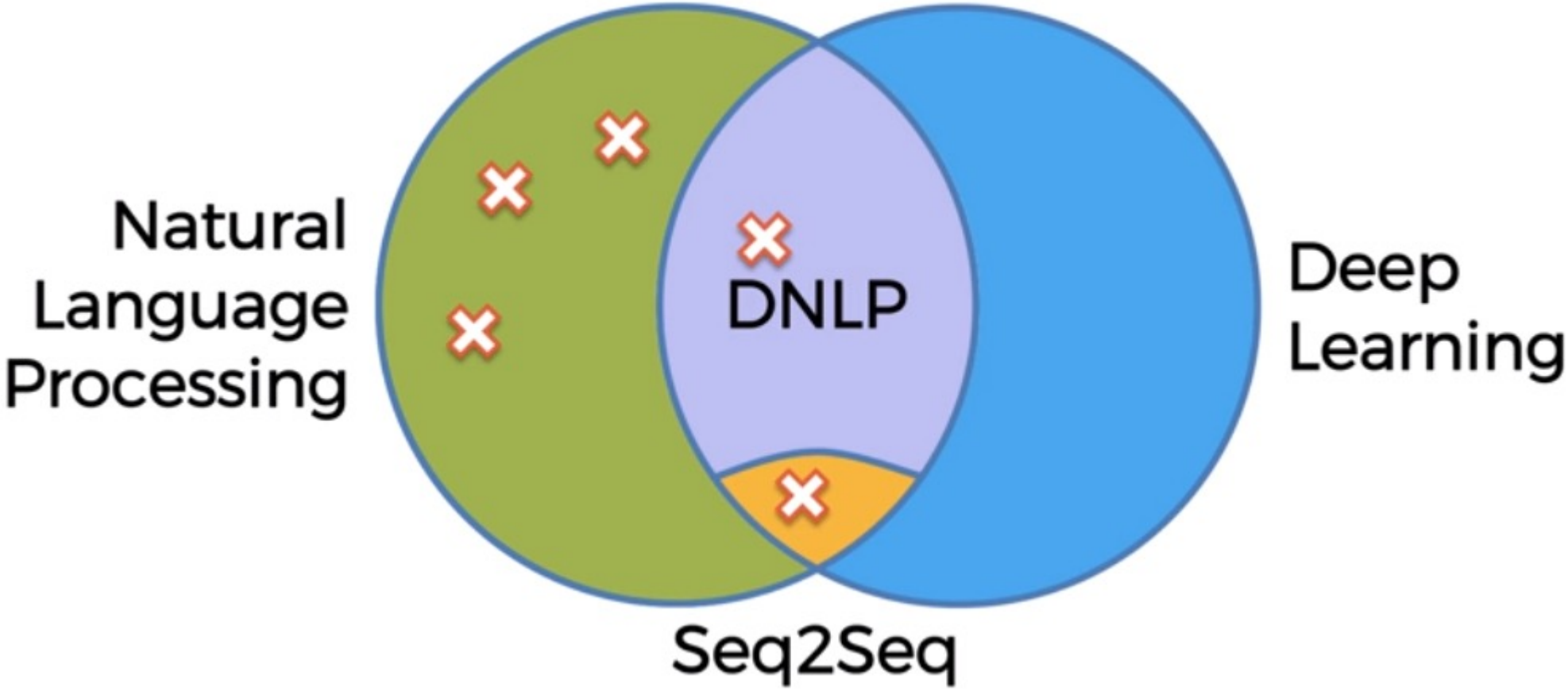
NLP-Natural Language Processing



NLP-Natural Language Processing

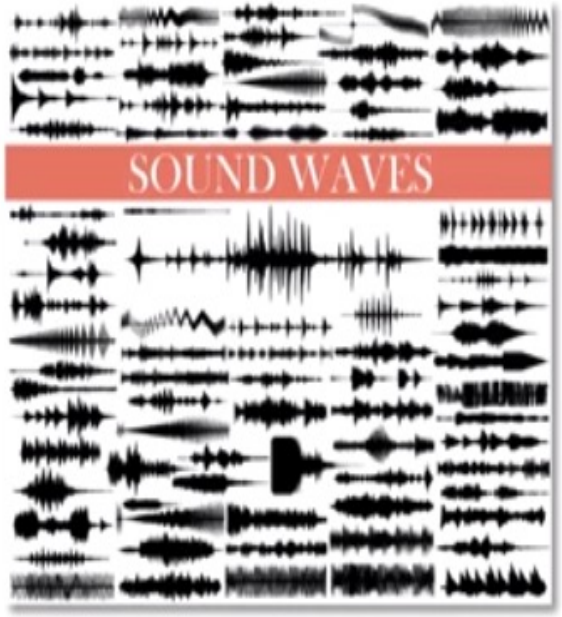
Some examples:

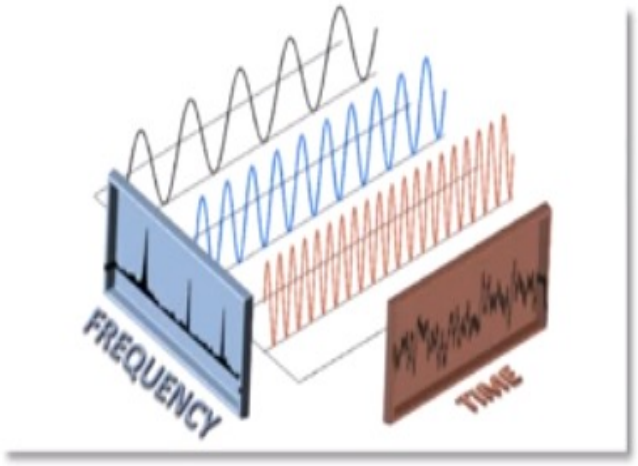
1. If / Else Rules (Chatbot)

- This is a way that we used to create chatbots back in the day. So if/else rules are located over here on our diagram in just the NLP part, and what they entail is a huge list of possible questions and answers to those questions.
- So once somebody in the chat asks a question, or we can identify that part of the sentence is the question that we have pre-recorded, then we will give them the correct answer, the answer that is associated with that question. But as you can imagine, such a mechanical approach to answering questions or chatting with people does not result in anything humanlike, anything realistic.
- But people want something tailored to them, something specific, they're asking about something else, but doesn't fit into that list of questions and answers. And it just very quickly becomes a mess.

NLP-Natural Language Processing

2. Audio frequency components analysis (Speech Recognition)





A very general overview of what happens. We look at the frequencies, certain mathematical operations and we're not doing any neural computations or not creating any neural networks. We're just doing mathematical calculations
around the frequencies that we can observe, comparing them to the mathematical calculations we have in our library of pre-analyzed frequencies. And then we are matching it up. We're finding what word a person is saying, what question they're asking or what the sentence is meaning. And then that is how we recognize speech.

NLP-Natural Language Processing

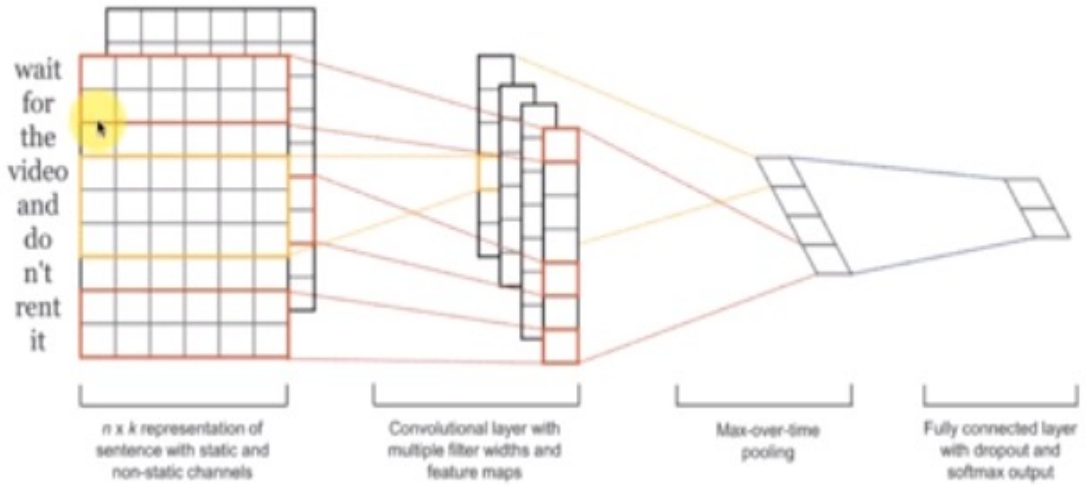
3. Bag-of-words model (Classification)

Comment	Pass/Fail
Great job!	1
Amazing work.	1
Well done.	1
Very well written.	1
Poor effort.	0
Could have done better.	0
Try harder next time.	0
...	...

It'll look at the words and try to classify these words or associate these words with either positive results or a negative result in our case. And so in this case, like amazing would be most likely associated with positive view. But then these other words like poor or harder would be associated with zeros. So then it'll remember and keep these words in a bag and next time something comes up, for instance, somebody says good, good job, keep, keep it up or something like that, it will analyze the words that are in that new sentence by pulling them out of the bag and looking at them and understanding are they mostly associated ones or zeros? And then it'll be able to predict or classify the new comment, even without knowing what the mark was, pass or fail.

NLP-Natural Language Processing

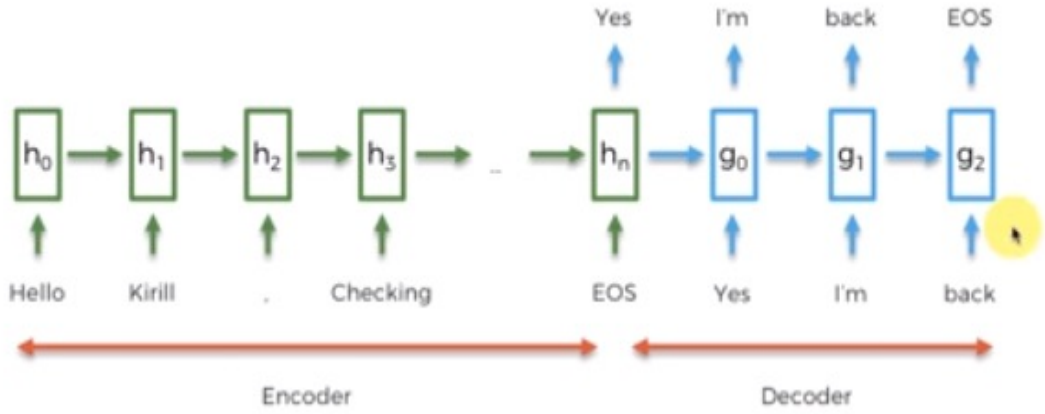
4. CNN for text Recognition (Classification)



It's called convolutional neural networks for text recognition then further for classification, which is indeed a deep natural language process model. It's a neural network that is used for mostly image recognition for like videos, self-driving cars use them to detect obstacles on roads, people and so on. So mostly it's used for image processing or video processing. The way it works is, these words are transformed into a matrix and that's done through an operation called embedding of words and once they're in a matrix, the same principles as were applied for images in convolutional neural networks, are applied. There's a convolution operation going through these images. Then they're pooled, max pooled or min pooled or sum pooled, and then they're flattened and we have the prediction. It’s just an overview

NLP-Natural Language Processing

5. Seq2Seq (many applications)



Each word that you used to type was converted to its target language giving no regard to its grammar and sentence structure. Seq2seq revolutionized the process of translation by making use of deep learning. It not only takes the current word/input into account while translating but also its neighbourhood. Seq2Seq (Sequence-to-Sequence) is a type of model in machine learning that is used for tasks such as machine translation, text summarization, and image captioning. The model consists of two main components:

- Encoder
- Decoder

Seq2Seq models are trained using a dataset of input-output pairs, where the input is a sequence of tokens and the output is also a sequence of tokens. The model is trained to maximize the likelihood of the correct output sequence given the input sequence.

