9/13

LP Duality

Primal LP:

Variables $\vec{x} = \langle x_1, \ldots, x_n \rangle$

maximize $\sum_{i=1}^{n} c_i x_i$ subject to constraints

$$\sum_{i=1}^{n} A_{ji} x_i \le b_j \quad , j \in \{1, \ldots, m\}$$

$$x_i \ge 0 \quad \forall i \in \{1, \ldots, n\}$$

To find the dual, construct a variable w_j for each j s.t.

(1) $w_j \ge 0 \Rightarrow$ any feasible $\vec{x}$ satisfies

$$\sum_{i=1}^{n} \left(\sum_{j=1}^{m} A_{ji} w_j \right) x_i \le \sum_{j=1}^{m} b_j w_j$$

(2) $\forall i, \ \sum_{j=1}^{m} A_{ji} w_j \ge c_i \Rightarrow$ any feasible $\vec{x}$ satisfies

$$\sum_{i=1}^{n} c_i x_i \le \sum_{i=1}^{n} \left(\sum_{j=1}^{m} A_{ji} w_j \right) x_i \le \sum_{j=1}^{m} b_j w_j$$

upper bound on objective of primal!

The **dual LP problem** is to find the <u>best upper bound.</u>
In other words,

variables $\langle w_1, \ldots, w_m \rangle = \vec{w}$

minimize $\sum_{j=1}^{m} b_j w_j$ subject to constraints

$$\sum_{j=1}^{m} A_{ji} w_j \ge c_i \quad \forall i \in \{1, \ldots, n\}$$

$$w_j \ge 0 \quad \forall j \in \{1, \ldots, m\}$$

So, we have a primal LP and the dual LP, which is the optimization problem for the best upper bound.

Theorem Weak LP Duality

(1) If the primal LP is unbounded ($+\infty$), the dual LP is infeasible.

(2) If the primal LP is finite, dual LP is finite & $\geq$ primal, or infeasible.

Proof:

(1) Suppose BWOC that the dual LP is feasible. Then, there is some upper bound on the primal LP. $\times$

(2) Any feasible solution of the dual must upper bound the primal.

$\square$

Theorem: Complementary Slackness

Consider feasible $\vec{x}$ for the primal and feasible $\vec{w}$ for dual. Then, the following are equivalent:

(1) $(w_j = 0 \text{ OR } \sum_i A_{ji} x_i = b_j \ \forall j)$ AND $(x_i = 0 \text{ OR } \sum_j A_{ji} w_j = c_i \ \forall i)$

(i.e. dual variable is 0 or primal bound j is tight, and vice versa)

(1*) $(\lambda_j = 0 \text{ OR } \sum_i A_{ji} x_i = b_j \ \forall j \in S)$ AND $\vec{x}$ optimal for LP_S^{λ}

(2) $\sum_i c_i x_i = \sum_j w_j b_j$ (i.e. $\vec{x}$ and $\vec{w}$ are both optimal)

(2*) $\vec{x}$ is optimal for LP & $\vec{\lambda}$ gives best upper bound

Proof: We can say the following:

$$\underbrace{\sum_i c_i x_i - \sum_j w_j b_j \leq \sum_i \Big(\sum_j A_{ji} w_j\Big) x_i - \sum_j w_j b_j}_{\text{holds because } \vec{w} \text{ is feasible}} = \underbrace{\sum_i \sum_j w_j (A_{ji} x_i - b_j) \leq 0}_{\text{holds because } \vec{x}, \vec{w} \text{ feasible}}$$

Condition (2) $\Leftrightarrow$ this whole inequality being tight.

Condition (1) $\Leftrightarrow$ no terms in the two middle sums.

So, (1) $\Leftrightarrow$ (2). $\square$

Def: Begin with any primal LP. Then, the Lagrangian relaxation w.r.t. $\vec{\lambda}$ ($\lambda_j \geq 0 \ \forall j \in S$ where $S \subseteq \{1, \ldots, m\}$) is

$$LP_S^{\lambda} = \text{Maximize } \sum_i c_i x_i + \sum_{j \in S} \lambda_j \Big(b_j - \sum_i A_{ji} x_i\Big) \text{ subject to}$$

$$\sum_i A_{ji} x_i \leq b_j \quad \forall j \notin S$$

$$x_i \geq 0 \quad \forall i$$

(move some j's from constraint to objective)

Theorem: Weak Lagrangian Duality

$\forall S \subseteq \{1, \ldots, m\}$ and $\forall \vec{\lambda}$, $LP_S^{\lambda} \geq LP$

Proof: Let $\vec{x}$ be feasible for LP. Then, $\vec{x}$ must be feasible for LP_S^{λ}.
Also, since $\lambda_j \geq 0 \ \forall j$, $(b_j - \sum_i A_{ji} x_i)\lambda_j \geq 0$
So, we relax by allowing a larger space of feasible solutions and also by increasing the optimum.

Observe that we can search for

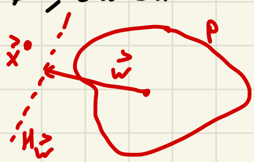
$$\text{Best upper bound} = \min_{\substack{\vec{\lambda} \text{ with} \\ \lambda_j \geq 0 \ \forall j \in S}} \left\{ \max_{\substack{\vec{x} \text{ feasible} \\ \text{for } \notin S}} \Big\{ \overbrace{\sum_i c_i x_i + \sum_{j \in S} \lambda_j \big(b_j - \sum_i A_{ji} x_i\big)}^{\text{objective for } LP_S^{\lambda}} \Big\} \right\}$$

Once S is fixed, every $\vec{\lambda}$ gives you a program, and every such program bounds the primal.

Note: when $S = \{1, \ldots, m\}$ this best upper bound search is equivalent to the dual LP.

Theorem: Seperating Hyperplane Theorem

Let P be a closed, convex region in $\mathbb{R}^n$ with $\vec{x} \notin P$.
Then, $\forall \vec{x} \notin P$, $\exists \vec{w} \in \mathbb{R}^n$ s.t. $\vec{x} \cdot \vec{w} > \max_{\vec{y} \in P} \{\vec{y} \cdot \vec{w}\}$



$H_{\vec{w}}$ is hyperplane $\vec{z} \in \mathbb{R}^n$ with $\vec{z} \cdot \vec{w} = \text{constant}$.
Then, there is some $\max_{\vec{y} \in P} \{\vec{y} \cdot \vec{w}\}$, and $\vec{x} \cdot \vec{w}$ is larger.

Lemma: Let $\vec{x}$ solve the primal LP, and let $S = \{j : \sum_i A_{ji} x_i = b_j\}$
Then there exist $\{\lambda_j\}_{j \in S}$ s.t. $\lambda_j \geq 0 \ \forall j \in S$ and $c_i = \sum_{j \in S} \lambda_j A_{ji} \ \forall i$.
(i.e. for each condition j that $\vec{x}$ tightly fits, there is a nice multiplier).

Proof: Let $X = \{\vec{y} : \exists \{\lambda_j \geq 0\}_{j \in S} \text{ s.t. } y_i = \sum_{j \in S} \lambda_j A_{ji}\}$
X is closed and convex. So, with the separating hyperplane theorem, if $\vec{c} \notin X$ we can improve our solution $\vec{x}$. So, $\vec{c} \in X$.
$\square$

Theorem: Strong LP Duality

(1) If primal is unbounded, the dual is infeasible.
(2) If the primal is finite, the dual and primal are equal.
(3) If the primal is infeasible, the dual is infeasible or unbounded.

Proof:

Set $w_j = \lambda_j \; \forall j \in S$, $w_j = 0 \; \forall j \notin S$

Because of the Lemma, this is a feasible dual solution.

Now,

$$\sum_j b_j w_j = \sum_{j \in S} b_j w_j + \sum_{j \notin S} b_j w_j = \sum_{j \in S} \left(\sum_i A_{ji} x_i\right) w_j = \sum_i \Big(\underbrace{\sum_{j \in S} A_{ji} w_j}_{= c_i \text{ from Lemma}}\Big) x_i = \sum_i c_i x_i$$

Lecture 9/15

LP Rounding

Motivating Vibes: Turn NP-Hard problem to integer program, solve normal LP, apply finesse to get integer solution.

ex/ Max-weight bipartite matching:

given bipartite $G(V = A \cup B, E \subseteq A \times B)$ and $w: E \to \mathbb{R}$ weights, find matching set M of edges s.t. no node appears > once that maximizes $\max \sum_{e \in M} w_e$

(x_e is indicator for $e \in M$)

Define x_e as follows: x_e is an integer $\in [0,1]$

$x_e = 1 \Leftrightarrow e \in M \qquad x_e = 0 \Leftrightarrow e \notin M$

The problem is to maximize $\sum_e w_e \cdot x_e$

subject to $0 \le x_e \le 1 \; \forall e$

$\forall a \in A, \sum_{b \in B} x_{(a,b)} \le 1$

$\forall b \in B, \sum_{a \in A} x_{(a,b)} \le 1$

AND x_e IS AN INTEGER

this condition makes this an integer program.

If we remove this we get a LP that can be solved in poly-time, but with fractional solutions.

We use the Birkhoff-Von Neumann Theorem, which states that any fractional matching to a set of convex integer matchings. Choosing any of these randomly will, in expectation, achieve the fractional expectation.

ex2/ vertex cover (NP-Hard):

Given $G=(V,E)$, weight $w: V \to \mathbb{R}$, output the set $S \subseteq V$ s.t. $\forall e \in E$, at least one endpoint of e is in S and S minimizes $\sum_{i \in S} w_i$

To convert this to an integer program, define indicator variables

$$x_i = \begin{cases} 1 & v_i \in S \\ 0 & v_i \notin S \end{cases}.$$

Then, we get the problem

variables: $x_i \quad \forall i \in V$ ~~(and x_i integer)~~ get rid of this, solve resulting LP

minimize $\sum_{i \in V} w_i x_i$

subject to $0 \le x_i \le 1 \quad \forall i \in V$

$\forall (u,v) \in E, \quad x_u + x_v \ge 1$ ← at least one node is in cover

For a solution $\tilde{x}$ to the LP, we can try to get an integer solution by rounding: place $i \in S$ iff $x_i \ge \frac{1}{2}$

(note: this is the best poly-time algo. for vertex cover)

Thm: Rounding outputs a valid vertex cover.

Proof: $\forall (u,v) \in E$, $x_u + x_v \ge 1$. So, at least one of u, v must be in S. $\square$

Thm: Rounding outputs a 2-apx for the best vertex cover (i.e. $\sum_{i \in S} w_i \le 2 \sum_{i \in V} w_i x_i$)

Proof: $$\sum_{i \in V} w_i x_i = \underbrace{\sum_{i \text{ s.t. } x_i \ge \frac{1}{2}} w_i x_i}_{\text{rounding at most doubled these}} + \underbrace{\sum_{i \text{ s.t. } x_i < \frac{1}{2}} w_i x_i}_{\text{rounding sent these to } 0} \ge \sum_{i \in S} \frac{w_i}{2} + 0 = \sum_{i \in S} \frac{w_i}{2}$$ $\square$

ex/ Distributed computing

The problem: n jobs, m machines, must assign each job
processing job i on machine j takes time P_{ij}

The goal: Finish all jobs as quickly as possible.

Def indicator $x_{ij} = \begin{cases} 1 & \text{job } i \text{ put on machine } j \\ 0 & \text{else} \end{cases}$

$$\text{minimize} \underbrace{\max_j}_{\text{(slowest machine)}} \left\{ \sum_i x_{ij} P_{ij} \right\}$$

A mathematical framework for this problem is to

$$\text{minimize} \max_j \left\{ \sum_i x_{ij} P_{ij} \right\} \quad \text{subject to} \quad \forall i, \sum_j x_{ij} \geq 1$$

$$\forall i,j, \quad x_{ij} \in [0,1]$$

$$x_{ij} \text{ is integer}$$

If we define T as a variable s.t. $T \geq \sum_i x_{ij} P_{ij} \; \forall j$, minimizing T solves the program.

There is an integrality gap, i.e. there are instances where the best fractional solution $\leq \frac{1}{n}$ (best integral solution). ← consider case with 1 job, m machines, $P_{ij} = 1$. fractional best is $\frac{1}{m}$, best actual solution is 1.

This proves that any rounding to the relaxed LP's solution will suck.

Proof: Any rounding algorithm takes as input feasible $\vec{x}^*$ and outputs an integral $\vec{x}$ s.t. quality$(\vec{x}) \geq c$ quality$(\vec{x}^*)$ for constant c. Integrality gap disallows this.

□

If we add a constraint that considers the lower bound that all jobs must go somewhere, we can build

parameter, think of t like the integral optimum → $LP(t)$: minimize T

subject to $x_{ij} \in [0,1]$ $\qquad T \geq \sum_i P_{ij} x_{ij} \; \forall j$

$\sum_j x_{ij} \underline{\geq} 1 \; \forall i$ $\qquad x_{ij} = 0$ if $P_{ij} > t$

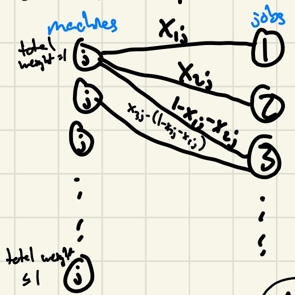
Key observation: If $t =$ integral optimum, the integral optimum is a feasible solution for $LP(t)$.

$\Rightarrow$ int. opt. $\geq t$ & int. opt. $\geq T^*(t)$ ← soln. to $LP(t)$

Rounding algorithm: $\forall$ machines j, make $\left\lceil \sum_i x_{ij} \right\rceil$ (ceil of # of jobs for machine j) copies of it as nodes $\in A$.

$\forall$ jobs i, make each one a single node $\in B$.

So, we get a bipartite graph. In A, there are multiple nodes for each machine. In B, there is one node for each job



↓ sorted in decreasing p_{ij}

Each job ends out exactly x_{ij} to some copy of j & higher copies of each machine get worse jobs

Note: every job has 1 or 2 edges to machines because it will either fit inside one copy or not.

In other words, we start at costliest jobs. The number of copies of machine j is given by the LP solution. Each copy has capacity 1. We go in decreasing order of jobs, putting/splitting it in the earliest copy we can to fill capacities. The last copy of machine j might not be filled.

Claim 1: Let T_j^c be the slowest job assigned to copy c of machine j. Then, $T^*(t) \geq \sum_{c=2}^{\lceil \sum_i x_{ij} \rceil} T_c^j \quad \forall j$ (# copies)

This is a consequence of the ordering of jobs in decreasing time. As we go to lower copies, they were filled by better jobs.

The algorithm is to find a complete matching in the graph and use it.

Claim 2: $T^*(f) \le t + \sum_{c=2}^{\lceil \sum_i x_{ij} \rceil} T_c^j \quad \forall j$

Lecture 9/20 - Ellipsoid Algorithm

Recall we refer to an LP

maximize $\sum_i c_i x_i$
subject to $\sum_j A_{ji} x_i \le b_i \quad \forall i$
$x_i \ge 0 \quad \forall i$

Sometimes we have disproportionately more constraints than variables.

Ex/ Semidefinite programming

$X \in \mathbb{R}^{n \times n}$ is positive semidefinite if $\forall \vec{a} \in \mathbb{R}^n$,
$\vec{a} X \vec{a}^T \ge 0 \iff \sum_{ij} X_{ij} a_i a_j \ge 0 \quad \forall \vec{a} \in \mathbb{R}^n$

If we want X being pos. semi def. to be a constraint, this is essentially infinitely many linear constraints.
This would still be an LP (linear objective, linear constraints), but you can't do anything in poly time over # of constraints.

