Introduction to Machine Learning in Python

Multi Layer Perceptrons
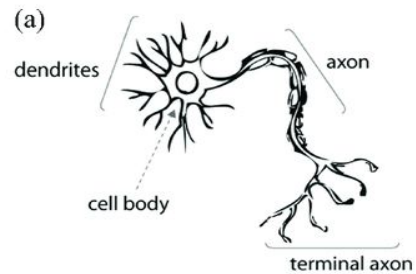
Instructor: Tugce Gurbuz

Aug 10th, 2023

Perceptron (1958)

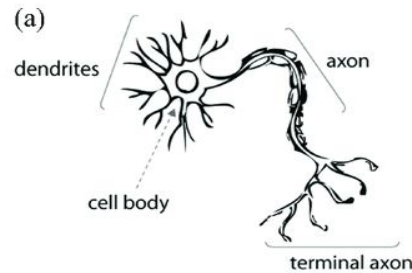
perceptron

(a)
dendrites / axon
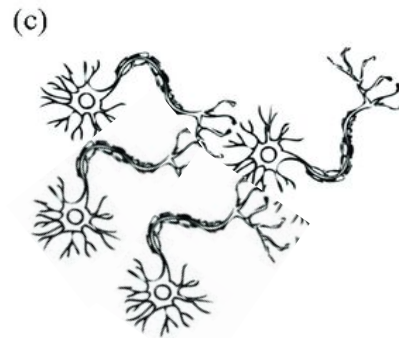cell body
terminal axon

**1: fire
0: not fire**

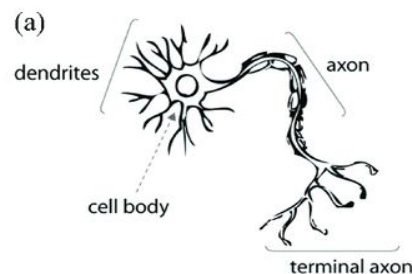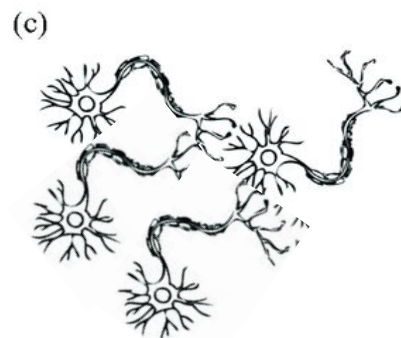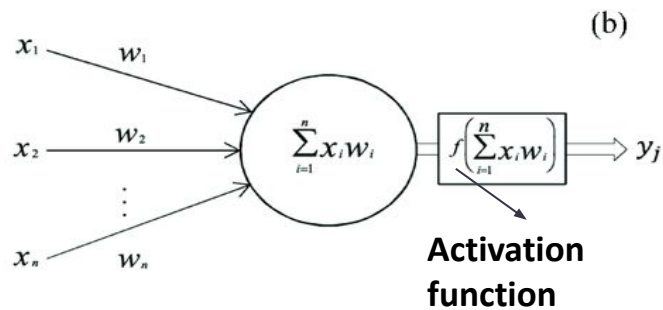Source

Perceptron (1958)

Source
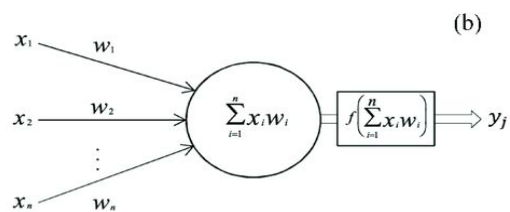

1: fire
0: not fire

Excitatory
and
inhibitory
connections

Perceptron (1958)

Source

(a) dendrites / axon / cell body / terminal axon

(c)

(b)

$$\sum_{i=1}^{n} x_i w_i \qquad f\left(\sum_{i=1}^{n} x_i w_i\right) \rightarrow y_j$$

$x_1 \rightarrow w_1$

$x_2 \rightarrow w_2$

$x_n \rightarrow w_n$

1: fire
0: not fire

Excitatory and inhibitory connections

Activation function

How does perceptron distinguish between 2 images?
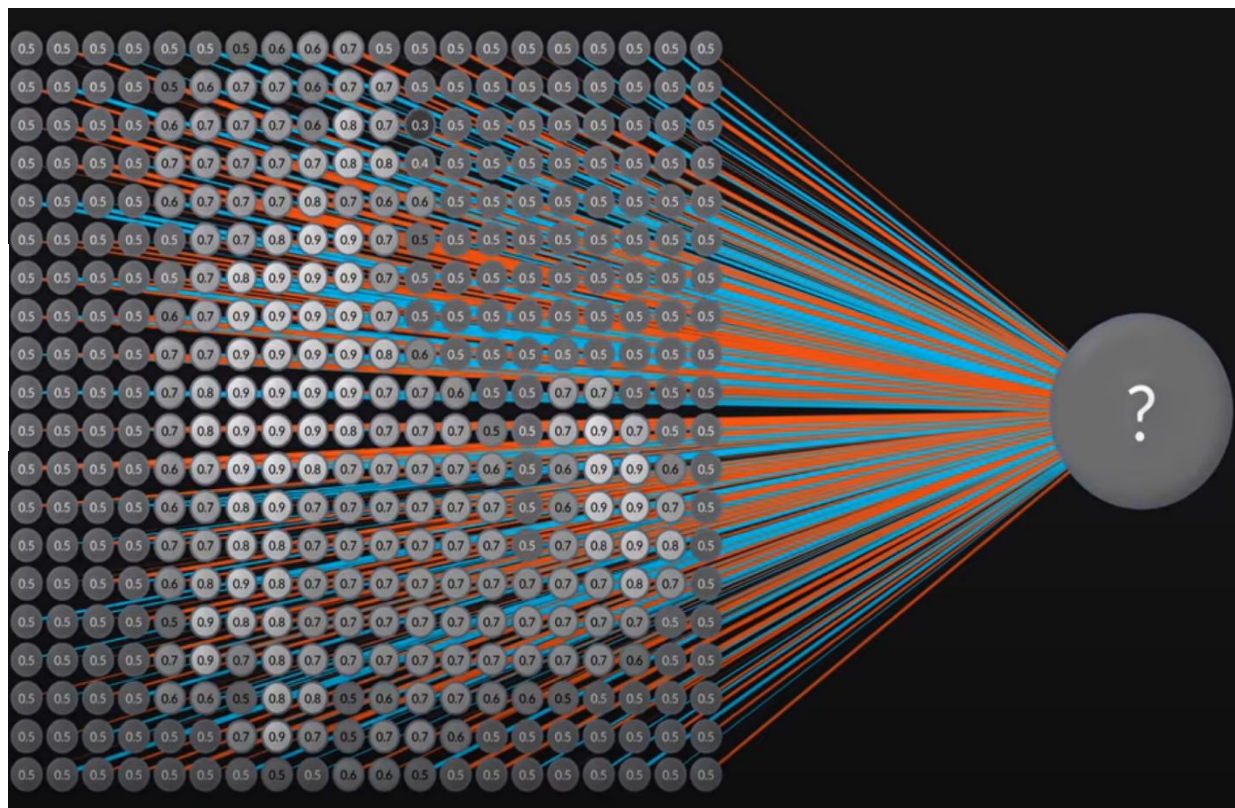
How does perceptron distinguish between 2 images?

