Loss functions in Deep learning

→ Loss function is a method of evaluating how well your algorithm is modelling your dataset.

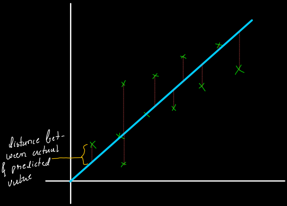
Loss function
- high value = poor model performance
- smaller value = great performance.

Why loss function is important..?

You can't improve what you can't measure

-Peter Drucker

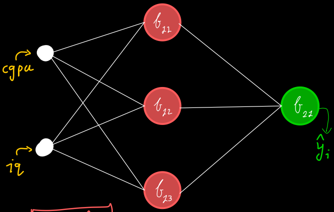
→ in machine learning.



mean-squared-error $= (y_i - \hat{y}_i)^2$

→ 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



$$L = \boxed{y_i - \hat{y}_i}$$

Steps

↳ Take first row of dataset [[7.1, 83]] and pass it thro... a...

NN

- At last, we will get some prediction ($\hat{y}_1$) Suppose $\hat{y}_1 = 3.7$
- Calculate loss based on prediction

  $(y_1 - \hat{y}_1) = (3.2 - 3.7) = -0.5$
- Using backpropogation & gradient descent algorithm, update parameters.
- Select next record & repeat the process.

-> Value of weights are optimal at which loss is minimum.

Types of loss functions in DL

Regression
1. Mean-squared-error
2. Mean-absolute-error
3. Huber-loss

Classification
1. Binary Cross Entropy
2. Categorical Cross Entropy
3. Hinge Loss

Autoencoders
1. KL divergence

GAN
1. Discriminator-loss
2. MinMax gan loss

Embedding
1. Triplet-loss

Object-detection
1. focal-loss

-> Even we can create our own loss-functions in keras library

Loss function VS Cost function

cgpa	iq	package (y)	pred ($\hat{y}$)
6.3	100	6.3	6.1
7.1	91	4.1	4
4.5	83	3.5	3.7
9.2	202	7.2	7

loss function -> On single training data

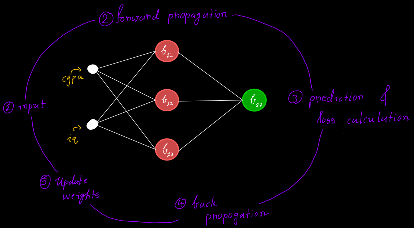
$(y_i - \hat{y}_i)^2 = (6.3 - 6.1)^2$

Cost function -> On entire training data

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

$\frac{1}{4}[(6.1-6.3)^2 + (4.1-4)^2 +$ [illegible] $5 - 3.7)^2 + ($ [illegible] $- 7)^2]$

1. MSE (Mean Squared Error)



$$\boxed{MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

-> As predicted values are far from original value, the error will be magnified.
-> Higher error will have drastic impact on weights.

• Advantages
-> Easy to interpret
-> Differentiable (Visulisable)
-> Only one local minima

• Disadvantages
-> Squared error unit
-> Not robust for outliers.

-> in-order to use MSE, the activation function of o/p layer must be "linear".

2. MAE (Mean Absolute Error) -> L1 loss

$$\boxed{Loss = |y_i - \hat{y}_i|}$$

$$\boxed{Cost = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|}$$

- Advantages
  - → Intuitive to understand
  - → Same unit
  - → Robust to outliers

- Disadvantages
  - → Not differentiable (Need to calculate subgradients)

3. huber loss

→ if in our dataset we have 10% or 15% outliers, they are not actually outliers.

→ In case of too many outliers in data, huber loss is here to help.

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

δ = hyperparameter

→ in-short, if your dataset has outlier then huber loss will behave like MAE, otherwise it will behave like MSE.

→ And by combining these, we will get best from both.

4. Binary Cross Entropy (log loss)

→ for binary class classification

$$\boxed{Loss = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})}$$

y = Actual value

$\hat{y}$ = predicted value

→ for Binary Cross Entropy, Activation function at output layer will always be sigmoid.

$$\text{cost} = \frac{-1}{n}\left[\sum_{i=1}^{n} Y_i \log \hat{Y}_i + (1-Y_i)\log(1-\hat{Y}_i)\right]$$

cgpa	iq	placement (y)
8	8.0	1
7	7.0	0
6	6.0	0

for [8, 8.0] suppose $\hat{y} = 0.73$

$$\text{Loss} = -1 \log(0.73) - (1-1)\log(1-0.73)$$

$$= 0.13$$

→ Repeat this for all data points

- Advantage
  → Differentiable
- Disadvantage
  → Multipal local minima
  → Intuitive

5. Categorical Cross Entropy (used in softmax regression)

→ for multi-class classification

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

k = number of o/p classes in dataset

→ for more than 3 output classes.
→ Number of neurons in o/p layer = number of o/p classes.
→ Activation function at o/p layer will be softmax

→ In Categorical - Cross_Entropy we encodes o/p classes using OneHotEncoder.

→ In Sparse-categorical-cross-entropy we encodes o/p classes using Categorical Encoder.

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

Conclution

Regression
- → No-outliers → MSE
- → Outliers → MAE
- → Mixed → Huber loss

Classification
- → Binary Classification → BCE
- → Multi-class classification
  - → few categories → CCE
  - → More categories → SCCE

Created with Notewise