9/6/22

Machine Learning Coursera

Week-2

Multiple features.

	Size (feet^2)	No. of bedrooms	No. of floors	Age of home	Price
1	a	2	1	10	100
2	b	3	3	15	200
3	c	4	2	20	300

$x_1^3 \Rightarrow$ size (feet) $\rightarrow 1$

$\Rightarrow$ c $\rightarrow 3$

$\therefore$ size in feet^2 of home 2

Cost function -

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x_2 \cdots \theta_n x_n$

$x_0 = 1$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

$\in \mathbb{R}^{n+1}$

$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 \cdots + \theta_n x_n$

Hypothesis function ↑

$\theta^T \times X = [n+1] \times 1$

Rows Cols

$\theta^T = [\theta_0 \quad \theta_1 \quad \theta_2 \quad \dots \quad \theta_n]$

1 - row, $n+1$ - cols.

$\theta^T \times X = \underset{\text{Rows}}{1} \times \underset{\text{Cols.}}{(n+1)}$

Vector can be defined as a 1-D data structure.

Eg. Housing dataset with an array containing info on →

a) No of bedrooms

b) sqft of the house

Vectors can be represented as a row as well as a column vector.

A vector can be a matrix but not vice versa.

Hypothesis ⇒

$h_\theta(x) = \theta^T x =$

$$\theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n$$

Parameters : $\theta_0, \theta_1, \cdots \theta_n \quad \simeq \theta$

Cost function —

$$J(\theta_0, \theta_1 \cdots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2$$

$J(\theta) \Rightarrow$ same.

Gradient Descent — [For $n = 1$]

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \cdots \theta_n)$$

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} [J(\theta)]$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right)^2$$

$$\theta_0 - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad \text{—— (i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - y^i\right) x^i \quad \text{—— (ii)}$$

New algorithm for $(n \geq 1)$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} h_\theta(x^i - y^i) x_j \quad \text{(i)}$$

θ_0 and θ_1 will remain same as for $n = 1$.

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left[h_\theta(x^i) - y^i \right) x_2^i \Big]$$

for second feature

10/6/22

Feature Scaling

x_1 = size $(0 - 2000$ sq ft$)$

x_2 = no. of bedrooms $(1 - 5)$

Scaling → $x_1 = \frac{\text{size}}{2000}$

$$x_2 = \frac{\text{no. of bedrooms}}{5}$$

$\therefore 0 \leq x_1 \leq 1 \quad , \quad 0 \leq x_2 \leq 1$

Goal of feature scaling is to bring every parameter with a range of $-1 \leq x_i \leq 1$

Mean normalization. —

Replace x_i with $x_i - \mu_i$

$$x_1 = \frac{\text{size} - 1000}{2000}$$

$$x_2 = \frac{\#\text{bedrooms} - 2}{5}$$

$\therefore -0.5 \leq x_1 \leq 0.5, \quad -0.5 \leq x_2 \leq 0.5$

Example →

$$x_1 \rightarrow \frac{x_1 - \mu_1}{s_1}$$

μ_1 = mean

s_1 = range.

$a = 4(3-5) + 2$

$= 4(-2) + 2$

$= -8 + 2 = -6$

$b = -4(5-3)$

$= -4(+2)$

$= -8$

$2b = -16$

$|2b| = +16$

20 + 40

$2|a| - |2b|$

$\Rightarrow 2 \times 6 - |-16|$

$\Rightarrow 12 - 16$

$\Rightarrow -4$

3T: 20

$1 < 2a < 18$

$0 < 2a < 17$

$0 < a < 8.5$

3/5

q

$x + y = 150$

$(x - 70) + 2y - (x + y) = 20$

$(x - 70) + 2y - 150 = 20$

$(x - 70) + 2y = 170$

$x + 2y = 240$

$x + y + y = 240$

$y = 240 - 150$

$= 90$

$x = 60$

$(x + y) - [(x - 70) + 2y] = 20$

$150 - 20 = (x - 70) + 2y$

$130 = x + y + y + 70$

$200 = 150 + y$

$y = 50$

$x = 100$

$y' = 100$

$x' = 30$

---

q

$\frac{14}{x} \quad \frac{14}{x - y}$

$\frac{14}{x - y} - \frac{14}{x}$

$\frac{14x - 14(x - y)}{x(x - y)}$

$\frac{14y}{x^2 - xy}$

$t = 5$

$u = -2$

$7, \quad -7$

$\sqrt{2} + \frac{2}{7} \; y = \frac{1}{2}$

$\frac{2}{7} + \sqrt{3} - 0.3$

$0.22 + 1.732 - 0.3$

$1.9 - 0.3$

$\Rightarrow 1.1$

$1.414 - 0.5 +$

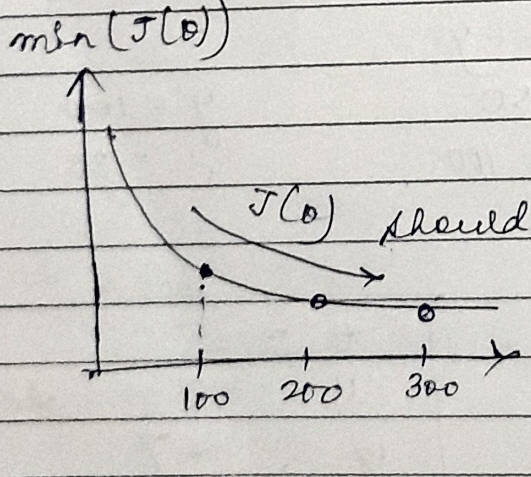
$100 \rightarrow 70 \rightarrow 56$

M T

$a\left[\frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5}\right] \quad a\left[\frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6}\right]$

22

Learning Rate



After 300th iteration we see that Gradient Descent has converged as there is no further reduction in the cost function.

Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

Acceptable learning rates →

α → 0.001, 0.01, 0.1, 1 ...

Normal Equation

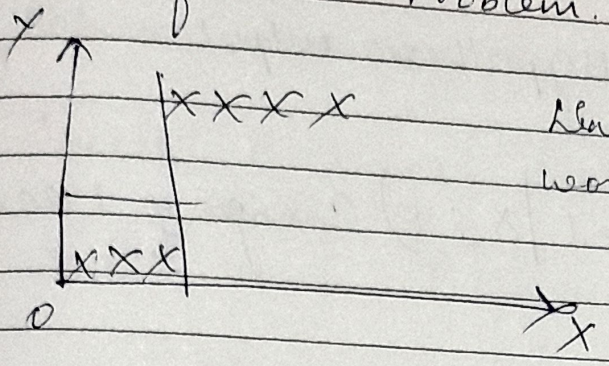
$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_n \end{bmatrix} \in R^{n+1} \qquad \theta = (X^T X)^{-1} X^T y$$

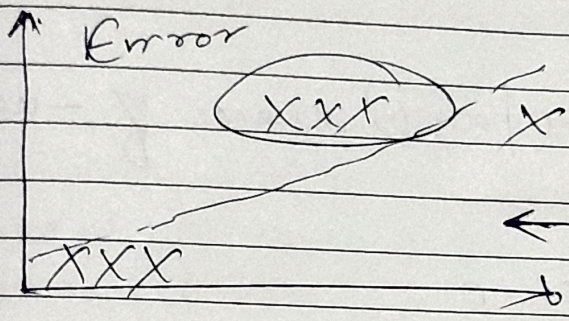
X design matrix $= \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}$

X_1	X_2	Price
4	89	100
5	81	150
6	60	200

$$X = \begin{bmatrix} 4 & 89 & 1 \\ 5 & 81 & 1 \\ 6 & 60 & 1 \end{bmatrix} \qquad Y = \begin{bmatrix} 100 \\ 150 \\ 200 \end{bmatrix}$$

Classification Problem.





