1)

Optimization examples:

(Diet & Cost balance → slides 'for question)

1) a) ~~[illegible]~~

$S_1 + S_2$ = Total cost, $(S_1, S_2 \geq 0)$

$\frac{S_1}{5} \times 60 + \frac{S_2}{10} \times 20 \rightarrow \geq 300$ = Carbs, Carbs $\geq$ 300gm constraint

$\frac{S_1}{5} \times 20 + \frac{S_2}{10} \times 80 \geq 800$

$3000 \geq \frac{S_1}{5} \times 100 + \frac{S_2}{10} \times 60 \geq 2000$

$\downarrow$

$-\left(\frac{S_1}{5} \times 100 + \frac{S_2}{10} \times 60\right) \geq -3000$

b) If we fix packets, → Minimize

$S_1 \times 5 + S_2 \times 10$ = total cost, $(S_1, S_2 \geq 0)$

$S_1 \times 60 + S_2 \times 20 \geq 300$

$S_1 \times 20 + S_2 \times 80 \geq 800$

	Prod 1	Prod 2	Requirements
Carbs	60	20	300
Prot	20	80	800
Energy	100	60	[2000, 3000]
Price	5Rs	10Rs	→

———— x ————

- Hyperplane $c^T x = b$ } For $x \geq 3$ length
  $x = 3 \rightarrow$ Plane

———— x ————

- Discrete variables → Discrete optimization
* Continuous variables → Continuous optimization

- Constrained & Unconstrained optimization.
- Linear vs NonLinear optimization.

———— x ————

*Linear Programming:

- Standard form → $\max_x c^T x$
such that
$Ax \leq b$
$x \geq 0$

- Equational form → $\max_x c^T x$
such that
$Ax = b$
$x \geq 0$

⇒ $A_{m \times n}$ matrix, m = ~~#Constraints~~, n = # variables

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & & \vdots \\ a_{m1} & \cdots & & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

(Problems like $Ax < b$ are hard to solve with LP)

⇒ Converting from standard to equational form is done by adding additional variab

———— x ————

Note: $Ax \geq b \Rightarrow -Ax \leq -b$ } Standard form

- $\min c^T x \approx \max -c^T x$

⇒ Solution ⇒ Assignment of values to x.

* Feasible solution ⇒ Assignment that satisfies given constraints.

Feasible region ⇒ Set of all feasible solutions.

Optimal solution ⇒ The feasible solution that maximizes / minimizes our objective.

---

* Problem 1

1) • max $c^T x$
given constraints

Problem 2 ($\alpha > 0$)

• max $\alpha c^T x$
given same constraints.

⇒ Let x^* be an optimal solution for problem 1

$x^* \in F(\text{Prob 1})$ & $c^T x^* \geq c^T x \ \forall x \in F(\text{Prob 1})$

⇒ Therefore x^* will also be a feasible solution for Problem 2, and $F(\text{Prob 1}) = F(\text{Prob 2})$.

$\Rightarrow c^T x^* \geq c^T x \ \forall x \in F(\text{Prob 1})$

$\Rightarrow \alpha c^T x^* \geq \alpha c^T x \ \forall x \in F(\text{Prob 2})$

$\therefore x^*$ is an optimal solution for problem 2.

---

2) Problem 1

max $c_1^T x + c_2^T x$
s.t $Ax \leq b$
$x \geq 0$

Problem 2

max $c_1^T x$
s.t $Ax \leq b$
$x \geq 0$

Problem 3

max $c_2^T x$
s.t $Ax \leq b$
$x \geq 0$

* ⇒ Is optimal(prob1) value = optimal(prob2) + optimal(prob3)

- No, since optimizing P2 & P3 seperately the x's could be different, so

max $c_1^T x$ + max $c_2^T x \geq$ max $c_1^T x + c_2^T x$

example: $\left(\frac{3}{2}x_1 - \frac{1}{2}x_2\right) + \left(\frac{-1}{2}x_1 + \frac{3}{2}x_2\right)$

$(x_1, x_2 \geq 0)$
$(x_1, x_2 \leq 1)$

$\frac{3}{2}$ } When we maximize

$\frac{3}{2}$ } When we maximize

$\Rightarrow \frac{3}{2}x_1 - \frac{1}{2}x_2 - \frac{1}{2}x_1 + \frac{3}{2}x_2$

$= x_1 + x_2 = 2$ } On maximizing

$\therefore \frac{3}{2} + \frac{3}{2} \neq 2$

Note: In linear optimization, optimal solutions always exists at atleast one of the corners of the feasible region. (They can exist elsewhere, but searching the corners guarantees that we will find a solution).

Note: Fitting a line for a given set of points
* is quadratic if we consider square distances etc but is a linear optimization problem if we consider absolute constraints.

Linear { $e_i \geq |ax_i + by_i - b|$ (Trick to convert into linear) } $\Rightarrow$ $te_i \geq ax_i + by_i - b$, $e_i \geq -ax_i - by_i + b$

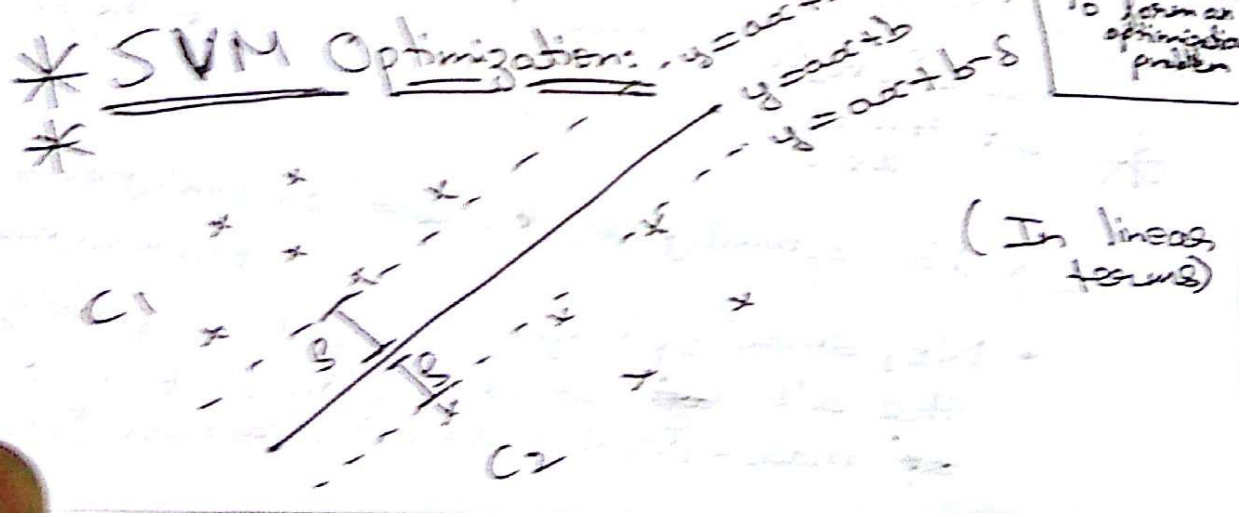
$\min \sum e_i$

Total error is $\geq$ each individual error $\geq$ 0 for all data

---

To form an optimization problem

* SVM Optimization:



(In linear terms)

$\Rightarrow$ Maximize $\delta \Rightarrow 0 \cdot a + 0 \cdot b + 1 \cdot \delta$

s.t

Class 1 $\begin{cases} x_1 a + b + \delta \le y_1 \\ \vdots \\ x_{n_1} a + b + \delta \le y_{n_1} \end{cases}$, a, b can be anything $\downarrow$ We need to transform this to standard form.

Class 2 $\begin{cases} -x_{(n_1+1)} a - b + \delta \le -y_{(n_1+1)} \\ \vdots \\ -x_{n_2} a - b + \delta \le -y_{n_2} \end{cases}$, $\delta \ge 0$

$\Rightarrow \max_{a_1, a_2, b_1, b_2, \delta} 0 \cdot a_1 + 0 \cdot a_2 + 0 \cdot b_1 + 0 \cdot b_2 + 1 \cdot \delta$

Class 1 $\begin{cases} x_1 a_1 - x_2 a_2 + b_1 - b_2 + \delta \le y_1 \\ \vdots \\ x_{n_1} a_1 - x_{n_1} a_2 + b_1 - b_2 + \delta \le y_{n_1} \end{cases}$

Class 2 $\begin{cases} -x_{n_1+1} a_1 + x_{n_1+1} a_2 - b_1 + b_2 + \delta \le -y_{n_1+1} \\ \vdots \\ -(x_{n_2}) a_1 + x_{n_2} a_2 - b_1 + b_2 + \delta \le -y_{n_2} \end{cases}$

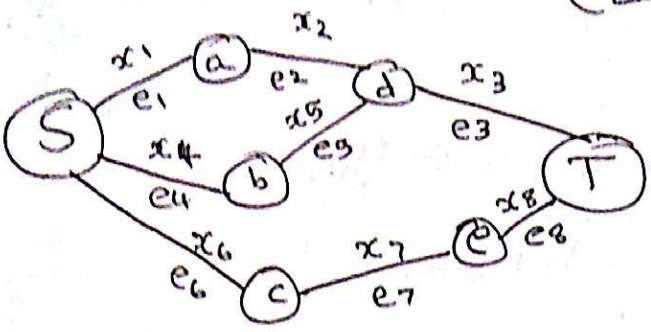
$a_1, a_2, b_1, b_2, \delta \ge 0$ } Standard form

(Here $a = a_1 - a_2$, $b = b_1 - b_2$, $a_1, a_2, b_1, b_2 \ge 0$, in this manner we have standard form constraints whereas earlier a, b had no constraints)

(We can generate any a, b from $a_1 - a_2$ & $b_1 - b_2$ keeping $a_1, a_2, b_1, b_2 \ge 0$)

Note: $\ni \Rightarrow$ S.t $\Rightarrow$ such that

*Graph Optimization: (Flows) (Linear Optimization)



- e_i = Capacity of edge
- x_i = Flow on edge to the right (-ve if to the left)

- $\max x_3 + x_8$ or $\max x_1 + x_4 + x_6$ (Optimization function)

$\Rightarrow -e_i \leq x_i \leq e_i$

$\Rightarrow x_1 = x_2$

$x_4 = x_5$

$x_6 = x_7$

$x_2 + x_5 = x_3$

$x_7 = x_8$

Converting to standard form,

→ Constraints

- $x_1 - x_2 \leq 0$, $x_1 \leq e_1$ $\left(a=b \Rightarrow a \leq b \text{ & } b \leq a\right)$

$x_2 - x_1 \leq 0$ $-x_1 \leq e_1$,

$x_3 - x_2 - x_5 \leq 0$ $x_2 \leq e_2$

$x_2 + x_5 - x_3 \leq 0$ $-x_2 \leq e_2$

$x_4 - x_5 \leq 0$ ⋮

$x_5 - x_4 \leq 0$ ⋮

$x_6 - x_7 \leq 0$ $x_8 \leq e_8$

$x_7 - x_6 \leq 0$ $-x_8 \leq e_8$

$x_7 - x_8 \leq 0$

$x_8 - x_7 \leq 0$

$(Ax \leq b)$ ↑ Constraint matrix form.

(16+10 = 26 constraints, Variables = 8)

→ Optimize $c^T x$, $c^T = [1,0,0,1,0,1,0,0]$

$\Rightarrow$ But this is also not in standard form since $x_i \neq 0 \ \forall i$

$\therefore$ Replace x_i with $x_{i_1} - x_{i_2}$ where $x_{i_1}, x_{i_2} \geq 0$.

