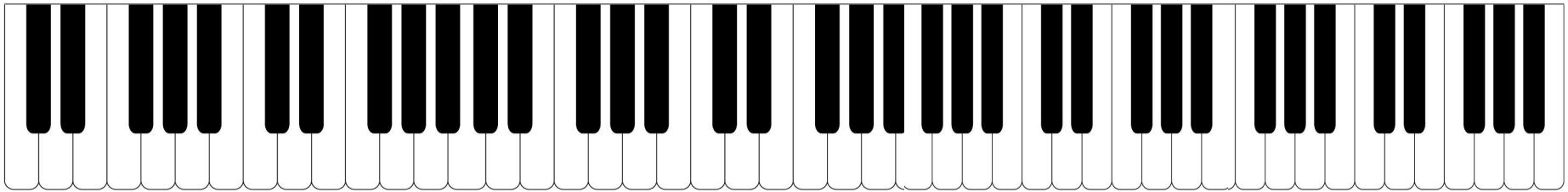
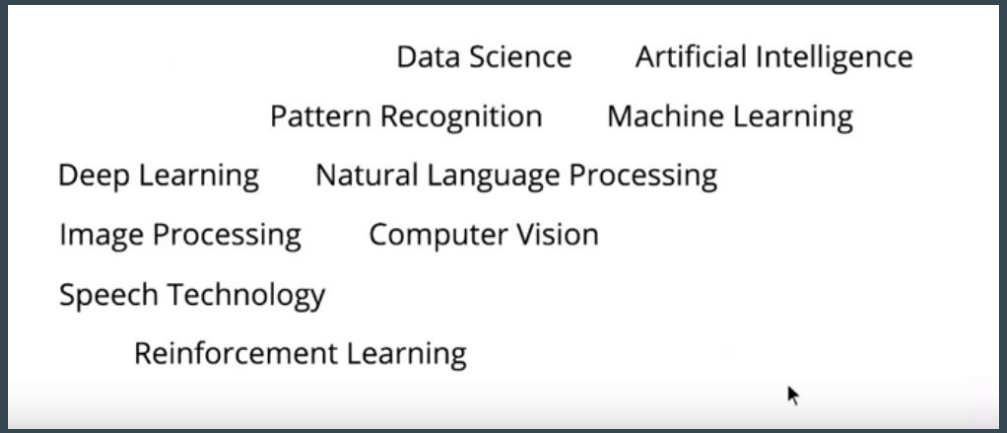
Music Generation Mini Project

By the Analytics Club



Before we dive in..



I am sure most of you would have come across these terms but are you sure what do they mean exactly?

Clearing up the Jargon

Let's start by understanding what is AI?

Artificial Intelligence means getting a computer to mimic human behaviour in some way.
So every human has a set of abilities (vision, hearing, etc) and translating this to computers - computer vision, NLP, etc. Similarly every there are some tasks involving the use of these abilities like classification, object detection, converting speech to text, etc. Likewise there are different methods to solve these tasks like Machine Learning, Deep Learning, etc.

Hence AI basically encompasses all of this. If someone says he/she is utilising an AI driven system then it means they are using any one of these methods to solve a specific task.

Machine Learning?

Machine learning as a method that is focused on the development of computer programs that can access data and learn from it automatically, without human assistance or intervention. The entire machine learning concept is based on the assumption that we should give machines access to information and let them learn from it.

Deep Learning?

Deep Learning is a subset of machine learning dealing specifically with neural networks and other complex techniques to help the AI learn. The main part where deep learning comes into play is when there is too much or too complex data for an ML model to handle. Just as the name suggests a deep learning model aims to replicate how the human brain thinks and learns using neurons.

ARTIFICIAL INTELLIGENCE

IS NOT NEW

ARTIFICIAL INTELLIGENCE

Any technique which enables computers to mimic human behavior

MACHINE LEARNING

AI techniques that give computers the ability to learn without being explicitly programmed to do so

DEEP LEARNING

A subset of ML which make the computation of multi-layer neural networks feasible

1950's 1960's 1970's 1980's 1990's 2000's 2010s

ORACLE

So where does Data Science come under all this?

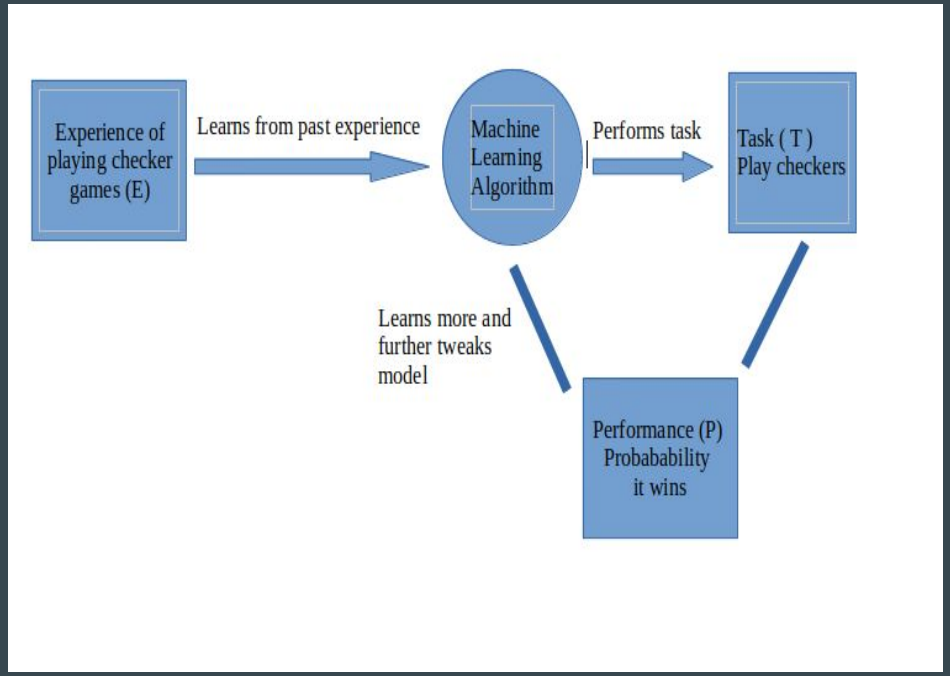
Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. Now anything that requires data and if we perform some kind of “science” or rather a systematic study of the data to provide insights is called data science. Now if this was the case doesn’t NLP or computer vision all come under this umbrella. Well yes! But if you can already be specific by saying that you are working on NLP or you are a computer vision engineer why use a generic term like Data Science.

A formal definition for Machine Learning

A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

WOAH sounds complex? Well, actually not! If there is a computer algorithm which can perform a task (T) after learning from some existing data (basically experience E) and if its performance (P) improves with more data (more experience) we call it a Machine learning algorithm.

A simple example

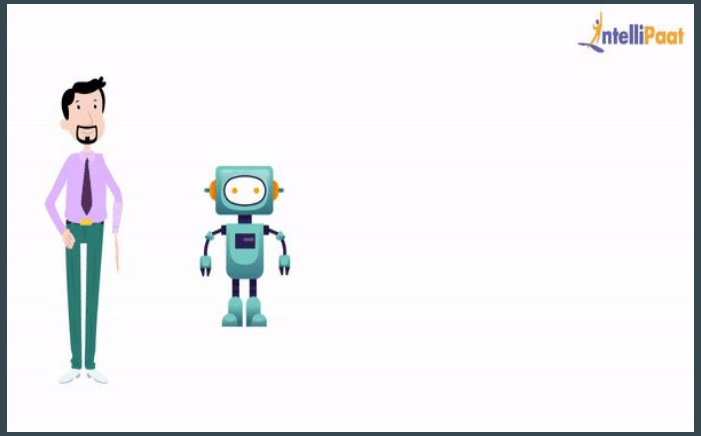


- Suppose we have to teach a computer to play a checkers game. The task (T) is to play a checkers game.
- Now to train the model we should have data of previous checkers games which is going to act as our experience E from which the algorithm learns.
- Now to evaluate how well the computer plays we keep a performance metric (P) which in this case we define as the probability that the computer wins the game.
- The computer can learn from the game it has played and further tweak the model to perform better.
- The process of learning happens over multiple iterations and with each iteration, the model should perform better.

Supervised and Unsupervised Learning

A Machine learning problem can be assigned into two broad categories - supervised learning and unsupervised learning.

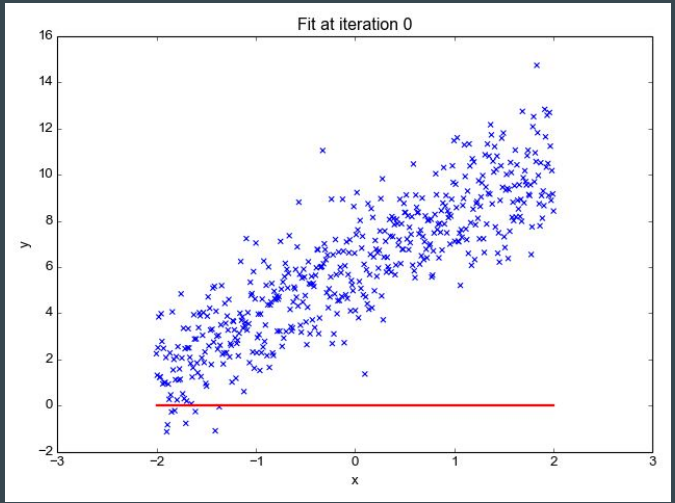
- **Supervised** : In supervised learning, we train a model with data which is already labelled or in simpler words we have a dataset which contains information about what the actual output must look like. So once the model learns from this “Training data” it tries to predict the output for a new set of examples by establishing a relationship between the input and output.
- For example: Given a huge set of images of cats and dogs, we train a model by giving it labelled data of a cat and dog and hence the model learns to identify these features when given a new image.



