upGrad
#LifeKoKaroLift

Post-Graduate Diploma in ML/AI

12-07-2020

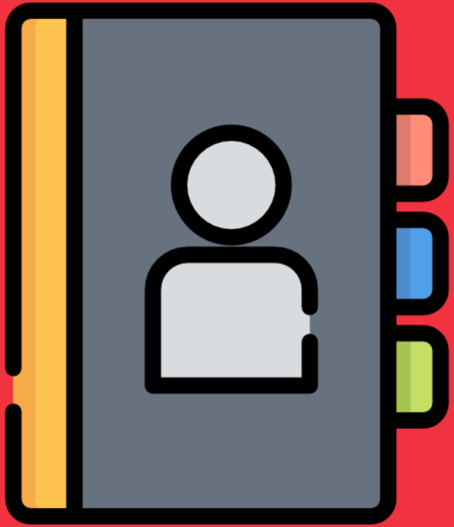
Course : Machine Learning

Lecture On : Neural Network - Intro

upGrad

Session - Agenda

- Introduction, Industry Use-cases
- Perceptron
- Feed Forward
- Backpropagation
- Assignment - Problem Statement
- Doubt Resolution



upGrad

In which of the following applications can we use deep learning to solve the problem?

A) Protein structure prediction
B) Prediction of chemical reactions
C) Detection of exotic particles
D) All of these

In which of the following applications can we use deep learning to solve the problem?

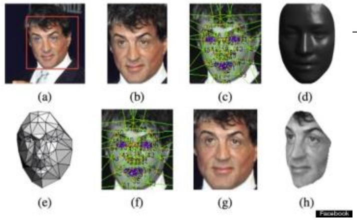
A) Protein structure prediction
B) Prediction of chemical reactions
C) Detection of exotic particles
D) All of these

We can use neural network to approximate any function so it can theoretically be used to solve any problem.

Neural Networks in Business

upGrad

Voice Assistants



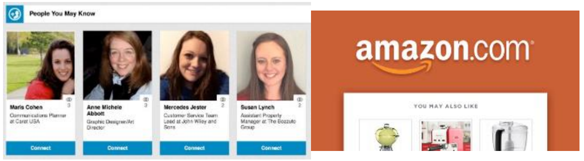
Facial Recognition

The Terminator

Self-driving cars

Google
- Search
- Translate
- maps

Recommendation Engines



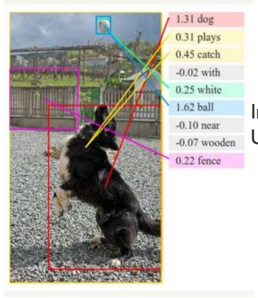
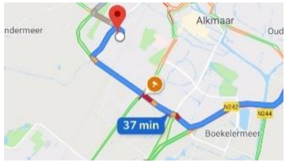


Image Understanding



Email filtering

Customer Support Queries (and Chatbots)

Video Surveillance

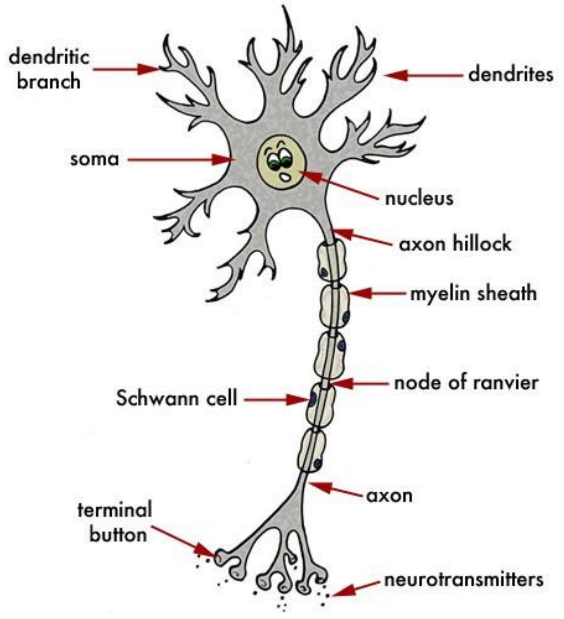
Catching Fraud in Banking

https://www.analyticsvidhya.com/blog/2018/05/24-ultimate-data-science-projects-to-boost-your-knowledge-and-skills/

Intuition behind the Perceptron

The human brain consists of neurons or nerve cells which transmit and process the information received from our senses. Many such nerve cells are arranged together in our brain to form a network of nerves. These nerves pass electrical impulses i.e the excitation from one neuron to the other.

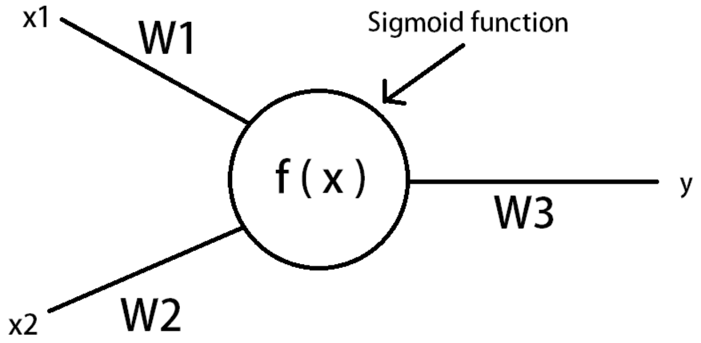
The dendrites receive the impulse from the terminal button. Dendrites carry the impulse to the nucleus Here , the electrical impulse is processed and then passed on to the axon



Construction of an Artificial Neuron

upGrad

In this case , the neurons are created artificially on a computer . Connecting many such artificial neurons creates an artificial neural network.



The data in the network flows through each neuron by a connection. Every connection has a specific weight by which the flow of data is regulated.

upGrad

Poll 1

Assume a simple MLP model with 3 neurons and inputs= 1,2,3. The weights to the input neurons are 4,5 and 6 respectively. Assume the activation function is a linear constant value of 3. What will be the output ?

A) 32
B) 643
C) 96
D) 48

upGrad

Poll 1

Assume a simple MLP model with 3 neurons and inputs= 1,2,3. The weights to the input neurons are 4,5 and 6 respectively. Assume the activation function is a linear constant value of 3. What will be the output ?

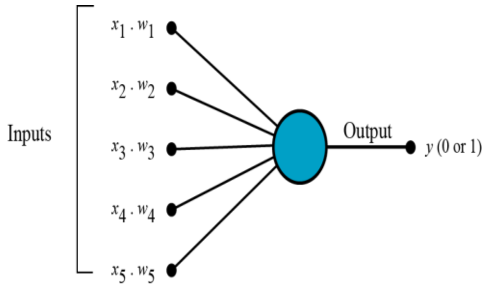
A) 32
B) 643
C) 96
D) 48

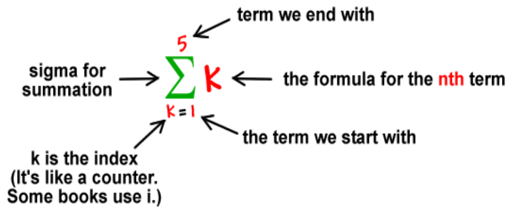
The output will be calculated as 3(1*4+2*5+6*3) = 96

But how does it work?

