Lecture 05

"loss function"

$\hat{y}$ → predicted label y → actual label

w → weight b → bias

→ we can call 'w' and 'b' as PARAMETERS

→ The purpose is to pick the 'w' an 'b' in such a way, where logistic regression give $\hat{y} \approx y$

→ we want that $y^{(actual)}$ an $\hat{y}$ (predicted) should be closer.

Training the logistic Regression mean:

$\hat{y}$ = weight * Input image pixels + bias

↳ predicted label

→ our aim is to pick (w, b) in a way that gives the value of $\hat{y}$ (predicted) should be closer to 1 for the positive class (here mean by cat class in which we are interested)

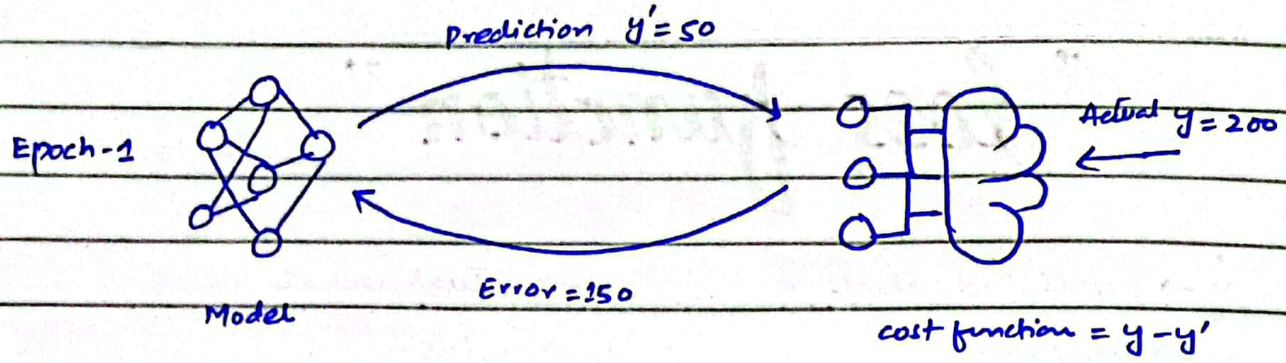
→ Note

How to find 'w' and 'b'

→ 'w' and 'b' can be calculated through loss function.

[Loss function is use to calculate distance between y and $\hat{y}$]

→ Distance in the loss function tell us how far off the predicted output is from the actual output.

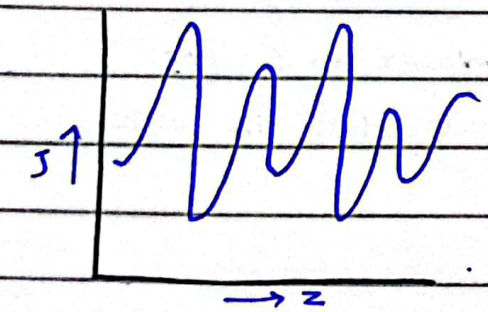


→ Loss function accept y and $\hat{y}$ as input and return a number (that represent distance) as an output

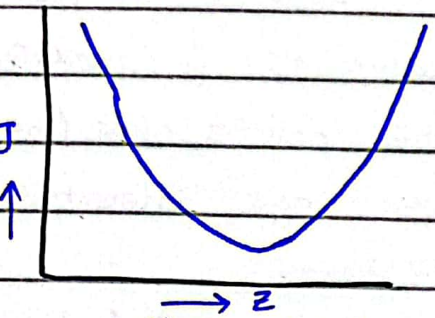
→ If the distance is closer to 0, it means there's a minimal distance b/w y and $\hat{y}$.

→ If that distance value is more than 0, it means there's a huge distance b/w y and $\hat{y}$

Non-conven vs conven functions



- Multiple local minima
- Global minimum harder to find



- Single global minimum
- local minimum is global

→ we can pick any function as a loss function but neural networks prefer conven functions for reliable optimization to find global minima.

