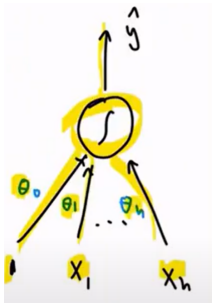
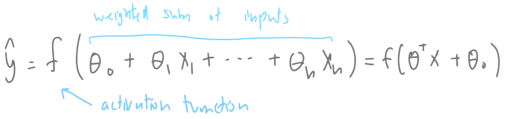
Sahil Sheikh : Assignment 02
CS577 - Fall 22

1.



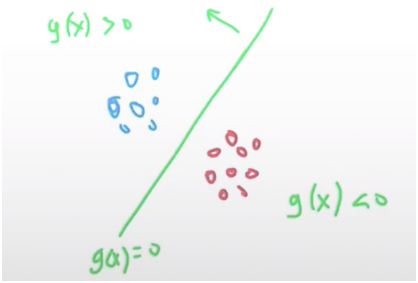
Weights = (0.1,0.2,0.3)
Input = (1,1)
Output: ŷ =?
Soln:

ŷ = f ($\theta_0 + \theta_1 x_1 + \theta_2 x_2$) Where f : activation function.
ŷ = f (0.1 + 0.2*1+0.3*1)
ŷ = f 0.6



2.

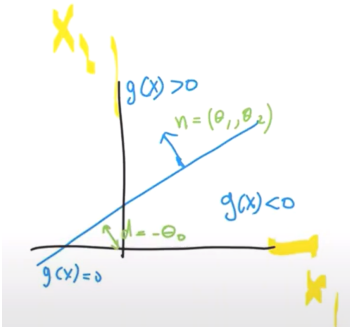
For g(x), a linear discriminant function, its values can be defined through the figure.
g(x) would be zero on the decision boundary
g(x) would be positive (g(x)>0) for one side of boundary
And for the other side of boundary g(x) would be negative (g(x) <0)



3. For $g(x_1,x_2) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$,

we can say θ_0 is the bias i.e. distance of decision boundary From origin.

$\theta_1$and θ_2 are the coefficients of normal vectors that define the decision boundary.



4.

The normal to the decision boundary defined by $g(x_1,x_2)$ is given by $\theta_1$and θ_2 = (2,3)

The distance of decision boundary from the origin is θ_0 = 1

5.

For $g(x) = \theta^T x$, the bias coefficient(θ_0)in θ such θ = [$\theta_0,\theta_1, \theta_2$ ….θn]
And for writing bias coefficient in θ we have add 1 in x in such a way that :

x = [1, x_1,x_2,.....xn]

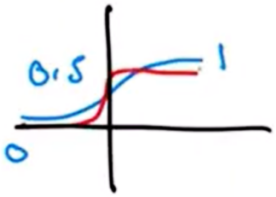
6.

Step activation function can be expressed as:

$$h_\theta(x) = \begin{cases} 1 \text{ if } \theta^T x > 0 \\ 0 \text{ for otherwise} \end{cases}$$

Sigmoid function can be expressed as:

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



where $g(x) = \theta^T x$

7.

Sigmoid is obtained using a linear function to model the log-likelihood ratio
Consider two classes c1 and c2.
Now probability for given x to be class c1 can be calculated as:

$P (y=1|x)$

And similarly, probability for given x to be class c2 can be calculated as:

$P (y= 0| x)$ … as $P(y=0|x) = 1 - P(y=1|x)$

To compare the above probabilities we find their ratio and this ratio is called likelihood ratio.

The likelihood ratio would be:

$$\frac{P (y=1|x)}{P (y= 0| x)} = \begin{cases} >1 \rightarrow c1 \\ <1 \rightarrow c2 \end{cases}$$

If the likelihood ratio is less than 1, the input x will belong to the C2 and if the result of the likelihood ratio is more than 1 then the input x will belong to the C1.

Taking log on both sides

$$\text{Log}\left(\frac{P (y=1|x)}{P (y= 0| x)}\right) = \begin{cases} >0 \rightarrow c1 \\ <0 \rightarrow c2 \end{cases}$$

Let the log of likelihood ratio be modeled as a linear function " $\theta^T x$"

$$\text{Log}\left(\frac{P(y=1|x)}{P(y=0|x)}\right) = \theta^T x$$

$$\frac{P(y=1|x)}{P(y=0|x)} = e^{\theta^T x}$$

$P(y=1|x) = P(y=0|x) * e^{\theta^T x}$ …. rearranging

$P(y=1|x) = (1 - P(y=1|x)) * e^{\theta^T x}$

$P(y=1|x) * (1 + e^{\theta^T x}) = e^{\theta^T x}$

$$P(y=1|x) = \frac{e^{\theta^T x}}{1 + e^{\theta^T x}}$$

$$P(y=1|x) = \frac{1}{1 + e^{-\theta^T x}}.$$

But we know :

$h_\theta(x) = \frac{1}{1 + e^{-g(x)}}$. is the sigmoid function

$P(y=1|x) = h_\theta(x)$

8.

The derivation of sigmoid :

For $\text{sigmoid}(x) = \frac{1}{1 + e^{-(x)}}$.

