algorithm that builds the graph as it flips coins to put vertices into one of the two sets A,B and at the end output the generated graph or report failure.

Definition 6 (Probability of success)**.** $p = \Pr\left[\sum_{e \in E} X_e \geq \frac{m}{2}\right]$

We want to find a lower bound on this p. It's not immediately apparent how to find this. We can't, for example, use a Markov bound. However, a little analysis will yield a useful lower bound.

Theorem 7 (Lower bound on P)**.** $P \geq \Omega\left(\frac{1}{m}\right)$

Proof. From the first part of the problem, and the definition of expectation, we know that

$$E\left[\sum_{e\in E} X_e\right] = \frac{m}{2} = \sum_{i=0} mi \Pr\left[\sum_{e\in E} X_e = i\right]$$

Let's assume, wlog, that m is even and separate it into two parts, using $\frac{m}{2}$ as the division point.

$$= \sum_{i<\frac{m}{2}} i \Pr\left[\sum_{e\in E} X_e = i\right] + \sum_{i\geq\frac{m}{2}} i \Pr\left[\sum_{e\in E} X_e = i\right]$$

Note that $i < m$. The second half of the previous expression can contribute at most pm to our total. From the fact that our function is integer-valued, we can also bound the left side. This gives us

$$\frac{m}{2} \leq pm + (1-p)\left(\frac{m}{2} - 1\right)$$

From this we find that

$$p \geq \frac{1}{1+\frac{m}{2}} \approx \frac{2}{m}$$

□

This means that, if we run our randomized algorithm $O(m)$ times, with probability $\frac{1}{2}$ I will get a partition with $\frac{m}{2}$ edges between A and B.

3.3 Max-Cut deterministic construction

We know that if we pick a partition uniformly at random,

$$E\left[C\left(A,B\right)\right] = \frac{m}{2}$$

But, we can give a greedy algorithm which will give us a cut of the size that we desire. Let $v_i \ldots v_n$ be a numbering of the vertices. Let's define an 0-1 random variable

Definition 8 (b_i)**.**

$$b_i = \begin{cases} 1 & \textit{if vertex } i \textit{ is in } A \\ 0 & \textit{if vertex } i \textit{ is in } B \end{cases}$$

What does $E\left[C\left(A,B\right)|b_1 b_2 \ldots b_n\right]$ indicate? This is the size of the cut, given that we have made all of the decisions on where to put vertices. We will give an algorithm that specifies these choices step by step such that

$$E\left[C\left(A,B\right)|b_1 b_2 \ldots b_i\right] \leq E\left[C\left(A,B\right)|b_1 b_2 \ldots b_{i+1}\right] \forall i$$

We can look at the expectation at the ith decision, and show that it always goes up at i+1. Then,

$$\frac{m}{2} = E\left[C\left(A,B\right)\right] \leq E\left[C\left(A,B\right)|b_1\right]$$
$$\leq E\left[C\left(A,B\right)|b_1 b_2\right] \leq E\left[C\left(A,B\right)|b_1 b_2 \ldots b_n\right]$$

At the final step of the algorithm, we have no randomness left. So a deterministic algorithm that behaves in this fashion would always output a partition where $C(A,B) \geq \frac{m}{2}$

Now, how do we know that we can make these choices of b_{i+1}? Let's consider what we know when we've picked $b_1 \dots b_i$. Now b_{i+1} can be randomly placed in either set. So,

$$E[C(A,B)|b_1 b_2 \dots b_i] = \frac{1}{2}E[C(A,B)|b_1 b_2 \dots b_{i+1} = 0] + \frac{1}{2}E[C(A,B)|b_1 b_2 \dots b_{i+1} = 1]$$

This tells us that the maximum of these two must be at least as much as the expectation. So if we can pick the one that is larger, we can always make the right decision as to where to place vertex i. A bit of thought indicates that we know a good forumla for calculating $\frac{1}{2}E[C(A,B)|b_1 b_2 \dots b_i]$. For the already placed vertices, we can count the number of edges crossing the partition A, B. Then we can estimate that $\frac{1}{2}$ of the remaining edges will cross the partition.

Let's start our algorithm by setting $b_1 = 0$ which tells us that $E[C(A,B)|b_1] = \frac{m}{2}$. Now, we can visualize a binary search tree. Each level corresponds to making a choice for b_i, resulting in n levels. The previous inequality guides us in computing the values of each choice efficiently. It turns out that we can make a greedy choice - if we always choose b_i such that the choice of where to put v_i maximizes the edge crossings between v_i and the previously placed vertices, we will maximize the size of the cut.

4 Set balancing revisited

4.1 Overview and Review

Definition 9 (The set-balancing problem)**.** *Find a* $b \in \{\pm 1\}^n$ *in order to minimize* $\|\mathbf{A}b\|_\infty$.