→ Loss function use conven functions that have single minimum / maximum point

→ conven function has only one global optimum point

→ In Logistic Regression, we use Negetive log loss as a loss function

Negetive log loss: (cross-Entropy loss OR Binary cross entropy loss)

$$L(y,\hat{y}) = \begin{cases} \hat{y} \approx y \Rightarrow L(y,\hat{y}) \to 0 \\ \hat{y} \neq y \Rightarrow L(y,\hat{y}) \to \infty \end{cases}$$

$$L(y,\hat{y}) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})] \quad \text{— eq (1)}$$

case: 01

$y=0$, $\hat{y}=0$ we can say $y \approx \hat{y}$

$$L(y,\hat{y}) = -[0 \log \hat{y} + (1-0)\log(1-0)]$$

$$= -[0 + 1 \log 1]$$

$$= -[1 \log^1] \qquad \because \log_{10} 1 = 0$$

$$= -[\log^1] \qquad \because 10^0 = 1$$

$$= -0$$

$$\text{Loss} = 0$$

↳ There is no distance b/w y and $\hat{y}$

case: 02

$y=1$, $\hat{y}=0.1$

$$L(y,\hat{y}) = -[1 \log(0.1) + (1-1)\log(1-0.1)]$$

$$= -[1 \log(0.1) + 0]$$

$$= -[1 \log 0.1]$$

$= -[-1]$ $\quad \because \log 0.1 = \log_{10}^{10^{-1}} = -1$

$= 1$

There's a significant distance b/w y and $\hat{y}$, that means the values of w and b should be optimized.

Cost function:

→ It is the combined loss of all training examples

For one sample:

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

For 'M' samples: Add $L(y^{(i)}, \hat{y}^{(i)})$ for all and take average

↑ cost function

$$J(w) = \sum_{i=1}^{M} \frac{L(y^{(i)}, \hat{y}^{(i)})}{M} \qquad \because \hat{y} = \sigma(w^T x + b)$$

$$J(w,b) = -\frac{1}{M}\left[y \log \hat{y} + (1-y) \log (1-\hat{y})\right]$$

$$J(w,b) = -\frac{1}{M}\left[y \log \{\sigma(w^T x + b)\} + (1-y) \log \left(1 - \{\sigma(w^T x + b)\}\right)\right]$$

$y^{(i)} \approx \hat{y}^{(i)}$ (predicted outputs are approximately equals to actual outputs)

Lecture : 06

Gradient Descent

→ Gradient Descent is use to optimize the parameters (w & b)

→ Gradient means → slope or derivatives or rate of change

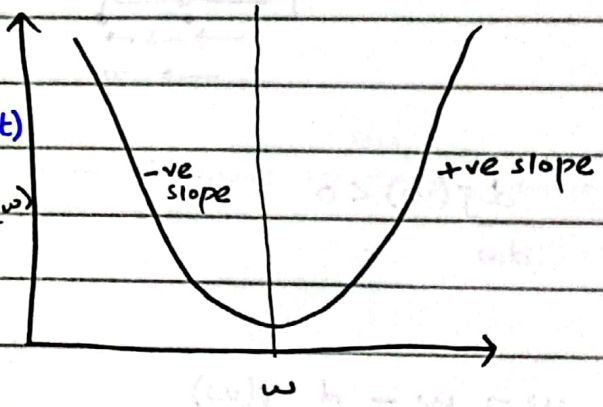
→ Descent means → to move in downward direction

> Gradient decent is an optimization algorithm which iteratively moves in the direction of the negetive gradient to find the minimum value of loss function.

$dJ(w) > 0$: 'w' decrease (move left)

$dJ(w) < 0$: 'w' increase (move right)

$dJ(w) = 0$: 'w' stays the same, optimal value found



$J(w)$ → cost of all training examples

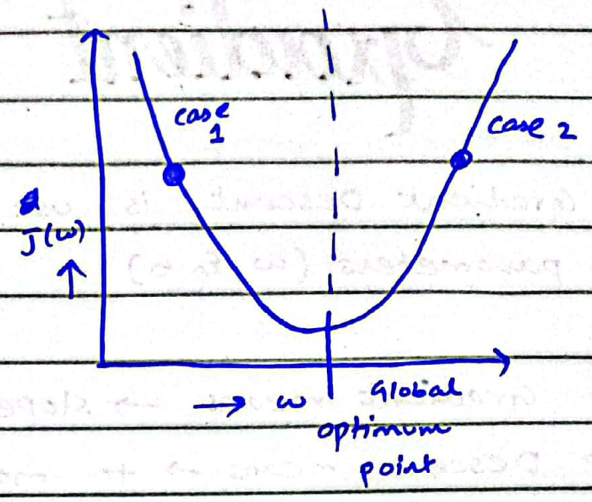
$$w = w - \frac{d}{dw} J(w)$$

(w updated as)