---

*Graph Matching: (Integer Optimization)

- Given a graph with $|E|$ edges, find $M \subseteq E$ such that no two edges share a vertex.

ex:

$\Rightarrow$ Define $x_e = 1$ if $e \in m$.
$= 0$ otherwise

- Optimize: $\max \sum_{e \in E} x_e$
- Constraints: $\sum_{e : e \text{ is incident on vertex } v} x_e \leq 1 \quad \forall \ v \in V$

$x_e \in \{0, 1\}$

Note: Replacing $x_e \in \{0,1\}$ with $0 \leq x_e \leq 1$, it is called the LP relaxation of integer program (Need not give optimal solution)

---

*Vertex Cover:

- Select minimum number of vertices such that every edge has atleast one selected vertex as one of its end points.

(Select $C \in V$), Minimize $|C|$

$\Rightarrow x_v = 1$ if $v \in C$

$= 0$ otherwise

$\Rightarrow \min \sum_{v \in V} x_v$ } Optimize

$x_u + x_v \geq 1 \quad \forall \underbrace{(u,v)}_{\text{end points of edge}} \in E$

$x_v \in \{0,1\}$

(Integer optimization problem)

(we can use LP relaxation here as well)

*LP Relaxation 1:

** • Consider we select a vertex if $x_v \geq 0.5$

$0 \leq x_v \leq 1$ } LP relaxation

$\Rightarrow$ This forms a valid vertex cover since $x_v + x_u \geq 1 \Rightarrow$ Atleast one of x_v or $x_u \geq 0.5$

$\therefore$ Every edge has atleast one selected vertex.

• Consider x^{**} to be an optimal solution for integer optimization & x^* is an optimal solution for LP relax. and $\tilde{x}$ is solution where we select vertices if value ≥ 0.5.

Linear opt gives less cost compared to integer prog. (Always)

$\sum_{v \in V} x^{**}_v \geq \sum_{v \in V} x^*_v$

($\tilde{x}_v = 1$ if $x^*_v \geq \frac{1}{2}$)

$\tilde{x}^*_v \leq 2x^*_v$

$\Rightarrow \sum_{v \in V} \tilde{x}_v \leq 2 \cdot \sum_{v \in V} x^*_v \leq 2 \cdot \sum_{v \in V} x^{**}_v = 2OPT$

$\therefore$ Our solution is not the optimal.

Convex set:

- Join any 2 points within a polygon, with a line segment, it should lie inside the polygon.

⇒ A set that curves outwards.

* • Set X is convex if $\forall x_1, x_2 \in X$, $\underbrace{\lambda x_1 + (1-\lambda)x_2}_{\text{Line segment points}} \in X$ for $\forall \lambda \in [0,1]$

⇒ Convex combination (Extension of line segment)

* $\lambda_1, \lambda_2, \ldots \lambda_k \ni \sum \lambda_i = 1$ for $x_1, \ldots, x_k \in X$

$\Rightarrow \sum \lambda_i x_i \in X$ (if X = Convex set)

- Intersection of convex sets is convex.
- Union of convex sets need not be convex.

—— x ——

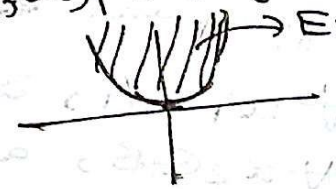
*Convex Functions:

- $f: R \to R$, $\text{Graph}(f) = \{(x_1, x_2) \in R^2 \mid x_2 = f(x_1)\}$

⇒ This is the graph of a function

a) $\text{Epigraph}(f) = \{(x, \mu) \mid \mu \geq f(x)\}$ → Epigraph.

* eg: $f(x) = x^2$,



- A function f is convex if it's epigraph is convex.

* ↳ $f: R \to R$, f (space must be a convex set)

⇒ If epigraph is convex, then on considering 2 points inside epigraph $(x_1, f(x_1))$ & $(x_2, f(x_2))$,

$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$

Holds true for points on curve, so will hold true for any points in epigraph.

* ∴ A function is convex if $\forall x_3 = \lambda x_1 + (1-\lambda) x_2$

$f(x_3) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$

—— x ——

Note: ex: $f(x) = x^2$, $f(x) = x^3$ are convex functions

$f(x,y) = xy$, is not convex

or

$f(x,y) = x^2 + y^2$

Hyperplane:

- $a_1x_1 + a_2x_2 + \dots a_nx_n = b$

$a^T = (a_1, a_2, \dots, a_n)$, b

$\Rightarrow$ A hyperplane divides the world in the current dimension into 2 half spaces.

$x \in R^n : a_1x_1 + \dots + a_nx_n \leq b$,

$x \in R^n, a_1x_1 + \dots + a_nx_n \geq b$

$\Rightarrow$ Both of these are convex.

---

* Seperation hyperplane theorem:

- If A, B are disjoint, nonempty, convex subsets of R^n. Then there exists a hyperplane (a^T, b) such that

$\forall x_1 \in A$, $a^Tx_1 + b \leq 0$ &

$\forall x_2 \in B$, $a^Tx_2 + b \geq 0$.

(Some non convex subsets can be seperated, if their convex hulls are non intersecting)

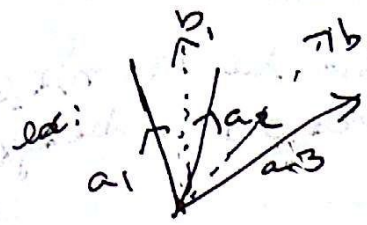
---

*Convex Cone:

- Given vectors $a_1, \ldots, a_n$

$\Rightarrow$ Convex cone of them is all linear combinations of a_i's with non negative coefficients.

$$a_x = \sum t_i a_i \mid t_i \geq 0 \; \forall i.$$

ex: 

b_i is formed by linear combinations

— x —

Convex Polyhedron:

- An intersection of a finite number of closed half spaces in R^n is a convex polyhedron.

$\Rightarrow$ Bounded polyhedron = convex polytope

ex: Cubes, Cross polytopes, Simplex etc. (These are polyhedrons)

— x —

Note: Vertex of polyhedron:

$c^T x < c^T v \; \forall x \in X - v$, then v is a vertex of X.

$c^T x = c^T v$ only when $x = v$.

— x —

*Convex Hull:

1) Given a set X, $\underbrace{C(X)}_{\text{Convex hull of } X} = \bigcap_\alpha C_\alpha$ } Intersection of all convex sets containing X.

2) $\Rightarrow$ Another definition is: $C(X) = \bar{C}$

$$\bar{C} = \sum_{i=1}^{m} \lambda_i x_i \mid x_i \in X \;\&\; \sum \lambda_i = 1$$

(All possible convex combinations)

⇒ Prove ① & ② are equivalent. (Proof in H.W questions)

ie. $\tilde{C} \subseteq C(X)$ & $C(X) \subseteq \tilde{C}$

- Using induction on m, $X \in C(X)$ ← any finite convex combination of any points in it

$\therefore \tilde{C} \subseteq C(X)$

⇒ If $x_1 \& x_2 \in \tilde{C}$, Show $\lambda x_1 + (1-\lambda) x_2 \in \tilde{C}$

$\tilde{C}$ is the convex set containing X

$\therefore C(X) \subseteq \tilde{C}$

$\therefore C(X) = \tilde{C}$

———— x ————

Note: For optimal solutions we only check vertices of feasible regions. Moving along an edge would either keep increasing or keep decreasing our optimization function.

———— x ————

Note: Corners of regions with m_x independent equality constraints will have at max m non zero values.

———— x ————

Basic Feasible Solution:

- For Linear program in equational form.
- Feasible solution $X \in R^n$ for which there exists an m element set $B \subseteq \{1, 2, \ldots n\}$ such that A_B is non singular & $x_j = 0 \; \forall j \notin B$

⇒ B – Basis (Feasible basis ⇒ Feasible solution exists with B)

↑ Variables selected = Basic variables

- For every set B, there exists at most one feasible solution $x \in R^n$ with $x_j = 0 \; \forall j \notin B$

$\Rightarrow Ax = A_B x_B + A_N x_N$ } other variables

Selected basis $\quad x_N = 0$ ↓ Non basic variables.

$\therefore Ax = A_B x_B = b$

$\therefore \boxed{x_B = A_B^{-1} b}$ $\quad \because A_B \Rightarrow$ Invertible (m x m matrix) (Non singular)

$\therefore$ we found a basic feasible solution and it is unique. (Sometimes we don't find a solution)

———— x ————

Note: For a LP in equational form, if an optimal solution exists, then there is a basic feasible solution that is the optimal solution.

———— x ————

* Solving LP: (Simplex Method) (see example)

$\Rightarrow$ We can try all basic feasible solutions, but to optimize & speed up, we look for basic feasible solutions such that,

$* \; z(\text{objective}) = c_0 - \pi^T x_N$ where $\pi \geq 0$

Non-basis variables

$\therefore Max(z) = c_0$

- We try to get that form above, ie find a basic feasible solution where $\pi \geq 0$.

Start with any solution, ($\because$ Non basic variables are 0, its optimal if $\pi \leq 0$)

$\Rightarrow$ Write basic variables in terms of non basic variables, and the objective in terms of non basic variables. → This is called simplex tableau

- If our objective is of the form $c_0 - \pi^T x_N$ we are done, else we try to move another variable in and then remove one older variable in the optimization func.

- If we get required form, we are done, else we repeat.

⇒ If we get a solution of required form, $c_0 - \pi^T x_N$ where $\pi \geq 0$, we can be sure that c_0 is our maximum solution since increasing x_N would just reduce our objective

———x———

ex: ** *

$$\max x_1 + x_2$$

s.t (slack variables)

$$-x_1 + x_2 + x_3 = 1$$
$$x_1 + x_4 = 3$$
$$x_2 + x_5 = 2$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

⇒ Start with a solution $(0,0)$

$(0,0,1,3,2)$ — Basic variables

⇒ $B = \{3,4,5\}$

- Write basic variables & objective func in terms of non basic variables

Leaves ← x_3 = $1 + x_1 - x_2$ → Enters

$$x_4 = 3 - x_1$$
$$x_5 = 2 - x_2$$

$$z = x_1 + x_2$$ } Choose a variable whose coefficient is +ve to enter

Enters

↑ Not of required form, so choose

- Choose a variable that will enter non basic variables and one that will leave.

⇒ x_2 enters & x_3 leaves

$x_2 = 1 + x_1 - x_3$

$x_4 = 3 - x_1$

$x_5 = 2 - 1 - x_1 + x_3$

$Z = 1 + 2x_1 - x_3$

- Write everything in terms of x_2, x_4, x_5 $\because (x_1, x_3) = (0,0)$

↳ Here x_1 has the coefficients, so we replace it

$\Rightarrow$ x_1 enters, x_2 does not constrain x_1. x_4 constraints x_1 to a max of 3. x_5 constraints x_1 to a max of 1 ($\because x_3 = 0$ & $x_5 = 1 - x_1$). So we remove x_5. x_5 = Leaving variable

↳ Non basic, non entering value

- $x_2 = 2 - x_5$

$x_4 = 2 - x_3 + x_5$

$x_1 = 1 + x_3 - x_5$

$Z = 3 - 2x_5 + x_3$

(x_4 is the leaving variable)

(x_3 is entering variable)

$(x_3, x_5) = (0,0)$

- Here x_3 enters, x_4 leaves since it gets constrained most.

- $x_2 = 2 - x_5$

$x_3 = 2 - x_4 + x_5$

$x_1 = 3 - x_4$

$Z = 5 - x_4 - x_5 \rightarrow Z_{max} = 5$

- Non basic variables (x_4, x_5)

$\Rightarrow (3, 2, 2, 0, 0)$

- Finally our solution is of the form $Z = C_0 - C_T \cdot X_N$. $\therefore$ This is our solution

---

Note: Entering variable = Whose coefficient is +ve in Z.

Leaving variable = The variable that limits the entering variable the most in the constraints

- Selection of these variables is called the pivot rule.

Note: Moving across these variables gives us the corners of feasible region.

