

Machine Learning (Sachin Mittal) go classes

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.

- Exhaustive Syllabus, covered in IISc and Sem Mtech
 - Also helpful for fresher level interview
 - Weightage is 15-20 Marks
- Book Recommendation
→ Bishop,
Pattern Recognition & Machine Learning.

Training data →

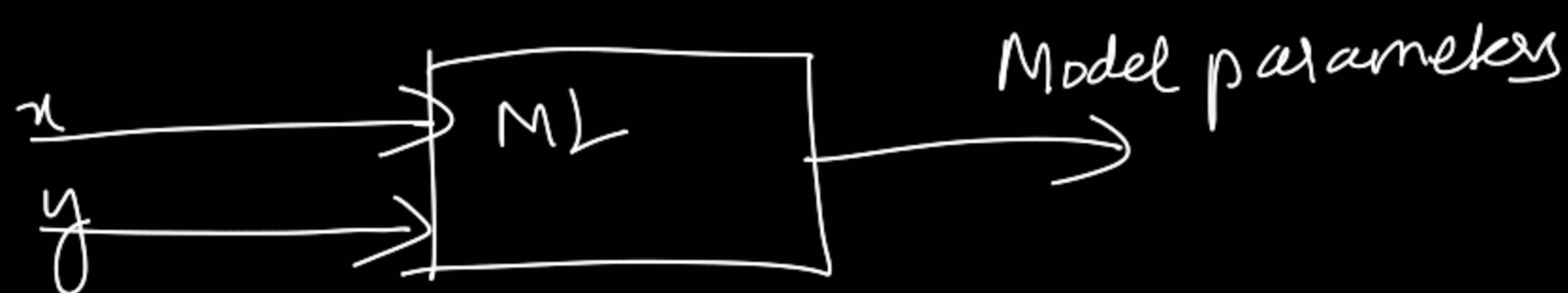
Testing data →

Machine learning → Learning parameters from training data using a model that relates input to output.

Book Recommendation → The Elements of Statistical Learning
ISLR



ML comes up with rule f from training data (x_i, y_i)

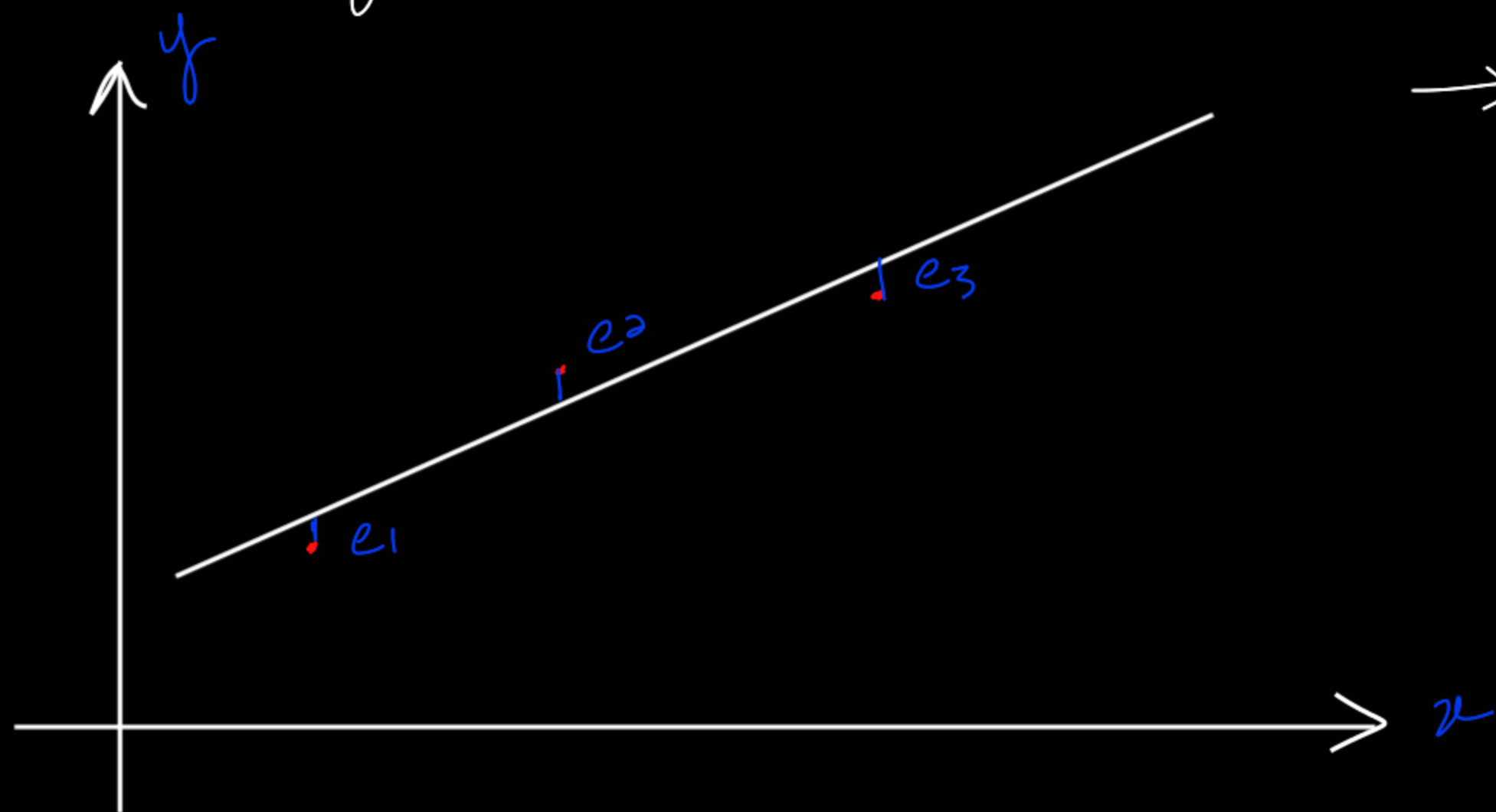


$f \rightarrow$ hypothesis

Classification Based on Data

- Supervised Learning → We have (x, y) pairs
- Unsupervised Learning → We don't have y .
- Reinforced Learning

Minimising error Least square Vs Modulus



→ error between prediction & actual point to be minimised
ie Minimise $|e_1| + |e_2| + |e_3|$
or Minimise $e_1^2 + e_2^2 + e_3^2$

→ Least square fits better for computers since completely differentiable

→ Modulus is not completely differentiable.

→ Also Minimising least square has a probabilistic interpretation in that it assumes the error is normally distributed

In two dimensions

$$f(a,b) = \sum_{i=1}^n (ax_i + b - y_i)^2$$

Minimise $f(a,b)$

→ projection → LA
differentiation → calculus

$n \rightarrow$ # datapoints in training data

$a, b \rightarrow$ parameters of model

$x_i \rightarrow$ i^{th} input

$y_i \rightarrow$ i^{th} o/p

$f(a,b) \rightarrow$ Loss function

→ These approaches work for small dimensional data, for large dimensions, we need to use gradient descent.

Using calculus, to find the minima, just take the first derivative & equate

$$\sum_{i=1}^n (ax_i + b - y_i)^2$$

$(Ax - y)^T (Ax - y)$ } scalar $\rightarrow$ sum of least squares

$A \rightarrow$ data

$y \rightarrow$ ground truth

$x \rightarrow$ parameter vector

$$A = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$x = \begin{bmatrix} a \\ b \end{bmatrix}$$

b - Not a multiple of input data
this is called bias.

→ We need to minimise

$$(Ax - y)^T (Ax - y) = \|Ax - y\|^2$$

(L2 Norm)

Q.)

$$y = c_1 x_1 + c_2 x_2 + c_3$$

$\hat{y} =$ prediction

$(y - \hat{y})^T (y - \hat{y}) \rightarrow$ minimise this w.r.t c_1, c_2, c_3

$$y = \begin{bmatrix} x_1 & x_2 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$A \quad x$

minimise $L(x)$ w.r.t x
scalars vector

$$\Rightarrow (Ax - y)^T (Ax - y) = L(x)$$

$x \rightarrow$ parameter vector

→ $L(x) \rightarrow$ Loss function ★

→ We want to differentiate scalar $L(x)$ w.r.t x which is a vector

→ $\frac{\partial f}{\partial x}$ → both f & x are scalar if $f(x) =$

$\begin{matrix} y \\ x \end{matrix}$	Scalar	Vector	Matrix
Scalar	$\frac{\partial y}{\partial x}$	$\frac{\partial y}{\partial x}$	$\frac{\partial y}{\partial x}$
Vector	$\frac{\partial y}{\partial x}$		
Matrix	$\frac{\partial y}{\partial x}$		

→ Derivative of y -scalar by vector x is given

$$\frac{\partial y}{\partial x} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \frac{\partial y}{\partial x_3} & \dots & \frac{\partial y}{\partial x_n} \end{bmatrix}$$

→ We can write the result as a row vector or column vector.

→ This is also called gradient.

Gradient: of differentiable function $f(x): \mathbb{R}^k \rightarrow \mathbb{R}$ w.r.t to its vector

argument $x = \nabla f(x)$

$$\nabla f(x) \triangleq \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_k} \end{bmatrix} \in \mathbb{R}^k$$

This convention is also fine but we are not taking.

We are taking the transpose of this

Q) Consider $f(x) = x^T b$ $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ & $b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$ find $\nabla f(x)$

Sol

$$\frac{\partial f}{\partial x} = \nabla f(x) = \begin{bmatrix} b_1 & b_2 & \dots & b_n \end{bmatrix} = b^T$$

Q) $f(x) = x^T x$ find $\nabla f(x) \rightarrow \begin{bmatrix} 2x_1 & 2x_2 \end{bmatrix} = 2x^T$
 $f(x) = x_1^2 + x_2^2$

if y is a vector & x is a scalar

then $\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x} \\ \vdots \\ \frac{\partial y_n}{\partial x} \end{bmatrix}$

if x, y are both vector in $\mathbb{R}^n$

then $\frac{\partial y}{\partial x} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \dots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \dots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \dots & \frac{\partial y_n}{\partial x_n} \end{bmatrix}$

FiViR

Function Increases Vertical; Variable Increases Right

Q) $Ax = f(x)$

$A = m \times n$
 $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$

differentiate Ax w.r.t x

$$\frac{\partial(Ax)}{\partial x} = \begin{bmatrix} \frac{\partial(a_{11}x_1 + a_{12}x_2)}{\partial x_1} & \frac{\partial(a_{11}x_1 + a_{12}x_2)}{\partial x_2} \\ \frac{\partial(a_{21}x_1 + a_{22}x_2)}{\partial x_1} & \frac{\partial(a_{21}x_1 + a_{22}x_2)}{\partial x_2} \end{bmatrix}$$

$$\frac{\partial(Ax)}{\partial x} = A$$

④ diff $(A^T x)$ wrt $x \rightarrow \frac{\partial (A^T x)}{\partial x} = A^T$ x is a vector.

y	$\frac{\partial y}{\partial x}$
	-
$A^T x$	A^T
$x^T A$	A^T
$x^T x$	$2x^T$
$x^T b$	b^T

☆☆☆☆☆

Results using FiVViR
Convention: otherwise transpose
the results.

$$\frac{\partial f(g(u))}{\partial x} = f'(g(x)) \cdot g'(u)$$

a) $f(x) = (Ax)^T Ax$ } use chain rule

$$f'(x) = \frac{\partial (Ax)^T}{\partial x} \cdot A$$

$$f'(x) = 2x^T A^T A$$

a) $f(x) = (Ax - y)^T (Ax - y)$

$$f'(x) = \frac{\partial (Ax - y)^T}{\partial x} \cdot A = \frac{\partial (x^T A^T - y^T)}{\partial x} \cdot A$$

now if $f'(x) = 0 \Rightarrow x^T A^T A - y^T A = 0$

$$\Rightarrow x^T A^T A = y^T A$$

or take
transpose
of whole
eq

$$\Rightarrow A^T A x = A^T y$$

Same result from
Linear algebra
for projections.

$$\nabla L(x) = 2A^T(Ax - b)$$

→ Instead of solving $Ax = b$
→ we are solving $A^T A x = A^T b$

⑤ find the eq. for the least sq. line for the data $(1,3)(2,2)(4,1)(5,1)$
 $y = \text{cost}_1 x$

$$Ax = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

$$\Rightarrow A^T A x = A^T b \Rightarrow$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 4 & 1 \\ 5 & 1 \end{bmatrix} = \begin{bmatrix} 46 & 12 \\ 12 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 16 \\ 7 \end{bmatrix}$$

$$= 10c_1 + 0c_0 = -5$$

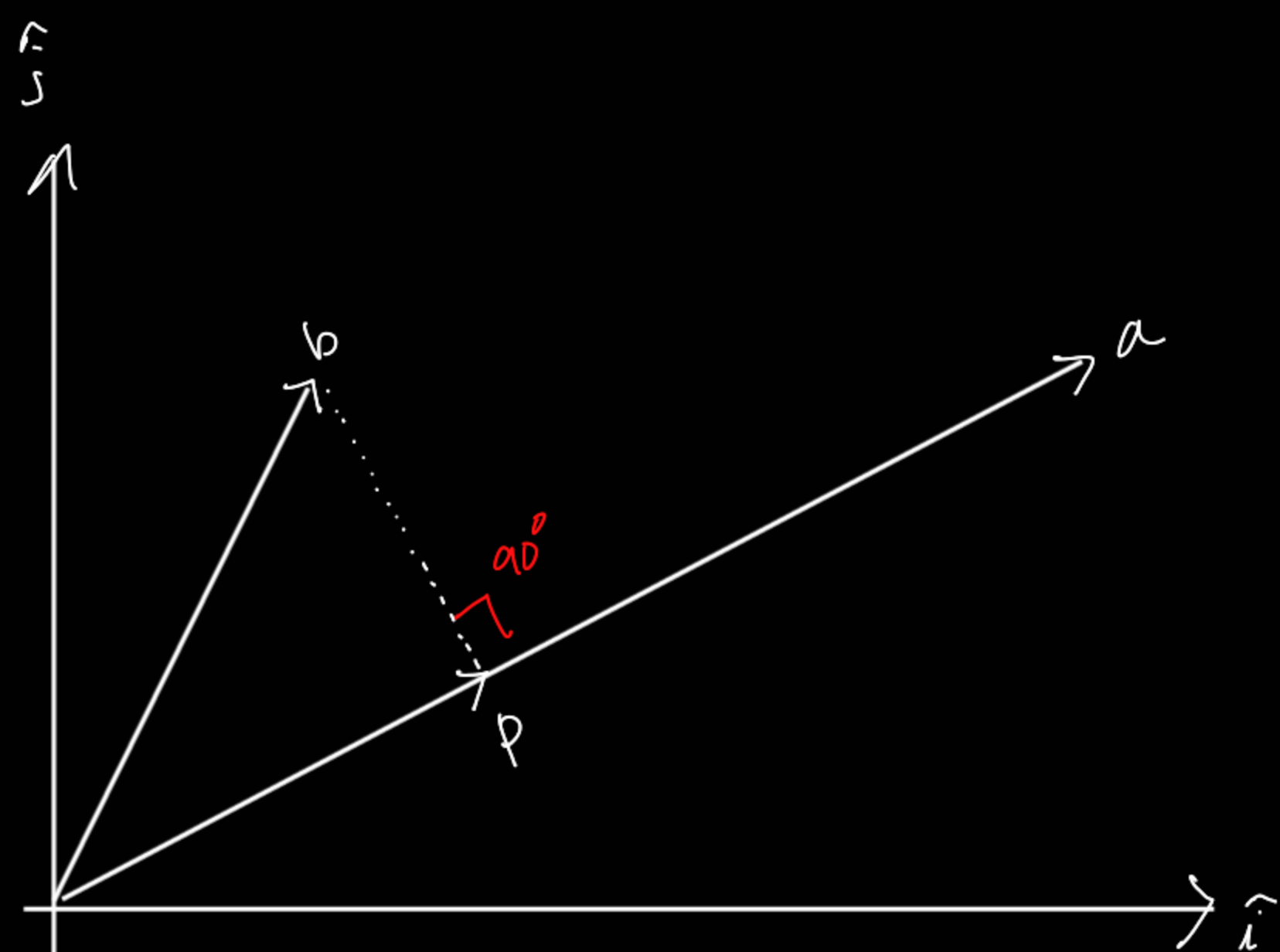
$$\Rightarrow c_1 = -1/2$$

$$-6 + 4c_0 = 7$$

$$c_0 = 13/4$$

Solution $y = -x/2 + 13/4$

Linear Algebra approach



$$p = \kappa a$$

$$b - p \perp a$$

$$\Rightarrow a^T (b - \kappa a) = 0$$

$$\Rightarrow \kappa = \frac{a^T b}{a^T a} \quad p = \frac{a a^T b}{a^T a}$$

$p \rightarrow$ projection

if a is unit vector,

$$\text{then } a^T a = 1 \Rightarrow$$

$$\kappa = a^T b$$

$$p = \hat{a} \kappa = \hat{a} a^T b$$

Q) True or false? If a is unit vector then $a^T b$ is the projection of b on a ? — **False**

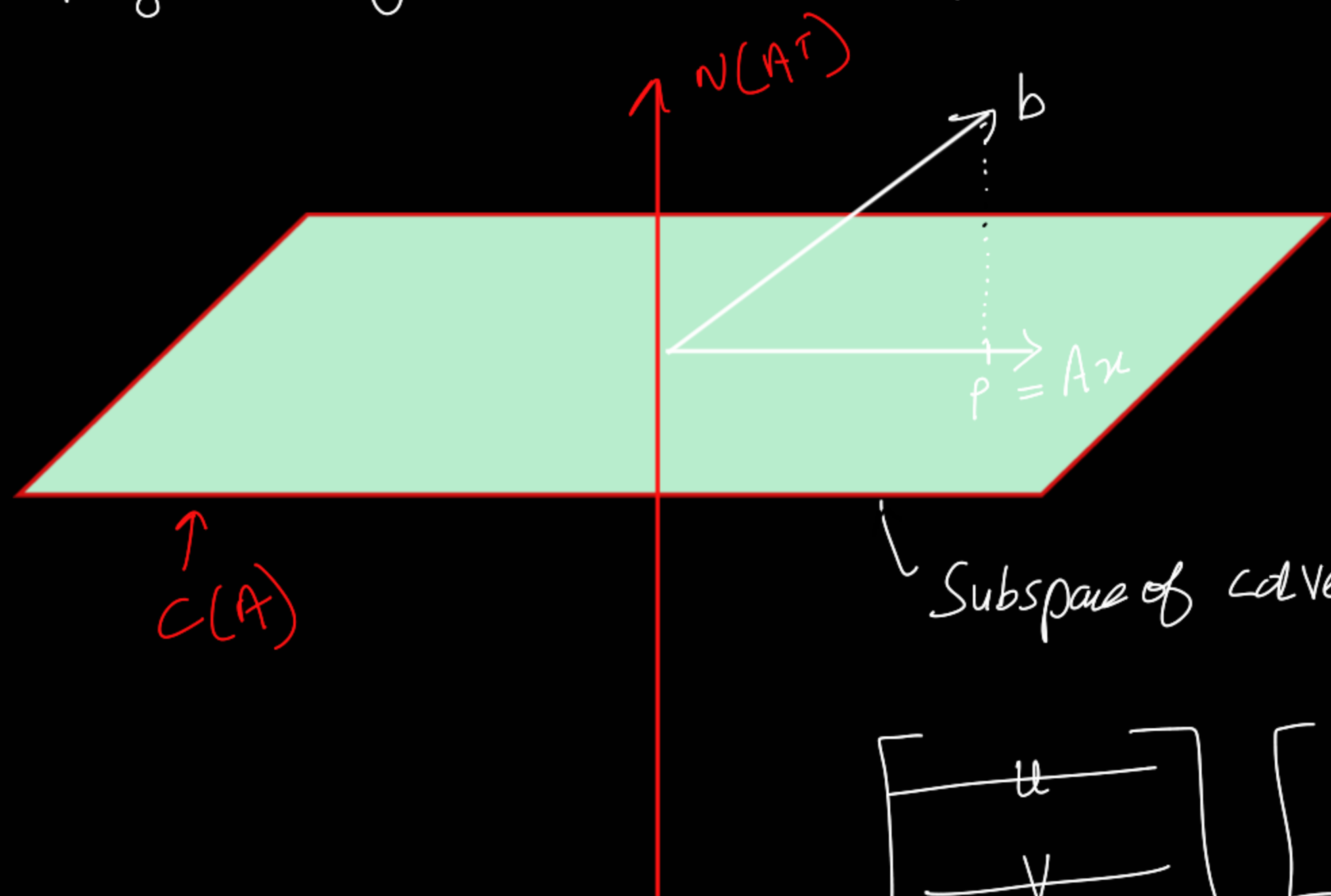
Q) True or false? If a is unit vector then $a^T b$ is the len of proj of b on a ? — **True**

Q) Find projection of b onto a and also compute it from $p = \hat{a} a$.

$$b = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{and} \quad a = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Solution:- $p = \kappa \hat{a}$; $\kappa = \frac{a^T b}{a^T a} \Rightarrow \kappa = 0 \Rightarrow p = 0$

Projection of vector onto subspace



$p = Ax$ where A is the matrix with column vectors as basis

$$\Rightarrow b - Ax$$

$$A^T (Ax - b) = 0$$

$$\begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} \downarrow & \downarrow \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \end{bmatrix}$$

$\rightarrow$ Using b we cannot solve for $Ax = b$, since b is not in the vector span of columns of A . So I am instead projecting b into the column space of A & then solve for $A^T A x = A^T b$; $p = A(A^T A)^{-1} A^T b$

$$P = A(A^T A)^{-1} A^T \rightarrow \text{Projection Matrix}$$

$$p = P b$$

$$\rightarrow P^n = P$$

$$\rightarrow P^T = P$$

$$\rightarrow \text{Eigen values } 0, 1$$

$$\rightarrow \text{Eigen vectors} \rightarrow \text{basis of column space, } N(A^T)$$

$$\rightarrow \text{for orthonormal basis; } P = A A^T \stackrel{\text{square } A}{=} I$$

$$\rightarrow \text{determinant } |P| = 0 \text{ unless } P = I \text{ (in square case)}$$

Since components in the $N(A^T)$ cannot be recovered

Q) Prove that for any arbitrary matrix A & B : $\text{Col}(AB) \subseteq \text{Col}(A)$

solution, for this we need to prove for any vector $x \in \text{Col}(AB) \Rightarrow x \in \text{Col}(A)$

$$\Rightarrow V = ABx$$

$$V = Ay$$

$$\Rightarrow V \text{ is in } C(A)$$

Column space of AB is the subset of column space of A
Null space of B is the subspace of Nullspace of AB

Q) Prove that $\text{Rank } A = \text{Rank}(A^T A)$ for any A

Soln

$$\text{if } x \in N(A) \Rightarrow Ax = 0$$

$$\Rightarrow A^T A x = 0$$

$$\Rightarrow x \in N(A^T A) \Rightarrow N(A) \subseteq N(A^T A) \text{ --- (1)}$$

$$\text{similarly if } x \in N(A^T A) \Rightarrow A^T A x = 0$$

$$\Rightarrow x^T A^T A x = 0$$

$$\Rightarrow (Ax)^T (Ax) = 0$$

$$\Rightarrow (Ax) = 0$$

$$\Rightarrow x \in N(A) \Rightarrow N(A^T A) \subseteq N(A) \text{ --- (2)}$$

$$\text{from (1) \& (2) } \Rightarrow N(A) = N(A^T A)$$

$$\Rightarrow \dim(N(A)) = \dim(N(A^T A))$$

$$\Rightarrow \text{Rank}(A) = \text{Rank}(A^T A)$$

$$\left. \begin{array}{l} \text{if } A = m \times n \\ \text{the } A^T A = n \times n \\ d(N(A)) = n - r \\ d(N(A^T A)) = n - r \end{array} \right\} \Rightarrow \underline{r = r}$$

Similar argument can be made for $\text{Rank}(A^T) = \text{Rank}(AA^T) = \text{Rank}(A)$
 using nullspace of (A^T) . i.e. $N(A^T) = N(AA^T)$

$$\Rightarrow \text{Rank of } A = \text{Rank of } A^T = \text{Rank of } A^T A = \text{Rank of } A A^T$$

$A^T A$ is invertible when column of A are linearly independent
 $A A^T$ is invertible when rows of A are linearly independent

Q Prove that $A^T A x = A^T b$ is always consistent

Soln We need to show $C(A^T A)$ is the column space of $A^T A$

$$\text{if } y \in C(A^T A) \Rightarrow y = A^T A x$$

$$\Rightarrow y = A^T (A x) \quad A x = v$$

$$y = A^T v$$

$$\Rightarrow y \in C(A^T) \Rightarrow C(A^T A) \subseteq C(A^T)$$

$$\text{further } R(A^T A) = R(A^T) + C(A^T A) \subseteq C(A^T)$$

$$C(A^T A) = C(A^T)$$

$$\Rightarrow \underbrace{A^T A x}_{\text{(Column space of } A^T A)} = \underbrace{A^T b}_{\text{(Column space of } A^T)} \quad \left. \begin{array}{l} \text{Since RHS is in the } C(A^T A) \\ \Rightarrow \text{it is always solvable} \\ \text{System is consistent} \end{array} \right\}$$

ML Problem Types

→ Based on data → Supervised, Unsupervised, Reinforcement
 eg. Linear regression → clustering with labels

→ Based on o/p → Regression, classification

→ Based on Type of Model → Generative, Discriminative

→ Probabilistic model that learn joint $P(x, y)$ or then prior & likelihood $P(y) \cdot P(x|y)$ to maximise use Bayes rule to maximise $P(y|x)$

→ Directly learns parameter of discriminative hyperplane or DB

→ Two types of Supervised Learning → Regression (output continuous domain)
 → Classification (output discrete)

→ We cannot treat a regression problem with classification algorithm or viceversa. This is because regression assumes inherent linearity between inputs & outputs. The output themselves have linear relation which is not true in classification

team A win } classification problem
team B win }

Convert to linear problem with output range 0,1

→ After learning there exists a linear relation in regression o/p/s which is not the case in classification

Setup

Given m points (pair of numbers)
 $\{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$

→ Task is to determine a model aka a function $f(x)$ of a simple form such that

$$\left. \begin{array}{l} f(x_1) \approx y_1 \\ f(x_2) \approx y_2 \end{array} \right\} f(x_m) \approx y_m$$

→ Strategy → ① Divide data into ^{practice quiz} training set & ^{midterm} test set
② Learn the function parameters using Loss function.
③ We test the model performance on Test set

→ Rule in Machine Learning → you cannot use the test data for training.
→ Midterm analogy → cheating on test
→ Paper leak jaisa bath he

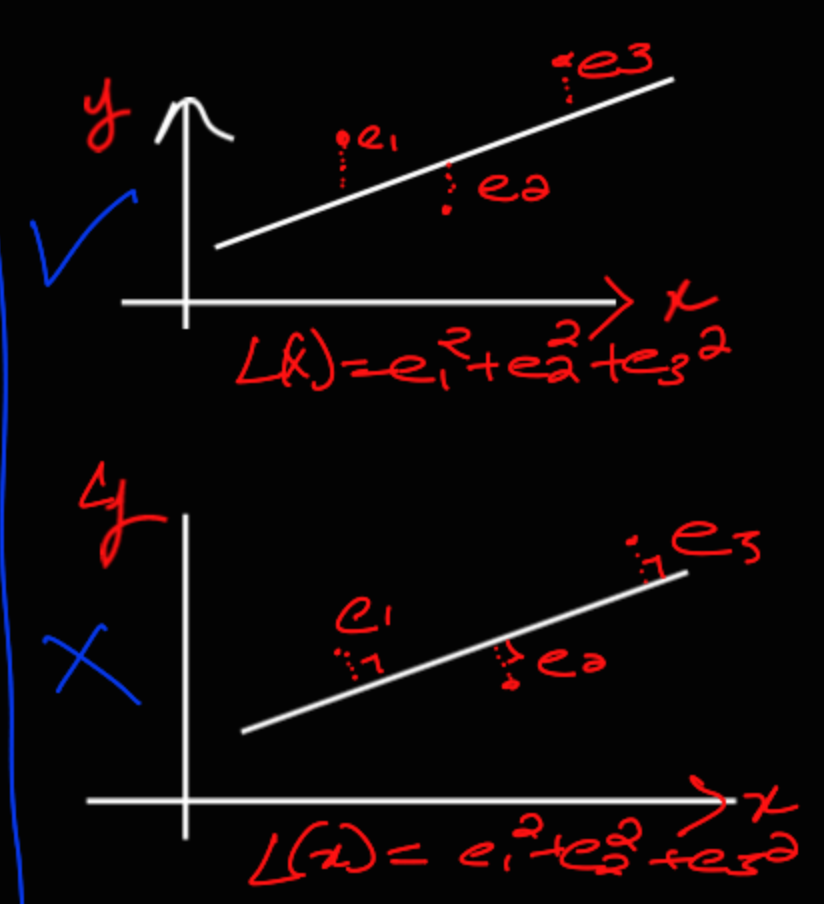
→ Sometimes we need validation set $\subseteq$ training set for tuning hyperparameters
parameters that is not learned by the model but the user has to set himself

Simple Linear Regression

Assumptions:- Relation btw x & y is linear

Model:- ① $\sum_{i=1}^n (y - \hat{y})^2$ → Loss function → Least sq. errors
 n → no. of datapoints
 y — actual o/p
 $\hat{y}$ — prediction
 $L(\beta) = \sum_{i=1}^n (y - \hat{y})^2$

Side note



② Hypothesis or model function $f(x) = A\beta$
 $\hat{y} = f(x) =$

$$\hat{y} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_n \end{bmatrix}$$

$A \rightarrow \text{data} = n \times m$
 $\beta \rightarrow \text{parameters}$
 $n \rightarrow \# \text{data points}$
 $m \rightarrow \# \text{features} + 1$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \end{bmatrix}$$

Objective to learn $\rightarrow$ Minimise Loss function w.r.t β .

$\rightarrow$ The value of β that minimises $L(\beta) = \beta^* \rightarrow$ learned parameters

Q)

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_0 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 2 \end{bmatrix}$$

Solution $A^T A \theta^* = A^T y$

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

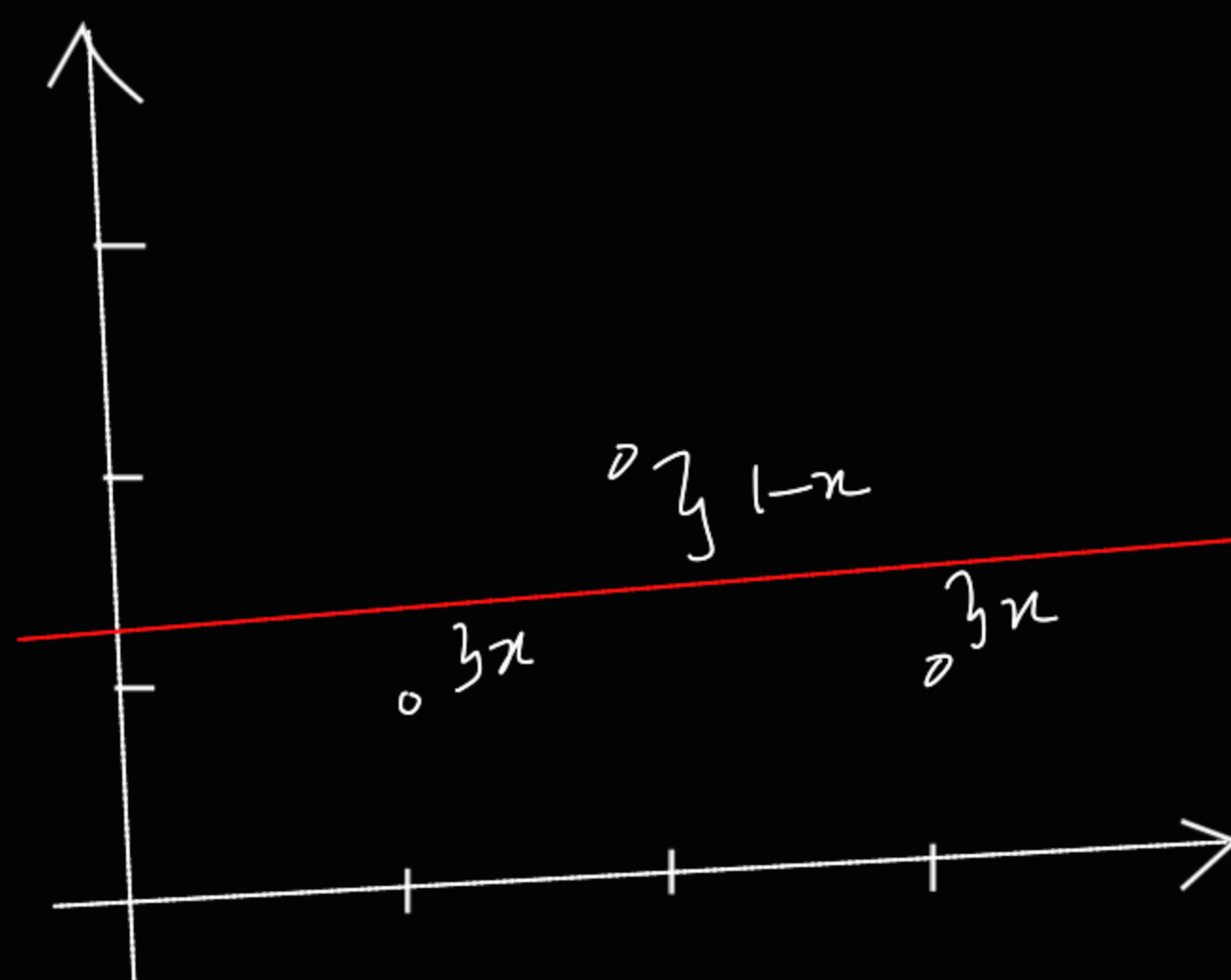
$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 2 \\ 3 & 1 & 4 \\ 3 & 1 & 2 \end{bmatrix}$$

$$A A^T \quad A^T y = \begin{bmatrix} 23 & 9 \\ 9 & 4 \end{bmatrix} \quad \begin{bmatrix} 23 \\ 9 \end{bmatrix}$$

$$\Rightarrow \begin{matrix} \theta_1^* = 1 \\ \theta_0^* = 0 \end{matrix}$$

x	y
1	1
2	2
3	1

Find MSE & SSE $\rightarrow$ Sum of squared error.
 $\rightarrow$ Mean squared error



Solve $MSE = \frac{x^2 + x^2 + (1-x)^2}{3} = L(x)$ $\nabla L(x) = 0 \Rightarrow x = 1/3$
 $MSE = 2/9$

Q. Dataset with N samples: $\{(x_i, y_i) \mid 1 \leq i \leq N\}$ $x_i, y_i \in \mathbb{R}$ for all i

data model $\hat{y}_i = w_0 + w_1 x_i$

Assuming least square has unique sol, which is true

$$L(w) = \sum_{i=1}^N (y_i - w_0 - w_1 x_i)^2$$

differentiating w.r.t w_0

$$L(w) = \sum_{i=1}^N \left[y_i^2 + w_0^2 + w_1^2 x_i^2 - 2y_i w_1 x_i + -2y_i w_0 + 2w_0 w_1 x_i \right]$$

$$\frac{\partial L(w)}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^N [2w_0 - 2y_i + 2w_1 x_i] = 0$$

$$\frac{\partial L(w)}{\partial w_1} = 0 \Rightarrow \sum_{i=1}^N [2w_1 x_i^2 - 2y_i x_i + 2w_0 x_i] = 0$$

$$\frac{\partial L(w)}{\partial w_0} = 0 \Rightarrow \sum_{i=1}^N \underbrace{[y_i - w_0 - w_1 x_i]}_{\text{residual/error}} = 0 \quad \frac{\partial L(w)}{\partial w_1} = 0 \Rightarrow \sum_{i=1}^N \underbrace{[y_i - w_0 - w_1 x_i]}_{\text{error}} x_i = 0$$

→ Intuition

- Summation of errors = 0 (only if intercept term is present)
- error is not correlated with input x_i / i.e w.r.t input, the error is completely random w.r.t input / input vector $\perp$ error vector

→ This means that in the 2-dimensional case asking for best line, we can just try to get the signed error sum to zero.

Locally weighted Linear Regression

→

$$\{w_0^*, w_1^*\} = \arg \min_{\{w_0, w_1\}} \sum_{i=1}^N \alpha_i^2 (y_i - w_0 - w_1 x_i)^2$$

Q) We want to estimate y | y is modelled as $y = \theta \sqrt{x}$ $\theta \rightarrow$ parameter

y	x
2	9
1	4
0	1

Find $\hat{\theta}$ that minimises MSE

$$L(\theta) = \sum_{i=1}^3 (y_i - \theta \sqrt{x_i})^2$$

$$\frac{\partial L(\theta)}{\partial \theta} = 0 \Rightarrow \sum_{i=1}^3 [2(y_i - \theta \sqrt{x_i}) \cdot -\sqrt{x_i}] = 0$$

$$\Rightarrow (2 - \theta \cdot 3) \cdot 3 + (1 - \theta \cdot 2) \cdot 2 + (0 - \theta \cdot 1) \cdot 1 = 0$$

$$\Rightarrow 6 - 9\theta + 2 - 4\theta + 0 - \theta = 0$$

$$\Rightarrow 8 = 14\theta \Rightarrow \theta = \frac{8}{14} = \underline{\underline{\frac{4}{7}}}$$

Side note
optimum value
 $\hat{\theta}$ or θ^*
people use
these notation

or we can use $t = \sqrt{x}$ & convert to L.R.

$$y = \theta t$$

y	t
2	3
1	2
0	1

$\Rightarrow$ Sum of errors shortcut is **not** applicable since there is no constant
no term in model

$$\Rightarrow (2 - \theta) \cdot 3 + (1 - 2\theta) \cdot 2 + (0 - \theta) \cdot 1 = 0$$

$$\Rightarrow 6 + 2 = 9\theta + 4\theta + \theta \Rightarrow \theta = 8/14 = \underline{\underline{\frac{4}{7}}}$$

Q. Model function $y_i = \omega^2 x_i + \omega x_i$ Derive a solution for ω

$$y_i = x_i(\omega^2 + \omega)$$

$$\text{Let } \theta = \omega^2 + \omega$$

$$\text{Solving for } \nabla L(\theta) = 0 \Rightarrow \sum_{i=1}^N (y_i - \theta x_i) x_i = 0$$

$$= \sum_{i=1}^N y_i x_i - \theta \sum_{i=1}^N x_i^2 = 0$$

$$= \sum_{i=1}^N y_i x_i = \theta \sum_{i=1}^N x_i^2$$

$$\Rightarrow \theta = \frac{\sum_{i=1}^N y_i x_i}{\sum_{i=1}^N x_i^2}$$

Side Note

$\rightarrow$ This shortcut method could have been used in the previous Q.

$$\text{eg } \theta = \frac{2 \times 3 + 1 \times 2 + 0 \times 1}{3^2 + 2^2 + 1^2} = \underline{\underline{\frac{8}{14}}}$$

$$\text{But } \theta = \omega^2 + \omega$$

Using Completing square

$$\theta + 1/4 = \omega^2 + \omega + 1/4 = (\omega + 1/2)^2$$

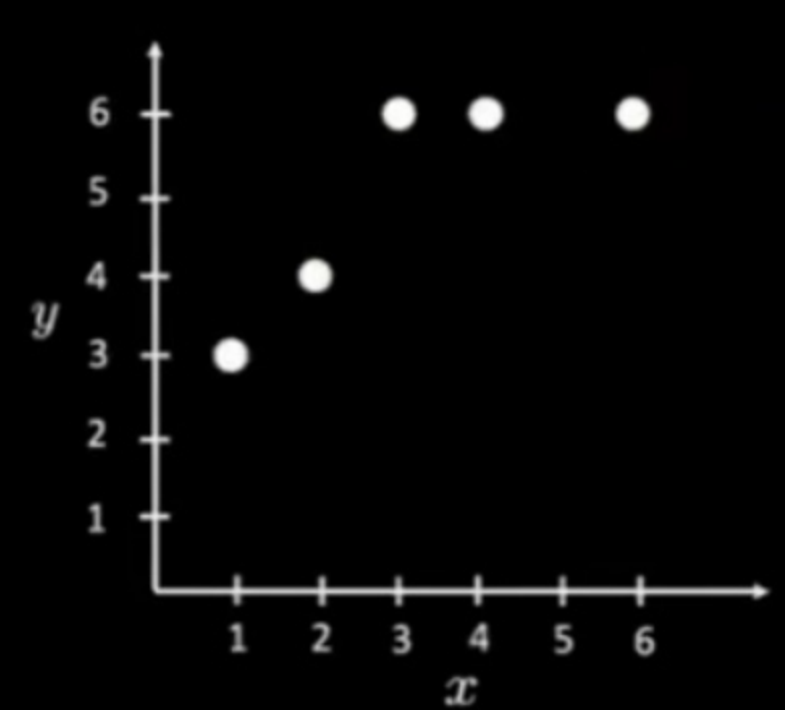
$$\Rightarrow \omega + 1/2 = \pm \sqrt{\theta + 1/4}$$

$$\omega = \pm \sqrt{\theta + 1/4} - 1/2$$

$$\Rightarrow \omega = \sqrt{\frac{1}{4} + \frac{\sum_{i=1}^N y_i x_i}{\sum_{i=1}^N x_i^2}} - \left(\frac{1}{2}\right)$$

Q)

Consider the following 1-D data set:



(a) What is the mean-squared error (MSE) on this data for the hypothesis $\hat{y}_i = 2x_i$?

(b) Beatriz decides that for her application, small errors in the predicted y -values are irrelevant. She designs a new loss function $\mathcal{L}(\hat{y}, y)$ which is 0 if $y - 2 \leq \hat{y} \leq y + 2$, and $(|y - \hat{y}| - 2)^2$ otherwise.

$$\mathcal{L}(\hat{y}, y) = \begin{cases} 0, & \text{if } y - 2 \leq \hat{y} \leq y + 2 \\ (|y - \hat{y}| - 2)^2, & \text{otherwise} \end{cases}$$

What is the average loss using $\mathcal{L}$ on the same data set as the previous question, assuming again the hypothesis $\hat{y}_i = 2x_i$?

a) $MSE = \mathcal{L}(x) = \frac{1}{n} \sum_{i=1}^n (y - \hat{y})^2 = \frac{1}{5} (3-2)^2 + (4-4)^2 + (6-6)^2 + (6-6)^2 + (6-12)^2$

Model $\hat{y}_i = 2x_i$

$$= \frac{1}{5} [1 + 0 + 0 + 0 + 36] = \frac{1}{5} \cdot 41 = \frac{41}{5}$$

b) MSE after changing loss function $MSE = \frac{1}{5} [0 + 0 + 0 + 0 + 16] = \frac{16}{5}$

Consider a modification of the standard linear regression setup where each datapoint is associated with an importance weight. Formally, we have a training dataset consisting of n examples $\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$, where each datapoint is associated with an importance weight $r_i > 0$. The weighted residual sum of squares objective is defined as

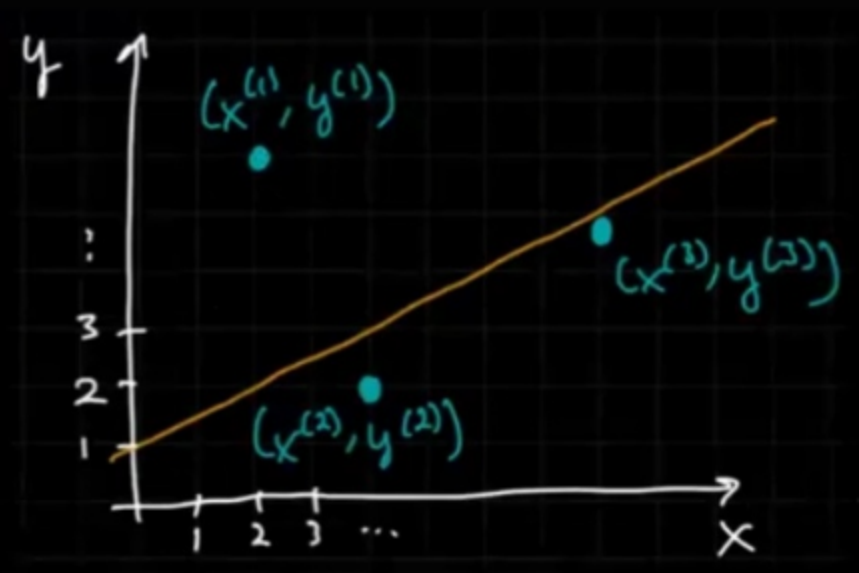
definition $WRSS(w) = \sum_{i=1}^n r_i (w x^{(i)} - y^{(i)})^2$.

The diagram below shows a training dataset with three examples $(x^{(1)}, y^{(1)})$, $(x^{(2)}, y^{(2)})$, and $(x^{(3)}, y^{(3)})$.

Here the inputs x are all one-dimensional. Soumya chooses some importance weights r_1, r_2 , and r_3 and then trains a model to minimize the WRSS objective. The model's predictions are illustrated by the orange line. (Please check image on next slide)

- (2 points) What are the values of w and b that Soumya learned?
- (4 points) Out of the values for r_1, r_2 , and r_3 that Soumya chose, which one was the smallest? Explain your reasoning.

$$WRSS(w) = \sum_{i=1}^n r_i (w x^{(i)} - y^{(i)})^2$$



- (a)
- What are the values of w and b that Soumya learned?
 - Out of the values for r_1, r_2 , and r_3 that Soumya chose, which one was the smallest? Explain your reasoning.

(c) Let w^* be any value of w that achieves the lowest possible value of the standard linear regression loss discussed in class. Let $\tilde{w}$ be any value of w that achieves the lowest possible WRSS loss when $r_i = 10$ for all i .

- Assume that the Normal Equation has a unique solution. Are w^* and $\tilde{w}$ guaranteed to be the same? Explain your reasoning.
- Now assume that the Normal Equation has more than one solution. Does this change your answer? Explain your reasoning.

→ After differentiation, what we get is the normal equation $\frac{\partial \mathcal{L}}{\partial w} = 0$

or $A^T X = b$

Sol

i) from the graph, we can see $b=1 \rightarrow$ y-intercept @ $x=0$
slope = $1/2$

ii) (x_1, y_1) has least important example

Part b why $\lambda_i < 0$ will be disastrous?

Answer $\lambda_i < 1 \Rightarrow$ Minimising the loss function $\Rightarrow$ error becomes infinity

ci) w^* & $\tilde{w}$ will be same since constant will vanish during differentiation.

cii) Multiple sol $\Rightarrow$ the solution is not guaranteed to be same even if we run the regression twice on same data. It is not however because of the 10

$\rightarrow$ Unique solution $\iff A^T A x = A^T b \implies A^T A$ invertible $\implies A^T$ has independent columns
 $\implies A$ has independent rows.
 $\rightarrow$ When $\# \text{features} < \# \text{datapoint}$, which is generally the case, A does have independent columns $\implies A^T A$ is invertible $\& A^T A x = A^T b$ has unique solution for all b .

Q) Using monthly stock averages, $x_1, x_2, \dots, x_n$, you run linear regression without regularization to estimate some future stock value y . However, your test error rate turns out to be very high. Your friend suggests that maybe the price differences between consecutive months may offer better features, so you add additional features $(x_1 - x_2), (x_2 - x_3), (x_3 - x_4), \dots, (x_{n-1} - x_n)$. Do you expect an improvement? Briefly justify your answer.

No because, the new features are linear combination of existing features $\&$ the regressor would have learned the optimum anyway.

Q In class, we discussed how problems can arise when certain types of examples are underrepresented in the data. One natural solution is to *reweight* the data. Suppose we have a training dataset D for linear regression that is the union of two (disjoint) datasets A and B , where $|A| \gg |B|$. For example, A and B might represent data from two different groups of individuals. Let $(x_a^{(i)}, y_a^{(i)})$ denote the i -th example in A and $(x_b^{(i)}, y_b^{(i)})$ denote the i -th example in B . The reweighted training loss is defined as

$$L(w) = \frac{1}{|A|} \sum_{i=1}^{|A|} (w x_a^{(i)} - y_a^{(i)})^2 + \frac{1}{|B|} \sum_{i=1}^{|B|} (w x_b^{(i)} - y_b^{(i)})^2,$$

where w is the weight parameter for linear regression (in this problem, we will omit the bias term).

Explain why this loss function is more likely to promote more equal treatment of individuals in dataset A and individuals in dataset B , compared with running normal linear regression on D .

$$\rightarrow \frac{1}{|A|} \& \frac{1}{|B|}$$

acts as local weights

$$\rightarrow \text{Conventional L.R} \\ L(w) = \frac{1}{|A|+|B|} \sum_{i=1}^{|A|+|B|} [y_i - \hat{y}_i]^2$$

$\rightarrow$ In contrast data point from B are given more weightage since $|A| \gg |B| \implies \frac{1}{|A|} < \frac{1}{|B|}$

Let w^* be the solution you obtain in standard least-squares linear regression. What solution do you obtain if you scale all the input features (but not the labels y) by a factor of c before doing the regression?

- ☒ $\frac{1}{c} w^*$ ☐ $c w^*$
☐ $\frac{1}{c^2} w^*$ ☐ $c^2 w^*$

(4) (3 points) Recall, the original training loss used for standard least squares regression:

$$\text{Loss}(w) = \sum_{i=1}^n (y^{(i)} - w x^{(i)})^2$$

The individual datapoints are indexed by i , so that $x^{(i)}$ is the input and $y^{(i)}$ is the output for the i th datapoint in your training set of n datapoints. Suppose $x^{(i)} \in \mathbb{R}^d$ and $y \in \mathbb{R}$.

Suppose that your friend has proposed a new training loss to use for learning a regression model:

$$\text{NewLoss}(w) = \log \sum_{i=1}^n (y^{(i)} - w x^{(i)})^2$$

Suppose the dataset is fixed - i.e. you are going to use the same training data with both losses - and that you are able to minimize both losses exactly during training. Will these two objectives always yield the same regression model? Why or why not? (Briefly justify your answer.)

Solution: Yes, both objectives, when applied to the same dataset, will always have the same *argmin* and thus yield the same w . Notice that:

$$\text{NewLoss}(w) = \log(\text{Loss}(w))$$

Since $\log$ is a strict monotonic increasing function, the w that minimizes Loss also minimizes NewLoss , and vice versa.

$\rightarrow$ Side note $\rightarrow$ Taking logs makes minimisation easier if the loss function is a product of sticky functions e.g.
 $L(x) = \sin x \cdot x \cdot e^x \cdot \cos x$ $\log L(x) = \log \sin x + \log x + x + \log \cos x$
 $\rightarrow$ This trick is used in Maximum likelihood estimate.

