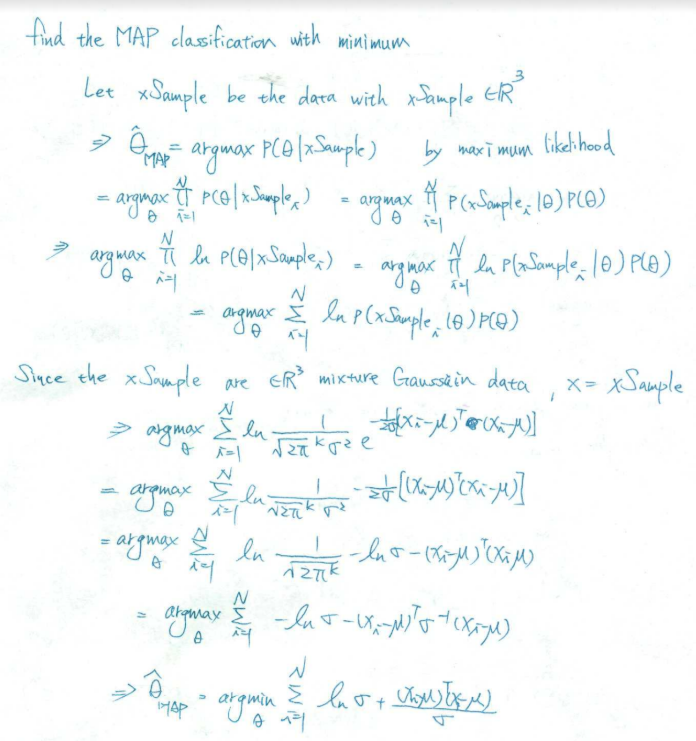
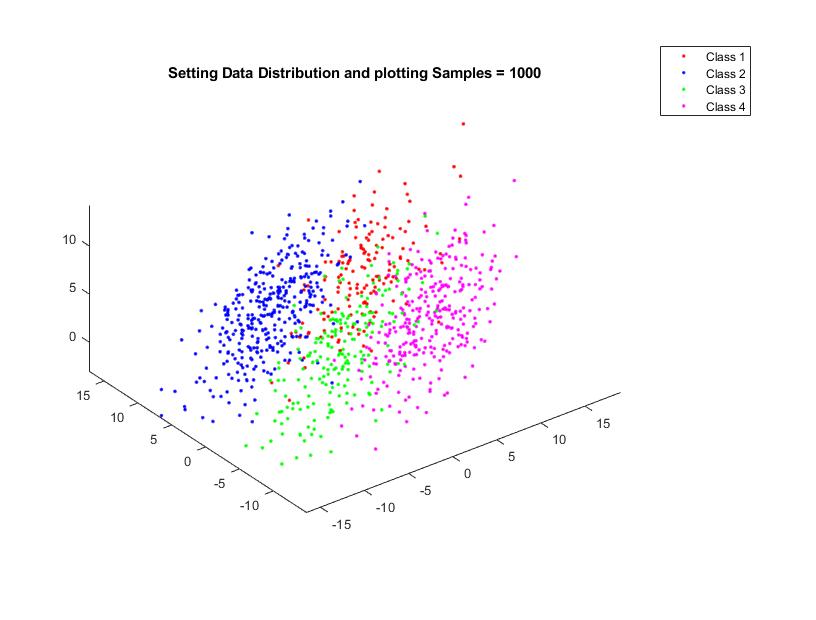
1. Question

