Module 7 : Machine Learning

Session 7D : Deeper into Machine Learning : Neural Networks

Dr Daniel Chalk

"You look a bit tensor"

Neural Networks

Neural Networks (or more specifically for our purpose, *Artificial Neural Networks*) are networks of *neurons* (nodes) with *connections* between them, and *signals* being sent along these connections.

Neural Networks mimic the way in which the human brain works, and were developed originally to try to better understand how the brain works.

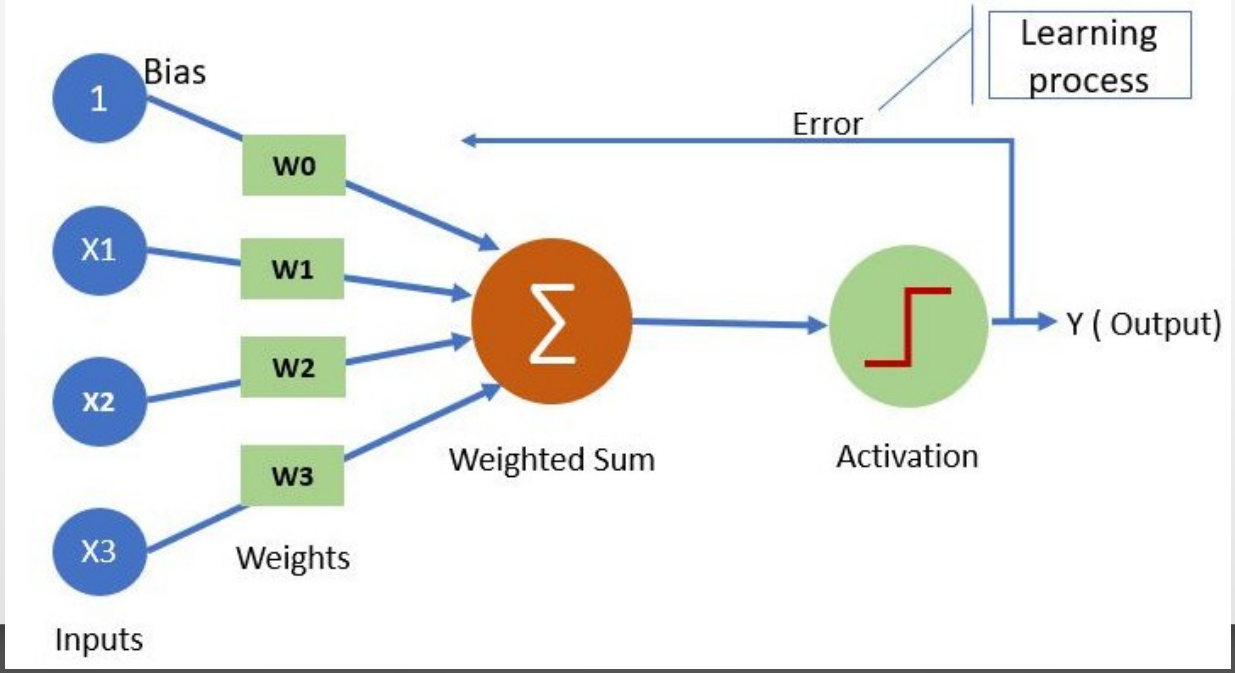
The irony is that whilst Neural Networks have been proven to be *superb* structures for artificially intelligent learning, we don't understand *why* they work…

So let's just pretend it's magic :)

The Neuron

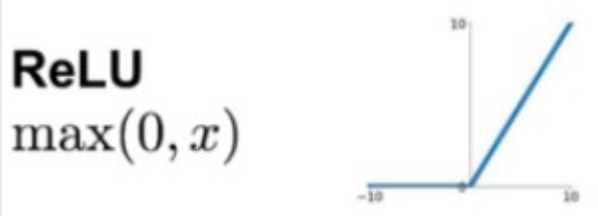
Neural Networks are made up of Neurons (originally named *Perceptrons*). Each Neuron performs a very simple task – to take all of its inputs (each multiplied by a *weight* representing the strength of a connection), add them all up, and spit them out according to an *Activation Function.*

The Activation Function takes the raw weighted sum, and converts it in some way.