How does perceptron distinguish between 2 images?
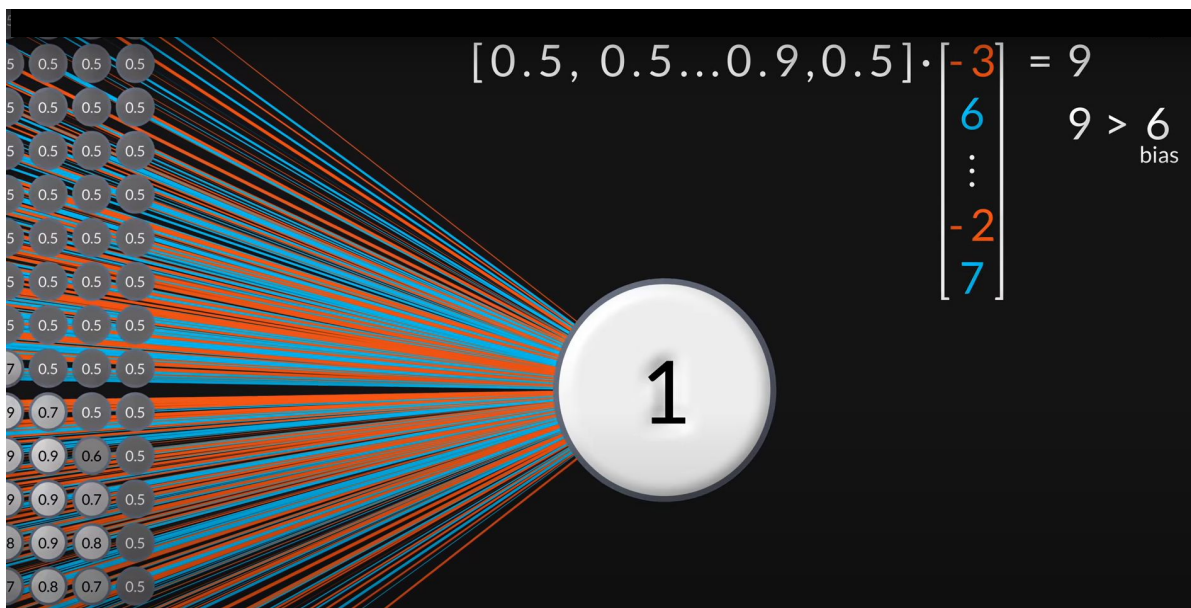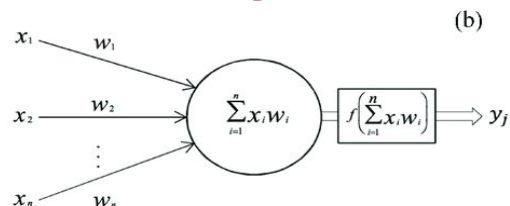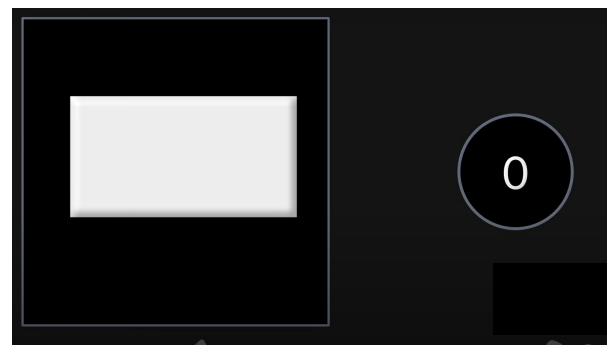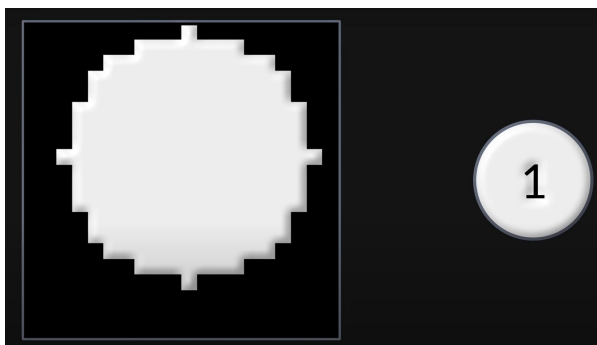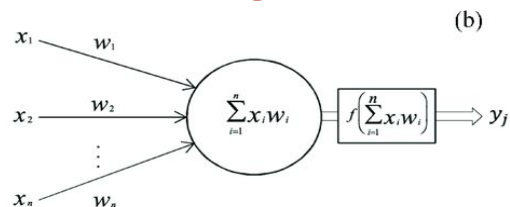