18.409: An Algorithmist’s Toolkit
Lecture 5

Jonathan Kelner

Administrivia
- Reading for this lecture
  - I posted two sources for approximating the permanent
  - The original paper
  - An excerpt from a textbook
    - I had to protect this one for copyright reasons
    - Tell me if you have a problem

Today
- Monte Carlo methods and approximate counting
- Approximately computing the permanent of dense matrices

MONTE CARLO METHODS AND APPROXIMATE COUNTING
Recall from Last Time
- Have some set $V$ of size $Z$, want to know how many elements are in some subset $S$
- Pick $N$ points of $V$ uniformly at random
- If $q$ of them are in $S$, return $Zq/k$
- Has the right expectation, but how tightly is our estimate concentrated around it?

More from Last Time
- **Chernoff bound**: If you want an $\epsilon$-approximation with probability 1-$\delta$, need
  \[ N \geq \Theta\left(\frac{\log \frac{\delta}{\epsilon^2}}{p\epsilon^2}\right) \]
  where $p$ is probability that a sample in your bigger set ends up in your smaller set
- In words: need enough trials to get $\Theta(\log \frac{\delta}{\epsilon^2})$ successes

Potential Problems
- Subset we’re studying is much smaller than ambient space
  - So need a very large number of samples to get a reasonable number of successes
- Have a complicated ambient space from which it’s hard to draw samples
- Today we’ll deal with both types of problems

A Little Terminology
- An $\epsilon, \delta$ approximation scheme for some quantity is an algorithm for computing an $\epsilon$-approximation with probability at least 1-$\delta$
- A fully polynomial randomized approximation scheme (FPRAS) is an $(\epsilon,\delta)$ approximation scheme that runs in time $\text{poly}(n, 1/\epsilon, \log 1/\delta)$
A Nontrivial Example: Counting DNF Solutions

- Have boolean variables $x_1, ..., x_n$
  - “Literal” means an $x_i$ or its negation
- Recall that a DNF formula $F$ is a disjunction (OR) of conjunctive (AND) clauses:
  $$F = C_1 \lor ... \lor C_m$$
  where each $C_i$ is an AND of literals, e.g.,
  $$F = (x_1 \land \overline{x_3}) \lor (x_2) \lor (x_2 \land \overline{x_1} \land x_3 \land x_4)$$

Counting DNF Solutions

$$F = (x_1 \land \overline{x_3}) \lor (x_2) \lor (x_2 \land \overline{x_1} \land x_3 \land x_4)$$

- Easy to check if there exists satisfying assignment
- Really hard to exactly count them
  - #P hard (so at least as hard as NP-complete problems)
- We’ll see how to, in poly time, get an $\varepsilon$-approximation with probability $> 1-1/2^n$
  - I.e., a FPRAS

How Not to Do It

- Naive Monte Carlo algorithm
- One trial:
  - Randomly assign 0 or 1 to each $x_i$
  - Check if this is a satisfying assignment
  - Repeat $m$ times, and say get $k$ successes
  - Estimate
    - # solutions $= (k/m) \cdot 2^n$
  - Why not?
    - #solutions/ $2^n$ can be exponentially small
    - So need exponentially many trials...

How to Fix the Problem

- Clauses
  - Assignments

<table>
<thead>
<tr>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0 0</td>
</tr>
<tr>
<td>0 1 0 1 1</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>1 1 0 1 1</td>
</tr>
<tr>
<td>0 0 0 1 0</td>
</tr>
<tr>
<td>0 0 0 0 1</td>
</tr>
</tbody>
</table>
How to Fix the Problem

- Want to count yellow boxes
- We’ll:
  1. Sample \( X = \{\text{blue}\} \cup \{\text{yellow}\} \)
  2. Count number yellow we get
  3. Multiply by \( |X| \)
- Now success prob \( \geq \frac{1}{m} \)
- Sample by col instead of by row
- If clause \( i \) has \( k_i \) vars in it, its col has \( 2^{n-k_i} \) satisfying assigs (and we know what they are)
- So sample each col proportional to \( 2^{n-k_i} \)
- Then pick a random sat. assig. of this clause and see if it’s first in row
- \( |X| = \sum 2^{n-k_i} \)

The Permanent

- Can easily compute in poly time
  \[ \det(M) = \sum_{\pi \in S_n} \text{sgn}(\pi) \prod_{i=1}^{n} m_{i, \pi(i)} \]
- Get permanent by dropping the signs
  \[ \text{per}(M) = \sum \prod_{\pi \in S_n} m_{i, \pi(i)} \]
- Problem becomes much harder— #P-complete
  - So no poly time algorithm unless \( P = \text{NP} \)
We're going to look at $M$ where all entries are 0 or 1
- Still #P-hard
- Has interpretation as number of perfect matchings in a bipartite graph