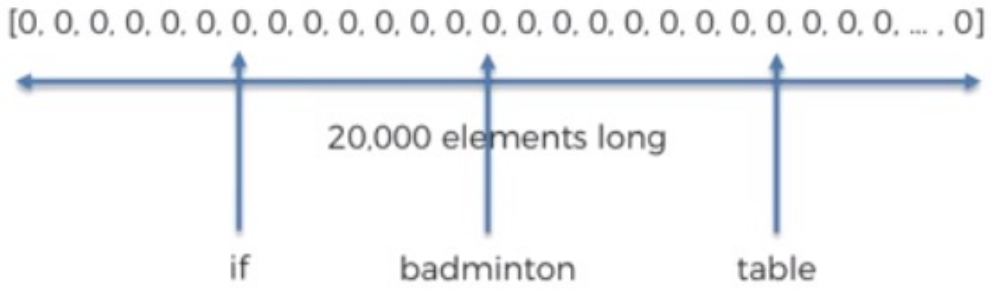
Bag Of Words

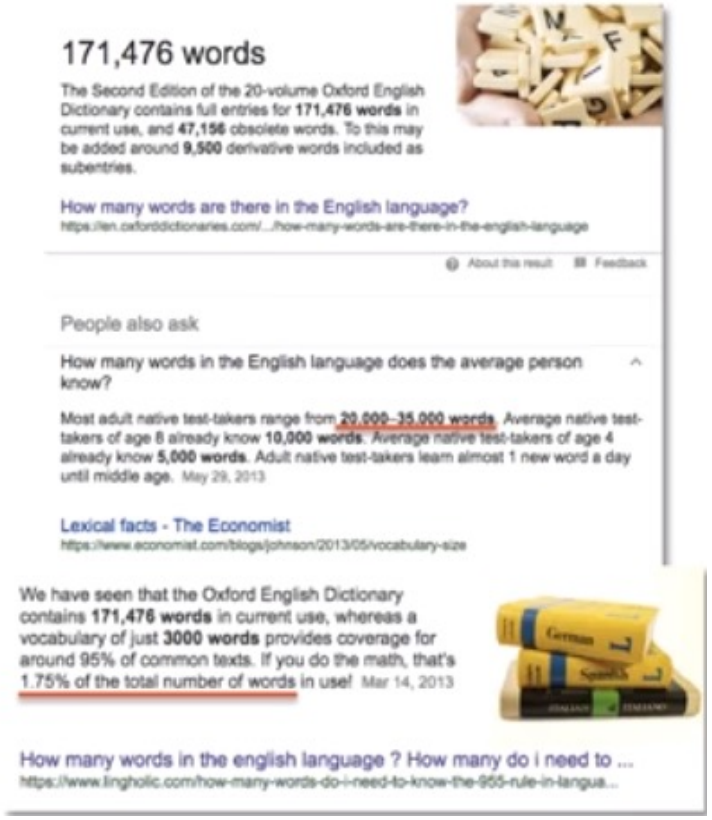
Checking if you are back to Oz. Let me know if you are around and keen to sync on how things are going. I defo could use some of your creative thinking to help with mine :)

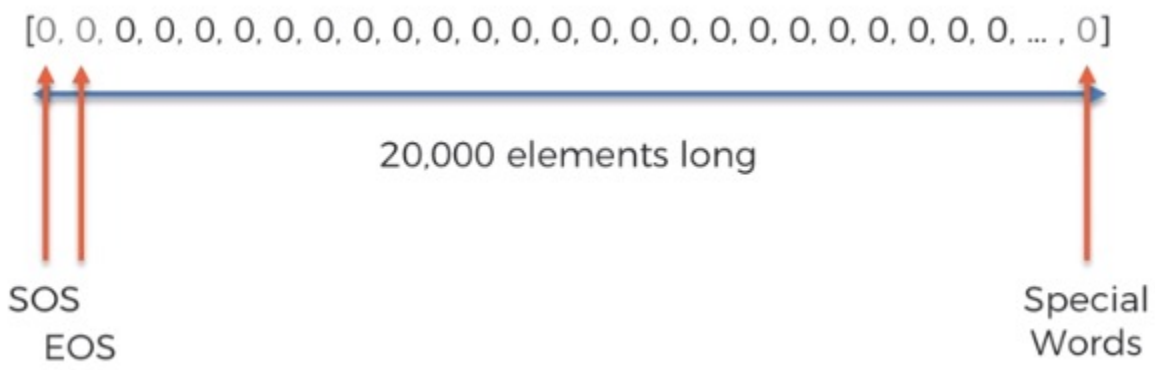
Cheers,
V

...

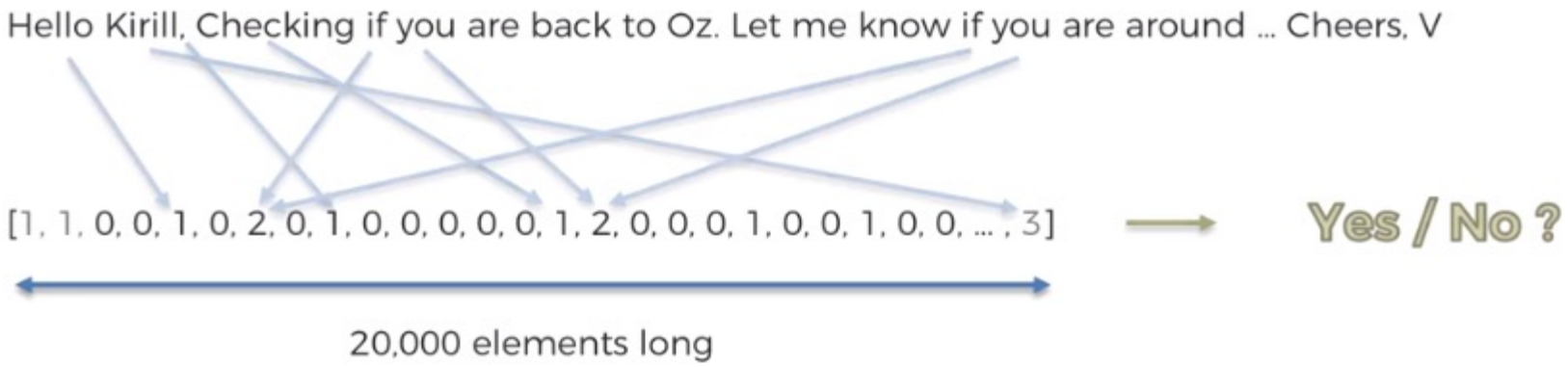
Yes, I'm around. | I'm back! | Sorry, I'm not.







Bag Of Words



Training Data:

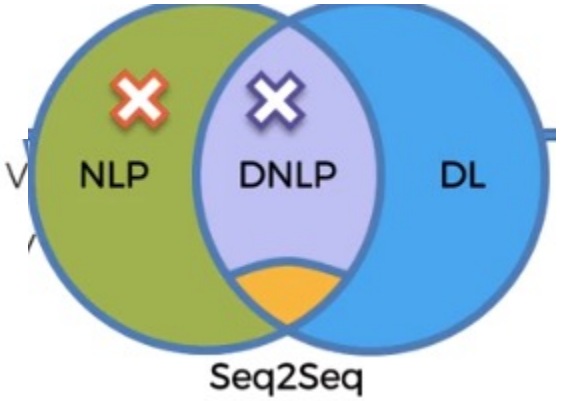
Hey mate, have you read about Hinton's capsule networks?	→	No
Did you like that recipe I sent you last week?	→	Yes
Hi Kirill, are you coming to dinner tonight?	→	Yes
Dear Kirill, would you like to service your car with us again?	→	No
Are you coming to Australia in December?	→	Yes
...	→	...

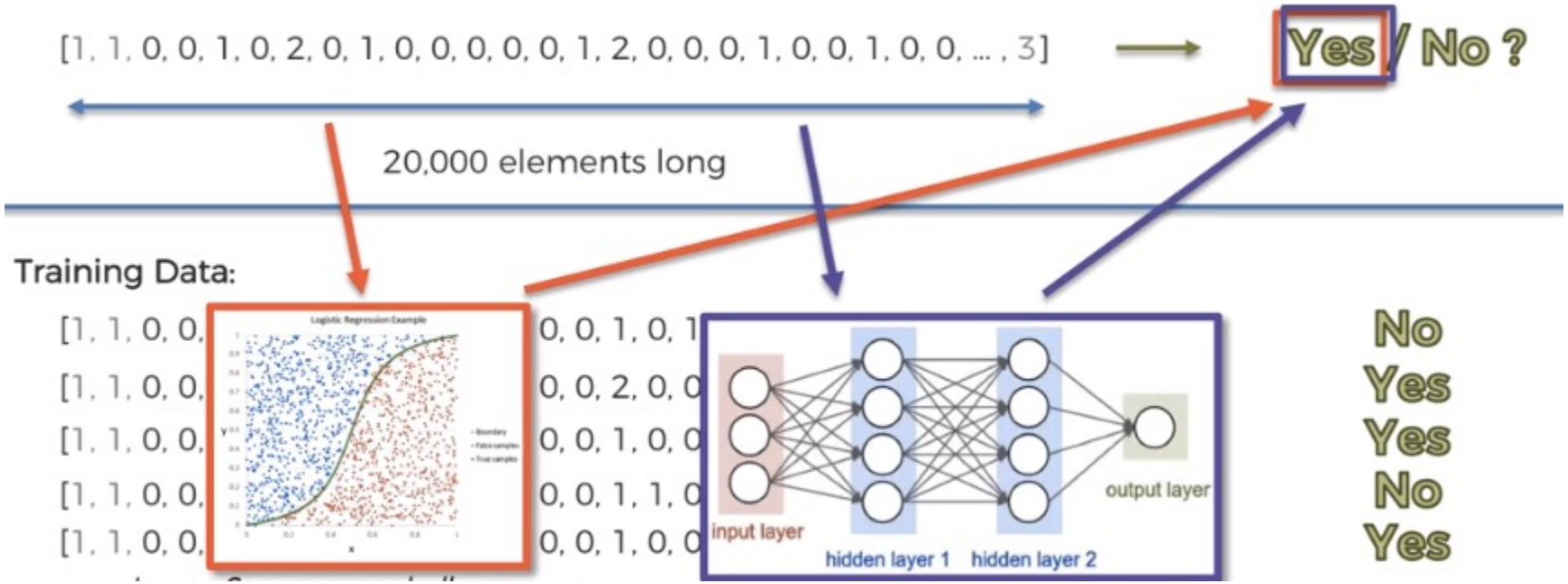
Training Data:

[1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, ... , 2]	→	No
[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, ... , 0]	→	Yes
[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, ... , 1]	→	Yes
[1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, ... , 1]	→	No
[1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, ... , 1]	→	Yes
...	→	...

Bag Of Words

Hello Kirill, Checking if you are back to Oz. Let me know if you are around … Cheers, V