Let's work out a two-way least-squares linear regression method. The i th observation is the ordered pair (x_i, y_i) . We're going to view these data in two ways:

1. x_i is a sample point in one dimension with label y_i , or
2. y_i is a sample point in one dimension with label x_i .

Since we do not use bias terms, both regression functions will pass through the origin. Our optimization problems are:

Find α that minimizes $\sum_{i=1}^n (\alpha x_i - y_i)^2$

Find β that minimizes $\sum_{i=1}^n (\beta y_i - x_i)^2$

(a) [7 pts]

Derive a closed-form expression for the optimal regression coefficient α . Express your final answer in terms of vector operations, rather than summations. Ensure you show all your work.

(b) [2 pts]

What is the closed-form expression for the optimal regression coefficient β ? This result follows from symmetry, so you don't need to repeat the derivation unless you'd like to.

$$a) \alpha = \frac{\sum_{i=1}^n y_i x_i}{\sum_{i=1}^n x_i^2}$$

$$\alpha = \frac{X^T Y}{X^T X}$$

$$b) \beta = \frac{Y^T X}{Y^T Y}$$

Note $\alpha \neq \frac{1}{\beta}$ after training

(2 pt) Kevin is trying to fit a simple linear regression model $\hat{y} = \theta_0 + \theta_1 x$ on a dataset with 900 observations.

After determining $\hat{\theta}_0$ and $\hat{\theta}_1$ by minimizing mean squared error, Kevin found the following:

- For 250 inputs, his predicted value $\hat{y}$ was 1 more than the actual value y .
- For another 500 inputs, his predicted value $\hat{y}$ was 0.5 less than the actual value y .
- All other points were predicted perfectly.

Calculate the **mean squared error** of Kevin's model. Select the closest answer. (Note that this question is not asking about mean nearest squared error.)

- ☐ 0
- ☐ 0.25
- ☐ 0.35
- ☒ 0.42
- ☐ 0.55
- ☐ 1

$$MSE = \frac{250 \cdot 1^2 + 500 \times (0.5)^2 + 0}{900}$$

$$= \frac{250 + 125}{900} = \frac{375}{900} = \frac{41.66}{100}$$

$$= 0.4166$$

Multiple Linear Regression

$$i \in \{1, N\} \quad \hat{y}_i = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \theta_3 x_{i3} + \dots + \theta_n x_{in}$$

$$\Rightarrow A\theta + b = \bar{y}$$

$\downarrow$ $\downarrow$ $\downarrow$ $\downarrow$
 $N \times n$ $n \times 1$ $N \times 1$ $N \times 1$

$n = \# \text{features}$

$$A = N \times n$$

$$\bar{y} = N \times 1$$

$\theta = n \times 1 \rightarrow \text{parameter vector}$

$N = \# \text{datapoints}$

$b = N \times 1 \rightarrow \text{Bias}$

$$\bar{b} = \begin{bmatrix} w_0 \\ w_0 \\ \vdots \\ w_0 \end{bmatrix}_{N \times 1}$$

Another notation

$$X\theta = \bar{y}$$

$\downarrow$ $\downarrow$ $\downarrow$
 $N \times n+1$ $n+1 \times 1$ $N \times 1$

$\Rightarrow$ This is being used throughout the literature

$\Rightarrow \theta$ or w are used interchangeably.

$$X\theta = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & & & \\ \vdots & \vdots & & & \\ 1 & x_{N1} & \dots & \dots & x_{Nn} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Least Square Estimate.

$$\bar{y} = Xw \Rightarrow \text{MSE} = \frac{1}{N} (y - \bar{y})^T (y - \bar{y}) \quad \text{Equivalent to} \quad \frac{1}{N} \sum_{i=1}^N (y_i - \bar{y}_i)^2 = \frac{1}{N} \sum_{i=1}^N (y_i - (w_0 + w_1 x_{i1} + \dots + w_n x_{in}))^2$$

scalars

$$\text{MSE} = \frac{1}{N} (y - Xw)^T (y - Xw) = L(w)$$

vectors

$$\nabla L(w) = 0 \Rightarrow \frac{1}{N} \cdot 2 \cdot (y - Xw)^T \cdot (-X) = 0$$

$$\Rightarrow (y - Xw)^T X = 0 \quad \Leftrightarrow \quad X^T (y - Xw) = 0$$

$$\Rightarrow X^T y = X^T X w \Rightarrow w = (X^T X)^{-1} X^T y$$

Reminder

$$\frac{\partial}{\partial x} (x^T x) = 2x^T$$

Normal equation

$X^T X$ invertible
when columns of
 X are linearly independent

$$\Rightarrow \text{Rank}(X^T X) = \text{Rank } X = n$$

i.e. $X \rightarrow$ full column rank
 $\Rightarrow X^T X \rightarrow$ invertible square
symmetric matrix

$\rightarrow$ Instead of solving $y = Xw$; we are solving $X^T y = X^T X w$
 $\rightarrow$ We would prefer $X^T X$ to be invertible for unique sol.
 $\Rightarrow$ we would prefer columns of X to be linearly independent to make $X^T X$ invertible & thus the solution unique

$\Rightarrow$ We want the features to be linearly independent as much as possible.

$\rightarrow$ Can we expect real world data to be linearly independent
Big YES $\rightarrow$ Since $N \gg n$ i.e. #datapoints $\gg$ #features. The matrix X will be very thin and long vertically. $A = \begin{bmatrix} \vdots \\ 1 \\ \vdots \end{bmatrix}$. Also even if we select features that have hidden linear relation. The sampling or measurement phase will be noisy & columns will be linearly independent

Q Suppose you have $m = 14$ training examples with $n = 3$ features (excluding the additional all-ones feature for the intercept term, which you should add). The normal equation is $\theta = (X^T X)^{-1} X^T y$. For the given values of m and n , what are the dimensions of θ , X , and y in this equation?

Your Answer

- ☒ X is 14×4 , y is 14×1 , θ is 4×1 ✓
- ☐ X is 14×3 , y is 14×1 , θ is 3×3
- ☐ X is 14×4 , y is 14×4 , θ is 4×4
- ☐ X is 14×3 , y is 14×1 , θ is 3×1

X is $N \times n+1 = 14 \times 4$
 θ is $(n+1) \times 1 = 4 \times 1$
 y is $N \times 1 = 14 \times 1$

Select one: The closed form solution for linear regression is $\theta = (X^T X)^{-1} X^T y$. Suppose you have $N = 35$ training examples and $M = 5$ features (excluding the bias term). Once the bias term is now included, what are the dimensions of X , y , θ in the closed form equation?

- ☐ X is 35×6 , y is 35×1 , θ is 6×1 ✓
☐ X is 35×6 , y is 35×6 , θ is 6×1
☐ X is 35×5 , y is 35×1 , θ is 5×1
☐ X is 35×5 , y is 35×5 , θ is 5×5

$$X = 35 \times 6$$

$$y = 35 \times 1$$

$$\theta = 6 \times 1$$

Suppose the scores of the midterm, homework, and final of 4 students are given by

Midterm	Homework	Final
80	85	96
72	68	74
94	97	90
69	70	78

You'd like to use linear regression to predict a student's final exam score from his/her midterm exam score and homework score. The normal equation of linear regression is $\theta = (X^T X)^{-1} X^T y$. What are X and y in this problem?

$$y = X\theta = \begin{bmatrix} 1 & 80 & 85 \\ 1 & 72 & 68 \\ 1 & 94 & 97 \\ 1 & 69 & 70 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 96 \\ 74 \\ 90 \\ 78 \end{bmatrix}$$

[0 pts] Given data $\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})\}$, we obtain $\hat{\mathbf{w}}$, the parameters that minimize the training error cost for the linear regression model $y = \mathbf{w}^T \mathbf{x}$ we learn from $\mathcal{D}$.

Consider a new dataset $\mathcal{D}_{\text{new}}$ generated by duplicating the points in $\mathcal{D}$ and adding 10 points that lie along $y = \hat{\mathbf{w}}^T \mathbf{x}$, for a total of $2N + 10$ points. Then the $\hat{\mathbf{w}}_{\text{new}}$ that we learn for $y = \mathbf{w}^T \mathbf{x}$ from $\mathcal{D}_{\text{new}}$ is equal to $\hat{\mathbf{w}}$.

- ☒ True
☐ False

We are not adding new different data points that change the minima $\rightarrow$ Adding the entire dataset again is as following

$$\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \xrightarrow{\text{becomes}} \left[\sum_{i=1}^N (y_i - \hat{y}_i)^2 + \sum_{i=N+1}^{2N+10} (y_i - \hat{y}_i)^2 \right] \frac{1}{2N+10}$$

Both terms are same

$$\Rightarrow \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \xrightarrow{\text{becomes}} \frac{1}{N+5} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

line of line = 0

Since the minima changes due to $N+5$, but the argmin does not change
 $\mathcal{D} \quad \mathbf{w}_{\text{new}}^* = \mathbf{w}^*$

3. (True or False) Assume that you have a dataset composed of N observations: the target $\mathbf{t}$ and features $\mathbf{X}$. You want to fit a linear regression model and find the weights $\mathbf{w}$, but you also know that more data is always helpful. Instead of fitting a model with $\mathbf{t} \in \mathbb{R}^n$ and $\mathbf{X} \in \mathbb{R}^{n \times d}$, you concatenate the data and fit a model using $\begin{bmatrix} \mathbf{t} \\ \mathbf{t} \end{bmatrix} \in \mathbb{R}^{2n}$ and $\begin{bmatrix} \mathbf{X} \\ \mathbf{X} \end{bmatrix} \in \mathbb{R}^{2n \times d}$. Running linear regression on this new dataset will give the same weights as on the original dataset.

Same reasoning as previous one
 $\rightarrow$ True

(a) (6 points) First, we use a linear regression method to model this data. To test our linear regressor, we choose at random some data records to be a training set, and choose at random some of the remaining records to be a test set. Now let us increase the training set size gradually. As the training set size increases, what do you expect will happen with the mean training and mean testing errors? (No explanation required)

- Mean Training Error: A. Increase; B. Decrease
- Mean Testing Error: A. Increase; B. Decrease ✓

For the remainder of this question, we will use the multiple linear regression model, which is of the form

$$\hat{y} = x \cdot \theta = \sum_{j=0}^p \theta_j x_j \quad \text{no } \theta_0 \text{ is bias}$$

As in class, assume:

- Y is a vector containing our observed responses (i.e. true y 's).
- X is a design matrix whose first column is 1 (i.e. $x_0 = 1$ for all observations), and X_i represents the i th row of X .
- We determine $\hat{\theta}$ by minimizing average squared loss.

$\hat{\theta}$ is the minimizer of which of the following quantities? Select all that apply.

- ☐ $\sum_{i=1}^n (y_i - \theta)^2$
- ☒ $\frac{1}{n} \sum_{i=1}^n (y_i - X_i \cdot \theta)^2$
- ☒ $\sum_{i=1}^n (y_i - X_i \cdot \theta)^2$
- ☒ $\frac{1}{n} \|Y - X\theta\|_2^2$
- ☒ $\|Y - X\theta\|_2^2$
- ☐ $\frac{1}{n} \sum_{i=1}^n (y_i - X_i \cdot \theta)$
- ☐ $\sum_{i=1}^n (y_i - X_i \cdot \theta)$
- ☐ None of the above

★ ★ ★ ★ ★ This part = 0 as we saw the normal equation for 2d case but

This does not mean we are minimizing this quantity

Remember that your work is graded on the quality of your writing and explanation as well as the validity of the mathematics.

- (1) Alice has collected a dataset of dependent and independent variables $\{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\}$; she does linear regression on it and obtains the weight vector w_{Alice} . Bob also collects the same dataset, but while recording the independent variables (the $x^{(i)}$'s) he uses different units. Specifically, Bob's dataset is $\{(z^{(1)}, y^{(1)}), \dots, (z^{(n)}, y^{(n)})\}$, where for each i , $z^{(i)} = cx^{(i)}$, where $c > 0$ is a scalar. Bob does linear regression on this dataset and obtains a weight vector w_{Bob} .

(a) (5 points) Do Alice and Bob have the same training loss? Is $w_{\text{Alice}} = w_{\text{Bob}}$? In either case, justify your answer.

(b) (5 Points) Suppose now that Bob records each feature in a different unit; that is, for each coordinate j , $z_j^{(i)} = c_j x_j^{(i)}$ for all i , where $c_j > 0$ is a scalar, and the c_j 's are not all equal. Do Alice and Bob still have the same training loss and is $w_{\text{Alice}} = w_{\text{Bob}}$? If yes, justify your answer. If no, provide an example of a dataset where this is not the case.

⑥ Notice c has subscript 'j' & not 'i' so, this is not equivalent to locally weighted regression, where each datapoint is weighted separately. Rather here, we find that each feature is 'just scaled' $\Rightarrow$ same result as part 1

Training loss will be the same as the model converges to the same optimum $\hat{y}_i$, even if input is scaled

$$z^i = cx^i \quad c > 0$$

$$\Rightarrow w_{\text{Bob}}^* z^i = w_{\text{Bob}}^* cx^i$$

$$\text{also } w_{\text{Alice}}^* x^i = w_{\text{Bob}}^* cx^i$$

Since the regression models give same line as solution.

$$\Rightarrow \boxed{w_{\text{Alice}}^* = c \cdot w_{\text{Bob}}^*}$$

Suppose we have a data set of 100 points whose first few rows are shown below, and that we'd like to predict $\hat{y}$ from $\vec{v}$ and $\vec{w}$. Suppose we create a design matrix X whose first column is $\vec{v}$, second column is $\vec{w}$, and third column $\vec{u}$ is a new feature $u_i = |v_i|$. The resulting model is $\hat{y}_i = \beta_1 v_i + \beta_2 w_i + \beta_3 |v_i|$. The top row is row 1, e.g. $y_1 = 4$.

y	v	w
4	-30	1
6	-40	2
5	20	3

notice the slope has no bias for

- (a) [3 Pts] For the data above, suppose we arbitrarily pick $\vec{\beta} = [0.1, 12, 0.2]^T$. What is $\hat{y}_1$?

$$\hat{y}_1 = 15$$

- (b) [2 Pts] For the data above, let $\vec{e}$ be the residual vector if $\vec{\beta} = [0.1, 12, 0.2]^T$. What is $|e_1|$?

$$|e_1| = 11$$

- (c) [3 Pts] For the data above, suppose that $\vec{e} \cdot \vec{e} = 9$. What is the MSE?

$$\text{MSE} = 3, 0.09$$

- (d) [1 1/2 Pts] For the data above, the matrix X has full rank (i.e. no columns are linear combinations of any others). Suppose we compute $Z = (X^T X)^{-1} X^T \vec{y}$. What is Z ? Select one and fill in its blank.

- ☒ It is a vector of length 100.
- ☐ It is a matrix with _____ rows and _____ columns.
- ☐ It does not exist because $|v_i|$ is not differentiable.

$$\begin{aligned} y_1 &= x_1 \cdot \beta = v_1 \beta_1 + w_1 \beta_2 + |v_1| \beta_3 \\ y_1 &= -30 \cdot (0.1) + 1 \cdot (12) + 30 \cdot (0.2) \\ y_1 &= -3 + 12 + 6 = 15 \end{aligned}$$

b) $\vec{e}$ Residual vector

$$\begin{aligned} &= y_i - \hat{y}_i = \\ &= y - A\beta \\ \vec{e} &= \begin{bmatrix} 4 - [-3 + 12 + 6] \\ 6 - [-4 + 24 + 8] \\ 5 - [-2 + 36 + 14] \end{bmatrix} \end{aligned}$$

$$\vec{e} = \begin{bmatrix} 4 - 15 \\ 6 - 28 \\ 5 - 38 \end{bmatrix} = \begin{bmatrix} -11 \\ -22 \\ -33 \end{bmatrix}$$

$$|e| = \sqrt{121 + 484 + 1089} = 169^{1/2} = 13$$

[1 points] True or False? When running linear regressions, it is a good idea to look at the largest (in absolute value) regression weights to see which features are most influential in determining the predictions.

→ Looking @ weights, we cannot determine whether a feature is important or not.

— False — The weights depends on the unit, scaling of the features
for eg. if we scale the input by 1000 the corresponding weight is scaled down by 1000.

→ $\|Xw - y\|^2 = 0$ if y is in the column space of X

→ if y is not in the column space of X then we are projecting y to column space of X & getting $\hat{y} = Xw^*$ $w^* = (X^T X)^{-1} X^T y$

& $\hat{y} = X(X^T X)^{-1} X^T y$ $\nearrow$ PP projection matrix. $X^T X w^* = X^T y$

Q

Let $X \in \mathbb{R}^{m \times n}$, $w \in \mathbb{R}^n$, and $Y \in \mathbb{R}^m$. Consider mean squared error $L(w) = \|Xw - Y\|_2^2$. What is $\nabla_w L(w)$?

- (a) $2Y^T(X^T X w - Y)$
- (b) $2X^T(X^T X w - Y)$
- (c) $2Y^T(Xw - Y)$
- (d) $2X^T(Xw - Y)$ ✓

Gradient Descent → Method of solving an optimising problem.

from calculus, we know minima occurs at $\frac{\partial f}{\partial \theta} = 0$

→ If we cannot solve analytically, we need iterative method.

$$\nabla_{\theta} f = \left[\frac{\partial f(\theta)}{\partial \theta_1}, \frac{\partial f(\theta)}{\partial \theta_2}, \dots, \frac{\partial f(\theta)}{\partial \theta_n} \right]$$

Strategy

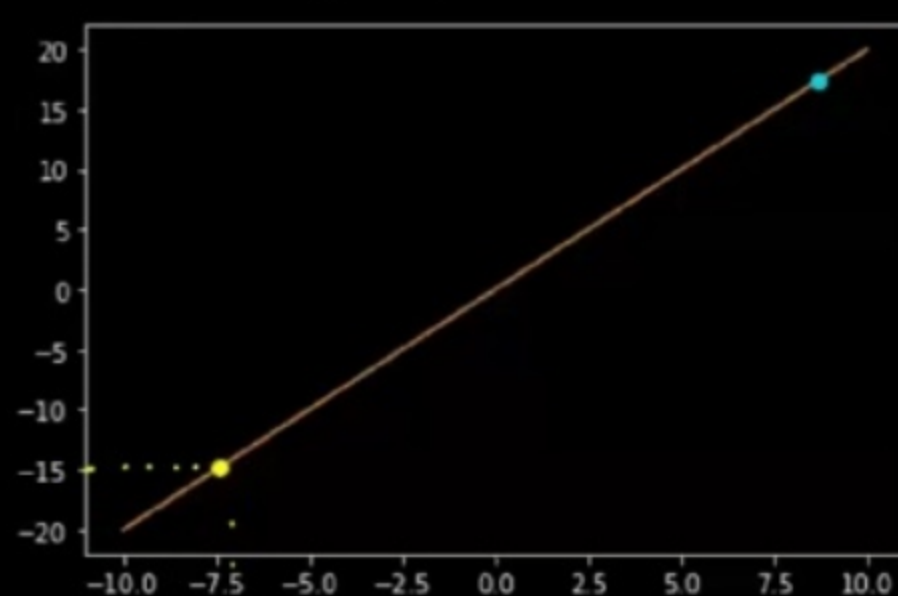
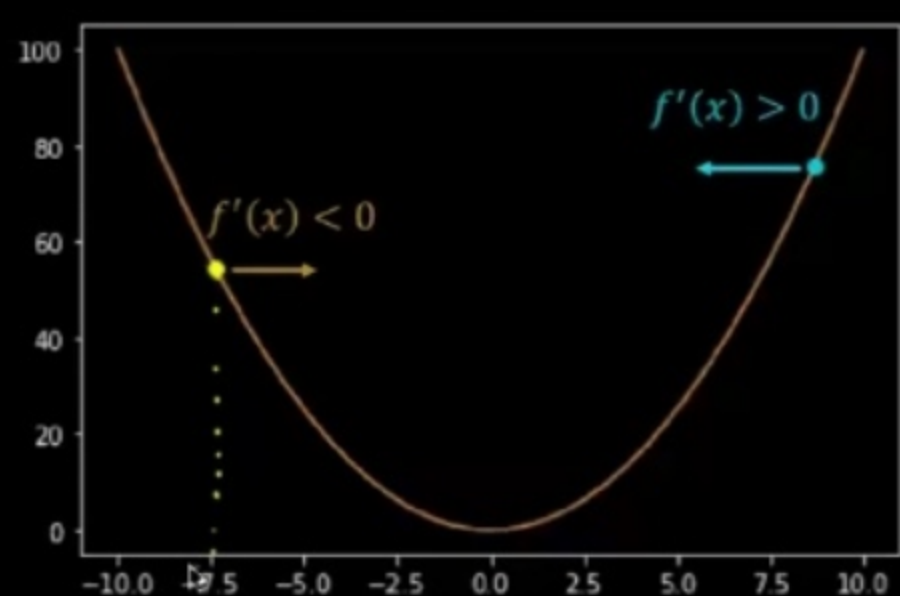
→ Begin with a guess for the optimum point & keep updating the point iteratively in the optimum direction

1D case

→ At time to randomly select a point x_0 . Evaluate $f'(x)$ at x_0
then put $x_1 = x_0 - \alpha f'(x_0)$ ($\alpha > 0$)

$$f(x) = x^2$$

$$f'(x) = 2x$$



$$x_{n+1} = x_n - \alpha f'(x_n) \rightarrow \text{gradient descent}$$

α = step size or learning rate

→ if $f'(x_0) > 0$ $x_1 < x_0$ → move left
→ if $f'(x_0) < 0$ $x_1 > x_0$ → move right

→ Basically go in the direction of derivative to find maxima or opposite direction to find minima.
 α = hyperparameters.

Observation

→ It is not necessary that gradient descent always converges. It may **diverge** or **oscillate**. → so it is best to keep α at optima.

slow algorithm
more chance for converge

α optimum

faster but may
oscillate & diverge

side note

α can be made variable &
can be updated over iteration

→ Gradient descent, the optimum point, may be **local minima**

→ This depends on the initial starting point.

→ One strategy is to repeat Gr. Descent with completely random points multiple times to find many diff. local minima to find the global minima.

N-Dimensional case (multivariate)

$$W_{t+1} = W_t - \alpha \nabla L(W_t) \rightarrow \text{repeat until stopping criteria}$$

stopping criteria

① $|\nabla L(w)| \leq \epsilon$; $\epsilon > 0$

② k # iteration

③ Combination of ② & ①

$$\begin{bmatrix} W_1^{t+1} \\ W_2^{t+1} \\ \vdots \\ W_n^{t+1} \end{bmatrix} = \begin{bmatrix} W_1^t \\ W_2^t \\ \vdots \\ W_n^t \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial L}{\partial W_1} & \frac{\partial L}{\partial W_2} & \dots & \frac{\partial L}{\partial W_n} \end{bmatrix}^T \bigg|_{\bar{w} = \bar{w}^t}$$

(2 points) Consider applying gradient descent with step size $\alpha = 0.1$ to find the $\mathbf{x}$ that minimizes the function $f(\mathbf{x}) = f((x^{(1)}, x^{(2)})) = (x^{(1)} - 1)^2 + (x^{(2)} - 2)^2$ starting from $\mathbf{x}_0 = (0, 0)$. Find the value $\mathbf{x}_1$ after one iteration.

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.1 \begin{bmatrix} -2 \\ -4 \end{bmatrix} \bigg|_{\substack{x=0 \\ y=0}}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$

$$f(x, y) = (x - 1)^2 + (y - 2)^2$$

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

(8 points) Mark each statement as true (T) or false (F).

- _____ Gradient descent can fail to converge on a convex function if step size α is such that we get stuck in a cycle, oscillating between two or several values. **ANSWER: T**
- _____ For a non-convex function, gradient descent can fail to converge by descending without bound. **ANSWER: T**
- _____ Gradient descent can fail to converge on a convex function if it gets stuck in a local minimum. **ANSWER: F**
- _____ Gradient descent can fail to converge on a convex function if the step size $\alpha > 0$ is too small. **ANSWER: F**

Consider the following loss function on vectors $w \in \mathbb{R}^4$:

$$L(w) = w_1^2 + 2w_2^2 + w_3^2 - 2w_3w_4 + w_4^2 + 2w_1 - 4w_2 + 4.$$

- What is $\nabla L(w)$?
- Suppose we use gradient descent to minimize this function, and that the current estimate is $w = (0, 0, 0, 0)$. If the step size is η , what is the next estimate?
- What is the minimum value of $L(w)$?
- Is there a unique solution w at which this minimum is realized?

Solution a)

$$\nabla L(w)$$

$$= \begin{bmatrix} 2w_1 + 2 \\ 4w_2 - 4 \\ 2w_3 - 2w_4 \\ 2w_4 - 2w_3 \end{bmatrix}^T$$

b) $w_1 = w_0 - \eta \begin{bmatrix} 2 \\ -4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -2\eta \\ +4\eta \\ 0 \\ 0 \end{bmatrix} = w_1$

c) $\nabla L(w) = 0 \Rightarrow \begin{cases} w_1 = -1 \\ w_2 = 1 \\ w_3 = w_4 \end{cases} \Rightarrow \min L(w) = 1 + 2 - 2 - 4 + 4 = 1$

d) No. for any $w_3 = w_4$ this is a minimum

Let

$$f(x, y) = (x - 1)^2(y - 3)^2$$

$$P_0 = (x_0, y_0) = (4, 5)$$

$$\eta = 0.05$$

Using the gradient descent algorithm find $P_2 = (x_2, y_2)$ and then calculate $f(P_2)$ to at least the nearest thousandth (at least 3 decimal places).

$$\Delta f = \begin{bmatrix} (y-3)^2 \cdot 2(x-1) \\ (x-1)^2 \cdot 2(y-3) \end{bmatrix}^T$$

$$P_1 = \begin{bmatrix} 4 \\ 5 \end{bmatrix} - 0.05 \begin{bmatrix} 24 \\ 36 \end{bmatrix} = \begin{bmatrix} 2.8 \\ 3.2 \end{bmatrix}$$

Similarly, $P_2 = P_1 - 0.05 \nabla f(2.8, 3.2)$
 $\Rightarrow P_2 = (2.7928, 3.152)$
 $f(P_2) = 0.0588$

(5 points) Consider the following loss function:

$$L(w) = \sum_{i=1}^n \log(1 + w^T x_i)$$

What is $\nabla L(w)$? Write down the update step for gradient descent.

$$\nabla L(w) = \sum_{i=1}^n \left(\frac{1}{1 + w^T x_i} \cdot x_i \right)$$

$$w_t = w_{t-1} - \alpha \sum_{i=1}^n \left(\frac{x_i}{1 + w_{t-1}^T x_i} \right)$$

[3 Pts] The learning rate can *potentially* affect which of the following? Select all that apply. Assume nothing about the function being minimized other than that its gradient exists. You may assume the learning rate is positive.

- ☒ The speed at which we converge to a minimum.
- ☒ Whether gradient descent converges.
- ☐ The direction in which the step is taken.
- ☒ Whether gradient descent converges to a local minimum or a global minimum.

Too high learning rate can jump from local to global or global to local

Which of the following could affect whether or not gradient descent converges to the global minimum?

(a) [1 Pt] Learning Rate
☒ True ☐ False

(b) [1 Pt] Initialization of parameters
☒ True ☐ False

(c) [1 Pt] The ordering of the data (ignoring issues related to numerical precision).
☐ True ☒ False

(m) [3 pts] Let $f: \mathbb{R} \rightarrow \mathbb{R}$ be a continuous, smooth function whose derivative $f'(x)$ is also continuous. Suppose f has a unique global minimum $x^* \in (-\infty, \infty)$, and you are using **gradient descent** to find x^* . You fix some $x^{(0)} \in \mathbb{R}$ and $\epsilon > 0$, and run $x^{(t)} = x^{(t-1)} - \epsilon f'(x^{(t-1)})$ repeatedly. Which of the following statements are true?

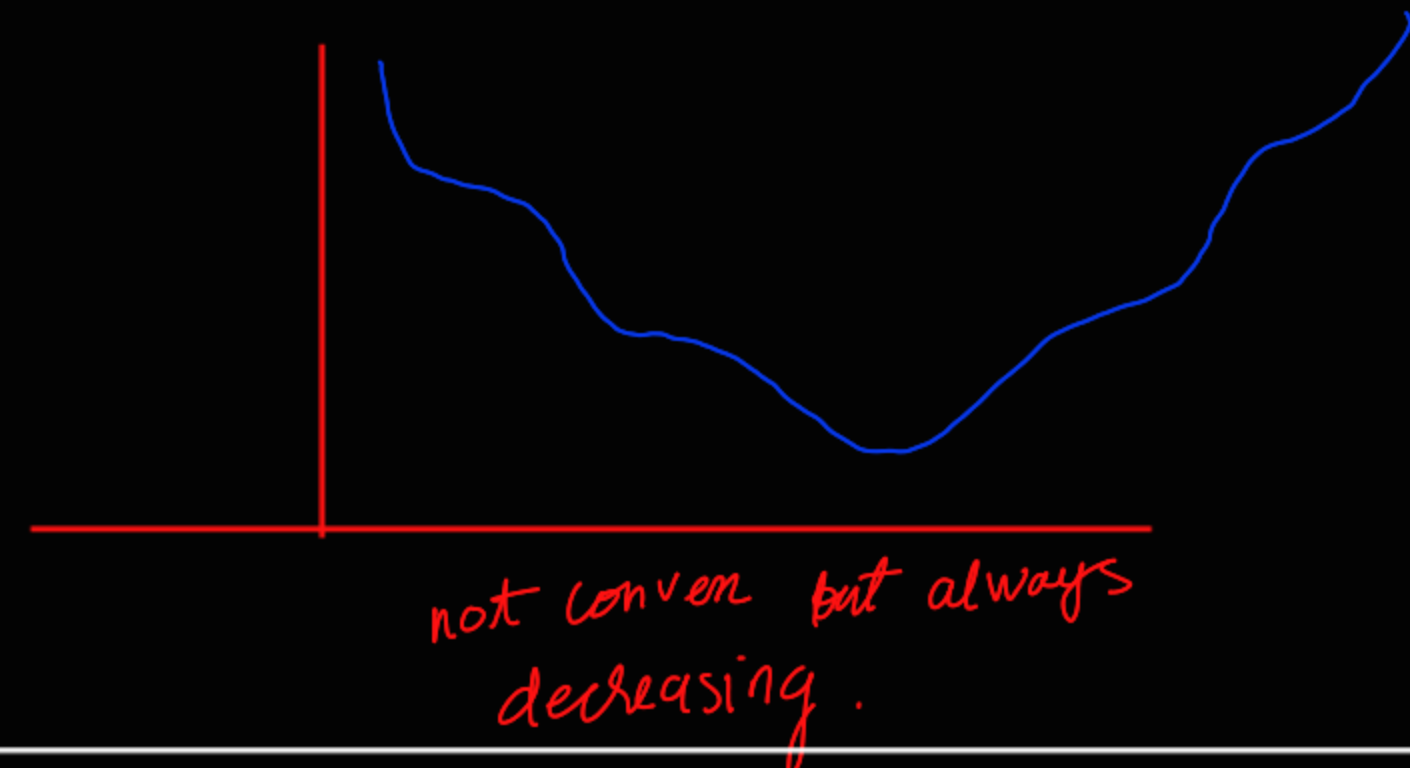
~~False~~ ☐ Gradient descent is sure to converge, to *some* value, for any step size $\epsilon > 0$

~~False~~ ☐ Assuming gradient descent converges, it converges to x^* if and only if f is convex

~~True~~ ☐ If f has a local minimum x' different from the global one, i.e., $x' \neq x^*$, and $x^{(t)} = x'$ for some t , gradient descent will not converge to x^*

~~True~~ ☐ If, additionally, f is the objective function of logistic regression, and gradient descent converges, then it converges to x^*

Logistic loss is convex



Select all that apply: Consider the convex function $f(z) = z^2$. Let α be our learning rate in gradient descent.

For which values of α will $\lim_{t \rightarrow \infty} f(z^{(t)}) = 0$, assuming the initial value of z is $z^{(0)} = 1$ and $z^{(t)}$ is the value of z after the t -th iteration of gradient descent?

- ☐ $\alpha = 0$
☐ $\alpha = \frac{1}{2}$
☐ $\alpha = 1$
☐ $\alpha = 2$
☐ None of the above

Numerical answer: Give the range of all values for $\alpha \geq 0$ such that $\lim_{t \rightarrow \infty} f(z^{(t)}) = 0$, assuming the initial value of z is $z^{(0)} = 1$.

$0 < \alpha < 1$

$f(z) = z^2$

if $z^0 = 1$

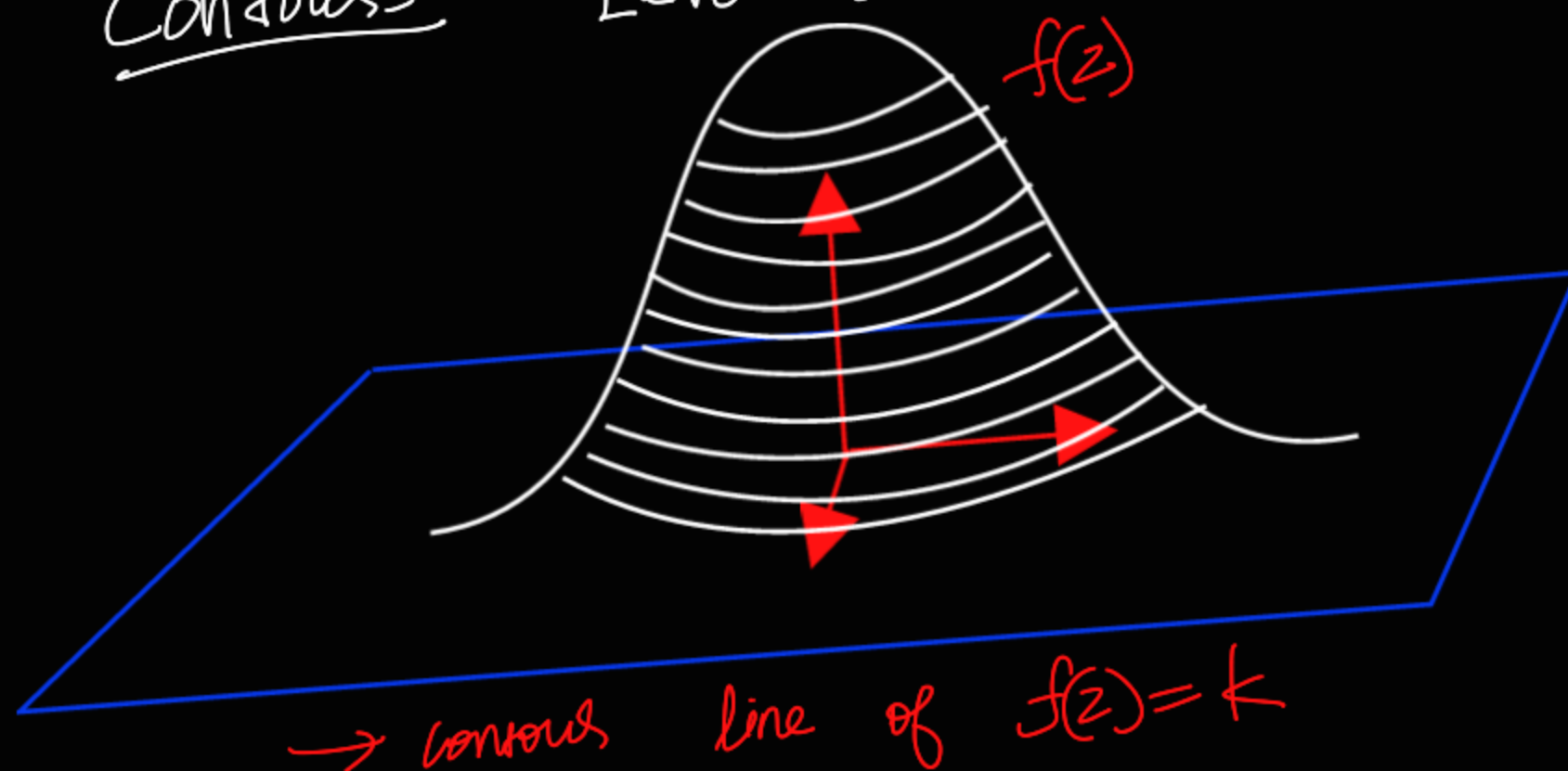
$z^1 = 1 - \alpha \cdot 2$

for $\alpha = 1$ $z^1 = 1 - 2 = -1$

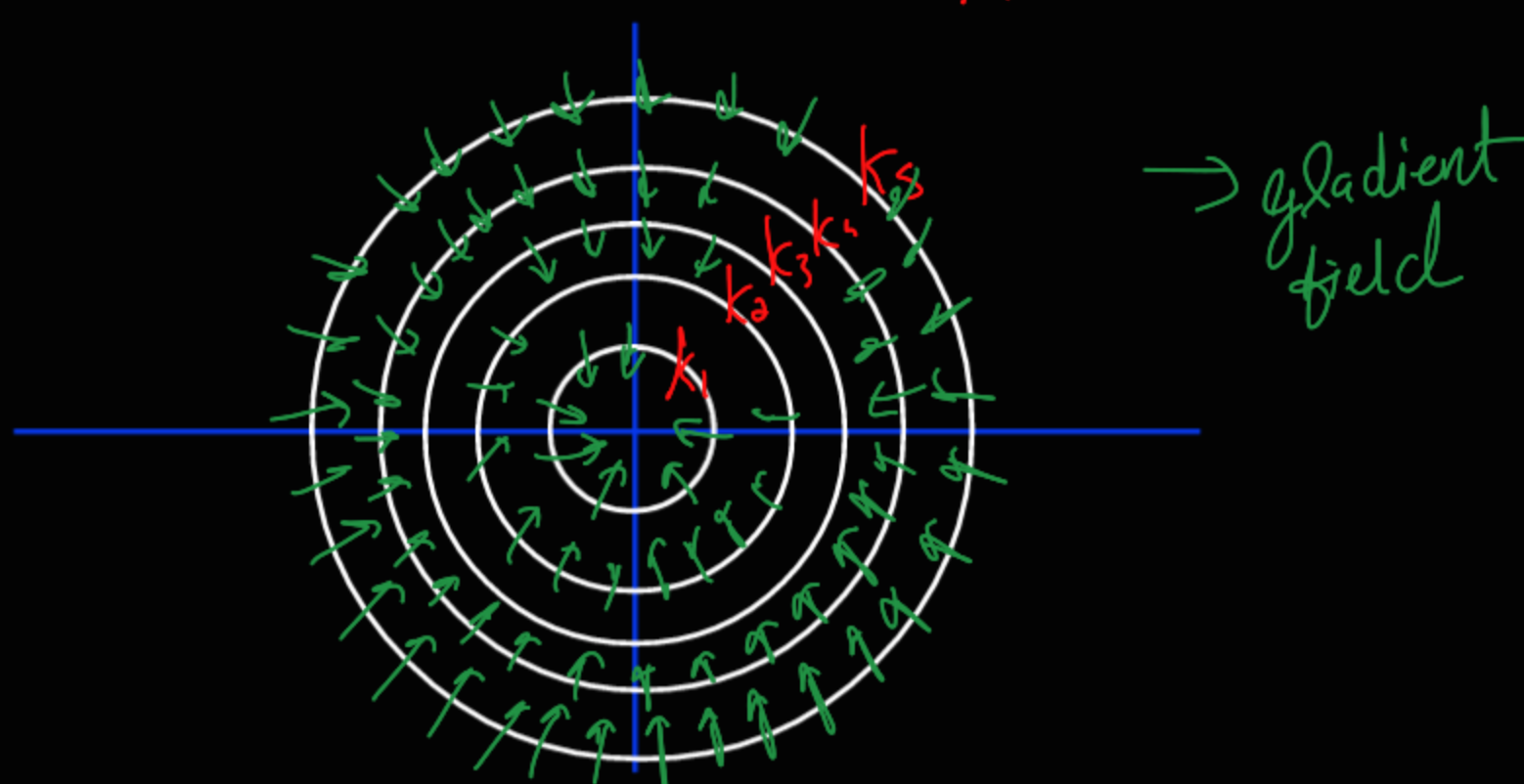
$z^2 = -1 - \alpha(-2) = -1 + 2 = 1$

$\therefore$ for $\alpha = 1$, G.D oscillates
 for $\alpha < 1$ it converges

Contours - Level Curves



$k_1 > k_2 > k_3 > k_4 > k_5$



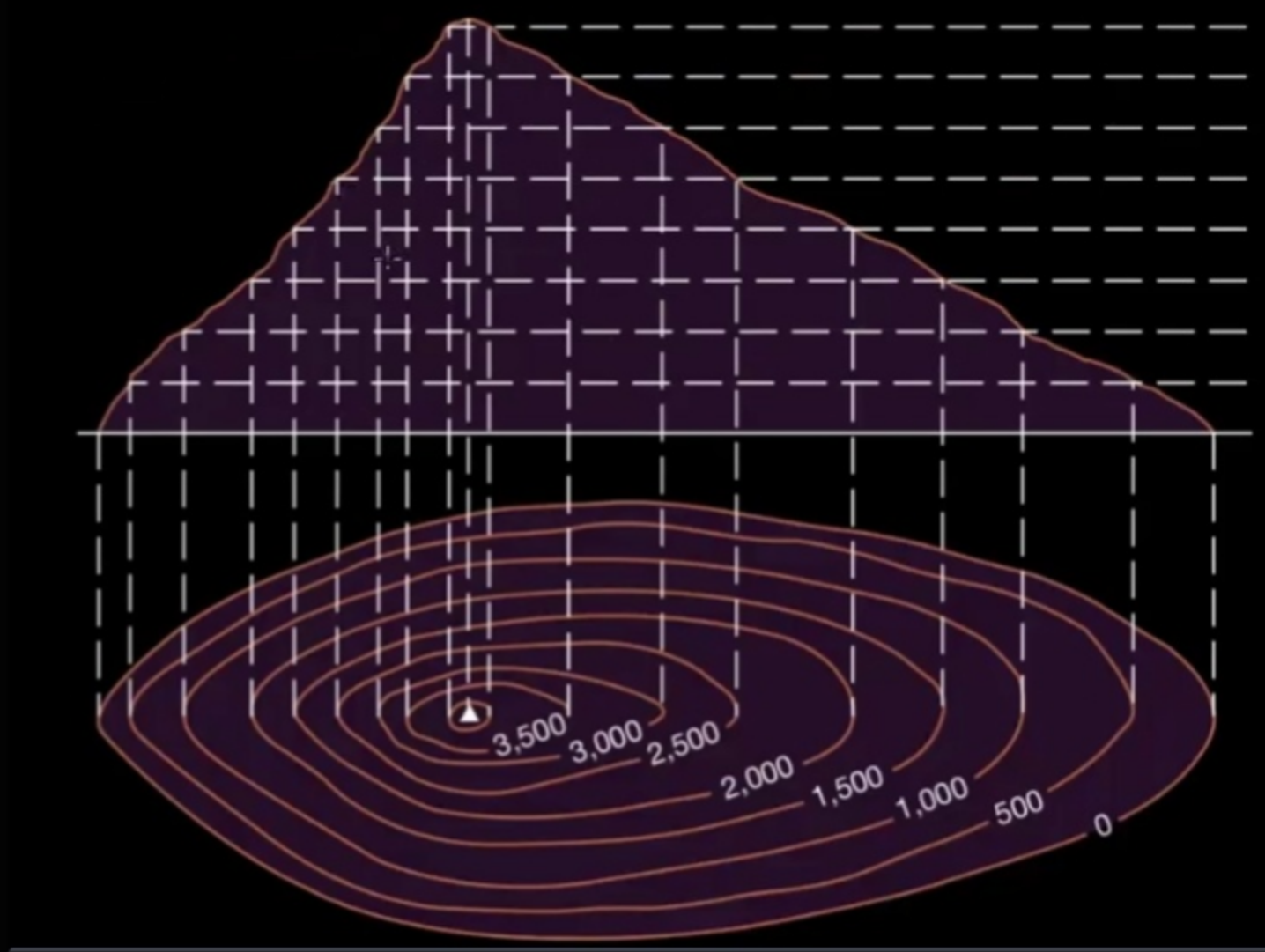
Intuition :- a) gradient field is perpendicular to the contour line.

b) For a function $f(z) \in \mathbb{R}^n \rightarrow \mathbb{R}$ then the contours $f(z) = k$ specifies a n dimensional shape

c) Contour lines are drawn for equally spaced values of $k = k_1, k_2, k_3, k_4$

ie $k_1 + d = k_2$
 $k_2 + d = k_3$ }

d) $\Rightarrow$ When $|\text{gradient}|$ increases, the contour lines are packed together.



$f(z)$ changes rapidly $\Rightarrow$ contours are close packed
 $|f'(z)|$ large

$f(z)$ changes slow $\Rightarrow$ contours are further apart.
 $|f'(z)|$ small

Linear Regression & Gradient Descent

- $\rightarrow$ Batch Gradient Descent $\rightarrow$ what we have seen above
- $\rightarrow$ Mini Batch GD
- $\rightarrow$ Stochastic Gradient Descent

Why we need Gradient descent

- $\rightarrow$ Closed form solution cannot be obtained
- $\rightarrow$ Using Linear algebra the complexity is $O(Nm^2 + m^3)$

$$X^T X \theta = X^T y$$

$\rightarrow$ Matrix multiplication

$m \times m$ matrix
 each entry is N multiplications
 $\Rightarrow$ Total result = Nm^2

$\rightarrow$ Solving for x involves Gaussian elimination

We have seen cost of elimination is $O(m^3)$ for m by m matrix in LA

N = # datapoints
 n = # features
 $X = N \times (n+1) \mid N \times m$
 $\theta = (n+1) \times 1 \mid m \times 1$
 $y = N \times 1 \mid N \times 1$
 $m = n+1$

$\rightarrow$ Cost involves matrix multiplication & Gaussian elimination

Gradient Descent

- Requires multiple iterations
- Need to choose α
- Works well when n is large

$\downarrow$
 number of datapoints

Closed Form Solution

- Non-iterative
- No need for α
- Slow if n is large
 - Computing $(X^T X)^{-1}$ is roughly $O(n^3)$

$$X = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \leftarrow \text{Big matrix if } n \text{ is high}$$

Gradient Descent cost
 $O(Nmt)$

Update Rule

$$w_0 \leftarrow w_0 - \alpha \frac{\partial L(w)}{\partial w_0}$$

$$w_1 \leftarrow w_1 - \alpha \frac{\partial L(w)}{\partial w_1}$$

$\vdots$

$$w_d \leftarrow w_d - \alpha \frac{\partial L(w)}{\partial w_d}$$

OR

$$\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial L(w)}{\partial w_0} \\ \frac{\partial L(w)}{\partial w_1} \\ \vdots \\ \frac{\partial L(w)}{\partial w_d} \end{bmatrix}$$

For example if cost function $L(w) = \frac{1}{2} (y - Xw)^T (y - Xw) = \sum_{i=1}^N (y_i - x_i w)^2$

then $\nabla L(w) = \left[\frac{\partial L(w)}{\partial w_1} \quad \frac{\partial L(w)}{\partial w_2} \quad \dots \quad \frac{\partial L(w)}{\partial w_m} \right]$

$$\nabla L(w) = -(y - Xw)^T X = -X^T (y - Xw) = X^T (Xw - y) \rightarrow \text{col vector}$$

where, $\frac{\partial L(w)}{\partial w_j} = \sum_{i=1}^N - (y_i - x_i w) x_{ij}$

or $(Xw - y)^T X \rightarrow \text{row vector}$

$\therefore$ update rule

$$w^t = w^{t-1} - \alpha \cdot X^T (Xw^{t-1} - y)$$

$$w^t = w^{t-1} - \alpha \underbrace{X^T}_{N \times m} \underbrace{(Xw^{t-1} - y)}_{N \times 1}$$

$N \times m$ multiplication
 $N \times m$ multiplication

$$\Rightarrow O(Nm + Nm)$$

$$\Rightarrow O(Nm) \text{ for 1 iteration}$$

$$\Rightarrow O(Nmt) \text{ for } t \text{ iteration}$$

$\rightarrow$ This is still very high for very large N .
 $\rightarrow$ Even for 1 iteration we take $O(Nm)$ time

Stochastic gradient descent

$\rightarrow$ Take 1 data at a time
 $\rightarrow$ Do not take entire train loss by $\sum_{i=1}^N$, but take only the i^{th} dataset for calculating the gradient

$$\nabla L(w)_{\text{stochastic } i} = x_i^T (x_i w - y_i)$$

Update rule

$$w^t = w^{t-1} - \alpha x_i^T (x_i w - y_i)$$

$\rightarrow$ Stochastic gradient descent converges faster because it doesn't need to go through all N datasets for even 1 iteration like batch gradient-descent. It does 1 iteration per dataset in $O(m)$ complexity.

each iteration is $O(m)$
 t iteration take $O(mt)$
but we have to do N iterations to cover the dataset once.

$\rightarrow$ It takes more iteration than batch GD but the speed up per iteration outweighs the increased no of iterations

- ▶ The answer to the problem of slow gradient descent is Stochastic Gradient Descent (SGD).
- ▶ Instead of looping through all the training examples to compute a single gradient and make one step, SGD loops through the examples (x, y) and updates the weights w based on each example.
- ▶ Each update is not as precise since it's based on one example, but we can make many more updates this way.
- ▶ **Aside:** There is a continuum between SGD and GD called minibatch SGD, where each update consists of an average over B examples.
- ▶ **Aside:** There are other variants of SGD. You can randomize the order in which you loop over the training data in each iteration. This is important, for instance, if your training data has all the positive examples first and the negative examples afterward.