Use a classifier for a classification problem.

Logistic Regression is a classification algorithm.

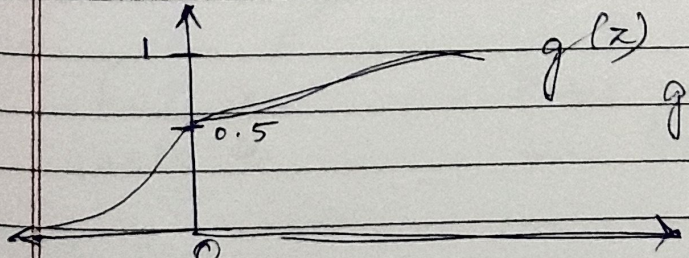
Logistic regression Model :-

$0 \leq h_\theta(x) \leq 1$

Sigmoid function.

$h_\theta(x) = g(\theta^T x)$ $\quad z = 1/1+e^{-z}$

$g(z) = \frac{1}{1+e^{-z}}$ (Sigmoid function



g(z) approaches 1 as it tends to $+\infty$ and 0 as it approaches to $-\infty$

14/6/22

Interpretation of Hypothesis output.

$h_\theta(x) = P(y=1 | x; \theta)$ Chance of +ve.

$y = 0$ or 1

$h_\theta(x) = P(y=0 | x; \theta)$ chance of -ve

Q $\theta_0 = 5, \theta_1 = -1, \theta_2 = 0$

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

$= 5 - x_1 + 0$

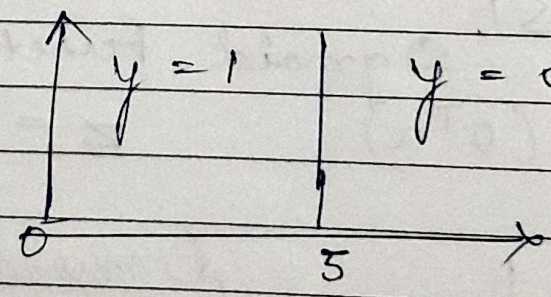
$= 5 - x_1$

$5 - x_1 > 0$

$-x_1 > -5$

$x_1 < 5$ $h_\theta(x)$ stays greater than 0

$\therefore$



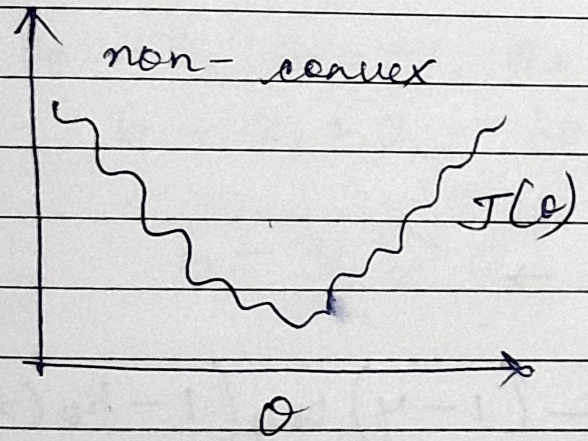
equation of a circle.

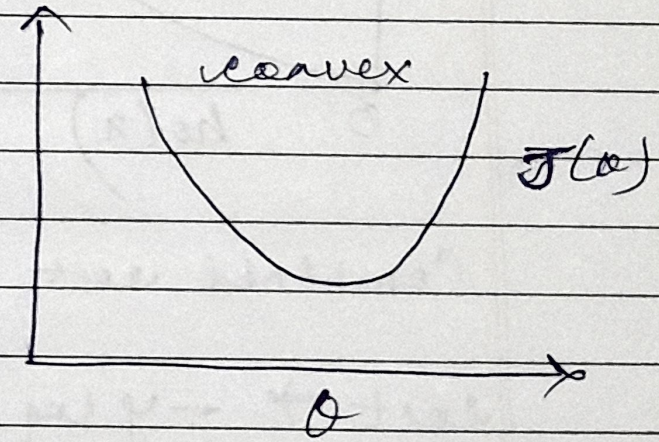
7/22

Training set = $\{(x^1, y^1), (x^2, y^2), (x^m, y^m)\}$

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$



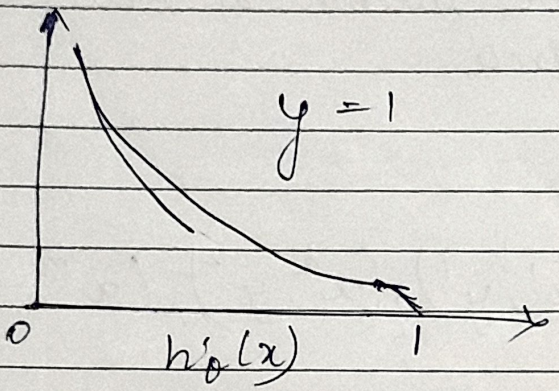


Logistic regression cost function –

$$Cost[h_\theta(x), y] \Rightarrow$$

$$\begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

a)



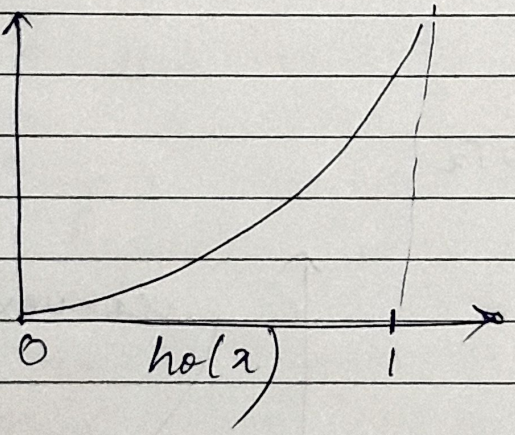
If $y = 1$,

$h_\theta(x) = 1 \rightarrow Cost = 0$

But as $h_\theta(x) \rightarrow 0$

$Cost \rightarrow \infty$

b) $y = 0$



Combined cost function →

$$cost \rightarrow -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

$$J(\theta) = \frac{-1}{m} \left[\sum_{i=1}^{m} y^i \log(h_\theta(x)) + (1-y^i) \log(1 - h_\theta(x)) \right]$$