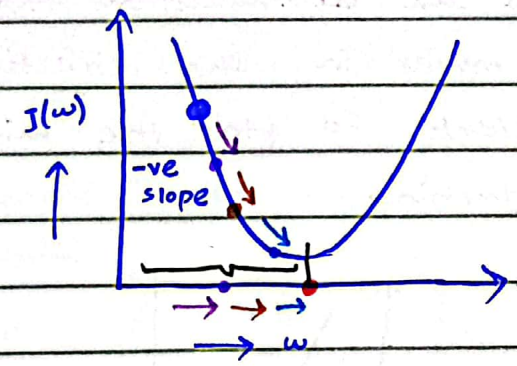
($\frac{d}{dw} J(w)$ → change in loss per unit change in weight)

$$\left[w = w - \frac{d}{dw} J(w) \right]$$ Gradient descent Algorithm

→ choose random 'w' value



Case: 01



$$\frac{dJ(w)}{dw} < 0$$

→ $w = w - \frac{d}{dw} J(w)$

$w = w - (-ve\ number)$

$w = w + num$

→ $w = w - (-ve\ number)$

$w = w + num$

$w = w + (0.1)$

→ $w = w - 0$

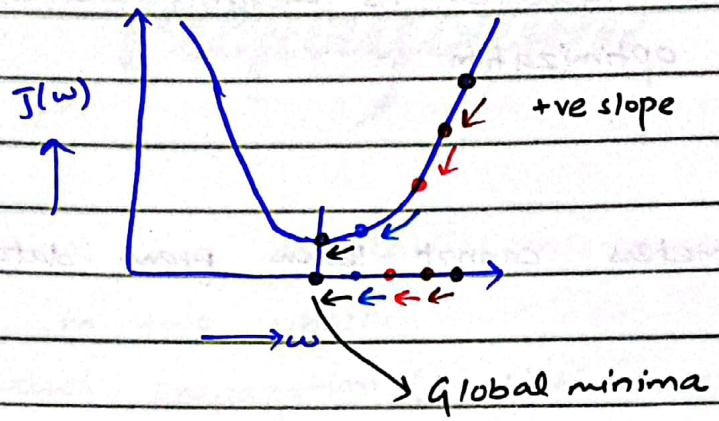
$w = w$ (reach at global minimum point)

(weight is converged now)

(loss is minimized now)

$J(w) \approx 0$ (minimum loss)

Case : 02



$dJ(w) > 0$

$\rightarrow w = w - \frac{d}{dw} J(w)$

$w = w - (+ve\ num)$ (w value decrease so move downward) (loss)

$\rightarrow w = w - (+ve\ num)$

$w = w - (very\ small\ number \approx 0)$

$w = w - 0 \rightarrow$

- w is optimized
- cost is minimized
- $y \approx \hat{y}$
- Nemal Network is trained now

$$w = w - \alpha \frac{d}{dw} J(w)$$

Learning rate (α) :-

The learning rate is a hyperparameter that controls the step size at which a model updates its weights and bias during optimization

→ hyperparameters cannot learn from data.

Lecture : 07 (part: 1+2)

forward and backward Propagation

➡ computations and calculations in neural network can be in two phases

1) Forward propagation (predictions, loss function)
2) Backward propagation (optimize parameters)

➡ For understanding forward and backward propagation we use computational graph.

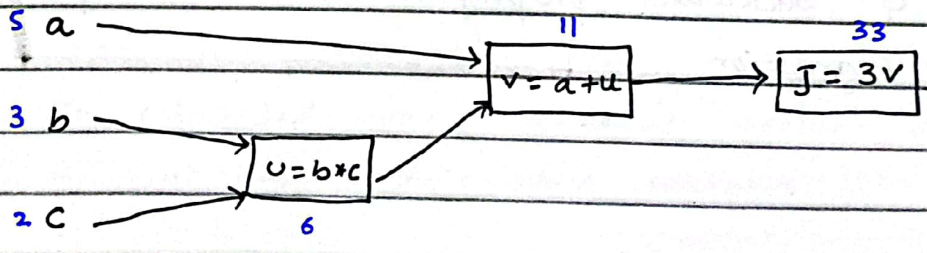
Computational Graph:-

➡ Explain different computations in the form of one graph node.

