上海科技大学
ShanghaiTech University

CS240 Algorithm Design and Analysis

Lecture 23

Randomized Algorithms

Fall 2021

2021.12.08

Chernoff Bounds

Beating Expectations

- Given a random variable X, the expected value E[X] is the value we're "supposed" to get when we take a sample of X.
- We won't always get E[X].
- What is the probability the sample is very different from E[X]?
- Ex If we flip a fair coin 100 times, we expect to get 50 heads. What's the probability we get 60 heads? 80 heads? 100 heads?

Beating Expectations

- Markov's Inequality Given a positive random variable X, $\Pr[X \geq a] \leq E[X]/a$ for any $a > 0$.
- Ex X = no. heads in 100 flips. $\Pr[X \geq 60] \leq 50/60 = 5/6$.
- Proof Suppose $\Pr[X \geq a] > E[X]/a$. By definition, we have

$$E[X] = \sum_x x \cdot \Pr[X = x] \geq \sum_{x \geq a} x \cdot \Pr[X = x] \geq$$

$$a \cdot \Pr[X \geq a] > a \cdot E[X]/a = E[X], \text{ contradiction.}$$

- Markov's inequality is weak.
  - Using the previous example, it states $\Pr[X \geq 101] \leq 50/101 \approx 0.495$, which is quite obvious!

Beating Expectations

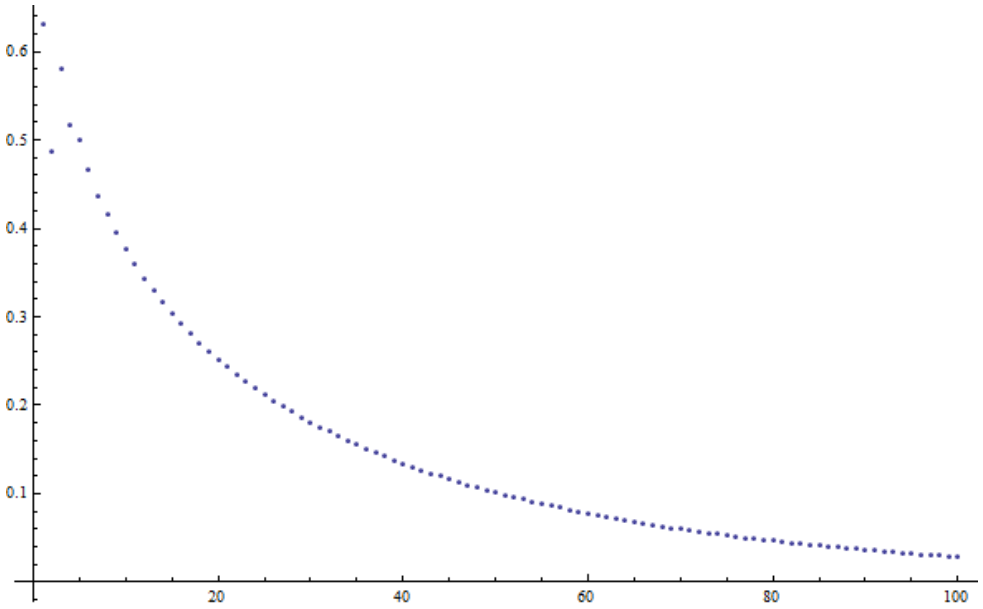
- But Markov's inequality is general.
  - X can be any positive random variable. It doesn't need to satisfy any special properties, as for some other inequalities.
  - Under some circumstances it can be used to prove stronger statements.
- Chebychev's Inequality Given a random variable X and any $a > 0$, we have $\Pr[|X - E[X]| \geq a] \leq Var[X]/a^2$.
- Proof $Var[X] = E[(X - E[X])^2]$, so by Markov's inequality,

$\Pr[|X - E[X]| \geq a] = \Pr[(X - E[X])^2 \geq a^2] \leq Var[X]/a^2$.

- Ex Let X=no. heads in 100 flips. Var[X]=100/4. So $\Pr[|X - 50| \geq 10] \leq 25/100 = 1/4$ by Chebychev.
  - Since X is symmetric about 50, $\Pr[X \geq 60] \leq 1/8$, which is much better than Markov's inequality.

Sum of independent random variables

- We'll consider a special kind of random variable X that is the sum of n independent random variables, for some n. We look at how likely X is to deviate from its expectation.
- Intuitively, as n gets larger, X should approach E[X], in relative terms, very quickly.
  - In fact, the convergence is exponential.
- Consider flipping a fair coin n times. What's the probability we get at least 60% heads?
  - For n=10, the probability is 37.7%.
  - For n=20, the probability is 25.2%.
  - For n=30, the probability is 18.1%.
  - For n=40, the probability is 13.4%.
  - For n=100, the probability is 2.84%.



Chernoff Bounds

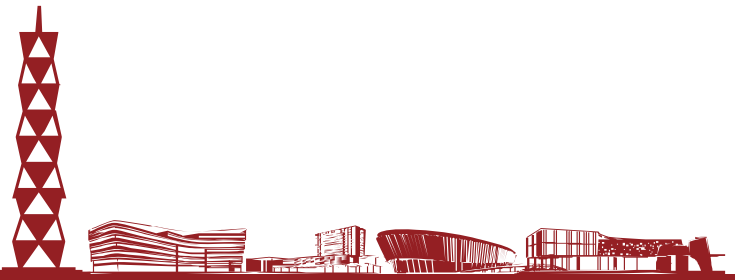
- Thm 1 Let $X_1,\ldots,X_n$ be independent random variables, s.t. $E[X_i] = p_i$ for all i. Let $X = \sum_i X_i$ and $\mu = E[X] = \sum_i p_i$. Then

  - For $0 < \delta \leq 1$, $\mathbf{Pr}[X \geq (1+\delta)\mu] \leq e^{-\mu\delta^2/3}$.
  - For $\delta > 1$, $\mathbf{Pr}[X \geq (1+\delta)\mu] \leq e^{-\mu\delta\ln\delta/3}$.
  - For $0 \leq \delta \leq 1$, $\mathbf{Pr}[X \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2}$.

