Linear Programming

A linear program is the problem of minimizing or maximizing a linear function subject to linear constraints.

Example:

variables

Maximize $x_1 + x_2$. } → Objective function

Subject to:

$$-x_1 + x_2 \leq 1$$
$$2x_1 - x_2 \leq 5$$
$$x_1 + 3x_2 \leq 9$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$

constraints.

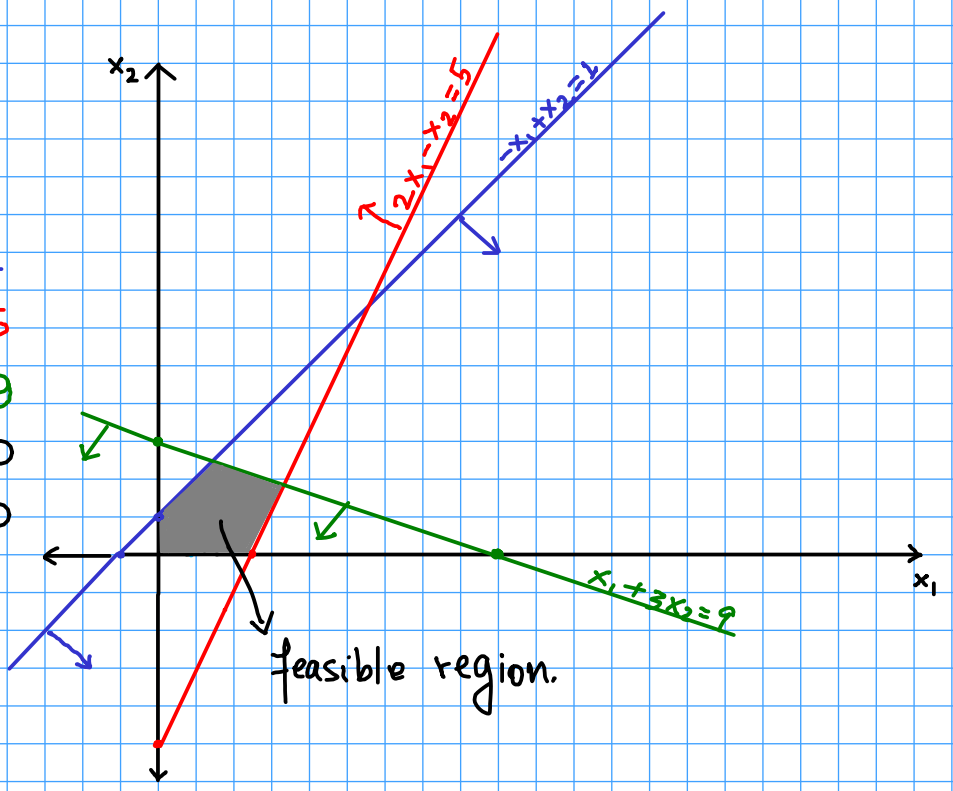
non-negativity constraints.

How does this problem look geometrically?

