**Deep Learning Concepts and Facts to remember from Lesson 2 and researches**

**(For your Pytorch DL - Stock knowledge @roBot):**

1. **Deep Learning** is a subset of Machine Learning which allows computational models that are composed of multiple processinglayers to learn representations of data with multiple levels of abstraction. It is used for applications such as the following:
   * beat humans in games
   * detect spam e-mails
   * forecast stock prices
   * recognize images in a picture
   * diagnose illnesses more
   * self-driving cars
2. **Pytorch** is a Python based package for serving as a replacement of Numpy and to provide flexibility as a Deep Learning Development Platform
   * **Getting started (Suggested tools to utilize):**
     + Install Anaconda, Python, Jupyter, Spyder

<https://medium.com/@Shreedharvellay/anaconda-jupyter-spyder-things-you-need-to-know-3c808d824739>

* + - Install Git on laptop
    - Create Github account
    - Google Colab GPU
    - TensorFlow with GPU
    - Pytorch (<https://pytorch.org/>)

1. **Neural Networks** function as the heart of Deep Learning
   * It interlinks Perceptrons by taking the output from one and tuning it into the output for another one.
   * In my own words, it is a computing system made up of a number of simple interconnected processing elements which then process information by their dynamic state response to external inputs
2. **Perceptron** is the main building block of Neural Networks
   * **Types of perceptron as logical operator:**

* AND
* OR
* NOT
* XOR

1. **Difference between the output of AND & OR Perceptron**
   * At OR Perception, the line is shifted down
2. **Two ways to go from an AND perceptron to an OR perceptron:**

* Increase the weights
* Decrease the magnitude of the bias

1. **Classification** is the task of identifying to which of a set of categories (sub-populations) a new observation belongs. It is decided on the basis of a training set of data containing observations (or instances) whose category membership is known.
   * **Examples of Classification**
     + Classifying e-mails as spam or not
     + Classifying flowers of a particular species (e.g. Iris dataset)
     + Classifying a credit card transaction as fraudulent or not
     + Face recognition
   * Remember the concept that any misclassified points want the line to move closer to them
2. **Parameters of Perceptron Algorithm**

* number of epochs
* learning rate
* randomized initial parameters

1. **Conditions which should be met in order to apply gradient descent:**

* The error function should be differentiable
* The error function should be continuous

1. **Continuous error functions are better than discrete error functions when it comes to optimizing**
2. **Activation(Transfer)function -** is the node that you add to the output end of any neural network or in between two neural networks for Neural Networks

* **Activation function for Neural Networks:**
  + Softmax
  + Sigmoid
  + ReLU
  + Leaky
  + tanh

**External reference (One page Cheat sheet reading for their definition and example):** <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

1. **exp** is a function that turns every number into a positive number
2. **For a very high value for P(all), such Probability Model classifies most pots correctly with P(all) which indicates the degree of its accuracy**
3. **log** is a function that turns product into sums
4. **Cross-Entropy** is a connection between probabilities and error functions. Its concept is popular in Machine Learning.
   * A higher (Multi-Class) Cross-entropy implies a lower probability of an event
5. **Logistic Regression** is the most popular and useful algorithms in Machine Learning. It is also the building block of Deep learning
   * **Logic Regression Algorithm process**:
     + Take the datasets
     + Select a random model
     + Calculate the error
     + Minimize the error
     + Obtain a better model as necessary

1. **Gradient Descent** is a first-order iterative optimization algorithm for finding the minimum of a function.

**External Reference (One page Cheat sheet reading for it’s definition and example):** <https://towardsdatascience.com/gradient-descent-simply-explained-1d2baa65c757>

1. There are 26 nodes required in the output layer if you try to classify all letters in the English Alphabet.
   * We can also utilize 52 nodes (26 uppercase and 26 lowercase letters)
2. **Feedforward** is the process which neural networks use to turn the input into an output
3. **Backpropagation** is one method which can be used to train a Neural Network

* **Backpropagation process:**
  + Do a feed forward operation
  + Compare the output of the model with the desired output
  + Calculate the error
  + Run the feedforward operation backwards to spread the error to each weight
  + Use this to update the weights and get a better model
  + Continue this until we have a good model

1. **High Level summary of steps in Training the Neural Network**
   * One-hot encoding the data
   * Scaling the data
   * Writing the backpropagation step
2. **High Level summary of steps in Predicting with Neural Networks:**
   * Load the data
     + Format the loaded data using Pandas and Numpy
     + <https://pandas.pydata.org/pandas-docs/stable/>
     + <https://docs.scipy.org/>

* Plot the data
* One-hat encode the rank (Use get\_dummies in Pandas)
* Scale the data
* Scale the data into Training and Testing
* Split the data into features and targets (labels)
* Train the 2-Layer Neural Network (Use sigmoid)
* Backpropagate the error
* Calculate the accuracy on the Test Data

1. **Overfitting** occurs when one part of Neural Network has very large weight and dominate all the training while another part of the network do not get much role.
2. **Regularization** is any modification we make to the learning algorithm that is intended to reduce the generalization error, but not its training error. It is the subtle way of preventing overfitting in a model.
   * **L1 Regularization** is used if we go for |Absolute values| (Large features selections)
   * **L2 Regularization** is used if we go for Squares (Maintains all small weights)
3. **Dropout** is another way to prevent overfitting. It turns a portion of the Neural Network off and let the rest of the network train.