$\frac{d}{dx}\text{sigmoid}(x) = \text{sigmoid}(x) * (1 - \text{sigmoid}(x))$

$\frac{d}{dx}$ log sigmoid(x) = $\frac{1}{\text{~~sigmoid(x)~~}}$ *~~sigmoid(x)~~ *(1- sigmoid(x)) = 1- sigmoid(x)

Derivatives w.r.t θ

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

$\frac{d}{d\theta} h_\theta(x) = (h_\theta)*(1- h_\theta)x$

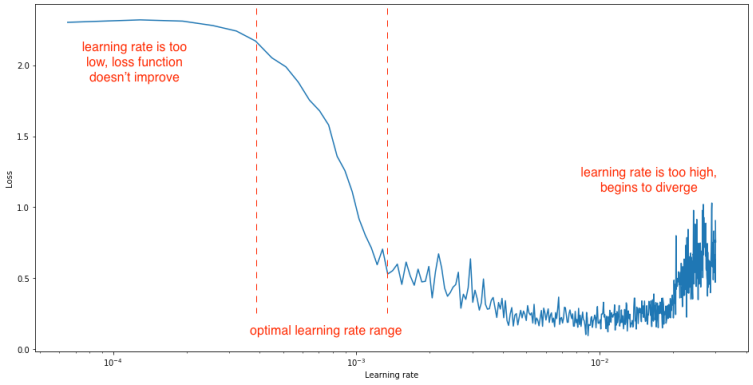
$\frac{d}{d\theta} \log h_\theta(x) = \frac{1}{\text{~~}h_\theta\text{~~}} *(\text{~~}h_\theta\text{~~})*(1- h_\theta)x = (1- h_\theta)x$

$\frac{d}{d\theta} (1-h_\theta(x)) = (1 -(1- h_\theta)x(-1)) = - h_\theta * x$

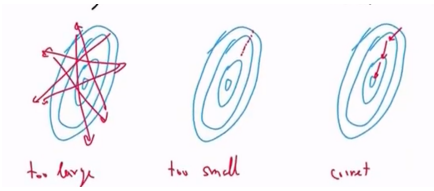
9. Vjbvub
10. Jjbjj

11. When the learning rate to too small or too large we observe the following cases:
    a. For a very large learning rate, the model might overshoot the minima and might not be able to converge. This will lead our loss function to behave in an undesired divergent manner.

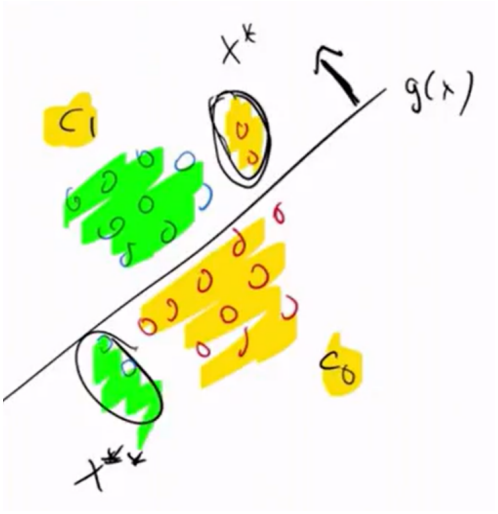
    b. For a very small learning rate, the model will train very slowly and there won't be improvement in the loss function



https://www.jeremyjordan.me/content/images/2018/02/lr_finder.png



12.



In empirical loss we calculate the samples the model classified in-correctly.

Empirical loss is calculated as follows:
Consider samples belonging two classes C0, C1
And the X* and X** represent the set of samples C0 and C1 classified in-correctly respectively.
So,

$X^* = \{ X^{(i)} | 1(y^{(i)} = 0 \wedge \theta^T x > 0 \}$

$y^{(i)} = 0$ represents the samples of class C0

$\theta^T x > 0$ is the distribution function representing the samples on above the decision boundary

$X^{**} = \{ X^{(i)} | 1(y^{(i)} = 1 \wedge \theta^T x < 0 \}$

$y^{(i)} = 1$ represents the samples of class C0

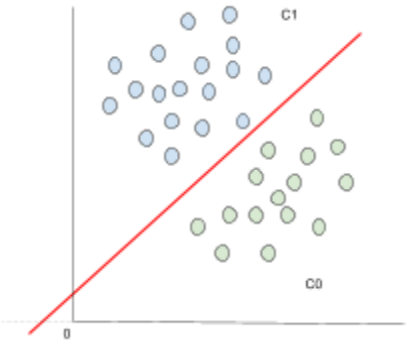
$\theta^T x < 0$ is the distribution function representing the samples on below the decision boundary

$E(\theta) = \#X^* + \#X^{**}$

$\Theta^* = \underset{\Theta}{\operatorname{argmin}}\ E(\theta)$

The problem with Empirical Loss function is that its a piecewise constant which produces a gradient of zero due to which empirical loss can not be minimized with Gradient Descent.

13.



Consider two classes C0 and C1 as follows:
Now to maximize the likelihood for binary classifier can be calculated as:

$$L(\theta) = \prod_{x^{(i)} \in C1} P(y =1| x^{(i)}) \prod_{x^{(i)} \in C0} P(y =0| x^{(i)})$$

Which can also be represented as:

$$L(\theta) = \prod_{i=1}^{m} P(y =1| x^{(i)})^{y^{(i)}} P(y =0| x^{(i)})^{1-y^{(i)}}$$

Now to minimize the negative log of likelihood
$l(\theta) = - \log L(\theta)$

Therefore,

$$l(\theta) = -\log \prod_{i=1}^{m} P(y=1|x^{(i)})^{y^{(i)}} \; P(y=0|x^{(i)})^{1-y^{(i)}}$$

$$l(\theta) = -\sum_{i=1}^{m} y^{(i)} \log\left(P(y=1|x^{(i)})\right) + 1-y^{(i)} \log\left(P(y=0|x^{(i)})\right)$$

$$l(\theta) = -\sum_{i=1}^{m} y^{(i)} \log\left(P(y=1|x^{(i)})\right) + 1-y^{(i)} \log\left(1 - P(y=1|x^{(i)})\right)$$

Let $P(y=1|x^{(i)})$ be substituted as $h_\theta(x)$

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

The above equation is the binary cross entropy for 2 classes.

14.