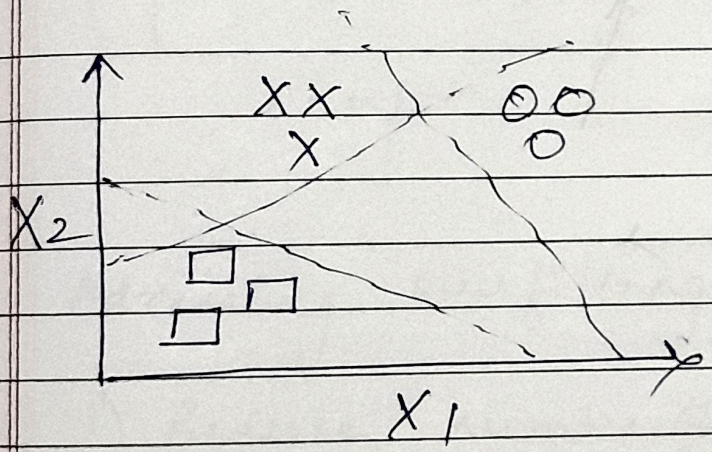
Multi class classification problem -

Email : Work, Friends, Family, Hobby

$y=1$ $y=2$ $y=3$ $y=4$

Weather ⇒ Sunny, Cloudy, Rain, Snow

$y=1$ $y=2$ $y=3$ $y=4$



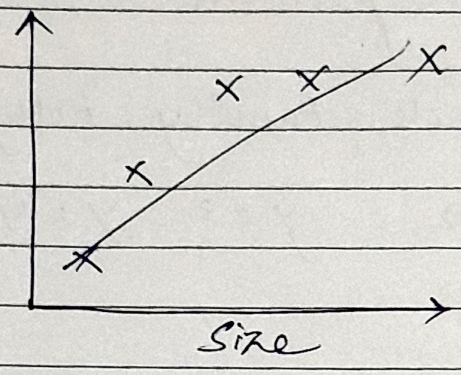
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$\theta_0 = 6, \quad \theta_1 = -1, \quad \theta_2 = 0$

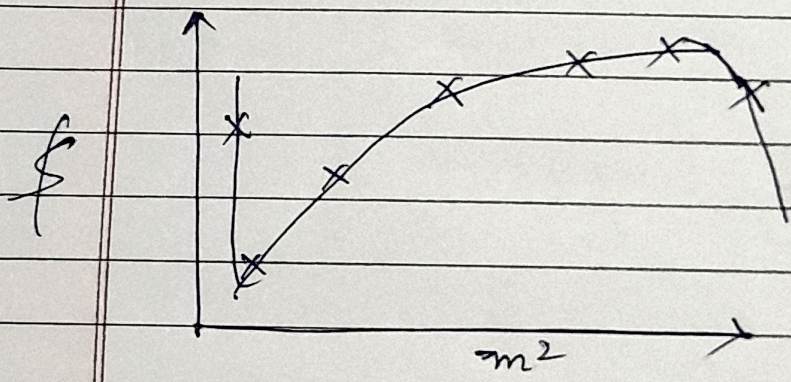
$6 - x_1 + 0 = h_\theta(x)$

$6 - x_1 \geq 0$

Overfit and Underfit. (Linear Regression)



Underfit
→ High bias
→ low variance.

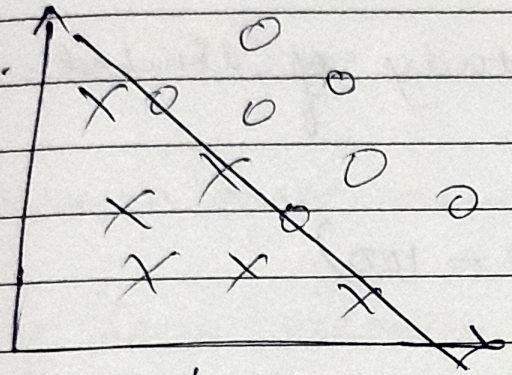


→ Overfit
→ High Variance
→ Low Bias

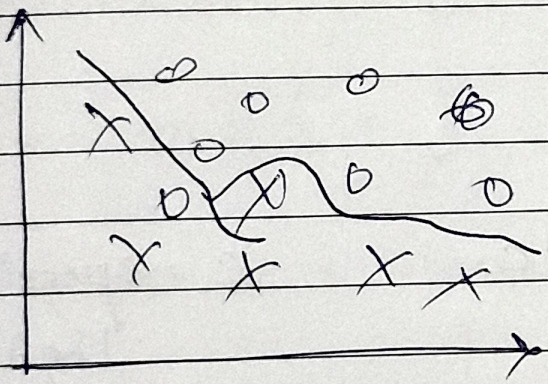
Overfit -

When we have too many features, the learned hypothesis may fit the training set very well but fail to generalize new examples

Logistic Regression.



Underfit



Overfit.

Addressing overfitting —

1) Reduce number of features

2) Regularization. (Reduce magnitudes/value of parameters).

15/7/22

Regularization ML Courseera.

$$1 - \alpha\frac{\lambda}{m} < 1$$

Normal Equation –

$$X = \begin{bmatrix} x^{(1)T} \\ \\ x^{(m)T} \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ \\ y^{(m)} \end{bmatrix}$$

$$\min(J(\theta))$$

$$\theta = \left(X^TX + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}\right)^{-1} X^Ty$$

$(n+1) \times (n+1)$ dimensional matrix.

Logistic regression maybe prone to overfitting if we use a higher order polynomial.

Thus we can regularize the cost function of logistic regression →

by adding, $\frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^{(i)} \cdot \log\left(h_\theta(x^i)\right) - (1-y^i) \log\left(1-h_\theta(x^i)\right) \right]$$

$$\text{grad} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^i) - (y^i)\right) x_j^i$$

Regularized cost function →

$$\frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 + J(\theta)$$

17/7/22

Week - 5

L = Total number of layers in the network.

S_L = no. of units in layer L

Classification

Binary

$y = 0$ or 1

$S_L = 1$

Multiclass

$y \in R^K$

pedestrian, car, motor

K output units

$S_L = K$

Cost function:

Logistic Regression:

$$J(\theta) = \frac{-1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^i) + (1-y^i) \log (1-h_\theta(x^i))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Backpropagation →

$$J(\theta) \quad -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{k} y_k \log h_\theta(x^i)_k + (1-y_k)\log\left(1-h_\theta(x)_k\right)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\theta_j^{(l)}\right)^2$$

$\min J(\theta)$

Need to compute :

1) $J(\theta)$

2) $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$

Gradient Computation!

$a^{(1)} = x$

$z^2 = \theta^{(1)} a^{(1)}$

$a^2 = g(z^2)$ — ①

$z^3 = \theta^{(2)} \cdot a^2$

$a^3 = g(z^3)$ — ②

$z^4 = \theta^3 \cdot a^3$

$a^4 = g(z^4)$ — ③

$a^4 = h_\theta(x)$

Intuition of back propagation algorithm is that for each node we are going to compute the term $\delta_j^{(l)}$

δ_j = error of node j in layer l

for each output unit $\rightarrow$ $L = 4$

$$\delta_j^4 = a_j^4 - y_j$$

y_j = ground truth

a_j^4 = value of the hypothesis we got.