Note: Degenerate cases:

* • If an entering variable has only 0 as its possible solution. (ie. Basic variable is zero)

ex: $x_3 = x_1 - x_2$ $(x_1 = 0) \Rightarrow$ Non basic, non entering
$x_4 = 3 - x_1$
$x_3 = 2 - x_2$ • x_3 constraints x_2 to 0.
$z = x_1 + x_2$ → Enter

$\Rightarrow$ If after swapping we get a case where we make x_3 enter again, we might end up looping.

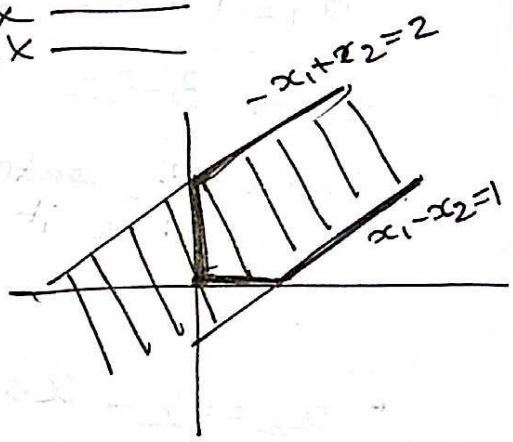
(These degenerate cases need to be handled properly.)

---

*

*(Q) max x_1
s.t
$x_1 - x_2 \leq 1$
$-x_1 + x_2 \leq 2$
$x_1, x_2 \geq 0$

$x_1 - x_2 + x_3 = 1$
$-x_1 + x_2 + x_4 = 2$



$\Rightarrow$ (0,0) is a basic feasible solution where $(x_1, x_2, x_3, x_4) = (0, 0, x_3, x_4)$
Basis variable
$B = \{x_3, x_4\}$

$x_3 = 1 - x_1 + x_2$
$x_4 = 2 + x_1 - x_2$
$z = x_1$
$x = (0, 0, 1, 2)$

• x_1 = Entering variable
x_3 = Leaving variable

$x_1 = 1 - x_3 + x_2$
$x_4 = 3 - x_3$
$z = 1 - x_3 + x_2$
$x = (1, 0, 0, 3)$

• Here x_2 would be entering

⇒ But all coefficients of x_2 in all constraints are true. Therefore none of the (x_1, x_4) constrain x_2. So we cannot proceed from here since the region is unbounded

⇒ If x_2 enters, solution is **unbounded** so we stop and do not proceed further.

———x———
———x———

- The simplex method assumes we know a basic feasible solution to start with.

⇒ Finding a basic feasible solution for an LP is as hard as finding an optimal solution for another LP.

Lemma:

eg: $A\bar{x} = b$, $\bar{x} \geq 0$, $\max c^T x$ → (1)

↓

Add m variables $x_{n+1}, \ldots x_{n+m}$

$\bar{x} = (\bar{x}, x_{n+1}, \ldots, x_{n+m})$

$x_{n+1}, x_{n+2}, \ldots = 0$

⇒ Add constraints, Modify existing constraints,

(2) ←

$a_{11}x_1 + \ldots + a_{1n}x_n + x_{n+1} \cdot 0 = b_1$

$\vdots$

$a_{m1}x_1 + \ldots + a_{mn}x_n + x_{n+m} \cdot 0 = b_m$

⇒ Modify objective as $\max(+x_{n+1} + x_{n+2} + \ldots + x_m)$

- Essentially feasible solution for (1) is the optimal solution for (2).

(Both are L.Ps)

———x———

Lemma: Given a basis B, there is a unique
*** Simplex Tableau

$$\frac{x_B = p + Q \cdot x_N \quad \cdot r^T \leq 0}{z = z_0 + r^T x_N}$$

- If this was not true then our method would not work.

Proof:

$$x = (x_B, x_N)$$

$$A = [A_B | A_N]$$

$$Ax = b \Rightarrow [A_B | A_N] \begin{bmatrix} x_B \\ x_N \end{bmatrix} = b$$

$$\Rightarrow A_B x_B + A_N x_N = b$$

$$(1) \leftarrow \Rightarrow x_B = A_B^{-1} b - A_B^{-1} A_N x_N$$

(A_B^{-1} exists since its non singular)

$$\Rightarrow z = c^T x = c_B^T x_B + c_N^T x_N$$

$$= c_B^T A_B^{-1} b - c_B^T A_B^{-1} A_N x_N + c_N^T x_N$$

$$(2) \leftarrow z = \underbrace{c_B^T A_B^{-1} b}_{z_0} + \underbrace{(c_B^T A_B^{-1} A_N + c_N^T)}_{r^T} \cdot x_N$$

$$\Rightarrow r = c_B^T A_B^{-1} A_N + c_N^T$$

- From (1), (2) a simplex tableau always exists.

$$* \Rightarrow x_B = P_0 + Q_0 x_N = P_1 + Q_1 x_N \quad \left.\right\} \text{For all } x_N$$

$$\Rightarrow P_0 = P_1 \quad (\because \text{If } x_N = [0, \ldots, 0])$$

$$\therefore Q_0 = Q_1$$

$\therefore$ This simplex tableau has to be unique

———x———

*Simplex Method Summary

Step1: Given a linear program, convert it to
$\max c^T x$ such that $Ax=b, x \geq 0$

Step2: If x_0 (Basic feasible solution) is not available, solve

$\max -(x_{n+1}+x_{n+2}+\dots x_{n+m})$

such that

$\bar{A}\bar{x}=b, \bar{x}\geq 0$

where $\bar{A}=[A|I_m]$, $\bar{x}=(x_1,\dots x_{n+1},\dots x_{n+m})$

with $\bar{x}_0=(0,\dots,0,b_1,\dots b_m)$

If $\bar{z}^*<0$, return INFEASIBLE

Else $x_0=(\bar{x}_1^*,\dots,\bar{x}_N^*)$

Step3: For a basis B, compute simplex tableau

$$x_B = p + Q x_N$$
$$z = z_0 + r^T x_N$$

Step4: If $r \leq 0$, return $x^*=(x_B^*,0,\dots,0), z_0^*$

Step5: If not, select entering variable x_v whose coefficient in r is positive. If multiple such x_v exist, use pivot rule.

Step6: If the column of the entering variable in tableau is positive return UNBOUNDED

Step7: Else, select a leaving variable x_u such that $q_{ui}<0$ & $\frac{-p_i}{q_{ui}} = \min\left\{\frac{-p_k}{q_{ui}}\right\}$

(p_i → Constant of i^{th} row; q_{ui} → Coefficient of x_u in i^{th} row; k → Basis; $k=(0,1,\dots,m)$)

Step8: $B \leftarrow (B \setminus \{u\}) + \{v\}$ Our new basis.

Go to step 3.

Q) max $9x_1 + 3x_2 + x_3$

such that

$$x_1 \leq 1$$
$$6x_1 + x_2 \leq 9$$
$$18x_1 + 6x_2 + x_3 \leq 81$$
$$x_1, x_2, x_3 \geq 0$$

⇒ Add slack variables

$$x_1 + x_4 = 1$$
$$6x_1 + x_2 + x_5 = 9$$
$$18x_1 + 6x_2 + x_3 + x_6 = 81$$
$$x_1, \ldots, x_6 \geq 0$$

Start with,

→ $B = \{4, 5, 6\}$, $(x_1, x_2, x_3 = 0)$ $(0,0,0)$

$$x_4 = 1 - x_1$$
$$x_5 = 9 - 6x_1 - x_2$$
$$x_6 = 81 - 18x_1 - 6x_2 - x_3$$
$$z = 0 + 9x_1 + 3x_2 + x_3$$

(x_1: Entering)

- Entering variable according to pivot rule
  ↓
  Dantzik: Choose one with max. the coefficient

- Leaving is x_4

→
$$x_1 = 1 - x_4$$
$$x_6 = 3 - x_2 + 6x_4$$
$$x_6 = 63 - 6x_2 - x_3 + 18x_4$$
$$z = 9 - 9x_4 + 3x_2 + x_3$$

(x_2: Entering) $(1,0,0)$

- Leaving is x_5

→
$$x_1 = 1 - x_4$$
$$x_2 = 3 - x_5 + 6x_4$$
$$x_6 = 45 - 18x_4 + 6x_5 - x_3$$
$$z = 18 + 9x_4 - 3x_5 + x_3$$

(x_4: Entering) $(1,3,0)$

- Leaving is x_1

$\rightarrow$ $x_4 = 1 - x_1$ (0,9,0)

$x_2 = 9 - 6x_1 - x_5$

$x_6 = 27 + 18x_1 + 6x_5 - x_3$

$z = 27 - 9x_1 + 3x_5 + x_3$

- And so on...

$\rightarrow x_4 = 1 - x_1$

$x_2 = 9 - 6x_1 - x_5$

$x_6 =$

- $(0,0,0) \rightarrow (1,0,0) \rightarrow (1,3,0) \rightarrow (0,9,0)$
  $\downarrow$
  $(0,9,27) \leftarrow (1,3,45) \leftarrow (1,0,63)$
  $\downarrow$
  $(0,0,81)$

Note: 1) Dantzig's Pivot Rule: Choose entering variable to be one with maximum +ve coefficient.

2) Bland's Pivot Rule : Choose entering variable to be the one with +ve coefficient & smallest index. (Give ordering to variables initially)

↓ Doesn't loop. → Proof? Text Book

(Leaving variable if multiple exist, choose smallest index)

3) Pivot Rule 3 (Largest Increase) : Choose entering variable which increases z the most.

Most practical ⇐ 4) Steepest Edge : Choose entering variable that moves current basic feasible soln closest to vector c.

$\max \left(\frac{c^T(x_{new} - x_{old})}{\|x_{new} - x_{old}\|} \right)$ are perfect & can take

- None of these are perfect & can take exponential time in the worst case.

Note: Inverse of matrices $\Rightarrow O(N^3)$ but precision errors etc

∴ We use decompositions into upper triangulars, lower triangulars matrices etc → To ease calculations.

———x———

Q) max x_1+x_2

such that $x_1+2x_2 \geq 1 \Rightarrow -x_1-2x_2 \leq -1$

$2x_1+3x_2 \leq 6$

$x_1, x_2 \geq 0$

- $x_1+x_2 \leq 2x_1+3x_2 \leq 6$

$x_1+x_2 \leq 2x_1+3x_2-x_1-2x_2 \leq 5$

$x_1+x_2 \leq x_1+\frac{3}{2}x_2 \leq 3$

(Find upper bounds, another method)

$2(x_1+x_2)+x_2 \leq 6$

$2(x_1+x_2) \leq 6 \quad \because x_2 \geq 0$

$(x_1+x_2) \leq 3$

$-y_1x_1-2y_1x_2 \leq -y_1$

$2y_2x_1+3y_2x_2 \leq 6y_2$ $\quad (\geq 1)$

$x_1+x_2 \leq (-y_1+2y_2)x_1 + (-2y_1+3y_2)x_2 \leq -y_1+6y_2$

Original objective $\quad (\geq 1)$

∴ Our new LP is,

min $-y_1+6y_2$

s.t

$-y_1+2y_2 \geq 1$

$-2y_1+3y_2 \geq 1$

* This new LP is called the dual problem.

———x———

Weak Duality:

- x^* be an optimal solution to (P) & $\tilde{y}$ be a feasible solution to (D), then (P → Primal, D → Dual)

$$c^T x^* \leq b^T \tilde{y}$$

$$\Rightarrow c^T x^* \leq b^T y^*$$

(Primal & Dual optimal solutions need not be equal – weak duality)

↳ Upper bound.