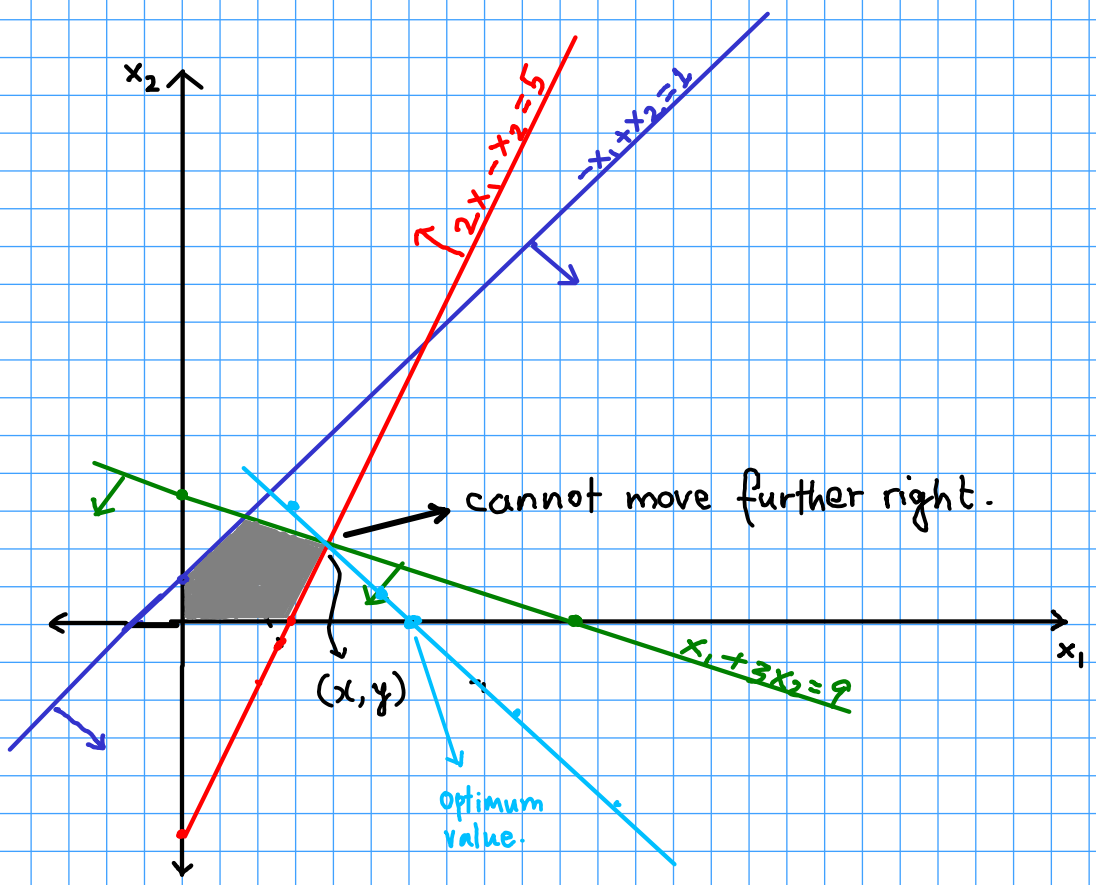
maximize $x_1 + x_2$

Subject to:

$$-x_1 + x_2 \leq 1$$
$$2x_1 - x_2 \leq 5$$
$$x_1 + 3x_2 \leq 9$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$



- The constraints define half-spaces.
- The feasible region is the intersection of the half-spaces.
- The intersection of half-spaces is a convex polyhedron. [we will see more about this soon]
- Any point inside the polygon shaded gray is called a feasible solution, since it satisfies all constraints.
- Different feasible points yield different values for the objective fn.
- For example, (0,0) and (2.5,0) are feasible. But, they yield objective fn. of value 0 and 2.5 respectively.
- We want a point in the feasible region or on the boundary that maximizes the value of the objective function.



- The value of the objective function at any point (a,b) inside the feasible region has value $a+b$ (since our obj. fn. is $x_1 + x_2$)
- Geometrically, this value is the intersection of the line $x_1 + x_2$ through the point (a,b) with the x_1-axis.
- So, we would like to shift the line $x_1 + x_2$ "as far to the right as possible" such that we still intersect the feasible region.
- The objective function cannot move beyond (x, y), for it would then be outside the feasible region.
- At this extreme point, the objective fn. is shifted by c, or equivalently, $x_1 + x_2 = c$ defines the line parallel to the objective fn.
- This gives us the optimal solution.

More generally, a linear program

- Consists of n variables, say $x_1, \ldots, x_n$.
  [or $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$]
- m linear constraints + n non-negativity constraints.
- A linear objective function that we want to maximize / minimize.

Ex:

$$\max. \quad \sum_{i=1}^{n} c_i x_i$$

subject to,

$$a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n \leq b_1$$
$$\vdots$$
$$a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n \leq b_m$$

$$x_i \geq 0, \quad i = 1 \ldots n.$$

In compact matrix notation:

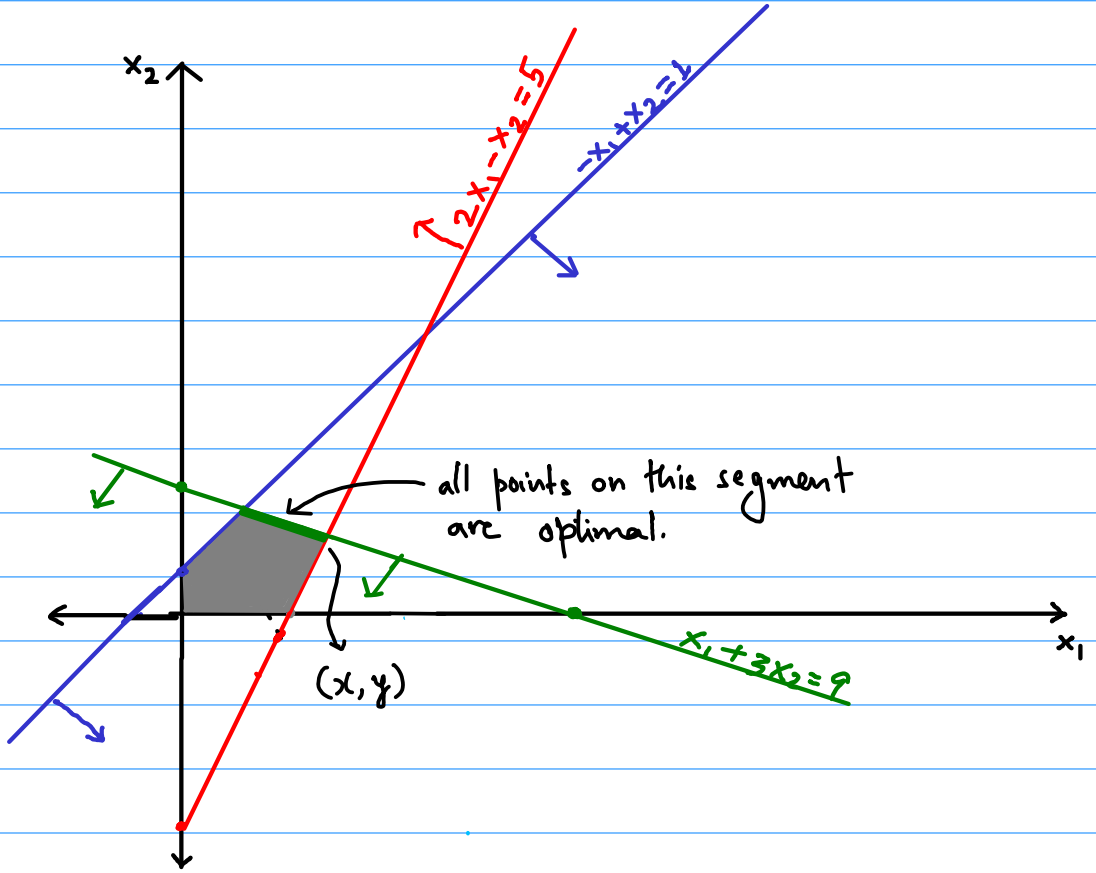
$$\max \quad c^T x$$
$$\text{s.t.}$$
$$Ax \leq b$$
$$x \geq 0$$