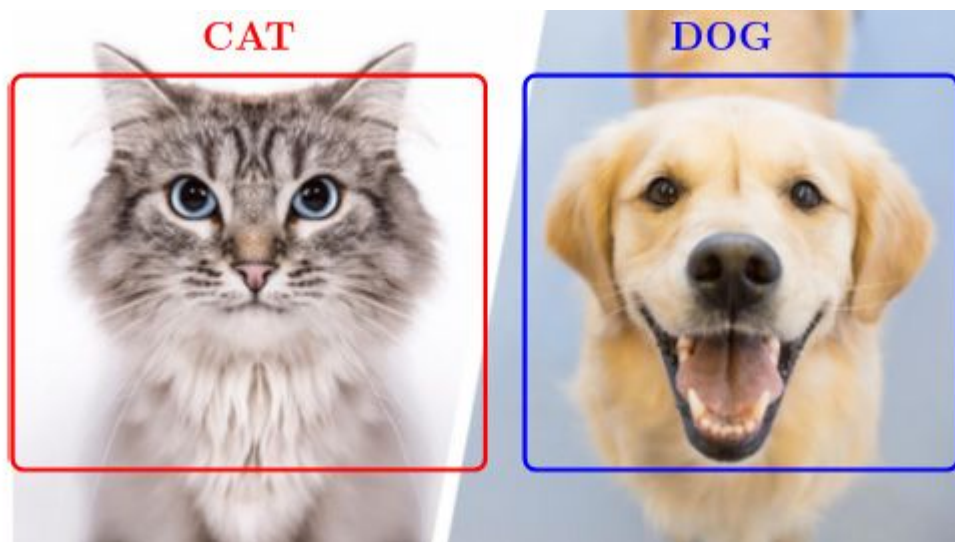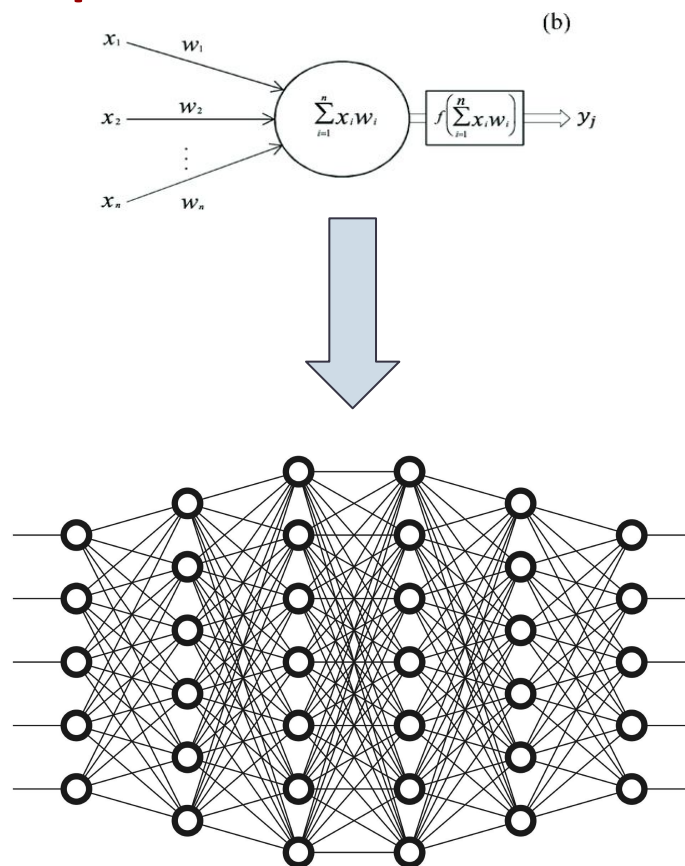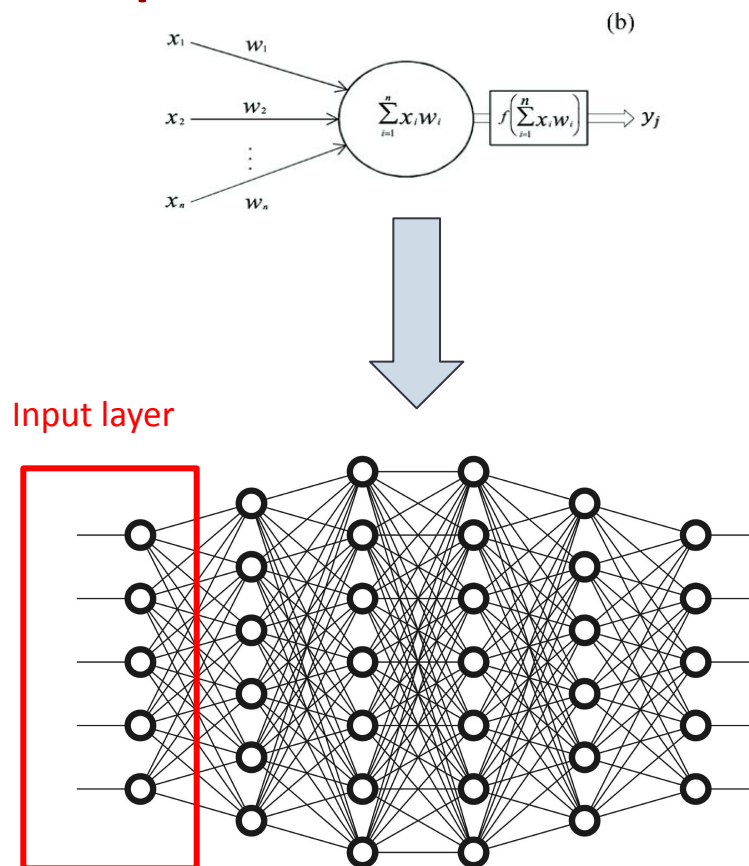
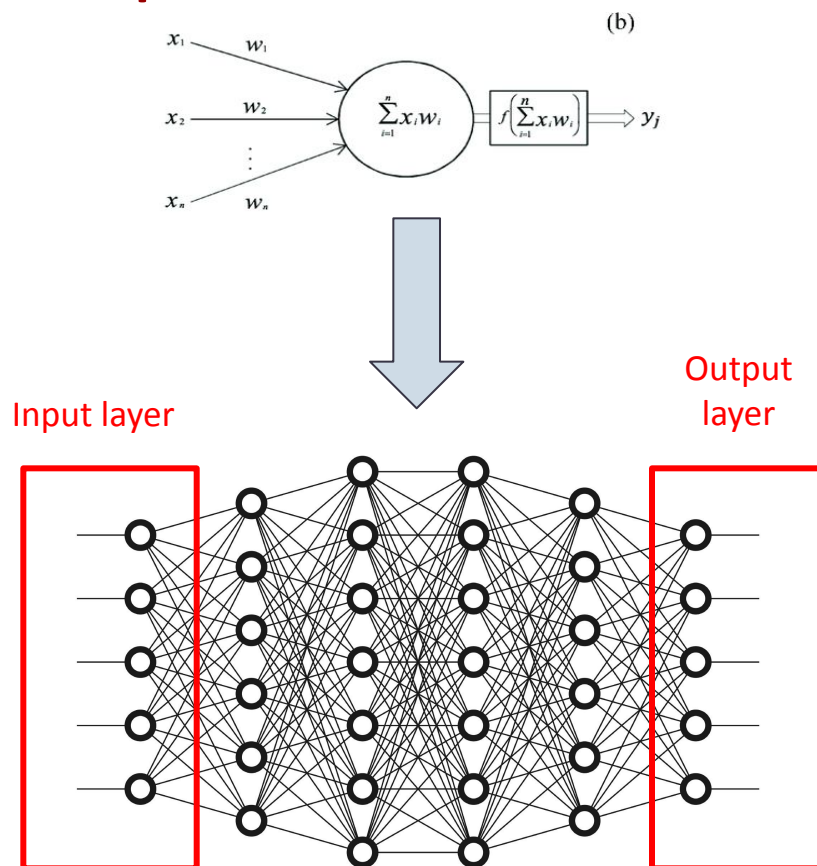# How does perceptron distinguish between 2 images?

