Artificial Neural Network - II

Back propagation

Looking back

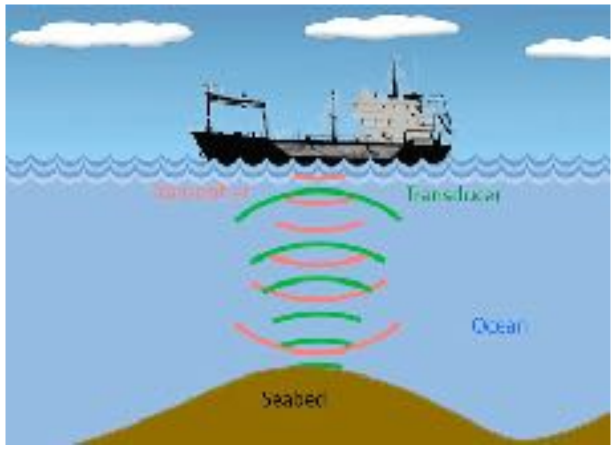
1. Artificial Neural Networks
2. Components of neural network
3. Feed forward in neural network
4. Representing NN with matrix
5. Numpy and tensorFlow implementation of Feed Forward NN

Today's topic

1. Activation functions
2. Bias in neural network
3. Loss functions in neural networks
4. Optimizers in neural network
5. Back propagation in neural network

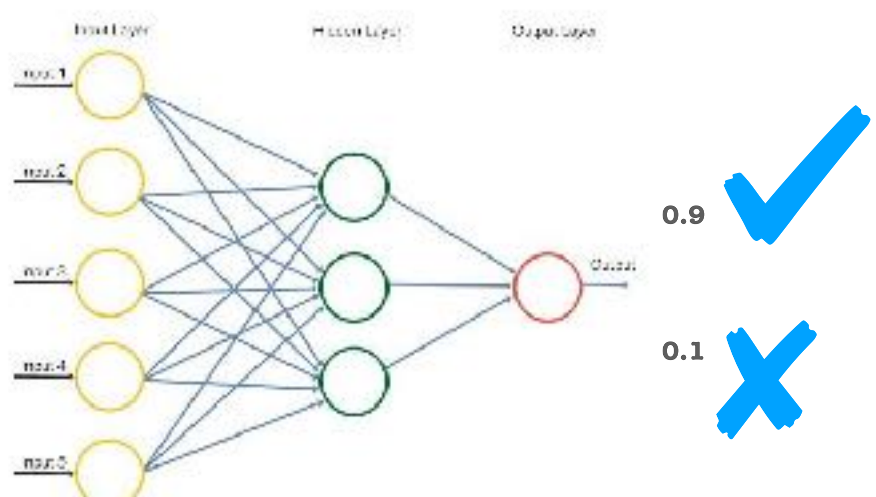
Dataset

Sonar dataset

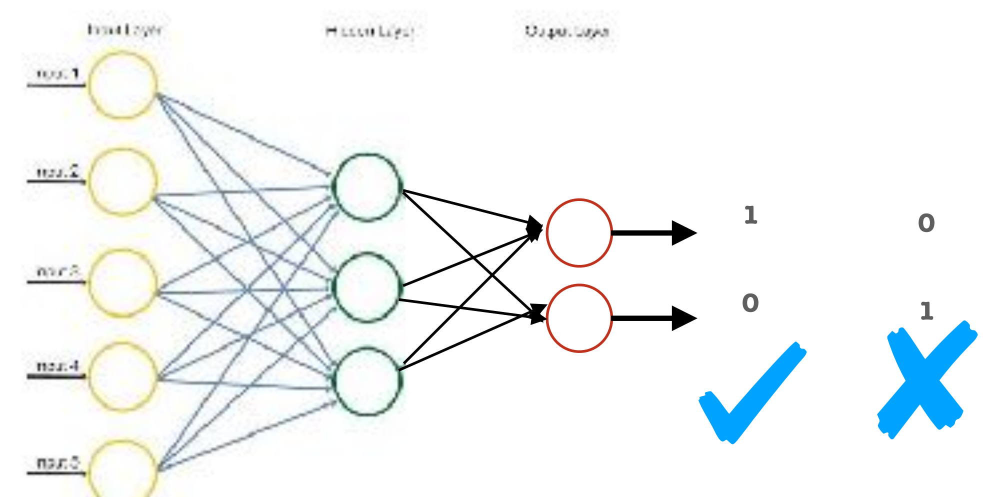


Glencore is a mining company. While mining in the ocean it was very difficult for them to identify whether floor is good for mining or not. They used traditional way of analysing the sonar data whenever there is a slight distress/ turbulence while mining. This procedure was very time consuming and sometimes led to the catastrophic disasters. Given the past sonar data about what they have observed how would you help them?

Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?
0.1	0.1	0.2	0.2	0.2	1
0.1	0.4	0.3	0.2	0.1	1
0.9	0.8	0.7	0.9	0.9	0



Regression problem



Classification problem

1 Activation functions

What are different activation function?

- Sigmoid or Logistic
- Tanh — Hyperbolic tangent
- ReLu -Rectified linear units

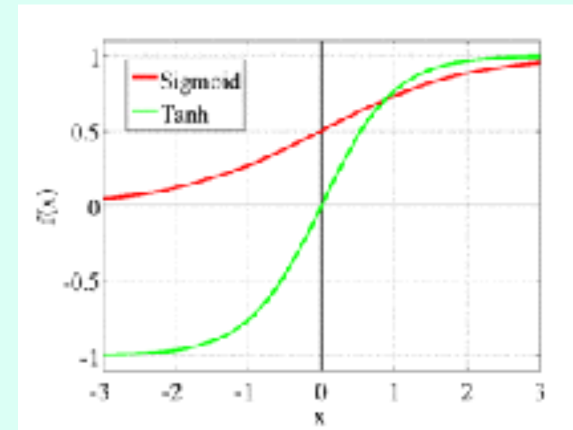
Why we need activation function?

The purpose of the activation function is to introduce non-linearity into the output of a neuron.

Why we need to introduce non - linearity?