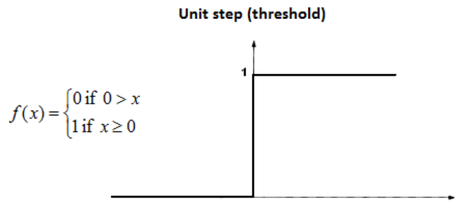
upGrad

The perceptron works on these simple steps





term we end with
sigma for summation
the formula for the nth term
the term we start with
k is the index (It's like a counter. Some books use i.)
$\sum_{k=1}^{5} k$



All the inputs ***x*** are multiplied with their weights ***w***. Let's call it ***k.***

Add all the multiplied values and call them Weighted Sum

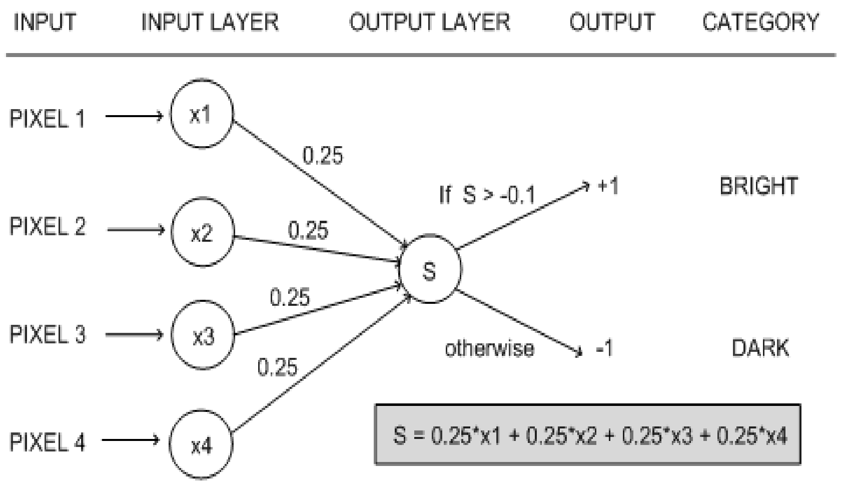
Apply that weighted sum to the correct Activation Function.

Why do we need Weights and Bias?

upGrad

Weights shows the strength of the particular node.

A bias value allows you to shift the activation function curve up or down.

