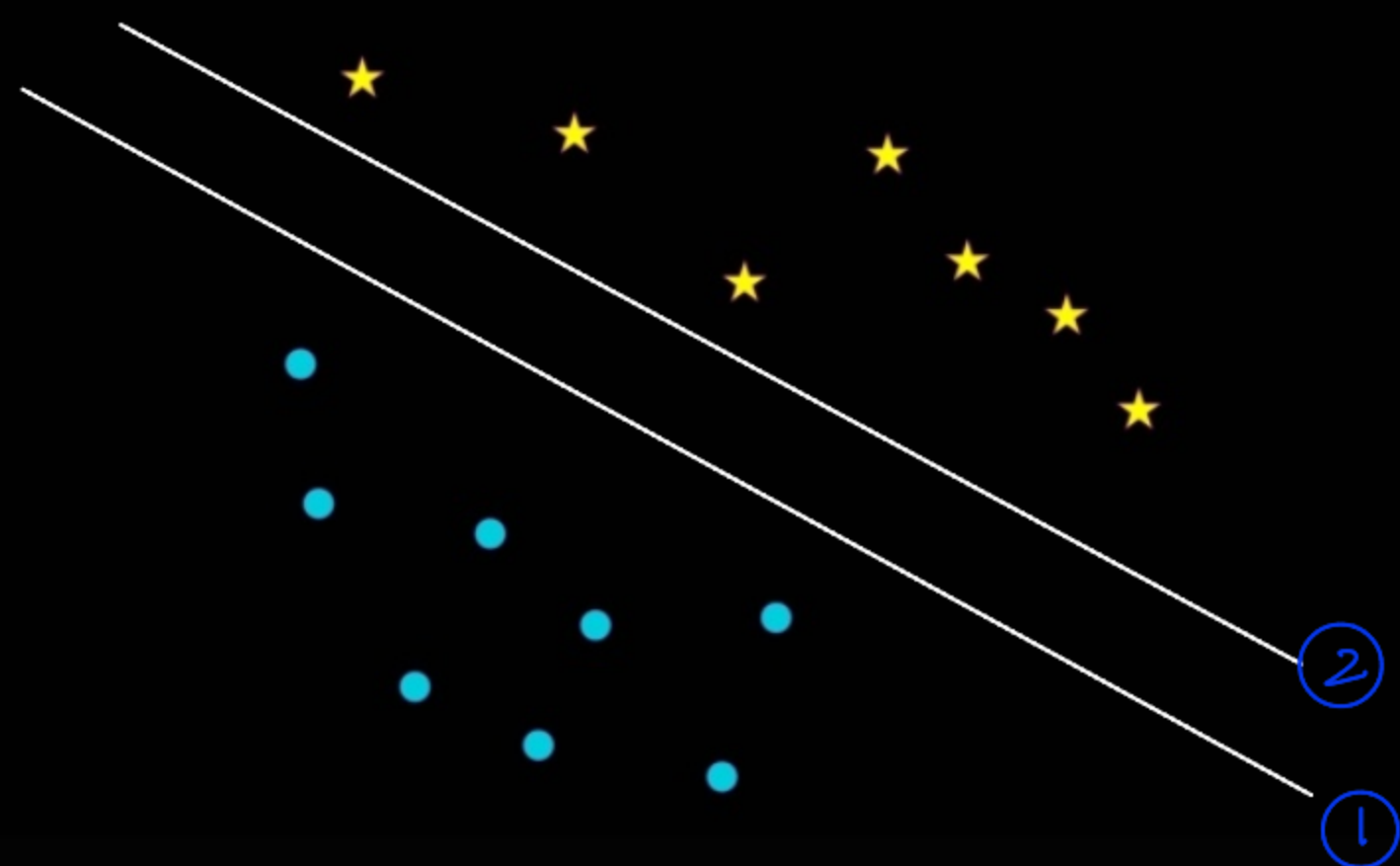


Support Vector Machine

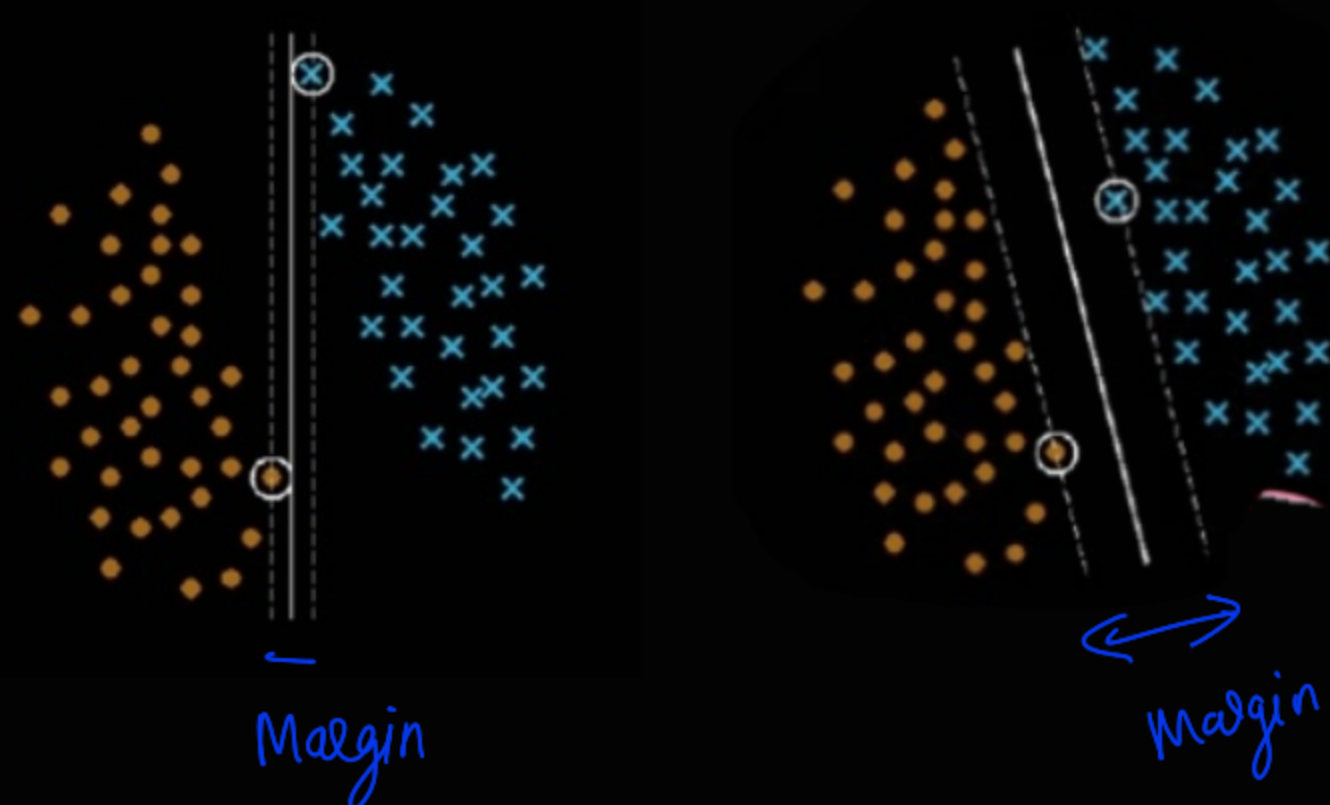
→ Most popular model

In this lecture & beyond we are following the notation $W^T X + b$
 $W \rightarrow$ column vector
 $X \rightarrow$ column vector
 $b \rightarrow$ bias

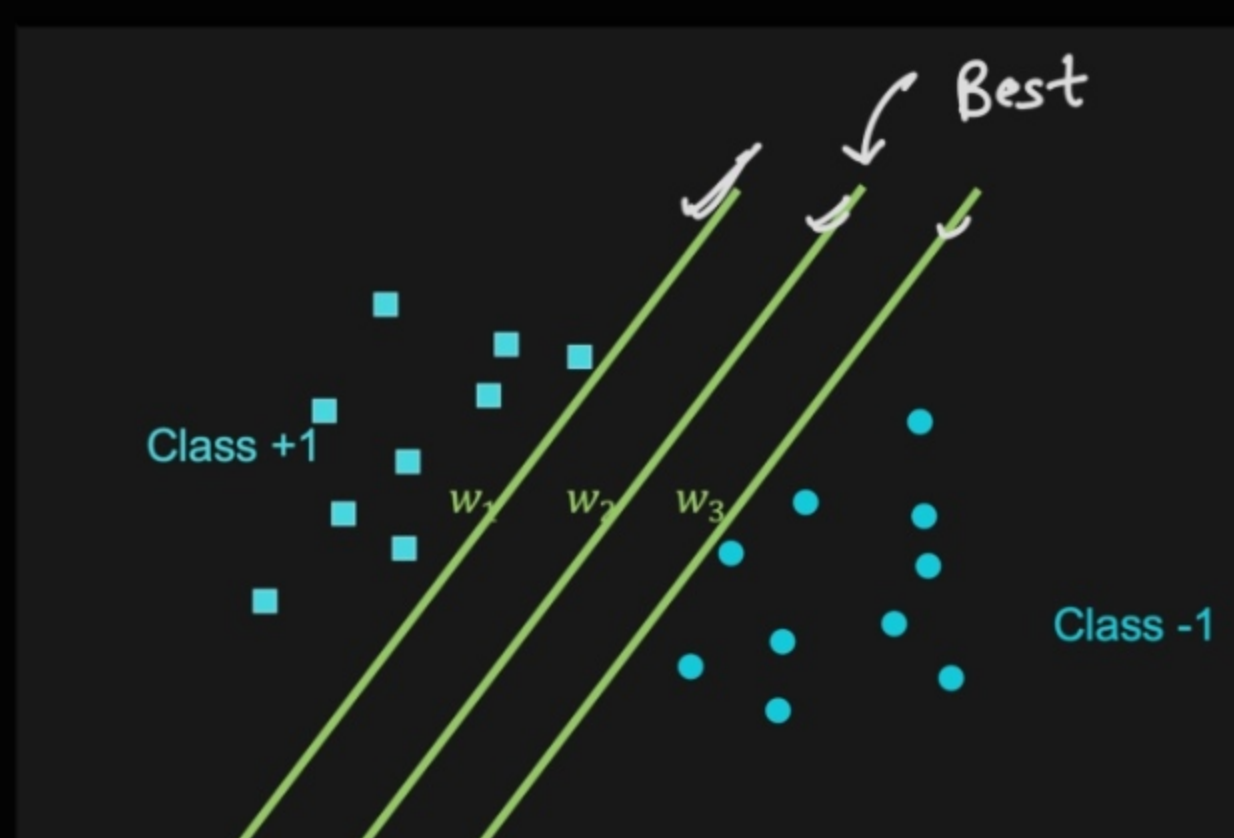
Suppose we are given these data points from two different classes and want to find a linear classifier that separates them.



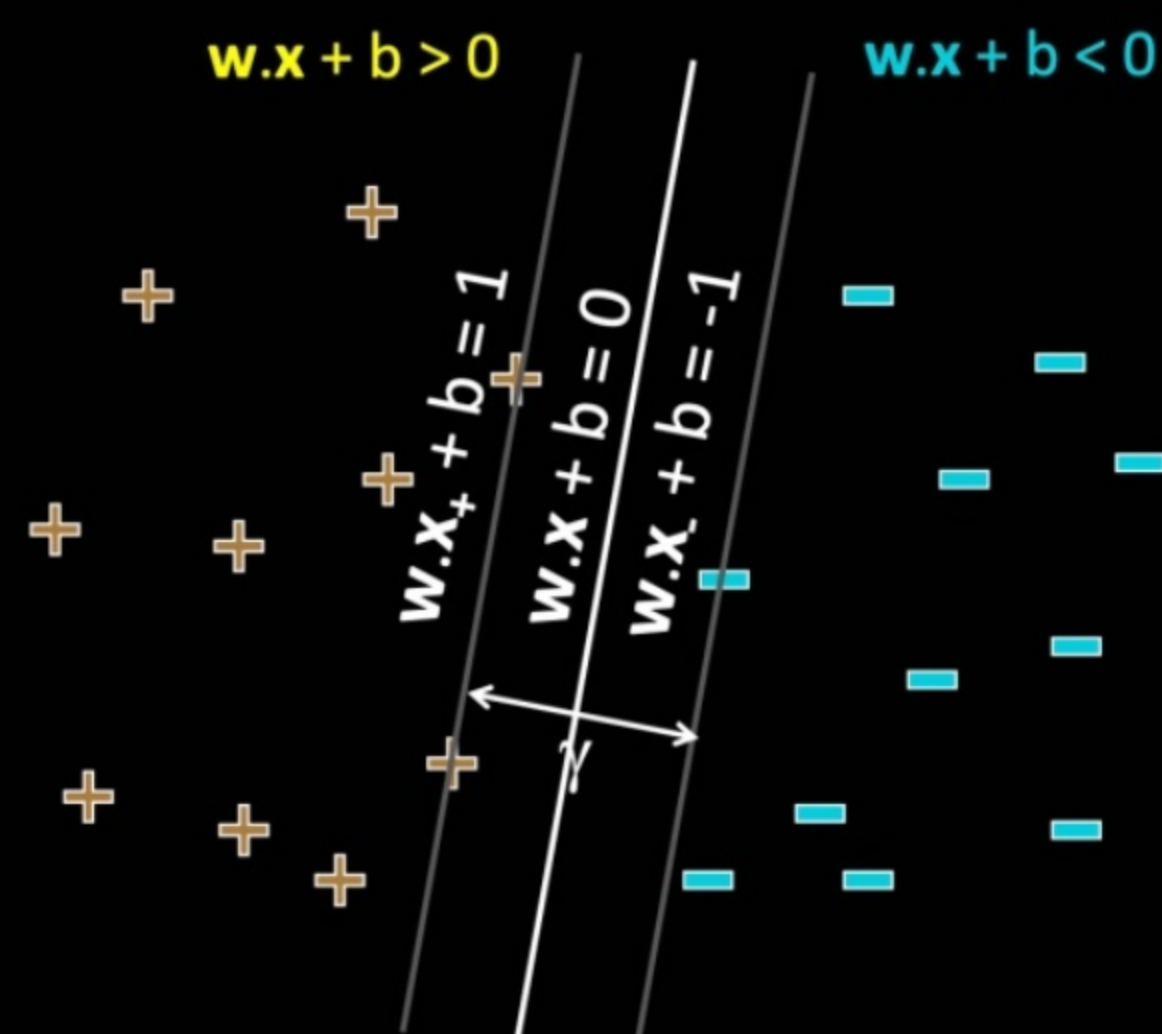
→ which is better
 → SVM gives best classifier
 → MARGIN → Sense of distance to closest training point
 → SVM maximises margin



This is best



Mathematical formulation



→ Given an hyperplane, we can divide & multiply by any constant & it won't change

$$\begin{aligned} W^T X + b &= 0 \\ \frac{W^T X}{K} + \frac{b}{K} &= 0 \\ K W^T X + K b &= 0 \end{aligned} \quad \left. \begin{array}{l} \rightarrow \text{same hyperplane} \\ \text{distance only scaled} \end{array} \right\}$$

→ Now, divide by C [$C \rightarrow$ margin for original $W^T X + b = 0$]

$$\begin{aligned} W^T X + b &= C \\ W^T X + b &= 0 \\ W^T X + b &= -C \end{aligned}$$

After scaling i.e. divide by C

$$\begin{aligned} W^T X + b &= +1 \\ W^T X + b &= 0 \\ W^T X + b &= -1 \end{aligned}$$

Since the normal to the hyperplane is same W does not change only bias changes.
 (parameter space without bias as parameters)

OBSERVATION

→ After scaling, $W^T X + b = 1$ is satisfied by the (+) training point which is closest to the separating hyperplane ($W^T X + b = 0$)
 → All other +ve training point $W^T X + b > 1$
 → After scaling, $W^T X + b = -1$ is satisfied by the (-) training point which is closest to the separating hyperplane ($W^T X + b = 0$)

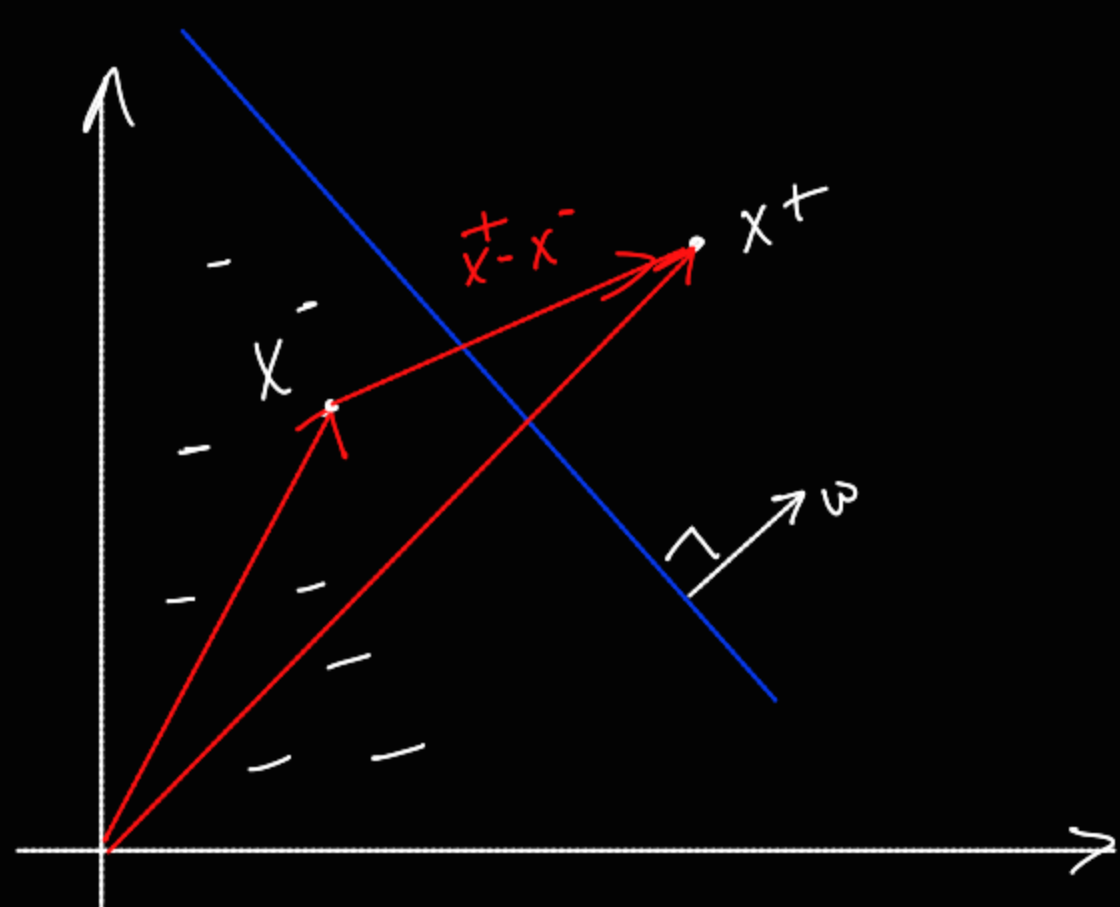
→ The closest points are called support vectors.

→ (classifier) won't change even if we remove points other than support vectors

$$\left. \begin{array}{l} w^T x_n + b \geq 1 \quad \forall y_n = +1 \\ w^T x_n + b \leq -1 \quad \forall y_n = -1 \end{array} \right\} y_n (w^T x_n + b) \geq 1 \rightarrow \text{Compact form}$$

→ Our objective is to maximise margin & for every point the above condition is satisfied.

METHOD 1



→ We want distance $x^+ - x^-$ in the direction of w

→ dot product & divide by $\|w\|$

this distance is not affected by scaling of w

$$\frac{w^T}{\|w\|} (x^+ - x^-)$$

remember $x = z^T b$ 2-normal vectors.

→ x^+ → closest (+) training point to hyperplane

→ x^- → closest (-) training point to hyperplane

$$\begin{array}{l} w^T x^+ + b = 1 \\ w^T x^- + b = -1 \end{array} \Rightarrow \begin{array}{l} w^T x^+ = 1 - b \\ w^T x^- = -1 - b \end{array} \Rightarrow \frac{w^T (x^+ - x^-)}{\|w\|} =$$

$$\frac{1 - b - (-1 - b)}{\|w\|} = \frac{2}{\|w\|}$$

Induction

→ scale just enough so that the closest + & - points satisfy ① ⇒ then Margin can be calculated from ③

→ if there is no scaling (i.e. $\|w\|=1$) then distance $\neq \frac{2}{\|w\|}$

since ① won't be satisfied without scaling.

→ scaled $w^* = \frac{w}{c}$ $\|w_s\| = \frac{\|w\|}{c}$

Margin without scaling

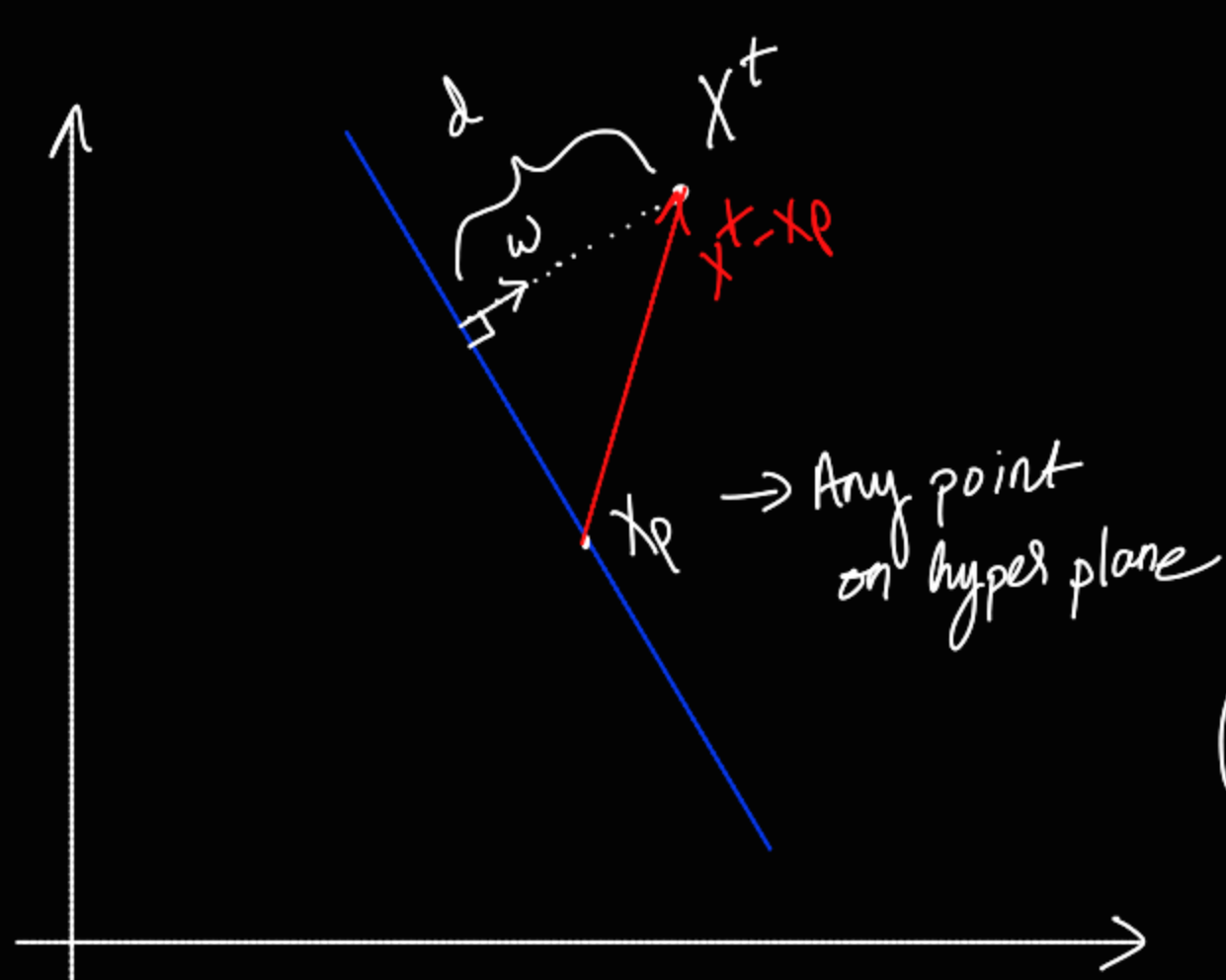
$$\begin{array}{l} w^T x^+ + b = c \\ w^T x^- + b = -c \end{array} \Rightarrow \begin{array}{l} w^T x^+ = c - b \\ w^T x^- = -c - b \end{array}$$

$$\Rightarrow \frac{w^T (x^+ - x^-)}{\|w\|} = \frac{c - b + c + b}{\|w\|} = \frac{2c}{\|w\|}$$

→ general equation ④

→ This will be satisfied even without scaling. i.e. $\|w\|=1$

METHOD 2



$$d = \frac{w^T}{\|w\|} (x^+ - x_p)$$

$$d = \frac{1 - b - (-b)}{\|w\|} = \frac{1}{\|w\|}$$

$$\begin{array}{l} w^T x^+ + b = 1 \\ w^T x_p + b = 0 \end{array}$$

without scaling

$$d = \frac{c}{\|w\|}$$

(d) distance from any general point x

$$= \frac{w^T x + b}{\|w\|}$$

OPTIMISATION

→ Maximise w, b

$$\frac{1}{\|w\|}$$

satisfying $y_n (w^T x_n + b) \geq 1 \quad \forall n$

without scaling

→ Maximise w, b

$$\frac{c}{\|w\|}$$

satisfying $y_n (w^T x_n + b) \geq c \quad \forall n$

for applying G.D we change target to $\min_{w,b} \frac{1}{2} \|w\|^2$ satisfying $y_n(w^T x_n + b) \geq 1 \forall n$

For Mathematical convenience

Argmin $\frac{1}{2} w^T w$ | $y_n(w^T x_n + b) \geq 1 \forall n$ } constrained optimisation problem

This is called hard margin or hard SVM.

if no constrain then model will put $w=0$ with constrain that the distance has to be ± 1 to closest points

→ What if data is not linearly separable? → Infeasible → No solution

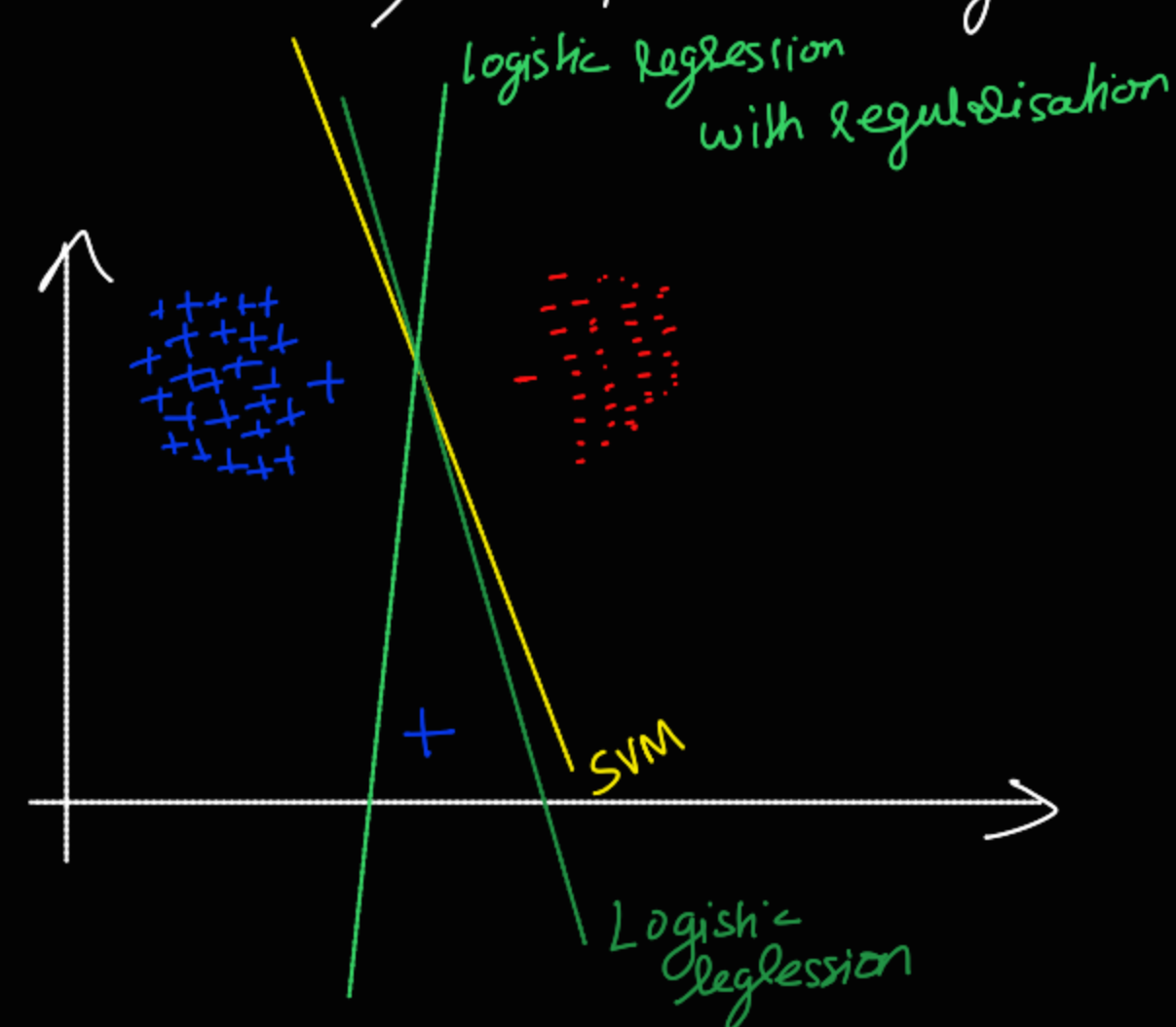
→ if linearly separable then classifier depends only on support vectors ($x^+ \& x^-$)

→ if data is linearly separable ⇒ unique solution

→ In logistic regression if data is linearly separable (even with outliers) then, it always fits the hyperplane to correctly classify.

→ loss = $-\ln(\sigma(y_n w^T x_n))$

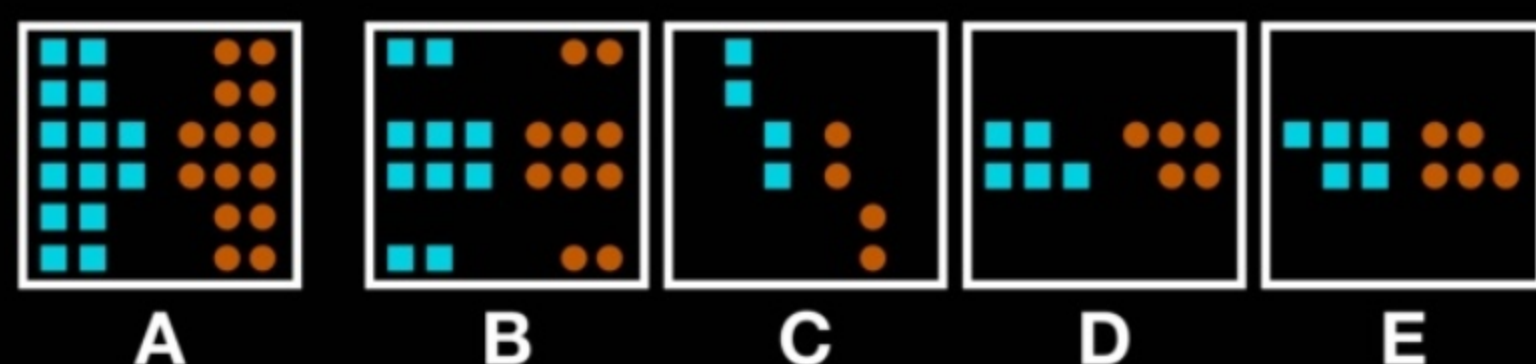
→ if bulk of the data is in one region, then logistic regression can sacrifice outlier to improve the likelihood for the majority? No. Since Logistic regression without regularisation will just correctly classify & then increase weights to improve likelihood rather than sacrifice



→ But Logistic regression with L1 or L2 regularisation may sacrifice the outliers to improve likelihood since it cannot do this by increasing weights.

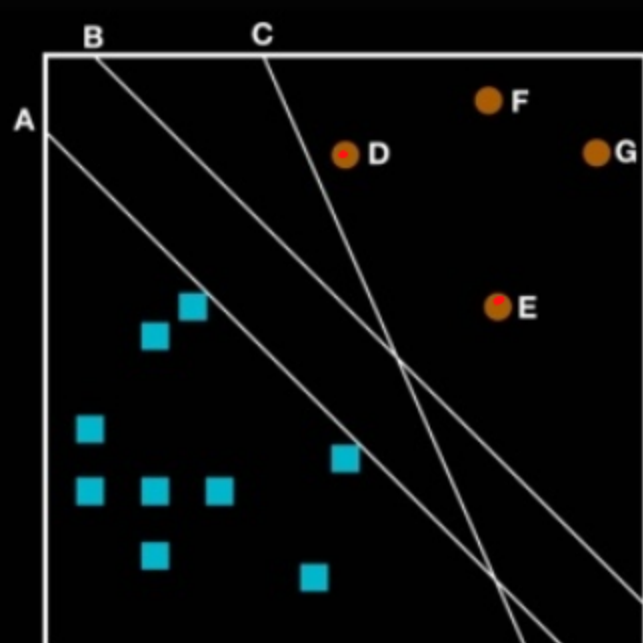
→ hard SVM will find the unique solution if data is linearly separable even with outliers. → Hard margin SVM can never miss classify

(2) (3pts) Hard SVM is run on each of the following training sets. Which (if any) of the training sets B-E will end up with the same decision boundary as training set A? Circle all that apply.



B, C, E

Question:



Find support vectors on the side D & E

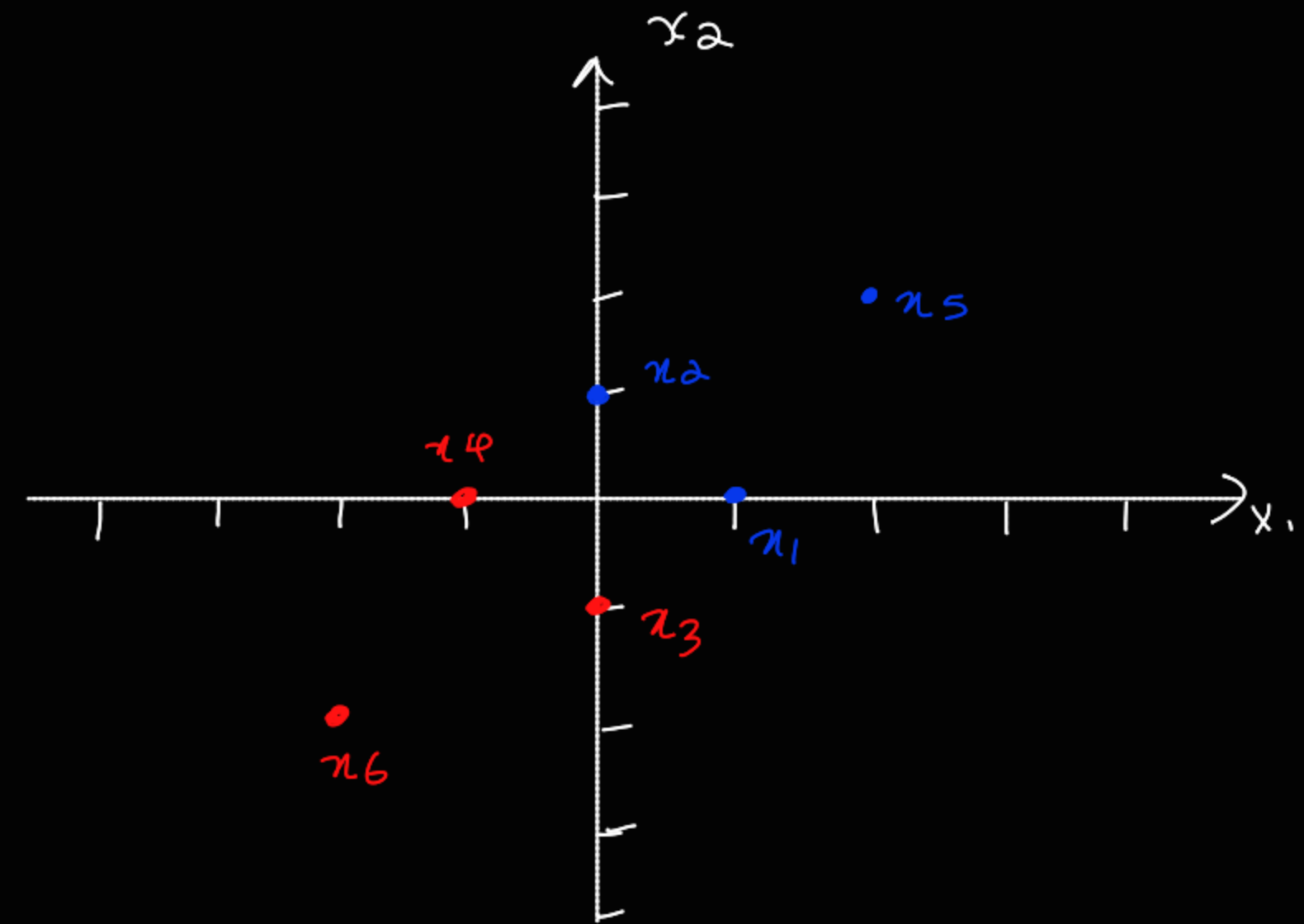
(b) Which (if any) of the decision boundaries could be returned after optimizing the hard SVM objective (with a bias parameter) on this dataset? Mark all that apply.

46. [1 points] Suppose you are given a binary labelled data set that is linearly separable. Using an SVM, you find a hyperplane H that separates the labels with maximum margin γ . Is it possible that there is another hyperplane, different from H , that also separates the labels with the same margin γ ?

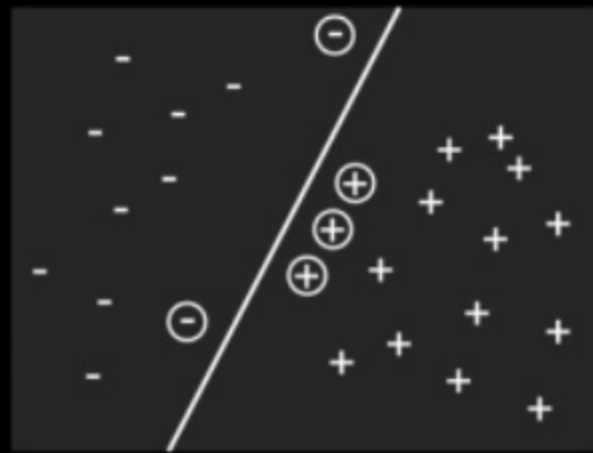
- (a) Yes
(b) No

No, hard SVM result is always unique for linearly separable data & No solution for linearly non separable

Q.17	Consider the dataset with six datapoints: $\{(x_1, y_1), (x_2, y_2), \dots, (x_6, y_6)\}$, where $x_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $x_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$, $x_4 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$, $x_5 = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$, $x_6 = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ and the labels are given by $y_1 = y_2 = y_5 = 1$, and $y_3 = y_4 = y_6 = -1$. A hard margin linear support vector machine is trained on the above dataset. Which ONE of the following sets is a possible set of support vectors?
(A)	$\{x_1, x_2, x_5\}$
(B)	$\{x_3, x_4, x_5\}$
(C)	$\{x_4, x_5\}$
(D)	$\{x_1, x_2, x_3, x_4\}$ ✓



45. [3 points] Consider a binary classification problem in a 2-dimensional instance space $\mathcal{X} = \mathbb{R}^2$. You are given a linearly separable training set. You run the hard-margin SVM algorithm and obtain the separating hyperplane below (support vectors are circled):

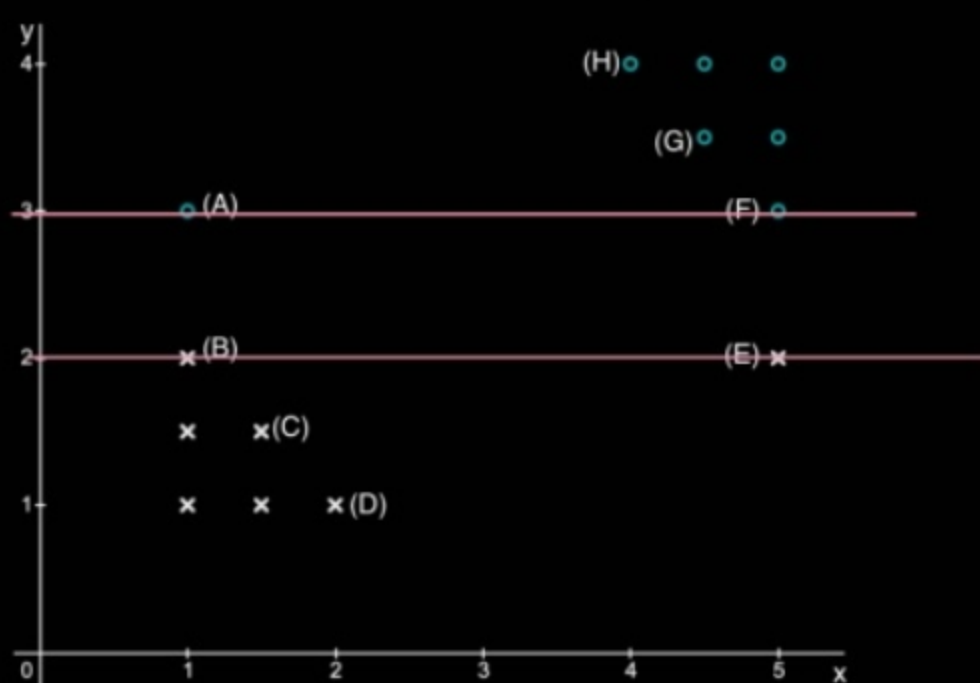


What is the smallest number of data points that would have to be removed from the training set in order for the SVM solution to change?

- (a) 1 ✓
(b) 2
(c) 3
(d) 4

removing support vectors will change the solution

3. SVMs. Consider the 2D datasets shown in the figure below. We want to classify the red circles and the black crosses using SVMs.



- (b) Let's first consider using a linear hard-margin SVM.
i. (3 points) What's the learned decision boundary?

Solution: $y =$

- ii. (2 points) Select all points that are support vectors.
☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G ☐ H

i) $y = 2.5$
ii) A F B E

(g) [3 pts] If a hard-margin support vector machine tries to minimize $|w|^2$ subject to $y_i(X_i \cdot w + \alpha) \geq 2$ instead, what will be the width of the slab (the point-free region bracketing the decision boundary)?

- ☐ $\frac{1}{\|w\|}$ ☐ $\frac{4}{\|w\|}$
☐ $\frac{2}{\|w\|}$ ☐ $\frac{1}{2\|w\|}$

$\frac{2c}{\|w\|}$ | here $c=2 \Rightarrow \frac{4}{\|w\|}$

(b) Recall the hard-margin SVM objective:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2} \|w\|_2^2 \\ &\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 \quad \forall i \end{aligned}$$

The constraints says that the (functional) margin of each example is at least 1. If we change the constraint to require the margin to be at least c ($c > 0$), i.e. solving

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2} \|w\|_2^2 \\ &\text{s.t.} \quad y_i(w^T x_i + b) \geq c \quad \forall i \end{aligned}$$

i. (3 points) Would it change the separating hyperplane? Why or why not?

no, orientation is same

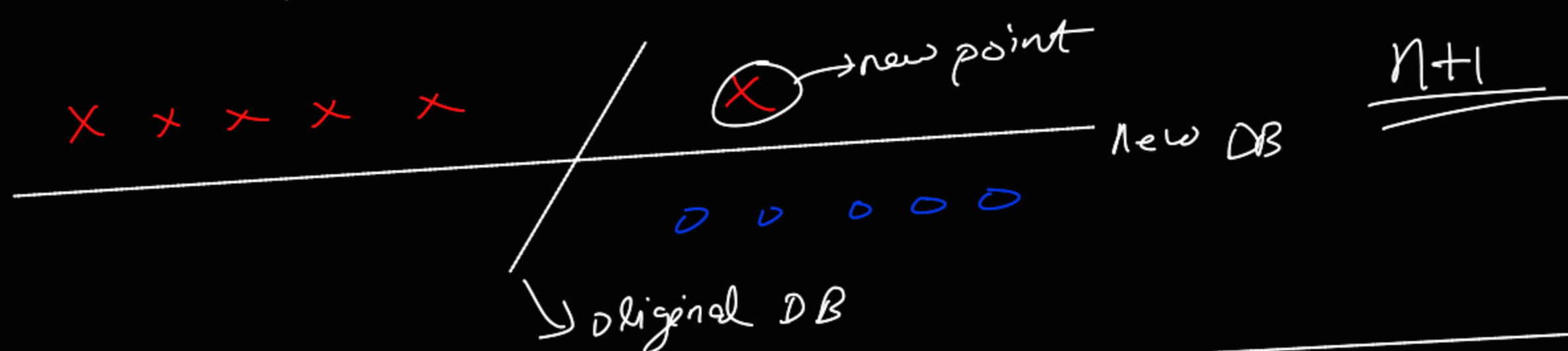
$$w^* = \frac{w^l}{c} \quad \text{This is scaling.}$$

ii. (3 points) Let w^* be the solution of the original hard-margin SVM, and w' be the solution of the modified problem with margin at least c . Write an expression of w' using w^* .

$$w' = c w^*$$

$$\begin{aligned} w^* &= \text{scaled} \rightarrow y_n(w^* x_n + b) \geq 1 \\ w^l &= \text{unscaled} \rightarrow y_n(w^l x_n + b) \geq c \end{aligned}$$

if you have n points in dataset (> 1000) & 2 support vectors, then what is the maximum no. of support vectors possible by adding just one point.



Suppose an SVM has $w = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ and $b = -1$. What is the actual distance between the two planes defined by $w^T x + b = -1$ and $w^T x + b = 1$?

$$\frac{2}{\|w\|}$$

$$\begin{aligned} d &= \frac{2c}{\|w\|} \quad \text{but } c=1 \\ &\text{since already scaled} \\ \Rightarrow \frac{2}{\sqrt{9+4+1}} &= \frac{2}{\sqrt{14}} \end{aligned}$$

- (A) $\frac{1}{\sqrt{14}}$ (B) $\frac{2}{\sqrt{14}}$ (C) $\frac{\sqrt{11}}{2}$ (D) $\sqrt{11}$ (E) none of the above

An SVM is trained with a linear kernel to do hard margin classification, i.e. the slack variables are not used, no misclassifications are allowed and the formulation is simply

$$\begin{aligned} &\min_{w, \{\xi_i\}} \quad \frac{1}{2} \|w\|^2 \\ &\text{subject to} \quad y_i \cdot (w^T x_i) \geq 1, \forall i. \end{aligned}$$

We know the support vectors and the margins after the training. Now, we introduce a new data point and retrain. Select the correct statements:

- ☒ The margin will never change if the new data point is on the correct side of the original decision boundary and inside the margin.
- ☒ The margin will not change if the new data point is on the correct side of the original decision boundary and outside of the margin.
- ☒ The margin will change if the new data point is in between the original margin regardless of its positioning with respect to the decision boundary.
- ☒ The margins will not change if the new data point is within the original margin and on the correct side of the original decision boundary.
- ☒ The number of support vectors will not change regardless of the position of the new data point.

What are support vectors:

- ☐ The examples farthest from the decision boundary.
- ☒ The only examples necessary to compute $f(x)$ in an SVM.
- ☐ The class centroids.
- ☒ All the examples that have a non-zero weight α_k in a SVM.

minimize $\frac{1}{2} \|w\|_2^2$

subject to $y^{(i)}(w^T x^{(i)} + b) \geq 1$

Convert to
KKT
Template

Min $\frac{w^T w}{2}$
subject to $1 - y_n(w^T x_n + b) \leq 0$

$N = \# \text{ datapoints}$

$\Rightarrow L(w, b, \mu) = \frac{w^T w}{2} + \sum_{n=1}^N \mu_n (1 - y_n(w^T x_n + b))$

KKT conditions

① $\Delta L(w, b, \mu) = 0 \Rightarrow$

$\frac{\partial L(w, b, \mu)}{\partial w} = w^T - \sum_{n=1}^N \mu_n y_n x_n^T = 0$

$\Rightarrow w = \sum_{n=1}^N \mu_n y_n x_n$

$\frac{\partial L(w, b, \mu)}{\partial b} = \sum_{n=1}^N \mu_n y_n = 0$

m equations

Stationary
condⁿ

$\rightarrow 1$ equation

$m = \# \text{ parameters}$
 $= \# \text{ features}$

②

$y_n(w^T x_n + b) \geq 1 \quad \forall n \text{ from } 1 \text{ to } N$

③

$\mu_n \geq 0 \quad \forall n \text{ from } 1 \text{ to } N$

④

$\mu_n (1 - y_n(w^T x_n + b)) = 0 \quad \forall n \text{ from } 1 \text{ to } N$

feasibility condⁿ

positivity condⁿ

complementary slack condⁿ

$\rightarrow N$ equations

Unknowns

$w_1, \dots, w_m \rightarrow m$

$b \rightarrow 1$

$\mu_1, \dots, \mu_N \rightarrow N$
 $m + N + 1$

Total $m + N + 1$ equation $\rightarrow$ Total $m + N + 1$ unknowns

☆ if $\mu_i = 0$ for some $i \rightarrow x_i \rightarrow$ non support vectors
 $\rightarrow$ not contribute to w

$m = \# \text{ features}$
 $m = \# \text{ w parameters}$
 $N = \# \text{ datapoints}$

☆ if $\mu_i \neq 0$ for some $i \rightarrow x_i \rightarrow$ support vector
 $\rightarrow$ contributes to w
 $\rightarrow$ For support vectors x_i

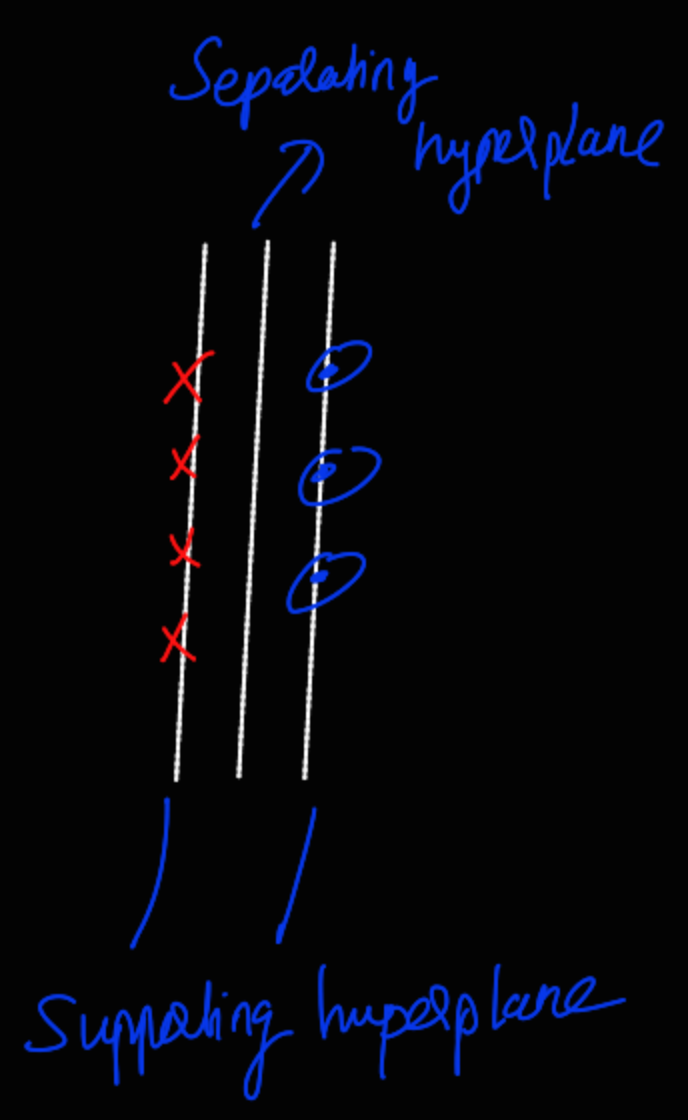
$\sum_i \mu_i y_i = 0$

Catch: $\mu_i \neq 0 \rightarrow$ Support vector that lies 1 algebraic margin away from hyperplane
 $\rightarrow w^T x_i + b = \pm 1$ or $y_i(w^T x_i + b) = 1$

exception: but inverse not true $\rightarrow$ lying on the hyperplanes $w^T x_i + b = \pm 1$ does not imply $\mu_i \neq 0$

lying on hyper plane $y_n(w^T x_n + b) = 1 \iff$ Support Vectors $x_i \iff \mu_i \neq 0$

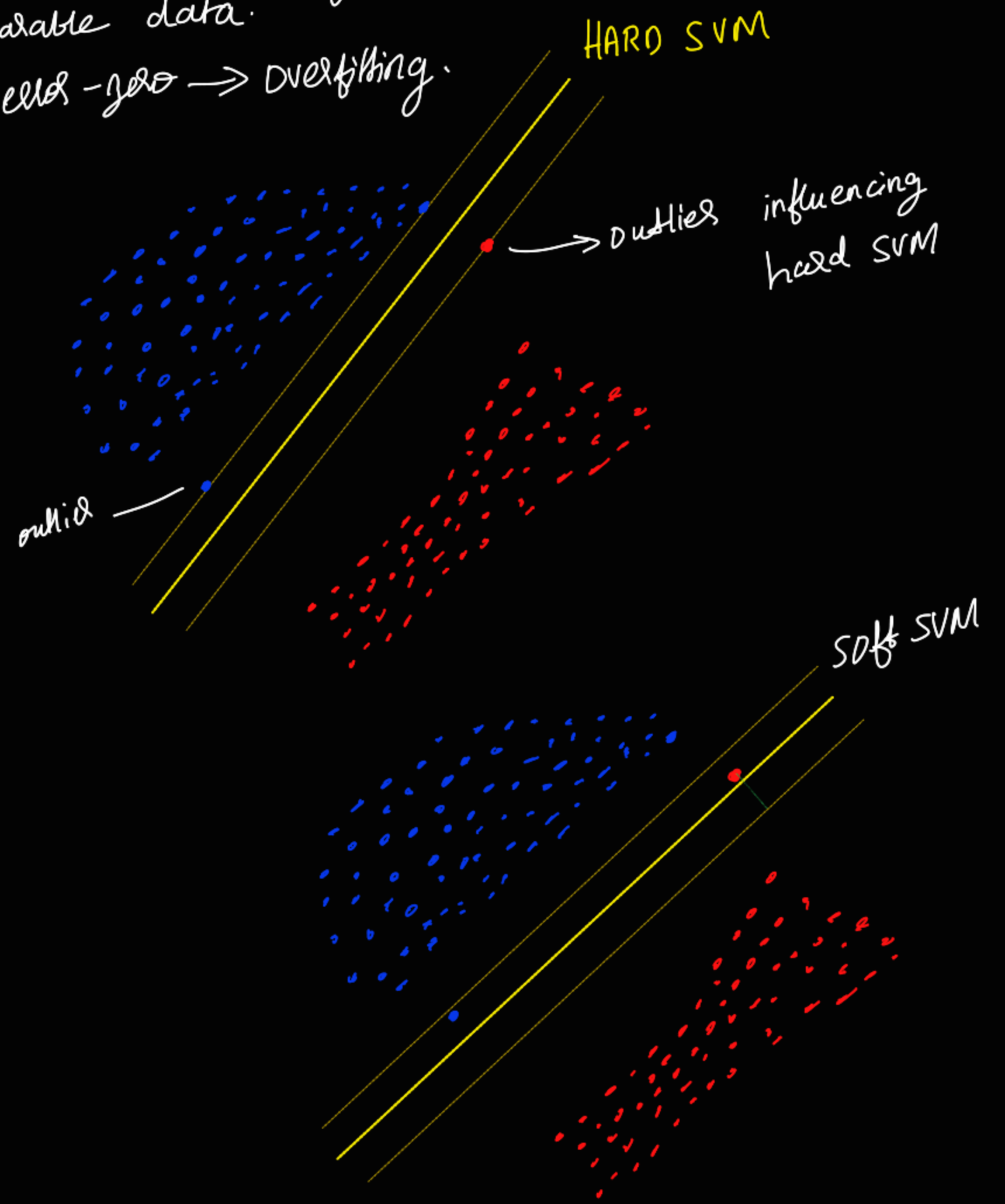
$y_n(w^T x_n + b) > 1$ Not on hyper plane $\iff$ Not support Vectors $x_i \iff \mu_i = 0$



- shortcomings of hard margin SVM
- outliers can influence the separating hyperplane too much
 - not suitable for linearly not separable data.
 - training error $\rightarrow 0 \rightarrow$ overfitting.

Soft SVM

- allow some misclassification
- Min misclassification
- Max margin
- Even for linearly separable soft margin SVM can misclassify
- points can lie in margin also
- Works even when data not linearly separable.
- generally better separating hyperplane because less overfit



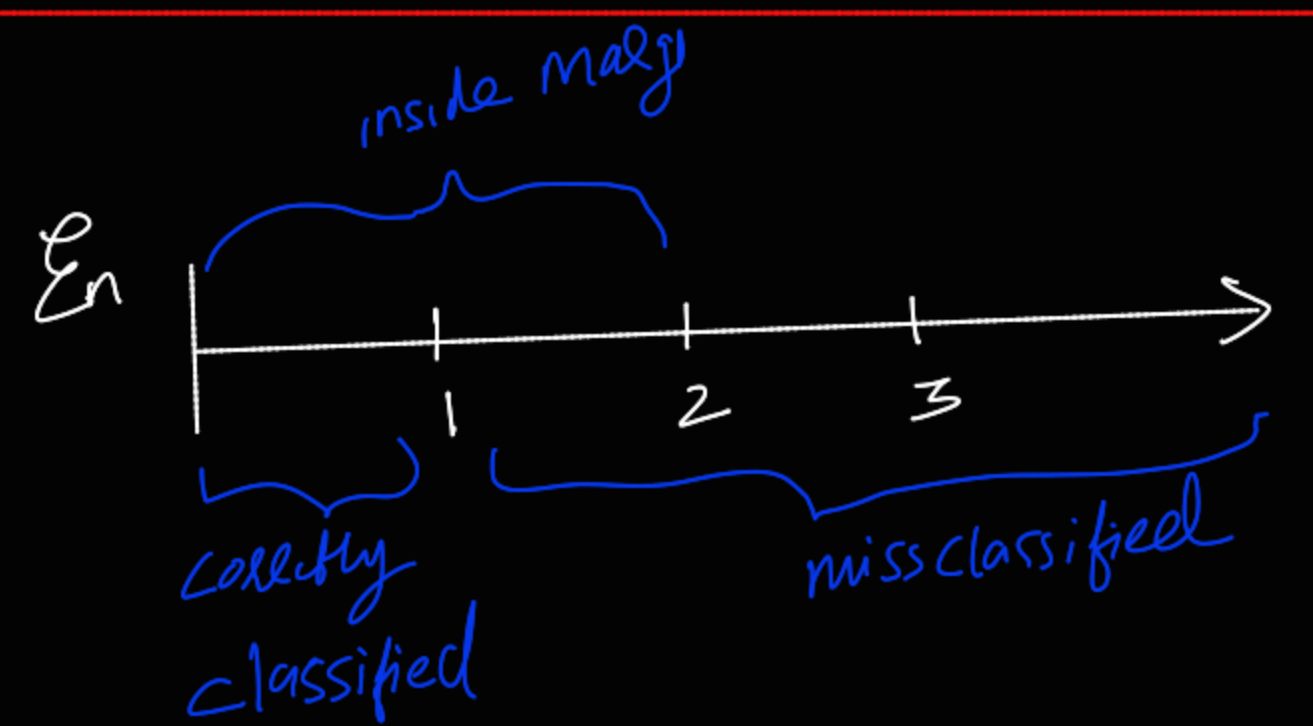
Slack

- allow slack $[e_n \geq 0]$
- Minimise sum of slack

$$\text{Minimise } \frac{\|w\|_2^2}{2} + \sum_{n=1}^N e_n \quad \text{subject to } y_n(w^T x_n + b) \geq 1 - e_n, e_n \geq 0$$

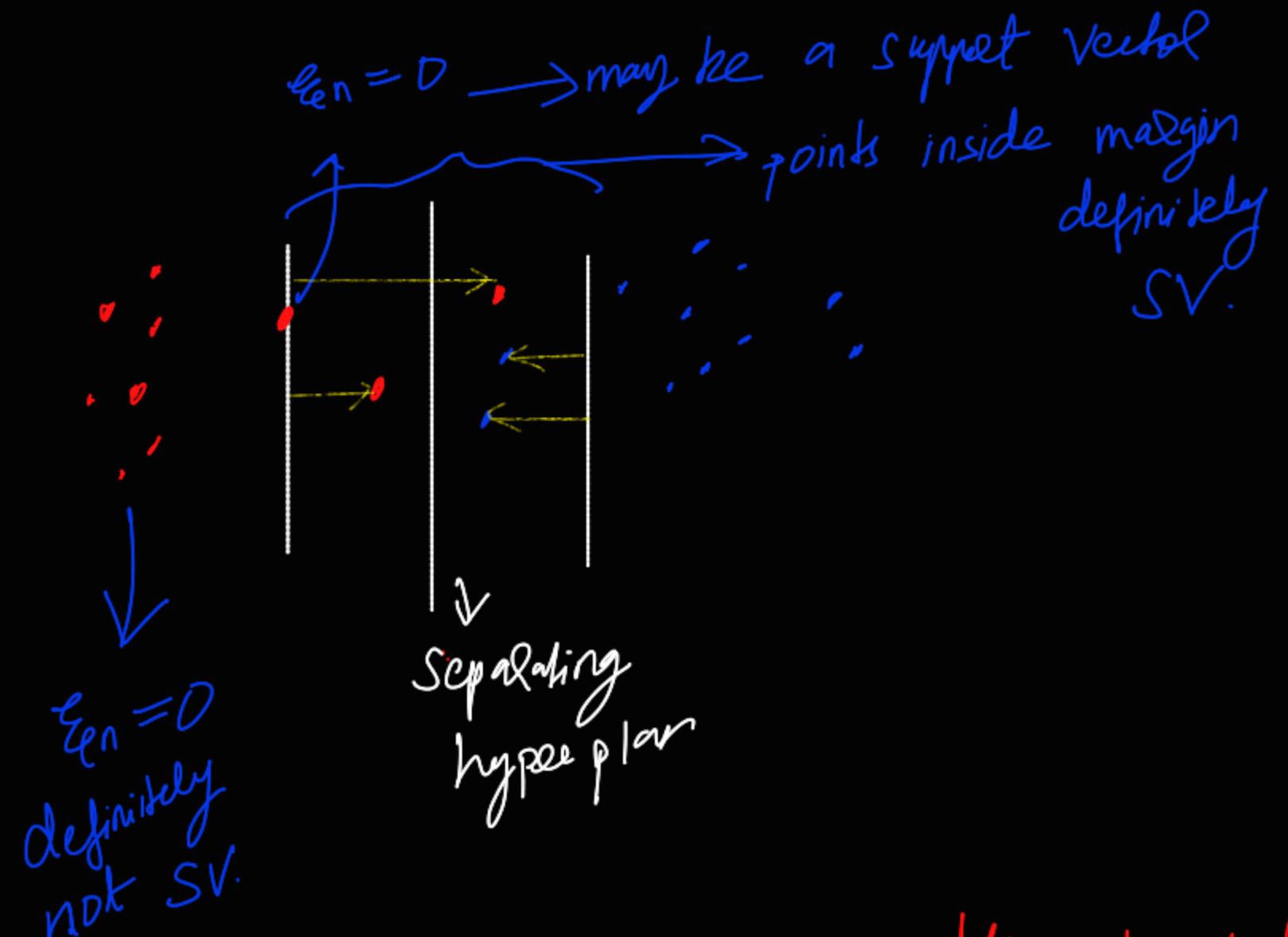
hyperparameters $\rightarrow$ validation set

- $e_n \neq 0 \rightarrow$ never
- $0 < e_n < 2 \rightarrow$ inside margin
- $0 < e_n < 1 \rightarrow$ correctly classified
- $1 < e_n \rightarrow$ miss classified



Support Vectors of Soft SVM

- Margin support vector $\Rightarrow e_n = 0$
- Non margin support vector $\Rightarrow e_n \neq 0$
- generally more no. of support vector than hard SVM.



Guideline

- If while calculating e_n if it turns out $-ve$ then point is correctly classified then put $e_n = 0$

effect of C

- C dictates which term ($\frac{\|w\|^2}{2}$ or $C \sum_{n=1}^N \xi_n$) will dominate the minimization
 - Small $C \Rightarrow \frac{\|w\|^2}{2}$ dominates $\Rightarrow$ prefer large margins
 - .. but allow potentially large # of misclassified training examples
 - Large $C \Rightarrow C \sum_{n=1}^N \xi_n$ dominates $\Rightarrow$ prefer small # of misclassified examples
 - .. at the expense of having a small margin

5. Suppose we have a soft-margin SVM for which $w = (-6, 3)$ and $b = 5$. Consider the example $(x = (2, 1), y = -1)$.

(a) How does the SVM classify x ?

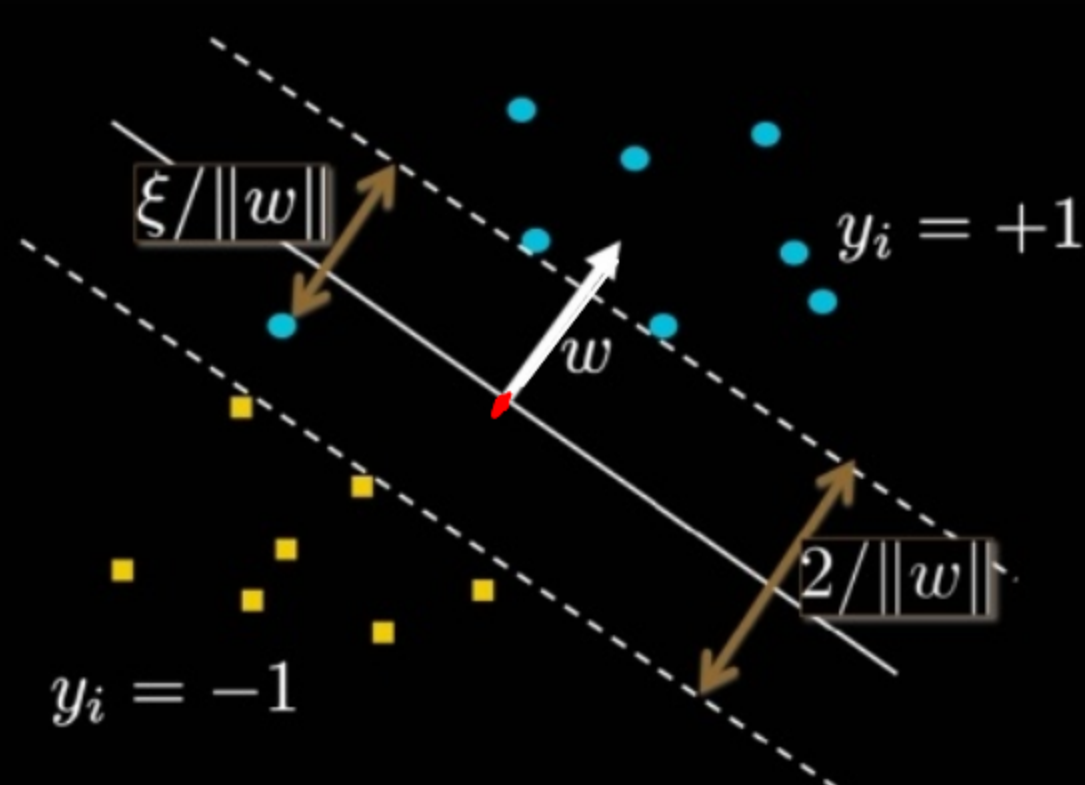
$$w^T x + b = -12 + 3 + 5 = -4$$

$$\therefore \hat{y} = -1 \text{ correctly classify}$$

$$\hat{y} < -1 \Rightarrow \xi = 0$$

$$\begin{aligned} -4 &= -(1 - \xi) \\ 4 &= 1 - \xi \\ \xi &= -3 \quad \text{Not possible} \\ \text{hence } \xi &= 0 \end{aligned}$$

(b) Does (x, y) satisfy the SVM constraint? (Answer Yes or No.)



$$\text{geometric margin} = \frac{1}{\|w\|}$$

$$\text{algebraic margin} = 1$$

$$\text{slack} = \xi$$

$$\text{geometric slack} = \frac{\xi}{\|w\|}$$

$$\text{hard} \quad y_n(w^T x_n + b) \geq 1$$

$$\text{soft} \quad y_n(w^T x_n + b) \geq 1 - \xi$$

scaling algebraic margin by C

$$\text{geometric margin} = \frac{C}{\|w\|}$$

$$\text{algebraic margin} = C$$

$$\text{slack} = C\xi$$

$$\text{geometric slack} = \frac{C\xi}{\|w\|}$$

$$y_n(w^T x_n + b) \geq C$$

$$y_n(w^T x_n + b) \geq C(1 - \xi)$$



Effect of C

Optimisation

Min

$$\frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n$$

hyperparameter

$$y_n(w^T x_n + b) \geq 1 - \xi$$

- $\rightarrow$ High penalty for slack
- $\rightarrow$ Do not try to increase margin
- $\rightarrow$ Do not want to misclassify
- $\rightarrow$ Same as hard margin SVM
- $\rightarrow$ Overfitting

$C=0$

- $\rightarrow$ No penalty for slack
- $\rightarrow$ Only priority is to Min $\|w\|$ i.e. Max Margin
- $\rightarrow$ Algorithm will increase margin to such an extent that all data points are in the margin.
- $\rightarrow$ extremely underfitting

C decreases

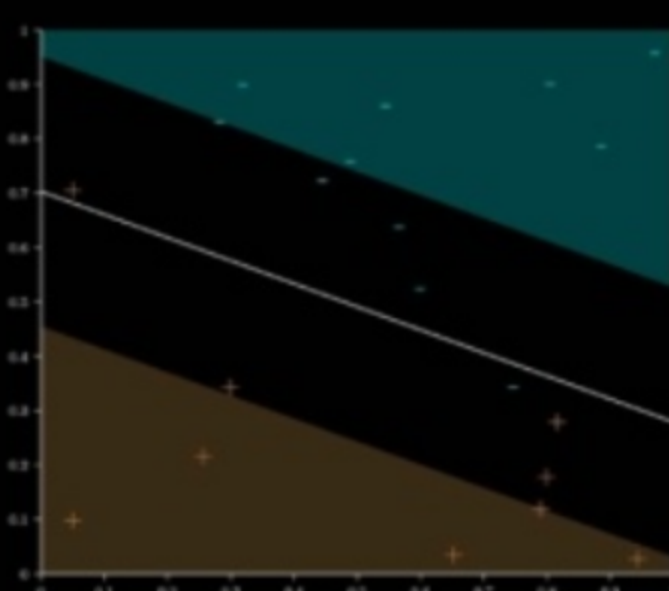
C increases

- $\rightarrow$ More underfit
- $\rightarrow$ More points misclassified
- $\rightarrow$ More points in margin
- $\rightarrow$ Margin increases

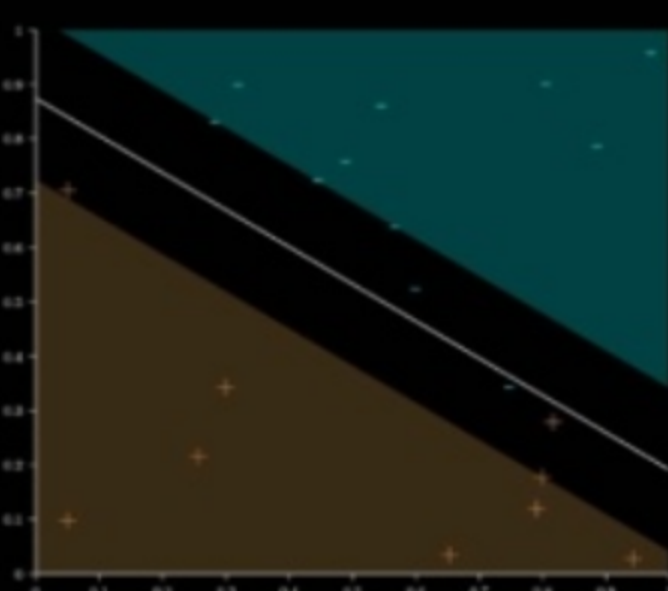
- $\rightarrow$ More overfit
- $\rightarrow$ less points misclassified
- $\rightarrow$ less points in margin
- $\rightarrow$ Margin reduces

Setting C

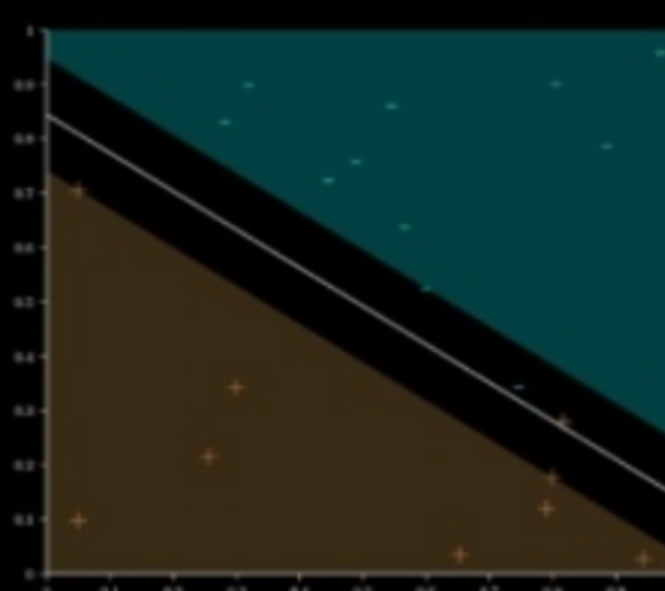
C is a tradeoff parameter (much like the tradeoff parameter in regularization)



Smaller C



Larger C



Hard Margin

Φ T/F, if we run hard margin SVM multiple times, we will get same boundary but differently scaled w & b .
 False w & b will be scaled such that the SV will lie on the SV boundary $y_n(w^T x_n + b) = 1$

$$y_i(\omega^T x_i + b) =$$

C) $\xi_i = \max(0, y_i (\mathbf{w}^T \mathbf{x}_i + b))$



(f) [3 pts] In a soft-margin support vector machine, if we increase C , which of the following are likely to happen?

