classmate
Date
Page

Loss function in deep learning.

- Start with loss function?

- Loss function is a model of evaluating how your algorithm is modelling your dataset.
- high value ⇒ Algorithm performance is poor
  low value ⇒ Algorithm performing great.
- Loss function is just mathematical function.
- Loss function is function of parameter in machin learning.
  e.g. for linear regression ⇒ L(m, b)
- How we change the loss function value?
  - By changing parameter value.

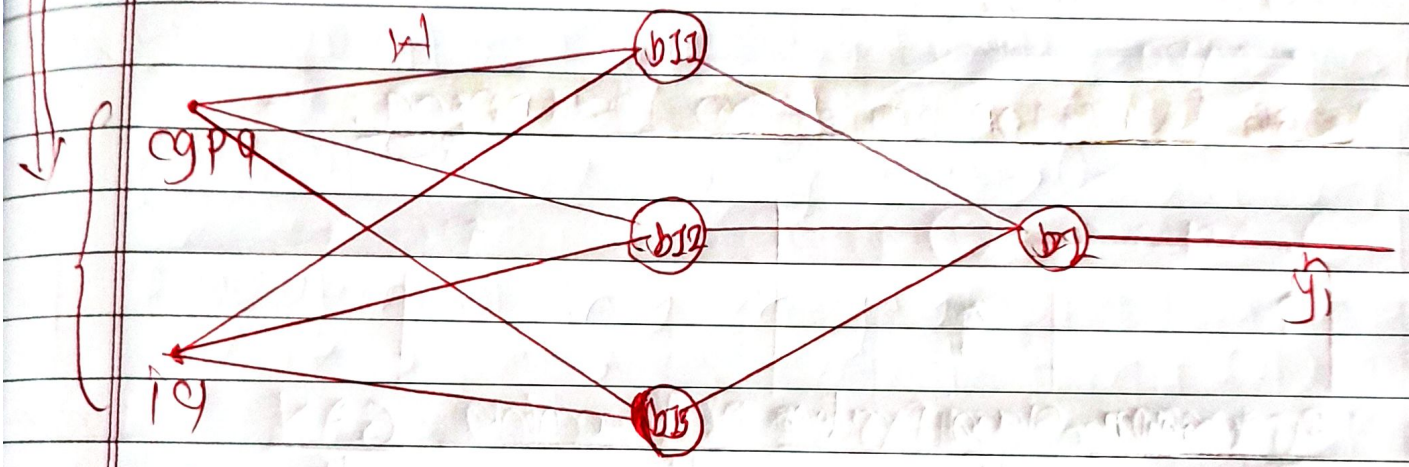
Why is loss function important?

- [You can't improve what you can't measure]
  - Peter Drucker
- This line by Peter Drucker perfectly explain importance of loss function in machine learning/ deep learning.

classmate
Date
Page

2) What is role of loss function in deep learning?

cgpa	iq	package(lpa)
7.1	83	3.2
8.5	91	4.5
6.3	102	6.1
5.1	87	2.7



known,

- To role we will discuss backpropgation for now,
- We put first student 'cgpa and iq' here and we will put this in neural network by using forward propogation it will predict for first student.
- Let's say for first student it comming out to be 3.7.
- Since this is regression problem, we will