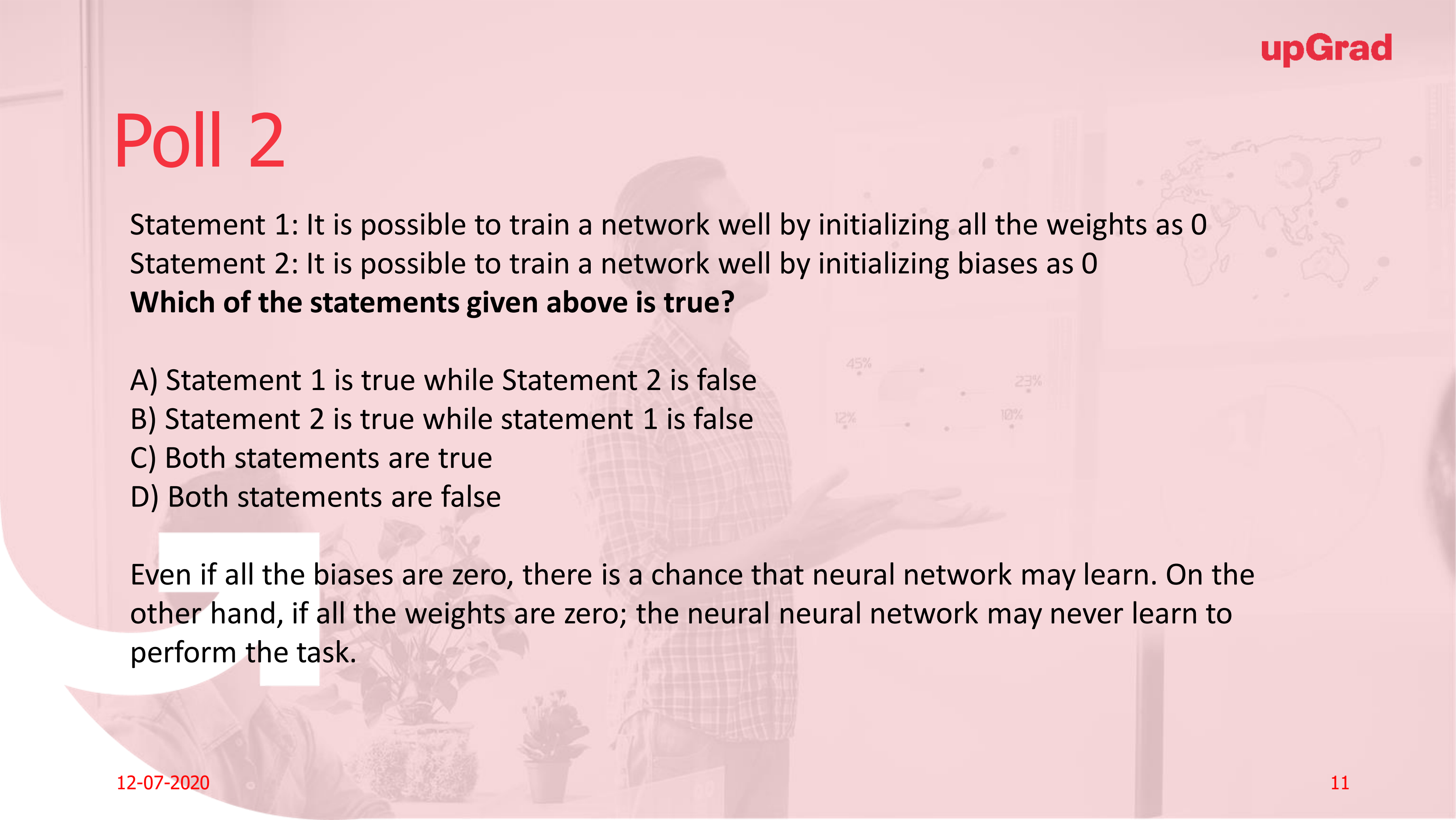


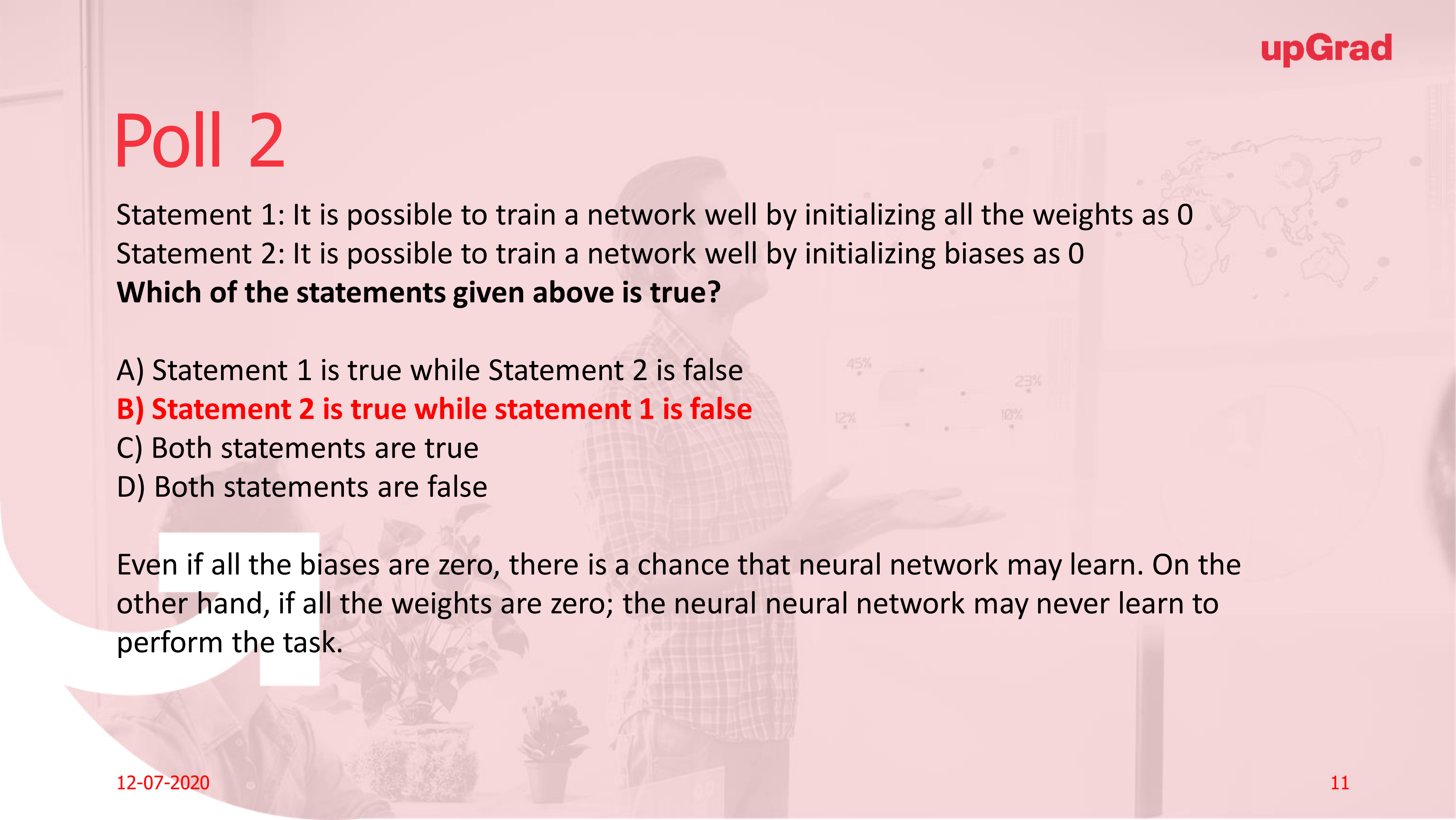
Poll 2

Statement 1: It is possible to train a network well by initializing all the weights as 0
Statement 2: It is possible to train a network well by initializing biases as 0
Which of the statements given above is true?

A) Statement 1 is true while Statement 2 is false
B) Statement 2 is true while statement 1 is false
C) Both statements are true
D) Both statements are false

Even if all the biases are zero, there is a chance that neural network may learn. On the other hand, if all the weights are zero; the neural neural network may never learn to perform the task.





upGrad

Poll 2

Statement 1: It is possible to train a network well by initializing all the weights as 0
Statement 2: It is possible to train a network well by initializing biases as 0
Which of the statements given above is true?

A) Statement 1 is true while Statement 2 is false
B) Statement 2 is true while statement 1 is false
C) Both statements are true
D) Both statements are false

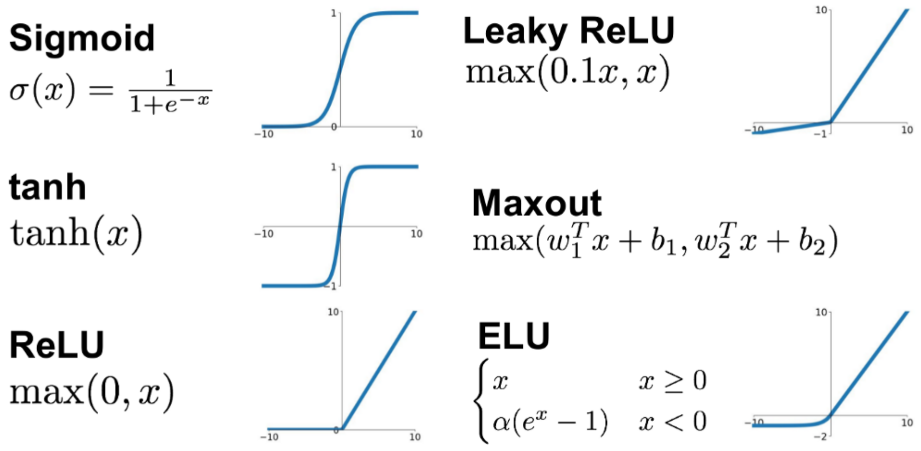
Even if all the biases are zero, there is a chance that neural network may learn. On the other hand, if all the weights are zero; the neural neural network may never learn to perform the task.

Why we use Activation functions

upGrad

***Activation** functions* are really important for a Artificial Neural Network to learn and make sense Non-linear complex functional mappings between the inputs and response variable.

They *introduce non-linear properties to our Network. Their **main purpose is to convert a input signal of a node in a A-NN to an output signal.*** That output signal now is used as a input in the next layer in the stack.