This is what we'll try to approximate
- **Surprising result:** Although #P-hard to compute permanent of 0-1 matrices exactly, it has a FPRAS

### A (Very) Little History

- **1989:** Jerrum and Sinclair
  - Permanent for *dense* graphs
    - Every vertex has degree at least $n/2$
    - Still #P-complete
- **2001:** Jerrum, Sinclair, and Vigoda
  - Permanent for arbitrary graphs
  - And thus for any matrix with nonnegative entries
- We'll just do the dense case

### What Won’t Work

- Can’t do any naive Monte Carlo algorithm
  - E.g., Guess a random permutation and see if it gives a matching
  - Anything like this will require you to estimate exponentially small quantities

### General Strategy

- Will look at all (possibly partial) matchings, not just perfect ones
- Let $M_k$ be set of matchings of size $k$
- Suppose we had a black box for sampling from $M_k \cup M_{k-1}$ uniformly at random for any $k$
- Let $r_k = |M_k|/|M_{k-1}|$
- Also suppose $1/\alpha \leq r_k \leq \alpha$ for some poly size $\alpha$
- Can estimate each $r_k$ within relative error $1+1/n^2$ with very high probability
- Estimate answer using $M_n = M_1 \cdot \prod_{i=2}^{n} r_i$
Main question: how do you sample from $M_k \cup M_{k-1}$?

Other issues:
- Will actually only sample approximately uniformly
  - Not a big deal, as long as we’re close enough
- All of our approximations will have probability of failure
  - Just make all of these numbers really small
- Don’t know that there’s a polynomial bound on the $r_k$
  - This will be guaranteed by our density assumption

Bounding the $r_k$

**Theorem:** Let $G$ be a bipartite graph with min degree $\geq n/2$. Then $1/n^2 \leq r_k \leq n^2$ for all $k$.

**Upper bound:**
- For every matching in $M_k$, (arbitrarily) pick a submatching in $M_{k-1}$ as a representative
- No matching in $M_{k-1}$ can be picked more than $(n-k+1)^2 \leq n^2$ times
  - Because has to pick one new edge $= 2$ unmatched vertices
- Implies $|M_k| \leq n^2 |M_{k-1}|$

Bounding the $r_k$ (cont.)

Upper bound follows from:

**Lemma:** In a graph of min degree $\geq n/2$, every partial matching in $M_{k-1}$ has an augmenting path of length $\leq 3$
- That is, you can augment it by either:
  - Adding an edge to the existing matching
  - Removing one edge then adding two more, one hitting each endpoint
- **Proof:** On blackboard
  - Fix some $m \in M_k$
  - At most $k$ matchings in $M_{k-1}$ can be augmented by path of length 1 to equal $m$
  - At most $k(k-1)$ can be augmented by path of length 3
  - So $|M_{k-1}| \leq k+k(k-1)|M_k| \leq n^2 |M_k|$

Canonical Paths

All that’s left is to show how to sample from $C_k = M_k \cup M_{k-1}$
- Will do this by making a random walk on $C_k$ with uniform stationary distribution and showing it mixes quickly
- Will show this by bounding $\Phi(C_k)$
- Will do so with a general technique known as canonical paths
Let \( G = (V, E) \) be any graph for which want to bound \( \Phi(G) \).

Will be applying this to (exponentially large) \( C_k \).

For every two vertices \( v, w \in V \), will specify a *canonical path* from \( v \) to \( w \).

Suppose no edge occurs on more than \( bn \) paths.

**Claim:** Implies \( \Phi(G) \geq 1/(2b d_{\text{max}}) \).

**Proof:**

\[
\Phi(G) = \min_{S \subseteq V} \frac{e(S)}{\min \{ \sum_{v \in S} d(v), \sum_{v \in \bar{S}} d(v) \}}
\]

For any \( S \subseteq V \), # of paths crossing cut is

\[
|S|(n - |S|) \geq |S|n/2
\]

At most \( bn \) paths through each edge, so # of edges crossing cut is \( \geq |S|/2b \).

Bound denominator using max degree.

---

**The Graph \( C_k \)**

- Will just do \( C_n \) for now.
- Vertices correspond to matchings in \( M_n \cup M_{n-1} \).
- (Directed) edges of 4 kinds:
  - Reduce: If \( m \in M_n \) and \( e \in m \), go to \( m' = m-e \).
  - Augment: If \( m \in M_{n-1} \) and \( \{u, v\} \) unmatched, move to \( m' = m+\{u,v\} \).
  - Rotate: If \( m \in M_{n-1} \) and \( u \) matched to \( v \), move to \( m' = m+\{u,v\} \).
  - Self loops: Stay where you are; add enough to make degree \( = 2|E| \) at each vertex.
- Want undirected graph—**Reduce** one way = **Augment** the other.

---

**The Canonical Paths**

To every node \( s \in M_n \cup M_{n-1} \), associate a “partner” \( s' \in M_n \).

- If \( s \in M_n \), \( s' = s \).
- If \( s \in M_{n-1} \) and has augmenting path of length 1, add corresponding edge to get \( s' \).
- If \( s \in M_{n-1} \) and has shortest augmenting path of length 3, let \( s' \) be matching it gives.

Path from \( s \) to \( t \) will have 3 segments:

\( s \) to \( s' \) (Type A),

\( s' \) to \( t' \) (Type B),

\( t' \) to \( t \) (Type A).
Type A Paths

- Type A paths connect a vertex s to its partner s'.
- If s ∈ M, s = s' and path is empty.
- If s ∈ M_{n-1} with augmenting path of length one, path is of length 1.
  - The edge of the corresponding Augment.
- If s ∈ M_{n-1} with augmenting path of length 3, path will be of length 2.
  - First Rotate, then Augment.
  - Picture on next slide.

Type B Paths

- s and t are each perfect matchings.
- Let d = s ⊖ t (symmetric difference).
- d consists of a collection of disjoint, even-length, alternating cycles of length ≥ 4.
  - Edge of s then t then s then t....
- Path from s to t will “unwind” each cycle.
  - Somehow order the cycles so there’s a canonical order to do this in.
  - Also need to canonically specify a “start” vertex in each cycle.
  - See picture on next slide.
Lemma: Let \( s \in M_n \). At most \( O(n^2) \) other nodes \( s' \in M_n \cup M_{n-1} \) have \( s \) as their partner.

Proof:
- Only one such \( s' \in M_n \)
  - \( s \) itself
  - Otherwise can get to \( s' \) from \( s \) by either a Reduce edge or (Rotate, Augment) pair of edges
- At most \( O(n) \) ways to Reduce, \( O(n^2) \) ways to (Rotate, Augment)

That lets us understand type A paths; let’s now look at type B paths

Fix one transition \( T \) (i.e., an edge of \( C_n \))

We’ll count how many pairs \( (s,t) \in M_n \times M_n \) contain \( T \) on their type B canonical path

Claim: Number is \( \leq |C_n| \)
- Implies \( T \) is on \( \leq O(n^2 \cdot |C_n| \cdot n^2) \) canonical paths
- So \( \Phi(C_n) \geq 1/(2n^4 d_{\text{max}}) = \Theta(1/n^6) \)
- Means lazy random walk mixes in poly number of steps, which is what we wanted

More on Counting Canonical Paths
- We’ll bound such pairs by injectively mapping them into \( C_n \)
  - So need a map \( \sigma \) \{ \( s,t \) \} that takes a pair (starting matching, ending matching) \( \rightarrow M_n \cup M_{n-1} \)
  - Given \( T \) and result, need to be able to figure out which \( s \) and \( t \) it came from
- Loosely:
  - \( \sigma_T \) will agree with \( s \) on cycles we’ve already unwound and on part of current cycle we’ve already unwound
  - \( \sigma_T \) will agree with \( t \) elsewhere
- Three cases:
  - \( T = \text{Reduce} \) at beginning of an unwinding
  - \( T = \text{Augment} \) at end of an unwinding
  - \( T = \text{Rotate} \) in the middle
- See next slide for pictures
Dealing with Smaller Matchings

- This was for perfect matchings, but need to get $r_k$ for all $k$
- Could do similar argument and show mixing
- But a better trick: reduce one problem to the other
- Create a (sufficiently dense) graph $G_k$ whose perfect and near-perfect matchings let you count $k$- and $(k-1)$-matchings of $G$

Whole Argument Review

- **General principle:** if we can sample, we can (often) approximately count
- Lots of exponential gaps, so need to go in stages
- Estimate ratios $|M_k|/|M_{k-1}|$ by sampling from $M_k \cup M_{k-1}$
- Sampling directly is hard, so instead take random walk
- Show random walk mixes (i.e., gets close to the uniform distribution) in a polynomial number of steps