The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

Tanh activation function

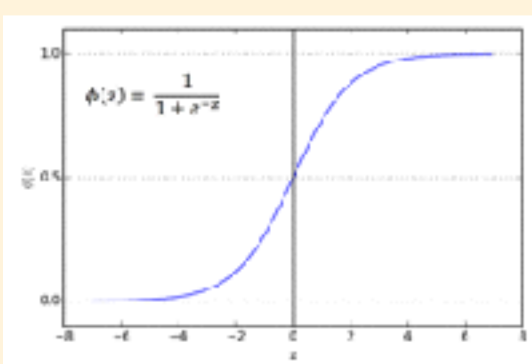


$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

It squashes values between -1 to 1

$$F(0.23) = \frac{e^{0.23} - e^{-0.23}}{e^{0.23} + e^{-0.23}} = 0.226$$

Sigmoid activation function

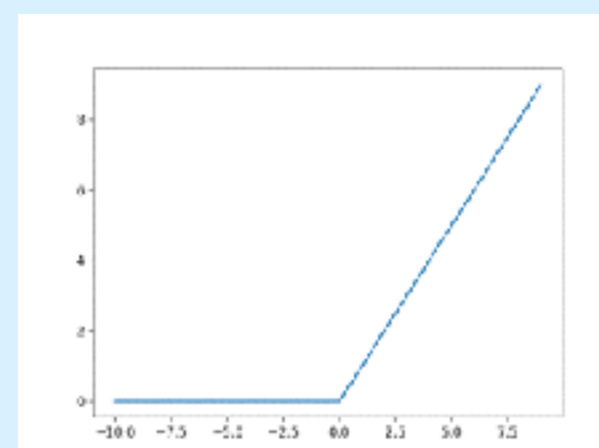


$$f(x) = \frac{1}{1 + e^{(x)}}$$

$$F(0.23) = \frac{1}{1 + e^{-0.23}} = 0.557$$

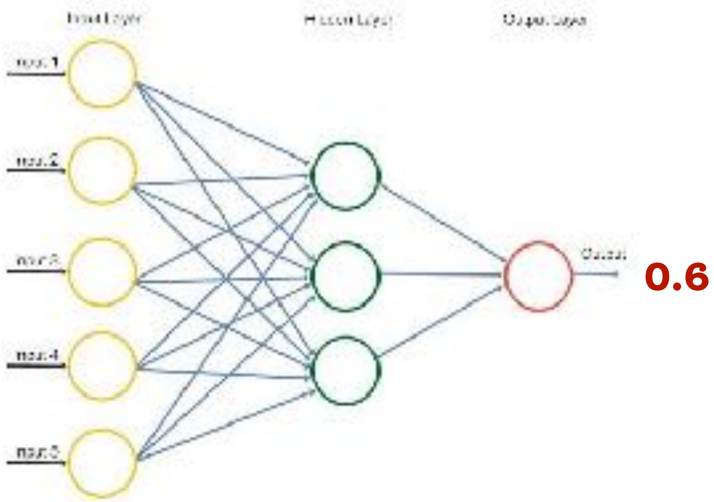
It squashes values between 0 and 1

ReLu activation function

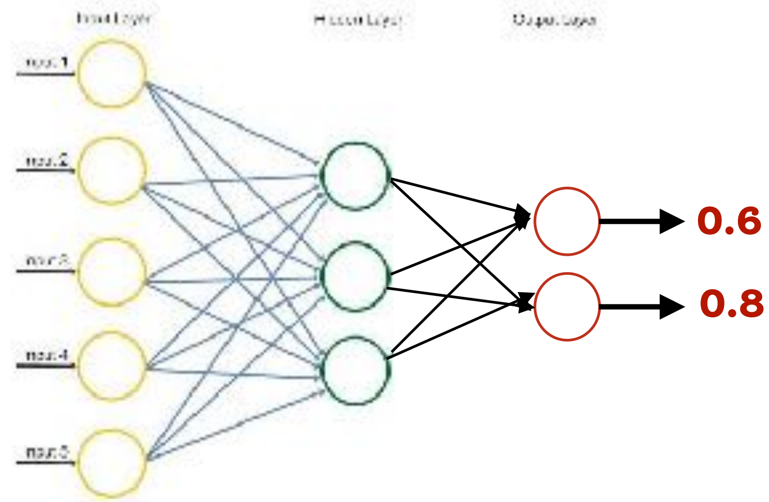


$$R(z) = max(0,\ z)$$

F(0.23) = max(0, 0.23) = 0.23



Regression problem



Classification problem

- **Sigmoid or Logistic**
- **Tanh — Hyperbolic tangent**
- **ReLu -Rectified linear units**

When we use sigmoid activation function, output of first neural network will be between 0 to 1

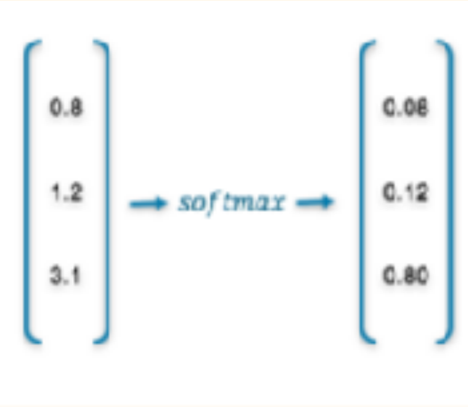
When we use sigmoid activation function, output of second neural network will be between 0 to 1 for both the neurons. One thing need to be observed here that when we add these 2 values corresponding to 2 neurons present in the output layer sum might be greater than 1.

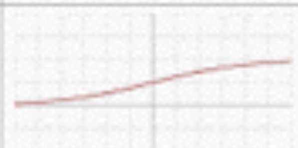
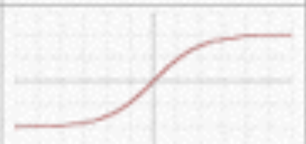
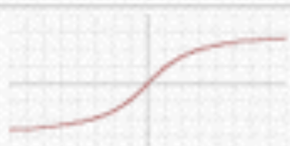
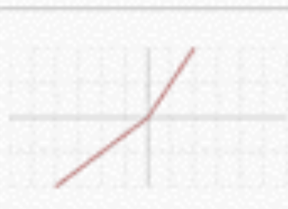
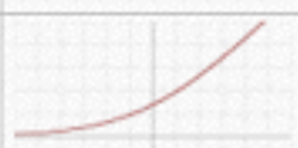
When we use tanh activation function, output of first neural network will be between -1 to 1

When we use tanh activation function, output of second neural network will be between -1 to 1 for both the neurons. One thing need to be observed here that when we add these 2 values corresponding to 2 neurons present in the output layer sum might be greater than 1.