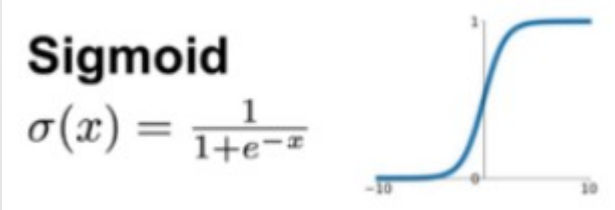


Activation

There are lots of different activation functions (which you'll see in Mike's section shortly). Two of the most common that you will encounter are :

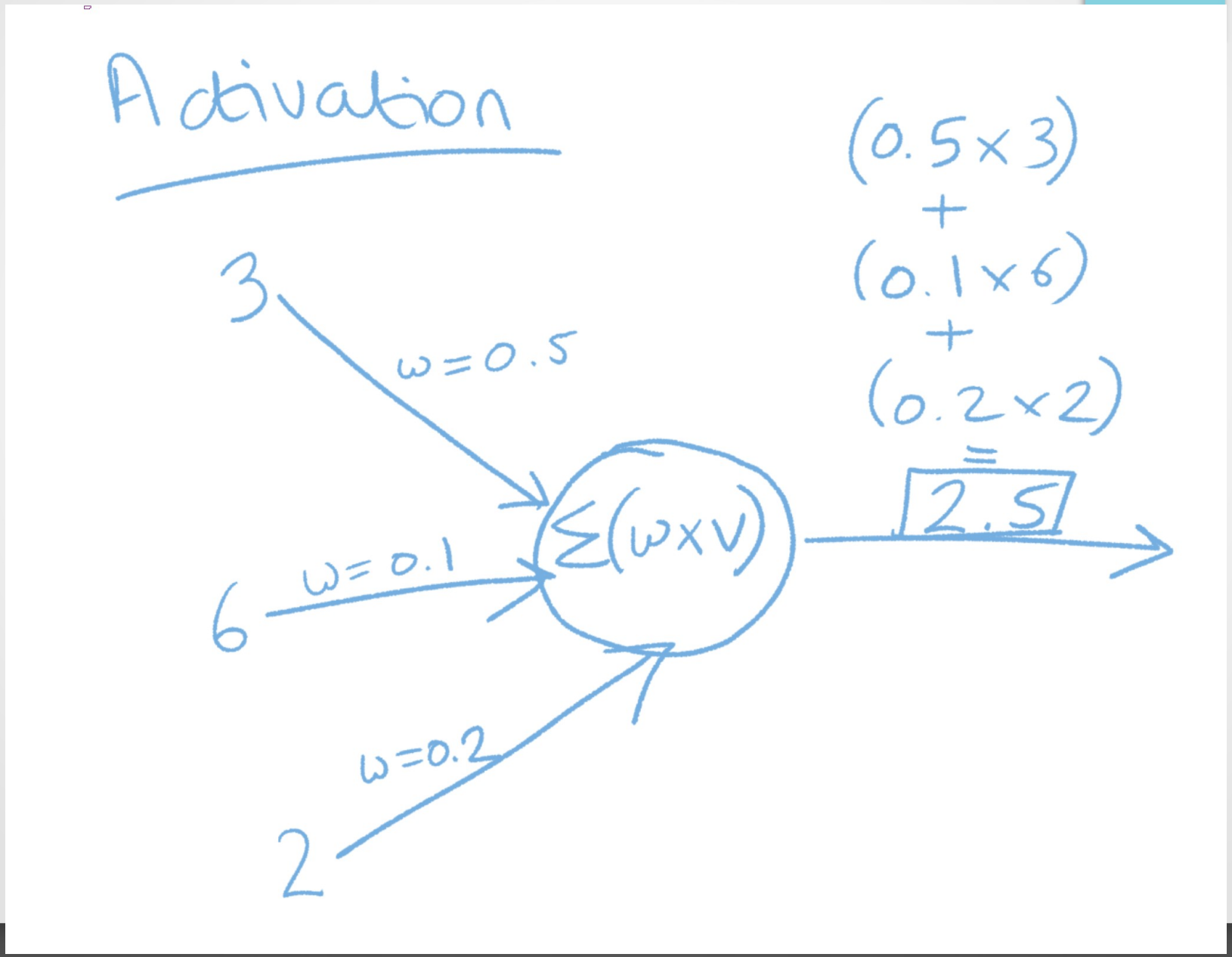


ReLU (Rectifying Linear Unit) – keep positive values unchanged, and convert all negative values to 0. Allows for modelling of non-linear functions and is widely used.