- Chernoff bounds say the probability the sum of a set of independent random variables is more than $1 \pm \delta$ times its expectation m decreases exponentially in δ and μ.

Chernoff Bounds

- Chernoff bounds are very useful in the analysis of randomized algorithms.
  - Often the algorithm does things in independent stages.
  - It's often easy to compute the expected cost of each stage.
  - The total cost is the sum of the cost of all the stages.
  - We use Chernoff bounds to show this is unlikely to be too big or small compared to its expectation.
- There are some variants of Chernoff's bound that differ in the precise bounds. Some giver tighter bounds for certain values of δ. Pick the best one to use for the situation.



Chernoff Bounds

- Thm 2 Let X, $X_1,\ldots,X_n$ be defined as earlier. Then for any δ >0 we have

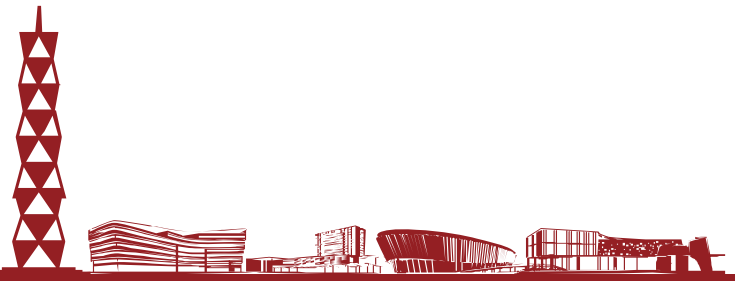
$$\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu}$$

- Thm 3 Let $X_1,\ldots,X_n$ be independent {-1,1} valued random variables, with $\Pr[X_i=1]= \Pr[X_i=-1]=1/2$ for all i. Let $X = \sum_i X_i$. Then for any $\delta \geq 0$, $\Pr[X \geq \delta] = \Pr[X \leq -\delta] \leq e^{-\frac{\delta^2}{2n}}$.

Load Balancing

- Suppose we have n computers. A set of equal sized jobs arrive online. We need to assign each to a computer for processing.
- To make all jobs finish fast, we want to give all computers (almost) same number of jobs, i.e., we want to balance their load.
- A simple way is to assign jobs round-robin. Keep giving next job to next computer, wrapping around if necessary.
- But this requires communicating with a centralized controller, which can be a bottleneck for large n.
- Instead, we do randomized load balancing.

❖ Algorithm Given a new job, assign it to a random computer.



Load Balancing

- How well does this balance the load?
- If there are m jobs, then in expectation every computer gets m/n jobs.
- Let X_i be the number of jobs computer i gets and let $X=\max_i \{X_i\}$ be max number of jobs any computer gets.
  - We'll bound probability that X_i or X are too large compared to the expectation m/n.
  - This shows the load is roughly balanced with high probability.
- Let Y_{ij} be a random variable that's 1 if the j'th job is assigned to computer i, and 0 otherwise.
  - $\Pr[Y_{ij}=1]=1/n$, since the jobs are assigned randomly.
  - $X_i = \sum_j Y_{ij}$ is the number of jobs computer i gets.
  - X_i is the sum of independent random variables Y_{ij}, so we can apply the Chernoff bound to X_i.

Analysis

- First consider the case $m = n$.
- We have $\mu = \mathbf{E}(X_i) = \mathbf{E}[\sum_{j=1}^{m} Y_{ij}] = \sum_{j=1}^{m} \mathbf{E}[Y_{ij}] = m\frac{1}{n} = 1$ for every i.
- We'll show $\mathbf{Pr}[X > O(\frac{\ln n}{\ln \ln n})] < \frac{1}{n}$.
  - So with high probability ($> 1 - 1/n$), every computer gets at most $O(\frac{\ln n}{\ln \ln n})$ times its expected number (=1) of jobs.
- Let's focus on X_i for some particular i. Let $\delta = O\left(\frac{\ln n}{\ln \ln n}\right) > 1$.
- By part 2 of Theorem 1, $\mathbf{Pr}[X_i > (1+\delta)\mu] \leq e^{-\delta \ln \delta/3}$, since $\mu = 1$.
  - So $e^{-\delta \ln \delta/3} = e^{O(\ln n)} \leq \frac{1}{n^2}$ for some choice of the constant in the big-O.
- We have $\mathbf{Pr}[X_i > O(\frac{\ln n}{\ln \ln n})] \leq \frac{1}{n^2}$ for every i. Hence by the union bound $\mathbf{Pr}[X_i > O(\frac{\ln n}{\ln \ln n})$ for any $i] \leq n\frac{1}{n^2} = \frac{1}{n}$.
- So $\mathbf{Pr}[X = \max_i X_i > O(\frac{\ln n}{\ln \ln n})] < \frac{1}{n}$.