- Supervised Learning problems can be further classified into two main categories:

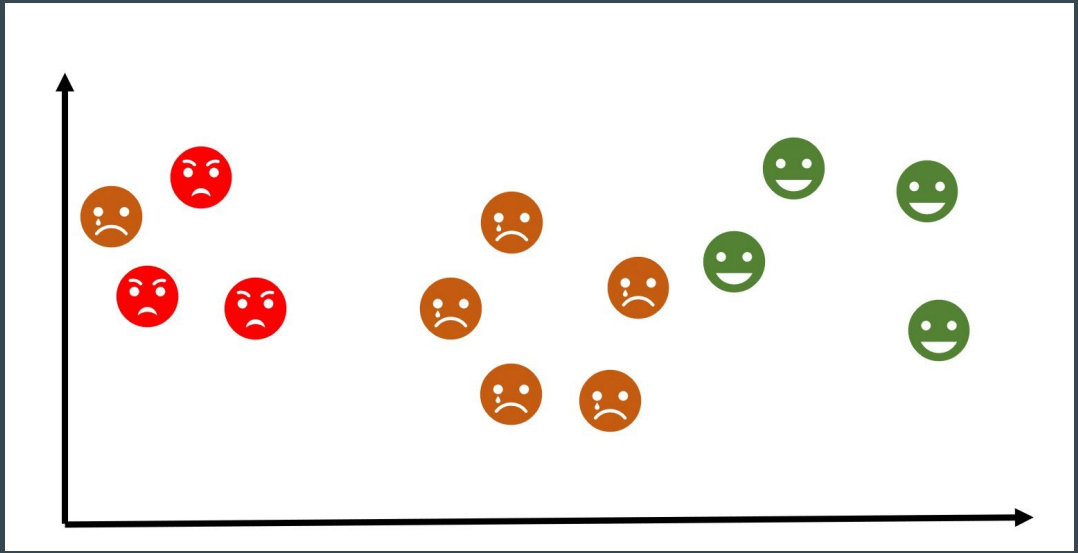
- **Regression Problem**: Whenever we are trying to predict results within a continuous output, meaning we are trying to map some input variables to some continuous function. For example: Given the characteristics of a mobile phone, we need to predict the approximate price of the mobile phone.

- **Classification Problem:** Whenever we are trying to classify the output into specific categories, meaning we need to map some input variables to output some discrete values. For example: Given an image of an animal, we need to classify into whether its a cat or a dog.



Supervised and Unsupervised Learning

- **Unsupervised :** Unsupervised learning is the training of machine using information that is neither classified nor labelled and allowing the algorithm to act on that information without guidance. Here the task of machine is to group unsorted information according to similarities, patterns and differences without any prior training of data.
- For example, A computer doesn't know what expressions are but given the data it tries to group similar expressions together and make some sense out of the data.



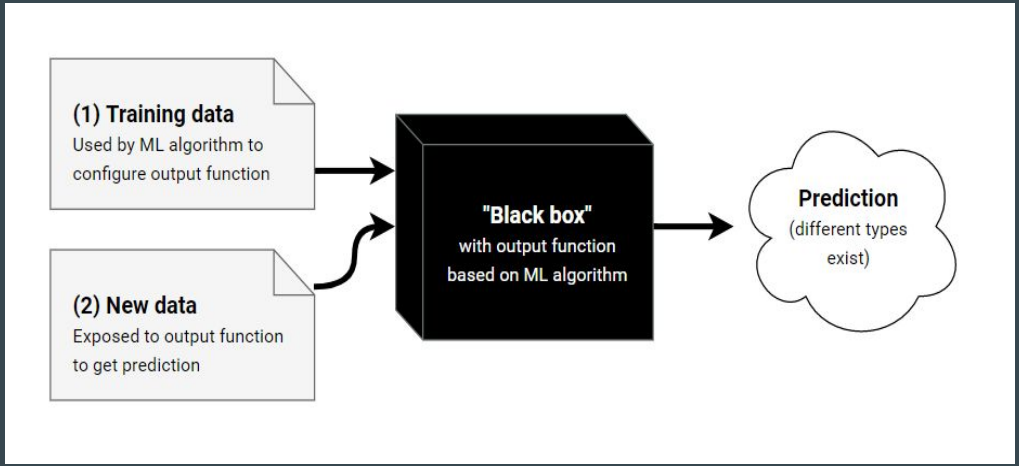
Understanding ML models

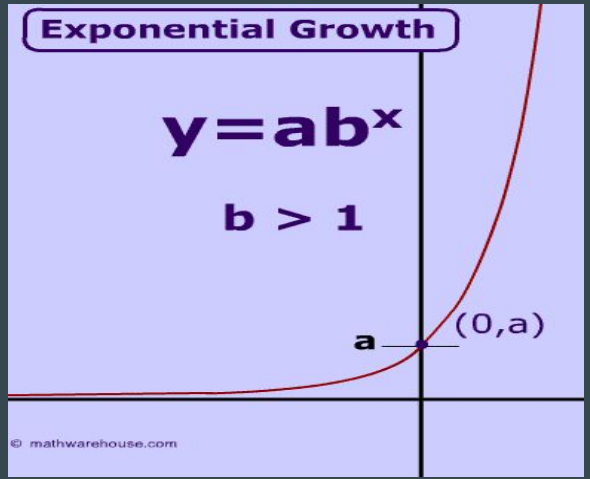
...

Basic Intuition of how an ML model works

Every model takes an input and tries to map the input to the corresponding output. This is done with the help of certain parameters which define the model.

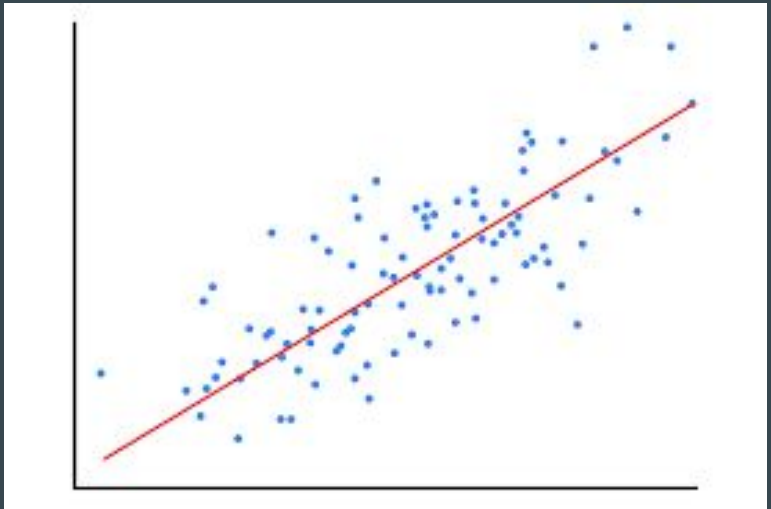
The model needs to find the ideal set of these parameters such that its predictions are as close as possible to the actual set of values.





- To understand this better we take a simple linear relationship between the output and the input.
- A simple linear function is of the form **Y = W*X + b** where X is the given input and Y is the corresponding output.
- Hence the parameters of this model is W and b and the model needs to find the ideal values of W and b such that the predictions are close as possible to the expected output.

Now how does it find these optimal values?



How does it work?

A very simple understanding is that the model is given a set of input data with the corresponding outputs as well. The model learns the best set of parameters using this data and hence this is called the training data and the process is called training.

Once the model is “trained”, unseen data is given to the model to test the models accuracy on the “test data”

The model is said to be ideal when it can perform well on the training data and the test data as well, meaning the model is not biased to either datasets.

Each and every set of data given to the model has a set of **features** which represents the input. In case of a univariate linear regression (as seen in the graph on the prev slide) there is one feature which represents the input.

	distance_to_city_center	rooms	size	price
0	2.4	1.0	19.35	191.565
1	2.4	2.0	13.08	221.568
2	5.0	1.0	24.66	185.936
3	1.9	1.0	24.82	275.502
4	1.9	1.0	25.39	241.205

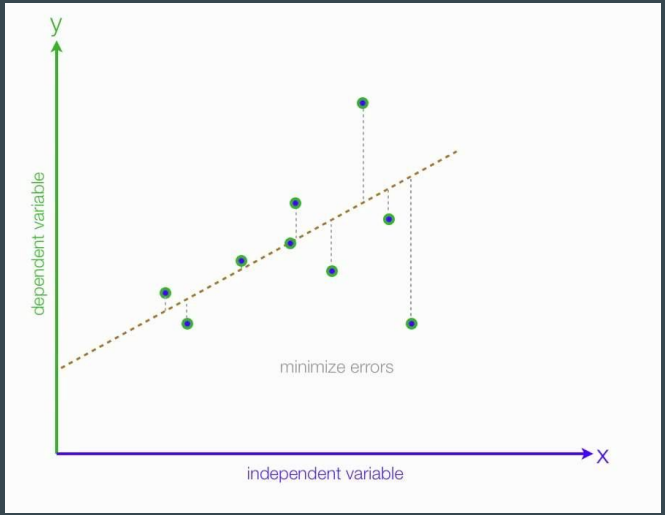
These features can also be called as independent variables and the output for each input the model is called the dependent variable as it depends on the set of features which describe it.