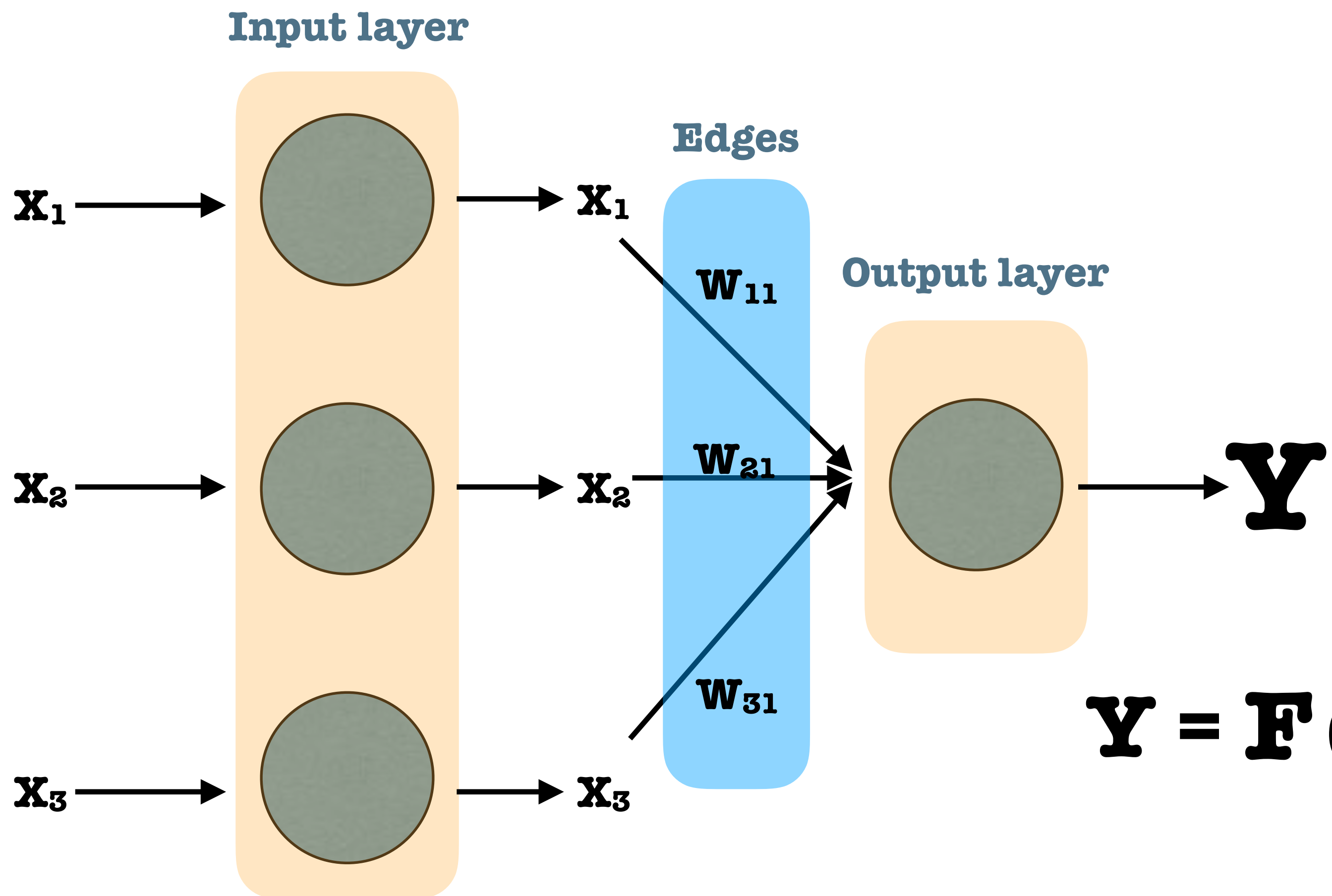
Softmax activation function

$$softmax(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1+e^{-x}}$

2 Bias in neural network



$$Y = F\left(\left[(X_1 * W_{11}) + (X_2 * W_{21}) + (X_3 * W_{31}) \right] + b_1 \right)$$

Why we need bias?

