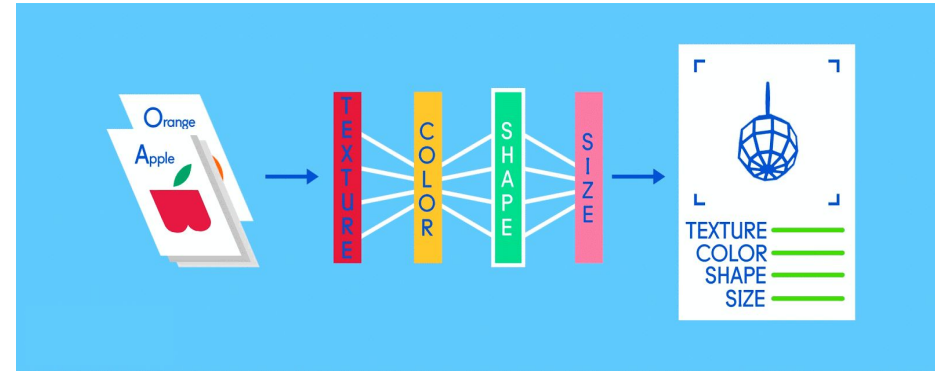
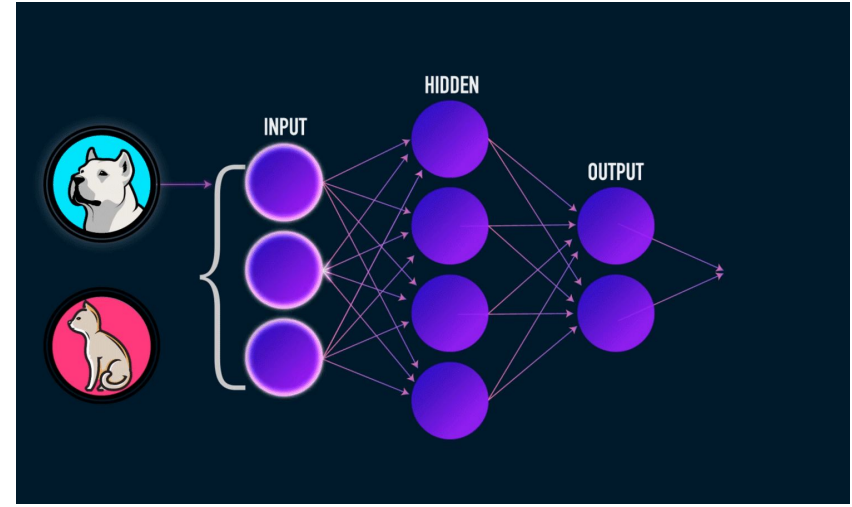


Understanding Neural Networks: Beginners guide



Scope

We would be: Focussing on understanding the concepts and would touch upon basic mathematical building blocks.

We will not be: Diving deep into calculus, vector mathematics.

Why: To make most out of the limited time we have. This content is sufficient for you to start solving real world business problems, but you should learn in depth to solve complex deep learning problem and develop specialisation.

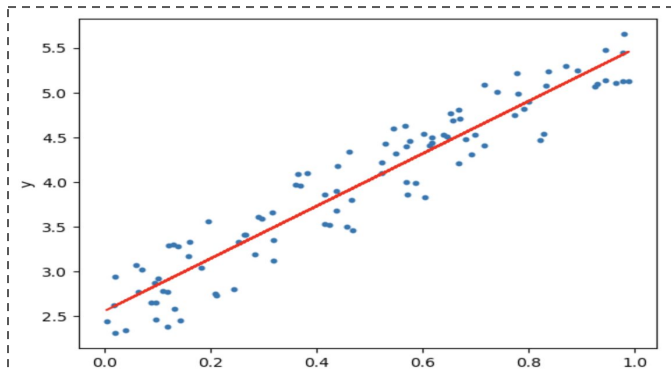
Topics covered:

- Neurons
- Activation functions
- How neural network learns/functions?
- Backpropagation
- Gradient descent & variants

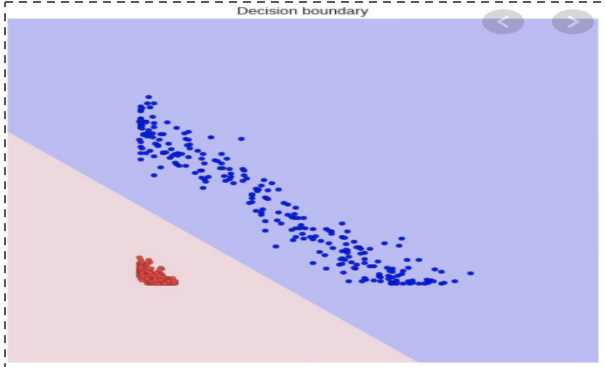
Why do we need Neural Networks?

Linear algorithms can predict linear trends (regression) and have linear decision boundaries (classification).

Linear regression line

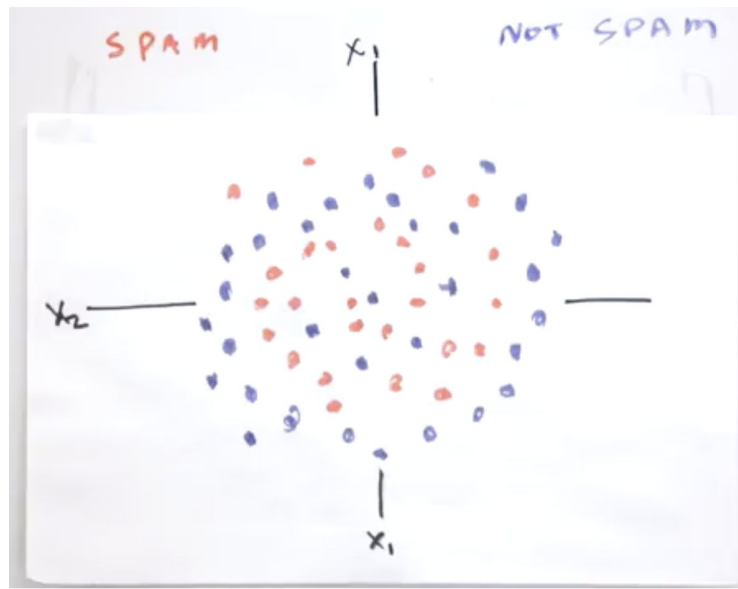


Decision boundary



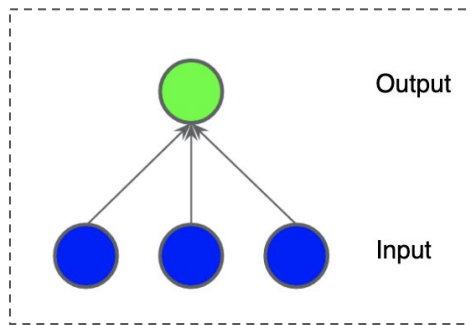
Linear decision boundary

But what can we do when our problem space is non-linear ?