Theory and math calculation of MAP classfication





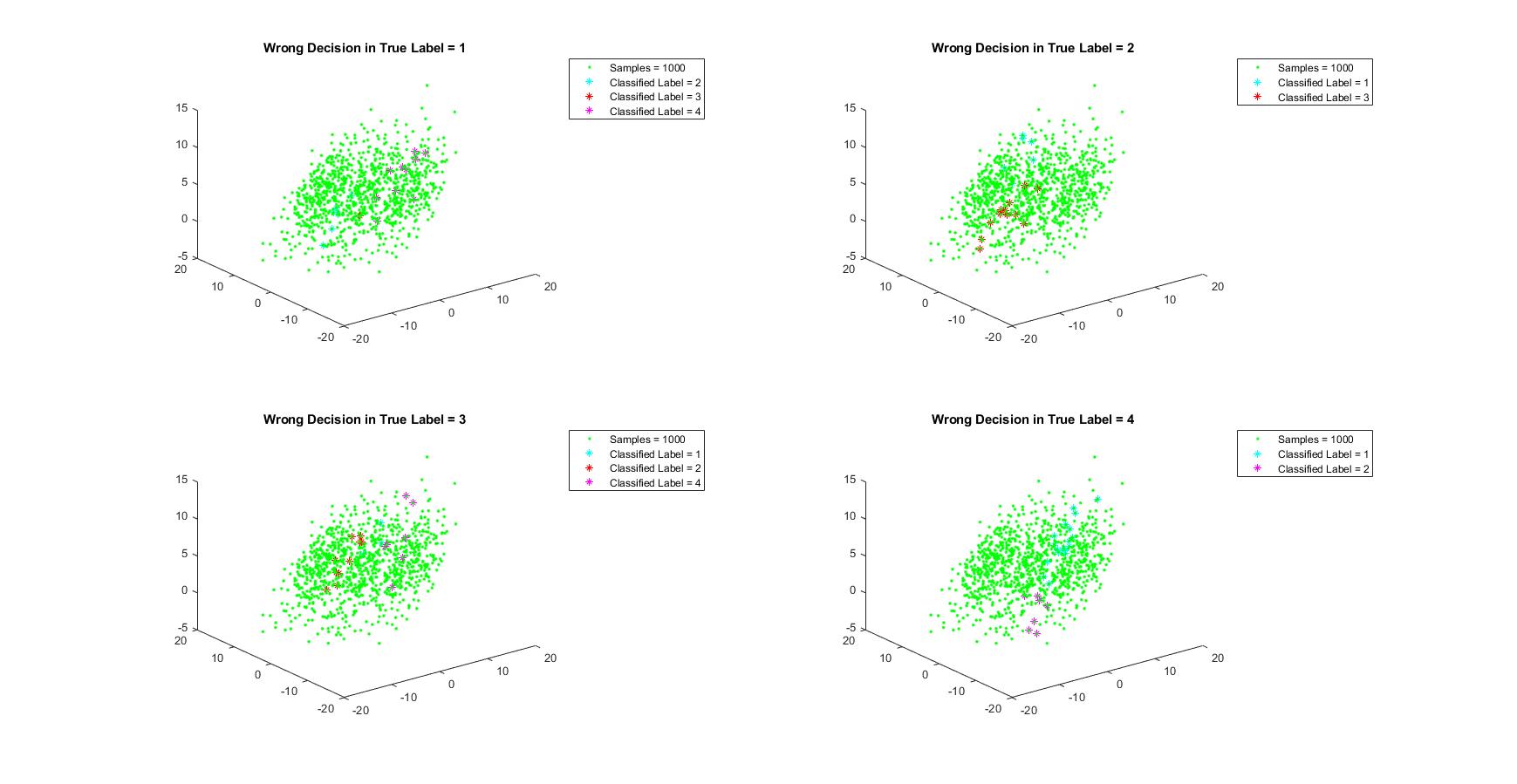
The distribution data is generated with

Prior = [0.15,0.35,0.2,0.3] where true sample number = [154,351,197,298]

Sigma = 0.8\*[5 1 2;1 5 0;2 0 3] /sqrt(2)\* covarianceVector for each class

covarianceVector = [1.3^2 0 0;0 1.2^2 0;0 0 1.4^2]

Apply this data distribution to the MAP classifier:



These plots show the MAP classification result

We get the confusion matrix = [137,6,1,10;6,333,12,0;3,10,177,7;20,0,7,271]

Where represent the right classification of

Class 1 = 137

Class 2 = 333

Class 3 = 177

Class 4 = 271 in 1000 true samples

Thus, we can calculate the whole MAP classification accuracy rate = (137+333+177+271) /1000 = 91.8%

