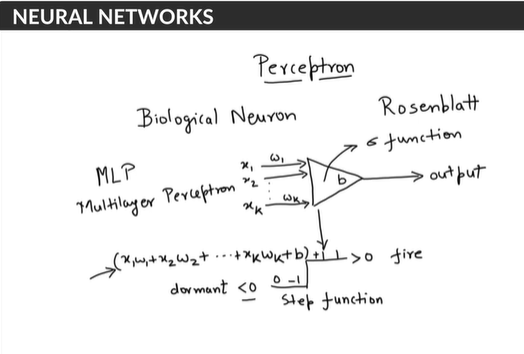
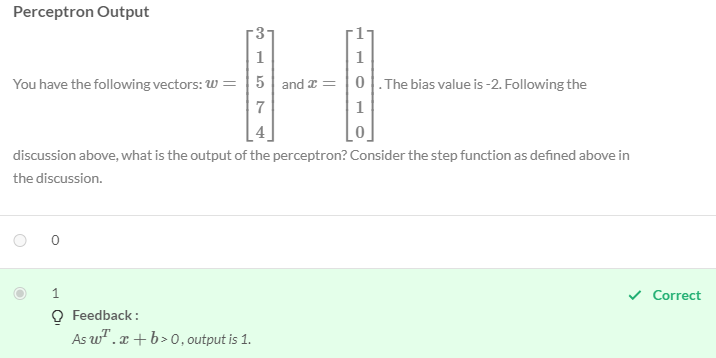
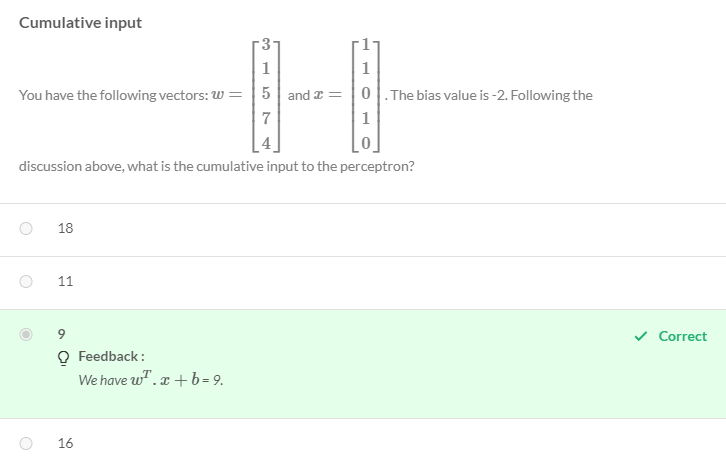
Neural Networks

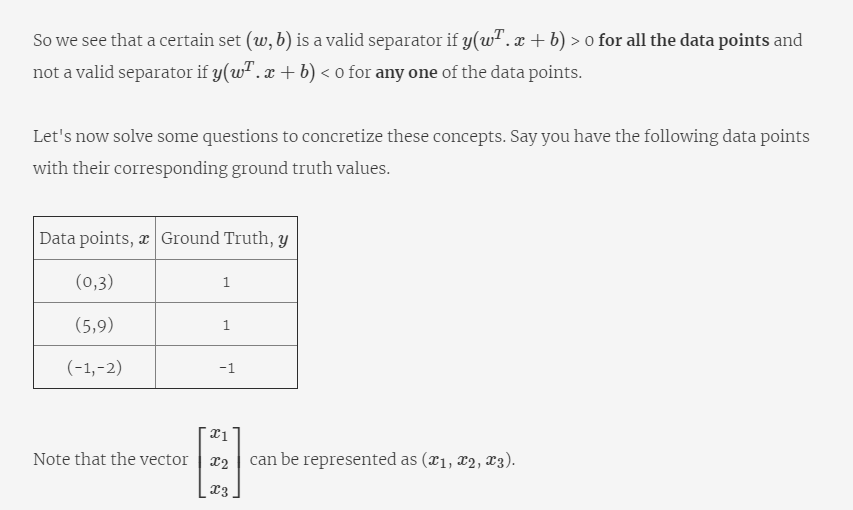
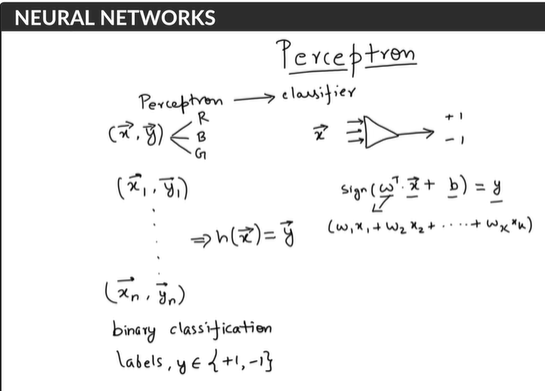