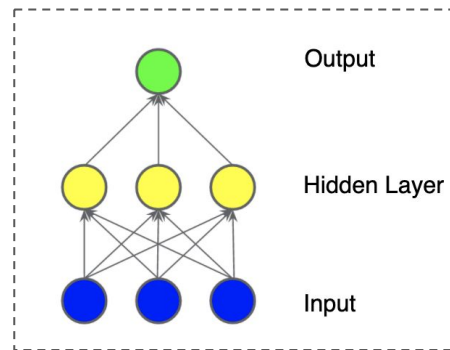
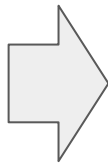


We need a non-linear decision boundary to classify these points.

How about? Stacking linear models on top of each other to model complex non-linear trends



Output = linear combination of inputs.
E.g. $y = x + y + z$



Output = linear combination of outputs from a linear model.

E.g. hidden layer = Yellow Dot (YD_1, YD_2, YD_3)

$YD_1 = a_1x + b_1y + c_1z$, $YD_2 = a_2x + b_2y + c_2z$, $YD_3 = a_3x + b_3y + c_3z$

Output = $a_4YD_1 + b_4YD_2 + c_4YD_3$

Where a_i, b_i, c_i are numerical constants. No matter how many times we multiply numerical constants we will still get a linear equation.

Linear combination of linear equation is linear equation.

Thus in order to model non-linear behavior, we need to somehow introduce non-linearity into our model. That is where Neural nets come to our rescue.

In layman terms: Neural Network = Stacked linear units + functions to introduce non-linearity

What makes Neural Nets so special?

Neural Nets represent complex human learning:

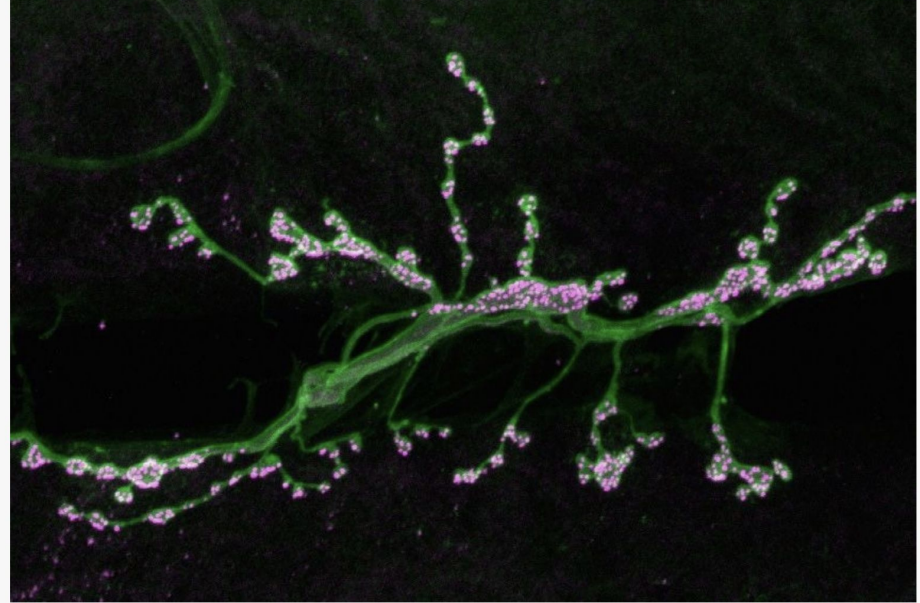
- Interconnected neurons in a neural network represent connections in human brain.
- When a human learn a new skill new connections are formed, connections relevant to performing skill/task becomes stronger (thicker), same happens in artificial neural network. Connections relevant to a problem increase in weight/importance.

Link to [article](#)

Neuroscientists reveal how the brain can enhance connections

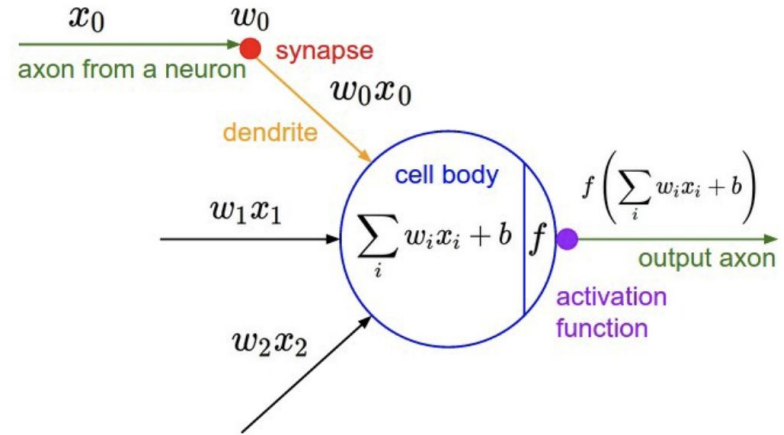
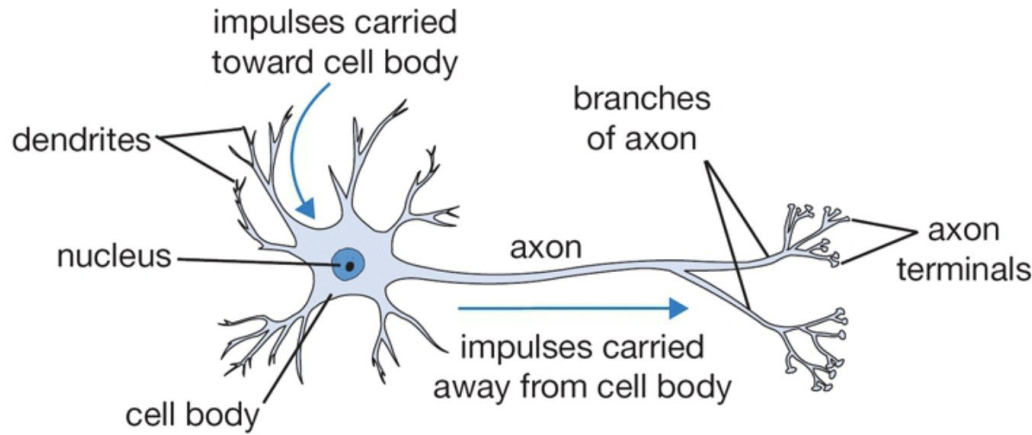
Newly identified mechanism allows the brain to strengthen links between neurons.

Anne Trafton | MIT News Office
November 18, 2015



When the brain forms memories or learns a new task, it encodes the new information by tuning connections between neurons. MIT neuroscientists have discovered a novel mechanism that contributes to the strengthening of these connections, also called synapses.