Ex/ Traveling Salesman (visit every node in graph along min weight path)
Let d_{ij} = dist. from i to j, $\quad X_{ij} = \mathbb{1}(i,j \text{ in path})$
We can write an IP

minimize $\sum_{ij} d_{ij} X_{ij}$
subject to $x_{ij} \in \{0,1\} \quad \forall i,j$ (integer constraint)
$\sum_j x_{ij} = 2 \quad \forall i$ (enter + exit & nothing else)
$\sum_{i \in S} \sum_{j \notin S} X_{ij} \ge 0 \quad \forall S \subseteq V, S \ne \emptyset, S \ne V$
(every cut is crossed)

To relax this into an LP, we could remove the integer constraint. However, there are 2^n cut constraints (power set), so we don't want this approach.

These examples show that sometimes we wish to do something else. We generalize.

Def Convex Programming   Recall convex means $\frac{f(i)+f(j)}{2} \geq f(k)$ $\forall k \in (i,j)$

A **convex program** is of the form

$$\begin{aligned} &\text{minimize} && f(\vec{x}) \\ &\text{subject to} && \vec{x} \in K \\ & && f \text{ is convex} \\ & && K \text{ is convex + closed} \end{aligned}$$

A hard problem would be to only do this with a membership oracle for K and a function evaluation oracle for f.

We can ask for a stronger assumption: a separation oracle.

Def: A **separation oracle** for a closed convex region K takes as input $\vec{x}$ and outputs $\begin{cases} \text{"yes"} & \vec{x} \in K \\ \text{separating hyperplane } \vec{w} & \vec{x} \notin K \end{cases}$

A separation oracle can be thought of as a constraint verifier, where we either return "yes" or a violated constraint.

Consider now a convex program where all we are given is a linear objective $f(\vec{x})$ and a separation oracle.

Practice:

- We can make a separation oracle (constraint verifier) for the Traveling Salesman by solving MinCut (poly time) for the graph with X_{ij} weights, and verifying that the weight of the mincut is ≥ 2.

- We can make a separation oracle for the Semidefinite Program by returning the eigenvector with a negative eigenvalue. (Poly time)

Ellipsoid Algorithm

bounded by box

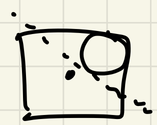
Given as input a separation oracle for $K \subseteq [-B,B]^n$, output
$$\begin{cases} \text{"yes"} & Vol(K) \geq \varepsilon^n \\ \text{"no"} & K \text{ is empty} \end{cases}$$

finite bits means there is some wiggle room for solutions, giving K volume

(Ex/ let $K = \{ \vec{x} \mid A\vec{x} \leq \vec{b} \}$ and $x \in [0,1]^n$ and A_{ij}, b_i are rational numbers of c bits.)

The plan is to check $K \cap \{ \vec{x} \mid f(x) \leq C \}$ emptiness with the ellipsoid algorithm, and run binary search on C. This is easy if f is linear, but if f is just convex we use the fact that convex functions lie above the gradient hyperplane; So, we need a gradient oracle for convex f.

Def: An ellipsoid is defined by a center $\vec{a}$ and a pos. semidefinite matrix B s.t.
$$E_{\vec{a},B} = \{ \vec{x} \mid (\vec{x}-\vec{a})^T B (\vec{x}-\vec{a}) \leq 1 \}$$



The algorithm follows these rules:

1. Query the origin; either it is in K and we are done, or we get a separating hyperplane and have shrunk the potential volume for K by a multiplicative factor.
2. Repeat a poly # of times until $vol\, K < \varepsilon^n$

More precisely the algorithm works by:

vars
E_0 = smallest ellipsoid containing $[-B,B]^n$ ← initial bounding box
Define $p_i = \text{center}(E_i)$

loop
while $(\text{vol}(E_i) \geq \varepsilon^n)$:
    if (separation oracle (p_i)):
        - return p_i
    else:
        - get separating hyperplane $\vec{w}_i, b_i$
        - update E_{i+1} = smallest ellipsoid containing $E_i \cap \{\vec{x} \mid \vec{w}_i \cdot \vec{x} \leq b_i\}$

funky space but smallest ellipsoid containing this space is nice

makes feasible region larger, but is easy to compute and gives you p_{i+1} for free

return False

Lemma 1: We can find E_{i+1} given $E_i, \vec{w}_i, b_i$

Lemma 2: $\text{Vol}(E_{i+1}) \leq \underbrace{\left(1 - \frac{1}{2n}\right)}_{\text{shrinking factor}} \text{Vol}(E_i)$

If we define the two problems for closed, convex K

separation oracle → $$\text{separate}_K(\vec{x}) = \begin{cases} \text{yes} & \vec{x} \in K \\ \vec{w} \text{ s.t. } \vec{x} \cdot \vec{w} > \max_{\vec{y} \in K} \{\vec{y} \cdot \vec{w}\} & \vec{x} \notin K \end{cases}$$

LP over convex space → $$\text{optimize}_K(\vec{c}) = \underset{\vec{x} \in K}{\text{argmax}} \{\vec{x} \cdot \vec{c}\}$$

We just saw a reduction from $\text{optimize}_K \to \text{separate}_K$.
We wish to prove a reduction from $\text{separate}_K \to \text{optimize}_K$

Theorem: Separate$_K$ $\rightarrow$ optimize$_K$

this LP searches for most violated constraint

Proof: Define an LP with vars $\vec{w}$ s.t. we

$$\text{maximize} \quad \sum_i x_i w_i = \vec{x} \cdot \vec{w}$$

$$\text{subject to} \quad \sum_i w_i y_i = \vec{y} \cdot \vec{w} \leq 1 \quad \forall \vec{y} \in K$$

constraint space

check this ☆

We see that at $\vec{x} \notin K$, $\exists \vec{w}'$ s.t. $\vec{x} \cdot \vec{w}' > \max_{\vec{y} \in K} \{\vec{y} \cdot \vec{w}'\}$

Let $\vec{w} = \dfrac{\vec{w}'}{\max_{\vec{y} \in K} \{\vec{y} \cdot \vec{w}'\}}$. This will clearly satisfy the constraint.

We seek a separation oracle for the region $\vec{w} \cdot \vec{y} \leq 1$ $\forall \vec{y} \in K$, which we can do by optimizing $\max_{\vec{y} \in K} \{\vec{w} \cdot \vec{y}\}$ and comparing this to 1. With this oracle, we can then optimize the initial LP via the ellipsoid algorithm to devise a separation oracle for K.

$\blacksquare$

Lecture 9/22 - Semidefinite Programs

Some linear algebra background:

Def: A symmetric matrix $A \in \mathbb{R}^{n\times n}$ is positive semidefinite if

$$\forall \vec{x} \in \mathbb{R}^n, \quad \vec{x}^T A \vec{x} \geq 0.$$

The following are equivalent:

(1) A symmetric matrix is PSD

(2) A has all nonnegative eigenvalues

(3) A can be written as $U^T U$ for some $U \in \mathbb{R}^{n\times n}$

$\iff A_{ij} = \langle \vec{u}_i, \vec{u}_j \rangle$ for n vectors $\vec{u}_1, \dots, \vec{u}_n \in \mathbb{R}^n$

Obs: The set of all PSD matrices in $\mathbb{R}^{n\times n}$ is convex. $\forall x, y \in S, \frac{x+y}{2} \in S$

Proof:

Let A_1, A_2 be PSD. Then, $A = \frac{A_1 + A_2}{2}$ has

$$\vec{x}^T A \vec{x} = \frac{1}{2} \vec{x}^T (A_1 + A_2) \vec{x} = \frac{1}{2}\left(\vec{x}^T A_1 \vec{x} + \vec{x}^T A_2 \vec{x}\right) \geq 0 \quad \forall \vec{x} \in \mathbb{R}^n$$

$\square$

A semi-definite program is a program of the form

maximize $\sum_{ij} c_{ij} X_{ij}$ — objective can actually be minimizing any convex fn. or maximizing any concave fn.

subject to

$\sum_{ij} A_{ijk} X_{ij} \leq b_k \quad \forall k$

X is PSD $\iff \exists \vec{v}_1, \dots, \vec{v}_n \in \mathbb{R}^n$ s.t. $X_{ij} = \langle \vec{v}_i, \vec{v}_j \rangle \ \forall i,j$

Equivalently, we can write a program to search over the vectors $\{\vec{v}_1, \dots, \vec{v}_n\}$

$\Rightarrow$ maximize $\sum_{ij} c_{ij} \langle \vec{v}_i, \vec{v}_j \rangle$

subject to $\sum_{ij} A_{ijk} \langle \vec{v}_i, \vec{v}_j \rangle \leq b_k \quad \forall k$

$\vec{v}_i \in \mathbb{R}^n \quad \forall i$

Ex/ Max-Cut

Consider the NP-Hard problem Max-Cut:

Given undirected, unweighted graph, find $S \subseteq V$ ($S \neq \emptyset$, $S \neq V$) maximizing # of edges between S & $\bar{S}$ $\left(\sum_{u \in S} \sum_{v \in \bar{S}} I((u,v) \in E) \right)$

The current best approach is to do an SDP relaxation (replace # w/ vectors) solve SDP, then round!

We write the integer program

$$\text{maximize} \quad \sum_{(i,j) \in E} \frac{1}{4} |u_i - u_j|^2$$

$$\text{subject to} \quad u_i \in \{-1, 1\} \quad \forall i$$

label for which side of cut u_i is on

To make this an integer SDP, we write the u_i labels as standard basis vectors:

(linear function of dot products) $\quad \text{maximize} \quad \sum_{(i,j) \in E} \frac{1}{4} \|\vec{u}_i - \vec{u}_j\|^2 = \sum_{(i,j) \in E} \frac{1}{4} \left(\langle \vec{u}_i, \vec{u}_i \rangle + \langle \vec{u}_j, \vec{u}_j \rangle - 2 \langle \vec{u}_i, \vec{u}_j \rangle \right)$

(linear constraints over dot products) $\quad \text{subject to} \quad \|\vec{u}_i\|^2 = 1 \ \forall i \iff \langle \vec{u}_i, \vec{u}_i \rangle \geq 1, \ \langle \vec{u}_i, \vec{u}_i \rangle \leq 1 \ \forall i$

$$\cancel{\vec{u}_i \in \{\hat{e}_1, -\hat{e}_1\} \ \forall i}$$

We can relax the basis element constraint to get an SDP, which is poly-time solvable. Now, we must round.

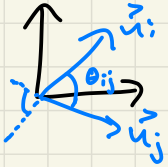
Random Hyperplane Rounding:

(1) Choose $\vec{c} \sim N(0,1)^n$ ($c_i \sim N(0,1)$ i.i.d)

(2) Set $u_i = \text{sign}(\langle \vec{c}, \vec{u}_i \rangle)$

The hyperplane tangent to $\vec{c}$ at the origin splits the space and cuts the graph. If $\vec{u}_i = \vec{u}_j$, they certainly are on the same side of the hyperplane. If $\vec{u}_i = -\vec{u}_j$, they are certainly on opposite sides. So, this has the properties we want.

Consider the space spanned by $\vec{u}_i, \vec{u}_j$.
We can show $\vec{c}$ lands randomly in this space.



So, $\mathbb{P}\{\text{round}(\vec{u}_i) \neq \text{round}(\vec{u}_j)\} = \frac{2\theta_{ij}}{2\pi} = \frac{\theta_{ij}}{\pi}$

Then, the number of edges in the cut is $\sum_{(i,j)\in E} \frac{\theta_{ij}}{\pi}$ in expectation

The LP yields a max $\sum_{(i,j)\in E} \frac{1}{4}\left(\|\vec{u}_i\|^2 + \|\vec{u}_j\|^2 - 2\langle \vec{u}_i, \vec{u}_j\rangle\right) = \sum_{(i,j)\in E} \frac{1-\cos(\theta_{ij})}{2}$

We can find numerically that $\forall \theta$, $\frac{\theta/\pi}{\frac{1-\cos\theta}{2}} \geq 0.878$

So, the rounded solution is a 0.878-approx of the optimal solution to the relaxed SDP.

Ex/ MAX 2SAT (NP Hard)

Given n literals and m clauses w/ 2 literals each,
i.e. $\text{clause}_i \in \{x_i \vee x_j,\ x_i \vee \neg x_j,\ \neg x_i \vee \neg x_j,\ \neg x_i \vee x_j\}$,
we want to set the literals to maximize the # of satisfied clauses.

We write

$$\text{maximize} \sum_{\ell} 1 - \frac{(1-y_\ell^1)(1-y_\ell^2)}{4}$$

$\left(y_\ell^j = x_i \text{ if } j^{th} \text{ literal in } \ell \text{ is } x_i,\ -x_i \text{ if } j^{th} \text{ literal in } \ell \text{ is } \neg x_i \right)$

clauses

subject to $x_i^2 = 1 \quad \forall i$
$x_i \in \{-1, 1\} \quad \forall i$

To vectorize this and relax it into an SDP, we want

filling the role of True

$$\text{maximize} \quad \sum_{\ell} 1 - \frac{\langle \vec{x}_0 - \vec{y}_\ell^{\,1}, \vec{x}_0 - \vec{y}_\ell^{\,2} \rangle}{4}$$

($\vec{y}_\ell^{\,i} = \vec{x}_i$ if ... else $-\vec{x}_i$)

$$\text{subject to} \quad \|x_i\|^2 = 1 \quad \forall i$$

($\vec{x}_0$ is any L_2-normalized fixed vector)

We get a solution to this SDP in poly-time.

Rounding:

(1) Pick a random direction $\vec{c} \sim N(0,1)^n$

(2) Set $x_i = \text{sign}(\langle \vec{c}, \vec{x}_i \rangle \cdot \langle \vec{c}, \vec{x}_0 \rangle)$

the sign of this is the "True" side of $\vec{c}$

Lecture 9/27 - Submodular Function Minimization

Submodular Functions

Def:
Let N be a a set N of n elements. A function $f: 2^N \to \mathbb{R}$ (power set of N) is submodular if

(1) $\forall A \subseteq B \subseteq N$ and $\forall j \notin B$, $f(A \cup \{j\}) - f(A) \geq f(B \cup \{j\}) - f(B)$ (Diminishing marginal returns)

or equivalently

(2) $\forall S, T \subseteq N$ $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$

Ex/ Cut Function
If $G=(V,E)$ is some graph and $N=V$, $f(S)$ is the weight of edges from S to $\bar{S}$, then f is submodular if all edges have nonnegative weight.

Ex/ Bipartite Coverage Functions
If $G=(V,E)$ bipartite, N is the set of left-hand nodes, $f(S)$ is the # of right hand nodes with an edge to something in S. Then, f is submodular.

SFM: Given submodular f, find $\underset{S \subseteq N}{\operatorname{argmin}} \{f(S)\}$

Note: Because submodular functions can be silly slow, we work in terms of value oracle access to $f(\cdot)$. So, we count polynomial runtime and counting the # of queries to this oracle.

Define a function $\hat{f}: [0,1]^N \to \mathbb{R}$ s.t. $\forall S \subseteq N$,
$\hat{f}(S) = \hat{f}(\text{vector with } x_i = 1 \ \forall i \in S, \ x_i = 0 \ \forall i \notin S)$ and $\hat{f}(S) = f(S)$ $\forall S$.

$\hat{f}$ is extension of f from discrete inclusion of elements to $[0,1]^n$

We want to show that $\hat{f}$ is convex $\iff$ f is submodular. Then, since $\hat{f}$ and f agree over 2^N, we can minimize $\hat{f}$, and we want to use this to minimize f.