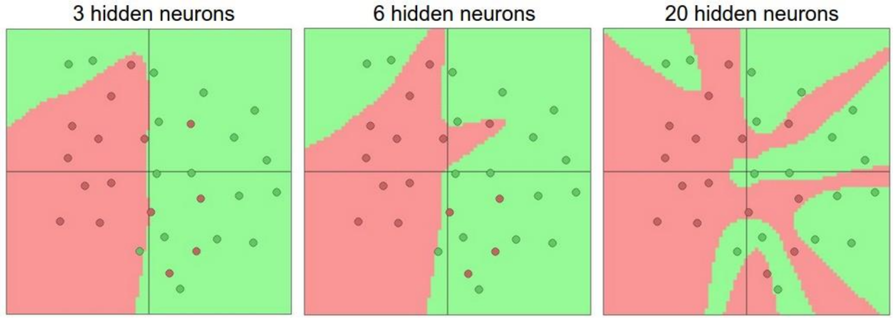


https://ai.stackexchange.com/questions/5493/what-is-the-purpose-of-an-activation-function-in-neural-networks

why can't we do it without activating the input signal

upGrad

A linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data. A Neural Network without Activation function would simply be a **Linear regression Model**, which has limited power and does not performs good most of the times.



That is why we use Artificial Neural network techniques such as ***Deep learning to make sense of something complicated ,high dimensional, non-linear -big datasets, where the model has lots and lots of hidden layers in between and has a very complicated architecture which helps us to make sense and extract knowledge form such complicated big datasets.***

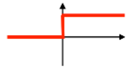
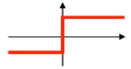
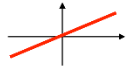
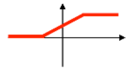
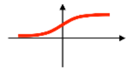
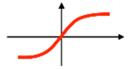
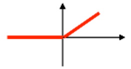
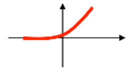
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

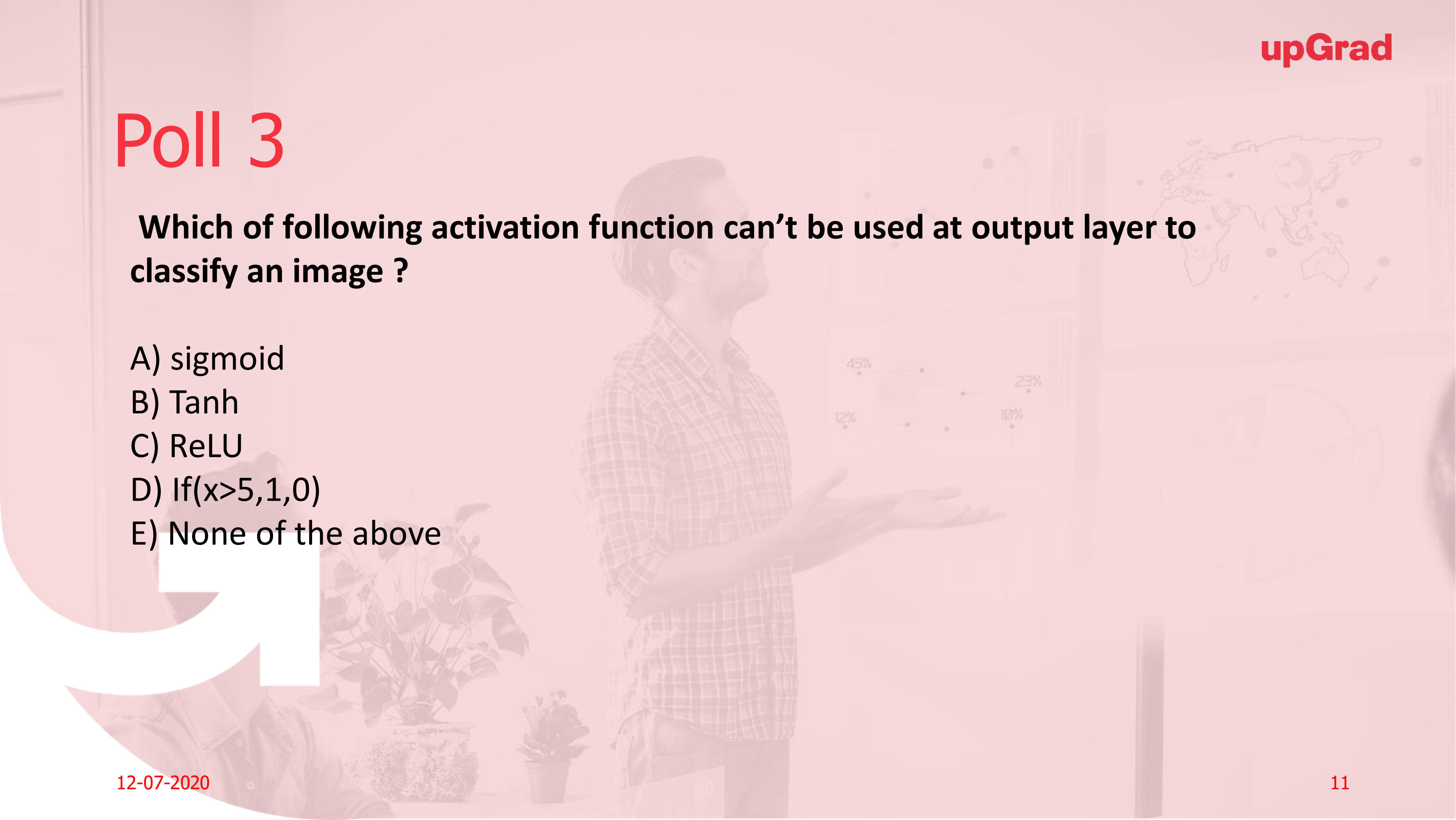
which one is better to use ?

upGrad

Nowadays we should use **ReLu** which should only be applied to the hidden layers. And if your model suffers form dead neurons during training we should use **leaky ReLu** or **Maxout** function.

It's just that *Sigmoid and Tanh* should not be used nowadays due to the **vanishing Gradient Problem** which causes a lots of problems to train,degrades the accuracy and performance of a **deep Neural Network Model.**