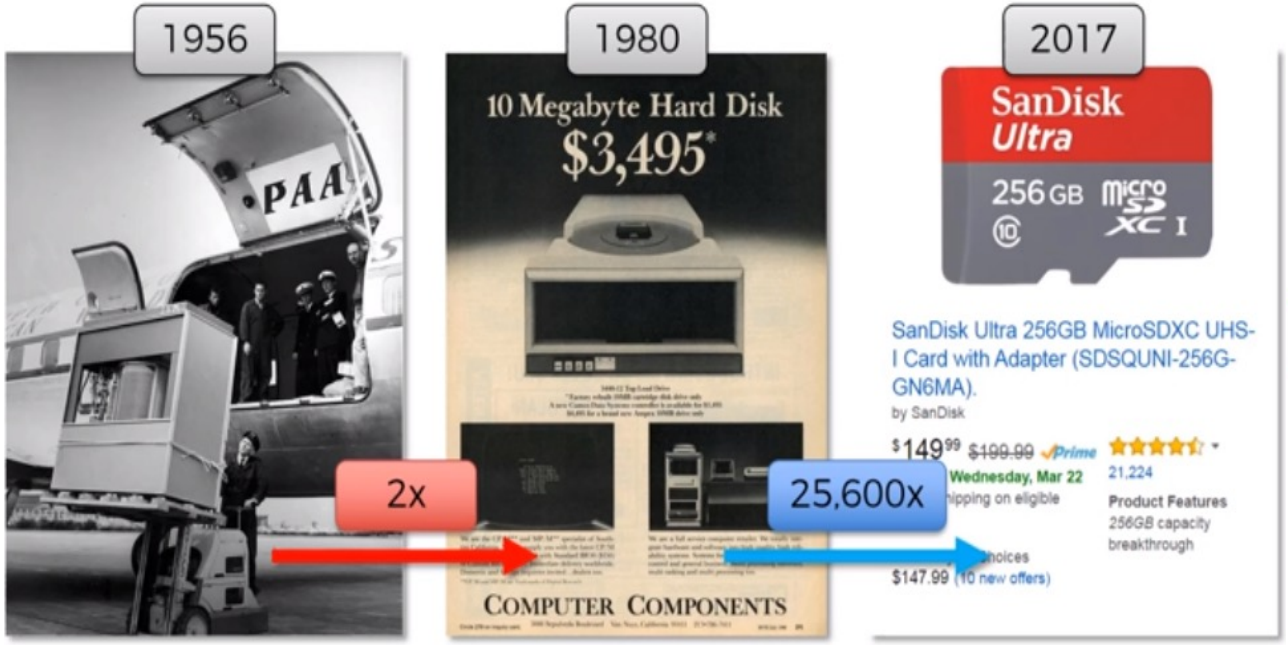


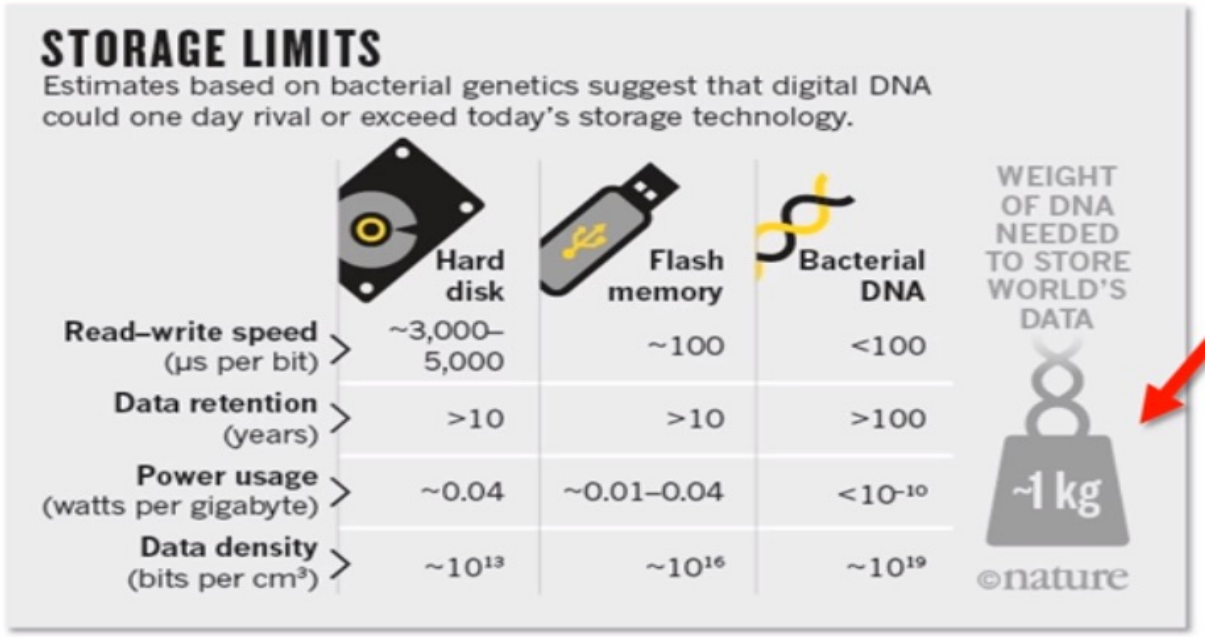
Deep Learning

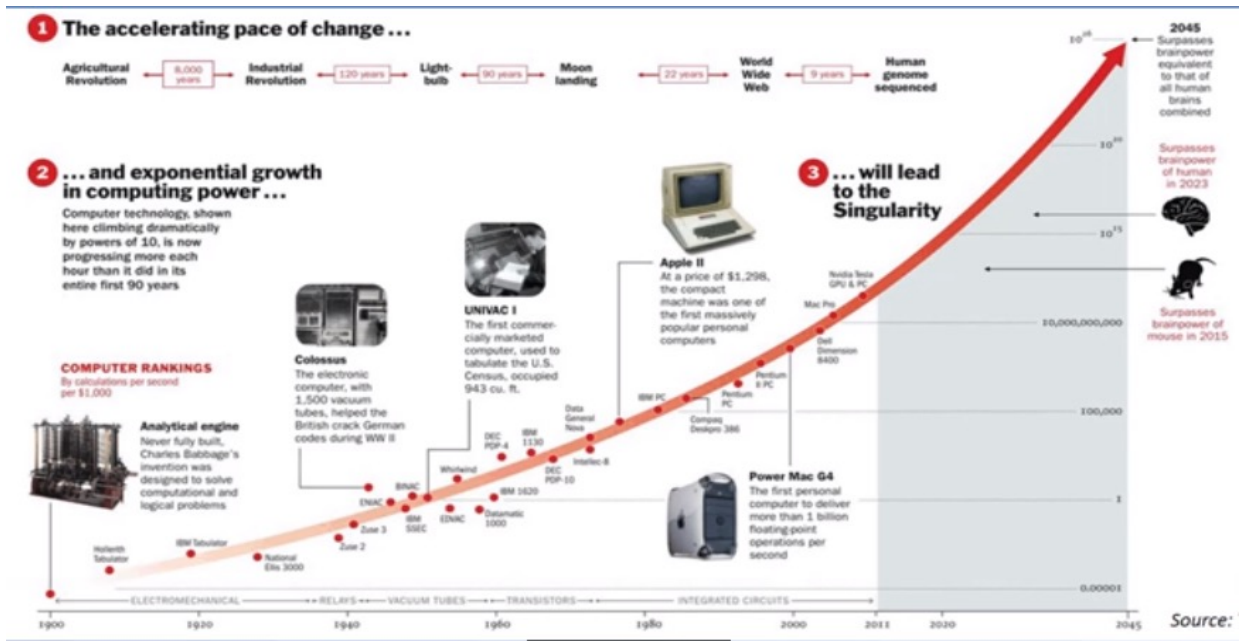
Deep Learning is the most exciting and powerful branch of Machine Learning. Deep Learning models can be used for a variety of complex tasks:

- Artificial Neural Networks for Regression and Classification
- Convolutional Neural Networks for Computer Vision
- Recurrent Neural Networks for Time Series Analysis
- Self Organizing Maps for Feature Extraction
- Deep Boltzmann Machines for Recommendation Systems
- Auto Encoders for Recommendation Systems