- The margin will grow wider

- ✓ Most nonzero slack variables will shrink

- There will be more points inside the margin

- ✓ The norm $|w|$ will grow larger

↓ less margin $\Rightarrow$
large $\|w\|$

$$\xi_i \geq 0$$

(a) [3 pt.] If the data is linearly separable, SVM minimizes $\|w\|^2$ subject to the constraints $\forall i, y_i w \cdot x_i \geq 1$. In the linearly separable case, which of the following may happen to the decision boundary if one of the training samples is removed? **Circle all that apply.**

- ✓ Shifts toward the point removed *☆ if we remove SV*

- Shifts away from the point removed

✓ Does not change if removed non SV

① A: Soft margin SVM: increasing the value of C

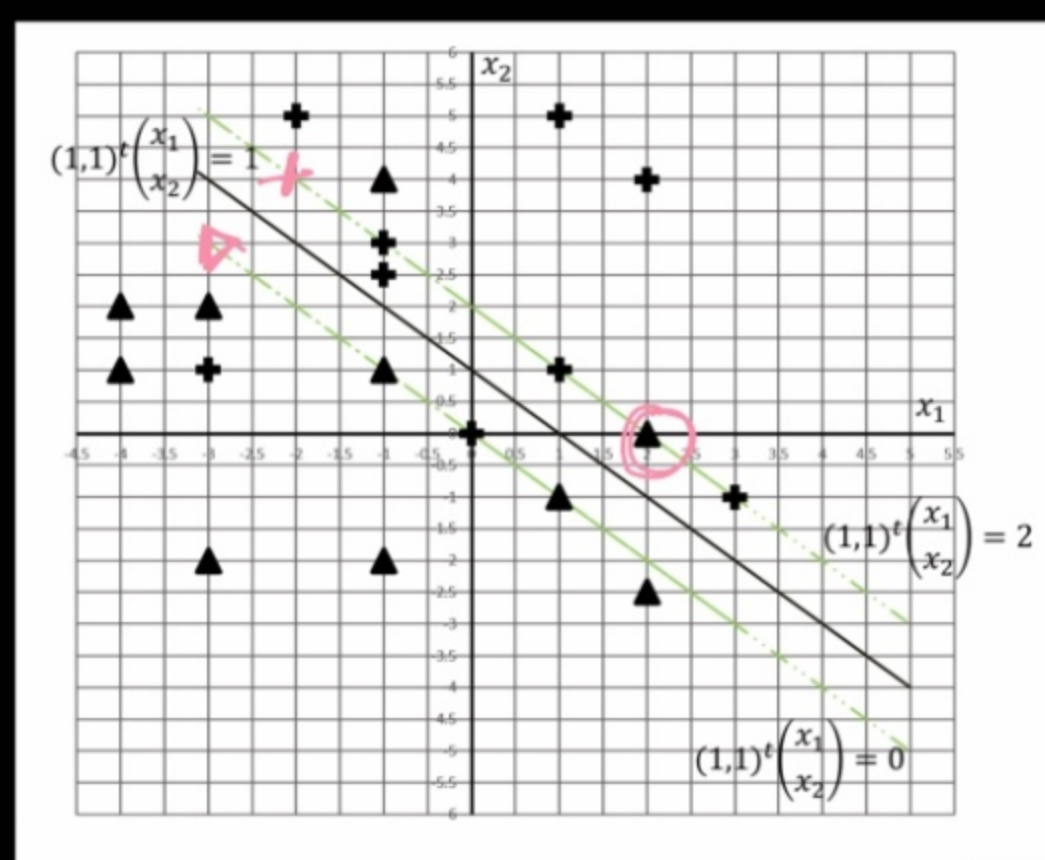
- C: Soft margin SVM: decreasing the value of C

- D: Hard margin SVM: adding a new feature to each sample point

↳ would increase margin.

For each of the following questions below, fill in the blanks with the best choice from $=$, $>$ or $<$:

$$\begin{array}{c} 0 \quad 2 \\ \xi(2,4) - \xi(2,0) \\ 0 \quad \xi(-1,1) - \xi(-1,-2) \quad 0 \\ \swarrow \quad \searrow \\ \xi(-3,1) - \xi(-1,2.5) \rightarrow 1 \quad 0.5 \\ \nwarrow \quad \nearrow \\ \xi(2,4) - \xi(-1,-2) \end{array}$$



1. The example **definitely IS** a support vector.
2. The example **definitely IS NOT** a support vector.
3. We cannot determine from the complementary slackness conditions whether or not the example is a support vector.

For each of the following training points, select the **ONE** best option from the possibilities above:

- i. (1 point) Example at $(2, 4)$ ☐ 1 ☒ 2 ☐ 3
- ii. (1 point) Example at $(1, 1)$ ☐ 1 ☐ 2 ☒ 3
- iii. (1 point) Example at $(2, 0)$ ☒ 1 ☐ 2 ☐ 3

Consider a Support Vector Machine (SVM) with decision boundary $\mathbf{w}'\mathbf{x} + b = 0$ for a three-dimensional feature space, where ' stands for vector transpose. The weight vector is $\mathbf{w} = (1 \ 2 \ 3)'$, and $b = 4$.

a. Which of the following points will be classified incorrectly by this SVM?

$q_1 = 5$ i. $x_1 = (0 \ 0 \ 0)'$, $y_1 = -1$ (minus one) *misclass*

$= 3$ ii. $x_2 = (-1 \ -1 \ -1)'$, $y_2 = 1$ (plus one) *misclass.*

$= 0.99$ iii. $x_3 = (-3.99 \ -3 \ 2)'$, $y_3 = 1$ (plus one) *correctly class.*

$$\omega = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

b. What would have been the slack variables ξ_i for the three points above, if they were in the training set? Assume w and b do not change. Hint: the constraints are $y_i(w'x_i + b) \geq 1 - \xi_i$

SOLVING SOFT MARGIN SVM

$$\min_{w,b} \frac{1}{2} \|W\|^2 + C \sum_{n=1}^N \xi_n \quad \left| \quad y_n(w^T x_n + b) \geq 1 - \xi_n, \xi_n \geq 0 \quad \forall n=1,2,3,\dots,N\right.$$

→ Convert to Lagrangian template

$$\min_{w,b} \frac{1}{2} W^T W + C \sum_{n=1}^N \xi_n \quad \left| \quad 1 - \xi_n - y_n(w^T x_n + b) \leq 0, \xi_n \leq 0 \quad \forall n \in \{1,2,\dots,N\}\right.$$

$$L(w,b,\xi,\mu,\lambda) = \frac{1}{2} W^T W + C \sum_{n=1}^N \xi_n + \sum_{n=1}^N \mu_n (1 - \xi_n - y_n(w^T x_n + b)) - \sum_{n=1}^N \lambda_n \xi_n$$

→ Notice here ξ_n is being considered as a parameter, even though it is determined by y_n, x_n, w, b
 → This is because, the untrained slack is part of an inequality not an equality. It is only after training that the optimum slack is found.

$$\frac{\partial L}{\partial W} = 0 \Rightarrow W^T - \sum_{n=1}^N \mu_n y_n x_n^T = 0$$

$$\Rightarrow W = \sum_{n=1}^N \mu_n y_n x_n$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{n=1}^N \mu_n y_n = 0$$

$$\frac{\partial L}{\partial \xi_n} = 0 \Rightarrow C - \mu_n - \lambda_n = 0 \quad \forall n \in \{1,2,3,\dots,N\}$$

Stationary
condⁿ

$$\xi_n \geq 0, y_n(w^T x_n + b) \geq 1 - \xi_n \quad \forall n \in \{1,2,\dots,N\}$$

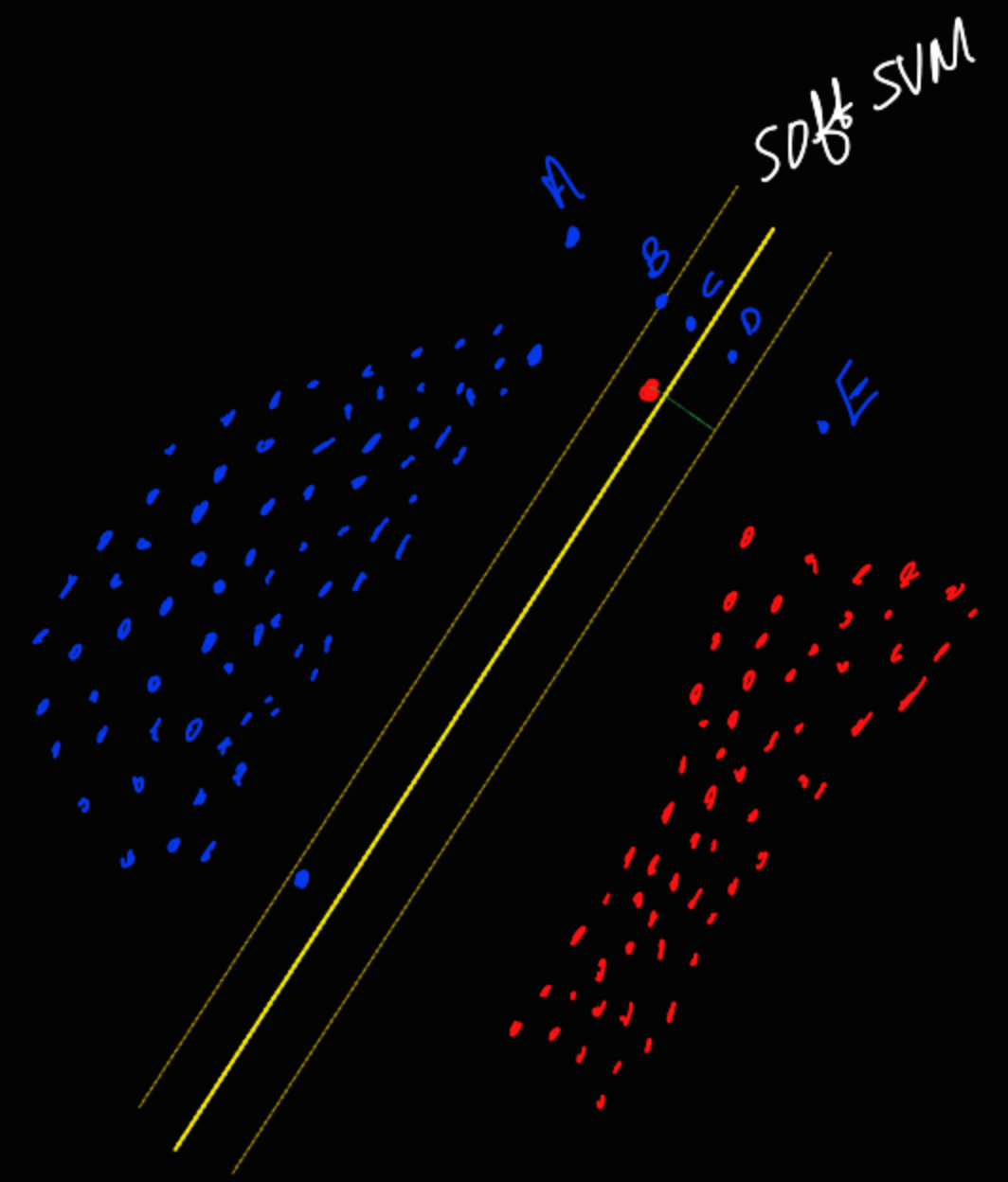
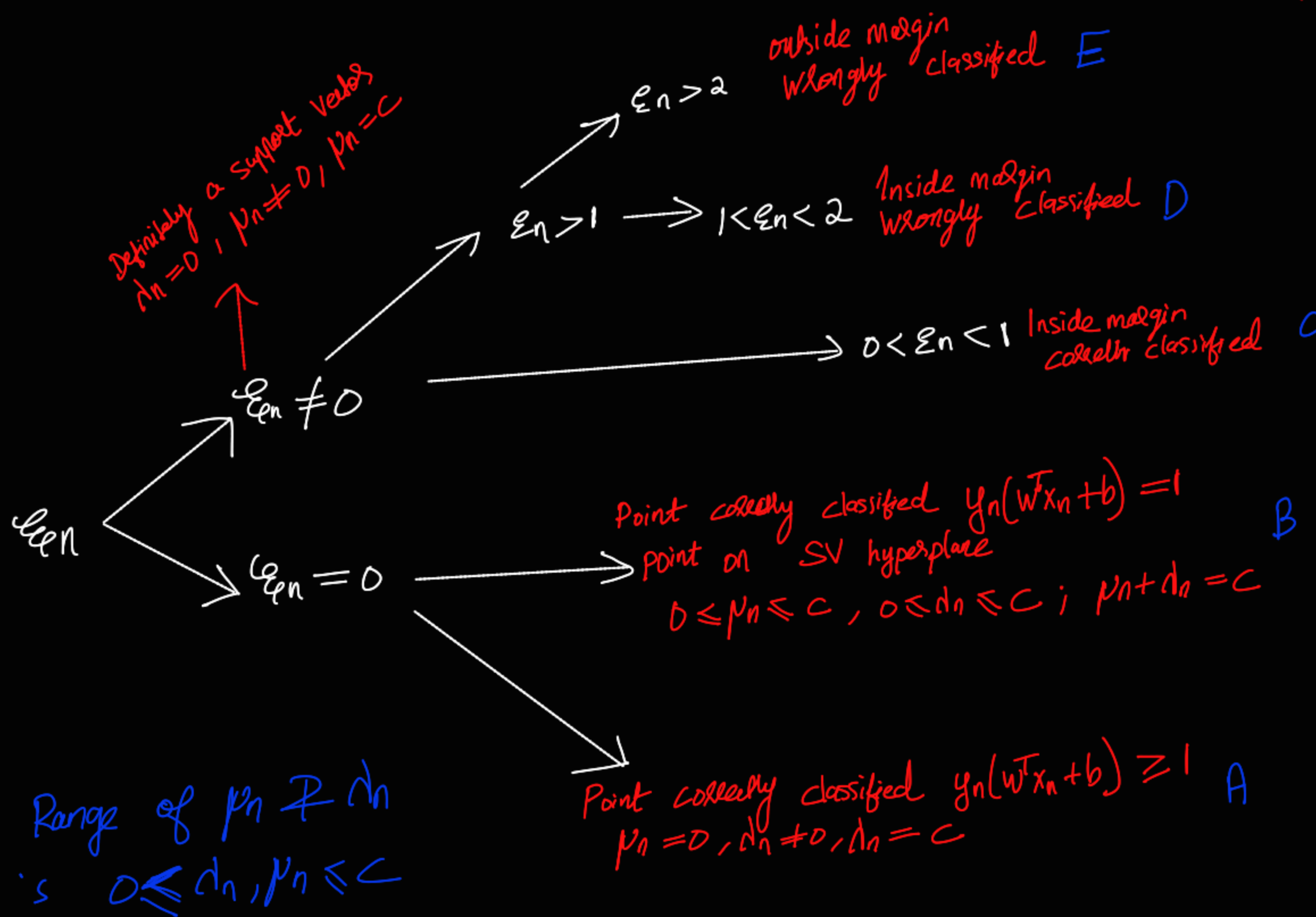
feasibility condⁿ

$$\mu_n (1 - \xi_n - y_n(w^T x_n + b)) = 0, \lambda_n \xi_n = 0 \quad \forall n \in \{1,2,\dots,N\}$$

Complementary slack condⁿ

$$\lambda_n \geq 0, \mu_n \geq 0 \quad \forall n \in \{1,2,\dots,N\}$$

positivity condⁿ



- HARD MARGIN $\Rightarrow 0 \leq \mu_n \leq \infty$
- SOFT MARGIN $\Rightarrow 0 \leq \mu_n \leq C$
- if $C \rightarrow \infty \Rightarrow$ SOFT $\rightarrow$ HARD

→ Point B is critical, may or may not be SV
 if it is an SV, then it is called marginal SV.
 → other points C, D, E are all non marginal SV.

Q) T/F $\rightarrow$ marginal SV $\Rightarrow 0 < \mu_n < C$ — False marginal SV $\Rightarrow 0 < \mu_n \leq C$

Q) T/F $\rightarrow 0 < \alpha < C \Rightarrow$ marginal SV — True no other SV can have these values.

Q) T/F $\rightarrow 0 < \alpha \leq C \Rightarrow$ marginal SV — False. It is an SV but if $\mu_n = C$ may be a non marginal SV also

SVM: Hinge Loss Formulation

$$\text{Min } \frac{1}{2} \|w\|_2^2 + C \sum_{n=1}^N \ell_n \quad | \quad y_n(w^T x_n + b) \geq 1 - \xi$$

rewriting for slack $\left\{ \begin{array}{l} \xi_n \geq 1 - y_n(w^T x_n + b) \\ \xi_n \geq 0 \end{array} \right. \Rightarrow$ After optimisation ξ_n will be minimised & it will be $\max(0, 1 - y_n(w^T x_n + b))$

optimum ξ

$$\xi_n = \max(0, 1 - y_n(w^T x_n + b))$$

$$\Rightarrow \text{Minimise } \frac{1}{2} \|w\|_2^2 + C \sum_{n=1}^N \max(0, 1 - y_n(w^T x_n + b))$$

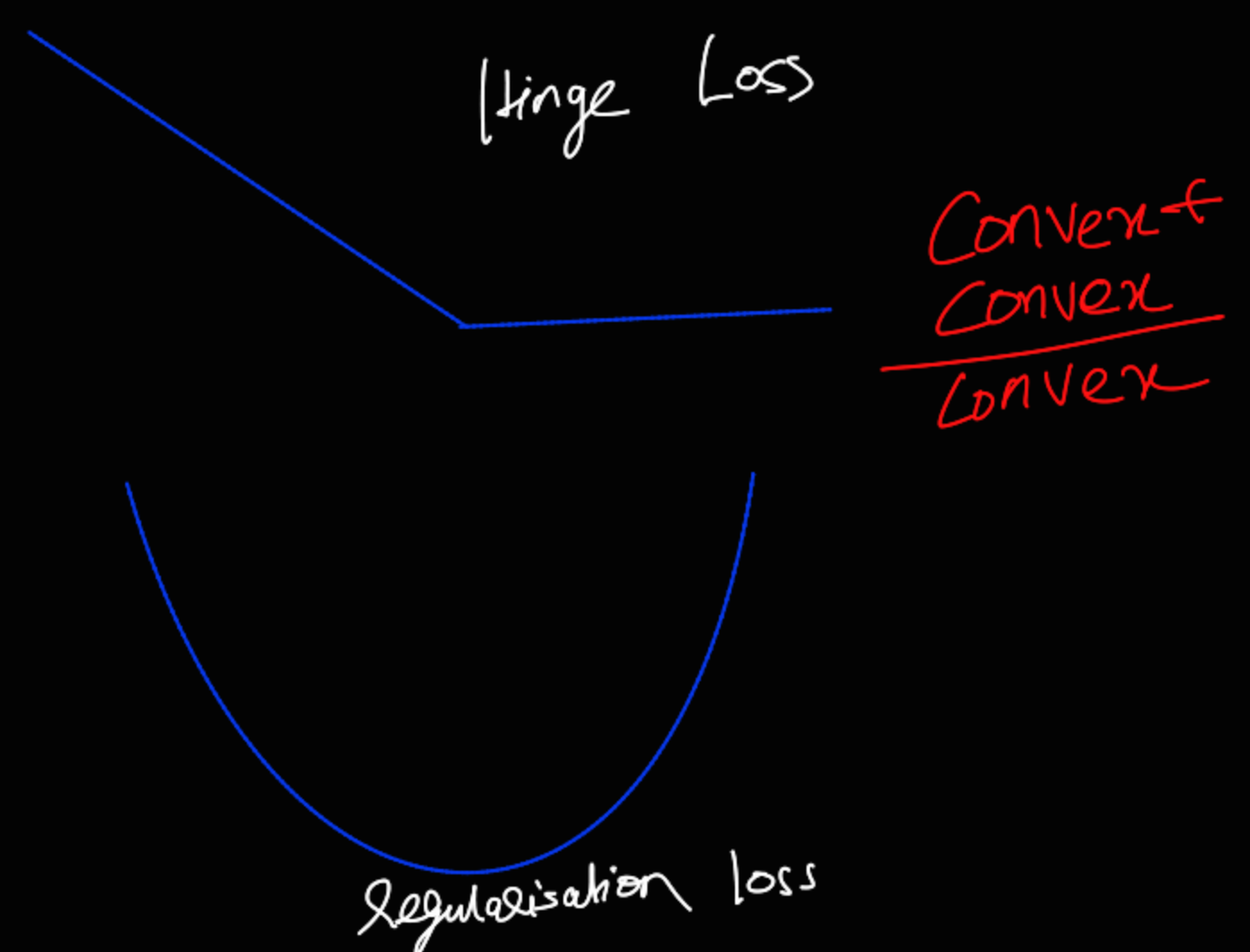
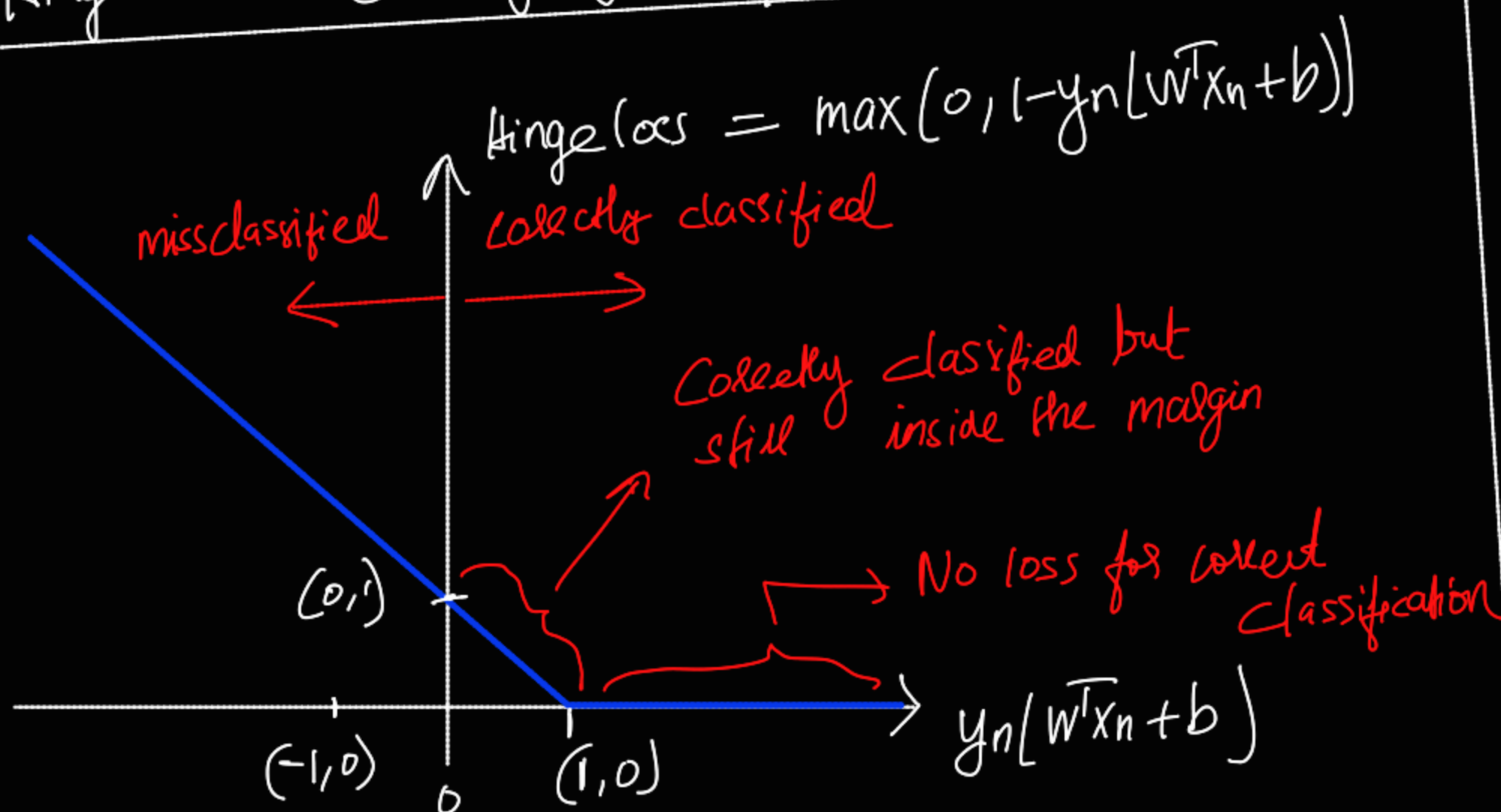
regularisation loss

- Trying to Reduce Weights
- Trying to increase margin
- Penalise large $\|w\|$
- Penalise small margin
- Promotes generalisation

Hinge Loss = $\sum \text{slack}$ on

- Trying to reduce slack
- Trying to minimise misclassification, points in margin
- Penalise misclassified points, points in margin

Hinge Loss for any given point



Both are convex $\Rightarrow$ soft SVM loss
is convex optimisation problem
 $\Rightarrow$ Unique global solution

Suppose we have a soft-margin SVM for which $\mathbf{w} = (-6, 3)$ and $b = 5$. Consider the example $(\mathbf{x} = (2, 1), y = -1)$.

- (a) How does the SVM classify $\mathbf{x}$? $w^T x > 0 \Rightarrow +1$
 $< 0 \Rightarrow -1$
- (b) Does $(\mathbf{x}, y)$ satisfy the SVM constraint? (Answer Yes or No.)
- (c) What is the hinge loss associated with $(\mathbf{x}, y)$?

$$y_n(w^T x_n + b) = -1(-12 + 3 + 5)$$

$$= 4 > 1 \Rightarrow \text{correctly classified}$$

$$\mathcal{E}_\ell = 0 = \max(0, 1 - 1) = \max(0, -3) = 0 //$$

hinge loss = 0

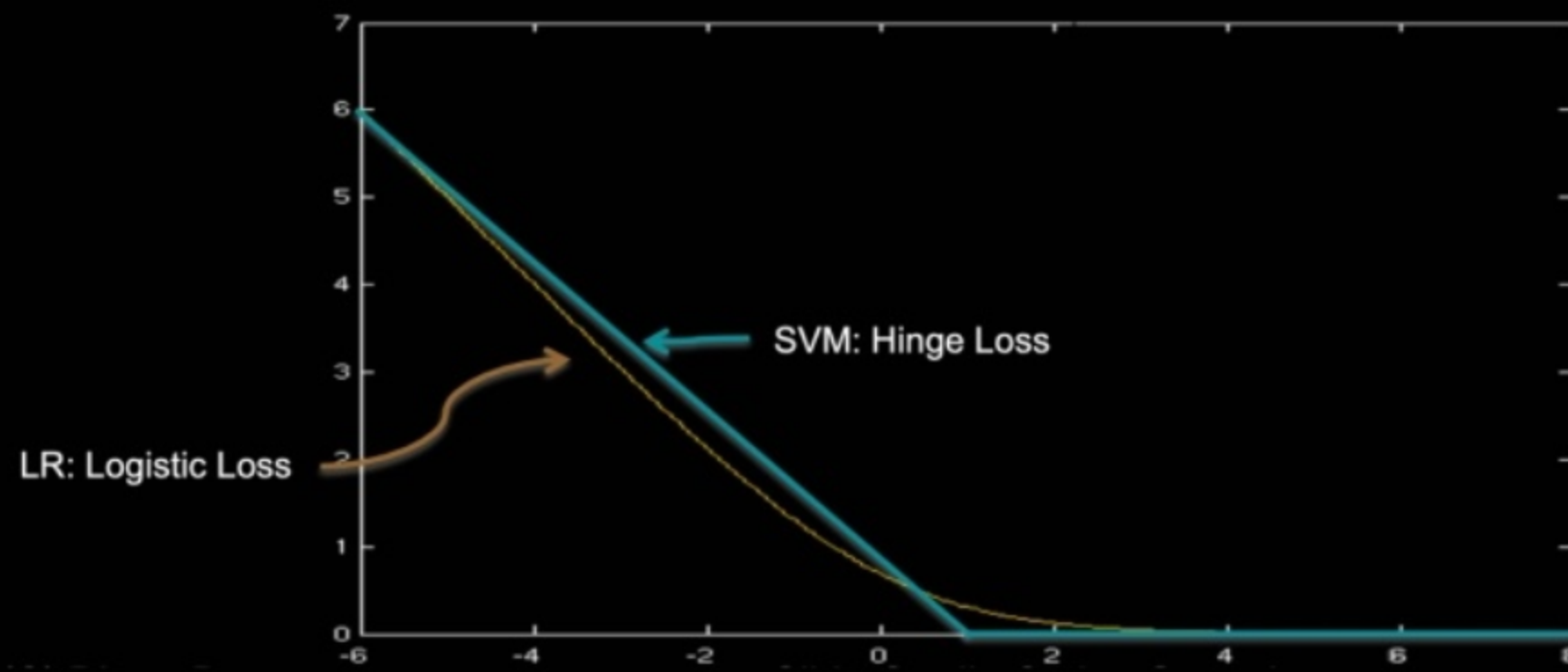
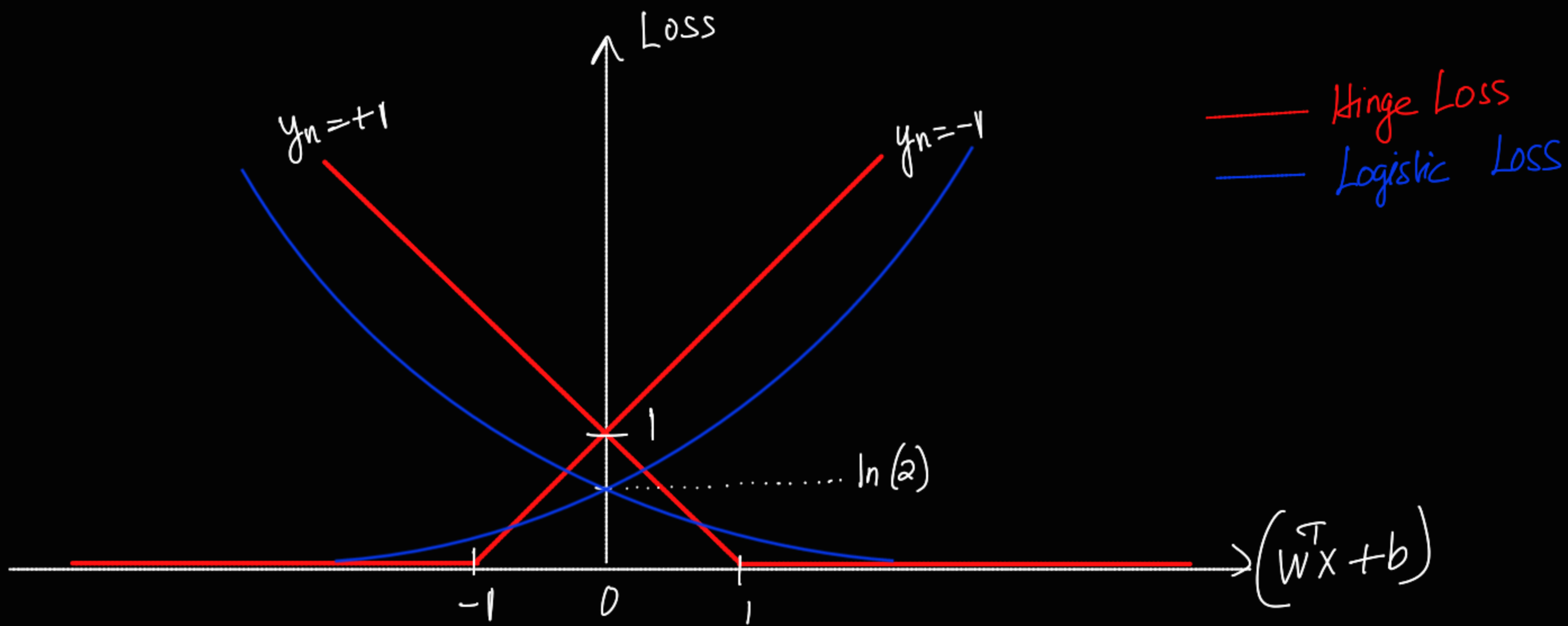
Plotting Logistic regression loss vs hinge Loss

Logistic regression
 Loss = $-\left[y_n \ln(\sigma(w^T x + b)) + (1 - y_n) \ln(1 - \sigma(w^T x + b)) \right] \rightarrow y_n \in \{0, 1\}$

Loss = $-\ln(\sigma(y_n(w^T x + b))) \rightarrow y_n \in \{-1, +1\}$

for $y_n = +1$
 Loss = $-\ln\left(\frac{1}{1 + e^{-(w^T x + b)}}\right)$

for $y_n = -1$
 Loss = $-\ln\left(\frac{1}{1 + e^{w^T x + b}}\right)$



→ Logistic Loss is approximately linear as we move away from boundary.

CROSS VALIDATION

→ Validation set

- Validate hyperparameters
- estimate generalisation error during or after training
- In other words, estimate test error, even before testing

→ hyperparameters vs parameter (weights)

- do not contribute to prediction
- To be fixed even before actual training can happen.
- learned through training model.
- Used for prediction.

- ▶ **Linear Regression:**
 - ▶ No major hyperparameters.
 - ▶ In SGD-based: **Learning rate**
- ▶ **Ridge Regression:**
 - ▶ λ : Regularization strength.
- ▶ **Logistic Regression:**
 - ▶ In SGD-based: **Learning rate**.
- ▶ **K-Nearest Neighbors (KNN):**
 - ▶ $n_neighbors$ (**K**): Number of neighbors.
- ▶ **Support Vector Machine (SVM):**
 - ▶ C : Trade-off between maximizing the margin and minimizing total slacks.

In addition → design questions

- 1) Degree of polynomial
- 2) Depth in decision tree
- 3) # neurons in NN
- 4) learning rate in SGD

→ Parametric vs non parametric (eg. KNN)

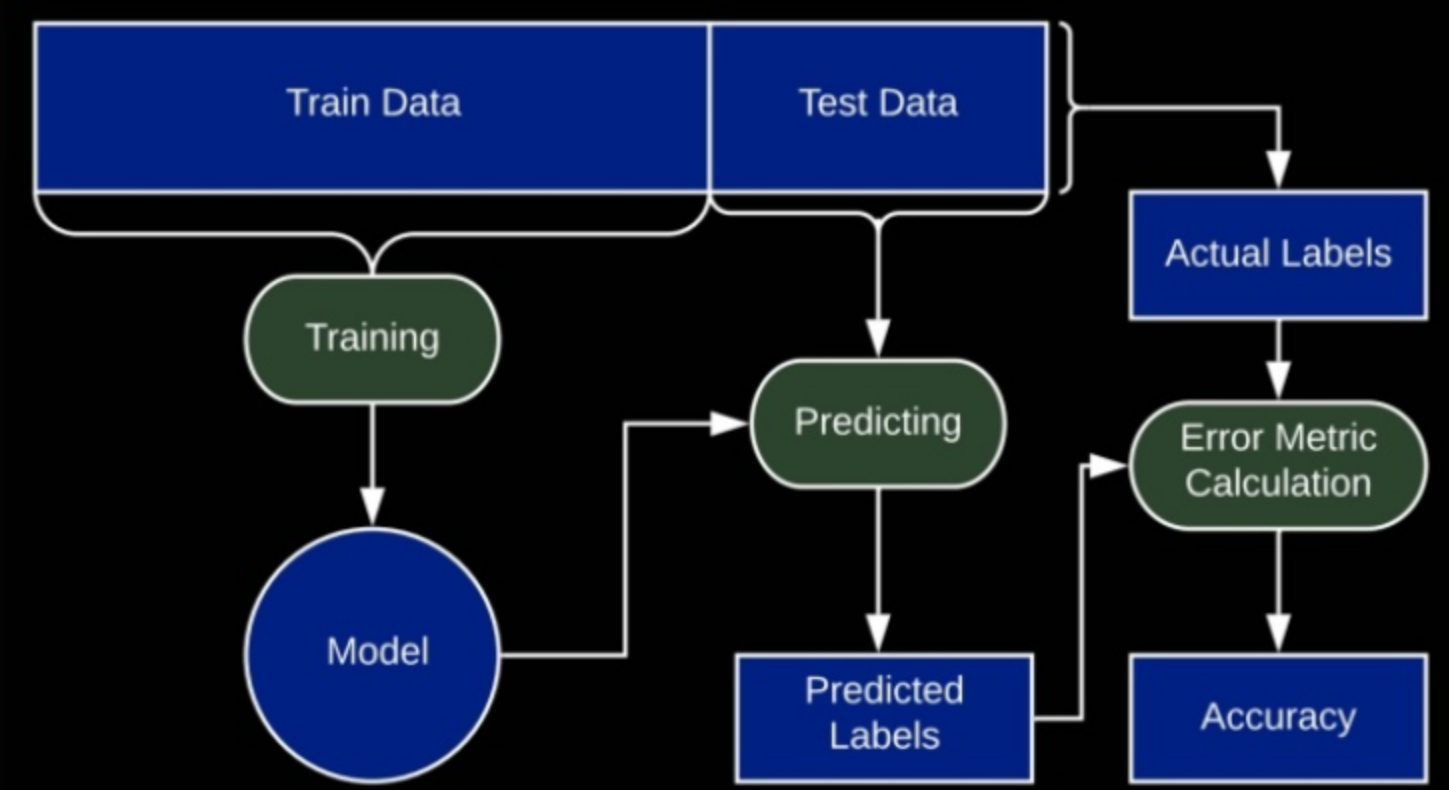
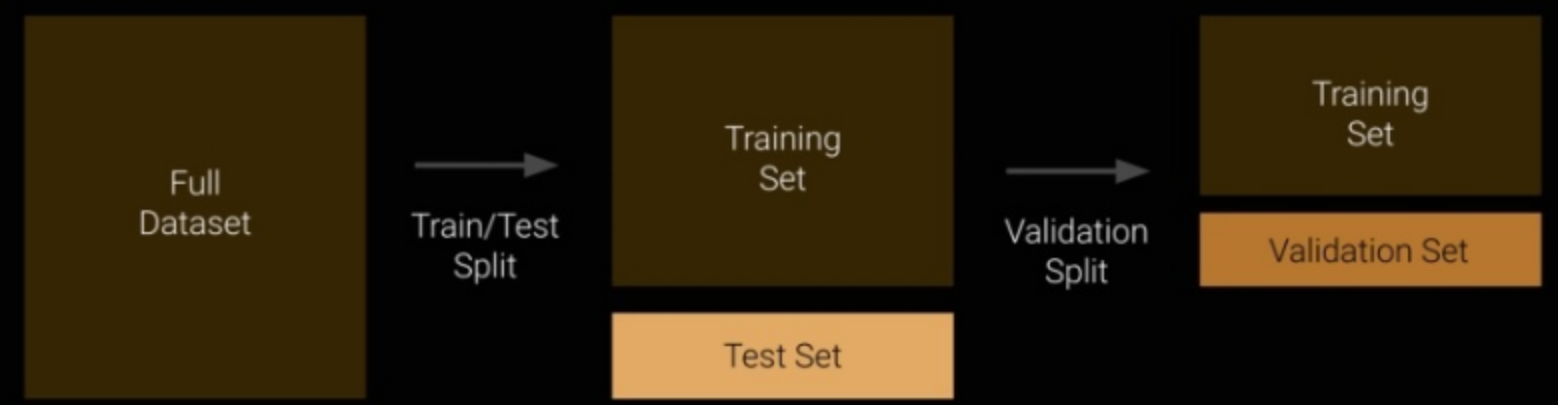
→ Can discard training data after learning parameter

→ Model based

→ No learned parameters

→ Memory based

→ Training data generally stored.



Accuracy for classification
other metric like MSE for regression problem.

HOLD OUT CROSS VALIDATION

For each value of hyperparameters, we will train the model on the training set. & then evaluate the performance on the validation set, then compare the performance (accuracy, error etc) to choose the best model.

Drawbacks

- Uses a static validation set and this is not a good generalisation.
- chance of unfortunate split
- Very risky when data is small
- Does not give good estimate on generalisation/test error because of the dependence on data split.

```

FOR each d in d-check-array
    model = Train_using(training set, d)
    performance = evaluate(model, validation set)
    perform-index.append(performance)

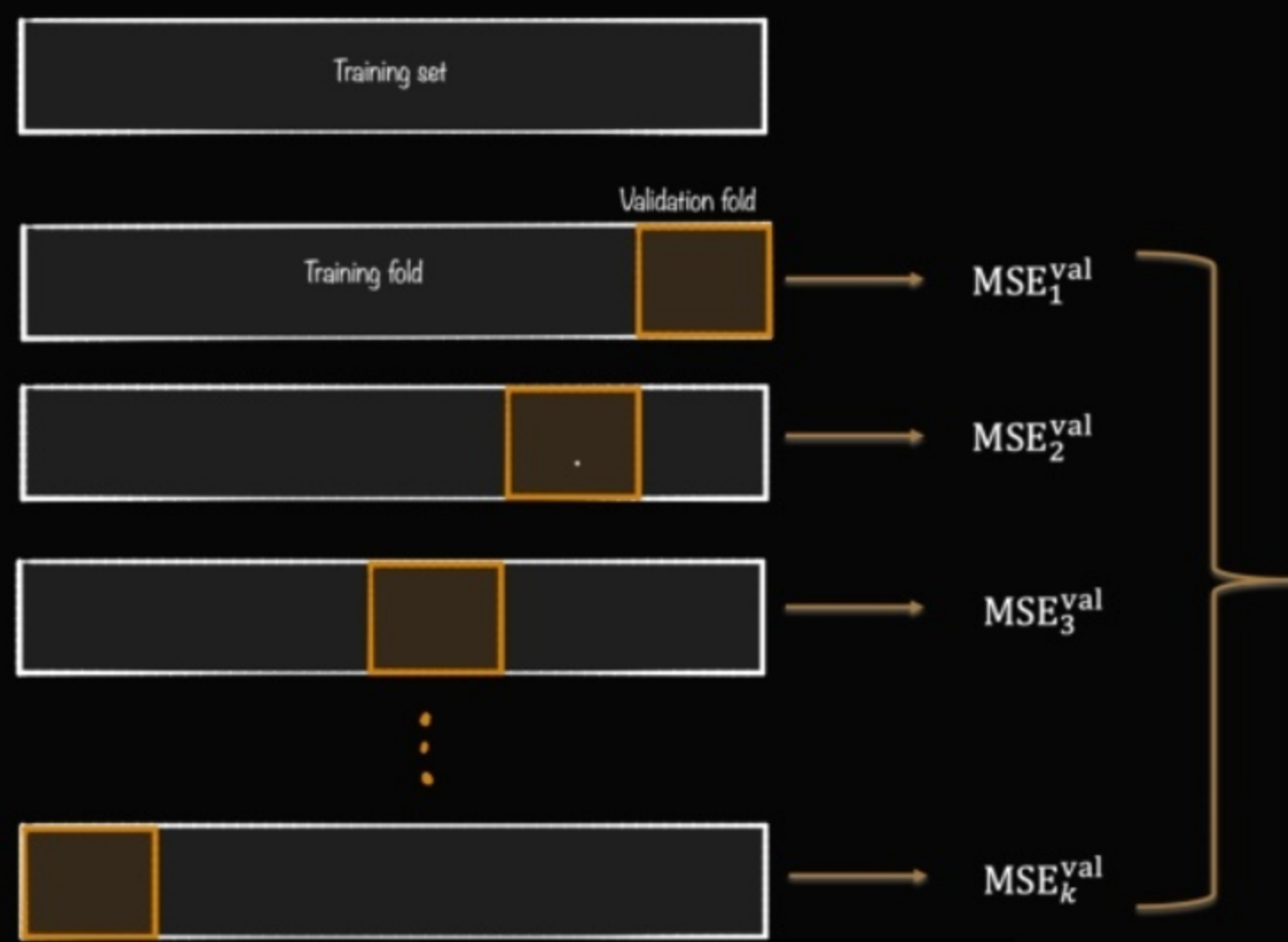
best_index = perform-index.argmax().index()
best_d = d-check_array[best_index]
  
```

→ Usually 20 to 30% as validation set

→ After finding best hyperparameter, retrain model on entire training set without leaving out anything.

NEVER TOUCH TEST DATA

K FOLDS CROSS VALIDATION



→ To estimate a single hyperparameter out from m possible values & k -fold validation

- ① → $m \cdot k$ training to be carried out — training fold
- ② → $m \cdot k$ error to be estimated — validation fold
- ③ → Average metric to be taken for all k folds
- ④ → m such average metric to be compared to find best fit hyperparameter.

$$MSE^{val} = \frac{1}{k} \sum_{i=1}^k MSE_i^{val}$$

→ For k FOLD VALIDATION DATA IS SPLIT INTO D/k } $D = |Training\ set|$
 $k = \text{Folds in CV}$

→ $|Validation\ Fold| = D/k$

→ $|Training\ Fold| = D \left(\frac{k-1}{k} \right)$

→ Average accuracy/error (metric) is to be taken from the k fold training session for one particular value.

Min out of k / Max out of k metric is not useful for validation

→ After selection, we have to retain model on the whole data, including the validation fold. This shall be the final trained model used for prediction

Pseudo code

```
lamda_choices = [10, 0.1, 0.01, 1]
avg_accuracies = []
for lamda in lamda_choices:
    current_accuracy = []
    for fold = 1 to k:
        validation_set = foldth session of training set
        current_train_set = training_set - validation_fold
        model = train(training_set, lamda)
        accuracy = evaluate(model, validation_set)
        current_accuracy.append(accuracy)
    avg_accuracies.append(current_accuracy.mean())
index = get_index(avg_accuracies, avg_accuracies.max())
return lamda_choices[index]
```

Leave one out Cross Validation

→ Special case for cross validation in which $\#folds = \#points\ in\ training\ data$

→ $|validation_fold| = \text{always one}$

→ only one data point left out during training each time

→ $n - samples \Rightarrow n$ training for testing 1 value of hyperparameter

→ $n-1 \rightarrow$ samples during training

→ 1 → sample for validation

(2) (2 points) Your friend needs help selecting between two linear regression models, desiring the one that generalizes best. Which model should they use? (Circle your answer and provide a short justification.)

Model 1

Training MSE: 100, Validation MSE: 105 ✓

Model 2

Training MSE: 50, Validation MSE: 110

} Validation error is the estimate of test error
 → small train & large test $\Rightarrow$ overfitting

14. What is cross-validation **not** used for?

- ✓ (a) To evaluate the performance of a machine learning model on unseen data. → validation set is used to evaluate/estimate the performance on test data.
- ✓ (b) To select a model's hyperparameters.
- ✓ (c) To determine the generalization of a machine learning model.
- ✗ (d) To train multiple machine learning models on different datasets.

test data $\Leftrightarrow$ unseen data

(ii) The model has a very high validation error.

- ☐ A. We should increase λ
- ☐ B. We should decrease λ
- ☒ C. Not enough information

Solution: A high validation error could either mean our model is overfitting and too complex or underfitting and not complex enough. Based on this information alone, we cannot determine which of these two situations is occurring and, therefore, do not know how to adjust λ .

What are the advantages of k -fold cross validation relative to

(a) Validation set approach

Solution. The estimate of the test error rate can be highly variable depending on which observations are included in the training and validation sets.

(b) Leave-one-out cross validation (LOOCV)

Solution. LOOCV is a special case of k -fold cross-validation with $k = n$. Thus, LOOCV is the most computationally intense method since the model must be fit n times.

In n -fold cross-validation, each data point belongs to exactly one test fold, which makes the test folds independent. However, if the data in test folds i and j are independent, are the error estimates e_i and e_j on test folds i and j also independent?

- A. True, because each test fold contains different data points.
- ☒ B. False, because data points can appear in multiple training sets, making error estimates dependent.
- C. True, because error estimates are calculated separately for each fold.

★ **SOLUTION:** False. Since a data point appears in multiple folds the training sets are dependent and thus test fold error estimates are dependent.

even though
validation fold independent
training folds are dependent
since same data point appears
in multiple training fold

(c) [3 pts] You train a linear classifier on 10,000 training points and discover that the training accuracy is only 67%. Which of the following, done in isolation, has a good chance of improving your training accuracy?

- ☒ Add novel features
- ☐ Use linear regression
- ☐ Train on more data
- ☒ Train on less data

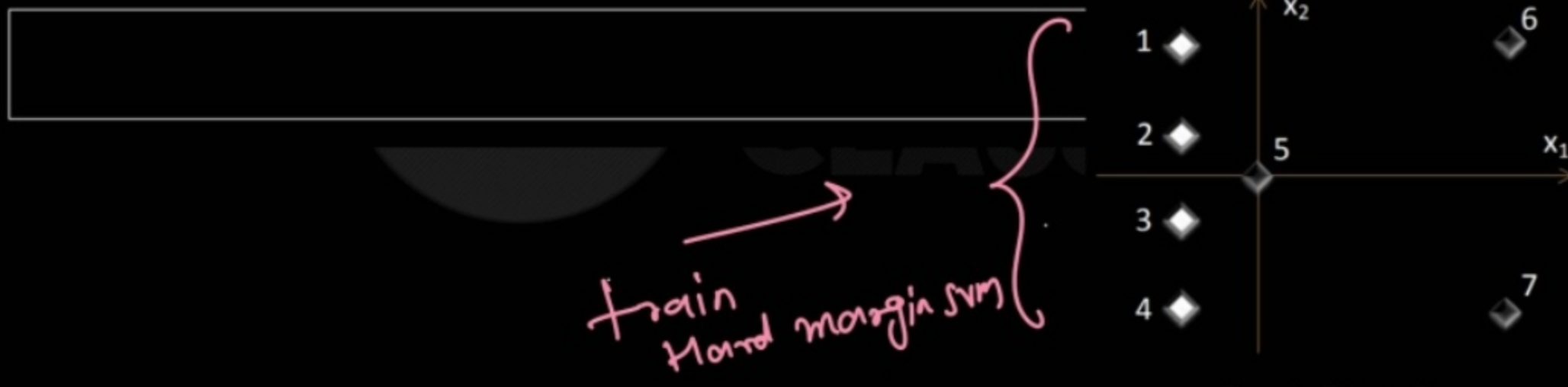
less data good fit training accuracy ↑.
But just training accuracy in practice
we don't care in selecting or adjusting model

(2.0 pt) Let's say we pick three hyperparameters to tune with cross-validation. We have 5 candidate values for hyperparameter 1, 6 candidate values for hyperparameter 2, and 7 candidate values for hyperparameter 3. We perform 4-fold cross validation to find the optimal combination of hyperparameters, across all possible combinations.

In this cross-validation process, how many random forests will we train? Your answer can be left as a product of multiple integers, e.g. "1 * 2 * 3", or simplified to a single integer, e.g. "6". (These are not the correct answers to the problem).

$$4 \times 5 \times 6 \times 7 = 840$$

v. (1 point) Suppose we use leave-one-out cross validation meaning we use 7-fold cross validation with a split of 6 to 1 between training set and validating set. Compute the average classification error over the 7-folds.



Assuming error
= $\begin{cases} 1 & \hat{y}_n \neq y_n \\ 0 & \hat{y}_n = y_n \end{cases}$
→ $y_n \neq \text{true value}$
→ where $\hat{y}_n$ is the predicted value after training w/o x_n & then predicting for x_n

→ Only point 5 gets misclassified during LOOCV ⇒ $\frac{1}{7}$

(20) Assume we have a data matrix X . Which of the following is a true statement when comparing leave-one-out cross validation (LOOCV) error with the true error?

- (A) LOOCV error is typically a slight underestimation of the true error of a model trained on X .
- (B) LOOCV error is typically a slight overestimation of the true error of a model trained on X .
- (C) LOOCV error is an unbiased estimator of the true error of a model trained on X .

Solution:

The solution is (B)

5. Suppose we have a design matrix $\mathbb{X}$ comprising of n observations, d features, and an additional intercept term. We decide to use $\mathbb{X}$ to create, tune, and evaluate a regularized linear regression model with two regularization hyperparameters, λ_1 and λ_2 . We have i different choices for λ_1 and j different choices for λ_2 , so there are $i \cdot j$ possible combinations of λ_1 and λ_2 .

We set aside 20% of our data to use as a test set and perform k -fold cross-validation to tune our hyperparameters. We compute the cross-validation error of each of the $i \cdot j$ possible combinations of λ_1 and λ_2 values, and our goal is to find the combination of values with the lowest cross-validation error. All following answers should be expressed in terms of i, j, n, d, k , and/or constants only (except for part c).

- (a) [2 Pts] For a single combination of hyperparameter values, how many model parameters do we fit?

For a single combination form $i \cdot j$, we do k folds $\rightarrow$ each fold trains on $n \cdot \frac{k-1}{k}$ sample points
 $\# \text{model parameters} = \# \text{feature} + 1 (\text{for bias}) = d+1 \Rightarrow \text{for } k \text{ fold} = (d+1)k$

- (b) [2 Pts] How many observations are in the validation set of each fold?

each fold size = $\frac{n}{k}$

- (c) [2 Pts] How many model parameters do we calculate in total? Your answer can be in terms of A , your answer to part a.

We do $i \cdot j$ times k is $(i \cdot j) \cdot (d+1)k$

35. **Extra credit:** Consider a nearest neighbor classifier that chooses the label for a test point to be the label of its nearest neighboring training example. What is its leave-one-out cross-validated error for the data in the following figure? (“+” and “-” indicate labels of the points).



Assuming
 $\text{error} = \begin{cases} 1 & \hat{y} \neq y \\ 0 & \text{otherwise} \end{cases}$
 $k\text{-NN with } k=1$

13. Why is it important to use a different test set to evaluate the final performance of the model, rather than the validation set used during model selection?

- (a) The model may have overfit to the validation set
 (b) The test set is a better representation of new, unseen data
 (c) Both a and b
 (d) None of the above

Correct answers: (c)

- (g) [4 pts] You are training a soft-margin SVM on a binary classification problem. You find that your model's training accuracy is very high, while your validation accuracy is very low. Which of the following are likely to improve your model's performance on the validation data?

- ☒ A: Training your model on more data.
☐ B: Adding a quadratic feature to each sample point.
☐ C: Increasing the hyperparameter C .
☒ D: Decreasing the hyperparameter C .

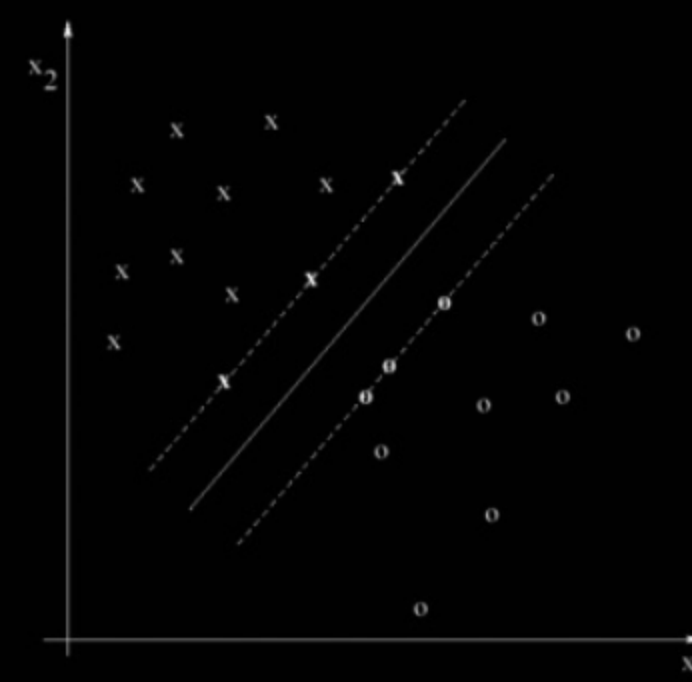
Which of the following statement(s) is(are) true for k -NN?

- ☐ k represents the number of classes.
☒ The final outcome of the algorithm may change with the distance measure.
☒ CrossValidation can be used to find the optimal k .
☐ As k increases, we overfit the data eventually.

[2 points] Suppose you have picked the parameter θ for a model using 10-fold cross validation. The best way to pick a final model to use and estimate its error is to

- (a) pick any of the 10 models you built for your model; use its error estimate on the held-out data
 (b) pick any of the 10 models you built for your model; use the average CV error for the 10 models as its error estimate
 (c) average all of the 10 models you got; use the average CV error as its error estimate
 (d) average all of the 10 models you got; use the error the combined model gives on the full training set
 (e) train a new model on the full data set, using the θ you found; use the average CV error as its error estimate

What is the leave-one-out cross-validation error estimate for maximum margin separation in the following figure? (we are asking for a number)



Answer: 0

Based on the figure we can see that removing any single point would not change the resulting maximum margin separator. Since all the points are initially classified correctly, the leave-one-out error is zero.

- (a) (1.0 pt) The test set is divided into k folds. For each fold of the test set, we use the entire training set to train the model, and use the given fold/subset of the test set for validation. The average error among all k folds is the cross-validation error.

True or False: This modification will result in overfitting.

- ☒ True
☐ False

- (c) (1.0 pt) We use normal k -fold cross-validation, but for each fold we use the entire training set for training.

True or False: This modification will result in overfitting.

- ☒ True
☐ False

- (b) (1.0 pt) We use normal k -fold cross-validation, but for each fold we only use half of the validation set for validation.

True or False: This modification will result in overfitting.

- ☐ True
☒ False

Training on training set & validating on test set $\rightarrow$ overfitting.
 why? model learns patterns based on training data, however hyperparameters is tuned looked at test data $\rightarrow$ model should never see the test data even for hyperparameters tuning because then we have no way to evaluate the performance on unseen data.
 $\rightarrow$ Test set error is no longer an unbiased estimate of generalization error.
 $\rightarrow$ CV error is optimistic underestimate of the true generalization error.
 Model overfits to the test set

How data leak leads to overfitting. Model is not learning anything bad like noise. It's just that on test validation will not be a good estimate. Model should generalise just fine

You're absolutely right — this is a subtle but very important distinction. Let's break it down carefully.

1. Classic overfitting (what most textbooks describe)

- Model capacity too high → fits **noise** in the training data.
- Outcome:
 - Training error ≈ 0
 - Test error much higher
- Cause: model is memorizing, not generalizing.

2. Data leakage scenario (your case)

- Here the model is not necessarily fitting noise.
- Instead, the **evaluation procedure is flawed**:
 - If validation/test data are leaked into training, then the model has “seen” them.
 - Reported validation/test error looks artificially low.
 - But on truly unseen data, error is higher.

So the **model itself might generalize just fine** if trained properly — but because the *evaluation* is biased, we think the model is better than it really is.

Why we still call it “overfitting to the test set”

Even though the model isn't memorizing random noise:

- The model selection process is indirectly tuned to the leaked test set.
- Over many iterations (e.g. choosing hyperparameters based on leaked validation/test), the final chosen model becomes tailored to *that dataset's quirks*.
- It might not be noise, but it's still dataset-specific information that doesn't generalize.

So the terminology “overfitting” is used in a broader sense:

- 👉 Either the model learns noise (classic case),
- 👉 Or the evaluation leaks cause us to **optimize for the wrong target** (biased test estimate).

⚖️ The Key Difference

- **Classic overfitting**: model truly generalizes poorly.
- **Data leakage “overfitting”**: model may generalize fine in principle, but we no longer know for sure because our performance estimate is overly optimistic.

✅ So you're right: in data leakage, the model itself isn't necessarily worse. The problem is that our **measure of generalization is broken** — so we might *deploy a model thinking it's great*, only to find it fails on truly unseen data.

Mark all the statements that are true, about cross-validation.

- ☒ Increasing k in k -fold cross validation decreases the bias of the model.
- ☐ We use cross validation because it is less computationally intensive than regular validation.
- ☒ We use cross validation over regular validation because cross validation allows us to use the entire training data set to test the model
- ☐ Cross-validation cannot be used in production, as the model is cheating by looking at validation samples.

EFFECT OF k IN k - FOLD VALIDATION

◆ 1. Small k (e.g. $k = 2, k = 5$)

- Training size per fold is smaller $\rightarrow$ model trained on less data.
- This tends to **overestimate error** (model looks worse than it really is).
- $\rightarrow$ **Higher bias, lower variance** (since folds are more different, averaging smooths out fluctuations).

◆ 2. Large k (e.g. $k = 10, k = n$, i.e. LOOCV)

- Training size per fold is almost the full dataset $\rightarrow$ model trained on nearly all data.
- This tends to **underestimate error** (model looks better than it really is).
- $\rightarrow$ **Lower bias, higher variance** (because folds are very similar, so error estimate depends heavily on a few points).

◆ 3. Extreme Case: LOOCV (Leave-One-Out Cross-Validation, $k = n$)

- Train on $n - 1$ points, validate on 1 point.
- Repeated n times.
- **Bias**: minimal (you almost always train on the full dataset).
- **Variance**: very high (error estimate depends on each single point – if a point is an outlier, error spikes).

◆ 4. Practical Choice

- $k = 5$ or $k = 10$ is commonly chosen.
- This gives a **good balance**:
 - Bias is not too high,
 - Variance is not too high,
 - Computation is reasonable.

Consider a scenario where we use leave-one-out cross-validation (LOOCV) with Support Vector Machines (SVM) for binary classification. Given a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)\}$, the cross-validation error for a sample $(\mathbf{x}_n, y_n)$ is defined as:

$$e_n = \begin{cases} 1 & \text{if } \hat{y}_n \neq y_n \\ 0 & \text{otherwise} \end{cases}$$

where $\hat{y}_n$ is the predicted label for $\mathbf{x}_n$ based on the model trained without the sample $\mathbf{x}_n$. The overall cross-validation error is given by:

$$E_{CV} = \frac{1}{N} \sum_{n=1}^N e_n$$

Which of the following statements is true regarding the relationship between the overall cross-validation error E_{CV} and the number of support vectors K ?

- A. $E_{CV} \geq \frac{K}{N}$
- B. $E_{CV} = \frac{K}{N}$
- C. $E_{CV} \leq \frac{K}{N}$ ✓
- D. E_{CV} is independent of K

EVALUATION METRICS FOR CLASSIFICATION MODELS

ACCURACY

$$\rightarrow \text{Accuracy} = \frac{\# \text{Correct classifications}}{\# \text{All classifications}}$$

→ Limitations of Accuracy — ① → Unfair if #datapoints are highly unbalanced
eg. - 2 class problem

class 0 examples = 9990
class 1 examples = 10

→ Stupid model predicts $\hat{y} = 0$ always
→ Still the accuracy = 99.9%

Usually occurs in medical statistics epidemic statistics.

Accuracy effectively weights the performance of the classes values based on the fraction of the training data that they represent, so if one class occurs 5 more times than another, it will have five times the impact on accuracy. Thus accuracy does not count the performance of the minority class enough and this may lead to strategies that always, or almost always, predict the majority class.

Terminology/Convention

→ POSITIVE CLASS → Minority class

→ NEGATIVE CLASS → Rest of the classes

→ Sensitivity → Accuracy for positive case = $\frac{\# \text{Correctly predicted +ve}}{\# \text{Total actual +ve}}$ → Also called Recall / True positive rate

→ Specificity → Accuracy for negative class = $\frac{\# \text{Correctly predicted -ve}}{\# \text{Total actually -ve}}$

Precision

→ out of all reported positives, how many were true positive

$$\frac{\# \text{true positives among predicted positives}}{\# \text{samples predicted as positive}}, \text{ defined as } 1 \text{ if denominator is } 0.$$

Recall / Sensitivity / True Positive Rate

→ out of all the ground truth positives, how many were predicted correctly.

Consider the following ground truth of 10 test examples.

Normal Pneumonia
Test 

Calculate the precision and recall of the Pneumonia class for all the given models.

Model1 

Model2 

Model3 

$$\text{Precision} = \frac{0}{0}; \text{Recall} = \frac{0}{1}; \text{Accuracy} = 90\%$$

$$\text{Precision} = \frac{1}{3}; \text{Recall} = \frac{1}{1}; \text{Accuracy} = 80\%$$

$$\text{Precision} = \frac{1}{6}; \text{Recall} = \frac{1}{1}; \text{Accuracy} = 50\%$$

(20 points) Each of the following scenarios describes the result of some (made up) classifier: there is a list of the correct labels, and the corresponding predictions. For each of the following situations, compute accuracy, recall, precision, and F_1 score. You may verify your answers with code, but you must show work, or some intermediate steps, to receive full credit on this problem.

(a) A binary classification result, where the correct labels are [T, T, F, T, F, T, F, T] and the predicted labels are [T, F, T, T, F, F, F, T]. Assume T means "true" (the desired class) and F ("false") is the "default" class. Compute accuracy, and compute recall, precision, and F_1 for T.

(b) A binary classification result, where the correct labels are [T, F, F, F, F, F, F, T] and the predicted labels are [F, T, F, F, F, F, F, F]. Assume T means "true" (the desired class) and F ("false") is the "default" class. Compute accuracy, and compute recall, precision, and F_1 for T.

$$\text{Accuracy} = 5/8; \text{precision} = 3/4; \text{Recall} = 3/5$$

$$\text{Accuracy} = 5/8; \text{precision} = 0; \text{Recall} = 0$$

CONFUSION MATRIX

From actual data & predicted value $\rightarrow$ Predicted
 $\rightarrow$ ground truth

	Actual	
	Positive	Negative
Positive	True positive	False positive
Negative	False negative	True negative

Actual Class	Predicted Class
0	0
1	1
0	1
0	1
0	0
1	1
0	0
0	0
1	1
1	0

	Actual class		
	Positive (1)	Negative (0)	Total
Predicted class	3	2	5
	1	4	5
Total	4	6	10

$\rightarrow$ No information lost

$$\text{Precision} = \frac{3}{5} \quad \text{Recall} = \frac{3}{4}$$

Number of common

$$\text{Precision} = \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

$$\text{Recall} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

$\rightarrow$ Intuition

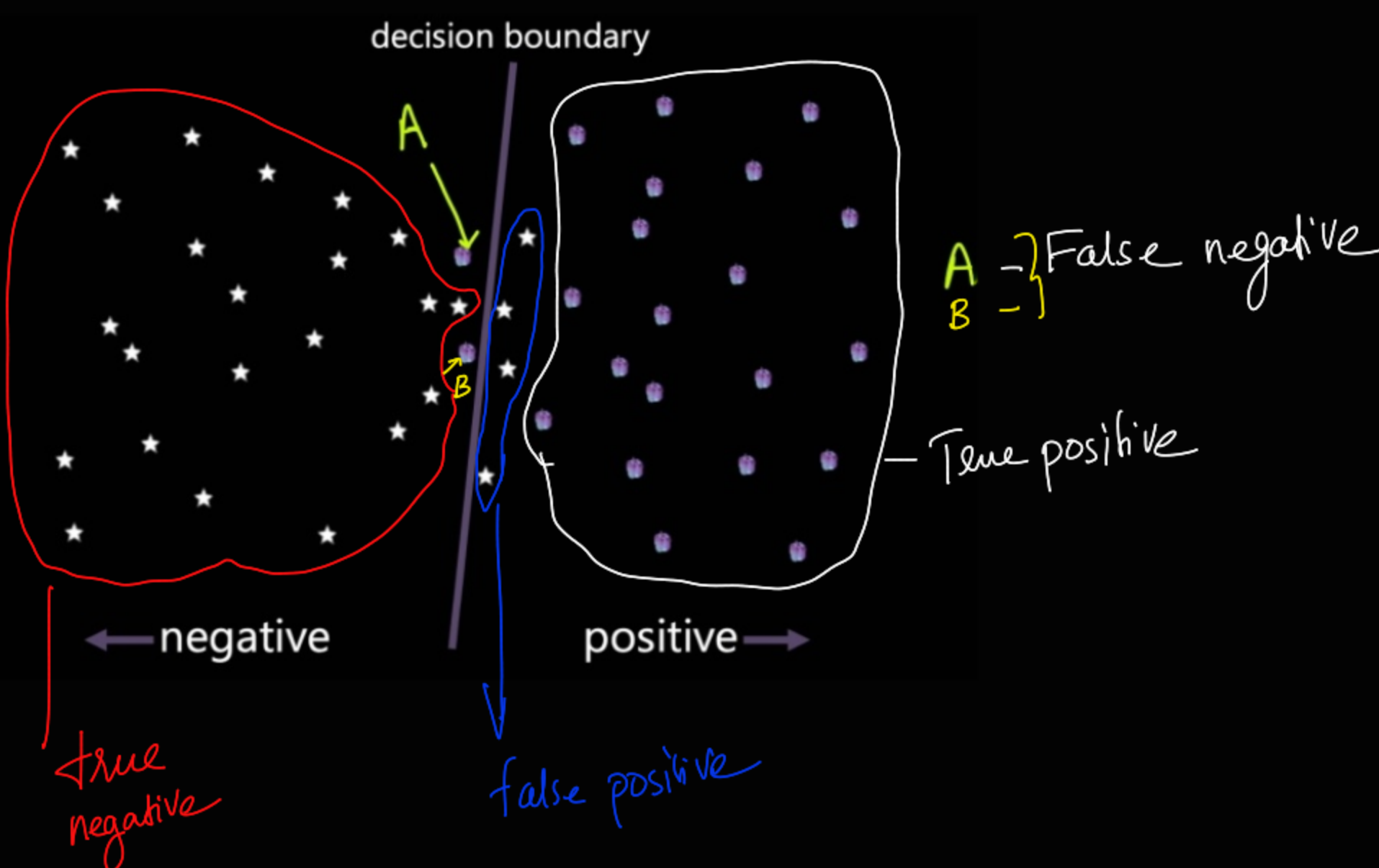
Precision > Recall $\Rightarrow$ False positive < False negative
 Recall > Precision $\Rightarrow$ False negative < False positive

Given the following information about a classifier's performance on a dataset of 12 pictures (8 cats and 4 dogs), where cats belong to class 1 and dogs belong to class 0:

- Actual labels:
actual = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]
- Predicted labels: $\times \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark \checkmark \times$
prediction = [0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1]

Using this data, can you construct the confusion matrix to summarize the classifier's results?