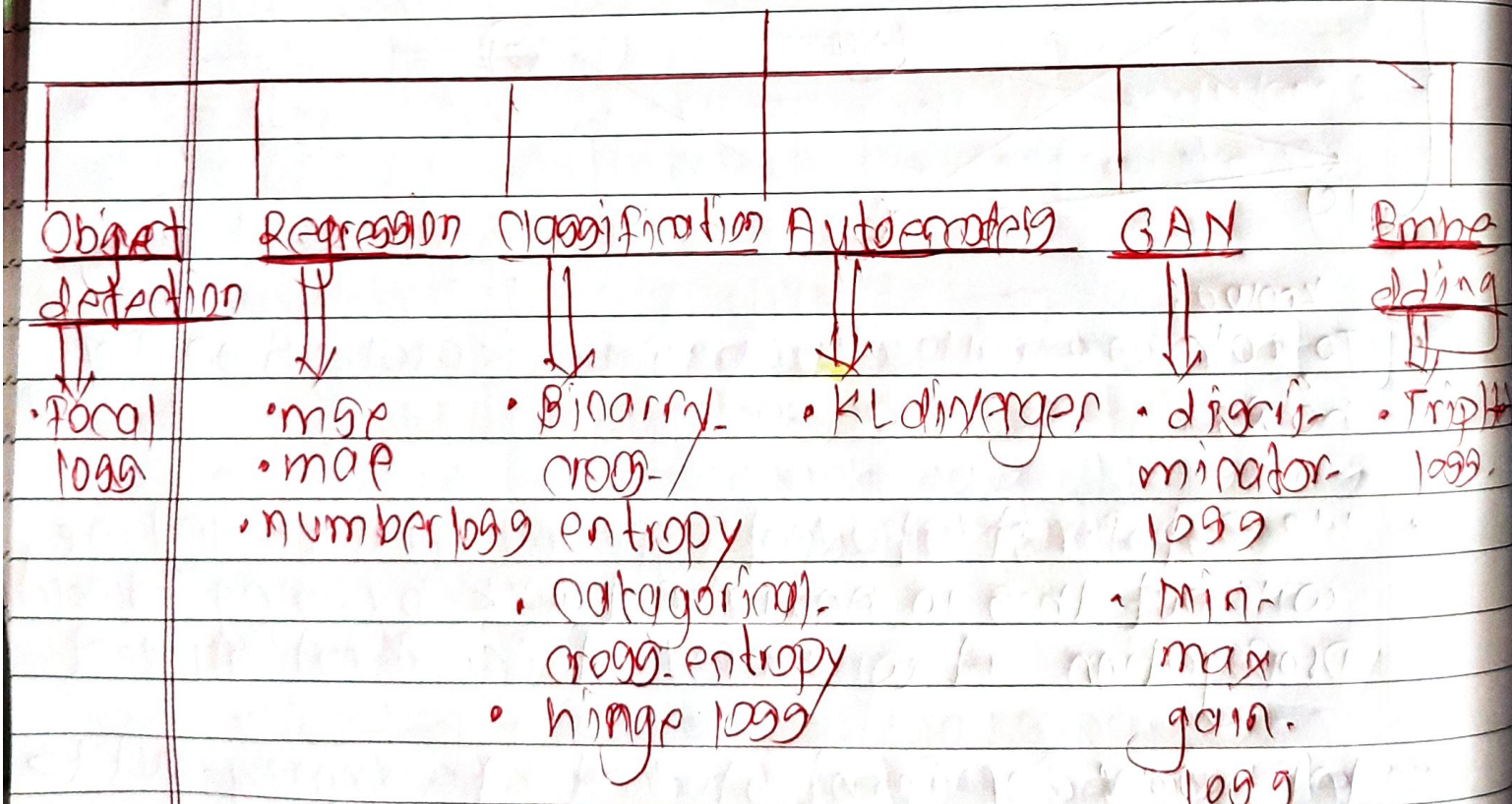
use, mean squared error.

$(3.2 - 3.7)^2 = (0.5)^2 = 0.25$

- Apply gradient descent on it and then update weight and biases
- We will do this for every point, ~~data~~ this is backpropogation.

Idea of updating weight and biases is to reduce loss.

Loss function in deep learning.