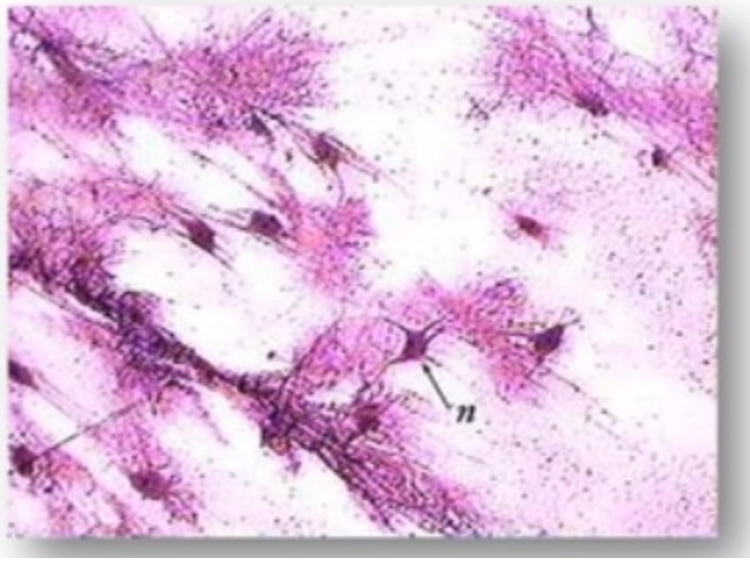
Deep Learning

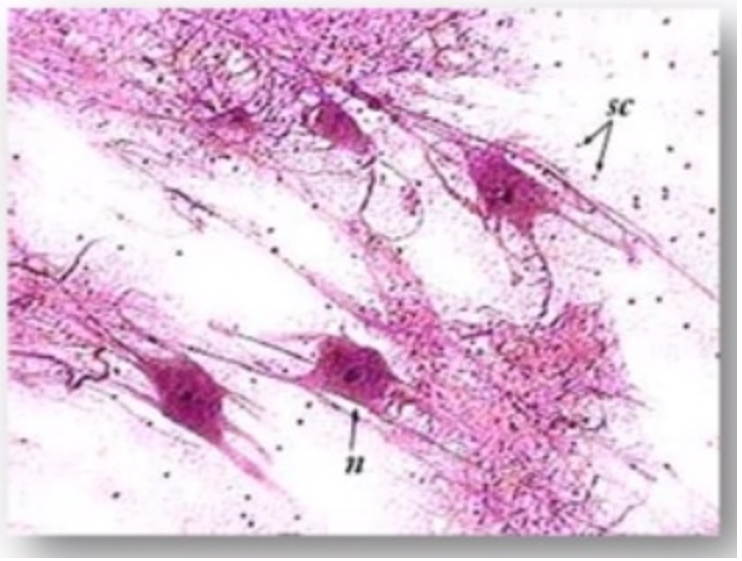


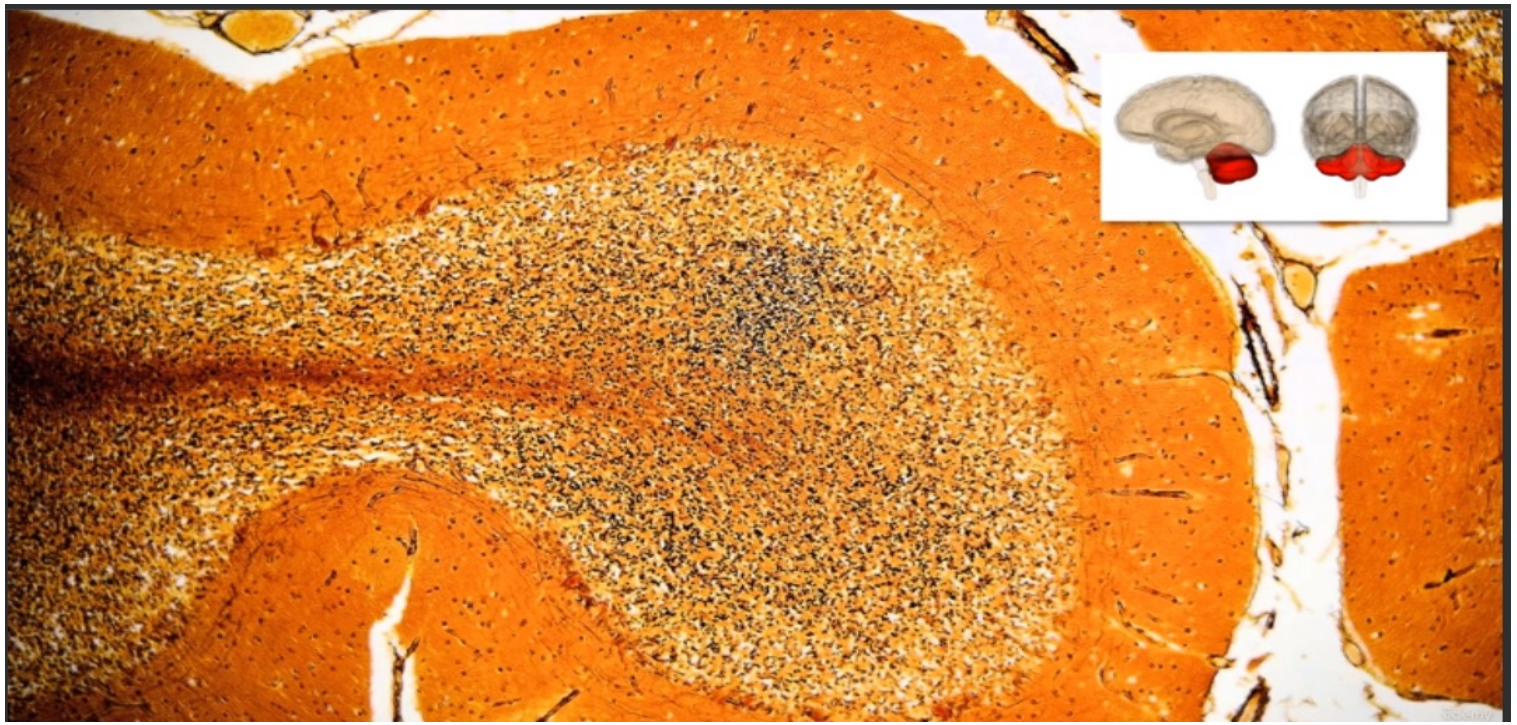




Deep learning

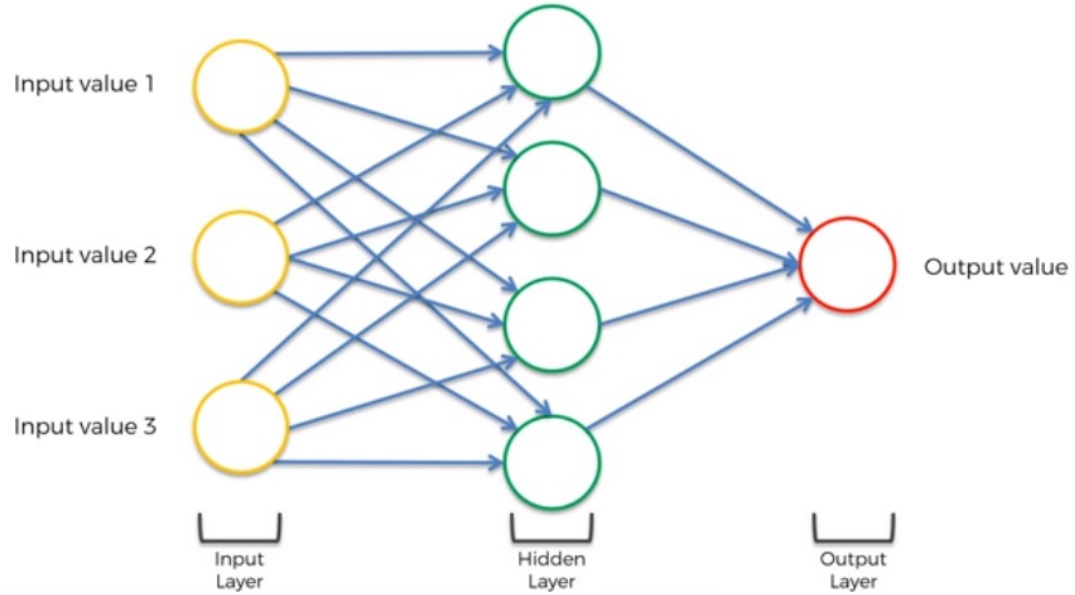


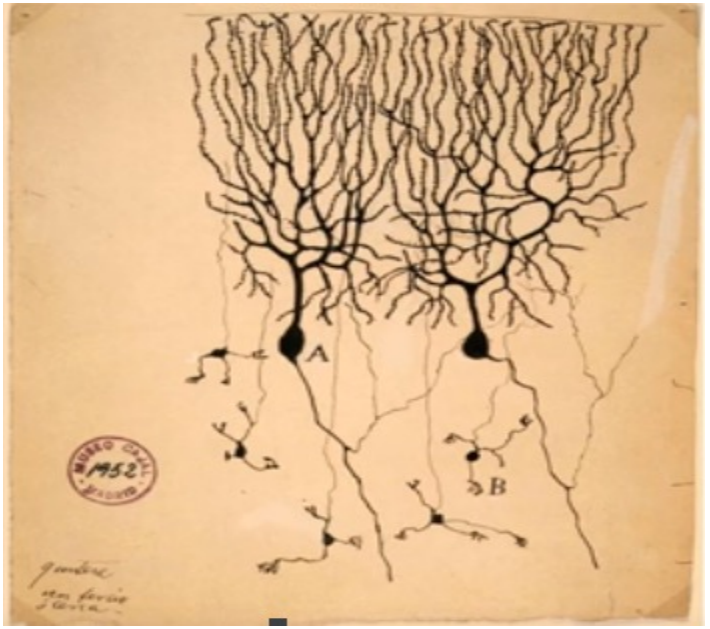


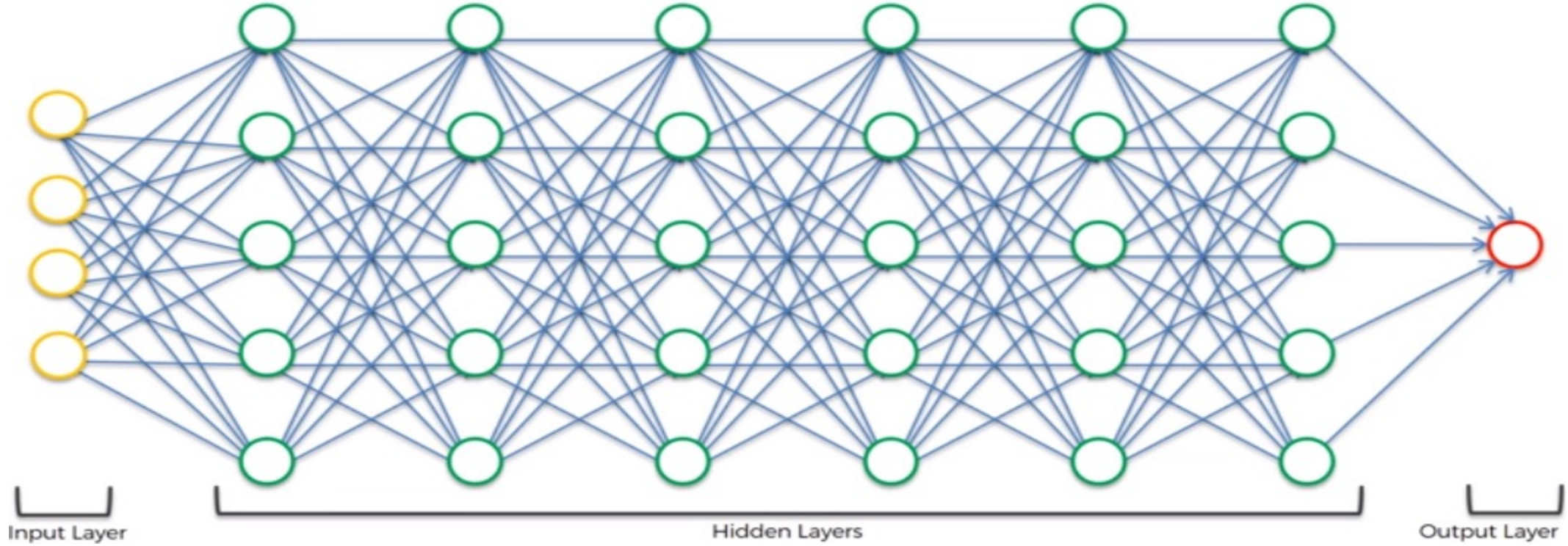


Here we've got some neurons, so these are neurons which are having smeared onto glass and being looked at under a microscope with some coloring. And this is, you can see what they look like. So they have a like a body, they have these branches and they have like tails and so on. We can see they have like a nucleus inside in the middle and that's basically what a neuron looks like. In the human brain, there's approximately a hundred billion neurons altogether. So these are individual neurons,there are actually modern neurons because they're bigger, they're easier to see. But, nevertheless, there's a hundred billion neurons in the human brain and each neuron is connected to as many as a thousand of its neighbors.