Neurons: Biological vs Artificial Neural Network



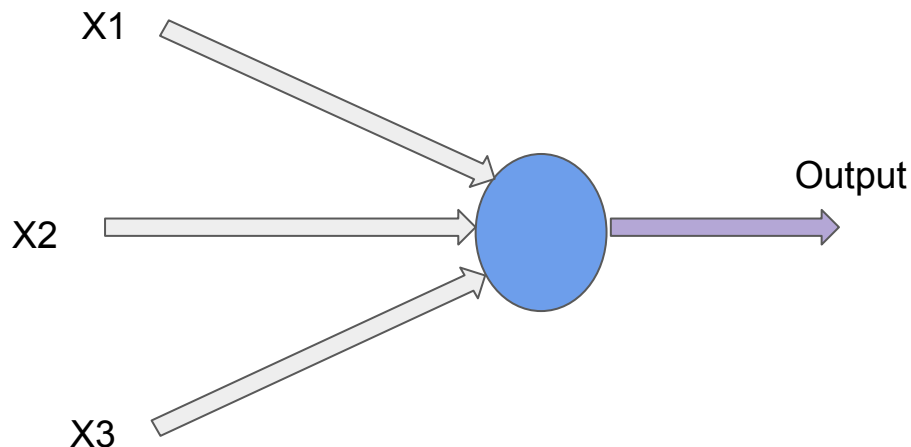
biological neuron (left) and a common mathematical model (right)

Neurons are the building block of any neural network

Understanding the most basic type of neuron: A Perceptron

The perceptron has three inputs, x_1, x_2, x_3 . Every input has corresponding weights, w_1, w_2, \dots , real numbers expressing the importance of the respective inputs to the output.

The neuron's output, 0 or 1, is determined by whether the weighted sum $\sum w_i x_i$ is less than or greater than some threshold value. Just like the weights, the threshold is a real number which is a parameter of the neuron.

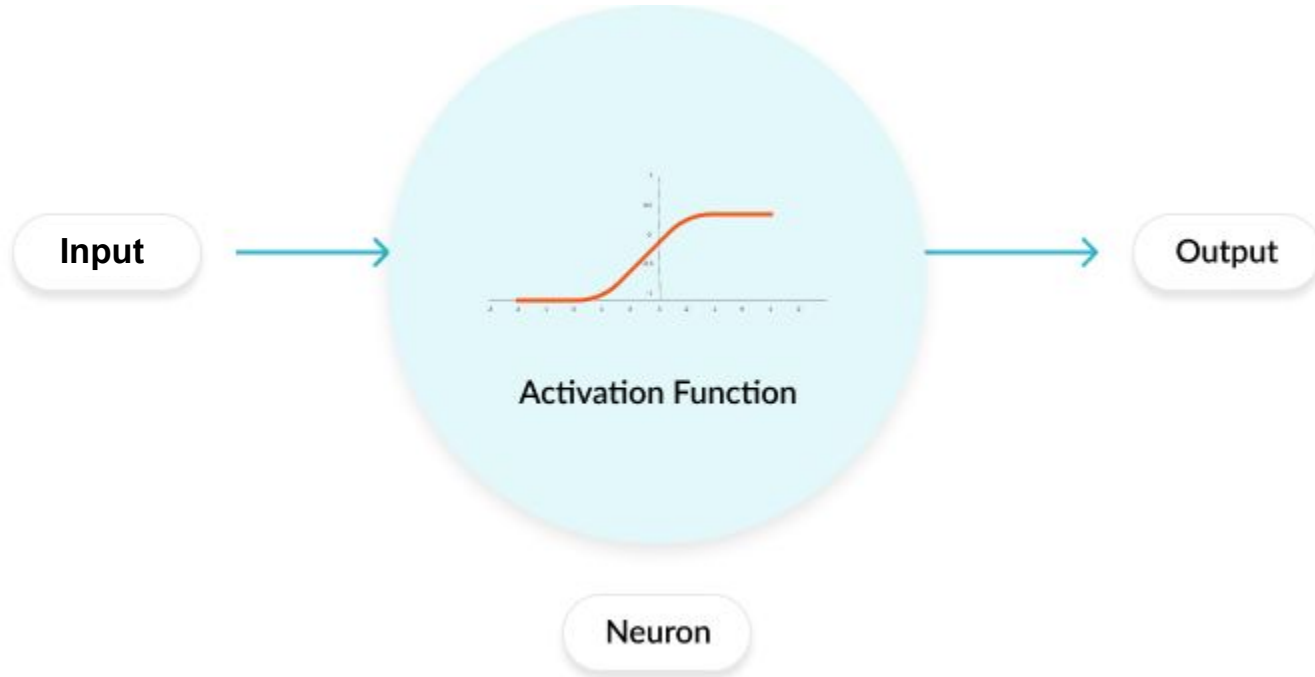


$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Output = 0, if $(x_1 * w_1) + (x_2 * w_2) + (x_3 * w_3) \leq \text{Threshold}$

Output = 1, if $(x_1 * w_1) + (x_2 * w_2) + (x_3 * w_3) > \text{Threshold}$

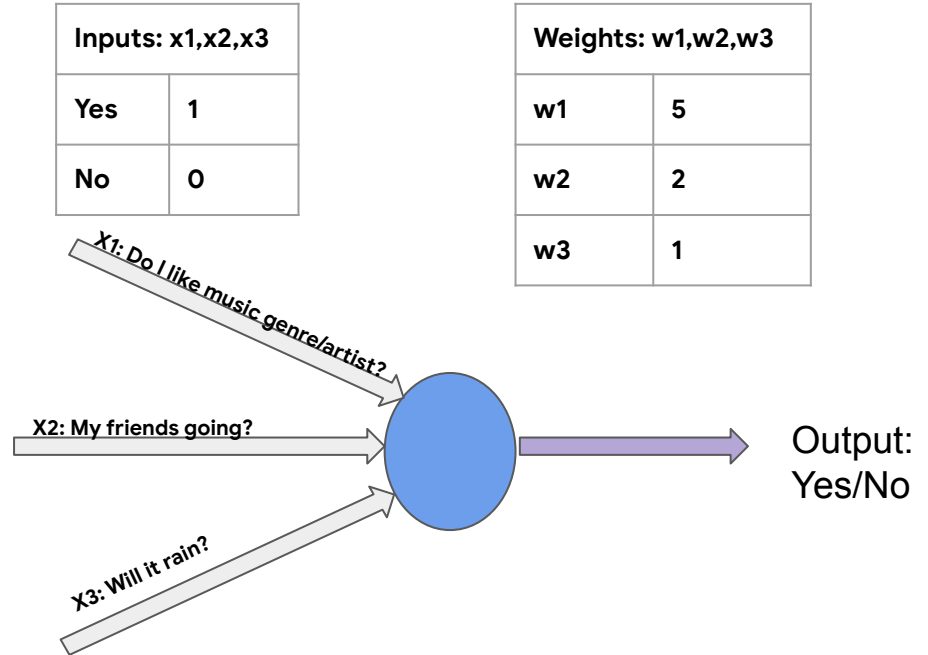
Equating 'Threshold' to 'Activation function'



Example problem: Should I go for a music-concert?