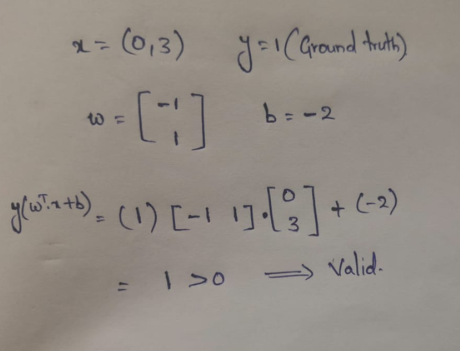
Architecture of ANN – Artificial Neural Networks



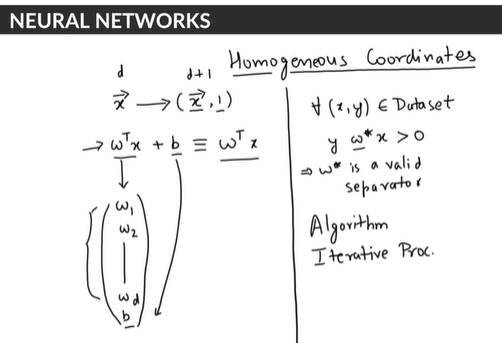


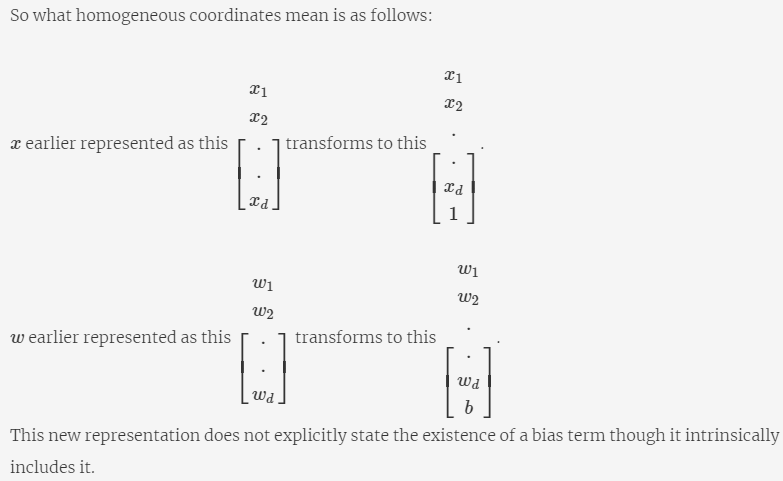


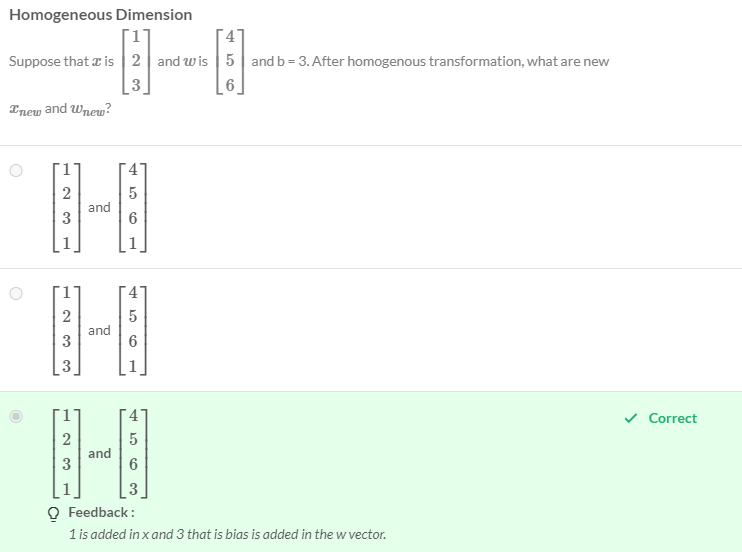