And we get the decision error of each class is

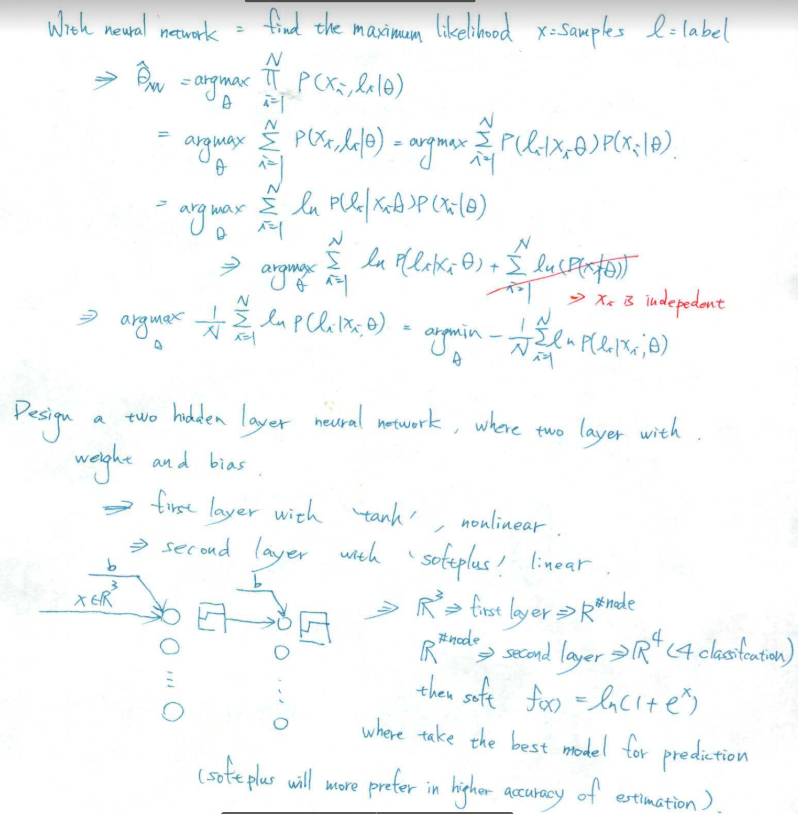
Class 1 error = 0.11

Class 2 error = 0.05

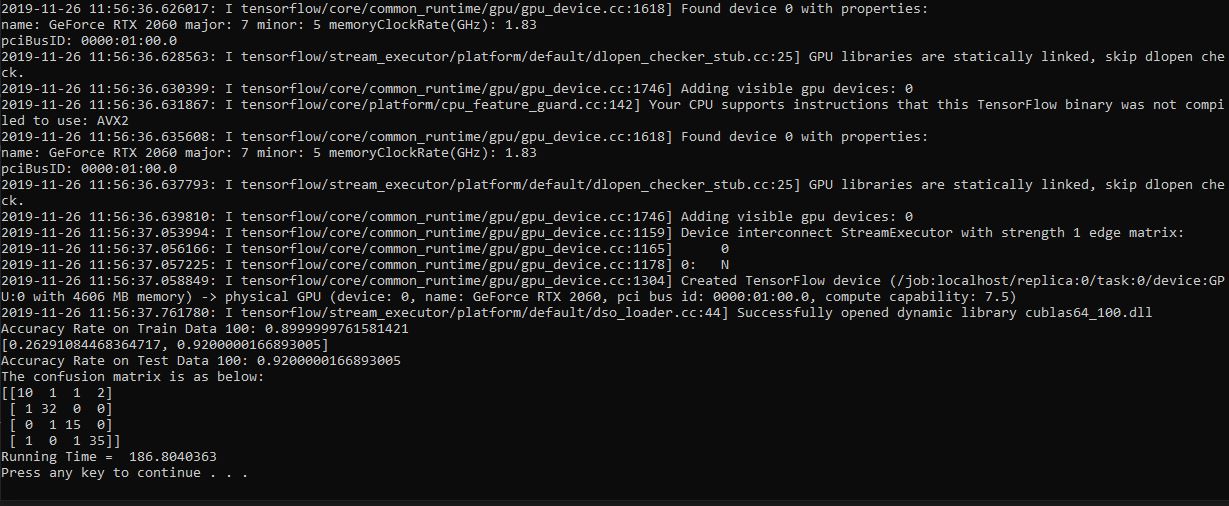
Class 3 error = 0.10

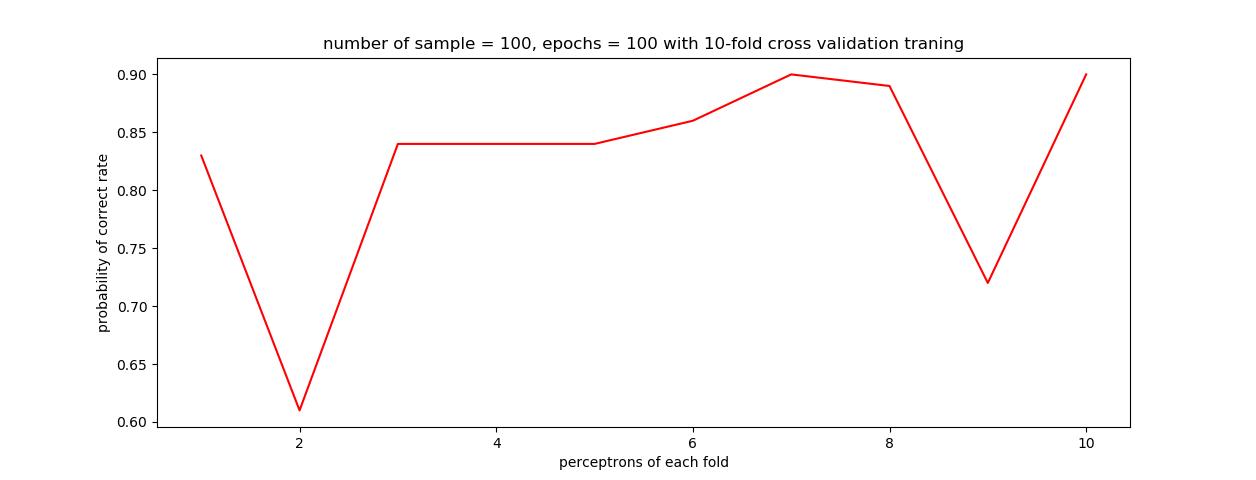
Class 4 error = 0.09

Theory and math calculation of neural network:



* 1. Training 100 data for getting the model in Keras library in python





Training the 100 samples data with neural network with 2 layers, with 1st layer setting the ‘tanh’ activation function, and the 2nd layer setting the ‘softplus’ activation function.

After the training, the best node, bias and weight will pass to the model of Keras object.

Let this object test the test data of 100 samples.

The result shows the confusion matrix, which allow us to find the accurate estimation

In 100 testing sample and 100 10-fold training.