Binary cross entropy is given by:

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

The gradient of binary cross entropy function with respect to parameter vector when using sigmoid activation is as follows:

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

$$= -\sum_{i=1}^{m} y^{(i)} (1-h_\theta(x^{(i)})x^{(i)} + \sum_{i=1}^{m} (1-y^{(i)}) (-h_\theta(x^{(i)})x^{(i)}$$

$$= -\sum_{i=1}^{m} \left(y^{(i)} - \cancel{y^{(i)}} * \cancel{h_\theta(x^{(i)})} - h_\theta(x^{(i)}) + \cancel{y^{(i)}} * \cancel{h_\theta(x^{(i)})} \right) x^{(i)}$$

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

Therefore ,

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

The gradient descent using this function would be:

$$\theta \leftarrow \theta - \eta \frac{d}{d\theta}(-l(\theta))$$

$$\theta \leftarrow \theta - \eta \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

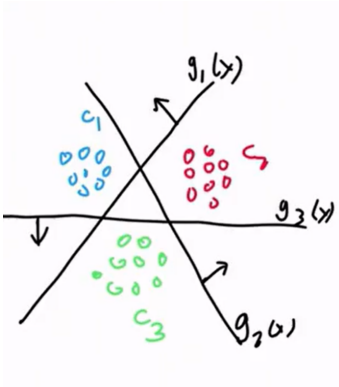
The above is the update rule, which calculates the difference between the previous value and learning rate times loss i.e. the difference between the prediction $h_\theta(x^{(i)})$ which can also be written as $\hat{y}^{(i)}$ and the original label $y^{(i)}$.
Whenever the prediction and true label agree there is no update to the parameter but when they don't agree there will be an update which will be feature vector $x^{(i)}$ times the learning rate.

15.

The two strategies used in multi class classification are as follows:

a. One vs All:
   It can be defined as a heuristic method which uses binary classification algorithms to classify multi class datasets.
   In this strategy the multi class data is split into multiple binary classification problems and
   then a binary classifier is trained on each one of the classes to give predictions.

   The number of discriminant functions would be equal to the number of classes.



b. One vs Each other:
This is also type of heuristic method which involves the use of binary classifiers to classify multi-class classification problems.
In this strategy we split the dataset into binary classification problems just like the one vs all approach. There's only difference in the approach, here the dataset is split into one set for each class against every other class.

The number of discriminant functions for "k" classes would be : $\frac{K(K+1)}{2}$

The one vs Each is easier to implement compared to One vs All strategy.

16. Will do later

17. A multi-class classification neural network model should be able to predict multinomial probability distribution for the given number of classes. The softmax is used as an activation function in the output layer of multi-class classification problems as it normalizes the output, converting them from weights sum values to probabilities which sum to one. Thus each value of the output of softmax is the probability of membership for each class.

18.
The softmax function is represented by

$$\text{softmax}(z_j) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}}.$$

Derivative of softmax is:

$$\frac{d}{d z_i}\text{softmax}(z_j) = \text{softmax}(z_j)(\delta_{ij} - \text{softmax}(z_j)$$

Where $\delta_{ij} = \begin{cases} 1 & i=j \\ 0 & \text{otherwise} \end{cases}$

Similarly for

$$\frac{d}{d\,\theta_j^T x}\text{softmax}(\theta_j^T x) = \text{softmax}(\theta_j^T x)(\delta_{ij} - \text{softmax}(\theta_j^T x)x$$

Therefore derivative for log $\text{softmax}(\theta_j^T x)$ is given by:

$$\frac{d}{d\,\theta_j^T x} \log\left(\text{softmax}(\theta_j^T x)\right) = \text{softmax}(\theta_j^T x)(\quad_{ij} - \text{softmax}(\theta_j^T x)x$$

19.

The log likelihood for a binary class classifier :

$$l(\theta) = -\log \prod_{i=1}^{m} P(y=1|x^{(i)})^{y^{(i)}} \; P(y=0|x^{(i)})^{1-y^{(i)}}$$

Indicator function:

$$1(y^{(i)} = j) = \begin{cases} 1 & y^{(i)} = j \\ 0 & \text{otherwise} \end{cases}$$

$$l(\theta) = -\log \prod_{i=1}^{m} P(y=1|x^{(i)})^{1(y^{(i)}=1)} \; P(y=0|x^{(i)})^{1y^{(i)}=0}$$

The negative log likelihood function:

$$l(\theta) = -\log\left(\prod_{i=1}^{m}\prod_{j=1}^{k} P(y=j|x^{(i)})^{1(y^{(i)}=j)}\right)$$

$$= -\sum_{i=1}^{m}\sum_{j=1}^{k} 1(y^{(i)}=j)\log P(y=j|x^{(i)})$$

$P(y=j|x^{(i)})$ can be represented as $h_{\theta_j}(x^{(i)})$

$$l(\theta) = -\sum_{i=1}^{m}\sum_{j=1}^{k} 1(y^{(i)}=j)\log h_{\theta_j}(x^{(i)})$$

This will calculate the negative log likelihood of the prediction. The above equation is called Categorical Cross - Entropy.

20.

The gradient of cross entropy loss function with respect to the parameter vector when using a softmax function:

$$\frac{d}{d\theta_j} l(\theta) = \frac{d}{d\theta_j} - \sum_{i=1}^{m} \sum_{j=1}^{k} 1(y^{(i)} = j) \log h_{\theta_j}(x^{(i)})$$

$$= \sum_{i=1}^{m} h_{\theta_j}(x^{(i)}) - 1(y^{(i)} = j) x^{(i)}$$

Therefore

$$\theta_j \leftarrow \theta_j - \eta \sum_{i=1}^{m} h_{\theta_j}(x^{(i)}) - 1(y^{(i)} = j) x^{(i)}$$

Answers

1.