Primal 2 Dual:

$\Rightarrow$ max $c^T x \rightarrow c_1x_1 + c_2x_2 + \dots + c_nx_n$ } Primal.

such that

$$Ax \leq b$$

$$x \geq 0$$

- To get to the dual form introduce variables $y_1, y_2, \dots, y_m$

Sign doesn't change only if all y's are +ve

$$\Rightarrow y_1(a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n) \leq b_1 y_1$$

$$y_2(a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n) \leq b_2 y_2$$

$$\vdots$$

$$y_m(a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n) \leq b_m y_m$$

- Add all to get,

$$(y_1a_{11} + y_2a_{21} + y_3a_{31} + \dots y_ma_{m1})x_1$$
$$+ (y_1a_{12} + y_2a_{22} + \dots + y_ma_{m2})x_2$$
$$+ \dots$$
$$+ (y_1a_{1n} + y_2a_{2n} + \dots + y_ma_{mn})x_n$$
$$\leq b_1y_1 + b_2y_2 + \dots + b_my_m$$

$\rightarrow$ (1)

Dual

- Our objective $\min_x c^T x \leq$ (1) → with constraints, so we define new objective as $b_1y_1 + b_2y_2 + \dots + b_my_m$ where our constraints are,

$y_1a_{11} + y_2a_{21} + \dots + y_ma_{m1} \geq c_1$ and so on till n constr.

$y_1, y_2, \dots \geq 0$

Note: Primal form → n variables & m constraints

⇓

Dual form → m variables & n constraints.

———x———

Note: Dual formulation for equational primal LP does not have constraint $y_i \geq 0$.
(∵ -ve multiplication is fine (no sign change))

———x———

**	Primal	Dual
Variables	$x_1, \ldots, x_n$	$y_1, \ldots, y_m$
Matrix	A	A^T
Objective func.	$\max c^T x$	$\min b^T y$
Constraints *	i^{th} constraint	
	$\leq$	$y_i \geq 0$
	$\geq$	$y_i \leq 0$
	$=$	$y_i \in R$
	$x_j \geq 0$	$\geq$ j^{th} constraint
	$x_j \leq 0$	$\leq$
	$x_j \in R$	$=$

———x———

(1) $\max x_1$
s.t
$x_1 - x_2 \leq 1$
$-x_1 + x_2 \leq 2$
$x_1, x_2 \geq 0$
} Primal

- This primal is unbounded so dual should give infeasible solution. (Since dual is an upper bound of objective)

$\Rightarrow y_1(x_1 - x_2) \leq y_1$
$y_2(-x_1 + x_2) \leq 2y_2$

$x_1 \leq (y_1 - y_2)x_1 + (y_2 - y_1)x_2 \leq y_1 + 2y_2$

min $y_1 + 2y_2$, such that

$\Rightarrow$ $y_1 - y_2 \geq 1$

$-y_1 + y_2 \geq 0 \Rightarrow y_2 \geq y_1$

$y_1, y_2 \geq 0$

$y_1 \geq 1 + y_2 \geq 1 + y_1$ → Not possible

∴ There is no feasible solution

———— x ————

* Strong Duality:

1) Neither (P) nor (D) has a feasible solution
2) (P) is unbounded & (D) is infeasible
3) (P) is infeasible & (D) is unbounded
4) Both (P) and (D) have a feasible solution.

If x^* is optimal solution to (P) & y^* is an optimal solution to (D), then

$c^T x^* = b^T y^*$ (Exact bound)

———— x ————

Note:

Define: max $c^T x$
* such that
* $Ax \leq b$
$-A^T y \leq -c$
$-c^T x + b^T y \leq 0$ $(c^T x \geq b^T y)$
$x, y \geq 0$

From max $c^T x$ (Primal)
s.t
$Ax \leq b$
$x \geq 0$
&
min $b^T y$
s.t
$A^T y \geq c$
$y \geq 0$ (Dual)

(We know $c^T x \leq b^T y$ from formulation of dual)

• If $(\tilde{x}, \tilde{y})$ is a feasible solution for this x^* then $(\tilde{x}, \tilde{y})$ is an optimal solution for both primal & dual since $c^T x = b^T y$.

———— x ————

* **Proving Strong Duality:** (Proof in text book) $(c^T x^* = b^T y^*)$

* (P) $\max c^T x$, s.t
  $Ax \leq b$
  $x \geq 0$

(P') $\max \bar{c}^T x$, s.t
$\bar{A}\bar{x} = b$
$\bar{x} \geq 0$

$\bar{x} = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+m})$
$\bar{A} = (A \mid I_m)$
$\bar{c} = (c_1, \ldots, c_n, 0, \ldots 0)$
} Add m slack variables

- $\bar{x}_B = \phi + Q\bar{x}_N$

$z = z_0 + r^T \bar{x}_N$

$x_B^* = A_B^{-1} b \rightarrow (1)$

- Consider
* $y^* = (\bar{C}_B^T \bar{A}_B^{-1})^T$
- This y^* is valid, we will show its a feasible solution

* $c^T x^* = \bar{C}^T \bar{x}^* = \bar{C}_B^T \bar{x}_B^* = (\bar{C}_B^T A_B^{-1}) b$ from (1)
$(y^*)^T b = b^T y^*$ $\}$ $\therefore c^T x^* = b^T y^*$
$\therefore y^*$ is valid

$\Rightarrow$ Show y^* is feasible,
- $y^* \geq 0$ & $A^T y^* \geq c$ (solution to dual)

$I_m y^* \geq 0$, $\bar{A}^T = \left[\frac{A^T}{I_m}\right]$, $\bar{A}^T y^* \geq \bar{C}$ we get this from above

$$\bar{A}^T y^* = \bar{A}^T (\bar{C}_B^T \bar{A}_B^{-1})^T = (\bar{C}_B^T \bar{A}_B^{-1} \bar{A})^T$$
$$= (\underbrace{\bar{C}_B^T \bar{A}_B^{-1} \bar{A}_B}_{\bar{C}_B^T})^T + (\underbrace{\bar{C}_B^T \bar{A}_B^{-1} \bar{A}_N}_{\text{Non basic}})^T$$

We know,
- $r^* = \bar{C}_N - (\bar{C}_B^T \bar{A}_B^{-1} \bar{A}_N)^T \leq 0$ → For optimality (x^*)
$\Rightarrow (\bar{C}_B^T \bar{A}_B^{-1} \bar{A}_N)^T \geq \bar{C}_N \rightarrow (2)$

$\bar{A}^T y^* \geq \begin{pmatrix} \bar{C}_B \\ \bar{C}_N \end{pmatrix} = \bar{C}$, $\therefore \bar{A}^T y^* \geq \bar{C}$

- Hence y^* is a feasible solution

~~$x_B = A_B^{-1} b + A_B^{-1} A_N x_N$~~

~~$C^T x = C_B^T x_B + C_N^T x_N$~~

$\therefore$ From strong duality proof, we have shown that $y^* = (\bar{C}_B^T \bar{A}_B^{-1})^T$ is a valid solution, feasible and gives us the same optimal value as $c^T x^*$, $\because \underbrace{c^T x^* = b^T y^*}_{\text{Validity.}}$

(A_B, C_B correspond to basis of x^*)

Note: We know $\bar{r}_N = \bar{C}_N - (\bar{C}_B^T \bar{A}_B^{-1} \bar{A}_N)^T$ from the lemma on unique simplex tableau.

Farkas Lemma

- Let $a_1, \ldots, a_m, b \in R^m$

Then exactly one of the two possibilities occurs

F1) The point b lies in the convex cone generated by $a_1, \ldots, a_n$

F2) There exists a hyperplane h passing through $\bar{0}$, of the form $h = \{x \in R^m \mid y^T x = 0\}$ for suitably chosen y s.t all the vectors $a_1, \ldots, a_n$ lie on one side of it & b lies strictly on the other side. That is,

(Sep. hyperplane theorem)

$y^T a_i \geq 0 \quad \forall i = 1, \ldots n$

& $y^T b < 0$

---

1) $Ax=b$ has non negative solution iff

Lem1: $\forall y$ (non negative) $\in R^m$, $y^TA \geq 0^T$ & $y^Tb \geq 0$

* $Ax=b$ means b lies in non-ve hypercone generated by A if $x =$ x solution.
So there cannot exist a separating hyperplane between points of A & b (From Farkas Lemma). where y is non negative. (y passes through 0)

$$\begin{bmatrix} \bar{a}_1 & \bar{a}_2 & \cdots & \bar{a}_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \bar{b}$$

——— x ———

* Note: Prove duality using Farkas theorem.

——— x ———
——— x ———

* Ellipsoid Methods:

* Hypersphere: $\sum_{i=1}^{n} x_i^2 = r^2$

Region inside hypersphere: $B_n(0,R) = \{x \in R^n \mid x^Tx \leq R^2\}$

* Ellipse: $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$$

$$\Rightarrow \sum_{i=1}^{n} \frac{x_i^2}{a_i^2} = 1 \quad \} \text{Hyperellipse}$$

$$x^T \begin{bmatrix} \frac{1}{a_1^2} & \cdots & 0 \\ \vdots & \frac{1}{a_2^2} & \vdots \\ 0 & \cdots & \frac{1}{a_n^2} \end{bmatrix} x = 1 \quad \} x^T M^{-1} x = 1$$

M^{-1}, $M = \begin{bmatrix} a_1^2 & \cdots & 0 \\ \vdots & a_2^2 & \vdots \\ 0 & \cdots & a_n^2 \end{bmatrix}$

* • Ellipse from a circle : $x \in B_n(0,1)$ (origin centered circle/ball, $r=1$)

$x \rightarrow Mx+S$ (M: non singular)

Ellipse from circle.

$\Rightarrow \{x \in R^n \mid M^{-1}(x-S) \in B_n(0,1)\}$

x belongs to ellipse if ... (Belongs to ball if $x^T x \leq r^2$)

$\Rightarrow \{x \in R^n \mid (x-S)^T M^{-1T} M^{-1}(x-S) \leq 1\}$

Let $Q = MM^T$, $Q^{-1} = M^{-1T} M^{-1}$

---

* $\Rightarrow$ Ellipsoid in n dim : $\{x \in R^n \mid (x-S)^T Q^{-1}(x-S) \leq 1\}$ (for some Q, S)

(Belongs to ellipsoid region)

Ball in n dim : $\{x \in R^n \mid (x-S)^T (x-S) \leq 1\}$ (for some S)

(Belongs to ball region)

---

* Ellipsoid Method:

* • $E(x, Q) = \{x \in R^n \mid (x-S) Q^{-1} (x-S) \leq 1\}$

Step 1: $k=0$, $E_0 = B_n(0,R)$, $S_0 = 0$

Step 2: Does $S_k \in P$? (S_k = Centre of ellipsoid)

If YES stop

If NO Let h_k be hyperplane that seperates S_k from P.

Step3: $H_{KH} = E_K \cap \{x \in R^n \mid h_k^T a_k \leq h_k^T s_k\}$

E_{KH} = the smallest ellipsoid containing H_{KH}

Step4: If Volume $(E_{KH}) < \varepsilon^n$

STOP, (Infeasible)

else, $k \Rightarrow k+1$

goto step2.

———— x ————

Note: the seperation hyperplane is provided by some other methods (assume Oracle).

Note: We generally choose planes through the center of ellipsoid since constructing enveloping ellipsoid is simple. Choosing other hyperplanes could potentially reduce convergence iterations, but hard to deal with.

———————

- We know that $\frac{Vol(E_{k+1})}{Vol(E_k)} \leq e^{\frac{-1}{2n+2}}$ → Geometrically proved

$\Rightarrow k > n(2n+2)\ln R/\varepsilon$ (Polynomial run time)

↑ After these iterations, it is guarenteed to converge.

i) Hyperplane Selection: (Seperation plane oracle for LP formulation)

⇒ Now, consider $\max c^T x$

$Ax \leq b$

$x \geq 0$

- $a_1x_1 + \ldots + a_nx_n \leq b$

$a_k^T s_k > b_k$ * • The constraint that is not satisfied by s_k consider that to be the hyperplane.