Analysis

- We see that when there are n jobs for n computers, the load can be quite unbalanced.
- Suppose now we have more jobs, $m = 16n \ln n$. Then $\mu = 16 \ln n$.
- By Theorem 2, $\mathbf{Pr}[X_i > 2\mu] < \left(\frac{e}{4}\right)^{16 \ln n} < \left(\frac{1}{e^2}\right)^{\ln n} = \frac{1}{n^2}$.
- Thus, by the union bound $\mathbf{Pr}[\max_i X_i > 2\mu] \leq n \frac{1}{n^2} = \frac{1}{n}$.
- By part 3 of Theorem 1, $\mathbf{Pr}[X_i < \frac{1}{2}\mu] \leq e^{-(\frac{1}{2})^2 16 \ln n/2} = e^{-2 \ln n} = \frac{1}{n^2}$.
- So by the union bound, $\mathbf{Pr}[\min_i X_i < \frac{1}{2}\mu] \leq \frac{1}{n}$.
- Thus, we have that with high probability, every computer has between half and twice the average load.
- These results hold for any $m > 16n \ln n$. So the more jobs there are, the better the load balancing random allocation achieves. This is similar to the phenomenon where the more coins we flip, the more likely we are to get the expected number of heads (in relative terms).

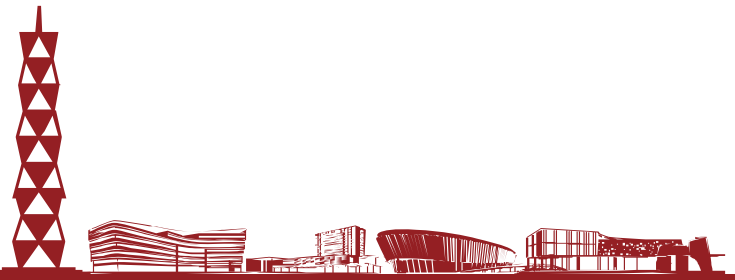
Set Balancing

- Suppose we have m sets, each a subset of {1,2,...,n}.
  - We can represent each set as an n-bit vector.
  - Ex If n=4 and S={1,3,4}, we can represent it as [1,0,1,1].
- We want to divide the sets into two groups, such that the sums of the sets in the groups are roughly equal.
- Ex S_1=[1,0,1,1], S_2=[1,1,1,0], S_3=[0,0,1,1], S_4=[0,1,1,0]. Then $S_1+S_4=[1,1,2,1]=S_2+S_3$.
- And exact balancing might not exist. We look for one that's as good as possible.
- Let G, G′ denote the two groups. We want to minimize the max imbalance $|\sum_{i\in G} S_i - \sum_{i\in G'} S_i|_\infty$.
  - Here $|v|_\infty$ denotes the L_∞ norm and is equal to the max component of v in absolute value. E.g. $|[1,2,-3,1]|_\infty = 3$.
  - Ex S_1=[1,1,0,1], S_2=[1,0,0,1], S_3=[1,0,1,0], S_4=[1,1,0,0].
  - There's no exact balancing, but S_1+S_3=[2,1,1,1] and S_2+S_4=[2,1,0,1], so the max imbalance is $|[2,1,1,1]-[2,1,0,1]|_\infty = |[0,0,1,0,]|_\infty = 1$.

Set Balancing

- Finding the grouping that minimizes the max imbalance is hard. Brute force would try 2^m possible groupings.
- We give a randomized algorithm that ensures the max imbalance is $\sqrt{4m \ln n}$ with high probability.
  - Max possible imbalance is m.

❖ Algorithm Assign each set to a random group.



Analysis

- Consider an item i. Suppose i belongs to k different sets.
- For the j'th such set S, define X_j=1 if S is in the first group, and X_j=-1 if it's in the other group.
- The imbalance from item i is simply $B_i = \sum_{j=1}^{k} X_j$.
- Since sets are assigned randomly, $X_1,\ldots,X_k$ are independent {1,-1} valued random values, we can apply Thm 3 to bound $\max_i B_i$.
- If $k \le \sqrt{4m\ ln\ n}, then\ B_i \le \sqrt{4m \ln n}$
- If $k > \sqrt{4m \ln n}$, then by Theorem 3 we have

$$\begin{aligned}\Pr[B_i > \sqrt{4m \ln n}] &\le e^{-\left(\sqrt{4m \ln n}\right)^2/(2k)} \\ &= e^{-4m \ln n/(2k)} \\ &\le e^{-4m \ln \frac{n}{2m}} \\ &= e^{-2 \ln n} \\ &= \frac{1}{n^2}\end{aligned}$$

- We showed the probability item i's balance is $\ge \sqrt{4m \ln n}$ is $\le \frac{1}{n^2}$. By the union bound, the probability that any of the n items' imbalance is $\ge \sqrt{4m \ln n}$ is $\frac{1}{n}$. So with high probability the max imbalance is $\le \sqrt{4m \ln n}$

Linear Programming

Linear Programming

- A farmer has the following problem.
  - He has 100 acres of land, on which he can plant wheat or barley or both.
  - He has 420 kg of fertilizer, and 160 kg of pesticide.
  - Each acre of barley requires 5 kg of fertilizer and 1 kg of pesticide.
  - Each acre of wheat requires 3 kg of fertilizer and 2 kg of pesticide.
  - Wheat sells for $3 per acre, barley sells for $4 per acre.
    - Actually, he could probably make $300 for wheat and $400 for barley. Choose $3 and $4 for simplicity.
- How many acres of wheat and barley should the farmer plant his field to maximize his income?
- Let w, b be acres of wheat and barley farmer plants.
- He wants to maximize 3w+4b, subject to the land, pesticide and fertilizer constraints.