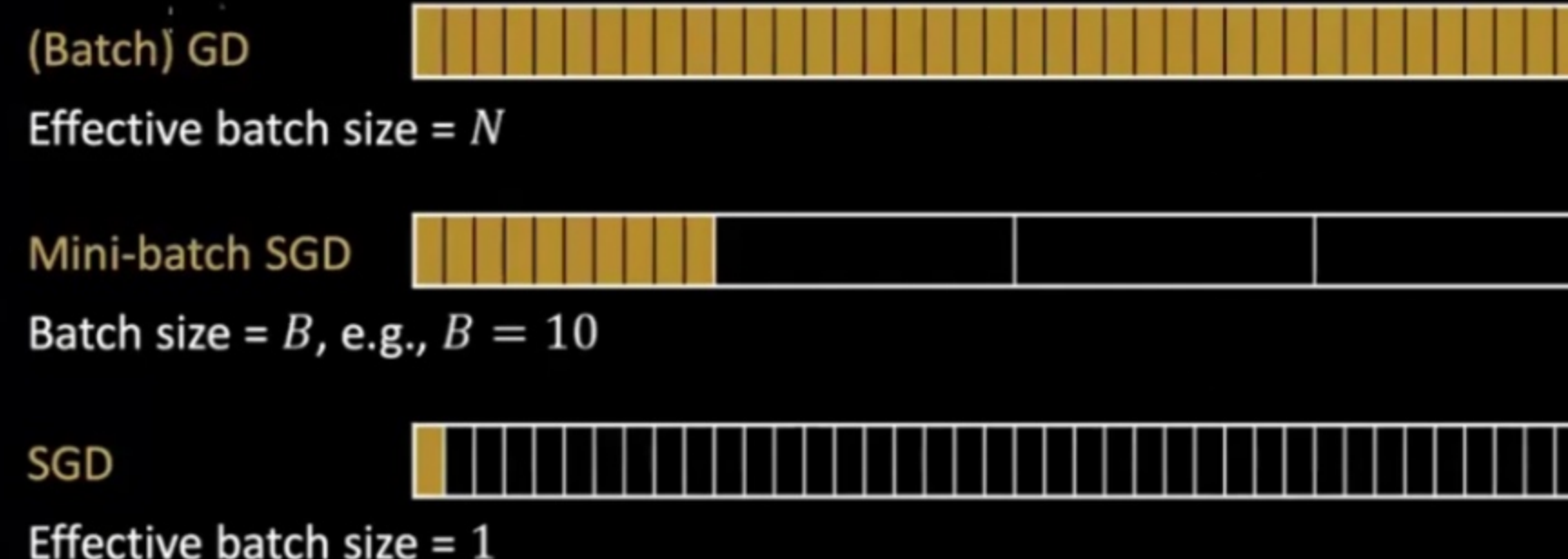


→ Linear regression ^{with cost = MSE} is a convex optimisation $\Rightarrow$ there is only one minima $\neq$ i.e the global minima $\Rightarrow$ We will always find global minima using GD

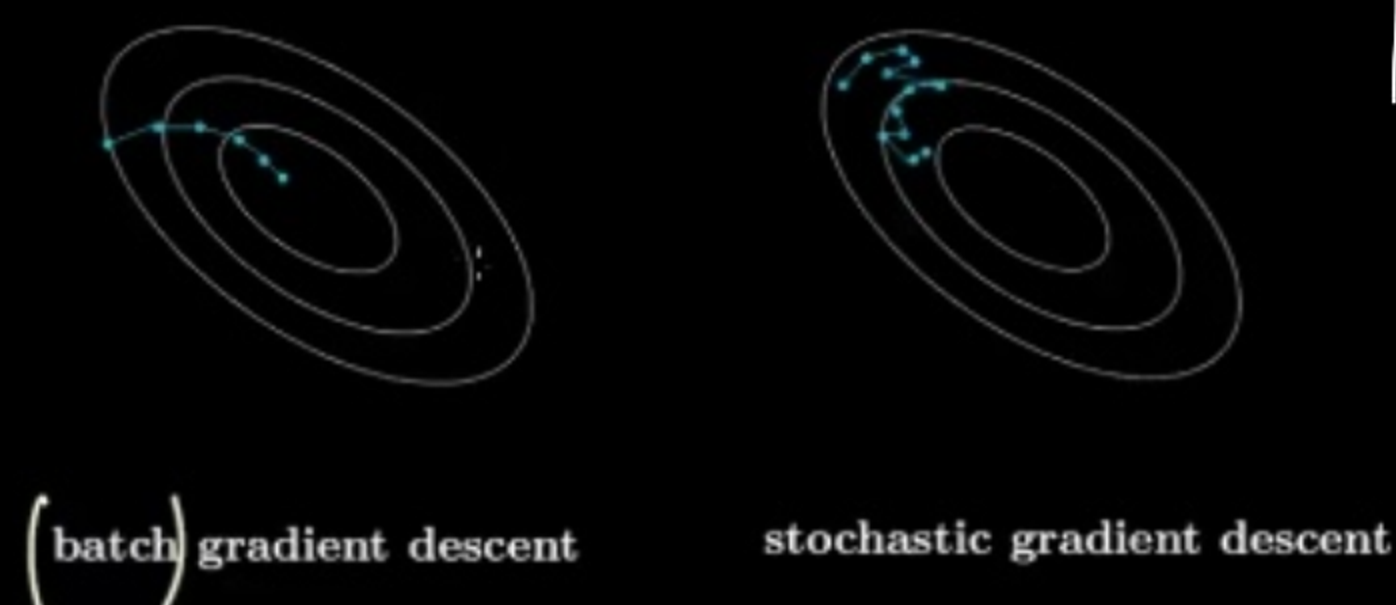
Gradient Descent

Batch $\rightarrow$ Mini-batch $\rightarrow$ Stochastic

Basically, just a difference in how many training points are each time we compute a gradient (and then take a step with that gradient)



- Batch gradient descent moves directly downhill. SGD takes steps in a noisy direction, but moves downhill on average.



Q) After a gradient descent update step, the objective function value at the weight vector is always lower after the update than before.

False : If the step size is too large, the objective function value can increase after an update.

Q) There is always a unique steepest descent direction in gradient descent.

False If the gradient is zero along some dimensions, there would not be a unique steepest descent direction.

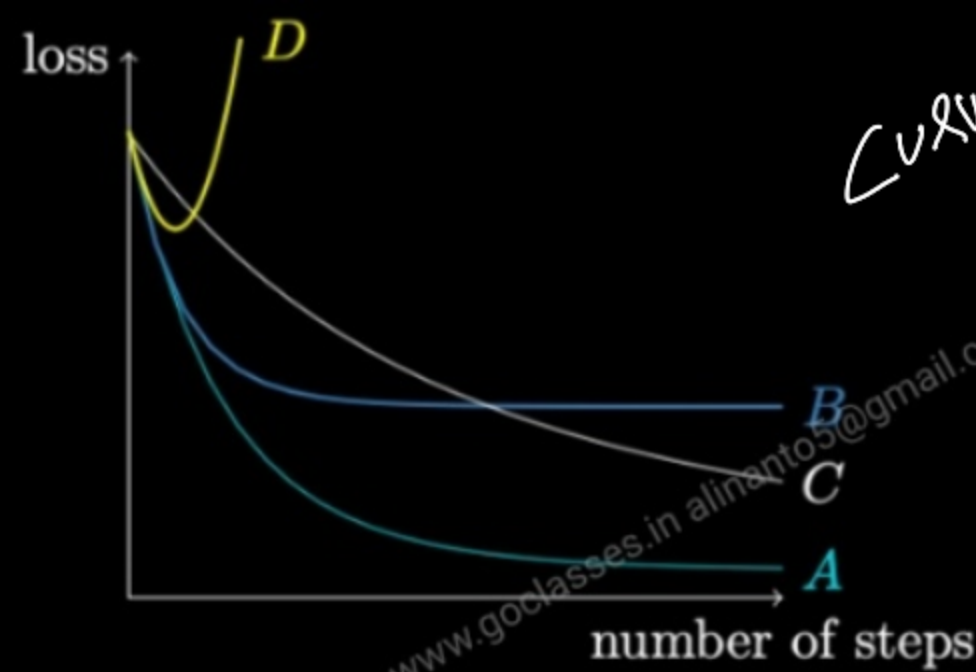
Q) When the algorithms converge, stochastic gradient descent always finds the same solution as gradient descent.

Solution: False. For example, if there are multiple local minima, even two different runs of gradient descent may converge to different local minima, depending on the random initial condition. Adding on top of this the stochastic/randomness from stochastic gradient descent makes the statement definitely false.

Q) [3 pts] Which of the following statements about optimization algorithms are correct?

- ☐ Newton's method always requires fewer iterations than gradient descent.
- ☐ Stochastic gradient descent always requires fewer iterations than gradient descent.
- ☒ Stochastic gradient descent, even with small step size, sometimes increases the loss in some iteration for convex problems.
- ☐ Gradient descent, regardless the step size, decreases the loss in every iteration for convex problems.

Arguments like one reasonable optimization algorithm dominates another for all problems are in general wrong. Gradient descent can work for some loss that Newton does not even converge.



Learning rate too low (select one):

- ☐ A ☐ B ☒ C ☐ D

Learning rate about right (select one):

- ☒ A ☐ B ☐ C ☐ D

Learning rate too high (select one):

- ☐ A ☒ B ☐ C ☐ D

Learning rate much too high (select one):

- ☐ A ☐ B ☐ C ☒ D

Solution: A low learning rate leads to slow decay of the loss— this occurs for *C*. A good learning rate leads to moderately quick decay to low loss, as in *A*. A high learning rate can prevent gradient descent from reaching the global minimum of the objective, and can cause it to oscillate between parameter values that give a higher loss value, as in *B*. A very high learning rate can cause gradient descent to diverge, as in *D*.

10. Below are a list of potential advantages and disadvantages of stochastic gradient descent (SGD). Select **all** that are true regarding SGD.

Advantages:

- (a) SGD is more memory-efficient because it takes a smaller number of samples at a time compared to classical gradient descent which takes the entire dataset into weight update
(b) In SGD, the update on weight w_{t+1} has lower variance because it doesn't take many samples into account at a time

Disadvantages:

- (c) The noise in the dataset has higher impact on the stability of SGD than on that of the classical gradient descent.
(d) SGD is more sensitive to learning rate compared to classical gradient descent
(e) It's more computationally inefficient to use SGD for a large dataset than to use classical gradient descent because it requires more resources to randomly sample a data point for the weight update

Correct answers: (a), (c), (d)

Explanation: Note: option (d) (SGD is more sensitive to learning rate compared to classical gradient descent) was deemed unclear and we accepted either (a, c) or (a, c, d) as correct.

Q Consider the quadratic loss function $L(w) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} (w - y_i)^2$, where w is a single parameter. What is the stochastic gradient update for w at time t with step size η ?

- A) $w^{(t+1)} = w^{(t)} + \eta(w^{(t)} - y_t)$
☒ B) $w^{(t+1)} = w^{(t)} - \eta(w^{(t)} - y_t)$
C) $w^{(t+1)} = w^{(t)} \times \eta \times (w^{(t)} + y_t)$

b) If the step size is dynamic, $\eta = \frac{1}{t}$, and $w^{(1)} = 0$, find the expression for $w^{(t)}$ after t steps of SGD.

A) $w^{(t)} = \frac{1}{t} \sum_{i=1}^t (y_i - w^{(i)})$

B) $w^{(t)} = \sum_{i=1}^t \frac{y_i - w^{(i)}}{t}$

C) $w^{(t)} = \sum_{i=1}^t \frac{y_i}{i}$

☒ D) $w^{(t)} = \sum_{i=1}^t \frac{y_i}{t}$

Solution

$$w^t = w^{t-1} - \frac{1}{t} (w^{(t-1)} - y^t)$$

$$w^t = w^{t-1} - \frac{(w^{t-1} - y^t)}{t} = \frac{w^{t-1} - y^t}{t}$$

$$\Rightarrow w^t = + \sum_{i=1}^t \left(\frac{y_i - w^i}{i} \right)$$

$$\text{but } w^2 = w^1 - \frac{1}{1} (w^1 - y^1) = y^1$$

$$w^3 = w^2 - \frac{1}{2} (w^2 - y^2) = \frac{1}{2} (y^1 + y^2)$$

$$w^4 = w^3 - \frac{1}{3} (w^3 - y^3) = \frac{2}{3} w^3 - \frac{y^3}{3} = \frac{y^1 y^2 + y^3}{3}$$

$$\text{so } w^t = \left(\sum_{i=1}^t \frac{y_i}{t} \right) \quad \square$$

Which of the following is *not* a true statement about stochastic gradient descent (SGD)?

- (A) It is possible to use the same data example across different iterations.
- (B) Stochastic gradient descent is faster because it has a higher convergence rate in optimization.
- (C) Stochastic gradient descent can create vastly different gradients in the first few steps comparing to those in gradient descent.

Solution:

The solution is (B).

(c) (2 points) Which of the following is true, given the optimal learning rate?

- (i) Batch gradient descent is always guaranteed to converge to the global optimum of a loss function.
- (ii) Stochastic gradient descent is always guaranteed to converge to the global optimum of a loss function.
- (iii) For convex loss functions (i.e. with a bowl shape), batch gradient descent is guaranteed to eventually converge to the global optimum while stochastic gradient descent is not.
- (iv) For convex loss functions (i.e. with a bowl shape), stochastic gradient descent is guaranteed to eventually converge to the global optimum while batch gradient descent is not.
- (v) For convex loss functions (i.e. with a bowl shape), both stochastic gradient descent and batch gradient descent will eventually converge to the global optimum.
- (vi) For convex loss functions (i.e. with a bowl shape), neither stochastic gradient descent nor batch gradient descent are guaranteed to converge to the global optimum.

already given learning rate is optimum, so both gradient will always converge
→ stochastic depends on data.
stochastic GD even if reaches the w^* may change after next iteration, while in GD it does not change at all.

Solution: (iii)

Insights from problems
→ Convex function → only 1 local minima which is always the global minima
reverse does not apply
unique global minima → no other local minima (a lot of such questions. Always assume there are other local minima unless convex)
→ For questions involving convergence & divergence, this depends on α .
so unless mentioned optimum learning right, always assume, GD or SGD can diverge because of $\alpha \rightarrow \text{high}$
→ Stochastic GD can sometimes go opposite to the gradient because it iterates using single or minibatch data points which may be outliers that take w away from w^* .
→ In questions involving GD where there ask w/ given a setup either write w_1, w_2, w_3 etc in terms of setup & recognise pattern or use general update rule. But always say both before giving answer.

POLYNOMIAL REGRESSION

- To model non linear relation using linear regression
- using polynomial terms (x^2, x^3 etc) in the hypothesis function

1-D Case

1 input
1 D/P

$$y = w_0 + w_1 x + w_2 x^2 + \dots + w_M x^M = \sum_{i=0}^M w_i x^i$$

} degree M
polynomial

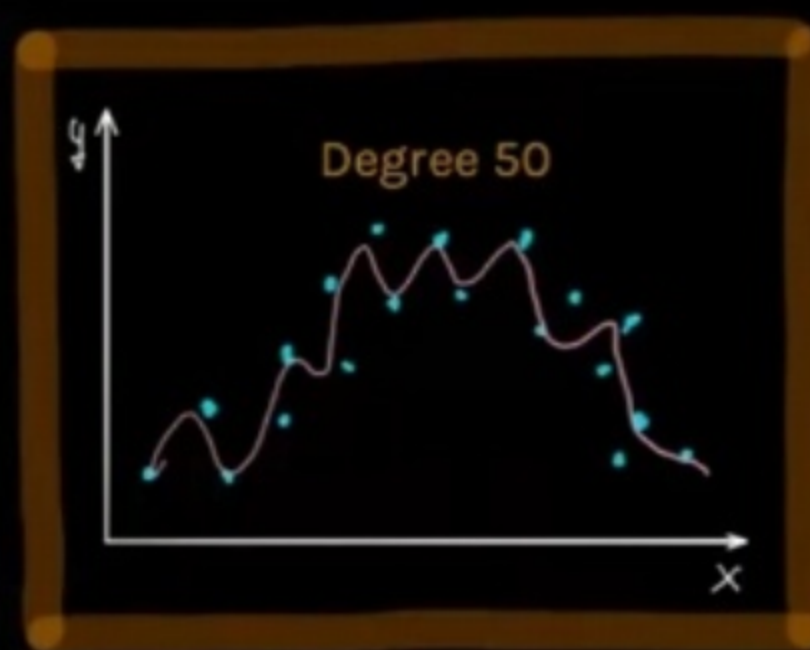
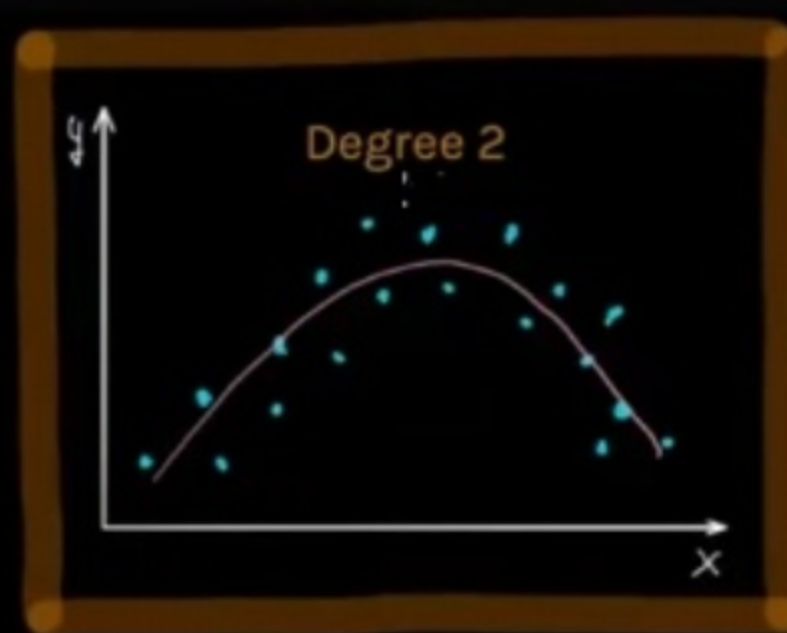
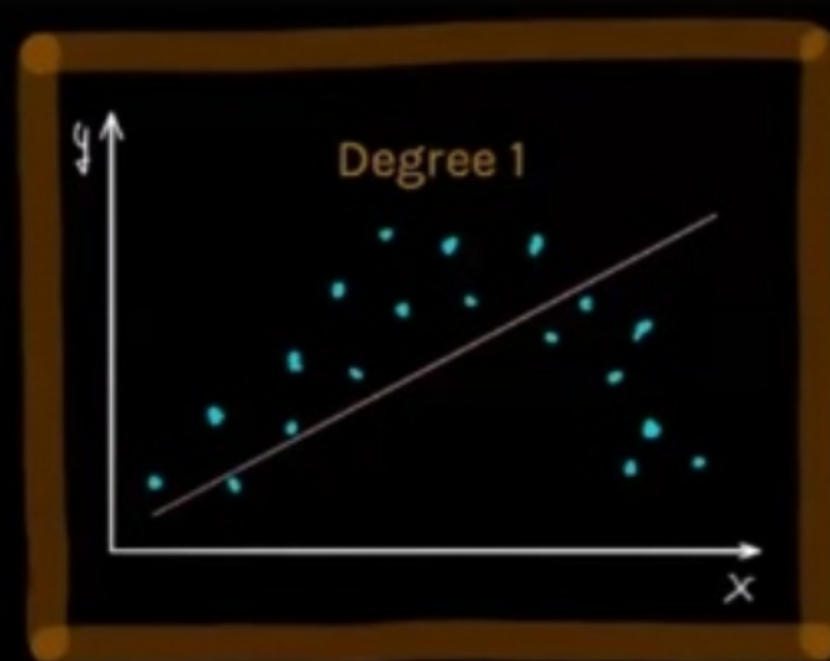
→ This is called poly regression or NONLINEAR BASIS EXPANSION } feature transformation

→ Now treat each term in the polynomial as an independent feature.

★ → y is linear in w since we can express $y = W^T \phi(x)$; $\phi(x)$ → potentially non linear fun of x

OVERFITTING

→ As we increase the degree the hypothesis tends to overfit the data.



Q) Suppose we are performing linear regression using a non-linear basis expansion Φ . Which of the following statements is true about the learned predictor?

- (a) It is a linear function of the inputs and a linear function of the weights.
- (b) It is a linear function of the inputs and a non-linear function of the weights.
- ✓ (c) It is a non-linear function of the inputs and a linear function of the weights.
- (d) It is a non-linear function of the inputs and a non-linear function of the weights.

Here inputs are x_1, x_2, x_3
weights are w_1, w_2, w_3

→ Inputs should not be considered as non linear basis expansion

Q) Linear regression. What is linear about it?
Select all that apply.

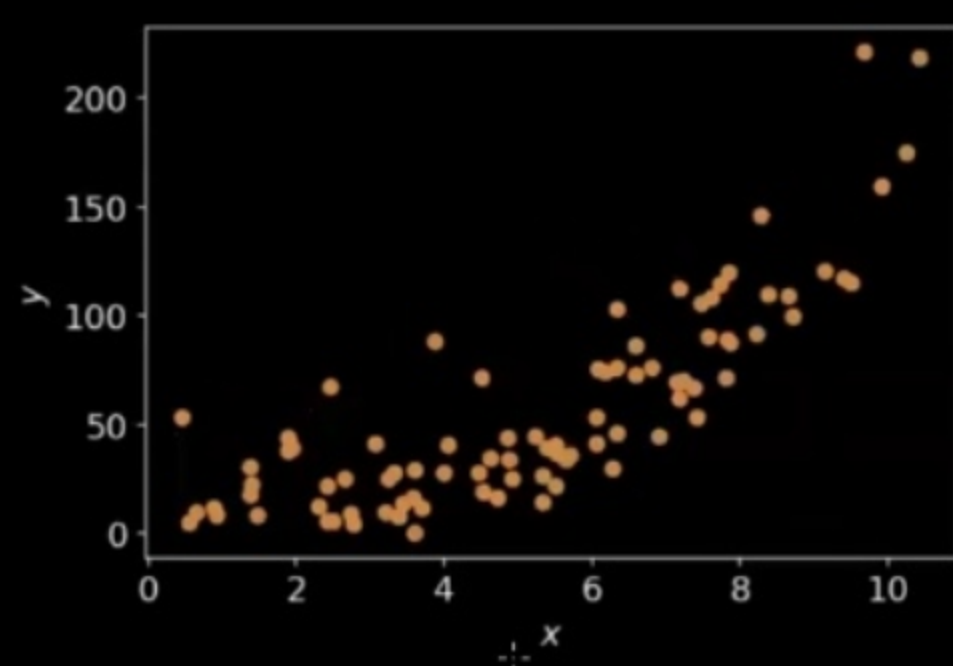
- A. Always fits a linear (or affine) shape to the data
- B. Linear objective function with respect to the input
- C. Linear objective function with respect to the parameters
- ✓ D. None of the above

} shape fitted can be non linear

} Objective function $\neq$ hypothesis
hypothesis linear in parameters $\Rightarrow$

Loss
Objective Cost MSE } is square (quadratic)

Q) [2 Pts] Pizza Pete creates the scatter plot of y against x shown below. Which of the following transformations is most appropriate to linearize the data?



- ☐ $\sqrt{x}$
- ☐ x^3
- ☐ y^3
- ☐ y^2

[5 Pts] Suppose we have a dataset $\{(x_i, y_i)\}_{i=1}^n$ and we model some particular observation y_i as $\hat{y}_i = f_{\theta_1}(x_i) = \theta_1 \ln(x_i)$. What is the minimizer $\hat{\theta}_1$ that minimizes MSE? Define MSE as $\frac{1}{n} \sum_{i=1}^n (y_i - f_{\theta_1}(x_i))^2$. You must show your work in the box on the following page to get full credit. The critical point you find is guaranteed to be a minimum, so you do not need to prove that this point is a minimum.

For full credit, you must show your work on the following page. You will receive partial credit for a correct answer without showing a correct approach.

- ☐ $\frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}$
- ☐ $\frac{\sum_{i=1}^n \ln(x_i) y_i}{\sum_{i=1}^n \ln(x_i)^2}$
- ☐ $\frac{\sum_{i=1}^n \ln(x_i) y_i}{\sum_{i=1}^n (\ln(x_i))^2}$
- ☐ $\frac{\sum_{i=1}^n y_i / x_i}{\sum_{i=1}^n x_i^2}$

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - f_{\theta_1}(x_i))^2 = L(\theta)$$

$$\nabla_{\theta_1} L(\theta) = \frac{\partial L(\theta_1, x_i)}{\partial \theta_1} = \frac{1}{n} \sum_{i=1}^n 2(y_i - f_{\theta_1}(x_i)) \cdot \frac{\partial f_{\theta_1}(x_i)}{\partial \theta_1}$$

$$= -\frac{2}{n} \sum_{i=1}^n [y_i - \theta_1 \ln(x_i)] \ln(x_i)$$

$$\frac{\partial L(x_i)}{\partial \theta_1} = 0 \Rightarrow \sum_{i=1}^n (y_i - \theta_1 \ln(x_i)) \cdot \ln(x_i) = 0 = \sum_{i=1}^n y_i \ln(x_i) = \sum_{i=1}^n \theta_1 \ln(x_i)^2$$

$$\Rightarrow \theta_1 = \frac{\sum_{i=1}^n y_i \ln(x_i)}{\sum_{i=1}^n \ln(x_i)^2}$$

→ This result we could have obtained by putting $V_i = \ln(x_i)$ & analysing loss function as applying in result from simple linear regression $\theta = \frac{\sum_{i=1}^n y_i x_i}{\sum_{i=1}^n x_i^2}$

prediction $\hat{y} = f_{\theta}(x) = \sum_{j=1}^M \theta_j \underbrace{\phi_j(x)}_{\text{kernel function}} \star$

$M \rightarrow \# \text{parameters}$

$M = \begin{cases} \# \text{features, no bias term} \\ \# \text{features} + 1, \text{ bias term} \end{cases}$

Consider the following training data:

x	y
1	3
2	1
3	0.5

→ non linear hypothesis fn.

Suppose the data comes from a model $\hat{y} = cx^{\beta}$, for unknown constants c and β . Use least squares linear regression to find an estimate of c and β .

$$\text{LSE } L(x) = \sum_{i=1}^3 (y_i - cx_i^{\beta})^2$$

$$\frac{\partial L(x)}{\partial c} = \sum_{i=1}^3 2(y_i - cx_i^{\beta}) \cdot -x_i^{\beta}$$

$$\frac{\partial L(x)}{\partial \beta} = \sum_{i=1}^3 2(y_i - cx_i^{\beta}) \cdot -c \cdot \beta \ln(x_i)$$

$$\text{Normal eq } \nabla L(x) = 0 \Rightarrow$$

$$\sum_{i=1}^3 (y_i - cx_i^{\beta}) x_i^{\beta} = 0 \text{ ie } (3 - c \cdot 1^{\beta}) \cdot 1^{\beta} + (1 - c \cdot 2^{\beta}) \cdot 2^{\beta} + (0.5 - c \cdot 3^{\beta}) \cdot 3^{\beta} = 0$$

Instead of solving this way we can use another method.

$$y_i = cx_i^{\beta} \xrightarrow[\text{ln on L-RHS}]{\text{taking}} \log y_i = \ln(cx_i^{\beta}) = \underbrace{\ln(c)}_{\theta_1} + \underbrace{\beta \ln(x_i)}_{\theta_2 V}$$

$\log x_i$	$\log y_i$
0	$\ln 3$
$\ln 2$	0
$\ln 3$	$\ln 0.5$

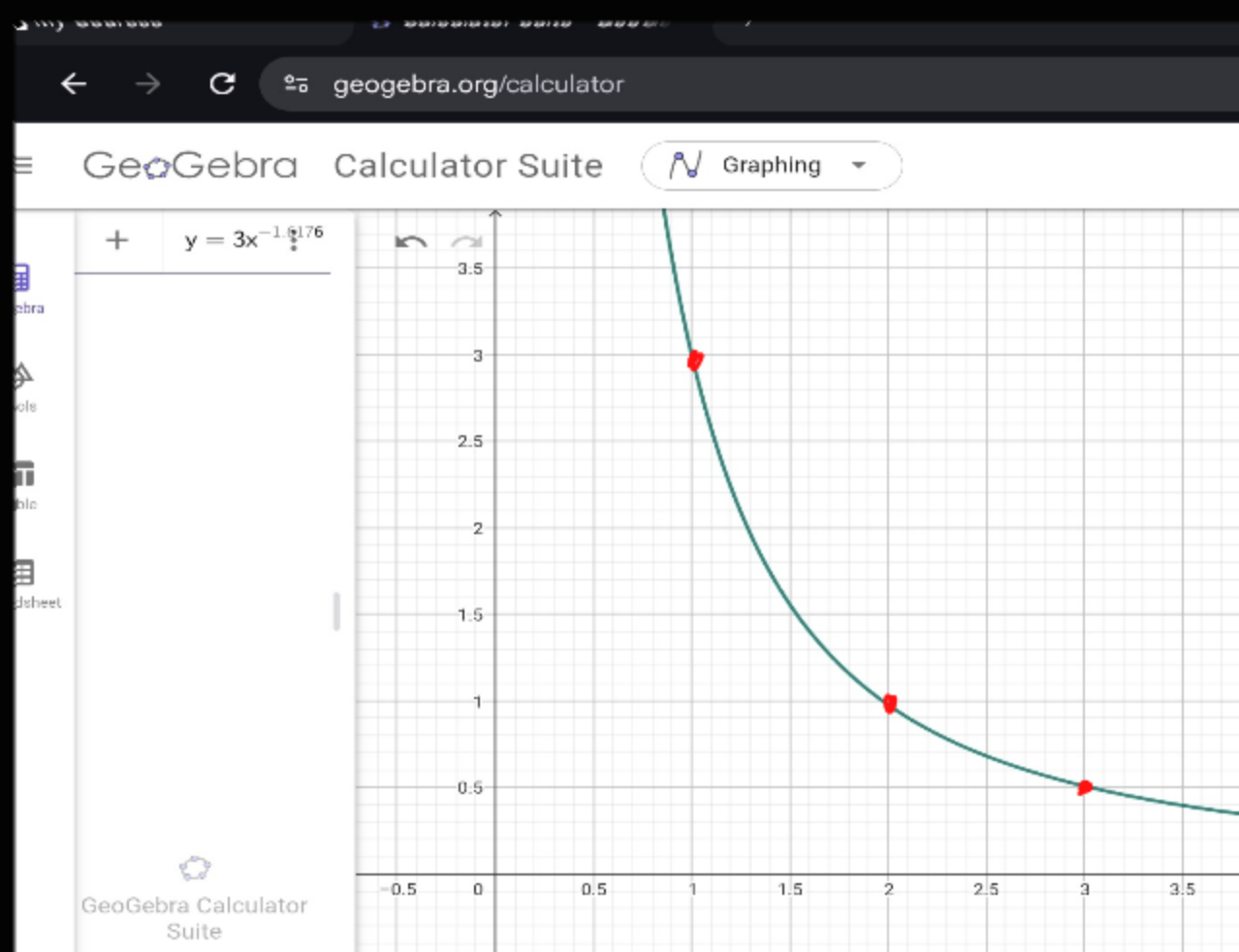
$$V^T V \theta^* = V^T \theta \Rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & \ln 2 & \ln 3 \end{bmatrix} \begin{bmatrix} 0 & \ln 3 \\ \ln 2 & 0 \\ \ln 3 & \ln 0.5 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 3 & \ln 6 \\ \ln 6 & \ln^2 2 + \ln^2 3 \end{bmatrix} \begin{bmatrix} \ln 1.5 \\ -\ln(2/3) \ln(2) \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 3 & 1.791 & 0.4055 \\ 1.791 & 1.6874 & -0.7615 \end{bmatrix} \xrightarrow{R_0 \rightarrow R_0 - 0.517 R_1} \begin{bmatrix} 3 & 1.791 & 0.4055 \\ 0 & 0.6182 & -1 \end{bmatrix}$$

$$\Rightarrow \theta_2 = -1.6176, \theta_1 = 1.1 \Rightarrow \boxed{c = e^{\theta_1} = 3, \beta = -1.6176}$$

$$\Rightarrow y = 3x^{-1.6176}$$



Overfitting

Definition

Overfitting is the phenomenon where the model is unnecessarily complex, in the sense that portions of the model captures the random noise in the observation, rather than the relationship between predictor(s) and response.

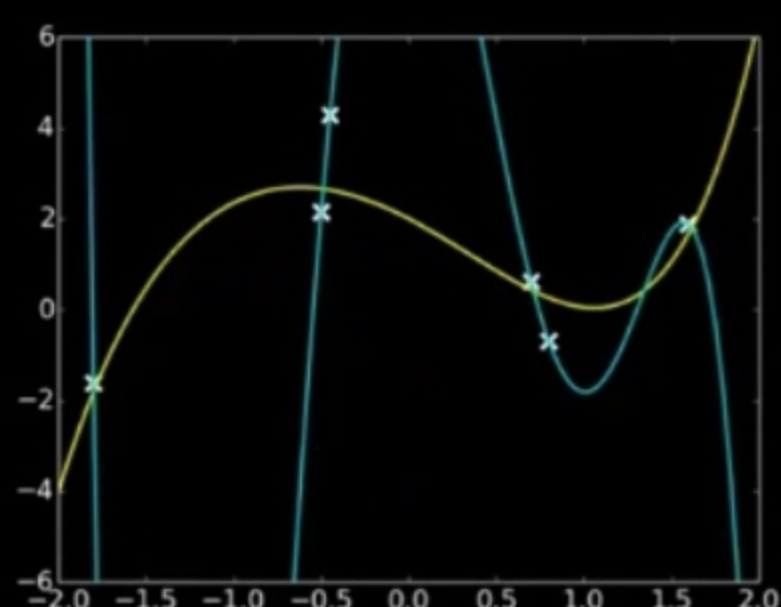
Overfitting causes the model to lose predictive power on new data.

- More features of x than required
- Model captures random noise
- Overfitting ↓ training error but disaster for future data.

Very important observation

→ Polynomials that overfit often have large coefficients

Observation: polynomials that overfit often have large coefficients.



$$y = 0.1x^5 + 0.2x^4 + 0.75x^3 - x^2 - 2x + 2$$

$$y = -7.2x^5 + 10.4x^4 + 24.5x^3 - 37.9x^2 - 3.6x + 12$$

3 overfitting

Why overfitting is bad

- learning noise
- fails to generalise
- performs poorly on test data (future / real world)

small training error large training error

small test error

generalise well
A
performs well

possible, but
B
unlikely

large test error

overfitting
C
fails to generalise

D
generalise well, perform poorly

Another reason to keep weights small

- ▶ Suppose the following two weight vectors, w_A and w_B , make nearly the same predictions:

$$w_A = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad w_B = \begin{pmatrix} -9 \\ 11 \end{pmatrix}$$

- ▶ But why is w_A better than w_B ?
- ▶ w_B is more sensitive—if we change the input even slightly, the output can change rapidly.
- ▶ In contrast, w_A is more stable, providing consistent predictions even with minor input variations.

www.goclasses.in
elinanto5@gmail.com
+91946062414

- Overfitting refers to a model that models the training data too well.
- Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.
- This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model.
- The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.

- Underfitting refers to a model that can neither model the training data nor generalize to new data.
- An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data.
- Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms.
- Nevertheless, it does provide a good contrast to the problem of overfitting.

Underfitting

- does not learn concepts
- fails to generalise
- easier to detect

Model complexity

Just right

- Small training & Test error
- generalises the concepts

dataset size increase

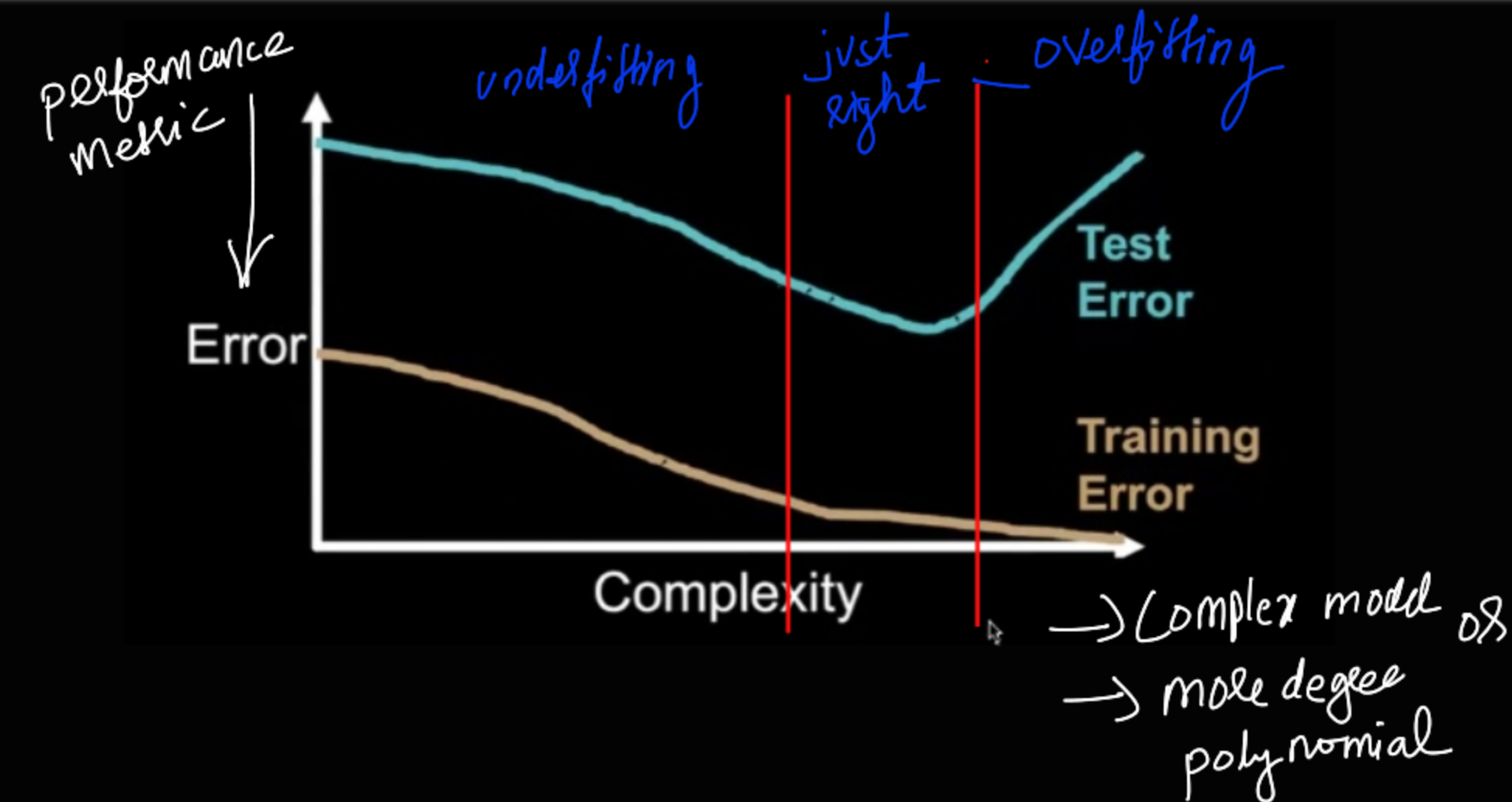
Less chance of underfitting or overfitting

increases

Overfitting

- learns random-fluctuation and noise
- performs poorly on test data compared to training data

dataset size increase



→ underfitting can also happen with correct model but the features are not informative

True/False: A regressor with fixed model complexity trained on a smaller dataset is **less likely** to overfit compared to one trained on a larger dataset.

False. A specific regressor (with fixed model complexity) is **more likely** to overfit to noise in the training data when trained on a smaller dataset. Therefore, it is more likely to overfit.

→ Large data set → fitting for all the noise & fluctuations is difficult to learn
→ Small dataset → Very easy to fit or learn noise & fluctuations

Question: True/False

If there is no noise, then there is no overfitting.

→ Even if given data has no noise and fits perfectly to a simple model if we then train the data on a sufficiently complex model, it may still overfit

False

Imagine that your data is two dimensional, where you predict $Y = f(X)$, where f is unknown, high-degree polynomial function. Say that you take every second datapoint as your training set, and the rest as test set. Your task in here would be simply to connect the dots using polynomial function. This seems simple, but you can easily choose wrong polynomial function that connects all the training samples, but misses the test ones. Since we've all seen hundreds of pictures of overfitting polynomials, I'm omitting the plot.

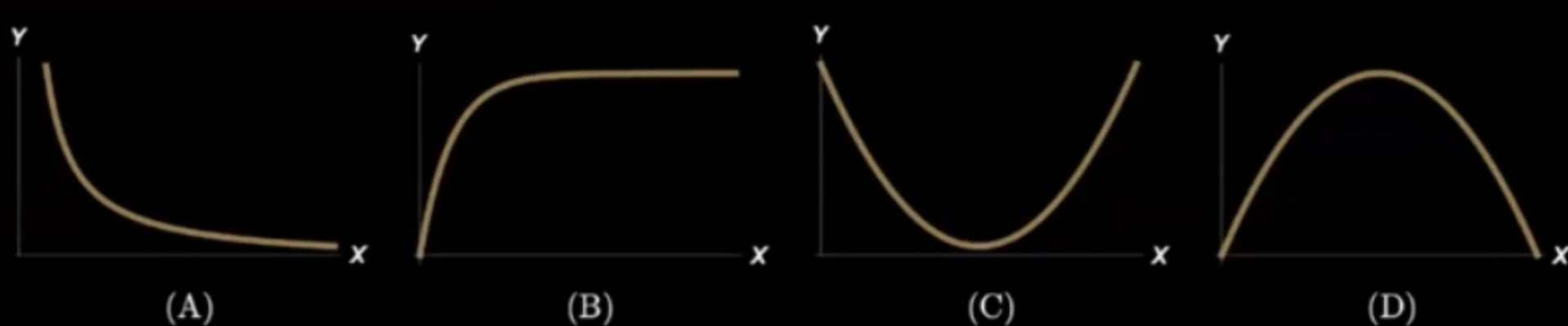
Share Cite Improve this answer Follow

answered Jan 11, 2019 at 18:57
Tim
140k 26 265 507

Yes, it is possible. Without noise, your X, Y pairs should completely satisfy $f(X) = Y$. From your N training points, you can find $N + 1, N + 2, \dots$ degree polynomials to fit in perfectly. And, since you don't know the truth of it, a wrong choice of polynomial degree will result in overfitting.

Generalisation ability → Perform well on unseen data or test data

Generalisation error → Test error



- (a) X axis: Number of training examples; Y axis: Test error.
(b) X axis: Number of training examples; Y axis: Training error.
(c) X axis: Order of polynomial features; Y axis: Test error.
(d) X axis: Order of polynomial features; Y axis: Cross-validation error.

A

B

C

C

Similar to test error

1. If there is no noise, then there is no overfitting.
2. Overfitting may improve the generalization ability of a model.
3. Generalization error is monotone with respect to the capacity/complexity of a model.
4. More training data may help preventing overfitting.

1. We can still have false correlation that leads to overfitting.
2. Overfitting would undermine the generalization ability.
3. Generalization error would first decrease and then increase as the model capacity increases.
4. Increasing training data size would help better approximate the true distribution.

1. Generalization and overfitting. For each of the following statements, which combination of italicized words correctly completes the statement? Circle one for each choice.

- A. If the model has low/high training set error and low/high test set error, the model generalizes.
- B. If the model has low/high training set error and low/high test set error, the model overfits.
- C. If the model has low/high training set error and low/high test set error, the model underfits.
- D. We can fix underfitting by either using a more/less complex model or adding/removing features.
- E. We can fix overfitting by either using a more/less complex model or adding/removing features.

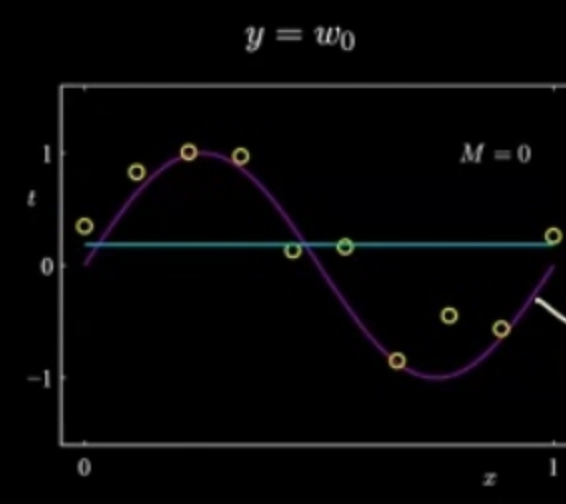
Regularisation → solution to overfitting

$$y = w_0 + w_1x + w_2x^2 \dots + w_Mx^M \quad \text{degree } M \text{ polynomial}$$

$$y = \sum_{i=0}^M w_i x^i \quad M = \text{degree}$$

$$M \geq 0$$

Suppose $M=0 \Rightarrow y=w_0$



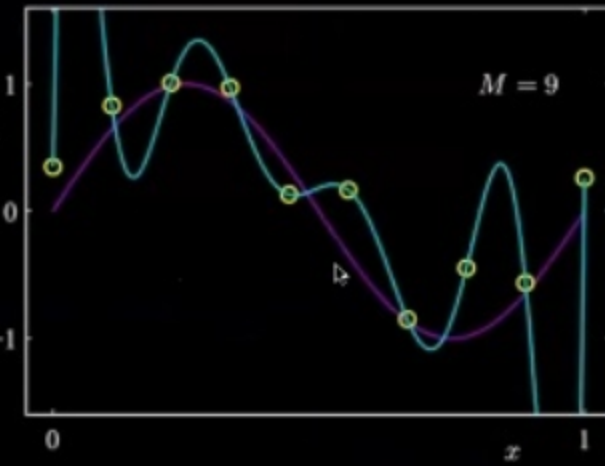
Side quest

True function in ML is the actual function that we are trying to model. This is the best fit function where the data actually came from with measurement errors

→ True function need not cover all data points since data came from noisy measurement

eg.

	M = 0	M = 1	M = 3	M = 9
w_0^*	0.19	0.82	0.31	0.35
w_1^*		-1.27	7.99	232.37
w_2^*			-25.43	-5321.83
w_3^*			17.37	48568.31
w_4^*				-231639.30
w_5^*				640042.26
w_6^*				-1061800.52
w_7^*				1042400.18
w_8^*				-557682.99
w_9^*				125201.43



Observation

→ As M increases, the magnitude of coefficients get larger
→ Between datapoints, the function exhibits large oscillation

Hence during training, instead of minimising the training error $L(w)$ we should minimise $L(w) + \text{???}$

- Some term which increases with overfitting
- Some term which increases with complexity
- Depends on model

Ridge Regression aka regularised linear regression

$$L(w) = \sum_{i=1}^N (y_i - x_i w)^2 + \lambda \sum_{j=1}^{m-1} w_j^2$$

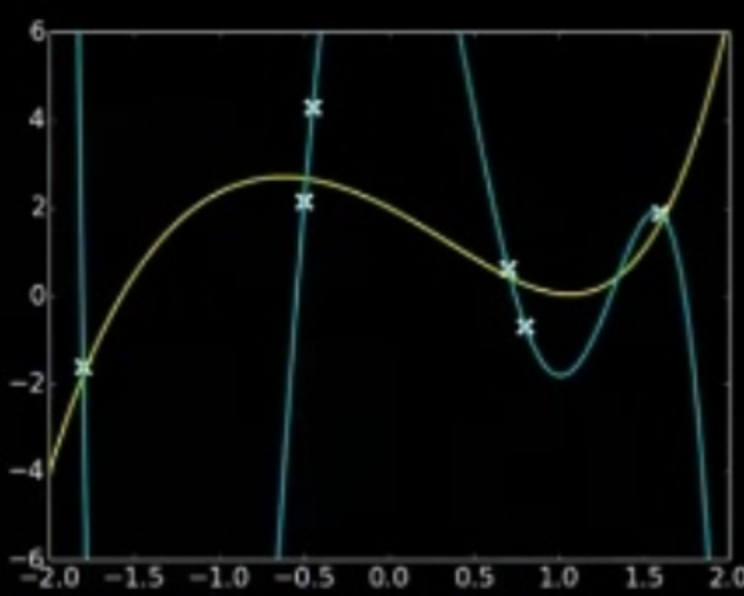
where $\lambda \geq 0$ is the regularisation parameter
→ controls the trade off between fitting & regularisation
→ Unregularised loss

Notice here we are not adding w_0

$$w = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{m-1} \end{bmatrix}_{N \times 1}$$

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

eg.



$$y = 0.1x^5 + 0.2x^4 + 0.75x^3 - x^2 - 2x + 2$$

$$y = -7.2x^5 + 10.4x^4 + 24.5x^3 - 37.9x^2 - 3.6x + 12$$

regularised loss function

conventional linear regression

decreases

$\lambda > 0$

increases

→ Focus on fitting data
→ Overfitting
→ Complex fitting

Optimum λ
→ Optimum complexity
→ Generalised data
→ Low Training & Test error

→ Too much focus on reducing weight
→ Simple fitting
→ Underfitting

Why no regularization on $w_0 \rightarrow$ Bias term?

- $\rightarrow w_0$ should be learned freely
- $\rightarrow w_0$ is bias which does not contribute for overfitting
- $\rightarrow w_0$ adjust for offset in data which should not be regularised

Regularised loss function

$$L(w) = \frac{1}{2} \sum_{i=1}^N (y_i - x_i w)^2 + \frac{\lambda}{2} \sum_{j=1}^{m-1} w_j^2$$

In vectorised form

$$L(w) = \frac{1}{2} (y - Xw)^T (y - Xw) + \frac{\lambda}{2} w^T w$$

} without w_0 term

$$\nabla L(w) = (y - Xw)^T \cdot -X + \lambda w^T$$

$$\nabla L(w) = X^T (Xw - y) + \lambda \cdot w$$

from this it $\nabla L(w) = 0$

$$X^T X w + \lambda w = X^T y$$

$$\Rightarrow (X^T X + \lambda I) w = X^T y$$

$$\Rightarrow w = (X^T X + \lambda I)^{-1} X^T y$$

- It penalizes the models that have a θ_i component to have a magnitude that is too big (either in positive or negative direction)
 - For instance, it prevents the model focusing on only one feature
- As this is square of length, it is called L_2 -regularization
 - It has a set of other fancy names: ridge regression, weight decay, Tikhonov regularization,
- Other regularization techniques are possible: L_1 -regularization (with the absolute values), combinations etc.

