Neural Net and deep learning

# Theory

* Needs a virtual Linux machine installed as the libraries can’t function on a windows machine.
* These are the first parts of reinforcement learning.
* Neural networks are the stepping stones to reinforcement learning.
* Neural networks are designed like the biological neural networks in humans with the goal of making computers learn like humans do.
* Use cases include pattern recognition, control such as in self driving vehicles, signal processing, time series predictions, anomaly de4tection etc.
* Neural networks are also known as Artificial Neural networks or ANN
* Neural networks attempt to solve with computers, problems that are easy for humans to solve but difficult for computers to solve.
* The simplest possible neural network is the Perceptron.
* A perceptron has 3 components: input (receives inputs); processes the inputs (by assigning weight to the inputs and summing the inputs); and output.
* To avoid the possibility of having zero values often for the inputs, one can add a bias which starts from 1
* To train the perceptron, there are five typical steps:
* Provide the perceptron with inputs for which there is a known answer
* Ask it to guess
* Computer the error
* Adjust all the weights with reference to the error
* Repeat the process
* You build a neural net by linking multiple perceptron together in layers.
* You keep training process until you reach a preset acceptable error rate
* Any layers between the input and output layers are known as hidden layers because you don’t directly see those
* Deep learning is simply a neural net with a lot of hidden layers.
* The library used for neural net in this course is the google Tensor Flow library which needs a virtual Ubuntu box to run on windows.

# What is Tensor Flow

* Tensor Flow is the most popular library for deep learning, open sourced, developed by google and runs faster on GPU machines than CPUs.
* The basic idea of Tensor Flow is to be able to create data flow graphs which have edges and nodes.
* The data passes (flows) from one layer of nodes (Tensor) to another Tensor.
* There are two ways to use tensor flow: Customizable Graph Session (which is very customizable but could be tedious with a poor background in the mathematics necessary) and scikit type interface with Contrib.Learn
* Documentation is at [www.tensorflow.org](http://www.tensorflow.org)

# Installing Tensor Flow

* Pip install tensorflow==version

# Tensor flow basics

* Import tensor flow as tf
* Tensor flow stores constants as a tensor object. To create one for instance, you can use a = tf.constant (‘hello world’) or b = tf.constant (100) to create string and integer constants respectively where a and b are the object names.
* You can check the shape, dtype etc. by simply calling the variable name.
* You can check to confirm it’s a tensor object by calling type of the variable name e.g., type (a) where a is the variable name.
* A tensor flow session is a class for running tensor flow operations. It can be created with c = tf.Session() where c is the instance name.
* Tensor objects are evaluated in operations and operations use sessions as running environments.
* You can perform typical math operations such as addition. Subtraction, multiplication, division etc. on a tensor object e.g., a + a, a \* 5 etc where a is a t6ensor object.
* You can transpose a tensor object that is an array (matrix) with tf.transpose(a) where a is the variable name of the tensor matrix
* You can concatenate matrices along any axis e.g., tf.concat ([a, b, c], axis = 0 or 1) where a, b and c are the variable names of the matrices to be concatenated. Note that all the tensors must be of same dimensions for it to work.
* You can sum all the elemnts of a tensor up e.g., tf.reduce\_sum (a)
* You can convert any value or array to a tensor with tf.convert\_to\_tensor([d, f, j, k]) where d, f, j, k are elements.