$$\delta^3 = (\theta^3)^T . \delta^4 . * g'(z^3)$$

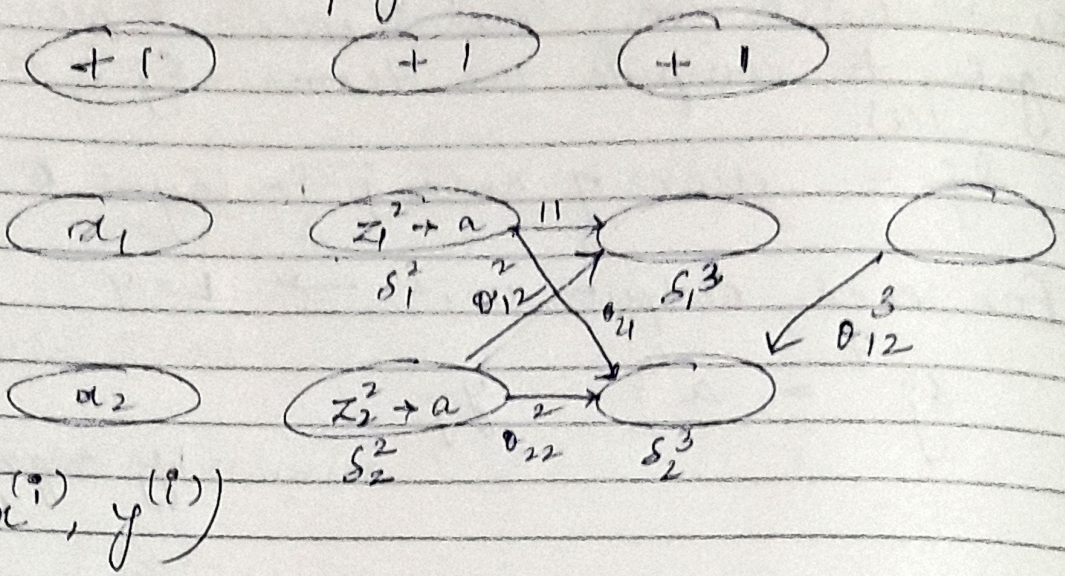
$$\delta^2 = (\theta^2)^T . \delta^{(3)} . * g'(z^2)$$

$$\Delta_{i,j} := \Delta_{ij} + a_j^{(l)} \delta_i^{(l+1)}$$

Suppose you have 2 training examples (x^1, y^1) and (x^2, y^2)
What is the correct sequence for computing the gradient.

Ans FP of x^1, followed by BP of y^1 then.
FP of x^2, followed by BP of y^2

Forward Propagation



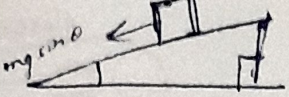
After we apply activation function to z_1 we get the activation values.

Activation function decides whether a neuron should be activated or not by calculating weighted sum and further adding bias with it.

Bias is a sort of a constant which is used to adjust the output along with the weighted sum.

In a NN, we update the weight and biases of the neuron based on the error at the output.

- Linear
- Sigmoid — Binary
- Tangent Hyperbolic - tanh
- ReLU
- Softmax - Multiclass



classmate
Date
Page

$\delta_1^4 \Rightarrow y - a_4$ (i) (4)

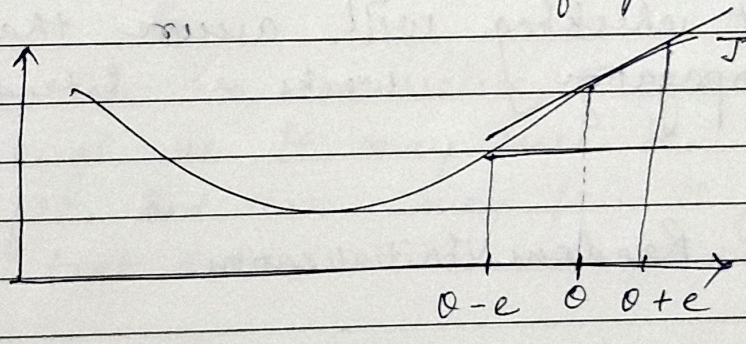
$\delta_2^2 = \theta_{12}^2 \delta_1^3 + \theta_{22}^2 \delta_2^3$

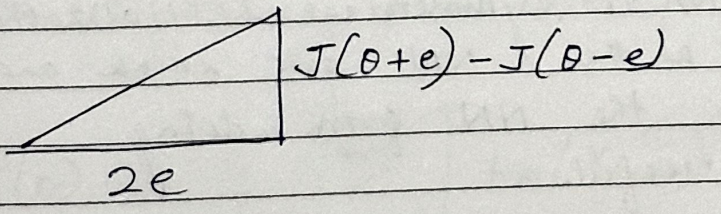
$= \theta^2$

$\delta_2^3 = \theta_{12}^3 \times \delta_1^4$

$\delta_1^3 \Rightarrow \theta_{12}^3 \times \delta_1^4$

18/7/22 Numesical estimate of gradients.





$$\frac{d}{d\theta} J(\theta) \simeq \frac{J(\theta+e) - J(\theta-e)}{2e} \qquad e = 10e^{-4}$$

$$\text{gradapprox} \approx \frac{\left[J(\text{theta}+e) - J(\text{theta}-e)\right]}{2*e}$$

Q $J(\theta) = \theta^3$. $\theta = 1$, $e = 0.01$

$$\frac{J(\theta+e) - J(\theta-e)}{2e} = \frac{dJ(\theta)}{d\theta}$$

$$\frac{(1+0.01)^3 - (1-0.01)^3}{2 \times 0.01} \approx 3$$

$$\Rightarrow 3.0001 \approx 3$$

Gradient checking will assure that backpropagation works as intended.

Random Initialization

Zero initialization or symmetrical initialization isn't helpful as the NN gets stuck and it prevents the NN from doing something useful.

So we go for Random Initialization. —

We initialize $(\theta i, j)$ to a random value in $[-\epsilon, \epsilon]$

Q, This is not same as the one from gradient checking.

Training a neural network -

a) No of input units : Dimensions of features

b) No of output units : Number of classes.

c) Reasonable Default → 1 Hidden unit.

if > 1, then have the same no of hidden units in every layer (more the better).

---

q $J(\theta) = 3\theta^4 + 4 \qquad \theta = 1, \epsilon = 0.01$

$$\frac{J(\theta+\epsilon) - J(\theta-\epsilon)}{2\epsilon}$$

- Just a low value of error is not enough as it may even be a case of overfit. and thus may fail to generalize the test data.

Training/Testing procedure for linear regression.

$$J(\theta)_{test} = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left[h_\theta(X_{test}^{(i)}) - y_{test}^{(i)} \right]^2$$

Avg squared error.

set

Why we need a val ~~set~~ →

A validation set is a set of data used to train AI with the goal of finding and optimizing the best model to solve a given ~~prob~~ problem.

It acts as an intermediate phase used for choosing the best model and optimizing it.

Overfitting is checked and avoided due to this.

$J_{CV}(\theta)$ is lower than $J_{test}(\theta)$ because an extra parameter (d) degree of polynomial had been fit to the cross validation set.