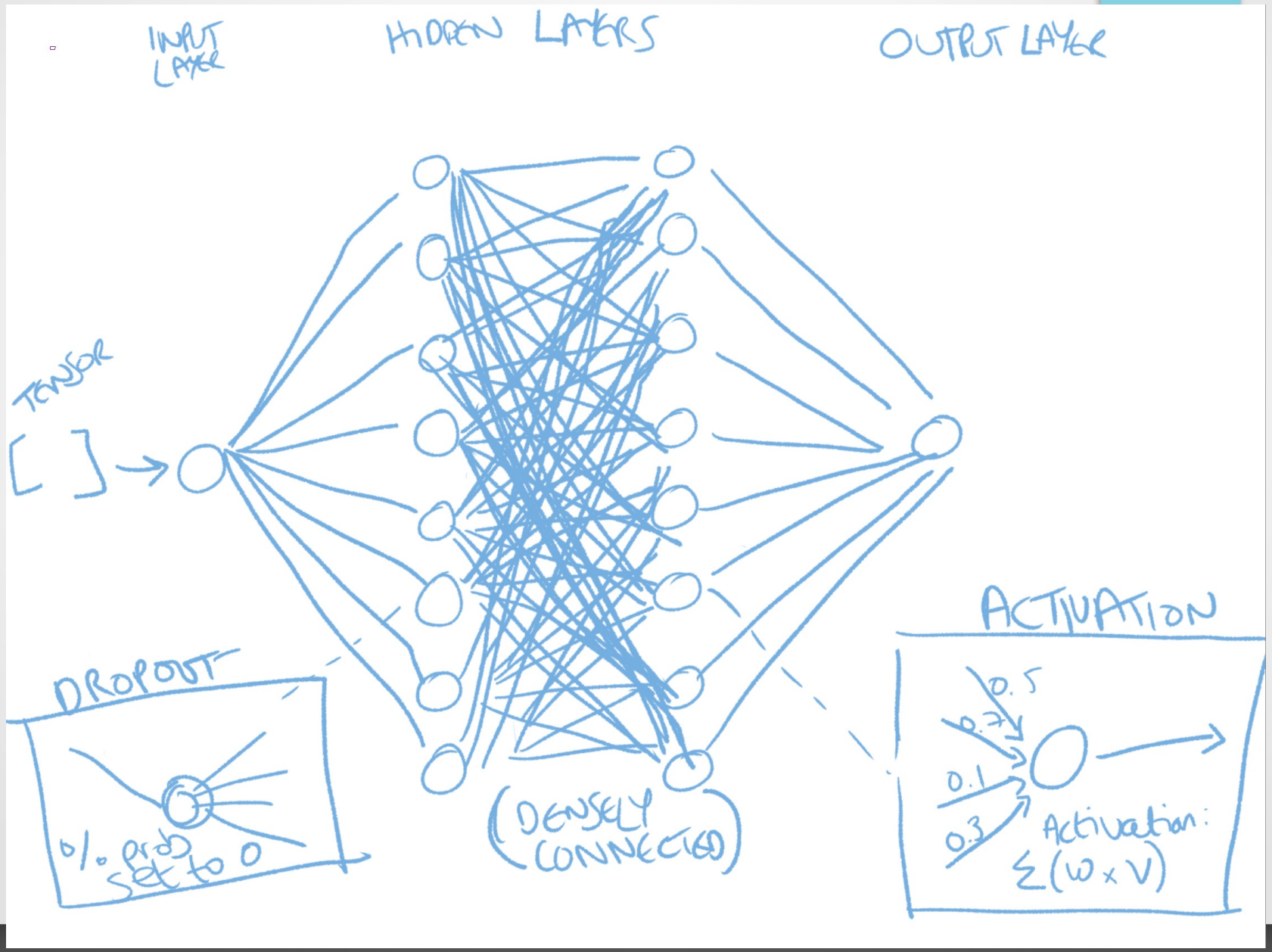


Sigmoid – scale output to between 0 and 1 using logistic function (as we saw in last session). Good for producing probabilities, and therefore often used as final output neuron in classification networks.

Activation



Neural Network



Tensors

Whilst we're using simple numbers here to describe how a neural network works, in reality the data being passed through a neural network is a bit more complex.

The numbers that we feed into a neural network need to be shaped as *Tensors*. A Tensor is quite a complex concept to understand, but a good way to think of a tensor is as a generalisation / abstraction of scalars and vectors (which are themselves Tensors).

A scalar represents a quantity that has magnitude but no direction. A vector has both scale and direction. Both of these are Tensors, but Tensors can also go into multi-dimensional space.

But don't worry about the complexities of this. All you need to know is that a Tensor is the shape that your numerical data needs to be in order to be pushed through a Neural Network. The packages you will use have methods to get your data into the right shape.

Loss and Backpropogation

The magic of a Neural Network lies in how they can learn by adapting the weights of their connections over time.

They do this using a process known as *backpropogation.* Backpropogation distributes the error (known as the *loss* – how far off the NN was in terms of the output it predicted vs the true output) across the network. In other words, it works out which connections / weights were most to blame for the error, and adjusts them accordingly.

There are various ways of measuring loss and also of backpropogating. You'll see some of these in Mike's section.

TensorFlow and Keras

Building and using Neural Networks were, until fairly recent times, the domain of expert mathematicians and Computer Scientists.