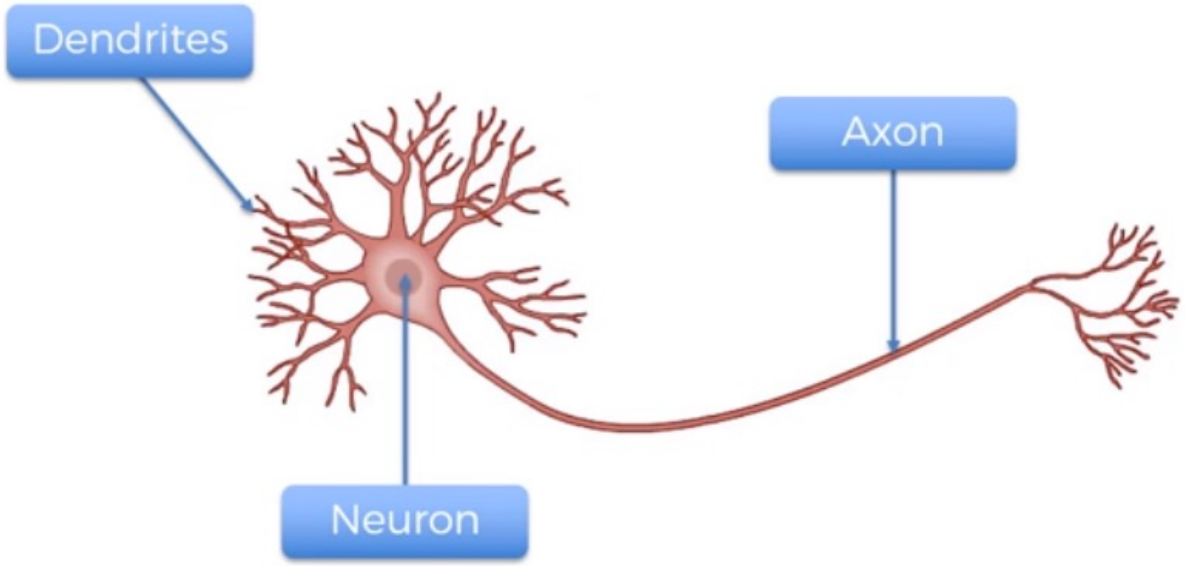
Deep learning

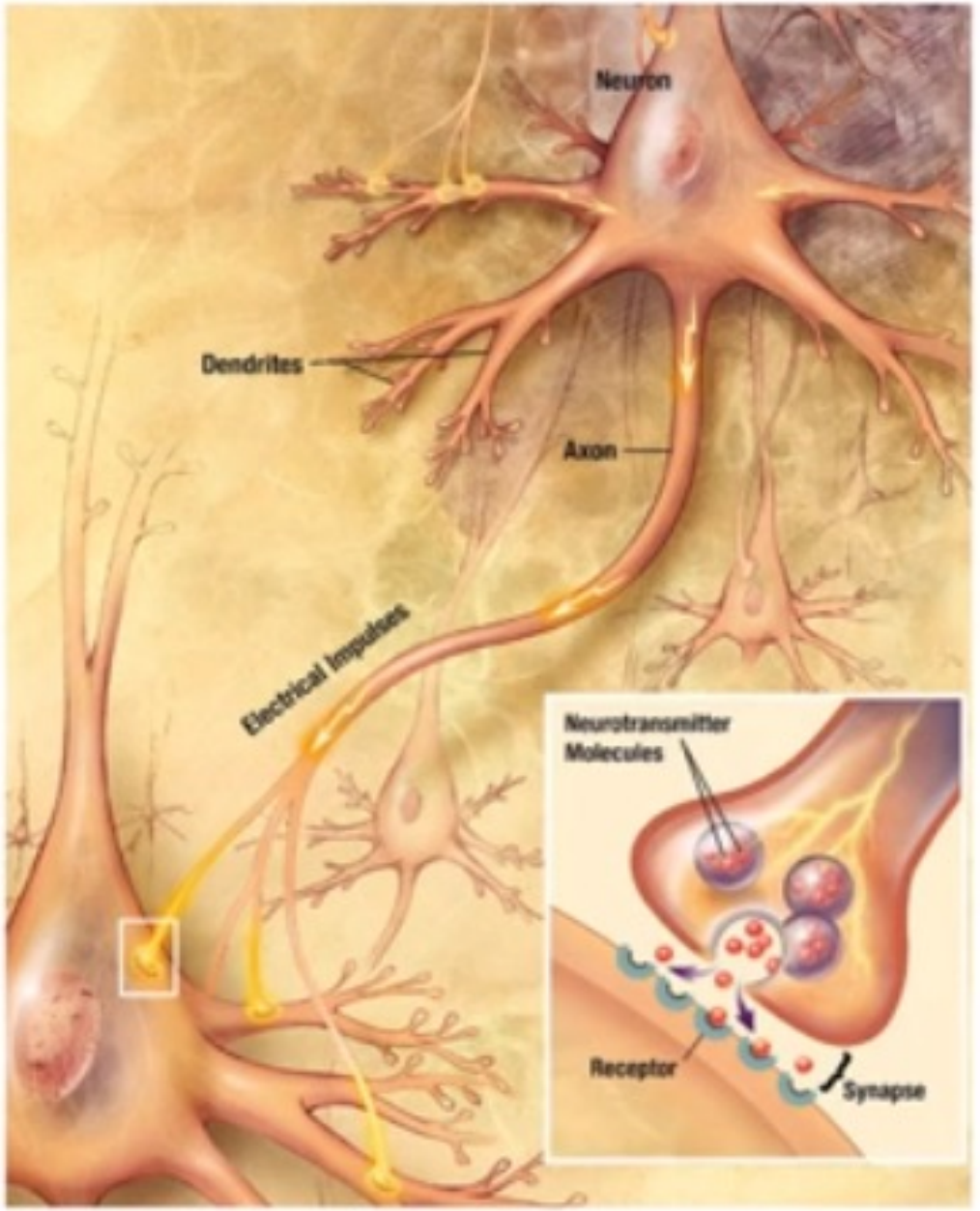




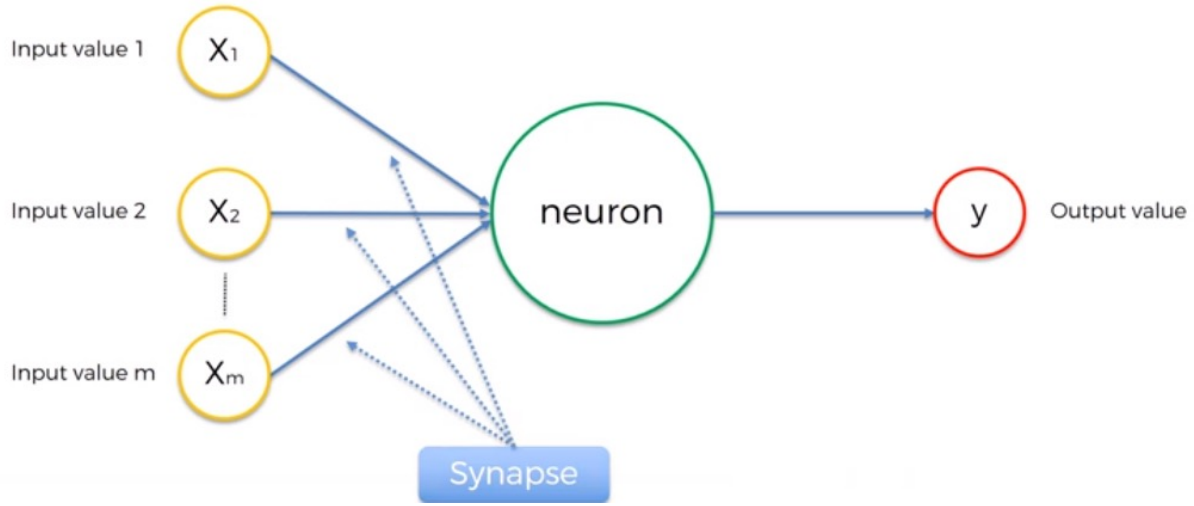


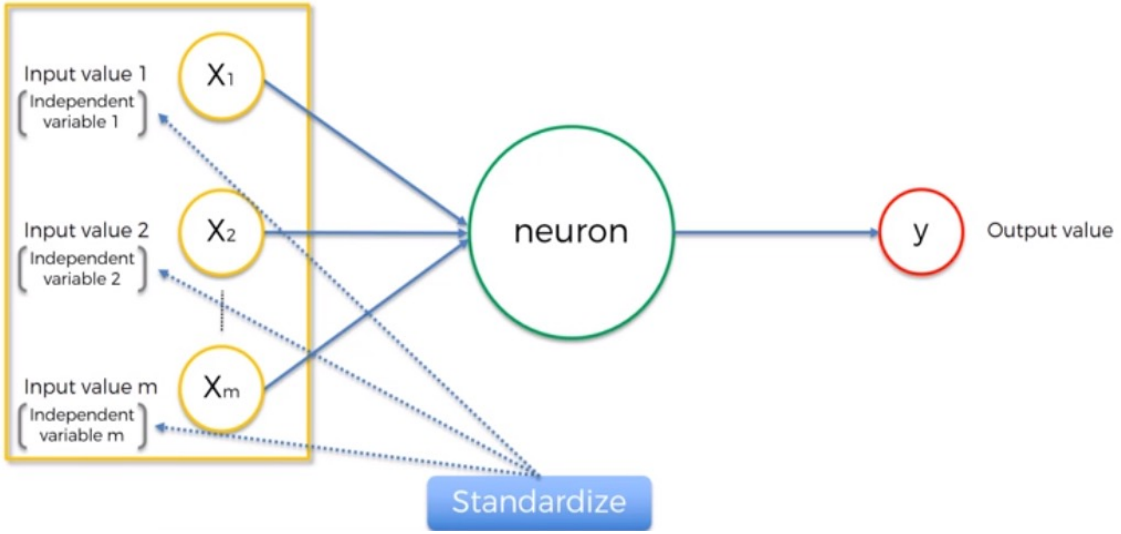
Deep learning- Neuron



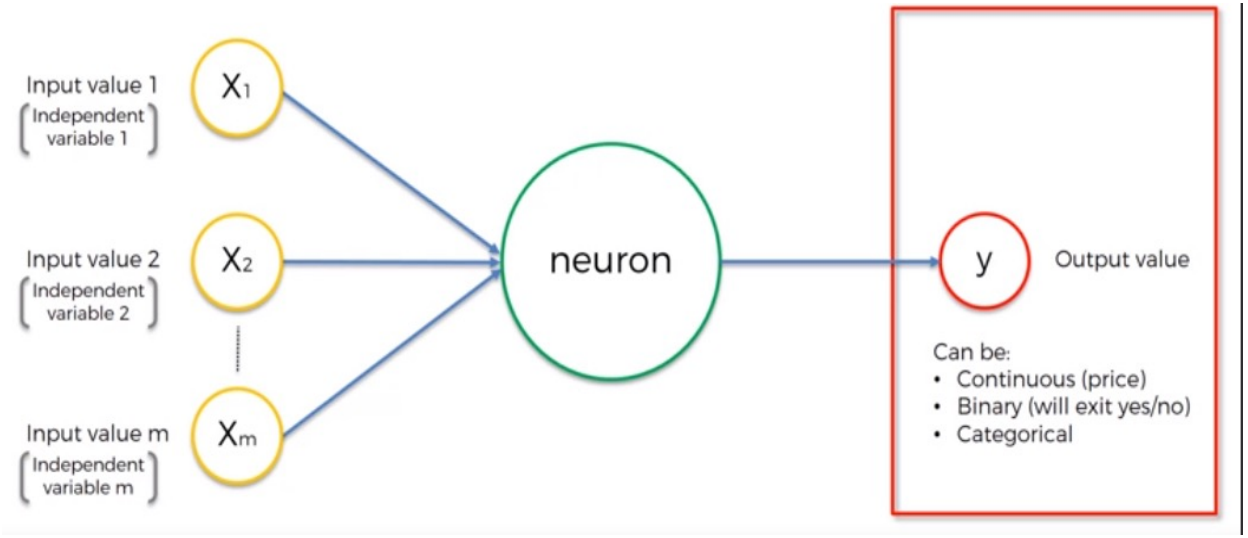


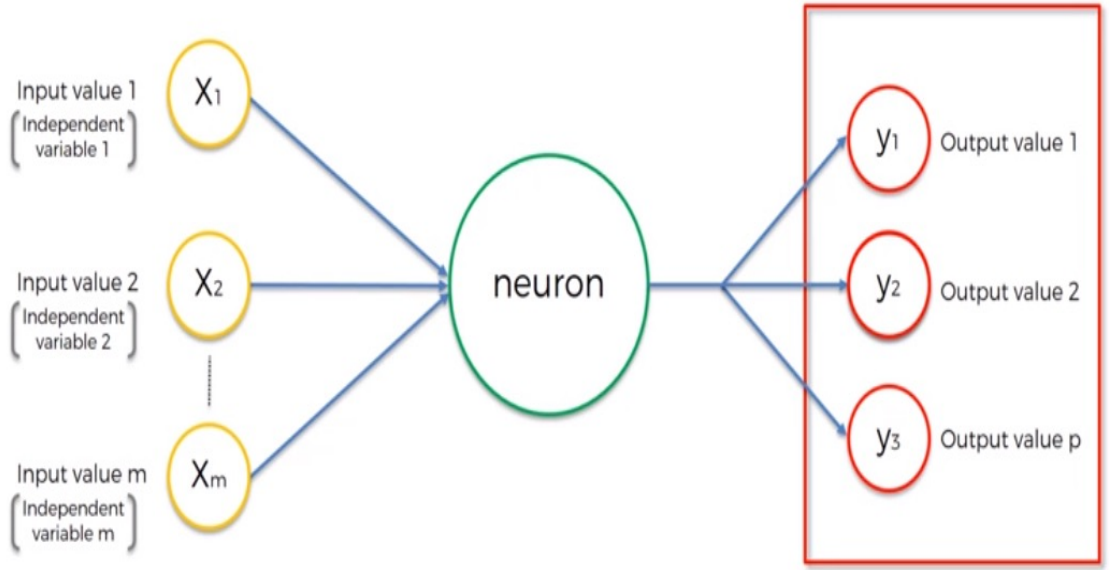
Deep learning- Neuron



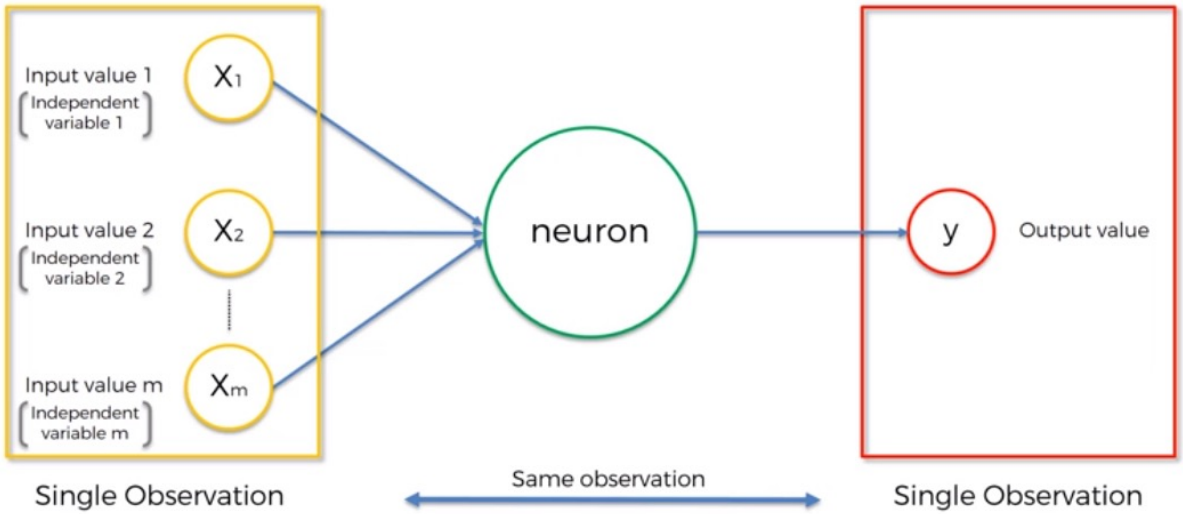


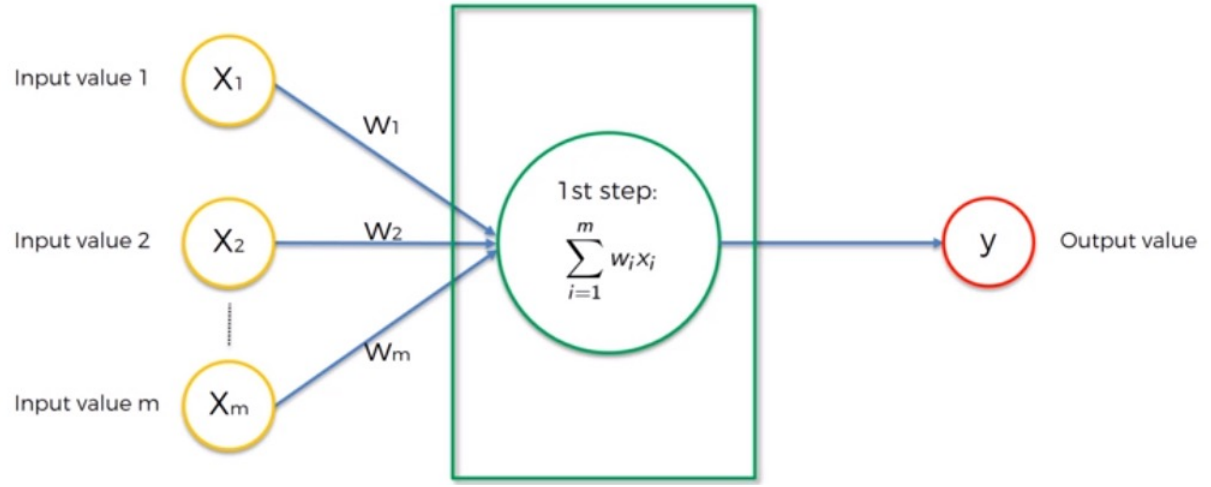
For Categorical target, we will have multiple Output values

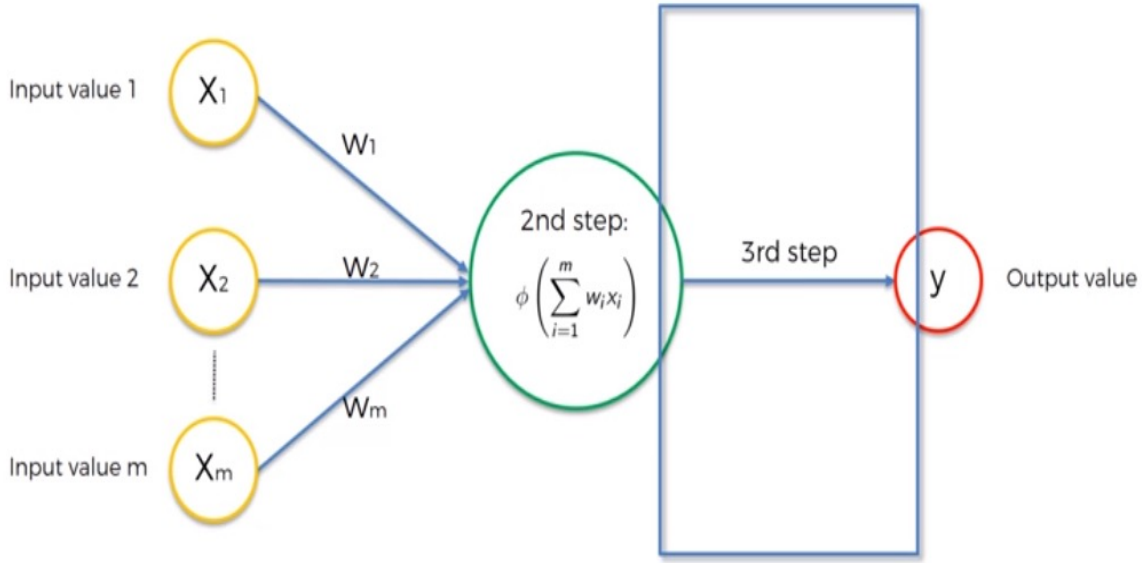




Deep learning- Neuron







- Weights are how neural networks learn by adjusting the weights. The neural network decides in every single case what signal is important, what signal is not important to a certain neuron. What signal gets passed along and what signal doesn't get passed along, or to what strength, to what extent signals get passed along. So weights are crucial.
- Signals go into the neuron and the weighted sum of all of the input values gets added in the first step and then it applies an activation function to this neuron or to this whole layer on top of this weighted sum and then from that, the neuron understands if it will either pass on a signal or it won't pass the signal on.

Deep learning- The Activation Function

Elements of a Neural Network

Input Layer: This layer accepts input features. It provides information from the outside world to the network, no computation is performed at this layer, nodes here just pass on the information(features) to the hidden layer.

Hidden Layer*:* Nodes of this layer are not exposed to the outer world, they are part of the abstraction provided by any neural network. The hidden layer performs all sorts of computation on the features entered through the input layer and transfers the result to the output layer.

Output Layer: This layer bring up the information learned by the network to the outer world.

What is an activation function and why use them?