$y = e^{4-3x}$

$u = 4 - 3x$

$y = e^{u}$

$\frac{du}{dx} = -3$

$\frac{dy}{du} = e^{u}$

$\frac{dy}{du} \cdot \frac{du}{dx} = \frac{dy}{dx}$

$R = 350, \quad V = 3V, \quad \frac{dV}{dt} = 1.8 V/min$

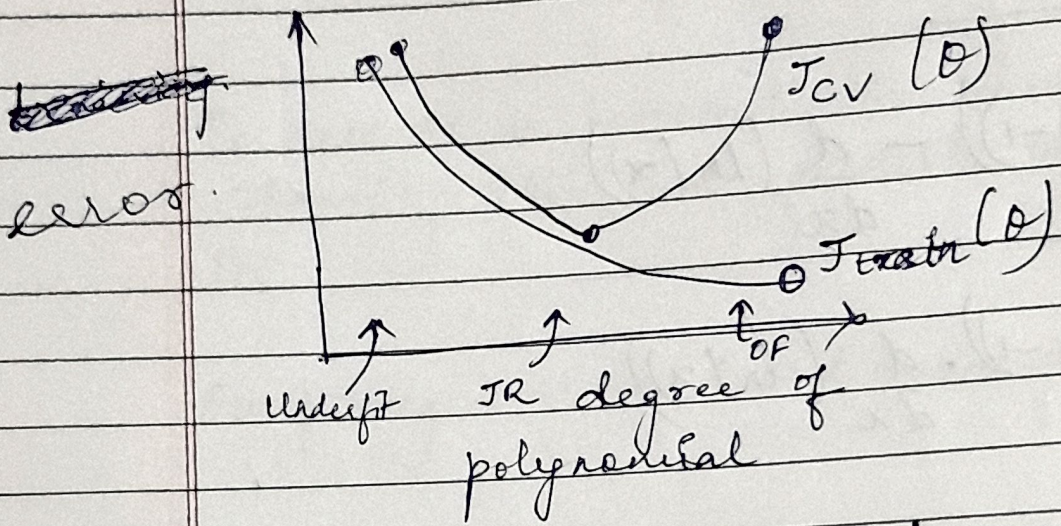
$\frac{dR}{dt} = 1.7 \Omega/min$

$V = IR.$

$\frac{dV}{dt} = \frac{dI}{dt} \cdot \frac{dR}{dt}$

High Bias – Underfit

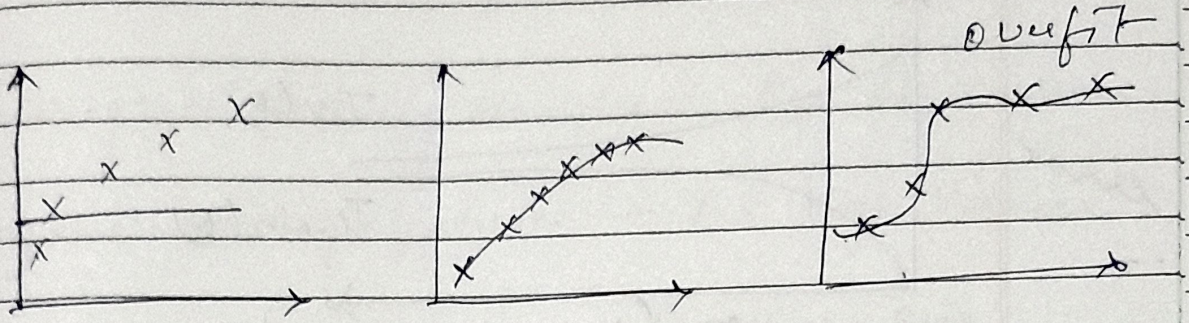
High Variance – Overfit.



High Bias –	High Variance
1) High training error	1) Low training error
2) High cross val error	2) High cross val error.

4/7 22

Linear regression with regularizat

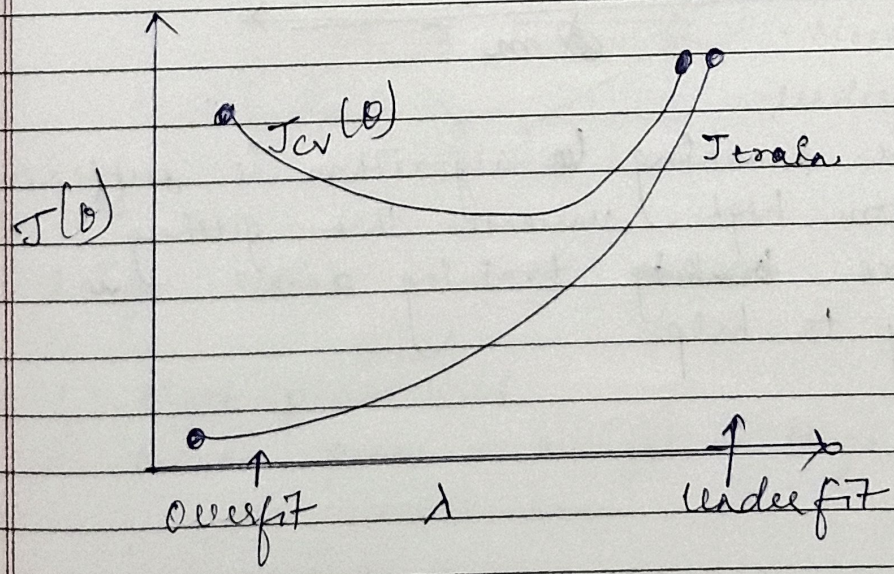


Underfit Just Right $\lambda = 0$

$\lambda = 10000$ $\lambda = 500$

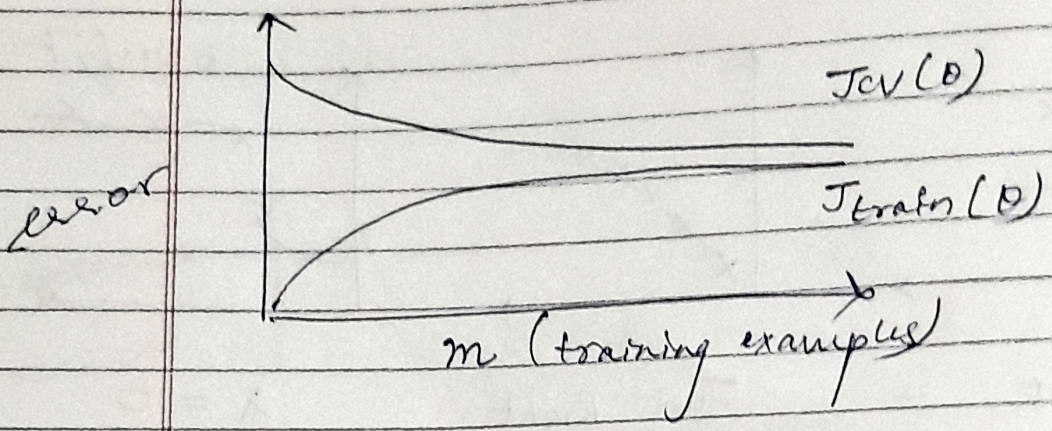
The way we find the value of λ is same as how we find the value of 'd' or degree of polynomial in a NN.