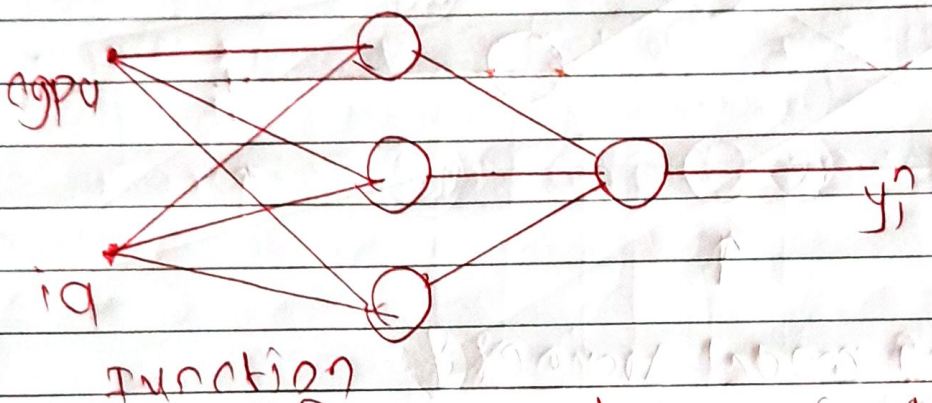


- We can create our custom loss function too, Keras allow to do that.

Loss function and cost function difference

- This is our data set,

cgpa	iq	package	Prediction
6.3	100	6.3	6.1
7.1	91	4.1	4
8.5	89	3.5	3.7
9.2	102	7.2	7



Function

- Loss ~~(student)~~ calculate on single training example

⇒ For first row loss function would be,

$$(y_i - \hat{y_i})^2 = (6.3 - 6.1)^2 \Rightarrow \text{loss function}$$

- When we calculate for single row, this is known as loss function

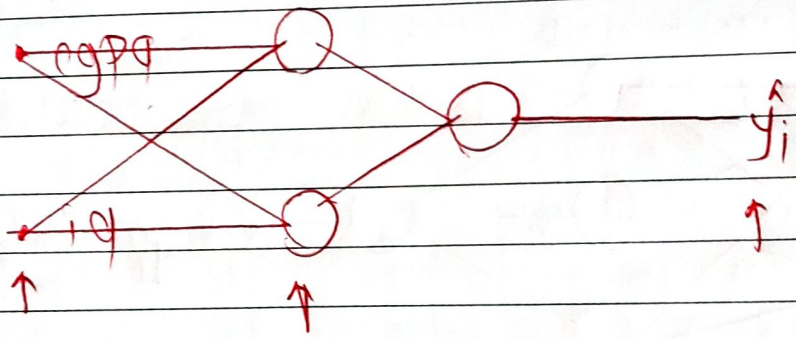
- When we calculate on batch it will become cost function

- **Cost function**

$$\frac{1}{n}\sum(y_i - \hat{y}_i)^2$$

Mean squared error

Also known as, squared Loss/ L2 loss, Let's study in context of deep learning.



- **How this model trained?**
  - We insert rows one by one and we get $\hat{y}_i$ (forward propagation)
  - After that we check how off is this prediction, By using loss function

$$(y_i - \hat{y}_i)^2$$

  - Based on this loss function we update weights and biases
  - We will do this for every point, Idea is to minimize value of loss function.

classmate Date Page

- We will discuss advantages of MSE also disadvantages of it to know which dataset it could apply on.

- ~~Why just $(y_i - \hat{y_i})$ instead of $(y_i - \hat{y_i})^2$~~

  - Because of squaring negative values get down
  - Total error reduced because of minus value, we want all error to be added not subtract.

- After squaring, $(y_i - \hat{y_i})^2$ becomes quadratic,

  $(\text{true}(y_i) - \text{predictive}(\hat{y_i})) \Rightarrow 1$ unit

  After squaring → 1 unit.

  2 unit → After squaring → 4 unit

  3 unit → After squaring → 9 unit

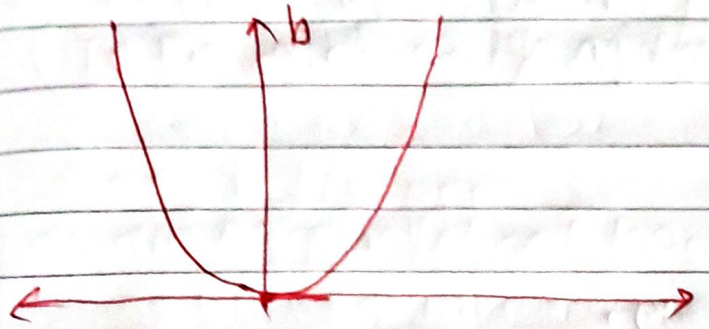
- Those points making more error/which are very far from actual line leads to change more weight and bias drastically.