Hidden layers are generally utilized for feature extraction and hierarchical classification.
The dimensionality increase is caused as the hidden layer has more units that compared to the input layer. This allows us to map the inputs to a higher dimensional space. The main benefit of dimensional increase it can compute nonlinear classification

$h > n$: dimensionality increase

2.

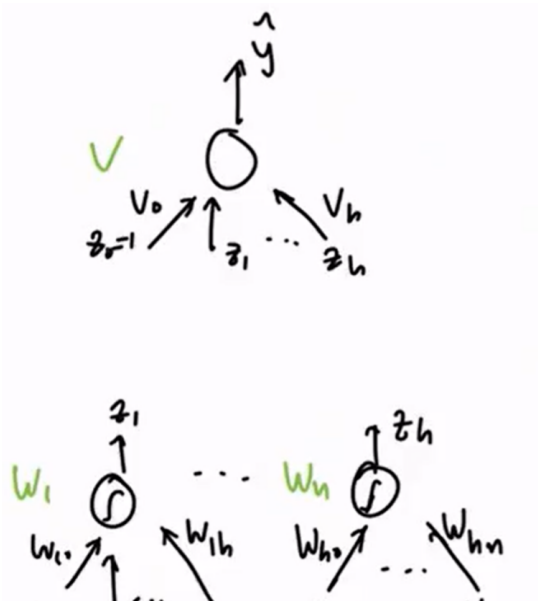
When the hidden layer has less units as compared to the input layer, this causes dimensionality reduction.
This allows the model to simplify the data.

$h < n$: dimensionality reduction

4.

Consider 2 layers as follows:

V being the output layer
W being the hidden layer.
ŷ: The final output
Z1,..., Zn: Outputs of hidden layer
x1,.., xn: Inputs
x0 : bias(1)



Loss can be calculated as:

$$E = \frac{1}{2} * \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

Now applying chain rule output layer:

$$\frac{dE}{dV} = \frac{dE}{d\hat{y}} \cdot \frac{d\hat{y}}{dV}$$

$$\frac{dE}{dV} = \frac{1}{2} * 2 * \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) * z^{(i)}$$

$$\frac{dE}{dV} = \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) * z^{(i)}$$

Now computing the gradient descent :

$$V = V - \eta \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) * z^{(i)}$$

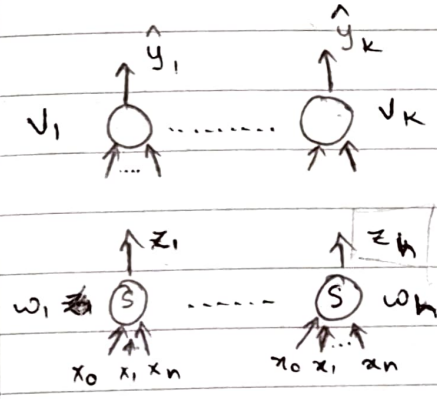
$$\frac{dE}{dW_j} = \frac{dE}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_j} \cdot \frac{dz_j}{dW_j}$$

$$\frac{dE}{dW_j} = - \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) * z^{(i)} \; Vj \; Z_j^{(i)} \; (1 -) \; X^{(i)}$$

No.

Date

So



2-layer regression network

Inputs: $\{ x^{(i)}, y^{(i)} \}_{i=1}^{m}$

$x^{(i)} \in R^n \quad y^{(i)} \in R^h$

Outputs:

$\hat{y}_j = V_j^T z$

$z_j = \text{sigmoid}(w_j^T x)$

Loss (~~E~~) =

$E(\cancel{V_1, w_h}) = E(\theta) = \frac{1}{2} \cancel{\sum_{i=1}^{m} \sum_{j=1}^{k} (\hat{y}_j^{(i)} - y_j^{(i)})^2}$, $n = h$, $m = k$.

$= -\sum_{i=1}^{m} \sum_{j=1}^{k} y_j^{(i)} \log(\hat{y}_j^{(i)})$

by Chain Rule

$\frac{dE}{dV_j} = \frac{dE}{d\hat{y}_j} \cdot \frac{d\hat{y}_j}{dV_j}$

$\frac{1}{2} - \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{y_j^{(i)}}{\hat{y}_j^{(i)}} * z^{(i)}$

$\sum_{i=1}^{m} \hat{y}_j^{(i)} - 1(y^{(i)} = j) x^{(i)}$

$\therefore \frac{dE}{dw_j} = \sum_{e=1}^{h} \frac{dE}{d\hat{y}_j} \cdot \frac{d\hat{y}_j}{dz_e} \cdot \frac{dz_e}{dw_e}$

$= \sum_{e=1}^{h} - \sum_{i=1}^{m} \sum_{j=1}^{k} \frac{y_j^{(i)}}{\hat{y}_j^{(i)}} * V_j * z_e^{(i)} (1 - z_e^{(i)}) x^{(i)}$

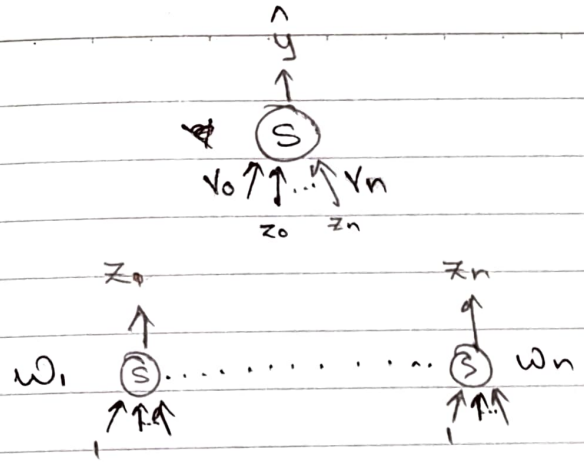
$= \sum_{g=1}^{k} \sum_{i=1}^{m} \hat{y}_j^{(i)} - 1(y^{(i)} = j) \cdot V_j * z_e^{(i)} (1 - z_e^{(i)}) x^{(i)}$

KOKUYO

No.