We test how well the model perfo across various values of lambda



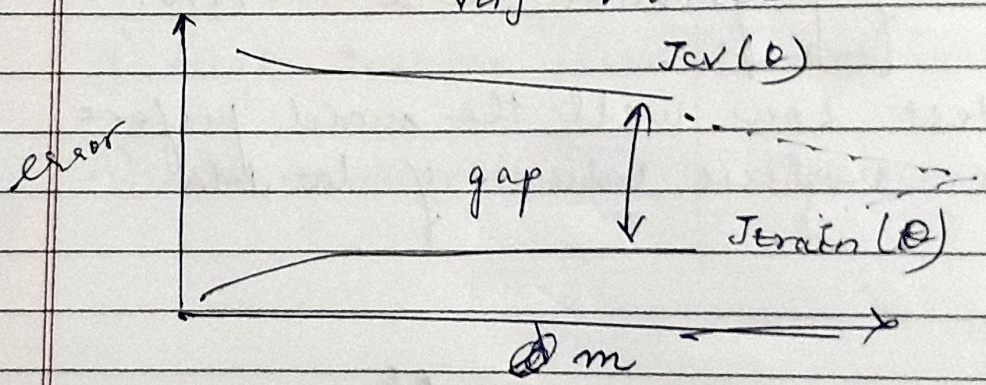
Learning Curve

Case →
High Bias



If a learning algorithm is suffering from high bias, getting more training data will not help much.

High Variance



If the learning algorithm is suffering from high variance then getting more training data is likely to help

Debugging learning algo —

Getting more training examples — Fix H. Variance

Try smaller set of features — Fix High Variance

Try getting additional features — Fix High Bias

Try adding polynomial features — Fix High Bias.

Decreasing λ — Fix High Bias

Increasing λ — Fix High Variance

Q → High error rate
↳ High variance, overfitting
↳ Increase

Precision / Recall.

Predicted \ Actual class	1	0
1	TP	FP
0	FN	TN

Precision → $\frac{TP}{TP+FP}$ How many had $y=1$

Recall → $\frac{TP}{TP+FN}$ How many did we correctly classify.

Q

	1	0
1	TP 80	FP 20
0	FN 80	TN 820

$P \rightarrow \frac{80}{80+20} = 0.8$

$R \rightarrow \frac{80}{80+80} = 0.5$

	TP	FP	FN	
0.8	30	2	3	$P = \frac{30}{32} = 93$, $R = 90$
0.3	30	10	2	$P = 75$, $R = 93$

$$\text{precision} = \frac{\text{true positives}}{\text{no. of predicted positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{no. of actual positives}}$$

Trading off precision and recall -

If we wish to increase the ~~thres~~ threshold of confidence from 0.5 → 0.9.

ie we are 90 % confident that someone has cancer then we ↑ precision but recall ↓

$y = 1$

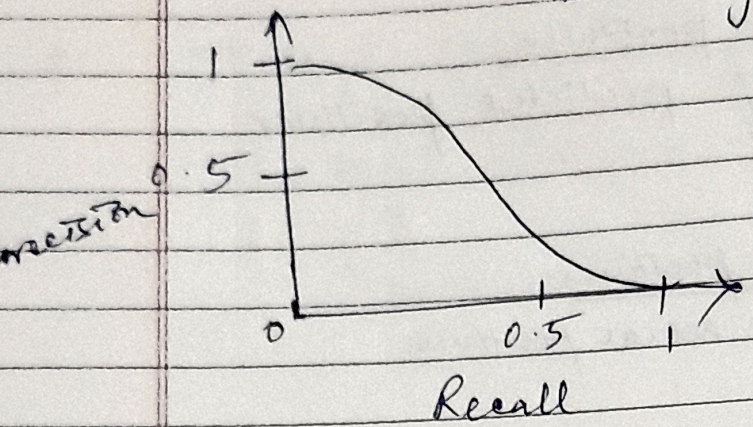
High Precision, Low recall

case 2 To avoid false negatives we may set the threshold to 0.3

Lower Precision, High Recall.

	T	TP	FP	FN	
I	0.8	30	2	3	$P = 93$, $R = 90$
II	0.3	30	10	2	$P = 75$, $R = 93$

$$\text{Accuracy} = \frac{TP + TN}{\text{Total Examples}}$$



F1 score →

	P	R
Algo 1	0.5	0.4
Algo 2	0.7	0.1
Algo 3	0.02	1.0

$$F_1 = \frac{PR \times 2}{P + R}$$

Large data rationale

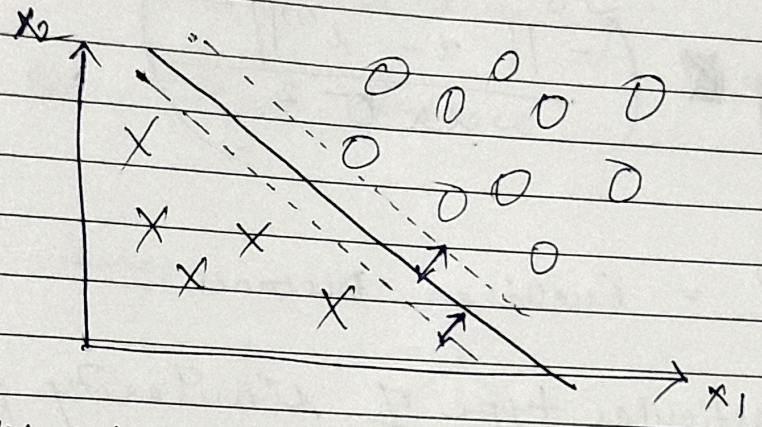
- Use a learning algorithm with many parameters.
  → Low bias
- Use a very large training set → unlikely to overfit

Q P → $\frac{85}{85+890}$

→ 0.087

22/7/22

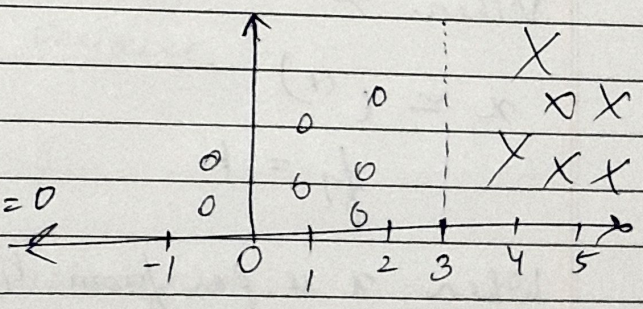
SVM



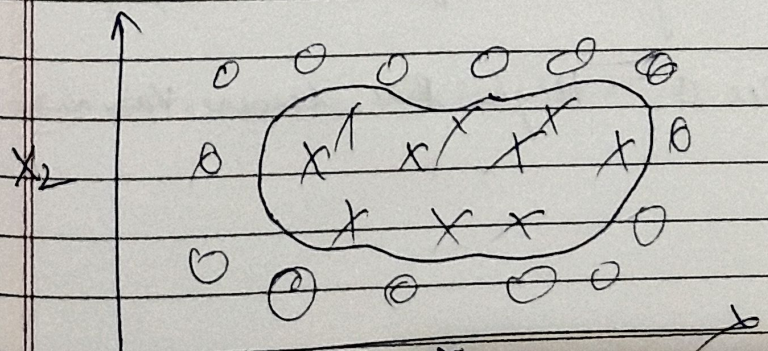
SVM tries to get the largest minimum distance between the two classes.