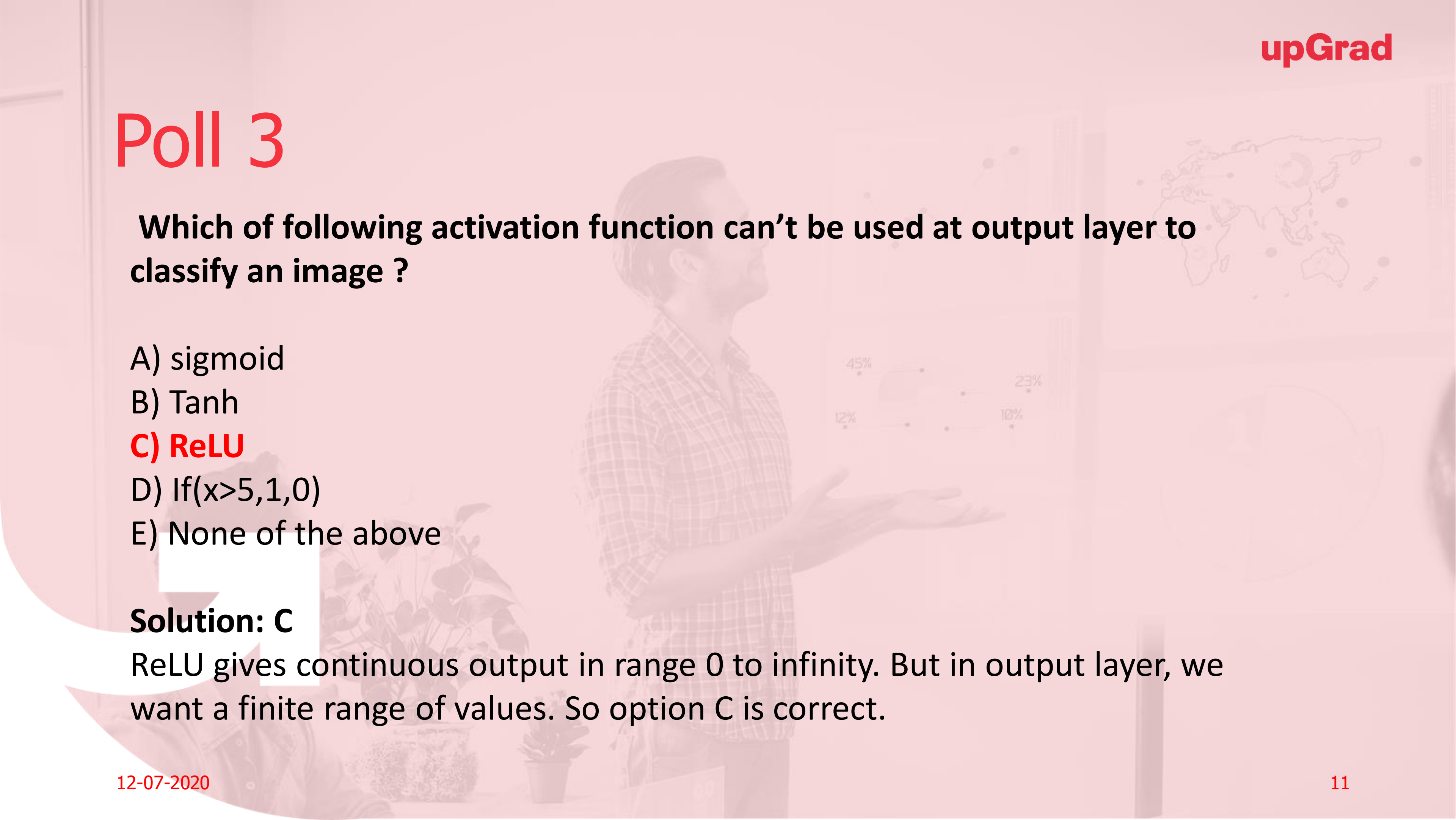
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	



Poll 3

Which of following activation function can't be used at output layer to classify an image ?

A) sigmoid
B) Tanh
C) ReLU
D) If(x>5,1,0)
E) None of the above



upGrad

Poll 3

Which of following activation function can’t be used at output layer to classify an image ?

A) sigmoid
B) Tanh
C) ReLU
D) If(x>5,1,0)
E) None of the above

Solution: C
ReLU gives continuous output in range 0 to infinity. But in output layer, we want a finite range of values. So option C is correct.

Multiple Neurons

- The average human brain has 100 billion neurons
- Allowing the brain to make extremely complicated decisions
- Each neuron is connected to 10,000 neurons
- Which together create a complicated network of about 1000 trillion connections

Let's Add a Bit of Complexity Now

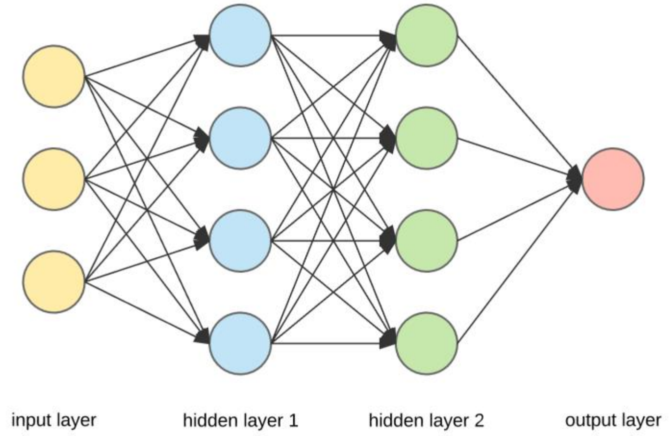
upGrad

3 types of layers in neural network:

•**Input layer** — It is used to pass in our input(an image, text or any suitable type of data for NN).

•**Hidden Layer** — These are the layers in between the input and output layers. These layers are responsible for learning the mapping between input and output. (i.e. in the dog and cat gif above, the hidden layers are the ones responsible to learn that the dog picture is linked to the name dog, and it does this through a series of matrix multiplications and mathematical transformations to learn these mappings).

•**Output Layer** — This layer is responsible for giving us the output of the NN given our inputs.

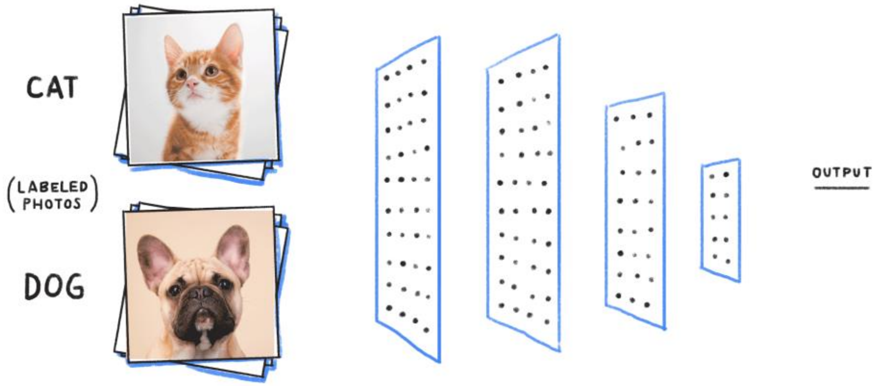


Feedforward

upGrad

the output from one layer is used as input to the next layer. Such networks are called **feedforward neural networks**.