  ~~Wrongly classified~~ / ~~$(y_i - \hat{y_i})^2$~~ too much } ⇒ makes more changes in W, b

classmate
Date
Page

Advantages

- Easy to interpreate
  Because of quadratic nature this function is always differentiable.



- There is only 'I' local minima.

Disadvantages

- Error unit is square.
- Not Robust to outlier.

Condition to apply MSE in deep learning.

- Last neuron activation function always should be 'linear'.

Mean Absolute error (MAE)

- Also known as 'L1 loss'
- $L = |y_i - \hat{y}_i| \implies$ Loss function
- $L = 1/n \Sigma |y_i - \hat{y}_i| \implies$ cost function.

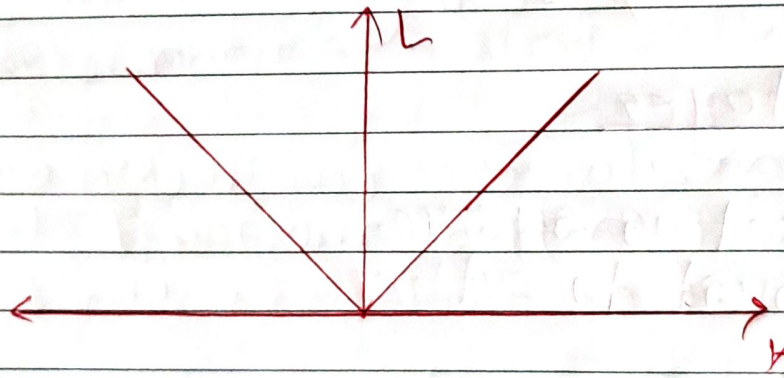
classmate
Date
Page

Advantages

- Easy to understand.
- Unit same as unit of y.
- Robust to outlier (so error cannot increase / punish, b drastically as in quadratic loss function i.e. MSE).

Disadvantages

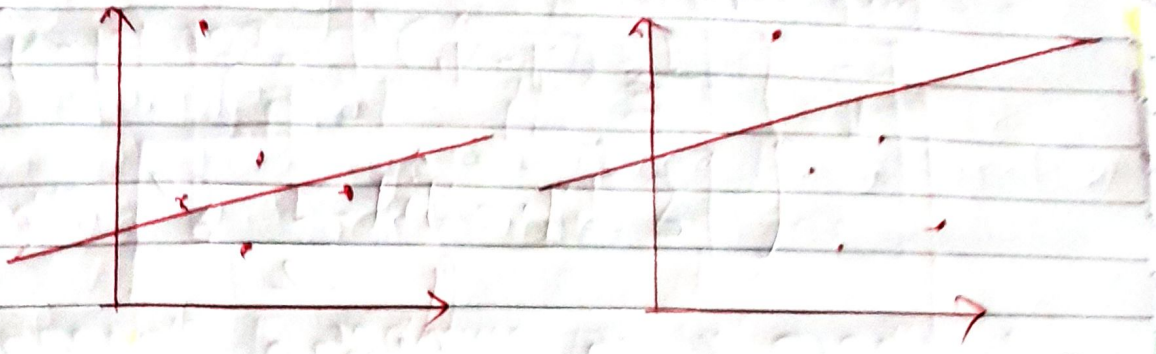
- Graph of MAE is,



- This graph is not differentiable, so we cannot apply gradient directly we have to calculate sub-gradient first.
- Calculating sub-gradient is computationally expensive.

