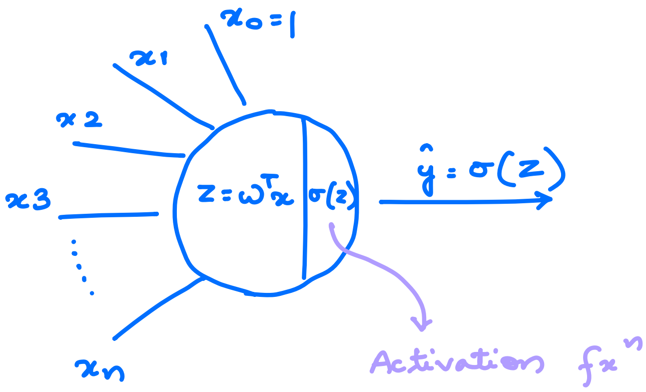
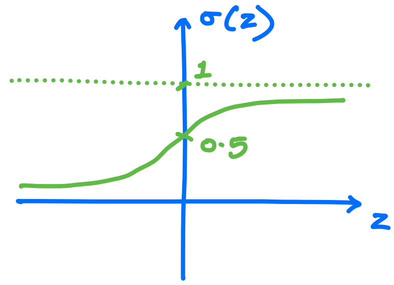
In this lecture we will see what is a Perceptron. Perceptron is a single layer neural network. We will be learning about the loss function and later on we will derive weight update rule.



$$z = w^T x = \begin{bmatrix} w^0 & w^1 & w^2 & \cdots & w^n \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

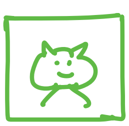
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

O/P is any no. b/w 0 and 1



Let us say we are doing a classification problem in which we want to predict if given image is of dog or cat.

dog → 0, cat → 1



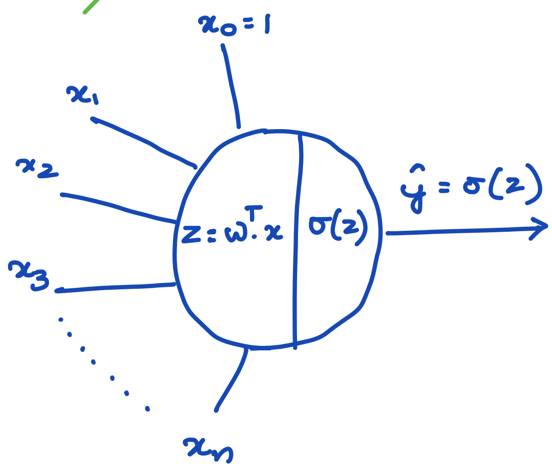
→ 0 or 1 ?

$\hat{y}$ lies in range 0 and 1.

If $\hat{y} = (0.8) \approx 1$ (Round off to 1)
$\hat{y} = (0.3) \approx 0$ (Round off to 0)

$\hat{y}$ tells which class a given input belongs to.

Perceptron will act as a linear classifier. It can do Binary Classification)



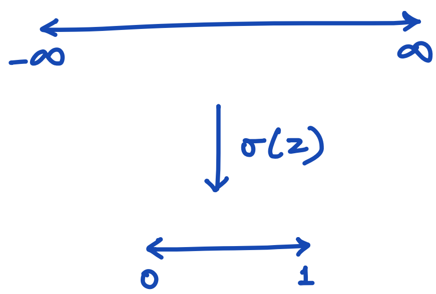
Basic Neural Network with only 1 node

Goal of learning Algorithm is to learn parameters w, if you get data of type 1 then output should be close to 1 and if you get data of type 0 then output should be close to 0.

If input is very large, lets say $z = 100$ then $\sigma(z) = 1$

$z = -200$ then $\sigma(z) = 0$

Activation fx^n is compressing your number line in the range 0 to 1.



$\sigma(z)$ tells the probability a given input belongs to which class.

Lets say $\sigma(z) = 0.7$, you are 70% confident that it belongs to class 1, you are 30% confident that it belongs to class 0.

$$P(y=1) \longrightarrow \sigma(z)$$

$$P(y=0) \longrightarrow 1-\sigma(z)$$

$$z = w^T \cdot x$$

This is how we can make predictions using a simple perceptron.