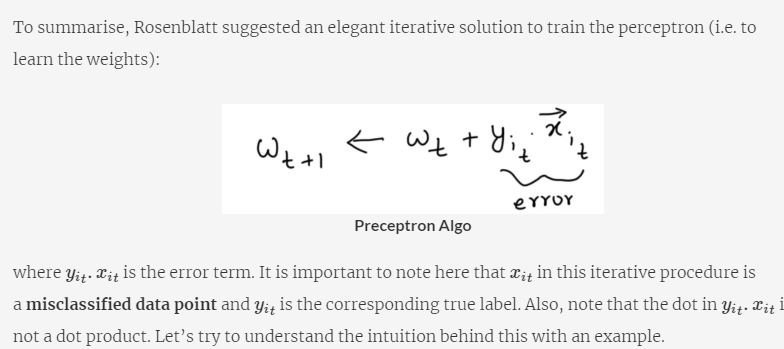


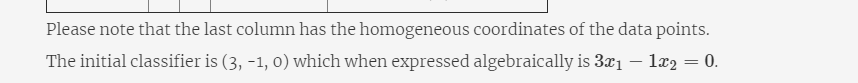


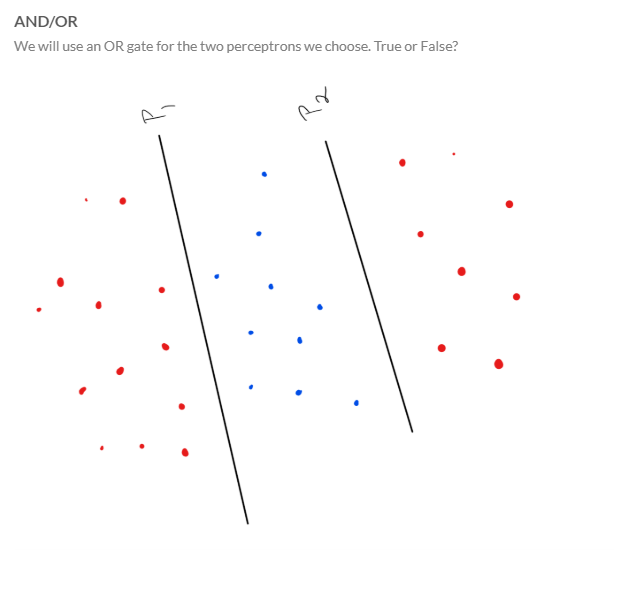


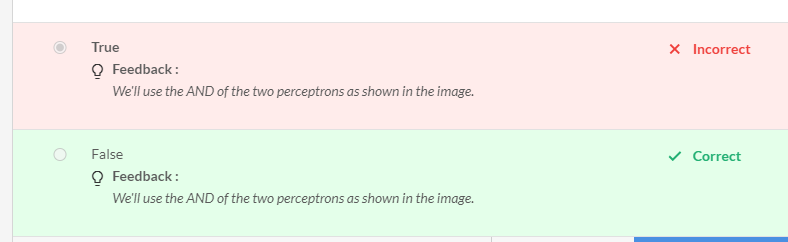


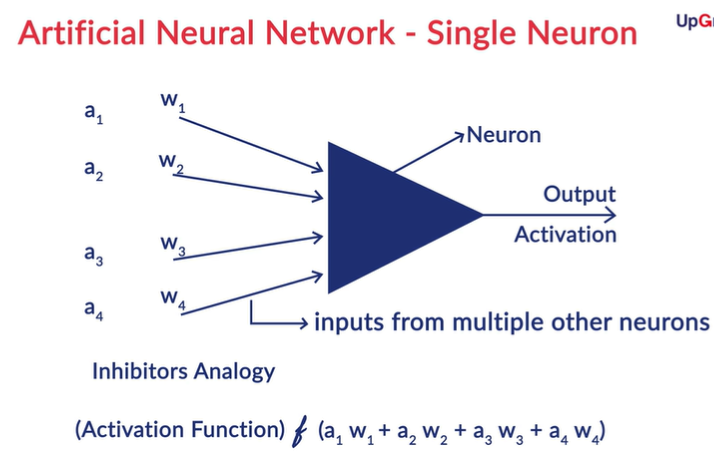


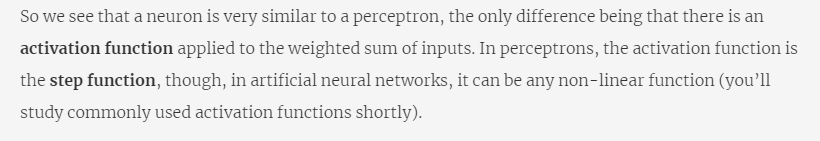