~~$a_1x_1 + \ldots + a_nx_n \leq b$~~

ii) Smallest ellipsoid: (containing given regions)

- $Q_0 = R^n I_n$, $x_0 = 0$

$$x_{k+1} = x_k - \frac{1}{(n+1)} \frac{Q_k h_k}{\sqrt{h_k^T Q_k h_k}}$$

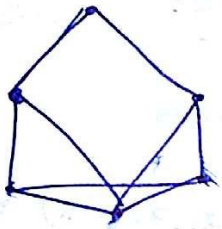
$$Q_{k+1} = \frac{h^2}{h^2-1}\left(Q_k - \frac{2}{n+1} \cdot \frac{Q_k h_k h_k^T Q_k}{h_k^T Q_k h_k}\right)$$

- (Derivation not required)

Ellipsoid

Note: ~~Ellipsoid~~ methods are polynomial in time. (But very slow)

*Travelling Salesman Problem:



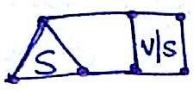
- $x_e = 1$ if e is selected, $x_e = \{0,1\}$
  $= 0$ otherwise
- $\min \sum_{e \in E} c_e x_e$

s.t

One edge to enter and one edge to exit. $\leftarrow \sum_{e \in \delta(v)} x_e = 2 \quad \forall v \in V$

$\delta(v) = \{e \in E$ s.t e is incident on $v\}$

$\Rightarrow$ This would not always work,



$\Rightarrow$ we might end up selecting 2 different cycles, which is not a travelling Salesman solution.

Solution:

* $\delta(S) = \{e \in E$ | one end of $e \in S$ & other $\in V/S\}$

$$\sum_{e \in \delta(S)} x_e \geq 2 \quad \forall S \subseteq V \text{ s.t } (S \neq \phi, S \neq V)$$

- These constraints ensure that every subset of vertices will be visited atleast once.
- Using LP relaxation, $0 \leq x_e \leq 1$ //

$\Rightarrow$ The number of constraints we have here are non-polynomial ($2^n - 2$ constraints)

- Using min cut, if min-cut(graph) ≥ 2, then ellipsoid method can be used with a hyperplane oracle that runs in polynomial time to get a [redacted] solution.
  - (If not, then the problem is infeasible)

———x———

Note: There are polynomial oracles for travelling salesman problem.

———x———
———x———

* Two player zero sum games:

*

- Consider Rock, Paper, Scissors

		R (y_1)	P (y_2)	S (y_3)
x_1	R	0	-1	1
x_2	P	1	0	-1
x_3	S	-1	1	0

$= A$

- Let $x = (x_1, x_2, x_3)$ be strategy by player 1 & $y = (y_1, y_2, y_3)$ be strategy by player 2.

$\rightarrow \sum x_i = 1, \; x_i \geq 0$

$\sum y_i = 1, \; y_i \geq 0$

- a_{ij} = [redacted] Payoff for player 1, if player 1 performs i^{th} action & player 2 performs j^{th} action.

$\Rightarrow \sum x_i \cdot y_j \cdot a_{ij} = \text{Payoff} = x^T A y$

$\Rightarrow \alpha(y) = \max_x x^T A y$ } Choose action x such that

$\Rightarrow$ Utility for column player is opposite, so

$\beta(x) = \min_y x^T A y$ } Opponent chooses y such that.

Note: (x^*, y^*) is Nash equilibrium if

* x^* is s.t it achieves $\alpha(y^*)$ & y^* is s.t it achieves $\beta(x^*)$

i.e $\alpha(y^*) = \max_x x^T A y^* = x^{*T} A y^*$

$\beta(x^*) = \min_y x^{*T} A y = x^{*T} A y^*$

ie. $\alpha(y^*) = \beta(x^*)$

Note:- One can prove that if (x^*, y^*) is Nash equilibrium (NE)

$\Rightarrow$ x^*, y^* are worst case optimal strategies

- If $\beta(x^*) = \alpha(y^*) \Rightarrow x^* y^*$ is NE.

---

* Worst case optimal strategies:

- $\max_x \min_y x^T A y$, x^* is worst case optimal for row player if x^* is s.t $\beta(x^*) = \max_x \beta(x)$

y^* is worst case optimal for the column player if y^* is s.t $\alpha(y^*) = \min_y \alpha(y)$

$= \min_y \max_x x^T A y$

(we try to force our opponent to take a bad move)
(Minimax)

Row player: $\max_x \min_y x^T A y$;

$$\sum x_i = 1$$
$$x_i \geq 0$$

$\approx \min_{j=1,n} x_1 a_{1j} + \dots + x_m a_{mj}$

Column player: $\min_y \max_x x^T A y$;

s.t

$$\sum y_j = 1$$
$$y_j \geq 0$$

$\approx \max_{i=1,m} y_1 a_{i1} + \dots + y_n a_{in}$

(Exam) → Prove

* • These two LP's are primal / dual forms.
(Optimal value in both cases is $x^* A y^*$)

Note: Minmax strategies are Nash Equilibriums only for 2 player zero sum games

---

Integer Programming

- min vertex cover
- max independent set
- maximum matching
- max flows | min cut

1) Maximum matching in bipartite graph

* $G(X \cup Y, E)$ → X union Y two sets of vertices is a bipartite graph

- $x_e = 1$ if e is selected
  $= 0$ otherwise
- $\max \sum_{e \in E} x_e$, $\sum_{e \in E \text{ edges incident on vertex } V} x_e \leq 1 \quad \forall V \in X \cup Y$ $x_e \in \{0,1\}$

⇒ Let $A_{n \times m}$ be incidat matrix of G denoted by

$a_{ij} = 1$ if $V_i \in e_j$
$= 0$ otherwise

$$\max \sum_{e \in E} x_e$$
$$Ax_e \leq 1$$
$$x_e \geq 0$$
$$x_e \in Z^m$$

- A matrix A is said to be totally unimodular if every square submatrix A has determinant 0, 1, or -1.

Lemma: Let A be a totally unimodular matrix. Consider the matrix $\bar{A}$ obtained by appending a unit vector e_j as new last column. (↳ Come not used here)

Then $\bar{A}$ is also totally unimodular

Proof: $\bar{A} = \left[\begin{array}{c|c} A & \begin{matrix}0\\0\\\vdots\\1\\0\\\vdots\end{matrix} \end{array}\right]$ ← e_j

- Let Q be $l \times l$ submatrix of $\bar{A}$

if 1) $\Rightarrow$ Q is a submatrix of A

$$\det(Q) \in \{0, 1, -1\}$$

2) Q contains elements from e

$$\det(Q) = \sum_{i=1}^{l} (-1)^{i+l} \underbrace{\det Q^{i,l}}_{\{0,1,-1\}}$$

$\therefore \det(Q) \in \{0, 1, -1\}$ ↳ Laplace expansion

Lemma: Consider a linear program

$$\min c^T x$$
$$\text{s.t}$$
$$Ax \leq b, \; x \geq 0$$
$$\& \; b \in \mathbb{Z}^n$$

- If A is totally unimodular & if LP has an optimal solution, then it also has an optimal integral solution

Proof: Convert to equational form,

$$\max \bar{c}^T \bar{x}$$

s.t

$$\bar{A}\bar{x} = b$$

$$\bar{x} \geq 0$$

$$\bar{A} = [A \mid I_m]$$

$$\bar{x} = (x_1, \ldots, x_n, x_{n+1}, \ldots x_{n+m})$$

$$\bar{c} = (c_1, \ldots, c_n, 0, \ldots 0)$$

$$\Rightarrow x^* = (\bar{x}_B^*, \bar{x}_N^{*\rightarrow 0})$$

$$\bar{x}_B = \bar{A}_B^{-1} b$$

$$\Rightarrow \bar{A}_B^{-1} = \frac{1}{\det A_B} \left[\text{Integers, cofactor matrix}^T \right]$$

$\det A_B$ ↓ $+1$ or -1 since basis is non singular

$$\Rightarrow \bar{x}_B = \frac{1}{\det A_B} \left[\text{Integers} \right] \left[\text{Integers} \right]$$

$\bar{x}_B$ is an integral solution.

$\bar{A}_B^{-1}$ ↓ Integers

b ↓ Integers

$\times$ Integers

$\Rightarrow$ Integral solution

Lemma: * Let $G = (X \cup Y, E)$ be an undirected bipartite graph. Then incident matrix A of G is totally unimodular.

Proof: Inductive, consider single element $\Rightarrow 1 \times 1$ matrix

$\det = 1$ or 0.

$l = 1 \Rightarrow$ True.

$l > 1$, Consider Q_{l+1}, at most two 1's in any column.

Assume $l-1 \Rightarrow$ True (Inductive Hypothesis)

- If there are no 1's $\Rightarrow$ det $= 0 \rightarrow$ True in any column
- If there is a single 1 in any column $\Rightarrow$ Expand on that column (From Laplace expansion) and since $l-1 \times l-1$ is true $1 \times \det(l-1 \times l-1) \in \{0, 1\} \rightarrow$ True

$*$ • If all columns have 2 1's in it

Every edge has one endpoint in X & another endpoint in Y.

$\Rightarrow \overset{\text{Sum}}{\Sigma}$ all the rows corresponding to vertices in X we get, $(1, \ldots\ldots, 1)^T$

$\Rightarrow$ Sum all the rows corresponding to vertices in Y, we get $(1, \ldots\ldots, 1)^T$

(∵ All columns have 2 1's ⇒ all vertices corresponding to column edges belong to one of X & Y)

$\Rightarrow$ This means that performing row operations we can get these $(1, \ldots\ldots, 1)^T$ & $(1, \ldots\ldots, \neq 1)^T$ for two rows & then on subtraction we get all zeroes $\Rightarrow$ Det $= 0$ //

$\therefore A =$ Totally unimodular

1) Dual of maximum matching: (Bipartite)

cont.

$$\min \sum_{u \in X \cup Y} y_u$$

s.t

$$A^T y \geq 1$$

$$y \geq 0$$

} Vertex cover on bipartite graph.

This is also totally unimodular (∵ A = Totally unimodular)
↳ So we will have an integral solution.

- Although we didn't say that $y \leq 1$, it will always be true for the optimal solution since we are minimizing it.

⇒ If $y_i = 2$ satisfies constraints ⇒ $y_i = 1$ also satisfies the constraints.

Note: This tells us that the size of a minimum vertex cover is equal to the size of a maximum matching in a bipartite graph (∵ These are dual forms of each other)

↑

* Konig's Theorem

---

2)

* Perfect Weighted Maximum Matching: (Bipartite)

- Perfect matching requires single edges from every vertex.

$$\Rightarrow \max \sum_{e \in E} w_e x_e$$

$x_e \in \{0, 1\}$
$0 \leq x_e \leq 1$ } LP relax

s.t

$$\sum_{e \text{ is incident on } v} x_e = 1 \quad \forall v \in X \cup Y$$

} Since its perfect matching

$\Rightarrow$ Let x^* be optimal solution to LP relaxation.

if $x^* \notin Z^n$, $\exists$ e_1, such that

$$0 < x^*_{e_1} < 1$$

$e_1 = (a_1, b_1)$ $\exists$ $e_2(b_1, a_2)$ such that
(since sum on $b_1 = 1$)

$0 < x_{e_2} < 1$

($\because$ Sum of x_e's on a vertex $= 1$)

$\Rightarrow$ $\exists$ $e_3 = (a_2, b_2)$ s.t $0 < x_{e_3}$

(Since sum on $a_2 = 1$)

⋮

This loops since number of edges are finite. (It forms a cycle, so the number of edges are even)

- Consider all edges $e_1, e_2, \ldots, e_t$ in the cycle,

Solution:

$$\tilde{x}_{e_1} = x^*_{e_1} + \varepsilon$$

$$\tilde{x}_{e_2} = x^*_{e_2} - \varepsilon$$

$$\tilde{x}_{e_3} = x^*_{e_3} + \varepsilon$$

⋮

$$\tilde{x}_{e_k} = x^*_{e_k} - \varepsilon$$

These are integers, either 0 or 1.

$$\left\{ \tilde{x}_e = x^*_e \quad \forall \; e \notin \{e_1, \ldots, e_k\} \right.$$

$\Rightarrow$ Now we can see that at each of the vertices outside the cycle the constraint holds and at any vertex within the cycle, $x^*_{e_i} + x^*_{e_j} \leq 1$

- And $x_{e_i}^* + \varepsilon + x_{e_j}^* - \varepsilon \leq 1$ //

∴ It satisfies the constraints.