Multi Layer Perceptron

Multi Layer Perceptron

(b)

Input layer

Multi Layer Perceptron

Multi Layer Perceptron

(b)

$$\sum_{i=1}^{n} x_i w_i$$

$$f\left(\sum_{i=1}^{n} x_i w_i\right)$$

x_1 w_1
x_2 w_2
x_n w_n

y_j

Input layer

Output layer

Hidden Layers
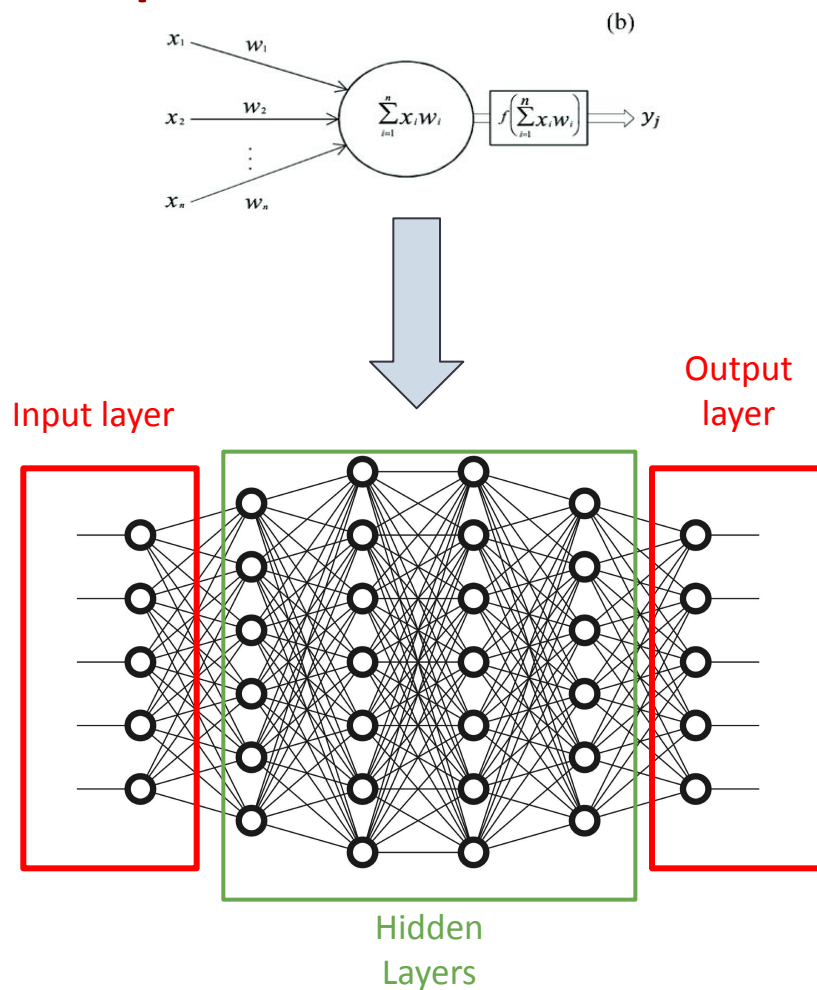
Multi Layer Perceptron

Activation Functions

Sigmoid

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

tanh

$\tanh(x)$

ReLU

$\max(0, x)$

Leaky ReLU

$\max(0.1x, x)$

Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
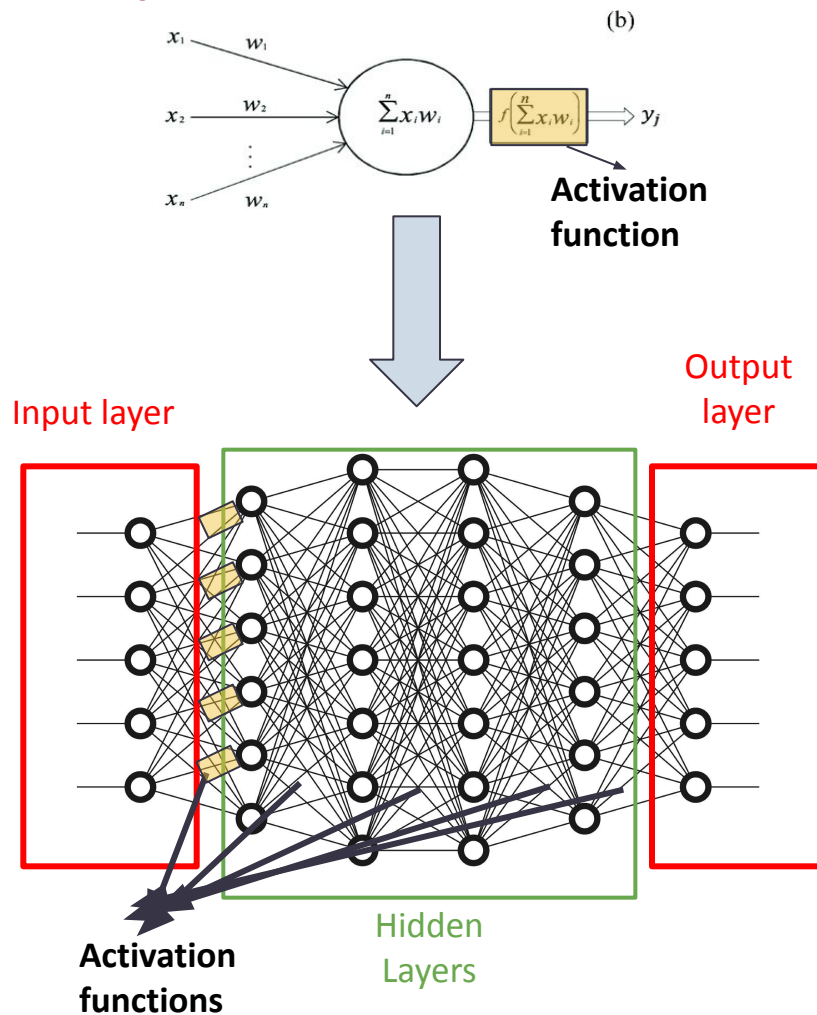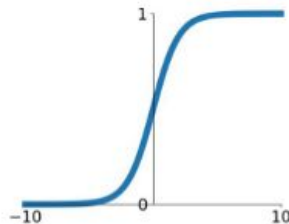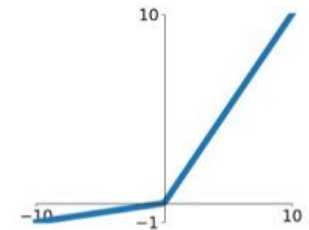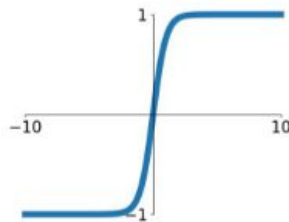
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$