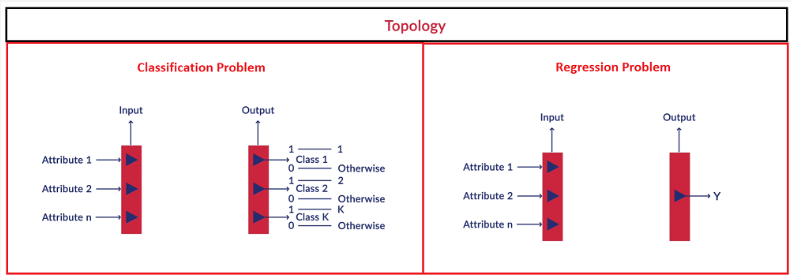


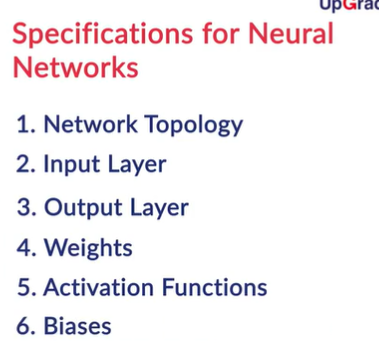




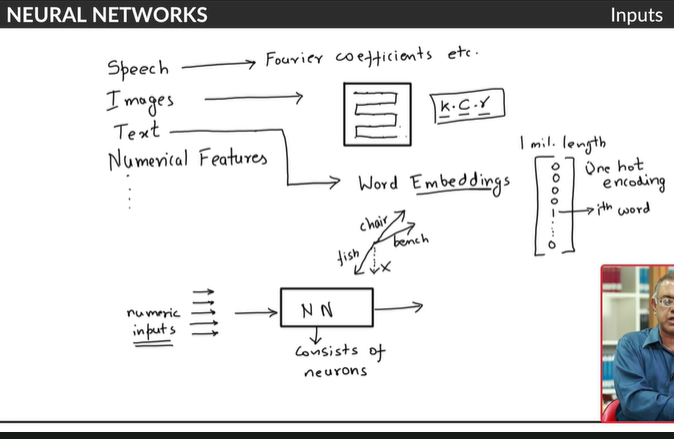




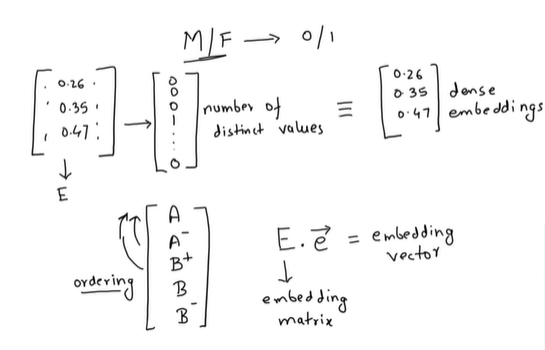




Inputs and outputs of a Neural Network:

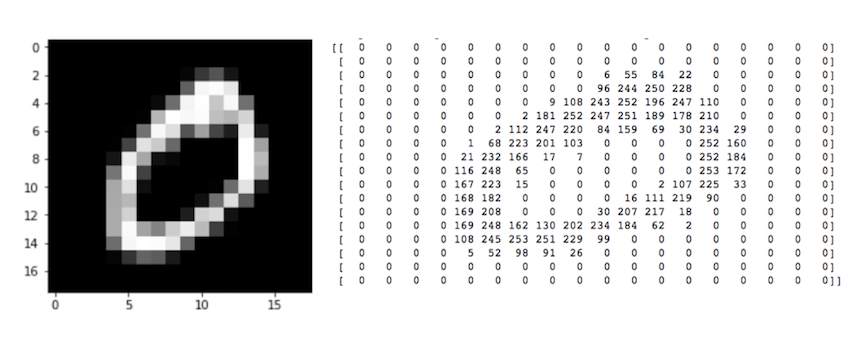
All Neural Networks would require a numerical inputx

Instead of using a one-hot encoding, we can use a dense embedding

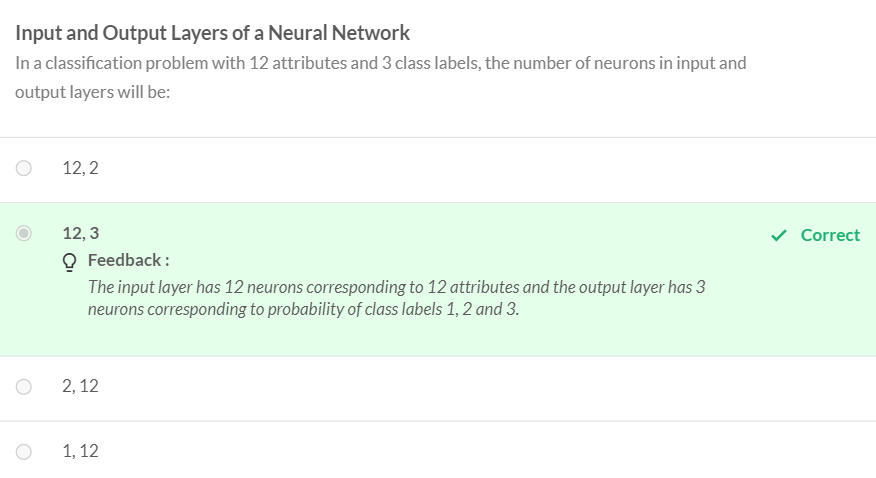


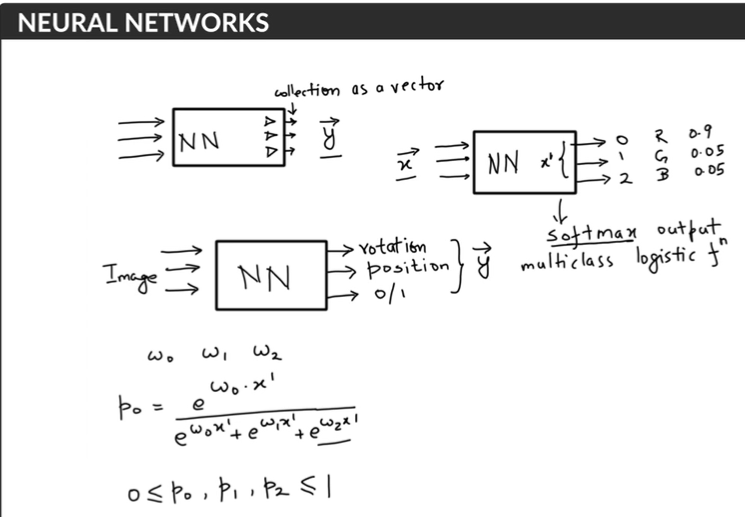
The most important thing to notice is that the inputs can only be numeric. For different types of input data, we use different ways to convert the inputs to a numeric form.

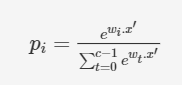
In case of **text data**, we either use **a one-hot vector** or **word embeddings**corresponding to a certain word. For example, if the vocabulary size is |V|, then you can represent the word wn as a one-hot vector of size |V| with a '1' at the nth element while all other elements being zero. The problem with one-hot representation is that usually the vocabulary size |V| is huge, in tens of thousands at least, and hence it is often better to use word embeddings which are a lower dimensional representation of each word.  
  