Correct in label 1 = 10

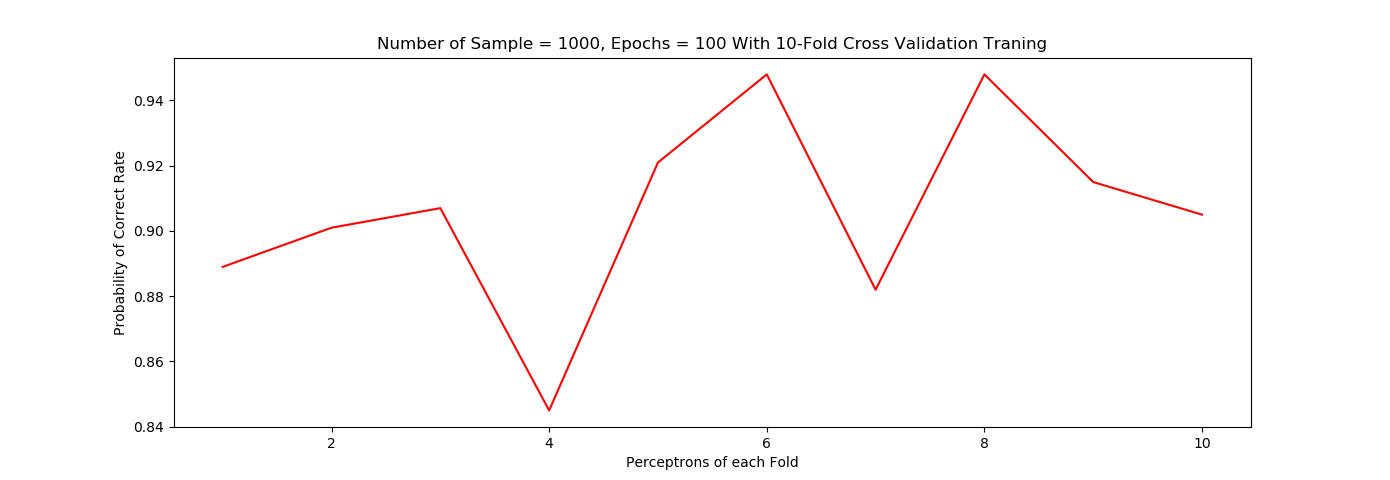
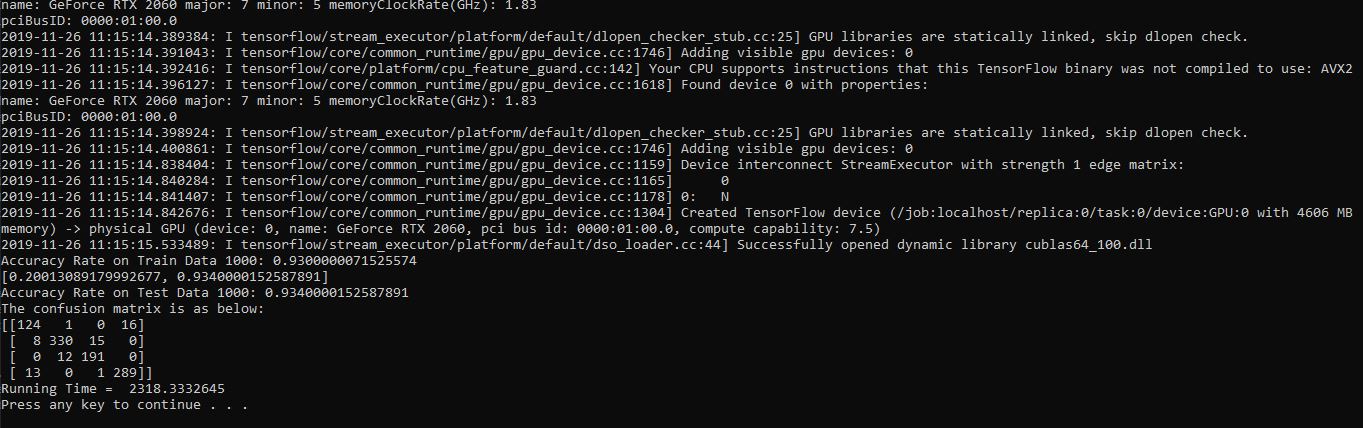
Correct in label 2 = 32

Correct in label 3 = 15

Correct in label 4 = 35 Accuracy = (10+32+15+35)/100 = 0.92

In this case, we can observe that there might have bad tanning case beget to low probability of accuracy rate.

* 1. Training 1000 data for getting the model in Keras library in python



Same as the former training in 100 samples. We take the first layer with activation function ‘tanh’, and the second layer with activation function ‘softplus’.

After getting the best model in the training, fit the model to test samples.

The result shows the confusion matrix, which allow us to find the accurate estimation

In 1000 testing sample and 1000 10-fold training.