where:

$c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ & $x \in \mathbb{R}^n$.

- A linear program can have
  - No solution
  - A unique solution
  - Infinitely many solutions.

Eg:



- If we change the objective fn. to:
  $\max\ x_1 + 3x_2$. instead of $\max.\ x_1 + x_2$

- then the optimal solution is all points on the green segment shown.

On the other hand, if we modify the red constraint to:

$-x_1 + x_2 \leq 5$

& drop the green constraint, we get the following LP: →

maximize $x_1 + x_2$

Subject to:

$-x_1 + x_2 \leq 1$

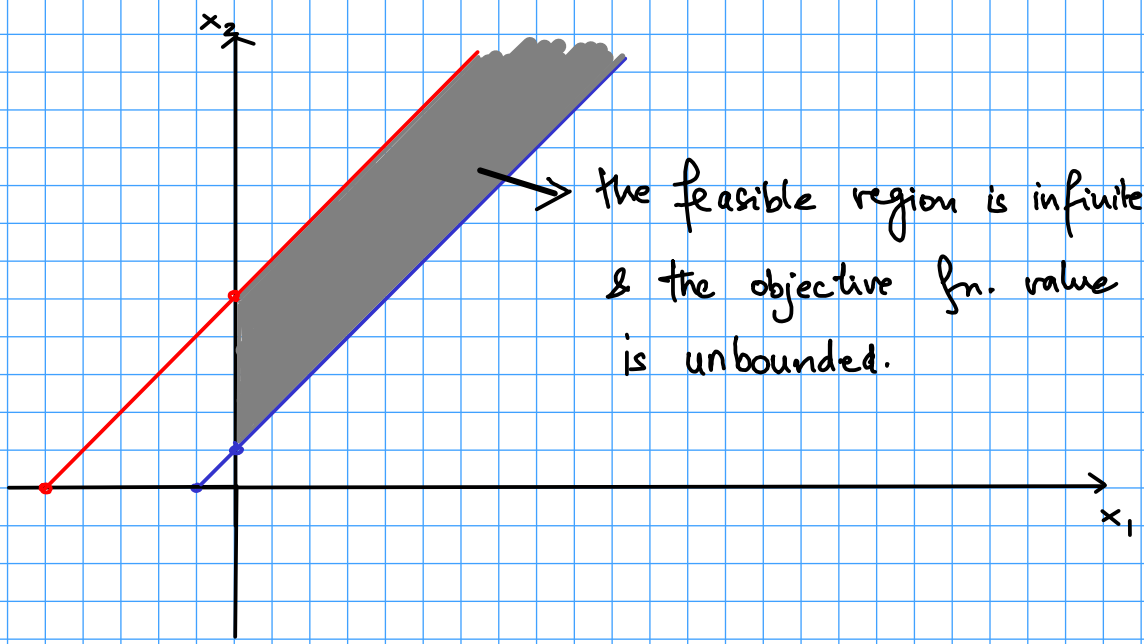
~~$-x_1 + x_2 \leq 5$~~

~~$2x_1 - x_2 \leq 5$~~

~~$x_1 + 3x_2 \leq 9$~~

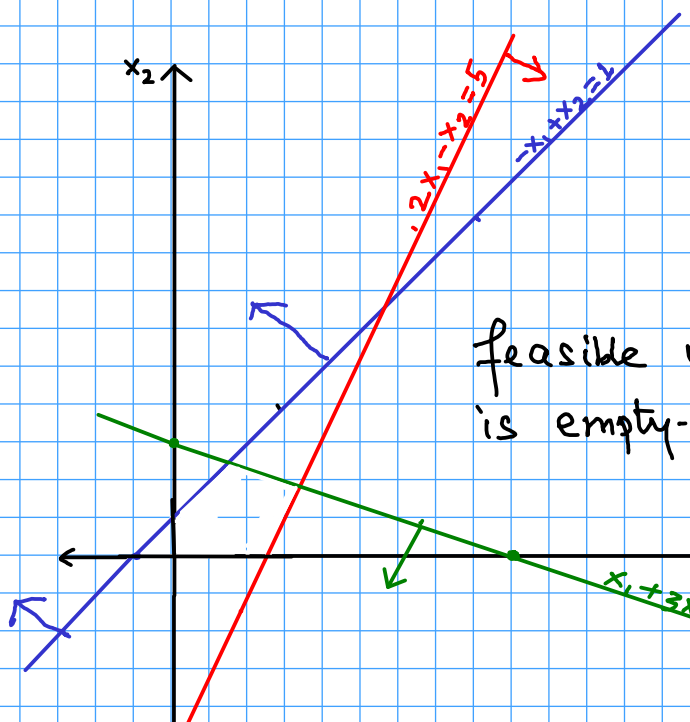
$x_1 \geq 0$

$x_2 \geq 0$



→ the feasible region is infinite.
& the objective fn. value is unbounded.

- Now, suppose we reverse the inequalities of our original LP:



feasible region is empty.

maximize $x_1 + x_2$