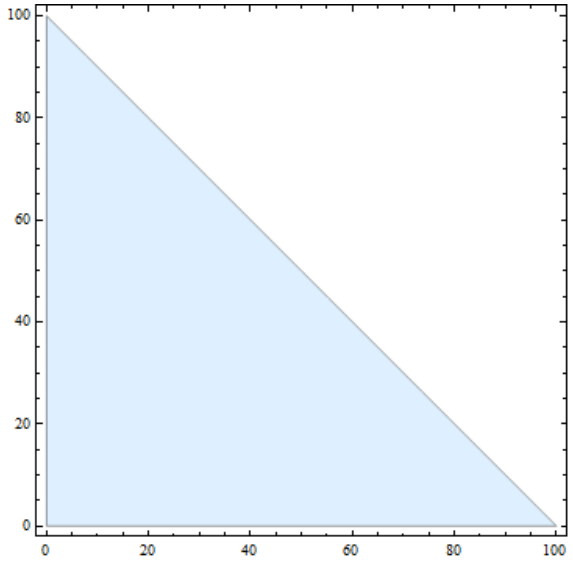
Linear Programming

- **maximize** 3w+4b **subject to**

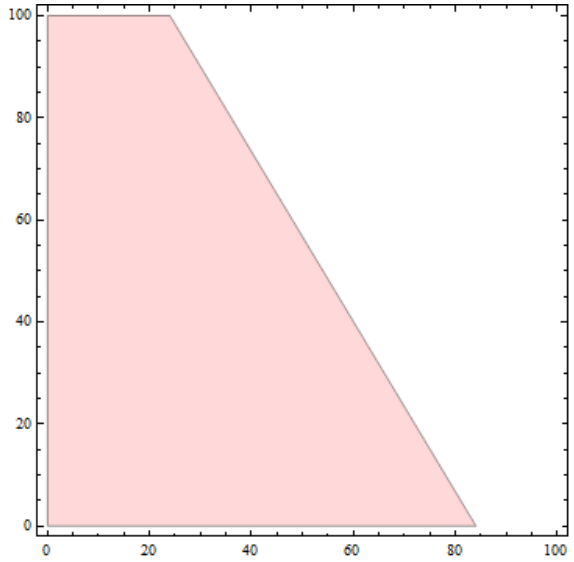
  w+b ≤ 100 (land)

  3w+5b ≤ 420 (fertilizer)

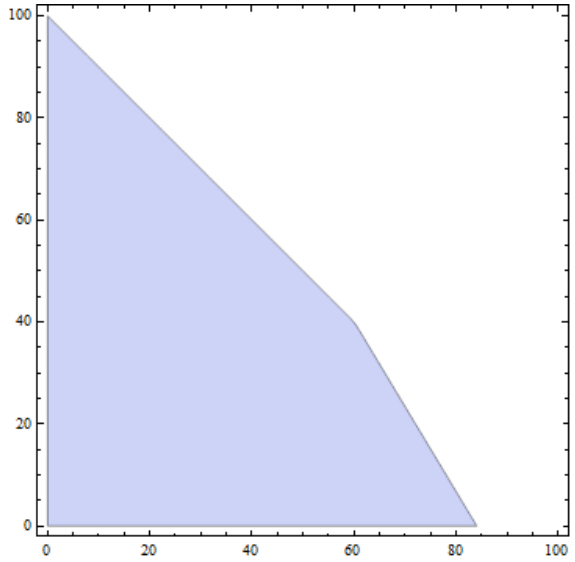
  2w+b ≤ 160 (pesticide)



land constraint



fertilizer constraint



land + fertilizer constraints

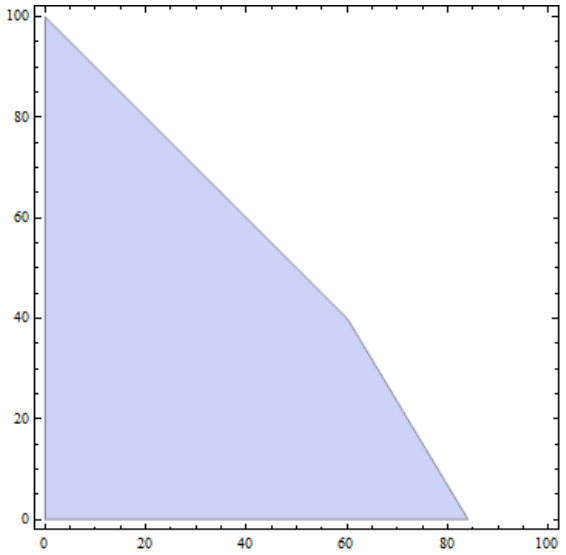
Linear Programming

- **maximize** 3w+4b **subject to**

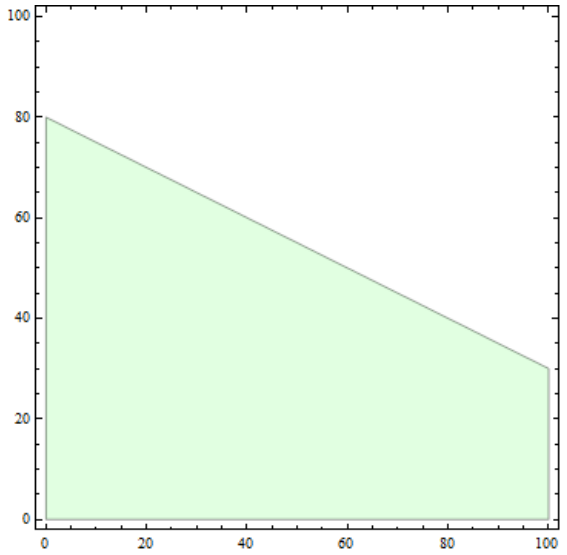
  w+b ≤ 100 (land)