Date

6.



$x_0 = 1$, $x_1 - x_n$: ~~inputs~~

Inputs : $\{x^{(i)}, y^{(i)}\}_1^m \quad x^{(i)} \in R^n \quad y^{(i)} \in R$

~~outputs~~ : ~~$\hat{y}_i = \text{sigmoid}(z_i$~~

outputs : $\hat{y} = \text{sigmoid}(z_i^T v)$

$z_i = \text{sigmoid}(w_i^T x)$

~~loss~~ ~~$E(\theta)$~~ :

loss $E(\{w_{ij}, v\}) \equiv E(\theta) = \sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + (1-y^{(i)})$

$\therefore E(\theta) = \sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + (1-y^{(i)}) \log(1-\hat{y}^{(i)})$

∴ output layer gradient with chain rule

$$\frac{dE}{dv} = \frac{dE}{d\hat{y}} \frac{d\hat{y}}{dv} = \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \cdot z_j$$

hidden layer :

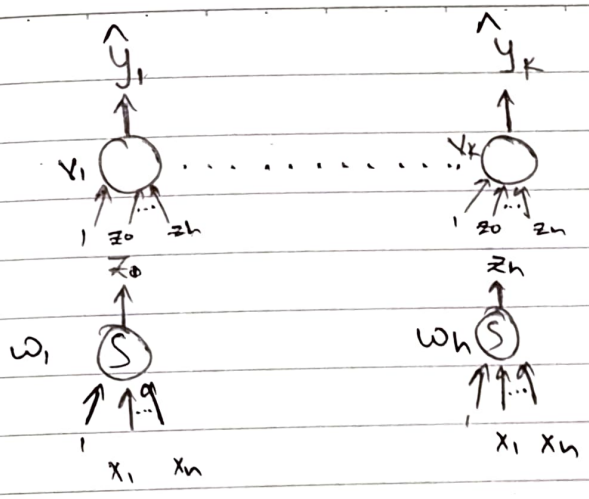
$$\frac{dE}{dw_j} = \frac{dE}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_j} \frac{dz_j}{dw_j}$$

$$= \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \cdot \hat{y}^{(i)}(1-\hat{y}^{(i)}) \cdot v_j \cdot z_j^{(i)}(1-z_j^{(i)}) x^{(i)}$$

No.

Date

7.



Inputs: $\{x^{(i)}, y^{(i)}\}_{i=1}^{m}$ $\quad x^{(i)} \in R \quad y^{(i)} \in R$

outputs:

$\hat{y}_j$: softmax $(z_j^T v)$

z_j : sigmoid $(w_j^T x)$

~~loss $E\{$~~

loss $E(\theta) = \sum_{i=1}^{m} \sum_{j=1}^{k} P(y=j \mid x^{(i)})^{1(y^{(i)}=j)}$

loss $E(\theta) = -\sum_{i=1}^{m} \sum_{j=1}^{k} 1(y^{(i)}=j) \log(\hat{y}_j^{(i)})$

$$\therefore \frac{dE}{dv} = \frac{dE}{d\hat{y}_j} \cdot \frac{d\hat{y}_j}{dv} = \sum_{i=1}^{m} (\hat{y}_j - 1(y^{(i)}=j)) v_j$$

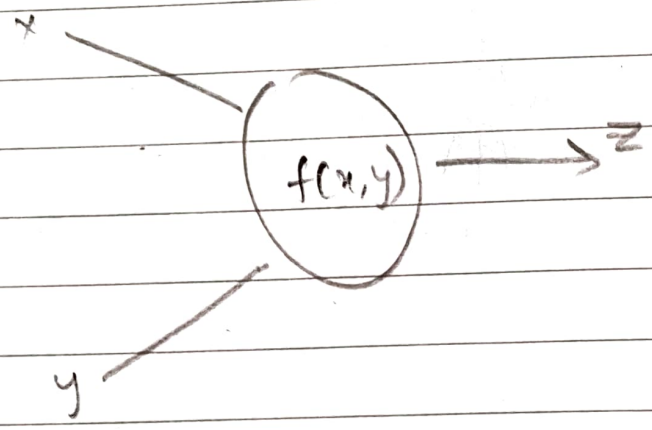
$$\therefore \frac{dE}{dw_j} = \sum_{j=1}^{k} \frac{dE}{d\hat{y}_j} \cdot \frac{d\hat{y}_j}{dz_i} \cdot \frac{dz_i}{dw_i}$$

$$= \sum_{j=1}^{k} \sum_{i=1}^{m} (\hat{y}_j^{(i)} - 1(y^{(i)}=j)) \cdot \text{softmax}(z_j^T v)(\delta_{ij} - \text{softmax} z_j^T v) v_j$$

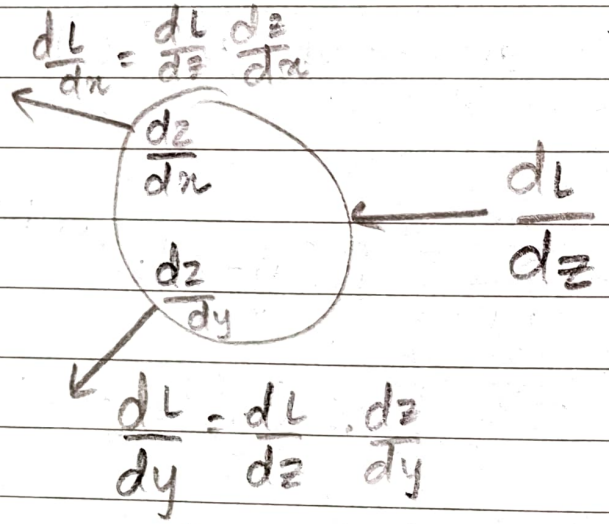
$\rightarrow (z_j^{(i)}(1 - z_j^{(i)}) x^{(i)}$

KOKUYO

1.