Correct in label 1 = 124

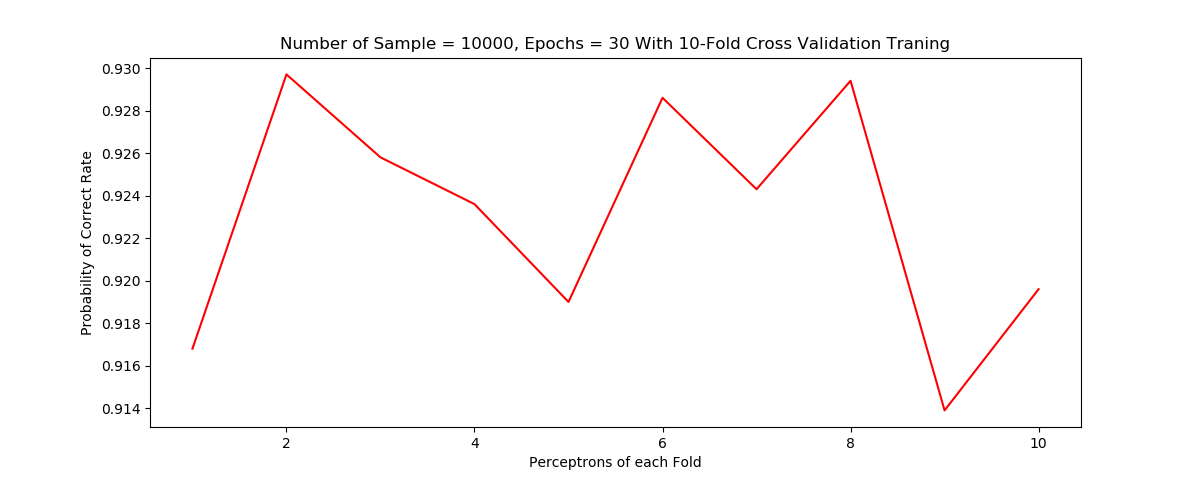
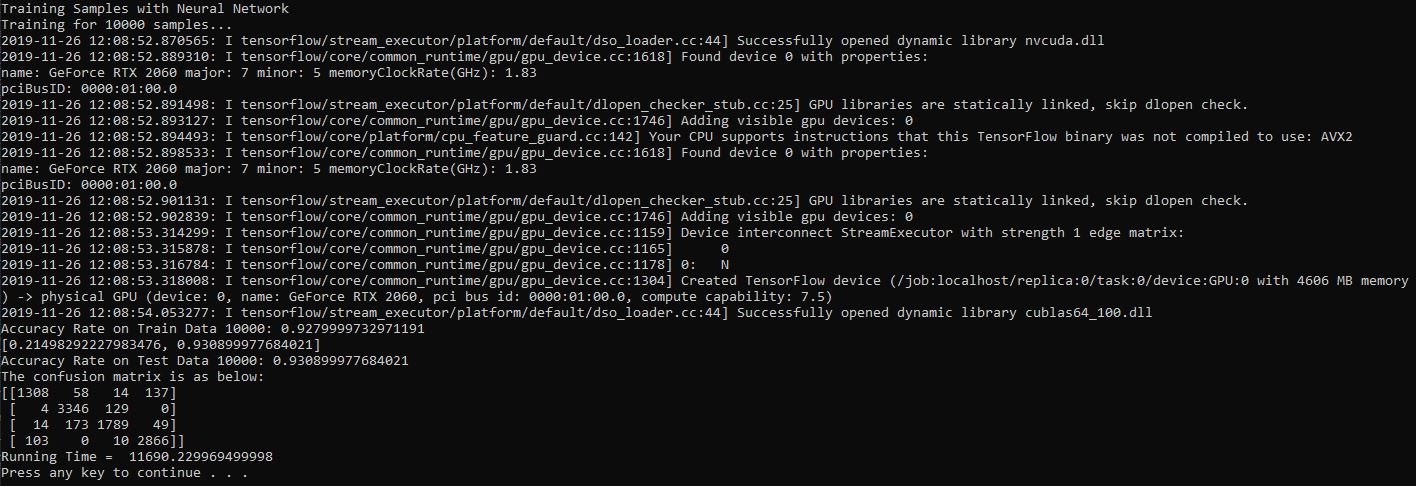
Correct in label 2 = 330

Correct in label 3 = 191

Correct in label 4 = 289 Accuracy = (124+330+191+289)/1000 = 0.934

Same here we could observe that some bas case might be caught for training, but the accuracy rate is better than 100 sample case. Which means the probability of pick bad training module would be lower.

* 1. Training 1000 data with epoch 30 for getting the model in Keras library in python



Same method as former two experiment

We get the confusion matrix that we could get the accurate rate and the wrong estimation number