A simpler way to understand what the bias is: it is somehow similar to the constant b of a linear function

$$y = ax + b$$

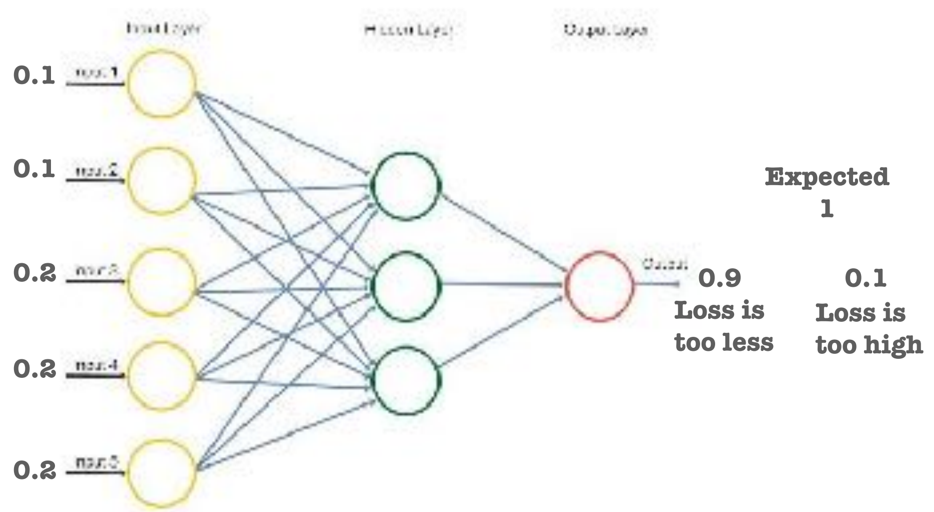
It allows you to move the line up and down to fit the prediction with the data better. Without b the line always goes through the origin (0, 0) and you may get a poorer fit.

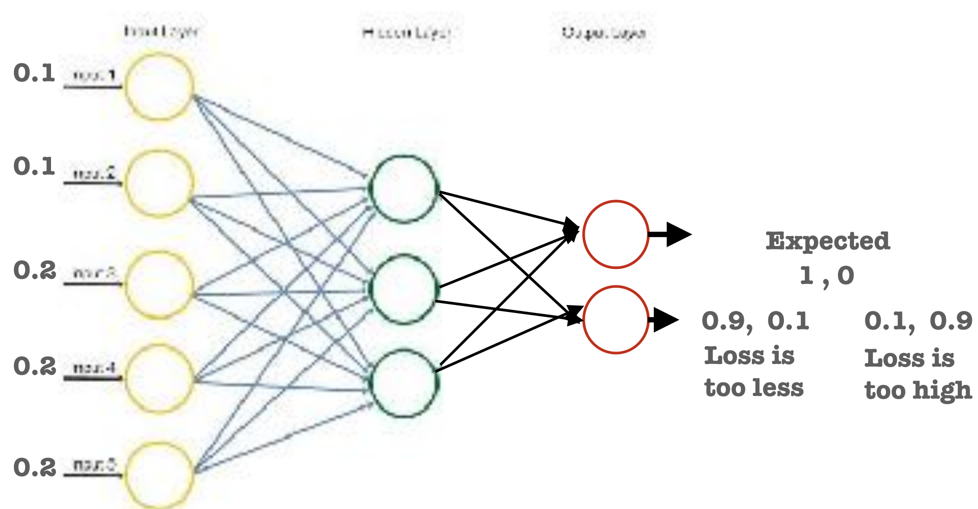
3 Loss functions in neural networks

What is loss function?

Loss function is a method of evaluating how well your algorithm models your dataset

Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?
0.1	0.1	0.2	0.2	0.2	1





If your predictions are totally off, your loss function will output a higher number. If they're pretty good, it'll output a lower number.

What are different loss function?

- Mean square error
- Likelihood loss
- Binary Log loss (Binary Cross entropy loss)
- Log loss (Categorical Cross entropy loss)
- Triplet loss
- Contrastive loss

When to use what activation function?

If the data has lot of missing values or if it is skewed then use the following loss functions:

- Binary Log loss (Binary Cross entropy loss)
- Log loss (Categorical Cross entropy loss)