Feeding **images** (or videos) is straightforward since images are naturally represented as **arrays of numbers**. These numbers are the raw **pixels** of the image. Pixel is short for picture element. In images, pixels are arranged in rows and columns (an array of pixel elements). The figure below shows an image of a handwritten 'zero' in the MNIST dataset (black and white) and its corresponding representation in Numpy as an array of numbers. The pixel values are high where the **intensity** is high, i.e. the colour is white-ish, while they are low in the black regions.

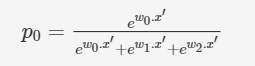
  
In a neural network, each **pixel** of the input image is a **feature.** For example, the image above is an 18 x 18 array. Hence, it will be fed as a **vector of size 324** to the network.

Note that the image above is a **black and white**image (also called **greyscale image**), and thus, each pixel has only one ‘channel’. If it were a **colour image** (called an RGB image - Red, Green, Blue), each pixel would have **three channels** - one each for red, blue and green as shown below. Hence, the number of neurons in the input layer would be 18 x 18 x 3 = 972. You’ll learn about this in detail in the next module on Convolution Neural Networks.



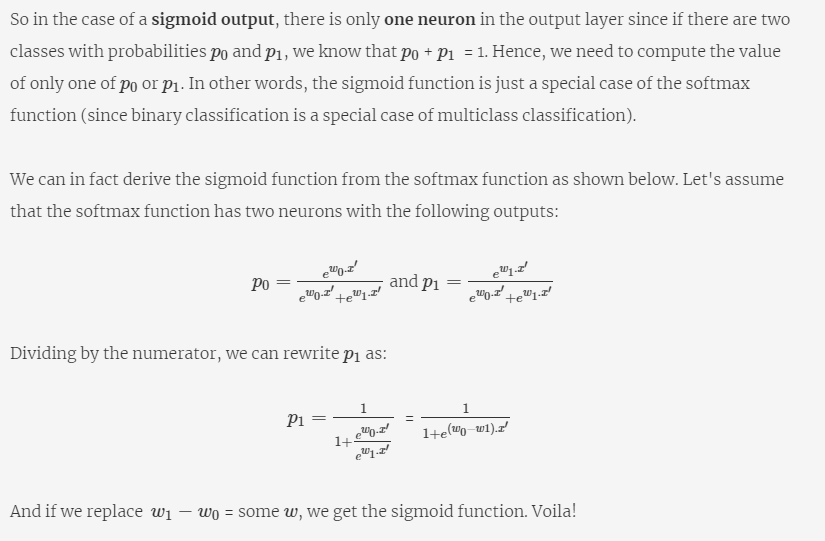


In this lecture, you were introduced to the **softmax output**. A softmax output is a **multiclass logistic function**commonly used to compute the 'probability' of an input belonging to one of the multiple classes. It is defined as follows:  
  
where c is the number of neurons in the output layer.

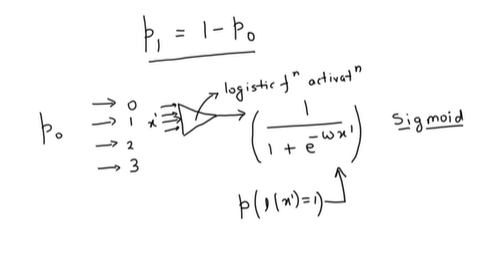
If the output layer has 3 neurons and all of them have the same input x′ (coming from the previous layers in the network), then the probability of the input belonging to class 0 is:  


The softmax function stated above is a general case for multiclass classification. Let’s see how the softmax function translates to a **sigmoid function** in the special case of **binary classification** in the next segment.

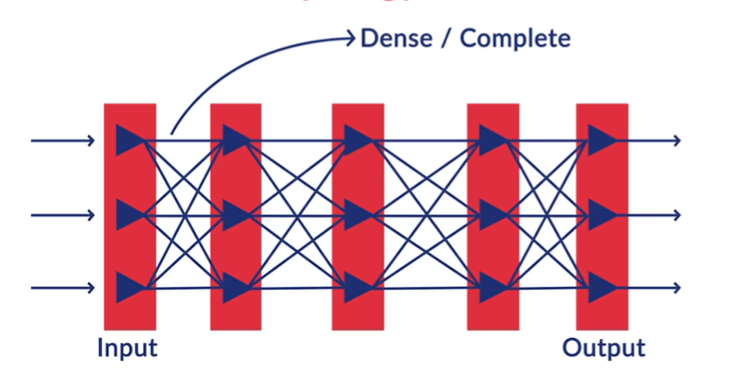
Recommended Reading: <http://neuralnetworksanddeeplearning.com/index.html>

