Part 1 Neural Networks for Classification

1. The network architecture :.

- **Number of Input Nodes** = 4 one for each feature of the data set Petal Width, Petal Length , Sepal Width & Sepal Length
- **Number of Output Nodes** = 1 one node which produced a continuous output between 0 & 2. Another choice was to use 3 nodes in the output layer, the result would have been a vector of length 3. The table below shows the output mapping to the classes in both cases.

Class	1 Node in output layer	3 Nodes in Output Layer
Iris-setosa	~0	[0,0,~1]
Iris-versicolor	~1	[0,~1,0]
Iris-virginica	~2	[0,0,~1]

- **Under 1 Hidden Layer Assumption**, the below table will show the accuracy percentage change as I increased the number of nodes in the hidden layer. After training 500 Epochs, I kept always a track of the weight saved to disk at which epoch.

Number of nodes	1	2	3	4	10	50	500	1000
Test Set Accuracy %	92-94	97-98.6	97-98.6	97-98.6	97-98.6	97-98.6	93-96	33-96

- From running lots of iterations I concluded that learning for a long time leads to over fitting and losing accuracy on the test set while lowering the error of the learnt model.
- **While Changing the number of hidden layers** while keeping hidden nodes to 5, I concluded after more than 5 hidden layers the network cannot even match the KNN accuracy rates as per table shown below.

Hidden Layers	1	2	4	6	50	500
Test Set Accuracy%	97-98.6	97-98.6	97-98.6	97-98.6	33	33

- So according to above evidence , and the processing time associated with learning I decided to go with two options either a super simple efficient network **1 hidden layer with 2 hidden nodes**, or a more complicated network of **4 hidden layers and 5 nodes** in each, just in case there are more patterns to be learnt the network would have the capacity to reflect that in the extra weights made available by a slightly more complicated architecture.

2. The learning Parameters:

- **Learning rate** was set to **0.1**, and momentum at **0.5** , as decreasing either prolong the learning process while increasing either would lead to being stuck in a local optimum bouncing between larger steps rather than slowly making small steps towards convergence.
- The table below will show the effect of learning rate change while keeping momentum at 0.5

Learning Rate	0.1	0.2	0.4	0.7	1
Test Set Accuracy%	97-98.6	97-98.6	97-98.6	33-94	33

- While setting the momentum to 0 and changing the learning rate it only affected the speed at which we reach 98.6% accuracy even at a learning rate of 1.

3. Termination Criteria:.

- I simply opted to record the best accuracy while training the network as well as graphing the error of the test data set while training almost using it as a validation technique as method to stop early before overfitting takes place. I know this is kinda cheating but the test set is too small to divide into a test set and a validation set, given a bigger data set I would have split it into a validation set and a test set to isolate the test set completely from the learning

process. I would have loved to implement other regularization techniques in literature such as L1 & L2 regularization and drop out, but I lack knowledge and time.

- Another Termination Criteria was to train the network for 200 epochs as most of the results were good enough at the stage.
- Error rates reported from most iterations yielded good accuracy for the test set at a range between 0.09-0.14 I could have terminated at 0.1 threshold of error.

4. Results from Training Network.

- I found that teaching the network for 100 epochs @ Learning rate 0.1 and momentum of 0 with 1 hidden layer and 5 hidden nodes for 10 times yielded accuracy of 98.6% for the test set. The 10 runs are saved as PNGs @ full resolution attached to the report, and below are the thumbnails in a separate page.

5. Performance Comparison between KNN & Neural Network

- Heavily over-fitted due to prolonged learning process of most NN architecture discussed above mimics the KNN algorithm @k=3. The KNN method @k3 yielded accuracy of 94.666% so did Neural Nets that was learning for over 400 epochs.
- Best Performance with the KNN method was 97.3% @K=11 , where an efficient architecture with a reasonable learning epochs the best performance for a Neural Network was 98.66667%.
- Neural Networks is superior method to KNN for the reasons below:.
    - Computation time is insignificant after the model has been optimized, unlike KNN where each test instance needs to be compared to all other instances regardless how many numbers of tests have been computed.
    - The larger the data set the better performance is expected from neural networks as it would map the patterns in the data set in the weightings of the layers and their corresponding nodes. Where as the effect of larger data sets on KNN highly affects the its computation time.
    - Transfer learning is applicable with Neural Networks, i.e we could deploy the network in this assignment to identify other plant species with slight modifications to the model given the features are similar.
    - The curse of dimensionality applied to KNN, is absent in Neural Networks as the higher the dimensions the more row are added to the input layer and we can adjust the architecture of Neural Network to deal with the dimensionality of the data set at hand.