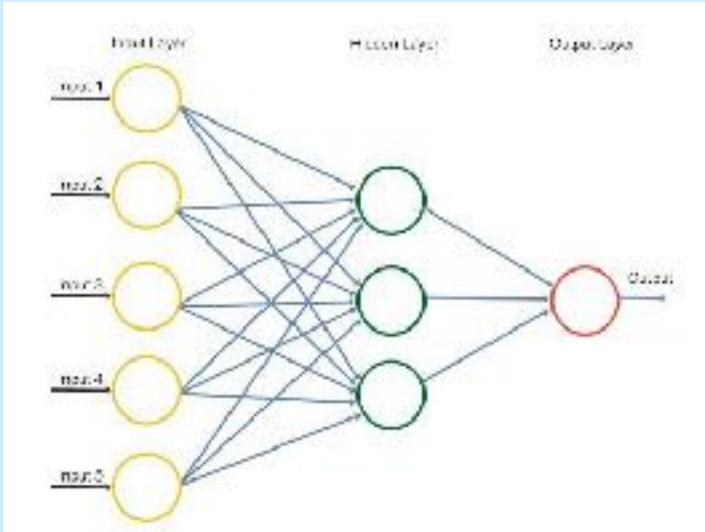
If the data is clean and you are very confident about it then you can use following loss functions:

- Mean square error
- Likelihood loss

What is the reason behind it?

Log losses penalise heavily for being confident about wrong classes. If a dataset has lot of missing values or skewness, we will be feeding lot of irrelevant data to the model. Model should closely examine the features even though we had replaced nans with averages and select good features.
If we have clean data model need not to look very close because if it looks very close there might be a chance of overfitting.

Mean square error



Regression problem

A regression predictive modelling problem involves predicting a real-valued quantity.

Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?	Predicted
0.1	0.1	0.2	0.2	0.2	1	0.8
0.1	0.4	0.3	0.2	0.1	1	0.6
0.9	0.8	0.7	0.9	0.9	0	0.4

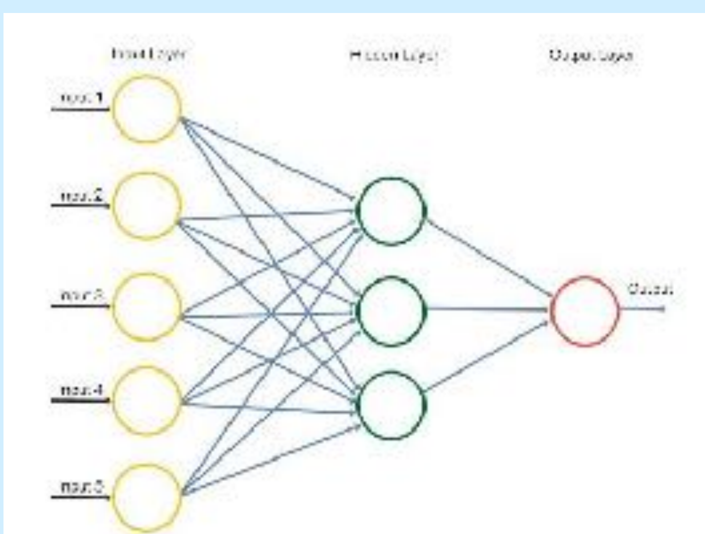
$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$*\, n$ is the number of data points
$*\, Y_i$ represents observed values
$*\, \hat{Y}_i$ represents predicted values

$$\frac{(1 - 0.8)^2 + (1 - 0.6)^2 + (0 - 0.4)^2}{3}$$

Example: Predicting price of a house $\dfrac{(1000 - 700)^2 + (600 - 900)^2 + (5000 - 1700)^2}{3}$

Likelihood loss



It is commonly used in classification problems especially when you are solving classification as a regression.

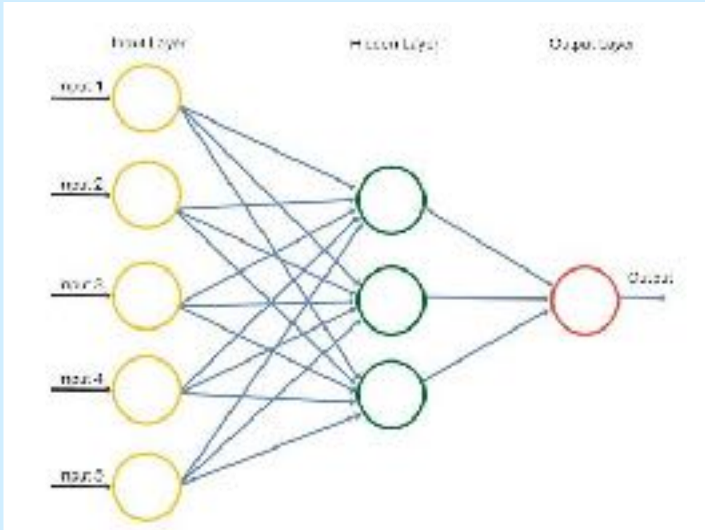
Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?	Predicted
0.1	0.1	0.2	0.2	0.2	1	0.8
0.1	0.4	0.3	0.2	0.1	1	0.6
0.9	0.8	0.7	0.9	0.9	0	0.4