Huber loss

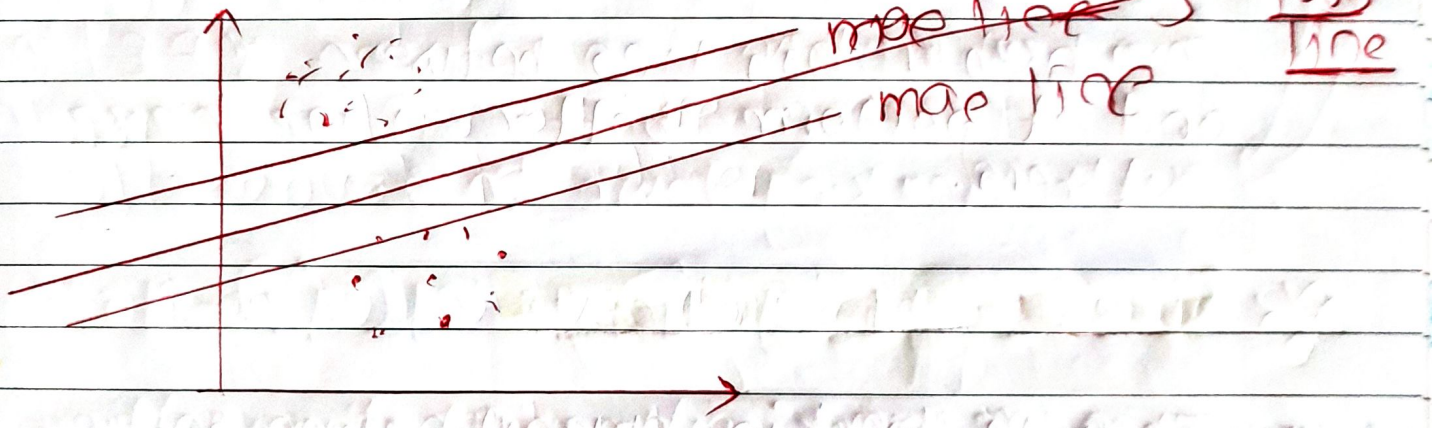
- mse cause drastic change in weights & biases and mae cause less changes in weight & biases



mae to outlier

mse to outlier

- Lets we have this kind of data that it have 25% of outlier (so that 25% is not a outlier firstely).



- In this case we want middle line which is we get after updating weight and biases by using Huber loss as Loss function.

- If outlier not present ⟹ behave like MSE.
  If outlier present ⟹ behave like MAE.

- mathamatically above is,

$$L = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y - \hat{y}_i| \le \delta \\ \delta |y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

(first branch → mse; second branch → mae)

- 'δ' parameter used to determine which point is outlier or not.
  'δ' by adjusting this parameter we can check performance.
- This huber loss lies between MAE / MSE and it perform better if we have more outliers.

Binary Cross Entropy

- This is used in classification setting
- It will be only used when you have only two classes.
- Loss function $\Rightarrow -y \log(\hat{y}) - (1-y)\log(1-\hat{y})$

  $\hat{y} \Rightarrow$ neural net architecture prediction
  $y \Rightarrow$ Actual value / taget.
- Activation function for last neuron is sigmoid

classmate
Date
Page

cost function ⟹ ~~=y log~~

$$-\frac{1}{n}\left[\sum_{i=1} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\right]$$

- After calculating loss we updates weight and biases by using gradient descent.

Advantages

- This loss function is differentiable (we can apply gradient descent).

Disadvantages

- Multiple minima could present.
- This is not intuitive.
- Some people called it log-loss too.

Categorical cross Entropy

- Used in classification scenarios, only when you have multi class classification problem.
- ~~This our set up,~~ This same loss function used in softmax regression too.
- To calculate above for single student, this is

Our formula,

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

- Where 'k' is number of classes in data
- This is our set up,

capital	city	placed
8	80	yes
6	60	no
7	70	maybe

(Classes after one-hot encoding)

yes	no	maybe
1	0	0
0	1	0
0	0	1

- After breaking down this,

$$L = -y_1 \log(\hat{y}_1) - y_2 \log(\hat{y}_2) - y_3 \log(\hat{y}_3)$$

- When we work with multi class classification some changes occurs, in output layers we use no. of neurons = no. of class.
- Activation function for output layer is soft max, this is formula.