$\rightarrow$ Regularization is penalizing unusually large weights. This unusually large weights usually happens when model tried to squeeze the model to fit even the noise in the data

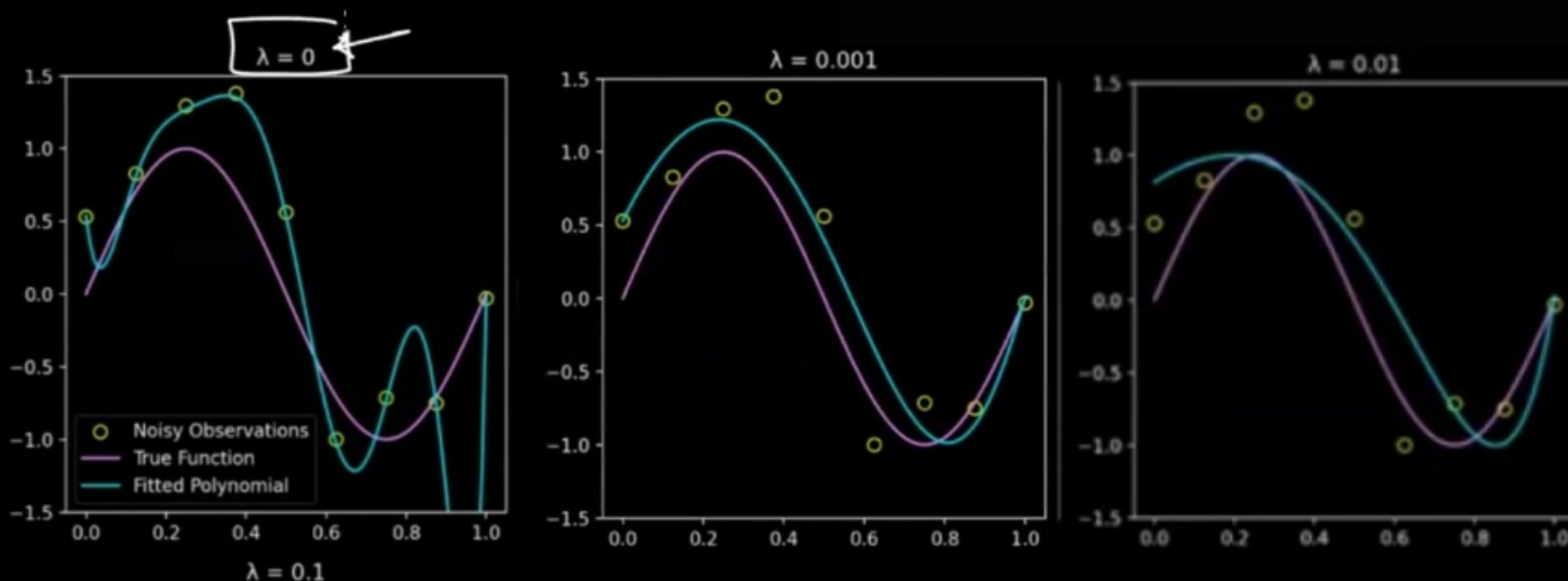
To prevent overfitting of our linear model, we would apply regularization on the model. Which of the following datasets should we evaluate using our model in order to decide the right amount of regularization?

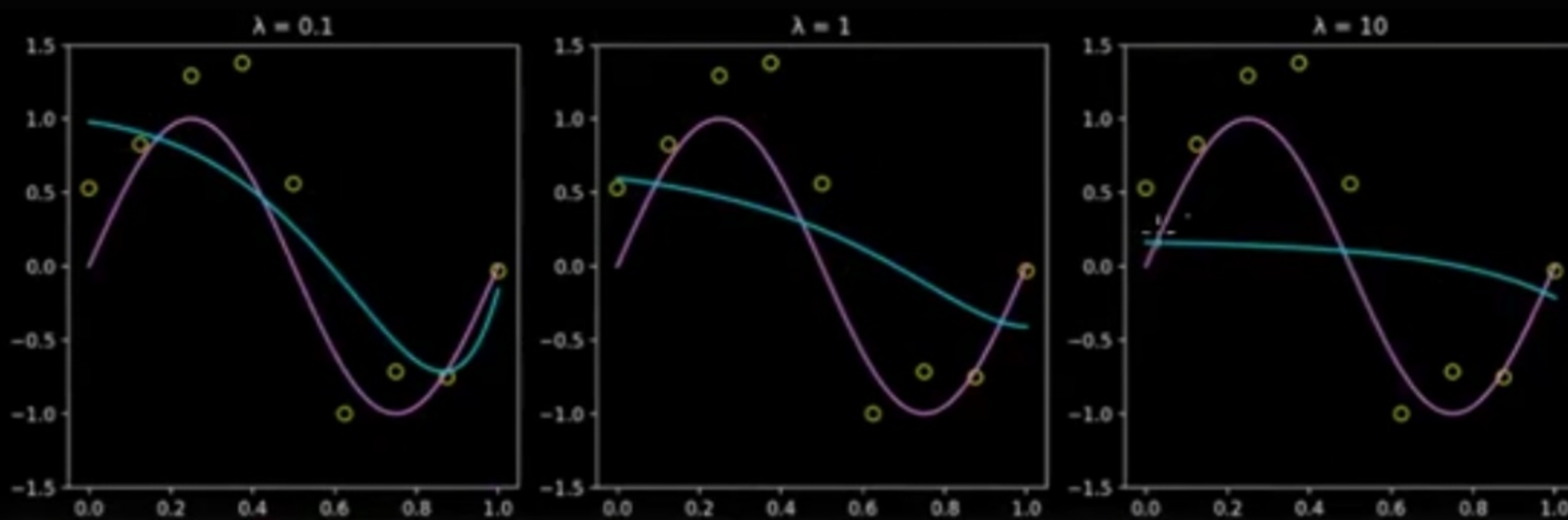
- (A) Train
- (B) Validation ✓
- (C) Test
- (D) All of the above

λ is a hyperparameter
Generally we cannot use test data for tuning hyperparameter even. Instead validation set is set aside specifically for this purpose

$\rightarrow$ Note that λ is a hyperparameter & is generally manually set using validation set - λ does not affect hypothesis function

Fitted polynomial as L_2 regularization weight increases





Observation — Even with highly complex model, regularisation can control it.

(27) For ridge regression, if the regularization parameter is too large, which of the following is true?

- ☐ (A) Large coefficients will not be penalized
- ☒ (B) The model will underfit the data
- ☐ (C) The loss function will be the as same as the ordinary least square loss function
- ☐ (D) The model will overfit the data

(2 points) If the model resulting from Ridge regression is currently overfitting, what are possible things to reduce overfitting? Select all that apply.

- ☒ Collect new data to increase the training data size.
- ☐ Repeat the current data twice to increase the training data size.
- ☒ Increase the ℓ_2 regularization penalty in the loss function.
- ☐ Add new features to the model.
- ☒ Remove features from the model.

Yes, More data $\Rightarrow$ it is difficult for model to learn the noise & less change of overfitting.

(g) [2 pts] Consider the L_2 regularized loss function for linear regression $L(w) = \frac{1}{2} \|Y - Xw\|^2 + \lambda \|w\|^2$, where λ is the regularization parameter. The Hessian matrix $\nabla_w^2 L(w)$ is

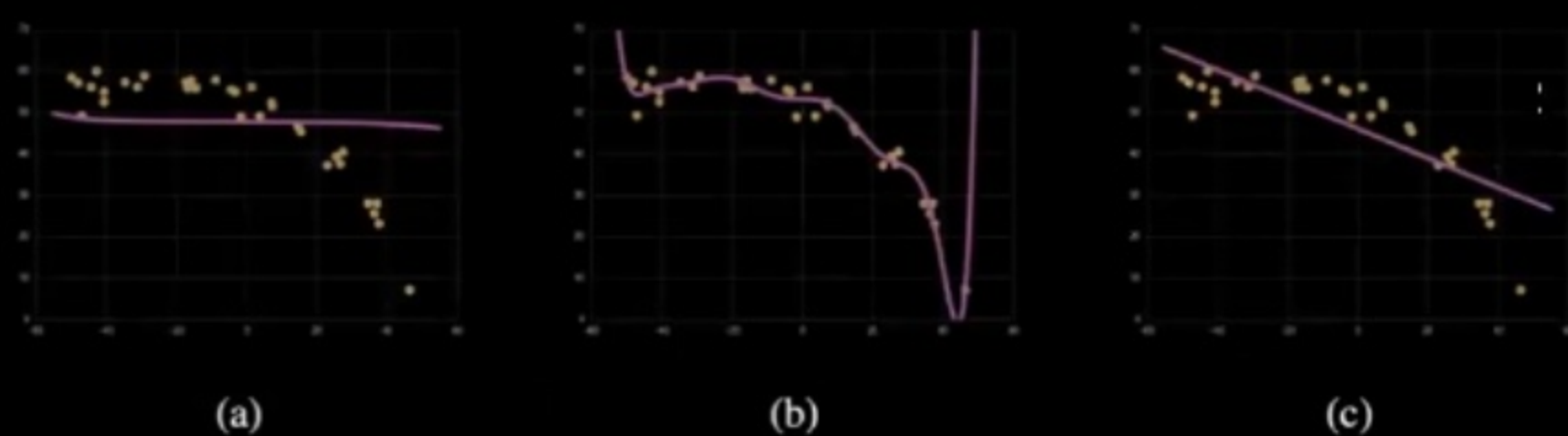
- ☐ $X^T X$
- ☐ $2\lambda X^T X$
- ☒ $X^T X + 2\lambda I$
- ☐ $(X^T X)^{-1}$

We know in this case

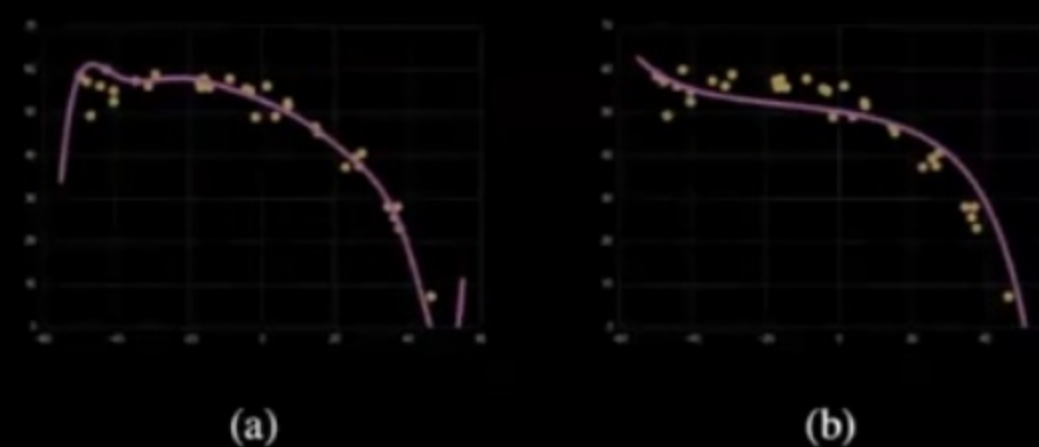
$$\nabla L(w) = X^T(Xw - y) + 2\lambda w$$

$$\nabla^2 L(w) = X^T X + 2\lambda I$$

(1) [3 Pts.] We have fit several models depicted as curves in the following plots:



(2) [2 Pts.] We fit two more models to these data. Again, the solid curves display the predictions made by each model.



Select the plot that best matches each of the models below. Each plot is used exactly once.

- Linear regression model
☐ (A) ☐ (B) ☒ (C)
- Linear regression with degree 10 polynomial features
☐ (A) ☒ (B) ☐ (C)
- Ridge regression with degree 10 polynomial features and substantial regularization.
☒ (A) ☐ (B) ☐ (C)

Select the plot that best matches each of the models below. Each plot is used exactly once.

- Ridge regression with degree 10 polynomial features, $\lambda = 0.1$.
☒ (A) ☐ (B)
- Ridge regression with degree 10 polynomial features, $\lambda = 1.0$.
☐ (A) ☒ (B)

Good practices to avoid overfitting include: **

- ☒ Using a two part cost function which includes a regularizer to penalize model complexity.
- ☐ Using a good optimizer to minimize error on training data.
- ☒ Building a structure of nested subsets of models and train learning machines in each subset, starting from the inner subset, and stopping when the cross-validation error starts increasing.
- ☐ Discarding 50% of randomly chosen samples.

optimizer refers to the learning algorithm like gradient descent, backpropagation etc. Newton method.

[3 pts] Given a design matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and labels $\mathbf{y} \in \mathbb{R}^n$, which of the following techniques could potentially decrease the empirical risk on the training data (assuming the loss is the squared error)?

- ☒ Adding the feature "1" to each data point.
 Value of Loss
 adding bias w_0
- ☒ Adding polynomial features to each data point.
 means more fit

☐ Penalizing the model weights with L_2 regularization.

$\rightarrow$ adding $\left(\sum_{j=1}^{n-1} w_j \right)$ does not decrease $L(w)$
 always +ve

True/False: Ridge regression can shrink all coefficients to exactly 0 if the regularization parameter λ is large enough.

- (A) True
 $\rightarrow$ high λ can decrease w low but exactly 0 happens only @ $\lambda = \infty$
- ☒ (B) False

(4) In a ridge regression model parameterized by w , what is the penalty term?

- (A) The square of the magnitude of w 's coefficients ☒
 (B) The square root of the magnitude of w 's coefficients
 (C) The absolute sum of w 's coefficients
 (D) The sum of w 's coefficients

Consider a new objective function with an added regularization term:

$$J_3(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^n (\theta^T x^{(i)} + \theta_0 - y^{(i)})^2 + 0.2(\|\theta\|^2 + \theta_0^2)$$

Will adding this particular form of regularization in J_3 improve the generalization of our model? Explain your answer.

$\rightarrow$ No, because we are penalising bias aka intercept component. Penalising bias is never good because model has to learn offset in data, without any limitation.

How does the irreducible error change if we increase the regularization coefficient λ in ridge regression?

- (A) Increase
 (B) Decrease
 ☒ (C) Not change
 (D) The answer depends on the dataset X and true weights w^* .

noise
 $\rightarrow$ refers to noise in measurement, not the error in model prediction
 $\rightarrow$ regularization only affects prediction error, inherent error in true value y is unchanged

3. Consider the least squares regression model, $\hat{\mathbf{Y}} = \mathbf{X}\theta$. Assume that $\mathbf{X}$ and $\mathbf{Y}$ refer to the design matrix and true response vector for our training data.

Let $\hat{\gamma}$ be the parameter vector that minimizes mean squared error without regularization. Specifically:

$$\hat{\gamma} = \arg \min_{\gamma} \frac{1}{n} \|\mathbf{Y} - \mathbf{X}\gamma\|_2^2$$

Let $\hat{\beta}$ be the parameter vector that minimizes mean squared error with L_2 regularization, using a non-negative regularization hyperparameter λ (i.e. ridge regression). Specifically:

$$\hat{\beta} = \arg \min_{\beta} \frac{1}{n} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \sum_{j=1}^p \beta_j^2$$

b) [2 Pts] For each of the remaining parts, you can assume that λ is set such that the predicted response vectors for our two models ($\hat{\mathbf{Y}} = \mathbf{X}\hat{\gamma}$ and $\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta}$) is different.

The **training** RMSE of the model $\hat{\mathbf{Y}} = \mathbf{X}\hat{\gamma}$ is _____ than the model $\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta}$.

- ☒ A. less than
 ☐ B. equal to
 ☐ C. greater than
 ☐ D. impossible to tell

c) [2 Pts] Now, assume we've fit both models using our training data, and evaluate both models on some unseen testing data.

The **test** RMSE of the model $\hat{\mathbf{Y}} = \mathbf{X}\hat{\gamma}$ is _____ than the model $\hat{\mathbf{Y}} = \mathbf{X}\hat{\beta}$.

- ☐ A. less than
 ☐ B. equal to
 ☐ C. greater than
 ☒ D. impossible to tell

$\rightarrow$ Because we don't know if regularization/ridge regression is helping or whether OLS is overfitting

(11 points) Regression

Suppose you are interested in predicting a one-dimensional quantitative random variable Y (outcome) in terms of a two-dimensional quantitative random variable X (covariates). You have a learning set $(x_1, y_1), \dots, (x_3, y_3)$ of three observations, given in the table to the right.

i	x_i	y_i
1	$\begin{bmatrix} 0 \\ 3 \end{bmatrix}$	-3
2	$\begin{bmatrix} 3 \\ 0 \end{bmatrix}$	0
3	$\begin{bmatrix} 6 \\ 0 \end{bmatrix}$	3

(2 pt) There are two derivations below: left and right. Select the one that expresses the regularized empirical risk (mean squared error) for ridge regression for this dataset in terms of β_1 and β_2 when $\lambda = 1$.

- ☐ $\frac{1}{3} [(-3+3\beta_2)^2 + (3\beta_1)^2 + (3+6\beta_1)^2] + (\beta_1^2 + \beta_2^2)$
 $= 16\beta_1^2 + 12\beta_1 + 4\beta_2^2 - 6\beta_2 + 6$
- ☒ $\frac{1}{3} [(-3-3\beta_2)^2 + (-3\beta_1)^2 + (3-6\beta_1)^2] + (\beta_1^2 + \beta_2^2)$
 $= 16\beta_1^2 - 12\beta_1 + 4\beta_2^2 + 6\beta_2 + 6$

(3 pt) Find the β_1 and β_2 that minimize the regularized empirical risk objective from part (c) above. Show your work for full credit.

$$\beta_1 = \boxed{} \quad \beta_2 = \boxed{} \quad \div$$

23. [6 Pts] What parameter estimate would minimize the following regularized loss function:

$$\ell(\theta) = \lambda(\theta - 4)^2 + \frac{1}{n} \sum_{i=1}^n (x_i - \theta)^2 \quad \text{Solution} \quad \nabla \ell(\theta) = 2\lambda(\theta - 4) - \frac{1}{n} \sum_{i=1}^n 2(x_i - \theta)$$

- ☐ $\hat{\theta} = \frac{1}{\lambda n} \sum_{i=1}^n x_i$
- ☐ $\hat{\theta} = 4 + \frac{1}{\lambda n} \sum_{i=1}^n x_i$
- ☐ $\hat{\theta} = \frac{1}{n(\lambda+1)} \sum_{i=1}^n x_i$
- ☐ $\hat{\theta} = \frac{\lambda}{\lambda+1} + \frac{1}{n(\lambda+1)} \sum_{i=1}^n (x_i - 4)$
- ☒ $\hat{\theta} = \frac{4\lambda}{\lambda+1} + \frac{1}{n(\lambda+1)} \sum_{i=1}^n x_i$

$$\nabla \ell(\theta) = 0 \Rightarrow 2\lambda\theta + 2\theta = \frac{2}{n} \sum_{i=1}^n x_i + 8\lambda$$

$$\Rightarrow \theta = \frac{\frac{2}{n} \sum_{i=1}^n x_i + 8\lambda}{2\lambda + 2}$$

$$\Rightarrow \theta = \frac{1}{(\lambda+1)n} \sum_{i=1}^n x_i + \frac{4\lambda}{(\lambda+1)}$$

Which model do you prefer, assuming both have zero training error?

Model structure (for both models):

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4 + \theta_5 x^5 + \theta_6 x^6 + \theta_7 x^7 + \theta_8 x^8$$

Model parameters:

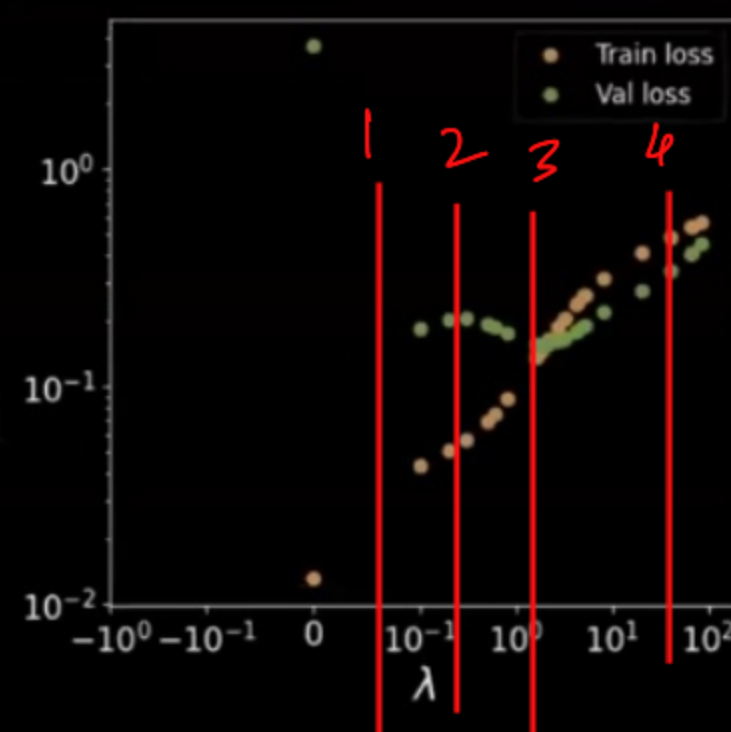
$$\theta = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8]^T$$

A. $\theta_A = [-190.0, -135.0, 310.0, 45.0, -62.0, 90.0, -82.0, -40.0, 29.0]^T$

B. $\theta_B = [25.5, -6.4, -0.8, 0.0, 6.6, -4.4, 0.2, -2.9, 0.1]^T$ ✓ Less weights $\rightarrow$ more stable, no overfitting

What is the best value for lambda?

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} J(\theta) + \lambda r(\theta)$$



λ @ Line 3

$\rightarrow$ Validation error is minimum here
 $\rightarrow$ Complexity is just right.

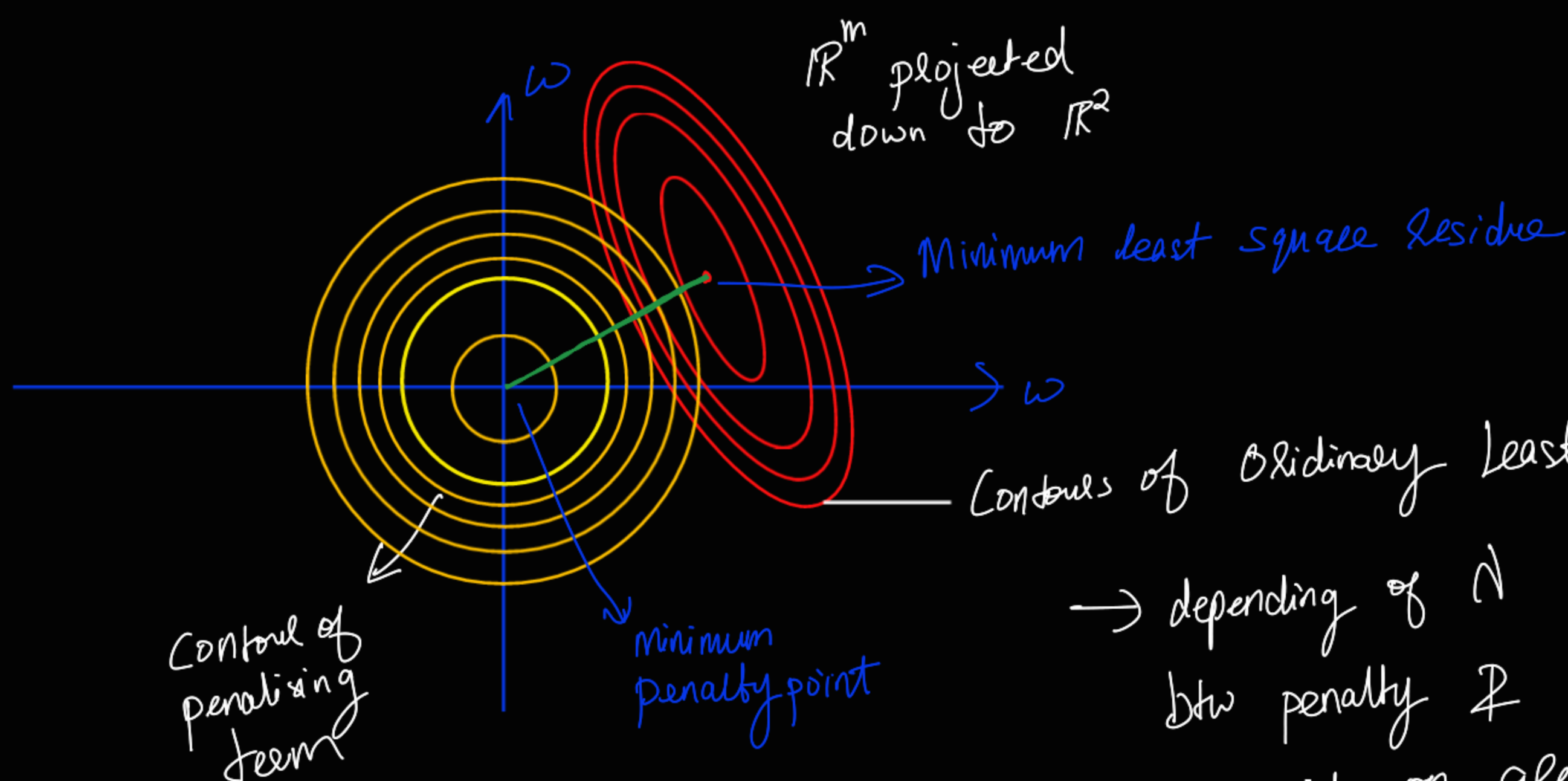
L_2 Regularised loss function $\rightarrow$ Ridge regression

parameter space $\mathbb{R}^m$
 $m = \# \text{ parameters}$

$$L(w) = \sum_{i=1}^N (y_i - x_i w)^2 + \lambda w^T w$$

$\rightarrow$ convex function
 not centered at
 origin in $\mathbb{R}^m$

$\rightarrow$ convex
 function centered
 at origin in $\mathbb{R}^m$



$\rightarrow$ depending of λ the final compromise
 b/w penalty & LS cost will be
 a point on green line

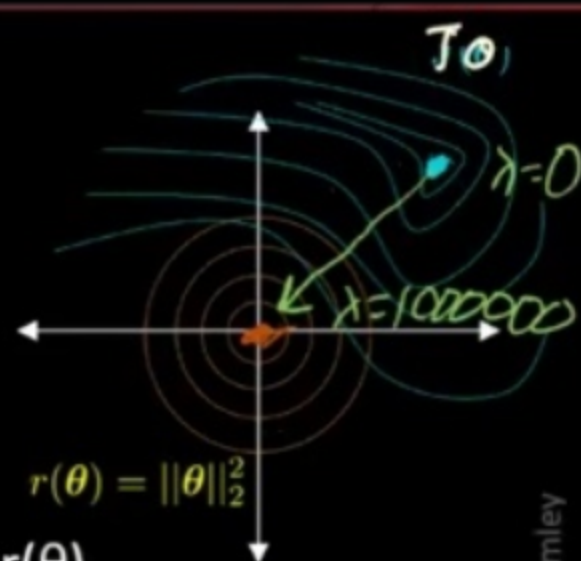
Poll 4

Suppose we are minimizing $J'(\theta)$ where

$$J'(\theta) = J(\theta) + \lambda r(\theta)$$

As λ increases, the minimum of $J'(\theta)$ will...

- A. ...move towards the midpoint between $J'(\theta)$ and $r(\theta)$
- B. ...move towards the minimum of $J(\theta)$
- ☒ C. ...move towards the minimum of $r(\theta)$ ✓
- D. ...move towards a theta vector of positive infinities
- E. ...move towards a theta vector of negative infinities
- F. ...stay the same



Slide credit: CMU MLD Matt Gormley

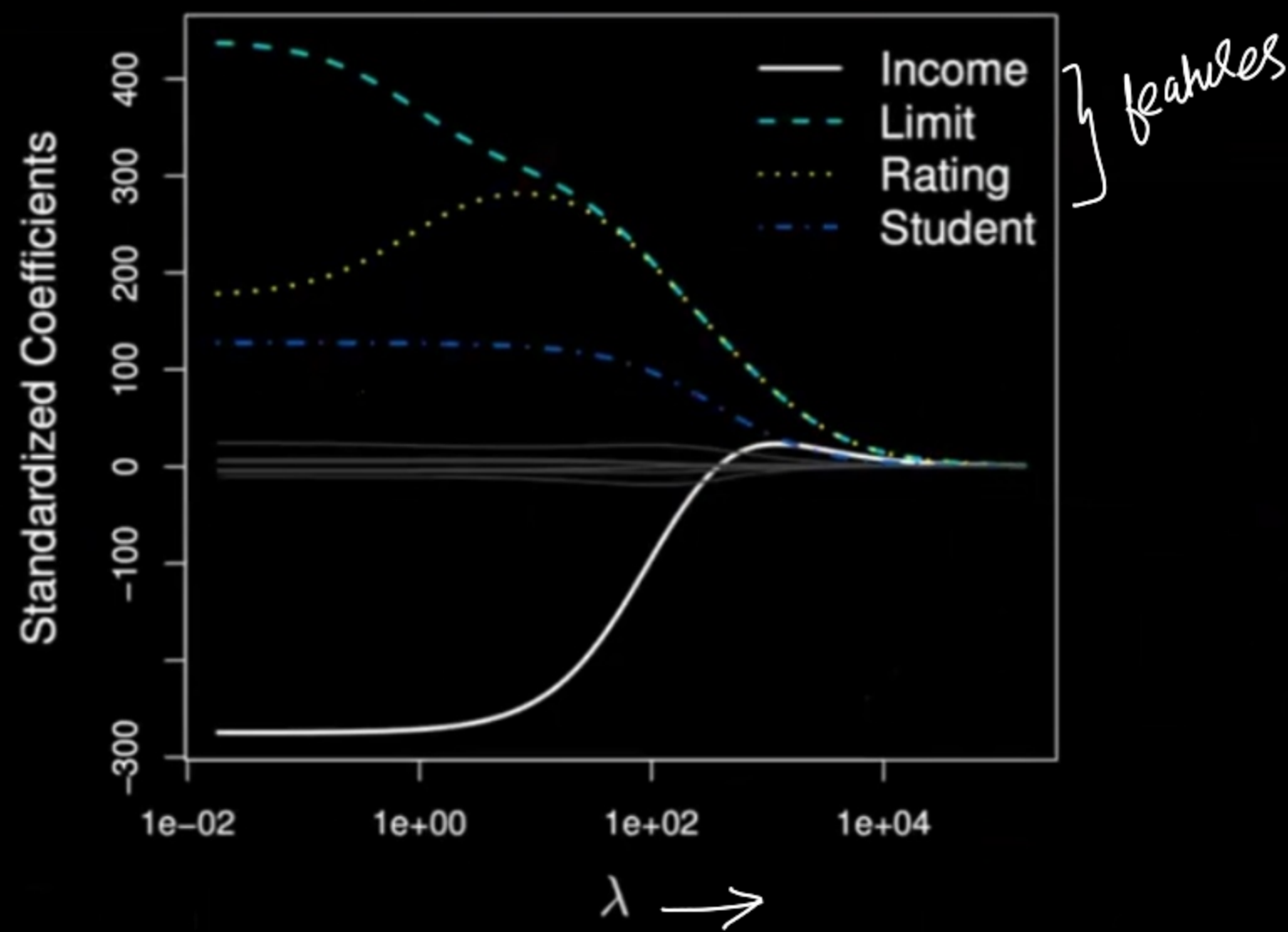
Regularization with L_1 norm
 $\rightarrow$ LASSO regression

$$\text{Loss function} = \sum_{i=1}^N (y_i - x_i w)^2 + \lambda \sum_{j=1}^{m-1} |w_j|$$

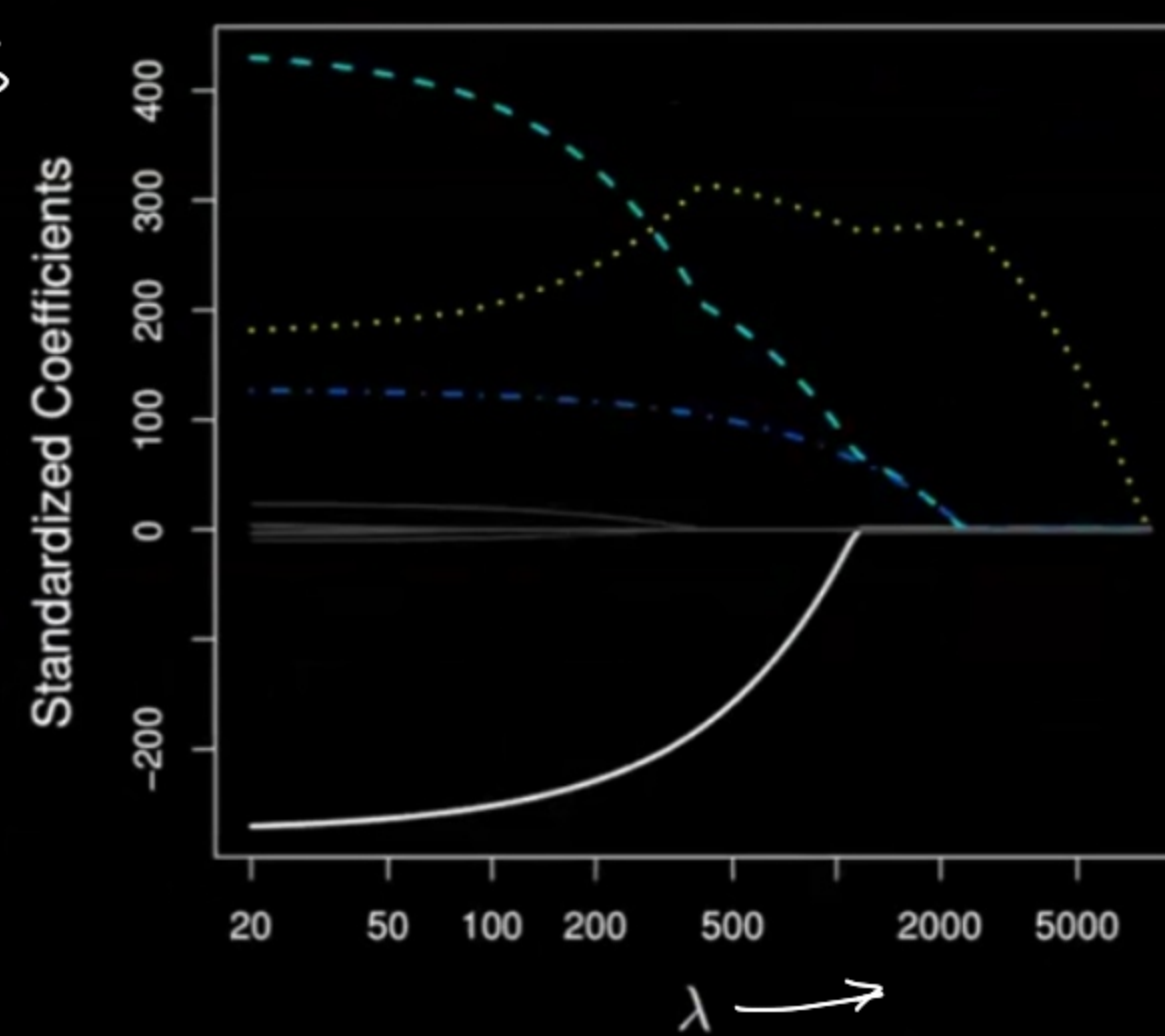
} NO closed form solution

$\rightarrow$ not differentiable @ $w=0$
 we can differentiate in
 piecewise form

Why we need LASSO REGRESSION?



Ridge Regression



The Lasso

Observation

- Increasing $\lambda \rightarrow$ Values of parameters w^* decreases
- At extreme λ , parameters are close to zero in L_2 regularisation
- At extreme λ , certain parameters are exactly zero in L_1 regression

Consider a problem where 3 vectors give minimum training error:

$$w_1 = \begin{bmatrix} 100 \\ 0.02 \end{bmatrix} \quad w_2 = \begin{bmatrix} 100 \\ 0 \end{bmatrix} \quad w_3 = \begin{bmatrix} 99.99 \\ 0.02 \end{bmatrix}$$

L_1 L_2

MS Cost same for all 3

Observation

- ▶ Without regularization, we could choose any of these 3.
- ▶ They all have the same error, so regularization will "break the tie".
- Meanwhile L_1 being a mod function doesn't give special attention to highest weight.
- L_1 regularisation promotes sparsity by driving some parameters to exactly zero.

- L_2 will try to minimise the highest weight on priority. By choosing the optimum point that has negligibly small components in features that are clearly not relevant.
- Since reducing δ from the highest weight is advantageous even if it has to tolerate 2δ in the weight almost close to zero weight.

• **L1-regularization gives sparsity but L2-regularization doesn't.**

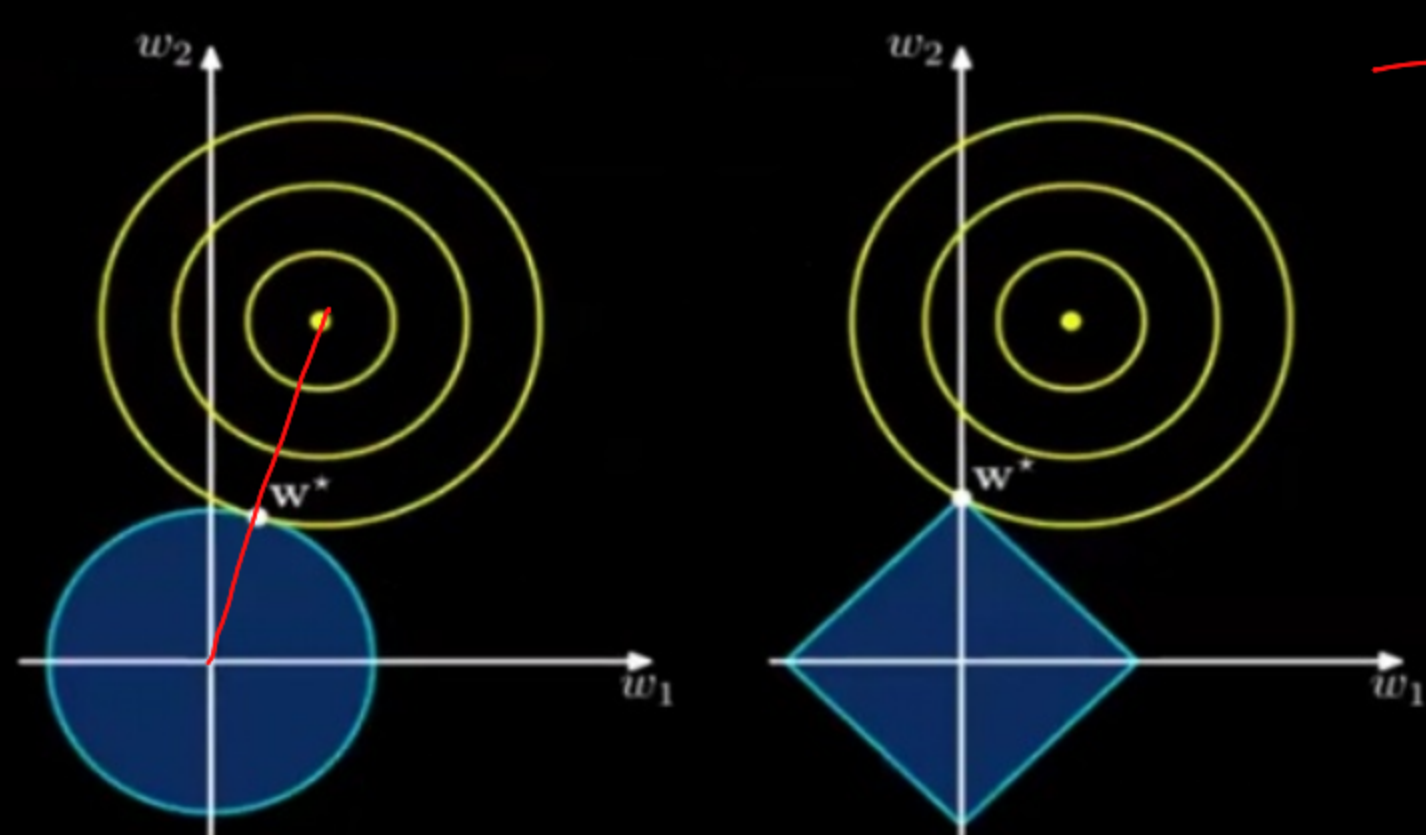
– But don't they both shrink variables towards zero?

i the instructors' answer, where instructors collectively construct a single answer

Imagine using L_2 regularization and you have a $w_j = 0.00001$. What's the penalty for this? Well, it's $(0.00001)^2 = 0.0000000001$. In other words, insanely tiny. So, there's little "incentive" (in terms of decreasing the loss) to set it to zero. And, critically, as you get closer to zero the "incentive" decreases further. That is exactly because of the smoothness of the function: the slope is going to zero smoothly, and the slope/derivative is exactly the "incentive to move even closer to zero".

With a non-smooth function like L_1 on the other hand, the slope stays constant the whole time, it doesn't diminish to zero. So even when the weight is really tiny, there's still something to be gained by making it even tinier... all the way until you're at zero.

- $L_2 \rightarrow$ less incentive to reduce tiny weights
- $L_1 \rightarrow$ equal incentive no matter the relative magnitude of weight



L2 regularization

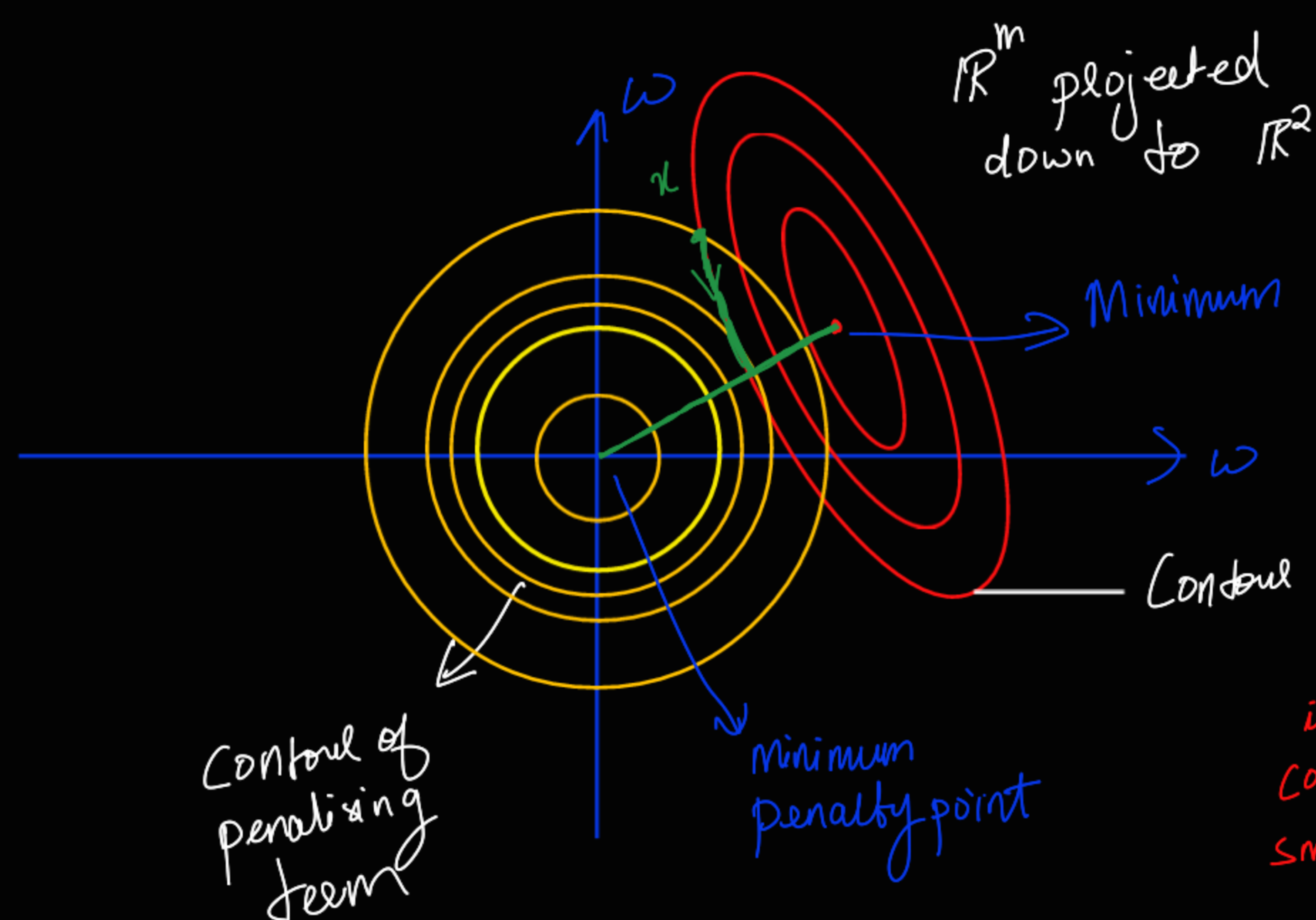
$$R = \sum_i w_i^2$$

L1 regularization

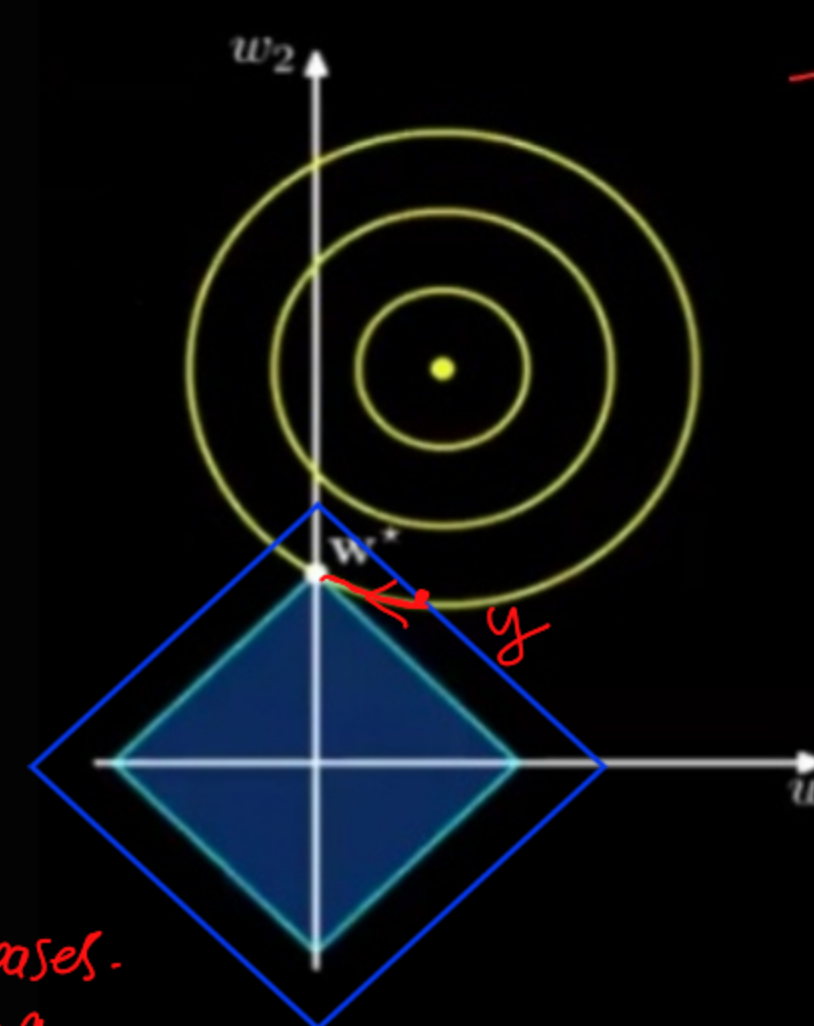
$$R = \sum_i |w_i|$$

→ L_2 regularisation always finds optimum solution along this red line, because any point x outside the line connecting two minima can further be minimised by moving along one of the contours.

— Similarly for L_1 regularisation the minimum point will most probably be along one of the axes in 2 dimension → equivalently, the same applies in more dimension



Insight
In L_1 , the size of diamond reduces as λ increases. so more chance of touching in parameter axes only compared to when λ is small.



L1 regularization

$$R = \sum_i |w_i|$$

Observation → L_1 promotes sparsity because it has high chances of finding the optimum point (touching point) in parameter axes in parameter space $\mathbb{R}^m$ ($m = \# \text{parameter}$)

- Ridge regression can not shrink all coefficient to exactly 0 even if regularization λ is large enough.
- Lasso regression can shrink all coefficients to exactly 0 if regularization λ is large enough.

(k) [4 pts] We are solving a least-squares linear regression problem without regularization. Suppose that the following two weight matrices both have the same cost: $w_1 = \begin{bmatrix} 1 \\ 0 \\ 6 \end{bmatrix}$ and $w_2 = \begin{bmatrix} -2 \\ 3 \\ 3 \end{bmatrix}$. That is, $\text{RSS}(w_1) = \text{RSS}(w_2)$. Which of the following statements become true if regularization is incorporated into the regression?

☒ A: If l_2 regularization is added with sufficiently high λ , w_1 will be preferred over w_2 .

☒ C: If l_1 regularization is added with sufficiently high λ , w_1 will be preferred over w_2 .

☒ B: If l_∞ regularization is added with sufficiently high λ , w_1 will be preferred over w_2 . (Recall that $\|w\|_\infty = \max_i |w_i|$.)

☒ D: If l_2 regularization is added with $\lambda > 0$, there might be infinitely many weight vectors that minimize $\text{RSS}(w)$.

$\|w\|_\infty = \max_i (|w_i|)$ → Component with the maximum magnitude

→ Adding l_2 regularization (Ridge) guarantees a unique solution w .

(e) [4 pts] Which of the following statements are true about Lasso and ridge regression?

- ☒ A: Both ridge regression and Lasso are methods used to reduce overfitting that might occur in standard linear regression.
- ☒ B: The ℓ_1 -norm regularization used in Lasso has a tendency to induce sparsity in the weight vector.
- ☐ C: Both ridge regression and Lasso have a cost function with a minimizing weight vector w^* that we can write as a closed-form algebraic expression.
- ☒ D: Ridge regression shrinks the weight vector but rarely drives its components to exactly zero.

Suppose you're using L2 regularization on a least squares objective. Some value λ^* will give you the best test error among all possible λ . You train your model using a $\lambda' \ll \lambda^*$. Which of the following do you expect to be true about your training error?

- ☒ a. The training error for λ' will be much smaller than for λ^*
- b. The training error for λ' will be much bigger than for λ^*
- c. The training error for each will be about the same – we regularize for the effect on test error.

Solution:

a, if we're not regularizing as much as we should, then we are likely to be overfitting to our training set – thus our training error will be smaller with λ' .

For a linear regression model, start with random values for each coefficient. The sum of the squared errors is calculated for each pair of input and output values. A learning rate is used as a scale factor, and the coefficients are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error is achieved or no further improvement is possible. What is this process called?

- (a) LASSO
- ☒ (b) Gradient Descent
- (c) Least squares
- (d) Regularization

MSE

(2 points) Which of the following statements are **TRUE** regarding Lasso and Ridge regression? Let $X \in \mathbb{R}^{n \times d}$ be the data matrix and $y \in \mathbb{R}^n$ be the observed labels for the n examples. Let $w \in \mathbb{R}^d$ be the weight parameters learned from Lasso or Ridge regression, and let λ be the regularization coefficient.

- ☒ In Lasso regression, as the regularization coefficient becomes very large ($\lambda \rightarrow \infty$), the learned weights from Lasso regression will be close to zero ($w \rightarrow 0$). *It will be exactly zero*
- ☒ Lasso regression performs both feature selection and regularization.