Now how do we find the optimal set of parameters

To find the optimal set of parameters there must be an evaluation metric which determines how well the model is performing and hence based on the performance we change the parameters.

This metric is modelled into a function called cost function. The cost function determines how well the model has fit to the given data.

Hence the model must find the optimal set of parameters by **minimizing the cost.**

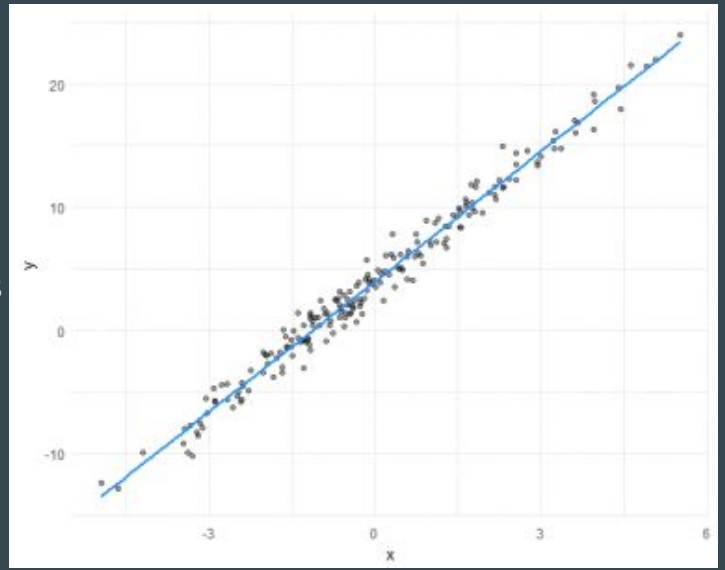


Cost Functions

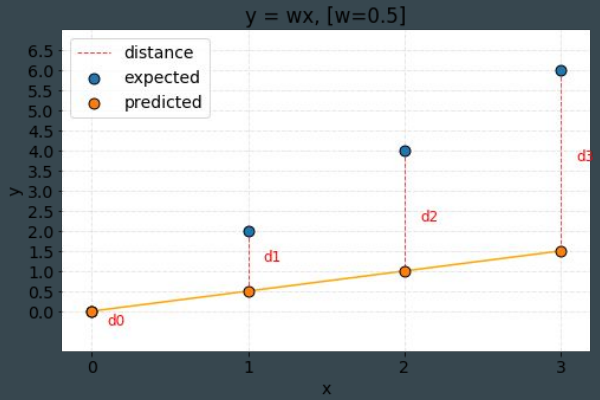
- A **cost function** is a measure of how wrong the model is in terms of its ability to estimate the relationship between X and y.
- We can use a cost function to convey to the computer how poorly its performing and get it to reduce it.

-Coming back to our linear regression model-

- We want the best possible line that passes through these points. If the line predicted by the computer is bad, such as in the previous case, then the cost function should return a very high value. If it predicts a good fit line like this, then it should return a value close to 0.



- That's great! But what can we use as a cost function?
- The answer varies for different problem types, like **Classification** and **Regression**.
- Intuitively what if our loss function calculate the distance between our predicted line and the data points.



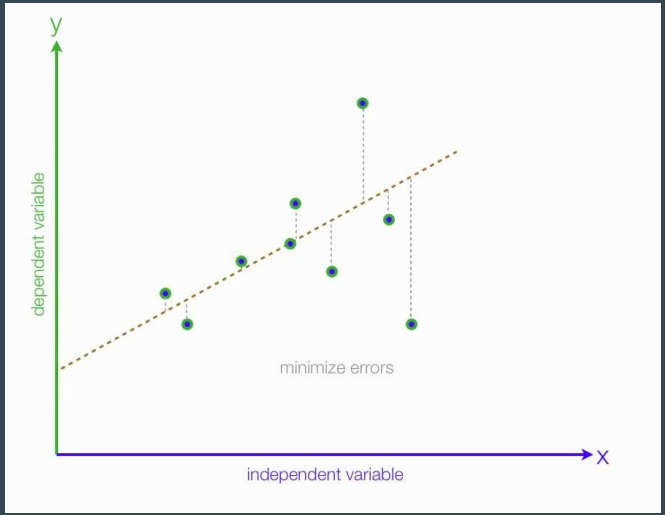
- Our cost function, with the parameters could look like:

$$f(W,b) = \text{average}(\Sigma(Y - (W*x - b))$$

- Our total cost would be, according to the graph:

$$= ((2-0.5) + (4-1) + (6 -3) + (0-0)/4 = -2.25$$

- The cost function returns a negative value, and it would make no sense if we asked a computer to minimize this function. And also,



- What if our line looked like this. Then the positive errors will cancel out with the negative ones.
- Hence we must look for another cost function.

- Mean Absolute Error (MAE) is a regression metric which measures the average magnitude of errors in a group of predictions.

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}^{(i)} - y^{(i)}|$$

where:

- i - index of sample,
- $\hat{y}$ - predicted value,
- y - expected value,
- m - number of samples in dataset.

- Yet still, this is not the cost function used (Any ideas why?)

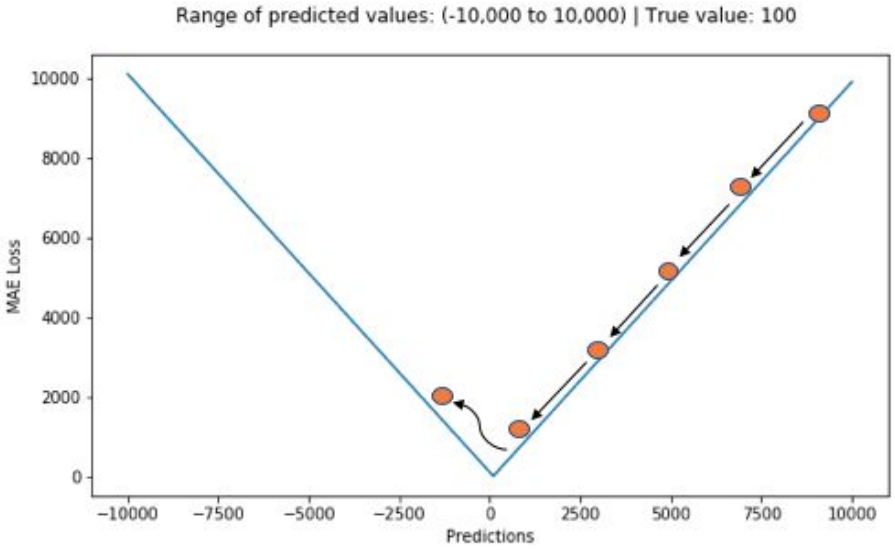
- The cost function used for linear regression is called:
  The Mean Squared Error (MSE)

$$MSE = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

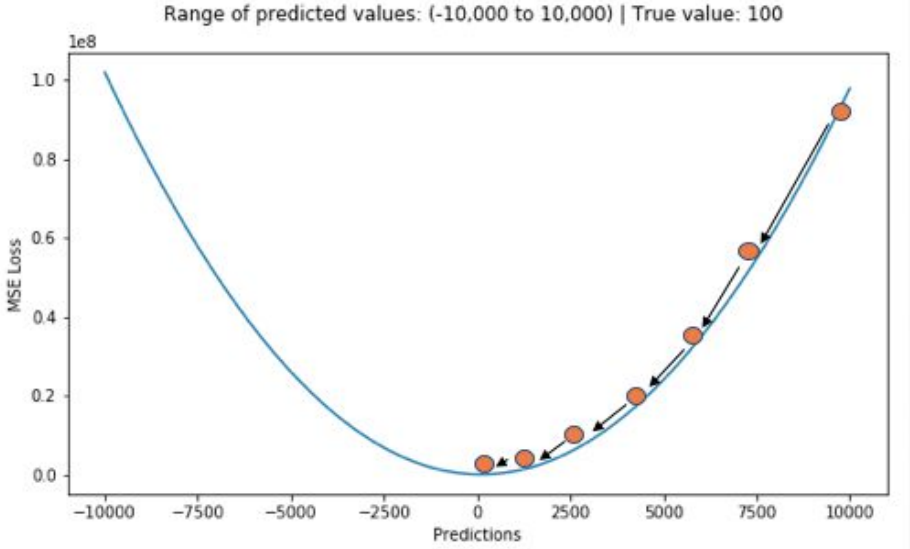
- i - index of sample,
- ŷ - predicted value,
- y - expected value,
- m - number of samples in dataset.