Correct in label 1 = 1308

Correct in label 2 = 3346

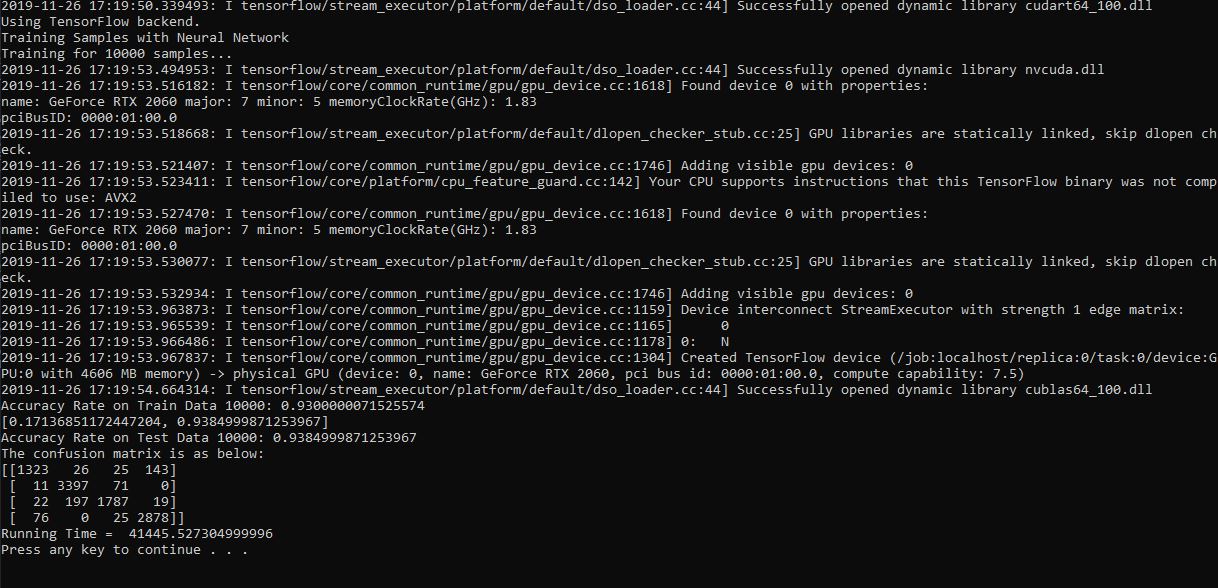
Correct in label 3 = 1789

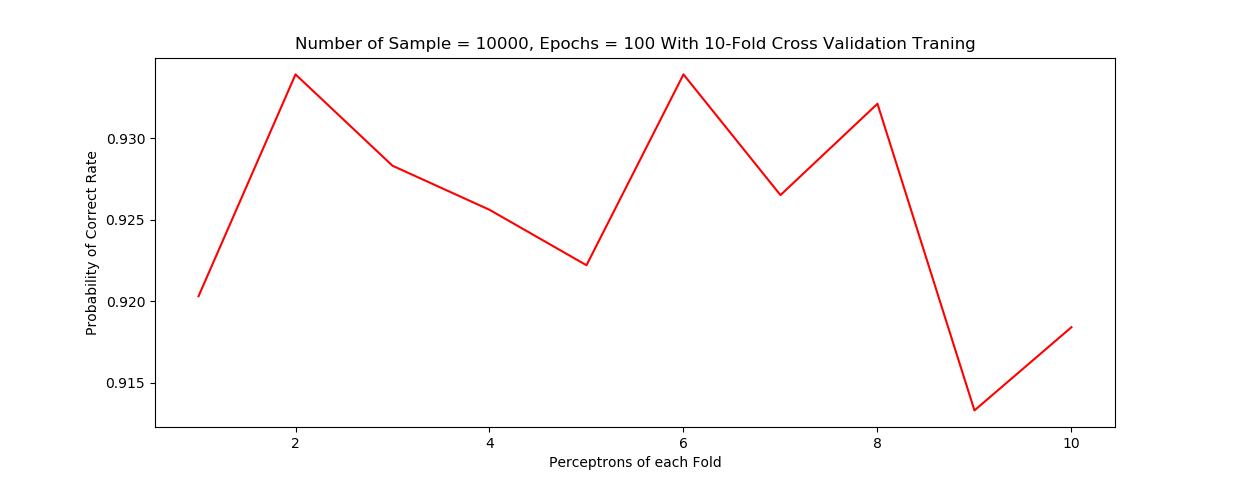
Correct in label 4 = 2866 Accuracy = 9309/10000 = 0.9309

This result does not show a better accuracy than previous two experiment since the epoch = 30 which is lower than 100

Thus, take epochs = 100 in the following experiment

* 1. Training 1000 data with epoch 100 for getting the model in Keras library in python





Here I tried the different if it can get a better result for accurate rate

Increase the epoch from 30 to 100.