  3w+5b ≤ 420 (fertilizer)

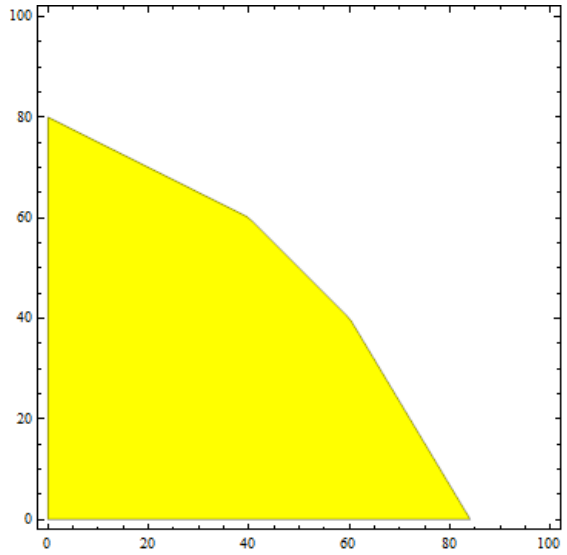
  2w+b ≤ 160 (pesticide)



land + fertilizer constraints



pesticide constraints



all three constraints

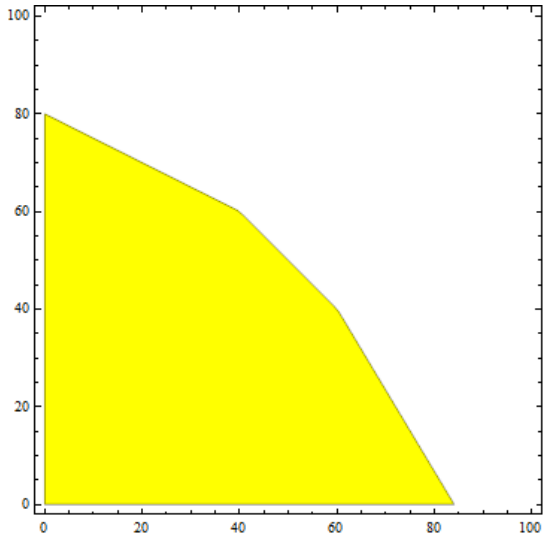
Linear Programming

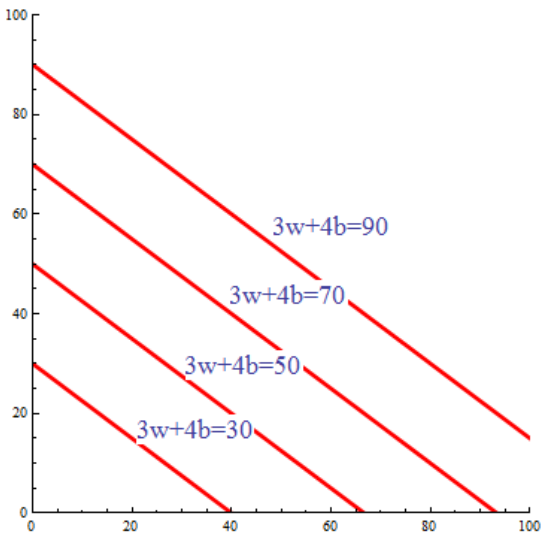
- **maximize** 3w+4b **subject to**

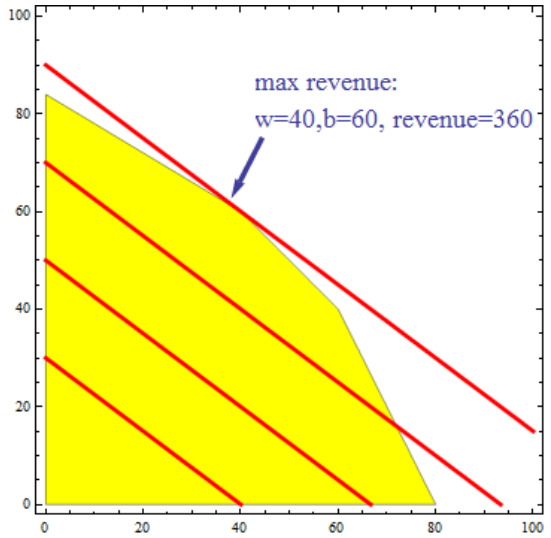
  w+b ≤ 100 (land)

  3w+5b ≤ 420 (fertilizer)

  2w+b ≤ 160 (pesticide)