X1	X2	X3	Output Threshold: 5
1	1	1	Yes
1	0	1	Yes
0	1	1	No
1	1	0	Yes
0	0	1	No
0	1	0	No
1	0	0	Yes
0	0	0	No

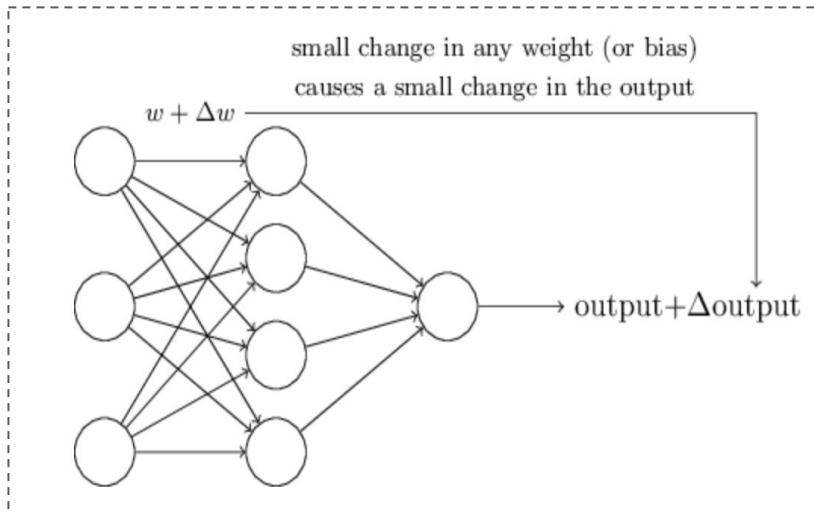
$$\text{Output} = \text{Threshold}(x_1w_1 + x_2w_2 + x_3w_3)$$



However, in real world, most of these variables are not binary in nature. X2 can be, how many of my friends are going? X3 can be probability of raining? Even human decision making is not binary, realistically it is about my likelihood of going (desire/mood/do I “feel” like).

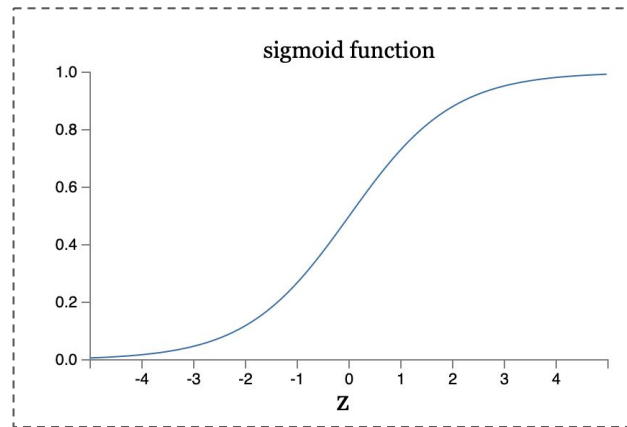
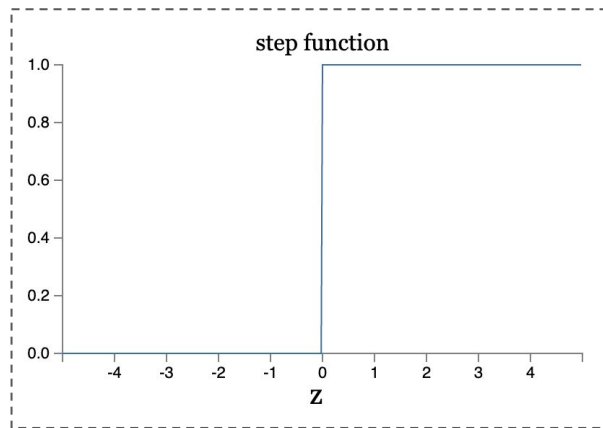
Thus our perceptron needs to adjust output with slight nudges in response to these slight changes in input data versus just having an output of 0 or 1.

Moving on to a sigmoid neuron

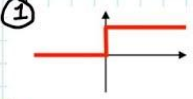
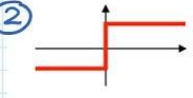
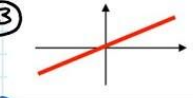
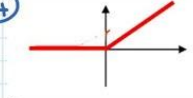
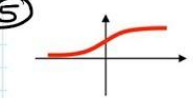
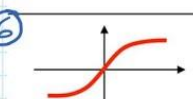


In order to have efficient learning, a small change in inputs should result in small change in output, i.e. slight adjustments in response to external stimuli.

However, with perceptrons a small change in input can flip a perceptron from 0 to 1 which in turn will change entire model (made up of multiple perceptron layer).



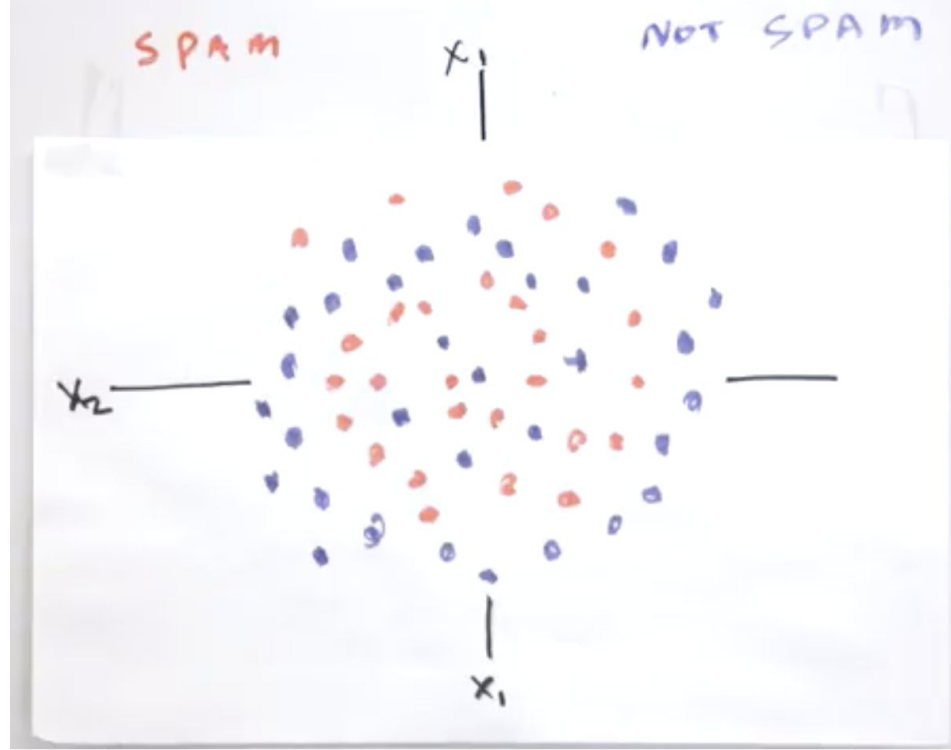
What other types of activation functions are available to us?