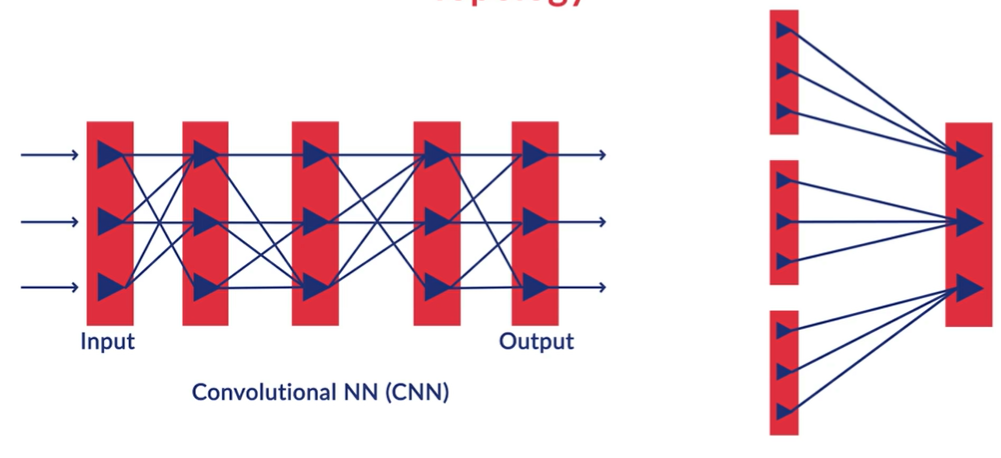


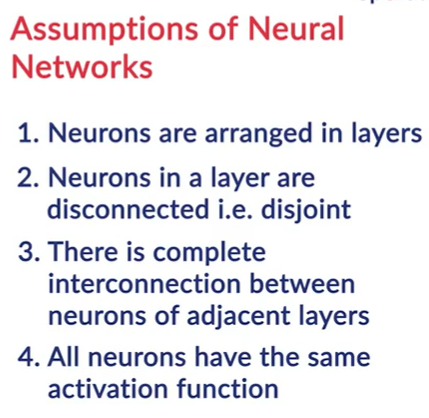
Multi class classification



Most common architecture for NN



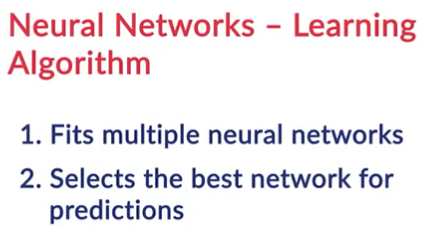
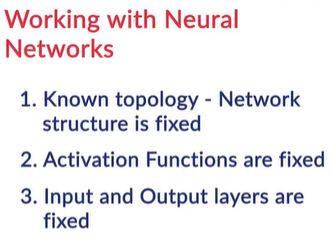




All neurons in the layer have the same activation function.

To summarise, commonly used neural network architectures make the following simplifying assumptions:

1. Neurons are **arranged in layers**and the layers are arranged **sequentially.**
2. Neurons **within the same layer do not interact** with each other.
3. All the inputs enter the network through the **input layer** and all the outputs go out of the network through the **output layer**.
4. Neurons in consecutive layers are **densely connected,**i.e. all neurons in layer l are connected to all neurons in layer l+1.
5. **Every interconnection** in the neural network has a **weight**associated with it, and **every neuron has a bias** associated with it.
6. All neurons in a particular layer use the **same activation function**.



During training, the neural network learning algorithm fits various models to the training data and selects the best model for prediction. The learning algorithm is trained with a **fixed set of hyperparameters** - the network structure (number of layers, number of neurons in the input, hidden and output layers etc.). It is trained on the**weights and the biases,**which are the **parameters of the network.**