⇒ Objective using new solution,

$$\sum w_e \tilde{x}_e = \sum w_e \tilde{x}_e \quad (e \notin \{e_1, \ldots, e_k\})$$

$$+ \sum_{i=1}^{n} w_{e_i} (-1)^{i+1} \varepsilon$$

$$+ \sum_{i=1}^{n} w_e x_e^*$$

$$= \text{Optimal} + \varepsilon \Delta$$

$$(|\varepsilon| \leq \min(\min x_e, \min(1 - x_e))) \rightarrow (1)$$

- Now Δ cannot be true since our optimal solution cannot be better than the LP relax. $\Delta = 0$ since its equal to our LP relax solution?

⇒ Based on (1), we can select an ε such that one of our $x_{e_1}, \ldots, x_{e_k}$ becomes an integer.

- We repeat this process for all the cycles that exist, a maximum of $n/2$ iterations, all the edges are now integer selections (all e_i's)

∴ In polynomial time we can convert LP relaxation solution to a valid matching solution.

3) Maximum Independent Set:

Optimize $\max \sum_i v_i$

$$v_i + v_j \leq 1 \quad \forall (i,j) = \text{edge}$$

$$v_i, v_j \in \{0,1\}$$

(Hard to solve)?

Note: (When not all constraints need to be satisfied.) (OR conditions)

ex: $|x| \leq 3$ } AND constraints

$-3 \leq x \leq 3$

OR constraints: $|x| \geq 3$

$+x \leq -3$ or $x \geq 3$

ex: $x \leq -3$

$-x \leq -3 + L$ } Case1 → Large value

or

$x \leq -3 + L$ } Case2

$-x \leq -3$

We add y and now we can solve this LP

$x \leq -3 + yL$

$-x \leq -3 + (1-y)L$

$y \in \{0,1\}$

- We add additional variables and solve this new LP

(Branch & Bound)

(Integer programming solve using LP relaxation)

(Q) $\max x_1 + x_2$

s.t

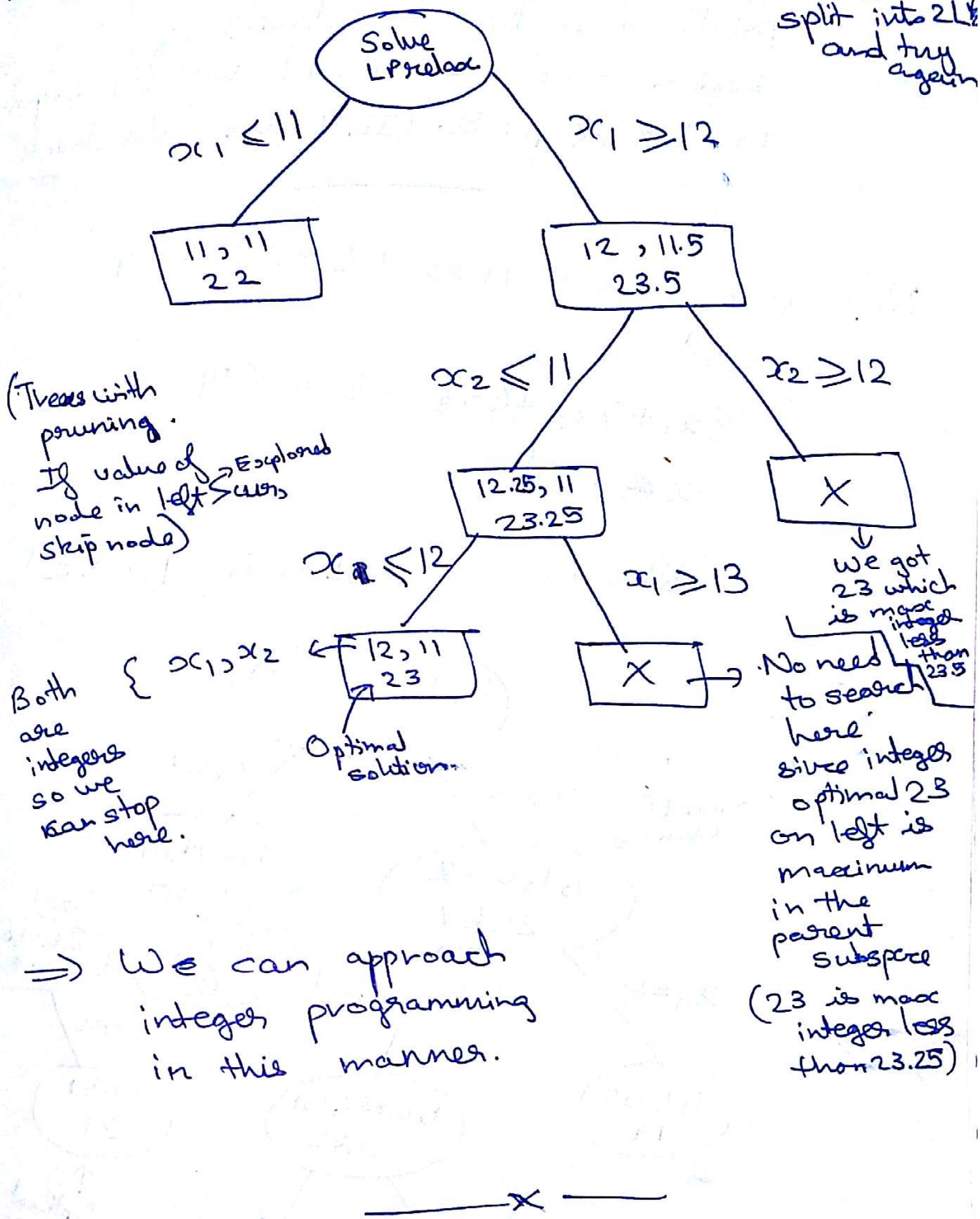
$x_1 + x_2 \leq 35$

$x_2 \leq x_1$

$x_2 \leq 35.5 - 2x_1$

$x_1, x_2 \geq 0$ & integers

- Solve using LP relax and for non integer results split into 2 LP's and try again



⇒ We can approach integer programming in this manner.

Note: If coefficients of IP in objective function are not integers but rational numbers, multiply with lcm (denominators of rational numbers) to get integer optimal values.
(i.e. Multiplying objective with constant does not change the optimal solution but just the optimal value)

Note: In IP, we can't check all integer points near the boundaries of the feasible region since this would be an exponential number of points. (In LP we only check corners)

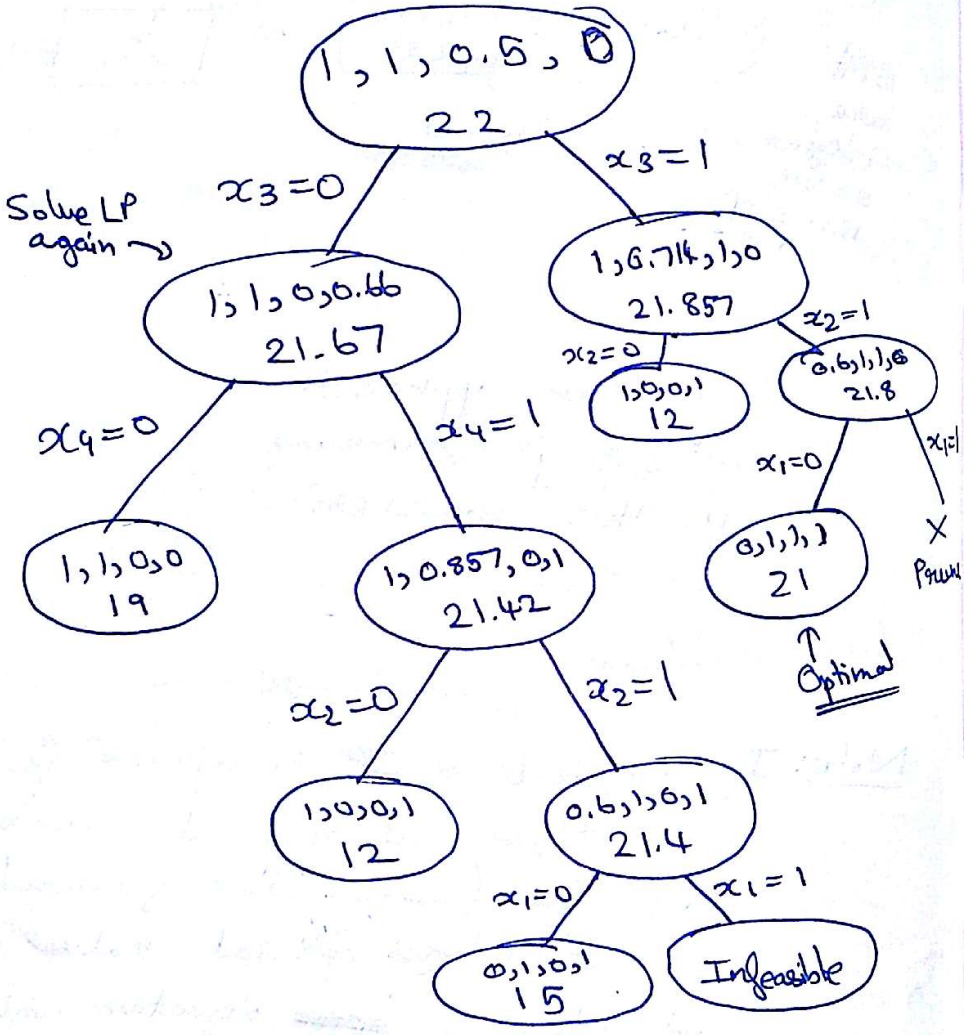
Q) max $8x_1 + 11x_2 + 6x_3 + 4x_4$

s.t

$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14$

$x \in \{0,1\}^4$

- Solve LP $\Rightarrow$ Solution is $1, 1, 0.5, 1$



---

* Branch & Bound could be either DFS or Best Fit (BFS). DFS makes it easier to formulate the problem since all we need to do is keep reducing the search space as we go down the tree. (Best Fit = Choose branch that seems best and expand)

Branch & Bound: (Pseudo code)

(0) $\bar{x} = [\,]$, $\bar{l} = -\infty$

(1) $P = \max c^T x$, $Ax < b, x \geq 0$

(2) PUSH (P, LIST)

(3) While (LIST) $\neq \phi$

(3a) $P' =$ POP(LIST)

(3b) Solve P'
Let x^* be the optimal solution

(3c) If $x^* \in Z^n$ & $c^T x^* > l^*$
$l^* = c^T x^*$ & $\bar{x} = x^*$

(3d) Else if $\lceil c^T x^* \rceil > l^*$
$\exists\, x_j \notin Z$
$P_1' = P' \cup x_j \leq \lfloor x_j^* \rfloor$ → Floor
$P_2' = P' \cup x_j \geq \lceil x_j^* \rceil$
PUSH (P_1', LIST)
PUSH (P_2', LIST)

(4) return $\bar{x}^*$, l^*

---

* Bala's Algorithm: (Binary Optimization)

$\min c^T x$ → If we want max, then do $\min -c^T x$

s.t

$Ax \geq b$

$x \in \{0,1\}^n$

$0 \leq c_1 \leq c_2 \leq \cdots \leq c_n$

↑

- If this constraint is not satisfied, just reassign the variables.

If we want the coefficients, do $y_i = (1-x_i)$ and this just adds an additional constraint to the objective function.