But in recent years, excellent packages have emerged that allow many more people to be able to (relatively) easily build their own Neural Networks, by hiding much of the “pure mathematical” complexity.

Two of the most well-established packages are *PyTorch* and *TensorFlow (developed by Google)*. In this session, you will have practice using TensorFlow using the *Keras* API, which sits on top of TensorFlow and makes things a bit more user-friendly.

Mike's Lecture

You're now going to watch a pre-recorded lecture from my colleague Mike Allen. The lecture will talk through the concepts I've just covered in more detail, before walking you through an example of how to build a simple Neural Network for the Titanic data, that we introduced in the last session.

The lecture is 95 minutes, and you should follow along using the provided notebook (HSMA_neural_nets.ipynb). If you don't have TensorFlow locally installed, you can open the notebook in CoLab instead. Simply click the "Open in Colab" button on Mike's GitHub repository here : https://github.com/MichaelAllen1966/2004_titanic

You should take a 10 minute break somewhere during the lecture, so we will resume in 1 hour 45 minutes. When we come back, you will work in your groups on an exercise to test out your new skills!

The lecture is here : https://bit.ly/3bI98EE

Exercise 1

You will now work in your groups to build a Neural Network that attempts to predict whether a patient receives clot busting treatment for stroke, using the same data that you used when attempting this with a Logistic Regression model in the previous session (using the notebook HSMA_exercise.ipynb).

You have until the end of the session. How well can you get your model to predict? Can you improve your accuracy by avoiding or reducing over-fitting?