Subject to:

$-x_1 + x_2 \geq 1$

$2x_1 - x_2 \leq 5$

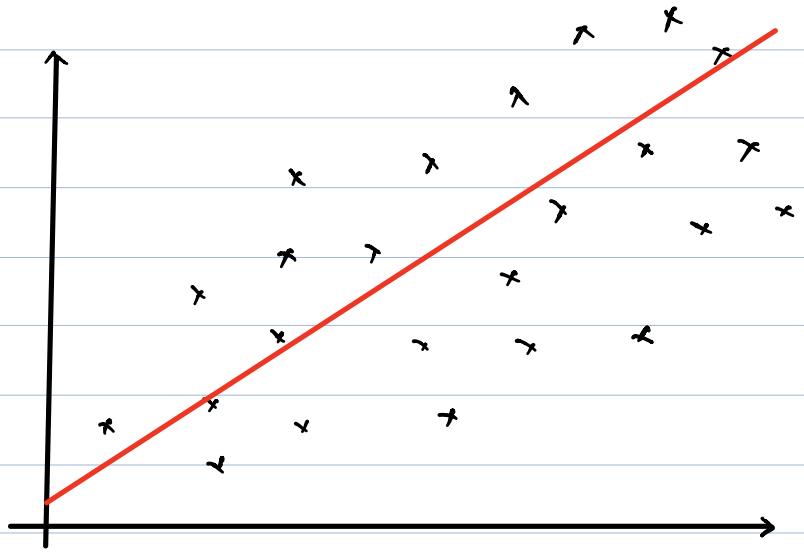
$x_1 + 3x_2 \leq 9$

$x_1 \geq 0$

$x_2 \geq 0$

Line fitting.

Given a set of observations, we want to fit a line that is a "best-fit" for the observations.



We look for a line $ax+b$, such that:

$$\sum_{i=1}^{n} (ax_i + b - y_i)^2 \quad \text{is minimized.}$$

ie; the avg. distance of the points to the line is minimized.

- This measure may not be always suitable, as avg. is sensitive to few large outliers

- Instead, we may want to minimize the absolute error:

$$\min \sum_{i=1}^{n} |ax_i + b - y_i|.$$

↳ But, this fn. is not linear.

- However, we can write an LP as follows:

- Set z_i be a variable.

$$\min \sum_{i=1}^{n} z_i$$

s.t.

$$z_i \geq ax_i + b - y_i \quad , i = 1..n$$

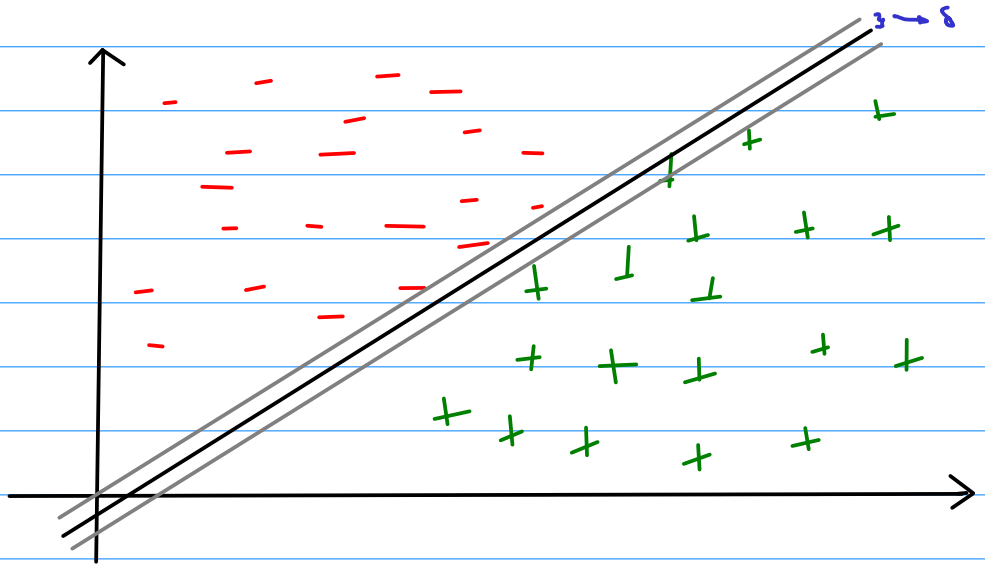
$$z_i \geq -(ax_i + b - y_i) \quad , i = 1..n$$

$z_i, i = 1..n$, a, b : variables.