Q $\theta_0 + \theta_1 x_1 + \theta_2 x_2$

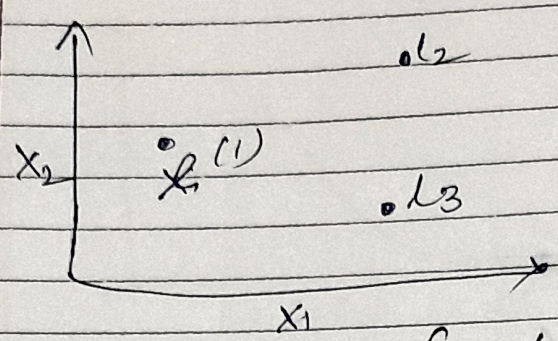
(a) $\theta = -3, \theta_1 = 1, \theta_2 = 0$



Kernels are used to adapt SVM in order to develop complex non linear classifiers.



l_1, l_2, l_3 are landmarks.



$f_1 = \text{similarity}(x, l^{(1)}) \rightarrow$ Kernel

$$\exp\left(\frac{-\|x - l^{(1)}\|^2}{2\times\sigma^2}\right)$$

$x - l^{(1)}$ = Euclidean Distance.

This particular type of similarity function is known as Gaussian kernel.

When →

$x \simeq l^{(1)}$

$f_1 = 1$

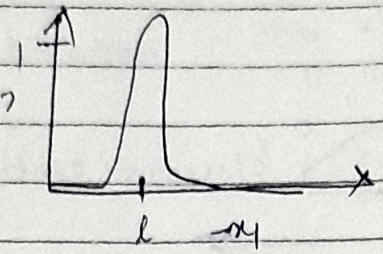
When x is far from l

$f_1 = 0$

$C = \frac{1}{\lambda}$ Larger C: High Variance, Low Bias

Smaller C: High Bias, Lower Variance

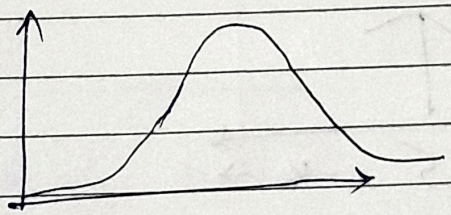
a) σ^2 = Smaller



Features vary less smoothly.

Low Bias, High Variance

b) σ^2 = Larger



Features vary smoothly,
High Bias, Low Variance,

Week - 8

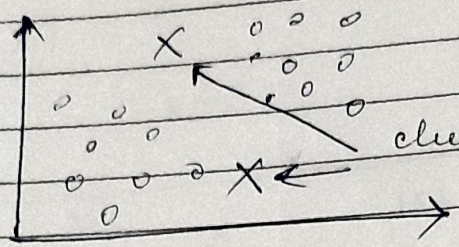
Applications of clustering →

a) Market Segmentation

b) Social network analysis.

c) Astronomical data analysis.

K-means is the most popular clustering algorithm.

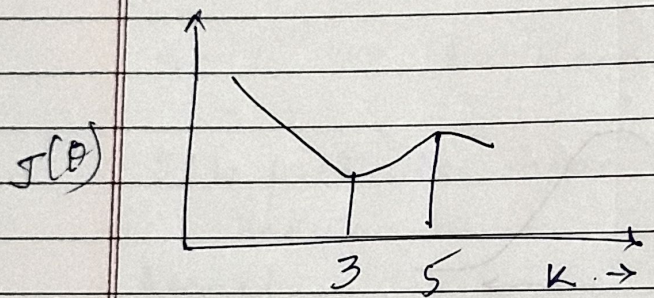


Steps →

a) Cluster Assignment

b) Move centroid step.

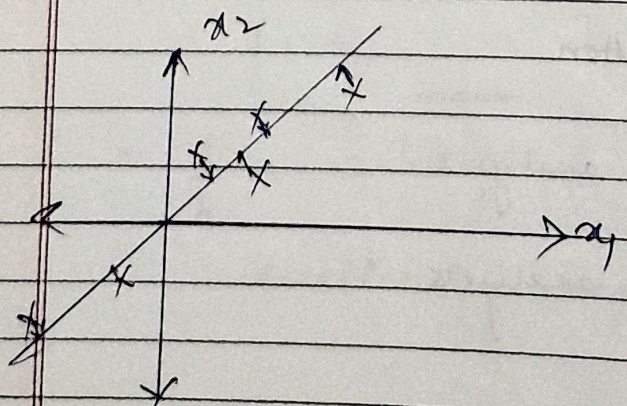
Loop



PCA

Principal Component Analysis.

Reduce from 2D to 1D : Find a direction onto which to project the data so as to minimize the projection error.



1, 2, 3, 4, 5 → Mean → 3.

Mean Normalization $\Rightarrow \frac{1}{m} \sum x_j - \mu_j$

$\Rightarrow \frac{1}{5}\left[(-2) + (-1) + 0 + 1 + 2\right] = 0$

∴ New mean = 0.

Feature scaling →

$x_j \Rightarrow \frac{x_j - \mu_j}{s_i}$ s_i = range

μ_j = mean.

Anomaly detection

1) Credit Card Fraud Detection.

2) Manufacturing. $p_x < \epsilon$

3) Data centres. Every time p_x goes ↓ ϵ, we flag it as an anomaly.

On a cross val/ test set →

$$y = \begin{cases} 1, & p(x) < \epsilon \text{ (anomaly)} \\ 0, & p(x) \geq \epsilon \text{ (normal)} \end{cases}$$

Use → F_1, P/R, as metrics.

Anomaly vs Supervised.

Very small number of positive examples (0-20)	Large no of positive and negative examples.
Large number of negative examples	Here we know what the positive examples maybe like tomorrow.
We dont know what they will be like to morrow.	
Ie for an aircraft engine a lot of things may go wrong.	

Q user = j | rated movie = i

$\therefore y^{(2,1)} = y^{(i,j)}$

$i = 2, \; j = 1$

r^{21} User 2, Movie 1

$r(i,j)$

Q $r(i,j) \Rightarrow$ i = rated movie

j = user

24/7/22

Stochastic gradient decent will allow us to scale GD to a much larger training set.

SGD

- Randomly shuffle the dataset
- Repeat for $i = 1 \dots m$.

$$cost(\theta, x^i, y^i) = \frac{1}{2}\left(h_\theta(x^i) - y^i\right)^2$$

$$J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^{m} cost\left(\theta, (x^i, y^i)\right)$$

Mini Batch Gradient Descent - Take a batch size of b ~ 10, and perform an iteration.

Photo OCR Pipeline

1. Text Detection
2. Character Segmentation.
3. Character classification.

Image → TD → CS → CC

Expansion operator : Brighter nearby pixels.