Please assign each plot in Figure 2 to one (and only one) of the following regularization methods. You can get some help from Figure 3. Please answer A, B, C or D.

- (a) (2 pts) No regularization (or regularization parameter equals to 0).

$$\sum_{i=1}^3 (y_i - \theta_1 x_i - \theta_0)^2$$

OLS C

- (b) (3 pts) L-2 regularization with λ being 5.

$$\sum_{i=1}^3 (y_i - \theta_1 x_i - \theta_0)^2 + \lambda(\theta_1^2 + \theta_0^2) \text{ where } \lambda = 5$$

Ridge ($\lambda=5$) D

- (c) (3 pts) L-1 regularization with λ being 5.

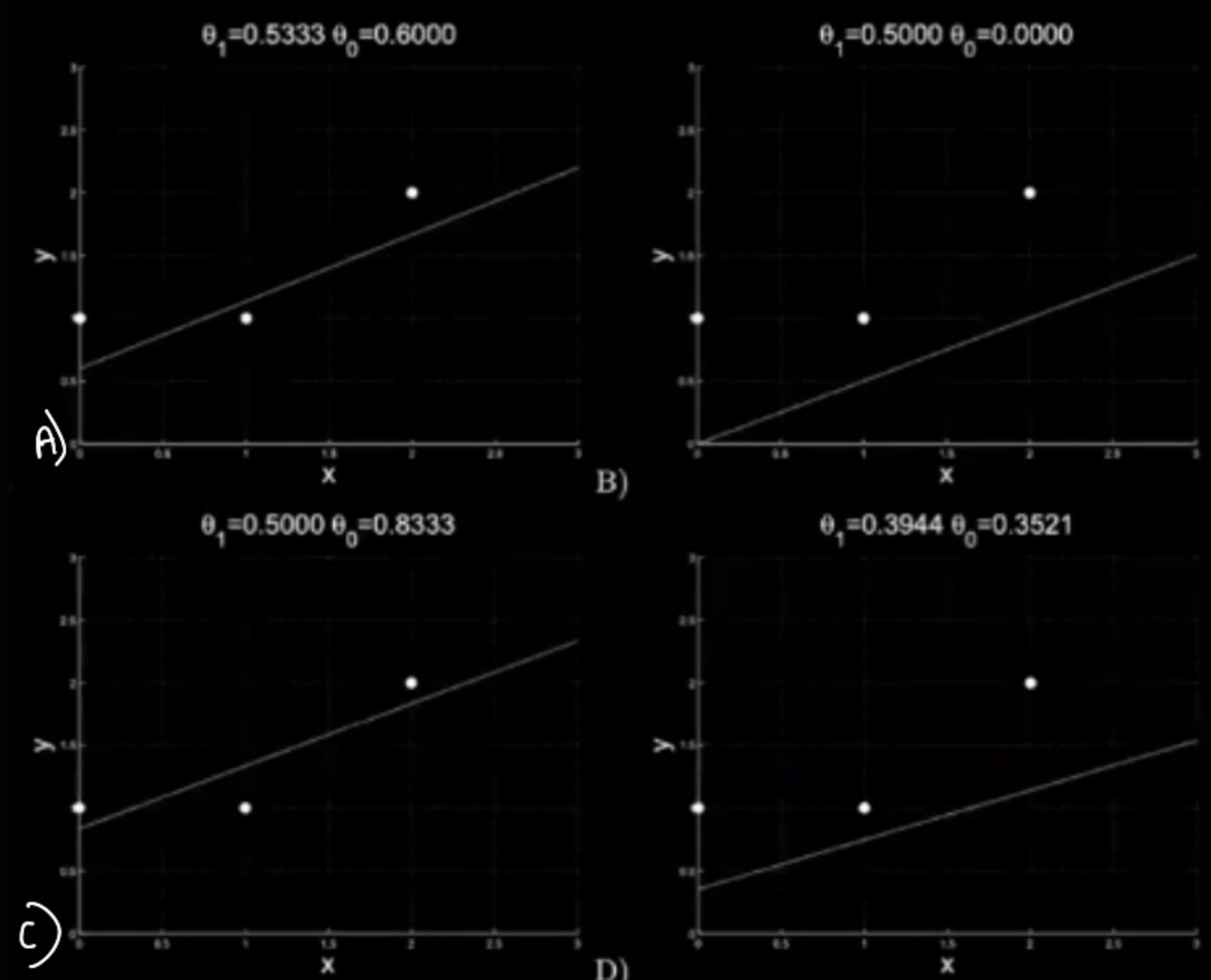
$$\sum_{i=1}^3 (y_i - \theta_1 x_i - \theta_0)^2 + \lambda(|\theta_1| + |\theta_0|) \text{ where } \lambda = 5$$

LASSO ($\lambda=5$) B

- (d) (2 pts) L-2 regularization with λ being 1.

$$\sum_{i=1}^3 (y_i - \theta_1 x_i - \theta_0)^2 + \lambda(\theta_1^2 + \theta_0^2) \text{ where } \lambda = 1$$

Ridge ($\lambda=1$) A



True/False: In LASSO regression, if the regularization parameter λ is very large and two informative features are highly collinear (i.e., that there exists an α such that $x_{ij} \approx \alpha x_{ik}$ for all $i \in [n]$), then LASSO will assign one of those coefficients to zero while ridge regression never will.

- ☒ (A) True
- (B) False

Intuition Lets say 2 parameters are exactly equal
ie $x_{ij} = x_{ik} \forall i \in [n]$; k th & j th features are same. Now their coefficients are w_j & w_k
Now assume the true function has $y = 2w_k = 2w_j$
L2 $\rightarrow$ it will put $w_j = w_k = 1$ so that $1^2 + 1^2 = 2$
it will not put $w_j = 2$ & $w_k = 0$ since $2^2 + 0^2 = 4$
 $\rightarrow$ L2 will try to balance weights w_j & w_k to minimise square of weight
L1 $\rightarrow$ L1 doesn't care about square so L1 forces one of the weight to 0.

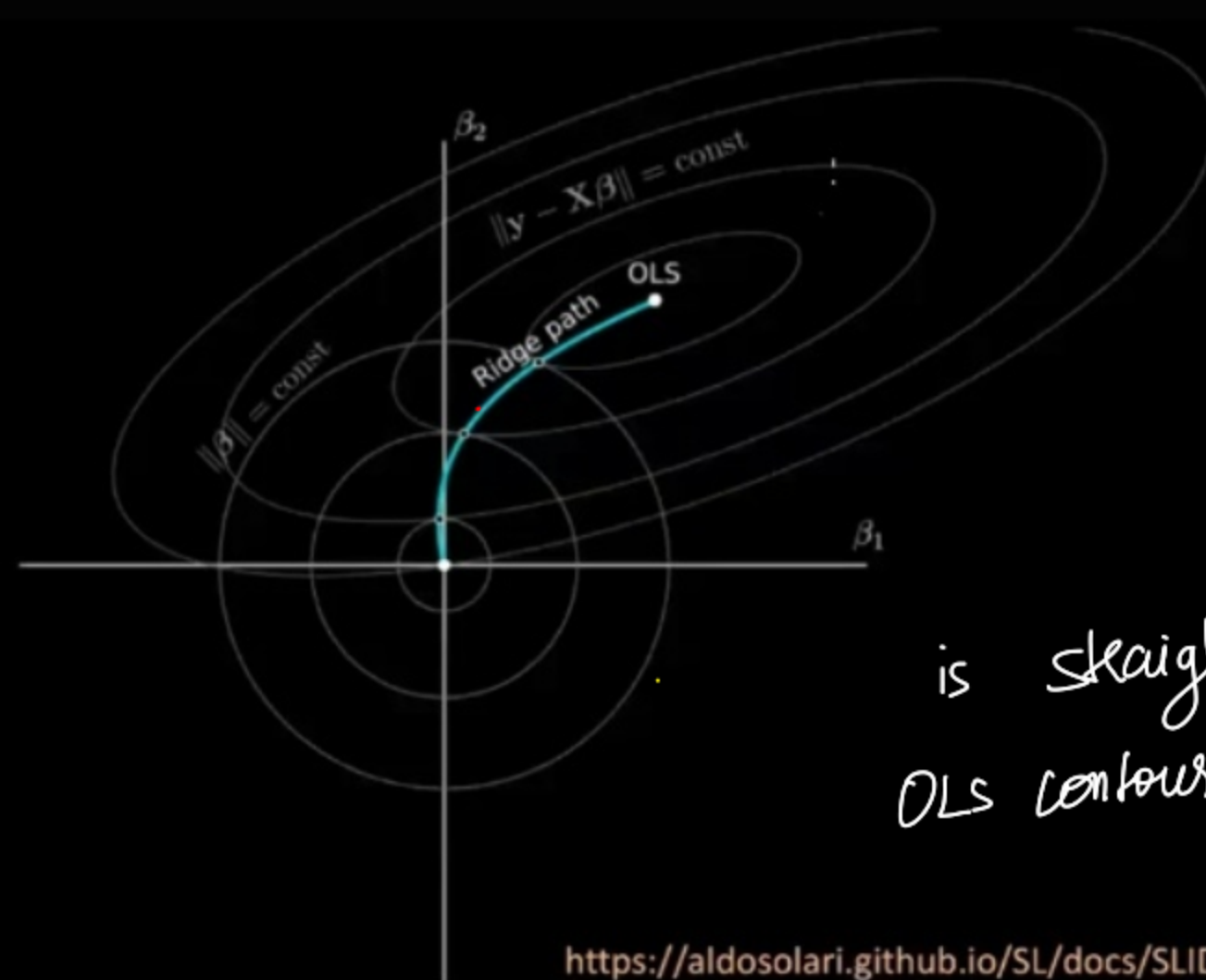
In a LASSO Regression, if the regularization parameter λ is very high, which of the following is true?

- ✓ (A) The model can shrink the coefficients of uninformative features to exactly 0
- (B) The loss function is as same as the ordinary least square loss function. ✗
- (C) The loss function is as same as the ridge regression loss function ✗
- (D) The bias of the model is no lower than the bias of the model with a smaller λ .

Comparison of LASSO with the other methods

Term	LS	Ridge	LASSO
x_0 Intercept	2.480	2.467	2.477
x_1	0.680	0.389	0.545
x_2	0.305	0.238	0.237
x_3	-0.141	-0.029	
x_4	0.210	0.159	0.098
x_5	0.305	0.217	0.165
x_6	-0.288	0.026	
x_7	-0.021	0.042	
x_8	0.267	0.123	0.059
Test Error	0.586	0.540	0.491

sparsity



Ridge path is straight line if OLS contours are circle

<https://aldosolari.github.io/SL/docs/SLIDES/F>

GRADIENT DESCENT IN LASSO & RIDGE

update rule

$$w^{t+1} = w^t - \alpha \left(\frac{\partial L}{\partial w} + \lambda \frac{\partial R}{\partial w} \right) \quad \text{for } R = \lambda w^T w \quad L2 \rightarrow \text{Ridge}$$

$$= w^t - \alpha \left(\frac{\partial L}{\partial w} + \lambda w^t \right)$$

$\alpha \rightarrow$ learning rate
 $\lambda \rightarrow$ regularisation factor

$$= w^t (1 - \alpha \lambda) - \alpha \frac{\partial L}{\partial w} \quad \left. \begin{array}{l} \text{Ridge regularisation} \\ \text{Matrix of } \pm 1 \text{ where } a_i = \text{sign}(w_i) \\ a_i = \begin{cases} +1, & w_i > 0 \\ -1, & w_i < 0 \end{cases} \end{array} \right\}$$

LASSO

$$w^{t+1} = w^t - \alpha \left(\frac{\partial L}{\partial w} + \lambda \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

Optimizing L1 regularized models The attentive reader will have noticed that the L1 penalty violates a key property that we want in optimization: it is not smooth. This is a major drawback of L1 regularization, and there are two strategies to avoid it, and we cannot naively apply gradient descent to L1-regularized models. One strategy to get around this is to apply specially designed optimization algorithms, which do not need the smoothness requirement. An alternative approach relies on the notion of *subgradient descent*, which generalizes gradient descent to convex but not-necessarily-differentiable functions. The theory of subgradients is covered in more advanced machine learning courses. However, the consequence in the case of L1 regularization is that we can use

$$\frac{\partial |w|}{\partial w} = \text{sign}(w) \quad (11.5)$$

as a stand-in for the derivative of the L1 norm (i.e., the absolute value) when performing gradient descent.

— Subgradient descent
if we denote the matrix $\begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$ $a_i = \frac{w_i}{|w_i|}$
as $\text{sign}(w)$
then

LASSO UPDATE RULE

$$w^{t+1} = w^t - \alpha \lambda \text{sign}(w^t) - \frac{\partial L(x)}{\partial w} \Big|_{w^t}$$

RIDGE UPDATE RULE

$$w^{t+1} = w^t (1 - \alpha \lambda) - \frac{\partial L(x)}{\partial w} \Big|_{w^t}$$

→ Observation: Here we can see why LASSO promotes parameter sparsity

★ L2 regularisation → w^t is being updated by decaying by some rate $\alpha \lambda$
L1 regularisation → w^t is being updated by taking away some fixed value $\alpha \lambda \text{sign}(w)$

side quest — L_p Norm

L_p norm of a vector $w \in \mathbb{R}^m$ denoted by $\|w\|_p$ for $p \in (0, \infty)$ is defined as

$$\|w\|_p = \begin{cases} \left[\sum_{i=1}^m |w_i|^p \right]^{1/p}, & p \neq 0, \\ \max_{i=1}^m (|w_i|), & p = \infty \end{cases}$$

Non zero entries in vector w , $p=0$

for $p=0$ & $p=\infty$, L_p norm is defined by taking limit

eg:-

$$\|w\|_1 = |w_1| + |w_2| + \dots + |w_m|$$

$$\|w\|_2 = \sqrt{w_1^2 + w_2^2 + w_3^2 + \dots + w_m^2}$$

Is least squares scaling invariant?

OLS

suppose Alice and Bob do the same experiment

- ▶ Alice measures distance in mm
- ▶ Bob measures distance in km

they each compute an estimator with least squares and compare their predictions

Q: Do they make the same predictions?

- A. yes ✓
- B. no

★★★★★

→ In OLS, scaling a feature x_i by a factor of say k , scales down the corresponding learned parameter w_j by the same factor k . $\Rightarrow$ Prediction invariant

→ But for ridge or lasso this is not the case because the algorithm is sensitive to weight the scaling of certain features does not make prediction invariant.

→ In ridge, the learning algorithm is biased towards reducing the larger weights

Normalisation / standardisation

→ if input data & therefore our design matrix X is not normalised & each feature column is a different range, then the weights of the true function may be biased towards features with lower range. This can cause Ridge or Lasso learning stage biased towards reducing these larger weights

→ To avoid this biasing, we use normalisation to make all features in the common range. $\left(x_i = \frac{x_i - \mu}{\sigma} \right)$

Non invertible $X^T X$ ★★★★★
in case of Ridge Regression

$$(X^T X + \lambda I) w = X^T y$$

I → In practice we omit the w_0 corresponding to w_0 .

$X^T X + \lambda I \rightarrow$ always invertible

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

$\therefore$ Regularisation $\Rightarrow$ unique solution

Logistic Regression

→ This is a method for classification

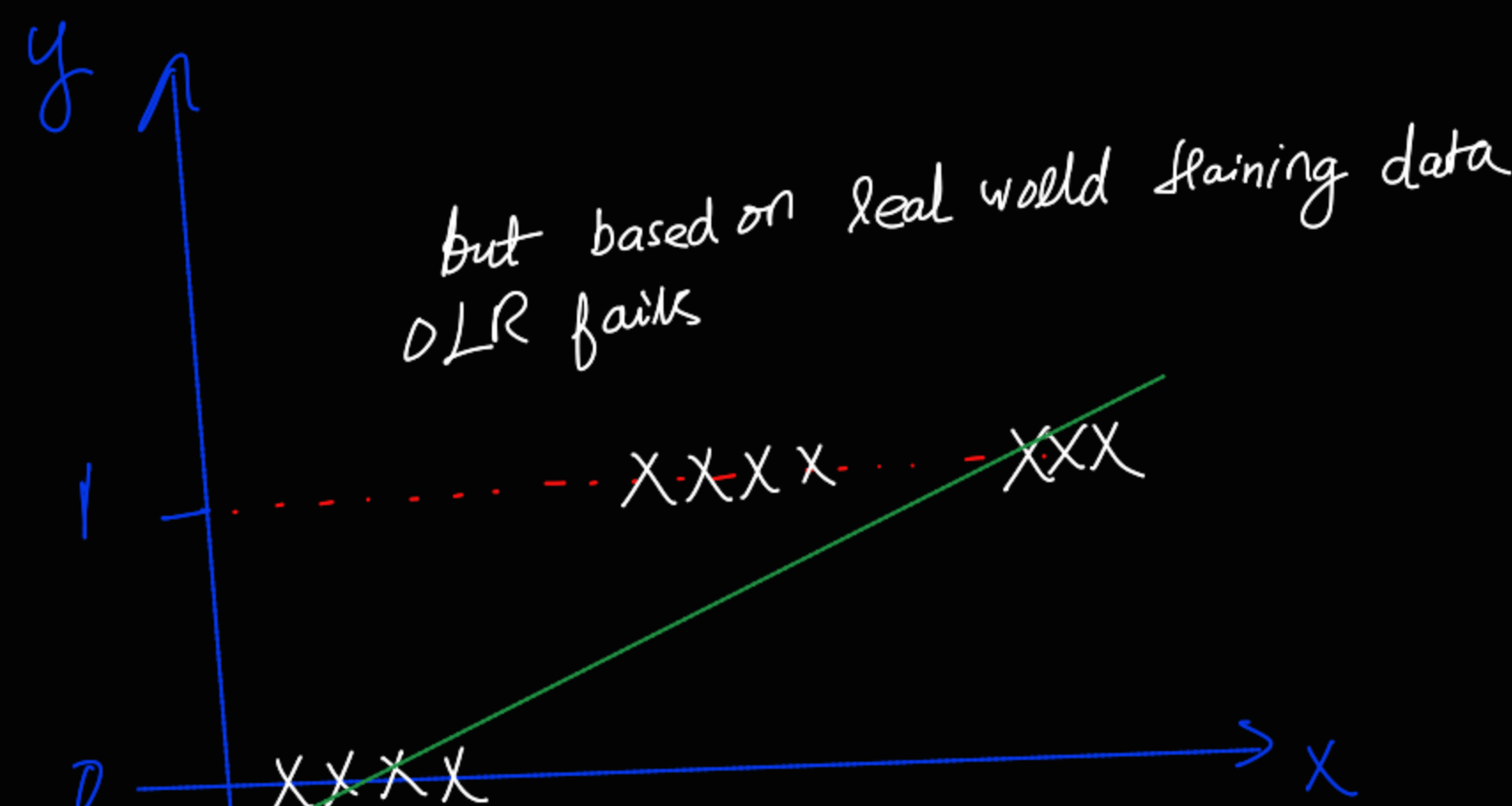
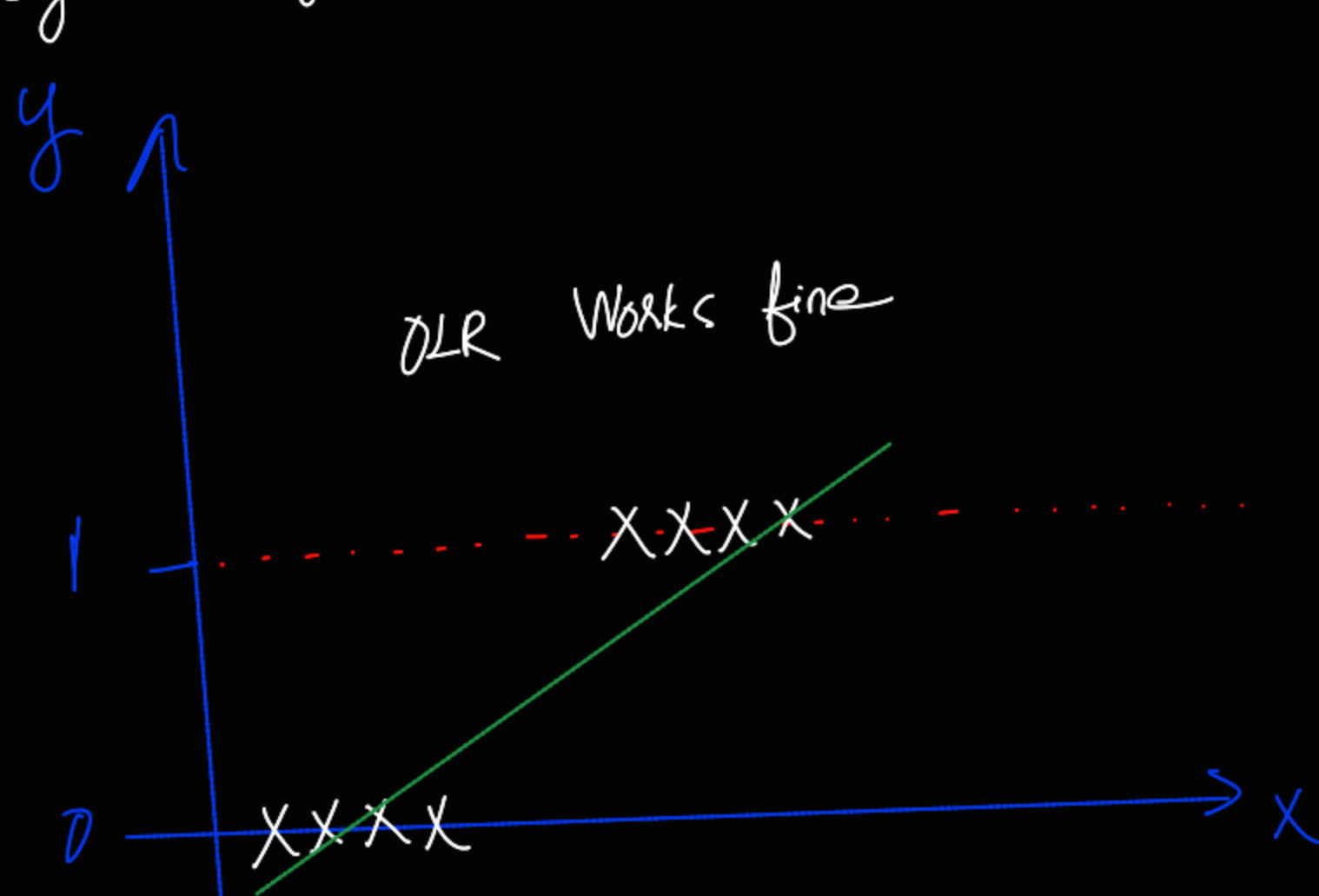
→ Why not use Linear regression for classification?

Insight:

→ Linear regression has linear relation btw outputs. Also the ordering is inherent in linear regression → This relation is not true in real world classification problem

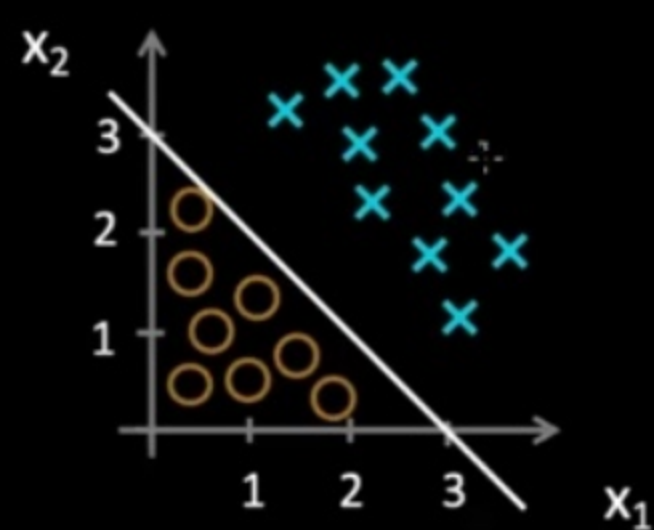
→ if the categorical response variable is ordinal (Variable is representing a category, but order is inherent among category eg: - class year, height category, weight category etc) then what?

- ① Non-sensical data for y
 - ② It will treat unit change from category 1 to category 2 the same as a change from cat 2 to cat 3
 - ③ The model would also imply specific ordering. If the same problem was encoded with different ordering then predictions will change entirely
- Linear regression model would make some sense but still not ideal



→ Goal of classifier → • Learn Decision Boundary

Decision Boundary

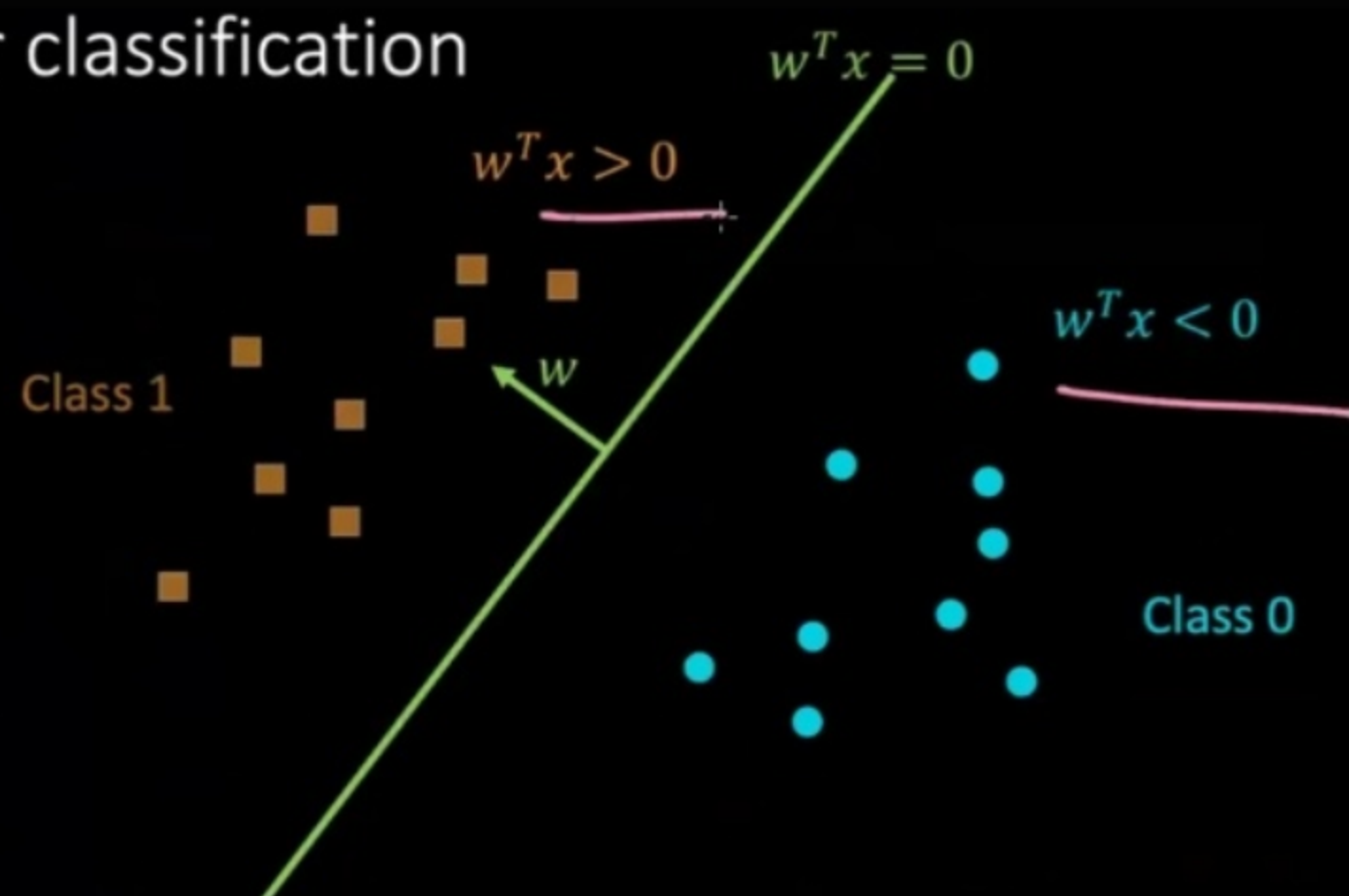


- Learn parameters of decision boundary (equation of hyperplane)
- Inside: - One side of the hyperplane, the eq gives >0 then other side gives <0 , on the boundary $=0$.

$$w_0 + w_1 x_1 + w_2 x_2$$

} Objective → learn parameters of best fit decision boundary → hyperplane in multi-dimension.

Linear classification



→ The vector normal to the hyperplane for x_0 is given by w .

Binary Classified Model

$$x_i \theta = 0 \rightarrow \text{decision boundary}$$

$$w_0 + w_1 x_{i1} + w_2 x_{i2} + \dots + w_m x_{im} = 0$$

$$\text{our model } \begin{cases} y_i = 1, & x_i \theta \geq 0 \\ y_i = 0, & \text{otherwise} \end{cases}$$

How to train? What is the loss function

$m = \# \text{ features}$

$$X_i = \begin{bmatrix} 1 & x_{i1} & x_{i2} & \dots & x_{im} \end{bmatrix}_{1 \times m} \quad \theta = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}_{m \times 1}$$

Linear classification: natural attempt

- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $f_w(x) = w^T x$ to minimize $\hat{L}(f_w) = \frac{1}{n} \sum_{i=1}^n \mathbb{I}[\text{step}(w^T x_i) \neq y_i]$

Indicator Variable

- Drawback: difficult to optimize
- NP-hard in the worst case

0-1 loss

either loss is 0 or 1 $\rightarrow$ It doesn't depend on $x_i w$

0-1 loss function

Linear classification: simple approach

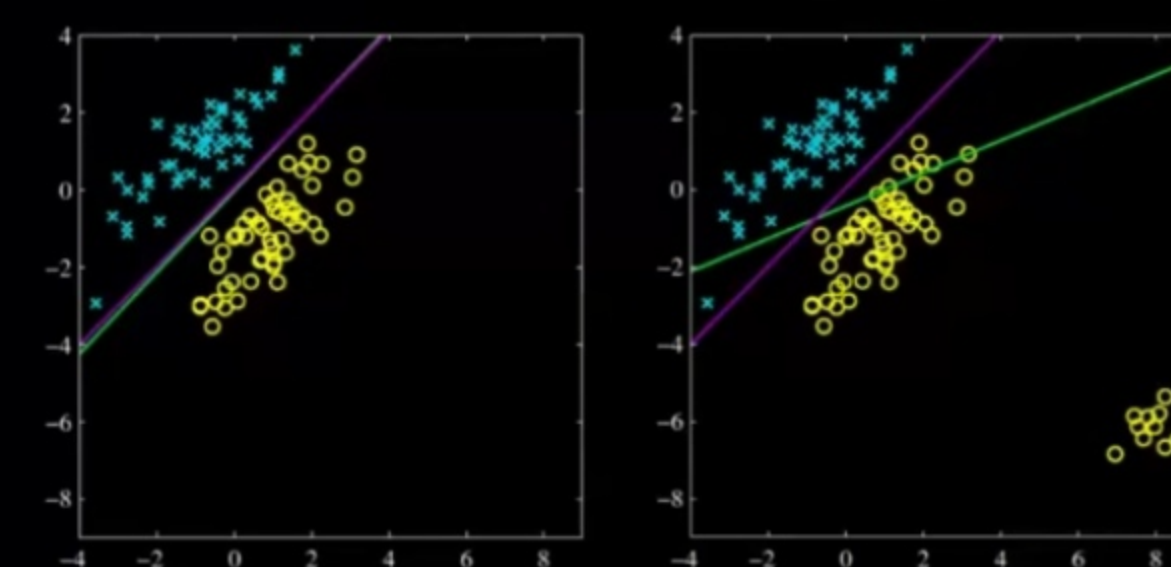
- Given training data $\{(x_i, y_i): 1 \leq i \leq n\}$ i.i.d. from distribution D
- Find $f_w(x) = w^T x$ that minimizes $\hat{L}(f_w) = \frac{1}{n} \sum_{i=1}^n (w^T x_i - y_i)^2$

Loss $\propto$ error now - OK

$\rightarrow$ But highly sensitive to outliers

Reduce to linear regression; ignore the fact $y \in \{0, 1\}$

Linear classification: simple approach



Drawback: not robust to "outliers"

Figure borrowed from Pattern Recognition and Machine Learning, Bishop

Figure 4.4 The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

$\rightarrow$ My loss function should consider predicted output in range $(0, 1)$

$\rightarrow$ It should not be highly sensitive to outliers

$\rightarrow$ The loss should be increase with errors

We want a model that can tell us: $\left. \begin{matrix} p(y=1|x; \theta) \\ p(y=0|x; \theta) \end{matrix} \right\} \text{prediction in range } (0, 1)$

The problem: z isn't a probability, it's just a number!

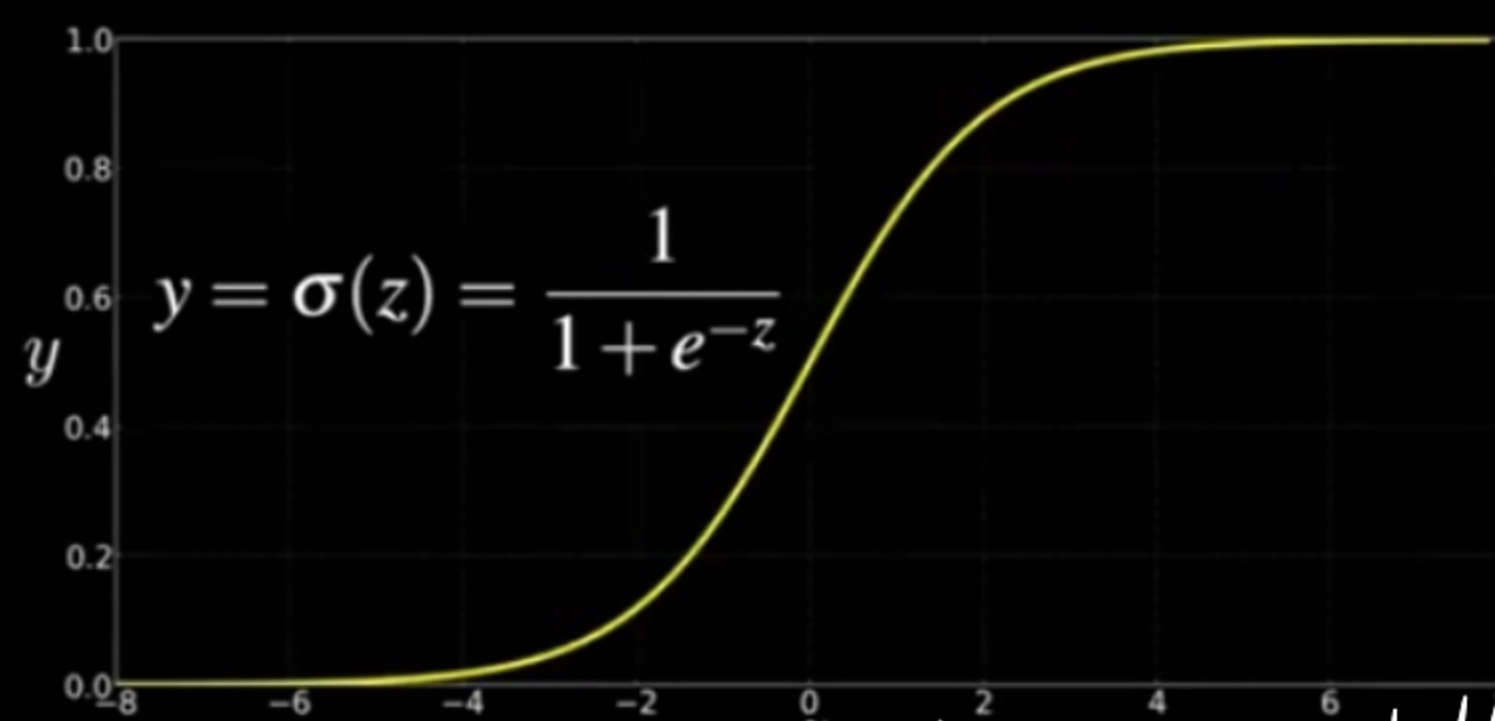
$$z = w \cdot x + b$$

Solution: use a function of z that goes from 0 to 1

$$y = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)}$$

Sigmoid function
logistic function

$$1 - \sigma(x) = \sigma(-x)$$



We will now treat $\sigma(x\theta)$ as a probability of $P(y=1|x,\theta)$ & $1 - \sigma(x\theta)$ as $P(y=0|x,\theta)$

$$\begin{aligned} P(y=1|x,\theta) &= \sigma(x\theta) \\ P(y=0|x,\theta) &= \sigma(-x\theta) \end{aligned}$$

Q) if $\theta = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}$ & $x = \begin{bmatrix} 7 & -1 & -1 \end{bmatrix}$, then what is $P(Y=1)$

$$= \sigma(7-4-3) = \sigma(0) = \underline{\underline{0.5}}$$

Note as per our notation
 $x \rightarrow$ row vector
 $\theta \rightarrow$ column vector

Since $P(Y=1) + P(Y=0) = 1$

our output $\hat{y} = \begin{cases} 1, & P(Y=1) > 0.5 \\ 0, & \text{otherwise} \end{cases}$

$\rightarrow$ This is equivalent to the model we adopted

$$\text{ie } y_i = \begin{cases} 1, & x_i \theta > 0 \\ 0, & x_i \theta \leq 0 \end{cases}$$

$$\text{Since } \frac{1}{1+e^{-x_i \theta}} > 0.5$$

$$\Rightarrow 1 > \frac{1+e^{-x_i \theta}}{2}$$

$$\Rightarrow e^{-x_i \theta} < 1$$

$$\ln \Rightarrow -x_i \theta < 0$$

$$\Rightarrow x_i \theta > 0$$

$\rightarrow$ Notice that the threshold of 0.5 here can be modified.

$\rightarrow$ Logical regression model gives only $P(Y=1)$; the categorical output if the threshold is defined later on

Q) Consider a logistic regression model with $w = (1, 2)$ and $b = 0$.

Use the model to estimate probability that y is 1, given that x is $(-3, 1)$.

Solution

$$\frac{1}{1+e^{-1}} = \frac{1}{1+e} = 0.269$$

TRUE / FALSE Logistic regression sends an example x through a linear function to get a real number, which it sends through an exponential function to get a positive number, which it sends through a logistic function to predict a probability between 0 and 1. (Then it optionally compares that probability to a threshold to predict $\hat{y} = 0$ or $\hat{y} = 1$.)

False, the logistic function includes the exponential so not need separately

7.2 (5pt) Suppose that you have trained a logistic regression classifier $h_\theta(x) = \sigma(1-x)$ where $\sigma(\cdot)$ is the logistic/sigmoid function. What does its output on a new example $x = 2$ mean? Check all that apply. (Hint: $\sigma(-1) \approx 0.27$)

☐ Your estimate for $P(y = 1|x; \theta)$ is about 0.73.

☐ Your estimate for $P(y = 0|x; \theta)$ is about 0.27.

☒ Your estimate for $P(y = 1|x; \theta)$ is about 0.27.

☒ Your estimate for $P(y = 0|x; \theta)$ is about 0.73.

$h_\theta(x) \rightarrow$ hypothesis function $\rightarrow$ model
 in LR $h_\theta(x) = P(Y=1|x;\theta)$

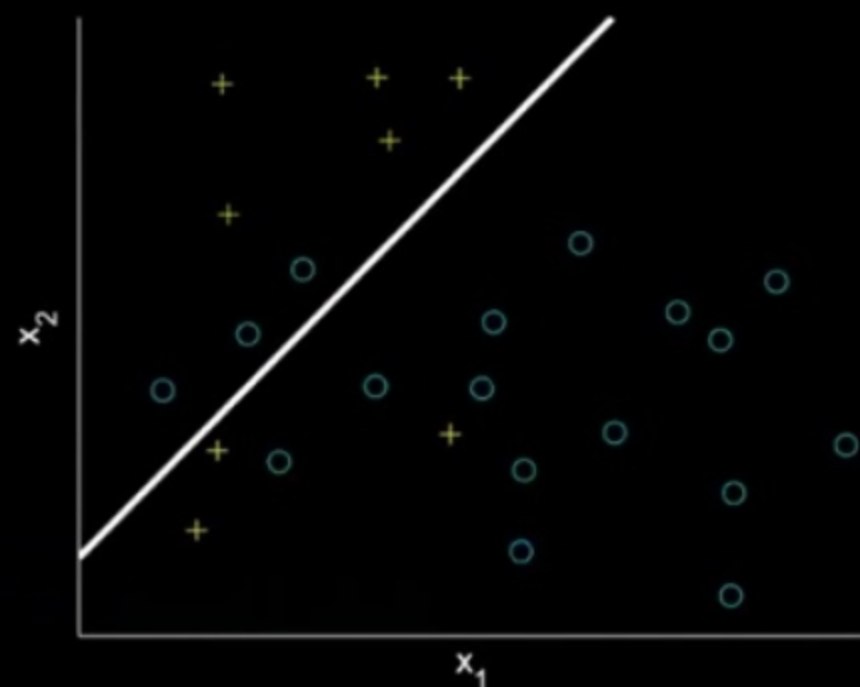
Q

Consider the dataset shown in the diagram below with a logistic regression decision boundary.

Questions:

1. Count the number of mistakes the model is making. **6**

2. Is the data linearly separable? **No**



Q Suppose we trained a logistic regression classifier for some binary classification task. The true labels y and predicted probabilities $P(Y = 1|x)$ are given below.

y	$P(Y = 1 x)$
1	0.9
1	0.6
0	0.6
0	0.55
1	0.4
0	0.3

In order to make predictions, we need to threshold $P(Y = 1|x)$. We use our thresholds as follows:

$$\hat{Y} = \begin{cases} 1 & P(Y = 1|x) \geq T \\ 0 & P(Y = 1|x) < T \end{cases}$$

We want to decide between three possible thresholds for $P(Y = 1|x)$.

- Model A uses a threshold of $T = 0.25$.
- Model B uses a threshold of $T = 0.5$.
- Model C uses a threshold of $T = 0.75$.

How many mistakes each model is making?

Consider linear regression and logistic regression. Circle the correct answer for each statement below. If a statement is false, explain why in one sentence.

1. (1 pt) (True or False) They both use linear functions. **No.**

Logistic Reg use Logistic Fn.

2. (1 pt) (True or False) They both can be used to solve regression problems. **No.**

3. (1 pt) (True or False) They both use the logistic activation function. **No**

OLR does not use logistic activation function

What is the purpose of the sigmoid function in logistic regression?

- (a) It converts continuous input into categorical data.
- (b) It standardizes the input to have zero mean and variance 1.
- (c) It optimizes the weights to reduce loss.
- ✓ (d) It transforms the output to a probability.

→ Notice here $\ln(\text{odds}) = \ln\left(\frac{P(Y=1|X, w)}{P(Y=0|X, w)}\right)$

$$= \ln(e^{xw}) = xw$$

→ So our linear part xw is the $\ln$ of odds
 → This is one interpretation of logistic regression

(2 pt) Bubble the expression that describes $P(Y=0|X)$ for a logistic regression model.

- ✓ ☒ $\sigma(-X^T\beta)$
- ☐ $1 - \log(1 + \exp(X^T\beta))$
- ☐ $1 + \log(1 + \exp(-X^T\beta))$
- ☐ None of these

★ (2 pt) Bubble the expression that describes the odds ratio $\frac{P(Y=1|X)}{P(Y=0|X)}$ of a logistic regression model. Recall: $P(Y=0|X) + P(Y=1|X) = 1$ for any X .

- ☐ $X^T\beta$
- ☐ $-X^T\beta$
- ✓ ☒ $\exp(X^T\beta)$
- ☐ $\sigma(X^T\beta)$
- ☐ None of these

$$= \frac{\sigma(X^T\beta)}{\sigma(-X^T\beta)} = \frac{1 + e^{X^T\beta}}{1 + e^{-X^T\beta}} = e^{X^T\beta} \frac{(e^{-X^T\beta} + 1)}{(1 + e^{-X^T\beta})}$$

$$= e^{X^T\beta}$$

LOG ODDS = xw (Linear)

(4 pt) Select *always*, *sometimes*, or *never* to describe when each statement below is true about a logistic regression model $P(Y=1|X) = \sigma(X^T\beta)$, where Y is binary and X is vector-valued. X and β are finite.

- ☐ Always ☒ Sometimes ☐ Never: $P(Y=1|X) > X^T\beta$
- ✓ ☒ Always ☐ Sometimes ☐ Never: $P(Y=1|X) = P(Y=0|-X)$
- ✓ ☒ Always ☐ Sometimes ☐ Never: $P(Y=1|X) < 1$
- ☐ Always ☒ Sometimes ☐ Never: $\sigma(X^T\beta) \leq \sigma(X^T(2 \cdot \beta))$

not true for $X^T\beta < 0$

$X^T\beta$ can be -ve. $p(y=1)$ cannot
 $\sigma(X\beta) = 1 - \sigma(-X\beta) = \sigma(X\beta)$
 → Assuming we are also negating bias — yes

accuracy → $\frac{\text{correctly classified}}{\text{total data points}}$

Recap

- Given: N labeled training examples $\{x_n, y_n\}_{n=1}^N$, $x_n \in \mathbb{R}^D$, $y_n \in \{0, 1\}$
- X : $N \times D$ feature matrix, y : $N \times 1$ label vector
- $y_n = 1$: positive example, $y_n = 0$: negative example
- Goal: Learn a classifier that predicts the binary label y_* for a new input x_*
- Want a **probabilistic model** to be able to also predict the **label probabilities**

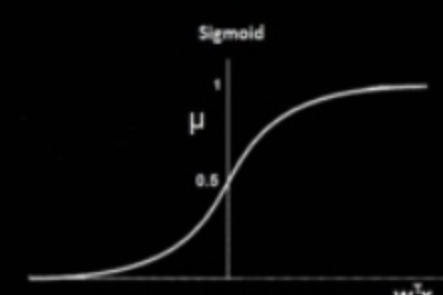
$$p(y_n = 1|x_n, w) = \mu_n \quad \sigma(w^T x_n) = \mu_n$$

$$p(y_n = 0|x_n, w) = 1 - \mu_n$$

- $\mu_n \in (0, 1)$ is the probability of y_n being 1
- Note: Features x_n assumed fixed (given). Only labels y_n being modeled
- w is the model parameter (to be learned)
- How do we define μ_n (want it to be a function of w and input x_n)?

- Logistic regression defines μ using the **sigmoid function**

$$\mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)}$$



- Sigmoid computes a real-valued "score" ($w^T x$) for input x and "squashes" it between $(0, 1)$ to turn this score into a **probability** (of x 's label being 1)

- Thus we have

$$p(y=1|x, w) = \mu = \sigma(w^T x) = \frac{1}{1 + \exp(-w^T x)} = \frac{\exp(w^T x)}{1 + \exp(w^T x)}$$

$$p(y=0|x, w) = 1 - \mu = 1 - \sigma(w^T x) = \frac{1}{1 + \exp(w^T x)}$$

Remember the factorization

- Note:** If we assume $y \in \{-1, +1\}$ instead of $y \in \{0, 1\}$ then $p(y|x, w) = \frac{1}{1 + \exp(-yw^T x)}$

- What's the underlying decision rule in Logistic Regression?
- At the decision boundary, both classes are equiprobable. Thus:

$$p(y=1|x, w) = p(y=0|x, w)$$

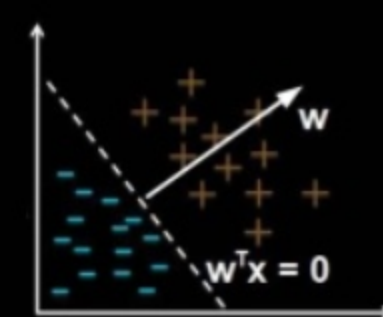
$$\frac{\exp(w^T x)}{1 + \exp(w^T x)} = \frac{1}{1 + \exp(w^T x)}$$

$$\exp(w^T x) = 1$$

$$w^T x = 0$$

- Thus the decision boundary of LR is nothing but a **linear hyperplane**, just like Perceptron, Support Vector Machine (SVM), etc.

- Therefore $y = 1$ if $w^T x \geq 0$, otherwise $y = 0$



Insight → xw is like the distance from our learned hyperplane ($xw=0$) if $|w|=1$ → Then it is the **actually distance** [normalised weight]

→ if the xw is already considerable, then further increase in distance gives incrementally less & less confidence in probability.

→ if xw is close to the decision boundary then $\sigma(xw)$ changes rapidly.

★ Instead of $\in [0, 1]$ if we consider categorical variable $y \in \{-1, 1\}$

then $P(y=y|x, \theta) = \sigma(yx\theta) = \frac{1}{1 + e^{-yx\theta}}$

→ if $P(Y_n=1|x_n, \theta) = p_n$
 $P(Y_n=0|x_n, \theta) = 1-p_n$ } Categorical variable $y_n \in \{0, 1\}$ } Then $P(Y_n=y_n|x_n, w) = p_n^{y_n} (1-p_n)^{1-y_n}$
COMPACT FORM

LOSS FUNCTION

MSE $\frac{1}{2n} \sum_{i=1}^n (y_i - \sigma(x_i w))^2$

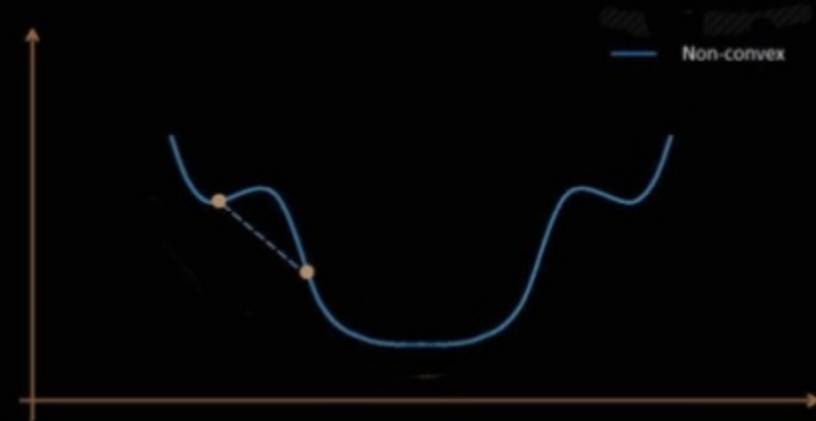
$\downarrow$ real y $\downarrow$ predicted probability of y=1

→ The cost/loss is proportional to the error in predicting probability
 → Outliers are not affecting the loss significantly.