		Predicted	
		+	-
Actual	+	6	2
	-	1	3



	Predicted +	Predicted negative
Actual +	TP	FP
Actual -	FP	TN

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

ϕ Consider classifier which labels everything 1
 Precision goes down
 Recall = 1

ϕ Consider a classifier which labels everything 0.
 Precision = $\frac{0}{0}$ by defn = 1 (remember, for precision if denominator = 0 then Precision = 1)
 Recall = $\frac{0}{\text{sum}} = 0$

Performance Metrics

Four Metrics

- Accuracy = $\frac{TP + TN}{TP + TN + FP + FN}$ The percentage of the correctly classified spams and non-spams in the dataset
- Precision = $\frac{TP}{TP + FP}$ The percentage of the real spams in the classified spams
- Recall = $\frac{TP}{TP + FN}$ The percentage of the truly classified spams in the real spams
- F1 Score = $2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

Actual

	Prediction		
	dog	cat	mouse
dog	4	0	0
cat	2	2	0
mouse	0	0	2

$$\text{Cat Precision} = \frac{2}{0+2+0} = 1$$

$$\text{Dog Precision} = \frac{4}{4+2} = \frac{2}{3}$$

$$\text{Cat Recall} = \frac{2}{2+2} = 1/2$$

$$\text{Dog Recall} = \frac{4}{4} = 1$$

COMBINING precision & Recall scores for Comparing models

→ F1-SCORE = $\frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$ → This is the harmonic average

→ Taking average alone is not a good indicator of model metrics
 eg:- Model A is always -ve classifier
 Model B is having 0.5 precision & 0.5 Recall } Avg for both case is 0.5
 But it is obvious that model B is better than dumb model A.

Instead here if we use F1 score, Model A gets 0 & Model B gets 0.5

$$F_{\text{Score}}(\beta) = \frac{(1+\beta^2) \cdot P \cdot R}{\beta^2 P + R}$$

$$F1_{\text{score}} = F_{\text{Score}}(\beta=1) = \frac{2 \cdot P \cdot R}{P + R}$$

- β controls the balance:
- $\beta > 1$: recall is given more weight.
- $\beta < 1$: precision is given more weight.
- $\beta = 1$: equal weight.

→ For any β

$$F_{\beta} = \frac{(1+\beta^2) \cdot P \cdot R}{\beta^2 P + R} = \frac{P \cdot R (1+\beta^2)}{P \cdot R (\frac{\beta^2}{R} + \frac{1}{P})}$$

$$= \frac{1+\beta^2}{\left(\frac{1}{R} + \frac{1}{R} + \dots + \frac{1}{R} \text{ } \beta^2 \text{ times}\right) + \frac{1}{P}} = \frac{n}{\text{sum of } n \text{ terms where each term } > 1} = \frac{n}{\text{atleast } n}$$

→ F_{β} score is always < 1 i.e. $0 < F_{\beta} < 1$

Numerator in F1 score = Product → Punish Model
 Denominator in F1 score = Average → Extreme
 eg: Prec = 1, Recall = 0
 or Prec = 0, Recall = 1

F1 score can also be defined as

$$F1_{\text{score}} = \frac{2}{\frac{1}{\text{prec}} + \frac{1}{\text{Recall}}} \quad \left. \begin{array}{l} \text{Harmonic Mean} \\ \text{used for rates/ratios} \end{array} \right\}$$

⇒ F1 score always between 0 & 1 since it is harmonic mean

Sara can't believe everyone she knows only decided to watch one of Barbie or Oppenheimer. She collects data and gives each person a label of $y_i = 1$ if a person watched Barbie, and $y_i = 0$ if a person watched Oppenheimer. Assume that everybody in the dataset only watched one of the movies.

Suppose Sara believes that wrongly predicting Barbie is worse than wrongly predicting Oppenheimer.

Which evaluation metric is the **BEST** option to **maximize** in this scenario?

- ☐ A. Accuracy
- ☒ B. Precision
- ☐ C. True Positive Rate
- ☐ D. False Positive Rate

Really a good question

Which quantity should she aim to **minimize**?

- ☐ A. True Positives
- ☒ B. False Positives
- ☐ C. True Negatives
- ☐ D. False Negatives

Wrongly predicting barbie

True 0 for false 1
 so we need to reduce False +ve
 ⇒ Maximise precision

		actual	
		positive	negative
predicted class	positive	true positives (TP)	false positives (FP)
	negative	false negatives (FN)	true negatives (TN)

$\frac{TP}{TP+FP} \rightarrow \text{precision}$

Definitions of True positive rate & False positive rate

true positive rate (recall) = $\frac{TP}{\text{actual pos}} = \frac{TP}{TP+FN}$

false positive rate = $\frac{FP}{\text{actual neg}} = \frac{FP}{TN+FP}$

$1 - \text{FPR} = \text{Specificity} = \frac{TN}{TN+FP}$

	Predicted: NO	Predicted: YES
Actual: NO	50	10
Actual: YES	5	100

$\text{TPR} = \text{Recall} = \frac{100}{105}$

$\text{Precision} = \frac{100}{110}$

False positive rate = $\text{FPR} = \frac{10}{60}$

Specificity = $\frac{50}{60}$

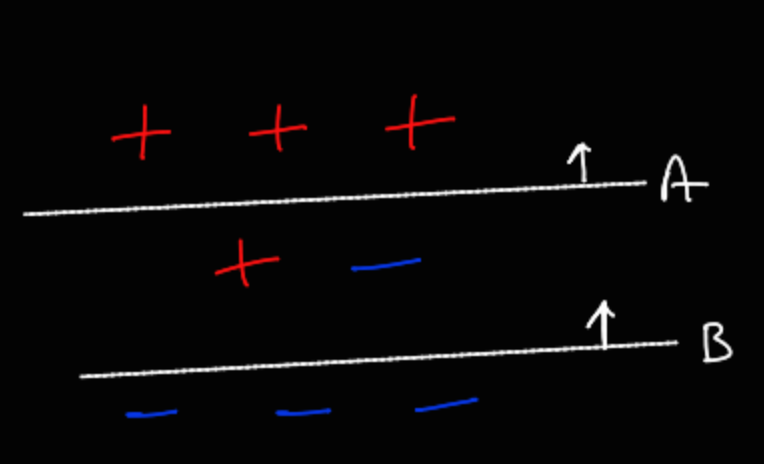
Accuracy = $\frac{150}{165}$

Guideline: Identify the row/column first & then write numerator, denominator is sum of row/column.

(i) (1 point) (T / F) In any binary classification problem, it is always possible to create a classifier that achieves a FPR of 0 by classifying every instance as positive.

Always true classifier FPR=1
All ground truth negatives will become false pos

if we instead classify everything as negative then FPR=0

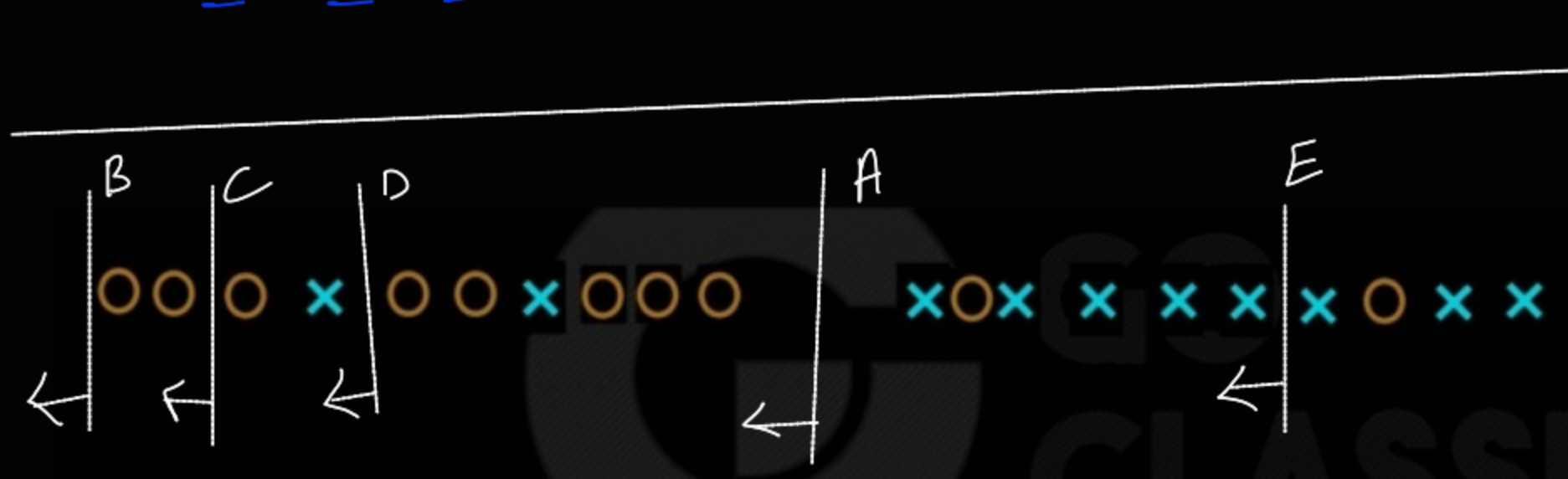


$\text{FPR}(A) = 0$

$\text{TPR}(A) = 3/4$

$\text{FPR}(B) = 1/4$

$\text{TPR}(B) = 4/4 = 1$



A	B	C	D	E
$\text{TPR} = \frac{8}{10}$	$\text{TPR} = 0$	$\text{TPR} = 2$	$\text{TPR} = \frac{3}{10}$	$\text{TPR} = \frac{9}{10}$
$\text{FPR} = \frac{2}{10}$	$\text{FPR} = 0$	$\text{FPR} = 0$	$\text{FPR} = \frac{1}{10}$	$\text{FPR} = \frac{1}{10}$

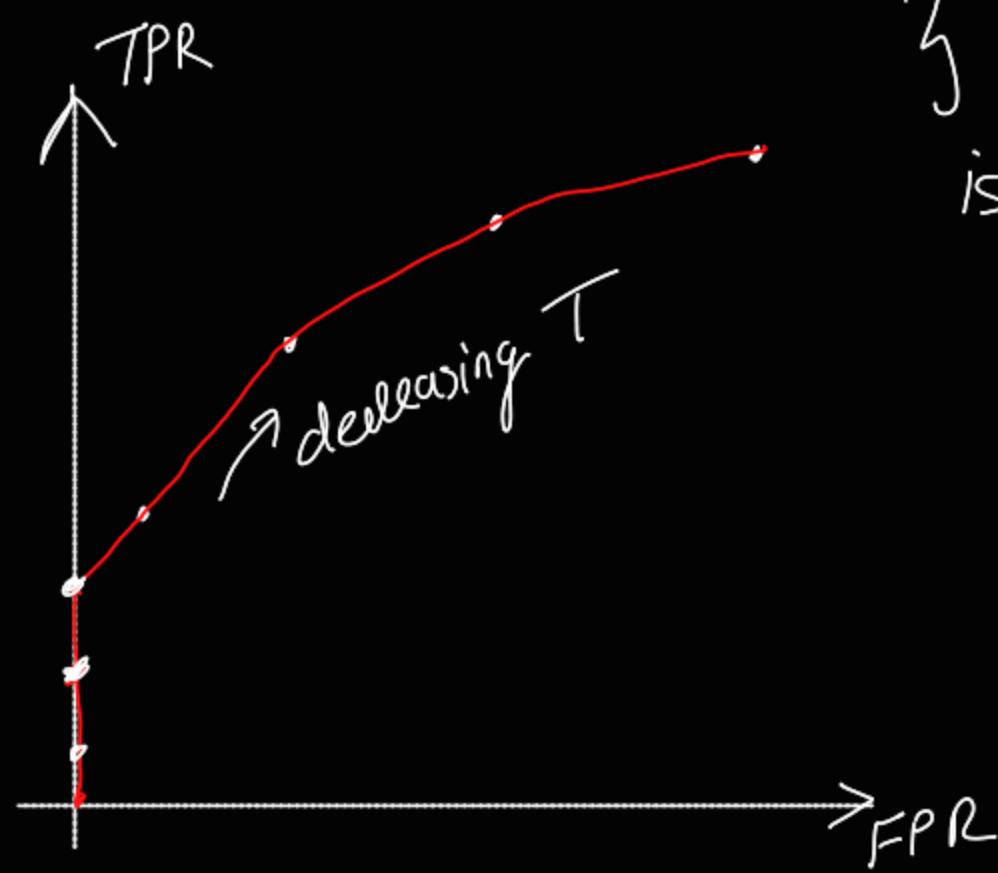
○ Positive
× Negative

Intuition — shifting the Decision Boundary from left to right only increases TPR & FPR since the denominators are constant while shifting.

One way to do this without changing the learned hyperplane is by changing the threshold

So by sliding the threshold we get different performance. We can tune this theta to get the balance/trade off between good precision & good recall.

ROC Curve



} plot of TPR vs FPR as the threshold of the decision boundary is changed.

① Binary classifier based on probabilistic model

$$\hat{y} = \begin{cases} +ve & P(y/x) > T \\ -ve & P(y/x) < T \end{cases} \quad \text{where } T \rightarrow \text{threshold}$$

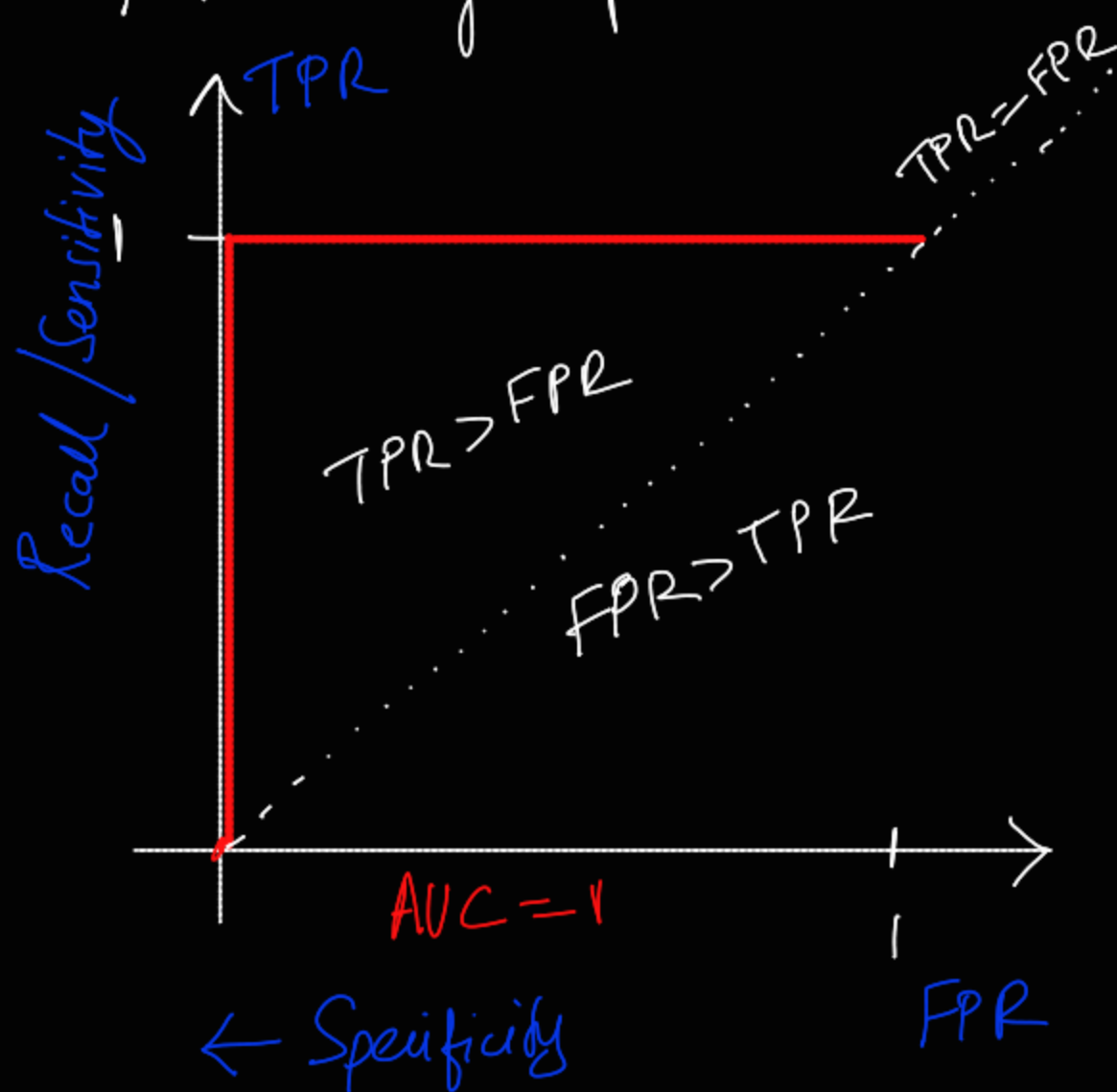
② Binary classifier based on geometric model

$$\hat{y} = \begin{cases} +ve & W^T x + b > T \\ -ve & W^T x + b < T \end{cases} \quad \text{where } T \rightarrow \text{threshold}$$

Intuition

- At the maximum value of threshold T in both cases, there are no +ve predictions
- This implies TPR & FPR is zero. Now as we decrease threshold, more and more datapoint gets classified as +ve (eitherly truly $\rightarrow$ TPR; or falsely $\rightarrow$ FPR).
- Therefore decreasing threshold from the top increases both TPR & FPR. ROC curve is always monotonically increasing.

For linearly separable data.



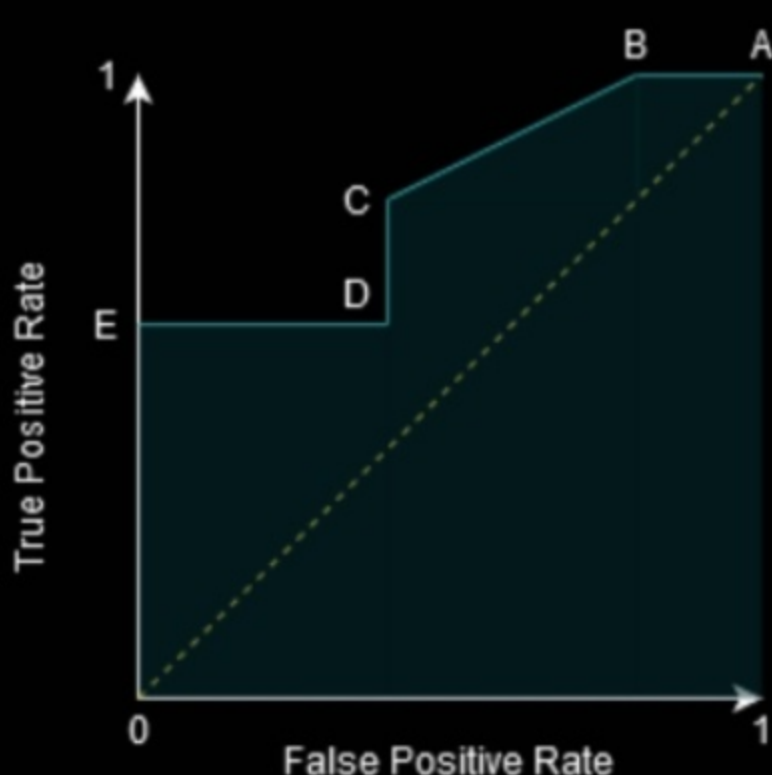
In the context of binary classification, if we continuously decrease the classifier threshold, how do the True Positive Rate (TPR) and False Positive Rate (FPR) behave?

- A) TPR increases while FPR decreases
- B) TPR decreases while FPR increases
- ☒ C) Both TPR and FPR increase
- D) Both TPR and FPR decrease

- Each ROC curve starts at (0,0) $\rightarrow$ classify all as -ve
- Each ROC curve ends at (1,1) $\rightarrow$ classify all as +ve
- ROC curve of perfect classifier goes through (0,1) } $AUC=1$
- if ROC is able to pass through (0,1) then $\Rightarrow$ data is linearly separable for some threshold
the model has to be correctly tuned also.

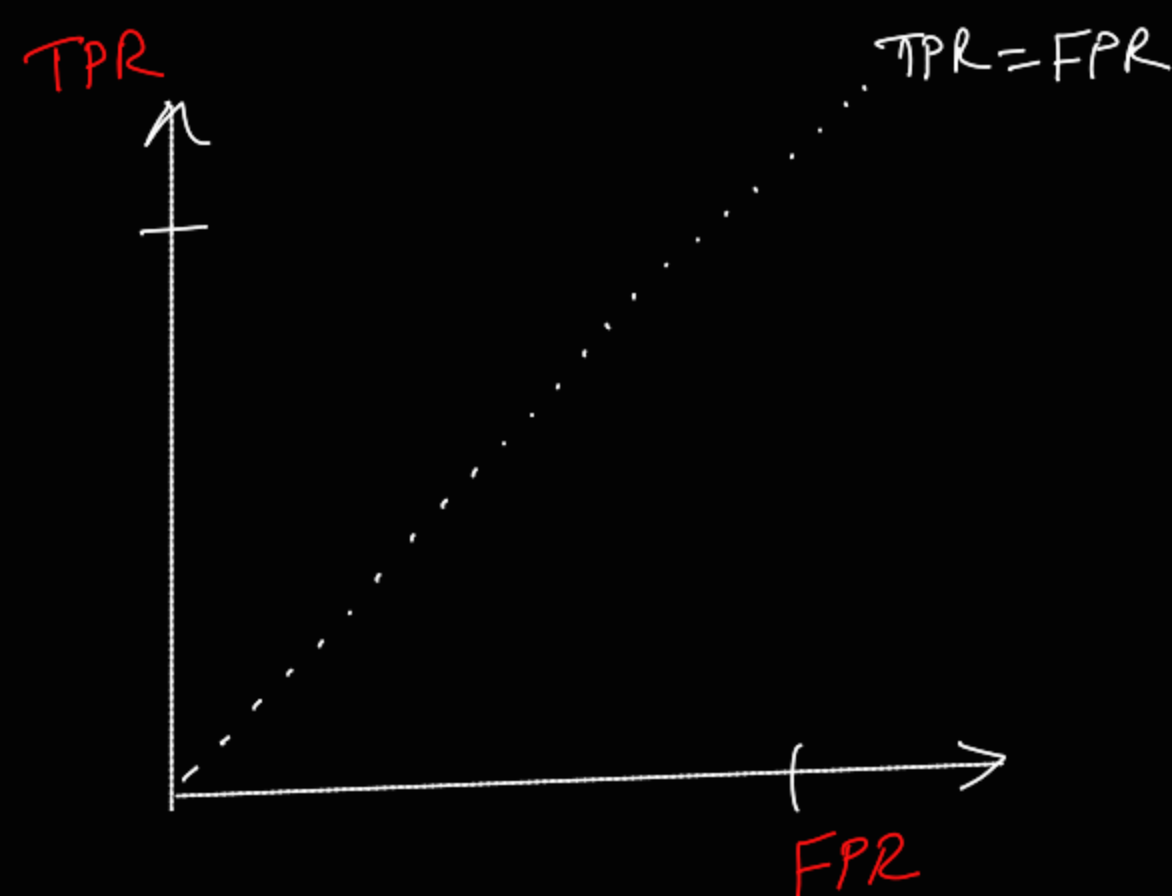
- Point A is where the Sensitivity is the highest.
- This means all the Positive class points are classified correctly and all the Negative class points are classified incorrectly.

- Although Point B has the same Sensitivity as Point A, it has a higher Specificity.
- Meaning the number of incorrectly Negative class points is lower compared to the previous threshold.
- This indicates that this threshold is better than the previous one.



- Between points C and D, the Sensitivity at point C is higher than point D
- This means, for the same number of incorrectly classified Negative class points, the classifier predicted a higher number of Positive class points.
- Therefore, the threshold at point C is better than point D.
- Now, depending on how many incorrectly classified points we want to tolerate for our classifier, we would choose between point B or C for predicting whether you can defeat me in PUBG or not.

Comparison among curves $\rightarrow$ Based on AUC (Area under the curve)



AUC for $TPR=FPR=0.5$

So $AUC > 0.5 \Rightarrow TPR > FPR$ for atleast some threshold.

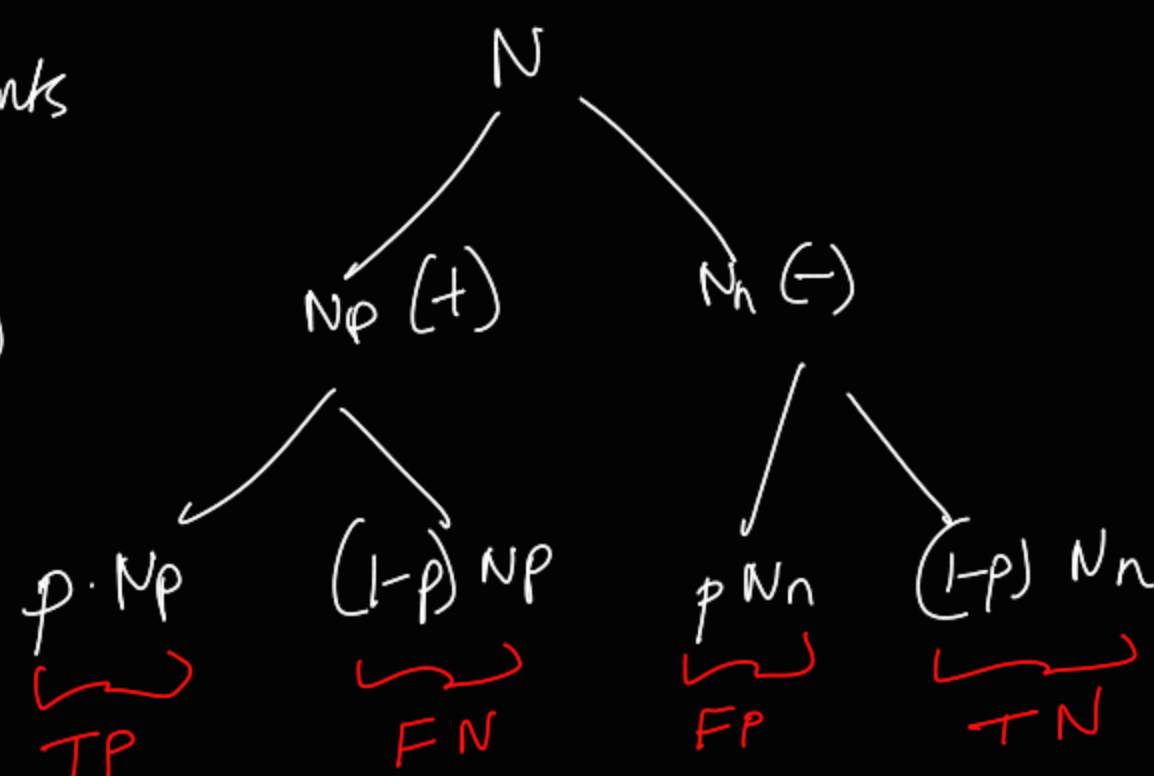
RANDOM CLASSIFIER

Imagine N datapoints

N_p # Actual (+)

N_n # Actual (-)

Random classifier
with $P(Y=1) = P$

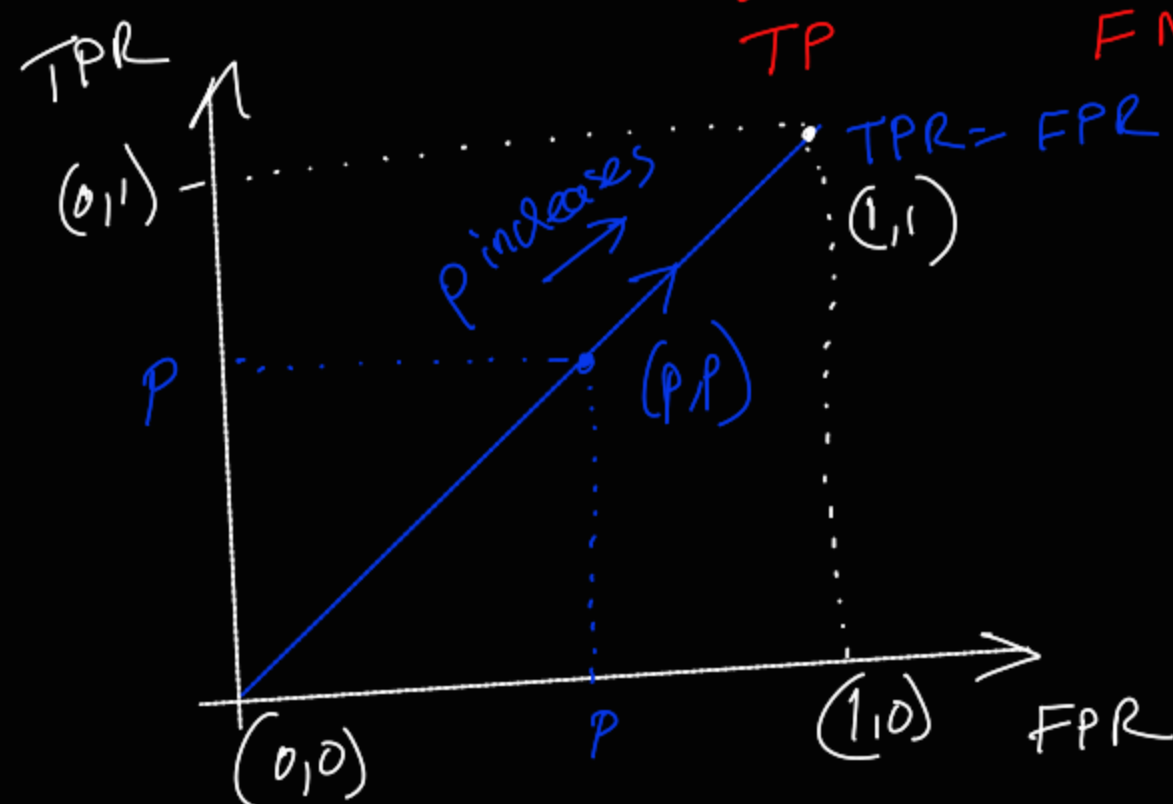


$$TPR = \frac{TP}{TP + FN} = \frac{pN_p}{pN_p + (1-p)N_p} = p$$

$$FPR = \frac{FP}{FP + TN} = \frac{pN_n}{pN_n + (1-p)N_n} = p$$

$\Rightarrow$ Random classifier $TPR = FPR = p$
where $p = P(Y=1|X) = P(Y=1)$

$\hookrightarrow$ For random classifier
 Y independent of X



Intuition

- $\rightarrow$ AUC for random classifier = 0.5
- $\rightarrow$ Models with $AUC > 0.5$ are better than random classifier
- $\rightarrow$ Models with $AUC < 0.5$ are worse than random classifier

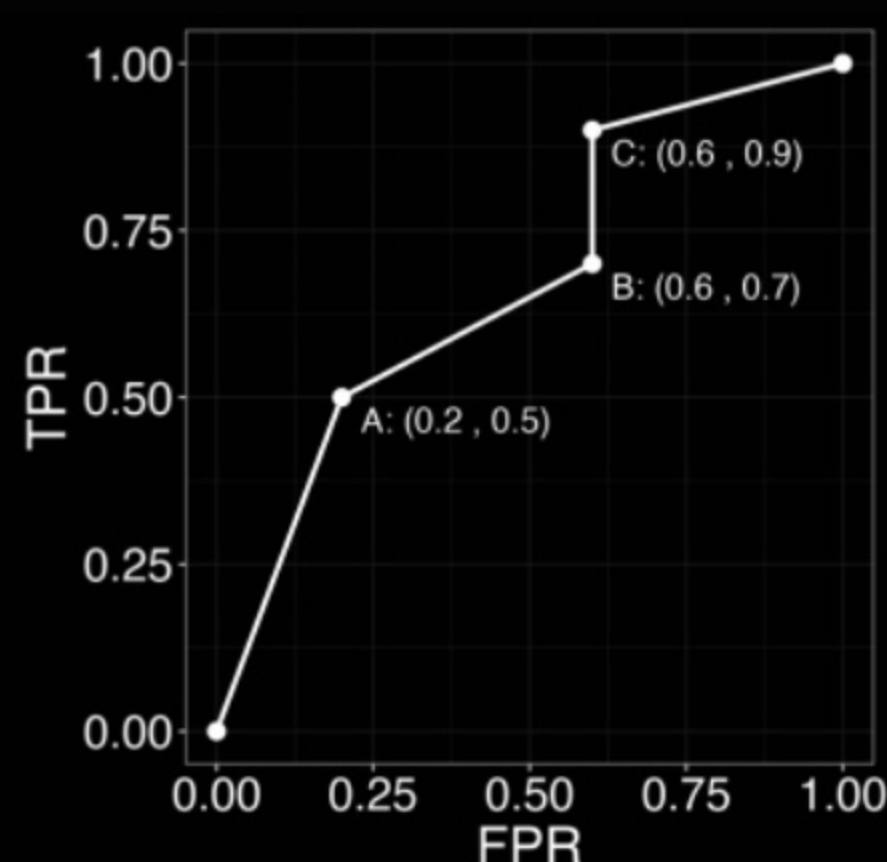
$AUC > 0.5$

$\downarrow$
Learn something
about data

$AUC < 0.5$

$\downarrow$
Learn the
exact opposite
from data

SSSES



(b) (2 points) Fill in the blanks below and explain your answer.

Among the points A, B, and C, C is strictly better than B.

BIAS VARIANCE

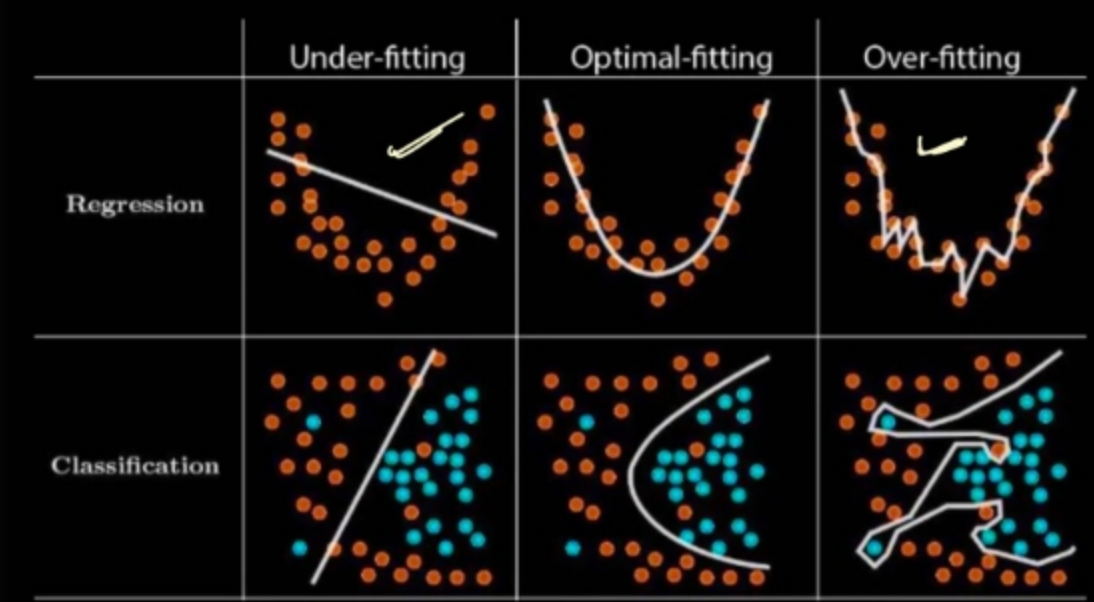
1. Bias and Variance: Core Idea

- Bias** = Error from wrong assumptions in the learning algorithm.
 - High bias → underfitting (model too simple, ignores data complexity).
 - Example: fitting a straight line to quadratic data.
- Variance** = Error from sensitivity to training data fluctuations.
 - High variance → overfitting (model too complex, learns noise).
 - Example: fitting a high-degree polynomial to noisy data.
- Generalization Error** = Test error = Bias² + Variance + Irreducible Noise.
 - Irreducible noise comes from randomness in data → cannot be removed.

$$\text{Let } f_{\mathcal{D}}(x) = E_{\text{over data points}} [\hat{f}(x)]$$

$$\text{Bias} = f_{\mathcal{D}}(x) - f(x)$$

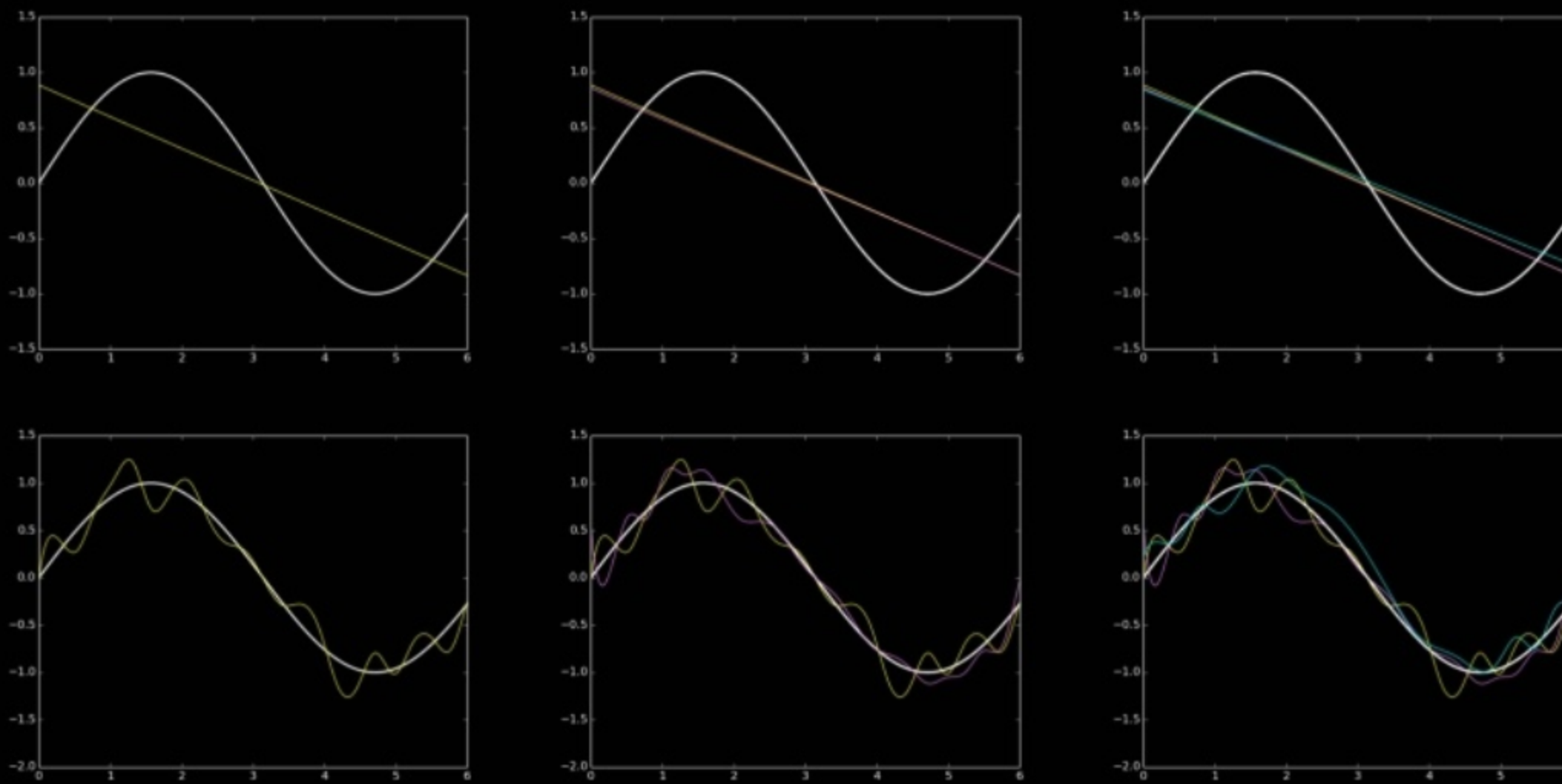
$$\text{Variance} = E[(f_{\mathcal{D}}(x) - \hat{f}(x))^2]$$



$f(x)$ → true function unknown

$\hat{y} = \hat{f}(x)$ → predicted model

$$\text{Generalization Error} = \text{test error} = \text{Bias}^2 + \text{Variance} + \text{irreducible noise}$$



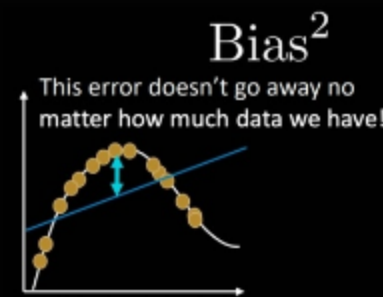
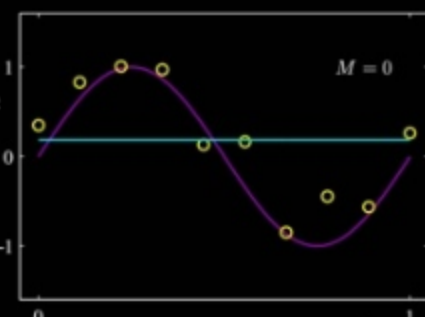
$E[\hat{f}(x)] = f_{\mathcal{D}}(x)$
average model over all possible data points in training

$y \neq f(x)$
 $y = f(x) + \epsilon$ → Noise gaussian

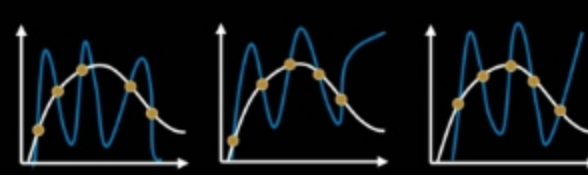
- for simpler models $f_{\mathcal{D}}(x)$ very far away from $f(x)$
- for complex model $f_{\mathcal{D}}(x)$ very close to $f(x)$
- However complex model has high variance over all possible data points
- Simpler models tend to have less variance

- Model too simple:** does not fit the data well
— A biased solution

- Model too complex:** small changes to the data, solution changes a lot
— A high-variance solution



Bias²
This error doesn't go away no matter how much data we have!



IRREDUCIBLE NOISE ← Assumption → LLN, CLT, Most Common

$$\epsilon \sim N(0, \sigma^2)$$

$$E(\epsilon) = 0$$

$$\text{Var}(\epsilon) = \sigma^2$$

$$y_i = f(x_i) + \epsilon_i$$

$$\text{Var}(y_i) = \text{Var}(f(x_i) + \epsilon_i)$$

$$\Rightarrow \text{Var}(y_i) = \text{Var}(\epsilon_i) = \sigma^2$$

$$E(y_i) = E(f(x_i) + \epsilon_i) = E(f(x_i)) + 0$$

Since $f(x_i)$ independent of ϵ

$$E(y_i) = f(x_i)$$

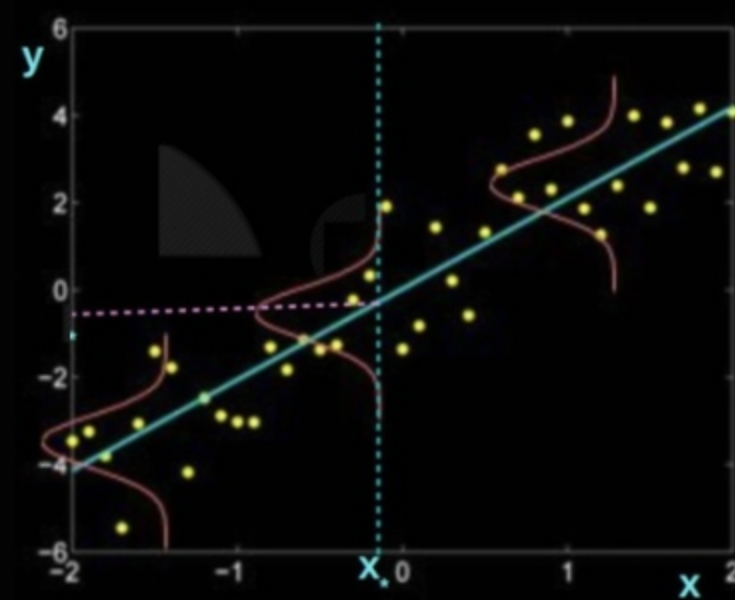
For the i^{th} point $f(x_i)$ is constant

$$\Rightarrow E(\epsilon) = 0$$

$$\text{Var}(\epsilon) = \sigma^2$$

$$\text{Var}(y_i) = \sigma^2$$

$$E(y_i) = f(x_i)$$



2. Effects of More Data

- High variance models benefit most → variance reduces, generalization improves.
- High bias models won't improve much (data can't fix a fundamentally too-simple model).

👉 More data reduces variance, not bias.

3. Effect of More Regularization

- Regularization (L1/Lasso, L2/Ridge, dropout, etc.) penalizes complexity.
 - Increases bias (simpler model, less flexible).
 - Decreases variance (prevents overfitting).

👉 Helps when variance is too high.

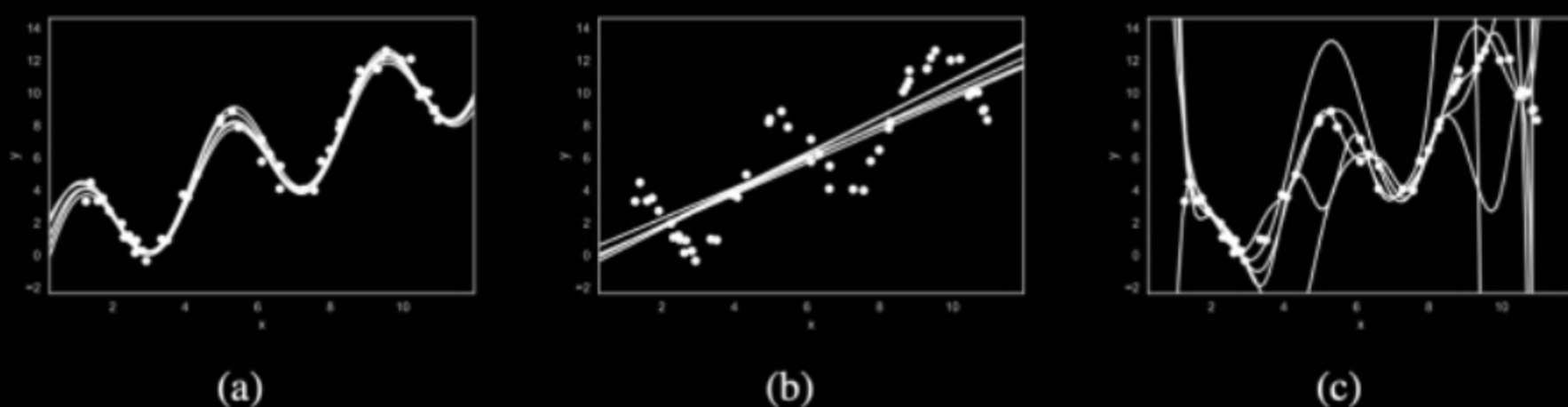
4. Effect of Cross Validation

- Used for model selection and hyperparameter tuning.
- Reduces variance in performance estimates (compared to a single split).
- Doesn't directly reduce bias/variance of the model, but helps you choose the best balance.

5. Bias-Variance Trends of Models

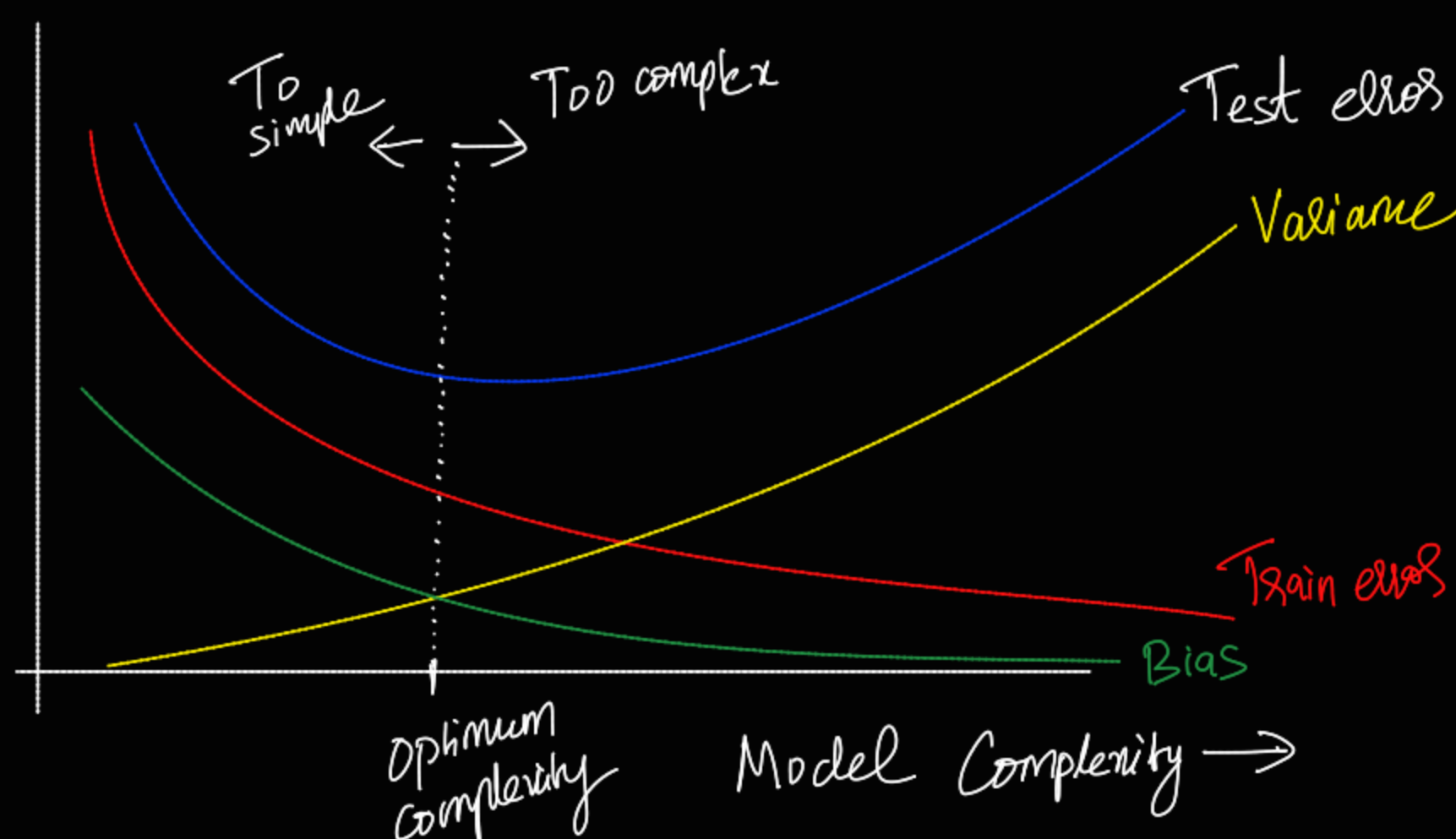
- Linear Regression (without regularization):**
 - Low variance, can have high bias if data is non-linear.
- Polynomial Regression (degree 1):**
 - Bias ↓, variance ↑ with degree.
- k-Nearest Neighbors (kNN):**
 - Small k → low bias, high variance (overfits).
 - Large k → high bias, low variance (underfits).
- Decision Trees:**
 - Deep trees → low bias, high variance.
 - Shallow trees → high bias, low variance.
- Random Forests (Ensembles):**
 - Lower variance than single trees (averaging effect).
 - Bias similar to deep trees (depends on depth).
- Naive Bayes:**
 - High bias (independence assumption), low variance.
- SVM:**
 - With large margin (C small) → high bias, low variance.
 - With small margin (C large) → low bias, high variance.
- Neural Networks:**
 - Small/under-trained → high bias.
 - Large/deep → low bias, but high variance if not regularized.

[1 Pt] Which of the following plots depicts models with the highest model variance?

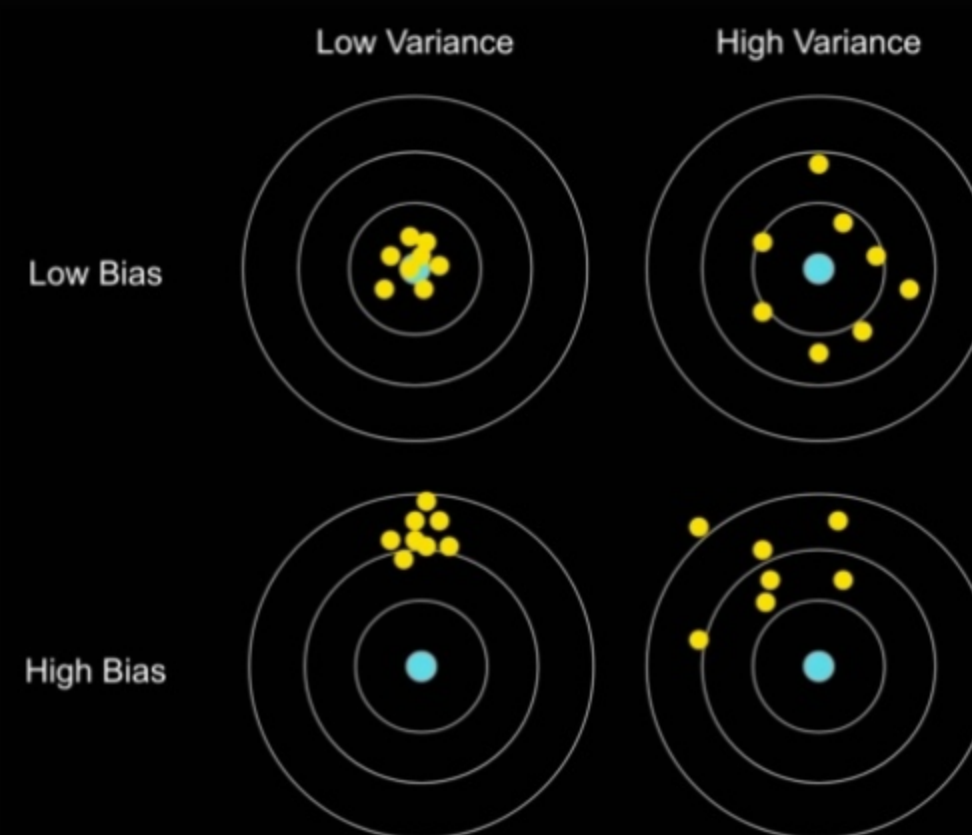


High bias High variance

Bias Variance vs error



$$\text{Test error} = \text{Bias}^2 + \text{Variance}$$



Underfitting and Overfitting



Question:



Figure 1: The following graphic will be used as a representation of bias and variance. Imagine that a true/correct model is one that always predicts a location at the center of each target (being farther away from the center of the target indicates that a model's predictions are worse). We retrain a model multiple times, and make a prediction with each trained model. For each of the targets, determine whether the bias and variance is low or high with respect to the true model.

In Figure 1, subplot I how are bias and variance related to the true model?

- (A) High bias, High variance IV
- (B) High bias, Low variance III
- (C) Low bias, High variance II
- (D) Low bias, Low variance I

[1 points] What best defines the relationship between a model's fit and its bias and variance

- (a) A model with low bias and high variance is underfitting
- (b) A model with high bias and high variance is underfitting
- (c) A model with low bias and low variance is underfitting
- (d) A model with high bias and low variance is underfitting ✓

How will regularizing the weights in a linear regression model change the bias and variance (relative to the same model with no regularization)?

- (A) Increase bias, increase variance
- (B) Increase bias, decrease variance ✓
- (C) Decrease bias, increase variance
- (D) Decrease bias, decrease variance

Regularization
↓

(6 pts) Suppose you have regression data generated by a polynomial of degree 3. Characterize the bias-variance of the estimates of the following models on the data with respect to the true model by circling the appropriate entry.

	Bias	Variance
Linear regression	low/high	low/high
Polynomial regression with degree 3	low/high	low/high
Polynomial regression with degree 10	low/high	low/high

62/10

[2 points] Suppose we have a regularized linear regression model: $\arg\min_w \|Y - Xw\|_2^2 + \lambda \|w\|_1$. What is the effect of increasing λ on bias and variance?

- (a) Increases bias, increases variance
- ✓ (b) Increases bias, decreases variance
- (c) Decreases bias, increases variance
- (d) Decreases bias, decreases variance
- (e) Not enough information to tell

[2 points] Suppose we have a regularized linear regression model: $\arg\min_w \|Y - Xw\|_2^2 + \|w\|_p$. What is the effect of increasing p on bias and variance ($p \geq 1$) if the weights are all larger than 1?

- (a) Increases bias, increases variance
- ✓ (b) Increases bias, decreases variance
- (c) Decreases bias, increases variance
- (d) Decreases bias, decreases variance
- (e) Not enough information to tell

(1 point) Peanut wants to train a model to accurately classify different types of animals from images. After training and testing his model, he observes that the model has high training error and high test error. What can we most confidently say about the bias/variance characteristics of Peanut's model?

- A. High bias. ✓
- B. Low bias.
- C. High variance.
- ✓ (D) Low variance.

High training error & High test error → Simple model

Also true in case of MSR

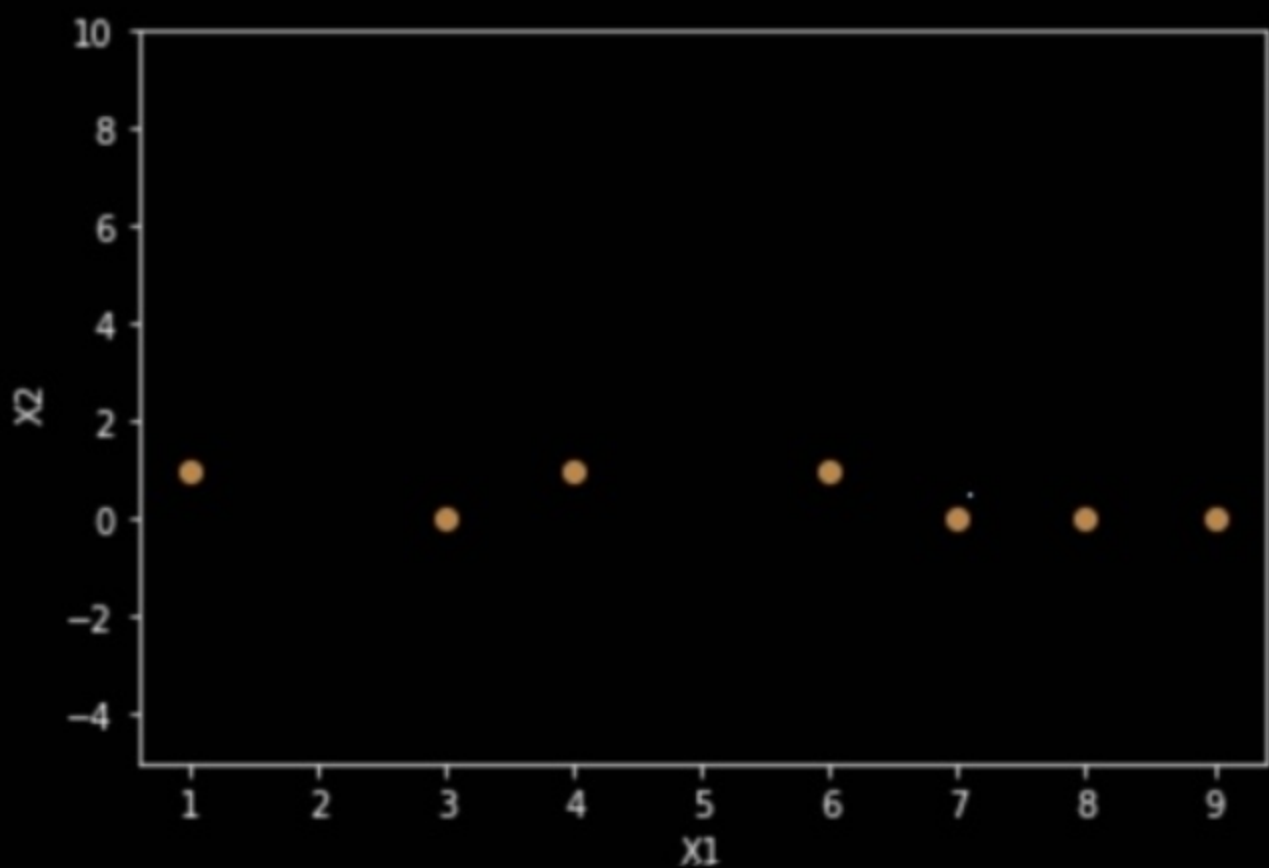
Which of the following can impact our model variance? Select all that apply.

- ☒ The regularization coefficient λ . $\rightarrow \lambda \downarrow \Rightarrow \text{Variance} \uparrow$
- ☒ The choice of features to include in our design matrix. $\rightarrow \text{Complex features} \Rightarrow \text{Complex model} \Rightarrow \text{High Variance}$
- ☒ The learning rate α in gradient descent.
- ☒ The size of the training set. $\rightarrow \text{More data} \Rightarrow \text{Variance decreases} \Rightarrow \text{Less change for overfitting \& learning complex model.}$

Which of the following are indications that you should regularize? Select all that apply.

- ☒ A. Our training loss is 0.
 - ☒ B. Our model bias is too high.
 - ☒ C. Our model variance is too high.
 - ☒ D. Our weights are too large.
- indicators of complex model

[1 Pt] Parth now decides to use only the two most important features, X_1 and X_2 , in the dataset to predict weekly watch time. Given below is the scatterplot of the two feature values along a representative sample of the dataset. Recall Parth's model was based on ridge regression. From the plot below, how would the model bias be affected if Parth were to use a LASSO regression model instead?



$\rightarrow$ More penalty
Earlier: Ridge
 $\rightarrow$ Less penalty
Now: LASSO
 $\Rightarrow$ Slight more complex
 $\Rightarrow$ More Variance
 $\Rightarrow$ Less Bias

[1 Pt.] When conducting linear regression, adding polynomial features to your data often decreases the variance of your fitted model. $\rightarrow \text{false}$

$\rightarrow$ Above leads to bias decreasing $\rightarrow$ True

[1 Pt.] Suppose your data are an i.i.d. sample from a population. Then collecting a larger sample for use as a training set can help reduce variance. True

More data $\Rightarrow$ Reduces Variance

Suppose that we are trying to fit a linear and 10th degree polynomial to data coming from a cubic function, corrupted by standard Gaussian noise. Let M_1 and M_2 denote the models corresponding to the linear and 10 degree polynomial. Then,

- (a) $\text{Bias}(M_1) \leq \text{Bias}(M_2)$, $\text{Variance}(M_1) \leq \text{Variance}(M_2)$.
- (b) $\text{Bias}(M_1) \leq \text{Bias}(M_2)$, $\text{Variance}(M_1) \geq \text{Variance}(M_2)$.
- ☒ (c) $\text{Bias}(M_1) \geq \text{Bias}(M_2)$, $\text{Variance}(M_1) \leq \text{Variance}(M_2)$.
- (d) $\text{Bias}(M_1) \geq \text{Bias}(M_2)$, $\text{Variance}(M_1) \geq \text{Variance}(M_2)$.

True: cubic

For ridge regression, how will the bias and variance in our estimate $\hat{w}$ change as the number of training examples N increases? Assume the regularization parameter λ is fixed.

- (a) $\downarrow$ bias, $\uparrow$ variance
- ☒ (b) same bias, $\downarrow$ variance
- (c) same bias, $\uparrow$ variance
- (d) $\downarrow$ bias, $\downarrow$ variance
- (e) same bias, same variance

More training examples generally doesn't affect the bias

[1 Pt.] Suppose your data are an i.i.d. sample from a population. Then collecting a larger sample for use as a training set can help reduce bias.

Solution: False. Increasing the dataset size without changing the modeling procedure can often reduce variance but is unlikely to address bias.

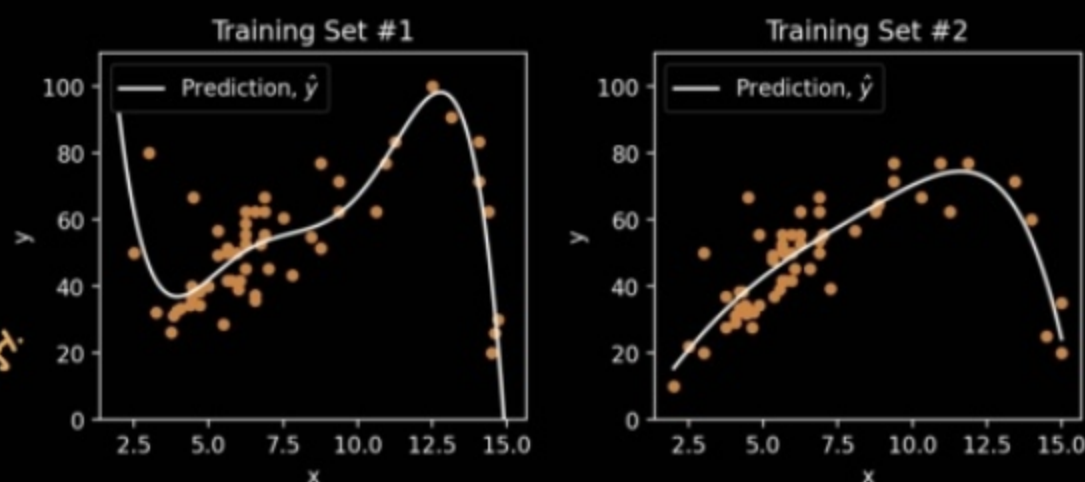
Which of the following is most indicative of a model overfitting?

- (a) High bias, low variance
- ☒ (b) Low bias, high variance
- (c) Low bias, low variance

Ridge regression

- ☒ (a) reduces variance at the expense of bias
- (b) adds an l_1 penalty norm to the cost function
- (c) often sets many of the weights to 0 when the regularization parameter λ is very large
- (d) is more sensitive to outliers than least squares

Consider a quintic (degree 5) model $\hat{y} = \hat{q}_0 + \hat{q}_1x + \dots + \hat{q}_5x^5$, where $\hat{q}_j$ is the MSE parameter estimate for feature j of this quintic model. You first fit this model to Training Set #1; then, you fit this model again to Training Set #2.



- (i) The primary contributor to this model's risk is: ☐ High bias ☒ High variance
- (ii) This contributor to model risk can be addressed by the following (select all that apply):
 - ☒ Reducing model complexity
 - ☒ Adding additional features to the model

$\rightarrow$ Increasing regularization

We obtain n data points (n is some large fixed integer) which have been generated from the true model $Y = f(x) + \epsilon$, where ϵ is random noise ($\mathbb{E}[\epsilon] = 0$, $\text{Var}(\epsilon) = \sigma^2$).

We fit linear models of varying complexity to our data, and plotted the bias, variance, and irreducible error below.



- (a) [1 1/2 Pts] Sketch the MSE on the above graph. Where does its minimum occur? Draw a star on your MSE plot where the minimum occurs.

- (a) [1 1/2 Pts] Sketch the MSE on the above graph. Where does its minimum occur? Draw a star on your MSE plot where the minimum occurs.
- (b) [1 Pt] Suppose we control the complexity of the linear models using a Ridge penalty term $\lambda \sum \beta_i^2$. Which of the following is true?
- ☐ The left side of the graph represents small λ .
 - ☒ The right side of the graph represents small λ .
- (c) [3 Pts] Which of the following can impact our model variance? Select all that apply.
- ☒ The regularization coefficient λ .
 - ☒ The choice of features to include in our design matrix.
 - ☒ The learning rate α in gradient descent.
 - ☒ The size of the training set.

[3 pts] Recall the data model, $y_i = f(X_i) + \epsilon_i$, that justifies the least-squares cost function in regression. The statistical assumptions of this model are, for all i ,

- ☒ ϵ_i comes from a Gaussian distribution.
- ☐ all y_i have the same mean
- ☒ all ϵ_i have the same mean
- ☒ all y_i have the same variance



$$\text{MSE} = \underbrace{\text{Bias}^2}_{(f_p(x) - f(x))^2} + \underbrace{\text{Variance}}_{E[(f_p(x) - \hat{f}(x))^2]} + \underbrace{\epsilon}_{\sigma^2}$$

where $f_p(x) = E(\hat{f}(x))$ over all possible data points

In class, we showed that the expected squared error can be decomposed into several important terms:

$$[Y - f_{\hat{\theta}}(x)]^2 = \sigma^2 + (h(x) - E[f_{\hat{\theta}}(x)])^2 + [(E[f_{\hat{\theta}}(x)] - f_{\hat{\theta}}(x))^2].$$

- (a) [1.5 Pts] Which of the following do we treat as fixed? Select all that apply.