In forward pass nodes computes functional values



In backward pass node computes local gradient and outputs product of incomming & local gradient.

computation graphs

1. The advantges of computational graphs are as follows:-

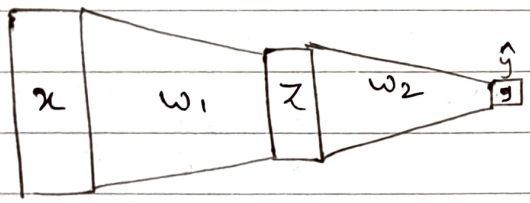
(i) They are flexible

(ii) The library is less invasive

→ In forward pass we pass the variables in forward direction to get the output

→ In backward pass the gradients which are essential for training of the neural network. These are computed in backward pass.

2.



x = input

$\hat{y}$ = output

f_1, f_2 : Activation function

w_1, w_2 : weights (unknown)

$$z = f_1(w_1, x)$$

$$\hat{y} = f_2(w_2, z)$$

$$\therefore \hat{y}(x) = f_2(w_2, f_1(w_1, x))$$

for a batch of $x^{(i)}, y^{(i)}\}_{i=1}^{m}$

$$L_2 \text{ Loss } (E_2) = \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2$$

$$= \sum_{i=1}^{m} (y^{(i)} - (f_2(w_2, f_1(w_1, x^{(i)})))^2$$

No.

Date

$$\frac{dE}{w_2} = \frac{dE}{d\hat{y}} \cdot \frac{d\hat{y}}{dw_2}$$

$$= \sum_{i=1}^{m} 2 \cdot (y^{(i)} - \hat{y}^{(i)}) \cdot \frac{d\hat{y}}{dw_2}$$

$$\frac{dE}{dw_1} = \frac{dE}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_j} \cdot \frac{dz_j}{dw_j}$$

$$= 2 \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)}) \cdot \frac{d\hat{y}}{dz_j} \cdot \frac{dz_j}{dw_j}$$

Now considering loss as binary cross entropy.

$$\text{Loss cross entropy } (E) = \sum_{i=1}^{m} y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)})$$

$$\therefore \frac{dE}{w_2} = \frac{dE}{d\hat{y}} \cdot \frac{d\hat{y}}{dw_2} = \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \cdot \frac{d\hat{y}}{dw_2} \; w_2 \; dy$$

$$\therefore \frac{dE}{w_{ij}} = \frac{dE}{d\hat{y}} \cdot \frac{d\hat{y}}{dz_j} \cdot \frac{dz_j}{dw_{ij}}$$

$$= \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \; w_1 \cdot \frac{dz_1}{dw_{ij}}$$

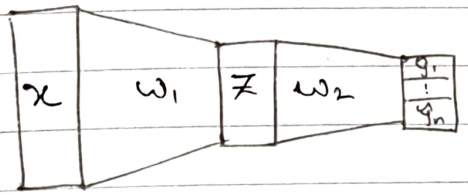
$$= \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)}) \; w_2 \cdot w_1 \frac{dz_j}{dw_{1j}}$$

KOKUYO

No.

Date

8.



Inputs : $\{ x^{(i)}, y^{(i)} \}_{i=1}^{m}$

Outputs : $\hat{y}_1 \cdots \hat{y}_K$

f_1, f_2 : Activation function.

$z = f_1(w_1, x)$

~~$\hat{y}_i = f_2(z_i,$~~

$w_{11} \cdots w_{1n}$: weight of w_1

$w_{21} \cdots w_{2n}$: weight of w_2

$z_j = f_1(w_{1j}, x)$

$\hat{y}_j = f_2(w_{2j}, z_j)$

Categorical cross entropy

$$(\epsilon) = -\sum_{i=1}^{m} \sum_{j=1}^{K} 1(y_j^{(i)} = j) \log \hat{y}_j^{(i)}$$

$$\therefore \frac{d\epsilon}{w_2} = \frac{d\epsilon}{d\hat{y}_j} \frac{d\hat{y}_j}{dw_2} = \sum_{i=1}^{m} \left(\hat{y}_j^{(i)} - 1(y^{(i)} = j) \right) \cdot w_2 \cdot \frac{d\hat{y}_j}{dw_2}$$

$$\frac{d\epsilon}{dw_1} = \frac{d\epsilon}{d\hat{y}_j} \sum_{j=1}^{n} \frac{d\epsilon}{d\hat{y}_j} \frac{d\hat{y}_j}{dz_k} \frac{dz_k}{dw_1}$$

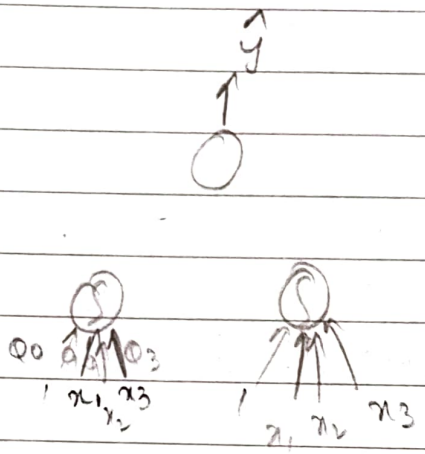
KOKUYO

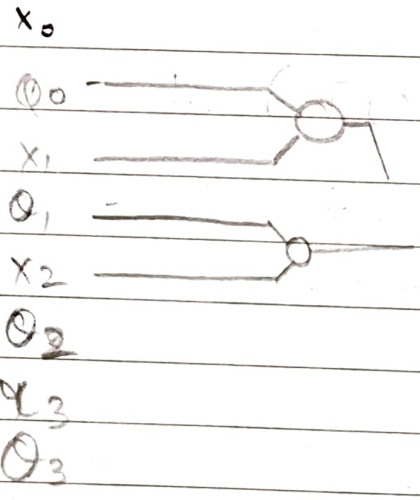
No.