Commonly Used Activation Functions				Range
1. Step function :	$f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$	①		$\{0, 1\}$
2. Signum function :	$f(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	②		$\{-1, 1\}$
3. Linear function :	$f(z) = x$	③		$(-\infty, \infty)$
4. ReLU function :	$f(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$	④		$(0, \infty)$
5. Sigmoid function :	$f(z) = \frac{e^x}{1+e^x}$	⑤		$(0, 1)$
6. Hyperbolic tan :	$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	⑥		$(-1, 1)$

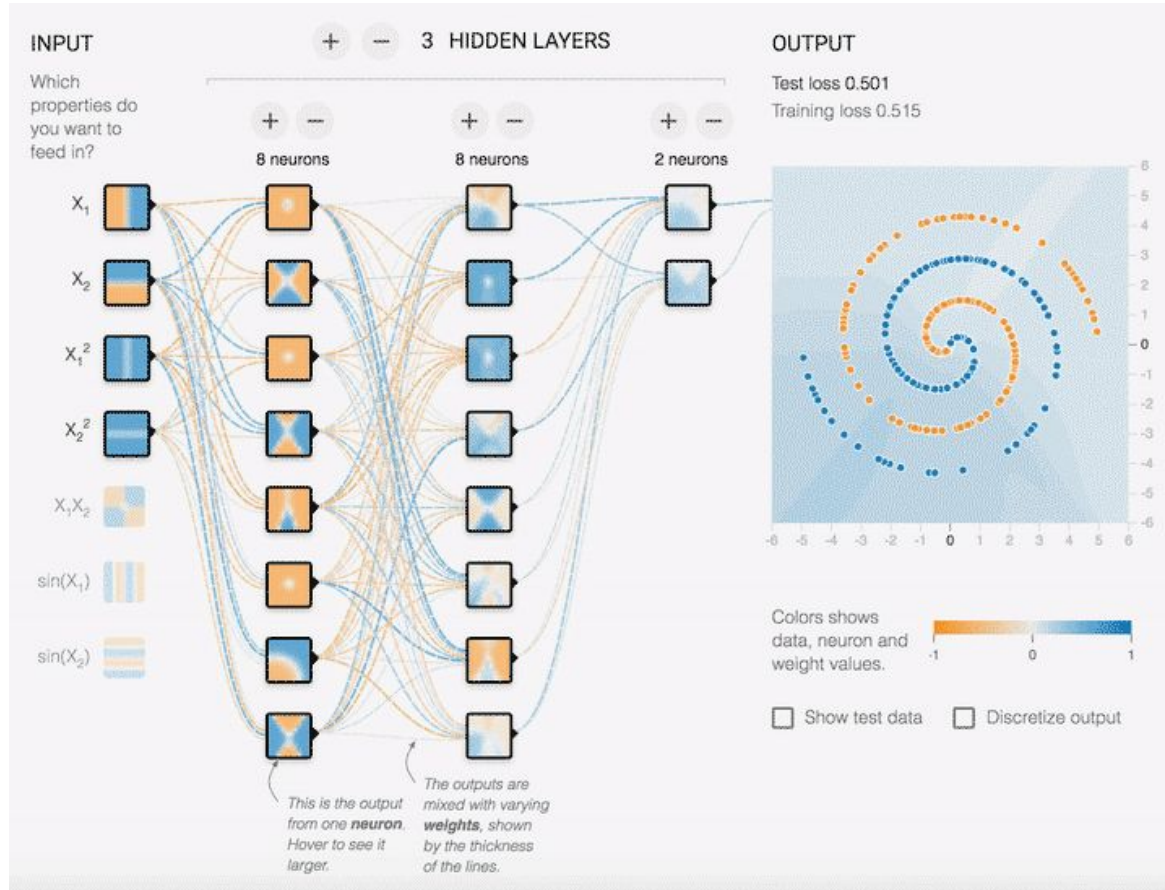
by Dr. Pankaj Kumar Porwal (BTech - IIT Mumbai, PhD - Cornell University) : Principal, Techno India NJR Institute of Technology, Udaipur

Choice of a activation function depends on the problem at hand, structure of classification boundary needed

So how a neural network would have solved our nonlinear problem?



Answer: Try it yourself [here](#)