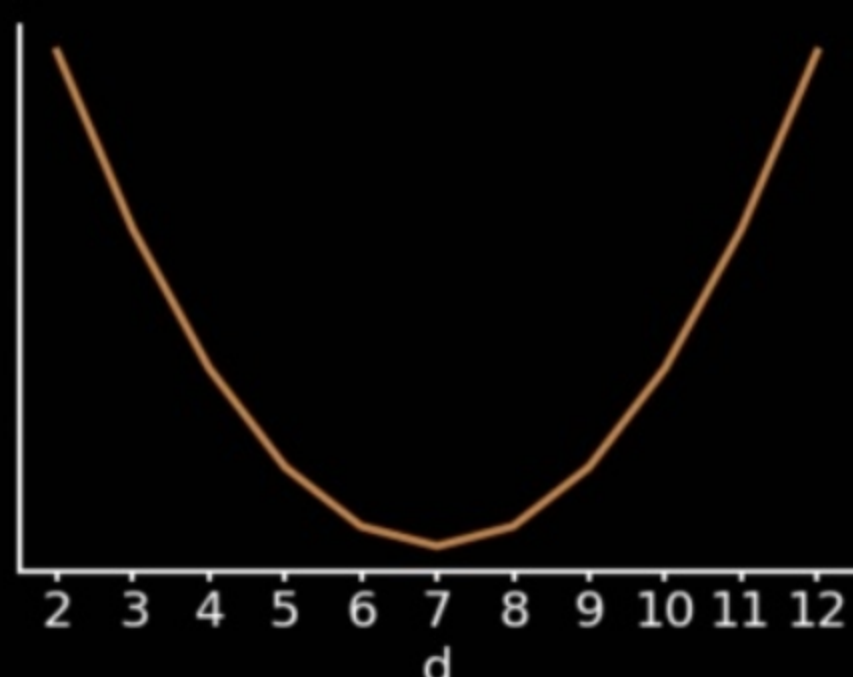
- ☐ A. $\hat{\theta}$
- ☒ B. σ^2
- ☒ C. $h(x)$

- (b) [1 Pt] By decreasing model complexity, we are able to decrease σ^2 .

- ☐ A. True
- ☒ B. False

Your team would like to train a machine learning model in order to predict the next YouTube video that a user will click on based on m features for each of the previous d videos watched by that user. In other words, the total number of features is $m \times d$. You're not sure how many videos to consider.

[2 Pts] Your colleague generates the following plot, where the value d is on the x axis. However, they forgot to label the y-axis.



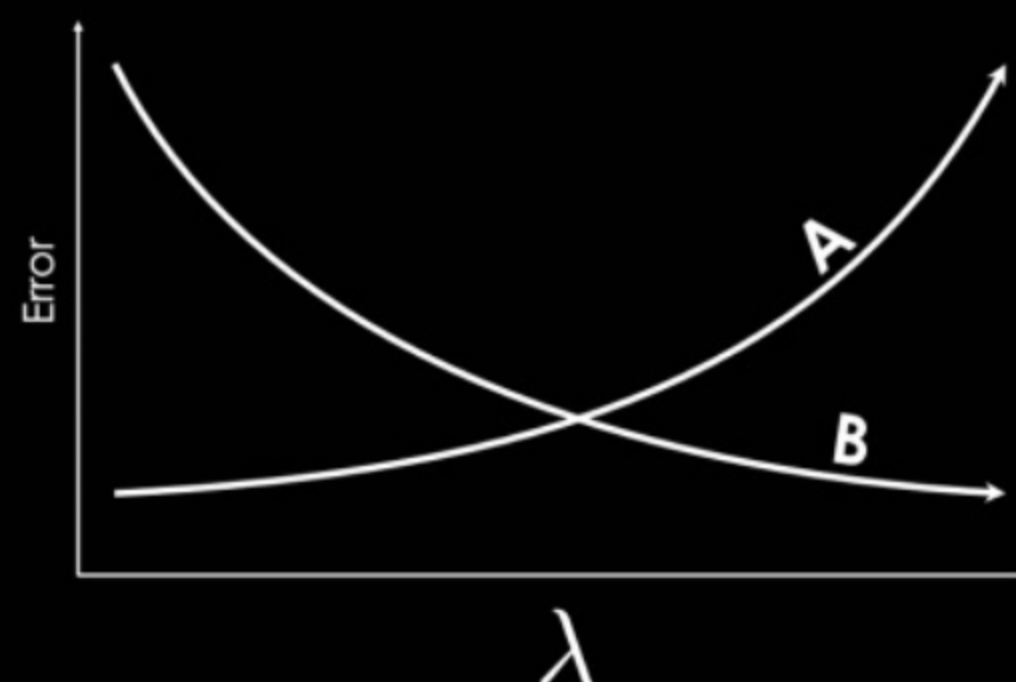
$d \rightarrow$
 Samp
 in

Which of the following could the y axis represent? Select all that apply.

- ☐ A. Training Error
- ☒ B. Validation Error
- ☐ C. Bias
- ☐ D. Variance

$\rightarrow$ generally approximates the test error
 $\rightarrow$ sh

[3 Pts] Assuming a regularization penalty of the form $\lambda R(\theta)$. Complete the following illustration. Note that the x-axis is the regularization parameter λ and not the model complexity.



- ☐ (A) is the **Test Error** and (B) is the error due to **(Bias)²**.
- ☐ (A) is the error due to **Model Variance** and (B) is the **Training Error**
- ☐ (A) is the error due to **Model Variance** and (B) is the error due to **(Bias)²**.
- ☒ (A) is the error due to **(Bias)²** and (B) is the error due to **Model Variance**.

BIAS VARIANCE DECOMPOSITION OF MSE

$$\text{MSE} = E[(y - \hat{f}(x))^2]$$

$\hat{f}(x) \rightarrow$ Learned model function

$\downarrow$
Over all the datapoints x_i
Over all the possible training set D

$$\text{MSE} = E_{x_0}[(y - \hat{f}(x))^2]$$

$$\text{MSE} = E[(f(x) - \hat{f}(x))^2] + 2E[(f(x) - \hat{f}(x)) \cdot \epsilon] + \sigma^2$$

$$\text{MSE} = E[(f(x) - \hat{f}(x))^2] + \sigma^2$$

$$\Rightarrow \text{MSE} = E\left[\underbrace{(f(x))}_a + \underbrace{\epsilon}_b - \underbrace{\hat{f}(x)}_a\right]^2$$

since $y = f(x) + \epsilon$
 $\epsilon \sim N(0, \sigma^2)$
 $E(\epsilon) = 0$
 $\text{Var}(\epsilon) = \sigma^2$
 $\text{Var}(y_i) = \sigma^2$
 $E(y_i) = f(x_i)$

$$= E[(f(x) - \hat{f}(x))^2] + 2E[(f(x) - \hat{f}(x)) \cdot \epsilon] + E(\epsilon^2)$$

$\rightarrow$ becomes 0

$$\text{Var}(\epsilon) = E(\epsilon^2) - E(\epsilon)^2$$

$$\Rightarrow \text{Var}(\epsilon) = E(\epsilon^2)$$

$$E[(f(x) - \hat{f}(x)) \cdot \epsilon] = E(\epsilon) \cdot E[f(x) - \hat{f}(x)]$$

since ϵ independent of $f(x)$ & $\hat{f}(x)$

$$E[(f(x) - \hat{f}(x))^2] \rightarrow \text{Bias}^2 + \text{Variance}$$

$$\text{Bias}^2 = (E[\hat{f}(x)] - f(x))^2$$

$$\text{Variance} = E[(\hat{f}(x) - E[\hat{f}(x)])^2]$$

$$E[(f(x) - \hat{f}(x))^2] = E\left[\underbrace{(f(x) - E[\hat{f}(x)])}_a + \underbrace{E[\hat{f}(x)] - \hat{f}(x)}_b\right]^2$$

Adding & Subtracting

$$= E\left[(f(x) - E[\hat{f}(x)])^2\right] + 2 \cdot E\left[(f(x) - E[\hat{f}(x)]) \cdot (E[\hat{f}(x)] - \hat{f}(x))\right] + E\left[(E[\hat{f}(x)] - \hat{f}(x))^2\right]$$

$E[\text{Bias}^2] = \text{Bias}^2$ $= 0$ Variance

$$= 2E\left[f(x) \cdot E[\hat{f}(x)] - f(x) \cdot \hat{f}(x) - E[\hat{f}(x)]^2 + \hat{f}(x) \cdot E[\hat{f}(x)]\right]$$

$$= 2 \cdot \left[\cancel{f(x) \cdot E[\hat{f}(x)]} - \cancel{f(x) \cdot E[\hat{f}(x)]} - \cancel{E[\hat{f}(x)]^2} + \cancel{E[\hat{f}(x)]^2} \right]$$

$$= 0$$

Since $f(x)$ is constant w.r.t $E_{x,p}[\]$
 $\Rightarrow$ i.e. $f(x)$ doesn't change over all possible training.

$$\Rightarrow \text{MSE} = E_{x,p}[(y - \hat{f}(x))^2] = \underbrace{(f(x) - E[\hat{f}(x)])^2}_{\text{Bias}^2} + \underbrace{E[(\hat{f}(x) - E[\hat{f}(x)])^2]}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Noise error}}$$

Generalisation Error = Test Error + Bias² + Variance + Noise error

★ In class we broke the least-squares error into three separate terms:

$$E[(y - f_\theta(x))^2] = E[(y - h(x))^2] + E[(h(x) - f_\theta(x))^2] + E[(f_\theta(x) - E[f_\theta(x)])^2]$$

where $y = h(x) + \epsilon$, $h(x)$ is the true model and ϵ is zero-mean noise. For each of the following terms, indicate its usual interpretation in the bias variance trade-off:

1. $E[(y - h(x))^2]$: A. Bias B. Variance **C. Noise**

2. $E[(h(x) - f_\theta(x))^2]$: **A. Bias** + **B. Variance** C. Noise $\rightarrow \text{Bias}^2 + \text{Variance}$

3. $E[(f_\theta(x) - E[f_\theta(x)])^2]$: A. Bias **B. Variance** C. Noise $(f_\theta(x))^2$

1st term

$$y = h(x) + \epsilon$$

$$y - h(x) = \epsilon$$

$$\Rightarrow E[(y - h(x))^2] =$$

$$E[\epsilon^2] = \sigma^2$$

2nd term

$\text{Bias}^2 + \text{Variance}$

3rd term

Variance

★ [3 Pts] Your manager hypothesizes that for a given novice player with x practice hours, their scrimmage performance is $g(x)$, where g is some unknown function you are trying to model. However, the *observed* performance is a random variable $Y = g(x) + \epsilon$, where ϵ is the player's random, zero-mean performance error on scrimmage day. A model's prediction $\hat{Y}(x)$ is also a random variable because the parameter estimates depend on the training set (which your manager assumes is a random sample of past novices' scrimmage performances).

Match the following expressions to the appropriate terms:

	Model Risk	(Model Bias) ²	Model Variance
(i) $\text{Var}(\hat{Y}(x))$	☐	☐	☒
(ii) $(E[\hat{Y}(x)] - g(x))^2$	☐	☒	☐
(iii) $E[(\hat{Y}(x) - E[\hat{Y}(x)])^2]$	☐	☐	☒

[2 Pts] Ishani has a model with a model risk of 32, a model variance of 8, and an observational variance of 3. Ishani adds more features to this model (without changing the number of data points) and sees the model risk decrease from 32 down to 17.

Which of the following are reasonable values for this newer model? **Select all that apply.**

- ☐ A. Model Variance: 5, Observational Variance: 2
- ☒ B. Model Variance: 10, Observational Variance: 3
- ☐ C. Model Variance: 15, Observational Variance: 2
- ☐ D. Model Variance: 15, Observational Variance: 3

$$\text{MSE} = 3$$

$\text{var}(\hat{f})$

★ Observational Variance

$$= \text{Var}(Y) = \text{Var}(f(x) + \epsilon) = \text{Var}(\epsilon) = \sigma^2$$

$\Rightarrow$ Observational Variance
 $\Rightarrow$ unavoidable errors

Solution: As we add more features to a model, we expect bias to decrease and variance to increase. As a result, Option A is incorrect.

Additionally, we are not changing anything about the underlying dataset, so the observational variance should remain the same, eliminating Option C (and Option A again) as

Matthew decides to try a new method. He first comes up with a model $h(x)$, which has a bias of B and a variance of V on the original dataset. Next, he builds a new model $f(x)$ using this process:

1. Obtain n independent, random samples from the original dataset.
2. For each dataset, train a new version of $h_i(x)$ for each i from 1 to n , where $h_1(x)$ was the $h(x)$ model trained on the first dataset and $h_n(x)$ was the $h(x)$ model trained on the n^{th} dataset.
3. Finally, define $f(x)$ as $f(x) = \frac{1}{n} \sum_{i=1}^n h_i(x)$.

[3 Pts] What is the model's **bias squared** of the $f(x)$ model in terms of B and n ? Grading will be done based on the work you show in the box below.

- ☒ A. B^2
☐ B. $\frac{1}{n} B^2$
☐ C. $(\frac{B}{n})^2$
☐ D. $n B^2$

Important intuition (why this matters)

- Averaging reduces variance, because independent fluctuations cancel.
- Averaging does NOT reduce bias, because bias is a systematic error.
- This is exactly why bagging helps high-variance models but not biased ones.

For completeness (even though not asked):

$$\text{Var}(f) = \frac{V}{n}$$

[3 Pts] Adrienne also gives Buck some **population parameters** for the "Score" column in reviews, displayed on the right. Let Y_i denote the "Score" value of entry i in Buck's sample, for $i \in \{1, 2, 3, \dots, 100\}$.

"Score"

population mean	7.1
population SD	1.2

Assume the underlying data generating process for the "Score" column is a constant model: $Y = \theta + \epsilon$, where $\theta = 7.1$ is a fixed population parameter and ϵ is a random noise with expectation 0 and variance 1.44.

Buck uses the following model to fit his sample of Y_i 's: $\hat{Y} = \hat{\theta}$.

Buck estimates θ using the mean of the first 10 "Score" values: $\hat{\theta} = \frac{1}{10} \sum_{i=1}^{10} Y_i$.

- (a) What is his **model bias**? Simplify your answer as much as possible.
 (b) What is his **model variance**? Simplify your answer as much as possible.
 (c) What is his **model risk**? Simplify your answer as much as possible.

a) Model Bias = $E[\hat{Y}] - 7.1$

$$= \frac{1}{10} \cdot E\left[\sum_{i=1}^{10} Y_i\right] - 7.1$$

$$= \frac{1}{10} \cdot \sum_{i=1}^{10} (E[Y_i])$$

} Since Y_i 's are independent for diff i

$$= \frac{1}{10} \cdot 10 \cdot (7.1) - 7.1 = 0$$

$Y_i = 7.1 + \epsilon_i$ | $\epsilon = N(0, 1.44)$
 $\sigma^2 = 1.44$
 $\sigma = 1.2$

Model $\hat{Y} = \frac{1}{10} \sum_{i=1}^{10} Y_i$

True function $Y = 7.1$

★ b) Model Variance = $E[(E[\hat{Y}] - \hat{Y})^2]$

$$= E[(7.1 - \hat{Y})^2] \rightarrow \text{wrong approach}$$

 Model Variance $\text{Var}(\hat{Y}) = \text{Var}\left(\frac{1}{10} \sum_{i=1}^{10} Y_i\right) \rightarrow \text{correct approach} \star$

$$= \frac{1}{100} \sum_{i=1}^{10} \text{Var}(Y_i) \rightarrow \text{Since } Y_i \text{ are iid}$$

$$= \frac{1}{100} \cdot 10 \cdot (1.44) = \frac{1.44}{10} = 0.144$$

c) Model Risk = $\text{Bias}^2 + \text{Variance} + \sigma^2$

$$= 0 + 0.144 + 1.44$$

 Model Risk = 1.584

We will analyze the model bias and variance of a simple linear regression model with no intercept term, $f_{\theta}(x) = \theta x$. Note that x and θ are both 1-dimensional.

- (a) [4 Pts] Suppose we estimate the values on the right using our model $f_{\hat{\theta}}(x)$ on the test data point $(x, Y) = (2, 3)$.

Calculate the model risk, assuming that there is no observational variance (i.e. $\epsilon = 0$). Show your work. (Hint: Calculate the individual quantities that make up the model risk.)

Model Risk = $\text{Bias}^2 + \text{Variance} + \sigma^2$

$$= 4 + 5 + 0 = 9$$

Quantity	Value
$E[f_{\hat{\theta}}(x)]$	1
$E[(f_{\hat{\theta}}(x))^2]$	6
σ^2	0

Model Bias²

$$(E[f_{\hat{\theta}}(x)] - f(x))^2$$

$$= (1 - 3)^2 = 4$$

Test data point
 $x = 2$
 $y = 3$
 $y = f(x) = \frac{3}{2} \cdot x$

Model Variance

$$E[(E[f_{\hat{\theta}}(x)] - f_{\hat{\theta}}(x))^2] = E[(1 - f_{\hat{\theta}}(x))^2]$$

$$= E[1 - 2f_{\hat{\theta}}(x) + f_{\hat{\theta}}^2(x)] = 1 - 2 + 6 = 5$$

 ★ c) $\text{Var}(f_{\hat{\theta}}(x)) = E[f_{\hat{\theta}}^2(x)] - E[f_{\hat{\theta}}(x)]^2$

$$= 6 - 1^2 = 5$$

(b) [4 Pts] Using a simple linear regression model without an intercept (i.e. $f_{\theta}(x) = \theta x$), we estimate the model variance on a *new* datapoint $(x, Y) = (3, 12)$ as $\text{Var}(f_{\hat{\theta}}(x)) = 3$ using some arbitrary model parameter $\hat{\theta}$.

★ Suppose we create a new model where $\theta_{\text{new}} = 2\hat{\theta}$. Calculate the new model variance, $\text{Var}(f_{\theta_{\text{new}}}(x))$ using θ_{new} . Show your work.

Model Variance $\text{Var}(f_{\theta}(x)) = 3$
 new model $f_{\theta_{\text{new}}} = \theta_{\text{new}} x = 2 \cdot \theta x$
 $\text{Var}(f_{\theta_{\text{new}}}(x)) = \text{Var}(2\theta x) = 4 \text{Var}(\theta x) = 4 \cdot 3 = 12$

(c) [2 Pts] Based on the models described in part (b), which of the following **may** be true? Note that the bias of the new model is $g(x) - \mathbb{E}[f_{\theta_{\text{new}}}(x)]$, and the bias of the original model is $g(x) - \mathbb{E}[f_{\hat{\theta}}(x)]$.

- ☒ A. The new model bias is smaller than the original model bias.
☐ B. The new model bias is larger than the original model bias.
☐ C. The new model bias is the same as the original model bias.
☐ D. The new model bias becomes undefined, but the original model bias is a real number.

New Bias = $g(x) - 2 \cdot \mathbb{E}[f_{\theta}(x)]$

old Bias = $g(x) - \mathbb{E}[f_{\theta}(x)]$

point $(x, y) = (3, 12)$

New Bias = $12 - 2 \cdot \mathbb{E}[f_{\theta}(x)]$

old Bias = $12 - \mathbb{E}[f_{\theta}(x)]$

[2 points] Which of the following is TRUE about the bias-variance decomposition of test error?

- ☒ (a) Expected test error is equal to $\text{bias} + \text{var}^2 + \text{noise}$
☒ (b) Noise can be reduced by using regularization
☒ (c) Low variance corresponds to high complexity
☒ (d) None of the above

[2 points] If we assume that the data is being generated from a source with no noise $y = w^T x$, choosing a classifier $f^*(x) = w^T x$ using OLS will make which of the following quantities zero?

- (a) Only Bias
 (b) Only Variance
☒ (c) Both Bias and Variance
 (d) None of the above

w

TRUE

Bias = $\mathbb{E}[f^*(x)] - f(x) = 0$

Variance = $\mathbb{E}[(\mathbb{E}[f^*(x)] - f^*(x))^2]$

= $\mathbb{E}[(f(x) - f(x))^2] = 0$

if no error in source
 $\Rightarrow f^*(x)$ will always be $= f(x)$

(2 pt) In the bias-variance decomposition, our model's risk was defined as $\mathbb{E}[(Y - f_{\hat{\theta}}(x))^2]$.

Which of the following is equal to $\mathbb{E}[Y - f_{\hat{\theta}}(x)]$?

- ☐ 0
☒ The negative of model bias
☐ Model bias
☐ Model bias squared
☐ Model variance



= $\mathbb{E}(Y) - \mathbb{E}[f_{\hat{\theta}}(x)]$
 = $y(x) - \mathbb{E}[f_{\hat{\theta}}(x)] = \text{Bias}$

Linearity

$\mathbb{E}(Y) = \mathbb{E}(y(x) + \epsilon)$
 = $\mathbb{E}[y(x)] + \mathbb{E}[\epsilon]$
 $\parallel \parallel$
 $y(x) \quad 0$

[2 points] Consider the error decomposition for a least squares regression model

$$\mathbb{E}_{x,y,D}[(h(x; D) - y)^2] = \underbrace{\mathbb{E}_{x,D}[(h(x; D) - \bar{h}(x))^2]}_{\text{Variance}} + \underbrace{\mathbb{E}_x[(\bar{h}(x) - \bar{y}(x))^2]}_{\text{Bias}^2} + \underbrace{\mathbb{E}_{x,y}[(\bar{y}(x) - y)^2]}_{\text{Noise}}$$

where $h(x; D)$ is a model learned over a training sample D , $\bar{h}(x) = \mathbb{E}_{D \sim P^n}[(h(x; D))]$ is the average model, and $\bar{y}(x) = \mathbb{E}_{y|x}[y]$ is the optimal Bayes model. Which of the following best describes the term labeled *variance*?

- ☒ (a) On average, how much your learned model differs from average model across different samples D
 (b) How far is the average model from optimal Bayes model
 (c) The variance between predictions for a fixed sample D
 (d) How accurate the model is in predicting y

$\bar{y}(x) \rightarrow$ true function $\rightarrow$ aka optimal Bayes model
 $y \rightarrow$ observation
 $h(x; D) \rightarrow$ hypothesis function



[3 pts] How does the bias-variance decomposition of a ridge regression estimator compare with that of ordinary least squares regression? (Select one.)

- ☐ Ridge has larger bias, larger variance
☒ Ridge has larger bias, smaller variance
☐ Ridge has smaller bias, larger variance
☐ Ridge has smaller bias, smaller variance

[3 pts] In terms of the bias-variance decomposition, a 1-nearest neighbor classifier has _____ than a 3-nearest neighbor classifier.

- ☒ higher variance
☐ lower variance
☐ higher bias
☒ lower bias

Classification on Bias, Variance decomposition

→ Involves 2 different expectations

- ① Over the randomness of the dataset / training sample → collecting new dataset ⇒ model fitted changes
- ② Over the randomness of test point x, y sampling from true data distribution

— Measure prediction error at a random test point (x, y)

— Algorithm that predicts $\hat{f}(x; D)$ after training on dataset D .

$$\text{MSE} = E_D E_{x,y} [(y - \hat{f}(x; D))^2]$$

For fixed x , $y \sim P(y|x)$, $E[y|x] = f^*(x) \rightarrow$ Bayes optimal predictor

then at given x , the error is

$$E_{D,y} [(y - \hat{f}(x; D))^2] = \underbrace{E_D [(\hat{f}(x; D) - E_D[\hat{f}(x; D)])^2]}_{\text{Variance}} + \underbrace{(E_D[\hat{f}(x; D)] - f^*(x))^2}_{\text{Bias}^2} + \underbrace{E[(y - f^*(x))^2]}_{\text{Irreducible noise}}$$

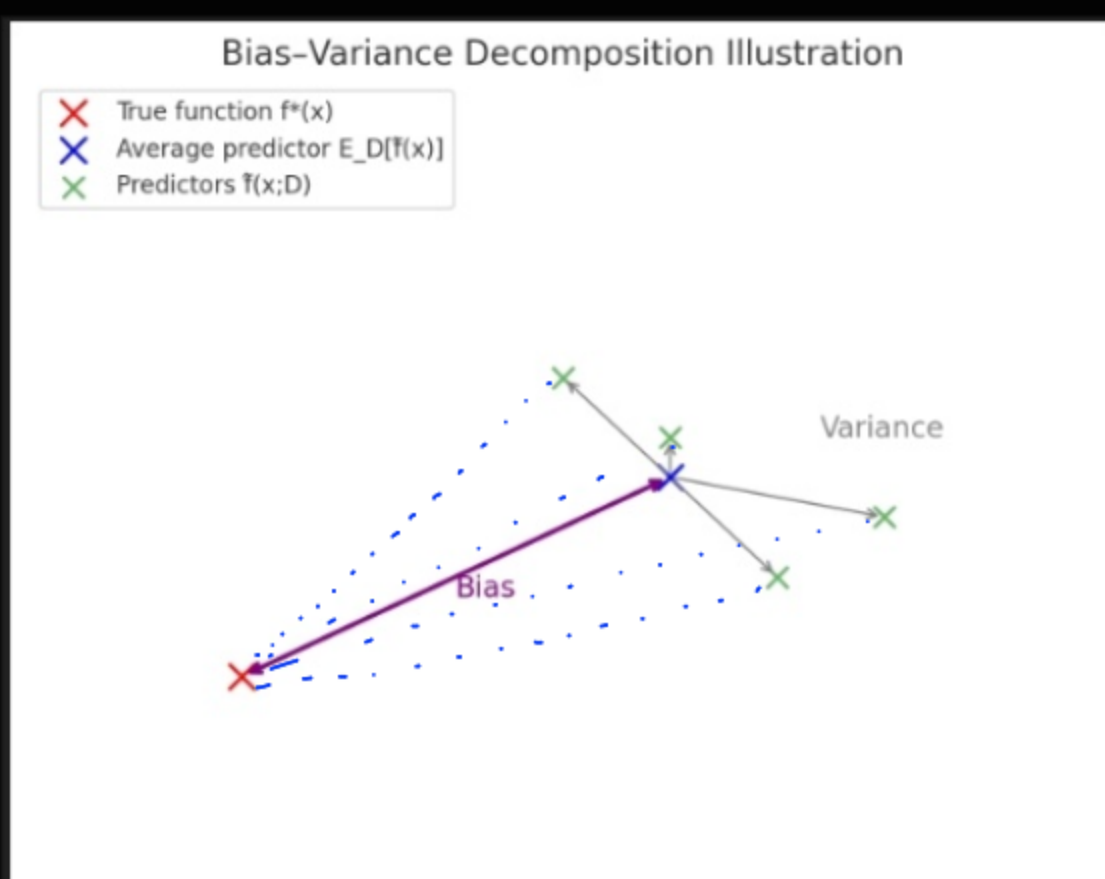
Bias² can be written in dual form

① $(E_D[\hat{f}(x; D)] - f^*(x))^2 \rightarrow$ standard form: difference between the average predictor & the true function

$$\begin{aligned} \text{② } E_D [(\hat{f}(x; D) - f^*(x))^2] &\rightarrow E_D [f^*(x)] = f^*(x) \\ E_D [(\hat{f}(x; D) - f^*(x))^2] &= (E_D[\hat{f}(x; D)] - E[f^*(x)])^2 \\ &= (E_D[\hat{f}(x; D)] - f^*(x))^2 \end{aligned}$$

③ Bias² + Variance

$$E_D [(\hat{f}(x; D) - f^*(x))^2] = \text{Bias}^2 + \text{Variance}$$



$E[(\hat{f}(x; D) - f^*(x))^2]$ = squared average of length from green to red (blue arrow) = Bias² + Variance

Objects in the picture (for a fixed x)

- $f^*(x) = E[y | x]$ (red point)
- $\hat{f}(x; D)$ for many datasets D (green points)
- $\mu(x) := E_D[\hat{f}(x; D)]$ (blue point)

Arrows:

- **Bias arrow:** $f^*(x) \leftrightarrow \mu(x)$
- **Variance arrows:** $\hat{f}(x; D) \leftrightarrow \mu(x)$

Decomposition at a fixed x

Start from the test squared error and insert and subtract $\mu(x)$:

$$\begin{aligned} E_{D,y} [(y - \hat{f})^2 | x] &= E_{D,y} [(y - f^* + f^* - \mu + \mu - \hat{f})^2 | x] \\ &= \underbrace{E[(y - f^*)^2 | x]}_{\text{noise at } x} + \underbrace{(\mu(x) - f^*(x))^2}_{\text{bias}^2 \text{ at } x} + \underbrace{E_D[(\hat{f}(x; D) - \mu(x))^2]}_{\text{variance at } x}. \end{aligned}$$

Why cross-terms vanish:

- $E[y - f^* | x] = 0$ by definition of f^* .
- $E_D[\hat{f} - \mu | x] = 0$ by definition of μ .
- The remaining two are independent of y vs. D , so mixed expectations are zero.

Interpretation vs. the picture:

- $(\mu - f^*)^2$ is the squared length of the purple "bias" arrow (red ↔ blue).
- $E_D[(\hat{f} - \mu)^2]$ is the average squared length of grey variance arrows (green ↔ blue).
- $E[(y - f^*)^2 | x] = \sigma^2(x)$ is irreducible noise (not drawn).

$$E[(\hat{f}(x_{iD}) - f^*(x))^2] = E[\hat{f}(x_{iD})^2 + f^*(x)^2 - 2\hat{f}(x_{iD}) \cdot f^*(x)]$$

$$= E[\hat{f}(x_{iD})^2] + f^*(x)^2 - 2f^*(x) \cdot E[\hat{f}(x_{iD})] \neq (E[\hat{f}(x_{iD})] - f^*(x))^2$$

Since

$$E[\hat{f}(x_{iD})^2] \neq E[\hat{f}(x_{iD})]^2$$

$$E[(\hat{f}(x_{iD}) - f^*(x))^2] = E\left[\underbrace{(\hat{f}(x_{iD}) - E[\hat{f}(x_{iD})])}_a + \underbrace{E[\hat{f}(x_{iD})] - f^*(x)}_b\right]^2$$

$(a+b)^2 = a^2 + b^2 + 2ab$

Adding the
Subtraction

$$= E\left[(\hat{f}(x_{iD}) - E[\hat{f}(x_{iD})])^2 + (E[\hat{f}(x_{iD})] - f^*(x))^2 + \underbrace{2ab}_{\text{term}}\right]$$

$$= \underbrace{E[(\hat{f}(x_{iD}) - E[\hat{f}(x_{iD})])^2]}_{\text{Variance}} + \underbrace{(E[\hat{f}(x_{iD})] - f^*(x))^2}_{\text{Bias}^2} + \text{cross term becomes zero.}$$

$$\Rightarrow \text{Var}(\hat{f}(x_{iD})) = E[\hat{f}(x_{iD})^2] - E[\hat{f}(x_{iD})]^2$$

$$= E[(\hat{f}(x_{iD}) - E[\hat{f}(x_{iD})])^2]$$

Useful approach in some Questions

Perceptron

Perceptron is a binary classifier
 Decision rule $y = \begin{cases} +1, & w^T x \geq b \\ -1, & w^T x < b \end{cases}$

$y = \begin{cases} +1, & w^T x + b \geq 0 \\ -1, & w^T x + b < 0 \end{cases}$ Similar to logistic regression without the Sigmoid activation

Correctly classified x_n if $y_n w^T x_n \geq 0$
 Bias / offset / threshold b is embedded into W by appending row of 1 to design matrix X .

Update Rule

Perceptron training algorithm is called perceptron update rule

$$W_{n+1} = W_n + y_n x_n \quad \text{if } y_n w^T x_n < 0$$

i.e. update only if incorrectly classified.

- Perceptron always converges if data is linearly separable.
- However the solution is not unique.
- The solution depends on the order in which data is used for update rule
- Multiple iterations over all datapoints may be required for the perceptron learning to converge.

Run the perceptron algorithm (with offset) on the following dataset where points are given in the form $(x^{(i)}, y^{(i)})$, showing the first iteration through the whole data set. Initialize θ to $(0,0)$ and θ_0 to 0. Iterate through the points in the order given. Show each step of the algorithm.

$((-1, 2), -1), ((0, 3), -1), ((2, 4), 1), ((4, 4), 1)$

datapoint x	datapoint y	mistake?	new θ	new θ_0
$(-1, 2)$	-1	Yes	$[1, -2]$	-1
$(0, 3)$	-1	No	$[1, -2]$	-1
$(2, 4)$	+1	Yes	$[3, 2]$	0
$(4, 4)$	+1	No	$[3, 2]$	0

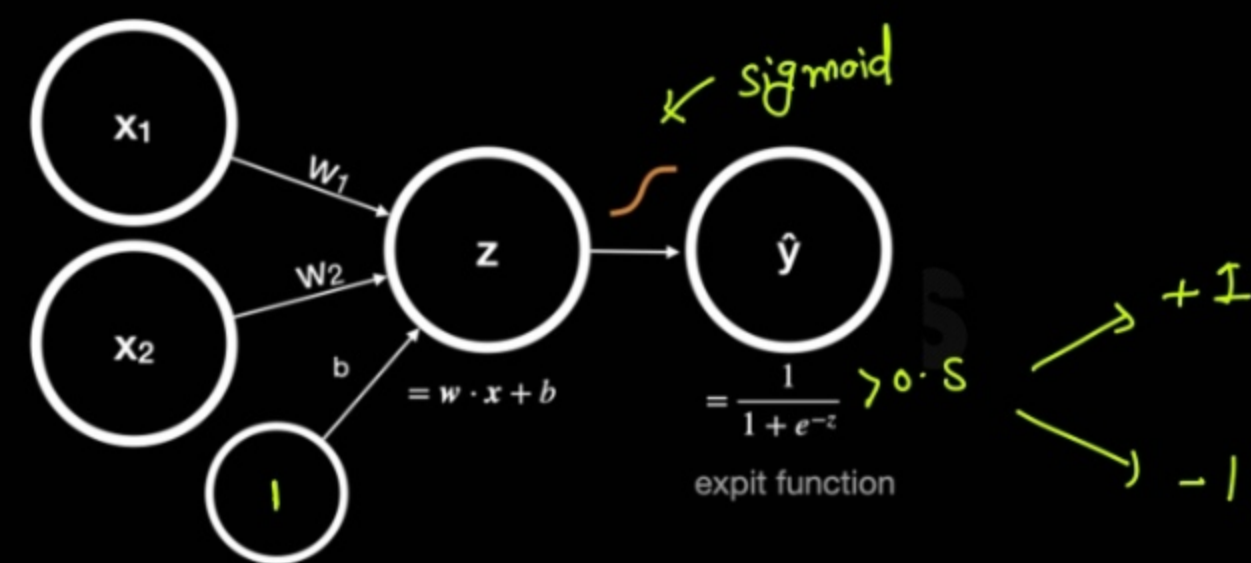
We are given a dataset $\mathcal{D}_n$ of points in 2D: $x \in \mathbb{R}^2$ and their corresponding labels $y \in \{+1, -1\}$. Consider the following two scenarios:

- We run the perceptron algorithm (with offset) on x , with θ initialized to $(0,0)$ and θ_0 initialized to 0.
- We construct a 3D dataset x' from x by concatenating a 1 to each x : $x' = [x \ 1]^T$. We run the perceptron algorithm (without offset) on x' with the parameters for this model, θ' , initialized to $(0,0,0)$.

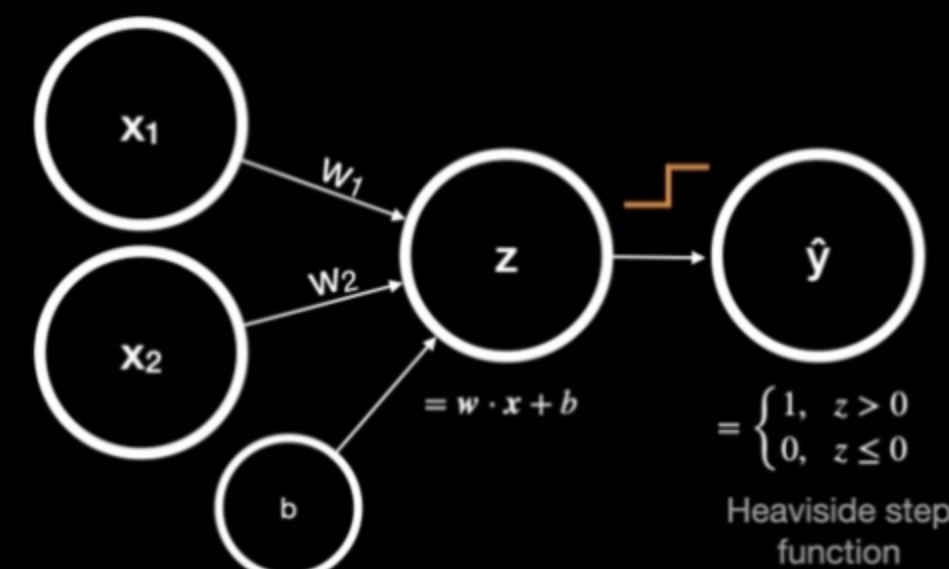
Assuming we iterate through the points in the same order, when we actually run these algorithms, will they make the same number of mistakes?

☒ Yes ☐ No

Logistic Regression



Perceptron



Let w_i represent w after the i -th iteration of learning.

We are given that $w_0 = (0, 0, 0)$.

$$\begin{aligned} w_1 &= w_0 + y_1 x_1 \\ &= (0, 0, 0) + (+1) \cdot (0, 0, 1) \\ &= (0, 0, 1) \end{aligned}$$

$$(y_1 w_0^T x_1 = 0)$$

$$\begin{aligned} w_2 &= w_1 + y_2 x_2 \\ &= (0, 0, 1) + (-1) \cdot (0, 1, 1) \\ &= (0, -1, 0) \end{aligned}$$

$$(y_2 w_1^T x_2 = -1)$$

$$\begin{aligned} w_3 &= w_2 + y_3 x_3 \\ &= (0, -1, 0) + (-1) \cdot (1, 0, 1) \\ &= (-1, -1, -1) \end{aligned}$$

$$(y_3 w_2^T x_3 = 0)$$

$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2$$

$$\theta = [w_1, w_2]$$

$$\theta_0 = w_0$$

DB

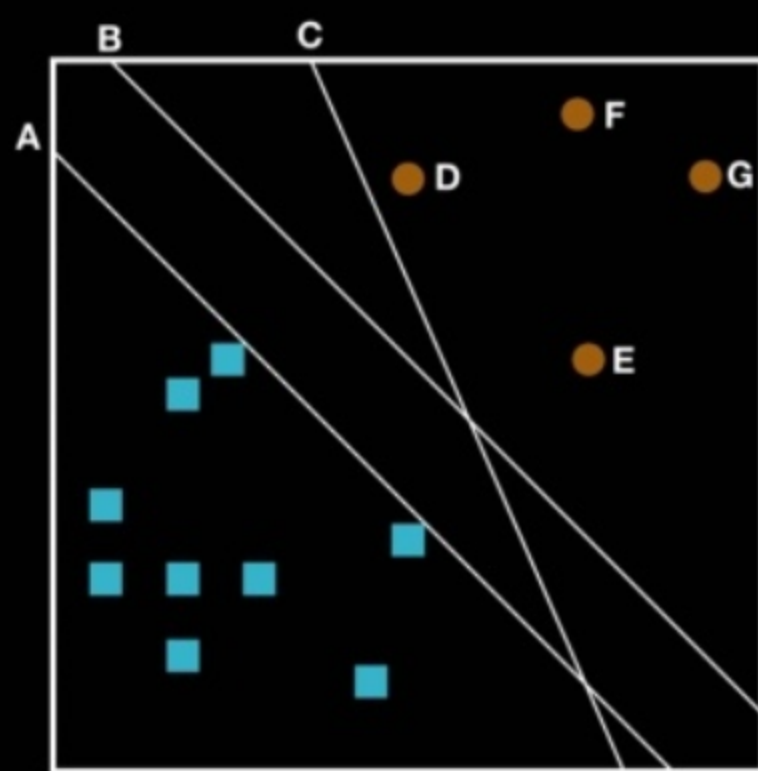
$$\begin{aligned} w^T x + b \geq 0 &\Rightarrow +1 \\ \text{else} &\Rightarrow -1 \end{aligned}$$

update rule

$$\begin{aligned} \theta_{\text{new}} &= \theta_{\text{old}} + y_n x_n \\ \theta_{0-\text{new}} &= \theta_{0-\text{old}} + y_n \end{aligned} \quad \text{if } y_n (w^T x_n + b) < 0$$

Constructing a 1 dimensional higher dataset x' by concatenating 1 to each original data point is equivalent for running the algorithm with separate bias term.

Consider the following binary classification dataset, where circles denote the positive class and squares the negative class:



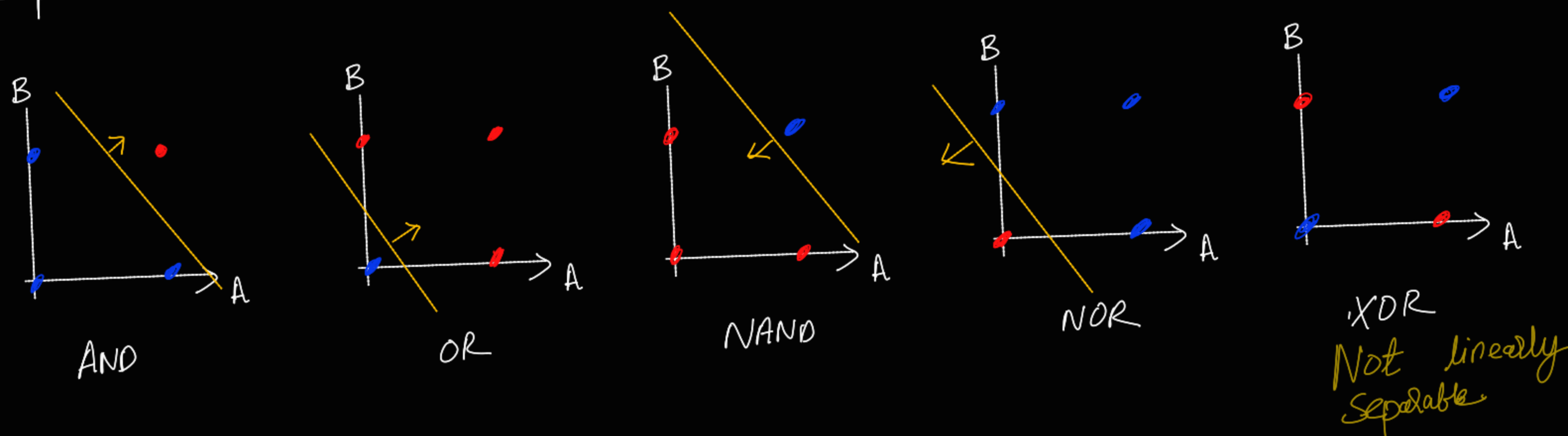
perceptron can have many solutions depending on the order in which we choose to train the data

SVM → Maximum margin
Perceptron → Any separating boundary

(a) Which (if any) of the decision boundaries could be returned by perceptron (with a bias parameter) after training until convergence on this dataset? Mark all that apply. A, B, C

(b) Which (if any) of the decision boundaries could be returned after optimizing the hard SVM objective (with a bias parameter) on this dataset? Mark all that apply. B

★ AND gate, OR gate and other linearly separable gates can be modelled using perceptron.
★ However perceptron cannot model the XOR gate since this data is not linearly separable



Which of the following is/are true about the Perceptron classifier?

- ✓ (a) It can learn a OR function
- ✗ (b) It can learn a XOR function
- ✓ (c) The obtained separating hyperplane depends on the order in which the points are presented in the training process.
- ✗ (d) For a linearly separable problem, there exists some initialization of the weights which might lead to non-convergent cases.

Reason for update rule

→ Suppose we misclassify x_n . i.e. $y_n w^T x_n < 0$: then we update

→ $w_{new}^T = w + y_n x_n$

→ check again the condition

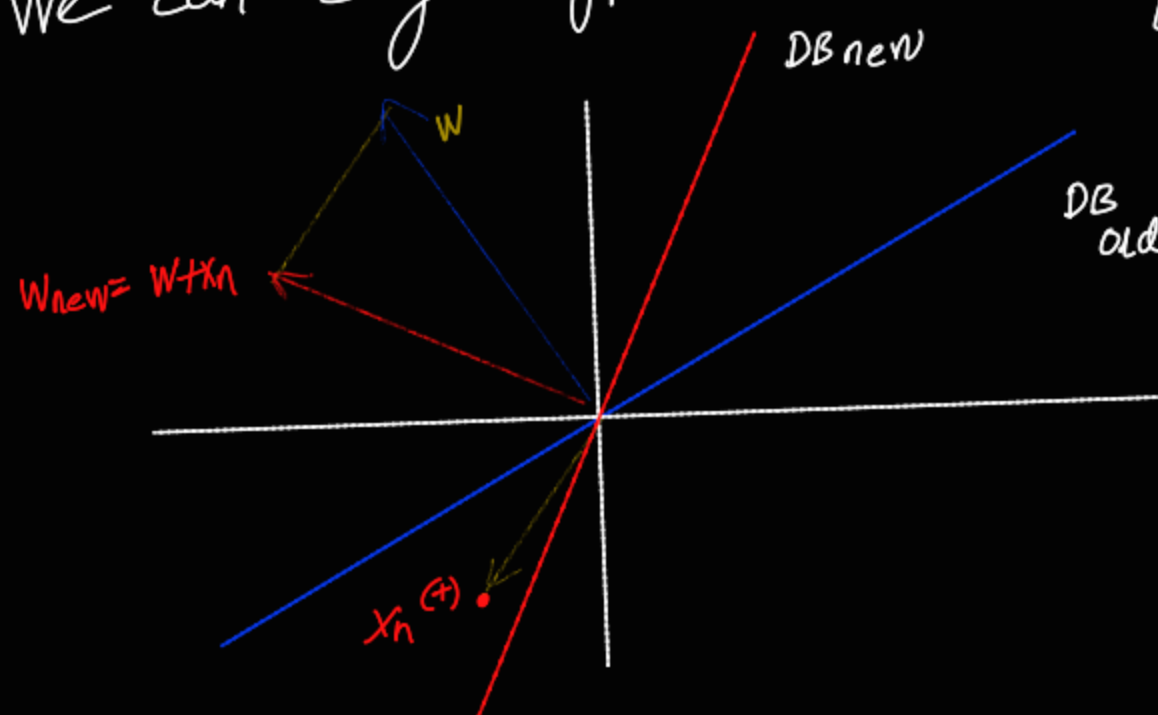
$$y_n w_{new}^T x_n = y_n (w + y_n x_n)^T x_n$$

$$= \underbrace{y_n w^T x_n}_{\text{original old quantity}} + \underbrace{y_n^2 x_n^T x_n}_{\text{Always +ve}}$$

original old quantity

Always +ve
 $y_n^2 \rightarrow +ve$
 $x_n^T x_n \rightarrow \|x_n\|^2 \rightarrow +ve$

→ New $\approx$ old + smt (+ve)
→ does not guarantee correctly classified with one iteration
→ But some progress towards correctly classifying
→ Margin from hyperplane to the misclassified point decreases in this step
→ We can say hyperplane moved towards this point (+ve) away from point (-ve)



x_n - misclassified

Consider a binary classification problem. Suppose I have trained a model on a linearly separable training set, and now I get a new labeled data point which is correctly classified by the model, and far away from the decision boundary. If I now add this new point to my earlier training set and re-train, in which cases is the learnt decision boundary likely to change?

- (a) When my model is a perceptron.
- (b) When my model is logistic regression.
- (c) When my model is an SVM.

if initialising with learned decision boundary won't update since already correctly classified

Not a SV

Logistic loss has changed

Point far away from DB and correctly classified, so negligible loss

But still loss not exactly 0

Decision boundary will slightly shift to the new point.

Figure 4 shows a training set in $\mathbb{R}^2$. Suppose that we use perceptron algorithm for classification. We record the total number of times each point occurs in the update step. Remember that if a point is misclassified, then it occurs in the update step.

x_1	x_2	y	times misclassified
-3	2	+1	0
-1	1	+1	0
-1	-1	-1	2
2	2	-1	1
1	-1	-1	0

Figure 4: Training Data

(3 points) Assume that the initial weight is $w^{(0)} = [-3, 2, 1]$ where 1 is the offset term. So the feature $x_3 = 1$ is constant.

What is the equation of the separating line expressed in terms of x_1 and x_2 determined by the algorithm?

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}_{t+1} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}_t + y_n \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}_n$$

$$w_{0 \text{ final}} = w_{0 \text{ init}} + \# \text{positive mistakes} - \# \text{negative mistakes} = 1 + 0 - 3 = -2 //$$

$$w_{1 \text{ final}} = w_{1 \text{ init}} + \sum_{n=1}^N x_1^n \left[\# \text{Ave mistake} - \# \text{(-ve) Mistakes} \right] = -3 + 2 - 2 = -3 //$$

$$w_{2 \text{ final}} = w_{2 \text{ init}} + \sum_{n=1}^N x_2^n \left[\# \text{Ave mistake} - \# \text{(-ve) Mistakes} \right] = 2 + 2 - 2 = 2 //$$

$$w_{\text{final}} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -2 \\ -3 \\ 2 \end{bmatrix}$$

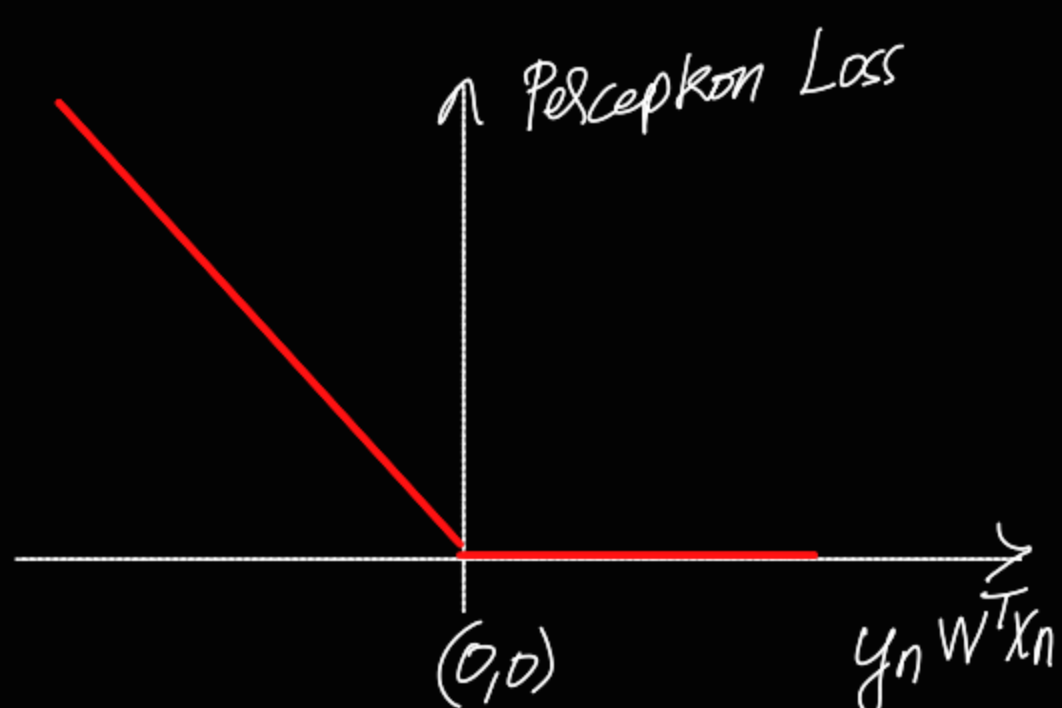
$$w_{\text{final}} = \begin{bmatrix} w_1 & w_2 & w_0 \end{bmatrix} = \begin{bmatrix} -3 & 2 & -2 \end{bmatrix}$$

Perceptron Cost function

$$L(w) = \frac{1}{n} \sum_{i=1}^n \max(0, -y_n w^T x_n)$$

No loss if $-y_n w^T x_n < 0$
 $\Rightarrow$ No loss for $y_n w^T x_n \geq 0$
 $\Rightarrow$ No loss for correctly classified
 $\Rightarrow$ Loss for misclassified

In logistic Loss, there is some Loss even for correct classification



Gradient descent with learning rate

$$w_{t+1} = w_t - \alpha \cdot \nabla L$$

$$w_{t+1} = w_t + \alpha y_n x_n$$

$\nabla L = -y_n x_n$

Find the range of decision boundary given by Perceptron.



[3 pts] The perceptron algorithm will converge:

- ☒ If the data is linearly separable
- ☐ As long as you initialize θ to all 0's
- ☐ Even if the data is linearly inseparable
- ☐ Always

We ran the perceptron algorithm, with offset, on the following dataset and recorded the number of mistakes we made for each point. Without running the algorithm, what is the final value of θ_0 ?

i	1	2	3	4	5
$x^{(i)}$	(-4, 3)	(-2, 2)	(-2, 0)	(1, 3)	(2, 0)
$y^{(i)}$	1	-1	-1	-1	-1
# mistakes	2	3	2	1	0

$$\theta_0: 2 - 3 - 2 - 1 \Rightarrow \theta_0 = -4$$

$$\theta_{n+1} = \theta_n + y_n$$

$$\theta_{\text{final}} = \# \text{mistakes} \cdot y_n$$

$$\star \theta_{\text{final}} = \# \text{positive mistakes} - \# \text{negative mistakes}$$

$y_n \neq +1$ misclassified +ve
 -1 misclassified -ve

Gradient update rule
 $w_{t+1} = w_t - \alpha \nabla L(w)$
 from the update rule if we infer loss function then this is the result.

Perceptron Convergence proof

We want to show that the number of mistakes made by the perceptron algorithm is bounded.

Let t be the number of iterations. We aim to prove:

t is the number of iterations required so that the perceptron no longer makes mistakes with w_{t+1} . $t \leq \frac{R^2}{\gamma^2}$

where R and γ are defined parameters.

Here each iteration = 1 mistake, so $t = \# \text{ mistakes}$ also

To prove this, we will establish lower and upper bounds on $\|w_{t+1}\|^2$.

1. Lower bound: $\|w_{t+1}\|^2 \geq t^2 \gamma^2$
2. Upper bound: $\|w_{t+1}\|^2 \leq t R^2$

$$\Rightarrow t^2 \gamma^2 \leq \|w_{t+1}\|^2 \leq t R^2$$

$$\Rightarrow t \leq \frac{R^2}{\gamma^2} \rightarrow t \text{ is bound} \Rightarrow \text{eventually no mistakes.}$$

Proof of the Upper Bound

Define $R = \max_{i=1, \dots, n} \|x_i\|$.

$$\begin{aligned} \|w^{(t+1)}\|^2 &= \|w^{(t)} + y^{(t)} x^{(t)}\|^2 \\ &= \|w^{(t)}\|^2 + \|x^{(t)}\|^2 + 2y^{(t)}(w^{(t)})^T x^{(t)} \\ &\leq \|w^{(t)}\|^2 + R^2 \end{aligned}$$

misclassified so always < 0

$$\|w^t\|^2 \leq \|w^{t-1}\|^2 + R^2$$

$$\|w^0\|^2 = 0$$

$$\therefore \text{By induction } \|w^{t+1}\|^2 \leq t R^2$$

Proof of the Lower Bound

Let w^* be the optimal solution, assuming the data is linearly separable.

Thus, w^* exists.

Define

$$\gamma = \min_{i=1, \dots, n} y_i x_i^T w^*$$

We want to show:

$$\|w_{t+1}\|^2 \geq t^2 \gamma^2$$

$y_i x_i^T w^*$ cannot be negative since it's the optimum classifier

$$y_i x_i^T w^* \geq 0$$

so $\min_{i=1, \dots, n} y_i x_i^T w^* \rightarrow$ algebraic Margin of this optimum classifier

$$y_1 x_1^T w^* = 2$$

$$y_2 x_2^T w^* = 0.1$$

$$y_3 x_3^T w^* = 5$$

$$\begin{aligned} (w^*)^T w^{t+1} &= (w^*)^T (w^t + y^t x^t) \\ &= (w^*)^T w^t + y^t (w^*)^T x^t \\ &= (w^*)^T (w^{t-1} + y^{t-1} x^{t-1}) + y^t (w^*)^T x^t \\ &= (w^*)^T w^{t-1} + y^{t-1} (w^*)^T x^{t-1} + y^t (w^*)^T x^t \\ &= (w^*)^T w^{t-1} + y^{t-1} (w^*)^T x^{t-1} + y^t (w^*)^T x^t \end{aligned}$$

$$(w^*)^T w^{t+1} = \underbrace{(w^*)^T w^0}_{=0} + \underbrace{y^0 w^* x^0 + y^1 w^* x^1 + y^2 w^* x^2 + \dots}_{\text{Taking min} = t\gamma}$$

$$\Rightarrow (w^*)^T w^{t+1} \geq t\gamma$$

$$\|w^*\| \|w^{t+1}\| \cos \theta \geq t\gamma$$

$$\|w^{t+1}\| \cos \theta \geq t\gamma$$

$$\|w^{t+1}\| \geq t\gamma$$

$$\|w^{t+1}\|^2 \geq t^2 \gamma^2$$

We don't invalidate the result by scaling w^* so that $\|w^*\| = 1$ since γ is algebraic margin will adjust automatically to geometric margin

$\rightarrow$ We can drop $\cos \theta$ because $\cos \theta \in [-1, 1]$ But $w^{t+1} \neq w^*$ are optimum $\Rightarrow \cos \theta \in [0, 1]$

$$\Rightarrow t \leq \frac{R^2}{\gamma^2} \rightarrow \gamma \rightarrow \text{geometric margin of optimum classifier} = \frac{w^T x_n + b}{\|w\|} \rightarrow \text{to the closest point}$$

$R \rightarrow$ distance from origin to the farthest point

$$= \frac{c}{\|w\|} \quad c \rightarrow \text{algebraic margin}$$

$$\text{so say } \gamma = 2$$

$$R = 10$$

$$\Rightarrow \text{Max iteration} = \frac{100}{4} = 25$$

$\Rightarrow$ Perceptron will not make mistake after 25 iteration/mistakes.

The Perceptron Convergence Theorem states that the upper bound on the number of mistakes is:

$$\text{or iteration } t \leq \left(\frac{R}{\gamma}\right)^2$$

where R is the maximum distance from the origin to a point in the data set, and γ is the minimum distance from a valid separator to a point in the data set.

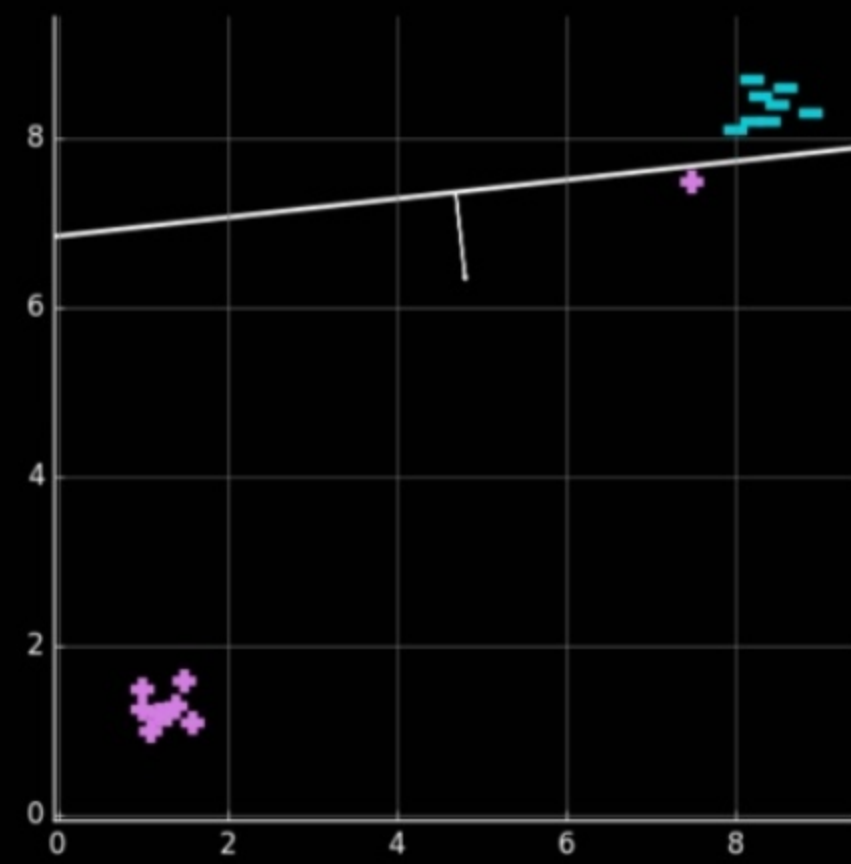
$\rightarrow$ min can be very close to zero for certain valid separators, but we are not getting restricted upper bound

$\Rightarrow$ So we actually want γ to be the maximum geometric margin possible like hard SVM.

Example:

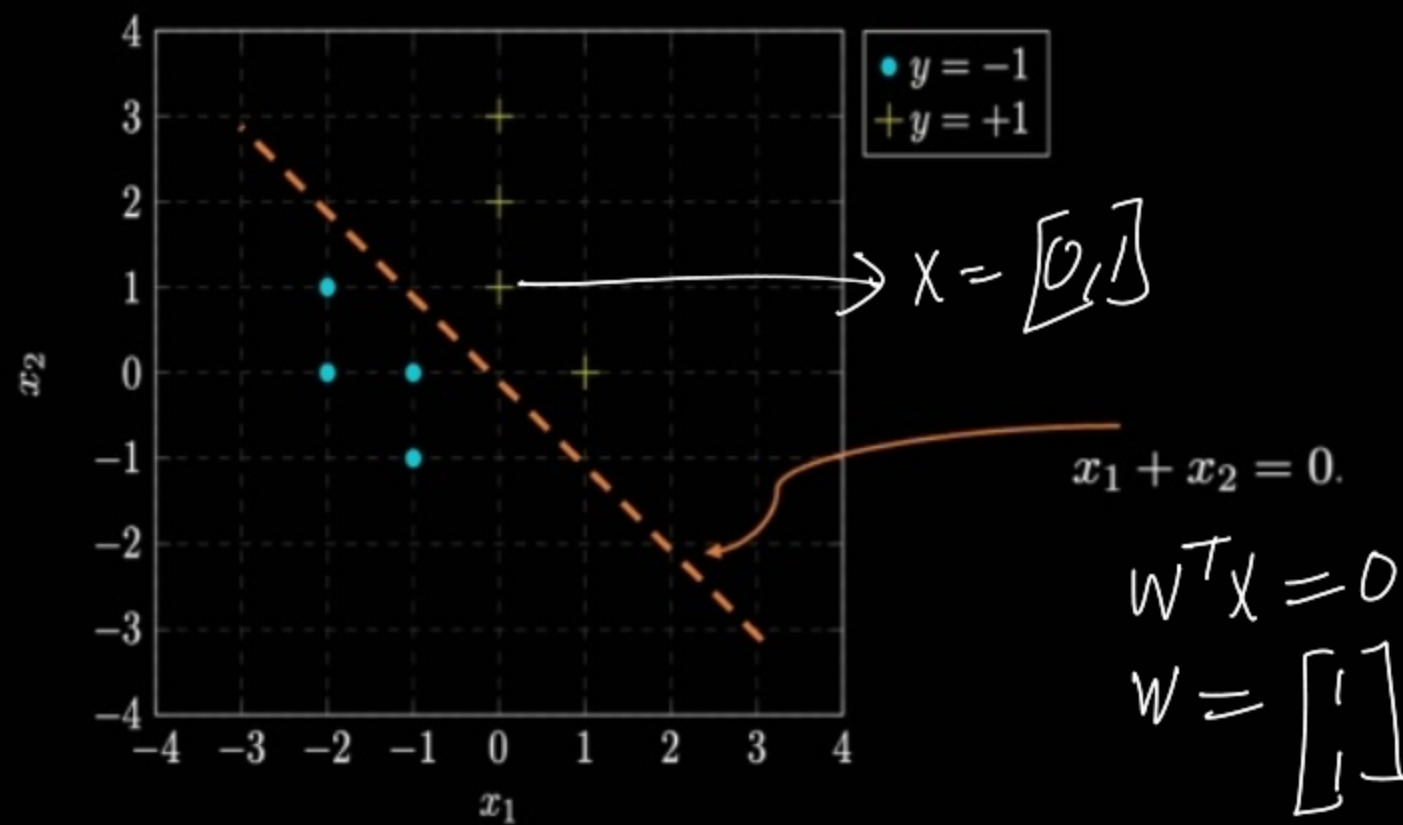
Find out the upper bound on number of mistakes.

$$R \approx 10, \gamma \approx 0.2.$$



$$t \leq \frac{100}{0.04} = \frac{100 \times 100}{4} = 2500$$

(17 points) Alice plans to apply the Perceptron algorithm to solve a classification problem on the data set shown in the figure below. For this problem, we will only consider linear separators that pass through the origin.



$$\gamma = \frac{1}{\sqrt{2}} \quad t \leq \frac{R^2}{\gamma^2} = \frac{9}{\left(\frac{1}{\sqrt{2}}\right)^2} = 18$$

$$R = 3$$

$$\gamma = \min \left(\frac{w^T x}{\|w\|} \right) = \frac{1}{\sqrt{2}}$$

What is the theoretical upper bound on the number of steps for the Perceptron algorithm

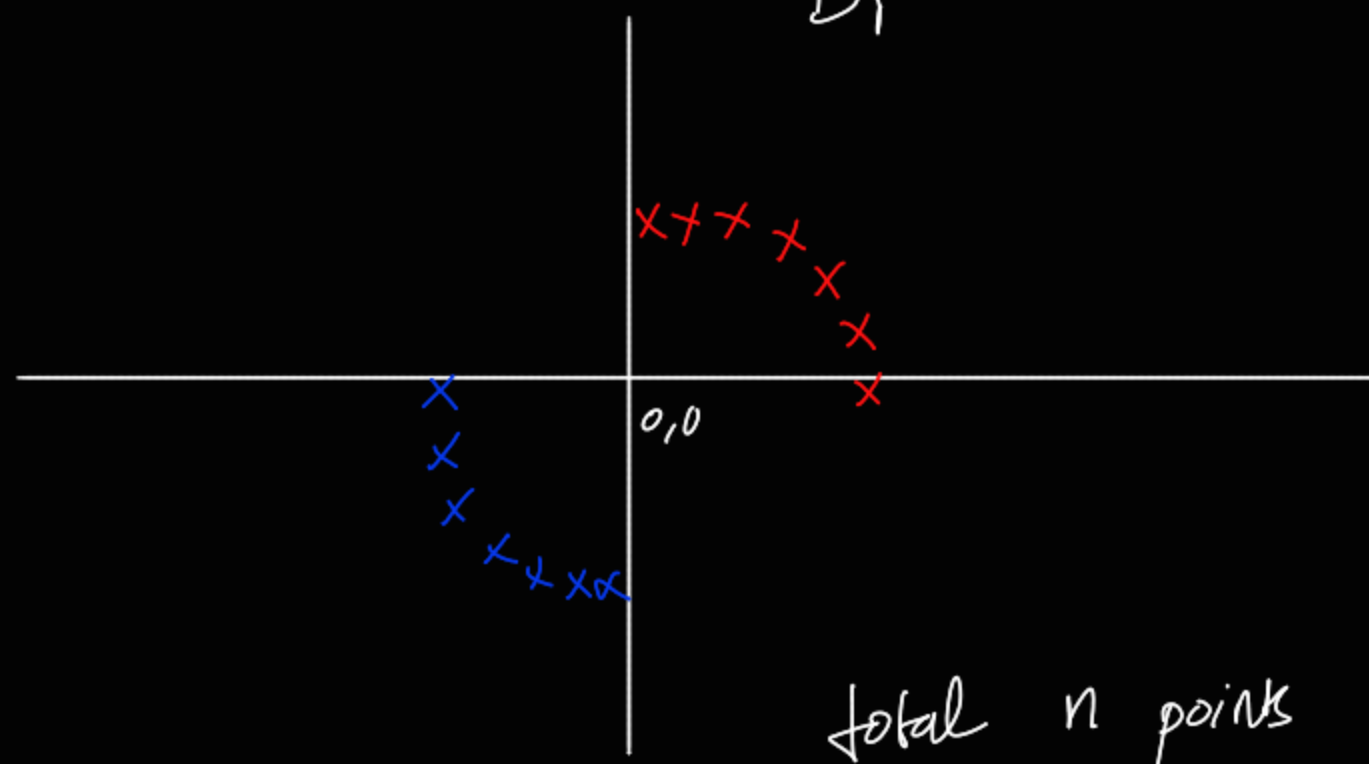
T/F

[2 pts.] Consider two datasets $D^{(1)}$ and $D^{(2)}$ where $D^{(1)} = \{(x_1^{(1)}, y_1^{(1)}), \dots, (x_n^{(1)}, y_n^{(1)})\}$ and $D^{(2)} = \{(x_1^{(2)}, y_1^{(2)}), \dots, (x_m^{(2)}, y_m^{(2)})\}$ such that $x_i^{(1)} \in \mathbb{R}^{d_1}$, $x_i^{(2)} \in \mathbb{R}^{d_2}$. Suppose $d_1 > d_2$ and $n > m$. Then the maximum number of mistakes a perceptron algorithm will make is higher on dataset $D^{(1)}$ than on dataset $D^{(2)}$.

upper bound depends on farthest datapoint
& Max geometric margin possible.