Using the logistic regression model

$h_{\theta}(x) = \frac{1}{1 + e^{-\theta^T x}}$

results in a **non-convex** optimization



→ Optimisation problem is non-convex

This is one idea

Case 1: real $y=1 \rightarrow \sigma(x_i w)$ is close to 1
 → The loss/cost is less
 → desirable

Case 2: real $y=1 \rightarrow \sigma(x_i w)$ is low/close to 0
 → The cost is high & proportion to error in probability.
 → desirable.

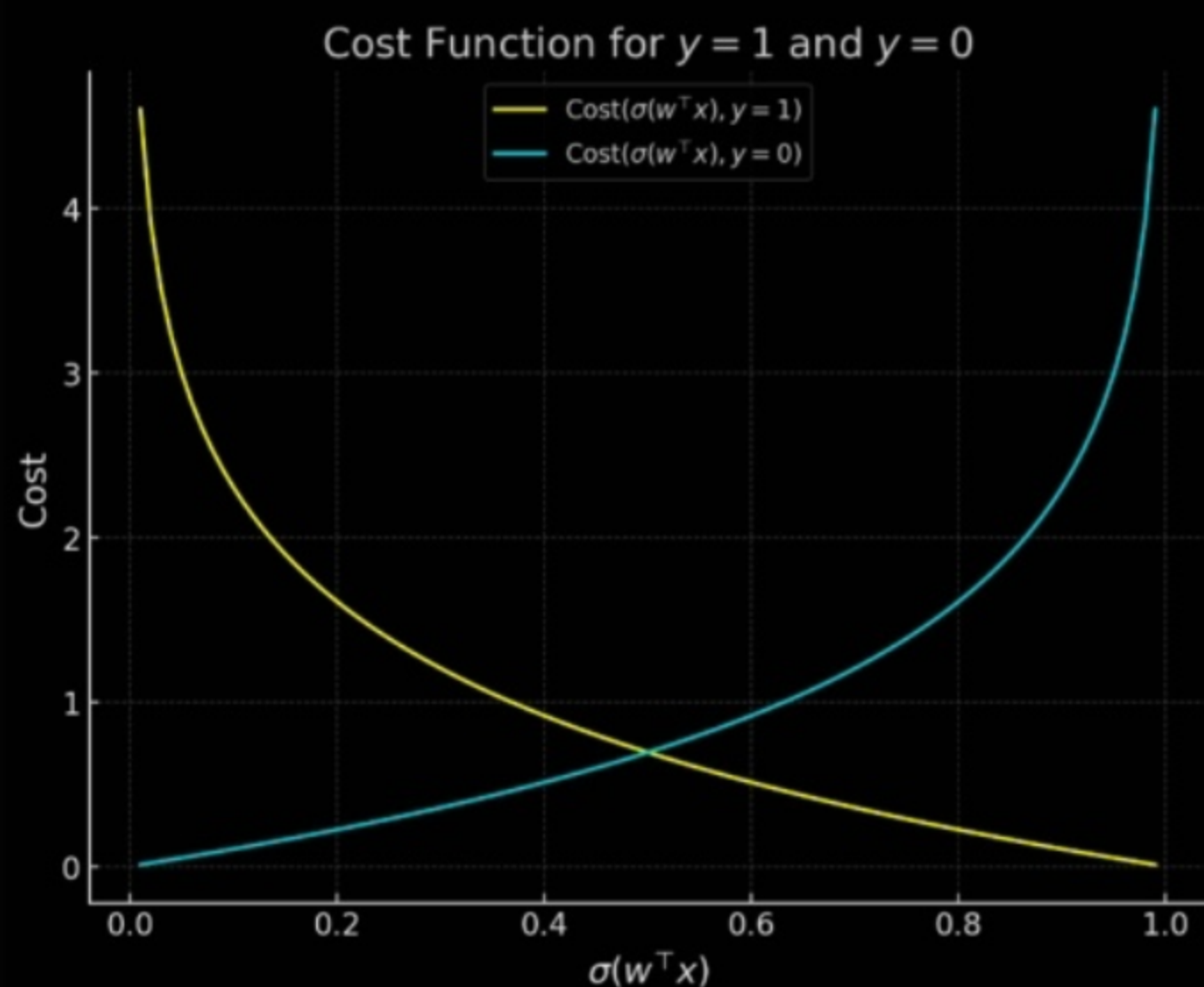
Case 3: real $y=0 \rightarrow \sigma(x_i w)$ is low/close to 0
 → The cost is low
 → desirable

Case 4: real $y=0 \rightarrow \sigma(x_i w)$ is high/close to 1
 → The cost is high & proportion to error
 → desirable.

CROSS ENTROPY LOSS

$\sigma(w^T x)$	Actual y	need Loss
0.7	1	less loss ??
0.2	0	less loss ??
0.1	1	high loss ??
0.4	1	high loss ??

→ what should be the loss here? Here we used random w to plot $\sigma(w^T x)$
 → This implies that our loss, should be a function of $\sigma(xw)$ & actual y
 ie. Cost/loss = $f(\sigma(xw), y) = \begin{cases} ?? & y=1 \\ ?? & y=0 \end{cases}$



→ We are separating out the loss for $y=0$ & $y=1$

$$L(\sigma(xw), y) = \begin{cases} -\ln(\sigma(w^T x)) & \text{if } y=1 \\ -\ln(1 - \sigma(w^T x)) & \text{if } y=0 \end{cases}$$

$\sigma(w^T x)$	Actual y	need Loss	Mean square loss	Our New loss
0.7	1	less loss ??	$(0.3)^2 = 0.09$	$-\ln(0.7) = 0.35667$
0.2	0	less loss ??	$(0.2)^2 = 0.04$	$-\ln(0.8) = 0.22314$
0.1	1	high loss ??	$(0.8)^2 = 0.64$	$-\ln(0.1) = 2.3025$
0.4	1	high loss ??	$(0.6)^2 = 0.36$	$-\ln(0.4) = 0.9162$

Loss function in compact form

$$\text{Cost}(\sigma(xw), y_n) = - \left(y_n \cdot \ln p_n + (1 - y_n) \ln (1 - p_n) \right)$$

CROSS ENTROPY LOSS (LOGISTIC LOSS)

Advantages

- Smooth
- Convex function
- Easy to derivate

[3 pts] Which of the following statements about **logistic regression** are correct?

- ☒ The cost function of logistic regression is convex
- ☒ The cost function of logistic regression is concave
- ☒ Logistic regression uses the squared error as the loss function
- ☒ Logistic regression assumes that each class points are generated from a Gaussian distribution

i. [2 Pts] Suppose our true labels are $\vec{y} = [0, 0, 1]$, our predicted probabilities of being in class 1 are $[0.1, 0.6, 0.9]$, and our threshold is $T = 0.5$. Give the total (not average) cross-entropy loss. Do not simplify your answer.

Total CE Loss = $\ln\left(\frac{1000}{324}\right) = 1.1270$

ii. [2 Pts] For the same values as above, give the total squared loss. Do not simplify your answer.

Squared Loss = 0.38

True y	$\sigma(xw)$	Logistic Loss	S Loss
0	0.1	$-\ln(0.7)$	0.1^2
0	0.6	$-\ln(0.4)$	0.6^2
1	0.9	$-\ln(0.9)$	0.1^2

iii. [2 Pts] Which of the following are valid reasons why we minimized the average cross-entropy loss rather than the average squared loss?

- ☐ To prevent our parameters from going to infinity for linearly separable data.
- ☐ There is no closed form solution for the average squared loss.
- ☒ To improve the chance that gradient descent converges to a good set of parameters.
- ☒ The cross entropy loss gives a higher penalty to very wrong probabilities.
- ☐ None of the above

→ Both CE Loss & MSE doesn't have closed form for Logistic regression

(1 pt) In this question, assume that we are using the logistic regression model $\hat{y} = \sigma(x^T \theta)$.

Suppose we want to modify cross-entropy loss to penalize predictions for observations that are truly positive twice as much as we penalize predictions for observations that are truly negative. Which of the following loss functions could we use? Recall that the average cross-entropy loss is:

$$R(\theta) = -\frac{1}{n} \sum_{i=1}^n (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

- ☐ $R(\theta) = -\frac{2}{n} \sum_{i=1}^n (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$
- ☒ $R(\theta) = -\frac{1}{n} \sum_{i=1}^n (2y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$
- ☐ $R(\theta) = -\frac{1}{n} \sum_{i=1}^n (y_i \log(\hat{y}_i) + 2(1 - y_i) \log(1 - \hat{y}_i))$
- ☐ $R(\theta) = -\frac{1}{n} \sum_{i=1}^n ((y_i + 2) \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$

$y = 1$

Suppose after training our model we get $\vec{\beta} = [-1.2 \quad -0.005 \quad 2.5]^T$, where -1.2 is an intercept term, -0.005 is the parameter corresponding to passenger's age, and 2.5 is the parameter corresponding to sex.

- i. [3 Pts] Consider Sīlānah Iskandar Nāsīf Abī Dāghir Yazbak, a 20 year old female. What chance did she have to survive the sinking of the Titanic according to our model? Give your answer as a probability in terms of σ . If there is not enough information, write "not enough information".

$P(Y = 1 | \text{age} = 20, \text{female} = 1) =$

- ii. [3 Pts] Sīlānah Iskandar Nāsīf Abī Dāghir Yazbak actually survived. What is the cross-entropy loss for our prediction in part i? If there is not enough information, write "not enough information."

cross entropy loss =

a) $X\beta = \begin{bmatrix} 1 & 20 & 1 \end{bmatrix} \begin{bmatrix} -1.2 \\ -0.005 \\ 2.5 \end{bmatrix} = -1.2 + -0.1 + 2.5 = 1.2$

$\therefore P(Y=1 | X\beta) = \sigma(1.2) = 0.768$

b) $CEL = -\ln(0.768) = 0.2637$

H.W.

Cross Entropy Loss

$$\text{Cost}(\sigma(X_n w), y_n) = - \left[\underbrace{y_n}_{\text{real prob}} \cdot \underbrace{\ln p_n}_{\text{predicted probability}} + (1 - y_n) \log(1 - p_n) \right]$$

For a binary classification problem, where the true class label $C \in \{0, 1\}$, the cross-entropy loss is defined based on the predicted probability P_C that a data point belongs to class C .

Which of the following expressions correctly represents the cross-entropy loss for this data point?

A) $-P_C \cdot \log(P_C) - (1 - P_C) \cdot \log(1 - P_C)$ ✗

B) $-(1 - P_C) \cdot \log(P_C) - P_C \cdot \log(1 - P_C)$ ✗

☒ C) $-\log(P_C)$

D) $-\log(1 - P_C)$ ✗

E) None of the above

→ Cross entropy loss is always one term.

Consider the sigmoid function defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\frac{d(\sigma(z))}{dz} = \frac{1 + e^{-z} \cdot 0 - 1(-e^{-z})}{(1 + e^{-z})^2} = \frac{+e^{-z}}{e^{-z}(e^z + 1)} = \frac{1}{(1 + e^z)}$$

Which of the following options correctly represents the derivative of the sigmoid function with respect to z ?

Options:

A. $\frac{d\sigma}{dz}(z) = \sigma(z)(1 + \sigma(z))$

B. $\frac{d\sigma}{dz}(z) = \sigma(z)(1 - \sigma(z))$ ✓

C. $\frac{d\sigma}{dz}(z) = (1 - \sigma(z))^2$

D. $\frac{d\sigma}{dz}(z) = \sigma(z)^2(1 - \sigma(z))$

$$\Rightarrow \frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z))$$

- The gradient of $L(\mathbf{w})$ is:

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^m \frac{\partial}{\partial \mathbf{w}} - (y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i)))$$

Our notation

$$L(\mathbf{w}) = -\sum_{i=1}^N y_i \ln(\sigma(\mathbf{x}_i \mathbf{w})) + (1 - y_i) \ln(1 - \sigma(\mathbf{x}_i \mathbf{w}))$$

- We will differentiate each term separately.

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{i=1}^N y_i \frac{\sigma(\mathbf{x}_i \mathbf{w})(1 - \sigma(\mathbf{x}_i \mathbf{w}))}{\sigma(\mathbf{x}_i \mathbf{w})} \cdot \mathbf{x}_i + (1 - y_i) \frac{\sigma(\mathbf{x}_i \mathbf{w})(1 - \sigma(\mathbf{x}_i \mathbf{w}))}{1 - \sigma(\mathbf{x}_i \mathbf{w})} \cdot \mathbf{x}_i$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{i=1}^N y_i (1 - \sigma(\mathbf{x}_i \mathbf{w})) \mathbf{x}_i - (1 - y_i) \sigma(\mathbf{x}_i \mathbf{w}) \mathbf{x}_i = -\sum_{i=1}^N [y_i - y_i \sigma(\mathbf{x}_i \mathbf{w})] \mathbf{x}_i - [\sigma(\mathbf{x}_i \mathbf{w}) - y_i \sigma(\mathbf{x}_i \mathbf{w})] \mathbf{x}_i$$

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = -\sum_{i=1}^N [y_i - \sigma(\mathbf{x}_i \mathbf{w})] \mathbf{x}_i$$

$N = \# \text{ datapoints}$
 $\nabla L_{\mathbf{w}} = \text{gradient}$

$$\nabla L_{\mathbf{w}} = \sum_{i=1}^N [\mu_i - y_i] \mathbf{x}_i$$

$\mu_i = \sigma(\mathbf{x}_i \mathbf{w})$
 $\mu_i = P(y_i = 1 / \mathbf{x}_i, \mathbf{w})$

→ Intuition ① large error on datapoint $\mathbf{x}_n \Rightarrow \mu_i - y_i \rightarrow \text{large}$
 ⇒ Large contribution of $\mathbf{x}_n$ to gradient.

partial derivative

$$\frac{\partial L(\mathbf{w})}{\partial w_j} = \sum_{i=1}^N [\mu_i - y_i] x_{ij}$$

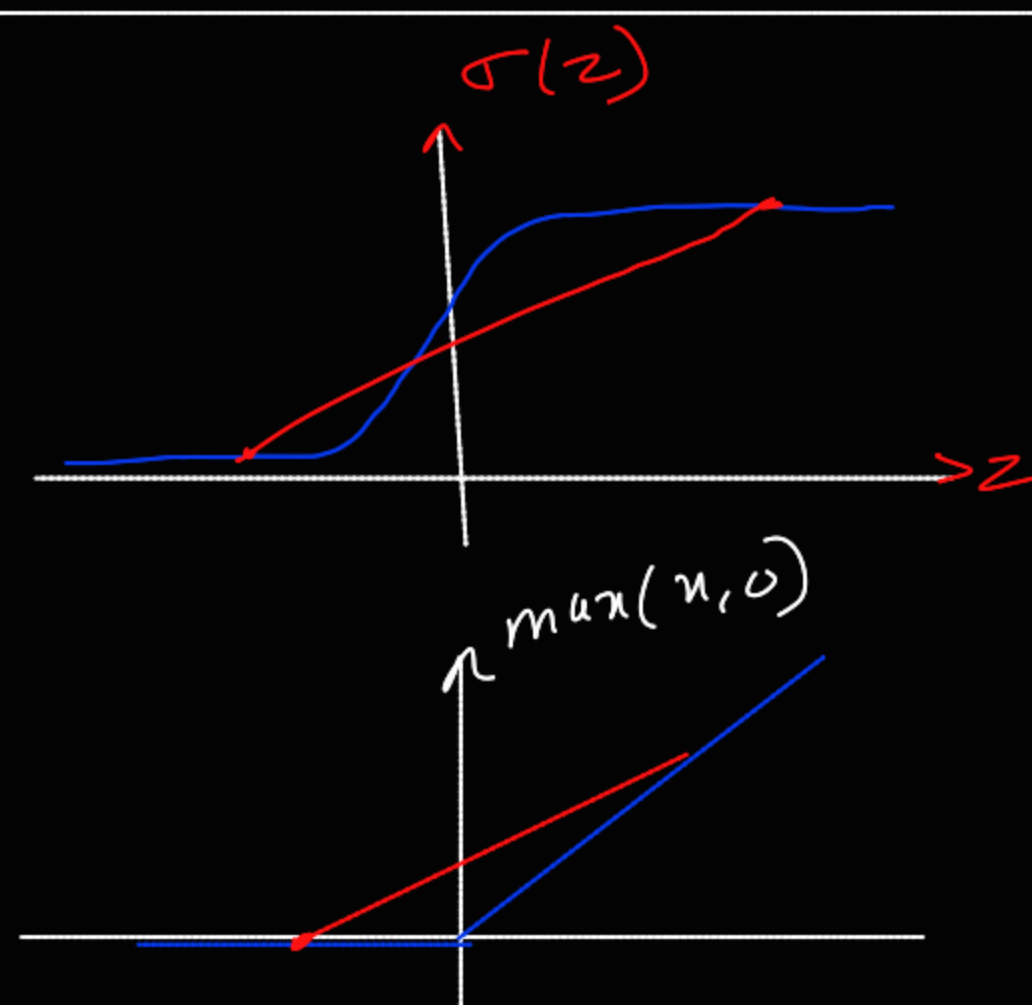
→ ② Closed form solution does not exist
 → ③ We need gradient descent

Which function is not a convex function?

- (A) Sigmoid/Logistic function: $f(x) = 1/(1 + e^{-x})$. → Not convex
- (B) Linear function: $f(x) = 3x$. → $\frac{\partial^2 f(x)}{\partial^2 x} = 0 \rightarrow \text{margin}$
- (C) Square function: $f(x) = x^2$. → Convex
- (D) ReLU $f(x) = \max\{x, 0\}$. → Convex

E) cross entropy loss
 → Convex

→ Convex marginal



(2 points) Suppose a classifier predicts each possible class with equal probability.

If there are 10 classes, what will the cross-entropy error be on a single example?

- ☐ $-\log(10)$
- ☐ $-0.1 \log(1)$
- ☒ $-\log(0.1)$
- ☐ $-10 \log(0.1)$

Solution: C. Cross-Entropy loss simplifies to negative log of the predicted probability for the correct class. At the start of training, we have approximately uniform probabilities for the 10 classes, or 0.1 for each class. So, that means $-\log(0.1)$ is what the loss should approximately be at the start.

Which of the following statements are **TRUE** about cross entropy?

- ☒ The value of cross-entropy loss is always non-negative.
- ☒ Minimizing the cross-entropy is equivalent to maximizing the likelihood with respect to the model parameters.

Maximum Likelihood Estimation (MLE)

PROBABILITY VS STATISTICS



Ber(p = 0.5) → Probability given model, predict data → P(THHTHH)



Ber(p = ???) ← Statistics given data, predict model ← THHTHH

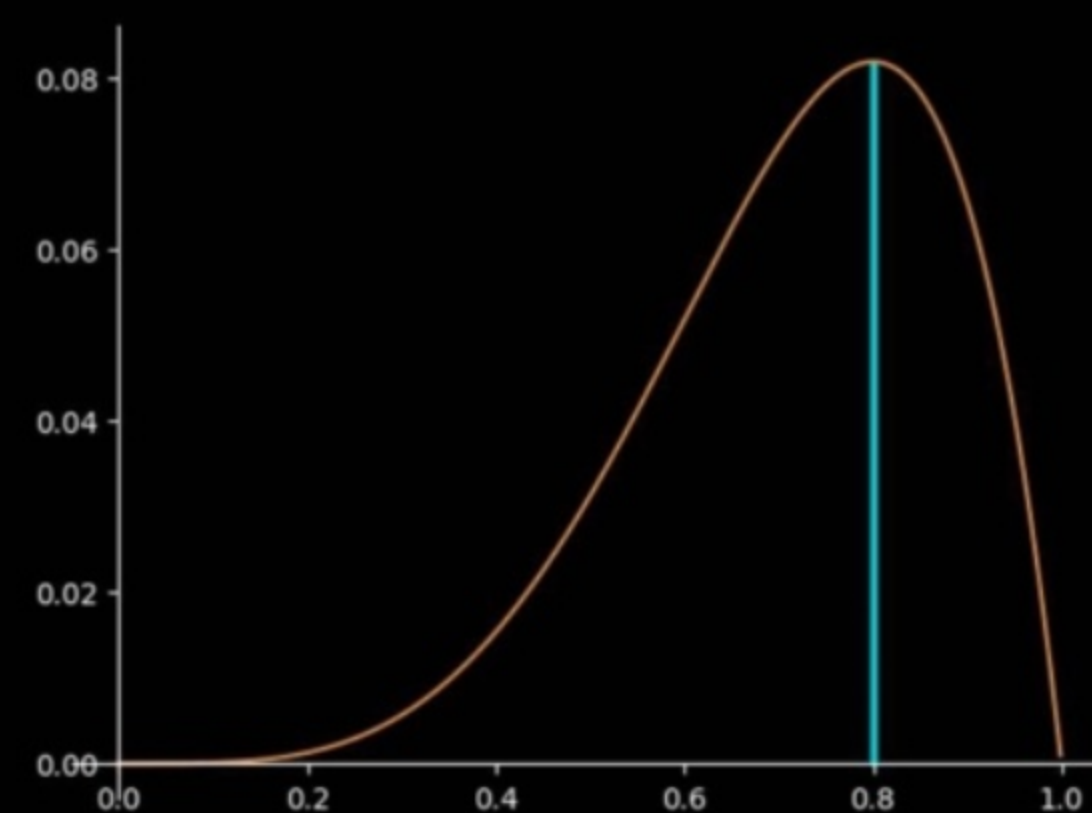
$P(\text{4 heads} | \text{5 tosses} | \theta) = 5C_4 \theta^4 (1-\theta) \rightarrow \text{likelihood of observation} = L(\text{observation} | \theta)$

Maximise the likelihood $\Rightarrow \frac{\partial L(\text{observation}(\theta))}{\partial \theta} \Rightarrow 5C_4 [4\theta^3(1-\theta) - 1(\theta^4)] = 0$

$\Rightarrow 4\theta^3 - 4\theta^4 - \theta^4 = 0 \Rightarrow 4\theta^3 = 5\theta^4 \Rightarrow \theta = 4/5$

Intuition

→ Maximise the likelihood w.r.t parameter of the distribution it came from.



Likelihood of Different Observations

Definition. The **likelihood** of independent observations $x_1, \dots, x_n$ is

$$\mathcal{L}(x_1, \dots, x_n | \theta) = \prod_{i=1}^n \mathbb{P}(x_i; \theta)$$

eg. Coin flipped $n_1 + n_2$ times observed n_1 head & n_2 tail

$\Rightarrow \text{Likelihood} = L(\text{observation} | p) = n_1 + n_2 C_{n_1} p^{n_1} (1-p)^{n_2} =$

MLE $\Rightarrow \frac{\partial}{\partial p} (n_1 \log p + n_2 \log(1-p)) = 0$

$\Rightarrow \frac{n_1}{p} - \frac{n_2}{(1-p)} = 0 \Rightarrow \frac{n_1(1-p) - n_2 p}{n_1 - n_1 p - n_2 p} = 0 \Rightarrow p = \frac{n_1}{n_1 + n_2}$

Can be used for always +ve function like Loss, Cost, probability.

★ LOG LIKELIHOOD

→ Taking log does not harm algman or alg men
→ Fn should be +ve always

Example 1: Suppose that X is a discrete random variable with the following probability mass function: where $0 \leq \theta \leq 1$ is a parameter. The following 10 independent observations

X	0	1	2	3
$P(X)$	$2\theta/3$	$\theta/3$	$2(1-\theta)/3$	$(1-\theta)/3$

were taken from such a distribution: (3, 0, 2, 1, 3, 2, 1, 0, 2, 1). What is the maximum likelihood estimate of θ .

$$P(\text{observation} | \theta) = \left(\frac{1-\theta}{3}\right)^2 + \left(\frac{2\theta}{3}\right)^2 + \left(\frac{\theta}{3}\right)^3 + \left(\frac{2(1-\theta)}{3}\right)^3$$

$$\text{MLE} \Rightarrow \frac{\partial}{\partial \theta} \left[2 \ln\left(\frac{1-\theta}{3}\right) + 2 \ln\left(\frac{2\theta}{3}\right) + 3 \ln\left(\frac{\theta}{3}\right) + 3 \ln\left(\frac{2(1-\theta)}{3}\right) \right] = 0$$

$$\Rightarrow 2 \cdot \frac{3}{1-\theta} \cdot (-1) + 2 \cdot \frac{3}{2\theta} \cdot (2) + 3 \cdot \frac{3}{\theta} \cdot (1) + 3 \cdot \frac{3}{2(1-\theta)} \cdot (-2) = 0 \Rightarrow \frac{-6}{1-\theta} + \frac{6}{\theta} + \frac{9}{\theta} - \frac{9}{1-\theta} = 0$$

$$\Rightarrow -15\theta + 15(1-\theta) = 0 \Rightarrow \theta = 15/30 = \underline{\underline{1/2}}$$

- Consider a sequence of N coin tosses (call head = 0, tail = 1)
- Each outcome x_n is a binary random variable $\in \{0, 1\}$
- Assume θ to be probability of a head (parameter we wish to estimate)
- Each likelihood term $p(x_n | \theta)$ is Bernoulli: $p(x_n | \theta) = \theta^{x_n}(1-\theta)^{1-x_n}$
- Log-likelihood: $\sum_{n=1}^N \log p(x_n | \theta) = \sum_{n=1}^N x_n \log \theta + (1-x_n) \log(1-\theta)$
- Taking derivative of the log-likelihood w.r.t. θ , and setting it to zero gives

$$\hat{\theta}_{MLE} = \frac{\sum_{n=1}^N x_n}{N}$$

- $\hat{\theta}_{MLE}$ in this example is simply the fraction of heads!
- MLE doesn't have a way to express our prior belief about θ . Can be problematic especially when the number of observations is very small (e.g., suppose we only observed heads in a small number of coin-tosses).

→ MLE doesn't consider priors
→ MAP works over MLE incorporating priors

OLR $\xrightarrow{\text{equivalent}}$ MLE

Regularised LR $\xrightarrow{\text{equivalent}}$ MAP with priors being that our weights are less

Consider the following probability distribution:

$$P_\theta(x) = 2\theta x e^{-\theta x^2}$$

where θ is a parameter and x is a positive real number. Suppose you get m i.i.d. samples x_i drawn from this distribution. The maximum likelihood estimator (MLE) for θ based on these samples is given by:

$$\theta = ?$$

Which of the following is the correct MLE for θ ?

A) $\theta = \frac{\sum_{i=1}^m x_i^2}{m}$

C) $\theta = \sqrt{\frac{\sum_{i=1}^m x_i^2}{m}}$

✓ B) $\theta = \frac{m}{\sum_{i=1}^m x_i^2}$

D) $\theta = \frac{\sum_{i=1}^m x_i}{m}$

$$P(\text{observation} | \theta) = \prod_{i=1}^m (2\theta x_i e^{-\theta x_i^2})$$

$$\text{MLE} \xrightarrow{\log \text{MLE}} \frac{\partial}{\partial \theta} \left(\sum_{i=1}^m \ln(2\theta x_i e^{-\theta x_i^2}) \right) = 0 = \frac{\partial}{\partial \theta} \sum_{i=1}^m \left[\ln(2\theta x_i) - \theta x_i^2 \right] = 0$$

$$\Rightarrow \sum_{i=1}^m \frac{1}{2\theta x_i} - x_i^2 \Rightarrow \frac{m}{\theta} - \sum_{i=1}^m x_i^2 \Rightarrow \theta = \frac{m}{\sum_{i=1}^m x_i^2}$$

There are 5 balls in a bag. Each ball is either red or blue. Let θ (an integer) be the number of blue balls. We want to estimate θ , so we draw 4 balls **with replacement** out of the bag, replacing each one before drawing the next. We get "blue," "red," "blue," and "blue" (in that order).

- (a) [5 pts] Assuming θ is fixed, what is the likelihood of getting exactly that sequence of colors (expressed as a function of θ)?
- (b) [3 pts] Draw a table showing (as a fraction) the likelihood of getting exactly that sequence of colors, for every value of θ from zero to 5 inclusive.

θ	$\mathcal{L}(\theta; \langle \text{blue, red, blue, blue} \rangle)$
0	?
1	?
2	?
3	?
4	?
5	?

$$p(\text{blue ball}/\theta) = \frac{\theta}{5}$$

$$p(\text{red ball}/\theta) = \frac{5-\theta}{5}$$

$$a) p(B, R, B, B/\theta) = \left(\frac{\theta}{5}\right)^3 \cdot \left(\frac{5-\theta}{5}\right)$$

$$M \log LE \rightarrow \frac{\partial}{\partial \theta} \left[3 \ln\left(\frac{\theta}{5}\right) + \ln\left(\frac{5-\theta}{5}\right) \right] = 0$$

$$\Rightarrow 3 \cdot \frac{5}{\theta} + \frac{(-1)5}{5-\theta} = 0$$

$$\Rightarrow 15(5-\theta) - 5\theta = 0$$

$$\Rightarrow 20\theta = 75$$

$$\theta = \frac{75}{20} = \underline{\underline{3.75}}$$

[3 pts] What is the maximum likelihood estimate for θ ? (Chosen among all integers; not among all real numbers.)

MLE in Logistic Regression

$$P(y_n | \theta, x_n) = p_n^{y_n} (1-p_n)^{1-y_n}, \text{ where } p_n = P(y_n=1/x_n, \theta) = \sigma(x_n \theta)$$

Likelihood over all observation x_i for i from 1 to N

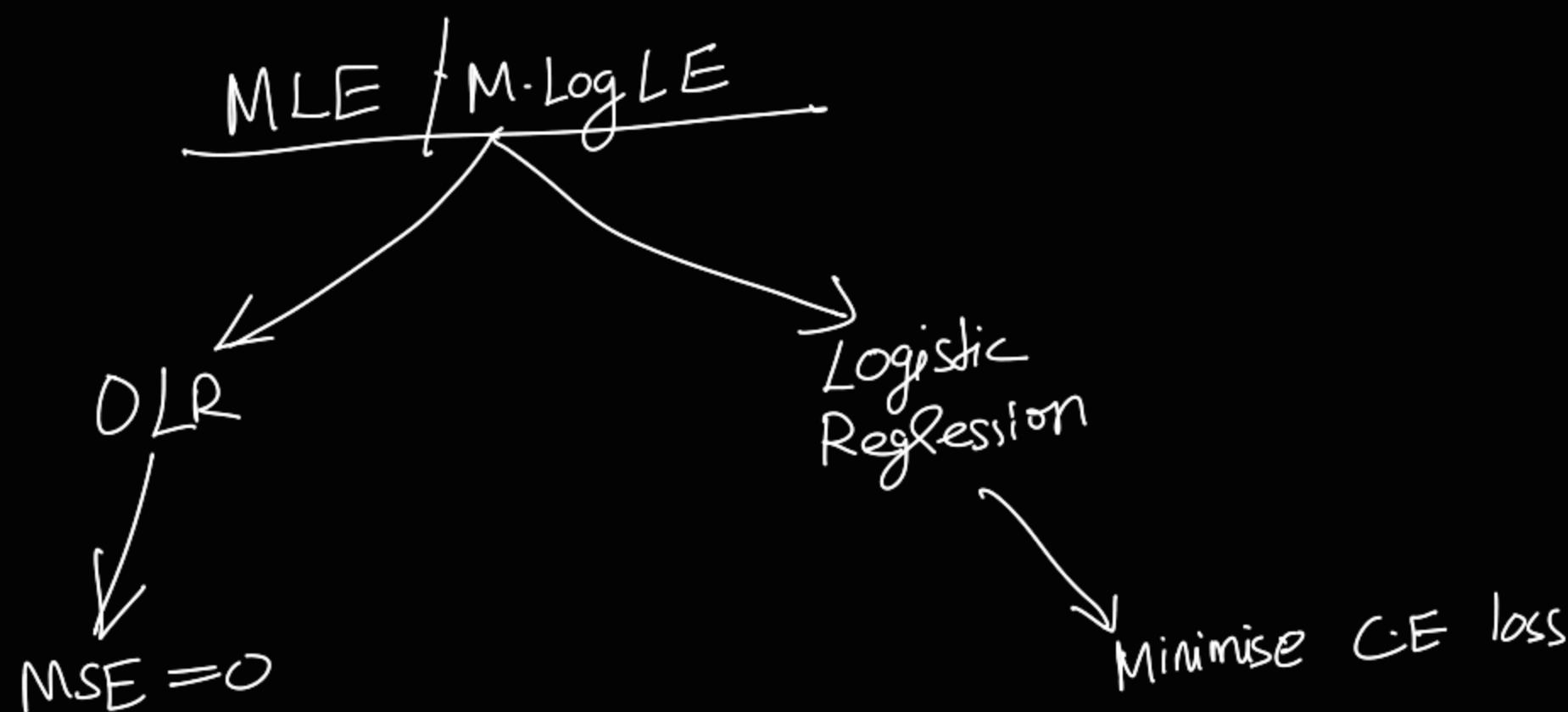
$$\text{Likelihood} = \prod_{i=1}^N \sigma(x_i \theta)^{y_i} (1-\sigma(x_i \theta))^{1-y_i} \xrightarrow{M.L.H} \sum_{i=1}^N y_i \ln(\sigma(x_i \theta)) + (1-y_i) \ln(1-\sigma(x_i \theta))$$

This is same as minimising the (-ve)

$\Rightarrow$ Minimise

$$-\sum_{i=1}^N y_i \ln(\sigma(x_i \theta)) + (1-y_i) \ln(1-\sigma(x_i \theta)) \xrightarrow{\frac{\partial}{\partial \theta}=0} \sum_{i=1}^N (\sigma(x_i \theta) - y_i) x_i = 0$$

Insight $\rightarrow$ Cross Entropy loss is -ve of log likelihood



1. (3 points) Which of the following models can be learned by MLE?

- ☐ Perceptron ☐ Ridge regression ☐ SVMs ☒ Logistic regression ☒ Ordinary Linear Regression

Solution:

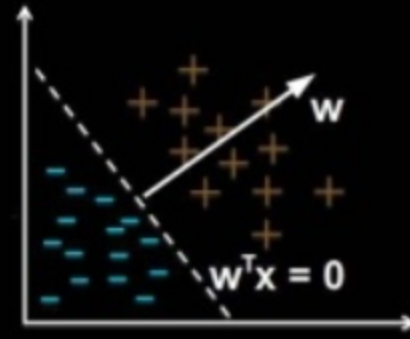
MAP

Perceptron & SVMs are not probabilistic models

- What's the underlying decision rule in Logistic Regression?
- At the decision boundary, both classes are equiprobable. Thus:

$$\begin{aligned} p(y=1|x, w) &= p(y=0|x, w) \\ \frac{\exp(w^T x)}{1 + \exp(w^T x)} &= \frac{1}{1 + \exp(w^T x)} \\ \exp(w^T x) &= 1 \\ w^T x &= 0 \end{aligned}$$

- Thus the decision boundary of LR is nothing but a **linear hyperplane**, just like Perceptron, Support Vector Machine (SVM), etc.
- Therefore $y = 1$ if $w^T x \geq 0$, otherwise $y = 0$

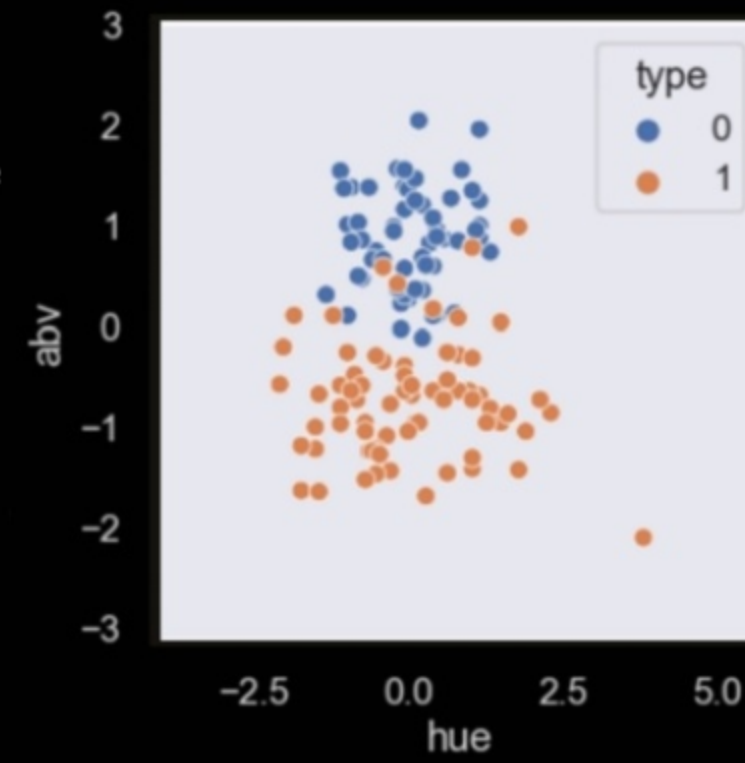


φ

Consider the following scatter plot of our two (standardized) features.

Which of the following statements are true about an unregularized logistic regression model fit on the above data? Select all that apply.

- ☐ After performing logistic regression, the weight for the hue feature will very likely have a negative sign.
- ☒ After performing logistic regression, the weight for the abv feature will very likely have a negative sign.



★ If no regularization is used and the training data is linearly separable, the optimal model parameters will tend towards positive or negative infinity.

- ☒ True ☐ False

$$P(y=1) = \frac{1}{1 + e^{-kxw}}$$

$$P(y=0) = \frac{1}{1 + e^{kxw}}$$

Since linearly separable → all points are correctly classified
→ Scaling up reduces loss

Gradient descent

$$\sum_{i=1}^N (p_i - y_i) x_i$$

→ Gradient descent will keep on increasing the mag w until $p_i(\sigma(xw))$ is as close to y_i as possible.

→ Now, what happens if not linearly separable

if Linearly Separable

⇒ $Xw \rightarrow$ perfect boundary

$$w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 \dots w_n x_n = 0$$

→ We are not changing the hyperplane but multiply all weights by k.

$$kw_0 + kw_1 x_1 + \dots + kw_n x_n = 0$$

→ intercepts are still same

→ Distance from hyperplane gets increased by factor k.

Which of the following statements are true about logistic regression?

☒ A: If the points are linearly separable (with a positive margin), then logistic regression (with suitable optimization software) can find a decision boundary with zero training error.

☒ B: If we don't use regularization, the logistic regression cost function might have infinitely many minimizers.

★ Let's think about what happens with linearly separable data in logistic regression.

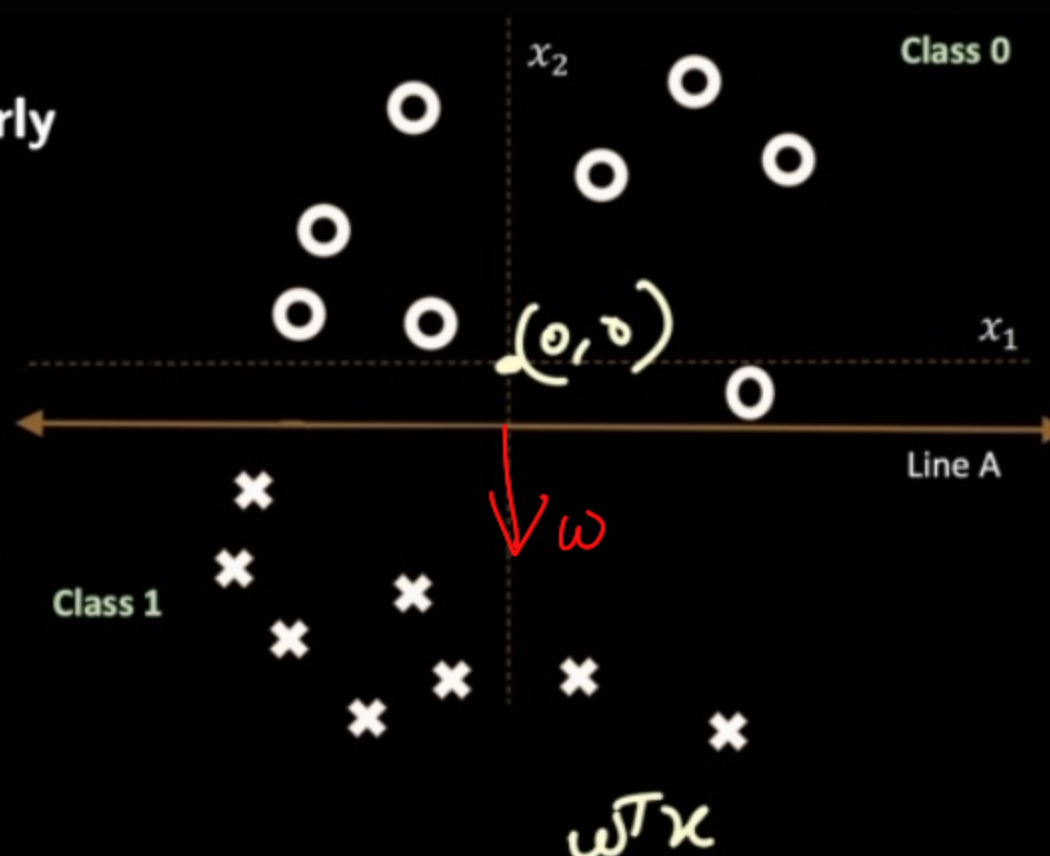
Consider the following training set:

Let the equation of horizontal line A be $w_0 + w^T x$, where intercept $w_0 = -1$.

Question: What would be the direction of w ?

Our model is logistic regression:

$$\Pr(y=1|x, w) = \frac{1}{1 + \exp(-w_0 - w_1 x[1] - w_2 x[2])}$$



Direction of w is perpendicular to hyperplane.

→ Toward class 1

$Xw \rightarrow$ dot product of X direction & w direction

We want Xw +ve → class 1
 Xw -ve → class 0

⇒ w is in dir of class 1

Slight change of previous Question:

→ Here it will be towards class 1

Let's think about what happens with linearly separable data in logistic regression.

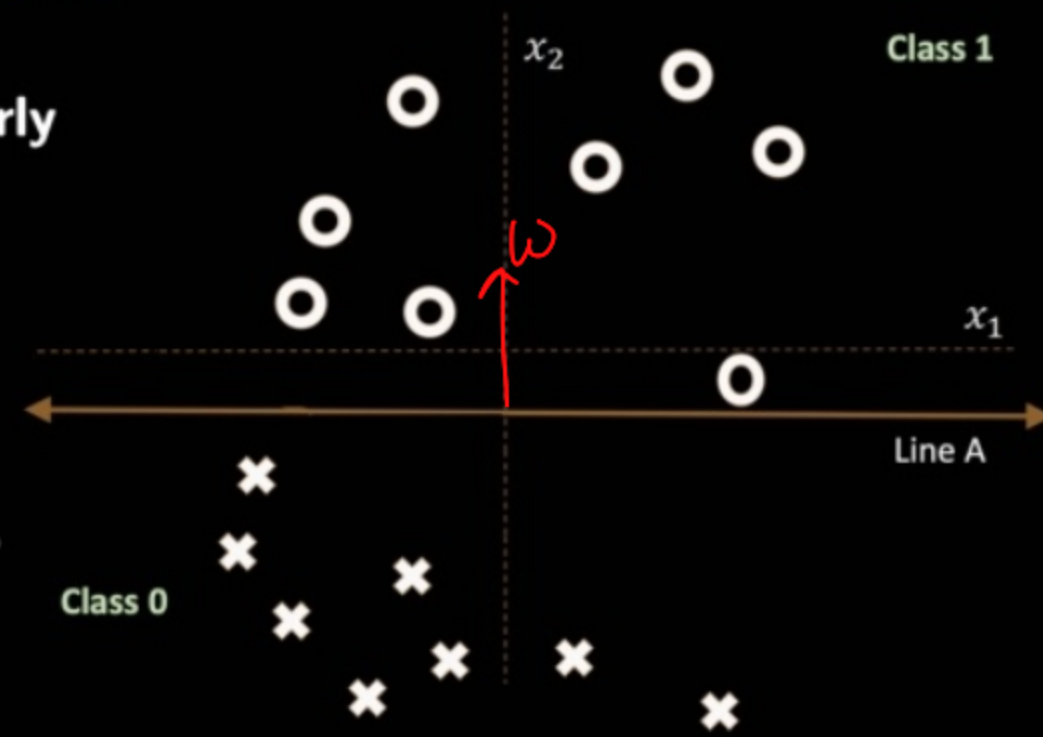
Consider the following training set:

Let the equation of horizontal line A be $w_0 + w^T x$, where intercept $w_0 = -1$.

Question: What would be the direction of w ?

Our model is logistic regression:

$$\Pr(y = 1 | x, w) = \frac{1}{1 + \exp(-w_0 - w_1 x[1] - w_2 x[2])}$$



→ Insight: w is towards class 1 direction.

Let's think about what happens with linearly separable data in logistic regression.

Consider the following training set on the right:

Each data point has two features - one shown on the horizontal axis, and the second on the vertical axis. Each point is in one of two classes - 'X' is class 1, 'O' is class 0.

Our model is logistic regression:

$$\Pr(y = 1 | x, w) = \frac{1}{1 + \exp(-w_0 - w_1 x[1] - w_2 x[2])}$$



(a) What are some possible values of w to produce line A and line B? Line A is horizontal with an intercept of -1. Line B has a slope of $-1/2$ and an intercept of -2 on the vertical axis.

$$\text{Line A } \begin{bmatrix} 1 & x_1 & x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\text{Line B } ax_1 + bx_2 + c = 0$$

$$\text{slope} = \frac{b}{a} = -1/2$$

$$x_1 \text{ intercept} = -\frac{c}{a}$$

$$x_2 \text{ intercept} = -\frac{c}{b} = -2$$

$$\begin{matrix} b = -1 \\ a = 2 \\ c = -2 \end{matrix} \quad \text{one solution}$$

$$\Rightarrow 2x_1 - x_2 + 2 = 0$$

$$\phi \quad \text{if } P(y|x, \theta) = \begin{cases} \sigma(x\theta) & y = +1 \\ 1 - \sigma(x\theta) & y = -1 \end{cases} \quad \text{then compact notation } P(Y=y|x, \theta) = \frac{1}{1 + e^{-y_n x_n \theta}}$$

★

$$\text{probabilistic model } P(Y=y|x, \theta) = \sigma(y_n x_n \theta)$$

$$\text{Loss fn conventional notation} = -\sum_{i=1}^N [y_n \ln(\sigma(x_n \theta)) + (1 - y_n) \ln(1 - \sigma(x_n \theta))]$$

$$\text{if } y \in \{+1, -1\} \text{ then Loss } L(w|x, \theta) = -\sum_{i=1}^N \ln(\sigma(y_n x_n \theta)) = \sum_{i=1}^N \ln(1 + e^{-y_n x_n \theta})$$

①

Given the probability model for a binary classification problem where the label $y_n \in \{+1, -1\}$ and the probability is defined as:

$$P(y_n | x_n) = \begin{cases} \sigma(w^T x_n) & \text{for } y_n = +1 \\ 1 - \sigma(w^T x_n) & \text{for } y_n = -1 \end{cases}$$

$$\begin{aligned} & -\sum_{i=1}^N \ln \sigma(y_n x_n \theta) \\ & = \sum_{i=1}^N \ln(1 + e^{-y_n x_n \theta}) \end{aligned} \quad \left| \begin{array}{l} -\ln x = \ln \frac{1}{x} \end{array} \right.$$

Which of the following represents the loss function for this classification problem using the logistic loss formulation?

A. $\sum_{n=1}^N (-y_n \log(\sigma(w^T x_n)) - (1 - y_n) \log(1 - \sigma(w^T x_n)))$

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

✓ C. $\sum_{n=1}^N \log(1 + \exp(-y_n w^T x_n))$

D. $\sum_{n=1}^N \log(1 + \exp(y_n w^T x_n))$

Using M-Log Likelihood

$$P(y|x, \theta) = \prod_{i=1}^N \sigma(y_n w_n x_n) = \sum_{i=1}^N \ln \sigma(y_n w_n x_n)$$

$$\text{-ve MLE is CE loss} \Rightarrow \text{CE} = -\sum_{i=1}^N \ln \sigma(y_n x_n w) = \sum_{i=1}^N \ln(1 + e^{-y_n x_n w})$$

MULTICLASS LOGISTIC REGRESSION

Laka SOFTMAX, multinoulli/Multinomial

→ Learn k weight vectors / Learn k hyperplane to classify into k classes.

→ Model: Predicted class → highest score of $x\theta_k$

$$= \underset{k}{\operatorname{argmax}} (x\theta_k)$$

Intuition → distance from hyperplane

$$k = \# \text{ classes} \\ = \# \text{ hyperplan}$$

We trained a three-way logistic regression and obtained weights

$$w_a = (1, 1, 0), w_b = (-1, 1, 1), w_c = (2, 1, 2).$$

What label would be given to the point $x = (0, 1, 1)$?

(A) A.

(B) B.