Machine learning:

. A fundamental task in m/c learning is the following:

- Given +ve & -ve observations, find a line that separates the +ve & -ve examples.



- Let $p_1, \ldots, p_n$ be the points, $p_i = (x_i, y_i)$.
- Let $ax + b = y$ be the line we want to find.
- A point p_i lies above this line if:

$$y_i > ax_i + b$$

- A point p_i lies below this line if:

$$y_i < ax_i + b.$$

- So, we can write a system of inequalities:

$y_i > ax_i + b$ $\forall$ -ve examples

$y_i < ax_i + b$ $\forall$ +ve examples.

- What is the objective fn?
- We get rid of the strict inequalities by introducing a variable δ, which tells us how far the points are from our line:
- So, we get the following LP:

$$\max. \quad \delta$$

subject to,

$$y_i \geq a x_i + b + \delta \quad \forall \text{ -ve examples}$$
$$y_i \leq a x_i + b - \delta \quad \forall \text{ +ve examples.}$$

What if we want to seperate the points not by a line, but by a parabola, or a cubic curve etc.?

Can we write an LP?

Indeed the same idea works. The coeffs. of the polynomial are variables.

So, if we want to fit: $ax^2 + bx + c = y$:

We can write:

$$\max \quad \delta$$

$$y_i \geq a x_i^2 + b x_i + c + \delta \quad \forall \text{ -ve examples.}$$

$$y_i \leq a x_i^2 + b x_i + c - \delta \quad \forall \text{ +ve examples.}$$

Solving Integer Programs:

- Linear programs can be solved in polynomial time in :
  - Number of Variables
  - Number of Constraints
  - maximum entry in the constraint matrix / RHS.

- The first polynomial-time algorithm was the Ellipsoid Algorithm of Khachiyan '81.
  - This algorithm unfortunately is not practical.
  - However, in many situations, the ellipsoid algorithm can also solve LPs with exponentially many constraints in polynomial time!

- Karmarkar '84 proposed an interior-point algorithm, that also runs in polynomial time & is practical.

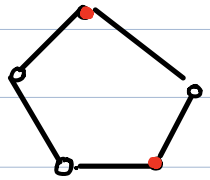
- One of the first algorithms to solve LPs is called the Simplex algorithm, proposed by Dantzig in the 1940's, all known implementations of which run in exponential time, but is still the most practical!

Integer Programs v/s Linear Programs

- Many problems we are interested in solving are "discrete optimization" problems., i.e. the variables take integer values & not rational values.

- Example: Let $G=(V,E)$ be a graph. An independent set is a set $S \subseteq V$, such that $\forall\, u,v \in S,\ \{u,v\} \notin E$.

  ie: S consists of a set of pairwise disjoint vertices.



- We want to compute an independent set of maximum size.

- Let us define an integer program for this problem:

$$x_u = \begin{cases} 1, & u \in S \\ 0, & \text{otherwise.} \end{cases}$$

$$\text{maximize} \sum_{u \in V} x_u$$

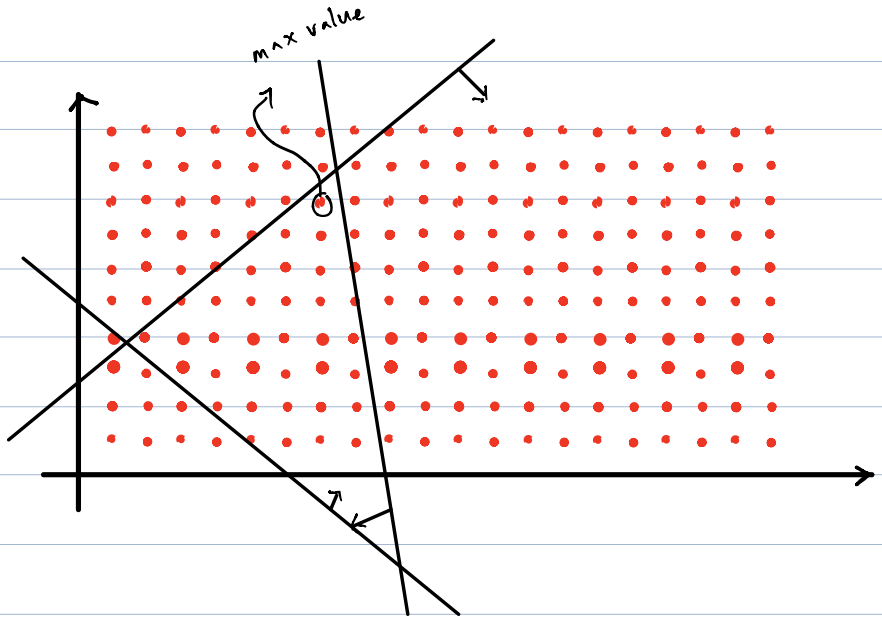
Subject to.

$$x_u + x_v \leq 1 \quad \forall\, \{u,v\} \in E$$

$$x_u \in \{0,1\} \quad \forall\, u \in V.$$ } integrality constraint.