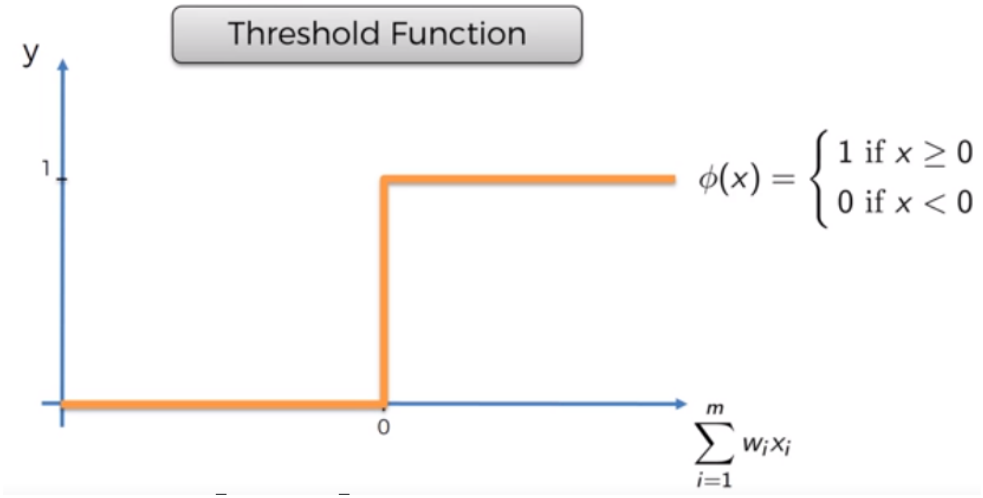
The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

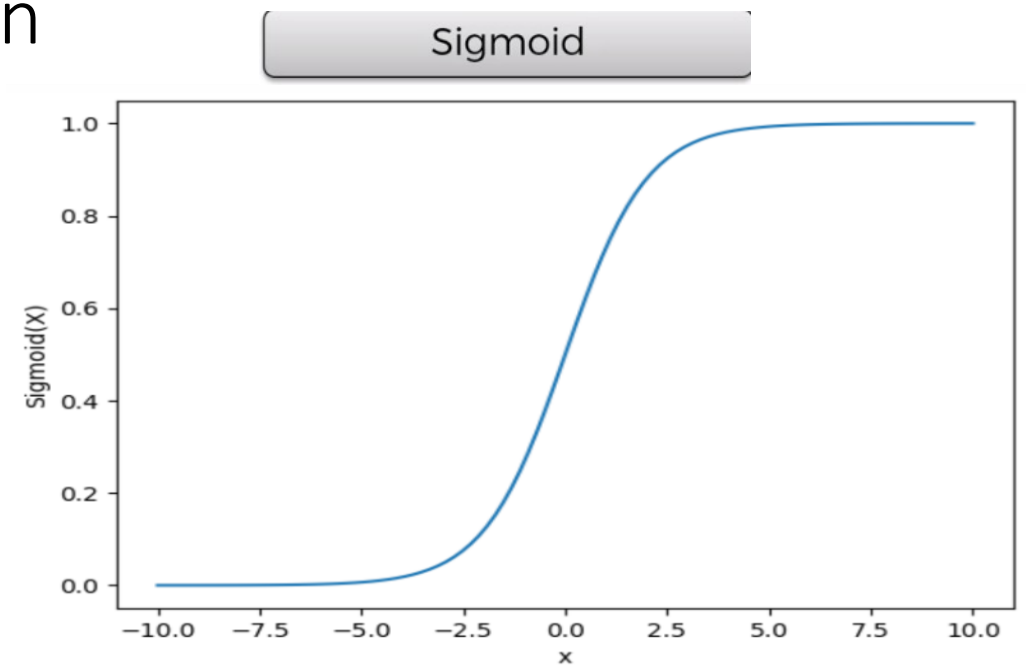
Explanation: We know, the neural network has neurons that work in correspondence with *weight, bias,* and their respective activation function. In a neural network, we would update the weights and biases of the neurons on the basis of the error at the output. This process is known as ***back-propagation***. Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases.

Why do we need Non-linear activation function?

A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks.

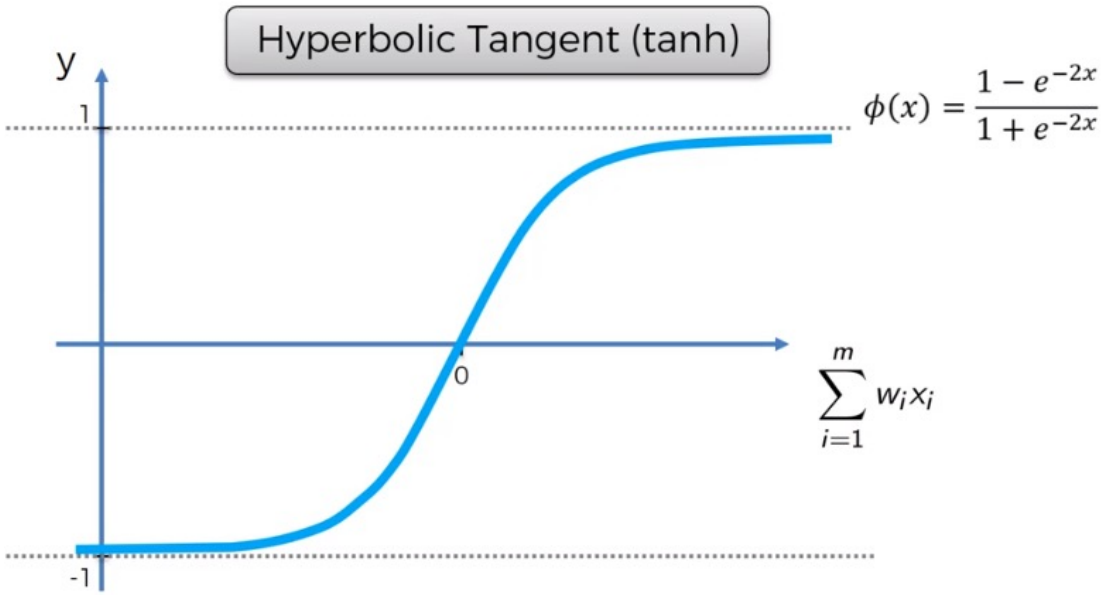
Deep learning- The Activation Function





- It is a function which is plotted as **'S'** shaped graph.
- **Equation :** $A = 1/(1 + e^{-x})$
- **Nature :** Non-linear. Notice that X values lies between -2 to 2, Y values are very steep. This means, small changes in x would also bring about large changes in the value of Y.
- **Value Range :** 0 to 1
- **Uses :** Usually used in output layer of a binary classification, where result is either 0 or 1, as value for sigmoid function lies between 0 and 1 only so, result can be predicted easily to be ***1*** if value is greater than **0.5** and ***0*** otherwise.

Deep learning- The Activation Function

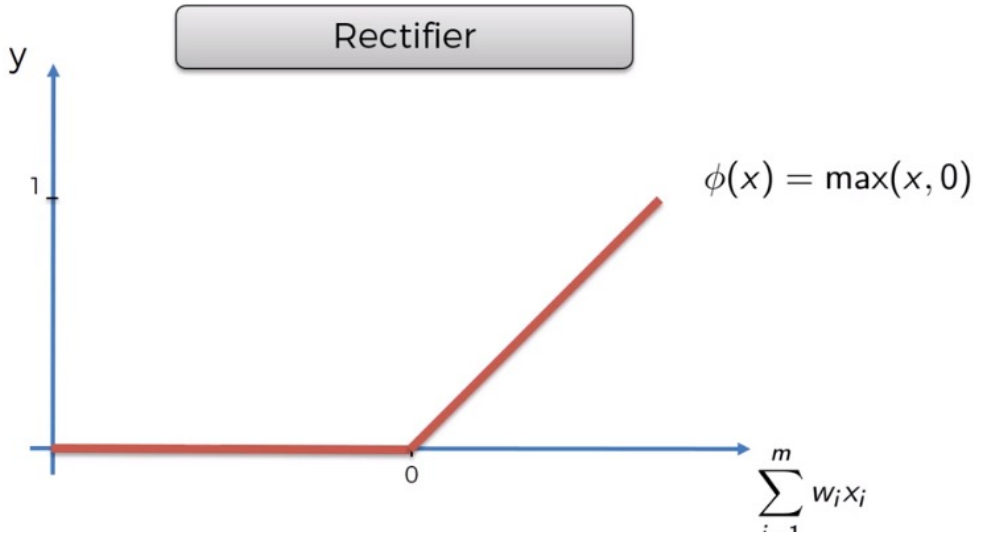


•The activation that works almost always better than sigmoid function is Tanh function also known as **Tangent Hyperbolic function**. It's actually mathematically shifted version of the sigmoid function. Both are similar and can be derived from each other.

•**Value Range :-** -1 to +1

•**Nature :-** non-linear

•**Uses :-** Usually used in hidden layers of a neural network as it's values lies between **-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.



•It Stands for *Rectified linear unit*. It is the most widely used activation function. Chiefly implemented in *hidden layers* of Neural network.

•**Equation :-** ***A(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.

•**Value Range :-** [0, inf)

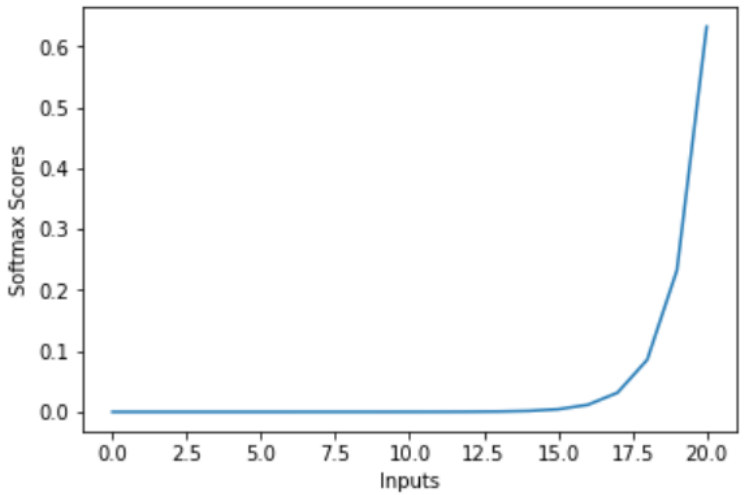
•**Nature :-** non-linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.

•**Uses :-** ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

In simple words, RELU learns *much faster* than sigmoid and Tanh function.

Deep learning- The Activation Function

Softmax Function



The softmax function is also a type of sigmoid function but is handy when we are trying to handle multi- class classification problems.

•**Nature :-** non-linear

•**Uses :-** Usually used when trying to handle multiple classes. the softmax function was commonly found in the output layer of image classification problems.The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs.

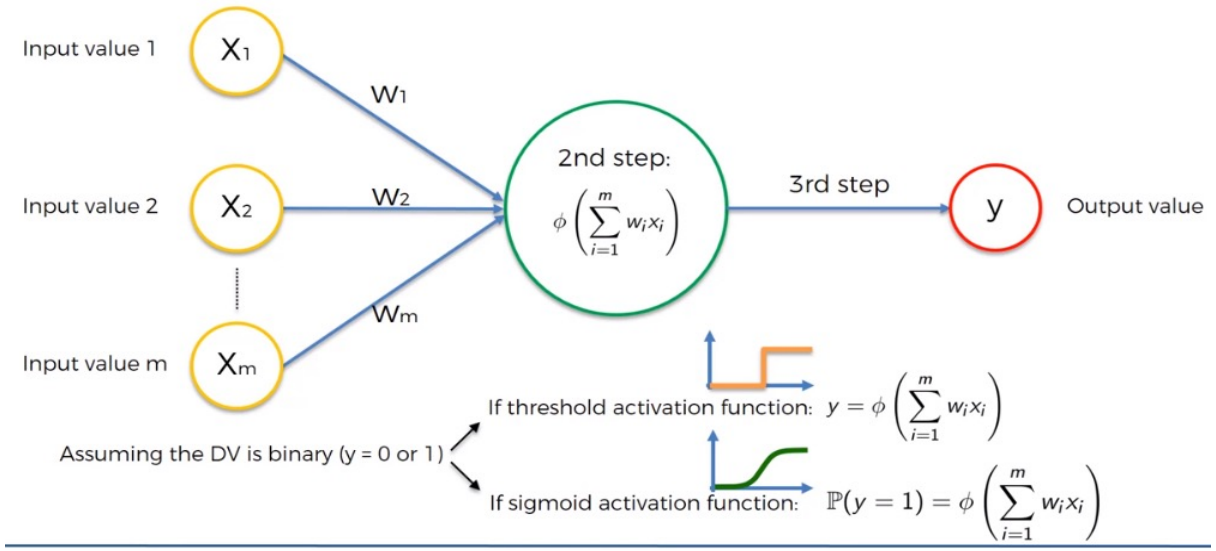
•**Output:-** The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

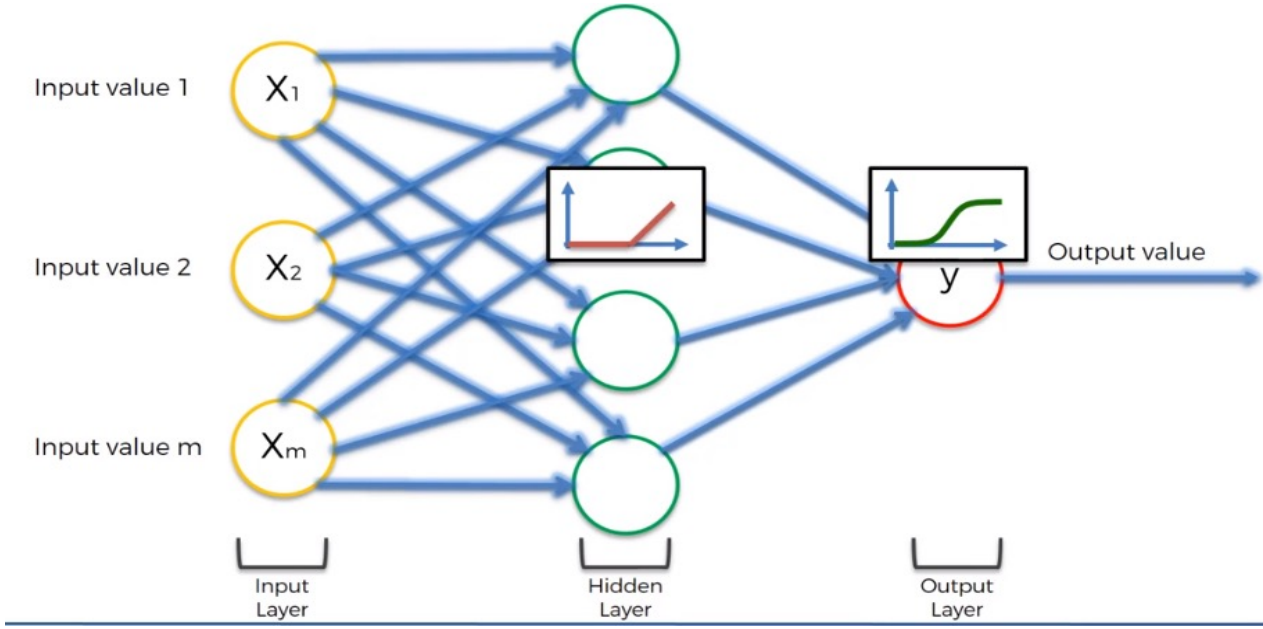
•The basic rule of thumb is if you really don't know what activation function to use, then simply use *RELU* as it is a general activation function in hidden layers and is used in most cases these days.