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,k} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ w_{j,0} & w_{j,1} & \cdots & w_{j,k} \end{bmatrix} \begin{bmatrix} a_0^0 \\ a_1^0 \\ \vdots \\ a_n^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$



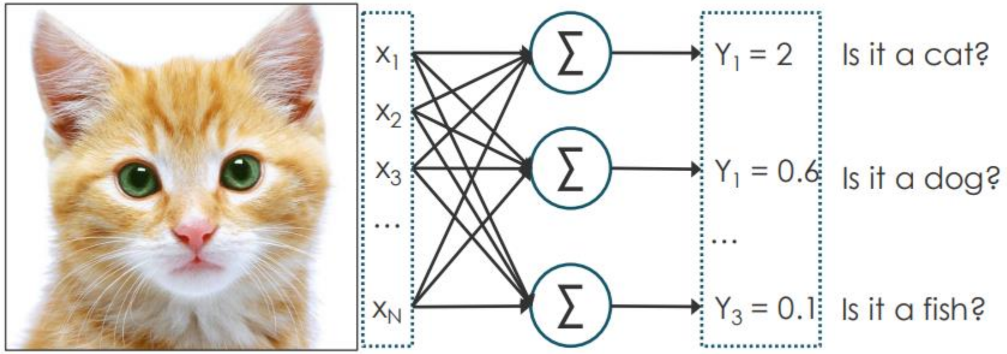
Example

upGrad

To create a classifier we want the output to look like as set of probabilities

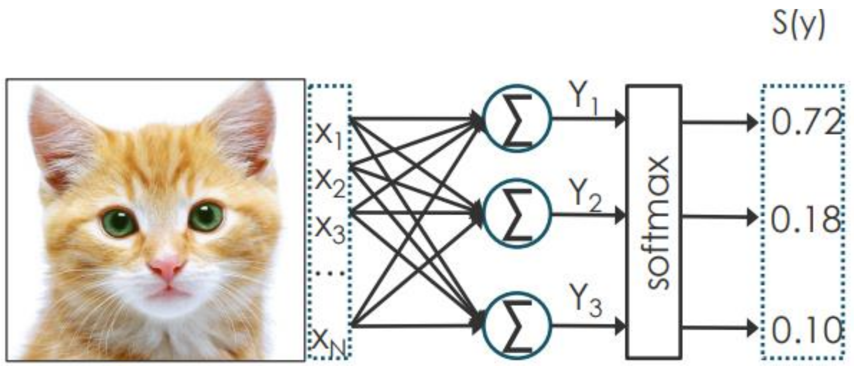
By feeding the input through the network we get a set of scores **Y** called **"logits"** but in here they cannot be used as probabilities

- They are not within the range [0,1]
- They do not add up to 1.

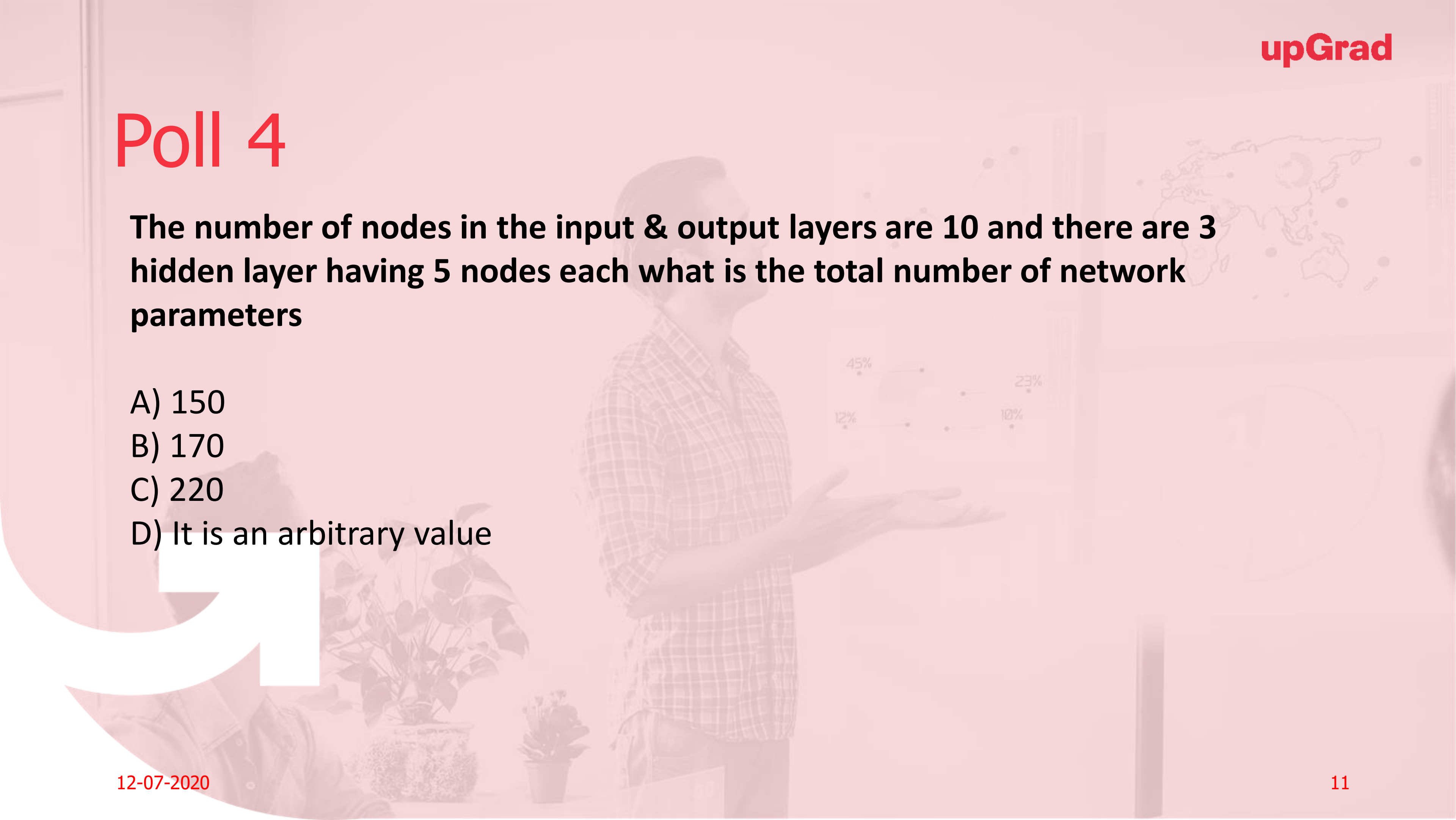


Example

To convert logits into probabilities we can use the softmax function $S(y_i) = \frac{e^{y_i}}{\sum e^{y}}$



This guarantees all values are between [0,z] and they add up to 1.



Poll 4

The number of nodes in the input & output layers are 10 and there are 3 hidden layer having 5 nodes each what is the total number of network parameters

A) 150
B) 170
C) 220
D) It is an arbitrary value

upGrad

Poll 4

The number of nodes in the input & output layers are 10 and there are 3 hidden layer having 5 nodes each what is the total number of network parameters

A) 150
B) 170
C) 220
D) It is an arbitrary value

Solution: B
(50 + 25+25+50) + 20.