Definition 10 (E_i)**.** E_i *is the event that* $|\mathbf{A}_i b| > 4\sqrt{n \ln n}$

Definition 11 (E)**.** E *is the event that* $\|\mathbf{A}b\|_\infty > 4\sqrt{n \ln n}$

When we wanted to find a probability bound for this problem, we found that if pick the elements b_i of b uniformly at random, then $\Pr[E_i] \leq \frac{2}{n^2}$ from the Chernoff bound, and $\Pr[E] \leq \frac{2}{n}$ where $E = \cup E_i$ from the union bound.

Observation 12. *Using the probabilistic method, we therefore know that there exists a* b *such that* $|\mathbf{A}b| \leq 4\sqrt{n \ln n}$.

We will derandomize to actually construct this b, in polynomial time, by finding each element b_i one by one such that the given property holds. We know the vector exists, from the probability being greater than 0 of finding it randomly. Looking at the decision tree of the b_is, we are trying to find a path through the decision tree so that at the bottom of the tree we have found the vector with the desired property. Let's say that, in the course of running the algorithm, we are somewhere in the middle of the decision tree. To procede, I need two things.

1. I need to associate the two children of my node with a number, which represents the probability of event E conditioned on the decisions so far, and follow the smaller one
2. I need to efficiently calculate these numbers

4.2 Decision tree traversal

The first problem is that it is difficult to compute $E\left[\|\mathbf{A}b\|_\infty | b_1 \dots b_i\right]$ efficiently, because of the *max* function inherent in the norm. However, computing the expectation of $\mathbf{A}_i b$ is easy.

Definition 13 (a)**.** *a is a node in the tree such that $b_1 \dots b_{i-1}$ is fixed, and the further decisions are not yet fixed.*

So if we try to calculate

$$\Pr\left[\|\mathbf{A}b\|_\infty > 4\sqrt{n \ln n} | a\right]$$

We must take into account all of the decisions thus far. Let's start at the root of the tree, which corresponds to making the decision for b_1.

Observation 14 (Initial bound)**.** *At the root,* $\Pr[root] \leq \frac{2}{n}$ *from the Chernoff bound proof.*

So I need to choose a path with the smaller probability of going over our limit. Ideally, I would compute the probablity for each child node and take the lower one.

Lemma 15 (Smaller child)**.** *We know that one child must have a smaller property, from the usual argument about expectations.*

Proof. If I call the two children of the root b and c,

$$\Pr[root] = \frac{1}{2}\Pr\left[\|\mathbf{A}b\|_\infty > 4\sqrt{n \ln n} | b\right] + \frac{1}{2}\Pr\left[\|\mathbf{A}b\|_\infty > 4\sqrt{n \ln n} | c\right]$$

□

So if I can compute these two numbers efficiently, I can walk down the tree in a straightforward manner and find the vector that I'm looking for.

Now let's consider what happens at a leaf e, when all of the decisions have already been made. Let *bad* denote picking a b that satisfies event E.

Observation 16.

$$\Pr[bad|e] \leq \Pr[bad] < \frac{2}{n}$$

Claim 17. *We have not picked a bad vector*

Proof. Consider that

$$\Pr[bad|e] \in 0, 1$$

This is because we have already made the decisions, so our vector is either bad or not. However, since we know that this is $\leq \frac{2}{n}$, then we know that the vector picked is good and $\Pr[bad|e] = 0$ □

This is similar to what we did in the max-cut problem, except we have followed probabilities down the tree instead of expectations.

4.3 Efficient computation

Remember that E is the event that we pick a bad vector, and a is a subsequence of the choices that we need to make.

Definition 18 ($\bar{P}(a)$)**.** $\bar{P}(a) = \sum_{i=1}^{n} \Pr[E_i|a]$

We note that

$$\Pr[E|a] \leq \sum_{i=1}^{n} \Pr[E_i|a] = \bar{P}(a)$$

Now $\bar{P}(a)$ is quite easy to compute. We can use this to efficiently generate a vector.

Theorem 19. *Using $\bar{P}(x)$ as a metric we can walk the decision tree of the set-balancing problem and generate a valid vector b.*

Proof. We know that $\bar{P}(root) \leq \frac{2}{n}$. We also know, that at node a, one child node c must have a $\bar{P}(c) \leq \bar{P}(a)$. So we can keep following the path of smaller children to a leaf node e. At this point, we know that

$$\bar{P}(e) < \bar{P}(root) < \frac{2}{n}$$

and from this

$$\bar{P}(e) = \sum_{i} \Pr[E_i|e] > \sum_{i} \Pr[E|e]$$

Given e, $\Pr[E|e]$ must be either 0 or 1. Putting these two equations together, we know that it must be 0, since it is bounded less than 1. So this e is a guaranteed good choice, and by looking at the decisions that we made to get to e, we can see what the vector b is.