200 × 200 / 4 × 4 → 50 × 50 = 2500

10 secs to get 1 ex

$$ITE = \frac{10}{60 \times 60}$$

$$= 2.77 \times 10^{-3} \text{ hrs to get 1}$$

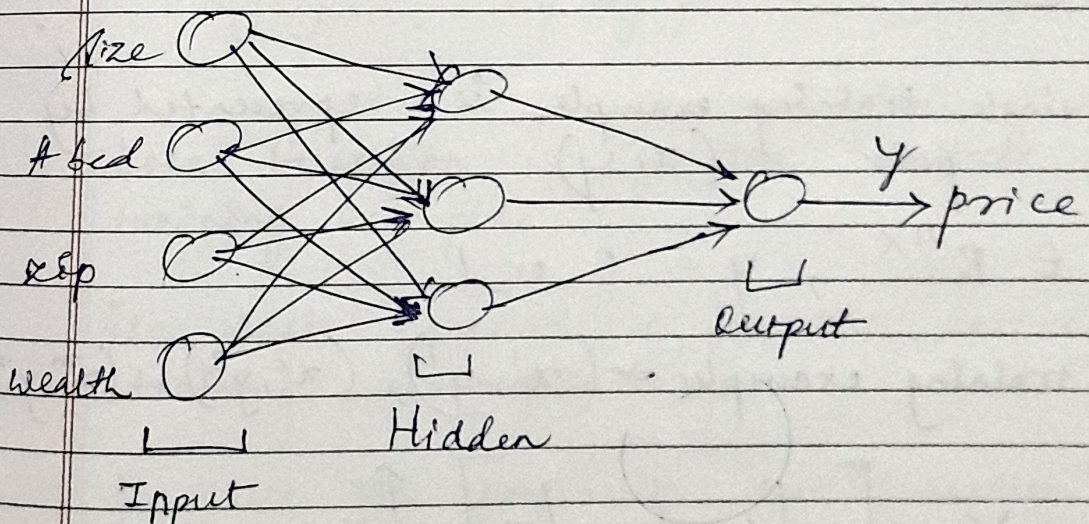
$10/hour

4 ex/min ⇒ 240 ex/hour = $10

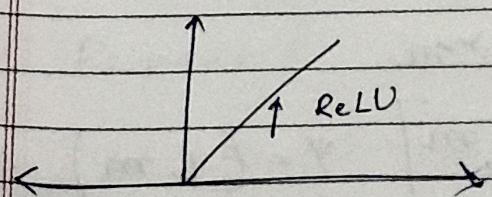
$$\frac{1000 \times 1000}{2 \times 2}$$

$$\$ \frac{500 \times 500}{}$$

Neural Network and DL



ReLU



The ReLU activation function allows better computation by allowing gradient descent to be equal to 1 and converge faster when compared to sigmoid.

Logistic regression is an algo for binary classification

RGB →
say we have an image of 64×64.

∴ R = 64 × 64
B = 64 × 64
G = 64 × 64

$$X = \begin{bmatrix} 255 \\ 231 \\ | \\ 255 \\ \vdots \\ 255 \\ 231 \end{bmatrix}$$

$X \rightarrow 3 \times 64 \times 64$
$n_x \rightarrow 12288$

dimension of this input feature vector.

A single training example is represented by a pair (x, y)

$x \in R^{n_x}$, $y = 0$ or 1

m training examples → $(x^1, y^1), (x^2, y^2) \dots (x^m, y^m)$

$$X = \begin{bmatrix} | & | & | & | \\ x^1 & x^2 & \dots & x^m \\ | & | & | & | \end{bmatrix} \quad n_x \text{ (rows)}, \quad m \text{ (columns)}$$

$$Y = \begin{bmatrix} y^{(1)}, y^2 \dots y^m \end{bmatrix} \quad Y = (1, m)$$

Output $\hat{y} = \sigma(w^T x + b)$

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

w is an n_x dimensional vector

b is a real number.

Loss Function →

$$L(\hat{y}, y) \Rightarrow \frac{1}{2}[\hat{y} - y]^2$$ Squared error loss Function.

Loss Func of Log Reg →

$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$$

Loss Function is for ~~the~~ ~~entire~~ Individual set of training examples.

The cost function is for the entire set of training examples

We use the particular cost function for logistic regression since we want a convex function.

Repeat {

$w := w - \alpha \frac{dJ(w)}{dw}$

}

α = learning rate.

Computing Derivatives

Chain Rule →

$$\frac{dJ}{da} = \frac{dJ}{dv} \cdot \frac{dv}{da}$$

$a \rightarrow v \rightarrow J$

Change in J due to change in a = Change in J due to change in v * Change in v due to change in a.

$$\frac{dJ}{db} = \frac{dJ}{du} \cdot \frac{du}{db}$$

$b = 3 \qquad u = bc$

$c = 2 \qquad u = 6$

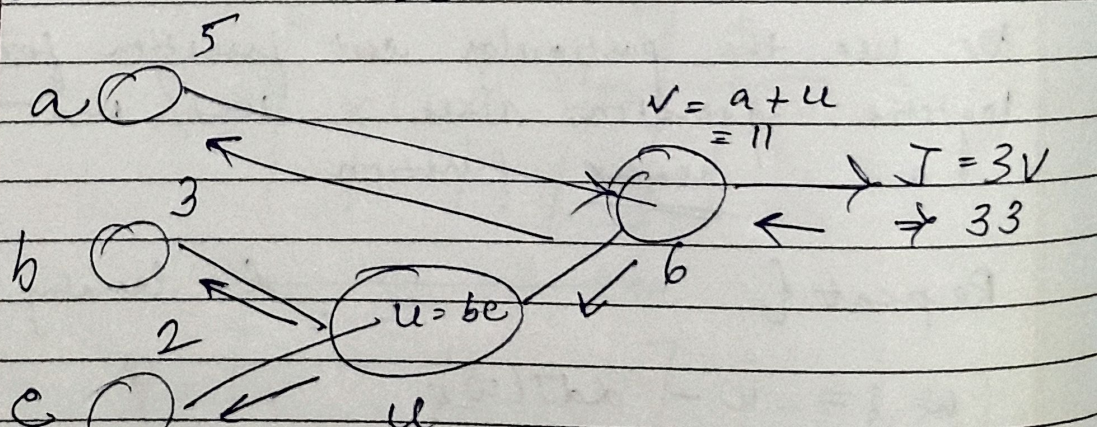
$b = 3.001 \qquad u = 6.002$

$c = 2$

$\Delta u = 0.002$

$$\frac{du}{db} = 2$$

$$\frac{dJ}{du} = 3. \qquad \frac{dJ}{db} = 3 \times 2 = 6.$$



x_1
w_1
x_2 — $z = w_1 x_1 + w_2 x_2 + b$ → $a = \sigma(z)$ → $L(a, y)$
w_2
b

$dz = \frac{dL}{dz}$ $\qquad$ $da = \frac{dL(a,y)}{da}$

$dz = a - y$ $\qquad$ $da = \frac{-y}{a} + \frac{1-y}{1-a}$

Vectorizing Logistic Regression

$z^{(1)} = w^T x^1 + b$	$z^2 = w^T x^2 + b$	$z^3 = w^T x^3 + b$
$a^1 = \sigma(z^1)$	$a^2 = \sigma(z^2)$	$a^3 = \sigma(z^3)$

$$X = \begin{bmatrix} | & | & & | \\ x^1 & x^2 & \cdots & x^m \\ | & | & & | \end{bmatrix} (n_x, m)$$

$$[z^1, z^2 \cdots z^m] = w^T X + [b \; b \cdots b]_{1 \times m}$$

$$[z^1, z^2 \cdots z^m] = [w^T x_1, w^T x_2 \cdots w^T x_m] + [b \; b \; b \cdots]$$

$$[1, 2, 3, 4, 5]_{1 \times 5} - [6, 6, 4, 3, 2]_{1 \times 5}$$