⇒ Most optimal solution here is always all $\bar{x} = \bar{0}$ } This might not be feasible though (since all $c_i \geq 0$)

- Solve using trees based on assignments & not constraints

---

* (2) $\min 3x_1 + 5x_2 + 6x_3 + 9x_4 + 10x_5 + 10x_6$

s.t

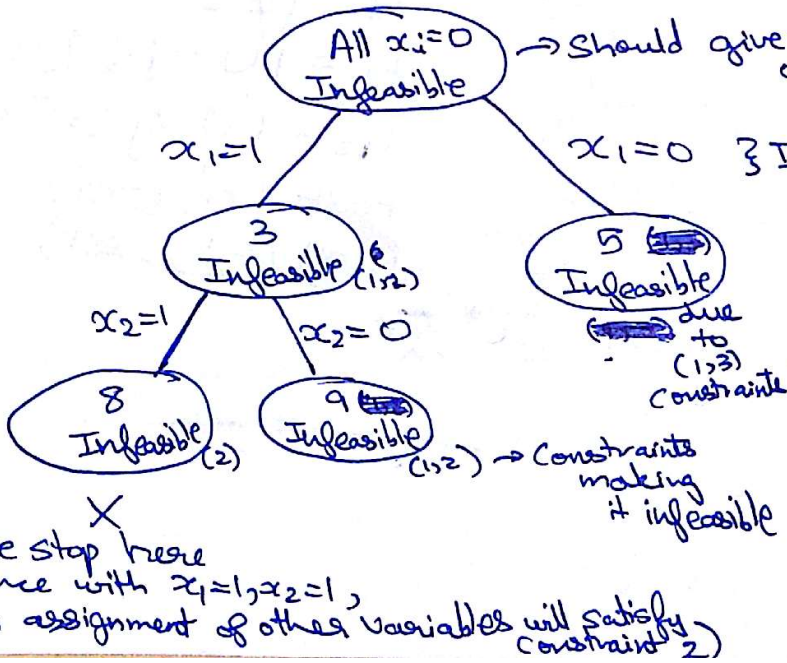
(1) $2x_1 + 6x_2 - 3x_3 + 4x_4 + x_5 - 2x_6 \geq 2$

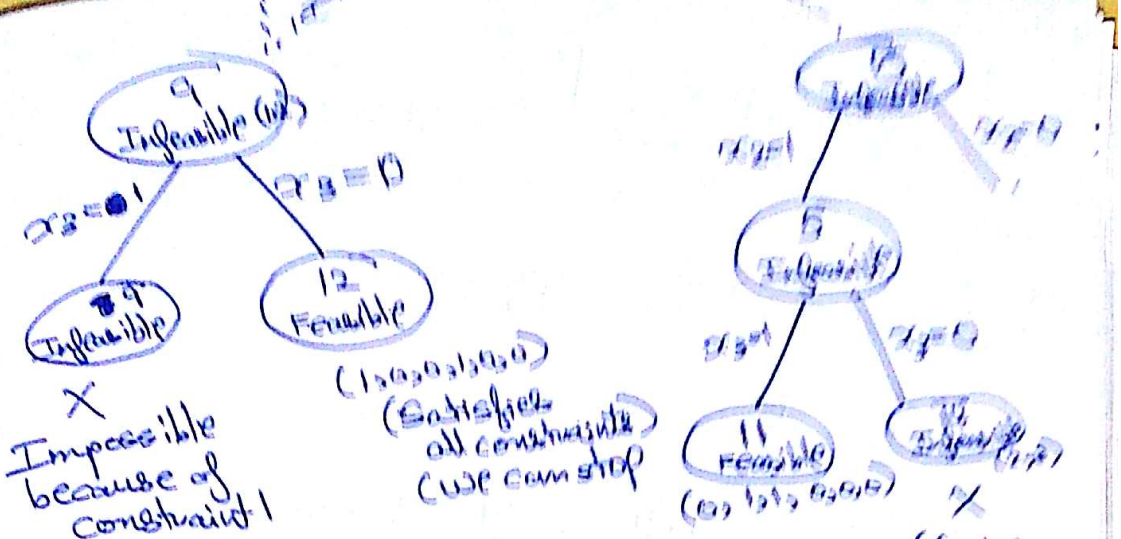
(2) $-5x_1 - 3x_2 + x_3 + 3x_4 - 2x_5 + x_6 \geq -2$

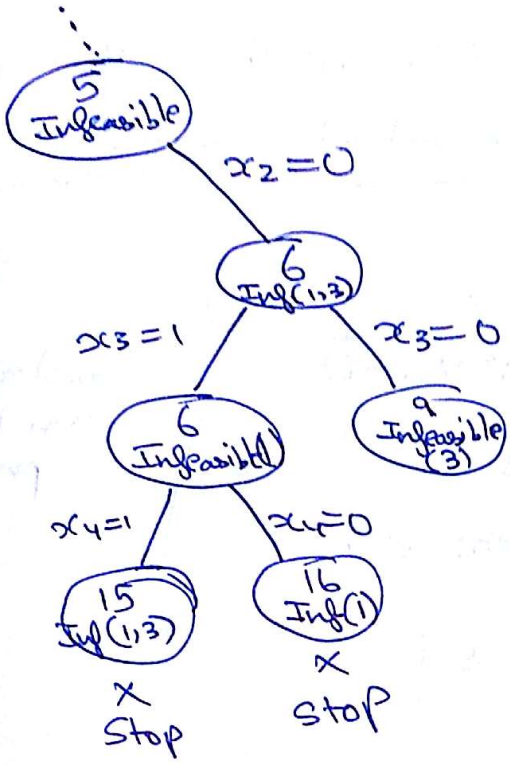
(3) $5x_1 - x_2 + 4x_3 - 2x_4 + 2x_5 - x_6 \geq 3$

$x_1, \ldots, x_6 \in \{0,1\}$

(Bala's Algorithm)







—— x ——

Non Linear Programming:

* (Fermat) Weber Problem:

$c_1, \ldots, c_m$ = facility centers

- $\min_{x \in R^n} \sum_{i=1}^{m} w_i \|x - c_i\|$ } Cannot be solved using LP

- $\min_{x \in R^n} f(x) = \max\left(\|x-c_1\|, \|x-c_2\|, \ldots, \|x-c_m\|\right)$
  → Can be modelled as a constrained programmas well.

ex: Minimize variance,

$$\min \sum_{i=1}^{k} x_i^2 \, Var(r_i)$$

$$s.t$$

$$\sum_{i=1}^{k} x_i - r_i \geq T$$

$$\sum_{i=1}^{k} x_i \leq B$$

} Constrained non linear program

Note: Local minima → Neighboring $\geq$ point
Strict Local minima → Neighboring $>$ point

Local Minimum:

- A vector $x^* \in R^n$ is an unconstrained local minimum of f if $\exists \varepsilon > 0$ such that $f(x^*) \leq f(x)$ $\forall$ x with $\|x - x^*\| \leq \varepsilon$

Global Minimum:

- A vector $x^* \in \mathbb{R}^n$ is an unconstrained global minimum of f if $f(x^*) \leq f(x) \; \forall x \in \mathbb{R}^n$.

---

* Gradient at Minima: (Intuition)

* $\Rightarrow$ Suppose f is differentiable, x^* = local minimum

$$f(x^* + \Delta x) - f(x^*) \geq 0$$

$$\nabla f(x^*)^T \cdot \Delta x \geq 0 \quad \rightarrow (1)$$

(Taylor expansion.)

$\forall \; \|\Delta x\| < \varepsilon$

$\nabla f(x^*)$: Gradient

- $$\nabla f(x^*) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x^*) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x^*) \end{pmatrix}$$

$\Rightarrow$ If we consider $\Delta x = (\delta, 0, 0, \ldots)$

then $\frac{\partial f(x^*)}{\partial x_1} \geq 0$ to satisfy (1)

$\Rightarrow$ If we consider $\Delta x = (-\delta, 0, 0, \ldots)$

then $\frac{\partial f(x^*)}{\partial x_1} \leq 0$ to satisfy (1)

$\therefore$ Since we know (1) is true for all Δx, such that $\|\Delta x\| < \varepsilon$, we can see that this is only possible if $\frac{\partial f(x^*)}{\partial x_i} = 0 \; \forall i$

- This is a necessary condition for minima but not sufficient (derivative = 0 even for maxima)

- If f is twice differentiable (x^* = Local minimum)

$$f(x^* + \Delta x) - f(x^*) \geq 0$$

$$\underbrace{\nabla f(x^*)^T \cdot \Delta x}_{0} + \frac{1}{2} \Delta x^T \nabla^2 f(x^*) \Delta x \geq 0$$

$\Rightarrow$ We just showed $\nabla f(x^*)^T \cdot \Delta x = 0$ for local minima/maxima.

- $\Delta x^T \nabla^2 f(x^*) \Delta x \geq 0$

$\ast \Rightarrow \nabla^2 f(x^*)$ is positive semi definite

↑ Hessian matrix

- This is ~~also~~ a ~~necessary~~ condition but ~~not~~ sufficient. (All these need to hold only if $f(x)$ is differentiable at x)

———— × ————

Proposition:

$\ast\ast$ Let x^* be an unconstrained local minimum of $f: \mathbb{R}^n \to \mathbb{R}^n$ & if f is continuously differentiable around x^*, then $\nabla f(x^*) = 0$. In addition, if f is twice continuously differentiable then $\nabla^2 f(x^*)$ is positive semi definite.

Proof:

Fix $d \in \mathbb{R}^n$, Let x^* be an unconstrained local minimum

$$g(\alpha) = f(x^* + \alpha d) - f(x^*)$$

$$\lim_{\alpha \to 0} \frac{f(x^* + \alpha d) - f(x^*)}{\alpha} \geq 0$$

$$\lim_{\alpha \to 0} \frac{g(\alpha)}{\alpha} = g'(0) = d^T \nabla f(x^*)$$

→ Derivative wrt α at $\alpha = 0$.

~~Taylor expansion of g(α)~~

- $d^T \nabla f(x^*) \geq 0$ & $-d^T \nabla f(x^*) \geq 0$

  ↑ Since for any d this should be true.

  $\Rightarrow \nabla f(x^*) = 0$ //

~~$g(\alpha) = \alpha \nabla f(x^*)^T d \Rightarrow g'(\alpha) = \nabla f(x^*)^T \cdot d$~~

~~1st order expansion~~

- Now, using Taylor series expansion of f around x^*

$$f(x^* + \alpha d) - f(x^*) = \alpha \nabla f(x^*)^T d + \frac{\alpha^2}{2} d^T \nabla^2 f(x^*) \cdot d + o(\alpha^2) \geq 0$$

$\left(\lim_{\alpha \to 0} \frac{o(\alpha^2)}{\alpha^2} = 0 \right)$

$$\Rightarrow \lim_{\alpha \to 0} \frac{f(x^* + \alpha d) - f(x^*)}{\alpha^2} \geq 0 \Rightarrow d^T \nabla^2 f(x^*) \cdot d \geq 0$$

$\nabla^2 f(x^*)$ is p.d

* Sufficient condition,

Let $f: \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable function & let x^* be s.t.

$\nabla f(x^*) = 0$ & $\nabla^2 f(x^*)$ is p.d (not semi)

Then x^* is strict unconstrained local minimum

(ex: $f(x) = x^3$ at $x = 0$ satisfies necessary conditions but is not local minima, (It does not satisfy sufficient condition))

———— x ————

Ex: $f(x) = x^2 - x^4$

$\nabla f(x) = 2x - 4x^3$

$x = 0, \pm 1/\sqrt{2}$

$\nabla^2 f(x) = 2 - 12x^2$

↑

At $x = 0$ it's a local minimum

At $x = \pm 1/\sqrt{2}$ it's a local maximum.