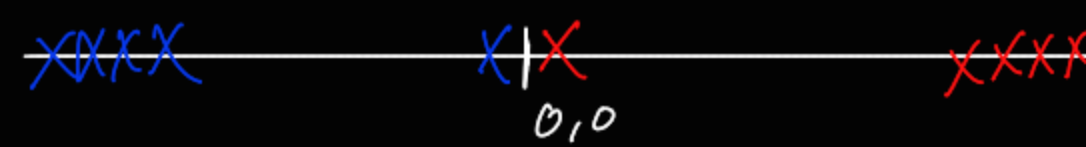
Counter example:

$$D_1 \in \mathbb{R}^2$$



total n points

$$D_2 \in \mathbb{R}^1$$



total m points

$$n = m$$

Multiclass - perceptron — Missing topic

PCA- PRINCIPLE COMPONENT ANALYSIS

→ Objective
To represent N dimension data in different basis $PC_1, PC_2, PC_3, \dots, PC_N$ such that the maximum variance deviation is captured in PC_1 , the second largest variance component is captured in PC_2 & so on until PC_N

→ Further after dimensionality reduction by discarding the last PC 's, change the data from N dimension to a K dimensional subspace of N ; $K < N$

→ original point $X_n = a_1 \hat{v}_1 + a_2 \hat{v}_2 + a_3 \hat{v}_3 + \dots + a_n \hat{v}_n$
for $\bar{X}_n = b_1 \hat{u}_1 + b_2 \hat{u}_2 + b_3 \hat{u}_3 + \dots + b_K \hat{u}_K$
where $K \leq N$ $v_1, v_2, v_3, \dots, v_K$ form a orthogonal basis for K -dim subspace

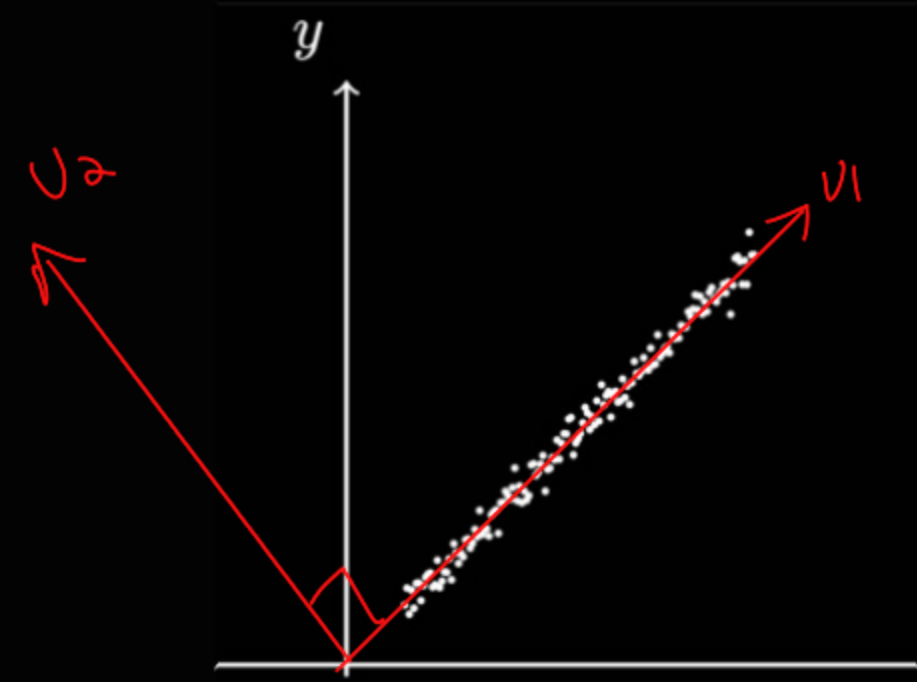
→ When $K=N$, only basis is changed but not discarded; Zero reconstruction error. $X_n = \bar{X}_n$

→ When $K < N$, some information is lost;

→ The data variance will be largest in PC_1

→ Second largest in PC_2

→ $\hat{v}_1, \hat{v}_2, \hat{v}_3, \dots, \hat{v}_K$ are linear combination of the original basis $\hat{v}_1, \hat{v}_2, \dots, \hat{v}_N$ } feature $\Rightarrow$ basis



Coordinates $b_1, b_2, \dots, b_K$ are projections of the original point X_n on the basis $v_1, v_2, \dots, v_K$ respectively.

$$X_n = a_1 \hat{v}_1 + a_2 \hat{v}_2 + \dots + a_n \hat{v}_n$$

$$\bar{X}_n = b_1 \hat{u}_1 + b_2 \hat{u}_2 + b_3 \hat{u}_3 + \dots + b_K \hat{u}_K \quad K \leq N$$

then $b_1 = \text{Project}(X_n \text{ on } v_1)$

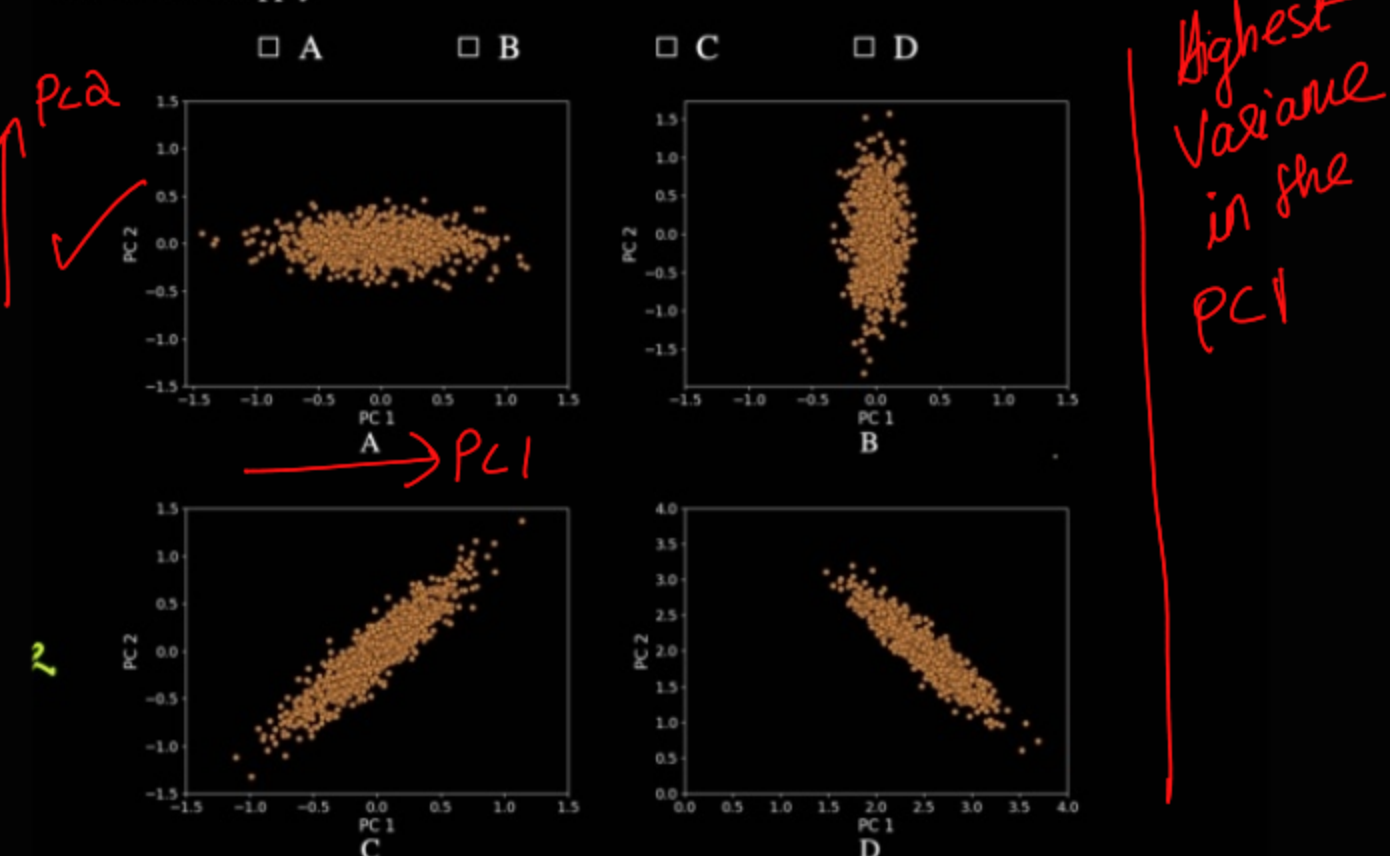
$$b_1 = \frac{u_1^T X_n}{u_1^T u_1} \rightarrow \|u_1\| = 1 \text{ since orthonormal basis}$$

scalar coefficient

$$\Rightarrow b_1 = u_1^T X_n$$

$$\bar{X}_n = (u_1^T X_n) \hat{u}_1 + (u_2^T X_n) \hat{u}_2 + \dots$$

[2 Pts] Suppose we project a new matrix A onto the directions of its first two principal components. Which of the following plots could possibly display the projected data with the first PC plotted along the horizontal axis and the second PC plotted along the vertical axis? Select all that apply.



PCA Wish List $\rightarrow$ How to choose principle components

Strategy

- Choose 1st principle component
- After choosing this then subtract PC_1 coefficients from the original data
- choose the principle component after subtracting
- get PC_1, PC_2 in order.

But how to choose PC_1 given data?

- optimise for representing maximum variance
- optimise for minimising reconstruction error.

OPTIMISE FOR REPRESENTING MAXIMUM VARIANCE

Let X_n be the n^{th} data point as a column vector & u_n be the unit vector in the direction of PC_n

then $X_n = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ \vdots \\ a_N \end{bmatrix} = a_1 \hat{v}_1 + a_2 \hat{v}_2 + \dots + a_N \hat{v}_N$
where $\hat{v}_1, \hat{v}_2, \dots, \hat{v}_N$ are the basis vector for standard basis.

then Let $\bar{X}_n$ be the projection of X_n on our PC basis $\hat{u}_1, \hat{u}_2, \dots, \hat{u}_K$

where $\bar{X}_n = b_1 \hat{u}_1 + b_2 \hat{u}_2 + \dots + b_K \hat{u}_K$

$$b_1 = \frac{u_1^T X_n}{u_1^T u_1} = u_1^T X_n \quad \text{Since } \hat{u}_1 \text{ is unit vector}$$

→ Now the Variance of the data is defined as $\text{Var}(X) = E[(x-\mu)^2]$

→ We want maximum variance in the projected data in PC1

→ ie the b_1 coefficient of the projected data to have maximum variance

⇒ Variance for X along $u \rightarrow \frac{1}{n} \sum_{i=1}^n \left[\underbrace{(u^T x_i - \underbrace{u^T \mu}_{\text{mean of projected data}})}^2 \right] = \frac{1}{n} \sum_{i=1}^n \left[u^T (x_i - \mu) \right]^2$

$= \frac{1}{n} \sum_{i=1}^n \left[\underbrace{u^T (x_i - \mu)}_{\text{since this quantity is scalar, we can transpose}} \underbrace{u^T (x_i - \mu)} \right] = \frac{1}{n} \sum_{i=1}^n \left[u^T (x_i - \mu) (x_i - \mu)^T \cdot u \right]$

$= u^T \left[\underbrace{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)(x_i - \mu)^T}_{\text{Covariance Matrix } S} \right] \cdot u$

⇒ Variance (X along unit vector $\hat{u}$) $= \hat{u}^T S \hat{u}$

→ Since we need to maximise variance, our optimisation becomes the constrained optimisation. i.e. Maximise $\hat{u}^T S \hat{u}$ subject to constrain $\|\hat{u}\| = 1$;

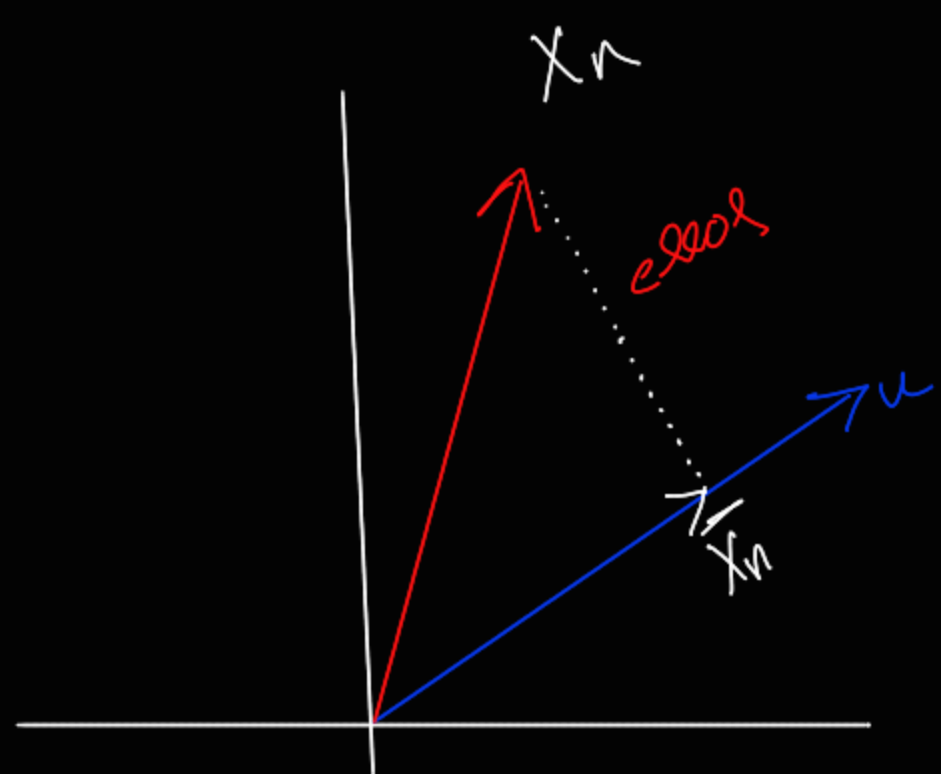
Solution = largest eigen value

∴ PC's are corresponding to $\lambda_1 > \lambda_2 > \lambda_3 \dots \lambda_N$
 $PC1, PC2, PC3 \dots PCN$ } $N \rightarrow \# \text{dimension}$
 $n \rightarrow \# \text{datapoints}$
 $\mu \rightarrow \text{mean of data.}$

$$\mu = \frac{1}{n} \sum_{i=1}^N x_i$$

PCA from Minimisation of Reconstruction error

if $u = \hat{u}$ (unit vector)



$$\bar{x}_n = \frac{u^T x_n}{u^T u} u$$

$$\bar{x}_n = (u^T x_n) \hat{u}$$

$$\begin{aligned} \bar{e}_n &= x_n - (u^T x_n) \hat{u} \\ \bar{e}_n &= x_n - \bar{x}_n \end{aligned} \rightarrow \text{Reconstruction error}$$

Minimise the L_2 norm / Length of $\bar{e}_n$

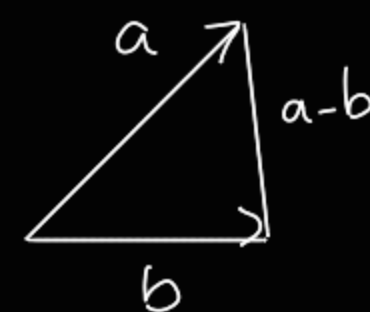
$$\sum_{i=1}^n \|x_i - \underbrace{u^T x_i}_{a-b} \hat{u}\|^2$$

st $\|u\| = 1$

$$= \sum_{i=1}^n \left\| x_i \right\|^2 + \left(u^T x_i \right)^2 \cdot \| \hat{u} \|^2 - 2 \cdot \left(u^T x_i \right)^2 - 2ab$$

$$= \sum_{i=1}^n \left\| x_i \right\|^2 - \left(u^T x_i \right)^2 \rightarrow \text{Minimise reconstruction error}$$

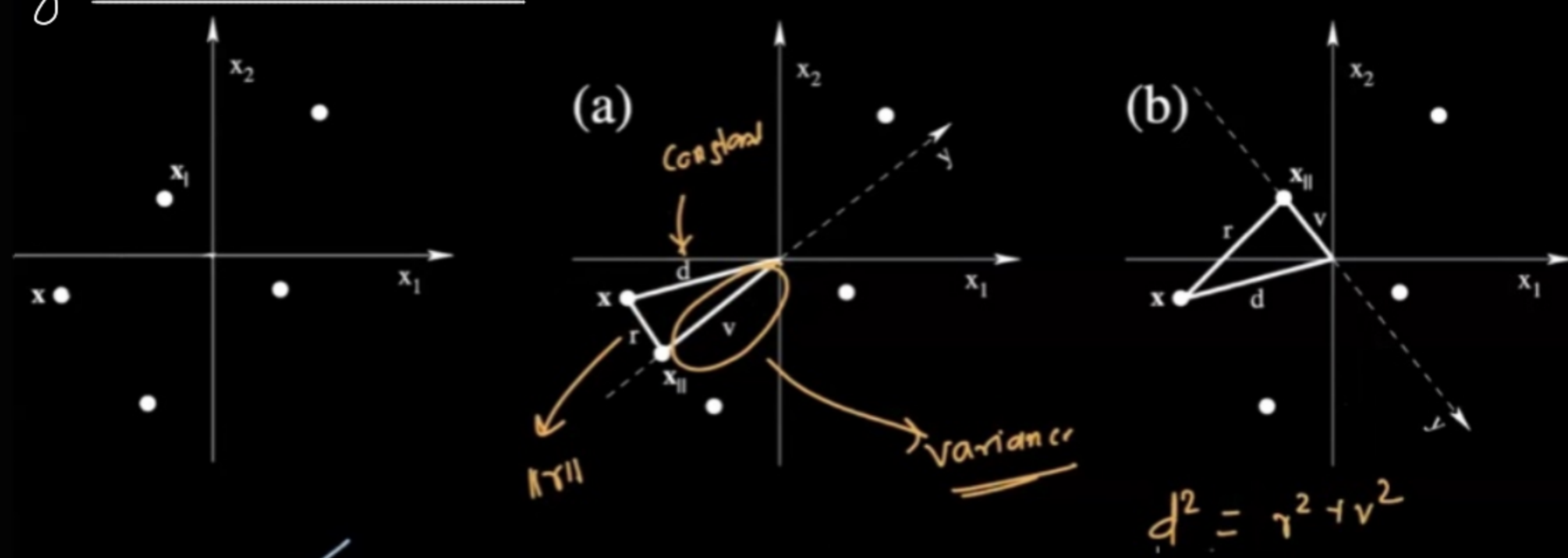
$$\rightarrow \text{Equivalent to Maximise } \sum_{i=1}^n \left(u^T x_i \right)^2 ; \| u \| = 1$$



$$\| a-b \|^2 = a^2 + b^2 - 2a \cdot b$$

$$\| \bar{a} - \bar{b} \|^2 = \| a \|^2 + \| b \|^2 + 2a^T b$$

Geometric interpretation



$$\text{reconstruction error} + \text{variance} = \text{datapoint}^2$$

Given 3 data points in 2-d space, (1, 1), (2, 2) and (3, 3),

(a) (1 pt) what is the first principle component?

(b) (1 pt) If we want to project the original data points into 1-d space by principle component you choose, what is the variance of the projected data?

(c) (1 pt) For the projected data in (b), now if we represent them in the original 2-d space, what is the reconstruction error?



$$a) u_1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right)$$

$$\text{also } \left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right)$$

$$\mu = \frac{1}{3} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \frac{2}{3} \begin{bmatrix} 2 \\ 2 \end{bmatrix} + \frac{3}{3} \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$b) \text{ Variance} = \frac{1}{n} \sum_{i=1}^n \left(u_1^T (x_i - \mu) \right)^2$$

$$= \frac{1}{3} \left(\begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}^T \begin{bmatrix} -1 \\ -1 \end{bmatrix} \right)^2 + \dots + \dots$$

$$= \frac{2 + 0 + 2}{3} = \underline{\underline{4/3}}$$

Method 2 - Using Matrix
Variance = $u^T S u$

$$S = \frac{1}{3} \begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}^T \begin{bmatrix} -1 & -1 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$= \frac{1}{3} \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

$$\text{Variance} = \frac{1}{3} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

$$= \frac{1}{3} \begin{bmatrix} 2\sqrt{2} & 2\sqrt{2} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \underline{\underline{4/3}}$$

[4 pts] Select the correct statements about principal component analysis (PCA).

- ☒ A: PCA is a method of dimensionality reduction
- ☒ B: If we select only one direction (a one-dimensional subspace) to represent the data, the sample variance of the projected points is zero if and only if the original sample points are all identical
- ☒ C: The orthogonal projection of a point x onto a unit direction vector w is $(x^T w)w$
- ☐ D: If we select only one direction (a one-dimensional subspace) to represent the data, PCA chooses the eigenvector of the sample covariance matrix that corresponds to the least eigenvalue

[4 pts] Consider a **centered** design matrix $X \in \mathbb{R}^{n \times d}$, where $X_i \in \mathbb{R}^d$ is the i -th sample point (a column vector) and X_i^T is the i -th row of X . Which of these optimization problems is a correct formulation of finding the first principal component v ?

- ☒ A: Subject to $\|v\|_2 = 1$, find $v \in \mathbb{R}^d$ that minimizes $\sum_{i=1}^n \|X_i - vv^T X_i\|_2^2$
- ☐ B: Find $v \in \mathbb{R}^d$ that minimizes $\frac{v^T X^T X v}{\|v\|_2^2}$
- ☐ C: Subject to $\|v\|_2 = 1$, find $v \in \mathbb{R}^d$ that maximizes $\sum_{i=1}^n \|X_i - vv^T X_i\|_2^2$
- ☒ D: Find $v \in \mathbb{R}^d$ that maximizes $\frac{v^T X^T X v}{\|v\|_2^2}$

Maximise $\frac{1}{n} u^T X^T X u$

Maximise $\frac{1}{n} \sum_{i=1}^n u^T X_i \cdot X_i^T u$

Variance given any unit vector v .

$$\rightarrow \text{Variance} = \frac{1}{n} \cdot \sum_{i=1}^n v^T X_i X_i^T v = v^T \left[\frac{1}{n} \sum_{i=1}^n X_i X_i^T \right] v =$$

$\rightarrow$ Now if v is eigen vector, then $Sv = \lambda v$ $\neq$ Variance $= v^T \lambda v = \lambda v^T v = \underline{\underline{\lambda}}$

PCA & Linear regression

PCA minimises

$$\sum_{n=1}^N x_n - \sum_{i=1}^k (u_i^T x_n) \hat{u}_i$$

$\rightarrow$ Reconstruction error

Linear regression minimises

$$\sum_{n=1}^N (y_n - w^T x_n)^2$$

$\rightarrow$ Residual error



Linear regression or PCA?



Linear regression or PCA?

Question: T/F

(2 points) **True or False:** In PCA, we project the mean-centered data matrix X to a lower dimensional space by choosing the eigenvectors of X corresponding to the largest eigenvalues.

False

We choose eigen vectors of Covariance matrix XX^T assuming datapoints are arranged in a column

Consider the following design matrix, representing four sample points $X_i \in \mathbb{R}^2$.

$$X = \begin{bmatrix} 4 & 1 \\ 2 & 3 \\ 5 & 4 \\ 1 & 0 \end{bmatrix}$$

We want to represent the data in only one dimension, so we turn to principal components analysis (PCA).

[5 pts] Compute the **unit-length principal component directions** of X , and **state which one the PCA algorithm would choose** if you request just one principal component. Please provide an exact answer, without approximation.

$$P = \begin{bmatrix} 3 & 2 \end{bmatrix}$$

$$X_{\text{centered}} =$$

$$\begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 2 & 2 \\ -2 & -2 \end{bmatrix}$$

$$\text{Covariance } S = \frac{1}{n} X^T X = \frac{1}{4} \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix}$$

eigen values are 4, 1; largest is 4

$$\lambda^2 - 20\lambda + 64 = 0$$

$$(\lambda - 16)(\lambda - 4) = 0$$

$$\lambda = 16, \lambda = 4$$

eigen vectors $\frac{1}{4} \begin{bmatrix} -6 & 6 \\ 6 & -6 \end{bmatrix} \begin{bmatrix} v \\ v \end{bmatrix} = D$ $V = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$

→ variance along this PC is 4.

MATRIX FORMULATION OF COVARIATION OF PROJECTED DATA

→ If data matrix X is formed using columns of data x_n , then $V^T X$ is given as

projected data = $\begin{bmatrix} \leftarrow V^T \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow x_1 \downarrow \\ \uparrow x_2 \downarrow \\ \dots \\ \uparrow x_n \downarrow \end{bmatrix} = V^T X$

→ In case of multiple $V_1, V_2, \dots, V_k$ then projected data is given as.

$$\begin{bmatrix} \uparrow \bar{x}_1 \downarrow \\ \uparrow \bar{x}_2 \downarrow \\ \dots \\ \uparrow \bar{x}_n \downarrow \end{bmatrix} = \begin{bmatrix} \leftarrow V_1^T \rightarrow \\ \leftarrow V_2^T \rightarrow \\ \vdots \\ \leftarrow V_k^T \rightarrow \end{bmatrix} \begin{bmatrix} \uparrow x_1 \downarrow \\ \uparrow x_2 \downarrow \\ \dots \\ \uparrow x_n \downarrow \end{bmatrix} = V^T X$$

where V is a matrix of principle components as columns

X is a matrix of data points as columns

$\bar{X}$ is a matrix of projected data as columns.

$$\Rightarrow \boxed{\bar{X} = V^T X}$$

→ Covariance Matrix of original data assuming centered data = $\frac{1}{n} X X^T$

→ Covariance Matrix of projected data

$$= \frac{1}{n} (V^T X) (V^T X)^T = \frac{1}{n} V^T X \cdot X^T V$$

$$= \frac{1}{n} V^T V \Lambda = \frac{1}{n} \Lambda$$

$$= \frac{1}{n} \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & d_3 \dots \end{bmatrix}$$

will be diagonal matrix Λ of $X X^T$ since V is a set of eigen vectors of $X X^T$

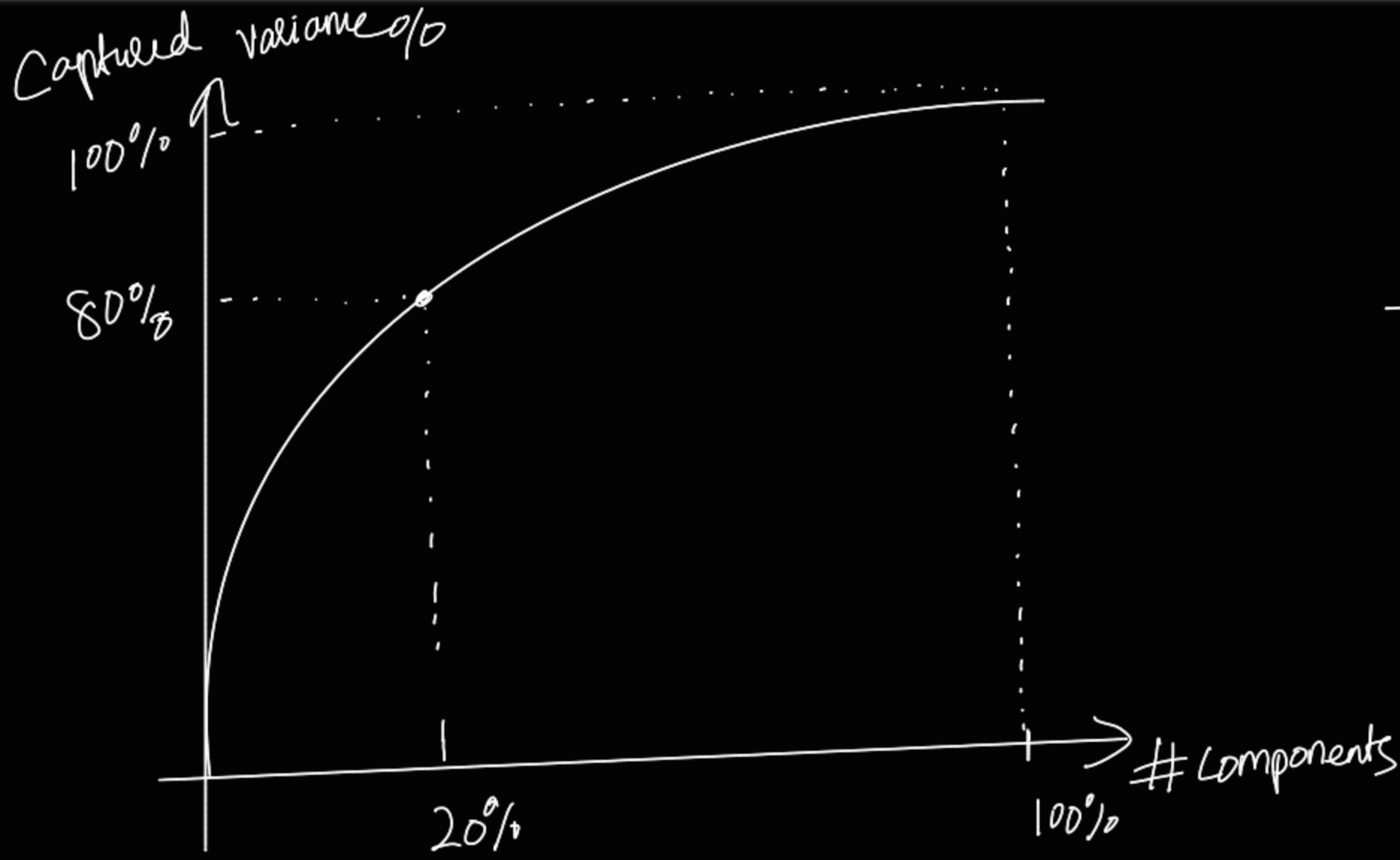
→ The features in the PCA are completely uncorrelated to each other.

→ Since the features are uncorrelated we can say $\text{Covariance}(\text{feature A}, \text{feature B}) = \text{Variance}(\text{feature A}) + \text{Variance}(\text{feature B})$

⇒ Total Variance captured by Top k Components

$$\frac{d_1 + d_2 + \dots + d_k}{d_1 + d_2 + \dots + d_N}$$

N - dim of data
 k - dim of projected data.



80-20 rule

→ generally it is seen for real world data that 20% components capture 80% of data variance

Reconstruction error notation for PCA into more than one component

$$\sum_{i=1}^n \|x_n - V V^T x_n\|^2$$

$V V^T x_n \rightarrow$ projected data in original basis

$V^T x_n \rightarrow$ projected data in new basis V

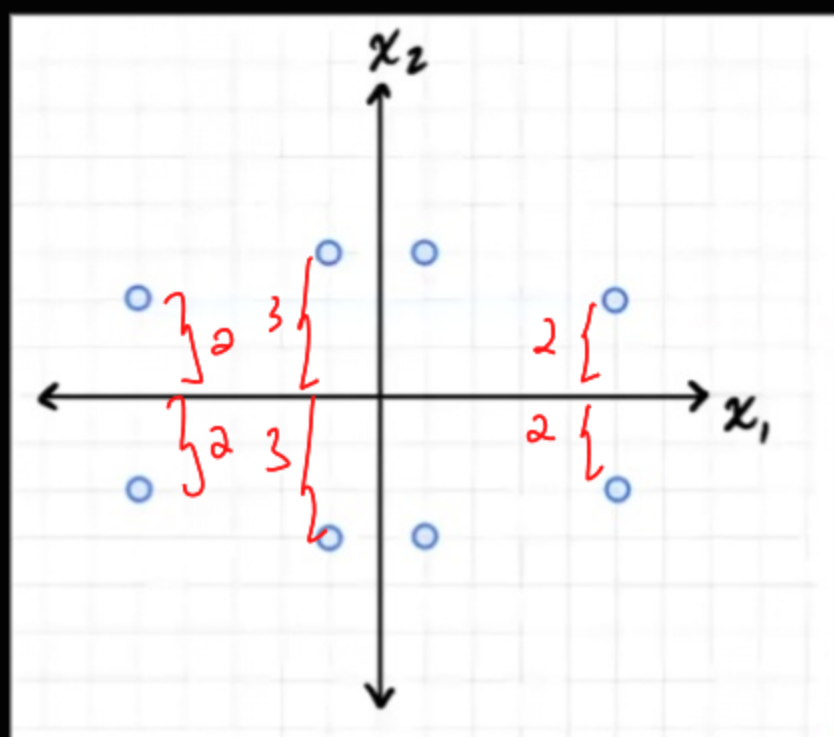
observation

$$\rightarrow V V^T \neq I$$

$$\rightarrow V V^T = I \text{ only if } V \text{ is square}$$

$$\Rightarrow V V^T = V^T V = I$$

→ if we don't discard any PCA component, then reconstruction error is zero.



Suppose PCA is used to reduce the dimension to 1.

What will be the reconstruction error?

$$R_e = 2^2 + 3^2 + 2^2 + 3^2 + 2^2 + 3^2 + 2^2 + 3^2$$

$$= 16 + 36 = 52 //$$

Consider the following data set, represented as three points in $\mathbb{R}^2$. Note: in this problem we will **not** demean the dataset. Perform all calculations as if the dataset were 0 mean.

$$\begin{bmatrix} 8 & 2 \\ 1.5 & 3 \\ 6 & 12 \end{bmatrix}$$

Covariance $\frac{1}{3} X^T X$

(a) What is the first principal component vector, v_1 ?

(b) What is the second principal component, v_2 ?

(c) If we use only the first principal component to compress the dataset, what will the representation of each point be?

(d) Will this representation be lossy, or perfectly preserve the data?

First component = $\begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \rightarrow$ captures all variance

Second component = $\begin{bmatrix} \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} \end{bmatrix} \rightarrow$ zero variance

Consider 3 data points in the 2-d space: $(-1, -1), (0,0), (1,1)$.

[6 points] What is the first principal component (write down the actual vector)?

$$u = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

[7 points] If we project the original data points into the 1-d subspace by the principal component you choose, what are their coordinates in the 1-d subspace? And what is the variance of the projected data?

$$\begin{aligned} \bar{x}_1 &= -\sqrt{2} \\ \bar{x}_2 &= 0 \\ \bar{x}_3 &= \sqrt{2} \end{aligned} \quad \left. \begin{array}{l} \text{Variance} = \frac{1}{3}(2+2) = \frac{4}{3} \end{array} \right\}$$

[6 points] For the projected data you just obtained above, now if we represent them in the original 2-d space and consider them as the reconstruction of the original data points, what is the reconstruction error?

$$L(=0)$$

Answer the same questions for the following, slightly larger dataset:

$$\begin{bmatrix} 1 & 1 \\ 1.5 & 1.5 \\ -2 & 2 \\ 4 & -4 \\ 6 & -6 \\ 2 & 2 \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} u_2$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} u_1$$

(a) What is the first principal component vector, v_1 ? u_1

(b) What is the second principal component, v_2 ? u_2

(c) If we use only the first principal component to compress the dataset, what will the representation of each point be?

(d) Will this representation be lossy, or perfectly preserve the data?

Yes

No

$$\bar{x}_1 = [\sqrt{2}, 0]$$

$$\bar{x}_2 = [\frac{3}{\sqrt{2}}, 0]$$

$$\bar{x}_3 = [0, 2\sqrt{2}]$$

$$\bar{x}_4 = [0, -4\sqrt{2}]$$

$$\bar{x}_5 = [0, -6\sqrt{2}]$$

$$\bar{x}_6 = [2\sqrt{2}, 0]$$

PC2 $\rightarrow$ Variance along $u_1 <$ PC1 $\rightarrow$ Variance along $u_2 =$

QUESTION 1

Consider n feature vectors $x \in \mathbb{R}^d$ that need to be projected into a lower-dimensional space $\mathbb{R}^k$ using k principal component basis vectors. The projection matrix W of size $d \times k$ consists of these orthonormal principal component vectors.

Which of the following expressions correctly represents the reconstruction error?

1. A. $\|x - WW^T x\|_2^2$ ✓

2. B. $\|x - W^T x\|_2^2$

3. C. $\|x - W^T W x\|_2^2$

4. D. $\|x - W x\|_2^2$

$$\begin{aligned} W^T x &\rightarrow k \text{ dim} \\ x &\rightarrow d \text{ dim} \\ WW^T x &\rightarrow k \text{ dim} \end{aligned}$$

www.alinau.com +9

PCA & SVD relation

In diagonalisation of A $AV = V\Lambda$
if V^{-1} exist $\rightarrow$ Linear independent eigen vectors.

where V is a matrix formed using eigen vectors of A as the column.
& Λ is the matrix formed with corresponding eigen values on diagonal

$$\rightarrow A = V\Lambda V^{-1}$$

choose orthogonal V

$$\Rightarrow A = V\Lambda V^T \rightarrow \text{Diagonalisation or Eigen value decomposition of } A$$

In SVD projected data $= \bar{x} = U^T X$
projected data covariance $= \frac{1}{n} U^T X (X U^T)$

X has data point as columns.

$$\text{Covariance of projected data} = \frac{1}{n} U^T X X^T U$$

→ We want PC_n & PC_k to be uncorrelated for $n \neq k$.

→ We want diagonal covariance matrix

$$\therefore \text{Covariance of projected data} = \text{diagonal Matrix} = \frac{1}{n} U^T X X^T U$$

→ We want U to be Matrix formed by eigen vectors of $X X^T$

$$\Rightarrow X X^T U = U \Lambda$$

$$\Rightarrow \text{Covariance} = \frac{1}{n} U^T U \Lambda = \frac{1}{n} \Lambda$$

→ if we choose the principal components as the eigen vectors of $X X^T$, then our projected data will have completely uncorrelated features. ★

QUESTION

Suppose a d -dimensional dataset is reduced to $\mathbb{R}^3$ with PCA. Suppose the covariance matrix of the new dataset is computed. Which of the below could possibly be this covariance matrix?

○ $\begin{pmatrix} 5 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 2 \end{pmatrix}$

○ $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 0 \\ 3 & 0 & 3 \end{pmatrix}$

○ $\begin{pmatrix} 7 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ ✓ → Projected Data Covariance = always diagonal.

○ $\begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 3 \\ 2 & 3 & 0 \end{pmatrix}$

$\begin{pmatrix} 7 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & -1 \end{pmatrix}$ → Variance can't be negative. Diagonal entry are value of PCs $X X^T$ → atleast +ve semi definite.

SVD of data Matrix

Given any $m \times n$ matrix A we can do singular value decomposition.

$$A_{m \times n} = U_{m \times m} \sum_{m \times n} V^T_{n \times n}$$

left singular right singular

where $\sum$ has exactly r non zero entry on the diagonal
 $r = \text{rank}(A)$

$$\frac{A^T A}{n \times n} = \underbrace{V^T}_{n \times n} \underbrace{\sum^T}_{n \times n} \underbrace{V}_{n \times n}$$

$$\frac{A A^T}{m \times m} = \underbrace{U}_{m \times m} \underbrace{\sum^T}_{m \times m} \underbrace{U}_{m \times m}$$

$$\frac{A V}{m \times n} = \underbrace{U}_{m \times m} \underbrace{\sum}_{n \times n}$$

Now, if we consider data matrix X , then in PCA.

$$XX^T = U \Sigma \Sigma^T U^T$$

singular
value's

$$XX^T = U \Lambda V^T$$

$U \rightarrow$ eigen vectors of XX^T

$\Sigma \Sigma^T = \Sigma^2$ since Σ is diagonal

$\Sigma^2 = \Lambda \rightarrow$ eigen values of XX^T

Suppose the singular values of a centered design matrix X are 1, 2, and 3. Find the percentage of the total variance captured by the first principal component.

$$= \frac{2^2}{1^2 + 2^2 + 3^2} \times 100 = \frac{4}{14} \times 100$$

Variance along PC1 = $\frac{3^2}{n}$

Let X be a matrix containing data samples as columns, and suppose X is centered such that each feature has a mean of zero. Which of the statements are TRUE?

- A: The principal components are eigenvectors of the centered data matrix. ✗
- B: The principal components are left singular vectors of the centered data matrix. ✓
- C: The principal components are eigenvectors of the sample covariance matrix. ✓
- D: The principal components are left singular vectors of the sample covariance matrix. ✓



Since covariance matrix is symmetric SVD will be equal to diagonalisation of (Covariance Matrix)².
Square $\Rightarrow$ eigen vectors don't change, only eigen values change.

Projected Data Matrix as SVD

Let data matrix $X = N \times D \rightarrow$ #dimension
 $\hookrightarrow$ # Datapoints

$X = U \Sigma V^T$
 $X^T X =$ Covariance Matrix
 $V \rightarrow$ Right singular Matrix
 $\hookrightarrow$ Represents principle components

first k rows of V
 $\leftarrow$ D vectors

$\Rightarrow$ Projected coordinates = $XV = U \Sigma V^T V = U \Sigma$
 $\hookrightarrow$ using all of V
 $\rightarrow$ Limiting to first k principle components.

Projected coordinates = $XV_k = U \Sigma_k V_k^T$
 $V_k \rightarrow$ Matrix with only first k components
 $V_k^T V_k \rightarrow$ Becomes a diagonal matrix very similar to identity but with only 1 in the k leading positions. $\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 & & 0 \dots 0 \end{bmatrix}$

$X_k = U \Sigma_k$ projected coordinates $\Sigma_k = \Sigma V_k^T V_k$

$\rightarrow X_k V_k^T = U \Sigma_k V_k^T$ projected data

k^{th} rank approximation

$X_k = X V_k$
 $X_k V_k^T = X V_k V_k^T$
 $\|X - X_k V_k^T\|$ reconstruction error.

Variance Shortcut

→ for A and eigen vector x $Ax = \lambda x$
 → if we take x as unit vector $\hat{w}_1 \Rightarrow$ then $A\hat{w}_1 = \lambda\hat{w}_1$
 The eigen value corresponding to $\hat{w}_1$ can be found out by

$$\begin{aligned}\hat{w}_1^T A \hat{w}_1 &= \hat{w}_1^T \lambda \hat{w}_1 \\ &= \lambda \underbrace{\hat{w}_1^T \hat{w}_1}_{\text{unit vector} = 1} \\ &= \lambda\end{aligned}$$

We can use this property to find variance in projected direction $\hat{w}_1$

Data Matrix X
 direction $\hat{w}_1 \Rightarrow$ Projected data = $X\hat{w}_1$
 coordinates

Assuming centered data

$$\begin{aligned}n \cdot (\text{Variance in this direction}) &= \|X\hat{w}_1\|^2 \rightarrow \text{Assuming centered data} \\ &= \hat{w}_1^T X^T X \hat{w}_1 \\ &= \hat{w}_1^T X^T X \hat{w}_1\end{aligned}$$

$$\text{Variance} = \frac{\hat{w}_1^T C \hat{w}_1}{n}$$

C = Covariance matrix
 n = # datapoints

PCA & OLS

→ Projecting data is unsupervised $\equiv$ Minimising Reconstruction error
 Minimising Mean of squared projection distance
 Maximising Sample Variance
 Happens in input featurespace
 $\neq$ o/p - i/p space

→ DLS $\rightarrow$ Happens in i/p o/p space

→ PCA removes info from dataspace $\equiv$ OLS performance can only degrade
 OLS loss can only increase never decrease

Proof that OLS loss never decrease after PCA projection

→ If any prediction done by PCA $= \hat{y}_k = X_k \theta_k$
 can be recreated by a θ on the unprojected data ie $\hat{y} = X\theta$, then PCA cannot improve the performance of OLS.

$$\begin{aligned}X &= n \times d \\ \theta &= d \times 1\end{aligned}$$

n — # samples
 d — i/p dimension

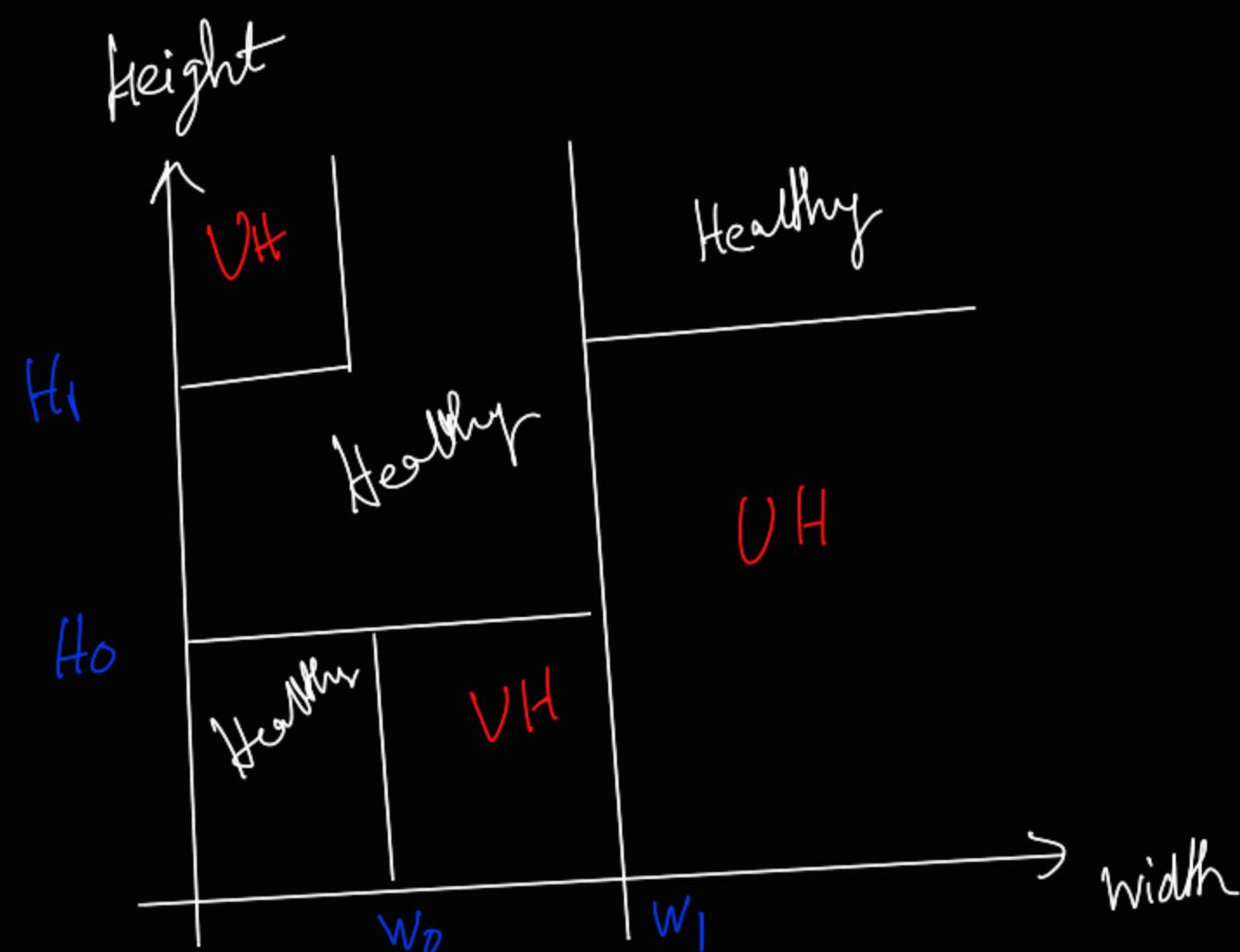
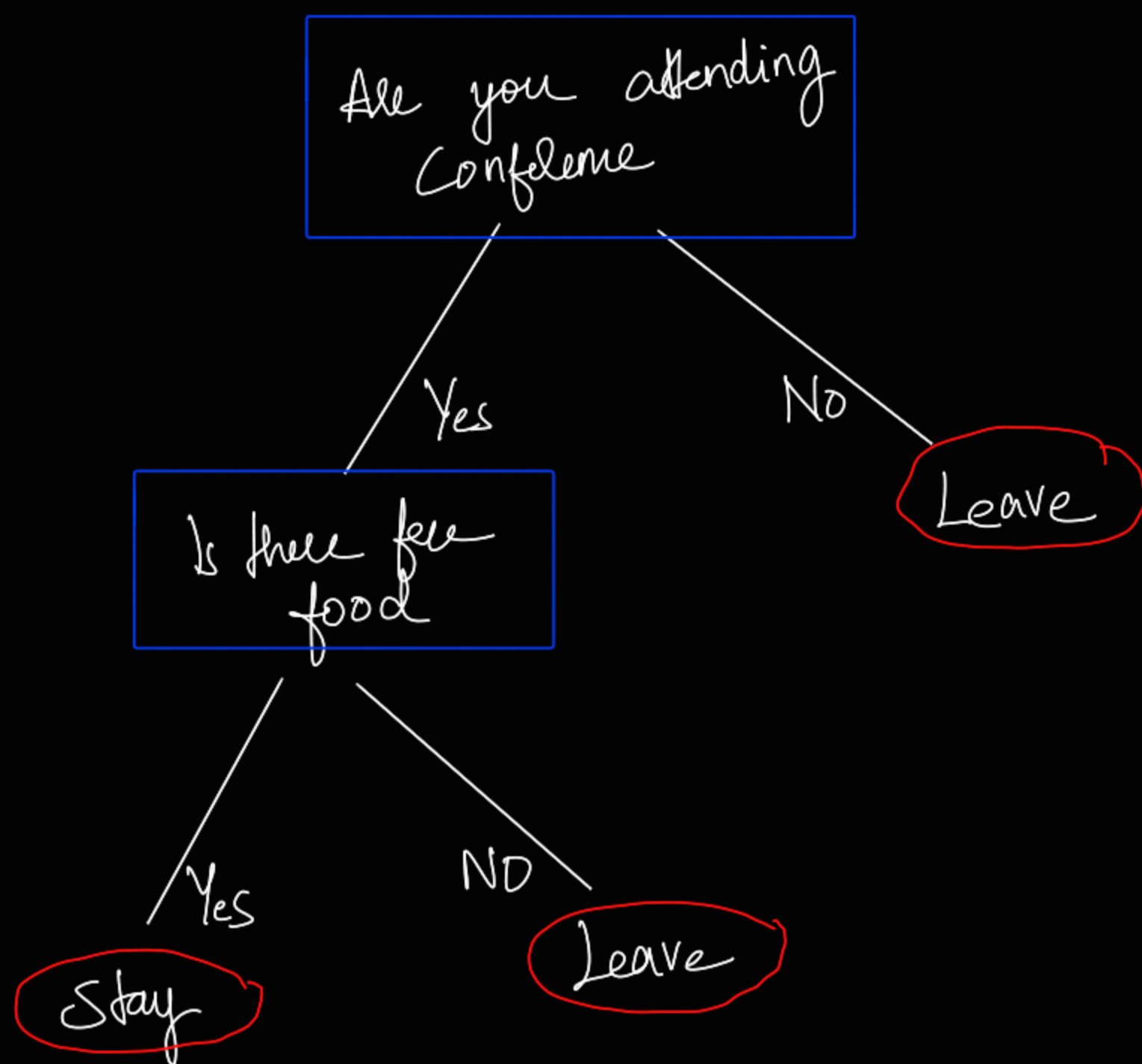
$$\begin{aligned}X_k \theta_k &= U \Sigma_k V^T \theta_k \\ &= U \Sigma_k V^T V_k V^T \theta_k \rightarrow \text{we have seen this} \\ &= U \Sigma_k V^T V_k V^T \theta_k \text{ Take as } \theta \\ &= U \Sigma_k V^T \theta \\ &= X \theta\end{aligned}$$

$\Rightarrow$ PCA doesn't improve OLS performance, it can stay same or worsen because of loss of information.

Decision Trees

Used for classification & regression

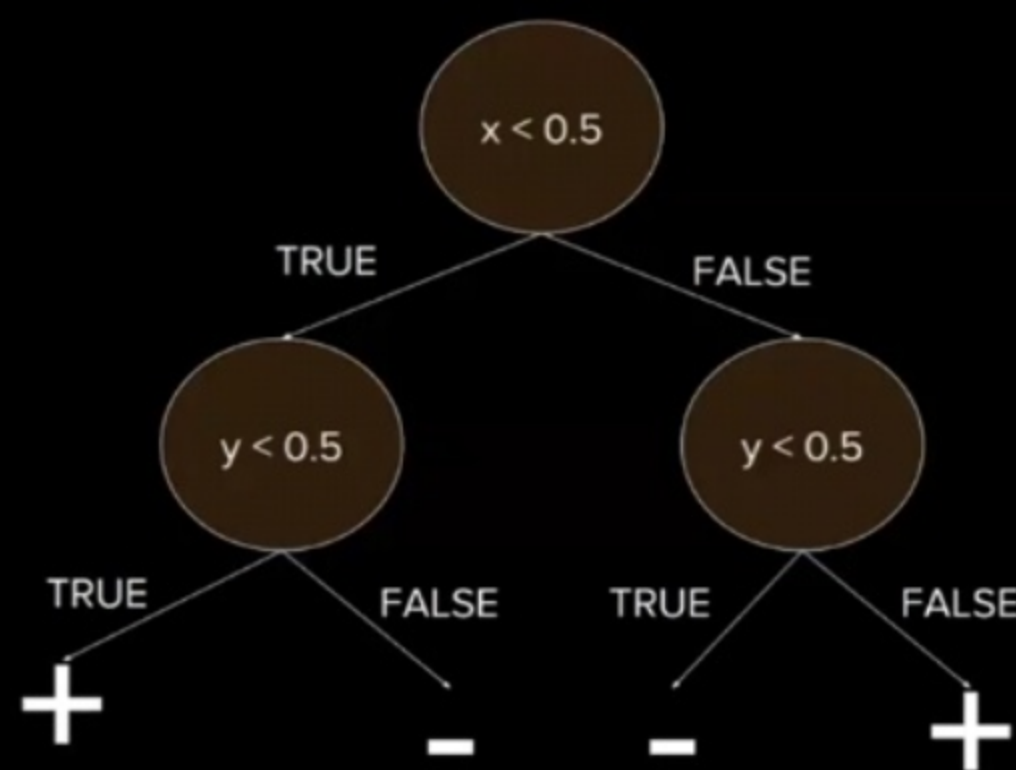
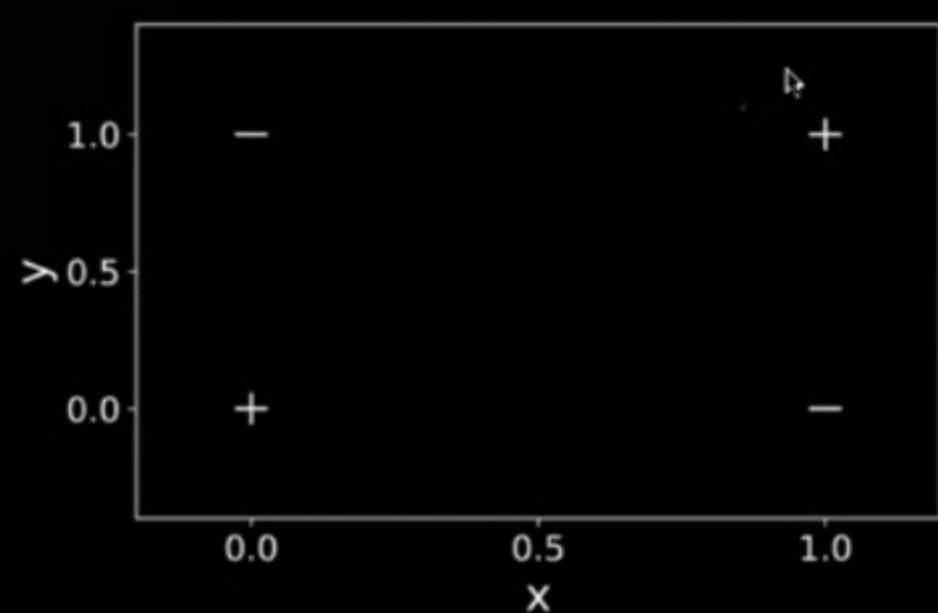
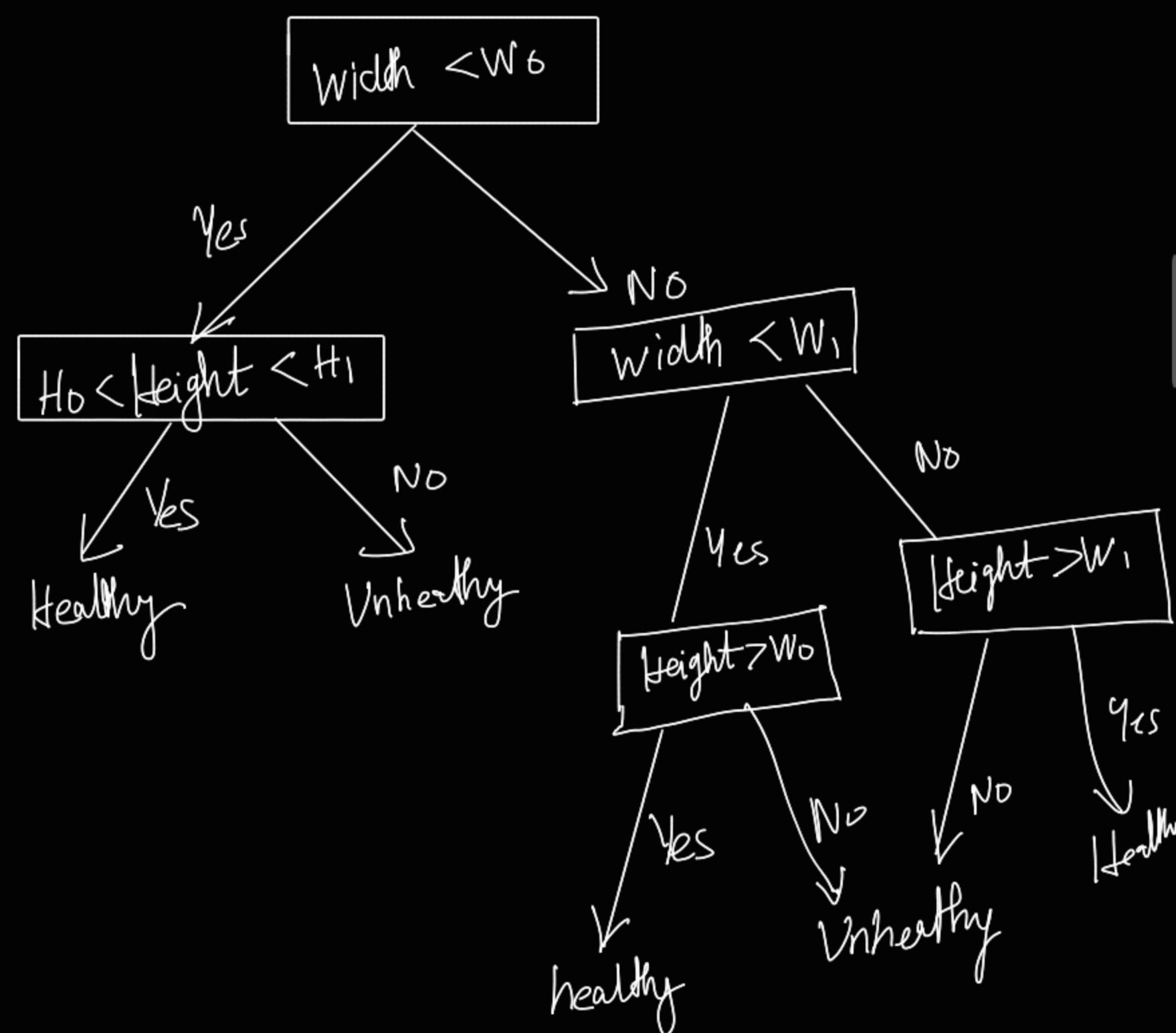
Example



Observations

- Decision tree need not be unique
- Some trees are better than others due to less depth or less error.

The key question is which topic to choose at each stage in a decision tree?



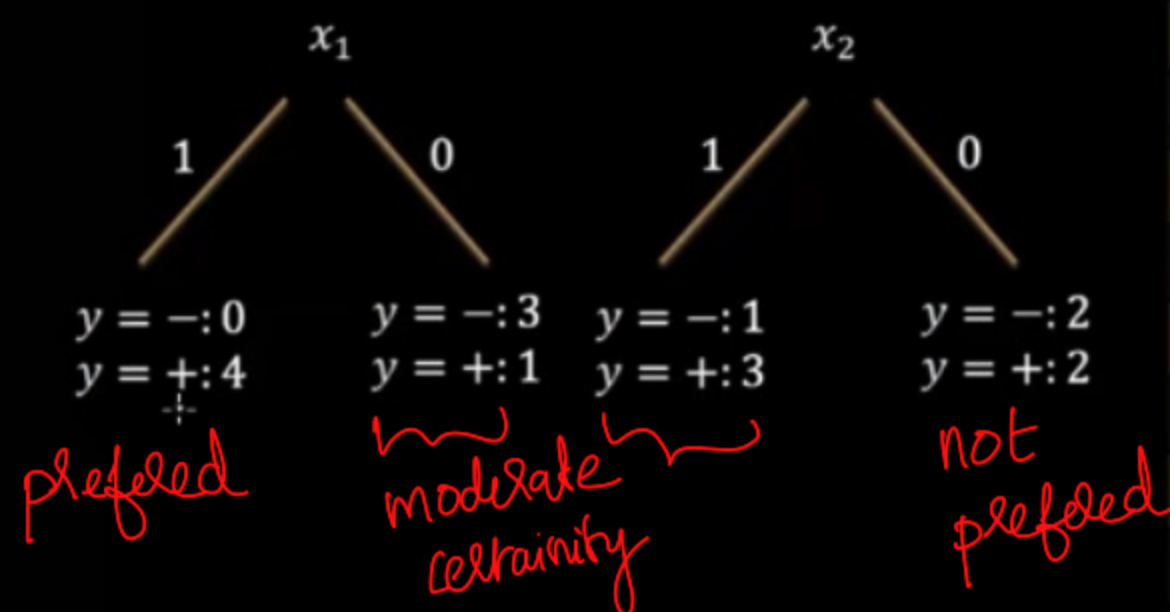
XOR problem

→ classify with depth 2

ENTROPY

$x_1, x_2 \in \{0,1\}$

Which attribute should you split on?



x_1	x_2	y
1	1	+
1	0	+
1	1	+
1	0	+
0	1	+
0	0	-
0	1	-
0	0	-

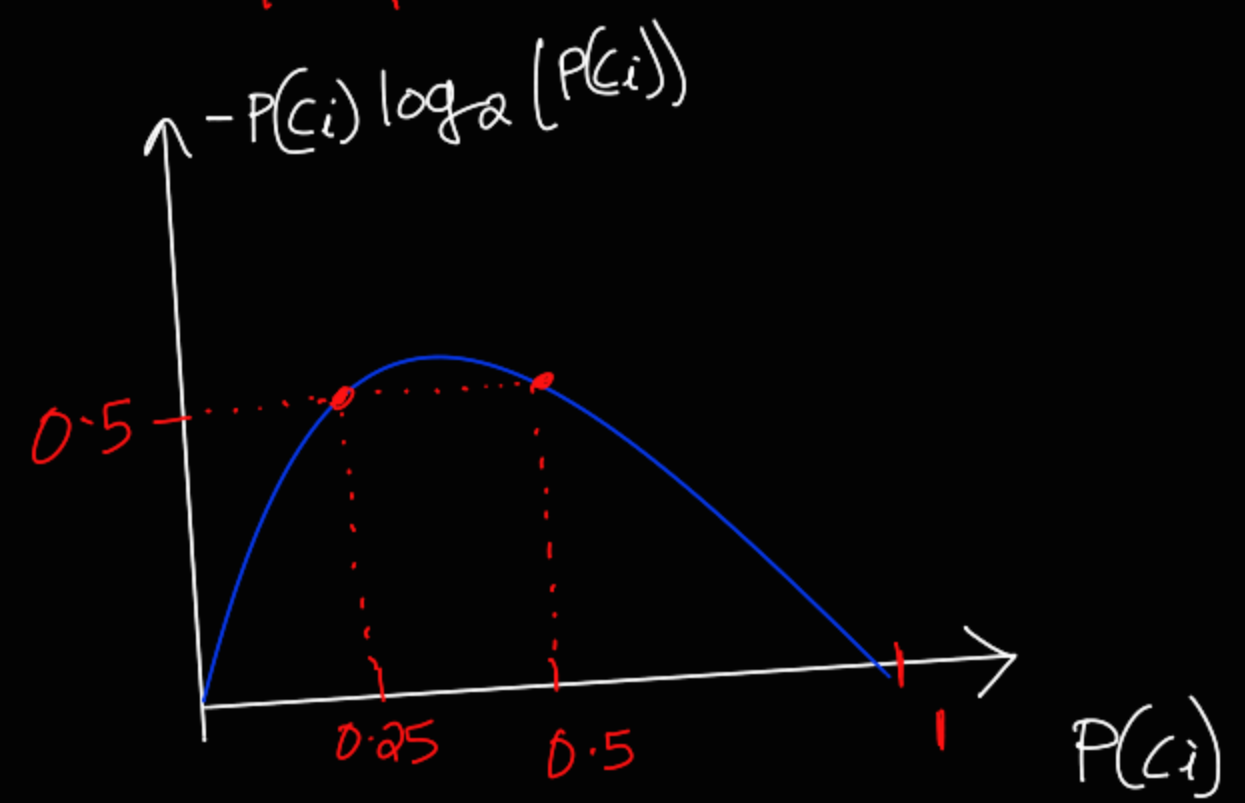
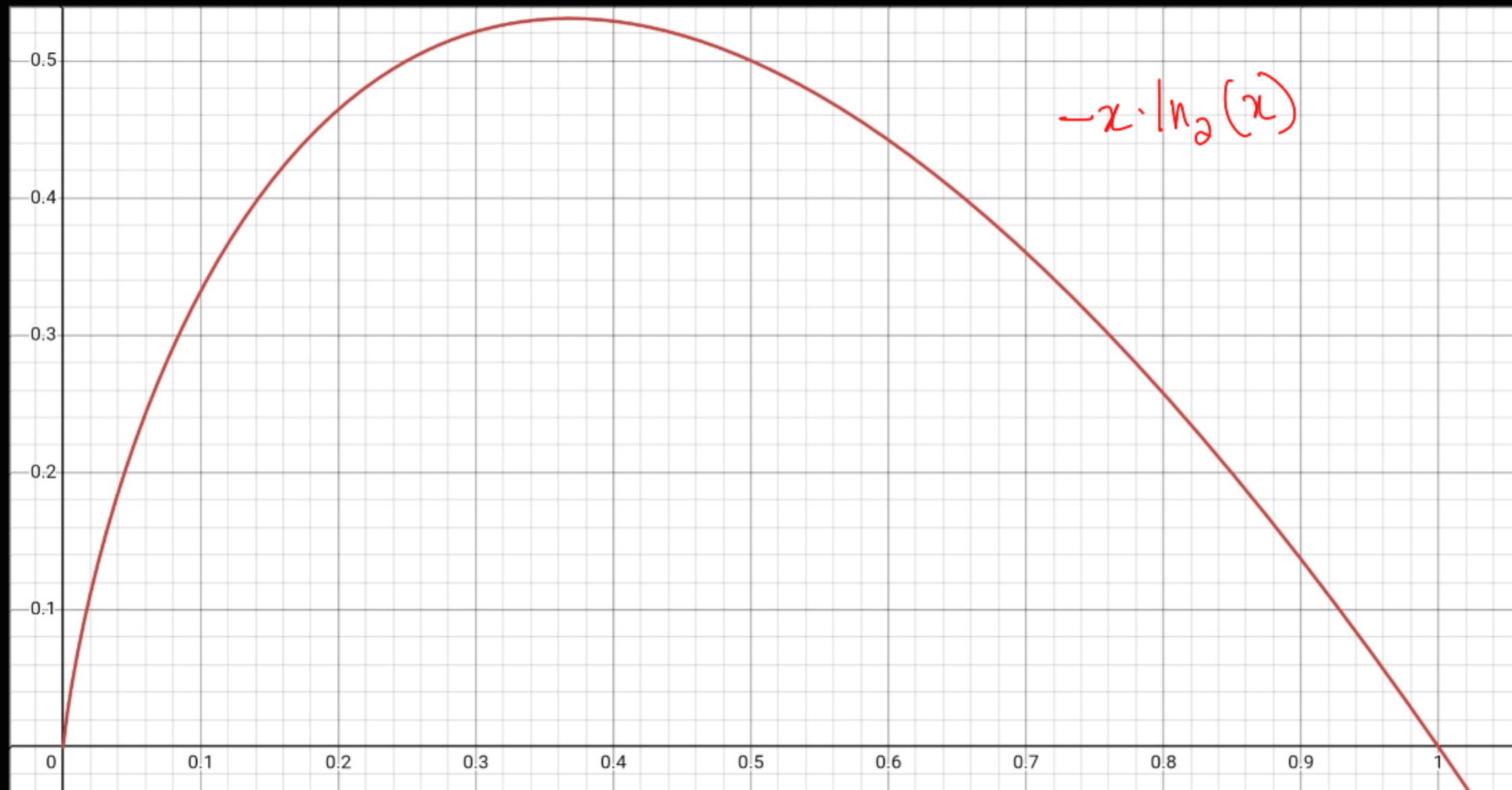
Uncertainty → Entropy

- Considers a probability distribution with k outcomes, $c_1, c_2, c_3, \dots, c_k$
- Probability of each outcome is $P(c_i)$

Then Entropy I is given by

$$I(p(c_1), p(c_2), \dots, p(c_k)) = - \sum_{i=1}^k p(c_i) \log_2(p(c_i))$$

→ Base 2 since information is measured in bits
 → Result is non negative & represents uncertainty.



eg. **Problem:** Consider a binary distribution $(p, 1-p)$ where $0 \leq p \leq 1$.

What is the entropy of the distribution $(0.5, 0.5)$?

- (A) 0.2 (C) 0.6
 (B) 0.4 (D) 1 ✓

$$-\frac{1}{2} \log_2\left(\frac{1}{2}\right) - \frac{1}{2} \log_2\left(\frac{1}{2}\right)$$

eg. $p(u) = 0.09$
 $p(u) = 0.99$
 entropy = $-0.09 \log_2(0.09) - 0.99 \log_2(0.99)$
 ≈ 0.08 less uncertainty.