Activation Functions

Sigmoid

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

tanh

$\tanh(x)$

ReLU

$\max(0, x)$

Leaky ReLU

$\max(0.1x, x)$

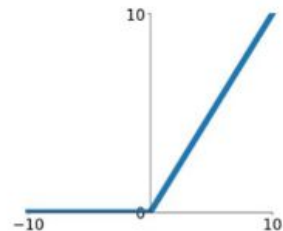**Maxout**

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU
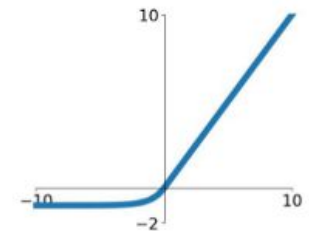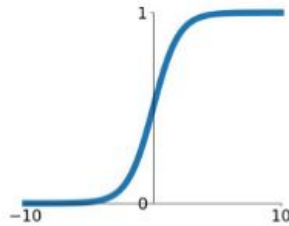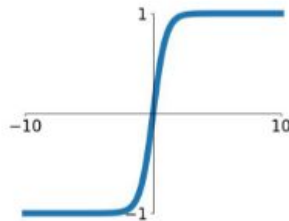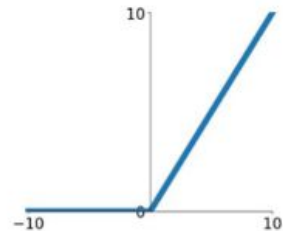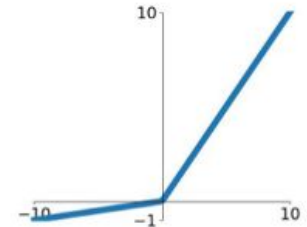
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$
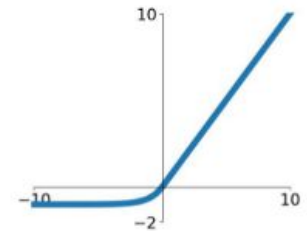
Let's implement our first ML model!
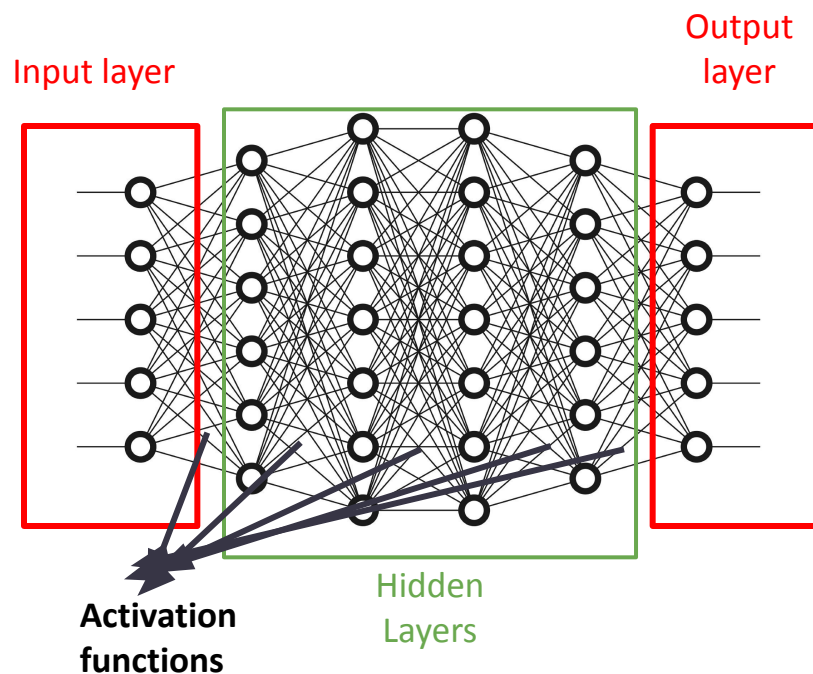
Let's go to our first tutorial and implement our first ML model!!

How do we teach MLP to classify?