✓ (C) C.

$$x w_a^T = 1$$

$$x w_b^T = 2$$

$$x w_c^T = 3$$

$$\therefore C$$

ALGEBRAIC NOTATION

$$W = \begin{bmatrix} w^1 & w^2 & \dots & w^d \end{bmatrix}$$

$$XW = \begin{bmatrix} xw^1 & xw^2 & \dots & xw^d \end{bmatrix}$$

Class 1 score Class 2 score Class k score

$$d = \# \text{ classes}$$



A linear predictor is used to solve a classification problem with three classes. The data is two-dimensional and the linear functions for each class are:

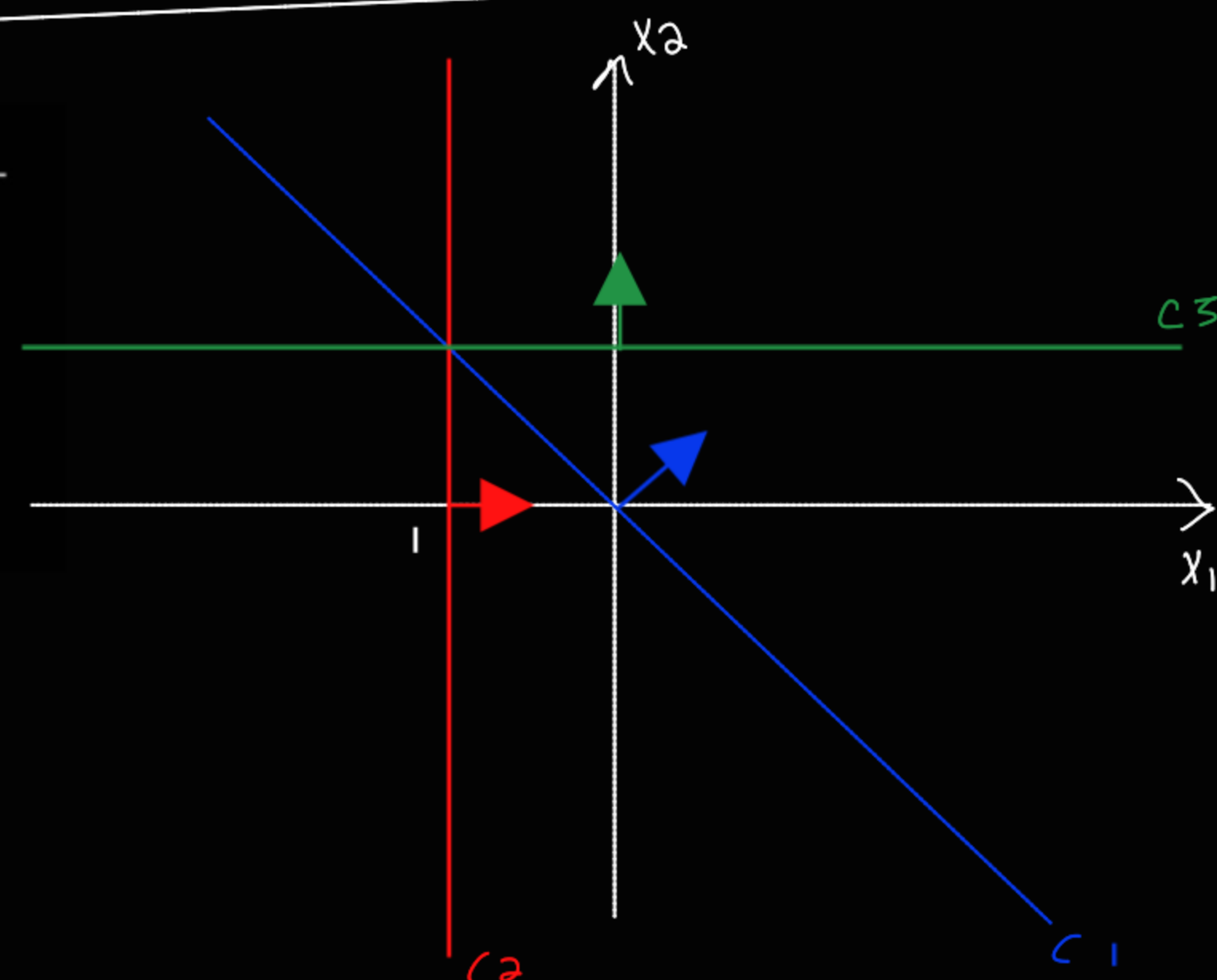
- Class 1: $w_1 = (1, 1), b_1 = 0$
- Class 2: $w_2 = (1, 0), b_2 = 1$
- Class 3: $w_3 = (0, 1), b_3 = -1$

Draw the resulting decision boundary and clearly mark the region corresponding to each class.

$$C1 \Rightarrow x_1 + x_2 = 0$$

$$C2 \Rightarrow x_1 + 1 = 0$$

$$C3 \Rightarrow x_2 - 1 = 0$$



Now this is the hyperplane only / Decision boundary $\neq$ hyperplane in softmax regression.

$$\text{class 1 score} > \text{class 2 score}$$

$$\Rightarrow x_1 + x_2 > x_1 + 1$$

$$\Rightarrow x_2 - 1 > 0$$

$$\underline{x_2 > 1}$$

$$\text{class 1 score} > \text{class 3 score}$$

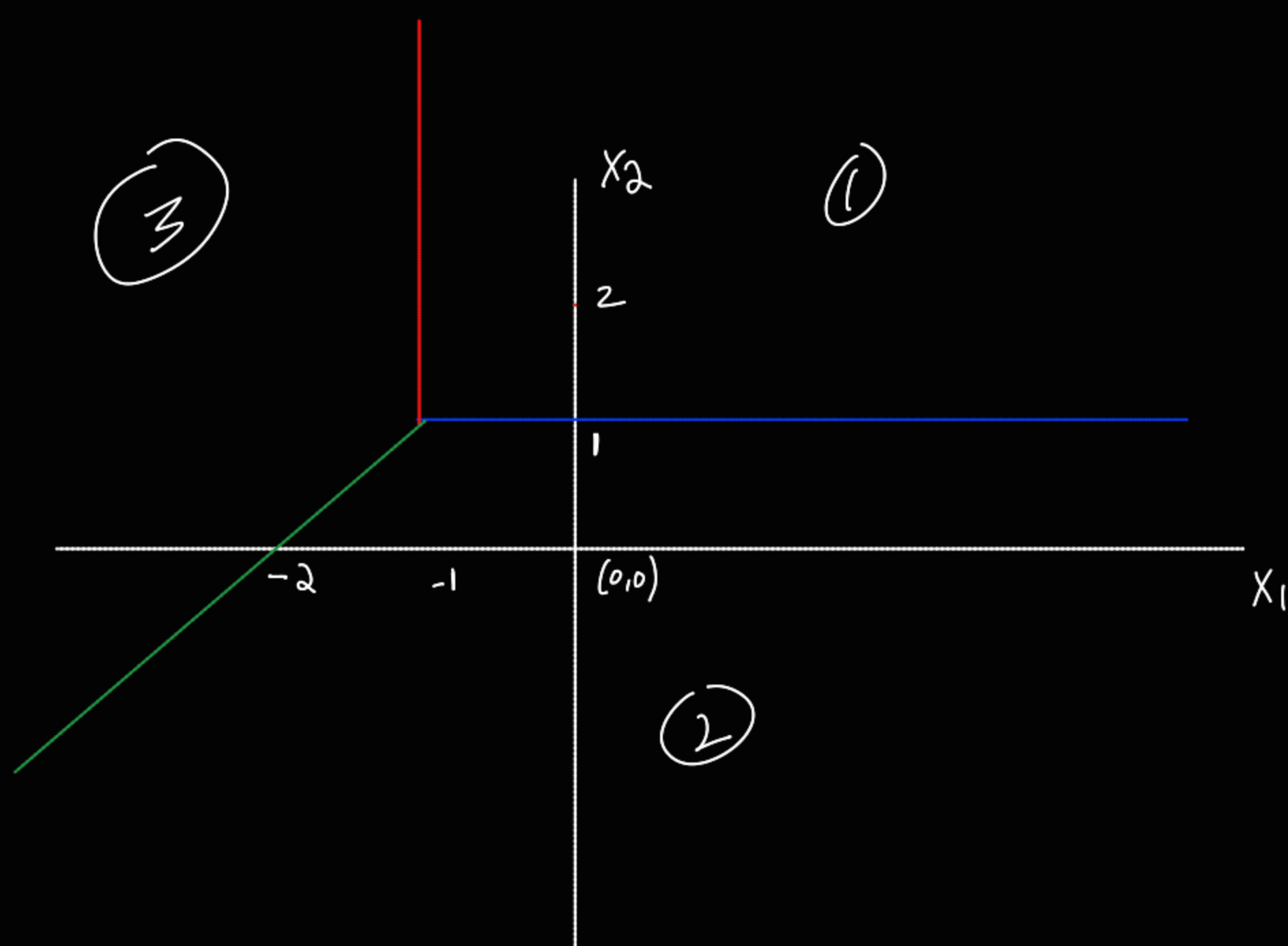
$$\Rightarrow x_1 + x_2 > x_2 - 1$$

$$\Rightarrow \underline{x_1 > -1}$$

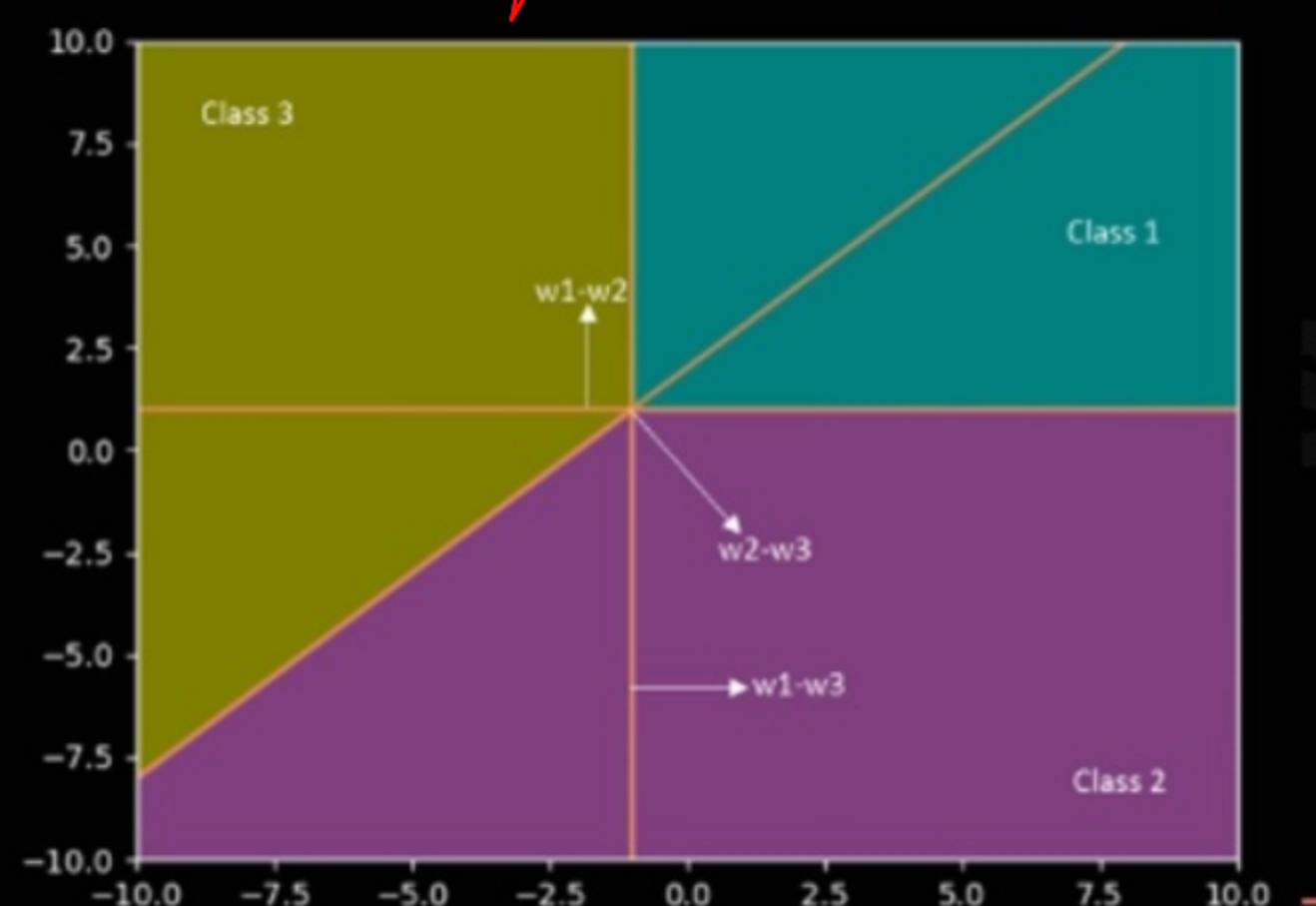
$$\text{class 2 score} > \text{class 3}$$

$$\Rightarrow x_1 + 1 > x_2 - 1$$

$$\Rightarrow \underline{x_1 - x_2 > -2}$$



→ Boundary comes after comparing the score.
→ Combined inequalities create boundary.



Turning into probability.

$$xw^1 \rightarrow s^1$$

$$xw^2 \rightarrow s^2$$

$$xw^3 \rightarrow s^3$$

$$xw^4 \rightarrow s^4$$

$$\vdots$$

$$xw^k \rightarrow s^k$$

k scores

to be

converted

to single probability

distribution

$$\textcircled{1} \rightarrow P(Y=y|x,w) = \frac{xw^y}{\sum_{i=1}^d xw^i}$$

→ Some terms xw^y can be negative
→ Some terms xw^y can be $> \sum_{i=1}^d xw^i$
→ Denominator could become zero

$$\textcircled{2} \quad P(Y=y|x,w) = \frac{\exp(xw^y)}{\sum_{i=1}^d \exp(xw^i)}$$

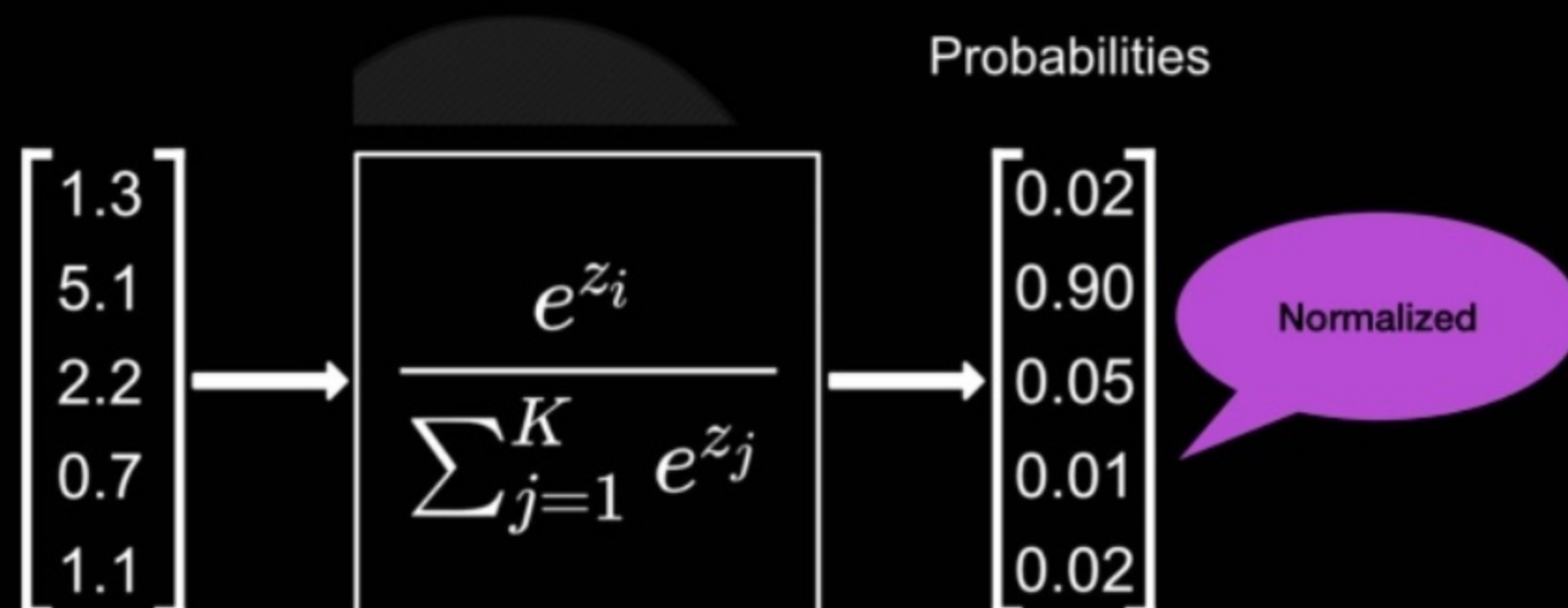
→ Each term in denominator is between 0, ∞
→ Numerator is never $>$ denominator $\Rightarrow P(Y|x,w) < 1$
→ Since each term is +ve $P(Y|x,w) > 0$
→ Sum of all probability = 1

This is softmax/activation

Softmax

Turns outputs z into probabilities (sum up to 1 across K classes)

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_K \end{bmatrix} \rightarrow \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ \vdots \\ e^{z_K} \end{bmatrix} \cdot \frac{1}{\sum_{k=1}^K e^{z_k}}$$



(2 points) What is the definition of $\text{softmax}(x_1, \dots, x_n)$? Recall that this function takes in a list of n real numbers $x_1, \dots, x_n$ and outputs a list of n real numbers.

A. $\text{softmax}(x_1, \dots, x_n) = \left[\frac{x_1}{\sum_{i=1}^n x_i}, \dots, \frac{x_n}{\sum_{i=1}^n x_i} \right]$

✓ B. $\text{softmax}(x_1, \dots, x_n) = \left[\frac{e^{x_1}}{\sum_{i=1}^n e^{x_i}}, \dots, \frac{e^{x_n}}{\sum_{i=1}^n e^{x_i}} \right]$

C. $\text{softmax}(x_1, \dots, x_n) = \left[\frac{e^{x_1}}{e^{\sum_{i=1}^n x_i}}, \dots, \frac{e^{x_n}}{e^{\sum_{i=1}^n x_i}} \right]$

D. $\text{softmax}(x_1, \dots, x_n) = [e^{x_1}, \dots, e^{x_n}]$

James likes big cats. He has a collection of color images, each of which has been accurately labeled as being a leopard, jaguar, cheetah, or lion. He wishes to design a four-way classifier that can classify new images as one of these four species. Your task is to assist James in creating a 4-way classifier for this dataset.

(3 points) It's best to try simple solutions first, so you tell James to try a linear model. Consider (1) linear regression, (2) logistic regression, and (3) softmax regression. Which one would you recommend? Why is it more suitable than the other two?

Given an n -dimensional vector y .

For any real number c , will the softmax of $c \cdot y$ be the same as the softmax of y ?

A. Yes

✓ B. No

$$y \rightarrow \begin{bmatrix} e^{y_1} \\ e^{y_2} \\ \vdots \\ e^{y_n} \end{bmatrix} \frac{1}{\sum_{i=1}^n e^{y_i}}$$

$$cy \rightarrow \begin{bmatrix} e^{cy_1} \\ e^{cy_2} \\ \vdots \\ e^{cy_n} \end{bmatrix} \frac{1}{\sum_{i=1}^n e^{cy_i}}$$

SOFTMAX as generalisation of Sigmoid

$$\text{Sigmoid} = \frac{1}{1+e^{-x}} \quad \text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

→ In binary classification we have two classes, if we use softmax

$$p(y=c|x,w) = \frac{e^{a_1}}{e^{a_1} + e^{a_2}}$$

$$= \frac{1}{1+e^{-(a_1-a_2)}} = \sigma(a_1-a_2) = \sigma(xw' - xw^2) = \sigma(x(w-w^2))$$

But why learn two separate parameters.

→ Apply multiclass/softmax regression on binary classification then it is overkill since it learns redundant weights.

Given input data $x \in \mathbb{R}^3$, how many parameters do the following models have? Assume that linear components of all models have bias terms.

1. (2 points) Binary logistic regression

4

parameter = # features + 1
in case of Bias

2. (2 points) Multiclass logistic regression with two output values, i.e. two classes

8

(3 points) A binary classification problem could be solved with the two approaches described below:

Approach 1: Simple Logistic Regression (one neuron)

Your output will be $\hat{y} = \sigma(W_1 x + b_1)$

Classify as 0 if $\hat{y} \leq 0.5$ and 1 otherwise.

Approach 2: simple softmax regression (two neurons)

Your output will be $\hat{y} = \text{softmax}(W_s x + b_s) = [\hat{y}_1, \hat{y}_2]^T$

Classify as 0 if $\hat{y}_1 \geq \hat{y}_2$ and 1 otherwise.

Approach 2 involves twice as many parameters as approach 1. Can approach 2 learn more complex models than approach 1?

If yes, give the parameters (W_s, b_s) of a function that can be modeled by approach 2 but not by approach 1. If no, show that (W_s, b_s) can always be written in terms of (W_1, b_1) .

$y=1$ if $\frac{e^{W_s^1 x + b_s^1}}{e^{W_s^1 x + b_s^1} + e^{W_s^2 x + b_s^2}} > \frac{e^{W_s^2 x + b_s^2}}{e^{W_s^1 x + b_s^1} + e^{W_s^2 x + b_s^2}}$

softmax

$\Rightarrow e^{W_s^1 x + b_s^1} > e^{W_s^2 x + b_s^2} \rightarrow$ since denominators same

$\Rightarrow W_s^1 x + b_s^1 > W_s^2 x + b_s^2 \rightarrow$ since log & exp are monotonic.

$\Rightarrow (W_s^1 - W_s^2)x + (b_s^1 - b_s^2) > 0$

$\rightarrow$ given a softmax model we can put $\begin{cases} W_1 = W_s^1 - W_s^2 \\ b_1 = b_s^1 - b_s^2 \end{cases}$ and achieve the same result using logistic regression.

$\rightarrow$ Intuition $\rightarrow$ There will be two hyperplanes, but the decision boundary is just one comparison statement. This means this decision boundary is also an hyperplane, which can be learn by a logistic regressor.

Consider the function $f: \mathbb{R}^d \mapsto \mathbb{R}^d$ whose components are given by

$$f_i(x) = \frac{x_i^2}{\sum_{k=1}^d x_k^2}$$

for $i = 1, \dots, d$ and $x \in \mathbb{R}^d$.

We can see that $0 \leq f_i \leq 1$ for each $i = 1, \dots, d$ and $\sum_{i=1}^d f_i = 1$. Thus, f normalizes the input vector x to a probability distribution, just like Softmax does! That said, what is one way in which our function f differs from the Softmax function?

$f_i(x)$ maps $\begin{matrix} -\infty & \text{to} & 1 \\ +\infty & \text{to} & 1 \\ 0 & \text{to} & 0 \end{matrix}$ } does not preserve order.
 $f_i(x) = f_j(x) \nRightarrow x_i = x_j$

softmax(x) maps $\begin{matrix} -\infty & \text{to} & 0 \\ +\infty & \text{to} & 1 \end{matrix}$

$\Rightarrow$ softmax preserves order
 $xw^k > xw^l \Rightarrow e^{xw^k} > e^{xw^l}$

Insight $\rightarrow$ vectors with all equal values $y \in \mathbb{R}^d$ $y = [k \ k \ k \dots k]^T_{1 \times d}$
then $\text{softmax}(y) = [\frac{1}{d} \ \frac{1}{d} \ \frac{1}{d} \dots \frac{1}{d}]^T_{1 \times d}$

$\rightarrow$ adding a equal value to all components in the vector does not changes the softmax
eg: $y+k = [y_1+k \ y_2+k \ y_3+k \dots y_d+k]^T_{1 \times d}$

$\text{softmax } y+k = \frac{1}{\sum_{i=1}^d e^{y_i+k}} [e^{y_1+k} \ e^{y_2+k} \dots e^{y_d+k}]^T$

$\text{softmax } y+k = \frac{1}{e^k \sum_{i=1}^d e^{y_i}} [e^{y_1} \ e^{y_2} \dots e^{y_d}]^T = \text{softmax } y$

Solution: No. For approach 1, the classifier is $\sigma(w_1 x + b_1) \geq 0.5$ which is equivalent to $w_1 x + b_1 \geq 0$.

Now, for approach 2, define w_1^1 as the weights for the first output and w_2^2 as the weights for the second output. Similarly, define b_1^1 as the bias for the first output and b_2^2 as the bias for the second output.

For approach 2, the classifier is $\frac{\exp(w_1^1 x + b_1^1)}{\exp(w_1^1 x + b_1^1) + \exp(w_2^2 x + b_2^2)} \geq \frac{\exp(w_2^2 x + b_2^2)}{\exp(w_1^1 x + b_1^1) + \exp(w_2^2 x + b_2^2)}$. Since the denominator is positive and same on both sides, this is equivalent to $\exp(w_1^1 x + b_1^1) \geq \exp(w_2^2 x + b_2^2)$.

Since $\exp$ is a monotonic increasing function, this is equivalent to $w_1^1 x + b_1^1 \geq w_2^2 x + b_2^2$.

This is equivalent to $(w_1^1 - w_2^2)x + (b_1^1 - b_2^2) \geq 0$.

Given any w_2 and b_2 in approach 2, we can get the exact same model by setting $w_1 = w_1^1 - w_2^2$ and $b_1 = b_1^1 - b_2^2$ in approach 1.

Alternatively, one can argue based on degrees of freedom.

Note that in this case, we are asking about model complexity, which is independent of loss function.

predicted $y = \text{softmax}(xw) = \begin{bmatrix} P(y=1/x,w) & P(y=2/x,w) & \dots & P(y=d/x,w) \end{bmatrix}$ $d = \# \text{ classes}$

real $y = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix}_{1 \times d}$ } correct class will have 1
ONE-HOT REPRESENTATION *

Loss $L(w|y_i, x_i) = \underbrace{-y_i \ln(p_i)}_{\text{either 1st term or 2nd term — but always only 1 term}} + \underbrace{-(1-y_i) \ln(1-p_i)}_{\text{either 1st term or 2nd term — but always only 1 term}}$

→ for logistic compact form → only 1 term

$= -\ln(\sigma(y_i x_i w))$
 $= -\ln(\text{predicted probability})$

Now, similarly for softmax

$L(w|y_i, x_i) = -y_i^T \ln(\hat{y}_i)$

→ example

$y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ $\hat{y} = \begin{bmatrix} 0.006 \\ 0.007 \\ 0.543 \\ 0.010 \end{bmatrix}$
 $y^T \hat{y} = 0.543 = \text{predicted probability}$

Total loss = $\sum_{i=1}^N -y_i^T \ln(\text{softmax}(x_i w)) = \sum_{i=1}^N \sum_{j=1}^d y_i^j \ln(\text{softmax}(x_i^j w))$

One of the objectives of cross-entropy loss is to both increase the probability of the true class and decrease the probabilities of the other classes. The formula for cross-entropy loss is given by:

Cross-Entropy Loss = $-\log(\text{Predicted Probability of True Class})$

From the formula, it is clear that minimizing the loss means increasing the probability of the true class. Argue that the cross-entropy loss function also ensures that the probabilities of the other classes are reduced.

A) As the probability of the true class increases, the total sum of probabilities across all classes decreases, which inherently reduces the probabilities of the other classes.

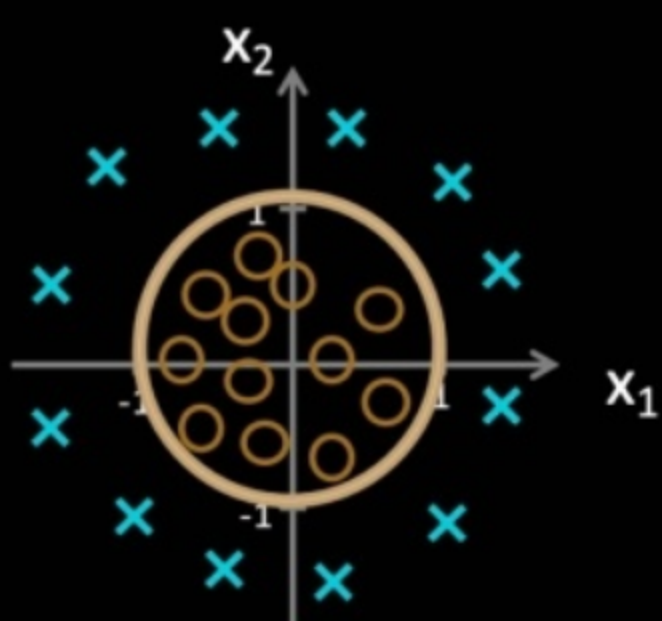
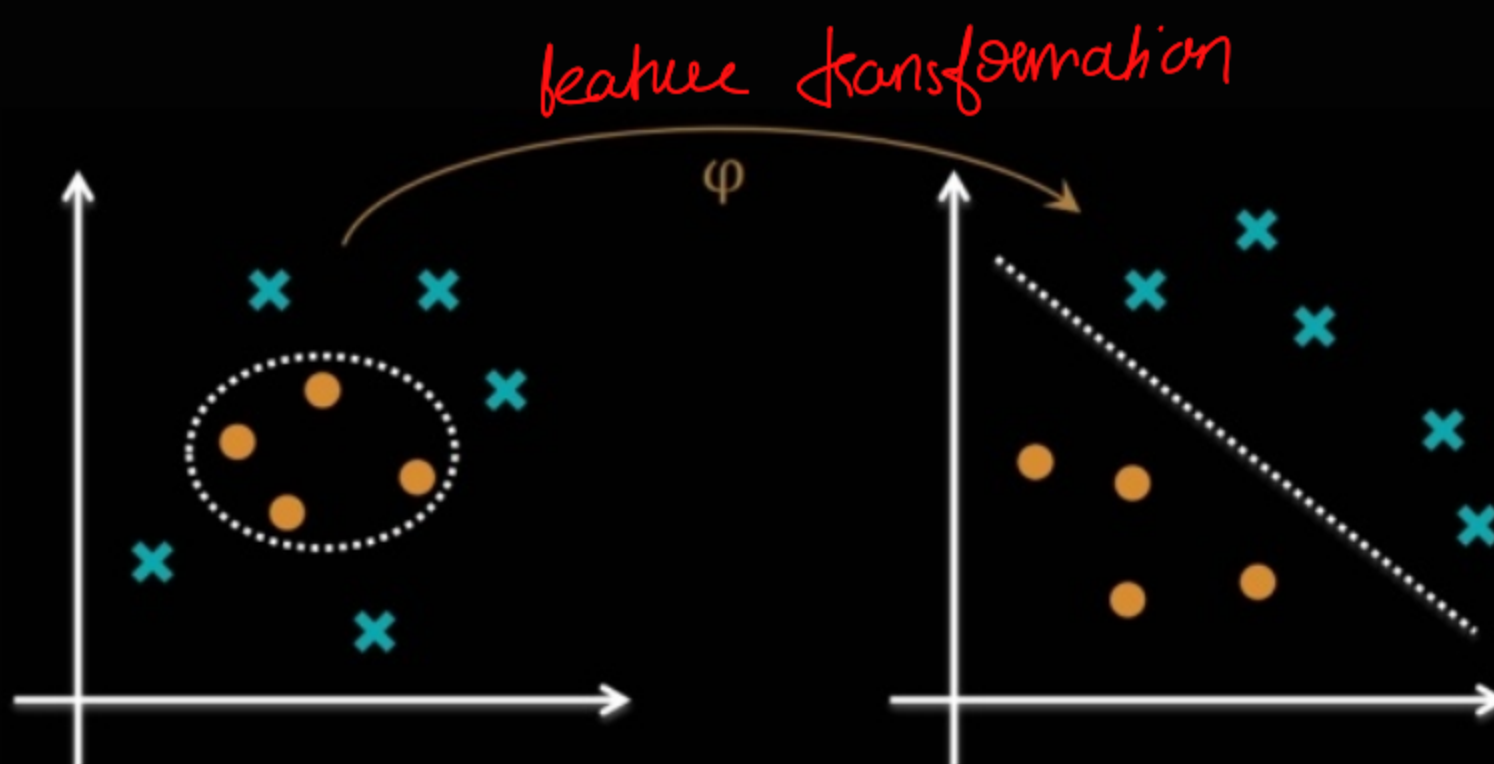
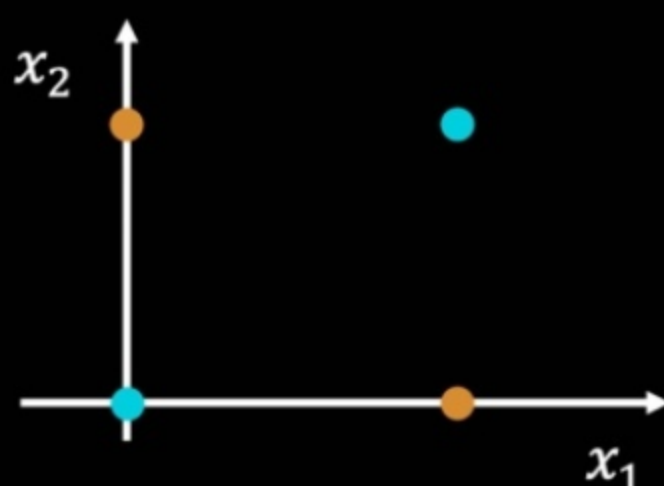
B) Cross-entropy loss does not affect the probabilities of the other classes; it only focuses on increasing the probability of the true class.

✓ C) Minimizing the cross-entropy loss requires that the probability of the true class is maximized, which leads to a natural reduction in the probabilities assigned to the other classes due to the constraint that probabilities must sum to one.

D) Cross-entropy loss is calculated only for the true class, so it does not influence the probabilities of other classes.

Feature transformation to separate with linear boundary

How to classify this ?



$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$

Predict $y = 1$ if $-1 + x_1^2 + x_2^2 \geq 0$

decision boundary is not linear
 → use feature x_1^2 & x_2^2
 instead of x_1, x_2 then
 we can convert the boundary
 into linear inequality.

linear inequality $\rightarrow$ linearly separable $\rightarrow$ linear hyperplane
 feature transformation
 non linear decision boundary.

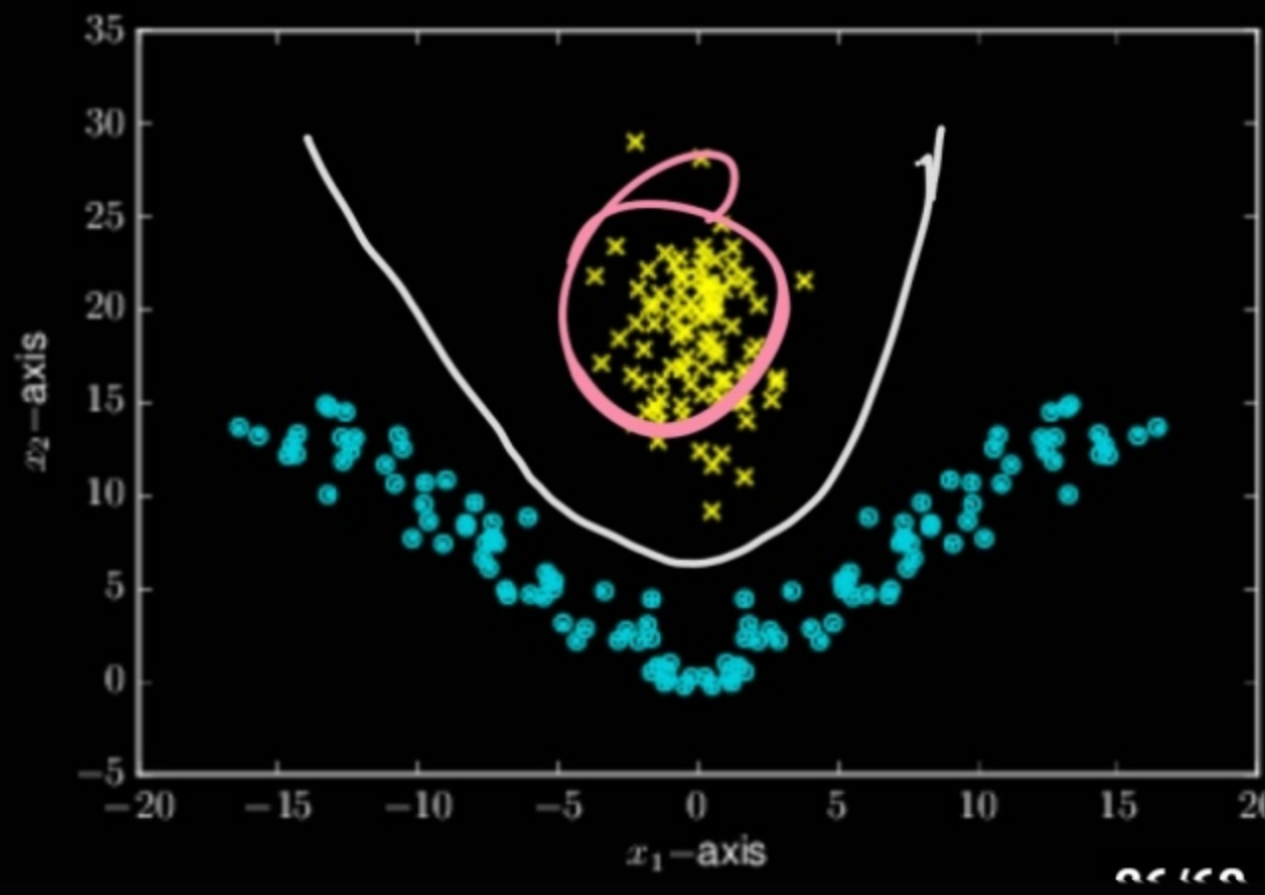
QUESTION.

$$x_2 = ax_1^2 + bx_1 + c$$

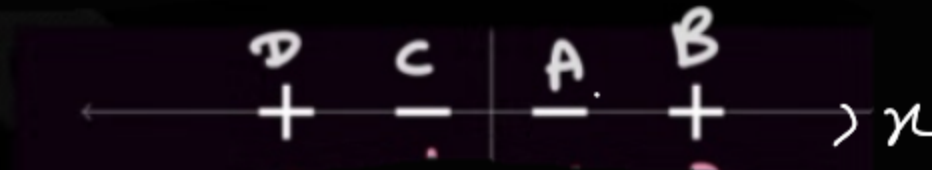
Give the possible feature transformation such that given data is linearly separable.

$$y = ax^2 + bx + c$$

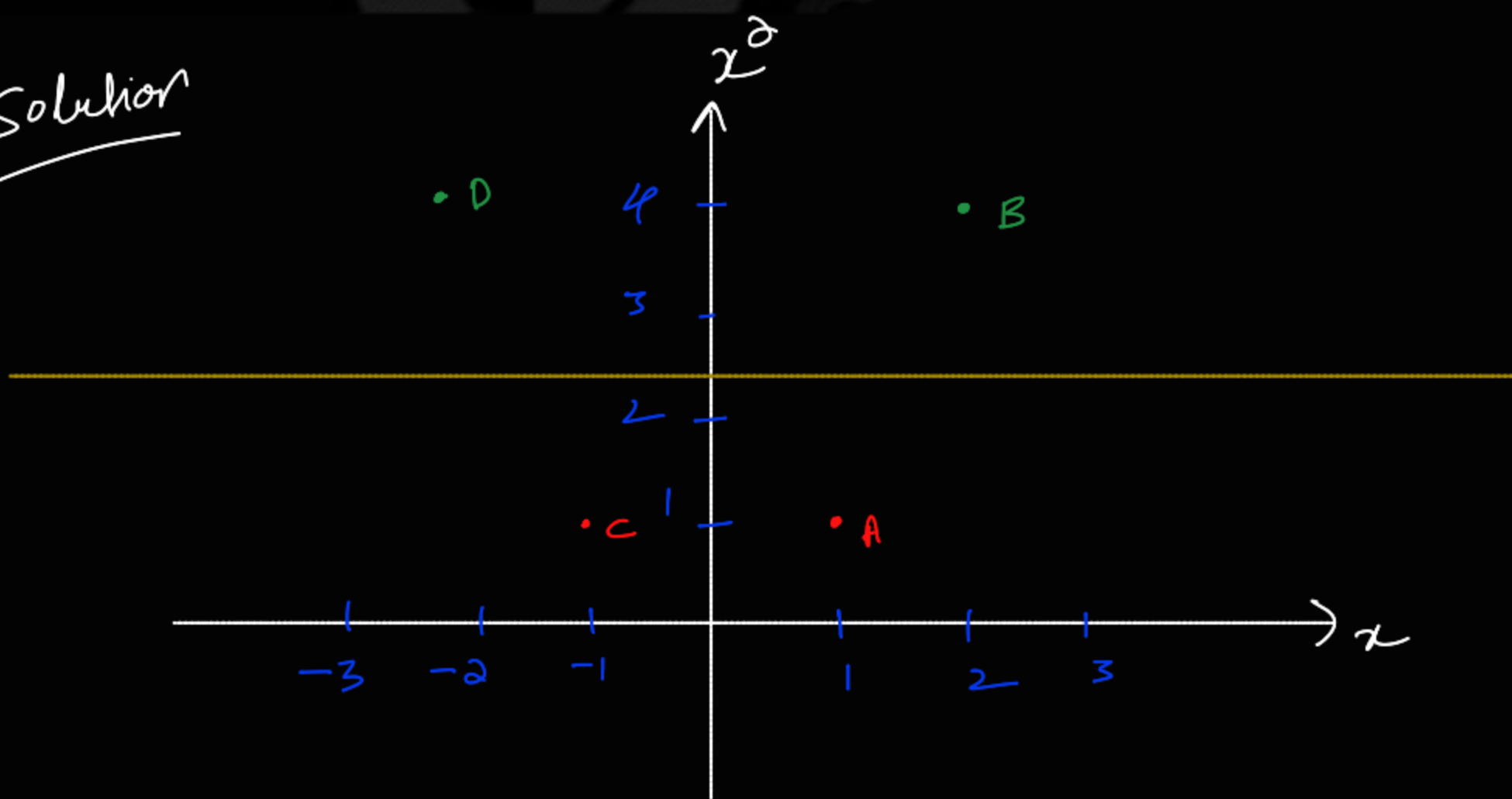
$$x_2 = ax_1^2 + bx_1 + c$$



Give the possible feature transformation such that given data is linearly separable.



Solution



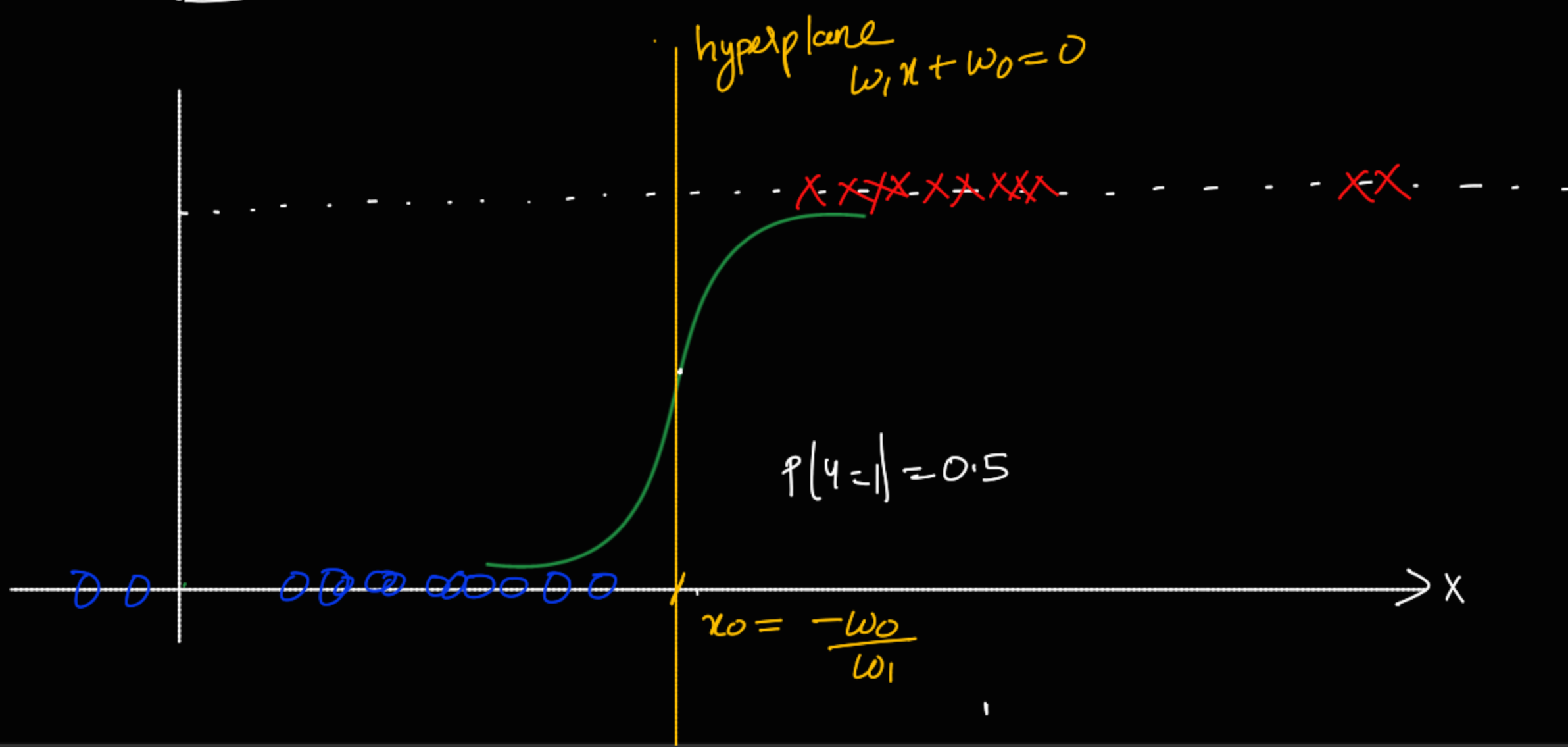
So introducing x^2 as a feature it became linearly separable.

UNDERSTAND SIGMOID GRAPHICALLY

$$P(y=1) = \frac{1}{1+e^{-xw}} = \frac{1}{1+e^{-(xw+w_0)}}$$

$\rightarrow w_0$ shifts the curve right or left
 w controls how steep the S-shaped curve is

Logistic regression as curve fitting



Logistic Regression & Log(odds)

☆ odds = $\frac{P(\text{success})}{P(\text{failure})} = k \Rightarrow k \text{ times more chance for success than failure } (k > 1)$
 $\frac{1}{k} \text{ time more chance for failure than success } (k < 1)$
 $\begin{cases} \rightarrow \in (0, \infty) \\ \rightarrow \mathbb{R}^+ \end{cases}$
 $k = 1 \Rightarrow \text{equally like to fail \& succeed}$

☆ $\log(\text{odds}) \in (-\infty, \infty) = \ln \left[\frac{P(\text{success})}{P(\text{failure})} \right]$
 $\hookrightarrow \text{aka logit} \in \mathbb{R}$

$\rightarrow$ Instead of using Logistic Regression to model $P(Y=1)$, we will model $\log$ odds i.e. $\ln \left(\frac{P(Y=1)}{P(Y=0)} \right) = \ln \left(\frac{p}{1-p} \right)$ using Linear Regression

$\rightarrow$ let $y = \ln \left(\frac{p}{1-p} \right)$ is modelled using xw $\neq$ Linear Regression

$$\Rightarrow \ln \left(\frac{p}{1-p} \right) = xw \Rightarrow \frac{p}{1-p} = e^{xw} \Rightarrow p(1+e^{xw}) = e^{xw}$$

$$\Rightarrow p = \frac{e^{xw}}{1+e^{xw}} = \frac{1}{1+e^{-xw}} = \sigma(xw) \quad \left. \begin{array}{l} \text{logistic function} \\ \text{Sigmoid function} \end{array} \right\}$$

Insight

$\rightarrow$ Modelling $\log(\text{odds})$ for a classification problem using Linear Regression is equivalent to using sigmoid function in the probabilistic model of Logistic Regression. Gradient descent stage the actual truth for $\log(\text{odds})$ will be $\pm \infty$. so meaningful learning won't happen so instead we use logistic regressor

k-Nearest Neighbours

$kNN \sim O(Nm)$ $m = \# \text{features}$
 $N = \# \text{data size}$

KNN $\rightarrow$ memory/instance based method
 DLR $\rightarrow$ parameter/model based model

To predict category label y of a new point x (classification):

- Find k nearest neighbors (according to some distance metric)
- Assign the majority label to the new point

To predict numeric value y of a new point x (regression):

- Find k nearest neighbors
- "Average" the values associated with the neighbors

Note: Changing k may result in a different prediction.

- $\rightarrow$ No training
- $\rightarrow k$ hyperparameter — fixed using validation dataset
- $\rightarrow$ We need all the data to predict the result
- $\rightarrow$ Unlike previous algorithms, we need data for prediction. We don't have learned weights.

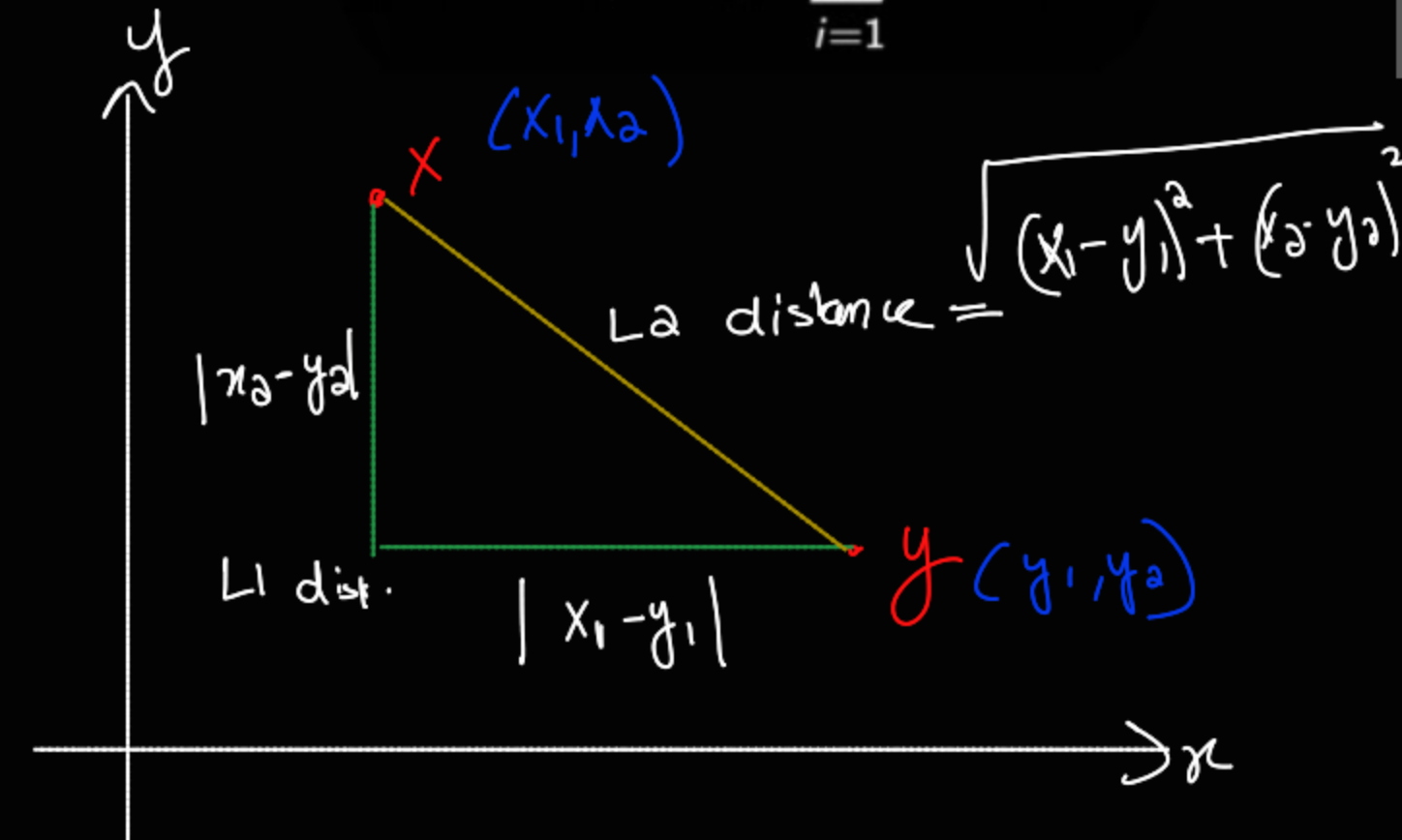
	Train Cost (Once)	Test Cost (Each)
KNN	NO	High
DLR, Ridge, Logistic	High	Vely Low $y = wx + b$

► **Euclidean Distance:**
 L_2 Distance

$$d_{\text{Euclidean}}(x, y) = \sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

► **Manhattan Distance:**
 L_1 distance

$$d_{\text{Manhattan}}(x, y) = \sum_{i=1}^k |x_i - y_i|$$



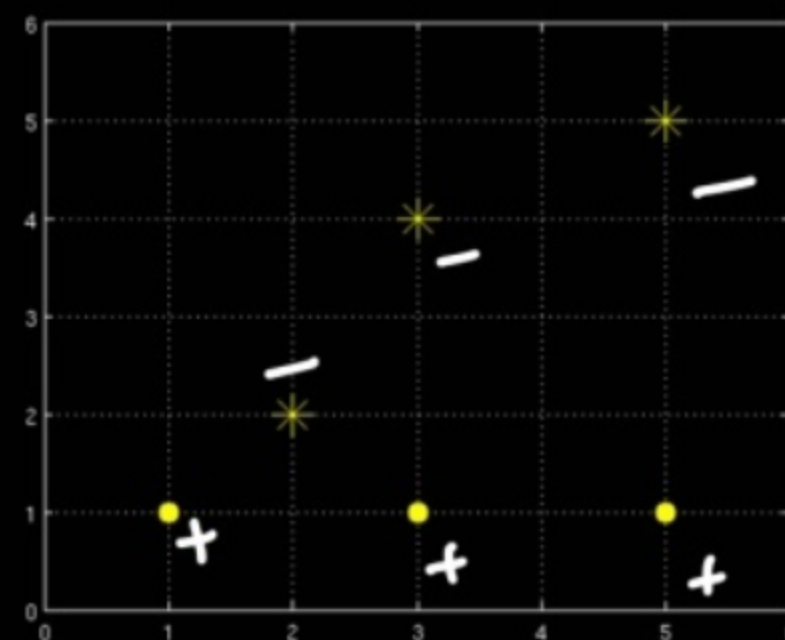
classify $(3, 6)$ using 3-NN & L1 distance

Training Example (x, y)	Class Label (c)
(1, 1)	-1
(1, 7)	+1
(3, 3)	+1
(5, 4)	-1
(2, 5)	-1

L1 dist
 $2+5=7$
 $2+1=3$
 $0+3=3$
 $2+2=4$
 $1+1=2$
 $\therefore \frac{2(+)}{3} = (+)$

[2 points] What is the training error (fraction labeled wrong) in the above picture using 1-nearest neighbor and the L_1 distance norm between points? If there is more than one equally close nearest neighbor, use the majority label of all the equally close nearest neighbors. Please make sure you really are computing training error, not testing error.