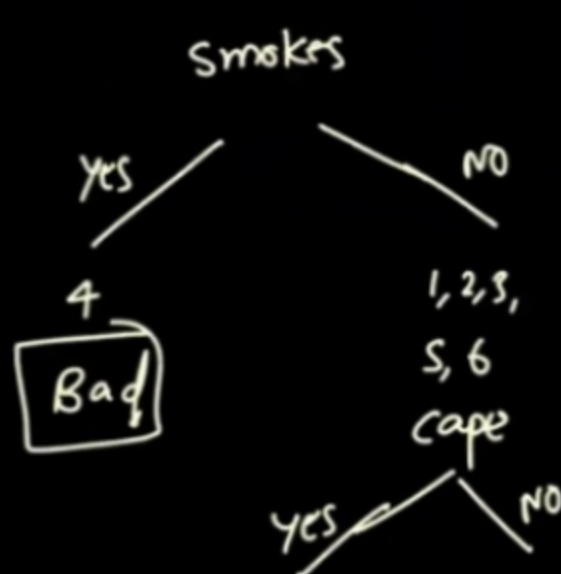


Curve of entropy for Binary outcomes

$$-p \log_2(p) - (1-p) \log_2(1-p)$$

For Binary classification, the entropy is between 0 & 1.

	mask	cape	tie	ears	smokes	class
training data						
1 batman	yes	yes	no	yes	no	Good
2 robin	yes	yes	no	no	no	Good
3 alfred	no	no	yes	no	no	Good
4 penguin	no	no	yes	no	yes	Bad
5 catwoman	yes	no	no	yes	no	Bad
6 joker	no	no	no	no	no	Bad



Entropy of smokes → yes category

$$-1 \log 1 + -0 \log 0 = 0 \text{ entropy.}$$

→ Here there is only one example in this class.
 → Entropy is zero
 → Therefore just looking for minimum

entropy is not a good idea.
 → A good classifying branch is decided by weighted entropy. [weighted by size of partition]

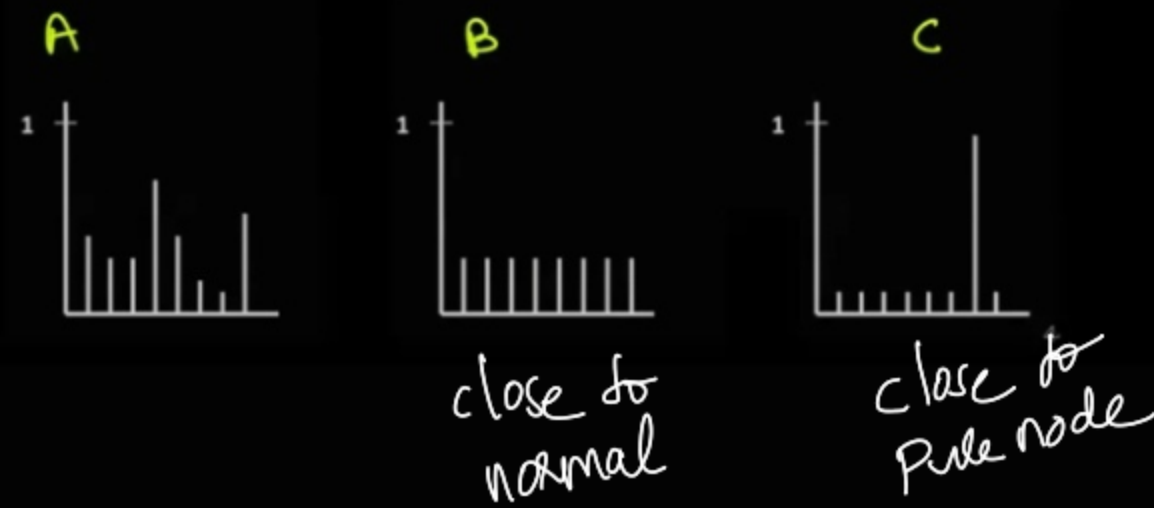
→ Pure Node's always have zero entropy

→ The more close to uniform a distribution, the higher the entropy

→ The more skewed & close to pure node, the lower is the entropy

Which of the following distribution has highest or lowest entropy?

$$B > A > C$$

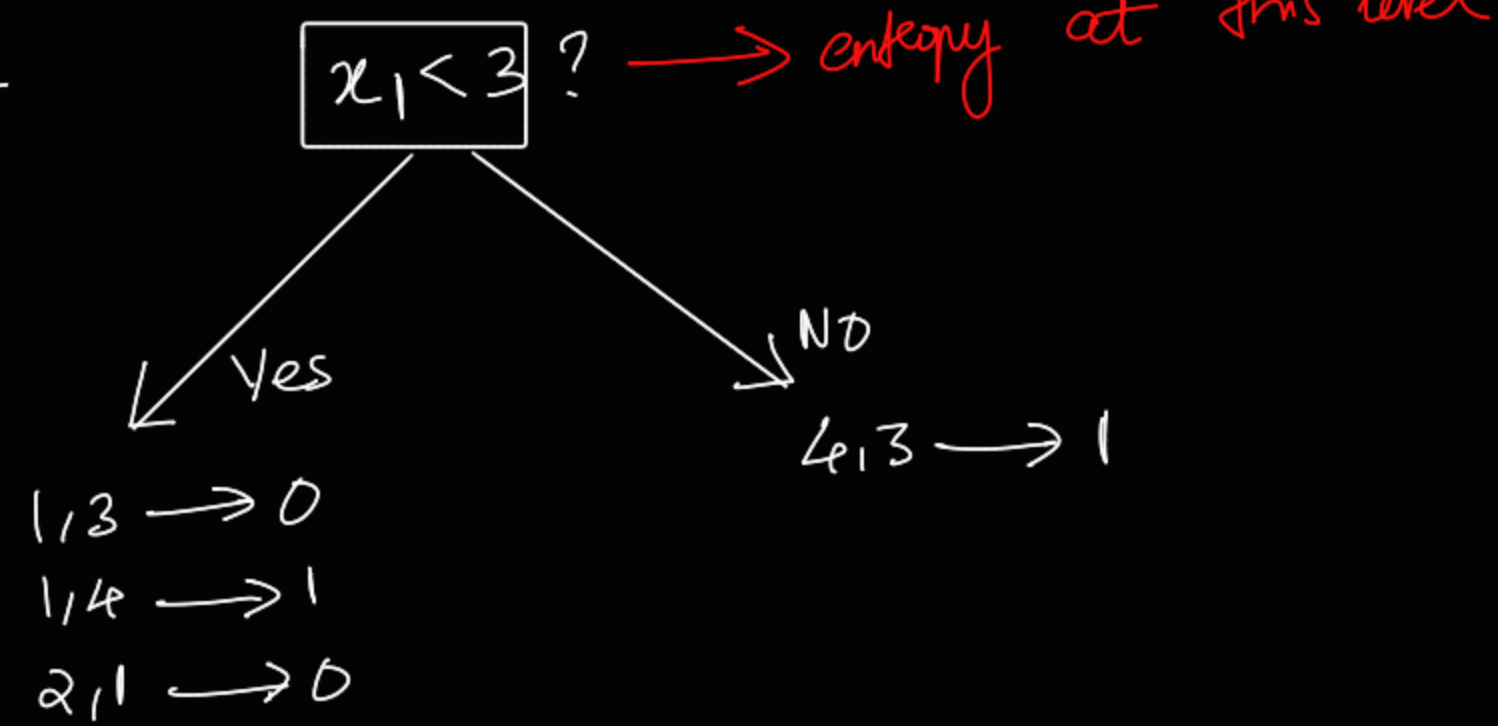


The top (root) node of a decision tree for this dataset has an entropy of 1

x_1	x_2	y
1	3	0
1	4	1
2	1	0
4	3	1

Top Node → sees all examples

example:-



$$= -\frac{1}{2} \log_2\left(\frac{1}{2}\right) - \frac{1}{2} \log_2\left(\frac{1}{2}\right) = \log_2(2) = 1$$

Below we give the design matrix X and response vector y for the training set of a binary classification problem with real-valued features:

$$X = \begin{bmatrix} 4 & 1 \\ 6 & 6 \\ 9 & 5 \\ 1 & 2 \\ 7 & 3 \\ 5 & 4 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

(2 points) For the data given above, what is the entropy at the root node of the tree before any splitting has occurred?

$$= -\frac{1}{2} \log_2\left(\frac{1}{2}\right) - \frac{1}{2} \log_2\left(\frac{1}{2}\right) = 1$$

[1 Point] True or False?

The distribution

$$P(A) = \frac{1}{2}, \quad P(B) = \frac{1}{2}$$

has higher entropy than the distribution

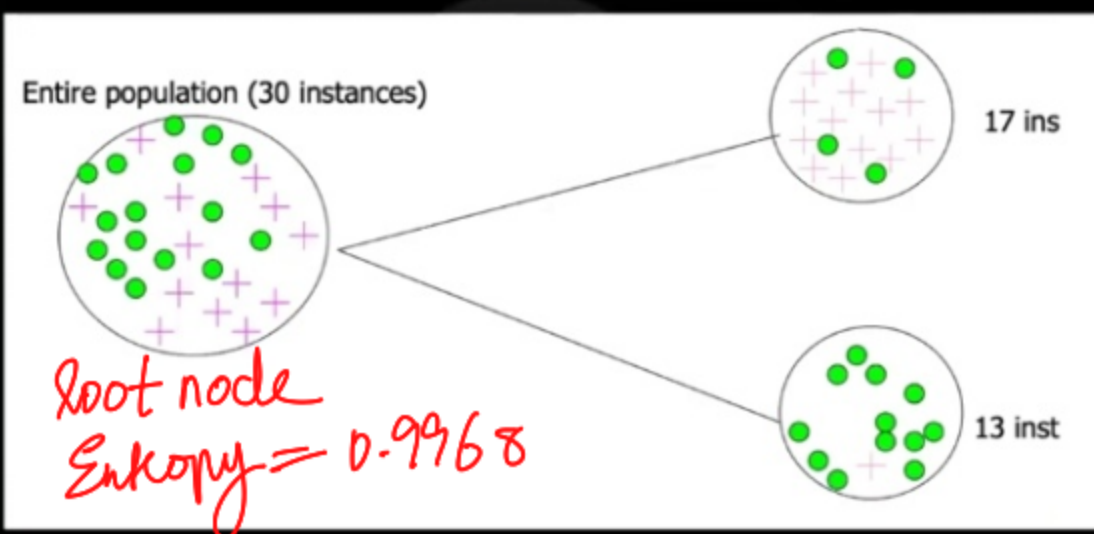
$$P(A) = \frac{1}{3}, \quad P(B) = \frac{1}{3}, \quad P(C) = \frac{1}{3}$$

$$-\frac{1}{2} \log_2\left(\frac{1}{2}\right) - \frac{1}{2} \log_2\left(\frac{1}{2}\right) = 1$$

$$-\frac{1}{3} \log_2\left(\frac{1}{3}\right) - \frac{1}{3} \log_2\left(\frac{1}{3}\right) - \frac{1}{3} \log_2\left(\frac{1}{3}\right) = \log_2(3)$$

→ But is the base two justified. The uncertainty is intuitive equal so should the entropy be equal?

Find out the weighted average of entropy of children.



$$-\frac{4}{17} \log_2 \left(\frac{4}{17} \right) - \frac{13}{17} \log_2 \left(\frac{13}{17} \right) \quad \mathcal{E}_a = 0.636410$$

$$= \frac{17}{30} (\mathcal{E}_a) + \frac{13}{30} (\mathcal{E}_b)$$

$$-\frac{12}{13} \log_2 \left(\frac{12}{13} \right) - \frac{1}{13} \log_2 \left(\frac{1}{13} \right) \quad \mathcal{E}_b = 0.39124$$

Weighted Average
Entropy / CONDITIONAL ENTROPY

$$= -\frac{4}{30} \log_2 \left(\frac{4}{30} \right) - \frac{13}{30} \log_2 \left(\frac{13}{30} \right) - \frac{12}{30} \log_2 \left(\frac{12}{30} \right) - \frac{1}{30} \log_2 \left(\frac{1}{30} \right) = 0.53017$$

Observation

→ After a particular split of data based on decision, the naive sum of child node entropies can increase (larger) than the parent.

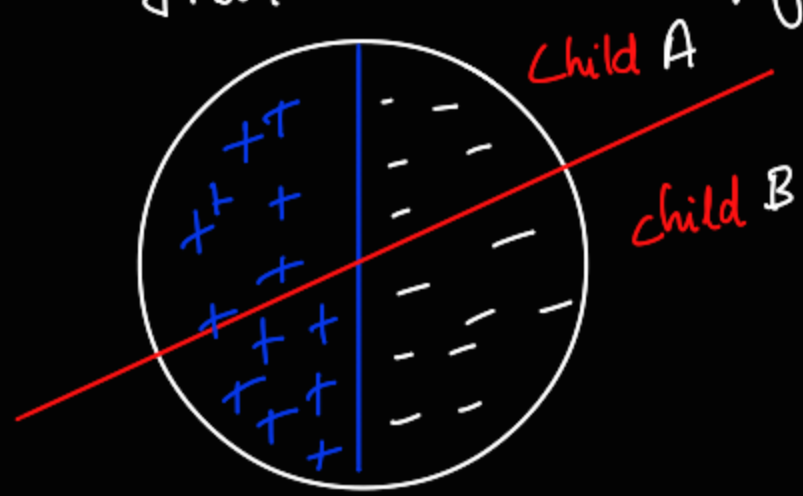
→ This is because, entropy doesn't care about the size of dataset but only about the distribution.

→ The maximum value of entropy a distribution can have, happens when the distribution is uniformly distributed across all its classes.

→ For "N" classes & base "b" the maximum entropy is when all classes C_i have equal probability.
ie) $P(C_i) = \frac{1}{N} \quad \forall i \text{ from } 1 \text{ to } N$

$$\Rightarrow \text{Entropy}_{\text{Max}} = -\sum_{i=1}^N \frac{1}{N} \log_b \left(\frac{1}{N} \right) = \log_b N$$

→ This is why weighted Average entropy is used since, it can never be higher than the entropy of the parent node (That is entropy without partition). Even for a bad split.



Original parent has P_1 instances of class 1
 P_2 instances of class 2

Child A has A_1 instances of class 1
 A_2 instances of class 2

Child B has B_1 instances of class 1
 B_2 instances of class 2

then $A_1 + B_1 = P_1$
 $A_2 + B_2 = P_2$

$$P(I=1)_{\text{parent node}} = \frac{P_1}{P_1 + P_2}$$

$$= \frac{A_1 + B_1}{A_1 + A_2 + B_1 + B_2}$$

$$= \frac{A_1}{A_1 + A_2 + B_1 + B_2} + \frac{B_1}{A_1 + A_2 + B_1 + B_2} = \frac{A_1}{A_1 + A_2} \cdot \frac{A_1 + A_2}{A_1 + A_2 + B_1 + B_2} + \frac{B_1}{B_1 + B_2} \cdot \frac{B_1 + B_2}{A_1 + A_2 + B_1 + B_2}$$

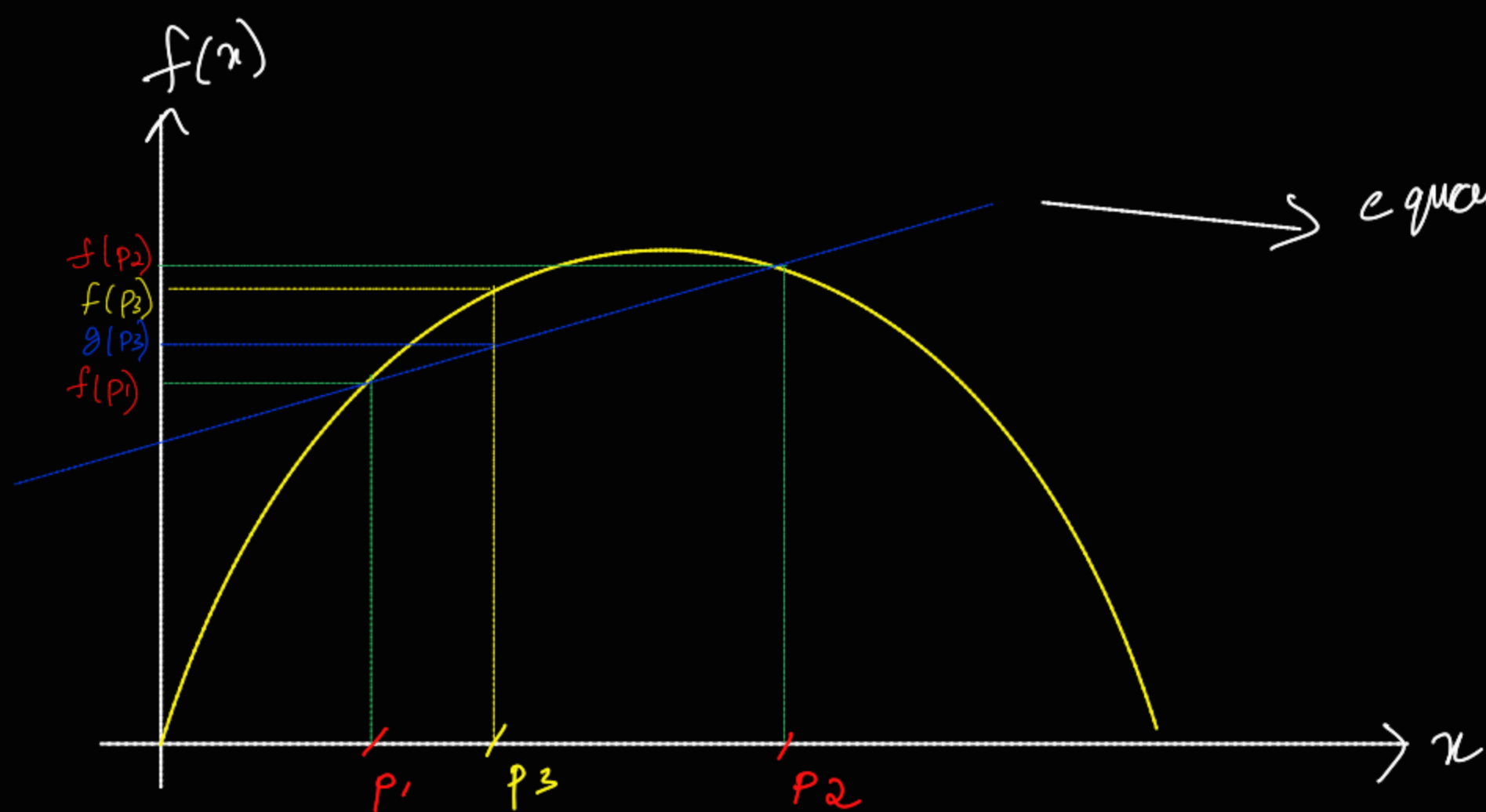
$$\Rightarrow P(I=1/\text{parent}) = P(I=1/\text{child A}) \cdot \frac{\text{Cardinality of child A}}{\text{Cardinality of Parent}} + P(I=1/\text{child B}) \cdot \frac{\text{Cardinality of child B}}{\text{Cardinality of parent}}$$

→ Probability distribution in parent = cardinality weighted distribution in children

→ Additionally entropy being a concave function the following always hold.

PROOF FOR CONCAVE FUNCTION being greater than its weighted average partition.

- The following is always true $kp_1 + mp_2 = p_3$ where $k+m=1$: $0 \leq k, m \leq 1$ for position of p_3
- i.e p_3 is the weighted average of p_1 & p_2
- Now imagine a concave function $f(x)$
- $f(p_3) = f(kp_1 + mp_2)$ compare this with $kf(p_1) + mf(p_2)$



$$\frac{g(x) - f(p_1)}{x - p_1} = \frac{f(p_2) - f(p_1)}{p_2 - p_1}$$

$$g(x) = (x - p_1) \cdot \frac{f(p_2) - f(p_1)}{p_2 - p_1} + f(p_1)$$

$$g(x) = f(p_2) \cdot \underbrace{\left(\frac{x - p_1}{p_2 - p_1}\right)}_B + f(p_1) \cdot \underbrace{\left(\frac{p_2 - x}{p_2 - p_1}\right)}_A$$

Let $A = k$
 $B = m$
 $k + m = 1$
 $0 \leq k, m \leq 1$

$$\Rightarrow \frac{p_2 - x}{p_2 - p_1} = k$$

$$x = p_2 - k(p_2 - p_1)$$

$$x = kp_1 + (1-k)p_2$$

$$\frac{x - p_1}{p_2 - p_1} = m \Rightarrow x = p_1 + m(p_2 - p_1)$$

$$x = p_1(1-m) + mp_2$$

$$\Rightarrow \boxed{x = kp_1 + mp_2 = p_3}$$

⇒ The equation of the line btw points p_1 & p_2 gives the weighted average

$$\text{i.e } g(kp_1 + mp_2) = kf(p_1) + mf(p_2)$$

⇒ Since $f(x)$ is concave, $f(x) \geq g(x)$ for any x btw p_1 & p_2

⇒ for Any concave function $f(x)$ $f(kp_1 + mp_2) \geq kf(p_1) + mf(p_2)$

$k+m=1$
 $0 \leq k, m \leq 1$ } k & m are weights.

Sidequest

$$kp_1 + (1-k)p_2 = p_1(1-m) + mp_2$$

$$p_1(1-(m+k)) = p_2(1-(m+k))$$

$$\text{but } p_1 \neq p_2$$

$$\Rightarrow 1-(m+k) = 0$$

$$\Rightarrow 1 = m+k$$

Entropy (cardinality weighted distribution of children) $\geq$ Cardinality weighted entropy (childrens)

Let Entropy of parent be $H(S)$
Let Entropy of child i be $H(S_i)$

$$H(S) \geq \sum_{i=1}^{\# \text{classes}} \alpha_i H(S_i)$$

$$\alpha_i = \frac{|S_i|}{|S|}$$

→ Splitting can only reduce the weighted entropy, it can't increase.

INFORMATION GAIN

Which attribute (Age, Salary, Marital Status) should be placed at the top of the decision tree?
Justify your choice using entropy.

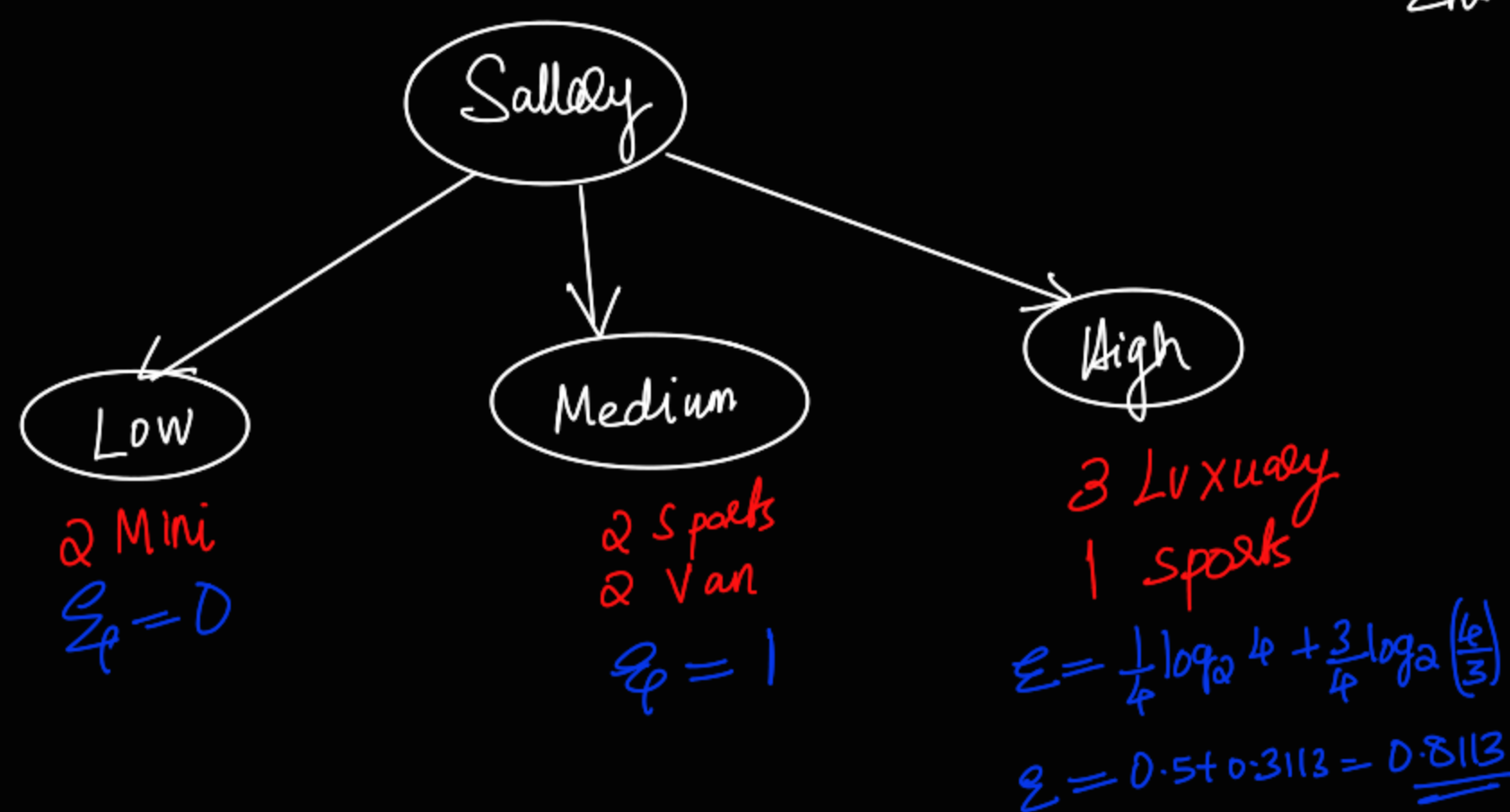
Car Type is the target attribute.

Age	Salary	Marital Status	Car Type
30s	Medium	Single	Sports
20s	Low	Single	Mini
40s	Medium	Married	Van
40s	High	Single	Luxury
60s	High	Married	Luxury
30s	High	Single	Sports
50s	Medium	Married	Van
50s	Medium	Single	Sports
60s	Low	Married	Mini
70s	High	Single	Luxury

Which should be the top root node in the decision tree.

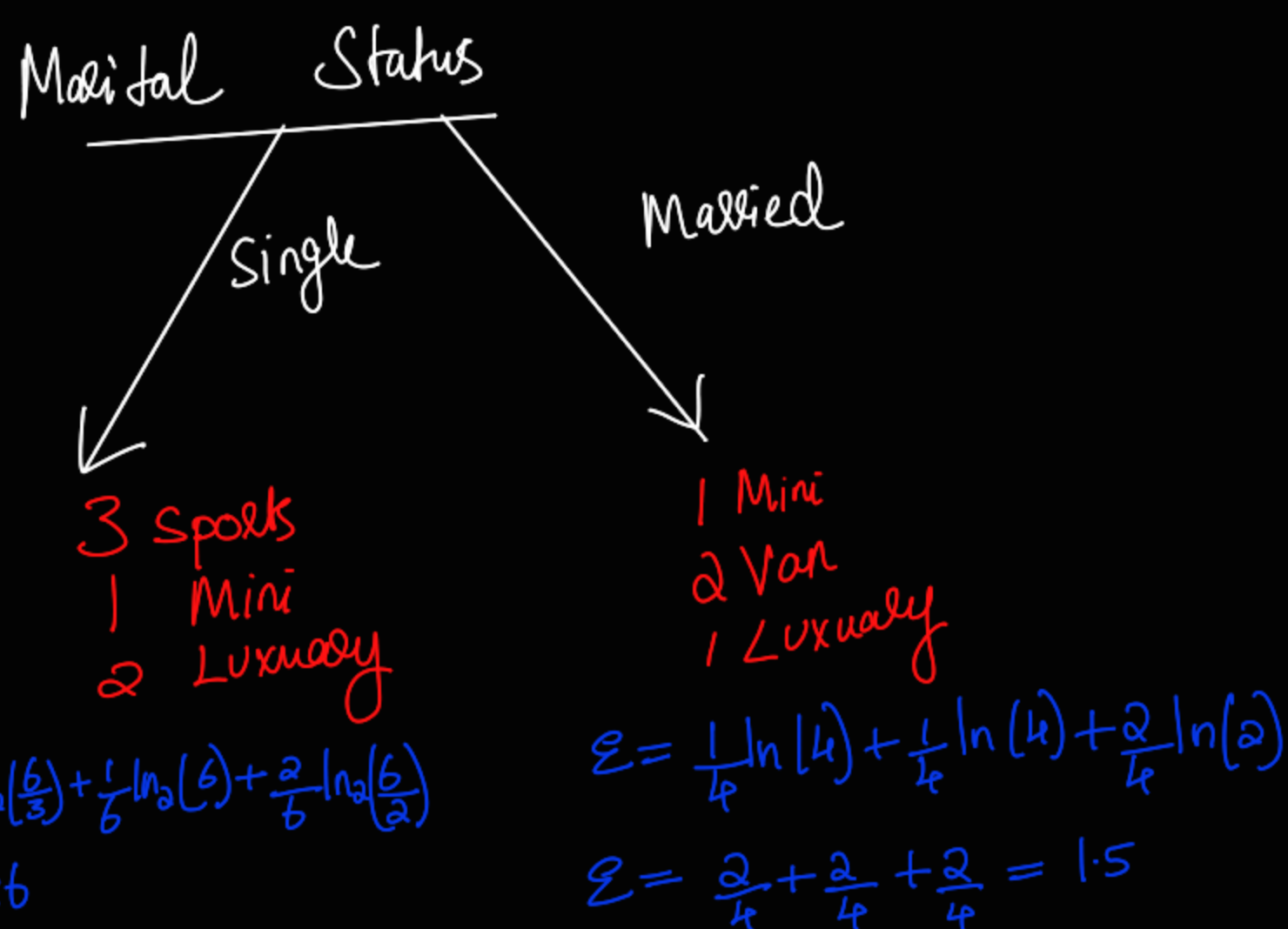
- ① Age
- ② Salary
- ③ Marital Status

$$\text{Weighted Entropy} = \frac{2}{10}(0) + \frac{4}{10}(1) + \frac{4}{10}(0.8113) = \underline{\underline{0.72452}}$$

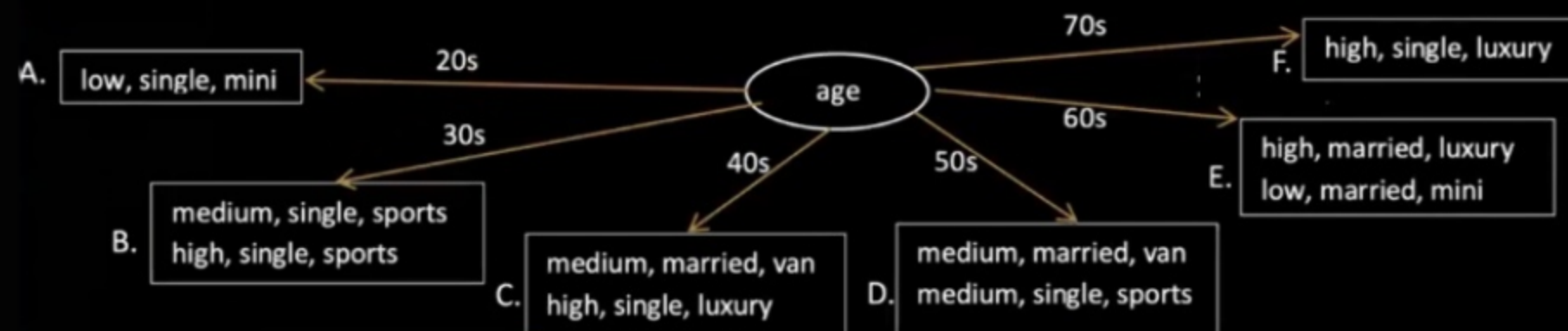


→ Here we have splitted into all separate classes as possible based on Salary attribute,

→ Since no further splits based on Salary is possible the minimum possible entropy possible by Salary is 0.72452



$$\text{Weighted Entropy} = \frac{6}{10}(1.46) + \frac{4}{10}(1.5) = \underline{\underline{1.476}}$$



Weighted entropy = 0.60

- A. Entropy(1,0)=0 D. Entropy(1/2,1/2)=1
B. Entropy(2/2,0)=0 E. Entropy(1/2,1/2)=1
C. Entropy(1/2,1/2)=1 F. Entropy(1,0)=0

Average entropy: $0.2*1 + 0.2*1 + 0.2*1 = 0.60$

→ Age is having minimum weighted entropy

INFORMATION GAIN given a classification or data split =

$$H(S) - \sum_i^{\text{all classes}} \alpha_i H(S_i) \quad \alpha_i = \frac{|S_i|}{|S|}$$

$H(x)$ → represent entropy function

$$H(x) = - \sum_{k=1}^{\text{all classes}} P_k^x \log_2(P_k^x)$$

where P_k^x → probability of class k @ node x

S - parent node
 S_i - i^{th} child node

Maximising Information gain $\Rightarrow$ Minimising entropy

Question: What is the conditional entropy of Y given X_1 , i.e., $H(Y | X_1)$?

X_1	0	0	1	1
X_2	0	1	0	1
Y	1	1	1	0

- A. 0
B. 0.25
C. 0.5
D. 0.75

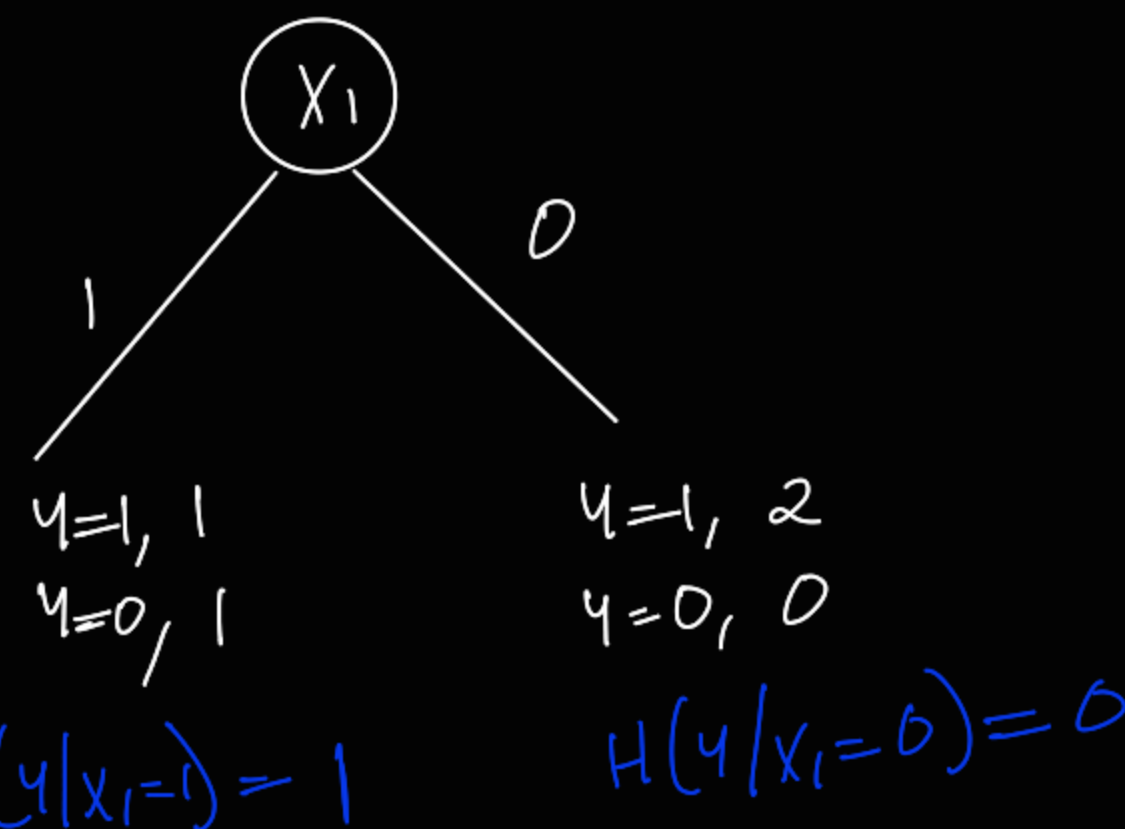
Use the following logarithm values:

- $\log_2 0.25 = -2$ ► $\log_2 1 = 0$
► $\log_2 0.5 = -1$ ► $\log_2 2 = 1$
► $\log_2 0.75 = -0.4$ ► $\log_2 4 = 2$

Convention: $0 \log_2 0 = 0$

Use this Formula:

$$H(Y | X_1) = P(X_1 = 0) \times H(Y | X_1 = 0) + P(X_1 = 1) \times H(Y | X_1 = 1)$$



weighted entropy / Conditional entropy = $\frac{2}{4}(1) + \frac{2}{4}(0) = \frac{1}{2}$

Problem 4 Consider the following set of training examples:

Instance	Classification	X_1	X_2
1	+	T	T
2	+	T	T
3	-	T	F
4	-	F	F
5	-	F	T
6	-	F	F

A. (3 points) What is the entropy of this collection of training examples with respect to the target function classification? Please write down necessary steps.

$$H(Y) = -\frac{2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6}$$

B. (3 points) What is the information gain of X_2 relative to these training examples? Please write down necessary steps.

$$H(Y) - H(Y|X_2)$$

$$H(Y|X_2=T) = \frac{2}{3} \log_2 \left(\frac{3}{2}\right) + \frac{1}{3} \log_2 \left(\frac{3}{2}\right)$$

$$H(Y|X_2=F) = 0$$

Weighted entropy } $= \frac{2}{6} \log_2 \left(\frac{3}{2}\right) + \frac{1}{6} \log_2 \left(\frac{3}{2}\right)$
Conditional " }

$$\begin{aligned} H(Y) - H(Y|X_2) &= \frac{2}{6} \log_2 \left(\frac{3}{2}\right) + \frac{4}{6} \log_2 \left(\frac{6}{4}\right) - \frac{2}{6} \log_2 \left(\frac{3}{2}\right) - \frac{1}{6} \log_2 3 \\ (\text{Information gain}) &= \frac{1}{6} \log_2 \left(\frac{3}{2}\right) + \frac{2}{6} \log_2 \left(\frac{3}{2}\right) \end{aligned}$$

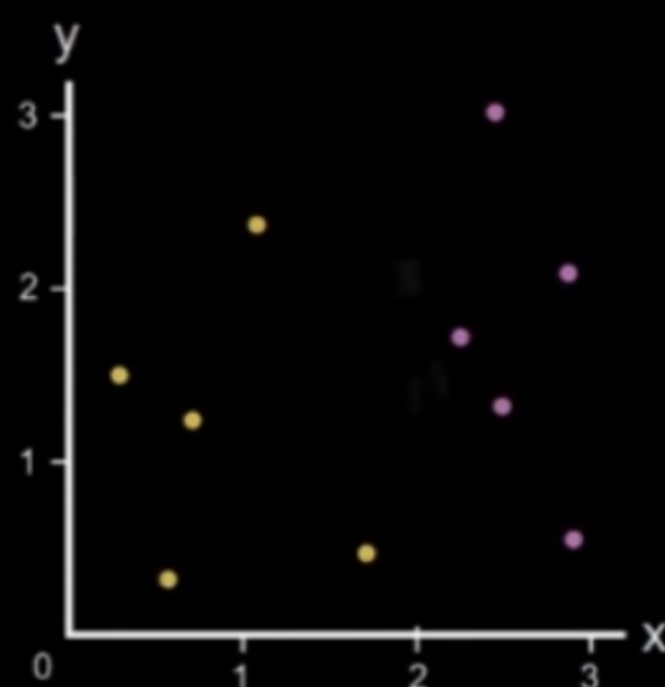
Gini Impurity

Suppose we:

Randomly pick a datapoint in our dataset, then **Randomly classify it according to the class distribution in the dataset.**

For our dataset, we'd classify it as blue 5/10 of the time and as green 5/10 of the time, since we have 5 datapoints of each color.

What's the probability we classify the datapoint incorrectly? The answer to that question is the Gini Impurity.



Dataset of size N . with p datapoints in +ve class
& q datapoint in -ve class
Random classifier with $P(Y=1/X) = p$
 $P(Y=-1/X) = q$ } $p+q=1$

Q) Given a random datapoint, what is the probability of incorrect classification
 $P(Y=1/X) \cdot P(X/Y=-1) + P(Y=-1/X) \cdot P(X/Y=+1) = pq + qp = 2pq$

eg:-

$$\begin{array}{c} + \\ + \\ + \end{array} \quad \begin{array}{c} - \\ - \\ - \end{array}$$

$$\frac{3}{9} = \frac{1}{3} \quad \frac{6}{9} = \frac{2}{3}$$

Gini impurity = $\frac{1}{3} \cdot \frac{2}{3} + \frac{2}{3} \cdot \frac{1}{3} = \frac{4}{9}$

$$= 2pq = 2 \cdot \frac{1}{3} \cdot \frac{2}{3} = \frac{4}{9}$$

In case of Multiple classes

→ Total N datapoints, & k classes $C_1, C_2, C_3, \dots, C_k$

→ $P(C_i) = p_i$ $\sum_{i=1}^k p_i = 1$

→ Random classifier with distribution same as data distribution

→ probability of missclassification = $\sum_{i=1}^k p_i (1-p_i) = \text{Gini Impurity} = 1 - \sum_{i=1}^k p_i^2$

GINI INDEX

eg. in case of 3 classes :- Gini Impurity = $p_1(1-p_1) + p_2(1-p_2) + p_3(1-p_3)$

→ Gini Impurity & Entropy are greedy strategies, since we are looking at only one node & looking at the best case scenario.

→ Optimisation involving a non optimum node in the current position, but still giving overall optimum solution are not considered.

A decision tree is trained on the following 2D dataset, where $(x_{i,1}, x_{i,2}, t_i)$ represents the features and the class label:

$(-2, 2, 0), (2, 1, 0), (7, -1, 0), (9, -7, 0), (0, -6, 1), (5, -2, 1), (8, -5, 1)$

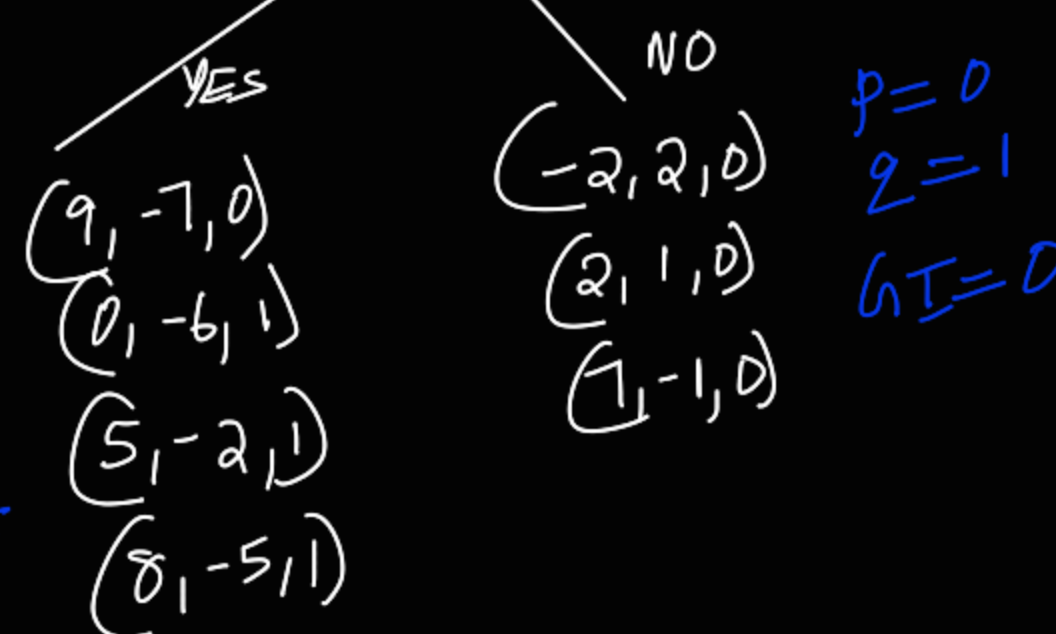
(a) Calculate the Gini index at the root node (without split).

(b) Calculate the weighted average Gini impurity after splitting the root node on $x_2 < -1.5$.

a) $P(1=0) = 4/7$
 $P(1=1) = 3/7$

Gini index = $2pq = 2 \cdot \frac{3}{7} \cdot \frac{4}{7} = \frac{24}{49}$

b) $x_2 < -1.5$



Weighted avg Gini Impurity

$$= \frac{4}{7} \left(\frac{3}{8} \right) + \frac{3}{7} \cdot 0$$

$$= \frac{12}{56} = \frac{3}{14}$$

Gini Gain = Parent gini gain - Weighted Gini Index

3. Decision trees. Consider the following dataset with input $x \in \mathbb{R}^3$ and $y \in \{0, 1\}$. Both x_i and y are binary random variables.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	1
0	1	0	1
1	1	1	0

You will use Gini index as the impurity measure to construct a decision tree.

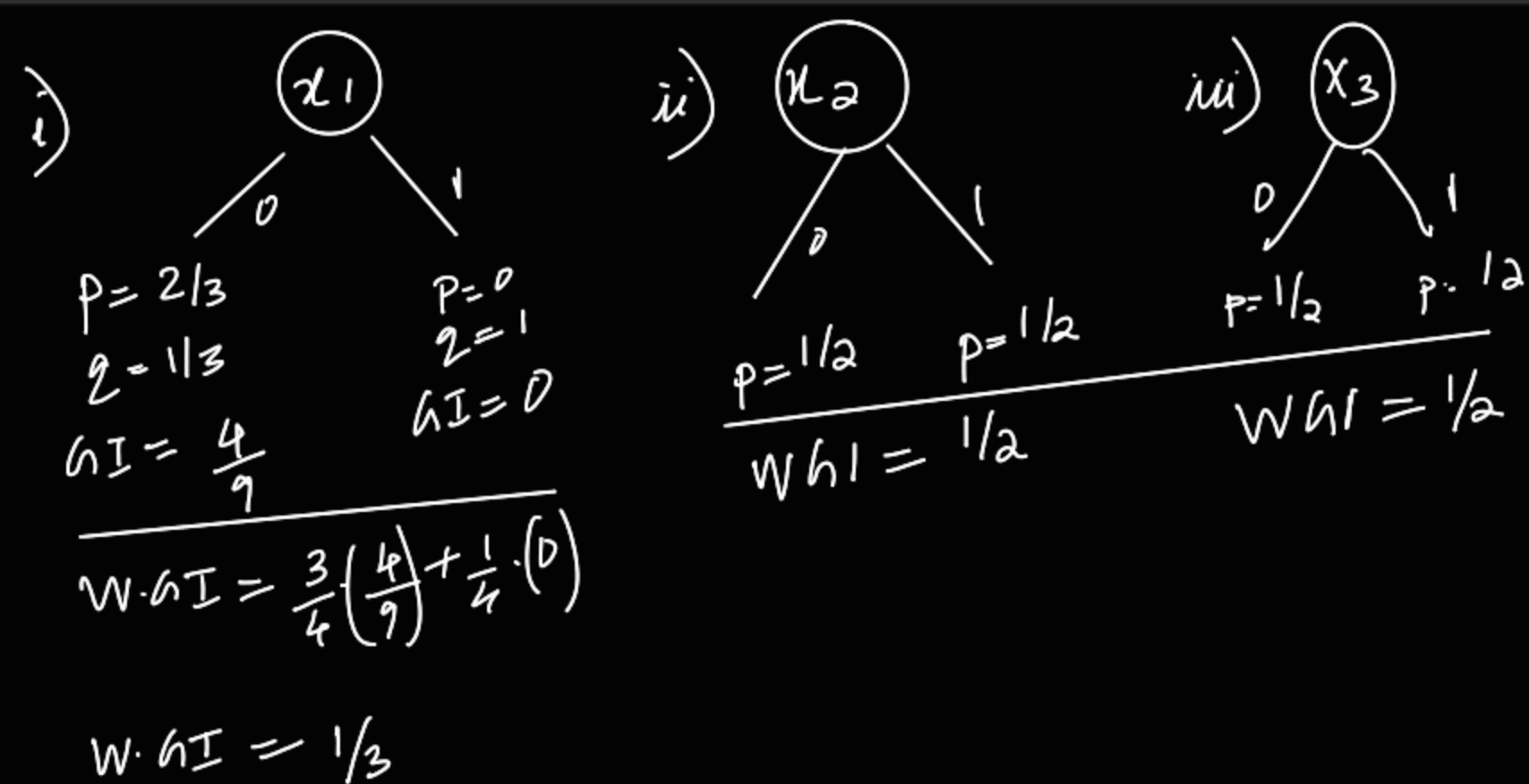
(a) Compute the weighted average of node impurities when splitting on each of the three features. Note that for each feature, since the value is binary, there is only one way to split. Your final answer should be a real number.

i. (2 points) Split on x_1

ii. (2 points) Split on x_2

iii. (2 points) Split on x_3

(b) (3 points) Draw the full decision tree and write the prediction for each leaf node. You can split ties arbitrarily.



→ It can definitely happen that using entropy & Gini impurity greedy learning can give different decision trees.

MISCLASSIFICATION ERROR & MAJORITY CLASSIFIER

→ Consider a classifier that always gives same classification which was the highest in the training dataset

$$p(y=C_i/x_i) = \begin{cases} 0, & \text{if } p(C_i) \neq \max(p_1, p_2, \dots, p_K) \\ 1, & \text{if } p(C_i) = \max(p_1, p_2, \dots, p_K) \end{cases}$$

→ Let $\max(p_1, p_2, \dots, p_K) = p^{\max}$ & the corresponding class be $C^{\max}$ } $\Rightarrow p(y=C_i/x_i) = \begin{cases} 0, & C_i \neq C^{\max} \\ 1, & C_i = C^{\max} \end{cases}$

majority classifier

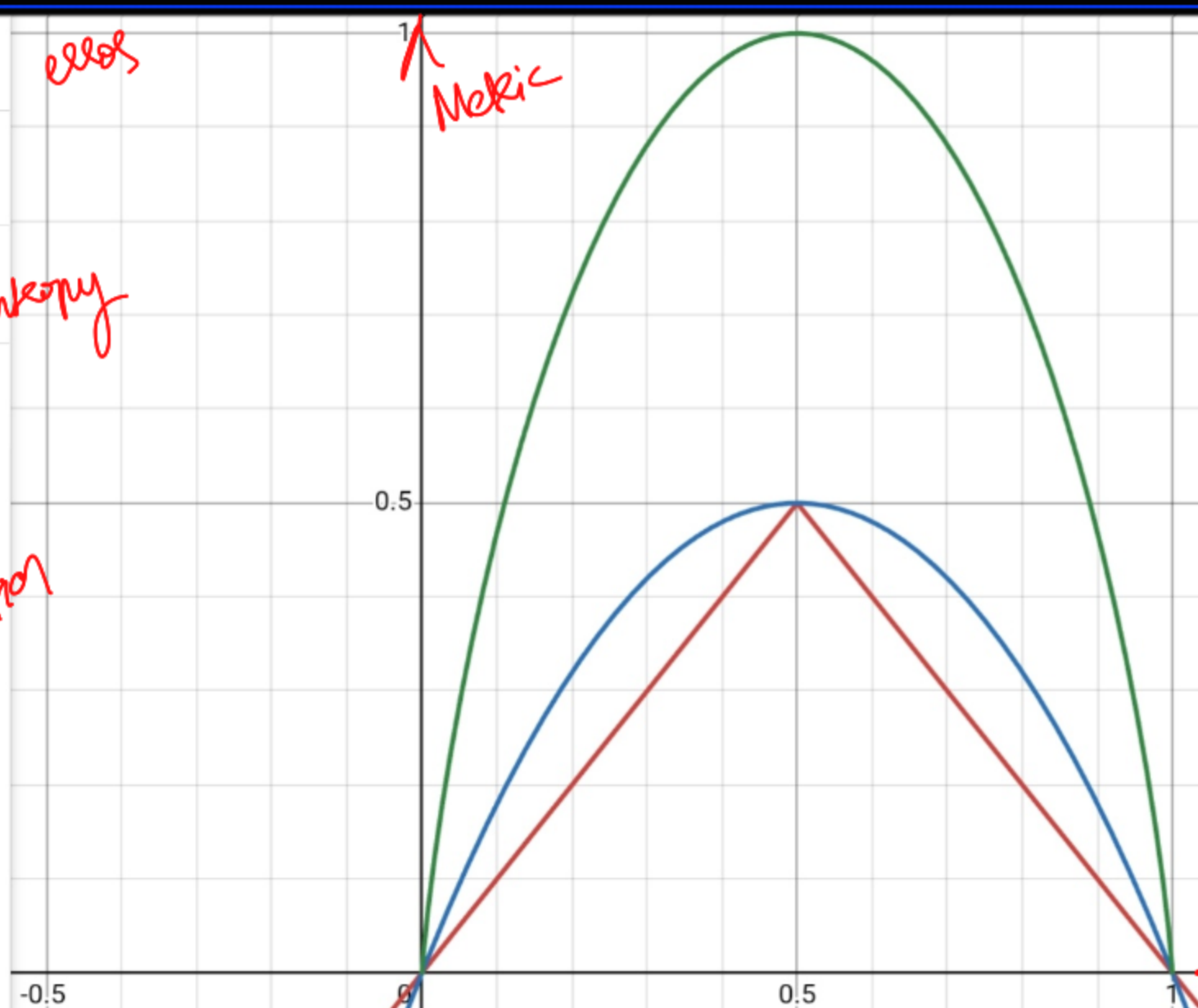
→ Probability of misclassification = MISCLASSIFICATION ERROR

$$= 1 - \max_{i=1}^K (p_i) = \min_{i=1}^K (1 - p_i)$$

→ This metric is not used much.

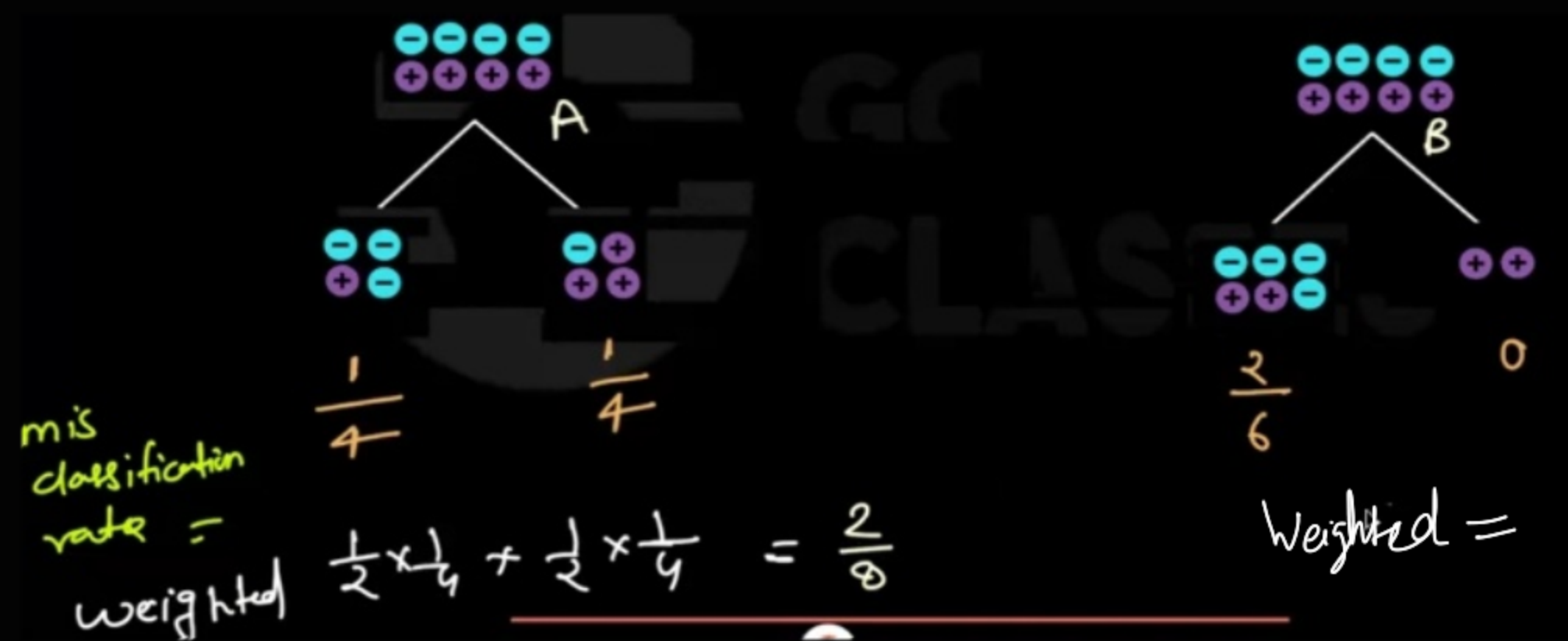
- 1 - max(x, 1-x) → Misclassification error
- 2x(1-x) → Gini Impurity
- x log₂(x) - (1-x) log₂(1-x) → Entropy

Binary classifiers / metric vs data distribution



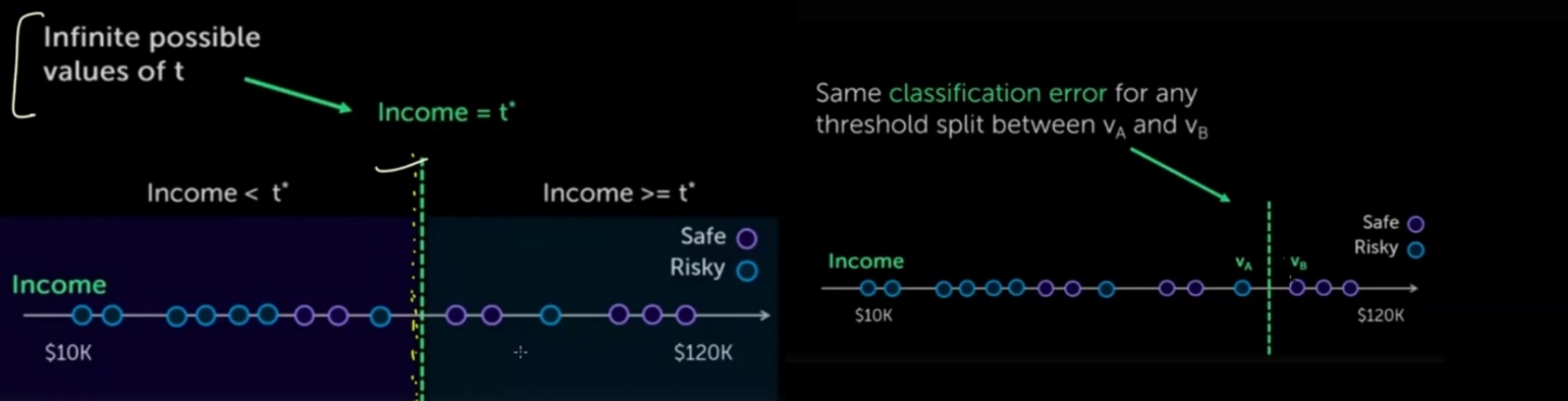
→ $p(y=0)$ data distribution

Why Misclassification Error is not a good metric



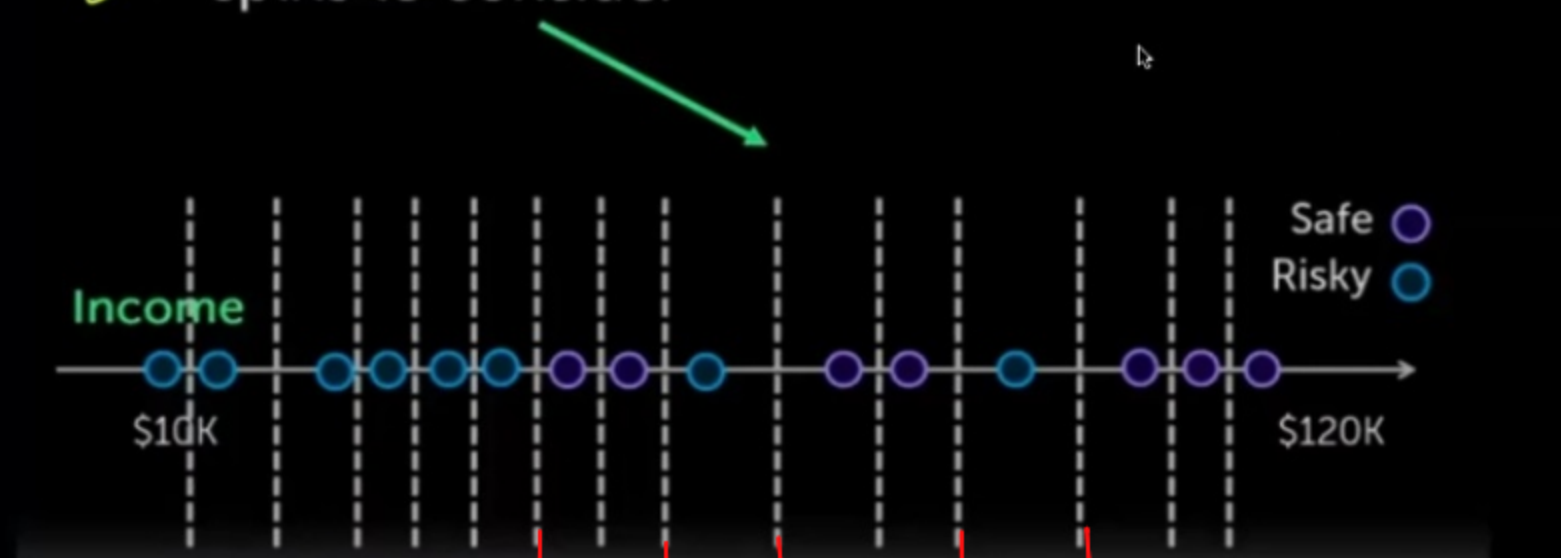
→ No Real Conclusion
→ Misclassification rate is not able to distinguish if no. of points misclassified in splits are equal with majority classifies.

DECISION TREE WITH CONTINUOUS VALUED FEATURES



Only need to consider mid-points

Finite number of splits to consider



if the nearest classes near a split are same class then it is not a good split

Good splits to consider

Candidate cuts which separate two classes

eg:-

What are the possible values of threshold of Temp to split ?

Temp	40	48	60	72	80	90
Tennis?	N	N	Y	Y	Y	N

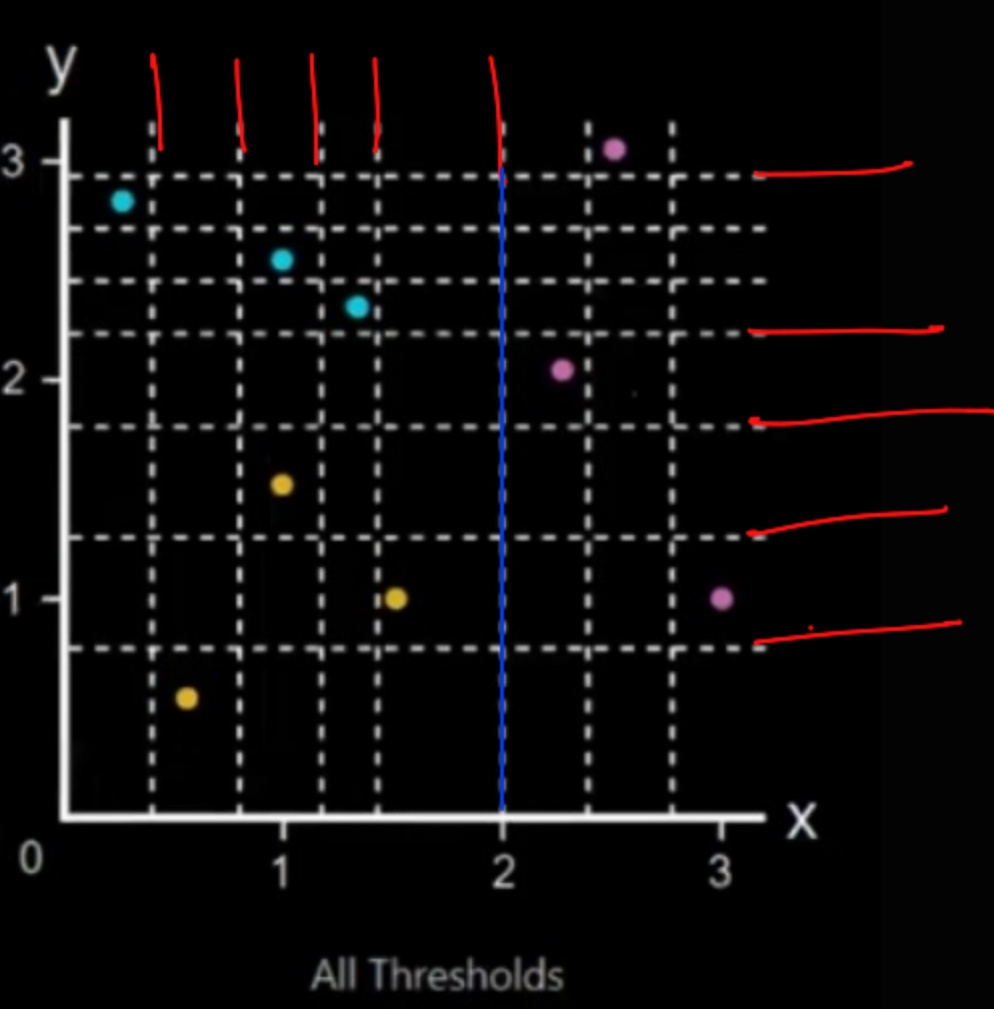
Temp < 54

Temp < 85

Further decide based on Metric
ie, entropy, gini impurity.

eg:-

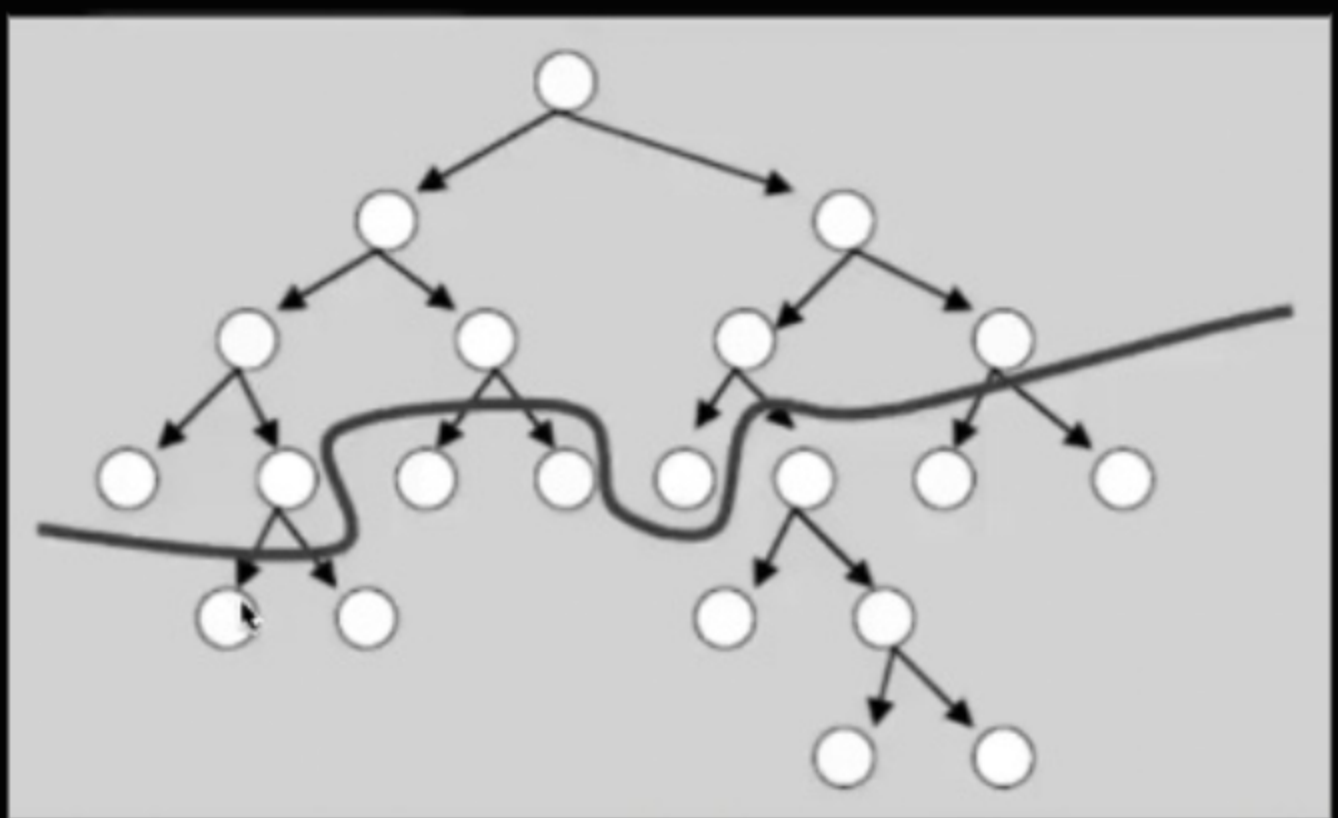
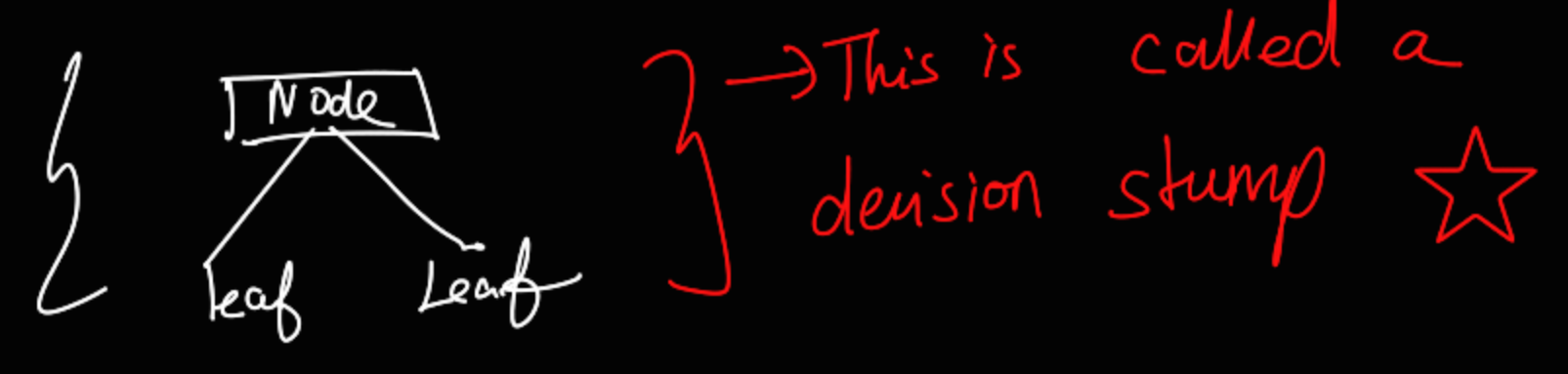
Split	Left Branch	Right Branch	Gini Gain
$x = 0.4$			0.083
$x = 0.8$			0.048
$x = 1.1$			0.133
$x = 1.3$			0.233
$x = 2$			0.333
$x = 2.4$			0.191
$x = 2.8$			0.083
$y = 0.8$			0.083
$y = 1.2$			0.111
$y = 1.8$			0.233
$y = 2.1$			0.233
$y = 2.4$			0.111
$y = 2.7$			0.048
$y = 2.9$			0.083



Maximise gini-gain
Minimise gini-impurity

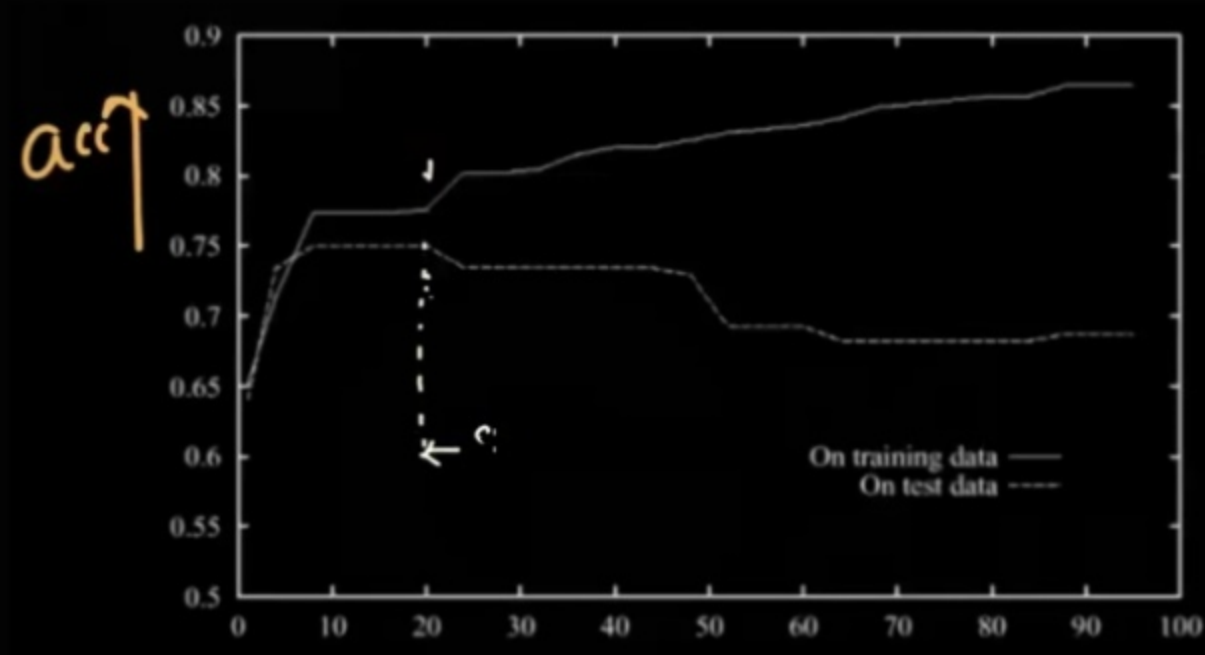
Tree pruning

- Decision tree - complexity increases with depth, #nodes → Larger trees
- Larger trees likely overfit
- Simplest decision tree has only 1 depth
- We can add a complexity penalty in the loss function $L + \alpha |T|$ where $|T|$ is #nodes



Tree pruning

- #Nodes can increase until every leaf is pure
- But this is likely overfitting
- pruning is the stopping criteria



→ No. of tree nodes

Method 1: Pre pruning

- Stop splitting the nodes if the number of samples is too small to make reliable decisions.
- Stop if the proportion of samples from a single class (node purity) is larger than a given threshold (Forward Pruning).