How to train a perceptron ?

In every ML Algo, we do the following:

- Model (Single Node of Perceptron, when we combine multiple such nodes we get a neural network which is called as multi layer perceptron)

- Loss

D	D	D	C	C

Algo predicts probability, with what probability it belongs to class Dog.

$P(y=1)$: $\hat{y}$:	0.7	0.8	0.8	0.3	0.4

with 0.3 probability it belongs to class Dog, it belongs to class cat with probability 0.7

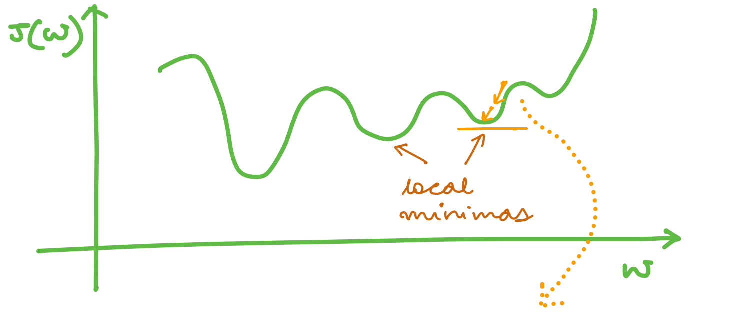
y :	1	1	1	0	0

$$J(w) \rightarrow MSE = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

→ Problem with this kind of loss function is it is a non-convex function.

$$\hat{y}^{(i)} = \sigma(z) = \sigma(w^T \cdot x)$$

$\hat{y}^{(i)}$ is a function of w.



If you start from this point and use gradient descent then you will be stuck in local minima. There are multiple local minimas and you will stuck inside it.

To overcome this difficulty we use another kind of loss called as log loss.

Log loss / Binary Cross Entropy

→ Proof for this is already covered in Maximum Likelihood Estimation for Logistic Regression.

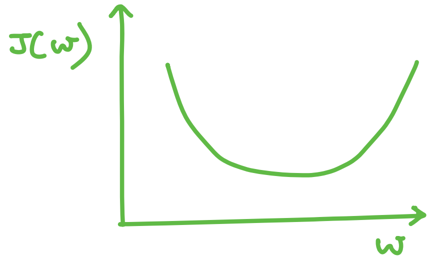
$$-\sum_{i=1}^{m} \left(y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)}) \right)$$

→ It is a convex fx^n

→ only 1 local and global minima

- if $y^{(i)} = 0$

  then loss $fx^n = \log(1 - \hat{y}^{(i)})$



- if $y^{(i)} = 1$

  then loss $fx^n = \log \hat{y}^{(i)}$

if predictions are not correct then we add log of misclassification.

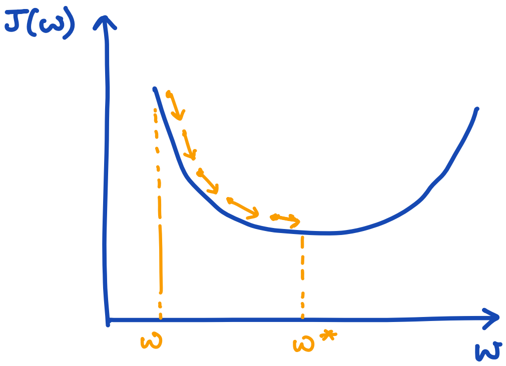
- if $y^{(i)} = 0$ and $\hat{y}^{(i)} = 0$

  then loss $fx^n = 0$ → that means when actual values matches with predicted value then you are adding 0 to loss.

- if $y^{(i)} = 1$ and $\hat{y}^{(i)} = 1$

  then loss $fx^n = 0$

- Weight update Rule



Start with some random w and update w as:

$$w = w - \eta \frac{\partial J}{\partial w} \Bigg\} \text{ Gradient update Rule}$$

-ve

+ve

$$J(w) = -\sum_{i=1}^{m} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

$$\hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)})$$

$J(w)$ is a fx^n of $\hat{y}^{(i)}$

$\hat{y}^{(i)}$ is a fx^n of $w^T x^{(i)}$