Welcome to Deep Learning Online Bootcamp

Working of Neural Network

dφ

Democratizing Data Science Learning

DPhi

Learning Objectives

Story time of A.N.N & Non Linear Boundary

Neural Network Architecture

Forward Propagation

Gradient Descent - Refresher

Backpropagation

DPhi

The Story of a Neural Network, also known as Artificial Neural Network - A.N.N.

ANN

ANN is a unique computer who doesn’t want to be programmed like others. She wants to learn on her own, even if it takes longer and involves a lot of trial and errors.

Was she able to succeed? Watch the video to find out.

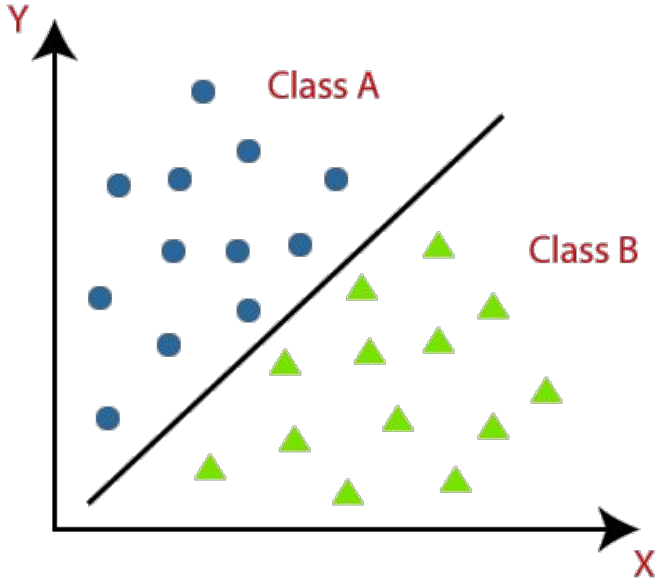
DPhi

ANN

DPhi

Classification

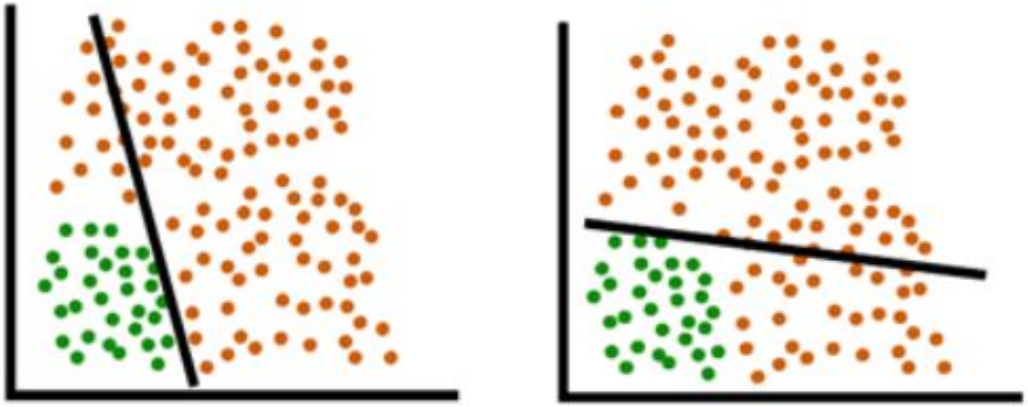
- **Classification** is a process of dividing the given data points into two or more classes.
- **For example,** in the given image we are classifying the data into two classes **class A** and **class B**.
- The data in the image are classified with a straight line (i.e. linear boundary)



DPhi

Non - Linear Boundary

Non-linear Boundary

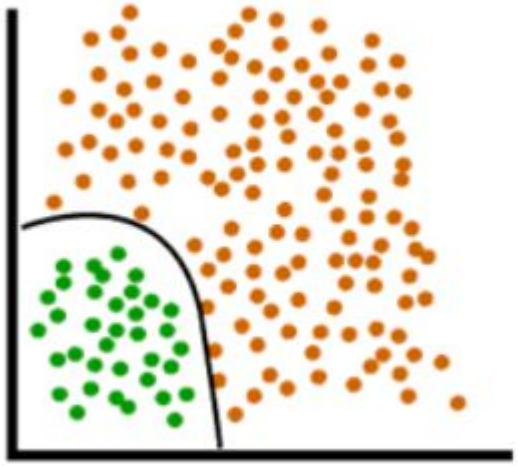


Both the models above are unable to classify our data into 2 classes because in both the cases the red points (significant in number) is present on the both sides of the line.

We thus need something better than a straight line to divide our data into 2 separate classes.

Non-linear Boundary

Realistic data is much more complex and not always classified by a straight line. For this purpose, we need a non-linear boundary to separate our data. Perceptron model works on the most basic form of a neural network, but for realistic data classification, we use **Deep Neural Network** or **Multi Layer Perceptrons**.

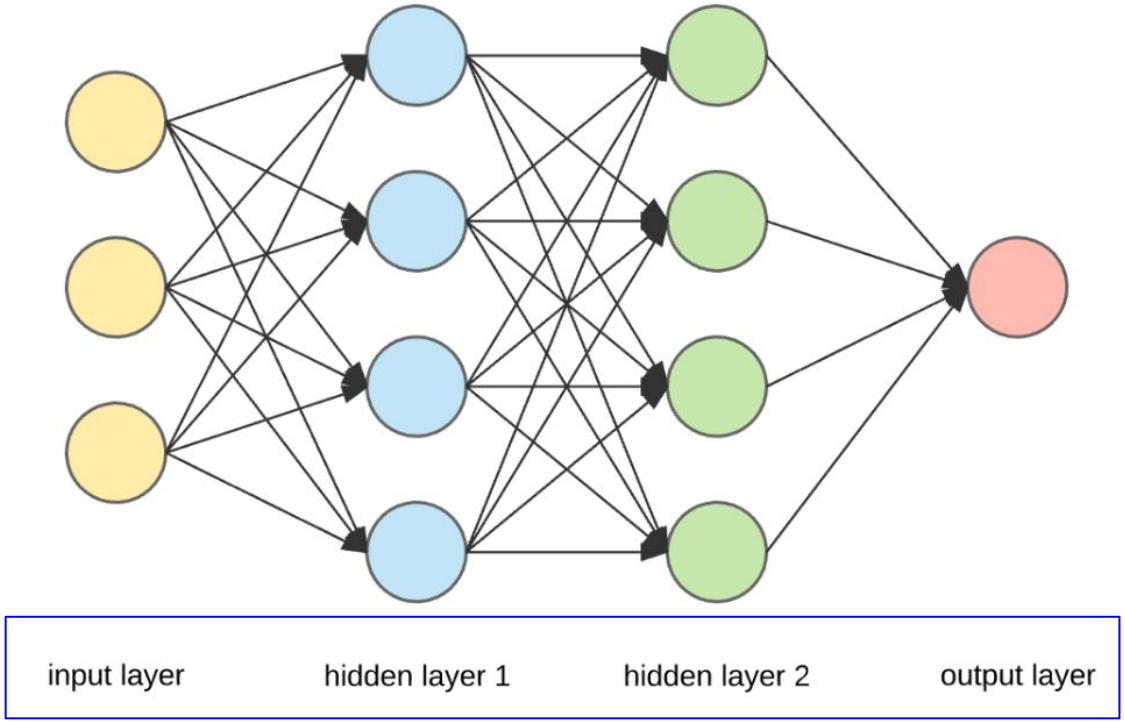


DPhi

Neural Network Architecture

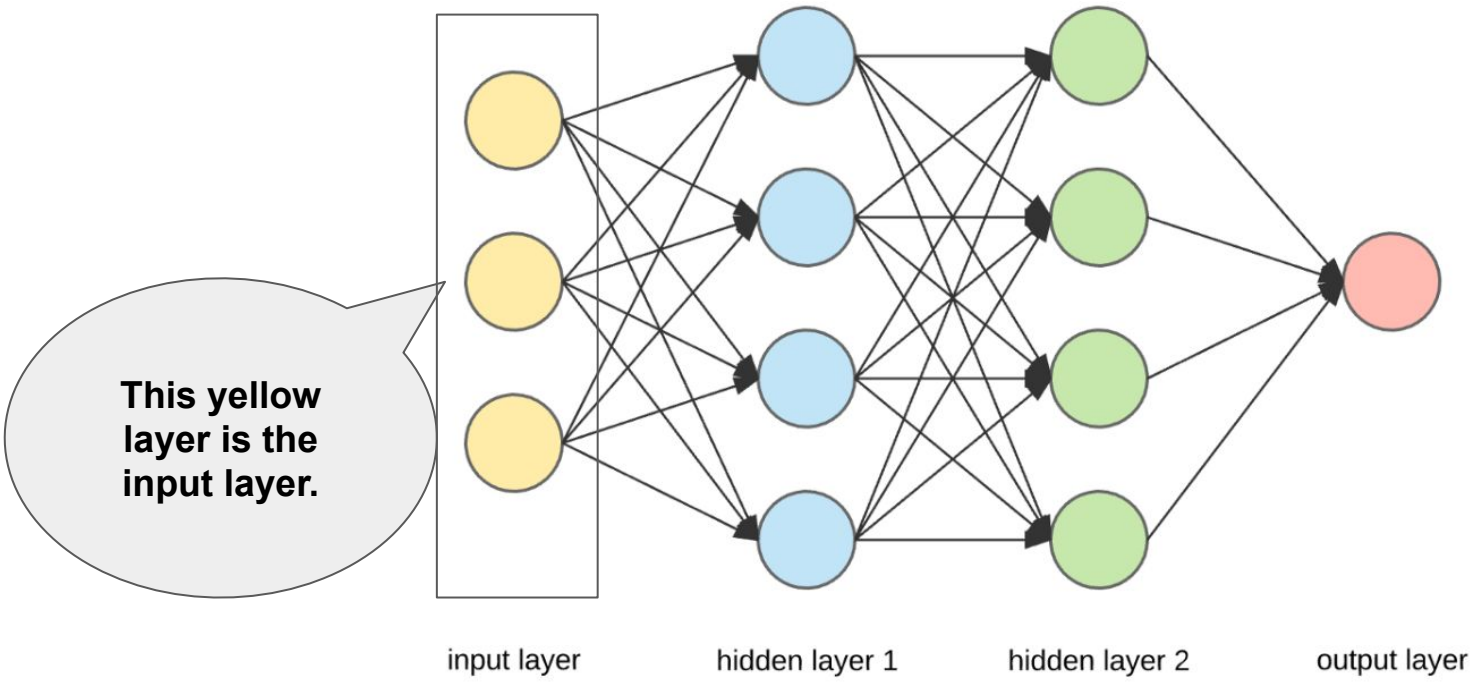
Neural Network: Architecture

- Neural Networks are complex structures made of artificial neurons that can take in multiple inputs to produce a single output.
- This is the primary job of a Neural Network
- Simple terms, to transform input into a meaningful output
- NN consists of an input and output layer with one or more hidden layers.



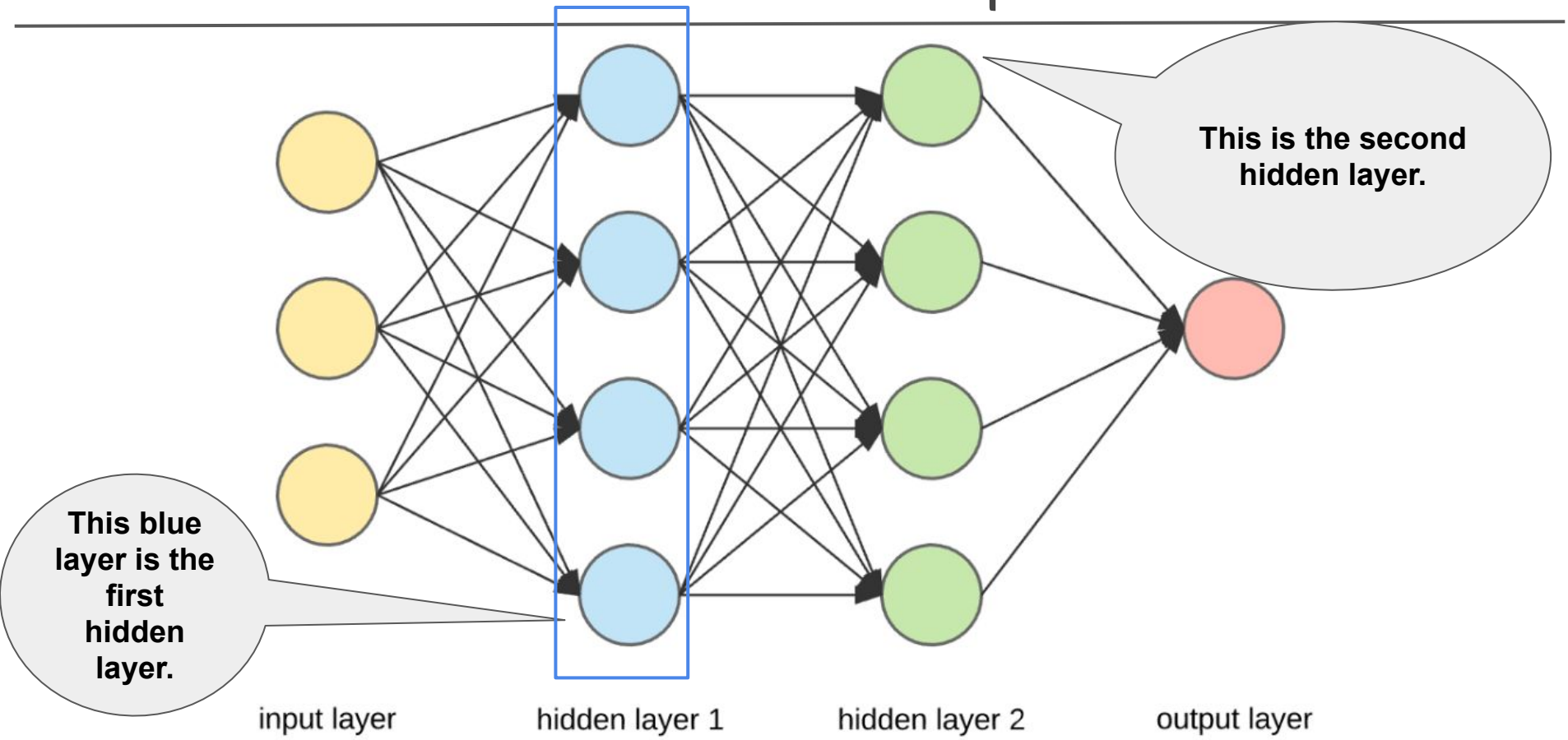
DPhi

Neural Network Components



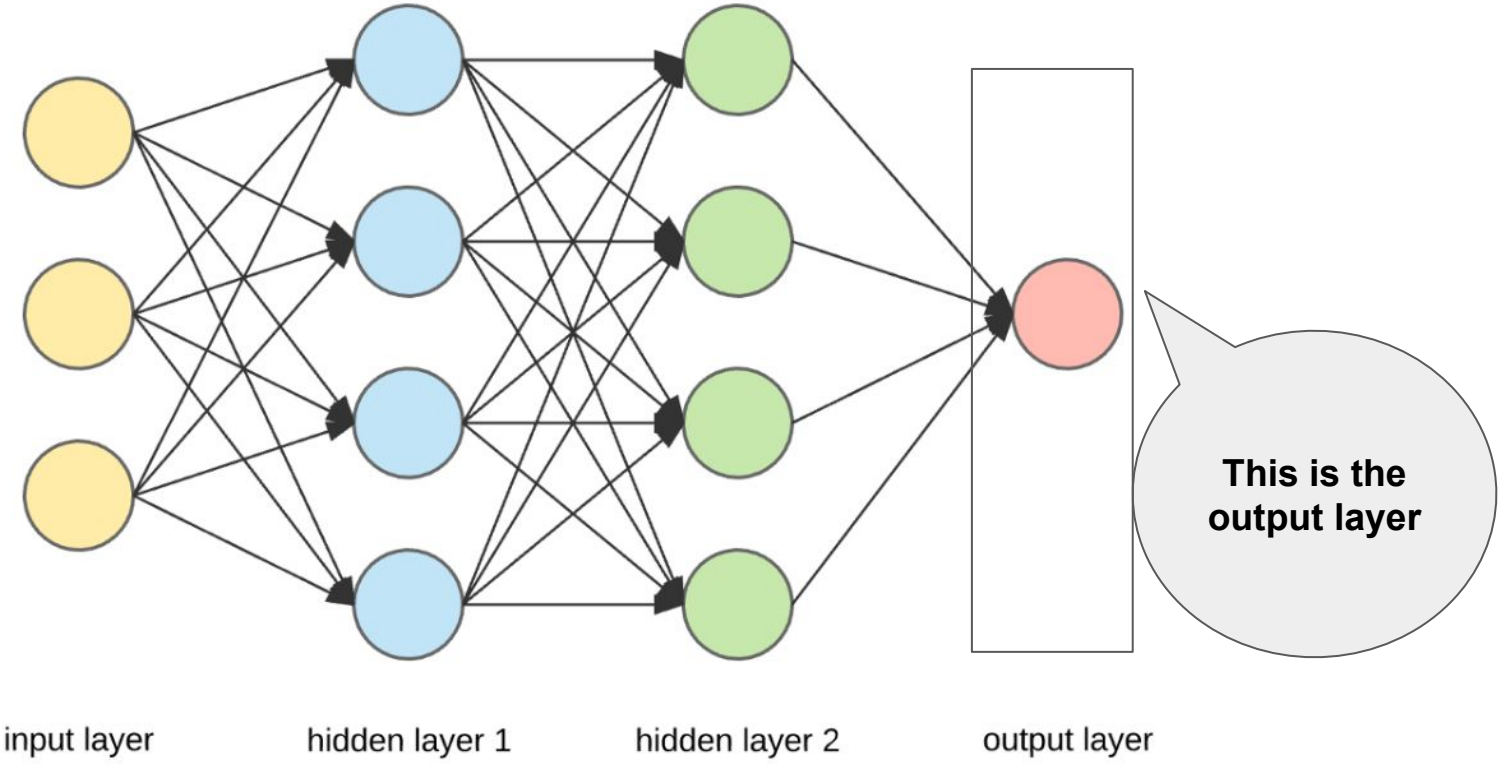
- Outermost yellow layer is the **input layer**. A neuron is the basic unit of a neural network. In this case the input layer has three neurons. The inputs are simply the measure of our features. For example, in the boston house price data in the Linear Regression with tf.keras notebook, there were 13 input features, so the input layer will have 13 neurons.

Neural Network Components



- The blue layer and the green layer are two **hidden layers** which are not directly observable while building the neural network. The number of neurons in these hidden layers are initially assigned by us and we can find the optimal number of neurons in hidden layer through hyperparameter tuning.

Neural Network Components

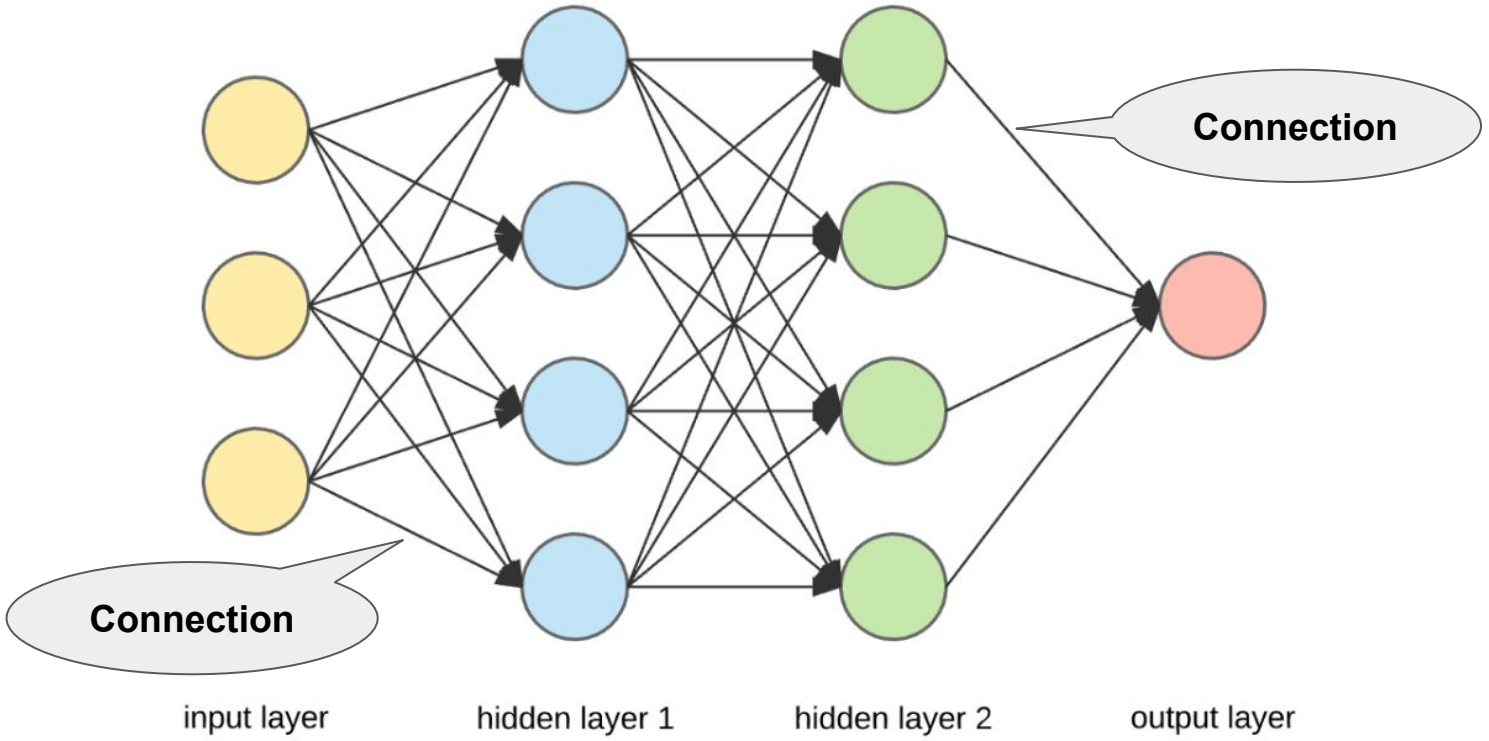


- The red layer is the **output layer**. This is the last layer of a neural network that produces the required output. For example, **'MEDV'** (Median value of owner-occupied homes in 1000 USD's) was the output layer in [boston house price dataset].

Image source: UpGrad

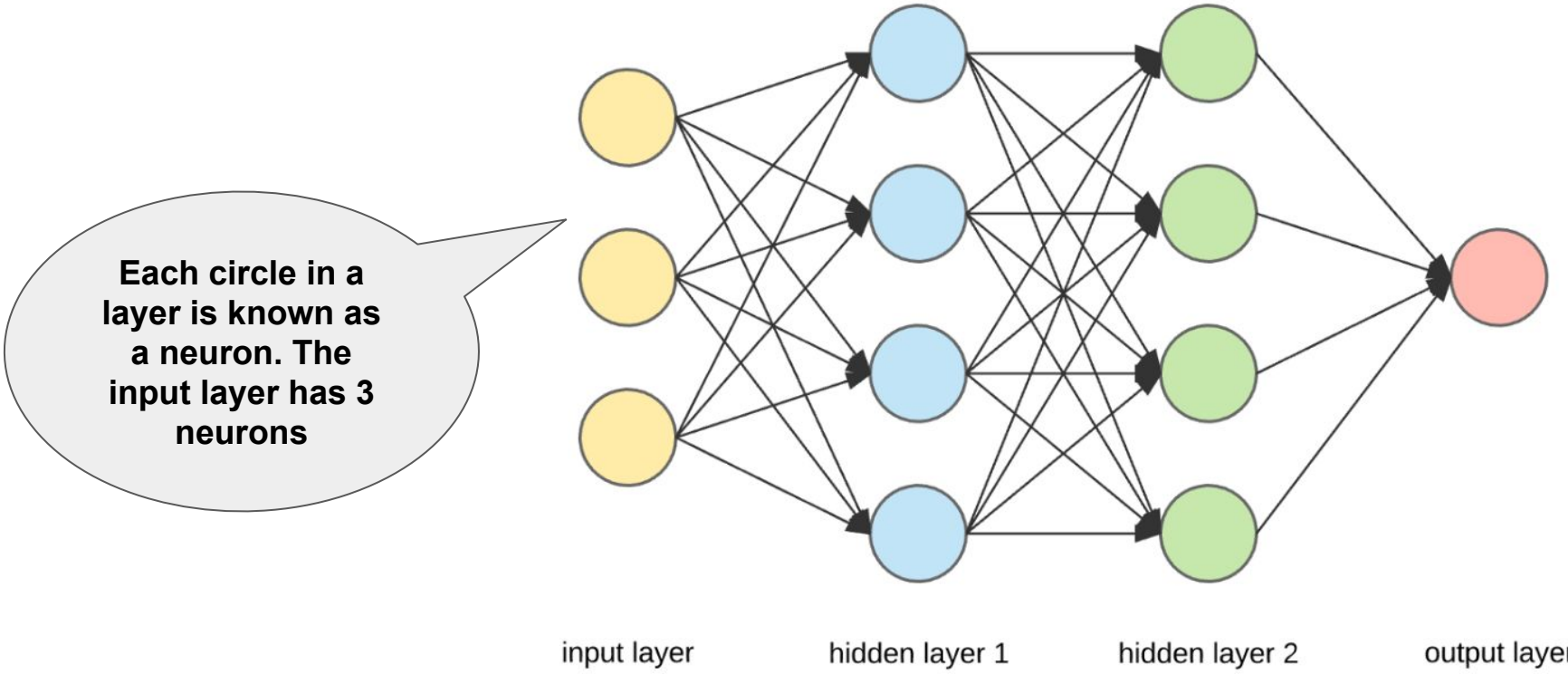
DPhi

Neural Network Components



- **Weights:** There is some weight assigned for each connection. Weights represent scalar (constant) multiplication. Initially these are assigned randomly, then these weights are updated as per their importance in predicting the output. The updation of weight is done through back propagation (you will know this in upcoming slides).

Neural Network Components

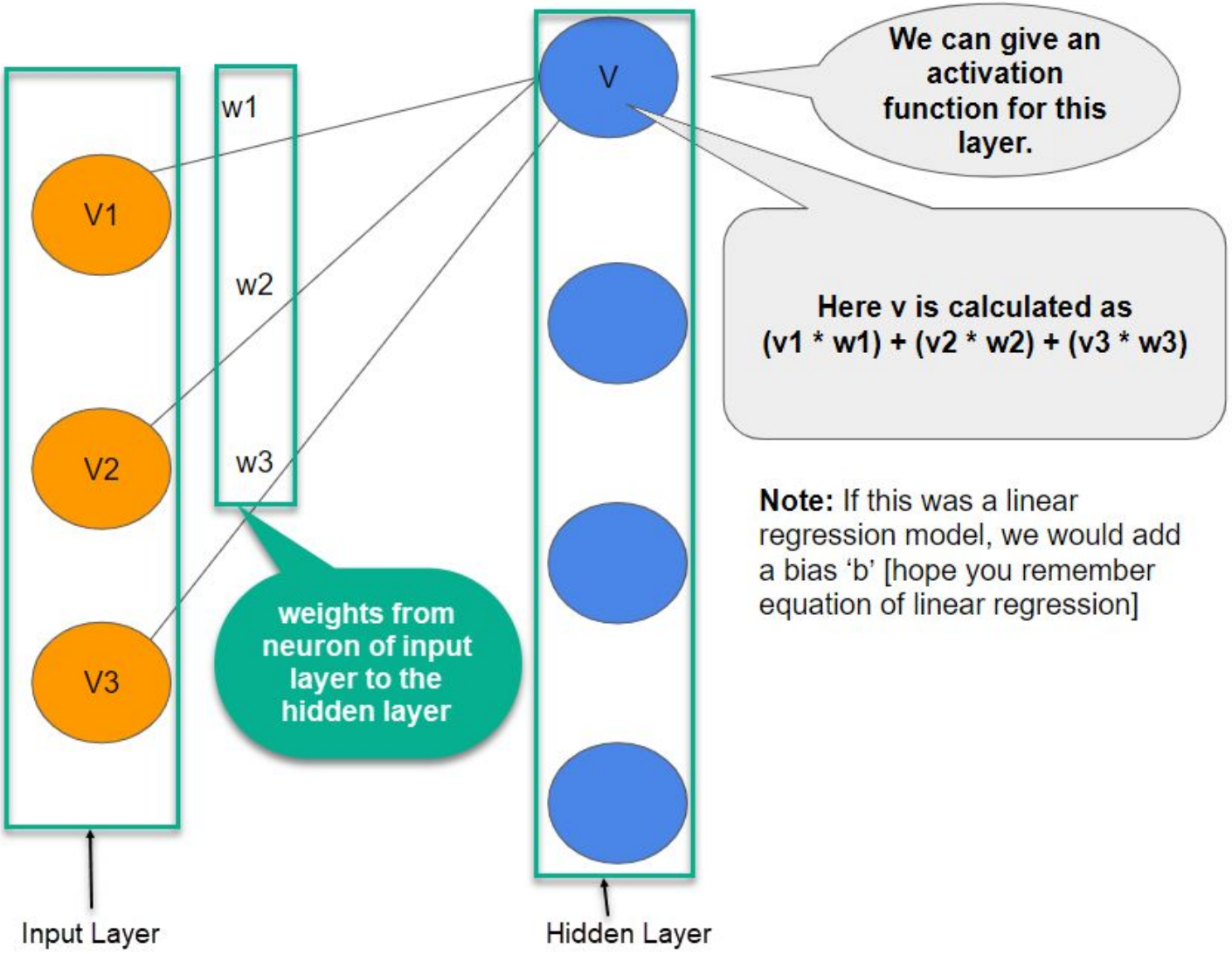


- Now we know each neuron in input layer has a value given in the dataset. Some random weights are assigned for each connection. When the value of each neuron in input layer is multiplied with its respective weight for the next connecting neuron and added all together, produces the value of next connecting neuron in the next layer. Don't worry if you don't understand here. An example of this calculation is shown in next slides.

DPhi

Calculating Value of a Neuron

A simple example on how the values of each neurons are calculated using weights and input values.



Calculating Value of a Neuron

- In the previous slide: w1 is the weight from first neuron of input layer to the first neuron of hidden layer. Same goes with w2 and w3
- The value of each other neuron in each hidden layer is calculated in the same way we calculated the value of the first neuron of the first hidden layer in the previous slide.
- Similarly, the value of output layer is calculated.

Let’s understand what we learnt using a simple task

Task - Making tea

Now imagine you are a Tea Master.....

The ingredients used to make tea (water, tea leaves, milk, sugar, and spices) are the "**neurons**" since they make up the starting points of the process. The amount of each ingredient represents the "**weight**." Once you put in the tea leaves in the water and add the sugar, spices, and milk in the pan, all the ingredients will mix and transform into another state. This transformation process represents the "**activation function**."

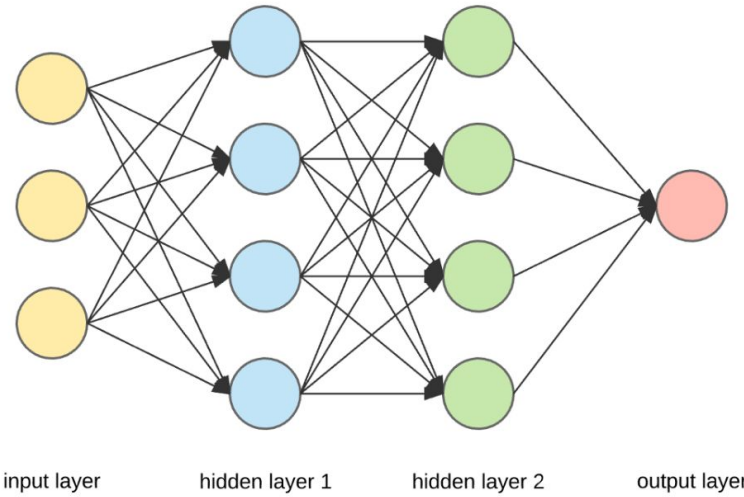
Hidden Layers and output Layer

- The layer or layers hidden between the input and output layer are known as the **hidden layers**. It is called the hidden layer since it is always hidden from the external world. The main computation of a Neural Network takes place in the hidden layers. So, the hidden layer takes all the inputs from the input layer and performs the necessary calculation to generate a result. This result is then forwarded to the output layer so that the user can view the result of the computation.

- In our tea-making example, when we mix all the ingredients, the formulation changes its state and color on heating. The ingredients represent the hidden layers. Here heating represents the activation process that finally delivers the result – tea.

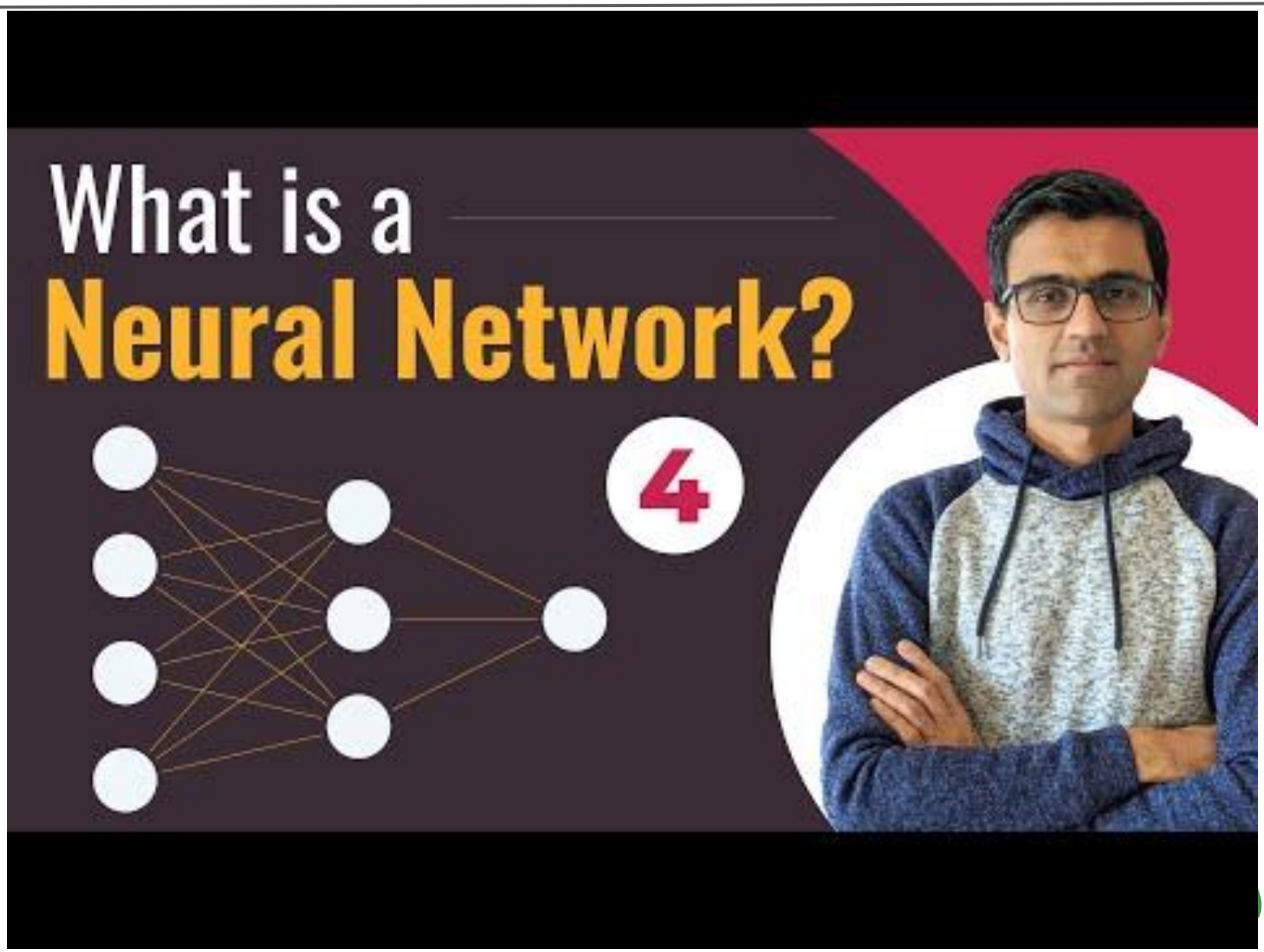
Summary of Tea Example

- **Input layers:** water, tea leaves, sugar, milk and spices
- **Hidden layers:** all the above ingredients mixed with certain weights
- **Activation function:** The heating process
- **Output layer:** Tea



DPhi

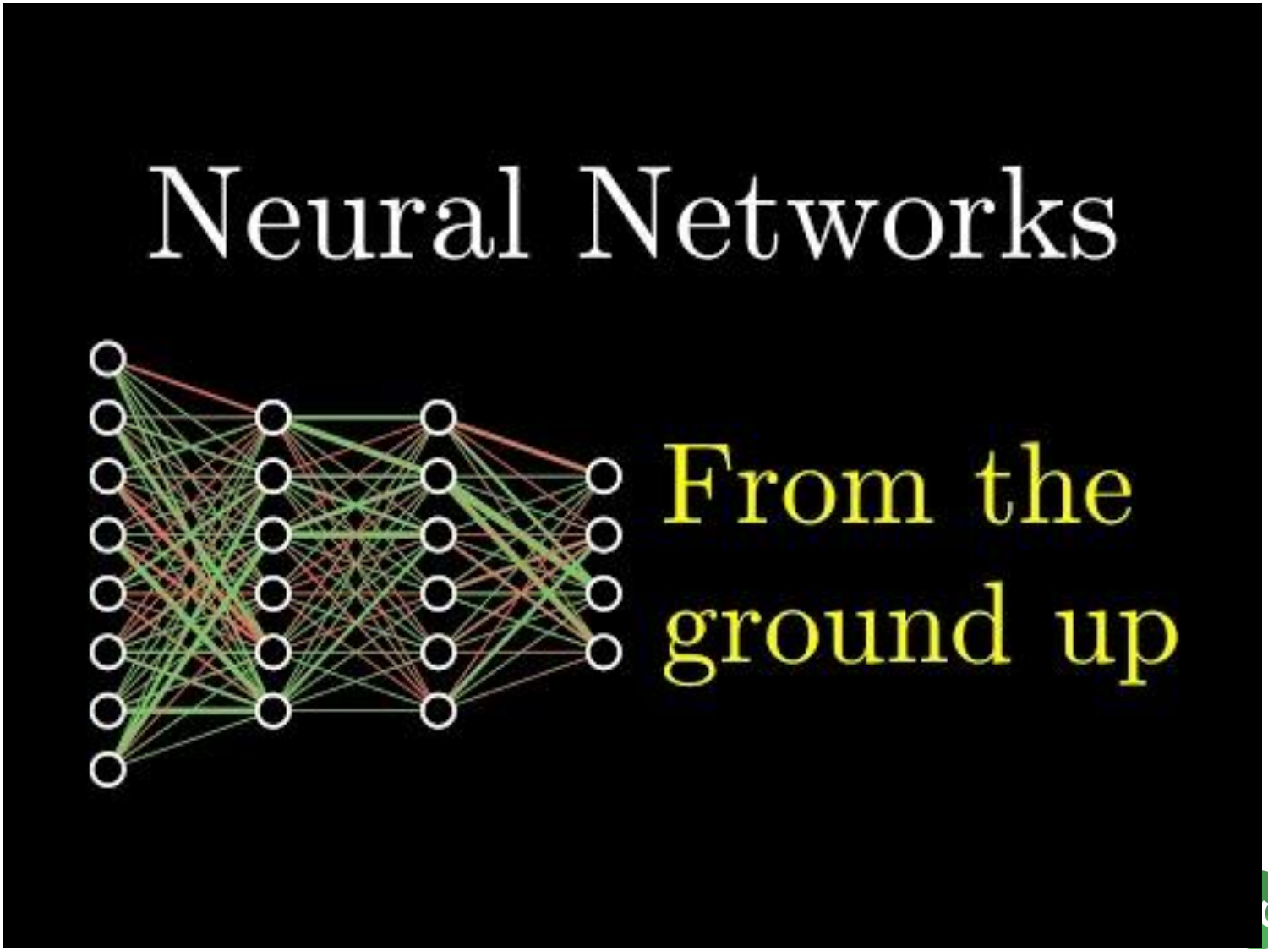
Another intuitive Explanation of Neural Networks



DPhi

Explanation of Neural Networks (Optional)

If you like understanding things through Mathematics, this is for you! 😆



DPhi

Working of a Neural Network

In the next set of slides, we will be discussing about the working on Neural Networks that includes 3 main steps:

1. Forward Propagation
2. Gradient Descent
3. Backward Propagation

What are they? These are the three important pillars of the neural network working and we will learn more about its prominence in next few slides.

Why Forward Propagation in NN?

- In order to generate output or desired result, we need to feed input data. Forward propagation helps us do that in Neural Networks. The data in NN should be fed in the forward direction only.

Why Gradient Descent?

- Just think of Gradient Descent as a technique to minimise the loss/cost function.
  It helps in making the model perform better.

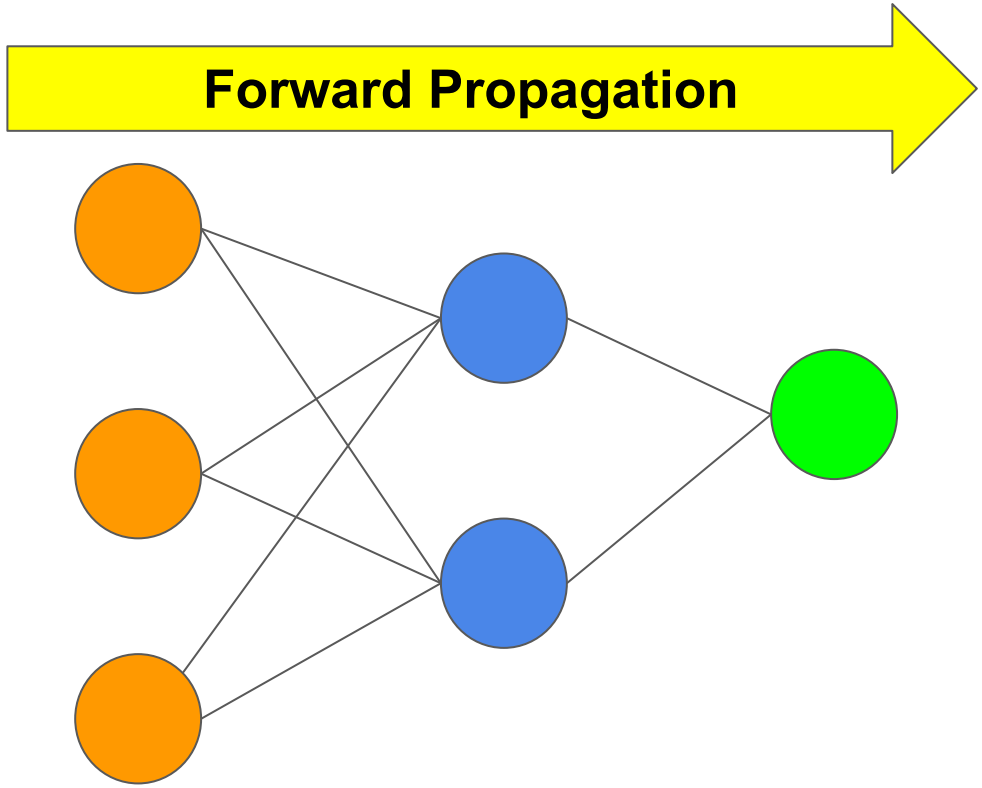
Why Backward Propagation in NN?

- When you generate some output through forward propagation, there are some errors generated in the process. To reduce these errors, we traverse back from the output layer to input layer and update the initially assigned weights.

1. Forward Propagation

What is Forward Propagation in NN?

- Well, if you break down the words, **forward** implies moving ahead and **propagation** is a term for saying spreading of anything.
- Forward propagation means we are moving in only one direction, from input to the output, in a neural network.



DPhi

What is Forward Propagation in NN?

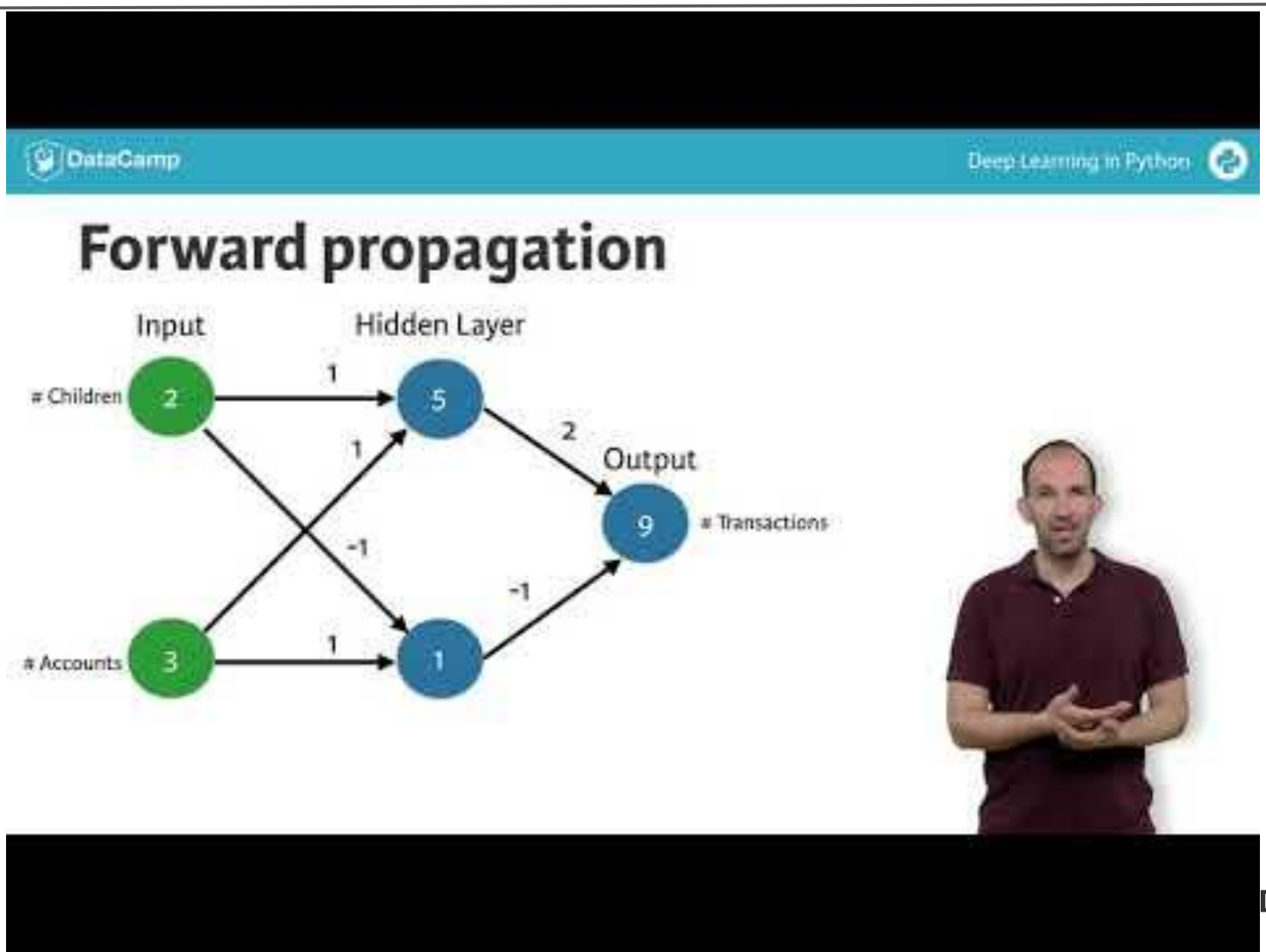
The video in the next slide takes a Bank Transaction Dataset Example. The dataset has two features; number of children and accounts and the objective is to predict how many transaction will a user make at bank.

The video covers the following aspects:

- How neural network model uses data to make predictions
- How the information is transferred from input layer to the output layer (through weights and hidden layers)
- The calculation of value of each neurons in hidden layers using weights and the input values
- And finally calculating the output

Note: In the video, the inputs (2,3) and weight allocation (1, -1 etc) are randomly assigned for explanation. NO need to bother to know how they started appearing all at a sudden!

Working of Forward Propagation in NN



2. Gradient Descent Refresher

Gradient Descent

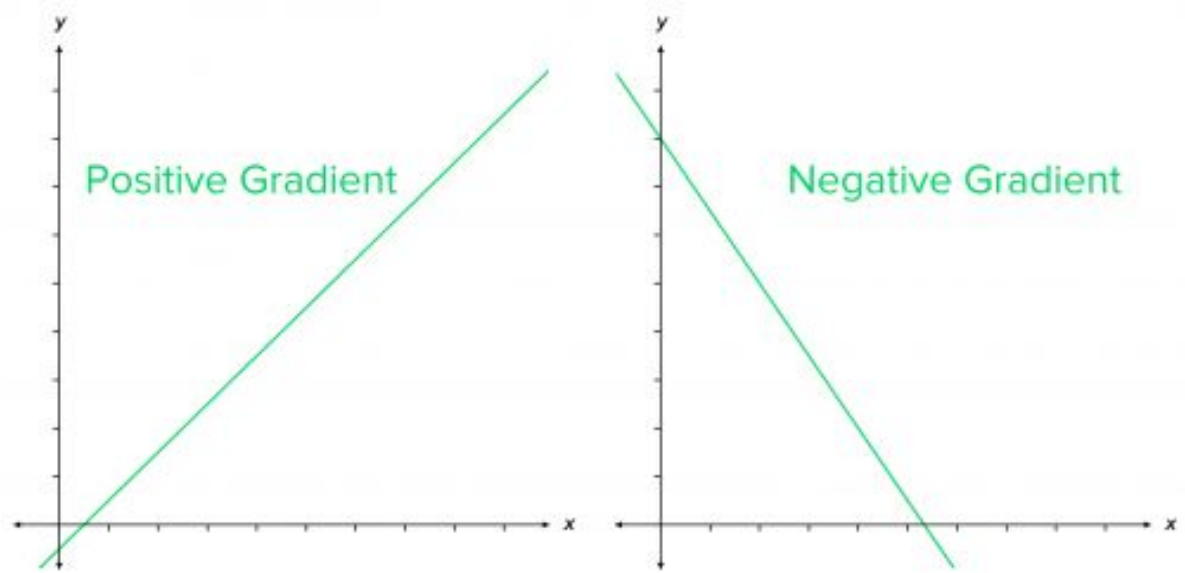
Gradient Descent!

Gradient Descent (GD) is an optimization technique in the machine learning process which helps us **minimize the cost function** by learning the weights of the model such that it fits the training data well.

Gradient Descent - let's break it down

Gradient = rate of inclination or declination of a slope (**how steep** a slope is and in **which direction** it is going) of a function at some point.

Descent = an act of moving downwards.



Simple way to memorize:

- line going up as we move right → positive slope, gradient
- line going down as we move right → negative slope, gradient

DPhi

Gradient Descent- A technique to minimize cost

At the start, the parameter values of the hypothesis are randomly initialized (can be 0)

Then, Gradient Descent **iteratively** (one step at a time) adjusts the parameters and gradually finds the best values for them by minimizing the cost

In this case we'll use it to find the parameters **m** and **c** of the linear regression model

m = **slope** of a line that we just learned about
c = bias : where the line crosses the Y axis.

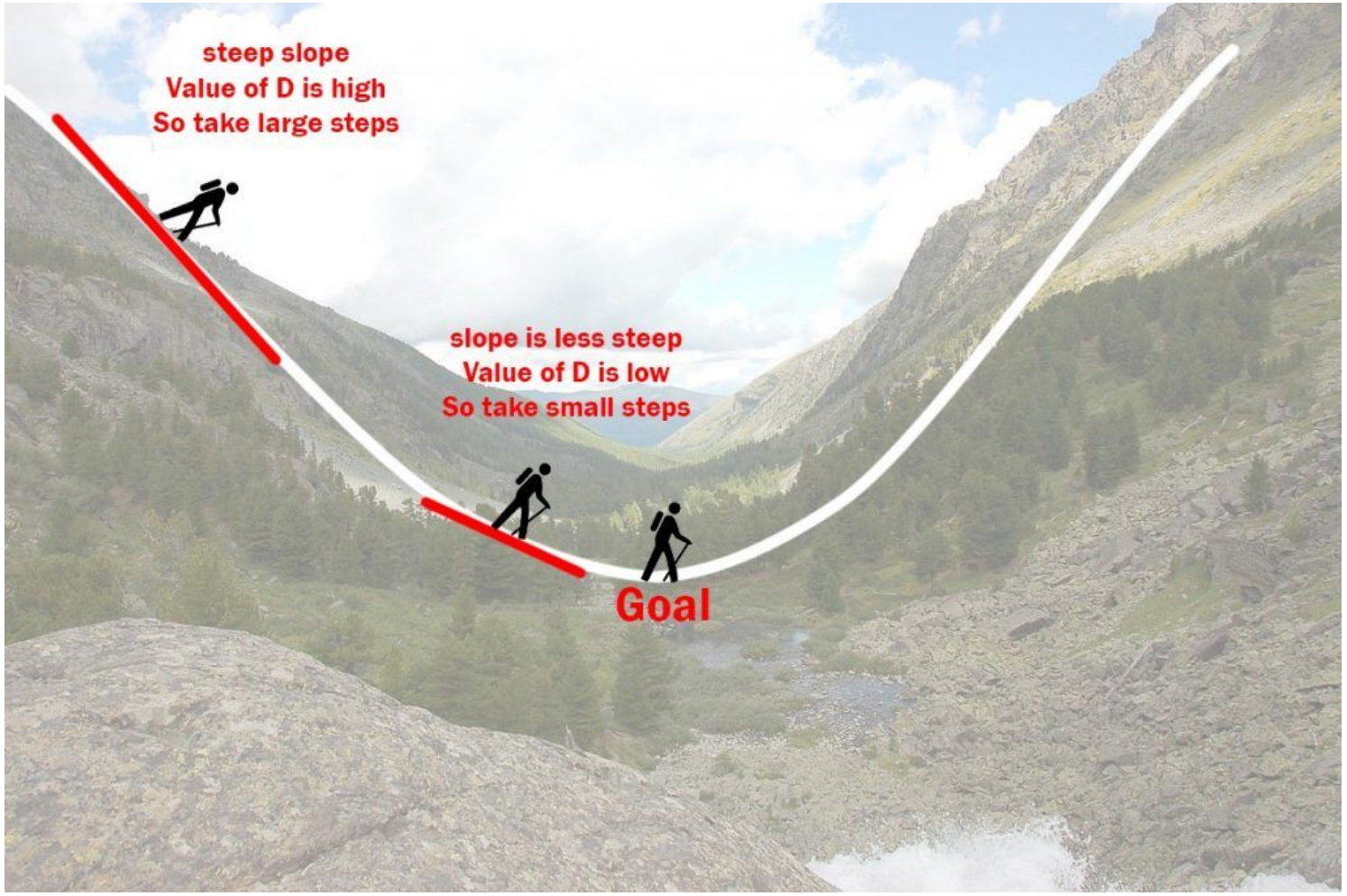
Gradient Descent - An Example

Imagine a person at the top of a mountain who wants to get to the bottom of the valley below the mountain through the fog (he cannot see clearly ahead and only can look around the point he's standing at)

He goes down the slope and takes **large steps when the slope is steep** and **small steps when the slope is less steep**.

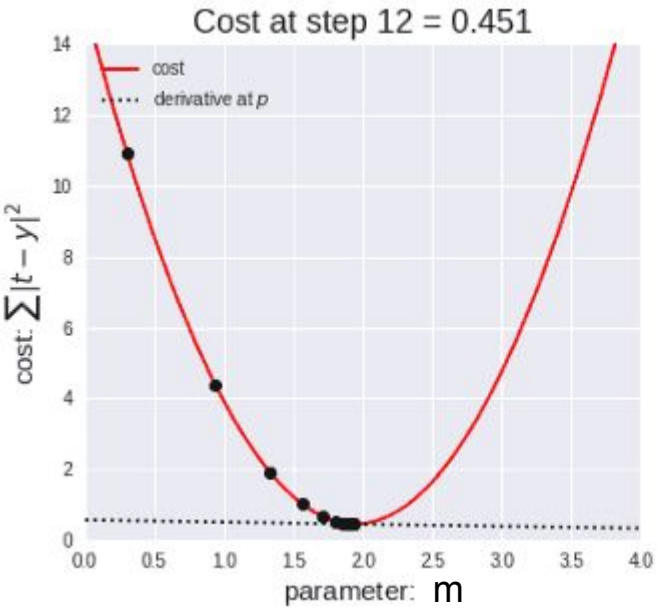
He decides his next position based on his current position and **stops when he gets to the bottom of the valley which was his goal.**

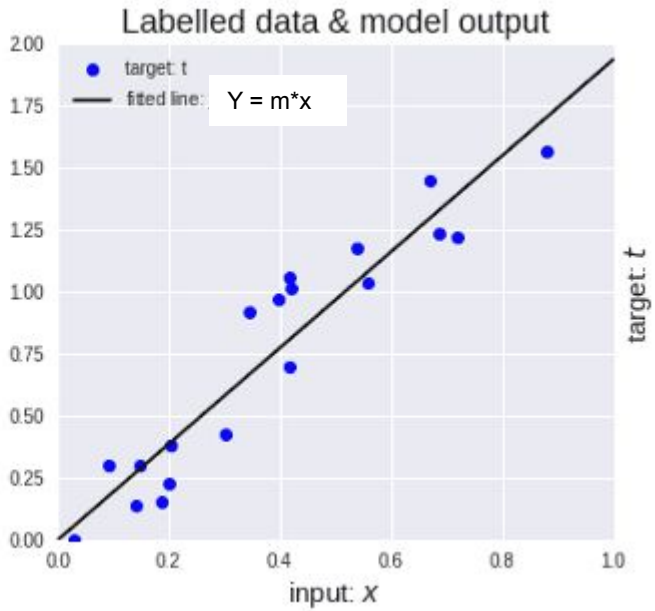
Gradient Descent



DPhi

Gradient Descent in action





Now, think of the valley as a cost function. The objective of GD is to minimise the cost function (by updating it's parameter values). In this case the minimum value would be 0.

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

DPhi

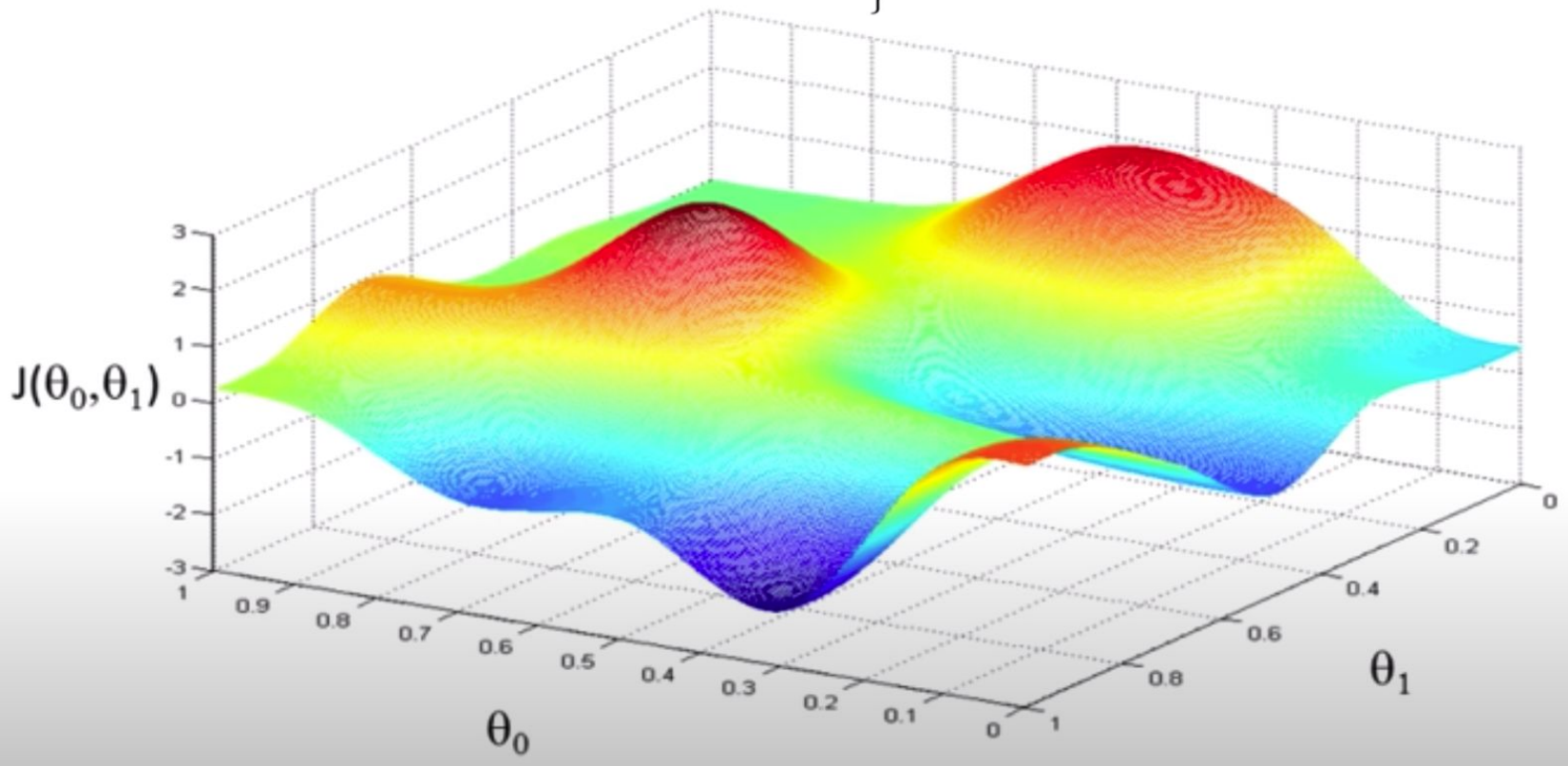
What does a cost function look like?

J is the cost function, plotted along it's parameter values. Notice the crests and troughs.

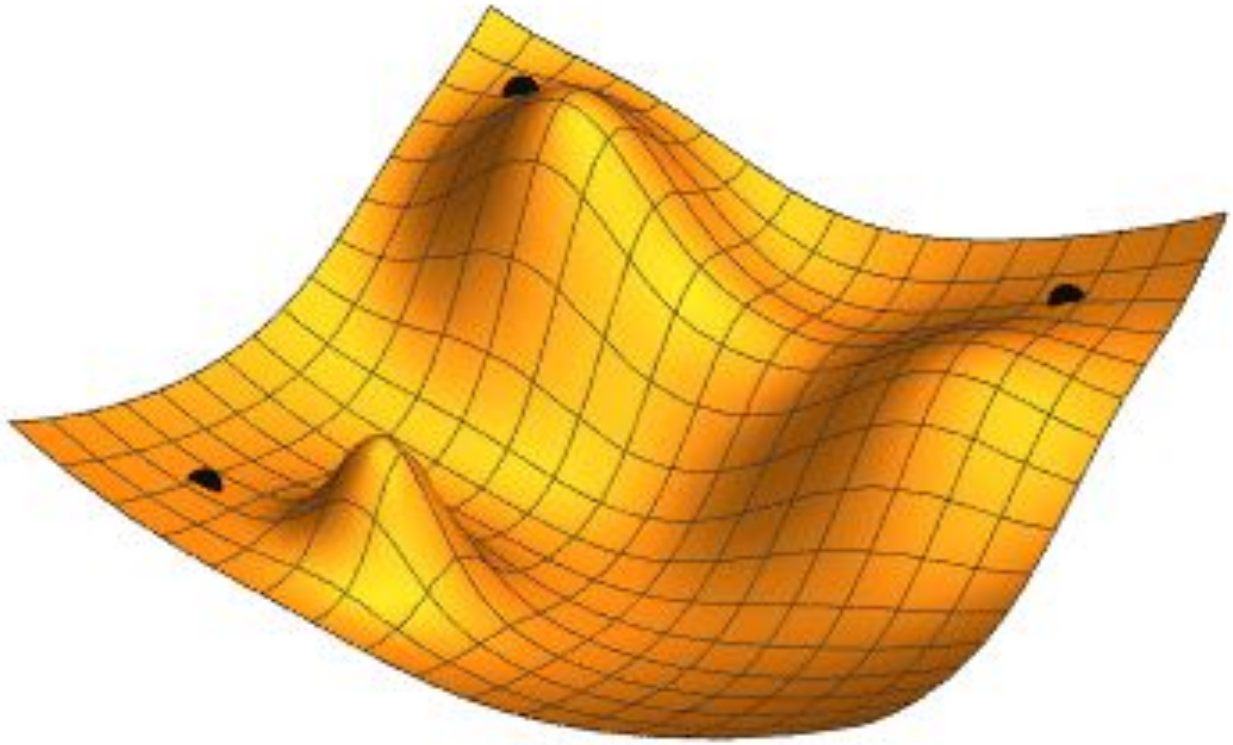
repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$ (simultaneously update $j = 0$ and $j = 1$)

}



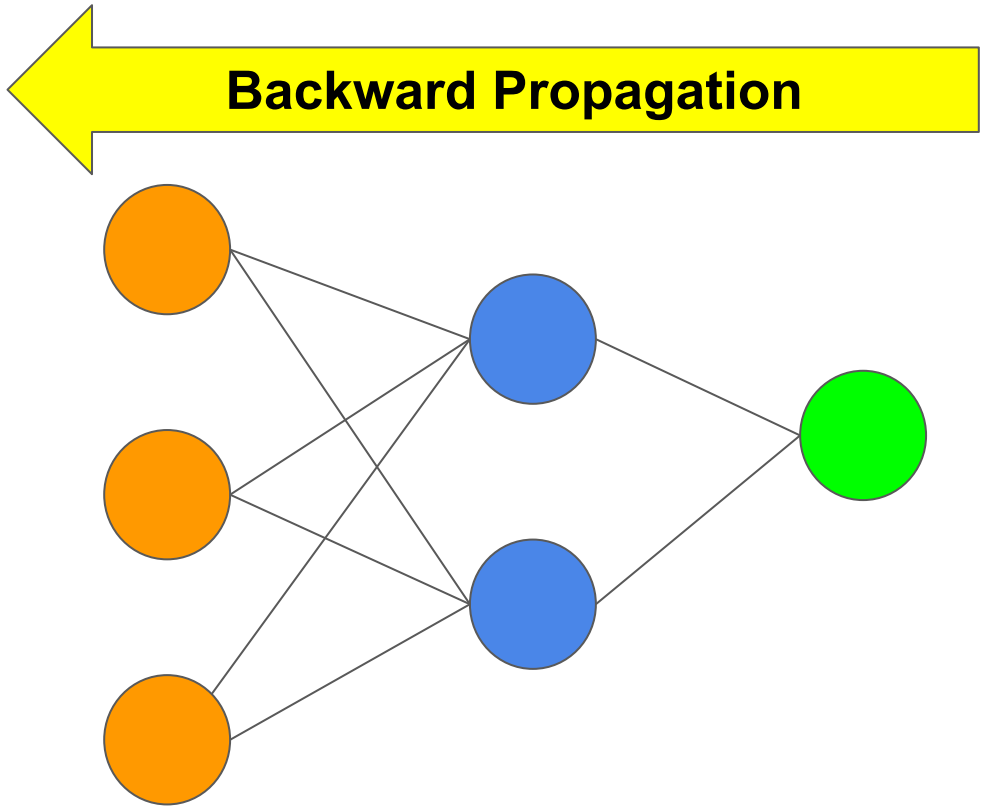
How we navigate the cost function via GD



3. Backward Propagation

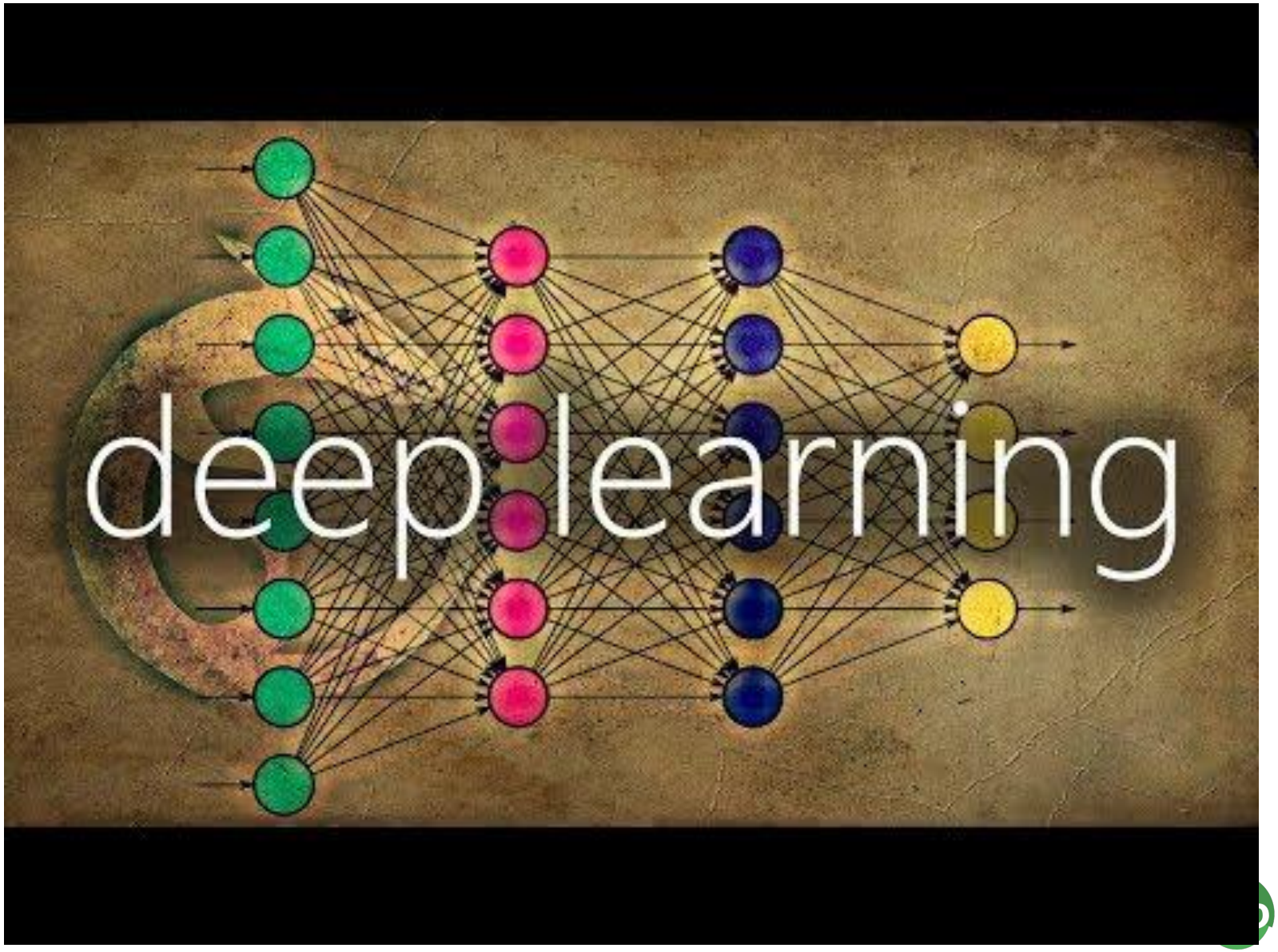
What is Backward Propagation in NN?

- **Backward propagation** means we are moving in only one direction, from output to the input, in a neural network.
- Backward propagation is also called as **Back Propagation** in short.



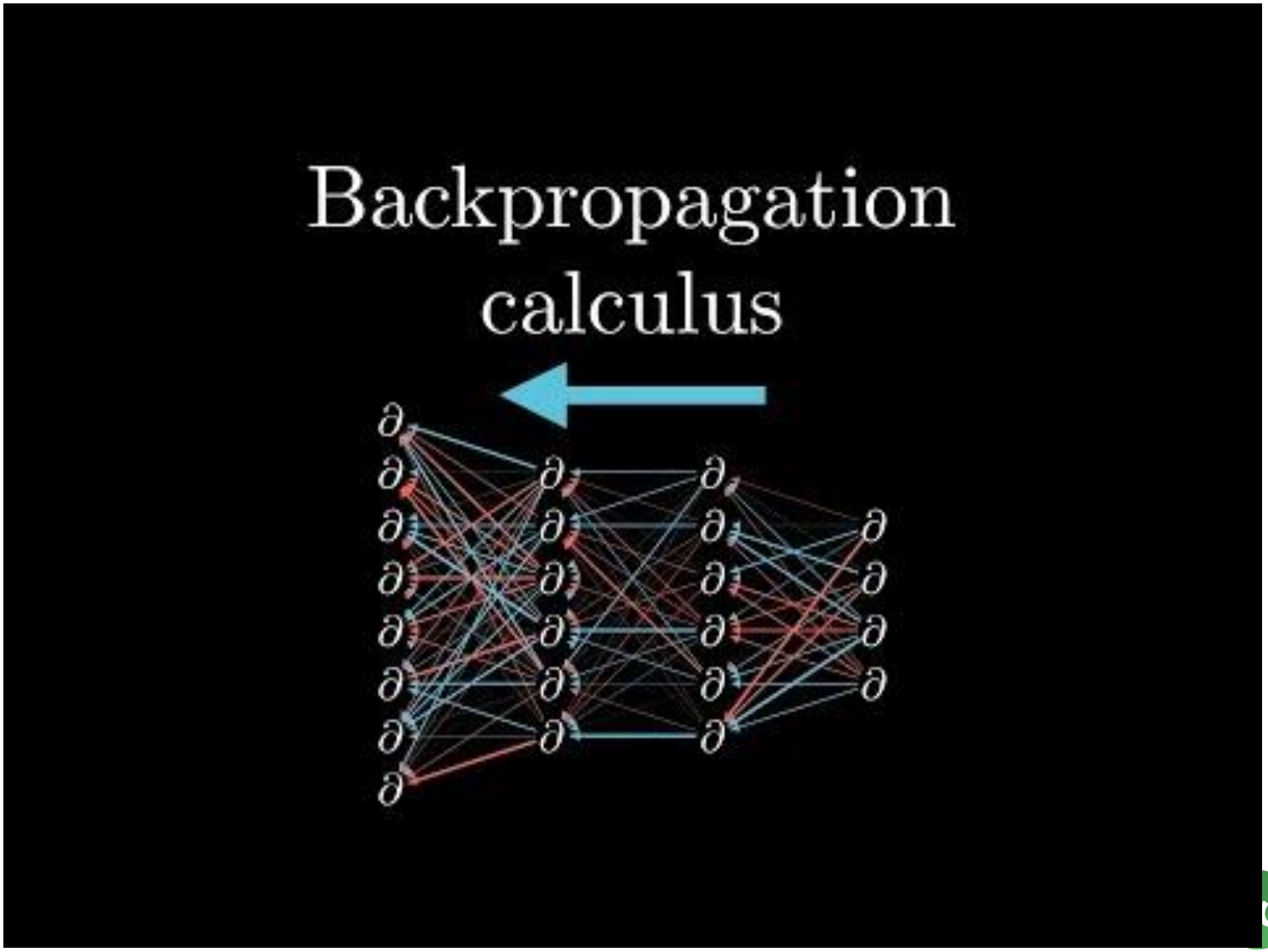
NOTE: Stochastic Gradient Descent (SGD) that the instructor is sometimes mentioning in the next video is just a type of Gradient Descent. Don't worry about its exact working.

What is Backward Propagation in NN?



Backpropagation Calculus (Optional)

If you like understanding things through Mathematics, this is for you! 😆



DPhi

Summarising the working of a NN

- A Neural Network passes data to model via **forward propagation**.
- When it reaches the last(output) layer, it calculates loss on output.
- It then **back propagates** information about the loss/error, in reverse through the network to the input layer, so that it can alter the parameters **weights**.
- Gradient Descent is calculated via backpropagation and works to minimise this loss:
  - By calculating gradient of loss function according to weights
  - Updating weights accordingly

Download link to slides

https://docs.google.com/presentation/d/1giN8HnKYKvLJ59OMJYglU4H0uMCjnqZb7QsQ1lXkXhk/edit?usp=sharing

DPhi

That’s it for this unit. Thank you!

Feel free to post any queries on Discuss
or in the #help channel on Slack

DPhi