Date

$$= \sum_{j=1}^{k} \sum_{i=1}^{m} \left(\hat{y}_j^{(i)} - 1(y^{(i)}=j) \cdot w_2 \cdot w_1 \cdot \frac{dz_j}{dw_1} \right.$$

4.



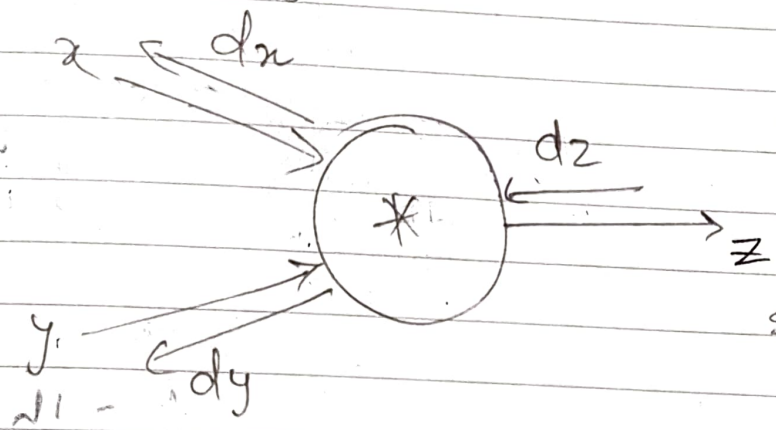


Date

5.

Multiplication of 2 inputs

$z = x * y$



$z = x * y$

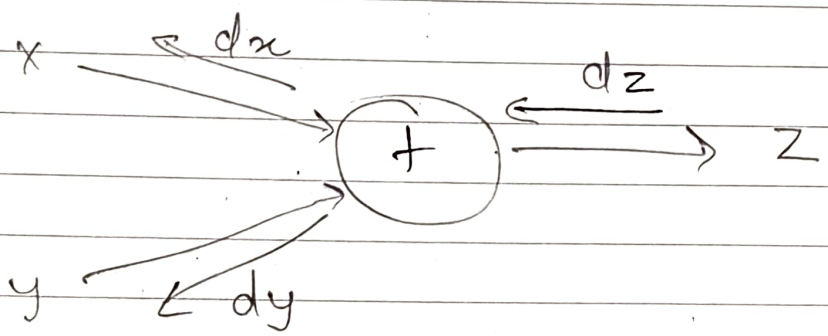
$\frac{dz}{dx} = y, \quad \frac{dz}{dy} = x$

so x & y are store in forward pass calculated.

$$dx = \frac{dz}{dx} * \frac{dL}{dz}, \quad dy = \frac{dz}{dy} * \frac{dL}{dz}$$

~~$\Rightarrow dx = y *$~~

$z = x + y$ $\quad \therefore \frac{dz}{dx} = 1, \quad \frac{dz}{dy} = 1$



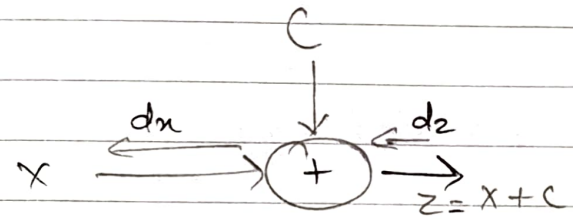
$$dx = \frac{dz}{dx} * \frac{dL}{dz}, \quad dy = \frac{dz}{dy} * \frac{dL}{dz}$$

$$\therefore dx = 1 * \frac{dL}{dz}, \quad dy = 1 * \frac{dL}{dz}$$

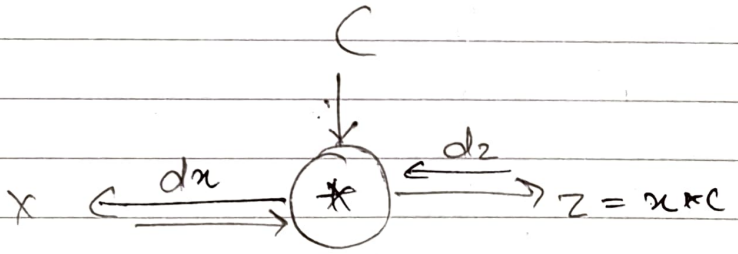
No.

Date

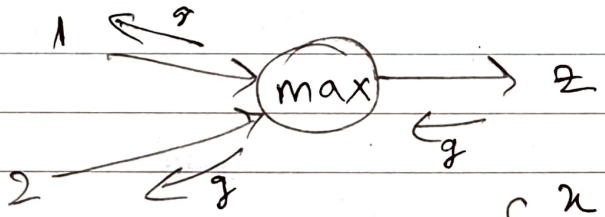
5.



$$\frac{dz}{dx} = 1$$



$$\frac{dz}{dx} = c$$



$$\max(x,y) = \begin{cases} x & x > y \\ y & y > x \end{cases}$$

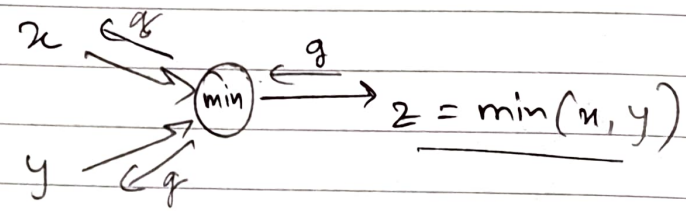
$$\therefore \frac{d}{dx} \max(x,y) = \begin{cases} 1 & x > y \\ 0 & y > x \end{cases}$$

$$\frac{d}{dy} \max(x,y) = \begin{cases} 0 & x > y \\ 1 & y > x \end{cases}$$

KOKUYO

No.