How do we teach MLP to classify?
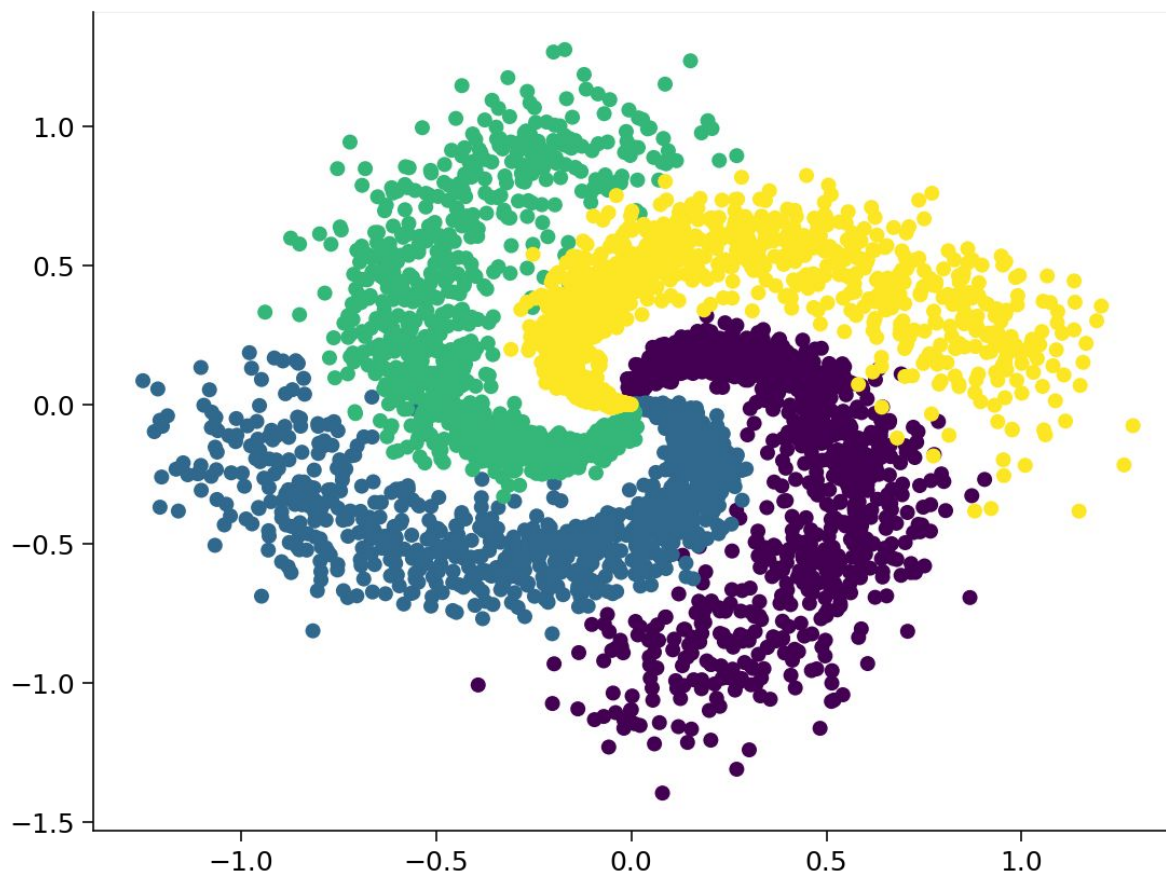
Dataset

How do we teach MLP to classify?
Dataset

How do we teach MLP to classify?
Dataset

Test data
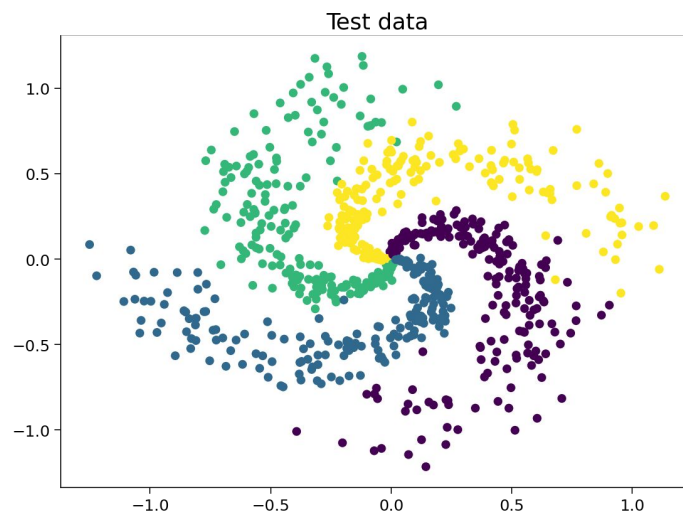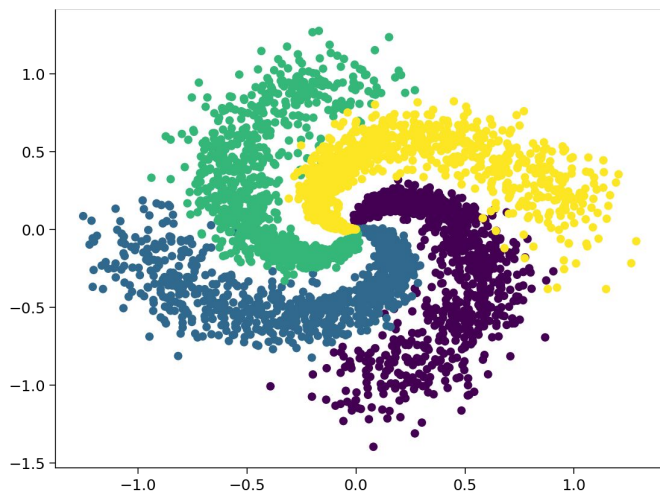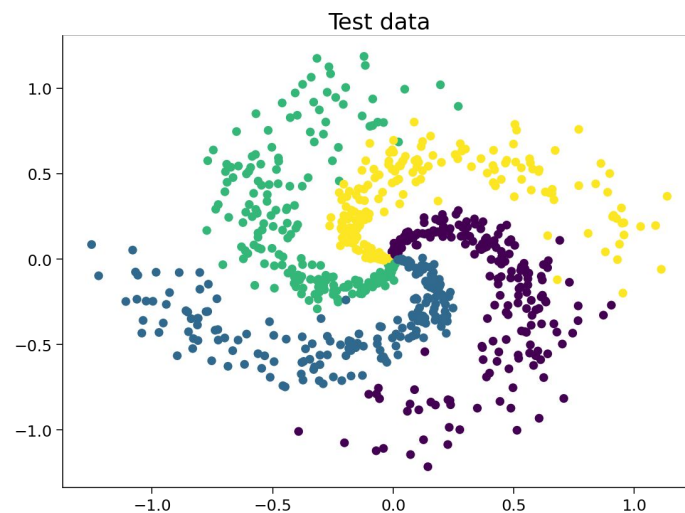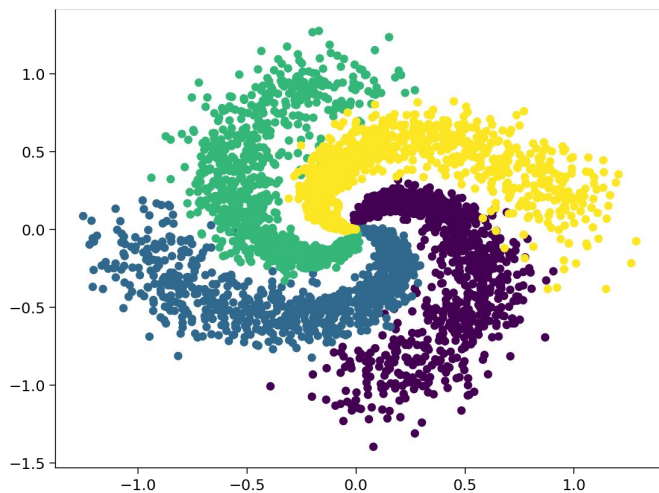
Let's prepare data for our model!