Method 2: Post-pruning

- Allow the tree to overfit and then prune it afterward.
- Use estimation of errors and tree size to decide which subtree should be pruned.

Limitation

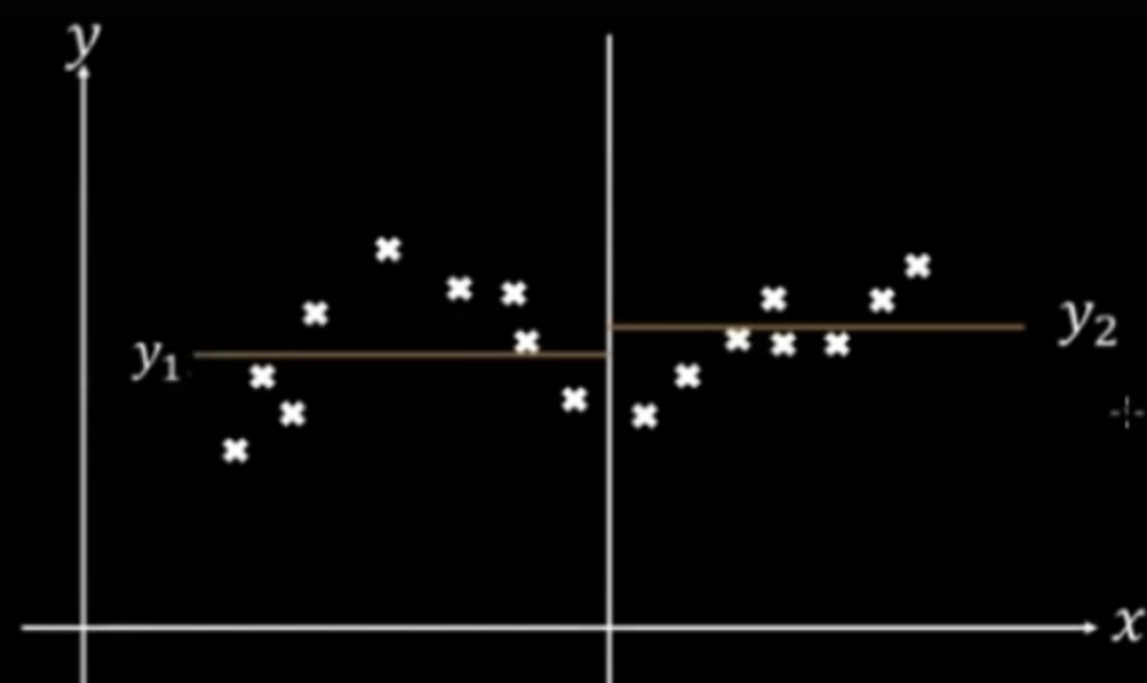


- Decision tree boundary is piecewise axis aligned
- There may be data distribution where SVM is a better choice

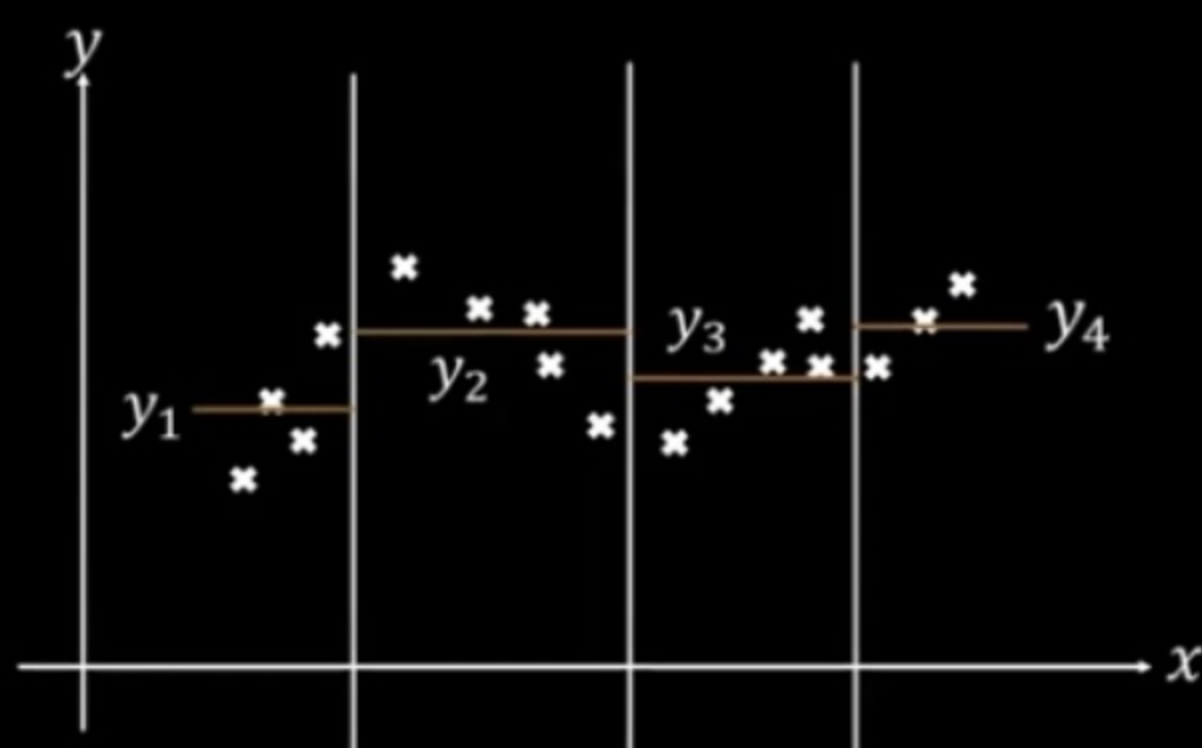
Regression with Decision Tree

- How do we use decision trees for regression?
- Partition the input into intervals

→ Feature split also



→ y_1 & y_2 are averages of their respective partitions



→ Information gain & Gini gain won't work as learning criteria. Since they are based on probability of classification

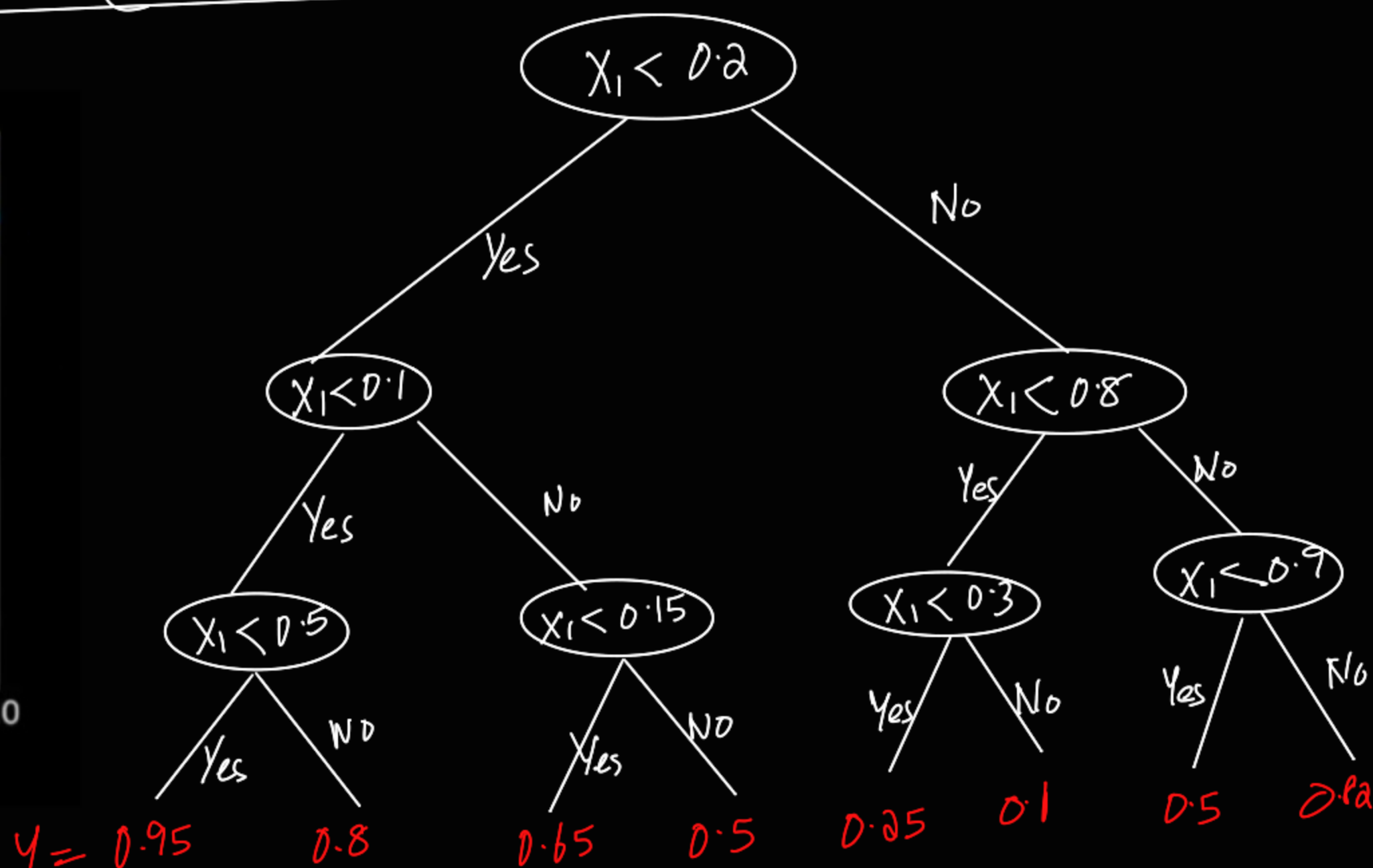
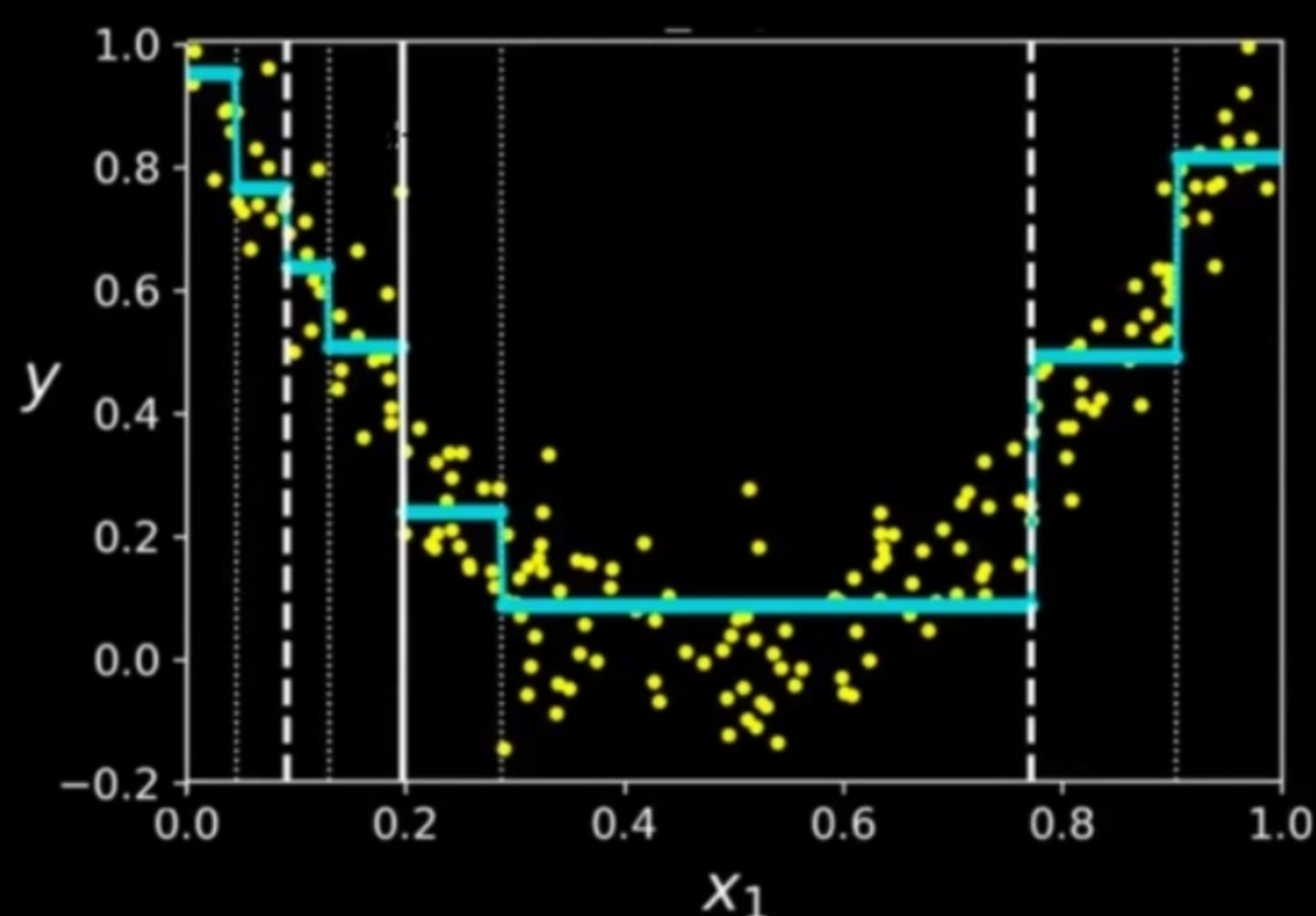
if we take MSE as criteria then calculating MSE of each child partition & then taking weighted average will result in computing MSE for SSE.

$$MSE_{weighted} = \sum \frac{|S_i|}{|S|} \cdot MSE(S_i)$$

$\frac{1}{|S|} = \sum \frac{1}{|S|}$ classes

possible criteria = MSE (Mean Squared error) / SSE (Sum of Squared error)

eg:-



Minimise $\sum_{i=1}^{\text{all regions}} \frac{|R_i|}{|R|} \cdot \frac{1}{|R_i|} \sum_{j \in R_i} (y_j - \bar{y}_i)^2$

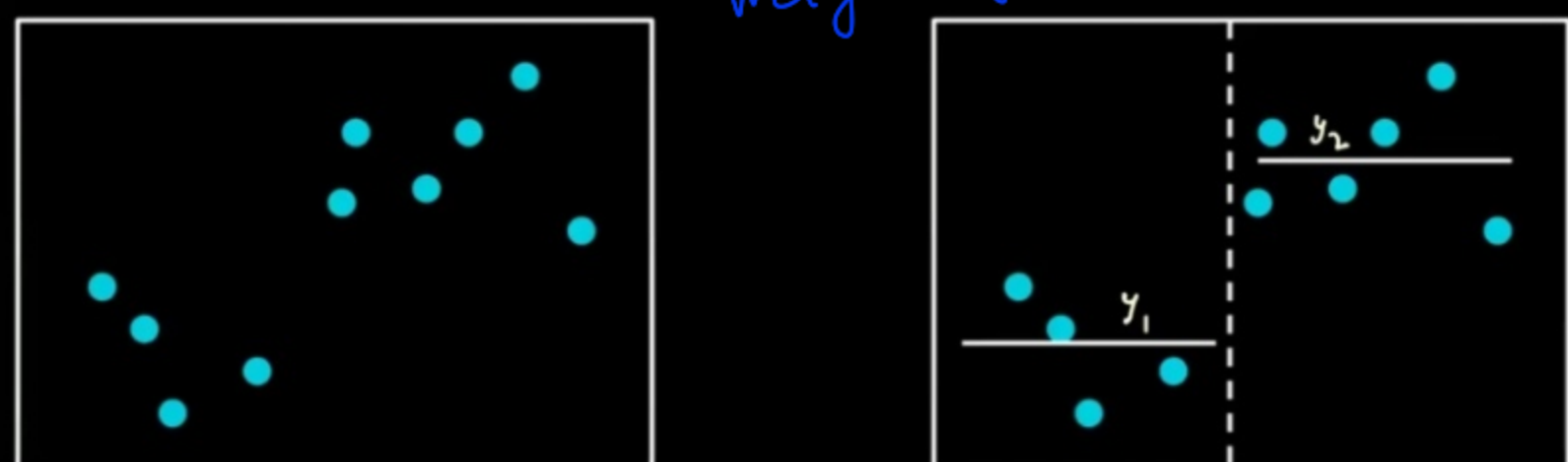
weighting

value of split in region i

$|R_i|$ → cardinality of split in region i
 R → cardinality of parent

→ where $\bar{y}_i$ is the mean in region R_i

→ This is equivalent to taking the squared distance from mean



$$\sum_{i \in R_1} (y_i - \bar{y}_1)^2 + \sum_{j \in R_2} (y_j - \bar{y}_2)^2 = \sum_i \sum_{j \in R_i} (y_i - \bar{y}_i)^2 \rightarrow \Rightarrow \text{Minimise Variance of split}$$

→ Minimising weighted variance of split is equivalent to maximising information gain

$$\text{Variance Reduction} = \text{Parent Variance} - \text{Weighted Variance}$$

In decision trees when used for regression, what is the criteria we use, reducing mean square error or reducing sum of squared errors

Excellent question 🍌

In **regression trees**, the splitting criterion is based on **reducing the variance of the target values** — which can be expressed equivalently in two ways:

- **Sum of Squared Errors (SSE)**
- **Mean Squared Error (MSE)**

Let's break it down clearly 🙌

→ We are minimising variance of each child
→ But because of weighting by size of child → more importance for reducing variance is given to larger child
→ if we don't do this → all children treated equally independent of size
→ Causes algorithm to be biased towards smaller nodes.
→ Weighted MSE of each child = MSE of parent - Variance Reduction
→ $\frac{\text{SSE of child}}{|\text{parent}|} = \text{MSE of parent} - \text{Variance Reduction}$

◆ Criterion used

At each split, the algorithm chooses the feature and threshold that **minimizes the impurity**, measured as:

$$\text{Impurity}(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y}_S)^2$$

This is the **Mean Squared Error (MSE)** within the node.

When computing the **total impurity** before and after a split, the algorithm multiplies each child node's MSE by its size — effectively using **Sum of Squared Errors (SSE)** to compare splits.

◆ In practice (what the algorithm does)

In implementations like **CART (Classification and Regression Trees)**:

$$\text{Split criterion} = \text{SSE}_{\text{parent}} - (\text{SSE}_{\text{left}} + \text{SSE}_{\text{right}})$$

where

$$\text{SSE}_{\text{node}} = \sum_{i \in \text{node}} (y_i - \bar{y}_{\text{node}})^2$$

So although you may **conceptually think** of minimizing MSE (variance), the **actual computation uses SSE** because it avoids dividing by the sample count (which cancels out when comparing splits anyway).

✅ Summary:

- Conceptually: decision trees for regression aim to **minimize MSE (variance)**.
- Computationally: they **compare splits using reduction in SSE**, since it's proportional to MSE and simpler to compute.

So, both are correct — but **SSE is the actual quantity minimized**.

Suppose instead of minimising SSE, the algorithm minimised the sum of MSE. What would be the effect

Excellent — this is a subtle but *very insightful* question. Let's unpack it carefully. 📌

◆ Recall what happens normally:

In standard regression trees (like CART):

We minimize **the total Sum of Squared Errors (SSE)** across the child nodes after a split:

$$\text{Criterion} = \text{SSE}_{\text{left}} + \text{SSE}_{\text{right}}$$

where

$$\text{SSE}_{\text{node}} = \sum_{i \in \text{node}} (y_i - \bar{y}_{\text{node}})^2$$

This means **larger nodes** (with more samples) naturally have more influence on the split decision — as they should, since they represent more of the data.

◆ Now suppose we minimize sum of MSE instead:

That is,

$$\text{Criterion} = \text{MSE}_{\text{left}} + \text{MSE}_{\text{right}}$$

where

$$\text{MSE}_{\text{node}} = \frac{1}{n_{\text{node}}} \text{SSE}_{\text{node}}$$

◆ What changes conceptually?

This version **ignores the size of the child nodes** — it treats both left and right nodes *equally*, no matter how many samples each one contains.

◆ Example

Imagine a split that produces:

- Left node: 100 samples, MSE = 4
- Right node: 2 samples, MSE = 0.1
- If we minimize **sum of SSE**:

$$100 \times 4 + 2 \times 0.1 = 400.2$$

- If we minimize **sum of MSE**:

$$4 + 0.1 = 4.1$$

So under the MSE-sum rule, the algorithm might **overvalue** the small node's tiny error, even though it barely affects total performance.

So under the MSE-sum rule, the algorithm might **overvalue** the small node’s tiny error, even though it barely affects total performance.

◆ **Effect on the resulting tree**

If you minimize the **sum of MSEs (unweighted)** instead of the **sum of SSEs (weighted)**:

- 1. ⚠️ **Splits become biased toward tiny nodes**
The algorithm will favor splits that produce small “pure” nodes even if they contain very few samples.
- 2. 🌳 **Tree becomes overly deep and fragmented**
Because small nodes can have small MSEs, it keeps splitting to isolate tiny subsets.
- 3. 🎯 **Worse generalization / more overfitting**
The model fits noise instead of real structure.

✅ **Summary**

Criterion	Weighting by Node Size?	Behavior	Result
Sum of SSE	✅ Yes	Balanced — large nodes have more influence	Standard CART behavior
Sum of MSE	❌ No	Overemphasizes small nodes	Overfitting, unstable splits

In short:

Minimizing the *sum of MSEs* instead of *sum of SSEs* would cause the tree to favor splits that create very small, low-error leaves — leading to **overfitting and instability**.

📄 🍌 💬 ↗️ ↺ ⋮

Is this related to K means and K means loss. In Kmeans. The algorithm aims to minimise the sum of squared distance rather than Mean of squared distance in each cluster. Is the same reason as above applicable as to why we don't minimise mean of squared distance.

100 Yes — you’ve spotted a deep and absolutely correct connection.

The reason K-Means minimizes the **sum of squared distances (SSD or SSE)** rather than the **mean of squared distances (MSE)** is **exactly the same logic** as in regression trees.

Let’s unpack that carefully. 🙋

◆ K-Means objective

K-Means minimizes:

$$J = \sum_{k=1}^K \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

This is the **sum of squared distances** of all points to their cluster centroids (SSE).

◆ Why not minimize the mean per cluster?

You could imagine an alternative:

$$J' = \sum_{k=1}^K \frac{1}{|C_k|} \sum_{x_i \in C_k} \|x_i - \mu_k\|^2$$

That's the **sum of mean squared distances** across clusters.

◆ What's the difference?

Just like in regression trees:

- The standard K-Means objective (SSE) **weights each cluster by its size** – clusters with more points have proportionally more influence on the total error.
- The alternative (sum of MSEs) would **treat all clusters equally**, regardless of how many points each cluster contains.

◆ What would happen if we minimized the sum of MSEs instead?

1. Bias toward small clusters:

The algorithm could make one very small cluster with a perfect centroid (near-zero error), since that small cluster counts equally to a large one.

2. Unstable / imbalanced clusters:

You might end up with many tiny, very pure clusters and a few huge, impure ones – exactly the same “over-splitting” problem seen in regression trees when minimizing sum of MSEs.

3. Loses probabilistic interpretation:

The K-Means objective is equivalent to maximum likelihood under a spherical Gaussian mixture with equal variances – this correspondence breaks if you divide by cluster size.

✔ Summary

Algorithm	Standard Criterion	Weighted by Cluster/Node Size?	If we use mean instead	Effect
Regression Tree	Sum of Squared Errors (SSE)	✔ Yes	Sum of MSEs	Overfitting small nodes
K-Means	Sum of Squared Distances (SSE)	✔ Yes	Sum of Mean Squared Distances	Bias toward small clusters

Question:

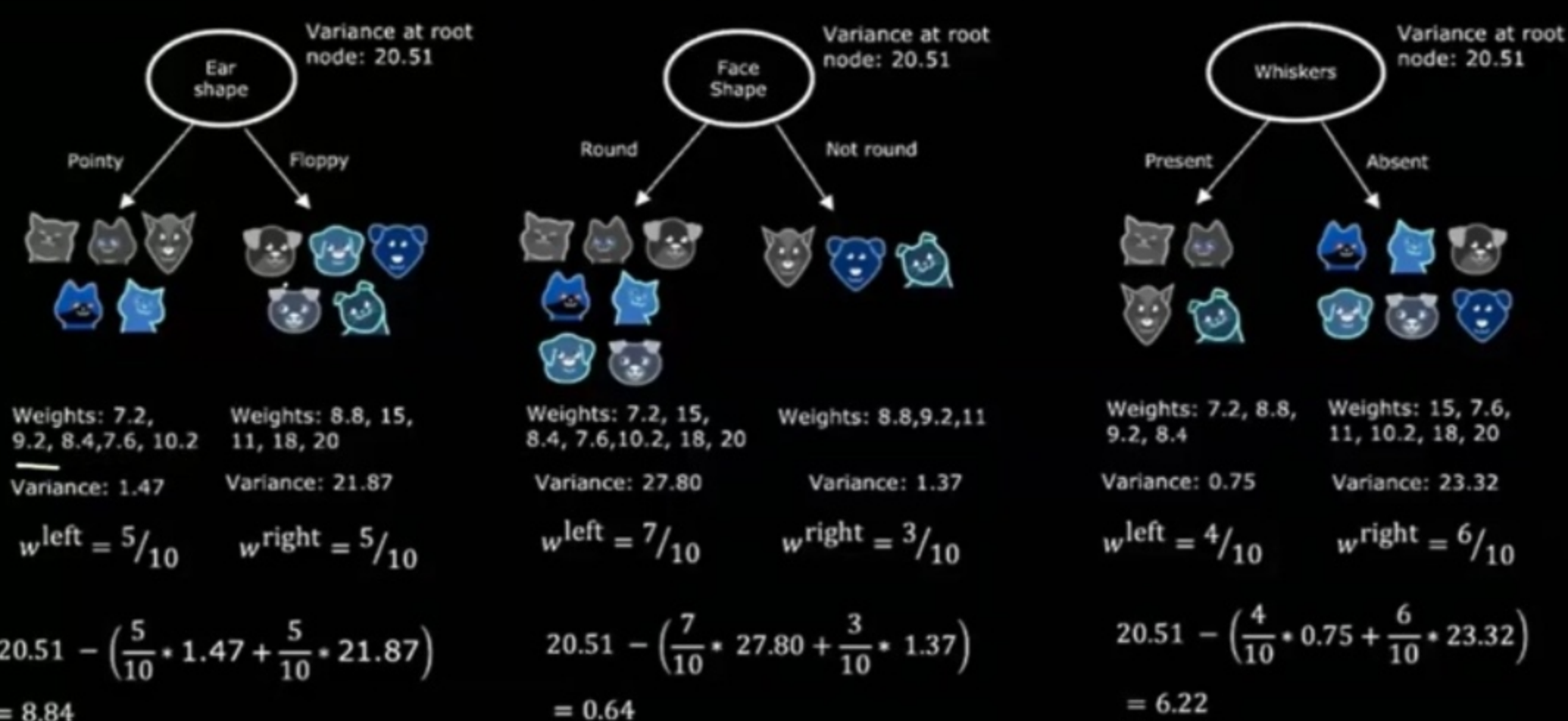
Consider this dataset and find out best attribute to split on at the root node based on variance reduction method.

X - data

	Ear Shape	Face Shape	Whiskers	Weight
1	Pointy	Round	Present	7.2
2	Floppy	Not Round	Present	8.8
3	Pointy	Not Round	Present	9.2
4	Pointy	Round	Present	8.4
5	Pointy	Round	Absent	7.6
6	Pointy	Round	Absent	10.2
7	Floppy	Round	Absent	15.0
8	Floppy	Round	Absent	11.0
9	Floppy	Round	Absent	18.0
10	Floppy	Round	Absent	20.0

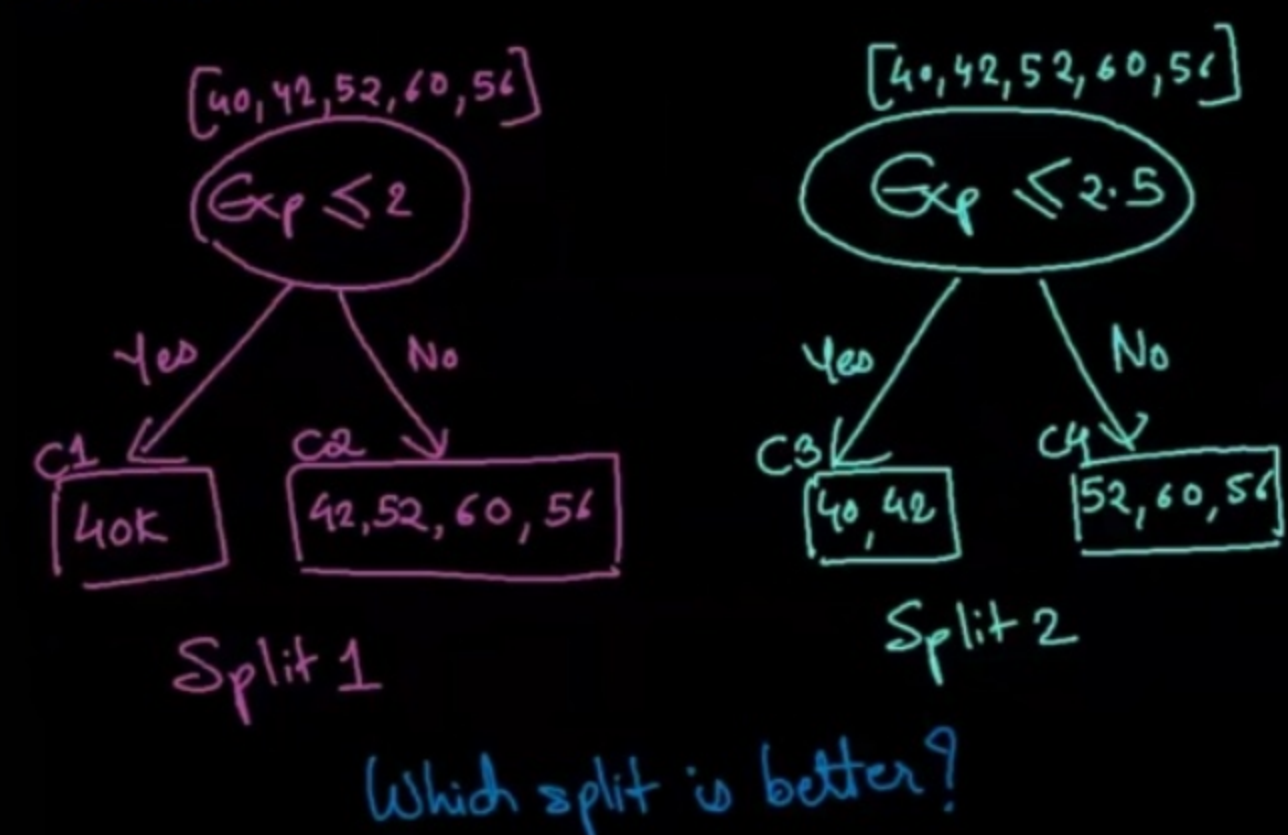
Dependent Variable (Y)

check H.W



Best criteria

Question:



DATASET

Exp	Gap	Salary ₹
2	Yes	40K
2.5	Yes	42K
3	No	52K
4	No	60K
4.5	Yes	56K

check H.W

$$\text{Variance of root node} = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$$

$$= \frac{1}{5} ((40-50)^2 + (42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2)$$

$$= 60.8$$

$$\text{Variance of } C1 = \frac{1}{1} ((40-50)^2) = 100$$

$$\text{Variance of } C2 = \frac{1}{4} ((42-50)^2 + (52-50)^2 + (60-50)^2 + (56-50)^2) = 51$$

$$\text{Variance of } C3 = \frac{1}{2} ((40-50)^2 + (42-50)^2) = 82$$

$$\text{Variance of } C4 = \frac{1}{3} ((52-50)^2 + (60-50)^2 + (56-50)^2) = 46.67$$

$$\text{Variance Reduction for split 1} = 60.8 - \left[\frac{1}{5} \times 100 + \frac{4}{5} \times 51\right]$$

$$= 0$$

$$\text{Variance Reduction for split 2} = 60.8 - \left[\frac{2}{5} \times 82 + \frac{3}{5} \times 46.67\right]$$

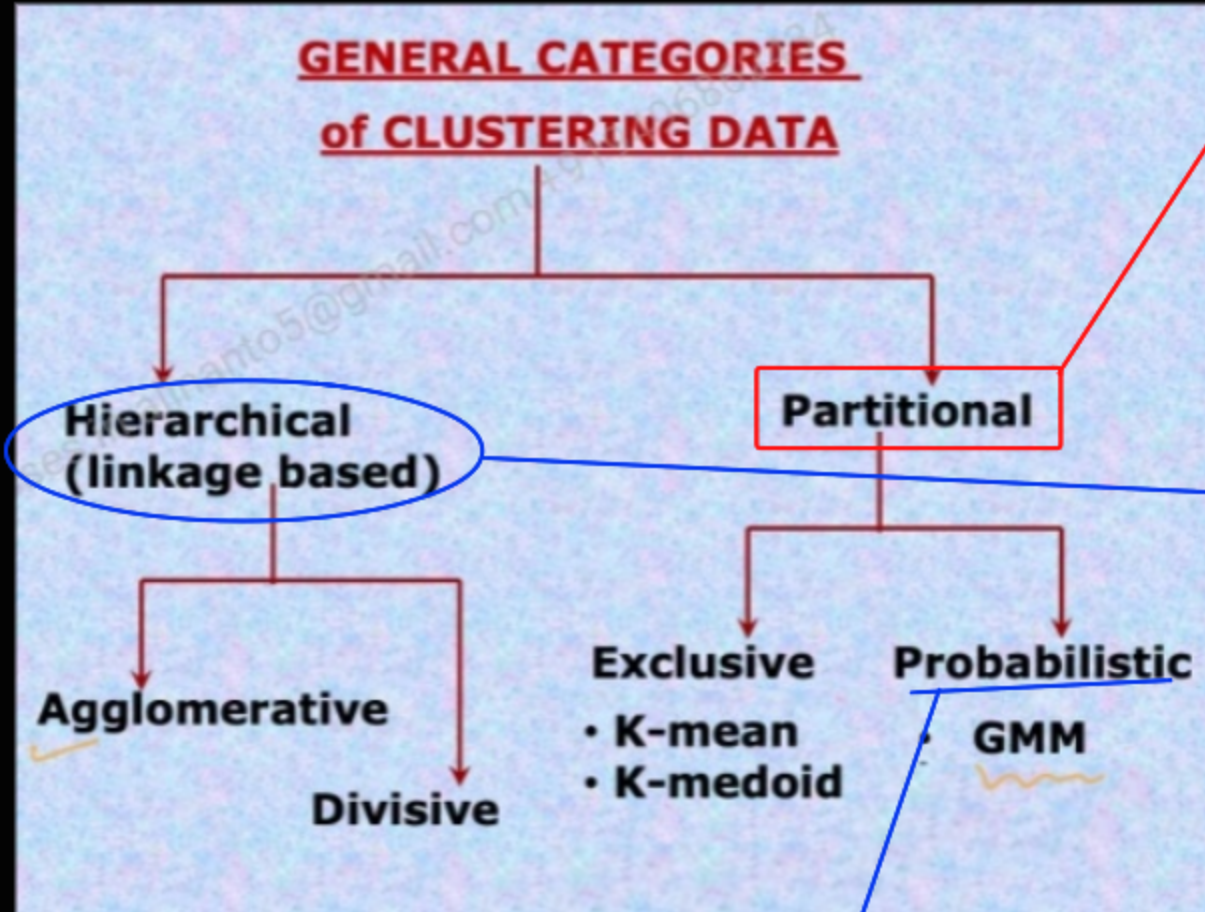
$$= 0.304$$

$exp \leq 2.5$ is better
 than $exp \leq 2$

Clustering

→ group similar datapoints together

Machine Learning: (i) Supervised Learning: regression and classification problems, simple linear regression, multiple linear regression, ridge regression, logistic regression, k-nearest neighbour, naive Bayes classifier, linear discriminant analysis, support vector machine, decision trees, bias-variance trade-off, cross-validation methods such as leave-one-out (LOO) cross-validation, k-folds cross-validation, multi-layer perceptron, feed-forward neural network; (ii) Unsupervised Learning: clustering algorithms, k-means/k-medoid, hierarchical clustering, top-down, bottom-up: single-linkage, multiple-linkage, dimensionality reduction, principal component analysis.



1. Partition-based Clustering

- **Idea:** Divide the dataset into a fixed number of non-overlapping clusters.
- **Assumption:** Each point belongs to exactly one cluster.
- **Examples:**
 - **K-Means:** Minimizes within-cluster variance (distance from centroid).
 - **K-Medoids (PAM):** Uses actual data points as cluster centers (more robust to outliers).
 - **CLARANS:** Efficient version of K-Medoids for large datasets.
- **Best for:** Spherical, well-separated clusters.

2. Hierarchical Clustering

- **Idea:** Build a hierarchy (tree/dendrogram) of clusters either by merging or splitting.
- **Types:**
 - **Agglomerative:** Start with individual points and merge clusters step by step (bottom-up).
 - **Divisive:** Start with all points in one cluster and split them recursively (top-down).
- **Linkage Criteria:** Single-link, complete-link, average-link, Ward's method.
- **Example:** Hierarchical Agglomerative Clustering (HAC).
- **Best for:** Discovering nested cluster structure.

4. Model-based Clustering

- **Idea:** Assumes data is generated from a mixture of underlying probability distributions.
- **Examples:**
 - **Gaussian Mixture Models (GMM):** Uses EM (Expectation-Maximization) to estimate parameters.
 - **Hidden Markov Models (HMM):** Sequential model-based clustering.
- **Best for:** Soft clustering (points have probabilities of belonging to each cluster).

K means

→ Partitions data into k clusters
→ Each cluster centre → Centroid

K-means algorithm

- Given k , the k -means algorithm works as follows:
 1. Choose k (random) data points (**seeds**) to be the initial **centroids**, cluster centers
 2. Assign each data point to the closest **centroid**
 3. Re-compute the **centroids** using the current cluster memberships
 4. If a convergence criterion is not met, repeat steps 2 and 3

▪ Based on the following two steps (applied in alternating fashion until not converged)

1. Assign each input to the current closest mean



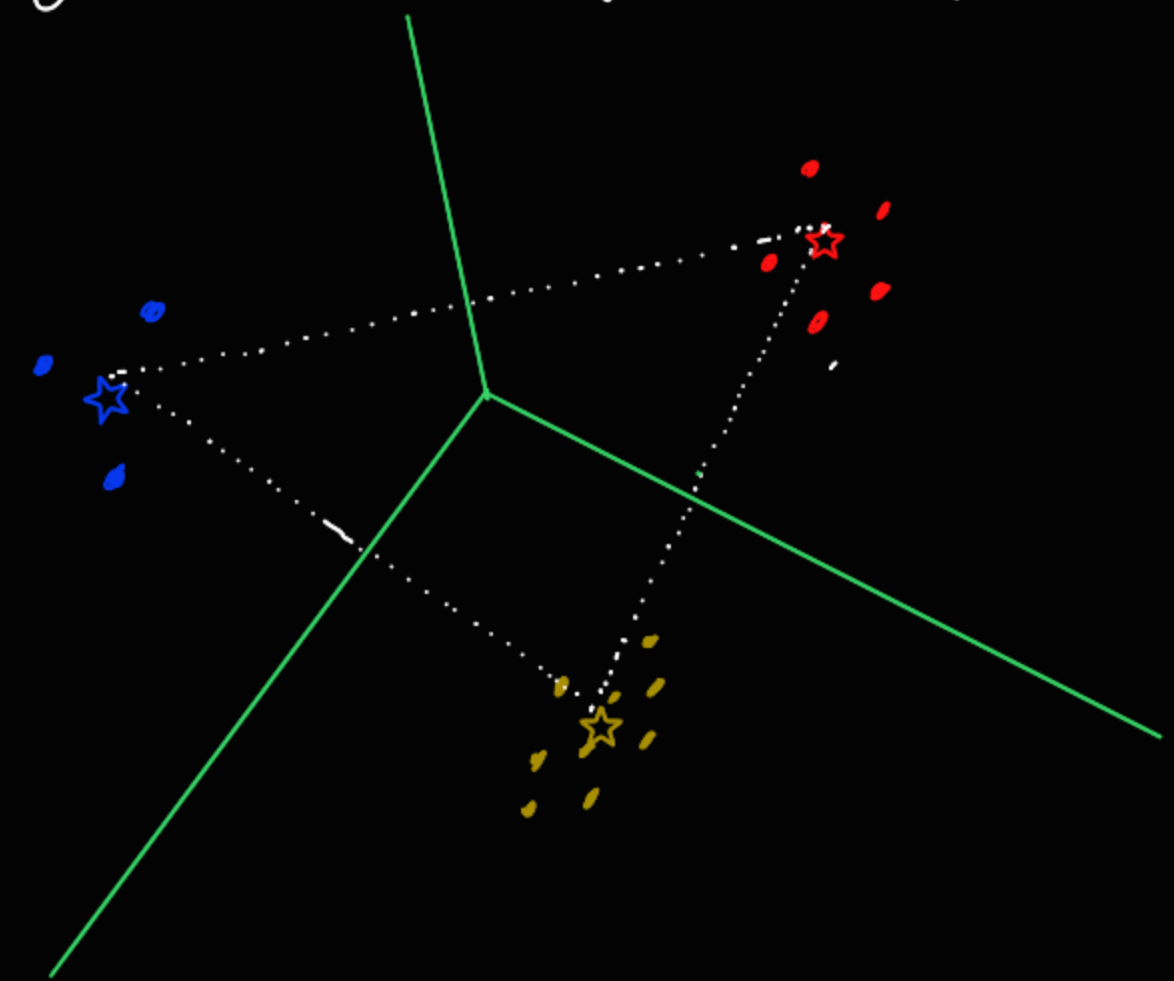
2. Recompute (update) the means



→ After convergence, the cluster center / centroids are stable & do not change

→ New unseen data is determined by checking the closeness to the centroids.

→ The decision boundaries are piecewise linear.



22. [4] You want to cluster 7 points into 3 clusters using the **k-Means Clustering** algorithm. Suppose after the first iteration, clusters C_1 , C_2 and C_3 contain the following two-dimensional points:

- C_1 contains the 2 points: $\{(0,6), (6,0)\}$
- C_2 contains the 3 points: $\{(2,2), (4,4), (6,6)\}$
- C_3 contains the 2 points: $\{(5,5), (7,7)\}$

What are the **cluster centers** computed for these 3 clusters?

✓ A. $C_1: (3,3), C_2: (4,4), C_3: (6,6)$
 B. $C_1: (3,3), C_2: (6,6), C_3: (12,12)$
 C. $C_1: (6,6), C_2: (12,12), C_3: (12,12)$
 D. $C_1: (0,0), C_2: (48,48), C_3: (35,35)$

OPTIMISATION VIEW OF CLUSTERING

K-means clustering problem:
 Partition the n observations into K sets ($K \leq n$) $S = \{S_1, S_2, \dots, S_K\}$ such that the sets minimize the within-cluster sum of squares:

$$\arg \min_S \sum_{i=1}^K \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$
 where μ_i is the mean of points in set S_i .

$$\text{Min} \left(\sum_{i=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|_2^2 \right)$$

→ This summation has just one term non zero

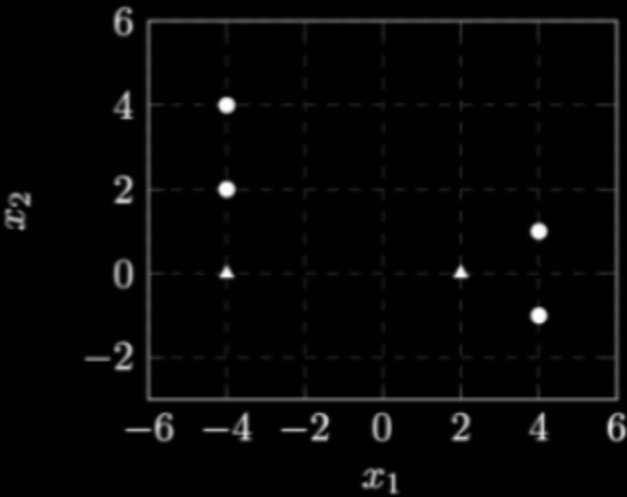
→ The non zero term corresponds to the distance between x_n & the cluster centre it belongs to

$r_{nk} \rightarrow$ Indicator RV for whether n^{th} data point belongs to cluster k .

$\mu_k \rightarrow$ centroid for the k^{th} cluster

Question:

Amazing, Inc. plans to open $k = 2$ stores in Megacity and has $n = 4$ customers located at positions $x^{(i)}$ as shown by circles in the plot below. The initial guesses for the store locations ($\mu^{(j)}$) are marked by triangles in the plot.



The loss function to minimize is:

$$L(\mu) = \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \|x^{(i)} - \mu^{(j)}\|^2$$

a) Starting Loss = $2^2 + 4^2 + 2^2 + 1^2 + 2^2 + 1^2$
 $= 12 + 16 + 2 = 30 //$

b) After convergence
 Loss = $1^2 + 1^2 + 1^2 + 1^2 = 4 //$

- a) What is the starting loss $L(\mu)$?
 b) What is the final Loss, after convergence

Question:

Recall the k -means objective function is

$$L(z_{1:n}, \mu_{1:k}) = \sum_{i=1}^n \|x^{(i)} - \mu_{z_i}\|^2,$$

where $z_{1:n}$ are the cluster assignments, $\mu_{1:k}$ are the corresponding cluster centroids, and $x^{(i)}$ is the i -th input datapoint. Now consider the following dataset consisting of three datapoints in a 2-D plane: $\{(0,3), (0,0), (4,0)\}$.

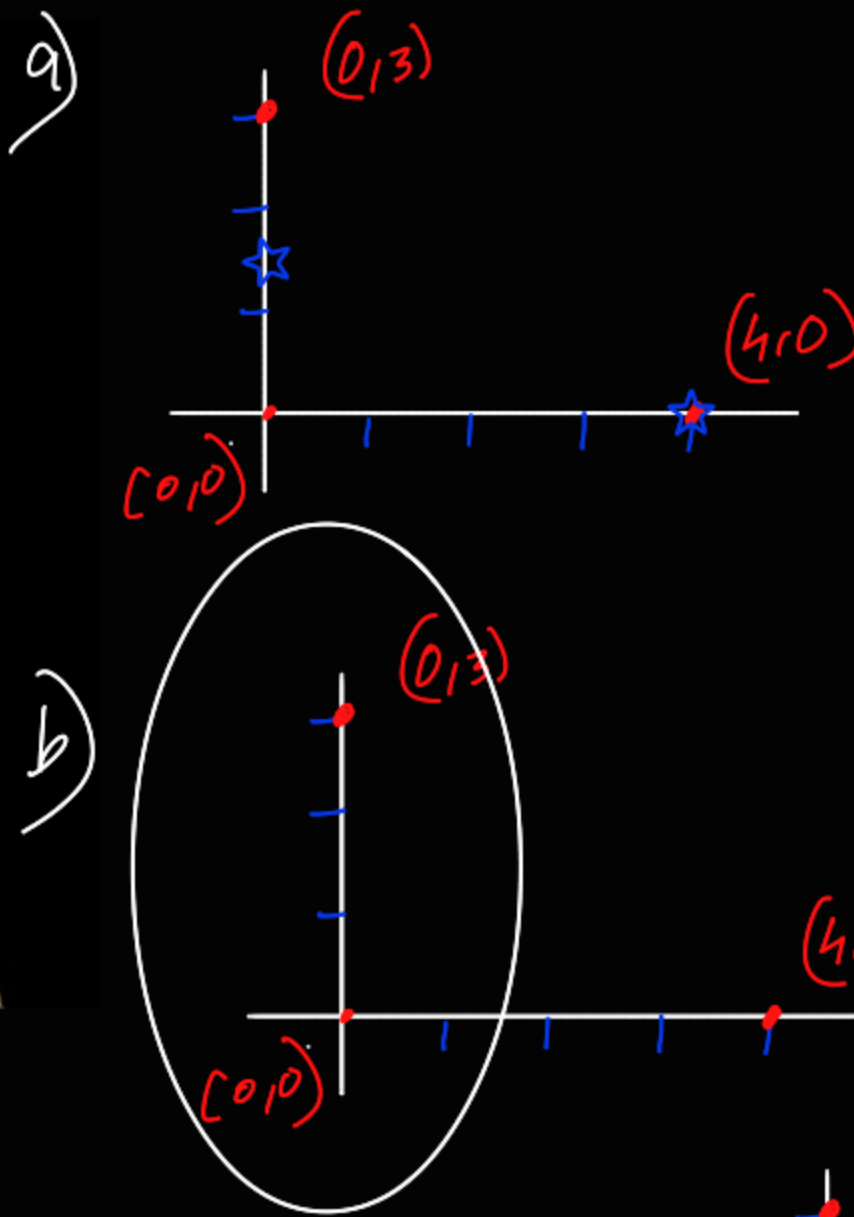
- (a) (6 points) Suppose you run k -means with $k = 2$ and have the initial cluster centers be $\mu_1 = (4,0)$ and $\mu_2 = (0,3)$. What will be the final cluster centers when k -means converges? Show your work and explain how you know that k -means has converged.
- (b) (6 points) Prove that the above initialization converges to the global minimum for $k = 2$. (Hint: Enumerate over all possible cluster assignments and compute the objective for each assignment.)

i) $C_1 = (0,3) \mid \text{Loss} = \left(\frac{3}{2}\right)^2 \times 2 + 0 = \frac{9}{2} = 4.5$
 $C_2 = (4,0)$

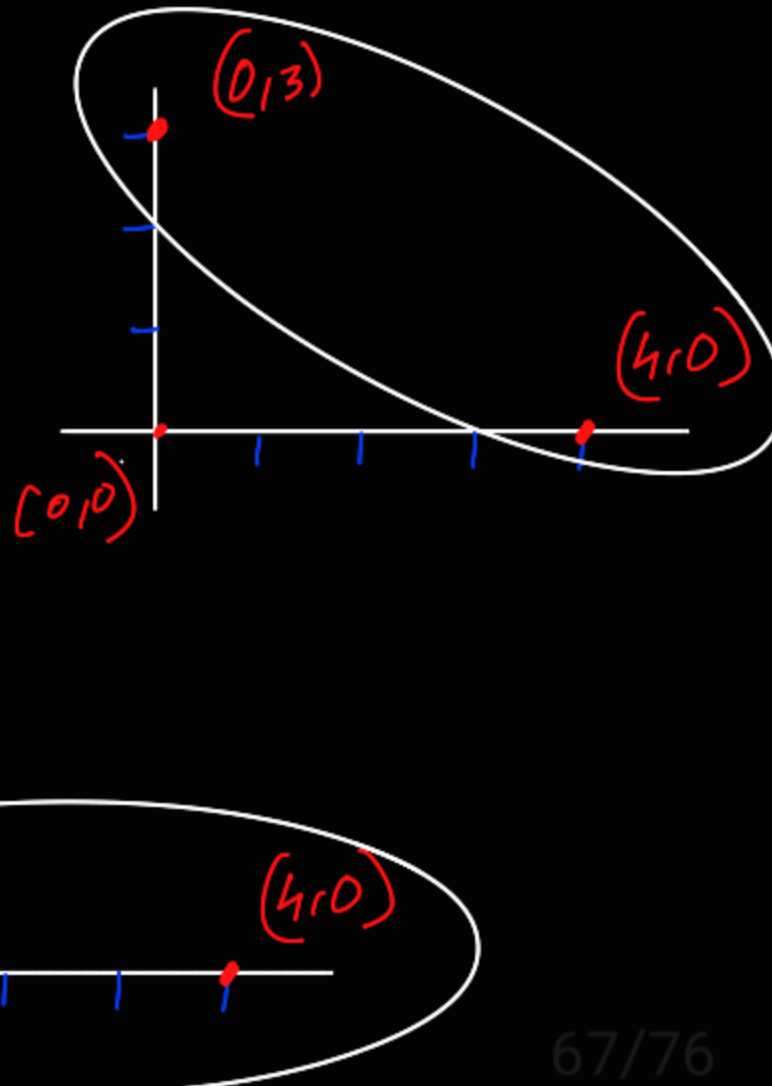
ii) $C_1 = (0,1.5) \mid \text{Loss} = \left(\frac{3}{2}\right)^2 \times 2 + 2^2 \times 2 = 8 + 4.5$
 $C_2 = (0,0)$

iii) $C_1 = (2,0) \mid \text{Loss} = 2^2 + 2^2 = 8$
 $C_2 = (0,3)$

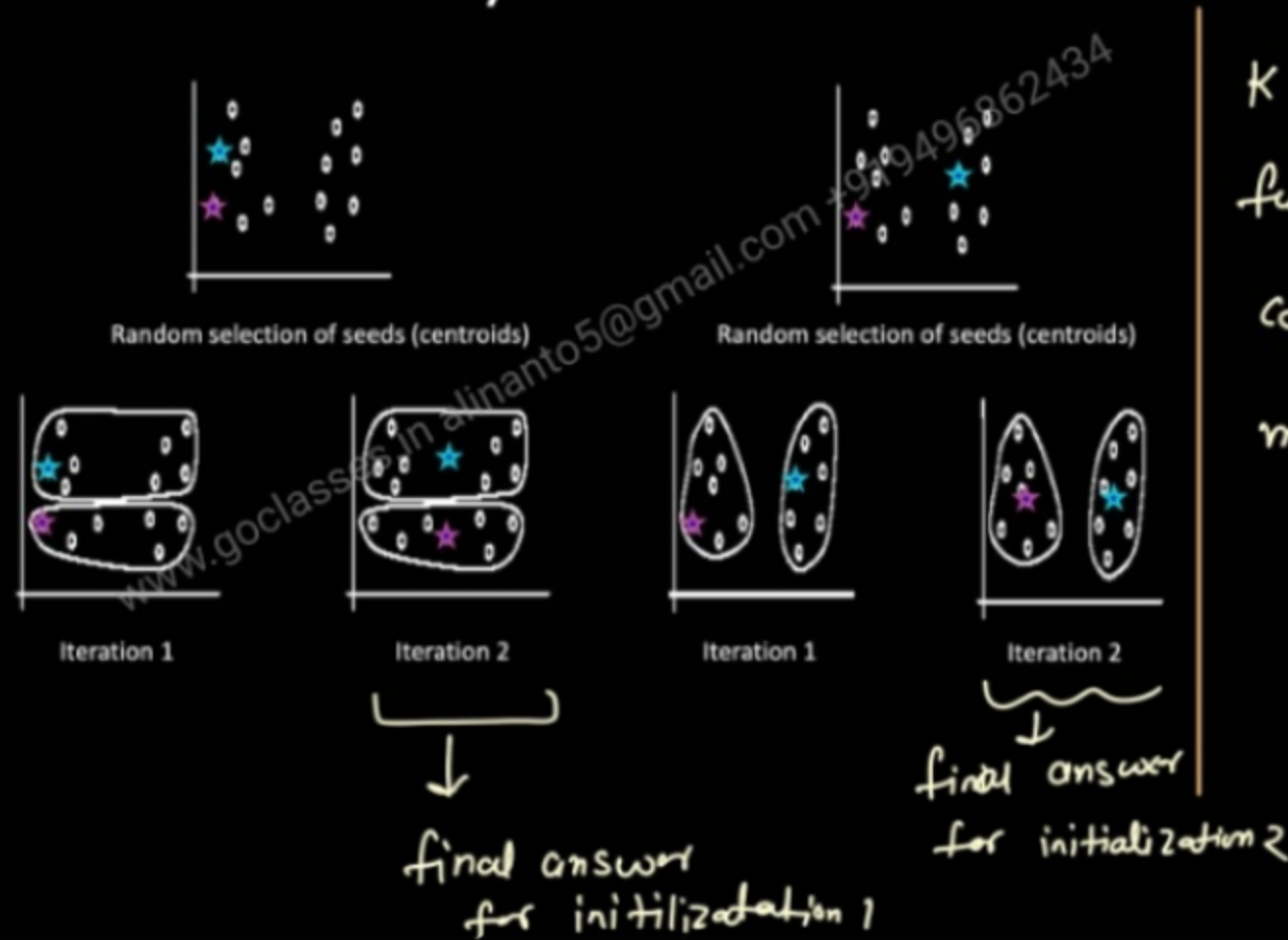
Comparatively Case (i) has lowest loss and is global minima.



cluster
cental
 $C_1 (0,1.5)$
 $C_2 (4,0)$



Sensitivity to initial seeds



K means loss function is not convex, hence may stuck to local minima.

1. K means can give different solⁿs based on the initialization
2. there is no guarantee that we will get optimal solⁿ as loss function is non convex and iterative algorithms may stuck at local minima

Recall the loss function for k-means clustering with k clusters, sample points $x_1, \dots, x_n$, and centers $\mu_1, \dots, \mu_k$:

$$L = \sum_{j=1}^k \sum_{x_i \in S_j} \|x_i - \mu_j\|^2,$$

where S_j refers to the set of data points that are closer to μ_j than to any other cluster mean.

- (1) [4 pts] Instead of updating μ_j by computing the mean, let's minimize L with **batch** gradient descent while holding the sets S_j fixed. Derive the update formula for μ_1 with learning rate (step size) ϵ .
- (2) [2 pts] Derive the update formula for μ_1 with **stochastic** gradient descent on a single sample point x_i . Use learning rate ϵ .

$$\Rightarrow \sum_{x_i \in S_j} x_i = \sum_{x_i \in S_j} \mu_j \Rightarrow \frac{\sum_{x_i \in S_j} x_i}{n_j} = \mu_j$$

→ Since during iteration each step only decreases the loss, the algorithm converges to the local minima.

a) update rule

$$\mu_1 \leftarrow \mu_1 + \alpha \sum_{x_i \in S_1} (x_i - \mu_1)$$

b) stochastic update

$$\mu_1 \leftarrow \mu_1 + \alpha (x_i - \mu_1) \text{ if } x_i \in S_1, \text{ no change otherwise}$$

Holding set S_j fixed $\Rightarrow$ points don't change clusters, when we update μ_j

$$\Rightarrow \frac{\partial L}{\partial \mu_j} = \sum_{x_i \in S_j} 2(\mu_j - x_i)$$

$$\frac{\partial L}{\partial \mu_j} = 0 \rightarrow \text{normal equation}$$

- Fixing the clusters, the minima is at the centroid
- In each iteration, while computing centroid the Loss only decreases.
- In each iteration, while recomputing cluster assignment of each point, points only shift cluster if the distance is found less to some other cluster
eg: point x shifts from cluster 1 to 2
 $\Rightarrow (x - \mu_1)^2 > (x - \mu_2)^2$
 $\Rightarrow$ In the loss summation, reassignment only decreases loss function.

Equivalent to above discussion

$$\frac{\partial}{\partial \mu_k} \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|_2^2 = 0$$

$$\Rightarrow 2 \sum_{n=1}^N r_{nk} (\mu_k - x_n) = 0$$

$$\Rightarrow \mu_k = \frac{\sum_{n=1}^N r_{nk} x_n}{\sum_{n=1}^N r_{nk}}$$

K-means Solution

A proposal: Minimise the *distortion function*, i.e., the sum of the squared distances of each data point to its closest vector μ_k .

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|^2$$

1. Given K , randomly select $\mu_{k=1, \dots, K}$
2. Minimise J with respect to r_{nk} , keeping the μ_k fixed. $\rightarrow r_{nk} = 1$ if $x_n \in \mu_k$
3. Minimise J with respect to μ_k , keeping the r_{nk} fixed.
4. Repeat steps 2 (Expectation) and 3 (Maximisation) steps until convergence
(reassignment) (calculate centroid)

K-means Solution

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

Step 2: Minimise J with respect to $\boldsymbol{\mu}_k$, keeping the r_{nk} fixed. **E step**

J is a linear function of r_{nk} . Also terms with n are independent.

Simply, $r_{nk} = 1$ for the closest cluster k , i.e. whichever k that gives the smallest value of $\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$. **Reassignment**

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|\mathbf{x}_n - \boldsymbol{\mu}_j\|^2 \\ 0 & \text{otherwise.} \end{cases}$$

Super optik

Standard k-means algorithm

Initialize cluster means arbitrarily (e.g., sample from datapoints)

– Alternate until convergence

- * Update cluster assignments: $z_{1:n} \leftarrow \arg \min_{z_{1:n}} J(z_{1:n}, m_{1:k})$
i.e., assign each point to the cluster with closest mean
- * Update cluster means: $m_{1:k} \leftarrow \arg \min_{m_{1:k}} J(z_{1:n}, m_{1:k})$
i.e., $m_j = \frac{\sum_{i=1}^n \mathbb{I}(z_i=j) x_i}{\sum_{i=1}^n \mathbb{I}(z_i=j)}$, the mean of points in cluster j

K-means algorithm always finds the clustering $(C_1, \dots, C_K)$ that minimizes the objective:

$$\text{Minimize } \sum_{j=1}^K \sum_{t \in C_j} \|x_t - r_j\|^2 \quad \text{where } r_j = \frac{1}{|C_j|} \sum_{t \in C_j} x_t \text{ is the centroid of cluster } j$$

F. K-means algorithm is only guaranteed to not make the objective worse. So it need not exactly minimize objective. It can get stuck in local minima.

false

Which of the following is true about k -means clustering?

- (a) k -means diverges and is non-convex.
- (b) k -means diverges and is convex.
- (c) k -means converges and is non-convex. **✓**
- (d) k -means converges and is convex.

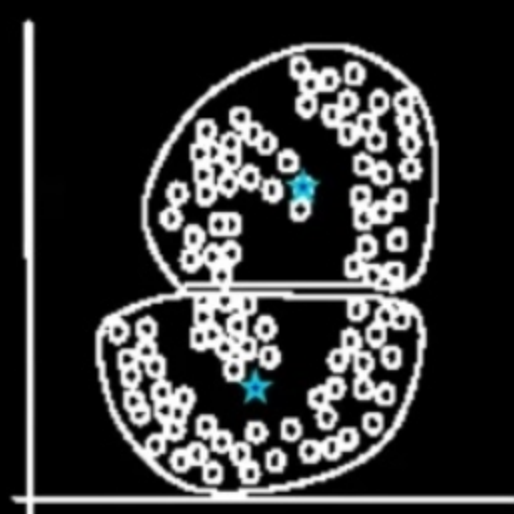
Weakness of K-means

- Sensitive to initial seeds → we have seen above
- Sensitive to outliers
- Not suitable for data distributions other than hyper-ellipsoid

- The k -means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).



(A): Two natural clusters



(B): k -means clusters

- k means DB is linear between any 2 classes
- if there are multiple classes, the decision boundary can be crudely approximated by hyper-ellipsoids.

K-means Solution

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$$

Step 3: Minimise J with respect to $\boldsymbol{\mu}_k$, keeping the r_{nk} fixed. **M step**

J is a quadratic function of $\boldsymbol{\mu}_k$ and can be minimised by setting its derivative with respect to $\boldsymbol{\mu}_k$ to zero, that is $\frac{\delta J}{\delta \boldsymbol{\mu}_k} = 0$.

$$\begin{aligned} \frac{\delta J}{\delta \boldsymbol{\mu}_k} &= \frac{\delta \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2}{\delta \boldsymbol{\mu}_k} = \sum_{n=1}^N r_{nk} \times (-1) \times 2(\mathbf{x}_n - \boldsymbol{\mu}_k) = 0 \\ &= \sum_{n=1}^N r_{nk} \mathbf{x}_n - \sum_{n=1}^N r_{nk} \boldsymbol{\mu}_k = 0 \quad \rightarrow \quad \boldsymbol{\mu}_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}} \end{aligned}$$

Calculating Centroid

Q.42 Given the dataset: (1, 1), (3, 3), (4, 4), (5, 5), (6, 6), (9, 9), (0, 3), (3, 0) and assuming the initial centroids for ($K = 3$ – means clustering) to be $C_1 = (3, 3)$, $C_2 = (5, 5)$ and $C_3 = (6, 6)$. One iteration of the Expectation Maximization Algorithm for K-means clustering, will update C_3 to (__, __)

(1, 1), (3, 3), (4, 4), (5, 5), (6, 6), (9, 9), (0, 3), (3, 0)

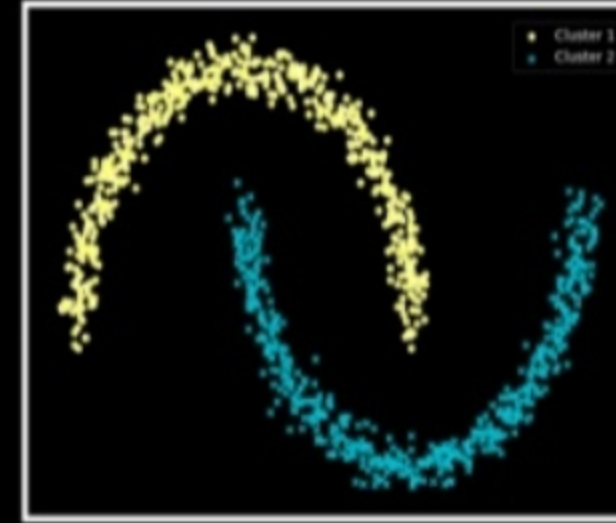
C_1 C_1 C_1 C_2 C_3 C_3 C_1 C_1

$C_3 \rightarrow (7.5, 7.5)$

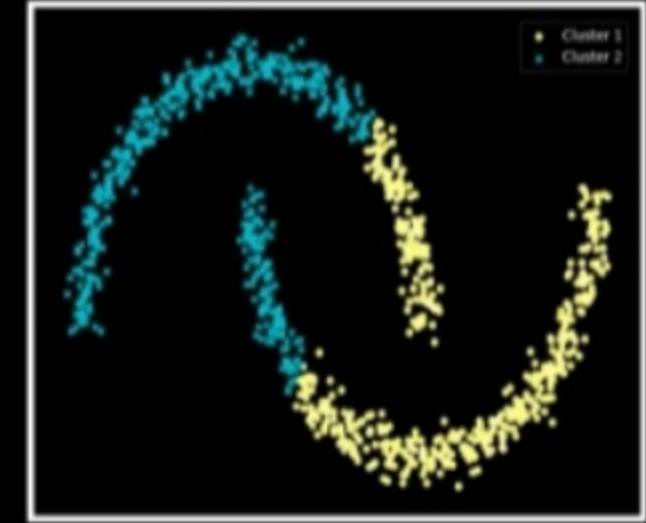
Q.19 Euclidean distance based k -means clustering algorithm was run on a dataset of 100 points with $k = 3$. If the points $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ are both part of cluster 3, then which ONE of the following points is necessarily also part of cluster 3?

- (A) $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$
(B) $\begin{bmatrix} 0 \\ 2 \end{bmatrix}$
(C) $\begin{bmatrix} 2 \\ 0 \end{bmatrix}$
(D) $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$

In which of the following plots are the points clustered by k -means clustering?