•If your output is for binary classification then, *sigmoid function* is very natural choice for output layer.

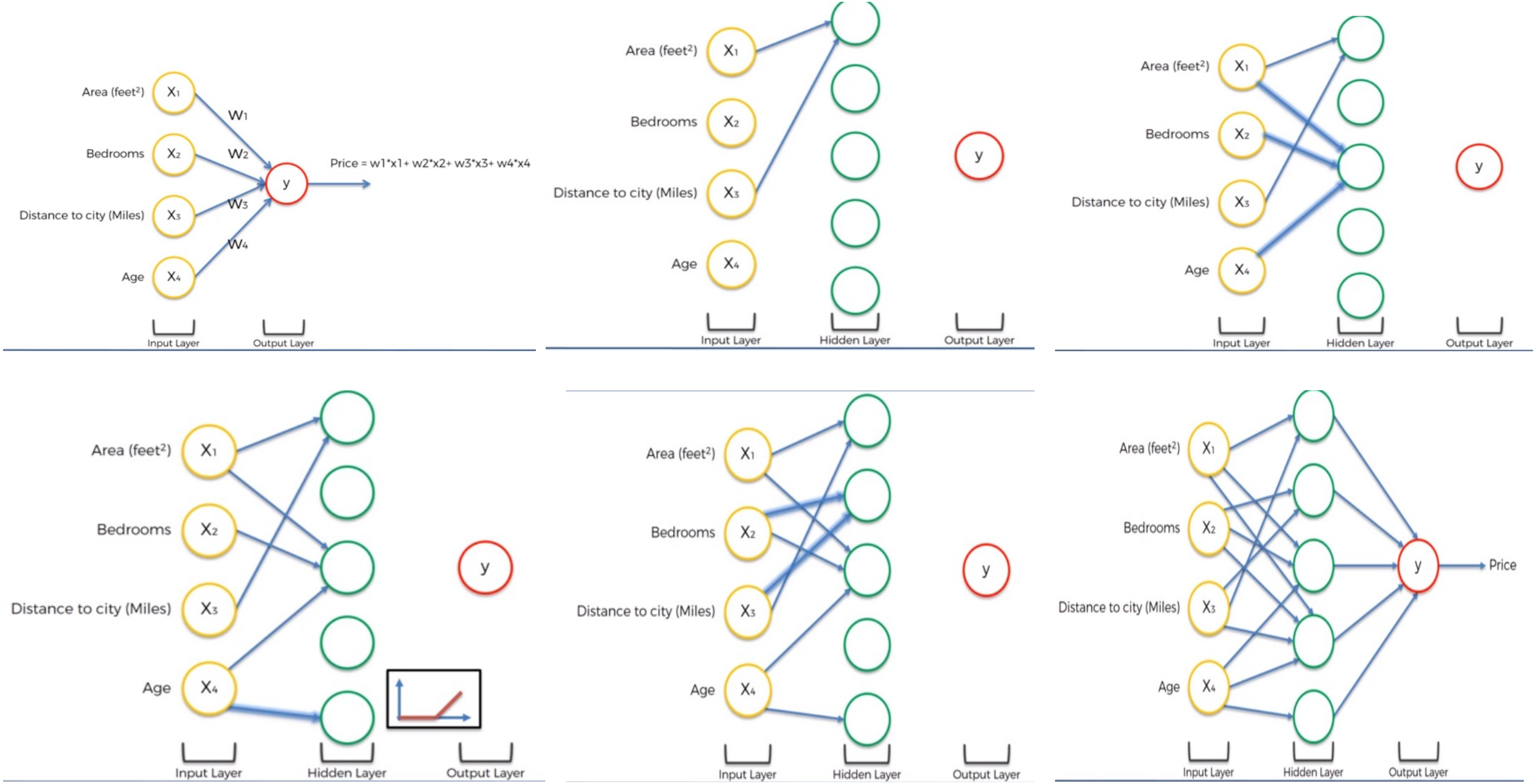
•If your output is for multi-class classification then, Softmax is very useful to predict the probabilities of each classes.

Deep learning- The Activation Function

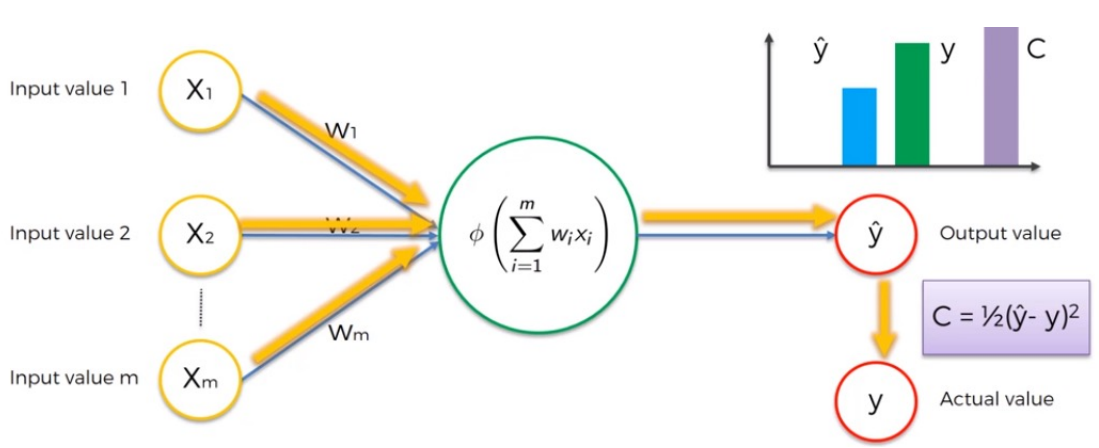


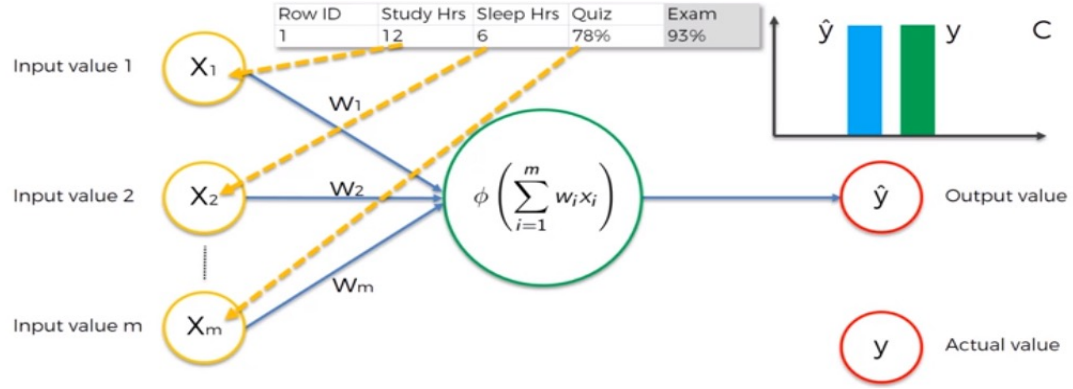


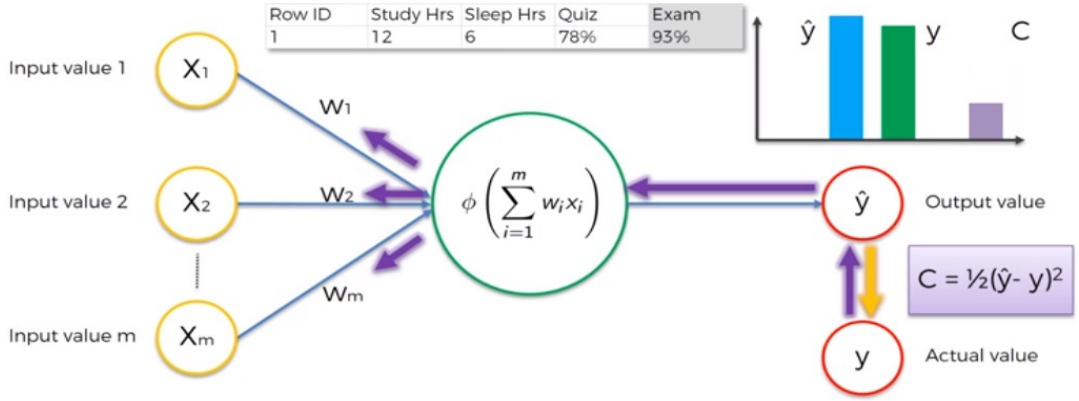
How do Neural Networks Work?

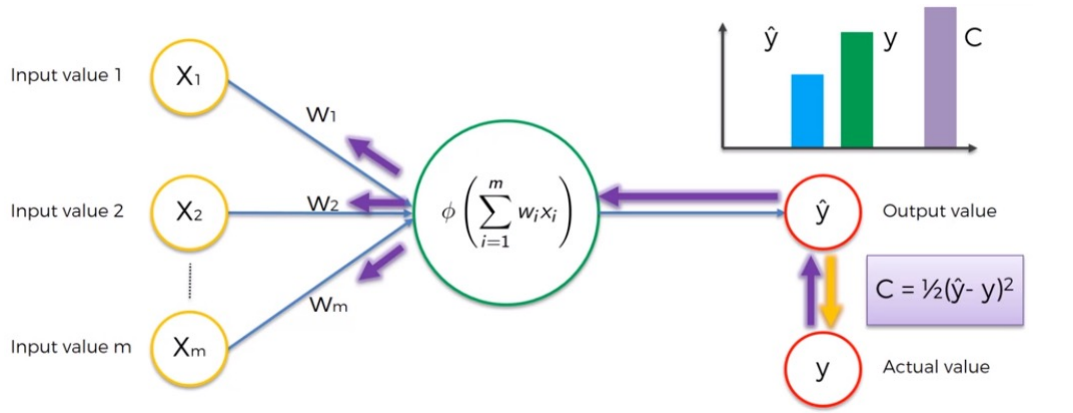


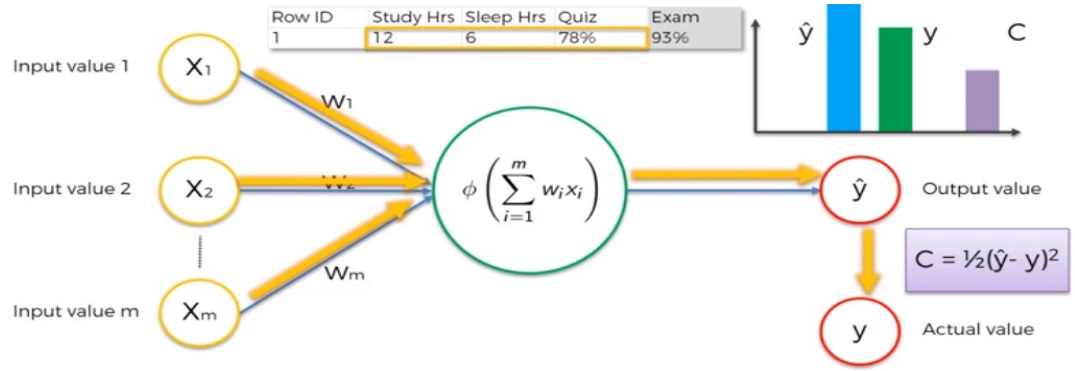
How do Neural Networks Learn?