How do we teach MLP to classify?
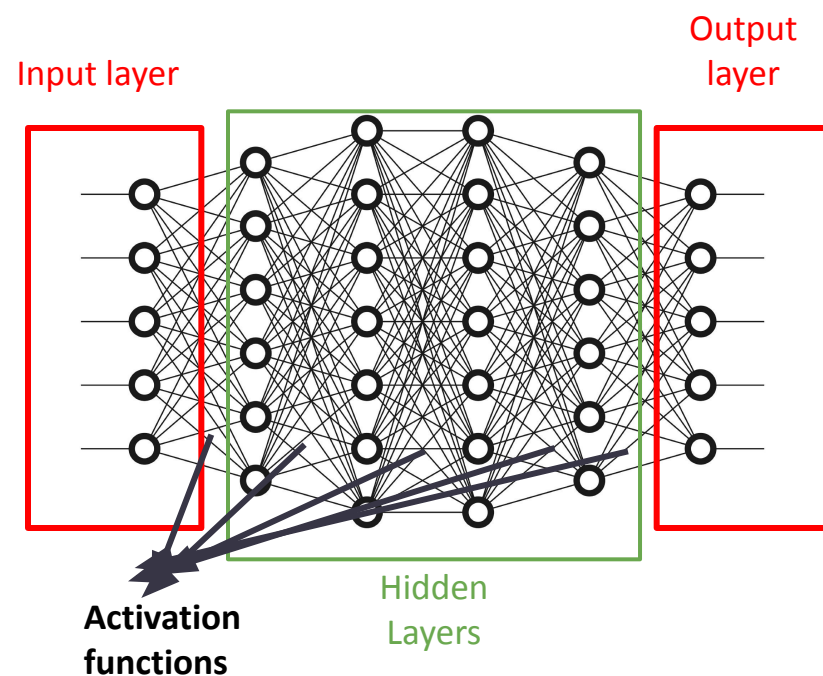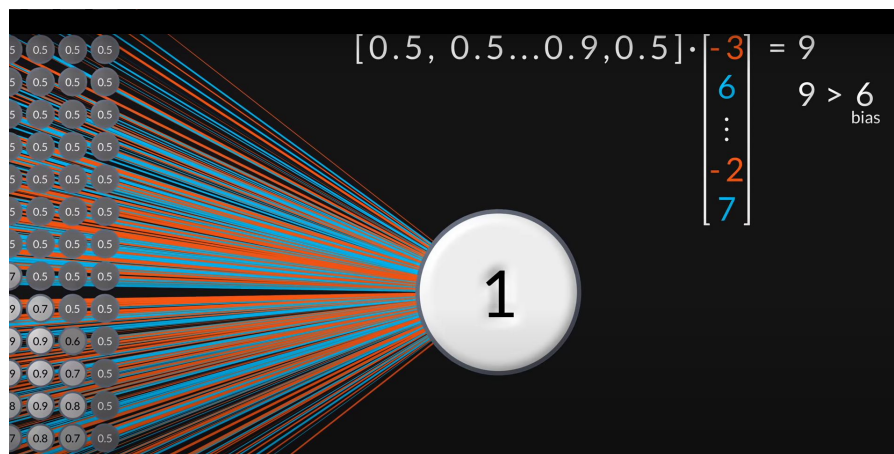
$[0.5, 0.5...0.9, 0.5] \cdot \begin{bmatrix} -3 \\ 6 \\ \vdots \\ -2 \\ 7 \end{bmatrix} = 9$

$9 > 6$ bias

1

Input layer

Output layer

Hidden Layers

Activation functions

How do we teach MLP to classify?
Learning good weights

Learning Recipe:

1-Initialize weights

2-Run a forward pass and make a prediction

3-Calculate loss to evaluate how good/bad your prediction was

4-Adjust your weights in a way so that loss will be minimized

Popular weight initialization methods: Xavier/Glorot Initialization , Normalized Xavier/Glorot Initialization, He Weight Initialization, Random Weight Initialization

Appendix: Andrew Ng's Lecture on Weight Initialization

How do we teach MLP to classify?
Learning good weights

Learning Recipe:

1-Initialize weights

2-Run a forward pass and make a prediction

3-Calculate loss to evaluate how

Input layer

Hidden layer

Hidden layer

Output layer

62% Dog

38% Cat
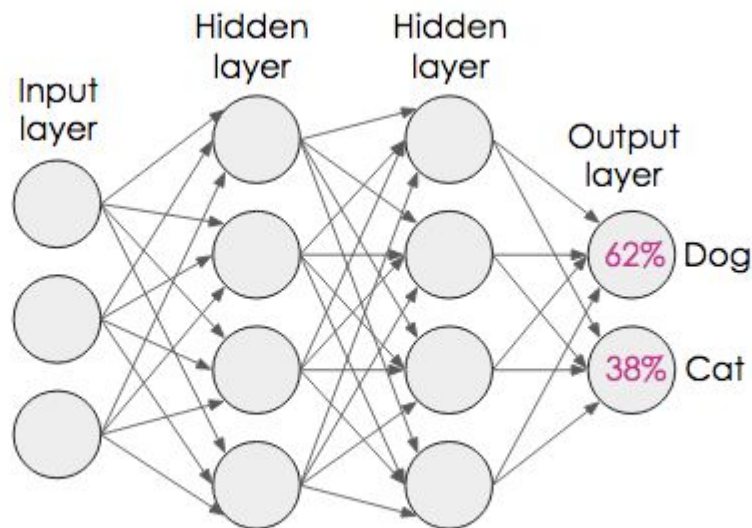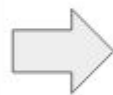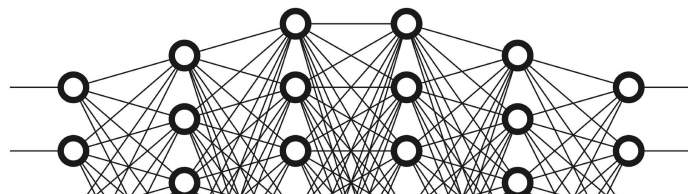
It should be 100% Cat :(