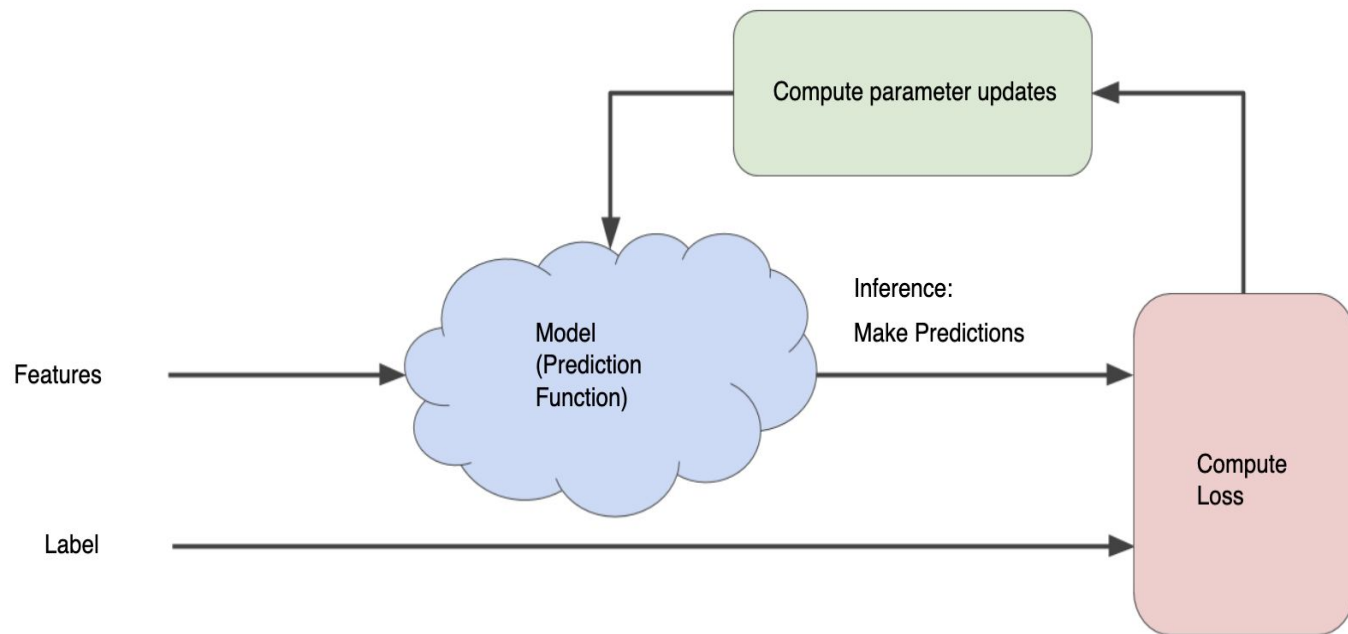


The output from one neuron feeds as an input to a neuron in the next layer

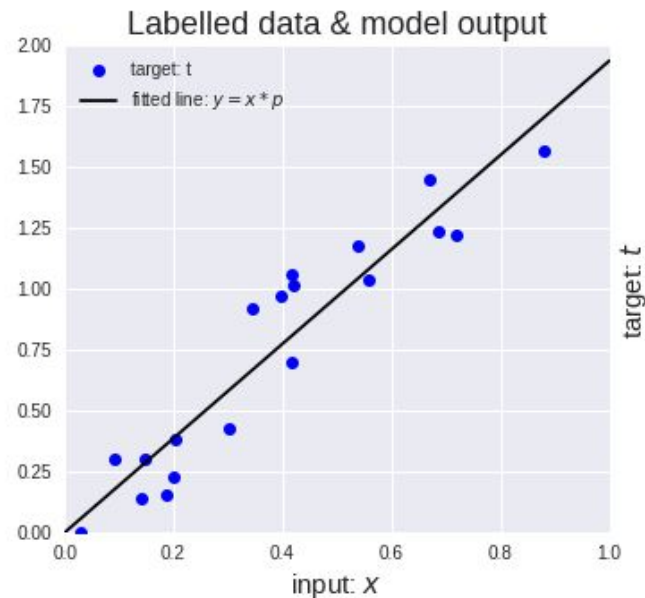
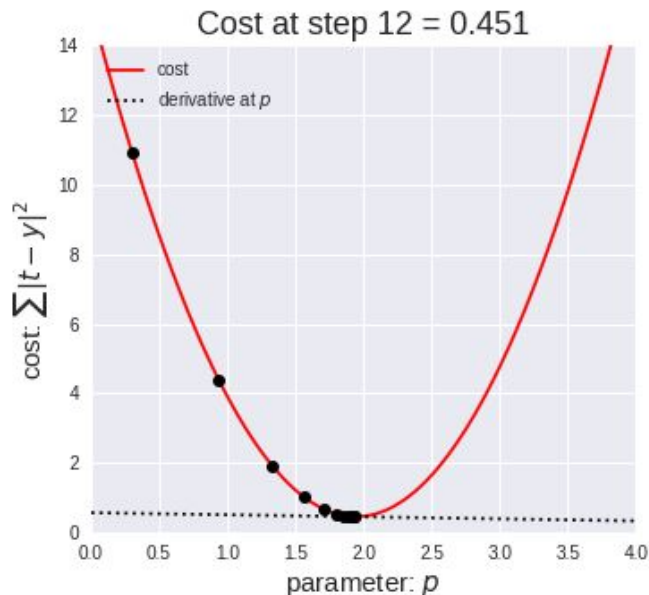
Notice the thickness of lines between neurons, those represent the weights.

That is a lot of neurons stacked on top of each other !!

So how is neural network actually learning?



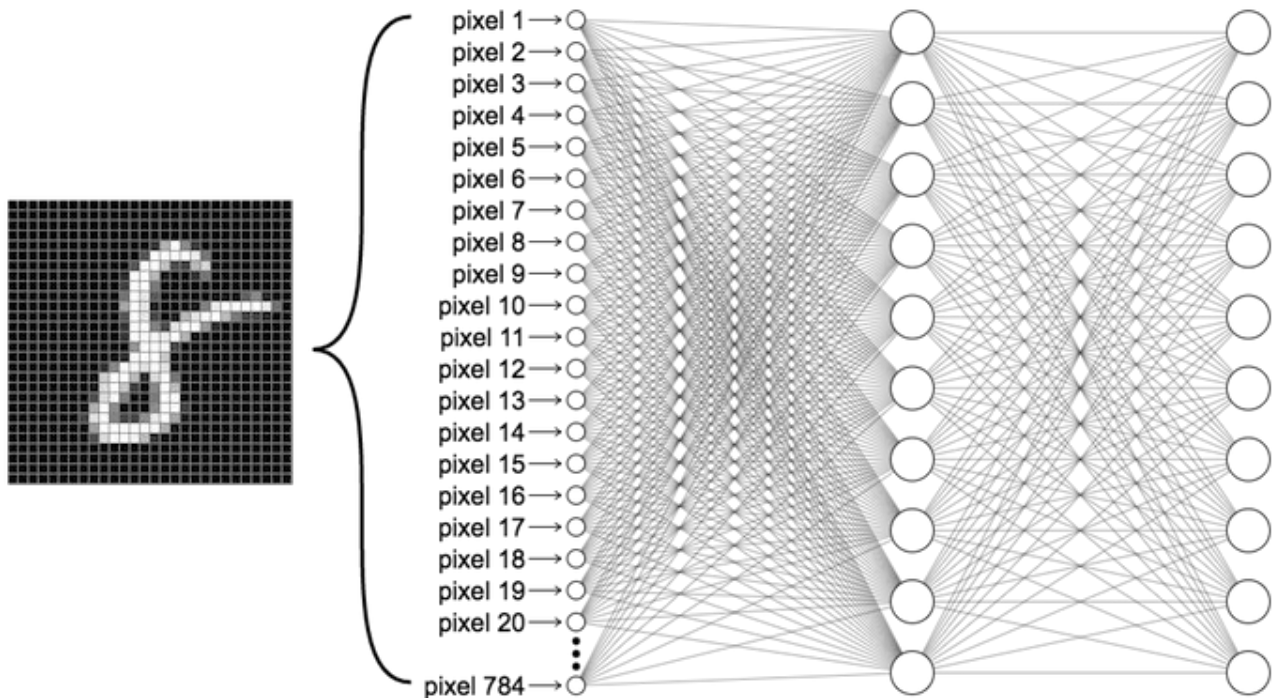
Example: Remember Linear regression model training, where we used to minimise value of loss functions?



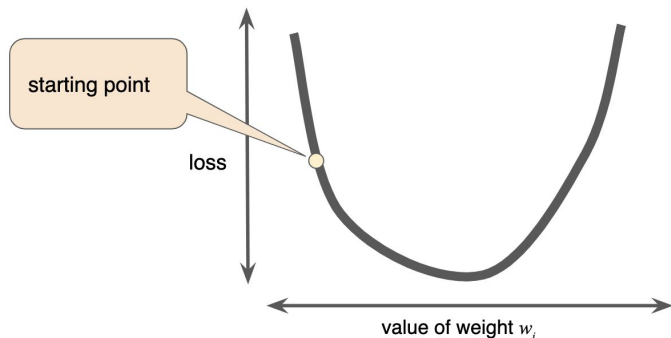
The loss function for this model is a convex/bowl/U shape with 1 minima