Let:

$$J(a,b,c) = 3(a+b*c)$$

input parameters



$\frac{dv}{da} = \frac{d}{da}(a+u)$

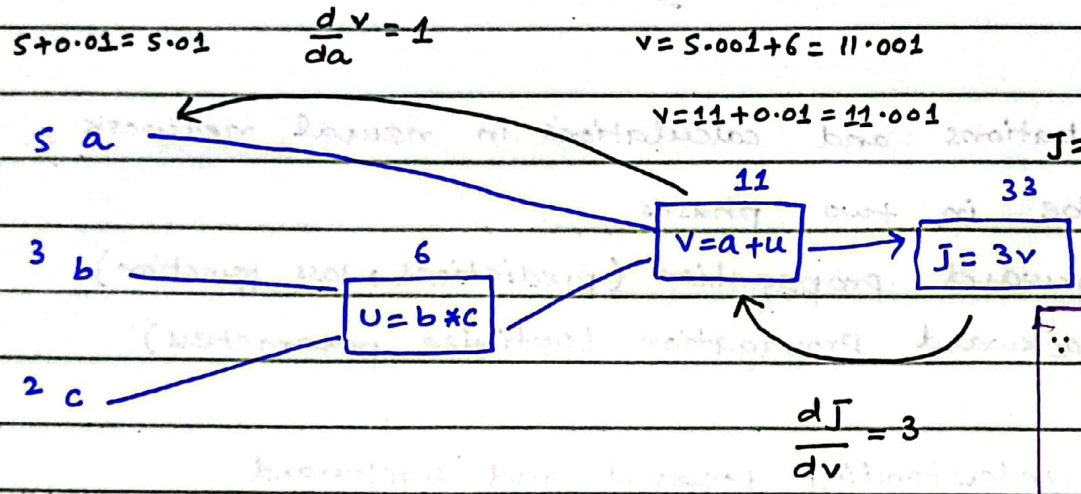
$= \frac{d}{da}(a) + \frac{d}{da}(u)$

$= 1 + 0$

$= 1$

$\frac{dJ}{da}$ (unit change in a, b, c effect J)

➡ (changing the unit of the independent variable and observing its effect on the dependent variable) Backward propagation



$\because \frac{dJ}{dv} = 3v$

$= \frac{d}{dv}(3v)$

$= 3\frac{d}{dv}(v)$

$= 3(1)$

$= 3$

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

$= 1 \times 3$

$= 3$

Forward propagation: Inference, loss find

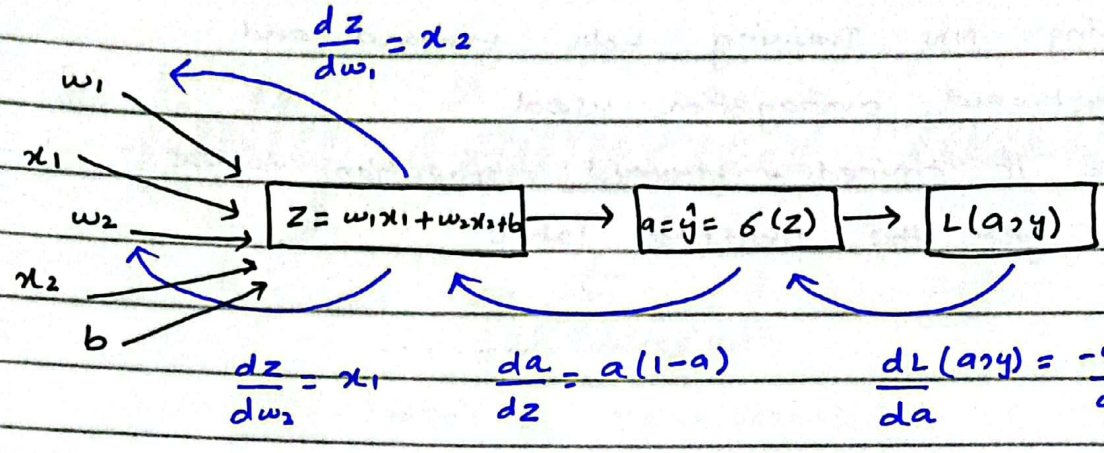
Backward propagation: Adjust parameters to minimize the loss

Forward & Backward propagation in terms of Logistic Regression

$$z = w^T x + b$$

$$\hat{y} = \sigma(z)$$

$$\hat{y} \approx y$$



→ Backward propagation provides relationship between parameters and loss

→ If it's positive means 'w' increase 'loss' increase

→ for negetive means 'w' increase 'loss' decrease

$$\frac{dL}{dw_1} = \frac{dz}{dw_1} \times \frac{da}{dz} \times \frac{dL}{da}$$

$$= x_2 \cdot a(1-a) \cdot \frac{-y(1-y)}{a(1-a)}$$

$$= x_1(a - y)$$

⇒ when weight is optimized and loss is minimize then

$$\hat{y} \approx y$$

our Neural Network is Trained

⇒ once the relationship b/w parameters and loss is calculated then gradient decsent algorithm is use to update the weights.