---

* Iterative Descent: (To reach local minimum)

$$\min_{x \in \mathbb{R}^n} f(x)$$

- $x_0 \to x_1 \to x_2 \cdots \to x_n$ } Initial guess x_0

$f(x_0) > f(x_1) > f(x_2) \cdots > f(x_n)$, if this is satisfied, we can hope that we will converge at some point.

$$* \Rightarrow x_{k+1} = x_k + \alpha_k \cdot d_k$$

$$* \quad d_k = \frac{-\nabla f(x_k)}{\|\nabla f(x_k)\|}$$

(Move in opposite direction of gradient)

Steepest descent.

→ Normal vector

$$\Rightarrow f(x_{k+1}) \approx f(x_k) - \frac{\alpha_k \nabla f(x_k)^T \nabla f(x_k)}{\|\nabla f(x_k)\|}$$

↑ Approximate by linear function

$$f(x_{k+1}) \approx f(x_k) - \alpha_k \|\nabla f(x_k)\|$$

---x---

Note: Fixing step size α_k is bad since it might just keep oscillating and not converge.

---x---

ex: $f(x,y) = 2x^2 + y^2$

$\nabla f(x,y) = \begin{bmatrix} 4x \\ 2y \end{bmatrix}$

- $x_0 = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$, $x_1 = x_0 - \nabla f(x_0)$ → If our step size is incorrect

$= \begin{bmatrix} -30 \\ -10 \end{bmatrix}$

$x_2 = \begin{bmatrix} 90 \\ 10 \end{bmatrix}$, $x_3 = \begin{bmatrix} -270 \\ -10 \end{bmatrix}$

- We can end up even performing ascent, step size should be set well.

———— x ————

Note: We can also try to optimize step size α_0

$x' = x + \alpha_k \cdot d_k$ (variable, known)

$f(x') = f(x + \alpha_k \cdot d_k)$ → Try to minimize this, at times this might be simpler than minimizing $f(x)$ directly

At a known x

(At each step we try to minimize the resulting $f(x')$)

(This usually takes lots of steps to reach min)

$\min_{x \in \mathbb{R}^n} f(x)$ ←

x ————

*Gradient Methods:

* • $f(x_{k+1}) = f(x_k) + \alpha \nabla f(x_k)^T d_k + o(\alpha)$

(Taylor expansion)

($\alpha > 0$ → Step size)

($\|d_k\| = 1$)

- $f(x_{k+1}) - f(x_k) < 0$, $\alpha \nabla f(x_k)^T d_k < 0$

↓ Minimize $f(x)$

$\boxed{\nabla f(x_k)^T d_k < 0}$

Methods satisfying this are called gradient methods.

1) $\nabla f(x_k)^T d_k \geq -\|\nabla f(x_k)\| \|d_k\|$ (Schwartz inequality)

$\geq -\|\nabla f(x_k)\|$

* a) $\boxed{d_k = \frac{-\nabla f(x_k)}{\|\nabla f(x_k)\|}}$ $\left(\nabla f(x_k)^T d_k = -\|\nabla f(x_k)\|\right)$

- This is called Steepest Descent (Greedy Approach) Method.

---

b) $d_k = -D_k \nabla f(x_k)$ (D_k = Identity for steepest descent)

- $D_k = (\text{Hessian})^{-1}$ in Newton's method.
- This generalization is a gradient method if D_k is +ve definite.

Note: To solve for x such that $f(x) = 0$, (a<b)

* if f is a polynomial, start with a, b such that $f(a) > 0$ & $f(b) < 0$, now perform binary search, $c = \frac{a+b}{2}$, if $f(c) > 0$, $a = c$ else $b = c$

Newton Rhapson method to solve for x

$f(x) = 0$, f = Polynomial (Start with guess x_0)

** $f(x) = f(x_0) + f'(x_0)(x - x_0) = 0$

2) $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$

↓ Approximation for x such that $f(x_1) \approx 0$

⇒ If $f(x_1) = 0$, done else $x_0 = x_1$ and repeat $\left(x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}\right)$

- If α is the root,

$$0 = f(\alpha) = f(x_k) + f'(x_k)(\alpha - x_k) + \frac{1}{2} f''(x_k)(\alpha - x_k)^2$$

some error term E_k

$$\Rightarrow \left(\frac{f(x_k)}{f'(x_k)} - x_k + \alpha\right) = \frac{-\frac{1}{2} \cdot f''(x_k)}{f'(x_k)} \cdot (\alpha - x_k)^2$$

$\alpha - x_{k+1} = E_{k+1}$, $|\alpha - x_k| = E_k$

$$\Rightarrow |E_{k+1}| = \frac{1}{2} \frac{f''(x_k)}{f'(x_k)} \cdot |E_k|^2$$

- This method converges very quickly given that we start at a good x_0 guess where $|\epsilon_0| < 1$.

⇒ To solve a minimization problem such as $\min f(x)$, we consider $g(x) = f'(x)$ and then get $g(x) = 0$.

Newton's Method

∴ Update rule ⇒ $x_{k+1} = x_k - \frac{g(x_k)}{g'(x_k)}$

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

- For n-dimensional functions, gradients and hessians are considered,

$$* x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \cdot \nabla f(x_k)$$

Newton's Method

- Comparing this with $x_{k+1} = x_k + \alpha_k \cdot d_k$, we get, Choose,

$$\alpha_k = 1, \quad d_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

$\Rightarrow$ Criteria for gradient descent is $\nabla f(x_k)^T d_k < 0$

$$\Rightarrow -\nabla f(x_k)^T \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) < 0$$

$$\Rightarrow \nabla f(x_k)^T \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) > 0$$

- When the Hessian is the definite Newton Rhapson method is a gradient descent.

---

ex: $f(x,y) = x^2 + y^2$, $\nabla f(x) = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$

$x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $\quad \nabla^2 f(x) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$

$$x_1 = x_0 - \left[\nabla^2 f(x_0)\right]^{-1} \cdot \nabla f(x_0)$$

Newton Rhapson method.

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Ex: $f(x,y) = 2x^2 + y^2$, $x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$$\nabla f(x) = \begin{bmatrix} 4x \\ 2y \end{bmatrix}$$

$$\nabla^2 f(x) = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} \quad , \left(\nabla^2 f(x)\right)^{-1} = \begin{bmatrix} 1/4 & 0 \\ 0 & 1/2 \end{bmatrix}$$

$$x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1/4 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$, Much faster than steepest descent.

$x_2 = \begin{bmatrix} -1 \\ 1/2 \end{bmatrix} - \begin{bmatrix} 1/2 & 0 \\ 0 & 1/4 \end{bmatrix} \begin{bmatrix} 4 \\ 1 \end{bmatrix}$

Note: In Newton's method we use quadratic approximation so all quadratic functions are minimized well.

3) Modified Newton's Method:

- Computing Hessian & inverse is hard so compute it once and use the same for a few iteration ($\nabla f(x_k)$ is computed at each step)

4) Discretized Newton's Method:

- $D_k = (H(x_k))^{-1}$, H is a symmetric approximation of $\nabla^2 f(x_k)$ using finite difference methods. ($H \Rightarrow$ p.d matrix)

5) Diagonally scaled: (steepest descent method)

$$D_k = \begin{bmatrix} \alpha_1 & \cdots & 0 \\ \vdots & \alpha_2 & 0 \\ 0 & \cdots & \alpha_n \end{bmatrix}$$

(Considers each dimension as independent for an approximation)

$$\alpha_k = \left(\frac{\partial^2 f(x_k)}{\partial x^2} \right)^{-1}$$

* (Similar to newton's method but performed independently)

---

* Neural Nets: (Gauss Newton's Method)

(Apply at each step & then backprop)

* Considers a neural network, (x = parameters of network)
* $\min_{x \in R^n} f(x) = \frac{1}{2} \|g(x)\|^2$ (n parameters)

$$\|g(x)\|^2 = \sum_{i=1}^{m} g_i(x)^2$$

g_i: i^{th} datapoint

- g has m dimensions (Total weights)
- x has n dimensions (m data points)
- Taylor (linear approx) → Errors

$$\Rightarrow \tilde{g}(x, x_k) = g(x_k) + \nabla g(x_k)^T (x - x_k)$$

$$\nabla g(x_k) = \left[\nabla g_1(x_k) \quad \nabla g_2(x_k) \cdots \nabla g_m(x_k) \right]$$

$$\therefore \frac{1}{2} \|\tilde{g}(x)\|^2 = \frac{1}{2} \left\{ \|g(x_k)\|^2 \right\} + \left\{ (x - x_k)^T \nabla g(x_k) \cdot g(x_k) \right\} + g(x_k)^T \cdot \nabla g(x_k)^T \cdot (x - x_k) + (x - x_k)^T \nabla g(x_k) \nabla g(x_k)^T \cdot (x - x_k)$$

($n \times m$; c_1; c_2)

$$\alpha = (x - x_k)^T$$

$$\frac{1}{2} \|\tilde{g}(x)\|^2 = \frac{1}{2} (c^2 + 2\alpha c_1 + \alpha^2 c_2)$$

$$c_1 = \nabla g(x_k) \cdot g(x_k) \qquad c_2 = \nabla g(x_k) \cdot \nabla g(x_k)^T$$

* Now, set $x_{k+1} - x_k = -C_2^{-1} \cdot C_1$

(Newtons method similar)

$$= -\left(\nabla g(x_k) \cdot \nabla g(x_k)^T\right)^{-1} \cdot \left(\nabla g(x_k) \cdot g(x_k)\right)$$

$$\Rightarrow \nabla f(x) = \sum_{i=1}^{n} g_i(x) \cdot \nabla g_i(x) \quad \rightarrow (1)$$

$$= \nabla g(x) \cdot g(x) = C_1$$

$$\alpha_k = 1$$

- Comparing (1) with $x^*_{k+1} = x_k + \alpha_k d_k$

$$d_k = -\left(\nabla g(x_k) \nabla g(x_k)^T\right)^{-1} \cdot \left(\nabla g(x_k) \cdot g(x_k)\right)$$

$\Rightarrow$ This is gradient descent if, (Gradient methods)

$\nabla g(x_k) \cdot \nabla g(x_k)^T$ is positive definite

$\Rightarrow$ This $d_k = -\left(\nabla g(x_k) \cdot \nabla g(x_k)^T\right)^{-1}$ selection is from Gauss Newton's method.

- Now, $\nabla^2 f(x) = \nabla g(x) \cdot \nabla g(x)^T + \sum_{i=1}^{n} g_i(x) \cdot \nabla^2 g(x)$

($n \times n$ matrix)

(Our d_k is like an approximation)

$\Rightarrow$ If we use $\left(\nabla^2 f(x)\right)^{-1}$ then its newtons method but we are simplifying it and considering d_k as above.

(This method works well in Neural Networks)

Note:
- $\nabla g(x_k) \cdot \nabla g(x_k)^T$ must be invertible
- Rank of this is bounded by $\min(m,n)$ but we want it to be atleast n, so we need atleast n datapoints ($m \geq n$)

- Geometric Mean $= (x_1 \cdot x_2 \cdot \dots \cdot x_n)^{1/n}$
- Arithmetic Mean $= \frac{x_1 + x_2 + \dots + x_n}{n}$

$x_1, x_2, \dots, x_n > 0$

$\Rightarrow$ Show G.M $\leq$ A.M

- Consider $y_i = \ln x_i$

$x_i = e^{y_i}$

- $e^{\frac{y_1 + \dots + y_n}{n}} \leq \frac{e^{y_1} + e^{y_2} + \dots + e^{y_n}}{n}$

$S = y_1 + \dots + y_n$, $y_n = S - y_1 - y_2 - \dots$

$\Rightarrow n e^{S/n} \leq e^{y_1} + e^{y_2} + \dots + e^{y_n}$

$\Rightarrow n e^{S/n} \leq e^{y_1} + e^{y_2} + \dots + e^{S - y_1 - y_2 \dots - y_{n-1}}$

- Consider gradient, Minimize,

? $\Rightarrow e^{y_i} = e^{S - (y_1 + \dots + y_n)}$

$\rightarrow$ We get $y_i = S/n$ for minimization and then LHS = RHS