- Integer (linear) programming is the problem of minimizing, or maximizing a linear function, subject to linear constraints, with the additional requirement that the variables take only integral values.

- Solving integer programs, and this integer program in particular is NP-hard.



$$\max c^T x$$

$$Ax \leq b$$

$$x \in \mathbb{Z}^n$$

Approximation algorithms

- Many discrete optimization problems of interest are NP-hard.
- One way of coping with the NP-hardness of these problems is to look for good approximate solutions that run in polynomial time.

- Formally, let Π be a maximization problem.
- An algorithm $\mathcal{A}$ that runs in polynomial time on any instance I of Π, and such that:

$$\text{Cost}(\mathcal{A}(I)) \geq \alpha \, OPT(I) \quad \forall I \in \Pi$$

is an α-approximation algorithm for Π.

ie: On all instances of the problem, the algorithm guarantees to produce a solution whose cost is at least an α-fraction of OPT (the optimal solution value of the problem).

- We can make a similar definition for minimization problems.

- Note that α need not be a constant, but a function of $|I|$, the input size.

Approximation Algorithms:

- Suppose we want to solve the maximum independent set problem.

- We can write an integer program:

$$\max \sum_{u \in V} x_u$$

s.t.

$$x_u + x_v \leq 1 \quad \forall \{u,v\} \in E$$

$$x_u \in \{0,1\}.$$

- Since we cannot solve this integer program in polynomial-time, we can instead solve its linear programming relaxation, ie:

$$\max \sum_{u \in V} x_u$$

s.t.

$$x_u + x_v \leq 1 \quad \forall \{u,v\} \in E$$

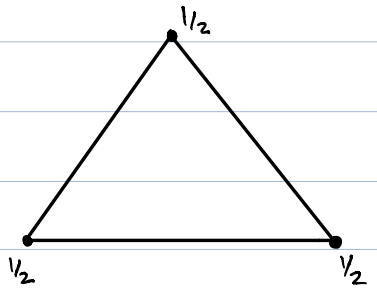
~~$x_u \in \{0,1\}$.~~ $x_u \in [0,1]$

- Now we solve this linear program in polynomial time.
- But, the optimal solution values may be fractional.

Let OPT_{IP} = an optimal solution of the integer program.

OPT_{LP} = an optimal solution of the relaxed linear program.

- Example: · Suppose G is a $\triangle$. The optimal independent set size = 1.



$OPT_{LP} = 3/2$. $OPT_{IP} = 1$.

- Observe that for any graph, $OPT_{IP} \leq OPT_{LP}$, since any feasible integer solution is feasible for the relaxed linear program.

- We could try to produce an integral solution A from OPT_{LP}, such that for some α

$$\text{Cost}(A) \geq \alpha\, OPT_{LP} \geq \alpha\, OPT_{IP}.$$

- There are several techniques to obtain approximation algorithms.

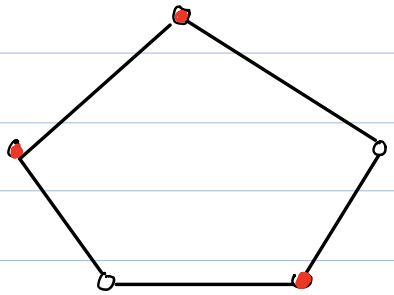
- For the particular case of computing a maximum independent set in a graph, we cannot obtain good approximation algorithms unless $P = NP$ (or other such conjectures are disproved).

Vertex Cover

Let $G = (V, E)$ be a graph. The vertex cover problem is to find a set $S \subseteq V$ of minimum cardinality so that $\forall \{u, v\} \in E$, either u, or v (or both) are in S.

i.e: Every edge is covered by (is adjacent to) a chosen vertex.

Example:



The red vertices form a vertex cover.

- The vertex cover problem is NP-hard.

- There are several approximation algorithms that yield a 2-approximation.

- Here we look at an algorithm based on LP-rounding:

Let $x_u = \begin{cases} 0, & u \notin S \\ 1, & u \in S. \end{cases}$

Integer Program:	Linear Program
$\min \sum_{u \in V} x_u$	$\min \sum_{u \in V} x_u$
$x_u + x_v \geq 1 \quad \forall \{u,v\} \in E$	$x_u + x_v \geq 1 \quad \forall \{u,v\} \in E$
$x_u \in \{0,1\}, \forall u \in V$	~~$x_u \in \{0,1\}, \forall u \in V$~~ $x_u \geq 0$

- Let x be an optimal solution returned by the linear program, and let $\sum_{v \in V} x_v = OPT_{LP}$.

- We round this potentially fractional solution to an integral solution as follows:

Obs:

- For each edge $\{u,v\} \in E$, either $x_u \geq 1/2$, or $x_v \geq 1/2$, since the constraint on each edge $\{u,v\}$ is: $x_u + x_v \geq 1$.

- Pick each v into S, whose x_v value is $\geq 1/2$. Then, we have atmost doubled each x_v value.