Need for an efficient learning algorithm

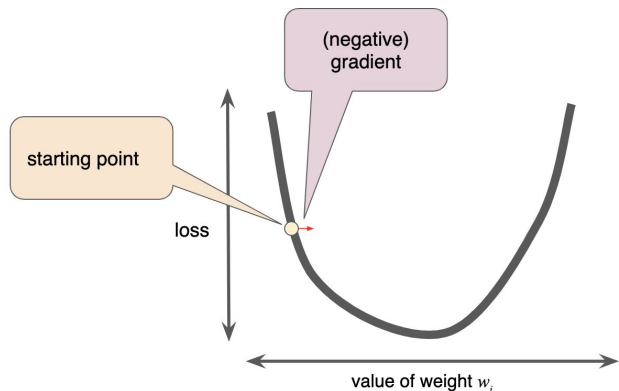
One way to reach minimum loss is to compute the loss curve for every possibility of input and weight combination for all neurons in neural network. But wait, it can be a infinitely large number of combinations for a complex prediction problem



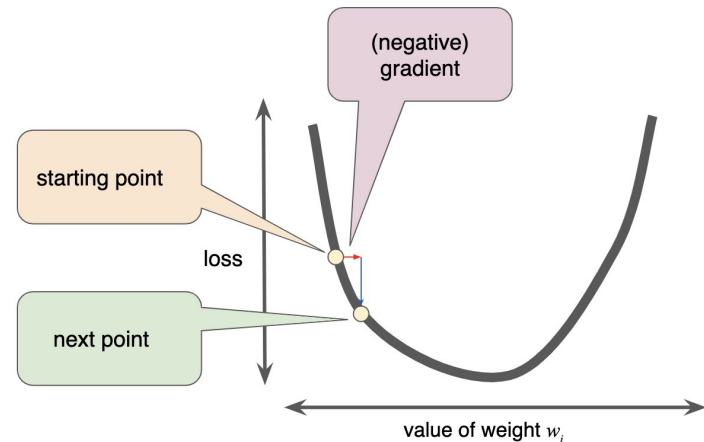
Introduction to gradient descent algorithm



Start: For a random value of weights, find the starting point on loss curve



Next: Calculate the gradient of the loss curve at the starting point. It is equal to the derivative (slope) of the curve. Thus figure out direction in which loss decreases.

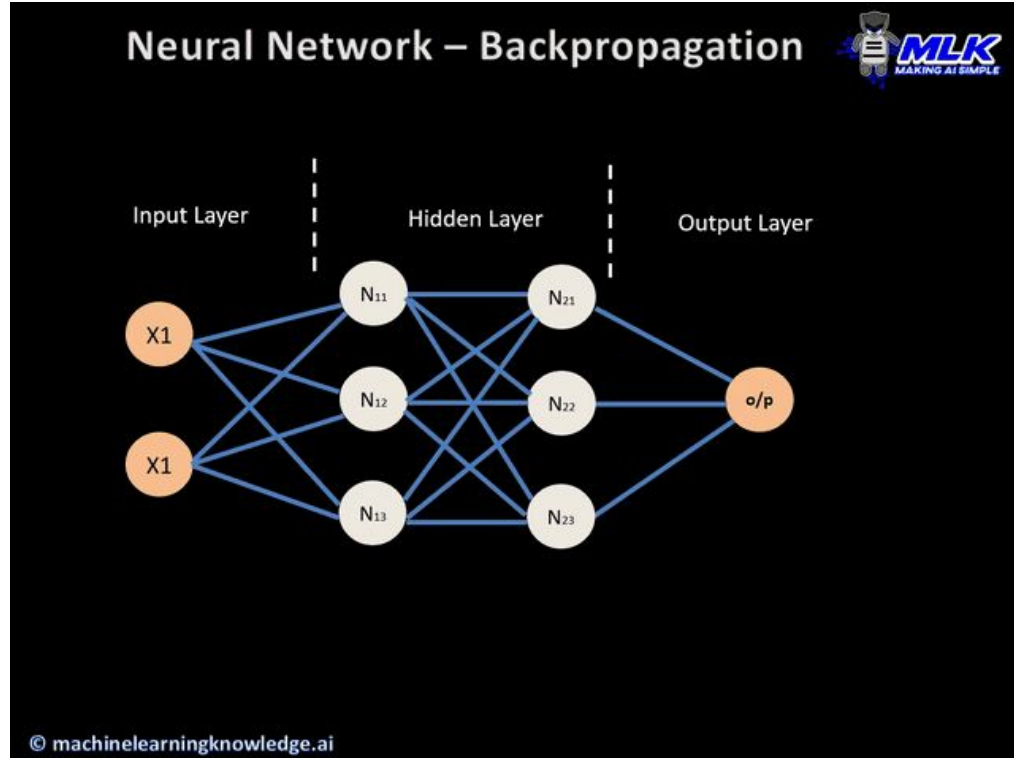


Next: To determine the next point along the loss function curve, the gradient descent algorithm adds some fraction of the gradient's magnitude to the starting point as shown in the following figure.

Repeat the process of calculating gradient and updating weights to get the next point till we reach minima.

Learning rate, the deciding factor for step size

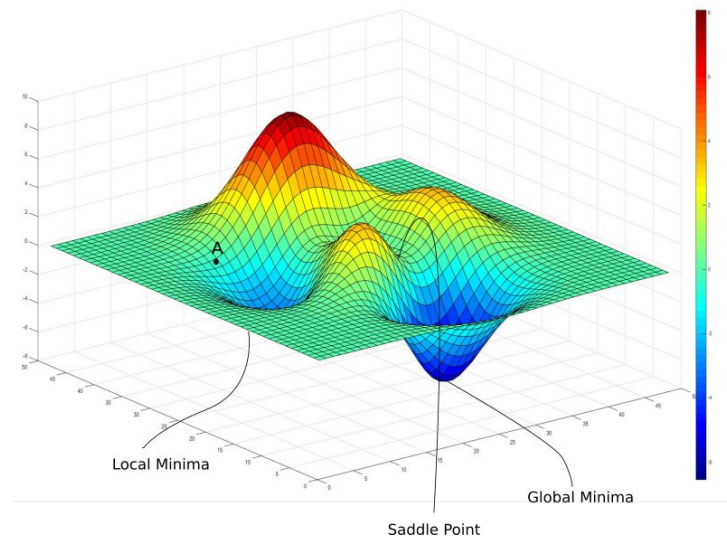
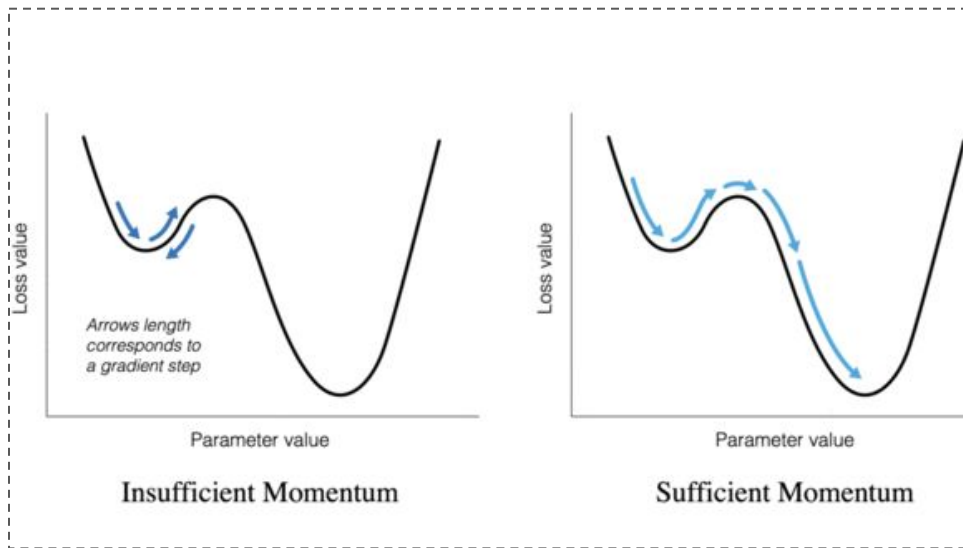
But how does a neural network know the effect of weight updates on loss function ?