□

5 Nash equliibrium

5.1 Problem overview

We have two matrices, $\mathbf{A}$ and $\mathbf{B}$, representing two players engaging in a game. Each is a $n \times n$, $0-1$ matrix. The entries $\mathbf{A}_{ij}$ represent the payoff to player 1 if he does action i and player 2 does action j, and likewise for player 2 and $\mathbf{B}$

Definition 20 (Δ_n)**.** *The set of all n-dimensional distributions $x \in R_+^n$ such that $\sum x_i = 1$*

Definition 21 (Nash equilibrium)**.** *A Nash equilibrium is a pair of distributions, $x, y \in \Delta_n$ where*

$$x^T \mathbf{A} y \geq \left(x'\right)^T \mathbf{A} y, \forall x' \in \Delta_n$$
$$x^T \mathbf{B} y \geq x^T \mathbf{B} y', \forall y' \in \Delta_n$$

Now $x\mathbf{A}y$ is the expected playoff for p_1 and likewise $x\mathbf{B}y$ is the expected payoff for p_2. So a Nash equilibrium is when neither player can increase their payoff by changing their distributions of x and y.

5.2 Approximation problem

Definition 22 (ϵ approximation)**.** *A Nash equilibrium is a pair of distributions, $x, y \in \Delta_n$ and an ϵ where*

$$x^T \mathbf{A} y \geq \left(x'\right)^T \mathbf{A} y - \epsilon, \forall x' \in \Delta_n$$
$$x^T \mathbf{B} y \geq x^T \mathbf{B} y' - \epsilon, \forall y' \in \Delta_n$$

so that x, y is worse than the optimal solution by at most ϵ.

This problem is interesting because a Nash equilibrium always exists, but may not be easy to find efficiently. But an epsilon approximation can be found efficiently. We will sketch this proof.

5.3 Efficiently finding an approximation

Theorem 23 (Efficiency bound)**.** *$\forall \epsilon > 0$, one can find an ϵ-approximate Nash equilibrium in time*

$$n^{O(\frac{\log n}{\epsilon^2})}$$

So for any constant ϵ, we can get a $n^{\log n}$ algorithm. However a polynomial time algorithm is only known for $\epsilon \geq \frac{1}{3}$.

Let's start with a definition.

Definition 24 (K-good)**.** *A distribution x is k-good if*

$$x_i \in \frac{0}{k}, \frac{1}{k}, \frac{2}{k}, \ldots \frac{k}{k}$$

$\forall i \in [n]$

Theorem 25 (Efficienty bound in terms of k-good distribution)**.** *$\forall \epsilon > 0$, $\exists \epsilon -$ approximate Nash equilibrium (x, y) such that (x, y) are k-good where*

$$k = O\left(\frac{\log n}{\epsilon^2}\right)$$

Observation 26. *From here, we can get to the original efficienty bound theorem by enumerating all of the k-good vectors.*

Also, it's easy to check if we have satisfied the ϵ-approximation property. We will try to prove the second theorem, by generating the desired x and y with the probabilistic method. We can't use the uniform distribution as we did in the previous problems; we will need to take the matrices $\mathbf{A}$ and $\mathbf{B}$ into account.

Proof. We will start by assuming that we know a Nash equilibrium (x, y). We will call the particular one (x^*, y^*). We know this must exist from Nash's theorem.

Define a distribution of k-good vectors x. We will use the balls and bins problem to generate this.

$$x : \text{put k balls into n bins, following } x^*$$

The first ball goes into bin i with probability x_i^*. In general,

$$x_i = \frac{\text{number of balls in bin i}}{k}$$

We define y_i similarly using y^*. If we let $k = \frac{100 \log n}{\epsilon^2}$, then

$$\Pr\left[(x, y)\text{is an } \epsilon\text{-approximate NE}\right] > 0$$

which can be proved using a Hoeffding bound.

□

Let's say I can show the above probability is close to 1.. Does this give me a Monte Carlo for this problem? It does not, because I don't know (x^*, y^*). Instead, we will give a sketch of how we can use a union bound to help prove the probability above is positive. By the definition of the Nash equilibrium, we only need to show that

$$\Pr\left[x^T \mathbf{A} y \geq \left(x'\right)^T \mathbf{A} y - \epsilon, \forall x' \wedge x^T \mathbf{B} y \geq x^T \mathbf{B} y' - \epsilon, \forall y'\right] > 0$$

$$\Pr\left[x^T \mathbf{A} y \geq \left(x'\right)^T \mathbf{A} y - \epsilon, \exists x' \vee x^T \mathbf{A} y \geq x^T \mathbf{A} y' - \epsilon, \exists y'\right] << 1$$

The problem with using a union bound is, x' is for all distributions. Here we can use a useful simplification from linear algebra to help us apply the union bound.

Lemma 27.

$$\max_{x' \in \Delta_n} \left(x'\right) \mathbf{A} y = \max_{i \in [x]} \mathbf{A}_{i*} y$$

using this, we can split the second probability into $2n$ events, so invoking the union bound we find that it is

$$\leq \sum_{i=1}^{n} \Pr\left[x^t \mathbf{A} y \leq \mathbf{A}_{i*} y - \epsilon\right] + \sum_{i=1}^{n} \Pr\left[x^t \mathbf{B} y \leq x \mathbf{B}_{i*} - \epsilon\right]$$

We then apply the Hoeffding bound on each probability above. Details can be found in the paper "Playing Large Games using Simple Strategies", by Lipton, Markakis, and Mehta.