P = predicted
A = actual
N = Number of samples

$$\prod_{i=0}^{N} \begin{cases} P & A==1 \\ 1-P & A==0 \end{cases}$$

Loss = 0.8 * 0.6 * (1 - 0.4)

Log loss or Binary Cross entropy



Binary cross entropy can be used to solve classification problem as regression

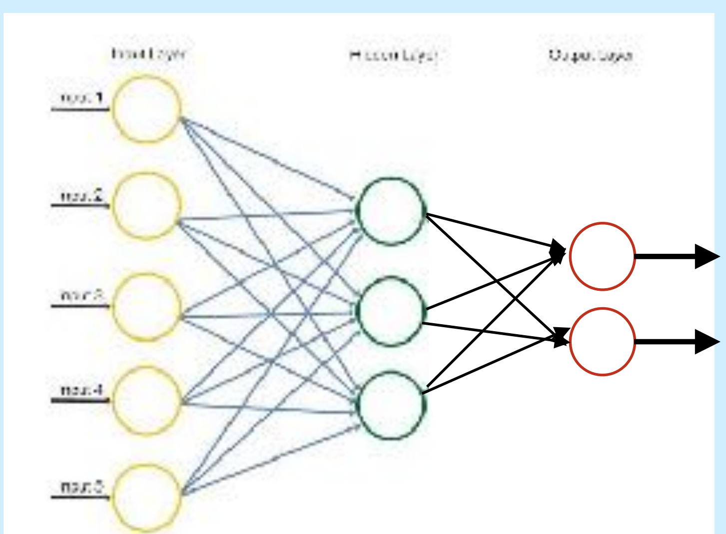
Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?	Predicted
0.1	0.1	0.2	0.2	0.2	1	0.8
0.1	0.4	0.3	0.2	0.1	1	0.6
0.9	0.8	0.7	0.9	0.9	0	0.4

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1-y_i) \cdot log(1-p(y_i))$$

$$\text{Loss} = -\frac{\left[\begin{array}{l} 1\log(0.8) + (1-1)\log(1-0.8) + \\ 1\log(0.6) + (1-1)\log(1-0.6) + \\ 0\log(0.4) + (1-0)\log(1-0.4) \end{array}\right]}{3}$$

It penalises heavily for being very confident and very wrong.

Categorical Cross entropy



Classification problem

It is commonly used in classification problems and it is used when you have softmax activation function in output layer

Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?	is_good	Predicted
0.1	0.1	0.2	0.2	0.2	1	1, 0	0.8, 0.2
0.1	0.4	0.3	0.2	0.1	1	1, 0	0.6, 0.4
0.9	0.8	0.7	0.9	0.9	0	0, 1	0.4, 0.6

$$\text{Loss} = -\frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N} Y_i * \log(P_j)$$

M = Number of samples
N = Number of neurons in output layer
Y_i = Target/Actual value
P_j = Predicted value

$$\text{Loss} = -\frac{\left[\begin{array}{l} 1\log(0.8) + 0\log(0.2) + \\ 1\log(0.6) + 0\log(0.4) + \\ 0\log(0.4) + 1\log(0.6) \end{array}\right]}{3}$$

4 Optimizers in neural network

When neural networks were introduced for the first time, loss has to be calculated for all the samples present in the dataset then back propagation had to be done.

Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?	Predicted	Loss
0.1	0.1	0.2	0.2	0.2	1	0	0.9
0.1	0.4	0.3	0.2	0.1	1	1	0.6
0.9	0.8	0.7	0.9	0.9	0	1	0.7
0.1	0.1	0.2	0.2	0.2	1	0	0.5
0.1	0.4	0.3	0.2	0.1	1	1	0.3
0.9	0.8	0.7	0.9	0.9	0	1	0.2
0.1	0.1	0.2	0.2	0.2	1	0	0.5
0.1	0.4	0.3	0.2	0.1	1	1	0.8
0.9	0.8	0.7	0.9	0.9	0	1	0.9
0.1	0.1	0.2	0.2	0.2	1	0	0.4
0.1	0.4	0.3	0.2	0.1	1	1	0.2
0.9	0.8	0.7	0.9	0.9	0	1	0.1

Avg Loss

Herbert Robbins

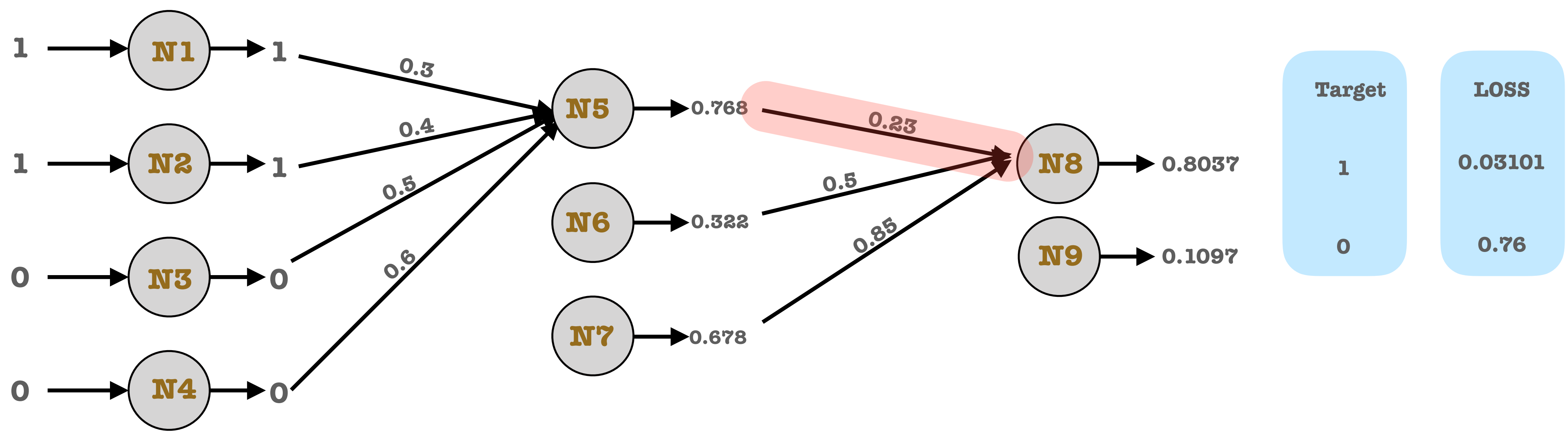
Attribute_1	Attribute_2	Attribute_3	Attribute_4	Attribute_5	is_good?	Predicted	Loss	
0.1	0.1	0.2	0.2	0.2	1	0	0.9	
0.1	0.4	0.3	0.2	0.1	1	1	0.6	
0.9	0.8	0.7	0.9	0.9	0	1	0.7	Avg Loss
0.1	0.1	0.2	0.2	0.2	1	1	0.7	
0.1	0.4	0.3	0.2	0.1	1	1	0.6	
0.9	0.8	0.7	0.9	0.9	0	0	0.7	Avg Loss
0.1	0.1	0.2	0.2	0.2	1	0	0.9	
0.1	0.4	0.3	0.2	0.1	1	1	0.6	
0.9	0.8	0.7	0.9	0.9	0	1	0.3	Avg Loss
0.1	0.1	0.2	0.2	0.2	1	1	0.9	
0.1	0.4	0.3	0.2	0.1	1	1	0.6	
0.9	0.8	0.7	0.9	0.9	0	0	0.3	Avg Loss

What are different optimisation techniques?

- Stochastic Gradient Descent
- Adagrad
- Adam
- RMSprop

5 Back propagation in neural network

Stochastic Gradient Descent



= sigmoid [(1*0.3) + (1*0.4) +(0*0.5) +(0*0.5) + b]
= sigmoid [(1*0.3) + (1*0.4) +(0*0.5) +(0*0.5) + 0.5]
= 0.768

weight_N5_to_N8 = weight_N5_to_N8 +(learning_rate * N8_loss * n5_output)
= 0.23 + (0.01 * 0.03 * 0.768)
= 0.2302304

Summary

1. Activation functions
2. Bias in neural network
3. Loss functions in neural networks
4. Optimizers in neural network
5. Back propagation in neural network