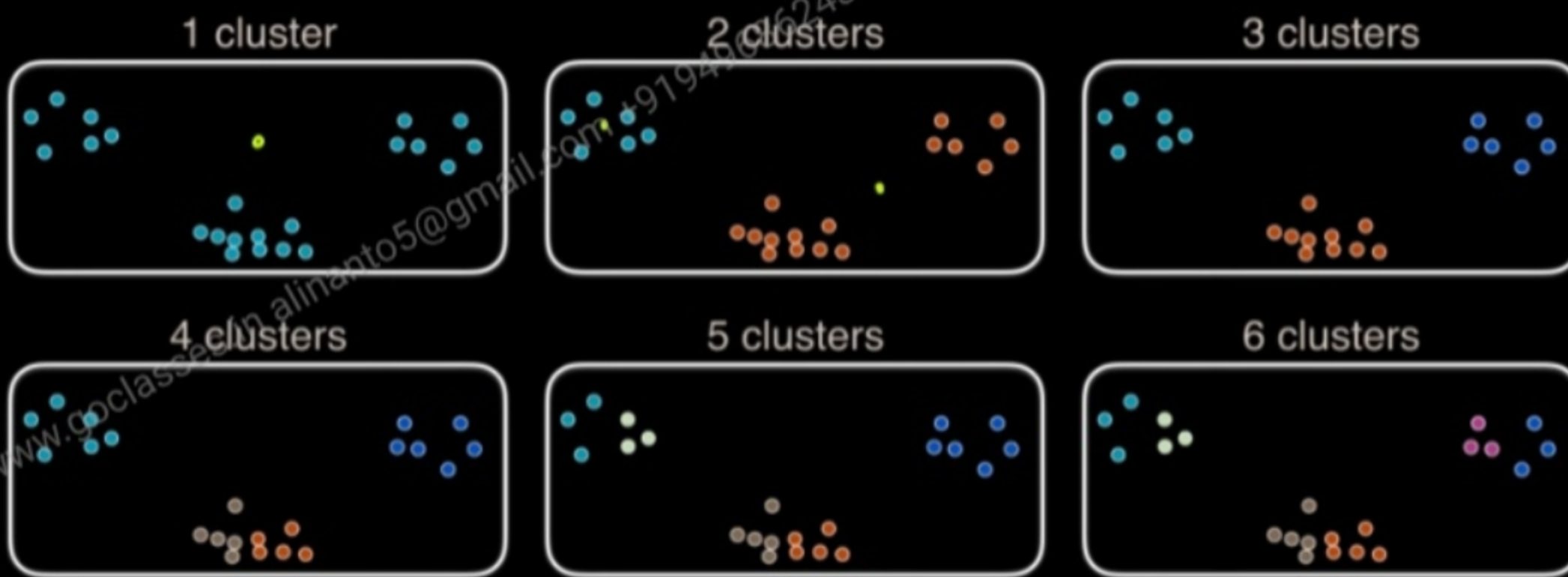
(a)



(b)

Elbow Method

→ Solution to how many clusters to take
→ Because k -means algo need k to be specified



The same dataset has different solution for k -means

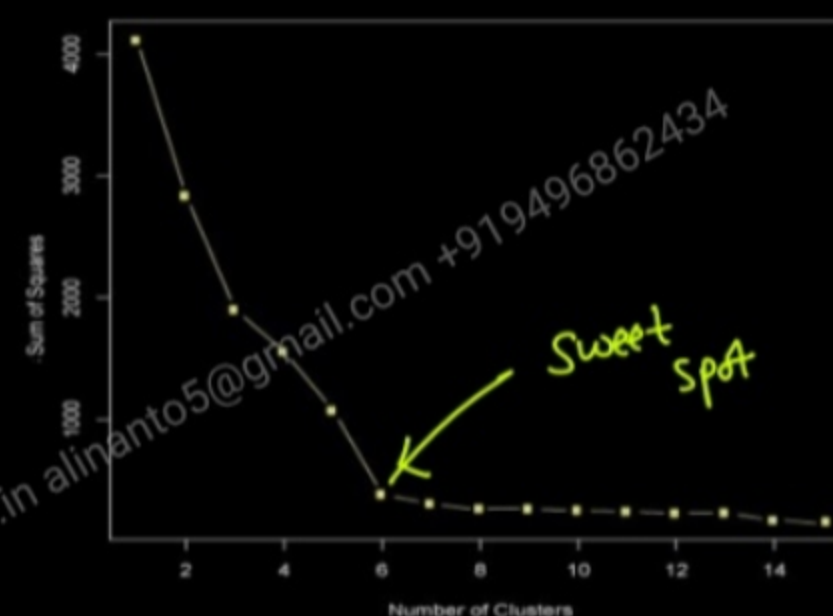
Elbow Method:

- First of all, compute the sum of squared error (SSE) for some values of k (for example 2, 4, 6, 8, etc.). The SSE is defined as the sum of the squared distance between each member of the cluster and its centroid. Mathematically:

$$SSE = \sum_{i=1}^K \sum_{x \in C_i} \text{dist}(x, c_i)^2 \rightarrow \|x - c_i\|^2$$

- If you plot k against the SSE, you will see that the error decreases as k gets larger; this is because when the number of clusters increases, they should be smaller, so distortion is also smaller. The idea of the elbow method is to choose the k at which the SSE decreases abruptly. This produces an "elbow effect" in the graph

K means
Loss



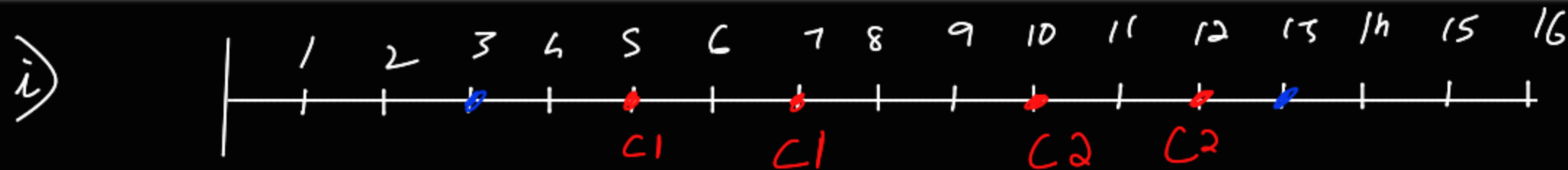
In this case, $k=6$ is the value that the Elbow method has selected.

(c) [9] Consider performing K -Means Clustering on a one-dimensional dataset containing four data points: {5, 7, 10, 12} using $k = 2$, Euclidean distance, and the initial cluster centers are $c_1 = 3.0$ and $c_2 = 13.0$.

- (i) [3] What are the initial cluster assignments? (That is, which examples are in cluster c_1 and which examples are in cluster c_2 ?)
- (ii) [3] What is the value of distortion for the clusters computed in (i)?
→ loss
- (iii) [3] What are the new cluster centers after making the assignments in (i)?

Distortion $\equiv$ k -means loss





ii) $distortion = 2^2 + 4^2 + 3^2 + 1^2 = 30$

iii) $\mu_1 = \frac{5+7}{2} = 6$; $\mu_2 = \frac{10+12}{2} = 11$

Question: True/False

[2] True or False: A good way to pick the number of clusters, k , used for k -Means clustering is to try multiple values of k and choose the value that minimizes the distortion measure.

False

Elbow point $\rightarrow$ knee

loss if k reach n — #data points then distortion $= 0$

Perform k -means clustering on six points $x_i = i$ for $i = 1 \dots 6$ with two clusters. Initially the cluster centers are at $c_1 = 1, c_2 = 6$. Run k -means until convergence. What is the reduction in distortion?

- (A) 2 (B) 4 (C) 6 (D) 8 (E) none of the above

$\Rightarrow$ distortion $= 1^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 = 4$
Initially distortion $= 0^2 + 1^2 + 0^2 + 0^2 + 1^2 + 0^2 = 10$
Reduction $= 10 - 4 = 6$



$c_1 = \{1, 2, 3\}$ } initially
 $c_2 = \{4, 5, 6\}$

$\mu_1 = 2$ } converges
 $\mu_2 = 5$

K-Medoids

K-medoids algorithm

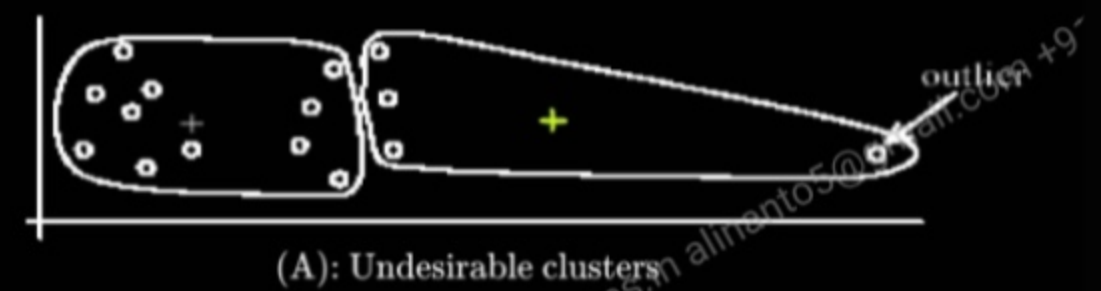
In some applications we want each center to be one of the points itself. This is where **K-medoids** comes in—an algorithm similar to the K -means algorithm, except when fitting the centers $c_1, \dots, c_K$, we restrict our attention to the points themselves

The mean in k -means clustering is sensitive to outliers. Since an object with an extremely high value may substantially distort the distribution of data.

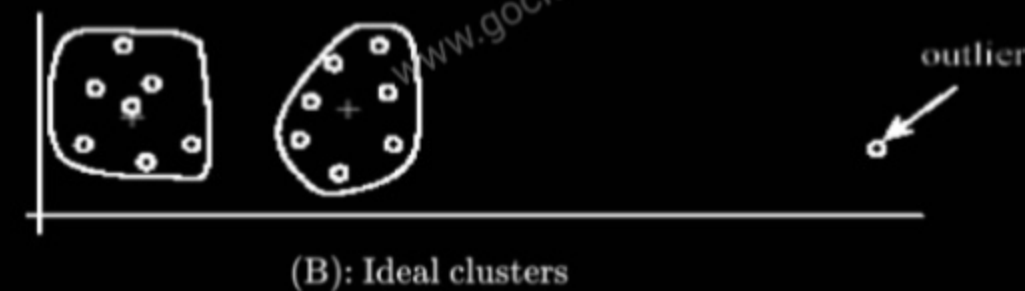
Hence we move to k -medoids.

Instead of taking mean of cluster we take the most centrally located point in cluster as it's center.

These are called medoids.



Kmeans



K medoids

Hierarchical Clustering

Agglomerative (bottom-up)

- Each point starts in its own cluster.
- Repeatedly merge the two most-similar clusters until only one remains.

Divisive (top-down)

- All points start in a single cluster.
- Repeatedly split the data into the two most self-similar subsets.

→ By convention hierarchical clustering by default means agglomerative (bottom up) since Divisive is very less common.

Agglomerative (Bottom-up) Hierarchical clustering

From K-means to hierarchical clustering

Recall two properties of **K-means (K-medoids)** clustering:

1. It fits exactly K clusters (as specified)
2. Final clustering assignment depends on the chosen initial cluster centers

Given pairwise dissimilarities d_{ij} between data points, **hierarchical clustering** produces a consistent result, without the need to choose initial starting positions (number of clusters)

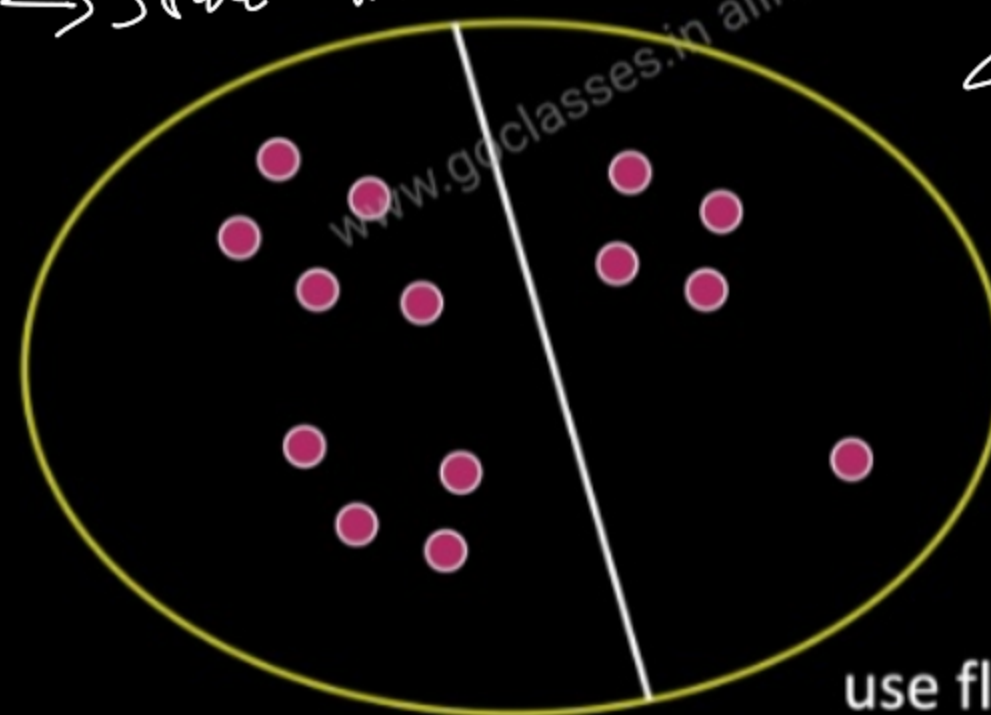
The catch: we need to choose a way to measure the dissimilarity between groups, called the **linkage**

Given the linkage, hierarchical clustering produces a sequence of clustering assignments. At one end, all points are in their own cluster, at the other end, all points are in one cluster

#clusters not predetermined.

Divisive clustering

→ Start with entire dataset as one cluster.



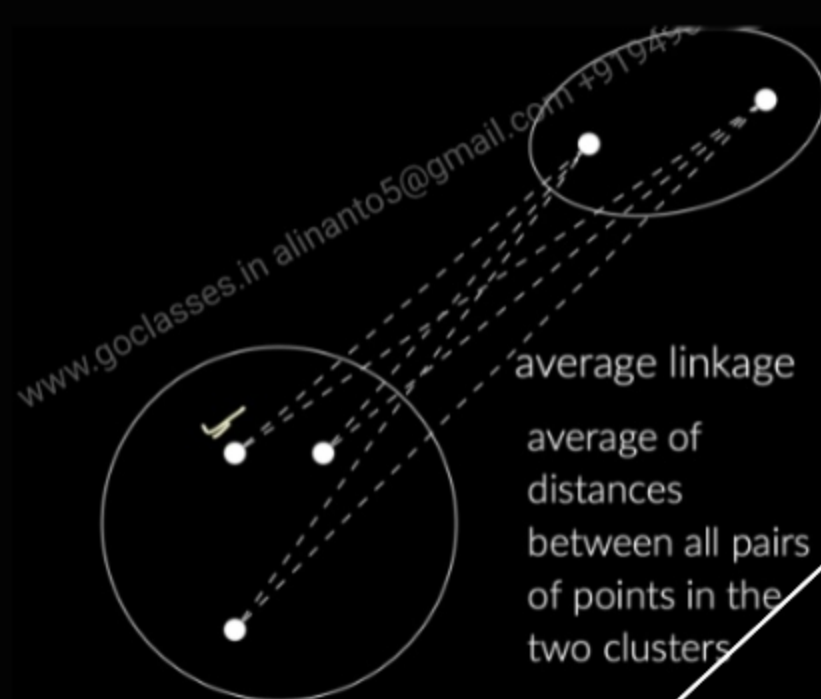
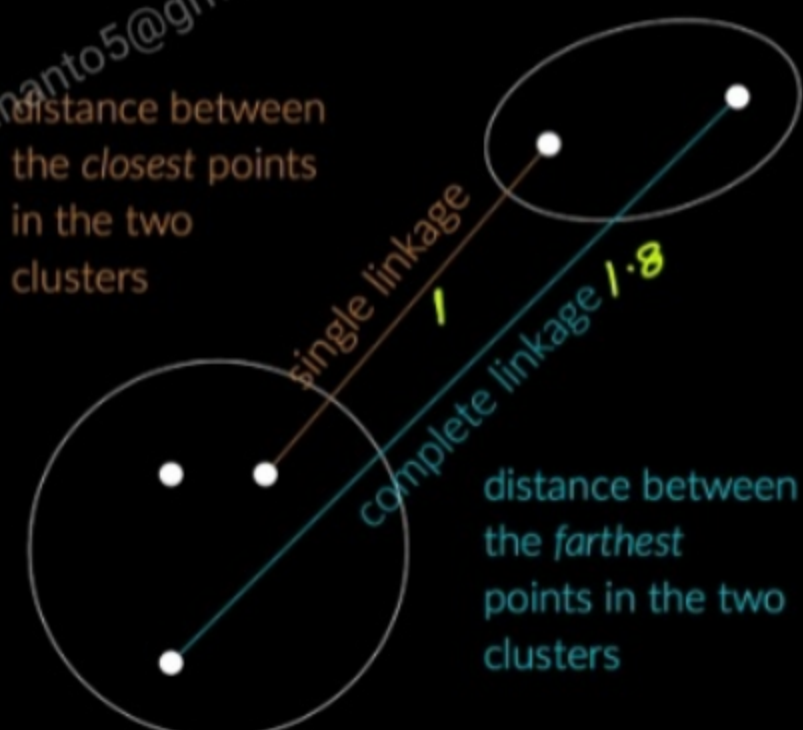
use flat clustering to split into two clusters (e.g., using K-means with $k=2$)

→ Stop when clusters reach some contain
→ Less common since it calls all the demerits of using flat clustering

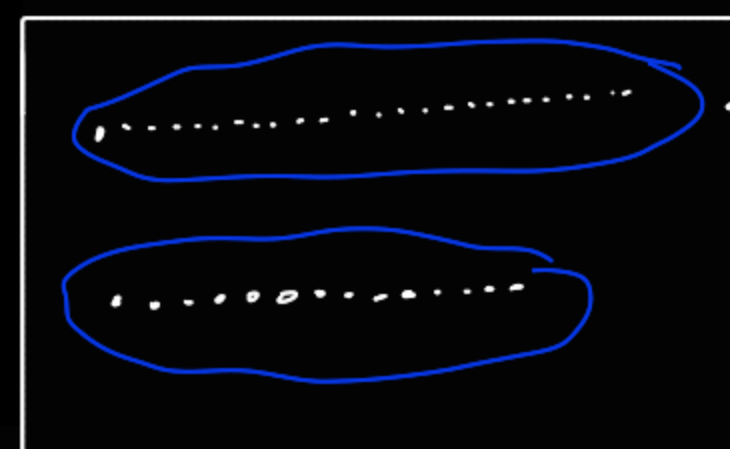
- ▶ **K-means or K-medoids** Recursively applies **2-means or 2-medoids** to a selected cluster.
- ▶ **Downsides of this approach:** Results depend on the initialization of k-means / k-medoids at each division step due to multiple stationary points.

LINKAGE

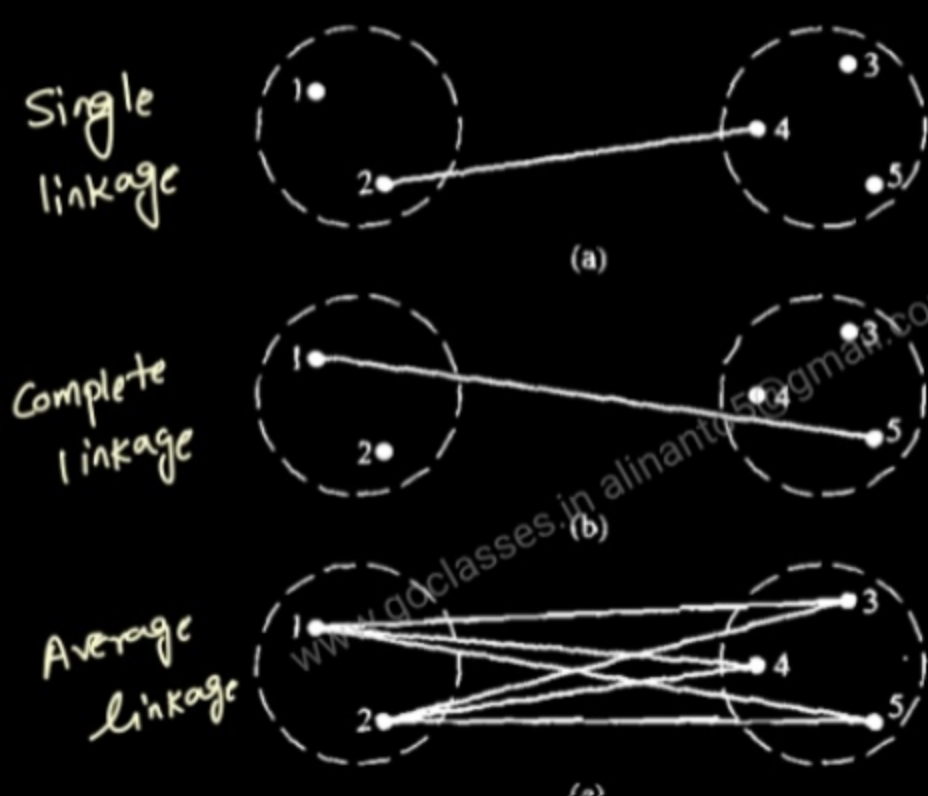
→ The choice of how to measure distance btw clusters is called linkage



why do we have multiple notions of linkage?



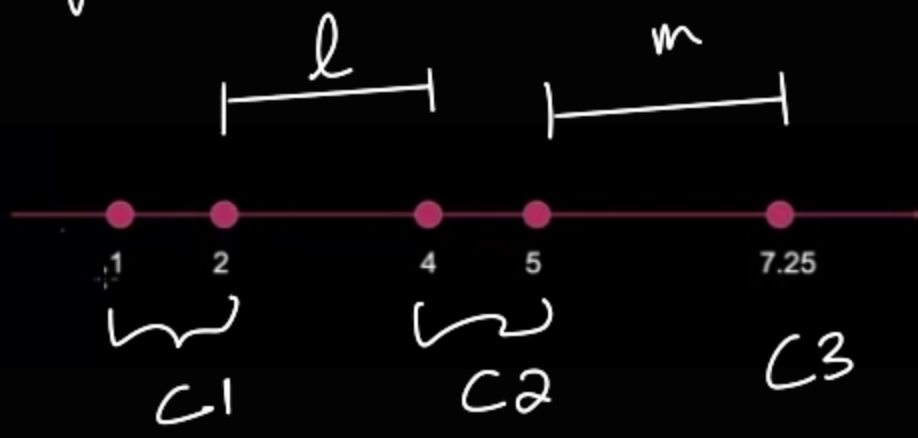
→ single linkage is useful when data is in form of chain



→ Similarly other data types might have different cluster property which can be selectively used based on the linkage we use.

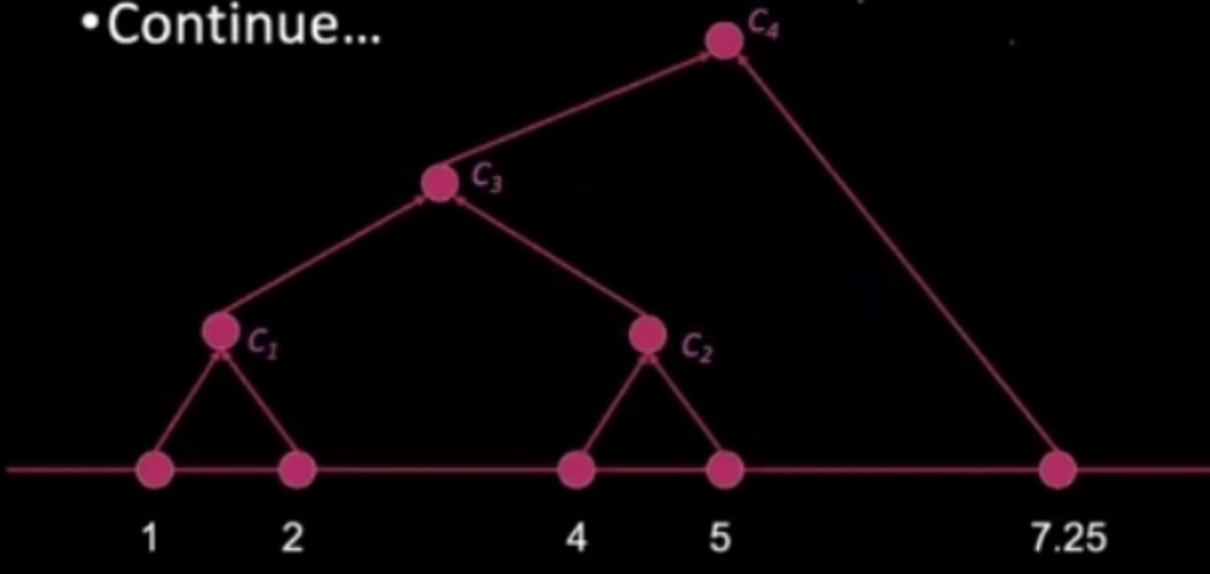
Single Linkage

eg



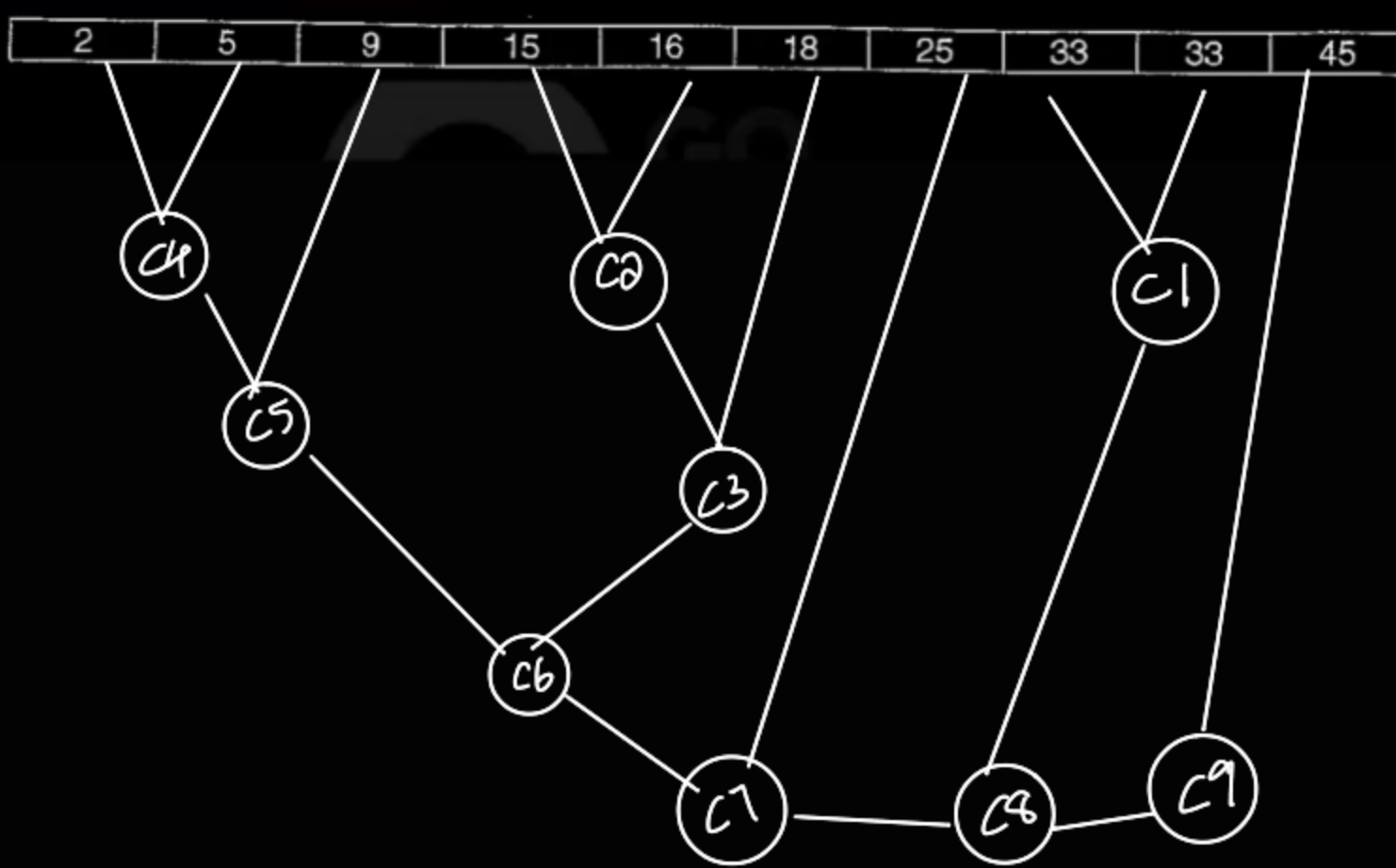
- Single Linkage $l=2, m=2.25$
- Complete Linkage $l=3, m=3.25$
- Average Linkage $l=2.5, m=2.75$

•Continue...

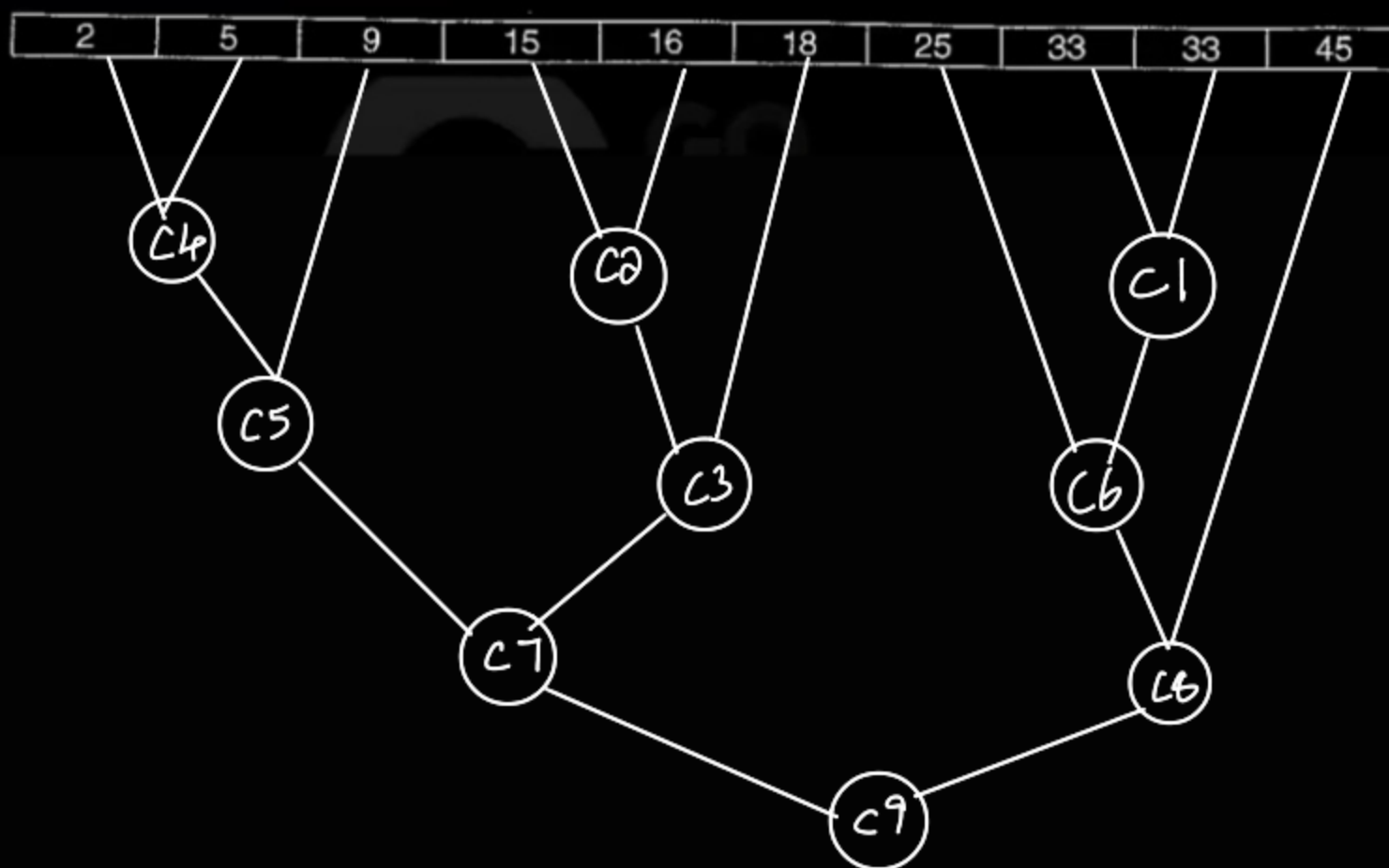


→ if we continue, we end up with just one cluster

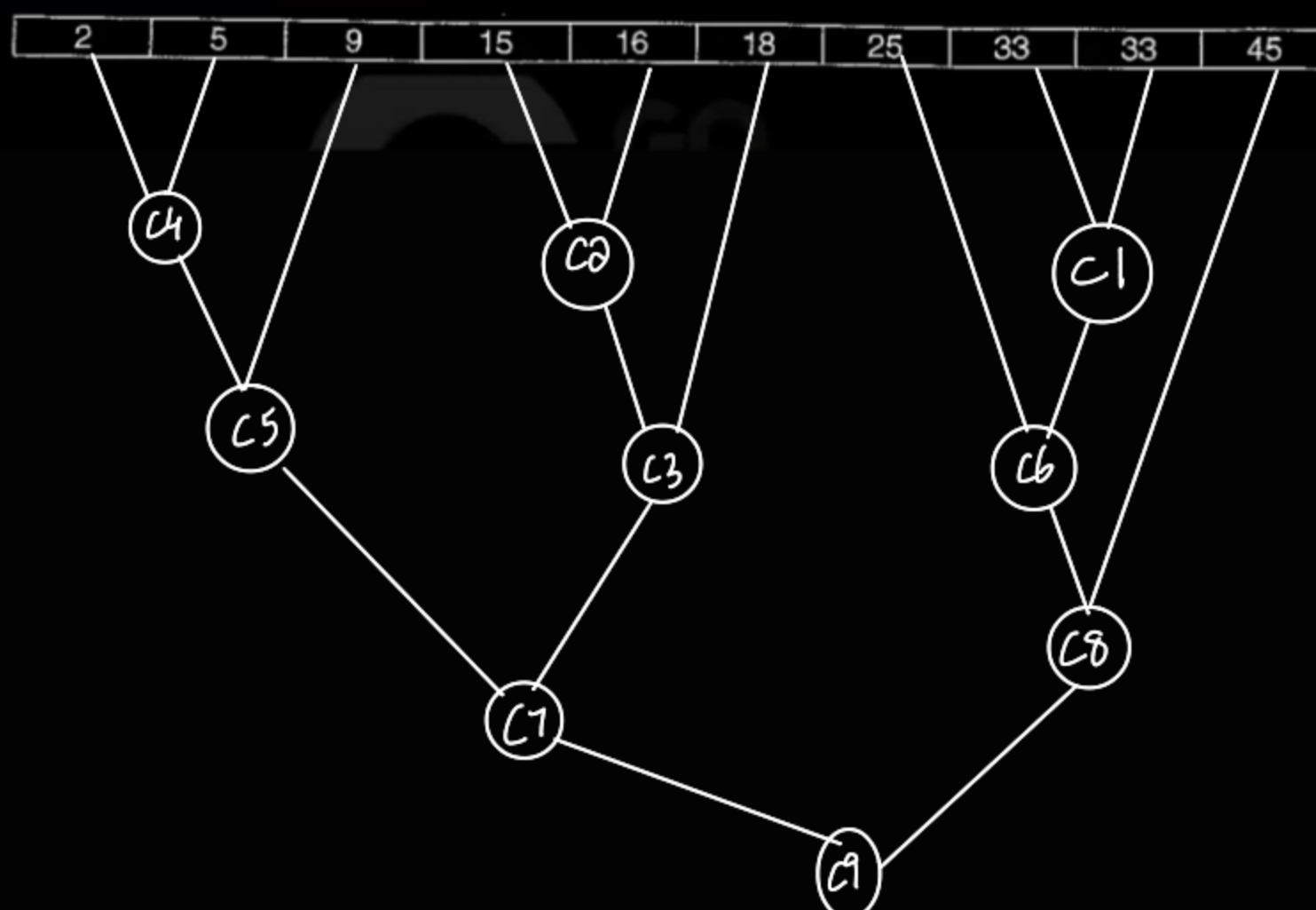
eg:-



Single Linkage.



Complete Linkage



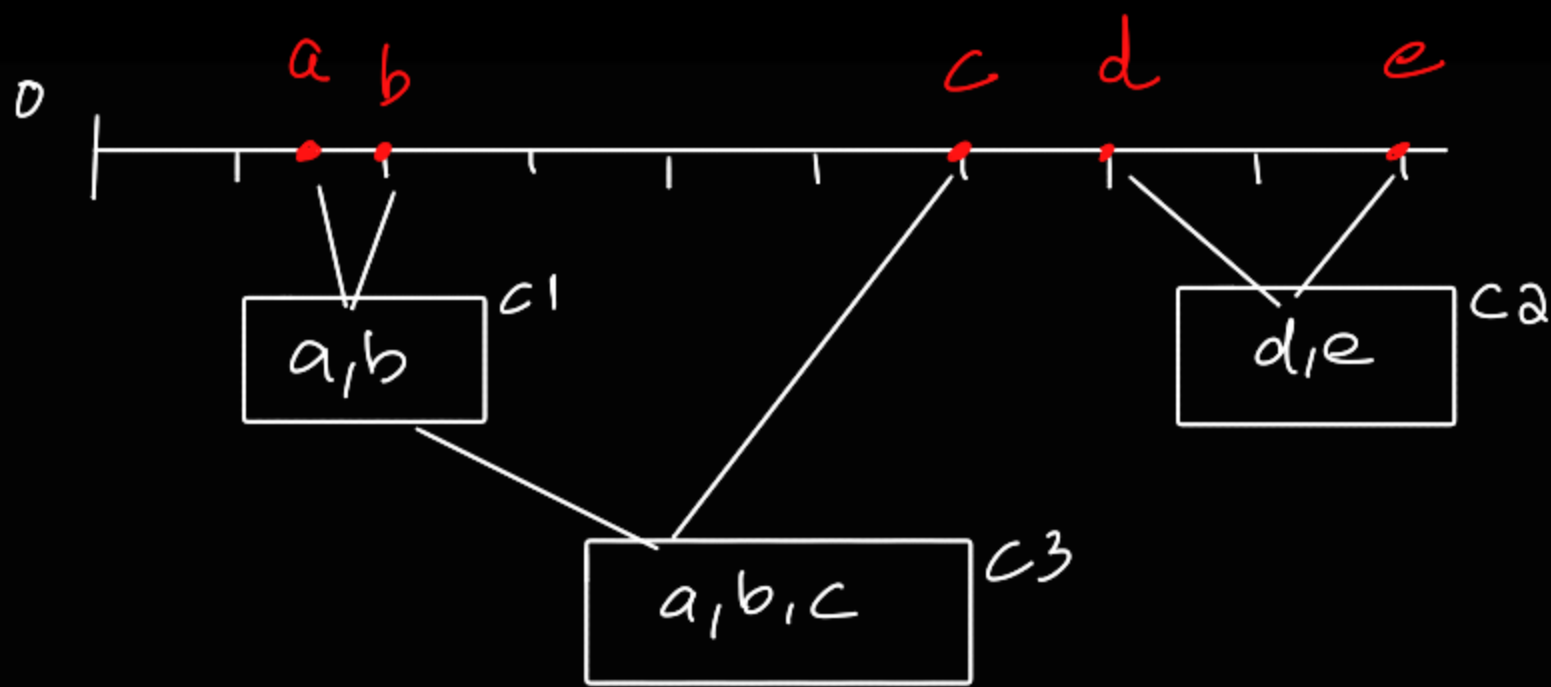
Average Linkage

Observation

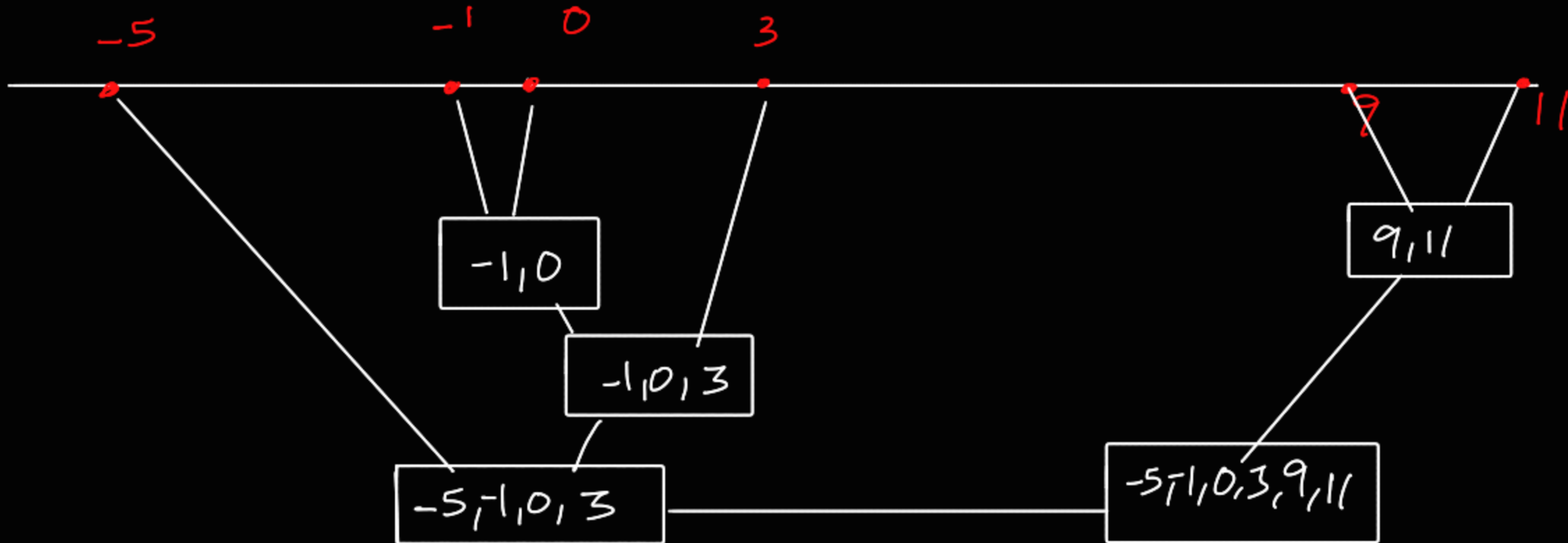
→ In this example, avg linkage leads to same structure as complete linkage.

→ generally: Avg linkage more similar to complete than to single

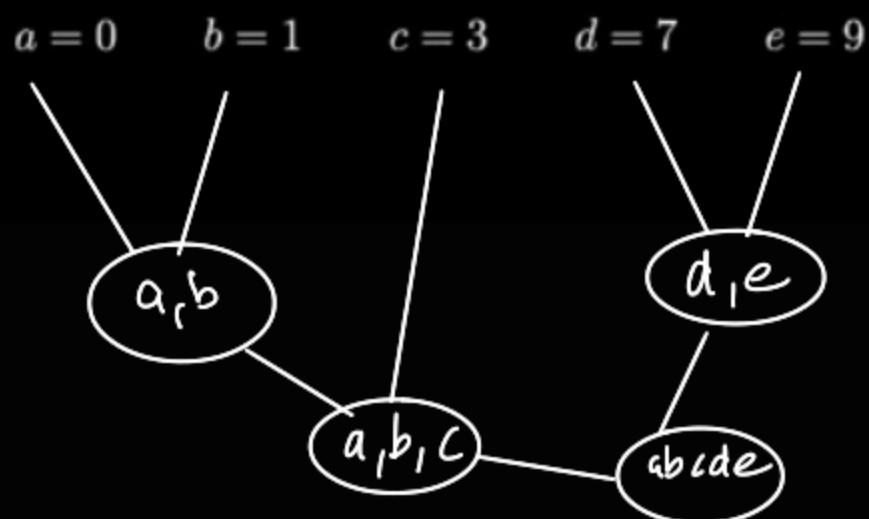
(Clustering) There are five points in one-dimensional space: $a=1.5, b=2, c=6, d=7, e=9$. Perform Hierarchical Agglomerative Clustering with Single Linkage, with the following extra constraint: at no time should c and d be in the same cluster. Complete the resulting clustering tree diagram (i.e., the dendrogram).



Perform hierarchical clustering with single linkage in one-dimensional space on the following points: $-5, -1, 0, 3, 9, 11$. Draw the clustering tree.



(Clustering) There are five points in one-dimensional space: $a = 0, b = 1, c = 3, d = 7, e = 9$. Perform Hierarchical Agglomerative Clustering with complete linkage. Complete the resulting clustering tree diagram (i.e., the dendrogram).



2. (Hierarchical Clustering) Consider six points in 2D, some of them overlap. We start with three clusters (rather than from scratch):

Cluster 1 = $(0,0), (1,1), (2,2)$

Cluster 2 = $(0,1), (1,0)$

Cluster 3 = $(0,0)$

With Euclidean distance and complete linkage, which clusters are merged in the next step of hierarchical clustering?

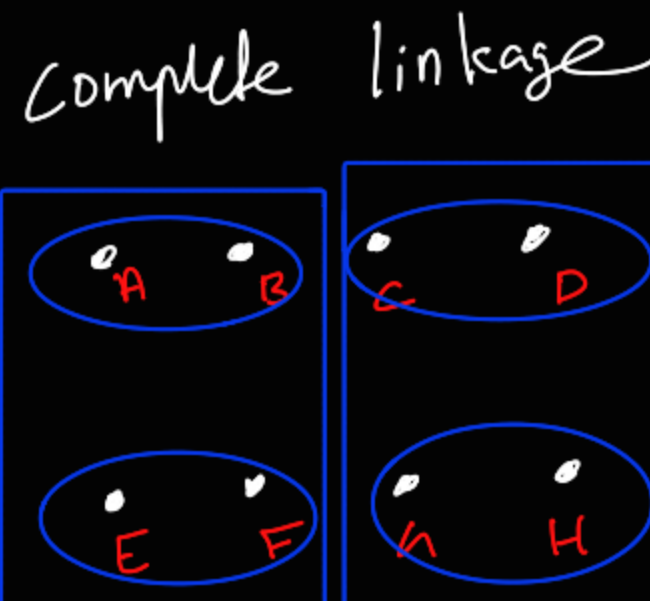
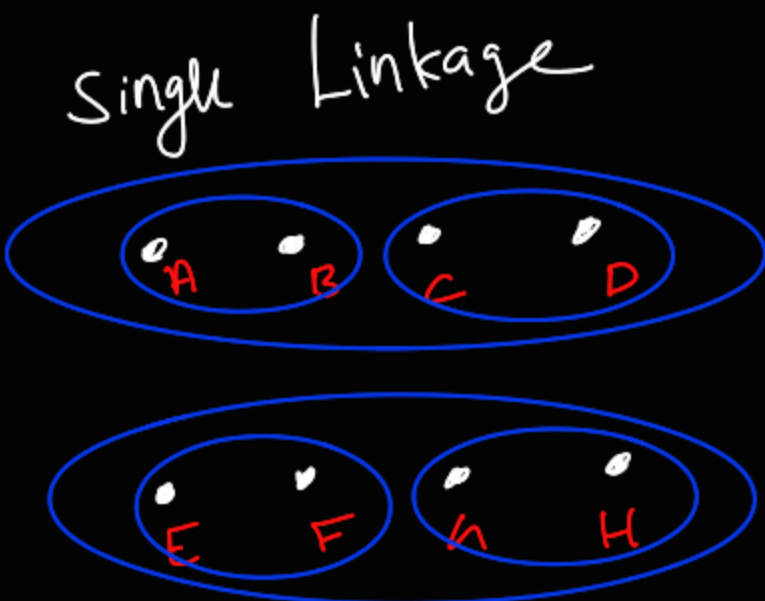
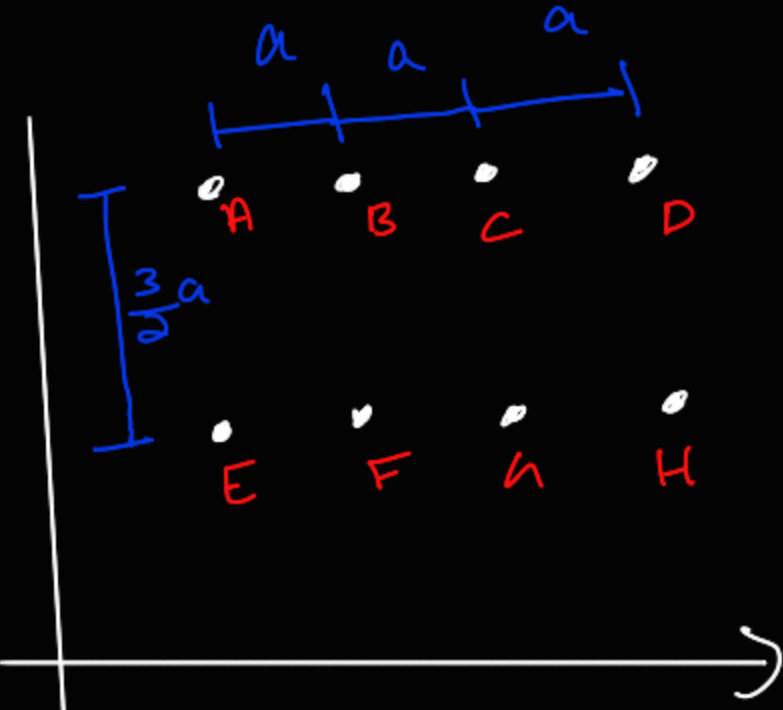
$$(d_{12})^2 = 2^2 + 1^2 = 5$$

$$(d_{23})^2 = 1^2 = 1 \rightarrow \text{clusters 2 \& 3 will be merged in the next step}$$

$$(d_{13})^2 = 2^2 + 2^2 = 8$$

→ Comparing square is also fine in complete linkage, since it won't affect monotonicity.

a) Perform both single & complete linkage



→ Single Linkage promotes chain like clusters

Consider the pairwise distances between 5 points given as matrix.

Since the matrix is symmetric, we have given only the lower triangle.

Find the final Clustering using single linkage.

	1	2	3	4	5
1	0				
2	2	0			
3	6	3	0		
4	10	9	7	0	
5	9	8	5	4	0

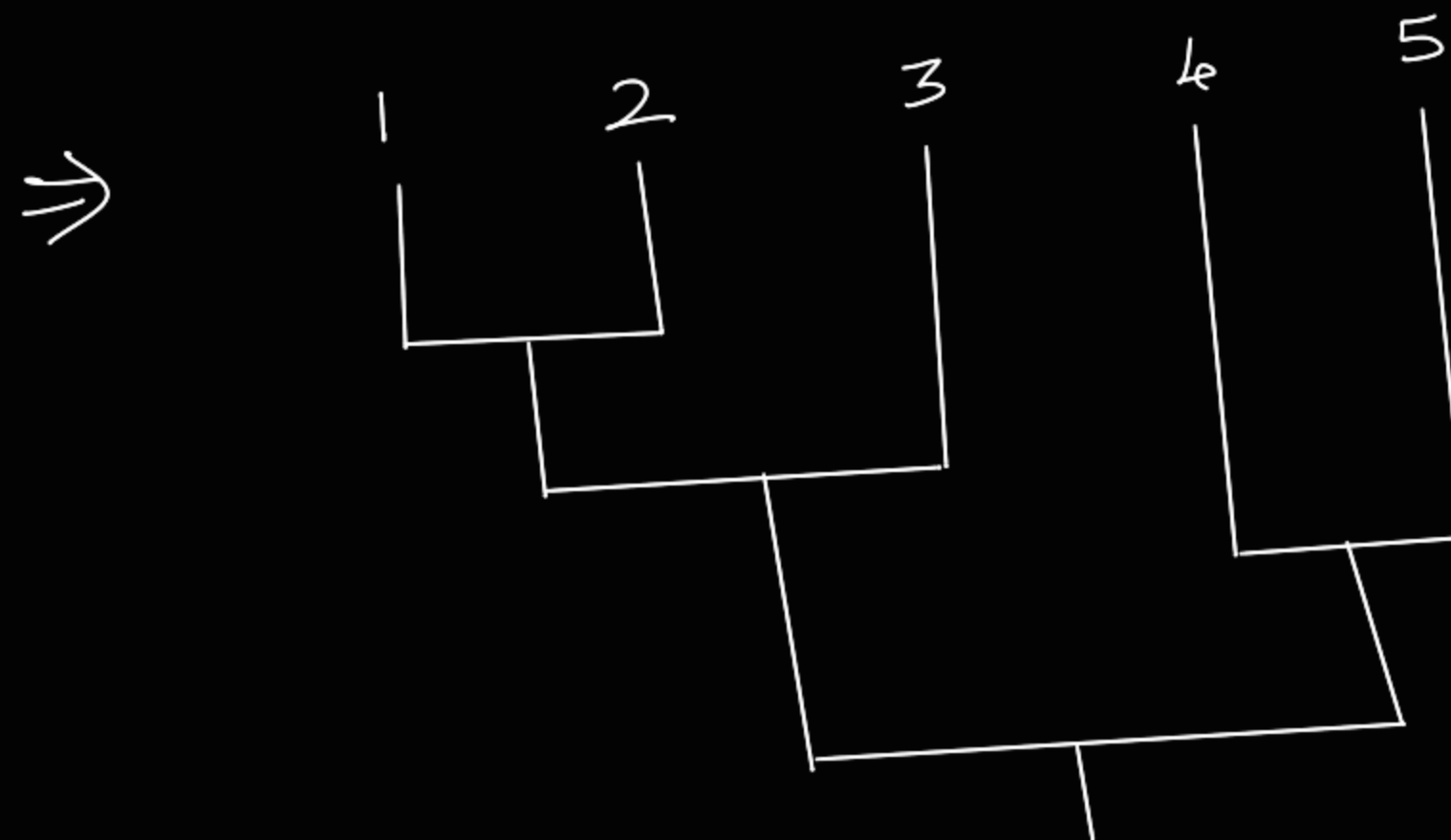
Smallest is {1,2}

	{1,2}	3	4	5
{1,2}	0			
3	3	0		
4	9	7	0	
5	8	5	4	0

Smallest is {1,2,3}

	{1,2,3}	4	5
{1,2,3}	0		
4	7	0	
5	5	4	0

Smallest is {4,5}



Q.42

Consider the table below, where the $(i,j)^{th}$ element of the table is the distance between points x_i and x_j . Single linkage clustering is performed on data points, x_1, x_2, x_3, x_4, x_5 .

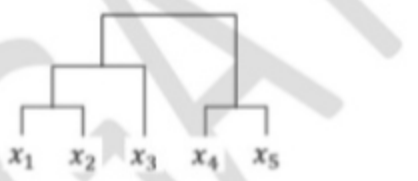
	x_1	x_2	x_3	x_4	x_5
x_1	0	1	4	3	6
x_2	1	0	3	5	3
x_3	4	3	0	2	5
x_4	3	5	2	0	1
x_5	6	3	5	1	0

Which ONE of the following is the correct representation of the clusters produced?

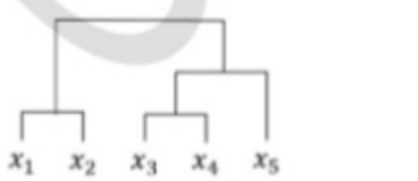
(A)



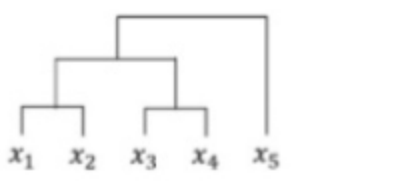
(B)



(C)



(D)

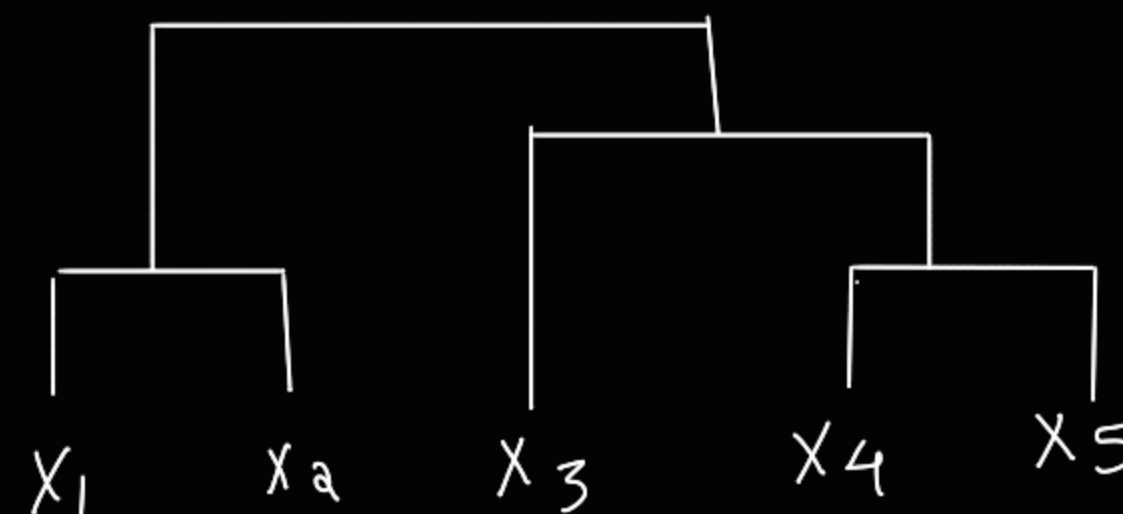


Smallest is {1,2} & {4,5}

	{1,2}	x_3	{4,5}
{1,2}	0		
x_3	3	0	
{4,5}	3	2	0

Smallest is x_3 with {4,5}

→ no need of further Matrices.



Complete Linkage for the following cluster →
 D_{ij} = distance btw points i & j

	1	2	3	4	5
1	0				
2	9	0			
3	3	7	0		
4	6	5	9	0	
5	11	10	2	8	0

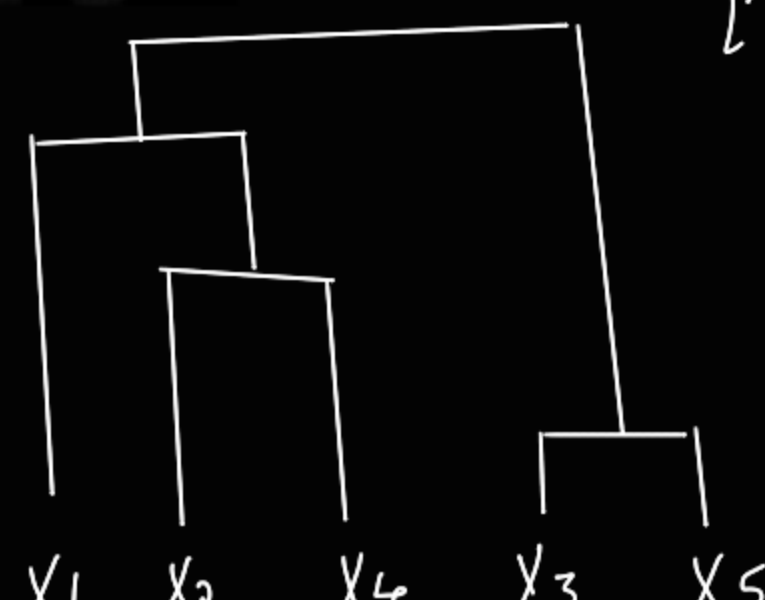
Smallest linkage for {3,5}

	x_1	x_2	x_4	{3,5}
x_1	0			
x_2	9	0		
x_4	6	5	0	
{3,5}	11	10	9	0

Smallest is {2,4}

	x_1	{2,4}	{3,5}
x_1	0		
{2,4}	9	0	
{3,5}	11	10	0

Smallest is {1,2,4} No need further Matrices

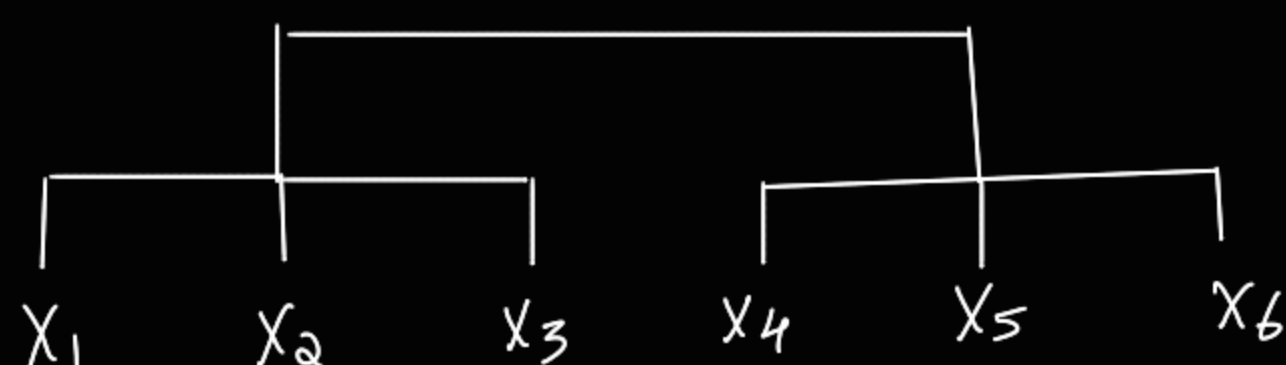


Perform **agglomerative Hierarchical Clustering** using **single linkage** as the cluster distance measure. Draw the associated tree.

	$\{x_1\}$	$\{x_2\}$	$\{x_3\}$	$\{x_4\}$	$\{x_5\}$	$\{x_6\}$
$\{x_1\}$	0	1	2	1.5	1.5	1.5
$\{x_2\}$	1	0	1	2.5	1.5	2.5
$\{x_3\}$	2	1	0	3.5	2.5	3.5
$\{x_4\}$	1.5	2.5	3.5	0	1	1
$\{x_5\}$	1.5	1.5	2.5	1	0	1
$\{x_6\}$	1.5	2.5	3.5	1	1	0

Smallest are $\{x_1, x_2\}$ & $\{x_4, x_5\}$ with linkage distance = 1

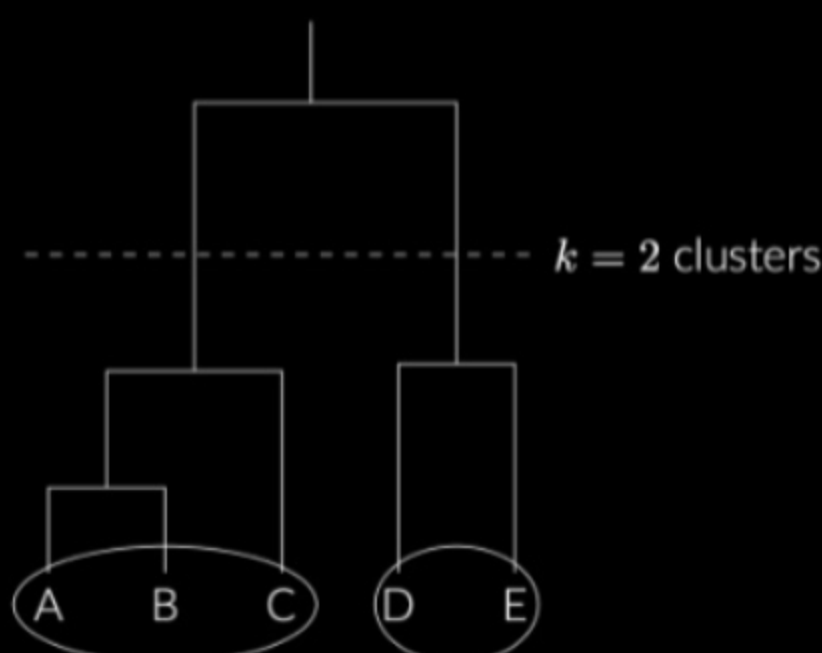
	$\{x_1, x_2\}$	x_3	$\{x_4, x_5\}$	x_6
$\{x_1, x_2\}$	0			
x_3	1	0		
$\{x_4, x_5\}$	1.5	2.5	0	
x_6	1.5	3.5	1	0



Dendrogram

Finding k Clusters

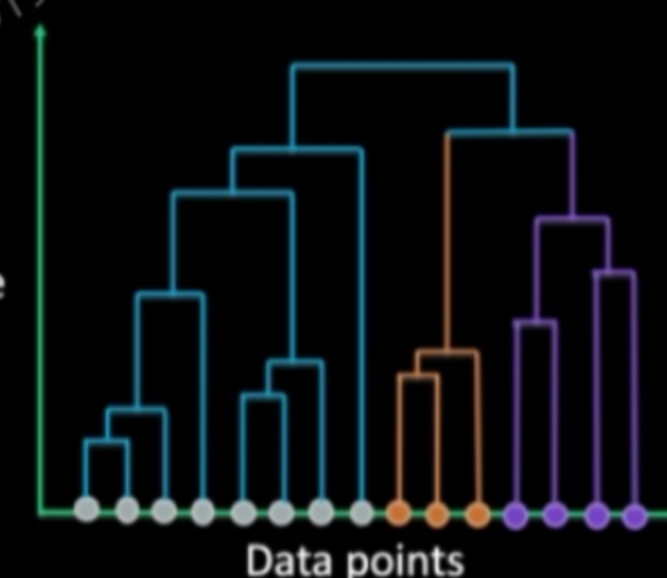
To get k clusters, we "cut" the dendrogram at some height.



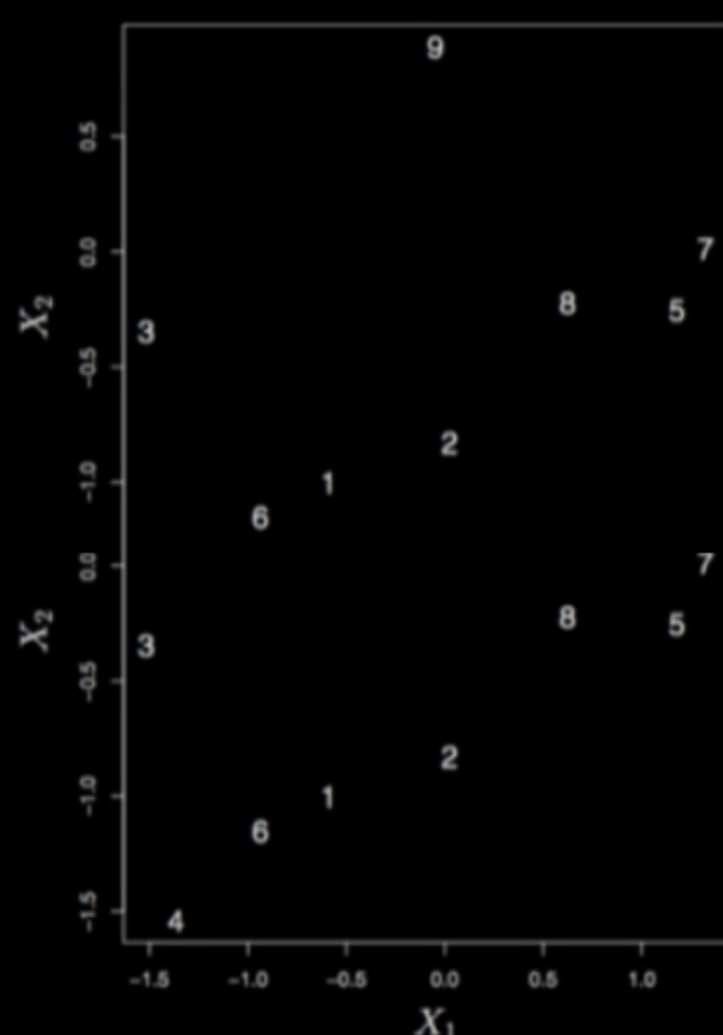
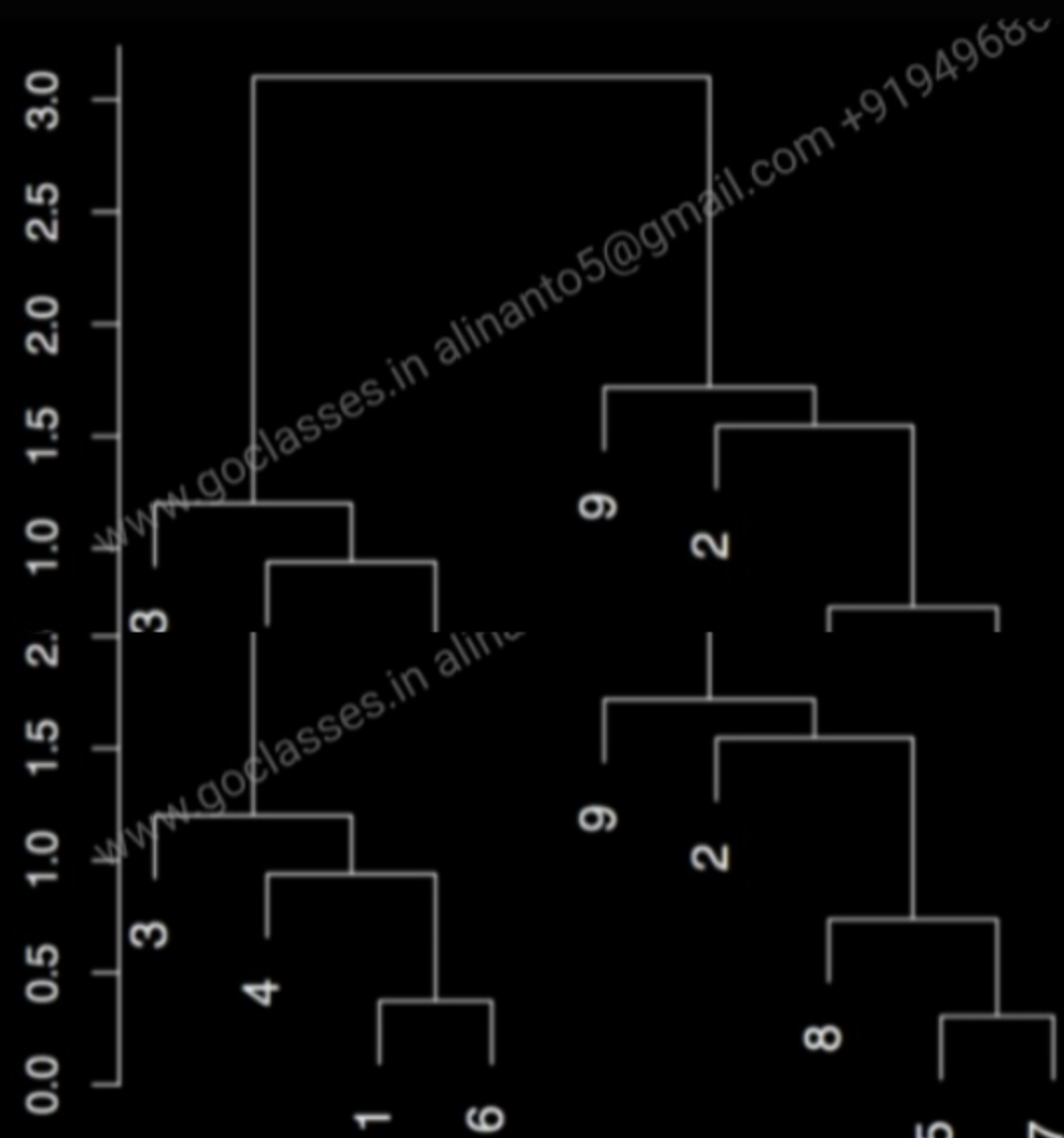
y-axis shows distance between pairs of clusters

Cluster distance

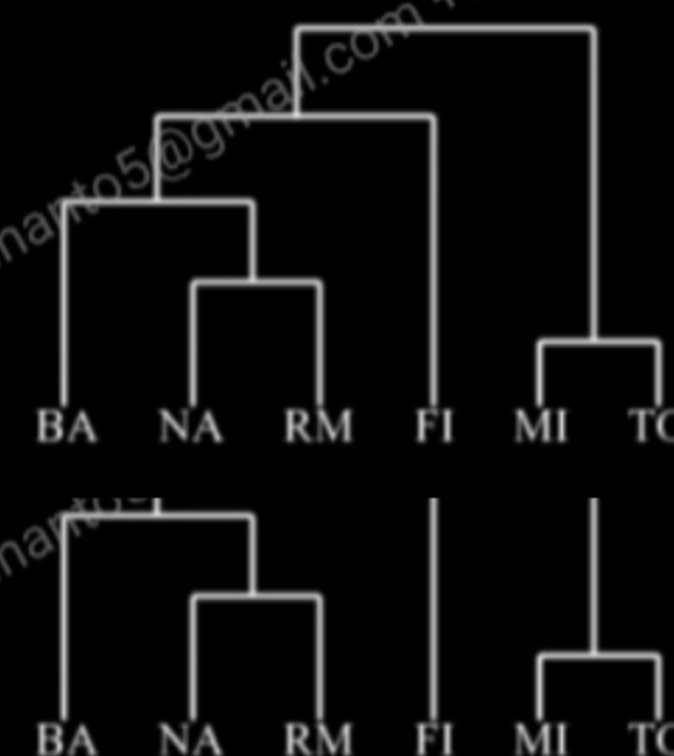
Based on the linkage we are using



Example



Consider the dendrogram:



Using this dendrogram to create 3 clusters, what would the clusters be?

- A. $\{BA, NA\}, \{RM, FI\}, \{MI, TO\}$
- B. $\{NA, RM\}, \{BA, FI\}, \{MI, TO\}$
- C. $\{BA, NA, RM, FI\}, \{MI\}, \{TO\}$
- D. $\{BA, NA, RM\}, \{FI\}, \{MI, TO\}$ ✓
- E. None of these