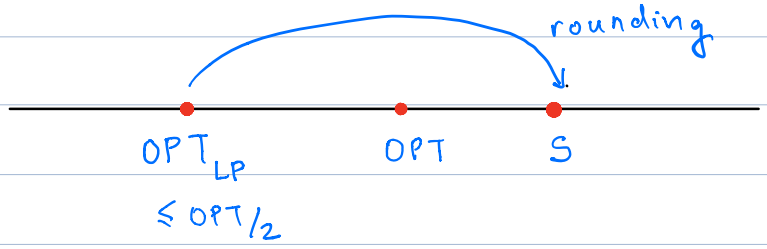
- Hence, $|S| \leq 2\,OPT_{LP} \leq 2\,OPT_{IP}$.

Questions: Given the simple analysis, we can ask if we can obtain a better 2-approximation via a cleverer analysis.

- If we can show an example where the ratio:

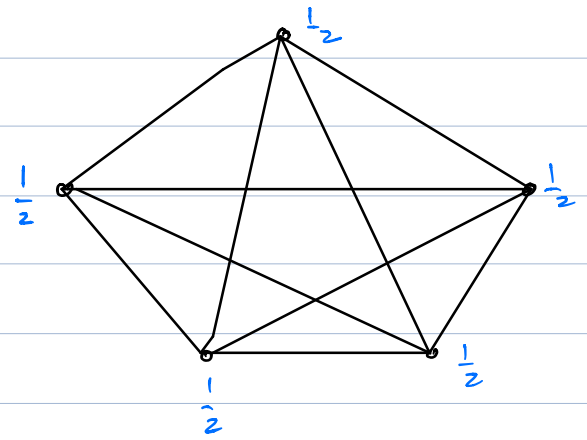
$$\frac{OPT_{LP}}{OPT} \leq \frac{1}{2}$$

- Then, this will show that our analysis is tight, ie: we will not be able to improve our analysis, as the following figure shows.



Here is an example:

- Consider the complete graph K_n.



- If the LP assigns a value $1/2$ to each vertex, then each of the constraints:

$$x_u + x_v \geq 1 \quad \forall \{u,v\} \in E$$

is satisfied.

- But, in a complete graph, there is an edge for every pair $\{u,v\} \in E$.

- Therefore, $OPT \geq n-1$.

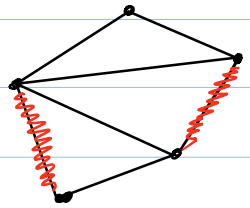
- Hence, $\frac{OPT_{LP}}{OPT} \leq \frac{n}{2(n-1)}$, which goes to 2 as $n \to \infty$.

□

A combinatorial algorithm:

- Since we can not improve the approximation factor via LP-rounding, may be we can try other algorithms?

- For a graph $G=(V,E)$, a set $M \subseteq E$ is a matching, if each vertex in V is incident to at most one edge of M.

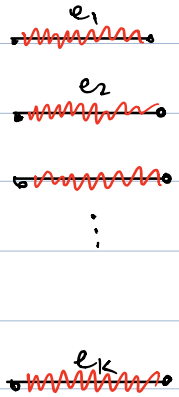
Eg: 

M

A matching, is maximal if we can not add any more edges to M without violating the matching condition.

The matching in the example above is a maximal matching.

- Suppose G has a maximal matching of size k:



Lemma: $OPT \geq k$.

ie: An optimal vertex cover has size $\geq k$.

Pf: Each edge in M requires a separate vertex to cover it $\blacksquare$

· This suggests the following algorithm:

1. Pick a maximal matching M
2. Choose both end-points of each edge in M into the solution S.

Theorem: $|S| \leq 2 \cdot OPT$.

Proof: $|S| = 2|M| \leq 2 \cdot OPT$ [By Lemma].

How do we select a maximal matching? A greedy algorithm will do:

1. Set $S = \phi$; $M = \phi$
2. While $\exists \{u,v\} \in E$ s.t. neither u nor v is in S,
3. . Add $\{u,v\}$ to M; u, v to S.
4. Return M, S.

At the end of the algorithm, M is a maximal matching, & $|S| = 2|M|$.

Lower bound:

- The combinatorial algorithm above is simpler than the LP-rounding algorithm.
- However, we still only managed a 2-approximation.
- Could there exist cleverer algorithms that yield better approximation factors?
- The algorithms above are from the 1970's. Surely, we could have better algorithms by now?
- Turns out, the answer is No! In-fact, Dinur & Safra proved in 2005 that one cannot obtain a better than 1.36-factor unless P = NP.
- Khot & Regev, '2008 proved that unless the Unique Games Conjecture is true, we cannot obtain a better than 2-approximation for vertex cover!