Date

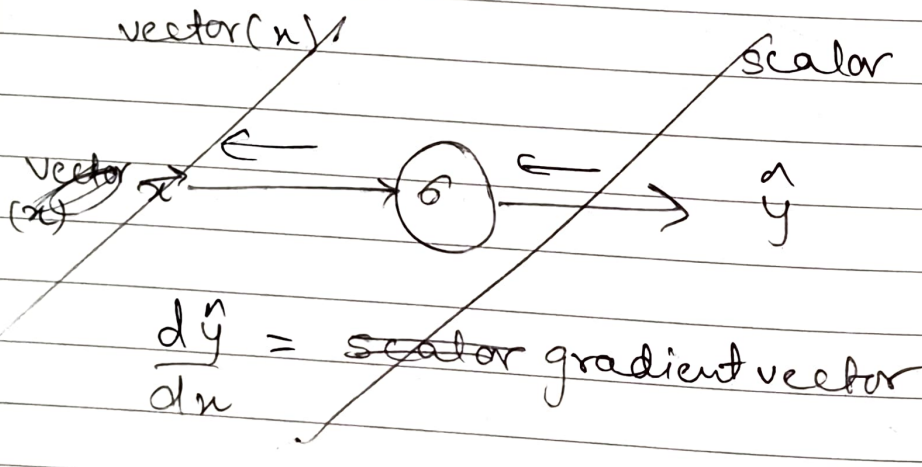


$$z = \begin{cases} x & x < y \\ y & y < x \end{cases}$$

$$\frac{dz}{dx} = \begin{cases} 1 & x < y \\ 0 & y < x \end{cases}$$

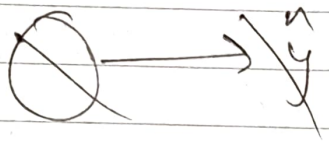
$$\frac{dz}{dy} = \begin{cases} 0 & x < y \\ 1 & y < x \end{cases}$$

6.

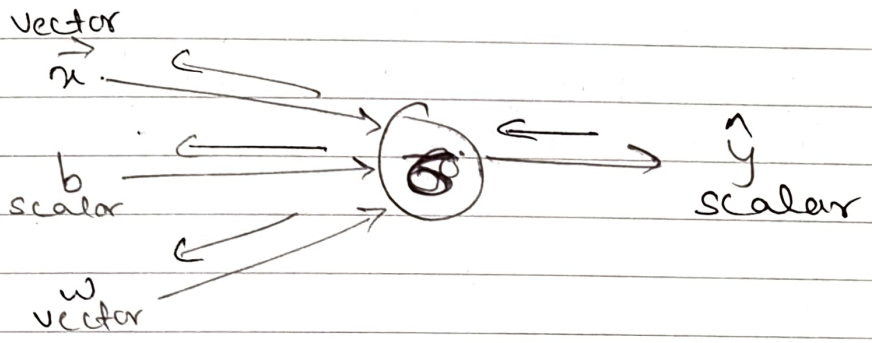


$$\frac{d\hat{y}}{dx} = \text{gradient vector}$$

7.



6. consider :



$\frac{d\hat{y}}{d\vec{x}}$ = gradient vector

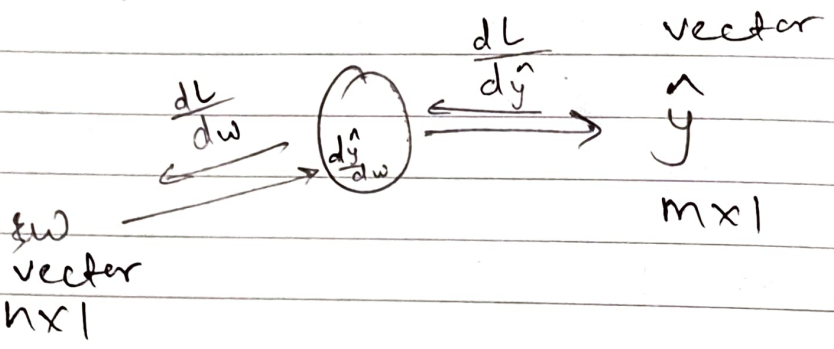
$\frac{d\hat{y}}{db}$ = scalar

$\frac{d\hat{y}}{dw}$ = gradient vector.

No.

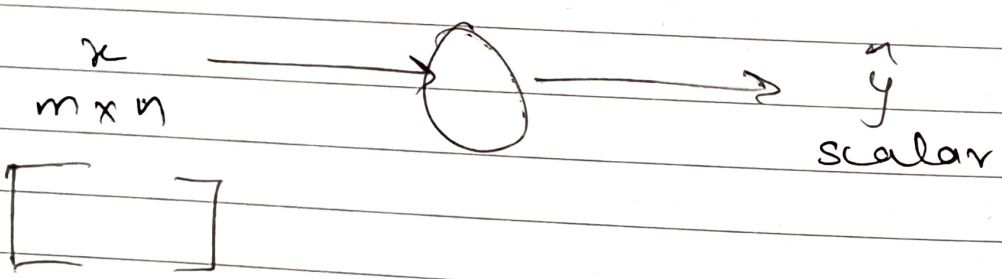
Date

7. conider



$\therefore \frac{d\hat{y}}{dw}$ = Rank 2 tensor (1+1)

8. consider :



$\therefore \frac{d\hat{y}}{dx}$ = Rank 3 tensor