Cost Function

upGrad

A **Cost Function/Loss Function** evaluates the performance of our Machine Learning Algorithm. The **Loss function** computes the error for a single training example while the **Cost function** is the average of the loss functions for all the training examples.

$$Cost = \frac{1}{N}\sum_{i=1}^{N}(Y' - Y)^2$$

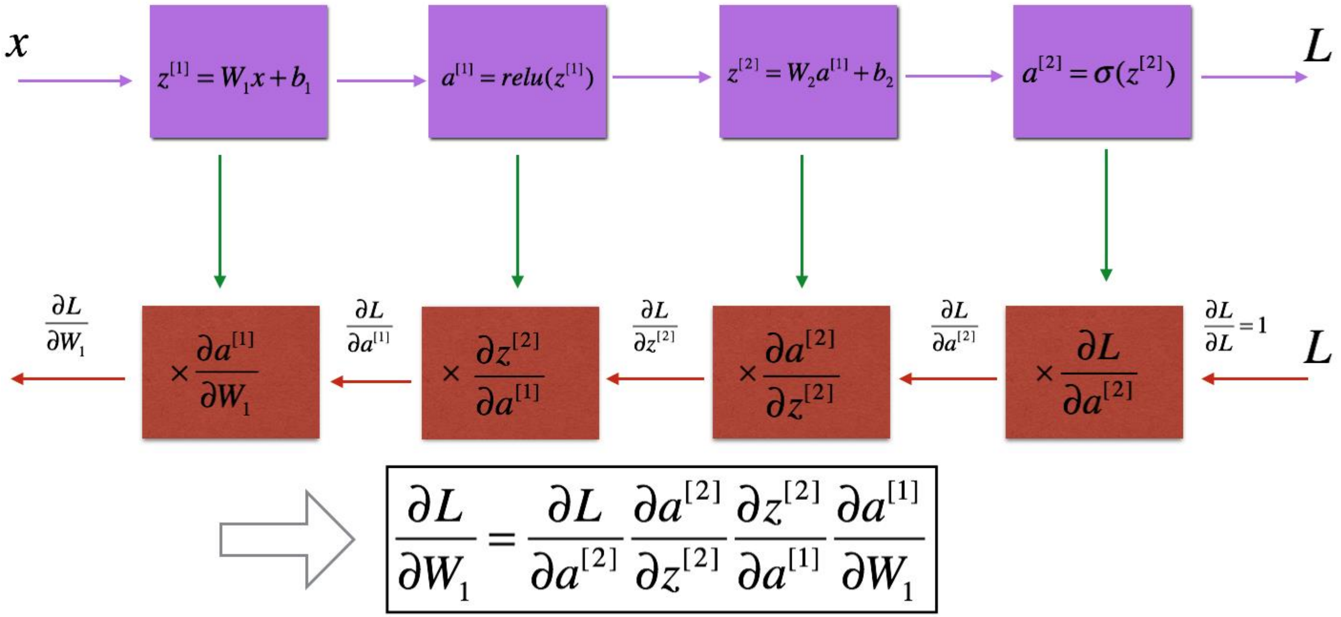
The goal of any Learning Algorithm is to minimize the Cost Function.

lower error between the actual and the predicted values signifies that the algorithm has done a good job in learning.

A common measure of the discrepancy between the two values is the ”Cross-entropy”

$$D(S(y), L) = -\sum_i L_i \log(S(y_i))$$

Backward Propagation

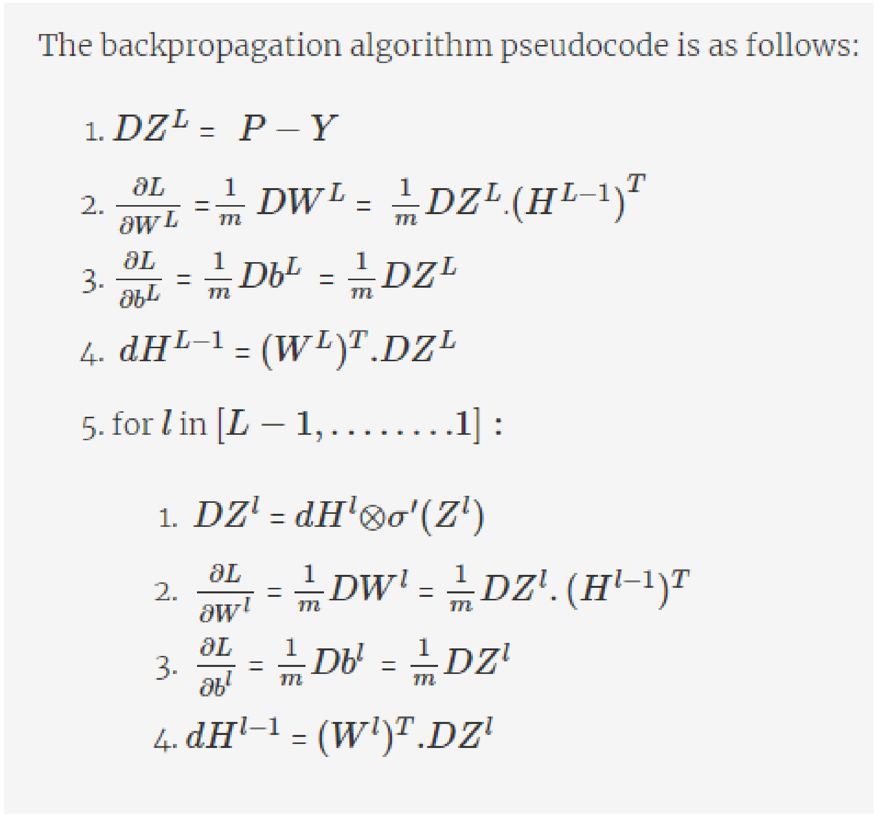


x
$z^{[1]} = W_1 x + b_1$
$a^{[1]} = relu(z^{[1]})$
$z^{[2]} = W_2 a^{[1]} + b_2$
$a^{[2]} = \sigma(z^{[2]})$
L
$\frac{\partial L}{\partial W_1}$
$\times \frac{\partial a^{[1]}}{\partial W_1}$
$\frac{\partial L}{\partial a^{[1]}}$
$\times \frac{\partial z^{[2]}}{\partial a^{[1]}}$
$\frac{\partial L}{\partial z^{[2]}}$
$\times \frac{\partial a^{[2]}}{\partial z^{[2]}}$
$\frac{\partial L}{\partial a^{[2]}}$
$\times \frac{\partial L}{\partial a^{[2]}}$
$\frac{\partial L}{\partial L} = 1$
L

$$\boxed{\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \frac{\partial z^{[2]}}{\partial a^{[1]}} \frac{\partial a^{[1]}}{\partial W_1}}$$