$$w_1 = w_1 - \alpha\left(\frac{dL}{dw_1}\right)$$

=> During NN Training both forward and backward propagation used

=> Once it trained, forward propagation is use to get the predicted labels

Lecture: 08

fully Connected neural network

(1980's by Yann Lecun)

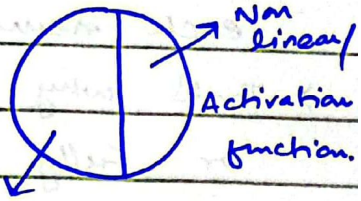
→ Fully connected neural network is also known as standard neural network or dense neural network.

→ Each neuron in the layer is connected to every neuron in subsequent layer

→ Each neuron has two parts

① Linear part

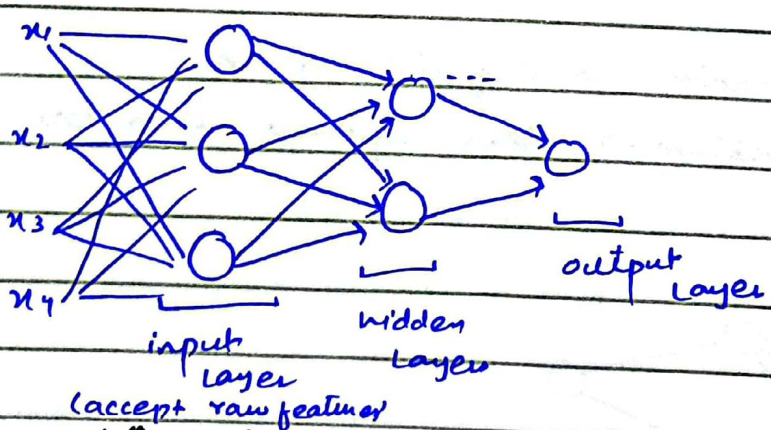
② Non Linear / activation function



→ activation function depends on the type of problem, it could be (ReLU, sigmoid, softmax etc)

1) Layers

It contains input-layer, hidden layers, output layers



∵ hidden layers find complex patterns

2) Activation function

It depends on the problem

→ for binary classification sigmoid is use as an activation function in output layer.

→ for regression problem RELU can be use as an activation function in output Layer.

→ Neuron is also known as activation function / non-linearity

→ Mostly, RELU is use as an activation function in Input and hidden layer.

3) Structure

→ Each neuron in a layer is connected to each neuron in subsequent Layer.

→ That's why it is known as Dense neural network or Fully connected neural network or standard neural network.

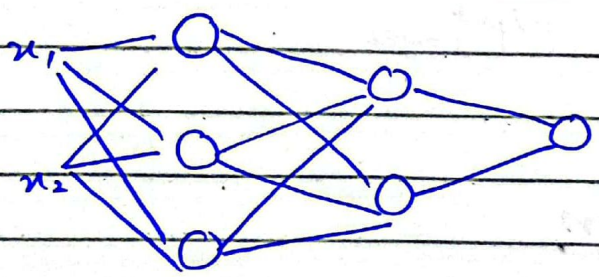
4) Forward Propagation

→ In forward propagation predicted labels and loss is calculated

→ loss function is choosen on the basis of problem

① Binary classification → Negative log loss

② Regression problem → Mean squared Error loss



(rows) (Columns)

$W \in R$ Hidden units $\times$ input features

$$z^{[1]} = w^{[1]} x + b \quad \dots \quad a^{[1]} = g(z^{[1]})$$

($a^{[1]}$ feeds back into the next step; g is the activation function)

$$z^{[2]} = w^{[2]} a^{[1]} + b \quad \dots \quad a^{[2]} = g(z^{[2]})$$

$$z^{[3]} = w^{[3]} a^{[2]} + b \Rightarrow \hat{y} = a^{[3]} = g(z^{[3]})$$

Then:

y is compared with $\hat{y}$

On Backward propagation, we optimize the weights to reduce the loss.

when $y \simeq \hat{y}$

Applications :-

→ Hand-written digits identification (cv problem)

→ House price prediction

→ For structured data, standard neural network is prefered.

→ speech recognition

→ NLP (sentiment classification)

→ Fraud detection in finance