How do we teach MLP to classify?
Learning good weights

Learning Recipe:

1-Initialize weights

2-Run a forward pass and make a prediction

3-Calculate loss to evaluate how good/bad your prediction was

4-Adjust your weights in a way so that loss will be minimized

Classifying categorical variables -> Cross-entropy Loss Function

Input image source: Photo by Victor Grabarczyk on Unsplash . Diagram by author.

$$S(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{n} \exp(y_j)}$$

Input image	Logits (L)	Softmax	Output probabilities (P)	Classes
NN Layers	3.2		0.775	Dog
	1.3		0.116	Cat
	0.2		0.039	Horse
	0.8		0.070	Cheetah

S		T
0.775	$L_{CE}(S,T)$	1
0.116		0
0.039		0
0.070		0
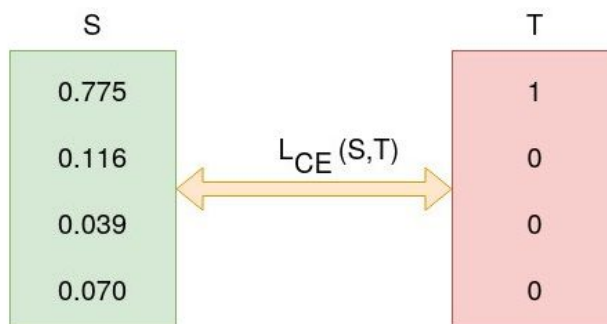
Cross Entropy (L) (Source: Author).

Input image source: Photo by Victor Grabarczyk on Unsplash . Diagram by author.
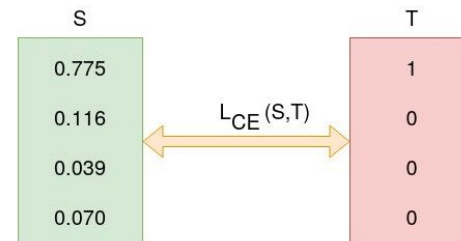
Cross Entropy (L) (Source: Author).

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \quad \text{for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the ith class.

$$L_{\text{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \quad \text{for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the ith class.

Cross Entropy (L) (Source: Author).

$$L_{CE} = -\sum_{i=1} T_i \log(S_i)$$

$$= -\left[1 \log_2(0.775) + 0 \log_2(0.126) + 0 \log_2(0.039) + 0 \log_2(0.070)\right]$$

$$= -\log_2(0.775)$$

$$= 0.3677$$

How do we teach MLP to classify?
Learning good weights

Learning Recipe:

1-Initialize weights

2-Run a forward pass and make a prediction

3-Calculate loss to evaluate how good/bad your prediction was

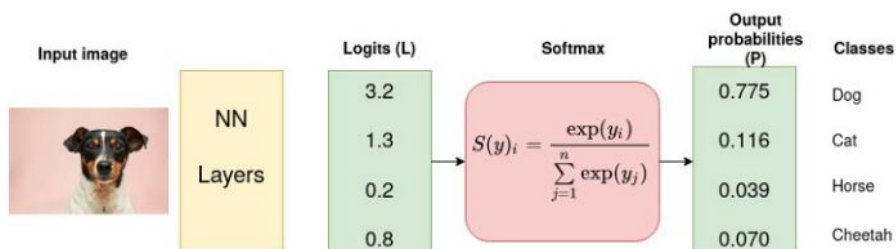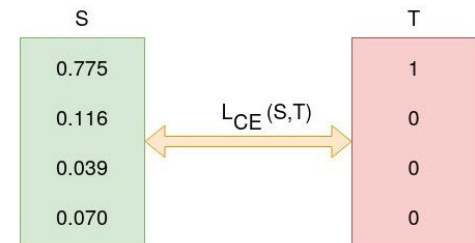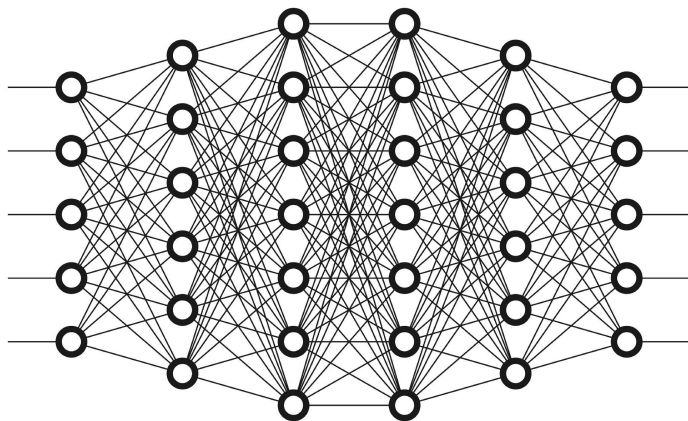4-Adjust your weights in a way so that loss will be minimized

Optimization! -> More details in the next lecture

We will use Adam Optimizer for this tutorial

How do we teach MLP to classify?
Learning good weights

Let's train our MLP model!!

Learning Recipe:
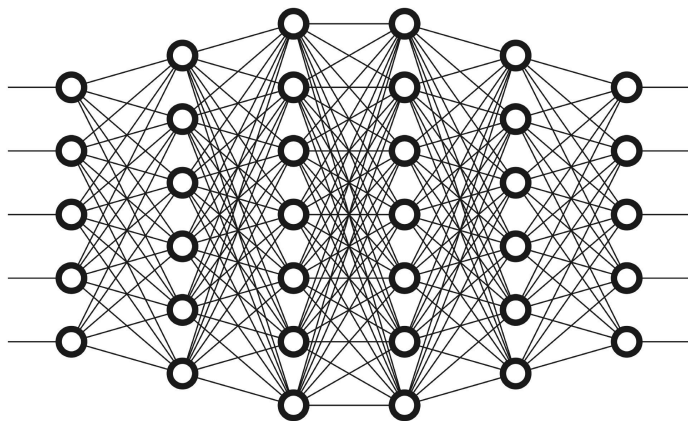
1-Initialize weights

2-Run a forward pass and make a prediction

3-Calculate loss to evaluate how good/bad your prediction was

4-Adjust your weights in a way so that loss will be minimized