Set Cover

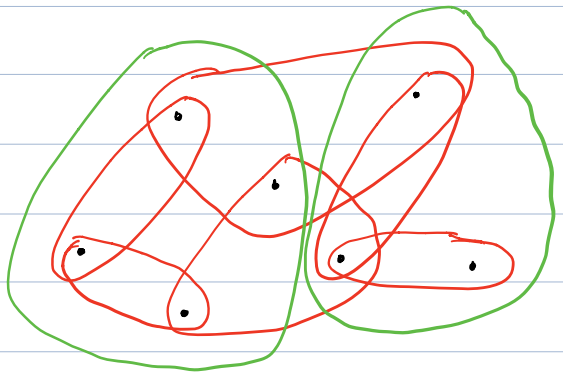
- Given a finite ground set X, $|X| = n$, and a collection of subsets $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$, $R_i \subseteq X$, such that $\bigcup_{i=1}^{m} R_i = X$

- Find a collection $\mathcal{S} \subseteq \mathcal{R}$, such that $\bigcup_{i \in \mathcal{S}} R_i = X$.

- ie: the sets in $\mathcal{S}$ also cover X.

Example:

OPT = the 2 green sets.



- The set-Cover problem is NP-hard.

Integer Program:	LP-relaxation
Let $x_i = \begin{cases} 1, & R_i \subseteq \mathcal{Q} \\ 0, & \text{otherwise.} \end{cases}$	
minimize $\sum_{R_i \in R} x_i$	minimize $\sum_{R_i \in R} x_i$
$\sum_{R_i : p \in R_i} x_i \geq 1 \quad \forall p \in X$	$\sum_{R_i : p \in R_i} x \geq 1 \quad \forall p \in X$
$x_i \in \{0,1\}$.	~~$x_i \in \{0,1\}$.~~
	$x_i \geq 0$

- Let $x = (x_1, \ldots, x_m)$ be an optimal solution to the linear program.

- $OPT_{LP} \leq OPT_{IP}$

- We will obtain an integer solution $y = (y_1 \ldots y_m)$, such that

$$\sum y_i \geq \alpha \sum x_i = \alpha \, OPT_{LP} \geq \alpha \cdot OPT_{IP}.$$

- Note that $x_i \in [0,1]$. Interpret each x_i as the probability of picking the set R_i.

- Construct a set $\mathcal{S}_0$ by picking each set R_i with probability x_i, independently of the other sets.

- $\mathbb{E}[|\mathcal{S}_0|] = \sum_{i=1}^{m} \mathbb{P}[R_i \text{ is chosen}] = \sum x_i = OPT_{LP}$ ($\because$ linearity of expectation)

- Let $p \in X$ be a point.

$$\mathbb{P}[p \text{ is not covered by } \mathcal{S}_0] = \prod_{R_i : p \in R_i} (1 - x_i)$$

$$\leq \prod e^{-x_i} \quad [\because e^y \geq 1+y \ \forall y \in \mathbb{R}]$$

$$= \exp\left(-\sum_{R_i : p \in R_i} x_i\right).$$

$$\leq 1/e \quad [\because \sum x_i \geq 1]$$

- The expected number of elements not covered by $\mathcal{S}_0$ is:

$$\sum_{p \in X} \mathbb{P}r[p \text{ is not covered by } \mathcal{S}_0] \leq n/e.$$

[by linearity of expectation]

- Let $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_t$ be the sets chosen in t rounds of the above procedure, using fresh random bits in each round.
- Let $\mathcal{S} = \bigcup_{i=1}^{t} \mathcal{S}_i$

- $\mathbb{P}[p \text{ is not covered in } \bigcup_{i=1}^{t} \mathcal{S}_i] = \prod_{i=1}^{t} \mathbb{P}[p \text{ is not covered by } \mathcal{S}_i]$

$$\leq \left(\frac{1}{e}\right)^t$$

- Let $t = 2 \ln n$, then.

$$\mathbb{P}[p \text{ is not covered in } \bigcup_{i=1}^{t} \mathcal{S}_i] \leq \frac{1}{n^2} \quad \text{---(1)}$$

Union Bound:

What is the probability that there is an element $p \in X$ that is uncovered?

$$\mathbb{P}[E_1 \cup E_2 \cup \cdots \cup E_k] \leq \sum_{i=1}^{k} \mathbb{P}[E_i] \quad \text{[Union bound]}$$

$$\mathbb{P}[\exists\, p \in X \text{ that is uncovered in } \mathcal{S}] \leq \sum_{p \in X} \mathbb{P}_r[p \text{ is uncovered in } \mathcal{S}]$$

$$\leq \frac{n}{n^2} = \frac{1}{n} \quad \text{(from (1))}$$

$$\mathbb{E}[|\mathcal{S}|] \leq \sum_{i=1}^{t} \mathbb{E}[|\mathcal{S}_i|] \leq 2 \log n \, OPT_{LP} \leq 2 \log n \, OPT_{IP}.$$

(randomized)

Hence, we have an $O(\log n)$-approximation algorithm for Set Cover.

- There are other algorithms that yield an $O(\log n)$-approximation

- However, there is no $(1-\epsilon)\log n$ approximation algorithm for any $\epsilon > 0$, unless $P = NP$.

- Therefore, under standard assumptions, this is the best approximation bound we can hope for.

- Can we improve the approximation ratio in certain geometric settings?

A Greedy algorithm for Set Cover

- Like the algorithm for Vertex Cover, we can obtain a combinatorial algorithm for Set Cover.
- This algorithm is a natural greedy algorithm.