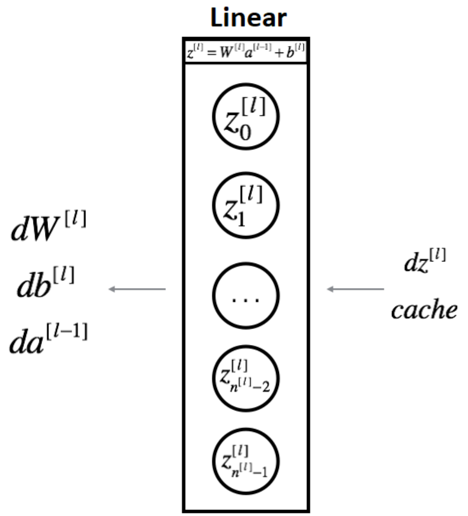
We know that propagation is used to calculate the gradient of the loss function with respect to the parameters.

Backward Propagation



The backpropagation algorithm pseudocode is as follows:

```
1. $DZ^L = P - Y$
2. $\frac{\partial L}{\partial W^L} = \frac{1}{m} DW^L = \frac{1}{m} DZ^L.(H^{L-1})^T$
3. $\frac{\partial L}{\partial b^L} = \frac{1}{m} Db^L = \frac{1}{m} DZ^L$
4. $dH^{L-1} = (W^L)^T.DZ^L$
5. for $l$ in $[L-1, \ldots\ldots\ldots 1]$ :
    1. $DZ^l = dH^l \otimes \sigma'(Z^l)$
    2. $\frac{\partial L}{\partial W^l} = \frac{1}{m} DW^l = \frac{1}{m} DZ^l.(H^{l-1})^T$
    3. $\frac{\partial L}{\partial b^l} = \frac{1}{m} Db^l = \frac{1}{m} DZ^l$
    4. $dH^{l-1} = (W^l)^T.DZ^l$
```



dZ — Gradient of the cost with respect to the linear output (of current layer l).

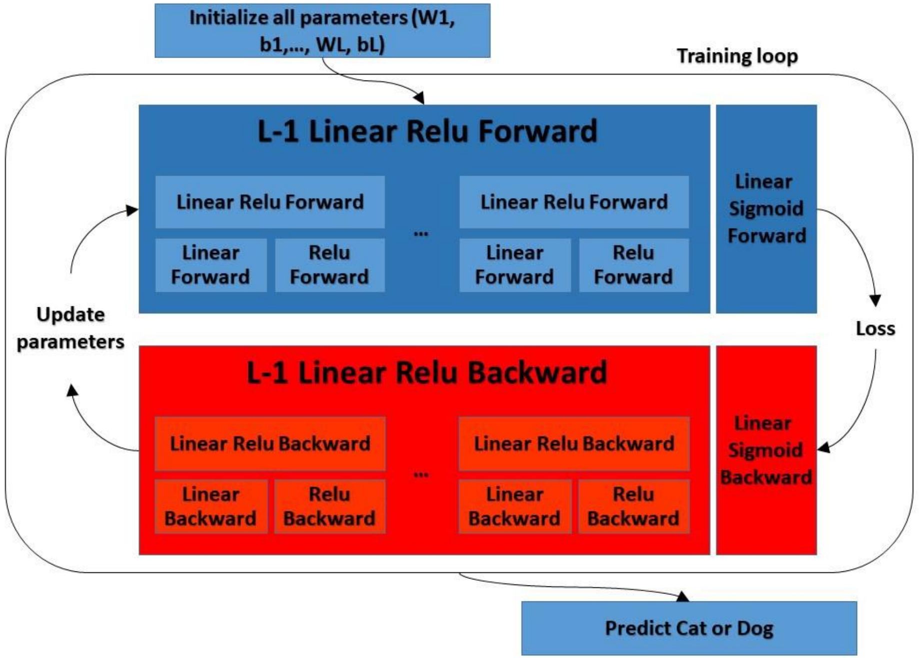
Training

upGrad

Training refers to the task of finding the optimal combination of weights and biases to minimize the total loss

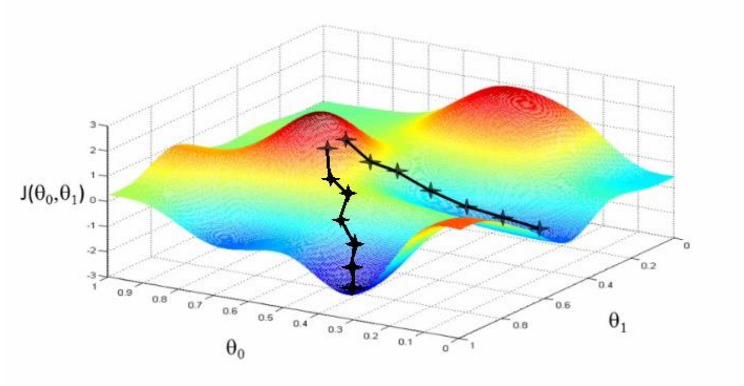
The optimization is done using the familiar **gradient descent** algorithm. In gradient descent, the parameter being optimized is iterated in the direction of reducing cost according to the following rule

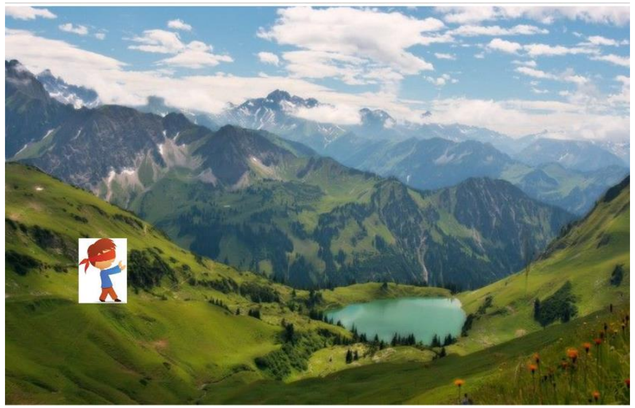
$$W_{new} = W_{old} - \alpha . \frac{\partial L}{\partial W} .$$



Gradient Descent

Starting at the top of the mountain, we take our first step downhill in the direction specified by the negative gradient. Next we recalculate the negative gradient (passing in the coordinates of our new point) and take another step in the direction it specifies. We continue this process iteratively until we get to the bottom of our graph, or to a point where we can no longer move downhill–a local minimum.





Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient

upGrad
#LifeKoKaroLift

Thank You!

Introduction to Reinforcement learning by Sutton and barto
https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf