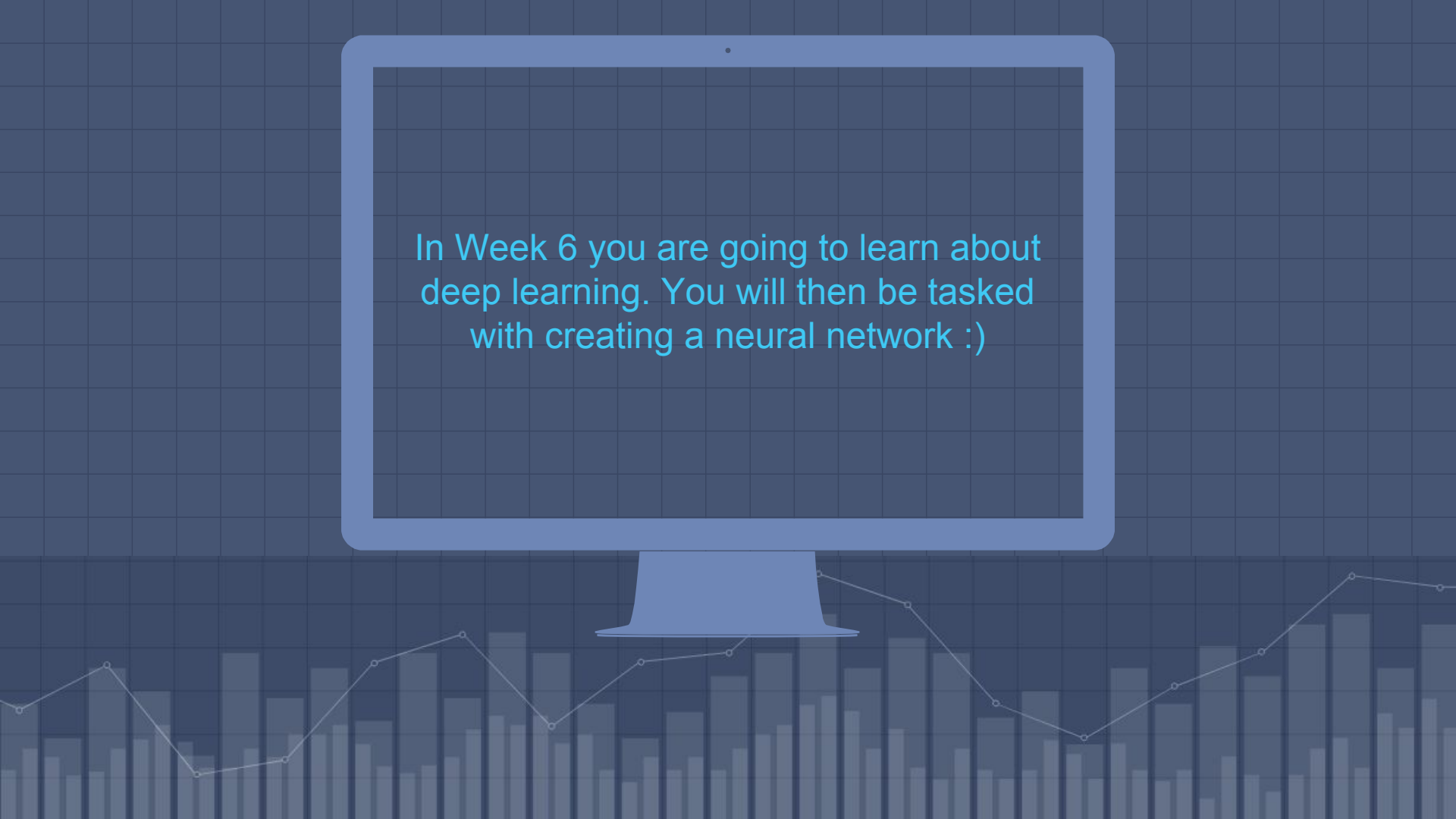


# arise:

## DeepLearning

### Week 6





In Week 6 you are going to learn about  
deep learning. You will then be tasked  
with creating a neural network :)

# Intro into Deep Learning



Deep Learning is a subfield of machine learning. The subfield is concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

Deep Learning is still a machine learning method that takes an input of  $X$ , and uses it to predict an output of  $Y$ .

The only difference is that deep learning algorithms use neural networks to find the associations between a set of inputs and outputs. Neural networks have what are known as hidden layers. The 'deep' part of Deep Learning models simply refers to the presence of these hidden layers.

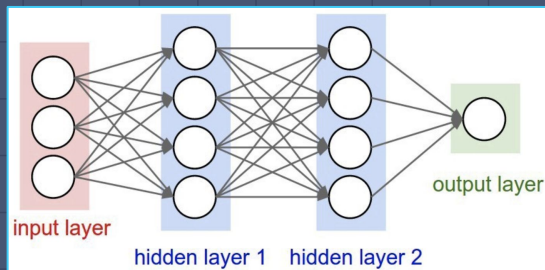
Deep learning algorithms can be applied to binary classification problems, such as training a model to predict whether an image contains a cat or a dog. They can also be applied to multi-class classification problems, such as predicting the specific digit contained in an image (this is the common MNIST digit challenge).

Deep learning algorithms can also be applied to regression problems. A very common example is predicting the future price of stocks on the stock market.

The purpose of a Deep Learning model will determine its structure. We will learn more about the structure of Deep Learning models on the next slide.

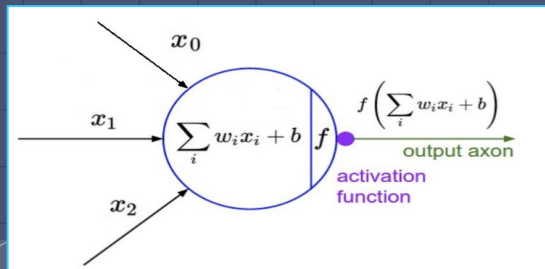
# Architecture of a NN

The structure of a Neural Network (NN) is referred to as its architecture. An example architecture can be seen in the first image on the left hand side of this slide.



The circles within each layer represent nodes and the arrows between nodes represent the weights. Within the input layer, each node represents a variable used within the model.

The structure of an individual node can be seen in the second image on the left hand side of this slide.



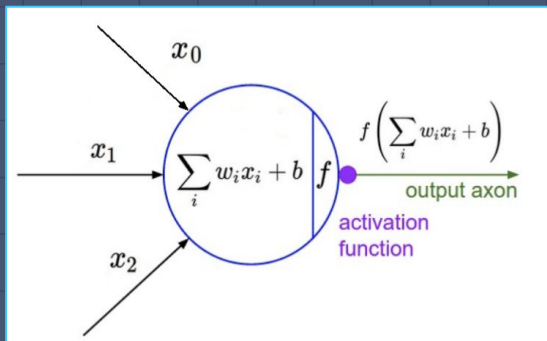
The inputs are mapped to each node of the first hidden layer. Each node of the first hidden layer will receive a weight from each input node. Mathematical calculations are then performed on the weights using what are known as activation functions. The outputs from the mathematical calculations performed in the first hidden layer are then fed to the second hidden layer. The second hidden layer then performs a set of mathematical calculations on the weights fed to them.

Finally, the output weights from the second hidden layer are mapped to the output layer. The output layer can be considered as the final Y value.

The input layer of a NN will have as many nodes as the number of variables used in the model. A NN will only have one input layer. NNs can have one or more hidden layers and each hidden layer can have a different number of nodes. A NN will always have one output layer. The number of nodes in the output layer is dependent on the number of classes being predicted by the model. A binary classification model will have an output layer that only has one node. A multi-class model will have as many nodes as classes. A regression NN will have a one node output layer.

# The Actual "Learning" Process

Let's take a closer look at the nodes. This is where the actual training of an NN is done. The weight and bias parameters, represented by  $w$  and  $b$  respectively in the activation function on the left, are essential to the actual "learning" process of a deep learning algorithm.



The parameters are initialised before the model is first run. After the neural network passes its inputs all the way to its outputs, the network evaluates how good its prediction was (relative to the expected output) through something called a loss function. As an example, the "Mean Squared Error" loss function is shown directly below.  $\hat{Y}$  represents the actual prediction, while  $Y$  represents the expected output.

$$\frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

A NN ultimately wants to minimize the loss function by altering the weights and biases within the network. This is done using something called "back propagation" with the use of stochastic gradient descent. The network backtracks through all its layers and updates the weights and biases of every node in the reverse direction of the network. In other words, every iteration of back propagation should result in a smaller loss function than before.

# Gradient Descent



Gradient Descent can be thought of as a climber walking down the steep side of a valley down to its bottom. Initially in very steep places the climber will take small steps, when the side flattens out they can then take bigger strides. The valley side in the analogy is the loss function in our deep learning model.

The equation below describes what Gradient Descent does. "b" describes the next position of our climber, while "a" represents their current position. The minus sign ensures gradient descent minimises the climber's path. The gamma in the middle is a waiting factor and the gradient term (  $\nabla f(a)$  ) is simply the direction of the steepest descent.

$$\mathbf{b} = \mathbf{a} - \gamma \nabla f(\mathbf{a})$$

The above formula essentially provides the next position of the steepest descent. In deep learning the fastest way to reduce the loss function is provided.



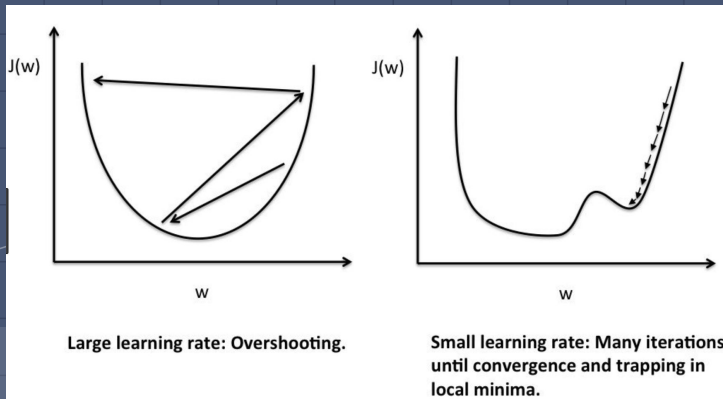
# Learning Rates



Learning rates are directly linked to GD. They define the rate at which the loss function descends towards its local minimum with each training iteration.

If a learning is too high the minimum of the loss function can be over shot or diverged from, meaning the loss function is never minimised while the model is training.

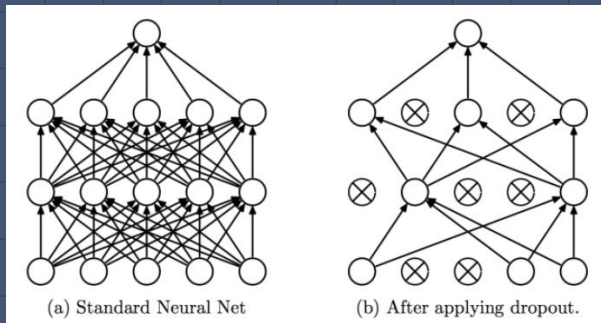
If a learning rate is too small then convergence to the minimum loss function can be too slow, which would also lead to the loss function not be minimised throughout training.



# Drop Out



Dropout is the random removal of nodes from the hidden layers on each training iteration of the deep learning model. It means that the model is essentially trained on numerous architectures. It is used to avoid overfitting.





# Summary



This reading provides a fantastic summary of all the terms and concepts associated with deep learning. The article will reinforce everything we have learnt and will add some additional snippets of knowledge.

# Deep Learning Model Types



The following are examples of very common deep learning techniques used in the data science industry. Please read through each reading and ensure you have a good understanding of each technique.

- [Convolutional Neural Networks](#)
- [Recurrent Neural Networks](#)
- [Transfer Learning](#)

# Practical Exercise



We are going to build a convolution neural network that is capable of classifying 10 different clothing items. We are going to use a Deep Learning framework called Keras. Other frameworks we could have chosen include Theano, Lasagne, Blocks, TensorFlow, MXNet, PyTorch and more. Each framework has its pros and cons, you can read more about the other framework options [here](#).

Keras is a very good beginners frame and it is tailored to handle pandas datasets.

Please open the Jupyter notebook titled `CNN.ipynb`.

Please make sure you have read all readings before looking through the practical work.

# Next Week



Next week you are going to receive your 2 week challenge!!!