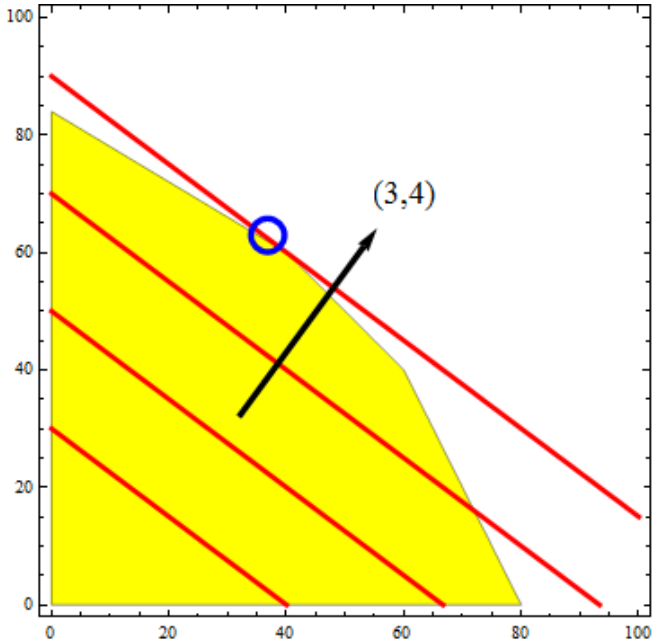






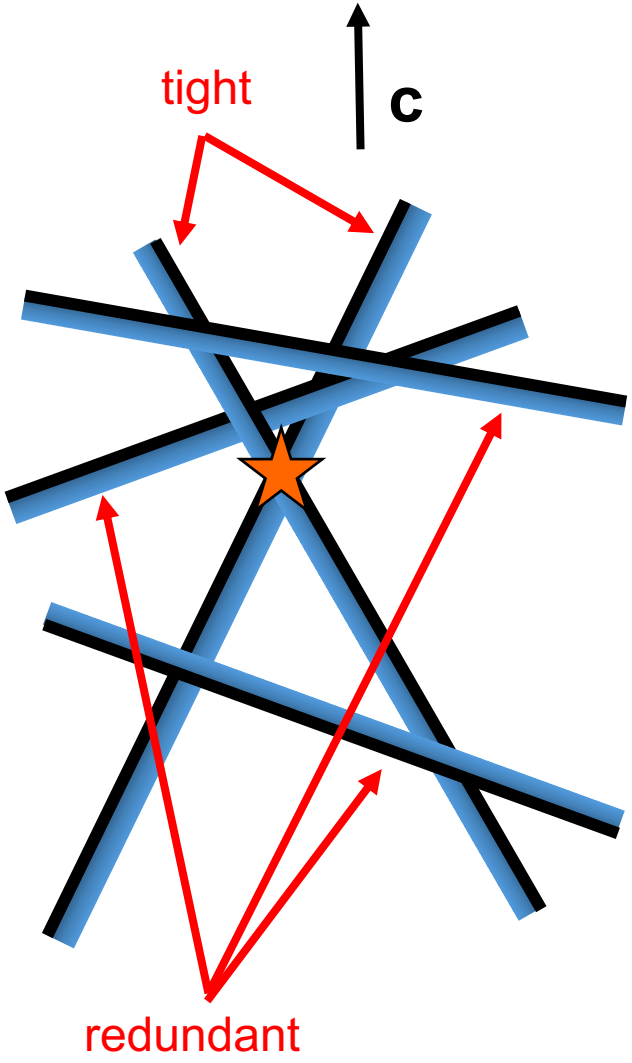
Linear Programming

- The feasible region is the area corresponding in which all the constraints are satisfied.
- Key Fact The optimum lies at an extreme point (corner).
- Find optimum by taking a line perpendicular to the direction pointed by the objective function, and shifting the line till when it will stop touching the feasible region.
- The optimum lies at the intersection of two constraints.
  - Call these the basis of the optimum.
  - For simplicity, assume constraints are general position, i.e. no 3 intersect at a point.



Randomized LP in 2D

- Since the optimum is defined by two constraints, the other constraints are redundant!
- A constraint is tight if the optimum lies on its defining line.
- Let H be set of n constraints. If pick random constraint, there's only 2/n probability it's tight.
- If constraint's not tight, we can discard it without changing optimum.
- How do we tell if it's tight?
  - For any constraint set G, let B(G) denote optimum.
- $h \in H$ is redundant iff $B(H)=B(H-\{h\})$.
  - i.e. the optimum is the same with or without h. So opt doesn't lie on h.



2D LP Algorithm

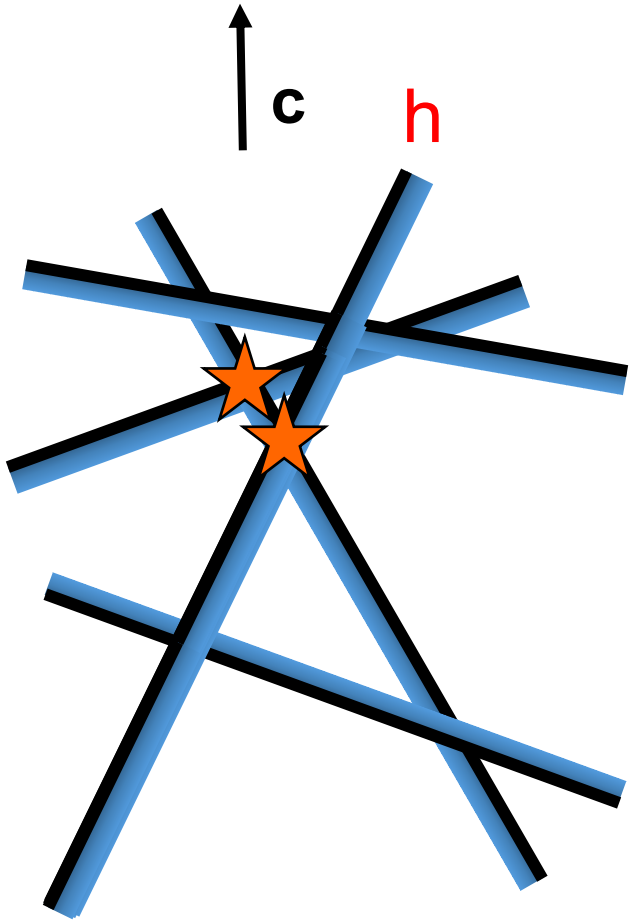
- If |H|=2, output intersection of the 2 halfplanes.
- Pick random constraint $h \in H$.
- Recursively find opt=B(H-{h}).
- If opt doesn't violate h, output opt.
  - opt violates h if opt lies outside h.
- Else project H-{h} onto h's boundary to obtain a 1D LP.
- Output the opt of the 1D LP.

上海科技大学 ShanghaiTech University

立志成才 报国裕民

Projection

- Given constraint h, let $\partial(h)$ be its boundary, i.e. the line defining h.
- Suppose B(H-{h}) violates h.
  - Then B(H) must lie on the boundary of h.
- Project a halfplane onto $\partial(h)$, reducing it to a line segment bounded on one or two sides.
- After projecting all H-{h} onto $\partial(h)$, we're left with a segment representing feasible region to 1D LP.
- Optimizing this is easy. The opt is one of the endpoints.



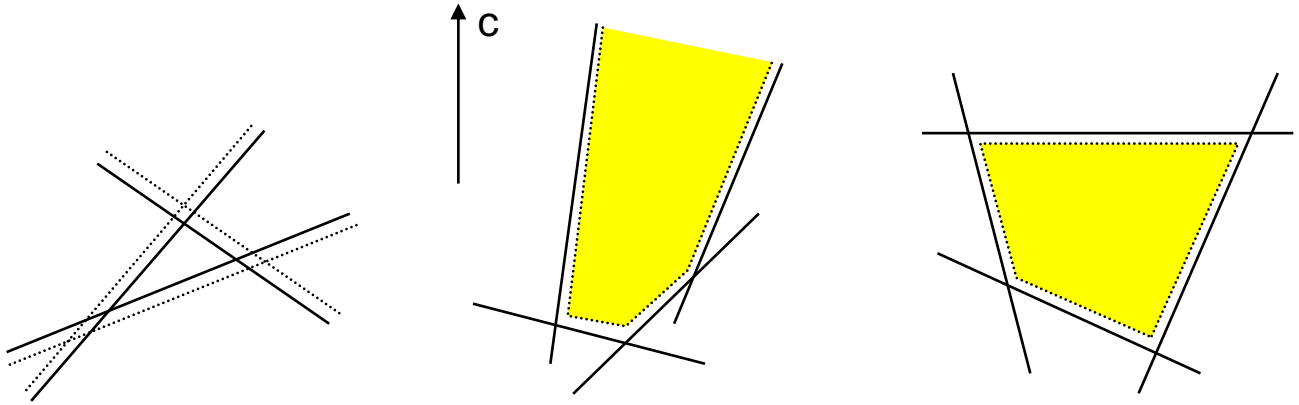
Analysis

- Let T(n) be expected time to solve 2D LP with n constraints.
- $T(n) \leq T(n-1)+O(1)+2/n(O(n)+O(1))$.
- T(n-1) time recursively find opt=B(H-{h}).
- First O(1) is time to check whether opt violates h.
- There's 2/n probability opt violates h, in which case we project all constraints onto $\partial(h)$.
  - O(n) to project H-{h} onto $\partial(h)$.
  - Final O(1) to solve 1D LP.
- T(n) solves to O(n).
  - So we can solve 2D LP with n constraints in expected linear time.

上海科技大学 ShanghaiTech University
立志成才 报国裕民

Corner Cases

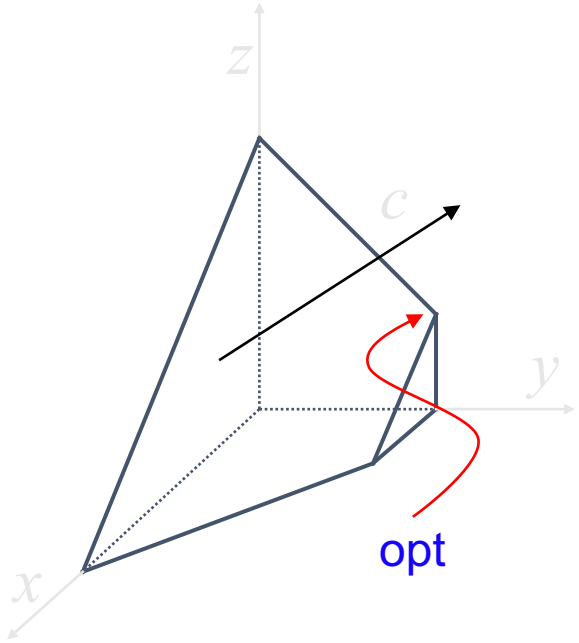
- For simplicity, we ignored several corner cases.
  - Infeasible means no points satisfy all the constraints.
    - Ex Constraints $x_1>1$ and $x_1<0$.
  - Unbounded means the optimum is infinite.
    - Ex Maximize x_2 s.t. $x_1+x_2>0$.
  - Non-unique optimum means an infinite number of points maximize the objective.
    - Ex Maximize x_2 s.t. $x_2 \leq 0$.
- Preprocess input to check for corner cases.

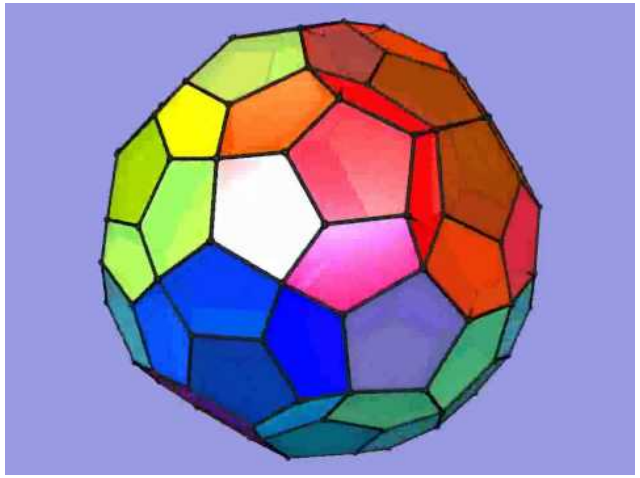


Infeasible　　Unbounded　　Non-unique optimum

Higher Dimensions

- In d>2 dimensions, lines become planes and each constraint corresponds to the space to one side of a plane, called a halfspace.
- The intersection of the halfspaces defines the feasible region.
  - This is a convex region called a polytope.
- Each extreme point (corner) of the polytope is the intersection of d halfspaces.
- The objective function defines a direction. Take a plane perpendicular to this direction and shift it till it stops touching feasible region.
- Hence optimum again lies at intersection of d halfspaces.
- Polytopes can be very complicated.





d-Dimensional LP Algorithm

- If |H|=d, output their intersection.
- Pick random constraint h∈H.
- Recursively find opt=B(H-{h}).
- If opt doesn't violate h, output opt.
- Else project H-{h} onto h's boundary to obtain a d-1 dimensional LP.
- Recursively solve the d-1 dim LP.

上海科技大学 ShanghaiTech University
立志成才 报国裕民

Analysis

- Let T(n,d) be expected time to solve d-dim LP with n constraints.
- $T(n,d) \leq T(n-1,d)+O(d)+d/n(O(dn)+T(n-1,d-1))$.
- T(n-1,d) time recursively find opt=B(H-{h}).
- O(d) time to check whether opt violates h.
- There's d/n probability opt violates h.
  - Because opt is defined by d of the n halfspaces.
  - In this case we project all constraints onto $\partial(h)$.
  - O(dn) to project H-{h} onto $\partial(h)$.
  - We obtain a d-1 dim LP with n-1 constraints.
  - T(n-1,d-1) time to solve this.
- T(n,d) solves to O(d! n)
  - Linear in number of constraints.
  - Exponential in dimensions.

Matrix Formulation

- **maximize** 3w+4b **subject to**

  w+b ≤ 100

  3w+5b ≤ 420

  2w+b ≤ 160

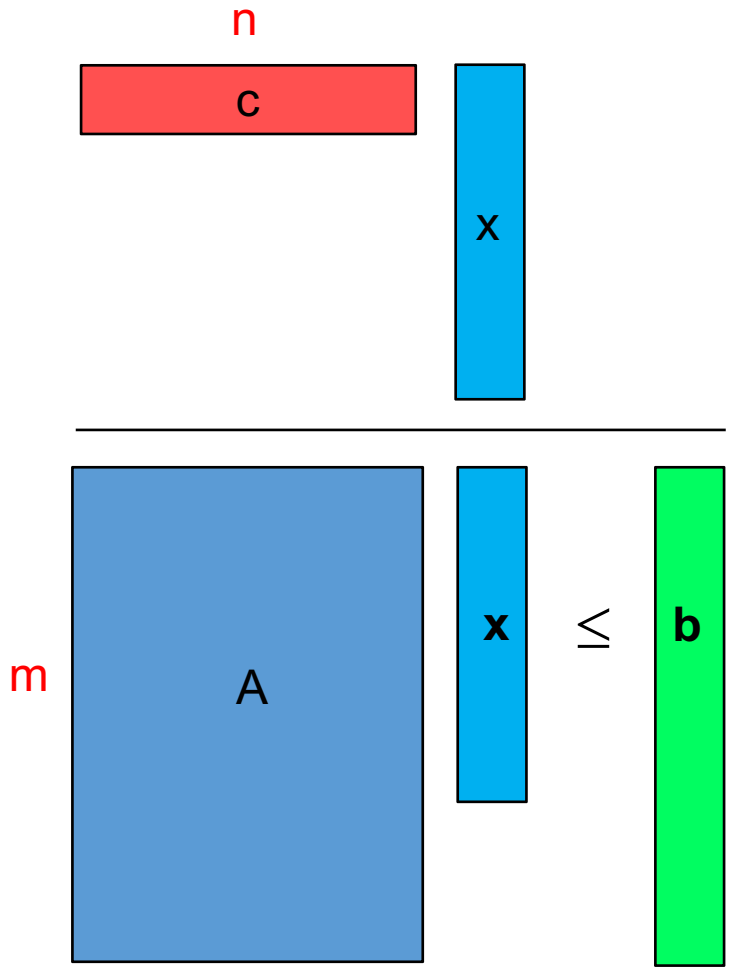
- $$\text{maximize}\begin{bmatrix}3,4\end{bmatrix}\cdot\begin{bmatrix}w\\b\end{bmatrix}\text{s.t.}$$

- $$\begin{bmatrix}1 & 1\\3 & 5\\2 & 1\end{bmatrix}\cdot\begin{bmatrix}w\\b\end{bmatrix}\le\begin{bmatrix}100\\420\\160\end{bmatrix}$$

  Let $x\in\Re^{n\times 1}, c\in\Re^{1\times n}, A\in\Re^{m\times n}, b\in\Re^{m\times 1}.$

  $\text{maximize}\, c\cdot x\ \text{s.t.}$

  $A\cdot x\le b$



Next Time:
Approximation algorithms