There are different forms of MSE formula, where there is no division by two in the denominator. Its presence makes MSE derivation calculus cleaner.

Graphs of Cost Functions



Graph of the MAE cost function

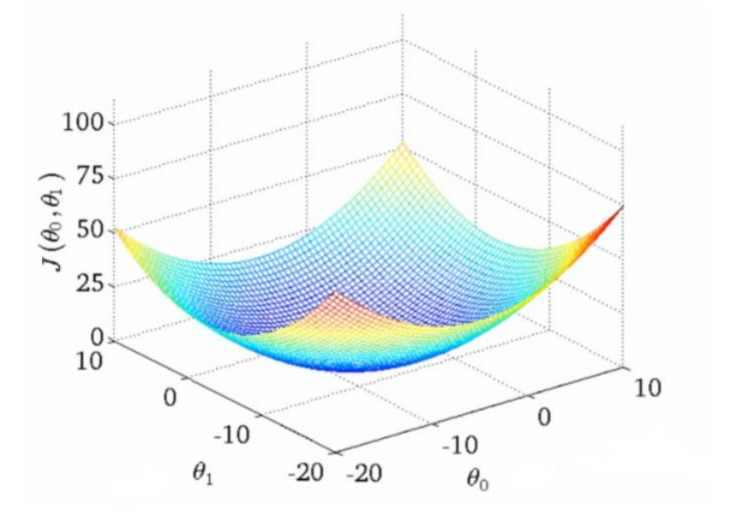


Graph of the MSE cost function

Gradient Descent

In essence, it refers to the updating of the parameters/weights to minimise the total cost.

Now, as you may know, the cost function for a simple regression problem with 2 variables will be a convex figure as follows:



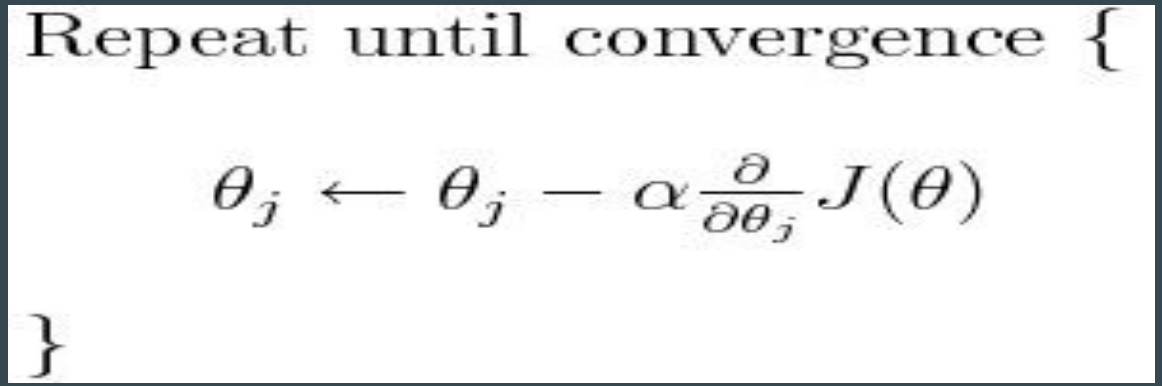
By implementing gradient descent, we wish to update the parameters repeatedly to reach the minima of the convex plot as shown below:

The process of gradient descent involves finding the partial derivatives of the cost function with respect to relevant parameters and using these partial derivatives to modify the parameter themselves to obtain more precise parameters.

We use the partial derivatives for this updation process because, the partial derivatives tell us how the changes of that particular parameter affect the total cost.

With time, we expect that these partial derivatives will approach 0, which would mean that the cost function is moving closer and closer to its minima and that our parameters are approaching their ideal values.

A gradient step for a simple linear regression problem looks as follows:



Where,

J(Θ) is the cost function,

α is the learning rate

Θ is the matrix containing the parameters Θj

Why is the formula working?

The values of partial derivatives (gradient) will give us the direction of maximum increase and hence the negative of the partial derivatives(gradient) will give the direction of maximum decrease of the cost function J(Θ), which is what we wish to achieve.

Also, the magnitude of the partial derivatives(gradient) tells us how far away we are from the minima of the function.

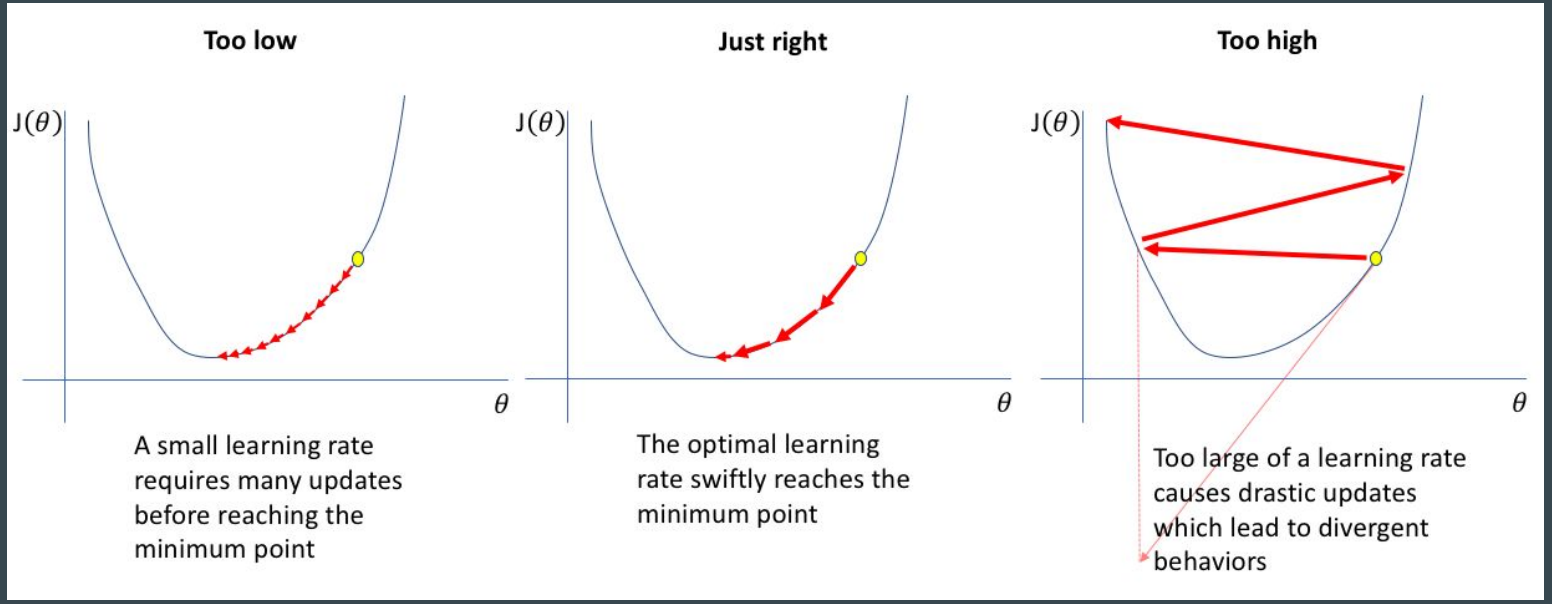
$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

The given statement basically moves the parameters towards the negative gradient which would in turn move the cost function towards the minima.

The distance moved is determined by the partial derivative (which reduces with time, hence reducing the distance moved after each step) and a constant called learning rate(α).

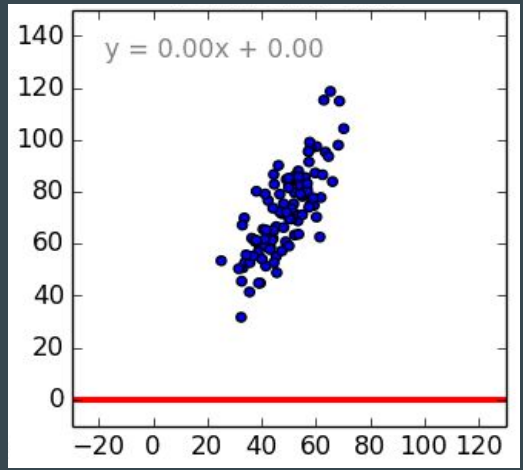
What is the learning rate α ?

The learning rate is a user defined constant that determines the magnitude by which the parameters change during each run of the loop.



Visualising the Training Process

- Coming back to our simple Linear regression problem, we started off with some random values of W and b, and then kept iterating to minimise the cost function, using the Gradient Descent method.

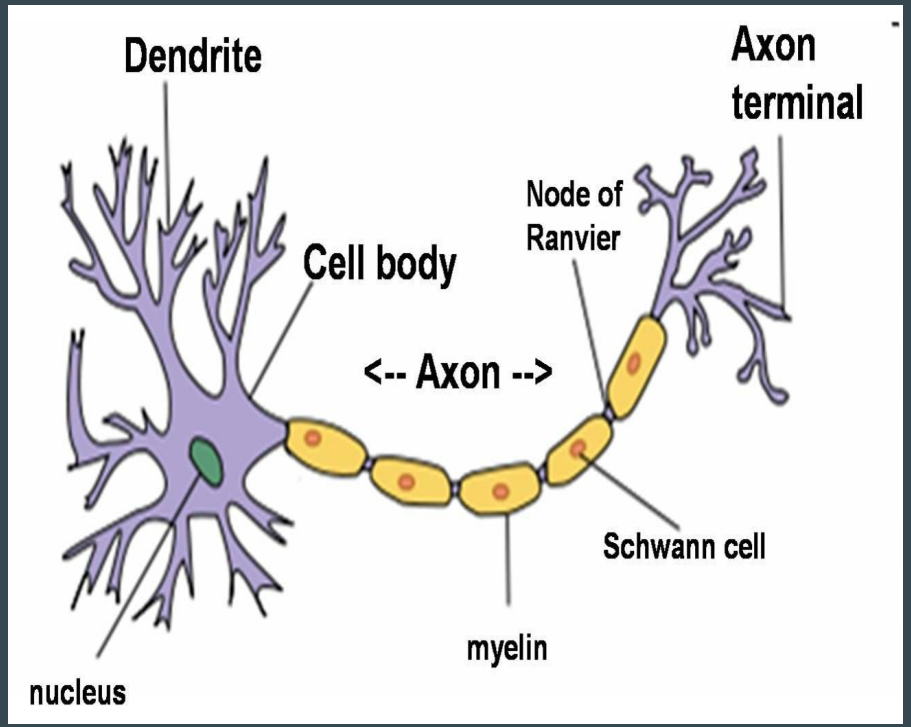


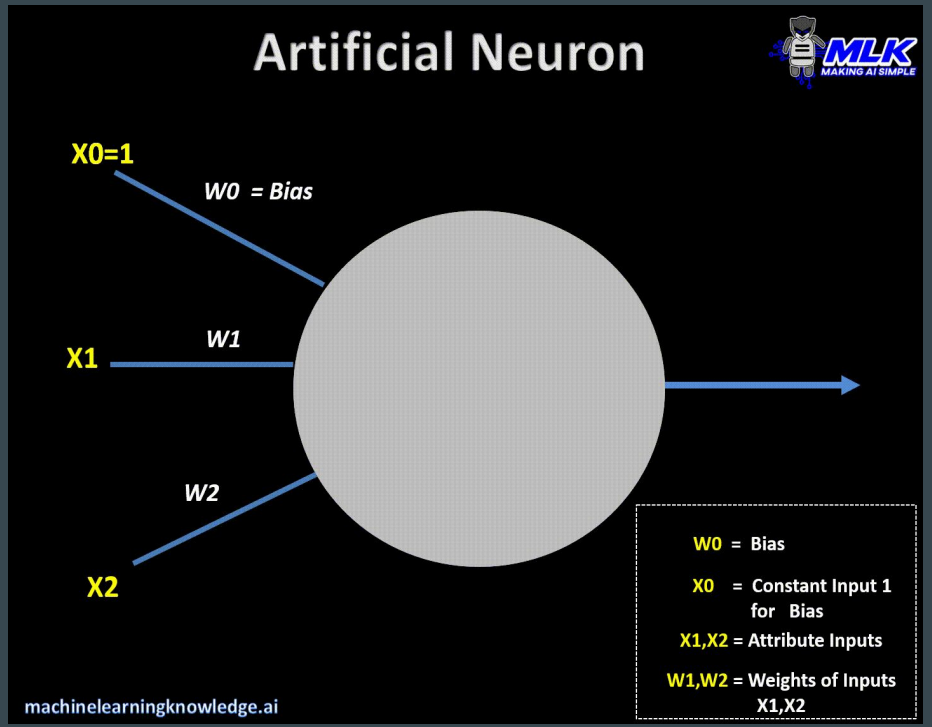
- Yay! Now we get a line which fits it the best. So, we get the ideal values as .... W=1.47, b= 0.03
- This line models the given set of data and can be used for predicting for new cases as well.

Neural Networks

And how they work

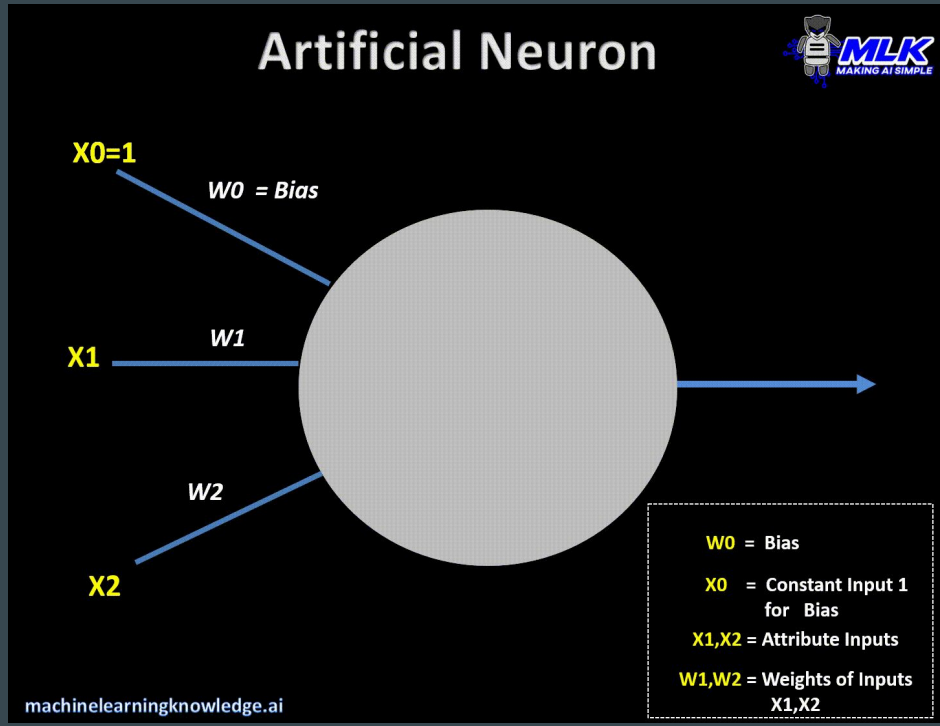
Inspiration from the Human Brain!





Artificial Neurons

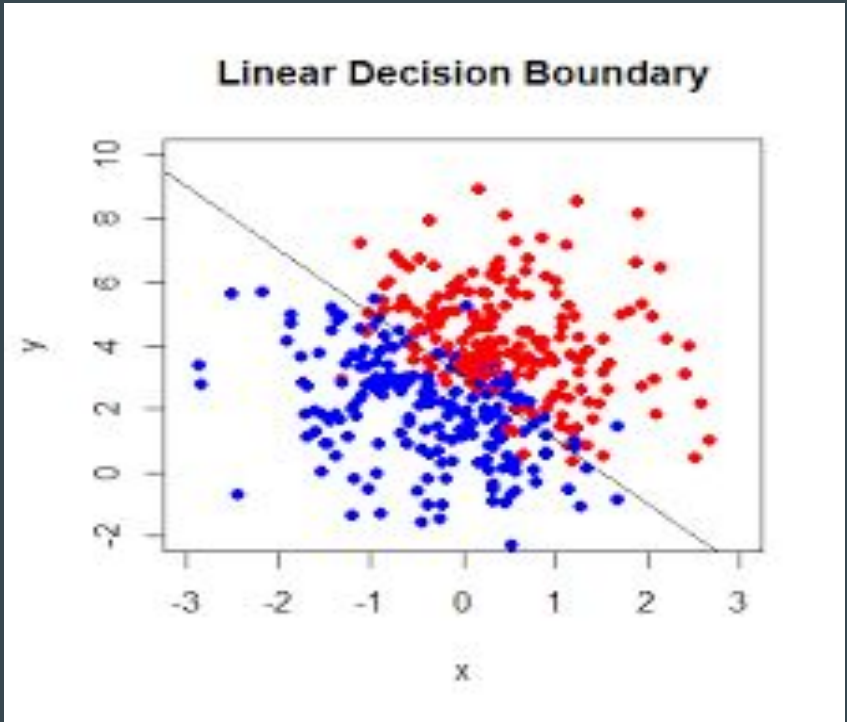
- An artificial neuron is the basic unit of any neural network
- Any neuron has multiple inputs and a single output
- Each input has a certain weight attached with it.
- We multiply each input with its respective weight and add all those products along with a bias term.
- We then pass this through an activation function to get the output from the neuron.

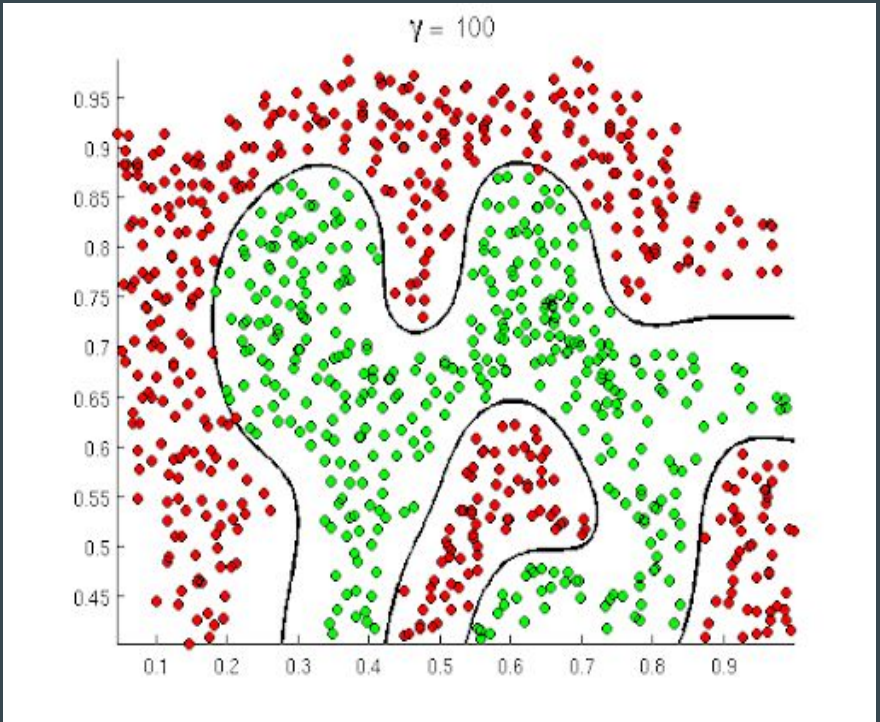


Why Neural Networks?

- Linear Classification models are useful only till a limited extent: After a certain point if we want to increase the complexity of a linear model we end up having too many features!
- E.g. if you decide to include all the 2nd degree polynomial terms in a problem with 1000 base features then we will end up having about 5,00,000 new features to consider!
- In such situations, neural networks come to our rescue as they have an inherent nonlinear nature in their construction.

Need for Nonlinear Hypothesis

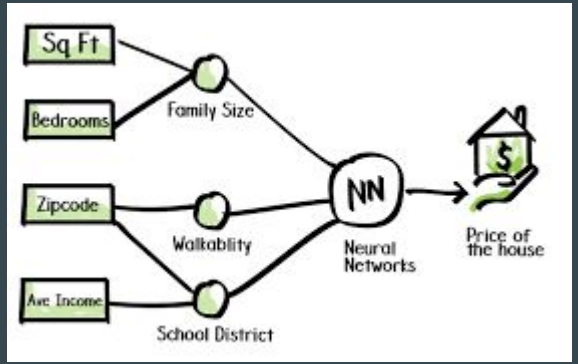




Let's get started with neural networks

Model a function that maps input to output given a training set i.e a set containing inputs with corresponding outputs.

A neural network for predicting housing prices:



- X1,X2 etc are features of your house

  Such as area, number of bedrooms etc which we

  Know and y is the price of the house.

- W1,W2 etc are weights for each of these features i.e they represent how important each feature is in predicting your output.
- Neural networks allow us to express y =f(x) (see fig)using some parameters (weights) that the model will learn once trained on the training set.

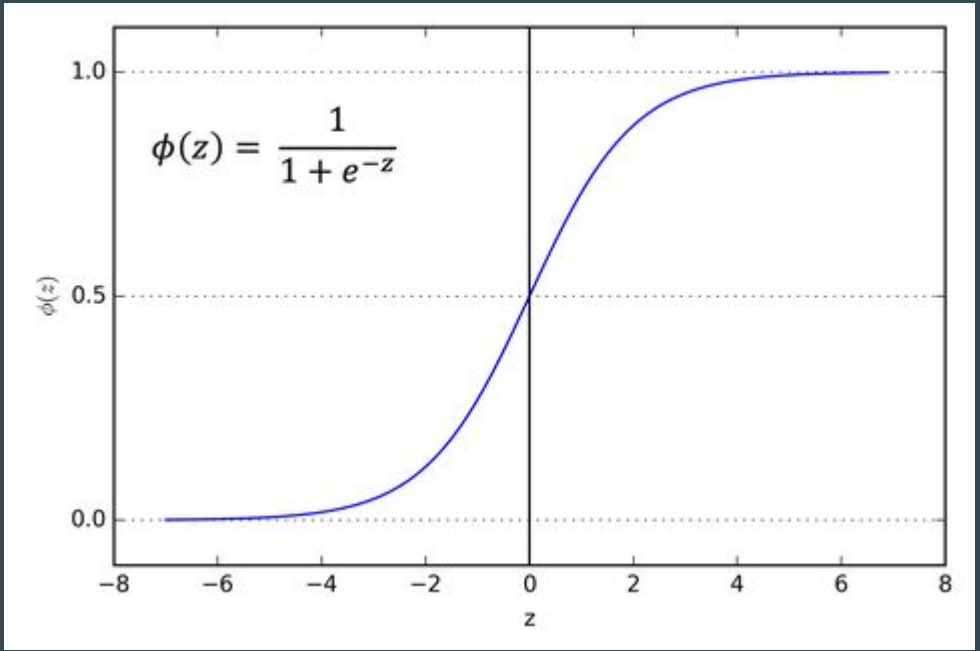
Jargons used when Describing a Neuron

Weights - These are the parameters that is multiplied with your input. Remember how your CGPA is calculated? The credits are the weights and the grades you received is the input.

Bias - This is another parameter that helps the network increase its capacity to solve problems. For a better intuition it is something like the intercept term in the normal linear regression.

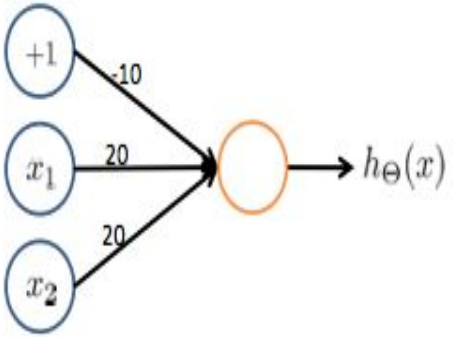
Activation Functions - A transformation which helps us map the input to the output. We use different kinds of activation functions depending on the situation.

The Sigmoid Activation Function



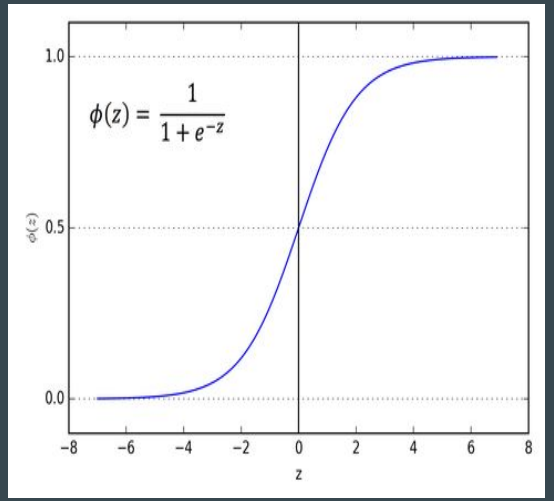
The OR gate!!

Example: OR function

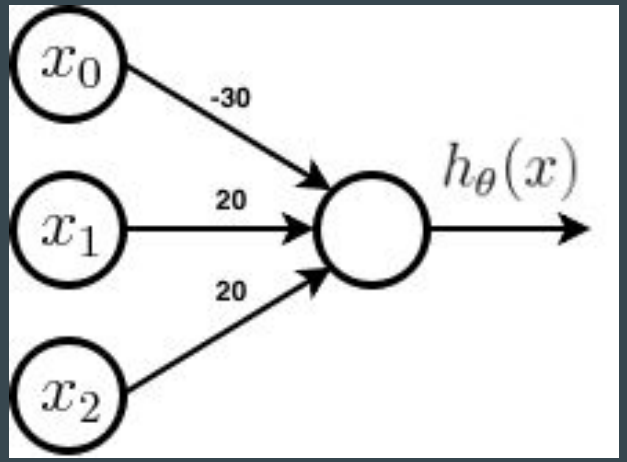


$g(-10 + 20x_1 + 20x_2)$

x_1	x_2	$h_\Theta(x)$
0	0	$g(-10) \approx 0$
0	1	$g(10) \approx 1$
1	0	≈ 1
1	1	≈ 1

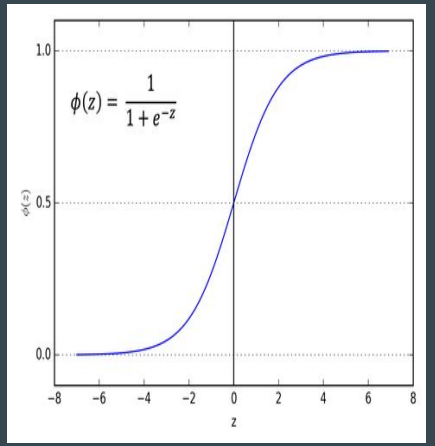


The AND gate!!



g(-30+20x1+20x2)

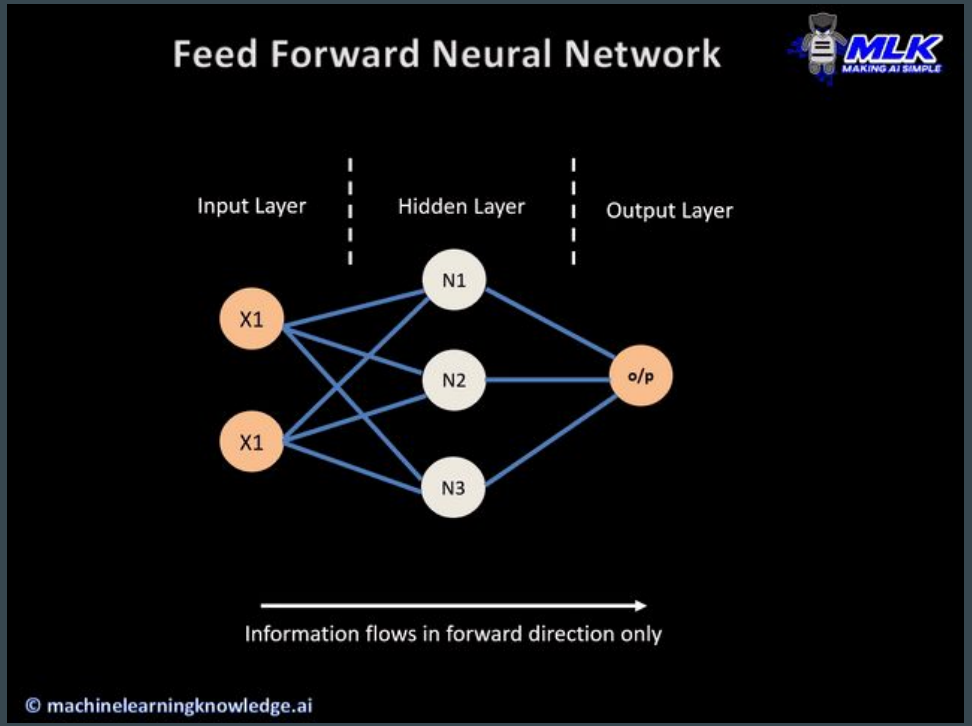
x1	x2	h(x)
0	0	g(-30)=0
1	0	g(-10)=0
0	1	g(-10)=0
1	1	g(10)=1



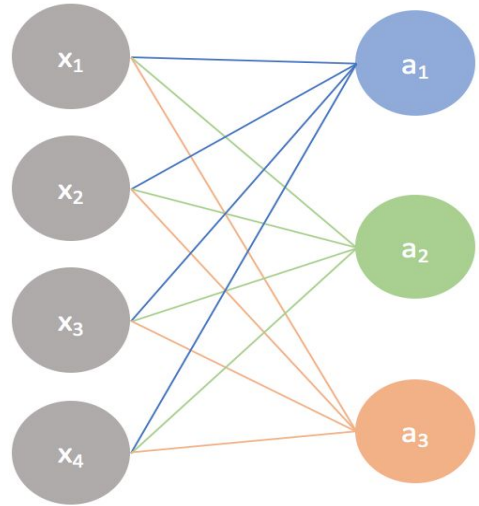
Feed - Forward Neural Network

Feed forward means basically to push the data forward. Each activation is multiplied with the corresponding weight and the sum is fed into the connected neuron in the next layer. A new term coming up.

Dense Layer : What you see in the picture is called a dense layer of neuron. This means that every neuron in that layer is connect to every neuron in the previous layer with a weight



Actual Mathematics Behind a Feed Forward NN

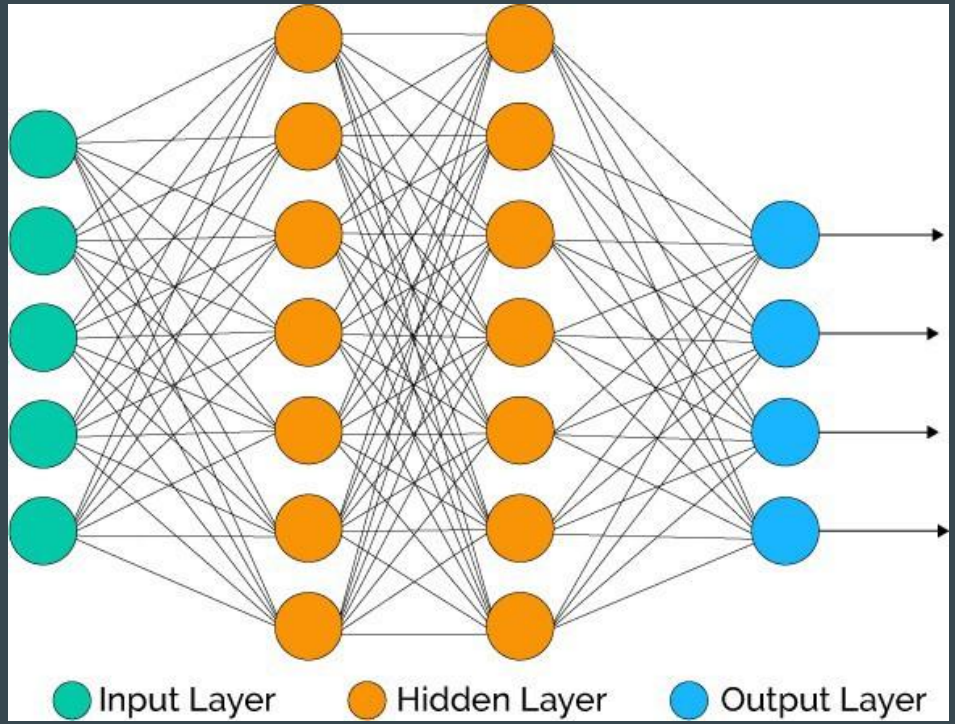


A simple neural network

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b \\ b \\ b \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \\ w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b \end{bmatrix} \xrightarrow{activation} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Layers in a Neural Network

- There are three parts to a neural network
- The input layer. This is where the user inputs their data to process
- The hidden layer. These layers basically try to find hidden patterns in the data and push it forward. There can be any number of hidden layers.
- The output layer. This is the layer from which the output is taken and is in human understandable form.

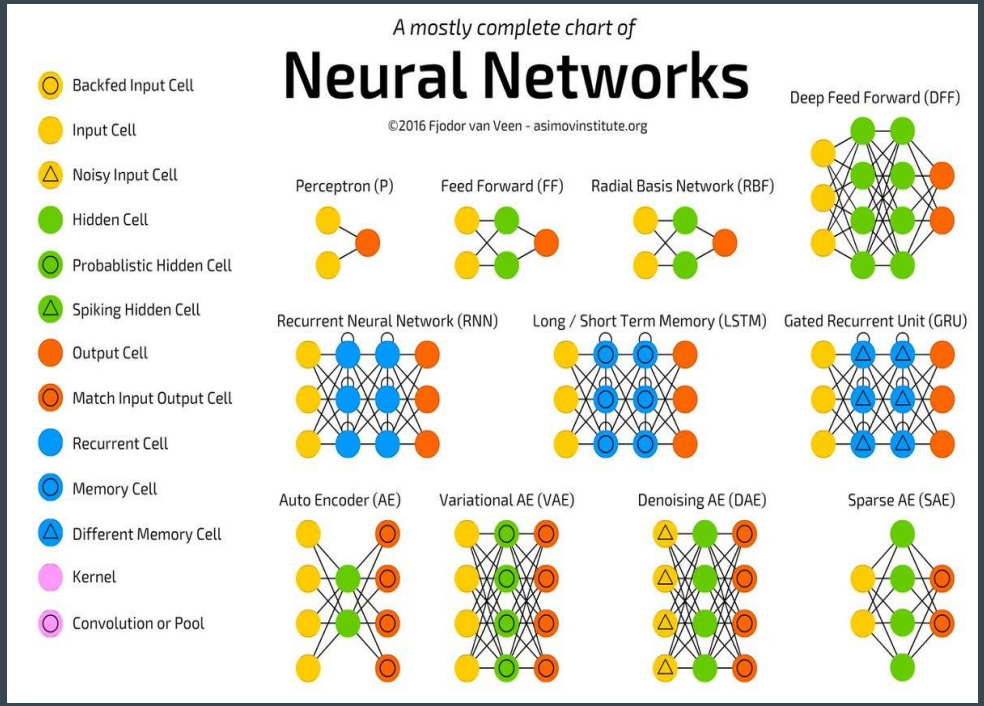


Neural Network architectures

Neural network architectures are basically different possible designs to connect the neuron to each other.

It includes answering questions like how many hidden layers you want or how many neurons you want in a layer

Every architecture is used for a different purpose.



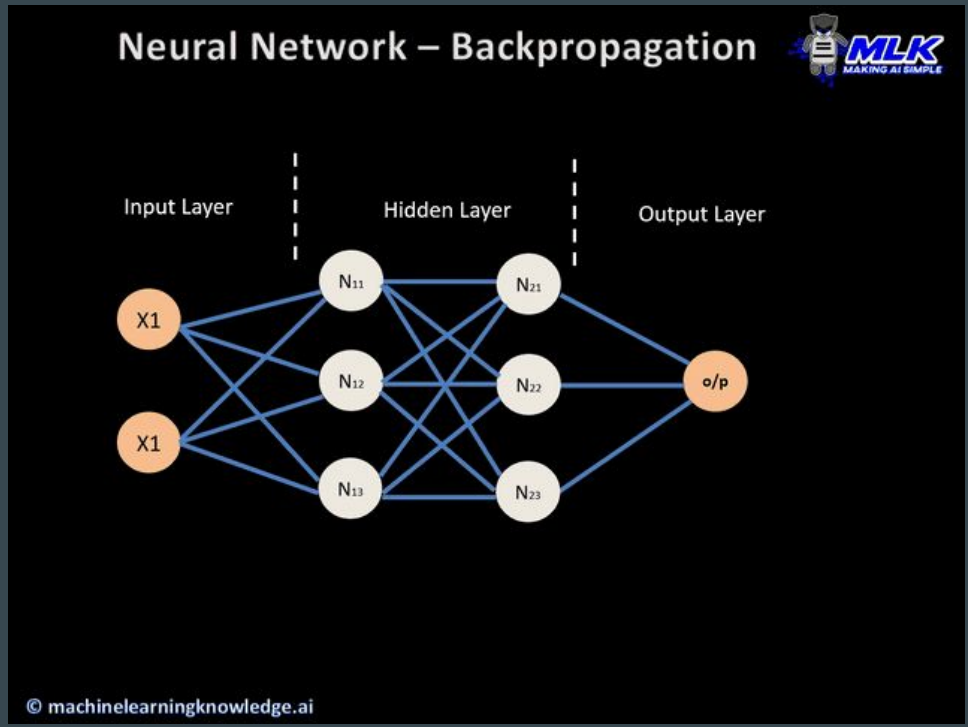
Backpropagation

Backpropagation in Neural Networks (intuition)

After the feed forward step the outputs are compared with the actual values and a loss is calculated. Then we try to figure out how changing each weight has an effect on the loss and we change the weights and biases such that the loss decreases.

The derivative of the loss wrt the weight is what gives us this value. Once this value is calculated for the weights and biases, they are updated so that the loss is lesser. This is what defined as learning, it is learning the weights and biases slowly and becoming better

These values are calculated layer by layer using the chain rule, and then the parameters are updated layer by layer. This is why its called Backpropagation.



Backprop in Neural Networks

Backprop in neural networks does the exact same thing as what we discussed in the case of simple regression problem.

The only difference is that, in case of a neural network, we have several layers and each layer has it own set of weights/parameters. Hence, while implementing backprop for neural networks, we use chain rule to represent the partial derivatives as the partial derivatives of more fundamental parameters.

We start by computing the error of the final output and then proceed backwards along the neural networks to compute error along every layer. We use this error along with the partial derivatives to compute the modifications required for each parameter. Hence this form of updation is called backpropagation or backward propagation of errors .

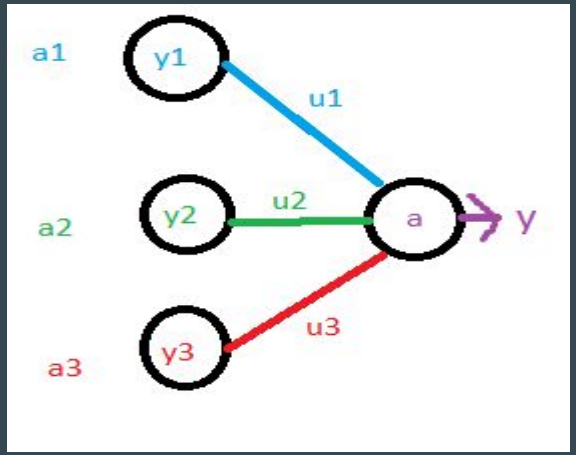
Math behind backpropagation

u - The weight in consideration

L - The loss function

y - the output value of the neuron

a - input given to the neuron in the next layer

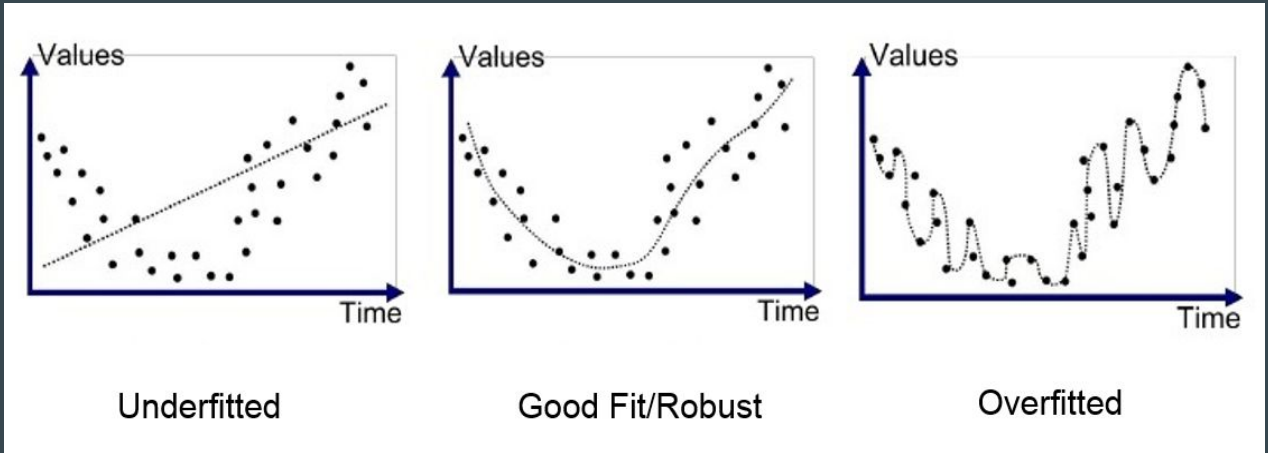


$$\mathbf{u} \leftarrow \mathbf{u} - \eta \nabla_{\mathbf{u}} L(\mathbf{u})$$

$$\frac{\partial L}{\partial u_{ij}} = \frac{\partial L}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial u_{ij}}$$

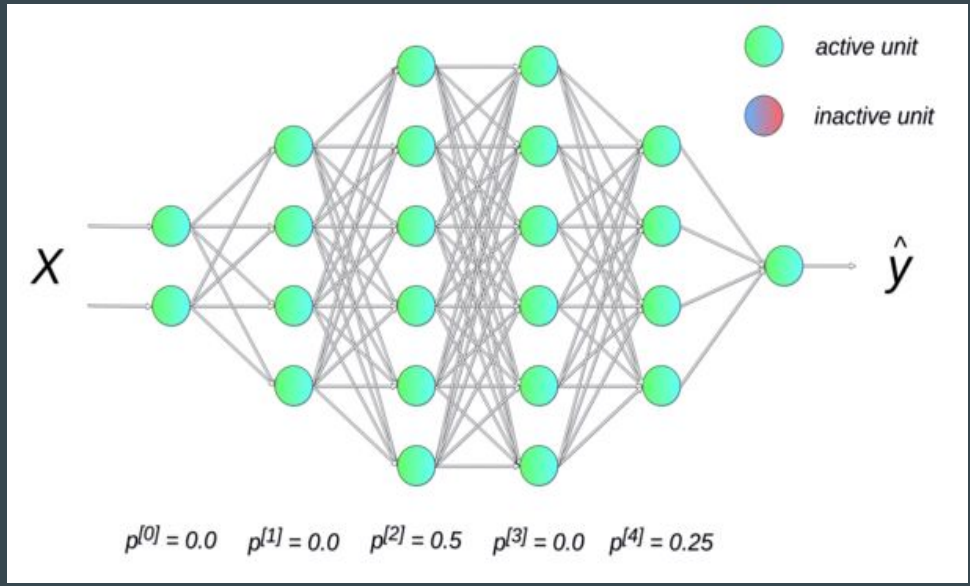
Regularization techniques

- Regularization are basically ways to ensure that model learns and recognizes trends instead of memorising the output. Lets now introduce some new terms
- Overfitting : it is a situation where the model has memorised all the the outcomes instead of trying to identify the general trend of the output with respect to the input features.
- Underfitting: In this case the model has not been able to generalise the outcomes and find relations with it to the input data.
- In both these cases, the problem is lack of generalisation which can be handled by different methods. In the case of overfitting, one of them is regularization



Dropout

- This is a regularization technique. What the dropout does it, when it has been added to a layer, a fraction of the neurons of that particular layer are deactivated during each forward propagation step
- This fraction is chosen by the user and can be adjusted so that the loss is minimised finally.
- To give an understanding think that, we are giving effectively “less data” so that the model will not draw the overfit line as you have seen in the previous slide



Weight Regularization

- In very simple words Weight Regularisation is just a means of preventing the layers from becoming very complex (to prevent overfitting)
- This is done by penalising the weights for being too large by using a regularisation parameter λ.
- The λ is multiplied with the square of the weights(β) or the summation of the absolute values in the cost function

$$Cost\ function = \sum\left(y_i - \sum \beta_k x_{ki}\right)^2 + \frac{1}{2}\lambda \sum {\beta_k}^2$$