Claim: Given an evaluation oracle and a gradient oracle for convex $\hat{f}$, we can minimize $\hat{f}$ over $[0,1]^n$ in poly time via the ellipsoid algorithm.

Proof: Recall that the ellipsoid algorithm works as follows:

Ellipsoid(K): given K convex and bounded ($\exists H$ s.t. $K \subseteq [-H,H]^n$) and a poly time separation oracle for K, determine in poly time whether K is empty.

To use ellipsoid as a subroutine, let $K_c = [0,1]^n \cap \{\vec{x} \mid \hat{f}(\vec{x}) \le C\}$
K_c is convex because $\hat{f}$ is convex, and it is bounded by $[0,1]^n$.
We can check if $\vec{x} \in K_c$ by checking $x_i \in [0,1]$ $\forall i$ and querying the evaluation oracle.
To find a separating hyperplane if $\vec{x} \notin K_c$, we can return

$$\text{hyperplane} = \begin{cases} \text{the hyperplane } \{\vec{y} \mid y_i \underset{\text{or} >}{<} x_i\} & \text{if } x_i \notin [0,1] \text{ for some } i \\ \{\vec{y} \mid \vec{\nabla}\hat{f}(\vec{x})(\vec{y}-\vec{x}) \le C - \hat{f}(\vec{x})\} & \end{cases}$$

$\square$

Def: For a function $f: \underset{\swarrow \text{power set}}{\{0,1\}^n} \to \mathbb{R}$, the Lovász extension $\hat{f}: [0,1]^n \to \mathbb{R}$ is

$$\forall \vec{x} \in [0,1]^n \quad \hat{f}(\vec{x}) = \mathop{\mathbb{E}}_{\lambda \sim U([0,1])} \{f(\{i \mid x_i \ge \lambda\})\}$$

Sample random threshold, include all coordinates above the threshold.

We observe that there are only $n+1$ sets to query on.
To see this, suppose WOLOG that $\vec{x}$ is s.t. $x_1 \ge \dots \ge x_n$. Then, the possible sets are $\{\emptyset\}, \{x_1\}, \{x_1, x_2\}, \dots, \{x_1, \dots, x_n\}$ with thresholds $1 \ge \lambda > x_1$, $x_1 \ge \lambda > x_2$, $x_2 \ge \lambda > x_3, \dots,$ $x_n \ge \lambda \ge 0$ that occur with $\mathbb{P}$'s

$\mathbb{P} = 1 - x_1$, $\mathbb{P} = x_1 - x_2$, $\mathbb{P} = x_2 - x_3, \dots,$ $\mathbb{P} = x_n - 0$

So, for such monotonically decreasing $\vec{x}$,

$$\hat{f}(\vec{x}) = \sum_{i=0}^{n} (x_i - x_{i+1}) f(\{1,\dots,i\}) \qquad (x_0 = 1,\ x_{n+1} = 0)$$

Then, $$\frac{\partial \hat{f}(\vec{x})}{\partial x_i} = f(\{1,\dots,i\}) - f(\{1,\dots,i-1\}).$$

We can create a gradient oracle in $n \log n$ time + n evaluations of f.
← needed to sort

Theorem: $\hat{f}$ is convex $\iff$ f is submodular

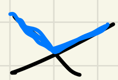
Proof: For simplicity, suppose WOLOG that
1) $x_1 \geq \dots \geq x_n$ (we can relabel)
2) $f(\emptyset) = 0$ (shifting f doesn't change submodularity)

(submodular $\Rightarrow$ convex)

Define $$P_f = \left\{ \vec{w} \;\middle|\; \forall S \subseteq \{1,\dots,n\},\ \sum_{i \in S} w_i \leq f(S) \text{ and } \sum_{i=1}^{n} w_i = f(\{1,\dots,n\}) \right\}$$
← power set

Define $g(\vec{z}) = \max_{\vec{w} \in P_f} \{ \vec{z} \cdot \vec{w} \}$. This forms a primal LP maximized by g.

(1) g is convex because the supporting line of a linear segment will lie below (or along) the max of the next part.



(2) f submodular $\Rightarrow g = \hat{f}$. To see this, we write the dual

$$\text{minimize} \sum_{S \subseteq \{1,\dots,n\}} y_S f(\{1,\dots,s\})$$
$$\text{subject to} \sum_{i \in S} y_S = z_i \quad \forall i$$
$$y_S \geq 0 \quad \forall S \subseteq \{1,\dots,n\}$$

We propose optimal $w_i^* = f(\{1,\dots,i\}) - f(\{1,\dots,i-1\}) = \nabla \hat{f}$

$$y_S^* = \begin{cases} z_i - z_{i+1} & S = \{1,\dots,i\} \\ z_n & S = \{1,\dots,n\} \\ 0 & \text{else} \end{cases}$$

We want to show

(1) $\vec{z} \cdot \vec{w}^* = \hat{f}(\vec{z})$

(2) $\sum_{S \subseteq \{1,\dots,n\}} y_S^* f(S) = \hat{f}(\vec{z})$

(3) $\vec{y}^*$ is feasible in the dual

(4) $\vec{w}^*$ is feasible in the primal

true $\forall f$

since this will imply that these are optimal solutions, and therefore that $\hat{f} = g$. We prove (4) by induction on $|S|$.

For a given S let i be the largest index in S.

Since f is submodular,

$$f(S) + f(\{1,\dots,i-1\}) \geq f(S \cup \{1,\dots,i-1\}) + f(S \cap \{1,\dots,i-1\})$$
$$= f(\{1,\dots,i\}) + f(S \setminus \{i\})$$

$$\Rightarrow f(S) \geq f(\{1,\dots,i\}) - f(\{1,\dots,i-1\}) + f(S \setminus \{i\})$$
$$= w_i^* + f(S \setminus \{i\}) \underset{IH}{\geq} \sum_{i \in S} w_i^*$$

So, w_i^* is feasible in the primal.

Thus, we can optimize $\hat{f}$ in poly time.

Lecture 9/29 - Concentration Bounds

Vibes: what can we say about a random variable and how close it usually/always is to its expectation?

Notation: For the below notes, S is a subset of the power set $\{0,1\}^N$

Recall Markov's Inequality:

Let X be a nonnegative random variable. Then,

$$\mathbb{P}\{X > c\,\mathbb{E}\{X\}\} \leq \frac{1}{c} \quad \forall c > 0$$

and Chebyshev's Inequality:

Let X be a RV with mean μ and variance σ^2. Then,

$$\mathbb{P}\{|X-\mu| \geq c\sigma\} \leq \frac{1}{c^2} \quad \forall c > 0$$

Chernoff Bounds

☆ We ask what if we draw n random variables that are independent and bounded. CLT means they approach Gaussian!

Formally, what if we have random variables $X_1, \ldots, X_n$ that are independent and s.t. $X_i \in [0,1]$ $\forall i$. What can we say about $X = \sum_{i=1}^{n} X_i$?

Theorem: Let $X_1, \ldots, X_n$ be independent with $X_i \in \{0,1\}$ $\forall i$. Then,

$$\mathbb{P}\left\{\sum_{i=1}^{n} X_i \geq (1+\varepsilon)\,\mathbb{E}\left\{\sum_{i=1}^{n} X_i\right\}\right\} \leq e^{\frac{-\varepsilon^2 \mathbb{E}\{\sum_{i=1}^{n} X_i\}}{3+3\varepsilon}}$$

← grows exponentially in expectation of the sum

Proof: Let $X = \sum_{i=1}^{n} X_i$. Let $p_i := \mathbb{E}\{X_i\}$ $\forall i$

Pick t to set later, and look at the random variable e^{tX}.

Observe that

$$\mathbb{E}\{e^{tX}\} = \mathbb{E}\left\{\prod_{i=1}^{n} e^{tX_i}\right\} \overset{X_i\text{'s independent}}{=} \prod_{i=1}^{n} \mathbb{E}\{e^{tX_i}\} = \prod_{i=1}^{n}\left((1-p_i) + p_i e^t\right)$$

$$= \prod_{i=1}^{n}\left(1 + p_i(e^t - 1)\right) \underset{1-x \leq e^{-x}}{\leq} \prod_{i=1}^{n} e^{p_i(e^t-1)} \underset{\sum_{i=1}^{n} p_i = \mathbb{E}\{X\}}{=} e^{(e^t-1)\mathbb{E}\{X\}}$$

We can see that, since e^{tz} is monotone,

$$\mathbb{P}\{X > (1+\varepsilon)\mathbb{E}\{X\}\} = \mathbb{P}\{e^{tX} > e^{t(1+\varepsilon)\mathbb{E}\{X\}}\}$$

By Markov's Inequality, this is bounded by

$$\mathbb{P}\{X > (1+\varepsilon)\mathbb{E}\{X\}\} \le \frac{e^{(e^t-1)\mathbb{E}\{X\}}}{e^{t(1+\varepsilon)\mathbb{E}\{X\}}}$$

Letting $t = \ln(1+\varepsilon)$ and noting $(1+\varepsilon)\ln(1+\varepsilon) > \varepsilon + \frac{\varepsilon^2}{3}$ for $\varepsilon \in [0,1]$,

$$\mathbb{P}\{X > (1+\varepsilon)\mathbb{E}\{X\}\} \le e^{\mathbb{E}\{X\}(\varepsilon - (1+\varepsilon)\ln(1+\varepsilon))} \le e^{-\frac{\varepsilon^2 \mathbb{E}\{X\}}{3}} \le e^{-\frac{\varepsilon^2 \mathbb{E}\{X\}}{3+3\varepsilon}}$$

so that it holds for $\varepsilon > 1$

READ NOTES HERE for Chernoff applications

Examples that look like sum of random variables we can use Chernoff on, but <u>aren't</u>!

Ex1/ Fixed unweighted graph G. Put v in a set S independently with probability p_v to get a random cut. What is the value of $\text{cut}_G(S)$.

Ex2/ Let F be a subset of the power set $\{0,1\}^N$. Put v in S independently with probability p_v. We can use Chernoff bounds on the size $|S|$, but not on functions like

a) $\max_{T \in F} \{|S \cap T|\}$ or b) $f(S) = \begin{cases} |S| & 0 \le |S| < \sqrt{n} \\ \sqrt{n} & \sqrt{n} \le |S| \le \frac{n-\sqrt{n}}{2} \\ \sqrt{n} + |S| & \frac{n-\sqrt{n}}{2} < |S| < \frac{n+\sqrt{n}}{2} \\ 2\sqrt{n} & |S| \ge \frac{n+\sqrt{n}}{2} \end{cases}$

<u>Defn</u>: A function f is c-Lipschitz if $\forall S \subseteq N$ and $j \in N$,

$$\underbrace{|f(S \cup \{j\}) - f(S)| \le c}_{\text{bounded differences}}$$

Theorem: McDiarmid's Inequality

Let $X_1, \dots, X_n$ be independent random variables, and let $f(\dots)$ satisfy bounded differences for $c_1, \dots, c_n$

$$\left(\text{i.e. } \forall i, \vec{X}_{-i}, x_i, x_i' \quad \underbrace{|f(\vec{X}_{-i}, x_i) - f(\vec{X}_{-i}, x_i')| \le c_i}_{\text{holding everything except } X_i \text{ constant makes } f \ c_i\text{-Lipschitz}}\right)$$

Then,

$$\mathbb{P}\{|f(\vec{X}) - \mathbb{E}\{f(\vec{X})\}| > \varepsilon\} \le 2e^{\frac{-2\varepsilon^2}{\sum_i c_i^2}}$$

(Note, when f is 1-Lipschitz, $\mathbb{P}\{\dots\} \le 2e^{-\frac{2\varepsilon^2}{n}}$. So, when $\varepsilon > \sqrt{n}$ this is cool.)

Better Theorem: Schoetman

$f(S \cup T) \le f(S) + f(T)$

Let f be <u>subadditive</u> and 1-Lipschitz, and let $X_1, \dots, X_n$ be independent. Let a be the median of $f(\vec{X})$. Then,

$$\mathbb{P}\{f(\vec{X}) \ge 3a + k\} \le 2^{2-k} \quad \forall k > 0$$

Note that Example 1 above is submodular, but non-monotone and Example 2(a) above is XOS/fractionally subadditive

Def: A function f is XOS if there exist additive functions $f_1, \dots, f_k$ s.t. $f(S) = \max_i \{f_i(S)\}$

Def: A function f is (a,b)-self-bounding if there exist $f_1, \dots, f_n$ s.t.

$$0 \le f(S) - f_i(S \setminus \{i\}) \le 1 \quad \forall i \in 1, \dots, n$$

$$\sum_{i=1}^{n} f(S) - f_i(S \setminus \{i\}) \le a f(S) + b \quad \forall S$$

Theorem: (a,b)-self-bounded functions are Chernoff bounded.

Corollary: Since XOS functions are $(1,0)$-self-bounded and non-monotone submodular functions are $(2,0)$-self-bounded, XOS & normed SM are Chernoff bounded.

Lecture 10/4 Streaming I

Streaming algorithms process large data in a small space (low memory usage).

The input stream is a sequence of inputs $a_1, \dots, a_n$ that is processed in sequence order.

Ex/ Approximate Counting

Maintain a counter n initialized to 0, supporting
- inc(): $n \leftarrow n+1$ (no more than N inc())
- query(): return an approximation $\tilde{n} = (1 \pm \varepsilon) n$ with high probability.

We can solve this trivially with $\log(N)$ bits by maintaining exact counter. This can be quite big.

Question:
Can we represent numbers $n \in \{1, \dots, N\}$ using $\ll \log N$ bits s.t. we can recover $\tilde{n} \in [\frac{n}{2}, 2n]$ from the encoding?

We can approximate n by only storing intervals, such as the nearest power of 2 (to get a 2-approx).
For $n \in [2^x, 2^{x+1})$, we can store x in $O(\log \log N)$ bits.

We can handle increments by incrementing x with probability 2^{-x}, such that we, in expectation, increment x when we should. The algorithm looks like

init(): $x \leftarrow 0$ after first inc() query(): return 2^x

inc(): $x \leftarrow \begin{cases} x+1 & \text{w.p. } 2^{-x} \\ x & \text{w.p. } 1-2^{-x} \end{cases}$

Analysis: Let X_n be the R.V. X after n calls to inc()

We WTS that $\mathbb{E}\{2^{X_n}\} = n$ and $\mathrm{Var}\{2^{X_n}\} \leq O(n^2)$

Proof.

$$\mathbb{E}\{2^{X_n}\} = \sum_x \mathbb{P}\{X_n = x\} 2^x = \sum_x \left(\mathbb{P}\{X_{n-1}=x\}\cdot(1-2^{-x}) + \mathbb{P}\{X_{n-1}=x-1\}\cdot 2^{-(x-1)}\right)2^x$$

$$= \sum_x \mathbb{P}\{X_{n-1}=x\}(2^x - 1) + \sum_x \mathbb{P}\{X_{n-1}=x-1\}\cdot 2 = \mathbb{E}\{2^{X_{n-1}} - 1\} + 2$$

$$= \mathbb{E}\{2^{X_{n-1}}\} + 1 \implies \mathbb{E}\{2^{X_n}\} = n.$$

Similar logic works for the variance.

□

We can apply Chebyshev's Inequality $\mathbb{P}\{|Y - \mathbb{E}\{Y\}| > T\} < \frac{\mathrm{Var}\{Y\}}{T^2}$ to get

$$\mathbb{P}\{|2^{X_n} - n| > T\} \leq O\left(\left(\tfrac{n}{T}\right)^2\right)$$

Means

We can reduce the variance by averaging s independent copies. Let $X^{(i)}$ denote X in the i^{th} copy. Then, letting $X^* = \frac{1}{s}\sum_i X^{(i)}$ be the average, $\mathbb{E}\{2^{X^*}\} = n$ and $\mathrm{Var}\{2^{X^*}\} = \frac{1}{s^2}\cdot s\,O(n^2) = O\left(\frac{n^2}{s}\right)$

Chebyshev now gives

$$\mathbb{P}\{|2^{X^*} - n| > T\} \leq O\left(\frac{1}{s}\left(\frac{n}{T}\right)^2\right)$$

If we set $T = \varepsilon n$, $s = \frac{1}{\delta\varepsilon^2}$, we get $\mathbb{P}\{|2^{X^*} - n| > \varepsilon n\} < \delta$

Total space used is $\Theta\left(\frac{1}{\varepsilon^2\delta}\log\log n\right)$

Median of means

Maintain $s_1 \cdot s_2$ independent copies. On query, divide into s_1 groups of size s_2. Let $X^{(ij)}$ be the j^{th} X of group i. For each group i, compute $\tilde{n}_i = \frac{1}{s_2}\sum_j 2^{(ij)}$. Let $\tilde{n}$ be the median of $\tilde{n}_1, \dots, \tilde{n}_{s_1}$.

If we set s_2 to $\Theta\left(\frac{1}{\varepsilon^2}\right)$, $\mathbb{P}\{|\tilde{n}_i - n| > \varepsilon n\} < \frac{1}{4}$

We can find that

$$\tilde{n} > (1+\varepsilon)n \iff \geq \frac{s_2}{2} \text{ groups have } \tilde{n}_i > (1+\varepsilon)n$$

$$\tilde{n} < (1-\varepsilon)n \iff \geq \frac{s_2}{2} \text{ groups have } \tilde{n}_i < (1-\varepsilon)n$$

Let $Y_i = \begin{cases} 1 & \text{if } \tilde{n} > (1+\varepsilon)n \\ 0 & \text{else} \end{cases}$

We know that

1. All the Y_i are independent
2. $\tilde{n} > (1+\varepsilon)n \iff \sum Y_i \geq \frac{s_2}{2}$
3. $\mathbb{E}\{Y_i\} < \frac{1}{4}$

With Chernoff, $\mathbb{P}\{\tilde{n} > (1+\varepsilon)n\} = \mathbb{P}\{\sum_{i=1}^{s_2} Y_i \geq \mathbb{E}\{\sum Y_i\} \cdot 2\} \leq e^{-\frac{s_1}{12}}$

Similarly, $\mathbb{P}\{\tilde{n} < (1-\varepsilon)n\} \leq e^{-\frac{s_2}{12}}$

If we set s_1 to $\Theta(\log(\frac{1}{\delta}))$, union bound yields

$$\mathbb{P}\{|\tilde{n} - n| \leq \varepsilon n\} > 1 - \delta$$

Total space used is $O\left(\frac{1}{\varepsilon^2} \log\left(\frac{1}{\delta}\right) \log \log N\right)$

Morris Counter

If we instead set the base to be $(1+\alpha)$ instead of 2,
we get $\mathbb{P}\{|\tilde{n} - n| \leq \varepsilon n\} > 1 - \delta$ with space $O\left(\log\left(\frac{1}{\varepsilon}\right) + \log\log N + \log\log\left(\frac{1}{\delta}\right)\right)$

Lecture 10/6 - Streaming II

Ex/ Distinct Elements

Input: a stream $a_1, \dots, a_n$ $(a_i \in \{1, \dots, U\})$,

Output: estimate $\tilde{F}$ of $\underbrace{\text{\# of distinct elements}}_{F}$
s.t. $\tilde{F} = (1 \pm \varepsilon) F$ w.p. $\geq 1 - \delta$.

Naive Solution

Store all distinct elements! $O(n \log U)$ space

Subset Sampling

Not accurate :(

(Recall that if $X_1, \dots, X_F$ are independent RV's with $X_i \sim U[0,1]$ uniform
and $X^{(k)}$ is the k^{th} smallest one, then $\mathbb{E}\{X^{(k)}\} = \frac{k}{F+1}$)

We can use this in reverse: find $X^{(k)}$ for some k to estimate F.

KMV (k-minimum value)

Algorithm

Ideally, assume access to a random hash function
$h: \{1, \dots, U\} \to [0,1]$. Have a parameter $k \geq 1$ to set later.

- initialize a set S to $\emptyset$ to store the k smallest hash values.
- for i in $\{1, \dots, n\}$:
  - $S \leftarrow S \cup \{h(a_i)\}$
  - if $|S| > k$: remove $\max\{S\}$ from S
- if $|S| = k$: return $\tilde{F} = \frac{k}{\max(S)} - 1$ ← -1 turns out to be unimportant

  else: return $\tilde{F} = |S|$

Analysis

We want two things

① upper bound on $\mathbb{P}\{\tilde{F} > (1+\varepsilon)F\}$
② upper bound on $\mathbb{P}\{\tilde{F} < (1-\varepsilon)F\}$

these are v. similar, so we focus on ①

We can find

$$\mathbb{P}\{\tilde{F} > (1+\varepsilon)F\} = \mathbb{P}\left\{\frac{k}{\max\{S\}} > (1+\varepsilon)F\right\} = \mathbb{P}\left\{\max\{S\} < \frac{k}{(1+\varepsilon)F}\right\}$$

where $\max\{S\}$ is the k^{th} smallest hash value.

Let $\underbrace{v_1, \dots, v_F}_{\text{independent}}$ be the hash values of the elements.

$$\forall v_i, \quad \mathbb{P}\left\{v_i < \frac{k}{(1+\varepsilon)F}\right\} = \frac{k}{(1+\varepsilon)F}$$

Let X be a RV denoting the # of v_i s.t. $v_i < \frac{k}{(1+\varepsilon)F}$

$$\Rightarrow \mathbb{E}\{X\} = \sum_{i=1}^{F} \frac{k}{(1+\varepsilon)F} = \frac{k}{1+\varepsilon}$$

$$\Rightarrow \mathrm{Var}\{X\} = \sum_{i=1}^{F} \mathrm{Var}\left\{v_i < \frac{k}{(1+\varepsilon)F}\right\} = F\left(\frac{k}{(1+\varepsilon)F} - \left(\frac{k}{(1+\varepsilon)F}\right)^2\right) < k$$

By Chebyshev,

$$\mathbb{P}\{X \geq k\} \leq \frac{\mathrm{Var}\{X\}}{\left(k - \frac{k}{1+\varepsilon}\right)^2} < \frac{\cancel{k}(1+\varepsilon)^2}{k^{\cancel{2}}\varepsilon^2} = O\left(\frac{1}{\varepsilon^2 k}\right)$$

If we set $k = \frac{c}{\varepsilon^2}$ $\quad \mathbb{P}\{\tilde{F} > (1+\varepsilon)F\} < O\left(\frac{1}{c}\right)$

We can apply similar logic to find that $\mathbb{P}\{\tilde{F} < (1-\varepsilon)F) < O\left(\frac{1}{c}\right)$

By Union Bound, $\quad \mathbb{P}\{\tilde{F} \in (1 \pm \varepsilon)F\} > 1 - O\left(\frac{2}{c}\right)$

using space $O\left(\frac{1}{\varepsilon^2}\right)$ "real numbers".

We can do better with the **median trick**: maintain T independent copies and output the median of the predictions. We saw last time that this yields

$$\mathbb{P}\{\text{median} \in (1 \pm \varepsilon)F\} \geq 1 - e^{-\Theta(T)}.$$ Setting $T = O(\log(\frac{1}{\delta}))$, $\mathbb{P}\{\dots\} \geq 1-\delta$.

space $= O\left(\frac{\log(\frac{1}{\delta})}{\varepsilon^2}\right)$ real #'s

Note that this algorithm assumes

① storing real numbers in $[0,1]$
② random hash function

Removing the Assumptions

① Discretize $[0,1]$ to $\{\frac{1}{M}, \frac{2}{M}, \dots, \frac{M-1}{M}, 1\}$. We get a "rounding error" $\leq O(\frac{1}{M})$.
If we set $M = U$, things work out the same.

② **Defn** Let $\mathcal{H}$ be a family of hash functions
$\{1, \dots, U\} \to \{1, \dots, M\}$. $\mathcal{H}$ is c-wise independent if
$\forall x_1, \dots, x_c \in \{1, \dots, U\}$ distinct, $\forall a_1, \dots, a_c \in \{1, \dots, M\}$,

$$\mathbb{P}_{h \in \mathcal{H}}\{\forall i \in \{1, \dots, c\}, h(x_i) = a_i\} = \frac{1}{M^c}$$

Recall that there exists pairwise independent $\mathcal{H}$ of size $\mathrm{poly}(U, M)$.
$\Rightarrow$ it takes $O(\log U + \log M)$ to encode one $h \in \mathcal{H}$.

Recall also that variance is linear for pairwise independent RVs.
For KMV, the only place that we use independence of the hash values v_i is when calculating $\mathrm{Var}\{X\}$.

So, the proof of the analysis is complete!
Total space amounts to

$$O\left(\log\left(\frac{1}{\delta}\right)\left(\log U + \frac{1}{\varepsilon^2}\log U\right)\right) = O\left(\frac{1}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)\log(U)\right) \text{ bits}$$

There is a better result: $O\left(\frac{1}{\varepsilon^2}\log\left(\frac{1}{\delta}\right) + \log(U)\right)$ (Błasiok 2018)

Ex/ **Frequency Moment**

Input: a stream $a_1, \dots, a_n$ $(a_i \in \{1, \dots, U\})$,

Denote by f_x the # of x in the stream and $F_p = \sum_{x \in \{1, \dots, U\}} (f_x)^p$

Output: We want $\tilde{F}$ s.t. $\mathbb{P}\{\tilde{F} \in (1 \pm \varepsilon)F\} \geq 1 - \delta$

Note that $p=0$ is # distinct, $p=1$ is counter.
For $p=2$, we use AMS.

AMS

Algorithm:

Assume access to a random hash $\sigma : \{1, \dots, U\} \to \{-1, 1\}$

- initialize $x \leftarrow 0$
- for i in $\{1, \dots, n\}$:
  $x \leftarrow x + \sigma(a_i)$
- return x^2

Correctness/Analysis

We have $X = \sum_{y \in \{1, \dots, U\}} f_y \cdot \sigma(y) \Rightarrow X^2 = \sum_{y_1, y_2} f_{y_1} f_{y_2} \sigma(y_1) \sigma(y_2)$

$$= \sum_y f_y^2 \cancel{\sigma(y)^2} + 2 \sum_{y_1 y_2} f_{y_1} f_{y_2} \underbrace{\sigma(y_1)\sigma(y_2)}_{\text{has } \mathbb{E}=0}$$

$$\Rightarrow \mathbb{E}\{X^2\} = \sum_y f_y^2 = F_2 \quad ✓$$

Similarly, we can find (if σ 4-wise independent) ← fill this in w/ lecture notes

$$\mathrm{Var}\{X^2\} = \mathbb{E}\{X^4\} - \mathbb{E}\{X^2\}^2 \leq O(F_2^2)$$

We can maintain $s_1 \cdot s_2$ copies of AMS, divided into $s_1 = O(\log \frac{1}{\delta})$ groups of size $s_2 = O(\frac{1}{\varepsilon^2})$. The median of the group means satisfies

$$\mathbb{P}\{\text{median} \in (1 \pm \varepsilon) F_2\} > 1 - \delta$$

with space $O\left(\frac{1}{\varepsilon^2} \log\left(\frac{1}{\delta}\right) \log U\right)$

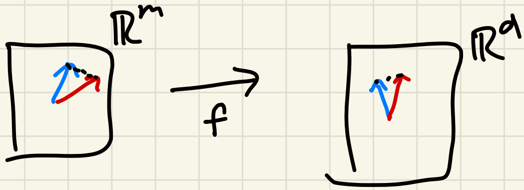
(Note: for $p > 2$, space lower bound $\Omega(n^{1-2/p})$)

Lecture 10/11 - Johnson-Lindenstrauss

We focus on dimensionality reduction.

Given vectors $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^m$, and $\varepsilon > 0$, find a mapping $f: \mathbb{R}^m \to \mathbb{R}^d$ $(d \ll m)$ s.t. $\forall i,j \in \{1, \ldots, n\}$,

$$\|f(x_i) - f(x_j)\|^2 \in (1 \pm \varepsilon)\|x_i - x_j\|^2$$



Theorem: (Johnson-Lindenstrauss)

For any $x_1, \ldots, x_n \in \mathbb{R}^m$ and any $\varepsilon > 0$, there exists $f: \mathbb{R}^m \to \mathbb{R}^d$ for $d = \Omega(\frac{1}{\varepsilon^2} \log n)$ s.t. $\|f(x_i) - f(x_j)\|^2 \in (1 \pm \varepsilon)\|x_i - x_j\|^2$ $\forall i,j \in [n]$.
Moreover, f is linear. $f(x) = \Pi X$ ← $d \times m$ matrix

Proof:

The plan is as follows:

① Find a distribution $\mathcal{D}$ over matrices in $\mathbb{R}^{d \times m}$ s.t.

$$\boxed{\forall \vec{x} \in \mathbb{R}^m, \quad \mathbb{P}_{\Pi \sim \mathcal{D}}\left[\|\Pi \vec{x}\|^2 \in (1 \pm \varepsilon)\|\vec{x}\|^2\right] > 1 - \delta} \quad (\varepsilon, \delta) \text{ J-L Property}$$

for $d = O(\frac{1}{\varepsilon^2} \log n)$, $\delta = \frac{1}{\text{poly}(n)}$.

② Union bound!

② Starting from ②, assume we have done ①. Then, sampling $\Pi \sim \mathcal{D}$, we get $f: \mathbb{R}^m \to \mathbb{R}^d$ s.t. $f(\vec{x}_i - \vec{x}_j) = f(\vec{x}_i - f\vec{x}_j)$. By (ε, δ) J-L, $\forall i,j \in [n]$,

$$\mathbb{P}_{\Pi \sim \mathcal{D}}\left[\|f(\vec{x}_i) - f(\vec{x}_j)\|^2 \in (1 \pm \varepsilon)\|\vec{x}_i - \vec{x}_j\|^2\right] = \mathbb{P}_{\Pi \sim \mathcal{D}}\left[\|\Pi(\vec{x}_i - \vec{x}_j)\|^2 \in (1 \pm \varepsilon)\|\vec{x}_i - \vec{x}_j\|^2\right] > 1 - \delta$$

For $\delta = \frac{1}{n^3}$, we can union bound over all pairs to see that what we want happens with probability $1 - \frac{1}{n}$.

$\square$

① There are two constructions of this distribution $\mathcal{D}$:

(a) $\Pi_{ij} \sim_u \frac{1}{\sqrt{d}} \cdot \{-1, 1\}$ (b) $\Pi_{ij} \sim_u \frac{1}{\sqrt{d}} \mathcal{N}(0,1)$ ← we prove with this

Using scheme (b), fix $\vec{x} \in \mathbb{R}^m$. Sample Π as above, and let $\vec{y} = \Pi \vec{x} \in \mathbb{R}^d$.
Then, $\|\vec{y}\|^2 = \sum_{i=1}^{d} y_i^2$, and $y_i = \sum_{j=1}^{m} \Pi_{ij} x_j \quad \forall i$.

$$\Rightarrow \mathbb{E}_{\Pi}[y_i^2] = \mathbb{E}_{\Pi}\left[\left(\sum_j \Pi_{ij} x_j\right)^2\right] = \mathbb{E}_{\Pi}\left[\sum_{j_1, j_2 = 1}^{m} \Pi_{ij_1} \Pi_{ij_2} x_{j_1} x_{j_2}\right]$$

$$= \mathbb{E}_{\Pi}\left[\sum_{j=1}^{m} \Pi_{ij}^2 x_j^2\right] + \mathbb{E}_{\Pi}\left[2 \sum_{j_1 < j_2} \underbrace{\Pi_{ij_1} \Pi_{ij_2}}_{\text{0 in expectation because independent}} x_{j_1} x_{j_2}\right]$$

$$= \frac{1}{d} \sum_{j=1}^{m} x_j^2 = \frac{\|\vec{x}\|^2}{d}$$

$$\Rightarrow \mathbb{E}\left[\|\vec{y}\|^2\right] = d \cdot \frac{\|\vec{x}\|^2}{d} = \|\vec{x}\|^2.$$ So, $\mathcal{D}$ behaves well in expectation. We want to show $\sum y_i^2$ <u>concentrates.</u>

We can say $$\mathbb{P}\left[\sum y_i^2 > (1+\varepsilon)\|\vec{x}\|^2\right] = \mathbb{P}\left[e^{t \sum y_i^2} > e^{t(1+\varepsilon)\|\vec{x}\|^2}\right]$$

Set $t = \frac{\varepsilon m}{8 \|\vec{x}\|^2}$

$$\underset{\text{markov}}{\leq} \frac{\mathbb{E}\left[e^{t \sum y_i^2}\right]}{e^{t(1+\varepsilon)\|\vec{x}\|^2}} = \frac{1}{e^{t(1+\varepsilon)\|\vec{x}\|^2}} \cdot \left(\frac{1}{1 - 2t \|\vec{x}\|^2 / d}\right)^{d/2}$$

Set $d = \frac{C}{\varepsilon^2} \log\left(\frac{1}{\delta}\right)$

$$\leq e^{-6\varepsilon^2 d}$$

Then $$\mathbb{P}\left[\sum y_i^2 > (1+\varepsilon)\|\vec{x}\|^2\right] \leq \delta.$$

We can perform similar bounds on the lower tail. This yields
$$\mathbb{P}_{\Pi \sim \mathcal{D}}\left[\|\Pi \vec{x}\|^2 \in (1 \pm \varepsilon)\|\vec{x}\|^2\right] > 1 - \delta$$ as desired.

$\square$

This reduces to dimension $d \approx \frac{1}{\varepsilon^2} \log(\text{poly } n)$, but takes $O(mn)$ time to transform each vector $\vec{x} \in \mathbb{R}^m$.

We can do better :)

Two strategies to speed up $\Pi\vec{x}$:

① use a sparse matrix Π (sparse JL transform)
- better for sparse $\vec{x}$

Consider random matrix Π. Fix parameter s.
- sample exactly s entries randomly in every column of Π to be nonzero
- fill all selected nonzero entries with random $\pm\frac{1}{\sqrt{s}}$

Theorem (KN, 2014)
$\exists c_1, c_2 > 0$ s.t., if we set $d = c_1 \cdot \frac{1}{\varepsilon^2}\log\left(\frac{1}{\delta}\right)$, $s = c_2\varepsilon d = \frac{c_1c_2}{\varepsilon}\log\left(\frac{1}{\delta}\right)$
then $\forall \vec{x} \in \mathbb{R}^m$, $\mathbb{P}_{\Pi}\left[\|\Pi\vec{x}\|^2 \in (1\pm\varepsilon)\|\vec{x}\|^2\right] \geq 1-\delta$

② use a structured matrix Π that allows for fast matrix-vector multiplication. (fast JL transform)
- better for average $\vec{x}$

Let Π be a product of 3 matrices, each with fast multiplication. In particular,

$$\Pi = \frac{1}{\sqrt{d}}\underbrace{S}_{d\times m} \cdot \underbrace{H \cdot D}_{m\times m}$$

(assume m is a power of d).

- S is a random variable, where $S\vec{x}$ picks d random coordinates of $\vec{x}$ to form a vector in $\mathbb{R}^d$.

Lemma: If $\|\vec{x}\|_\infty$ is small, then $\mathbb{P}\left[\|S\vec{x}\|^2 \in (1\pm\varepsilon)\|x\|^2\right]$ is large.

So, we want $H\cdot D$ to preprocess $\vec{x}$ to maintain the norm, but have small $\|H\cdot D\vec{x}\|_\infty$.

- H is a deterministic Hadamard matrix $H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}$, $H_0 = [1]$

$\forall \vec{x} \in \mathbb{R}^m$, $H\vec{x}$ can be computed in $O(m\log m)$

- D is a randomly diagonal matrix that randomly negates coordinates. $D = \begin{bmatrix} \pm 1 & & 0 \\ & \pm 1 & \\ 0 & & \ddots \ \pm 1 \end{bmatrix}$

  $D\vec{x}$ can be computed in $O(n)$ time.

We know that both $\frac{1}{n}H$ and D are unitary, preserving the norm. There is a nontrivial lemma

sort of a rotation

<u>Lemma:</u> $\forall \vec{x} \in \mathbb{R}^m$, $\mathbb{P}\left[\|\frac{1}{n}HD\vec{x}\|_\infty \text{ is "small"} \right]$ is "large".

This yields that $\Pi = SHD$ has the same properties, but can be multiplied in $O(n \log n)$ time.

Insert spinner notes here

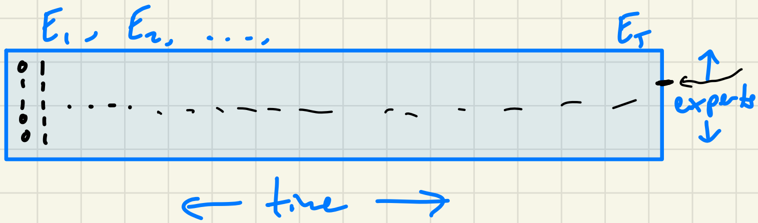
Lecture 10/25 - Learning from Experts

no distributional assumptions, can be adversarial

Consider a sequence of events $E_1, \ldots, E_T \in \{0,1\}$ where each event E_t's outcome is revealed at time t.

There are also n experts, each one predicting E_t before time t.

The goal is to predict events before they happen, minimizing # of mistakes



suppose there is some good expert that makes $m^* \ll \frac{T}{2}$ mistakes

We will show that, without knowing which one is the best expert, we can also make about m^* mistakes as well.

Warmup: If $m^*=0$, we follow the majority advice among all experts that haven't made a mistake yet at step t. There are 2 cases

1. the majority is correct
2. the majority is incorrect, and so we reduce the # of experts we follow by a factor ≥ 2

(2) can only happen $\log n$ times, and so we make $\leq \log n$ mistakes.

Weighted Majority

Init: Fix parameter $\eta \in (0, \frac{1}{2}]$, give weight $w_i = 1$ to expert i

For $t \in [T]$:

- follow the weighted majority of all experts
- for all incorrect experts, $w_i \leftarrow w_i(1-\eta)$

Theorem: the # of mistakes M is at most $2\cdot(1+\varepsilon)m^* + \frac{2\ln n}{\varepsilon}$

Proof: Denote by $w_i^{(t)}$ the weight of expert i at time t.

Let $W^{(t)} = \sum_{i=1}^{n} w_i^{(t)}$ be the total weight.

Every time we make a mistake,

$\geq \frac{W^{(t)}}{2}$ because weighted majority

$$W^{(t+1)} = \sum_{i=1}^{n} w_i^{(t+1)} = \sum_{i \text{ correct}} w_i^{(t)} + (1-\varepsilon)\sum_{i \text{ incorrect}} w_i^{(t)} = W^{(t)} - \varepsilon \sum_{i \text{ incorrect}} w_i^{(t)}$$

$$\leq W^{(t)} - \varepsilon \frac{W^{(t)}}{2} = \left(1 - \frac{\varepsilon}{2}\right) W^{(t)}$$

The best expert i_* has $w_{i_*}^{(T)} = (1-\varepsilon)^{m^*}$

The final total weight is

$$W^{(T)} \leq \left(1 - \frac{\varepsilon}{2}\right)^M W^{(0)} = n\left(1 - \frac{\varepsilon}{2}\right)^M \geq (1-\varepsilon)^{m^*}$$

$$\Rightarrow (1-\varepsilon)^{m^*} \leq n\left(1-\frac{\varepsilon}{2}\right)^M \Rightarrow m^* \ln\left(\frac{1}{1-\varepsilon}\right) \geq \ln n + M \ln\left(\frac{1}{1-\frac{\varepsilon}{2}}\right)$$

Since $|\varepsilon| \leq \frac{1}{2}$, $\varepsilon \leq \ln\left(\frac{1}{1-\varepsilon}\right) \leq \varepsilon + \varepsilon^2$

$$\Rightarrow (\varepsilon + \varepsilon^2) m^* \geq \ln n + \frac{\varepsilon}{2} M \Rightarrow M \leq 2(1+\varepsilon)m^* + \frac{2\ln n}{\varepsilon}. \quad \square$$

Randomized Weighted Majority

The same idea, but in each round, we return

$$b \in \{0,1\} \text{ w.p. } \sum_{i:\text{ expert } i \text{ predicts } b} \frac{w_i^{(t)}}{W^{(t)}}$$

Theorem: The randomized version makes at most

$$\mathbb{E}\{M\} \leq (1+\varepsilon)m^* + \frac{\ln n}{\varepsilon}$$

mistakes in expectation.

Proof: Denote by $q^{(t)}$ the probability that we make a mistake in step t. So, $q^{(t)} = \sum_{\text{incorrect } i} \frac{w_i^{(t)}}{W^{(t)}}$. Then,

$$W^{(t+1)} = \sum_i w_i^{(t+1)} = \sum_{\text{correct } i} w_i^{(t)} + (1-\varepsilon)\sum_{\text{incorrect } i} w_i^{(t)} = W - \varepsilon \sum_{\text{incorrect } i} w_i^{(t)}$$

$$= W^{(t)} - \varepsilon q^{(t)} W^{(t)} = (1-\varepsilon q^{(t)}) W^{(t)}$$

So, the final total weight is

$$W^{(T)} = n \prod_{t=1}^{T} (1-\varepsilon q^{(t)}) \leq n \prod_{t=1}^{T} e^{-\varepsilon q^{(t)}} = n e^{-\varepsilon \sum_{t=1}^{T} q^{(t)}} = n e^{-\varepsilon \mathbb{E}\{M\}}$$

Also, as before,

$$(1-\varepsilon)^{m^*} \leq W^{(T)} \Rightarrow (1-\varepsilon)^{m^*} \leq n e^{-\varepsilon \mathbb{E}\{M\}}$$

$$\Rightarrow m^*(\varepsilon + \varepsilon^2) \geq -\ln n + \varepsilon \mathbb{E}\{M\} \Rightarrow \mathbb{E}\{M\} \leq (1+\varepsilon) m^* + \frac{\ln n}{\varepsilon}. \qquad \square$$

Multiplicative Weights

In the general setting, there are T rounds.

- each round has n choices $\{1, \dots, n\}$, and we choose one.
- there is a cost $m_i^{(t)} \in [-1, 1]$ for choosing i in round t.
- We wish to minimize total cost.

Init: Fix parameter $\varepsilon \in (0, \frac{1}{2}]$ and give weight $w_i = 1$ to each choice.

For $t \in [T]$:

- return i w.p. proportional to w_i
- observe costs $\{m_i^{(t)}\}_{i \in [n]}$
- update $w_i \leftarrow w_i (1 - \varepsilon m_i^{(t)})$

Theorem: For every $i \in [n]$, the expected total cost is at most

$$\mathbb{E}\{M\} \leq \sum_{t=1}^{T} m_i^{(t)} + \varepsilon \sum_{t=1}^{T} |m_i^{(t)}| + \frac{\ln n}{\varepsilon}$$

$\leq T$ if $m_i^{(t)} \in [-1, 1]\ \forall i, t$

If we set $\varepsilon = \sqrt{\frac{\ln n}{T}}$, we make $O(\sqrt{T \ln n})$ more mistakes than the expert in question.

Lecture 10/27 - Online Algorithms

Ski Rental -

Every day that you ski, you can either:

(1) use skis you already bought
(2) rent skis for R
(3) buy skis for B

* An important part of the model is that you don't know, until it happens, whether you plan to ski.

On day 1, you go skiing and must decide.
After day i, you may never ski again, or you go skiing on day i+1.

We measure the result using the competetive ratio: $\max_{\text{all inputs}} \left\{ \frac{\text{your cost(input)}}{\text{OPT of (input)}} \right\}$

↑ input $= D \in \mathbb{N}$, the number of days I will ski

The offline problem has an $OPT(\text{input } D) = \min\{DR, B\}$
We wish to design an online algorithm that does well under the competetive ratio metric.

Any deterministic online algorithm is <u>fully defined</u> by T, the number of days we are willing to rent (rent $\forall t \le T$, buy on $t = T+1$, $t \in [D]$)

<u>Claim</u>: For any algorithm T, the competitive ratio is achieved at $D = T+1$.

<u>Proof</u>: For fixed T, D, get

$$\frac{R \min\{D,T\} + B \mathbb{1}_{D>T}}{\min\{DR, B\}} \quad \begin{array}{l} \leftarrow \text{price we pay} \\ \leftarrow \text{opt} \end{array}$$

(1) max cannot be achieved at $D > T+1$, since numerator doesn't change and denominator may grow.
(2) max cannot be achieved at $D < T$, since it is always ≤ 1.
(3) $D = T$ is $\le D = T+1$, using marginal logic.

(worst case is stopping skiing right after buying) $\square$

So, for any T, the competitive ratio is $\dfrac{RT+B}{\min\{R(T+1), B\}}$.

Claim: This is minimized at $T = B/R - 1$ (assuming $B/R \in \mathbb{N}$), yielding a competitive ratio of $2 - \dfrac{R}{B}$

Proof:

(1) the min is not achieved at $T > B/R - 1$; the denominator is constant while the numerator increases

(2) the min is not achieved at $T < \frac{B}{R} - 1$; the denominator and numerator both grow by R, and the numerator is larger than the denominator, so the competitive ratio decreases for each additional T.

List Update -

You manage a linked list. Online, you get requests to access x. You scan the list until you hit x. You are allowed to move x up in the list however much for free after returning.

Frequency count -

(1) Initialize $C(x) = 0 \ \forall x$

(2) If x is queried, increment $C(x)$

(3) move x up above all y with $C(x) > C(y)$

Claim: FC has competitive ratio $\Omega(n)$

Proof: Start by querying element i i times. Then, for some large K,

for _ in [K]:
    for $j \in [n]$:
        query j n times.

query back of the list n times

The offline optimum is, for each new query, move to front. This has total order $O(Kn^2)$

FC will pay $n+(n-1)+\dots = O(n^2)$ for each time we query j n times.
So, FC has cost $\Omega(Kn^3) \Rightarrow$ C.R. $= \Omega(n)$

$\square$

Move to front

Every time you query something, move it to the front.

MTF fails when querying in a cycle, which also causes offline optimum to suck.

Theorem: MTF has C.R. ≤ 2

offline opt

Proof: Imagine running MTF and OPT side-by-side. $\forall$ times t, denote by $\Phi(t)$ the # of pairs (x,y) s.t. $x >_{MTF} y$ but $y >_{OPT} x$

# of inversions

We can see that

(1) $\Phi(0) = 0$ (2) $\Phi(t) \geq 0$. Let $MTF(t), OPT(t)$ be the costs for query t.

Claim: $\forall t$, $MTF(t) + (\Phi(t) - \Phi(t-1)) \leq 2\,OPT(t)$

Proof: Consider accessing x @ time t. Let $MTF(x) = p$.
Suppose that k elements in front of x in MTF are also ahead of x in OPT. $\Rightarrow MTF(t) = p$, $OPT(t) \geq k+1$

the idea is if $k \sim p$, we did good; if $k \ll p$, we fix many inversions and help Φ

The MTF operation creates k inversions, but fixes $p-k$ inversions, if we were to not change OPT. Moving x forward in OPT can only improve things, since it can only fix inversions by agreeing that x is ahead of things. So, $\Phi(t) - \Phi(t-1) \leq 2k - p + 1 \leq 2\,OPT(t) - MTF(t)$

$\square$

Repeated application of the claim shows

$$MTF + \Phi \leq 2\,OPT \Rightarrow \text{C.R.} \leq 2.$$

$\square$

Lecture 11/8 - Communication Complexity

Def: A two-party communication problem consists of a function $f: \{0,1\}^a \times \{0,1\}^b \to \{0,1\}$. Alice receives input $A \in \{0,1\}^a$ and Bob receives $B \in \{0,1\}^b$. The goal is to compute $f(A,B)$.

Def: A deterministic communication protocol specifies for Alice as a function of her input A and all previous messages $a_1 b_1 a_2 b_2 \ldots a_k b_k$ what is the next message a_{k+1} Alice should send? Similarly for Bob.

Def: The communication cost is the maximum # of bits in all messages.

Ex/ Equality $\quad f(X,Y) = 1$ iff $X = Y$

Protocol: In each message i, Alice sends X_i and Bob sends Y_i

$\Rightarrow O(n)$ cost

Insert rest of lecture here

Lecture 11/10 - Computation of Nash

Consider rock-paper-scissors

	R	P	S
R	0, 0	-1, 1	1, -1
P	1, -1	0, 0	-1, 1
S	-1, 1	1, -1	0, 0

- This is a zero-sum game
- Consider the matrix of payoffs for the row player & the matrix for the column player. The rank of the sum of these matrices determines tractibility of computation of Nash.

Recall: • Nash Equilibrium is when both players are best responding to each other.
  ◦ Can be pure or mixed NE

We generalize: A 2 player game is given by two $n \times n$ matrices A, B where A_{ij} denotes the payoff to A if row plays i and col plays j, and B_{ij} . . .

mixed strategy distribution over actions $(\vec{x}, \vec{y})$ is a Nash Eq. if

$$\vec{x} A \vec{y} \geq A_{i\cdot} \vec{y} \quad \forall i \quad \& \quad \vec{x} B \vec{y} \geq \vec{x} B_{\cdot j} \quad \forall j$$

Def: $(\vec{x}, \vec{y})$ are ε-Nash Equilibrium if (Almost Nash)

$$\varepsilon + \vec{x} A \vec{y} \geq A_{i\cdot} \vec{y} \quad \forall i \qquad \varepsilon + \vec{x} B \vec{y} \geq \vec{x} B_{\cdot j} \quad \forall j$$

Computation of Nash

Given A, B, find a NE. This problem is PPAD-Complete. (define this later)
We can find ε-Nash or use LP rounding, etc.

Lipton/Markakis/Mehta

Theorem: There exist two multi-sets S, T, each of size $O(\log n/\varepsilon^2)$ s.t. it is an ε-Nash for A, B to randomly sample strategies uniformly from S, T, respectively.

set that can have repeat elements

This implies a brute force algorithm to exhaust all $n^{O(\frac{\log n}{\varepsilon^2})}$ pairs of multisets and check if any is ε-Nash.

quasi-polynomial time $O(n^{\log n})$

Side note: If $\forall \varepsilon$, $\exists$ a $o(n^{\log n})$ time algorithm for finding ε-Nash, there exists a $2^{o(n)}$ algorithm for PPAD.
(Rubinstein)

(If you can do better than the LMM alg. above, you can do sub-exponential PPAD!)

Proof of Theorem:

Let $(\vec{x}, \vec{y})$ be a NE (one must exist). Consider randomly sampling k strategies from $\vec{x}$ (call it S) and k strategies from $\vec{y}$ (call it T).

Define

$$x_i^* \equiv \frac{\#\text{ of times } i \in S}{k} \qquad y_i^* = \frac{\#\text{ times } i \in T}{k}$$

(Use empirical distribution gotten by sampling from Nash)

We want to show:

(1) $\forall i, \ |A_{i,\cdot} \cdot \vec{y} - A_{i,\cdot} \cdot \vec{y}^*| < \varepsilon$

(2) $\forall i, \ |B_{i,\cdot} \cdot \vec{x} - B_{i,\cdot} \cdot \vec{x}^*| < \varepsilon$

(3) $|\vec{x} A \vec{y} - \vec{x}^* A \vec{y}| < \varepsilon$

(4) $|\vec{x} B \vec{y} - \vec{x} B \vec{y}^*| < \varepsilon$

Chernoff City: Population Weinberg

From this, we want to show that $\vec{x}^* A \vec{y}^* \geq A_{i,\cdot} \vec{y}^* - 3\varepsilon \quad \forall i$

(1) gives $\vec{x}^* A \vec{y}^* \geq \vec{x}^* A \vec{y} - \varepsilon$ (each row is ε-accurate, and so is a distribution over rows)

(3) then gives $\geq \vec{x} A \vec{y} - 2\varepsilon$

N.E. then gives $\geq A_i \vec{y} - 2\varepsilon \quad \forall i$

(1) again gives $\geq A_i \vec{y}^* - 3\varepsilon \quad \forall i$

$\square$

Exponential Time Alg. For Exact Nash

(1) Assume WOLOG that $A = B^T$ (we can reduce anything to this by swapping players for half the actions)
This will look like playing against ourselves?

Consider the following: $\langle \vec{A}_{i,\cdot}, \vec{x} \rangle \leq 1 \quad \forall i$ (i doesn't give payoff more than $\frac{1}{|\vec{x}|_1}$ against $\frac{\vec{x}}{|\vec{x}|_1}$)
(LH polytope) $x_i \geq 0 \quad \forall i$ ($\vec{x}$ has pos. entries and is normalizable)

Being in this polytope means no strategy does better than $\frac{\vec{x}}{|\vec{x}|_1}$.

(2) We call an action i covered if $\langle \vec{A}_{i,\cdot}, \vec{x} \rangle = 0$ or $x_i = 0$ or both.

We claim: if $\vec{x}$ satisfies LHP and has all i covered (at least one ineq. is tight), then $\frac{\vec{x}}{|\vec{x}|_1}$ is Nash.

Proof: Consider using i against $\frac{\vec{x}}{|\vec{x}|_1}$. If $x_i = 0$, not used and we don't care. Otherwise, i covered $\Rightarrow \langle \vec{A}_{i,\cdot}, \frac{\vec{x}}{|\vec{x}|_1} \rangle = \frac{1}{|\vec{x}|_1} \Rightarrow i$ is a BR $\square$

The Algorithm: (Pivoting) (also a proof that N.E. exists)

Start from a vertex of the polytope, "walk" along the boundary (keeping all but one constraint tight) until the next vertex (pivoting).

Start at $\vec{0}$.
"relax" $x_i = 0$, see which constraints tighten to get next $\vec{x}$ (can be done w/ LP, simplex)
If $\vec{x}$ covers all i, done!
If not, $\exists i$ that is "double-covered", relax one and continue. $\square$

See lecture notes for details

Note that any decision problem (is there a Nash s.t. ...) is NP-Hard, but no such problem implies that no Nash exists. So, Nash $\notin$ NP-Hard

PPAD-Complete Examples

Given a graph on 2^n nodes s.t. every node has indegree ≤ 1 and outdegree ≤ 1 with a source node (indegree 0), find a sink (we can nonconstructively prove a sink exists)

Given $|S| = 2^n$, $|T| = 2^{n-1}$, there exists a function $f: S \to T$ that maps s_1, s_2 to the same t.

Lecture 11/15 – Low-Rank Approx.

Let $\vec{a}_1, \dots, \vec{a}_n \in \mathbb{R}^d$ be data points. We seek $b_1, \dots, b_k \in \mathbb{R}^d$ $(k \ll d)$ and $\{c_{ji}\}_{j\in[k], i\in[n]}$ s.t. $\vec{a}_i \approx \sum_{j=1}^{k} c_{ji} \vec{b}_j$ (approximately in low dimensional subspace)

Equivalently, let $A = \begin{pmatrix} | & & | \\ a_1 & \cdots & a_n \\ | & & | \end{pmatrix} \in \mathbb{R}^{d\times n}$. We seek $B = \begin{pmatrix} | & & | \\ b_1 & \cdots & b_k \\ | & & | \end{pmatrix} \in \mathbb{R}^{d\times k}$

and $C \in \mathbb{R}^{k\times n}$ s.t. $A \approx BC$.

We wish to minimize the following for a given matrix $A \in \mathbb{R}^{d\times n}$ and $k \ll d$:

Frobenius norm $\|M\|_F^2 = \sum_{i,j} |M_{ij}|^2$

$$\underset{B\in\mathbb{R}^{d\times k},\, C\in\mathbb{R}^{k\times n}}{\operatorname{argmin}} \text{ error}, \qquad \text{error} = \sum_{i=1}^{n} \Big\|\vec{a}_i - \sum_{j=1}^{k} c_{ji}\vec{b}_j\Big\|_2^2 = \|A - BC\|_F^2$$

SVD

Theorem: (SVD exists)

Let $A \in \mathbb{R}^{d\times n}$ be a matrix. Let $r = \min(d,n)$. Then, there exist matrices U, Σ, V^T

- $U \in \mathbb{R}^{d\times r}$; the columns of U (left singular vectors) are orthonormal
- $\Sigma \in \mathbb{R}^{r\times r}$; Σ is $\operatorname{diag}(\sigma_1, \dots, \sigma_r)$ s.t. the singular values have $\sigma_1 \geq \dots \geq \sigma_r \geq 0$
- $V \in \mathbb{R}^{n\times r}$; the columns of V (right singular vectors) are orthonormal

$$☆ \; U\Sigma V^T = A \; ☆$$

This leads to some interesting properties:

- $A^TA = V\Sigma^2V^T$
- $AA^T = U\Sigma^2U^T$
- Singular values are square roots of eigenvalues
- If $\vec{v}_i$ column of V, $A\vec{v}_i = U\Sigma(V^T\vec{v}_i) = U\Sigma\begin{pmatrix}0\\ \vdots \\ 1 \\ \vdots \\ 0\end{pmatrix} = \sigma_i \vec{u}_i$

Theorem: (SVD is best)

For any $k \in [1, r]$, let U_k be the matrix of size $d \times k$ consisting of the first k columns of U. Let $V_k \in \mathbb{R}^{n \times k}$ and $\Sigma_k \in \mathbb{R}^{k \times k}$ be defined similarly. Then,

$$\|A - U_k \Sigma_k V_k^T\|_F^2 = \min_{\substack{B \in \mathbb{R}^{d \times k} \\ C \in \mathbb{R}^{k \times n}}} \|A - BC\|_F^2$$

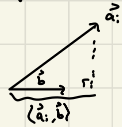
Proof:

(k=1) Consider the case $k=1$. We seek $\vec{b} \in \mathbb{R}^d$, $\vec{c} \in \mathbb{R}^n$ s.t.

$$\|A - \vec{b}\vec{c}^T\|_F^2 = \sum_{i=1}^n \|a_i - c_i \vec{b}\|_2^2 \quad \text{is minimized.}$$

For any given $\vec{b}$, (suppose WOLOG that $\|\vec{b}\| = 1$, since we can scale down $\vec{b}$ and scale up c_i) this is minimized for the ith term $\|\vec{a}_i - c_i \vec{b}\|^2$ when $c_i = \langle \vec{a}_i, \vec{b} \rangle$.

The minimum is then $\|\vec{a}_i - c_i \vec{b}\|_2^2 = \|\vec{a}_i\|^2 - \|c_i \vec{b}\|^2 = \|\vec{a}_i\|^2 - c_i^2$



So, we wish to maximize

$$\max \sum c_i^2 = \sum \langle \vec{a}_i, \vec{b} \rangle^2 = \|A^T \vec{b}\|_2^2$$

For given A, we find unit vector $\vec{b}$ maximizing $\|A^T \vec{b}\|_2^2$, and set $\vec{c} = A^T \vec{b}$.

Let $A = U \Sigma V^T \Rightarrow \|A^T \vec{b}\|_2^2 = \|V \Sigma U^T \vec{b}\|_2^2 \underset{\substack{V \text{ orthonormal} \\ \text{columns} \\ \forall \vec{x},\ \|V\vec{x}\| = \|\vec{x}\|}}{=} \|\Sigma U^T \vec{b}\|_2^2 = \sum_{i=1}^r \sigma_i^2 \langle \vec{u}_i, \vec{b} \rangle$

Since $\|\vec{b}\|_2 = 1$, $\sum_i \langle \vec{u}_i, \vec{b} \rangle = 1$ since $\{\vec{u}_i\}$ orthonormal

So, we maximize when $\vec{b} = \vec{u}_1$ ($\langle \vec{u}_1, \vec{b} \rangle = 1$ and $\vec{u}_1$ has largest singular value σ_1)

Therefore, the error is minimized for $\vec{b} = \vec{u}_1$, $\vec{c} = A^T \vec{u}_1 = V \Sigma U^T \vec{u}_1 = \sigma_1 \vec{v}_1$.

The claim holds for $k=1$.

(k>1) We do the same thing. For any given $B \in \mathbb{R}^{d \times k}$ with orthogonal columns, the best C is $C = B^T A$. We seek the B maximizing

$$\|B^T A\|_F^2 = \|B^T U \Sigma V^T\|_F^2 = \|B^T U \Sigma\|_F^2 = \sum_{i=1}^r \sigma_i^2 \|B^T \vec{u}_i\|_2^2 \le \sum_{i=1}^k \sigma_i^2$$

Since B has orthonormal columns, $\sum_{i=1}^r \|B^T \vec{u}_i\|_2^2 \le k$, $\|B^T \vec{u}_i\|_2^2 \le 1$.

So, we want $\|B^T \vec{u}_i\|_2^2 = 1 \ \forall i \Rightarrow B = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_k \\ | & & | \end{pmatrix} = U_k$. Then, $C = B^T A = B^T U \Sigma V^T = \Sigma_k V_k^T$

∎

Algorithm (SVD Solver)

- Initialize $A^{(1)} = A$
- For $i = 1, \ldots, r$:
  - compute the optimal rank 1 approximation of $A^{(i)}$ $\left(\begin{array}{l}\text{find } \vec{b}^{(i)} \in \mathbb{R}^d, \ \vec{c}^{(i)} \in \mathbb{R}^n \text{ s.t.} \\ \|A^{(i)} - \vec{b}^{(i)}\vec{c}^{(i)T}\|_F^2 \text{ minimized}\end{array}\right)$
  - update $A^{(i+1)} \leftarrow A^{(i)} - \vec{b}^{(i)}\vec{c}^{(i)T}$, $\vec{u}_i = \dfrac{\vec{b}^{(i)}}{\|\vec{b}^{(i)}\|_2}$, $\vec{v}_i = \dfrac{\vec{c}^{(i)}}{\|\vec{c}^{(i)}\|_2}$, $\sigma_i = \|\vec{b}^{(i)}\|_2 \|\vec{c}^{(i)}\|_2$
- Set $U = (\vec{u}_1, \ldots, \vec{u}_r)$, $V = (\vec{v}_1, \ldots, \vec{v}_r)$, $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_r)$.

We need to show that $A^{(r+1)} = 0$ and that $\{\vec{u}_i\}_i$ and $\{\vec{v}_j\}_j$ are both orthonormal.

<u>Claim:</u> For any round i with $\vec{b} = \vec{b}^{(i)}$, $\vec{c} = \vec{c}^{(i)}$, $A = A^{(i)}$,
1. $\vec{b} \in$ column space of A
2. $\vec{b} \perp$ column space of $(A - \vec{b}\vec{c}^T)$

This claim (if we were to prove it) shows

$$\vec{b}_i \in \text{span}\{A_i\} \text{ and } \vec{b}_i \perp \text{span}\{A_{i+1}\} \Rightarrow \vec{b}_i \perp \vec{b}_j \text{ for } i \neq j.$$

and $\quad \text{span}\{A_{i+1}\} \subseteq \text{span}\{A_i\} \Rightarrow$ dim reduces by 1 each round.

These two results show U is orthonormal and $A^{(r+1)} = 0$.

$\square$

We need to fill in one piece: finding the best rank 1 approximation for A. We will use the **Power Method**. The idea is we wish to find the top eigenvalue of A^TA. We keep multiplying a vector by A^TA, which will push it more in the direction of the top eigenvector of A^TA (or AA^T, same spectrum).

Power Method

- initialize $\vec{z}$ to random vector with i.i.d $\mathcal{N}(0,1)$ entries. Set $\vec{z}_1 = \dfrac{\vec{z}}{\|\vec{z}\|_2}$.
- For $t = 1, \ldots, T$: ← large # rounds
  - set $\vec{z}_{t+1} \leftarrow AA^T\vec{z}_t$. Normalize $\vec{z}_{t+1} \leftarrow \dfrac{\vec{z}_{t+1}}{\|\vec{z}_{t+1}\|_2}$
- Return $\vec{z}_T$ as $\vec{b}$. Return $\vec{c} = A^T\vec{b}$.

This works because

$$\vec{z}_1 = \sum_i \alpha_i \vec{u}_i \Rightarrow \vec{z}_2 \propto AA^T \vec{z}_1 = \sum_i \sigma_i^2 \alpha_i \vec{u}_i \Rightarrow \dots \Rightarrow \vec{z}_T = \sum_i (\sigma_i^2)^T \alpha_i \vec{u}_i$$

largest eigenvalue will dominate

If we set $T = O\left(\frac{\log d}{\varepsilon}\right)$, we have

$$\|A - \vec{b}(A^T\vec{b})^T\|_F^2 \le (1+\varepsilon)\underbrace{\|A - \sigma_1 \vec{u}_1 \vec{v}_1^T\|_F^2}_{\text{top-1 SVD}}$$

The total time to find $\vec{b}, \vec{c}$ is $O\left(\frac{\log d \, nd}{\varepsilon}\right)$

Therefore, the total time to find the K-rank SVD approximation is

$$\boxed{O\left(\frac{knd \log d}{\varepsilon}\right)}$$

Lecture 11/29 - State Data Structure Lower Bound

Polynomial Evaluation

Given a polynomial $P \in \mathbb{F}_q[x]$ of degree n, where q is prime, we would like to preprocess it into a data structure of size S s.t. given a query $x \in \mathbb{F}_q$, $P(x)$ can be computed in time T

We focus on minimizing S and T.

There are two trivial structures:

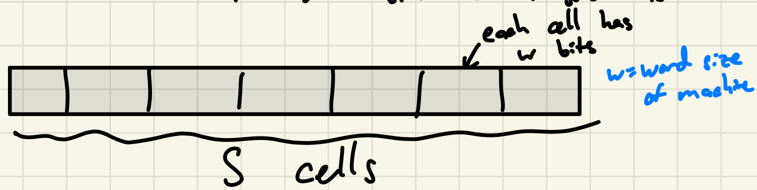
① Store coefficients of P — $O(n)$ space + $O(n)$ query time
② Precompute $P(x)$ $\forall x \in \mathbb{F}_q$ — $O(q)$ space + $O(1)$ query time

A nontrivial result from [Kellaga Umans '08] is that $\forall \delta > 0$, we can achieve $S = O(n^{1+\delta} \operatorname{polylog} q)$, $T = O(\operatorname{polylog}(n, q))$

Today we will prove lower bounds on (S, T)!

Def: [Yao '81] The Cell-Probe Model for data structure analysis is

Memory of size S:



① Cells are indexed by $[S]$
② Can read/write a cell in unit time
③ Computation is free $\Rightarrow$ $T :=$ # of cell probes (read/writes) the algorithm makes

Since this model is stronger than actual computing, lower bounds here apply everywhere!

We make the usual assumption $w \geq \Omega(\log n + \log q)$ (can store pointers and elements of $\mathbb{F}_q$)

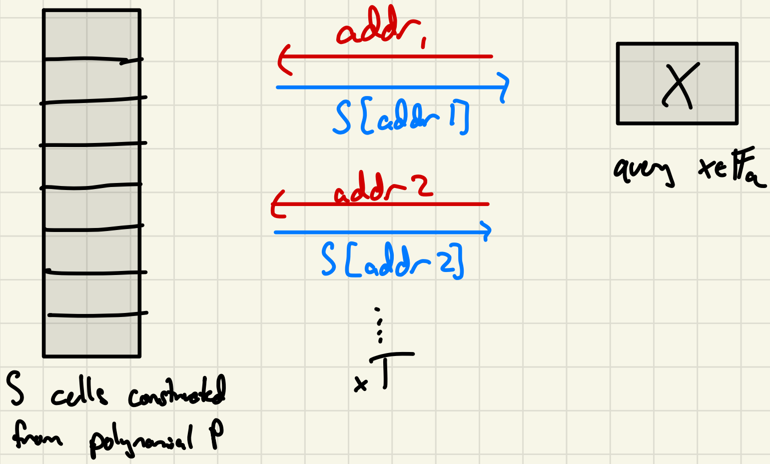
An interesting setting we will focus on is when $w = \Theta(\log n)$ and $q = \text{poly}\, n$.

Reduction from Communication Complexity Problem [Miltersen, Nisan, ... '94]

Recall: Alice gets input X and Bob gets input Y, and the goal is to compute a function $f(X,Y)$ using minimal communication.

We can view polynomial evaluation as a communication game as follows:

Alice knows the polynomial   Bob knows the query



Memory accesses are a 2-way communication! Formally,

Lemma: Suppose that $\exists$ a data structure for polynomial evaluation w/ space S and query time T. Then, there is a protocol for the following communication protocol:

- Alice gets P. Bob gets x. The goal is to send $P(x)$ s.t. Bob sends $T \cdot \log S$ and Alice sending Tw bits.

Proof: Alice preprocesses P into a data structure of size S locally. Then, Bob & Alice simulate the query alg:

For $t \in [T]$:

- Bob sends an address in $\log S$ bits
- Alice responds with the cell contents in w bits

$\square$

So, a lower bound in this communication problem is a lower bound for PE. Note that this reasoning works for any data structure in the cell-probe model!

Claim: To compute $P(x)$, for any $c \in [0, \min\{\log n, \log q\}]$,

- Bob sends $\geq 2^c \log q$ bits
- Alice sends $\geq \log q - c$ bits

← Communication lower bound

Proof: Omitted :)

This yields that

$$\forall c \quad \text{s.t. } Tw < 2^c \log q, \quad T \cdot \log S \geq \log q - c$$

$$\Rightarrow S^T \geq \frac{q}{2^c} \quad \forall c \text{ s.t. } 2^c > \frac{Tw}{\log q}$$

$$\Rightarrow S^T \geq \frac{q \log q}{Tw} \Rightarrow \boxed{S \geq \left(\frac{q \log q}{Tw}\right)^{1/T}}$$

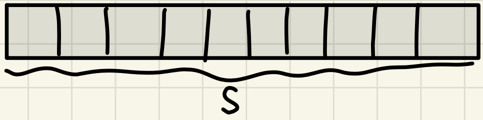
When $T = 1$, this means $S \geq \dfrac{q \log q}{w}$ ← # bits to store q elements of $\mathbb{F}_q$; ← # bits/cell
(which is the second trivial solution!)

Cell Sampling

Suppose now that T is a large constant s.t. $\left(\frac{q \log q}{Tw}\right)^{1/T} < n$ ← when $q = \text{poly } n$

We focus on when $w = \log q$ (each cell stores one element of $\mathbb{F}_q$) and $q = \text{poly } n$ for convenience.

The idea is to find a small set of cells C s.t. too many queries can be answered by accessing C.



We want to produce contradictions by being able to reconstruct the polynomial too easily.

For $T=1$, $\exists$ one cell that is accessed by q/S different queries.
In general,

<u>Lemma</u>: Let $\varepsilon>0$. Suppose that $\exists$ a data structure for polynomial evaluation w. space S and time $T(\ll n)$. Then, there exists a set C of εn cells st. $\geq q\left(\frac{\varepsilon n}{S}\right)^{O(T)}$ different queries can be answered by only accessing the cells in C.

<u>Proof:</u> We will do this by a probabilistic argument.

1. Sample a random set C of εn cells.
2. Fix a query $x \in \mathbb{F}_q$.

Then, $$\mathbb{P}_C\left\{\begin{matrix}x \text{ can be answered} \\ \text{by accesses of only } C\end{matrix}\right\} = \frac{\binom{S-T}{|C|-T}}{\binom{S}{|C|}} = \frac{S-T!}{S!}\cdot\frac{|C|!}{(|C|-T)!}$$

$$= \frac{|C|\cdot(|C|-1)\cdot\ldots\cdot(|C|-T+1)}{S\cdot(S-1)\cdot\ldots\cdot(S-T+1)} \geq \left(\frac{|C|-T}{S}\right)^T \geq \left(\frac{\varepsilon n}{S}\right)^{O(T)} \quad (T\ll\varepsilon n)$$

3. By linearity, $\mathbb{E}\{\#x \text{ that can be answered within } C\} \geq q\left(\frac{\varepsilon n}{S}\right)^{O(T)}$. So, there exist this many queries for some set C.

$\square$

So, this setup would allow us to answer enough queries to reconstruct an n-degree polynomial with εn cells if $q\left(\frac{\varepsilon n}{S}\right)^{O(T)} > n$. Therefore,

<u>Theorem</u>: We must have $q\left(\frac{\varepsilon n}{S}\right)^{O(T)} \leq n \iff T \geq \Omega\left(\frac{\log(q/n)}{\log(S/n)}\right)$

if $S=O(n)$, $T\geq\Omega(\log q/n) = \Omega(\log n)$

if $T=O(1)$, $S\geq n^{1+\Omega(1)}$

<u>Proof:</u> Suppose BWOC that $q\left(\frac{\varepsilon n}{S}\right)^{O(T)} \geq n+1$.

encode

1. Construct the data structure and find the set C with the claimed property (from the Lemma).
2. Write down the (address, content) pairs for all cells in C. This is the encoding.
3. To decode,

decode

   (a) Recover C from reading the encoding
   (b) Query the algorithm $\forall x \in \mathbb{F}_q$. Collect the answers for all queries that can be answered within C. The Lemma condition implies that we will have $\geq q\left(\frac{\varepsilon n}{S}\right)^{O(T)} \geq n+1$ different $(x, P(x))$ pairs.

This uniquely determines the polynomial by interpolation.

So, we get a procedure that can encode the whole polynomial in $\leq |C|(\log S + w) = \varepsilon n(\log S + \log q) < (n+1)\log q$

Therefore, $\exists P_1, P_2$ with the same encoding. Contradiction!

$\square$

Lecture 12/1 - Fine-Grained Complexity

We focus on K-SAT: given n variables $x_1, \dots, x_n$ and a K-CNF formula $C_1 \wedge C_2 \wedge \dots \wedge C_m$, where each C_i is of the form $y_1 \vee y_2 \vee \dots \vee y_k$ and y_i is either x_t or $\neg x_t$ for some $t \in [n]$, compute if $\exists$ an assignment $\vec{x} \in \{0,1\}^n$ that satisfies all C_i.

Brute force: try all 2^n assignments.
Best known: $O(2^{n(1-\frac{c}{k})} \cdot m^d)$ for constants $c, d > 0$

There is a hypothesis that this is the best we can do.

Strong Exponential Time Hypothesis - (implies $P \neq NP$)
$\forall \varepsilon > 0$, $\exists k \geq 3$ s.t. K-SAT cannot be solved in $O(2^{(1-\varepsilon)n} \text{poly } m)$

Theorem [Impagliazzo, Paturi, Zane '01]
SETH $\iff$ SETH with $m = O(n)$

Consider the following problem:

Orthogonal Vectors (OV)

Input: a set of N vectors in $\{0,1\}^d$ $(d = O(\log N))$
Output: if $\exists u, v$ s.t. $\langle u, v \rangle = 0 \iff u \wedge v = \emptyset$

Brute force: $O(N^2 d)$ time, compute $\langle u, v \rangle \ \forall u, v$

Theorem [Williams '04]

$$\text{SETH} \Rightarrow \forall \delta > 0 \text{ OV cannot be solved in } O(n^{2-\delta} \text{poly } d) \text{ time}$$

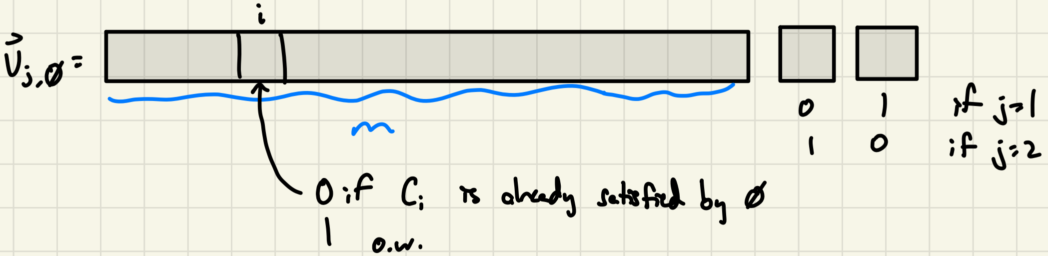
Proof: We want to prove the contrapositive: with an $O(n^{2-\delta} \text{poly } d)$ time OV alg., we can construct a $O(2^{(1-\frac{\delta}{2})n} \text{poly } m)$ K-SAT alg.

Consider a K-SAT instance $C_1, \dots, C_m$, $m = O(n)$

- divide the variables $\{x_1, \dots, x_n\}$ into V_1, V_2 of size $\frac{n}{2}$
- for $j = 1, 2$ consider all possible $2^{n/2}$ partial assignments $\emptyset$ to V_j

  construct a vector of dimension $m+2$ for each $(j, \emptyset)$

The vectors are constructed below



- there are $2^{\frac{n}{2}+1}$ vectors in total.

<u>Claim:</u> two vectors are orthogonal iff they combine to satisfy.

i.e. $\langle \vec{v}_{j_1,\emptyset_1}, \vec{v}_{j_2,\emptyset_2} \rangle = 0$ iff $j_1 \neq j_2$ and $(\emptyset_1, \emptyset_2)$ satisfies.

So, $\exists \emptyset_1, \emptyset_2$ s.t. $\langle \vec{v}_{1,\emptyset_1}, \vec{v}_{2,\emptyset_2} \rangle = 0 \iff C_1 \wedge C_2 \wedge \dots \wedge C_m$ is satisfiable

$\Rightarrow$ alg for OV in $O(N^{2-\delta} \text{poly } d) \Rightarrow$ K-SAT in time

$$O\left(\left(2^{\frac{n}{2}}\right)^{2-\delta} \text{poly } n\right) = O\left(2^{n(1-\frac{\delta}{2})} \text{poly } n\right)$$

$\square$

<u>Graph Diameter</u>

Given an undirected, unweighted graph $G = (V, E)$ w/ $|V| = n$, $|E| = m$

Compute $D = \max_{u,v \in V} d_G(u,v)$

<u>Brute force:</u> breadth-first search in $O(mn)$ time

[RV '13] $\frac{3}{2}$-approx in time $O(m^{1.5} \text{polylog } m)$

$\left(\frac{2D}{3} \leq D' \leq D\right)$

Theorem:

SETH $\Rightarrow \forall \varepsilon > 0$, $(\frac{3}{2} - \varepsilon)$-approx must take $m \cdot n^{1-o(1)}$ time

Proof: Reduce from OV on N vectors. Check lecture notes.

This shows that if $\exists$ alg. for diameter in $mn^{1-\delta}$ time,

$\Rightarrow$ OV is solved in $O(N^{1-\delta} N d) = O(N^{2-\delta} d)$

$\square$

Now let us look at this through 3-SUM:

given a set S of n numbers, output whether

$\exists\, a,b,c \in S$ s.t. $a+b=c$

There is also 3-SUM convolution:

given $A[1,\dots,n]$, output whether $\exists\, x,y \in [n]$ s.t. $A[x]+A[y]=A[x+y]$

Naive: $O(n^2)$ for both

3-Sum Conjecture - $\forall \delta > 0$, no alg. solves 3-SUM in $O(n^{2-\delta})$ time.

$\iff$

$\forall \delta > 0$, no alg. solves 3-SUM-conv in $O(n^{2-\delta})$ time

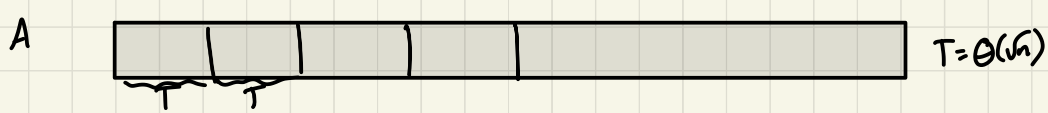
Exact triangle (ET)

Given a weighted, undirected graph G, output if $\exists\, a,b,c \in V$ s.t.

$w(a,b) + w(b,c) + w(c,a) = 0$

Theorem: If $\exists\, O(n^{3-\delta})$ alg. for ET, then $\exists$ 3-Sum-conv alg. with $O(n^{2-\frac{\delta}{2}})$ time

Proof: Consider an input A to 3-Sum-conv. We will construct $O(\sqrt{n})$ graphs $G_1, \dots, G_{\sqrt{n}}$ of size $O(\sqrt{n})$.



G_i is tripartite U_i, V_i, W_i

- U_i has vertices $j \in [n/T]$
- V_i has vertices $s \in [T]$
- W_i has vertices $t \in [2T]$

- for $j \in U_i$, $s \in V_i$ add edge (j,s) with weight $A[jT+s]$
- for $j \in U_i$, $t \in W_i$ add edge (j,t) with weight $-A[(i+j)T+t]$
- for $s \in V_i$, $t \in W_i$ add edge (s,t) with weight $A[iT+(t-s)]$

<u>Claim:</u> G_i has a zero-Δ iff $\exists x$ in block i s.t. $A[x]+A[y]=A[x+y]$

Since $|G_i| = O(T+\frac{n}{T})$, $i \in [n/T]$, if $\exists$ ET alg in time $O(n^{3-\delta})$, total time is $O(\sqrt{n}\cdot(\sqrt{n})^{3-\delta}) = O(n^{2-\delta/2})$.

$\square$

Lecture 12/6 - Differential Privacy

Def: A database D is a tuple $(\vec{x}_1, \dots, \vec{x}_n)$.

Def: A counting query q is a predicate that takes input $\vec{x}$ and outputs $q(\vec{x}) \in \{0,1\}$. Over a whole DB, $q(D) = \sum_{i=1}^{n} \frac{q(\vec{x}_i)}{n}$

In the worst case, if the whole world were out to get you or an attacker had all the possible outside information, even a large-scale survey where you answer honestly is not private (even when n large, $0 \ll q(D) \ll 1$)

examples

① all other respondents know what they put and can find your answer

② Netflix de-anonymization via pattern-matching with external IMDB DB.

We would like machinery to robustly prove that no matter what an attacker knows, they can't break your privacy.

Def: A randomized algorithm M is α-accurate for q if, w.h.p.,

$$|M(D) - q(D)| \leq \alpha \quad \forall D$$

Def: A randomized algorithm M is ε-differentially private if $\forall i$, all D, D' s.t. $D_{-i} = D'_{-i}$, $\forall$ sets S of possible outputs,

$\forall$ pairs of databases differing by at most 1 respondent

$$\mathbb{P}\{M(D) \in S\} \leq e^{\varepsilon} \, \mathbb{P}\{M(D') \in S\}$$

$$\Longleftrightarrow$$

$$\forall \text{ outputs } r, \quad \left| \ln\left(\frac{\mathbb{P}\{M(D) = r\}}{\mathbb{P}\{M(D') = r\}} \right) \right| \leq \varepsilon$$

We want to ensure that ε-DP ensures that your participation <u>cannot</u> affect anyone else's (insurance, Mom, etc.) Bayesian prior about you.

<u>Prop:</u>

Formally, suppose that someone has a Bayesian prior P about the database state that they will update to $\bar{P}$ after seeing M on the database. ε-DP guarantees

$$\mathbb{P}_{D\in\bar{P}}\{D \mid M(D)=r\} \in e^{\pm 2\varepsilon}\, \mathbb{P}_{D\in\bar{P}'}\{D \mid M(D)=r\}$$

<u>Proof:</u>

$$\mathbb{P}_{D\in\bar{P}}\{D \mid M(D)=r\} = \frac{\mathbb{P}\{D\in P \wedge M(D)=r\}}{\mathbb{P}\{M(D)=r\}} = \frac{\mathbb{P}\{D\in P\}\ \mathbb{P}\{M(D)=r\}}{\sum_{\hat{D}} \mathbb{P}\{M(\hat{D})=r\}\ \mathbb{P}\{\hat{D}\in P\}}$$

$$\in \frac{\mathbb{P}\{D\in P\}\cdot e^{\pm\varepsilon}\,\mathbb{P}\{M(D')=r\}}{\sum_{\hat{D}'} \mathbb{P}\{M(\hat{D}')=r\}\ e^{\pm\varepsilon}\ \mathbb{P}\{\hat{D}\in P\}} = e^{\pm 2\varepsilon}\, \mathbb{P}_{D\in\bar{P}'}\{D \mid M(D)=r\}$$

$\leftarrow \bar{P}$ if they saw $M(D')=r$ instead of $M(D)=r$

$\square$

<u>Idea 1:</u> Randomized response

With probability p give correct answer $q(\vec{x}_i)$, w.p. $1-p$ flip it $\forall i$. Your response will be more private without worrying about the total dataset or the algorithm. The output vector is $\vec{r} = (r_1, \dots, r_n)$.

$$\Rightarrow \frac{\mathbb{P}\{M(D)=\vec{r}\}}{\mathbb{P}\{M(D')=\vec{r}\}} = \frac{\cancel{\mathbb{P}\{M_{-i}(D)=\vec{r}_{-i}\}}\ \mathbb{P}\{M_i(D)=r\}}{\cancel{\mathbb{P}\{M_{-i}(D')=\vec{r}_{-i}\}}\ \mathbb{P}\{M_i(D')=r\}} = \frac{p}{1-p} \approx \varepsilon$$

if $p = \frac{1}{2} + \frac{\varepsilon}{2}$.

The estimate should then be $\frac{1}{2p-1}\left(\left(\sum_i \frac{r_i}{n} - (1-p)\right)\right)$

which is correct in expectation with variance $\frac{p(1-p)}{n(2p-1)^2}$

Idea 2: Add noise

Add noise to each response r_i drawn from $\text{Lap}\left(\frac{1}{\varepsilon n}\right)$. So, the PDF of the noise is $f(x) = \dfrac{1}{2\left(\frac{1}{\varepsilon n}\right)} e^{-|x| / \left(\frac{1}{\varepsilon n}\right)}$

We are concerned with the density ratios between D and D'

$$\frac{f_{M(D)}(r)}{f_{M(D')}(r)} = \frac{e^{-\varepsilon n |r - q(D)|}}{e^{-\varepsilon n |r - q(D')|}} \leq e^{-\varepsilon}$$

differ by at most $\frac{1}{n}$ because count changes by ≤ 1 when changing one response

It is correct in expectation with variance $\dfrac{2}{(\varepsilon n)^2}$.

Def: A randomized algorithm M is (ε, δ)-differentially private if $\forall i$, all $\underbrace{D, D' \text{ s.t. } D_{-i} = D'_{-i}}_{\text{pairs of databases differing by at most 1 respondent}}$, $\forall$ sets S of possible outputs,

$$\forall \text{ outputs } r, \quad \frac{\mathbb{P}\{M(D) = r\}}{\mathbb{P}\{M(D') = r\}} \leq e^{\varepsilon} + \delta\, \mathbb{P}\{M(D') = r\}$$

← check this

Theorem: If $M_1, \ldots, M_k$ all ε-DP, then the algorithm that answers all queries $(M_1, \ldots, M_k)$ is $k\varepsilon$-DP.

If $M_1, \ldots, M_k$ all (ε, δ)-DP, then $(M_1, \ldots, M_k)$ is $\left(k\varepsilon^2/2 + \sqrt{2k \ln\left(\frac{1}{\delta}\right)}\,\varepsilon, \delta\right)$-DP.

Proof: Lol no :)

D

ε-DP is also robust to groups of individuals!

ε-DP is also robust to postprocessing:

$\forall$ algorithms A, M is ε-DP $\Rightarrow$ $A \circ M$ is ε-DP

Lecture 12/8 - Smoothed Analysis

① given a worst-case input $\vec{x}$ (adversarial)

② randomly smooth $\vec{x}$ to $\vec{y}$ using some distribution of magnitude σ
(maintain adversarial big picture, but randomize lower-order bits)

③ $\vec{y}$ is the true input

④ $\forall \vec{x}$ (even adversarial), $\mathbb{E}_{y \in \text{smooth}_\sigma(\vec{x})} \{\text{runtime}_A(\vec{y})\} = \text{poly}(|\vec{x}|, \frac{1}{\sigma})$

Super cool result we won't prove:

Theorem: (Spielman, Teng '01)

Let $\vec{c}$, A, $\vec{b}$ define a LP.

objective

constraint matrix

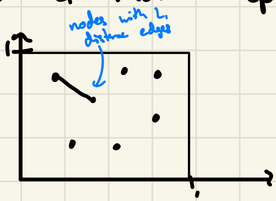
Let smooth$(\vec{c}, A, \vec{b})$ add i.i.d. $\mathcal{N}(0, \sigma^2)$ to A_{ij}, b_j $\forall i, j$.
Then, the simplex algorithm is smoothed poly-time in this model.

exponential worst-case

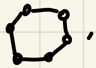
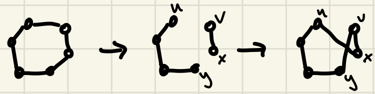
check Tim's notes for discussion about what this means about simplex in practice

Metric TSP

Consider a metric space $[0,1] \times [0,1] \subset \mathbb{R}^2$ with L_1 metric.



TSP is to find the lowest cost Hamiltonian cycle.

There is an algorithm called 2-OPT that performs local searches. Basically, for any current tour , consider replacing each pair of nonadjacent edges one-by-one as follows: 

Keep the improvements and terminate when no pair replacement helps.

2-OPT has <u>worst case exponential time</u> to terminate but smoothed polynomial runtime, as we will see below.

<u>Theorem:</u> If node x_i is smoothed to y_i independently according to any distribution w/ PDF f_i s.t. $f_i(y) \le \frac{1}{\sigma}$ $\forall y$ (bounded density), then 2-OPT is smoothed poly-time.

<u>Def:</u> A swap $(u,v),(x,y)$ is ε-bad if

$$\|u-v\|_1 + \|x-y\|_1 - (\|u-x\|_1 + \|v-y\|_1) \in (0,\varepsilon)$$

Swapping makes $<\varepsilon$ progress, so alg. continues with very little progress

<u>Lemma:</u> $\forall \vec{x}$, $\mathbb{P}_{\vec{y} \in \mathrm{smooth}_\sigma(\vec{x})} \{ \underline{\text{any}} \text{ swap in } \vec{y} \text{ is } \varepsilon\text{-bad} \} \le \frac{\varepsilon n^4}{\sigma}$

<u>Proof of Lemma:</u> First, observe that there are n^4 possible choices of $((u,v),(x,y))$ and so if all of these are good, there can be no graph with an ε-bad swap. For each $(u,v),(x,y)$, let

$$\star = |u_1-v_1| + |u_2-v_2| + |x_1-y_1| + |x_2-y_2| - |u_1-x_1| - |u_2-x_2| - |v_1-y_1| - |v_2-y_2|$$

If we fix the relative ordering of $\{u_1,v_1,x_1,y_1\}$ and $\{u_2,v_2,x_2,y_2\}$, then $\star$ is linear in all vars and all coeffs are in $\{-2,0,2\}$. So, $\star \in \{$linear fns with coeffs $\in \{-2,0,2\}\}$. There are $(4!)^2 n^4$ possible linear functions. Now, for any linear function in this set, in our smoothed model, we want to show that it is $\in (0,\varepsilon)$. If all the coefficients are 0, it holds trivially.

Now, suppose WOLOG that u_1 has a ± 2 coefficient. Sample the smoothed versions of all other variables except u_1.
The function is
$$\pm 2 u_1 + \underbrace{\alpha_2 u_2 + \alpha_3 v_1 + \ldots}_{= C \text{ for some } C}$$
So, the function is in $(0, \varepsilon)$ iff $u_1 \in \left(\frac{-C}{\pm 2}, \frac{\varepsilon - C}{\pm 2} \right)$

of width $\frac{\varepsilon}{2}$. The max probability that u_1 can lie in this range is $\leq \frac{\varepsilon}{2\sigma}$ because of the bounded densities.

For each possible swap, a union bound over the $(4!)^2$ possible functions yields that $\mathbb{P}\{\varepsilon\text{-bad swap}\} \leq \frac{(4!)^2}{2} \frac{\varepsilon}{\sigma}$

Now, a union bound over the n^4 possible swaps proves the Lemma. $\square$

Proof of Theorem:

Note first that since each edge weight ≤ 1, the initial tour is $\leq 2n$. So, if no ε-bad swaps, there will be $\leq \frac{2n}{\varepsilon}$ iterations.
The Lemma gives that
$$\underbrace{\mathbb{P}\{\text{more than } M \text{ iterations}\} \leq O\left(\frac{2n}{M} \frac{n^4}{\sigma}\right)}_{\text{w.h.p. polynomial}}$$
The expected # of iterations, since there must be $\leq n!$ possible tours, is
$$\mathbb{E}\{\#\text{iters}\} = \sum_{M=1}^{n!} \mathbb{P}\{\text{more than } M \text{ iters}\} \leq \sum_{M=1}^{n!} O\left(\frac{n^5}{\sigma}\right) \cdot \frac{1}{M}$$
$$= \underbrace{O\left(\frac{n^5}{\sigma}\right) n \log n}_{\text{poly in expectation}}$$
Both together give smoothed poly runtime.

$\square$