Variants of gradient descent

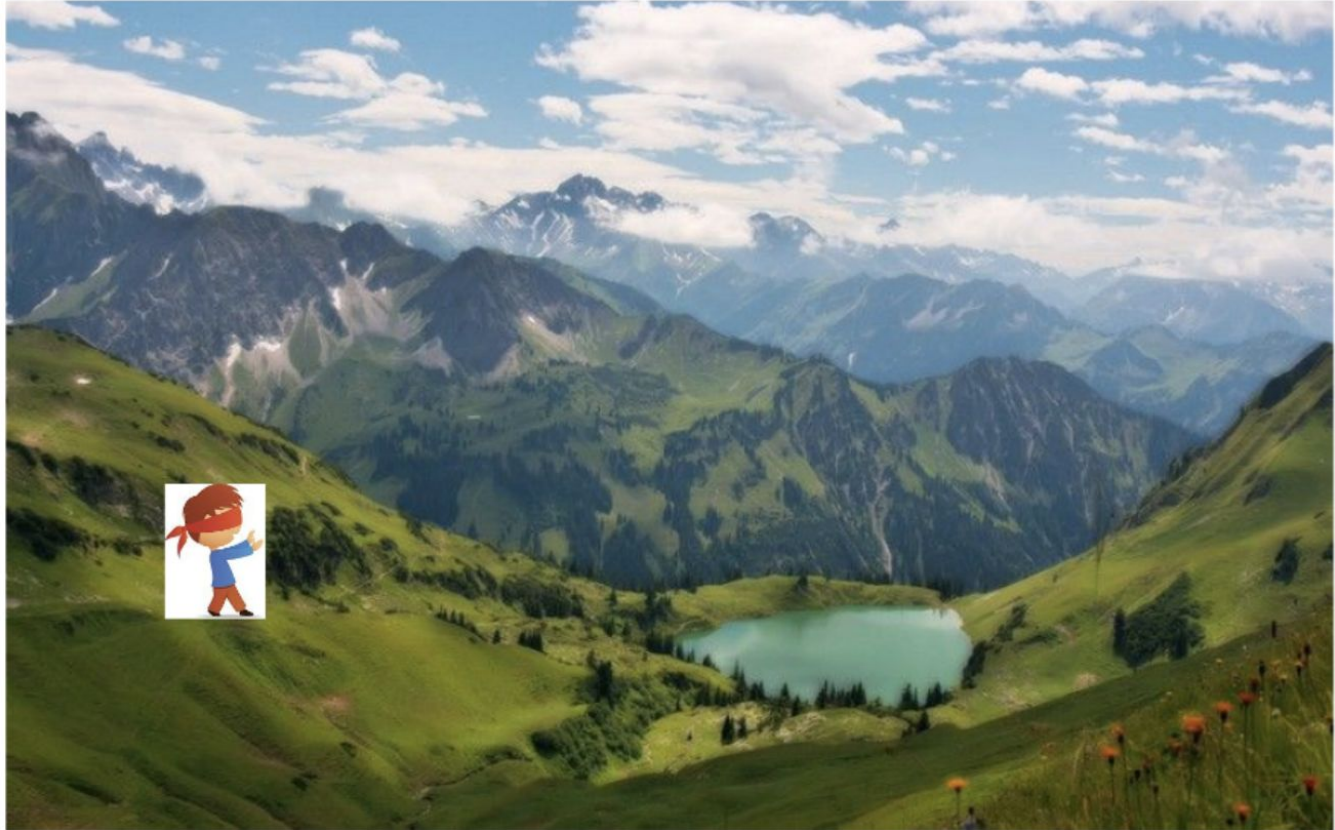
- Based on batch size: Stochastic vs Full batch
- Based on algorithm:
 - **Vanilla:** Means pure / without any adulteration. Its main feature is that we take small steps in the direction of the minima by taking gradient of the cost function.
 - **Gradient Descent with Momentum:** we introduce a new term called velocity, which considers the previous update and a constant which is called momentum.
 - **Adagrad:** Uses adaptive technique for learning rate updation. On the basis of how the gradient has been changing for all the previous iterations we try to change the learning rate.
 - **ADAM:** Builds on adagrad and further reduces its downside. In other words, you can consider this as momentum + ADAGRAD.

But why we need so many variants?



Again, it depends on the problem space/shape of cost function for which you are developing the solution.

My favorite example



Practice

Complete the tasks in MLCC [playground](#)

References:

- <https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/playground-exercises>
- https://docs.google.com/document/d/1DEyo_IWoxX89AkbdyStrvOkhOIH_ywGM9cKXwFO4cU0/edit
- <https://developers.google.com/machine-learning/glossary#neuron>
- <https://people.csail.mit.edu/dsontag/courses/ml16/slides/lecture24.pdf>
- https://drive.google.com/corp/drive/folders/1vWDj8cTDm_Q93v742f-Q9o716W8w7dSJ
- <https://github.com/vkosuri/CourseraMachineLearning/blob/master/home/week-4/lectures/pdf/Lecture8.pdf>
- <https://news.mit.edu/2015/brain-strengthen-connections-between-neurons-1118#:~:text=When%20the%20brain%20forms%20memories,the%20connections%2C%20also%20called%20synapses.>
- https://www.google.com/search?q=how+humans+learn+%2B+stronger+neural+connections&rlz=1C5CHFA_enIN852IN853&sxsrf=ALeKk00-0BpHK4-qFIORZCGQRxQg-Twn2Q:1599599128540&source=inms&tbm=isch&sa=X&ved=2ahUKEwjmt9uOu9rrAhUFVH0KHRyDD2kQ_AUoAXoECBkQAw&biw=1678&bih=895
- <https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc>
- <https://biodifferences.com/difference-between-dendrite-and-axon.html>
- <http://neuralnetworksanddeeplearning.com/chap1.html>
- <https://www.analyticsvidhya.com/blog/2017/03/introduction-to-gradient-descent-algorithm-along-its-variants/>