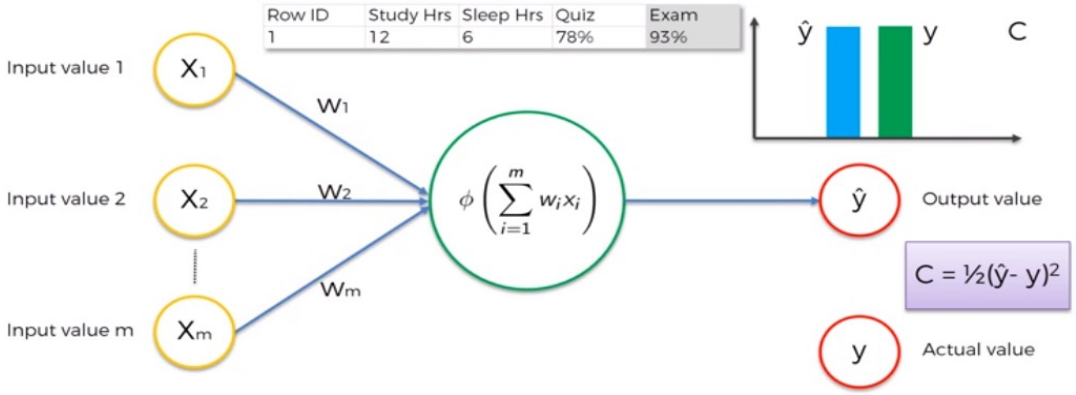




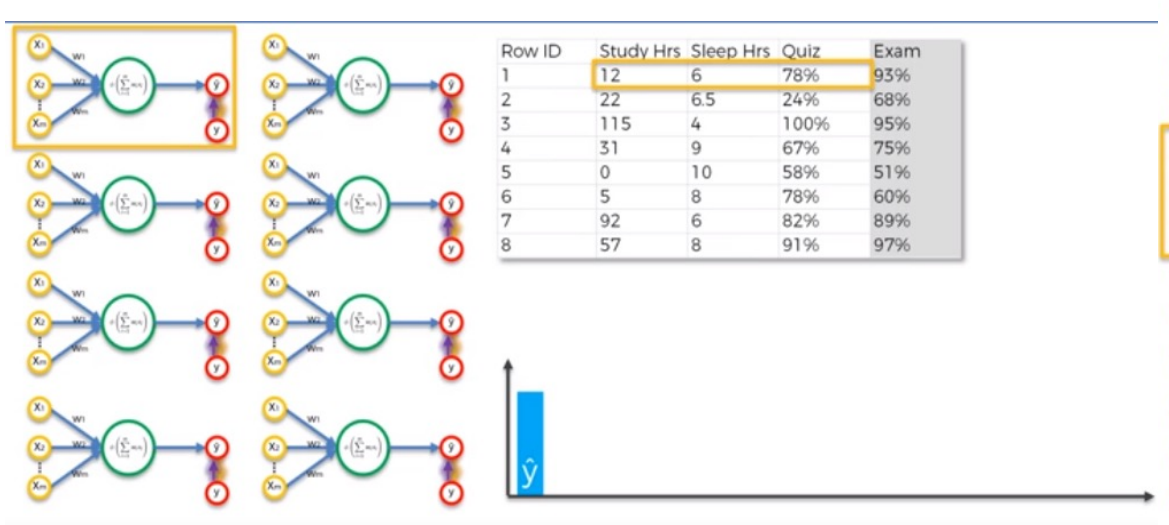




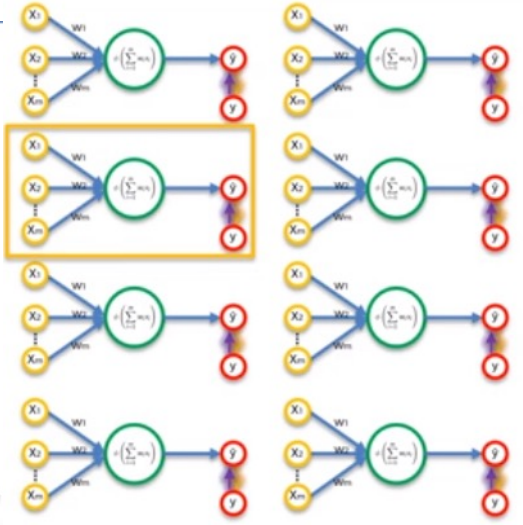




How do Neural Networks Learn?

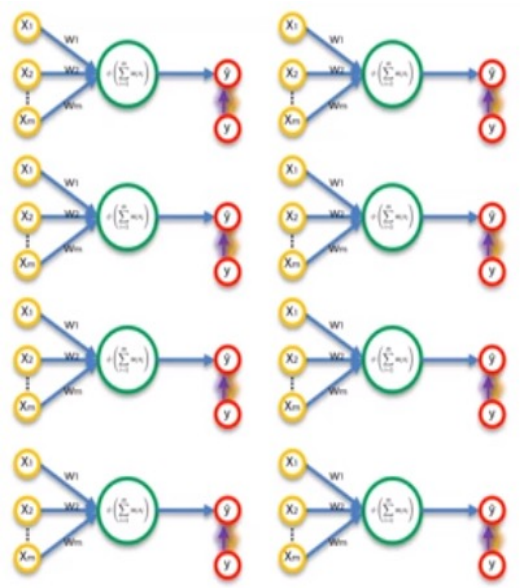


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%



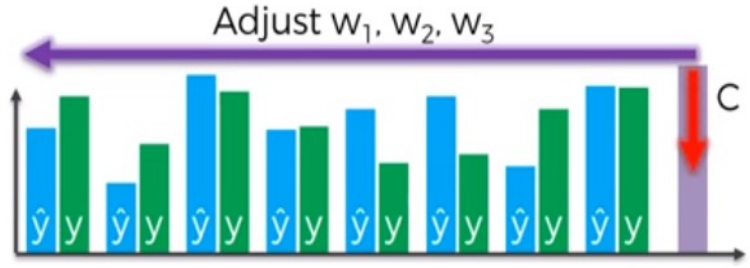
Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

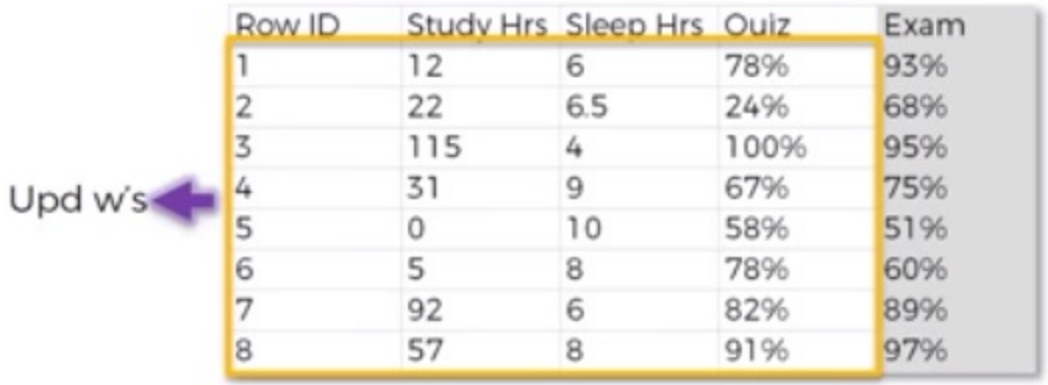


Row ID	Study Hrs	Sleep Hrs	Quiz	Exam
1	12	6	78%	93%
2	22	6.5	24%	68%
3	115	4	100%	95%
4	31	9	67%	75%
5	0	10	58%	51%
6	5	8	78%	60%
7	92	6	82%	89%
8	57	8	91%	97%

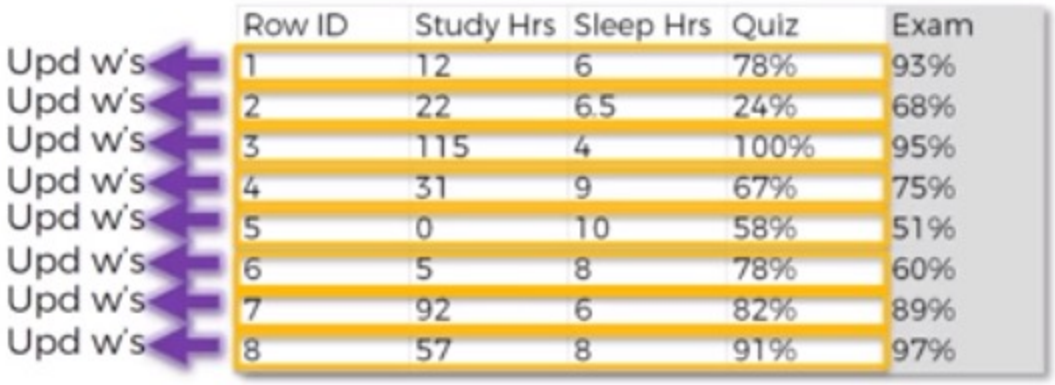
$C = \sum \frac{1}{2}(\hat{y} - y)^2$



Gradient Descent vs Stochastic Gradient Descent

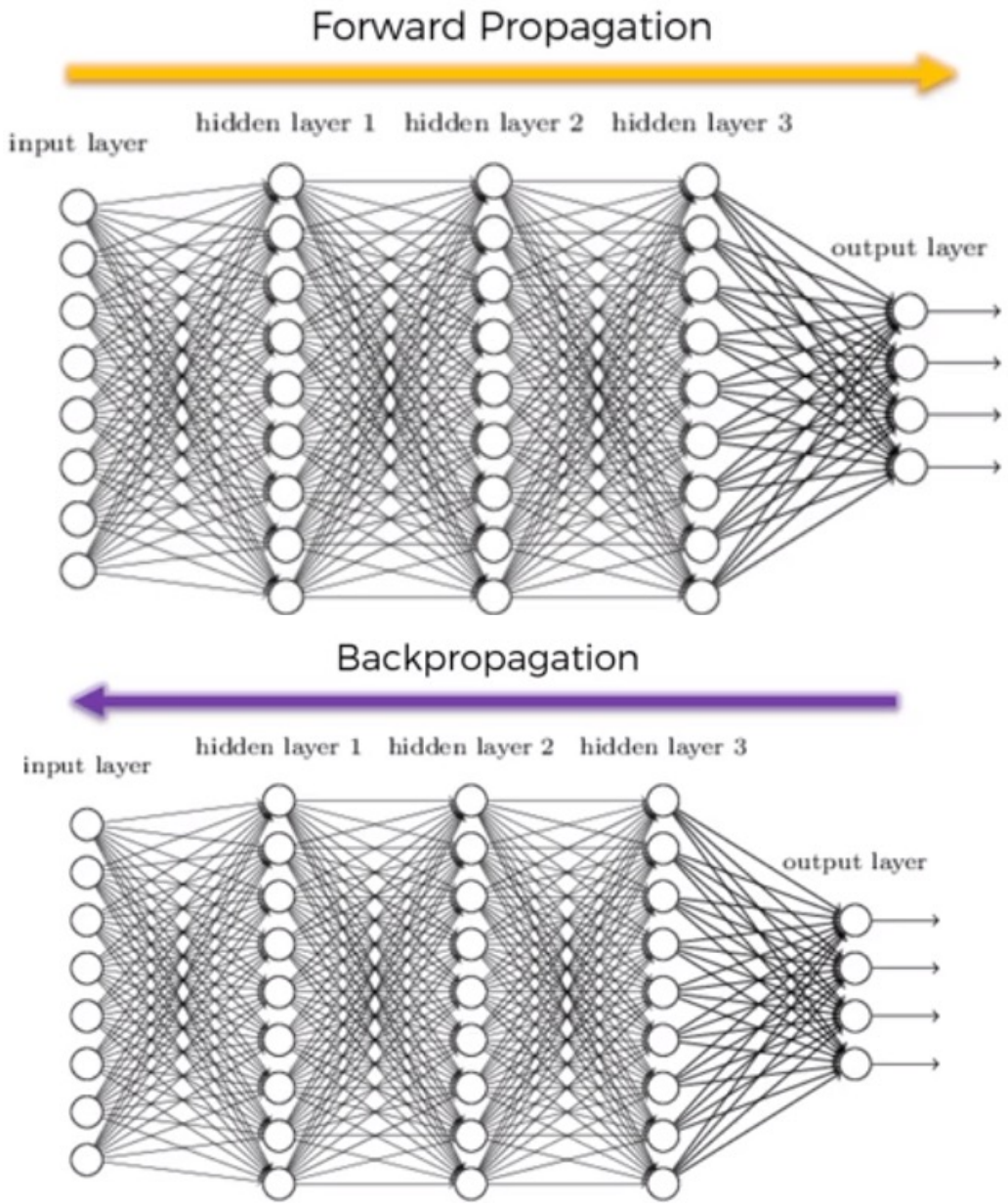


Batch Gradient Descent



Stochastic Gradient Descent

BackPropagation



STEP 1: Randomly initialise the weights to small numbers close to 0 (but not 0).

STEP 2: Input the first observation of your dataset in the input layer, each feature in one input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.

STEP 4: Compare the predicted result to the actual result. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated. Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 6: Repeat Steps 1 to 5 and update the weights after each observation (Reinforcement Learning). Or: Repeat Steps 1 to 5 but update the weights only after a batch of observations (Batch Learning).

STEP 7: When the whole training set passed through the ANN, that makes an epoch. Redo more epochs.

Number of Neurons In Input and Output Layers

The number of neurons in the input layer is equal to the number of features in the data and in very rare cases, there will be one input layer for bias. Whereas the number of neurons in the output depends on whether is the model is used as a regressor or classifier. If the model is a regressor then the output layer will have only a single neuron but in case if the model is a classifier it will have a single neuron or multiple neurons depending on the class label of the model.

There are many rule-of-thumb methods for determining the correct number of neurons to use in the hidden layers, such as the following:

The number of hidden neurons should be between the size of the input layer and the size of the output layer.
The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
The number of hidden neurons should be less than twice the size of the input layer.

Moreover, the number of neurons and number layers required for the hidden layer also depends upon training cases, amount of outliers, the complexity of, data that is to be learned, and the type of activation functions used.

Most of the problems can be solved by using a single hidden layer with the number of neurons equal to the mean of the input and output layer. If less number of neurons is chosen it will lead to underfitting and high statistical bias. Whereas if we choose too many neurons it may lead to overfitting, high variance, and increases the time it takes to train the network.