- (a) 0
- (b) 1/6
- (c) 2/6
- (d) 3/6
- (e) 4/6

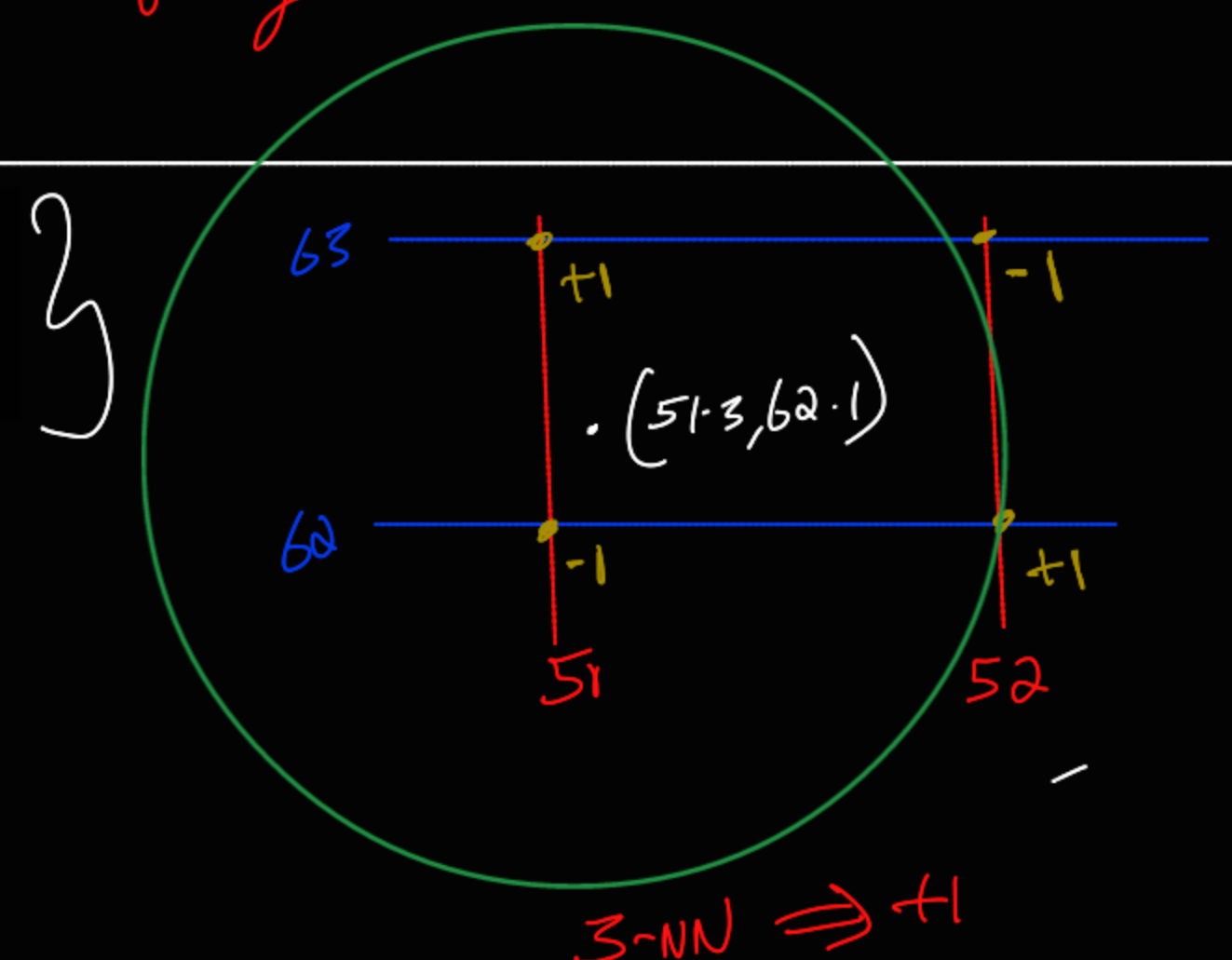


The closest point is itself
 $\therefore$ Training error always zero in 1-NN

Training error here is the error you'll have when you input your training set to your KNN as test set. When $K = 1$, you'll choose the closest training sample to your test sample. Since your test sample is in the training dataset, it'll choose itself as the closest and never make mistake. For this reason, the training error will be zero when $K = 1$, irrespective of the dataset. There is one logical assumption here by the way, and that is your training set will not include same training samples belonging to different classes, i.e. conflicting information. Some real world datasets might have this property though.

1-NN $\rightarrow$ overfitting

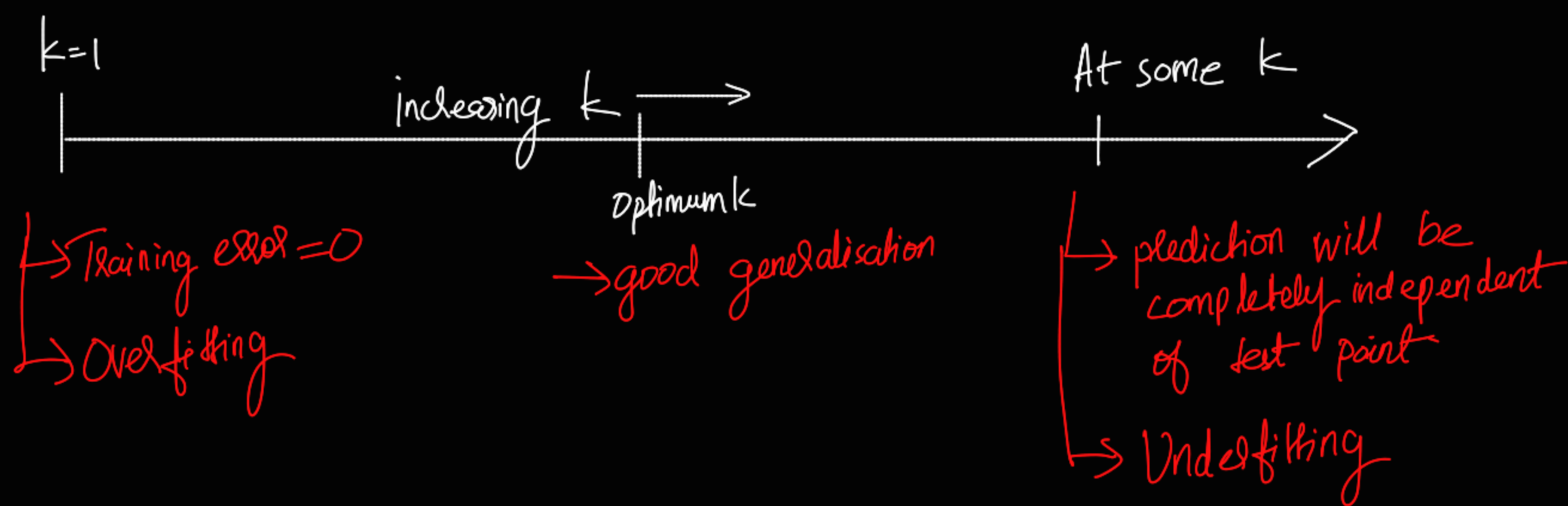
(kNN) Consider a 2D dataset where training points are on the integer grid (x_1, x_2) , and both dimensions x_1 and x_2 range from 1 to 100 (inclusive). The binary label for point (x_1, x_2) is $(-1)^{(x_1+x_2)}$. What is the label for test point $(51.3, 62.1)$ with a 3-NN classifier?



(kNN) Consider points in 2D and binary labels. Given the following training data: $x_1=(1,1)$, $y_1=1$, $x_2=(-1,1)$, $y_2=0$, $x_3=(-1,-1)$, $y_3=1$, $x_4=(1,-1)$, $y_4=0$, and use Manhattan distance, how will 1NN classify any points in 2D?



1st & 3rd quadrant $y=1$
 2nd & 4th quadrant $y=0$



You are using K-Nearest Neighbors (KNN) regression with $K = 3$. Below is the dataset of the nearest neighbors and their corresponding target values:

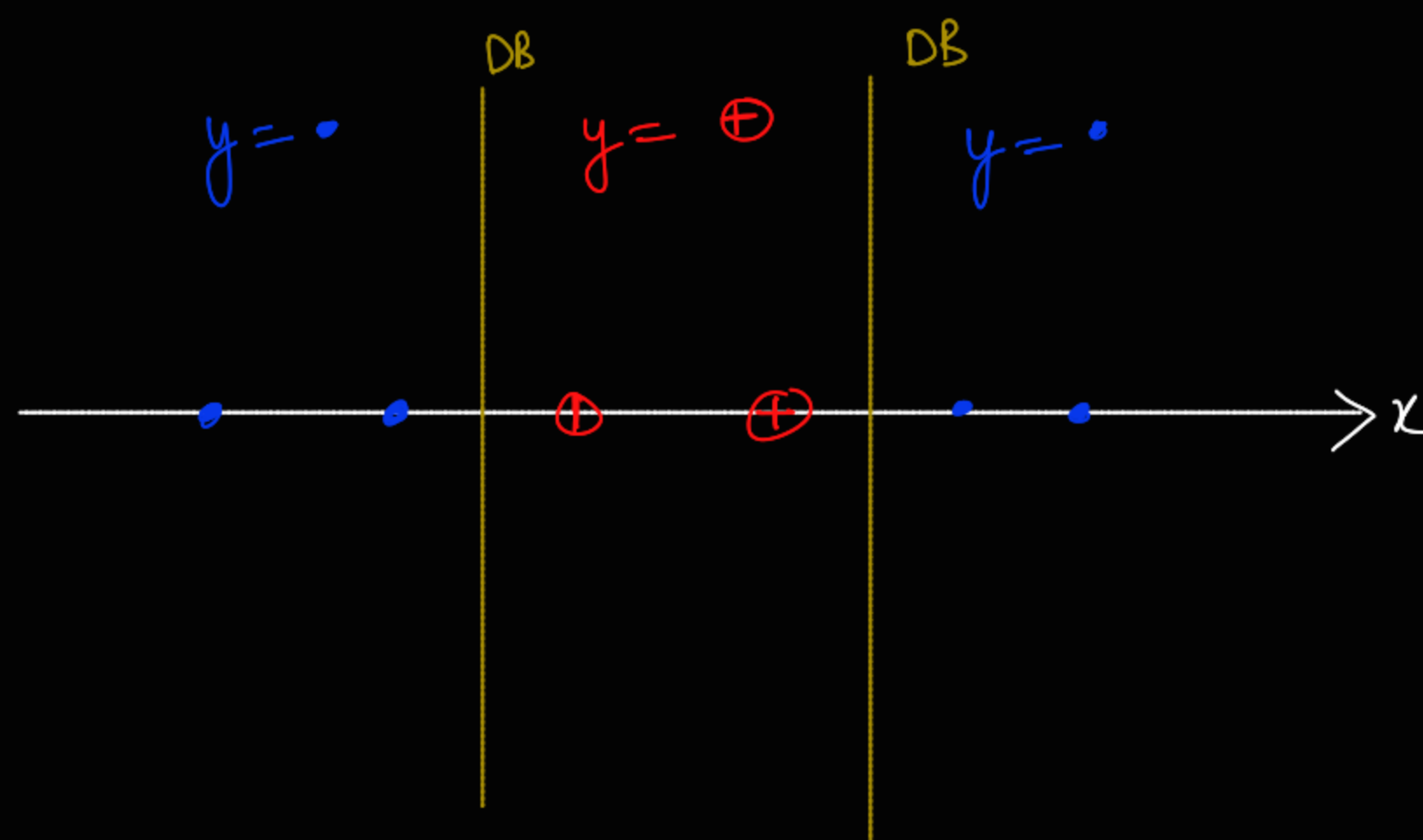
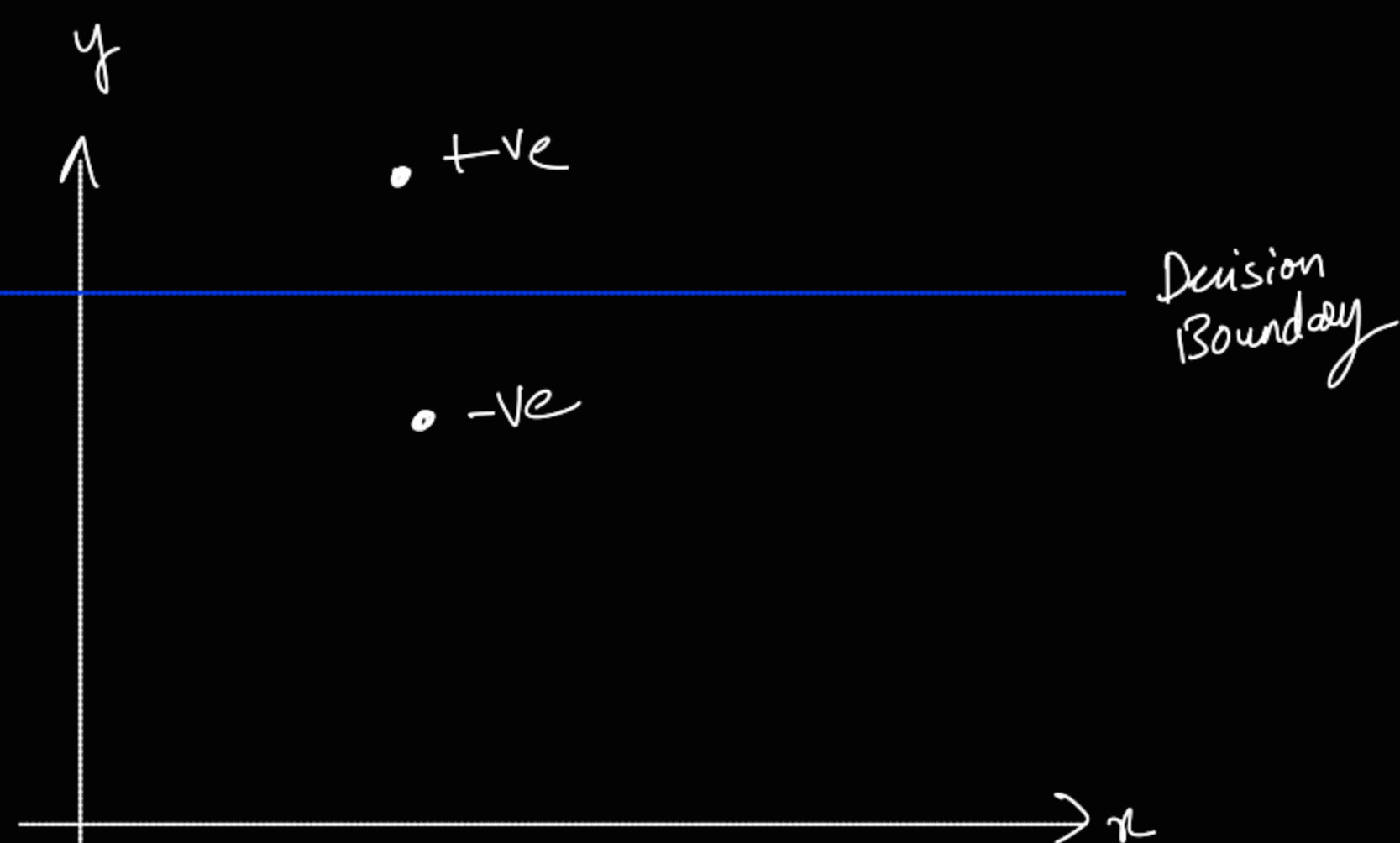
Neighbor	Distance from q	Target Value
N1	1.2	15
N2	1.5	20
N3	1.7	25
N4	2.0	30
N5	2.5	35

$$\hat{y} = \frac{15 + 20 + 25}{3} = 20$$

Using simple averaging (mean) of the target values of the 3 nearest neighbors, what will be the predicted value for the new data point?

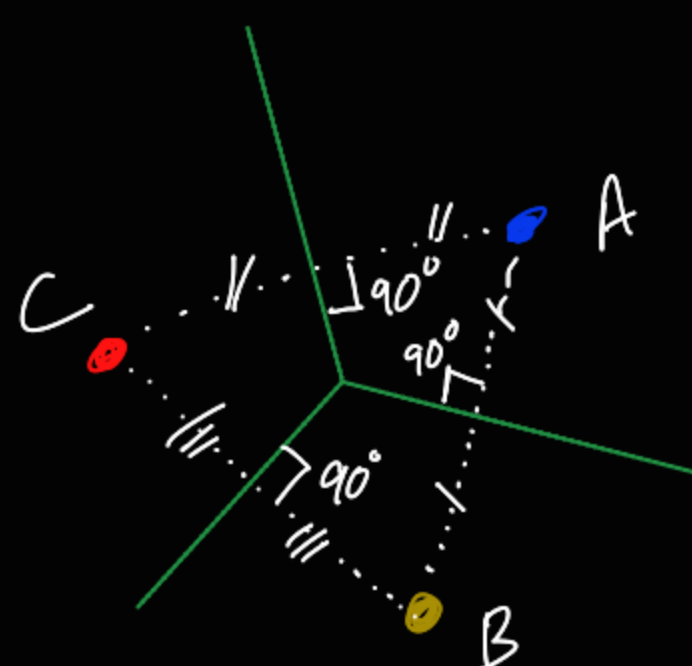
DECISION BOUNDARY

① Decision Boundary → the k nearest points change in a way that the majority vote shifts.

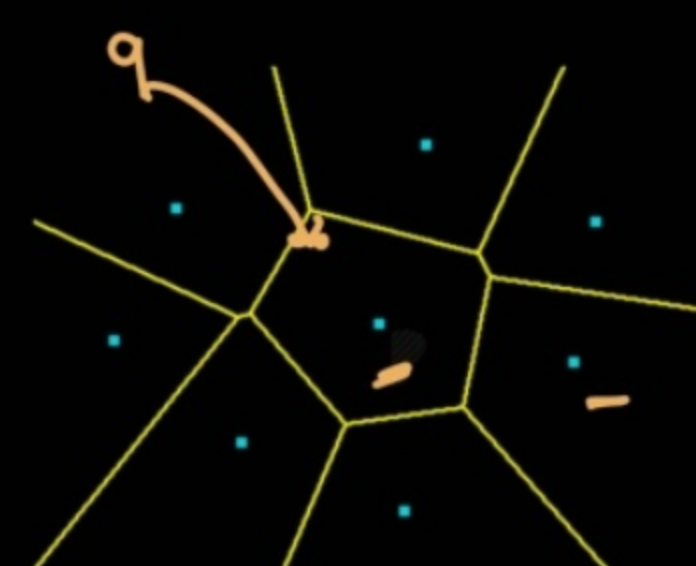


→ How to find decision boundary

- ① Join the datapoints using straight line
- ② Find the perpendicular bisector of this straight line
- ③ if DB is between 2 points of same class then skip



- Given a set of points, a Voronoi diagram describes the areas that are nearest to any given point.
- These areas can be viewed as zones of control
- Each zone is controlled by one of the points

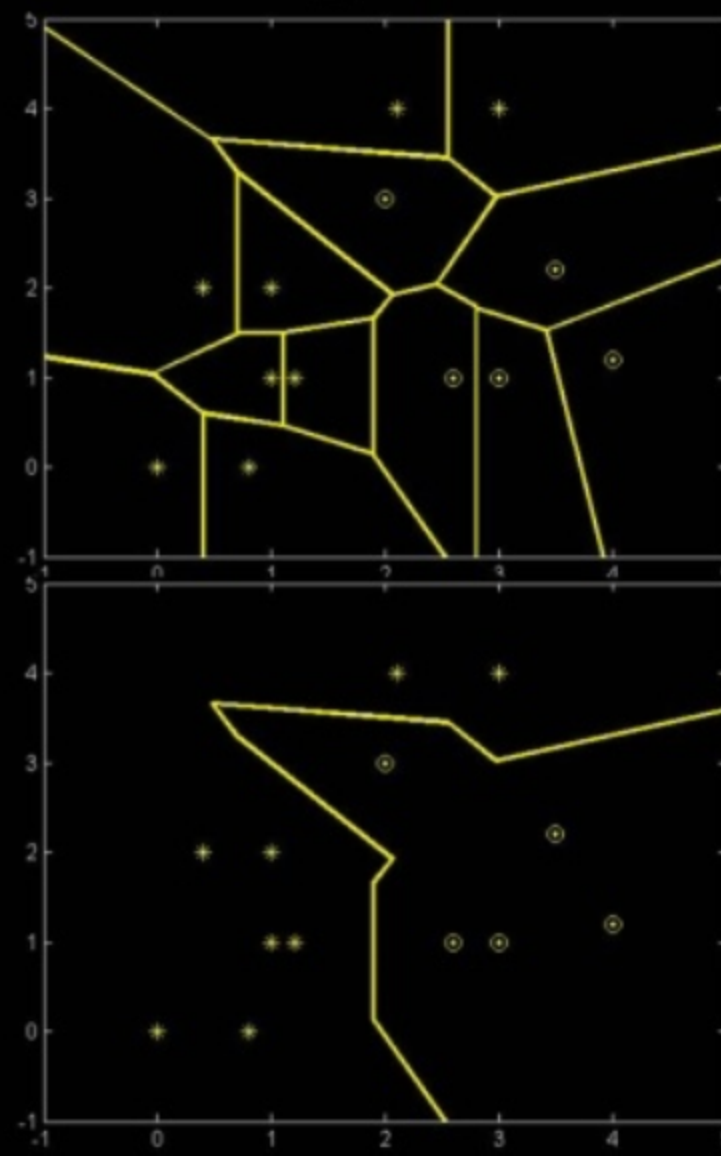


1NN

Voronoi diagram

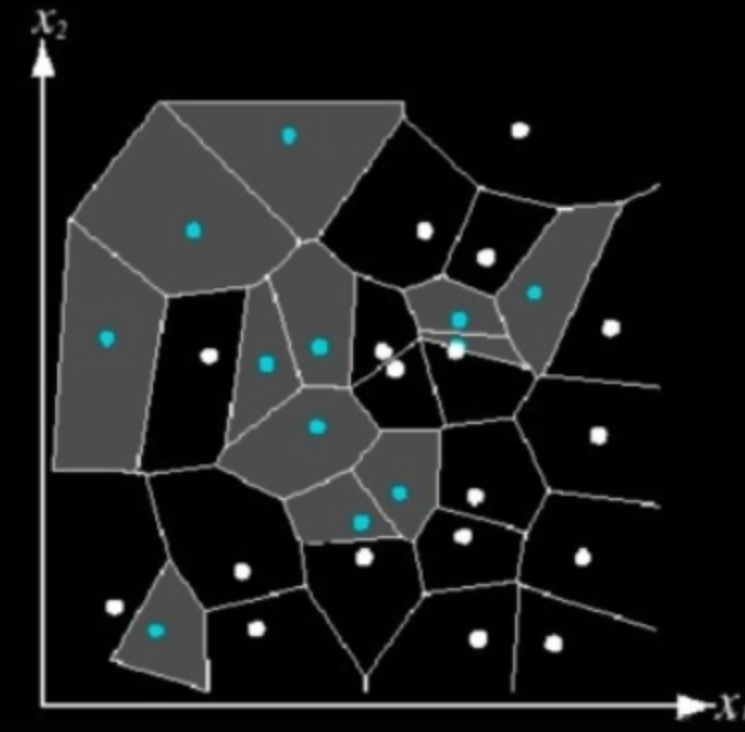
DB is Subset of the Voronoi Diagram

- Each example controls its own neighborhood
- Create the voronoi diagram
- Decision boundary are formed by only retaining these line segments separating different classes.
- The more training examples we have stored, the more complex the decision boundaries can become



- Decision boundaries: **Voronoi diagram** visualization

- ▶ show how input space divided into classes
- ▶ each line segment is equidistant between two points of opposite classes

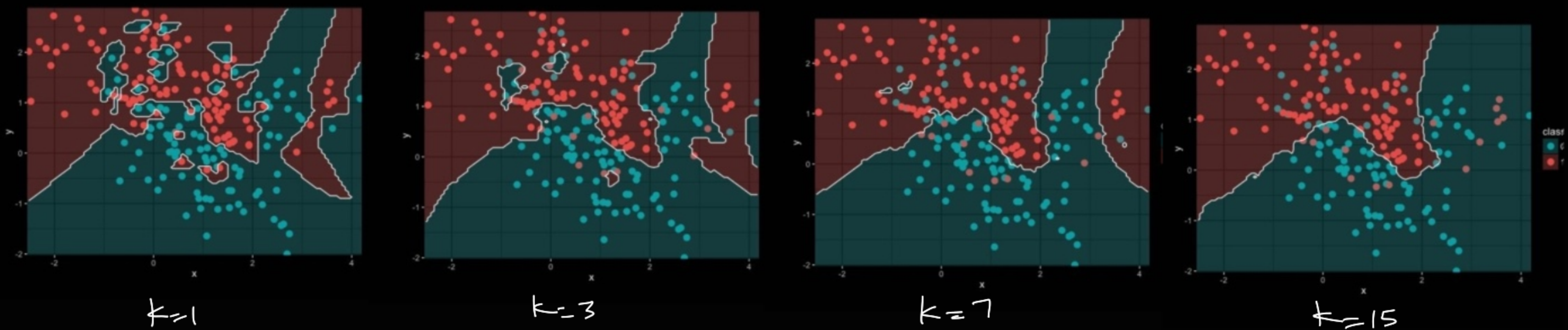


Decision Boundary is piecewise linear

[6 points] TRUE or FALSE: for any two neighboring Voronoi cells C_1 and C_2 corresponding to two training examples $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$, the boundary between C_1 and C_2 is part of the decision boundary of the 1-nearest-neighbor classifier. Briefly justify your answer.

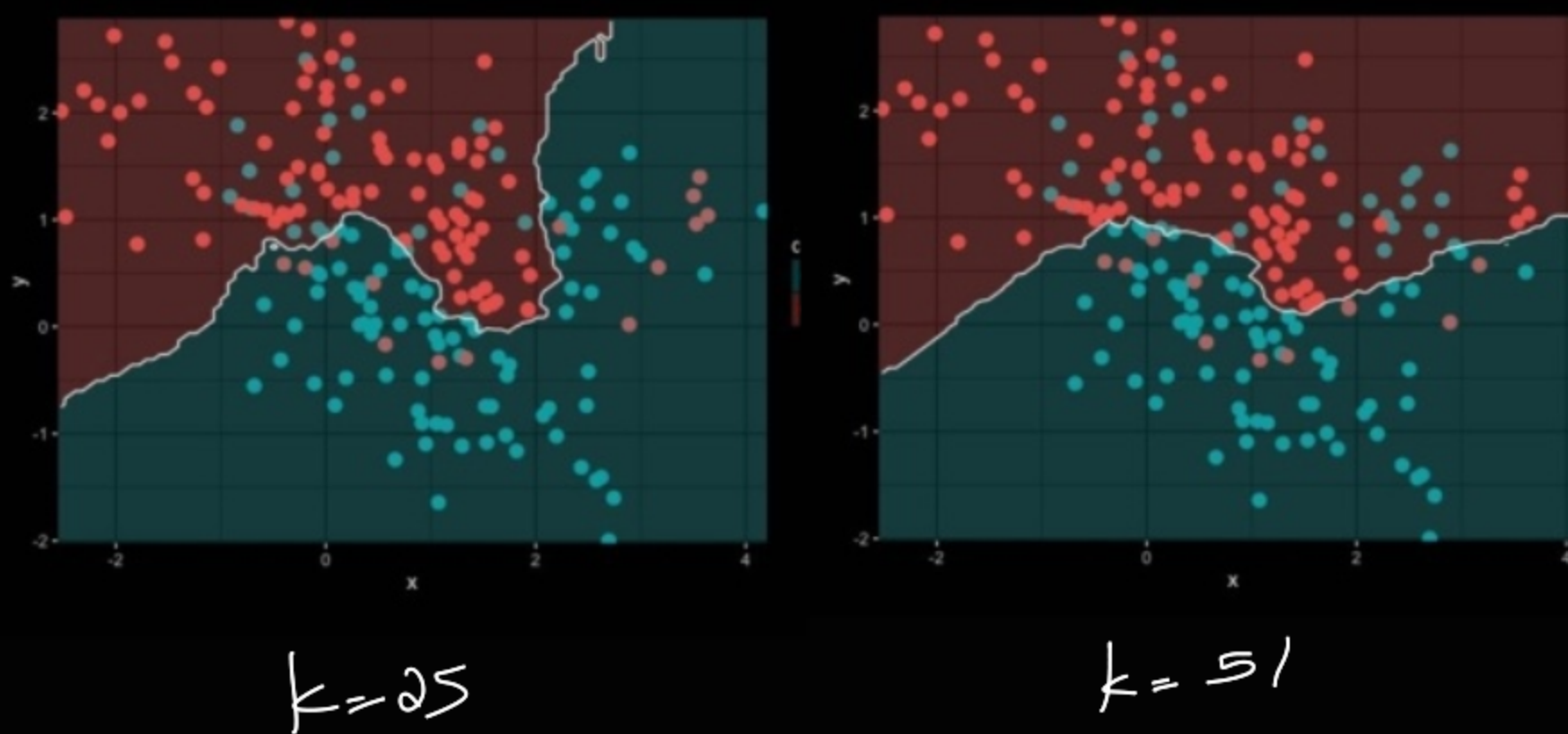
False
if C_1 class = C_2 class then boundary is not in decision boundary

→ k-NN algorithm inherently doesn't frame or learn the decision boundary. It if just what can be inferred.



Small k

- $k = 1$
- Low bias
- Models like this are **overfit**
 - Read too much into the training data extrapolating based on things that aren't necessarily relevant
- High variance: model varies greatly with the data

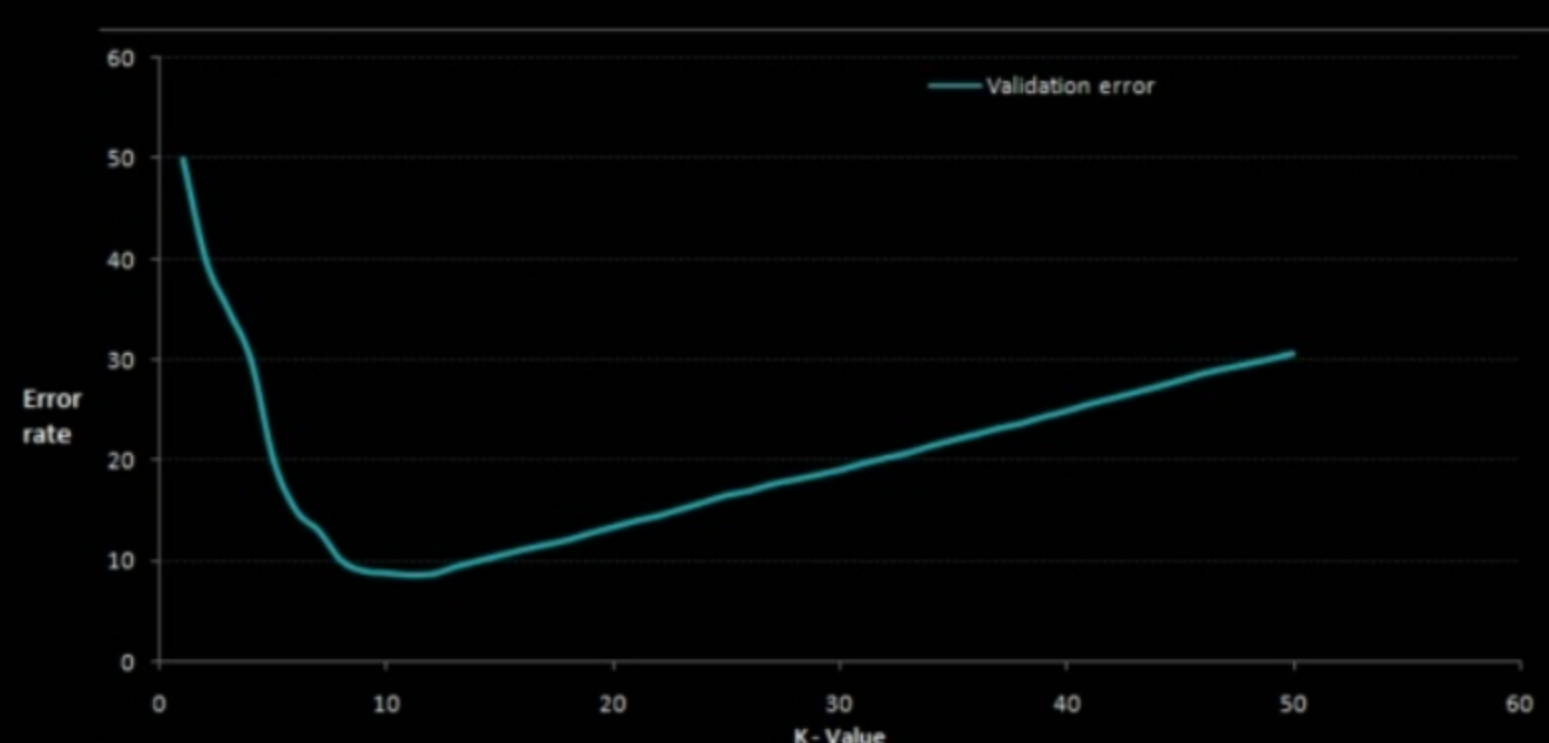


Large k

- $k = 101$
- High bias
- Rather than being **overfit**, this model is **underfit**
 - The decision boundary doesn't capture enough of the relevant information encoded in the training data
- Low variance: model barely varies with the data

In the image below, which would be the best value for k, assuming you are using the k-nearest neighbor algorithm?

- 3
- 10
- 20
- 50



(2 points) Which of the following statements are true?

- ☐ Training a k-nearest-neighbors classifier takes more computational time than applying it.
 - ☒ The more training examples, the more accurate the prediction of a k-nearest-neighbors.
 - ☐ k-nearest-neighbors cannot be used for regression.
 - ☒ A k-nearest-neighbors is sensitive to outliers. for small k yes
- Noise

Any algorithm → more data is good-

Select one: Imagine you are using a k-Nearest Neighbor classifier on a dataset with lots of noise. You want your classifier to be *less* sensitive to the noise. Which of the following is likely to help and with what side effect?

- ☒ Increase the value of k → Increase in prediction time
- ☐ Decrease the value of k → Increase in prediction time
- ☐ Increase the value of k → Decrease in prediction time
- ☐ Decrease the value of k → Decrease in prediction time

Naive Bayes Classifier

BAYES OPTIMAL CLASSIFIER

$$\hat{y} = \underset{y}{\operatorname{argmax}} (P(Y=y | X=x))$$

Softmax we had same thing
But in bayes optimal
we have different model
for $P(Y=y | X=x)$

BAYES rule

$$P(Y/x) = \frac{\overbrace{P(X/Y)}^{\text{likelihood}} \cdot \overbrace{P(Y)}^{\text{prior}}}{\underbrace{P(X)}_{\text{normalizes}}} = \frac{P(X,Y)}{P(X)}$$

This is from
 $P(Y/x) P(X) = P(X/Y) \cdot P(Y) = P(X,Y)$

Bayes classifier

$$\hat{y} = \underset{y}{\operatorname{argmax}} \left[\frac{P(Y=y, X=x)}{P(X=x)} \right]$$

$P(X=x)$ in denominator can be ignored
since it is same for all $y = 1, 2, 3, \dots, d$
 $d = \# \text{classes}$

$$\Rightarrow \hat{y} = \underset{y}{\operatorname{argmax}} \left[\underbrace{P(Y=y, X=x)}_{\text{Joint probability distribution}} \right] = \underset{y}{\operatorname{argmax}} \left[\underbrace{P(X/Y) \cdot P(Y)}_{\text{calculated from data}} \right]$$

Logistic Regression

- In logistic regression, the model for $P(y | X = x)$ is:

$$P(y | X = x) = \frac{1}{1 + e^{-w^T x}}$$

- This directly models $P(Y | X)$, which is known as a "discriminative model".

eg. You have a green-black avocado, and want to know if it is ripe. Based on this data, would you predict that your avocado is ripe or unripe?

Train Data:

color	ripeness
bright green	unripe
green-black	ripe 1
purple-black	ripe 2
green-black	unripe
purple-black	ripe 3
bright green	unripe
green-black	ripe 4
purple-black	ripe 5
green-black	ripe 6
green-black	unripe
purple-black	ripe 7

Test Point: green-black avocado

$$P(\text{ripe} | \text{green-black}) = ? \quad 3/5$$

$$P(\text{unripe} | \text{green-black}) = ? \quad 2/5$$

$$P(\text{ripe}) = \frac{7}{11}$$

$$P(\text{unripe}) = \frac{4}{11}$$

prior

if we don't calculate posterior & only look for prior, then our classification will be fixed irrespective of datapoint.

$$P(\text{GB} | \text{ripe}) = 3/7$$

likelihood

$$P(\text{GB}) = 5/11$$

Normalised

Since $P(R|GB) > P(U|GB)$
 $\Rightarrow \hat{y} = \text{Ripe} | \text{greenblack}$

$$P(\text{GB} | \text{unripe}) = 2/4$$

→ Note: Here only 1 feature → Calculate posterior directly from data.

→ Calculation prior is not complex & computationally inexpensive

eg.

x_1	x_2	x_3	x_4	x_5	x_6	y
Sky	Temp	Humid	Wind	Water	Forecast	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

→ $P(y)$ or $P(y')$ is easy

Just count → for d # classes we need to estimate $(d-1)$ probabilities since $\sum_{i=1}^d P(y=i) = 1$ → so the last prob is computed from previous $d-1$

$$P(y/x_1, x_2, \dots, x_d) = \frac{P(y) \cdot P(x_1, x_2, \dots, x_m | y)}{P(x_1, x_2, \dots, x_m)}$$

Training → given a class then, for each possible input datapoint, we need to calculate likelihood. Suppose there are m features & each feature is discrete and can take k different values. then total no. of possible data point = k^m eg: Binary feature ($x_n \in \{0,1\}$) & 10 features
 $\Rightarrow 2^{10}$ likelihood calculation per class for d classes this is $d \times 2^{10}$

In general for d # classes, m # features, k # discrete values a feature can take $\Rightarrow d k^m$ likelihoods has to be learned in training

prediction is $O(1)$ → constant time.

→ Some optimisation can be done.

$$\checkmark P(A|B) + P(A^c|B) = 1$$

$$\times P(A|B) + P(A|B^c) = 1 ?$$

⇒ out of k^d possibilities per class we can reduce 1

⇒ Total probabilities to be estimated → $d-1$ priors
 $= d-1 + d(k^m - 1)$ → $d(k^m - 1)$ likelihood per class
 $= d + dk^m - d - 1 = dk^m - 1$ → After optimisation.

→ Instead of this route, we can think about joint distribution $P(y, x_1, x_2, x_3, \dots, x_m) = \frac{1}{d k^m - 1}$ } Real algo would rather do this

Consider a Bayes classifier with 3 boolean input variables, X_1, X_2 , and X_3 , and one boolean output, Y .

How many parameters must be estimated to train such a Bayes classifier?

Here $m=3$
 $k=2$
 $d=2$
 $2 \times 2^3 = 16$ likelihood estimations
 2 priors
 with optimisation
 $= 2 \times 2^3 - 1 = 15$ estimations
 14 likelihood
 1 prior

Suppose that you are trying to solve a binary classification problem, and your data set has 4 attributes. Each attribute can take 3 possible values. If you modeled the full joint distribution of the attributes and the class label, how many parameters would you need?

$m=4$
 $k=2$
 $d=2$
 with optimisation
 #parameters to be calculated
 $= 2 \times 3^4 - 1 = 161$

MSQ Question: 1

What is the total number of parameters needed to model $P(Y, X_1, X_2, X_3, \dots, X_d)$?

- k is the number of possible values Y can take.
- t is the number of possible values for each X_i , where $i \in \{1, 2, \dots, d\}$.

✓ A. $k \cdot t^d - 1$

B. $k \cdot t^{d-1}$

✓ C. $k \cdot (t^d - 1) + (k - 1)$

D. $t^d \cdot k$

To handle this question, we have discussed optimisation

NAIVE BAYES

Naive Bayes assumption the features are conditionally independent given the class.

ie) $X_1/Y, X_2/Y, \dots, X_m/Y$ are independent

$$\Rightarrow P(X_1, X_2, X_3, \dots, X_m/Y) = P(X_1/Y) \cdot P(X_2/Y) \cdot \dots \cdot P(X_m/Y) = \prod_{i=1}^m P(X_i/Y)$$

Naïve Bayes classifier

$$\hat{y} = \operatorname{argmax}_y P(Y = y | X = x)$$

$$= \operatorname{argmax}_y P(Y = y) P(X = x | Y = y)$$

→ ignore denominator

$$= \operatorname{argmax}_y P(Y = y) \prod_{i=1}^d P(X_i = x_i | Y = y)$$

→ Training time calculation of probabilities is greatly reduced.

Bayes rule

+ Conditional independence assumption

→ Note if asked probability, then do not ignore denominator. Denominator cannot be split since condi. Independence $\nRightarrow$ independence

eg d

Suppose we are given the following dataset, where A, B, C are input binary random variables, and y is a binary output whose value we want to predict.

A	B	C	y
0	0	1	0
0	1	0	0
1	1	0	0
0	0	1	1
1	1	1	1
1	0	0	1
1	1	0	1

How would a naive Bayes classifier predict y given this input:

$A = 0, B = 0, C = 1$. Assume that in case of a tie the classifier always prefers to predict 0 for y .

$$\begin{array}{l|l}
 1) P(y=1) = 4/7 & P(y=0) = 3/7 \quad \text{prior} \\
 2) P(A=0|y=1) = 1/4 & P(A=0|y=0) = 2/3 \\
 3) P(B=0|y=1) = 2/4 & P(B=0|y=0) = 1/3 \\
 4) P(C=1|y=1) = 2/4 & P(C=1|y=0) = 1/3
 \end{array}$$

$$\Rightarrow \frac{16}{64 \times 7} = \frac{1}{28} \quad \frac{3 \times 2}{3 \times 3 \times 3 \times 7} = \frac{2}{63}$$

$$\frac{2}{63} < \frac{1}{28} \Rightarrow \text{prediction } y=1$$

0.0317 < 0.0357

$P(y|A=0, B=0, C=1)$
 We need to compare
 $P(y=1|A=0, B=0, C=1)$ with
 $P(y=0|A=0, B=0, C=1)$
 using bayes's theorem, & naive assumption.
 Compare
 1) $P(y=1) \cdot P(A=0|y=1) \cdot P(B=0|y=1) \cdot P(C=1|y=1)$
 2) $P(y=0) \cdot P(A=0|y=0) \cdot P(B=0|y=0) \cdot P(C=1|y=0)$

7

A_1	A_2	Class Label Y
True	True	+
True	True	+
True	False	-
False	False	+
False	True	-
False	True	-
False	False	-
True	False	+

$$\begin{array}{l|l}
 P(y=+) = 4/8 = 1/2 & P(A_1=T, A_2=T) = 2/8 = 1/4 \\
 P(y=-) = 1/2 & \\
 P(A_1=T|+) = 3/4 & P(A_1=T|-) = 1/4 \\
 P(A_2=T|+) = 2/4 = 1/2 & P(A_2=T|+) = 2/4 = 1/2
 \end{array}$$

$$\Rightarrow P(y=+|A_1=T, A_2=T) = \frac{P(y=+) \cdot P(A_1=T, A_2=T|y=+)}{P(A_1=T, A_2=T)}$$

What is the probability $P(Y = + | A_1 = \text{True}, A_2 = \text{True})$ by applying the Naïve Bayes classifier (show each step in computing the probability)?

What label should you predict for input ($A_1 = \text{True}, A_2 = \text{True}$) based on your result above?

$$\frac{P(y=+) \cdot P(A_1=T|+) \cdot P(A_2=T|+)}{P(A_1=T, A_2=T)} = \frac{\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{1}{2}}{1/4} = \frac{1}{4}$$

$$= \frac{P(y=+) \cdot P(A_1=T|+) \cdot P(A_2=T|+)}{P(A_1=T, A_2=T)} = \frac{\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{1}{2}}{\frac{1}{4}} = \frac{3}{4}$$

$\therefore$ prediction $y=+$

★

Consider a classification problem with 10 classes $y \in \{1, 2, \dots, 10\}$, and two binary features $x_1, x_2 \in \{0, 1\}$. Suppose $p(Y = y) = 1/10$, $p(x_1 = 1 | Y = y) = y/10$, $p(x_2 = 1 | Y = y) = y/540$. Which class will naive Bayes classifier produce on a test item with ($x_1 = 0, x_2 = 1$)?

(A) 1 (B) 3 (C) 5 (D) 8 (E) 10

$$P(x_1=1|y=y) = \frac{y}{10} \Rightarrow P(x_1=0|y=y) = \frac{10-y}{10}$$

$$P(x_2=1|y=y) = \frac{y}{540} \rightarrow \text{prediction } \underset{y}{\operatorname{argmax}} \left[\frac{10-y}{10} \cdot \frac{y}{540} \right] = \underset{y}{\operatorname{argmax}} [(10-y) \cdot y] = \underline{5}$$

We need, $P(y | x_1=0, x_2=1)$
 $= P(y) \cdot P(x_1=0|y) \cdot P(x_2=1|y)$
 compare for all y
 then $P(y) = \frac{1}{10} \forall y \rightarrow \text{ignore}$

$$P(y/x_1=1, x_2=0) = P(y) \cdot P(x_1=1/y) \cdot P(x_2=0/y)$$

$$= \frac{1}{32} \cdot \frac{y}{46} \cdot \frac{62-y}{62}$$

$$\arg \min_y [(62-y) \cdot y] = \underline{\underline{31}}$$

- $$P(y/x_1, x_2, x_3) = P(y) \cdot P(x_1/y) \cdot P(x_2/y) \cdot P(x_3/y)$$
- $\downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow \qquad \qquad \downarrow$
 $2^{-1} + 2 \times (2^{-1}) \qquad 2 \times (2^{-1}) \qquad 2 \times (2^{-1})$
- In general $(d-1) + \underbrace{d(k-1) + \dots + d(k-1)}_{m \text{ times}}$

$$= 1 + 2 + 2 + 2 = 7 //$$
$$\begin{aligned} &= d-1 + md(k-1) \\ &= d-1 + mdk - md \\ &= dkm - dm + d - 1 \end{aligned}$$

Here $= 2 \cdot 2 \cdot 3 - 2 \cdot 3 + 2 - 1$

$$= 12 - 6 + 1$$

$$= 7 //$$

Optimal Bayes = $d k^{m-1}$
Naive Bayes = $d k^m - d m + d - 1$

$$(d-1) + m d \cdot (k-1) = 1 + n \cdot 2 \cdot 1 = \underline{\underline{2n+1}}$$
$$(k-1) + k \cdot d(v-1) = k-1 + kd v - kd$$

$$\Rightarrow kd v - k(d-1)$$

$C \rightarrow$ closest match

ignoring -1
 $k + kd v$

H.W.

$$P(y=1|X) = \frac{P(y=1) \cdot P(x_1=1/y=1) \cdot P(x_2=0/y=1) \cdot \dots \cdot P(x_{100}=0/y=1)}{P(y=1) \cdot P(x_1=1/y=1) \cdot \dots \cdot P(x_{100}=0/y=1) + P(y=0) \cdot P(x_1=1/y=0) \cdot \dots \cdot P(x_{100}=0/y=0)}$$
$$\begin{array}{l} p(x_i=1|y=0)=a \\ p(x_i=1|y=1)=b \end{array} \Rightarrow \begin{array}{l} p(x_i=0|y=0)=1-a \\ p(x_i=0|y=1)=1-b \end{array}$$

$$= \frac{0.5 \cdot b^{50} (1-b)^{50}}{0.5 \cdot b^{50} (1-b)^{50} + 0.5 \cdot a^{50} (1-a)^{50}} = \frac{b^{50} (1-b)^{50}}{b^{50} (1-b)^{50} + a^{50} (1-a)^{50}}$$