$$f(z_1) = \frac{e^{z_1}}{e^{z_1} + e^{z_2} + e^{z_3}}$$

$z_1 \Longrightarrow$ yes
$z_2 \Longrightarrow$ no
$z_3 \Longrightarrow$ maybe

- $f(z_1) + f(z_2) + f(z_3) = 1$

- This is loss function for single row,

$$L = -y_1 \log(\hat{y_1}) - y_2 \log(\hat{y_2}) - y_3 \log(\hat{y_3})$$

- For first row, all neuron's which basically three neuron output is,

$$[\underset{\downarrow}{0.2} \quad \underset{\downarrow}{0.3} \quad \underset{\downarrow}{0.5}]$$
$$\text{yes} \quad \text{no} \quad \text{maybe}$$

$$[\underset{\downarrow}{1} \quad \underset{\downarrow}{0} \quad \underset{\downarrow}{0}] \Longrightarrow \text{true lable}$$
$$y_1 \quad y_2 \quad y_3$$

- We apply this loss function,

$$L = -y_1 \log(\hat{y_1}) - 0 - 0$$

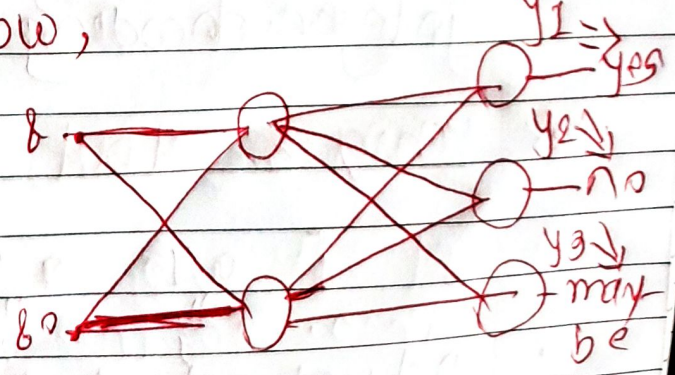
$$L = -1 \log(0.2) \Longrightarrow$$ We update's weight & biases accordingly.

- This is loss for first row.

- Similarly for second row,

prediction $[0.3 \quad 0.6 \quad 0.1]$

$$[\underset{\downarrow}{0} \quad \underset{\downarrow}{1} \quad \underset{\downarrow}{0}]$$
$$y_1 \quad y_2 \quad y_3$$



- Loss becomes,

$$L = -y_2 \log(\hat{y}_2)$$
$$= -1 \log(0.6)$$

- similarly for third point/row,

$$L = -y_3 \log(\hat{y}_3)$$

After that we update weight biases accordingly

Sparse categorical cross Entropy

cgpa	iq	placed		After integer encoding
7	70	yes	⇒	1
8	80	no	⇒	2
6	60	maybe	⇒	3

- If we use categorical cross entropy we use one-hot encoding
- For sparse categorical cross entropy we use integer encoding
- This is output three neurons for first row,

$$[\underset{\downarrow}{0.1} \quad \underset{\downarrow}{0.4} \quad \underset{\downarrow}{0.5}]$$

- Actual output for first row is (yes = 1)

- so loss becomes,

Forgot about them.

$$L_1 = -y_1 \log(\hat{y}_1)$$
$$= -1 \log(0.2) \Rightarrow$$

$-y_2 \log(\hat{y}_2)$ & $-y_3 \log(\hat{y}_3)$

After this weights are updated accordingly.

- Similarly for second and third row,

$$L_2 = -y_2 \log(\hat{y}_2) = -1 \log(\hat{y}_2)$$
$$L_3 = -y_3 \log(\hat{y}_3) = -1 \log(\hat{y}_3)$$

- If we have too much classes we should use sparse categorical entropy, since other calculation not needed and we don't have to one-hot encode it.

$-y_3 \log(\hat{y}_3)$ & $y_2 \log(\hat{y}_2)$

$= -y_2 \log(\hat{y}_2)$ & $y_1 \log \hat{y}_1)$

} Forgot about them.

- Cost function,

$$C = \frac{-1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} y_{ij} \log(\hat{y}_{ij})$$

Summary

- Regression

- use mae
- use mae if outlier present. } for combinat outlier and normal use number loss.

- **classification**
  - Binary classification ⟹ Binary cross entropy.
  - Multi-class classification ⟹ categorical cross entropy. (less category).
    - If we have more category/classes then use sparse categorical cross entropy.