The accurate rate: 0.9384

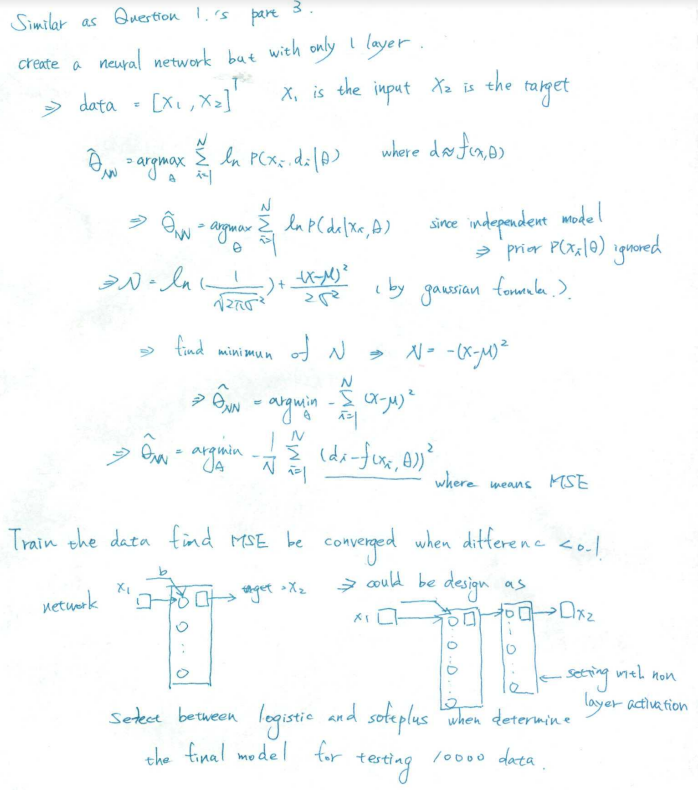
This accurate is a little better than the former 1000 training case

Conclusion of Question 1:

By the previous experiment, NN training is a little better than MAP classification. But it depends on the amount of training sample. As the number of training epochs increase, the accuracy supposed to be increase.

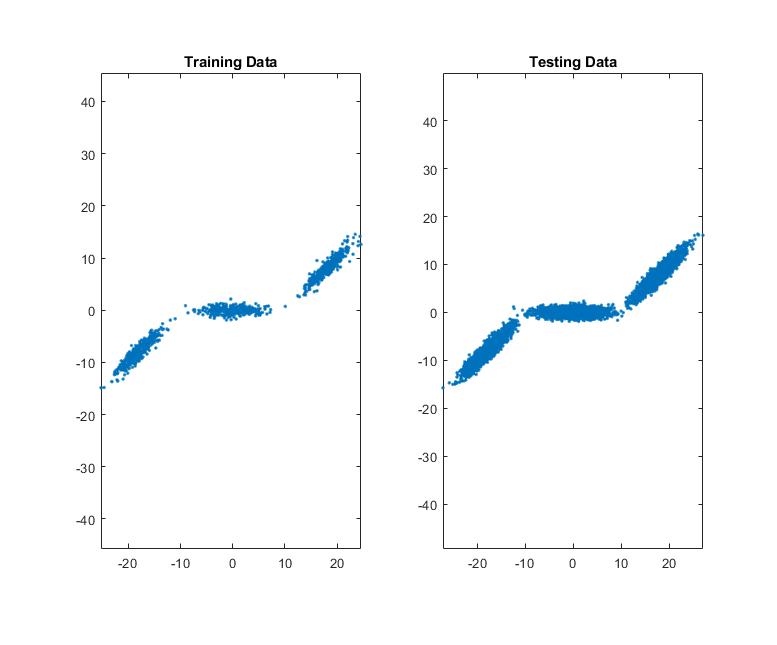
1. Question

Theory and mathematical calculation:



Training and experiment:

The original data of training and testing data



The plots show the original data of training and testing which are 1000 and 10000 samples

Training is base on finding the mean square error of inputs x1. when find a newMSE, compare the newMSE to the old MSE value, if the difference is smaller than a value (define as epsilon), determine as converged.

With the method finding the best argmin value in the mathematical part show as the best model

Separate the training data as 10-Fold to train and validate the model.

Then apply this model to 10000 testing data

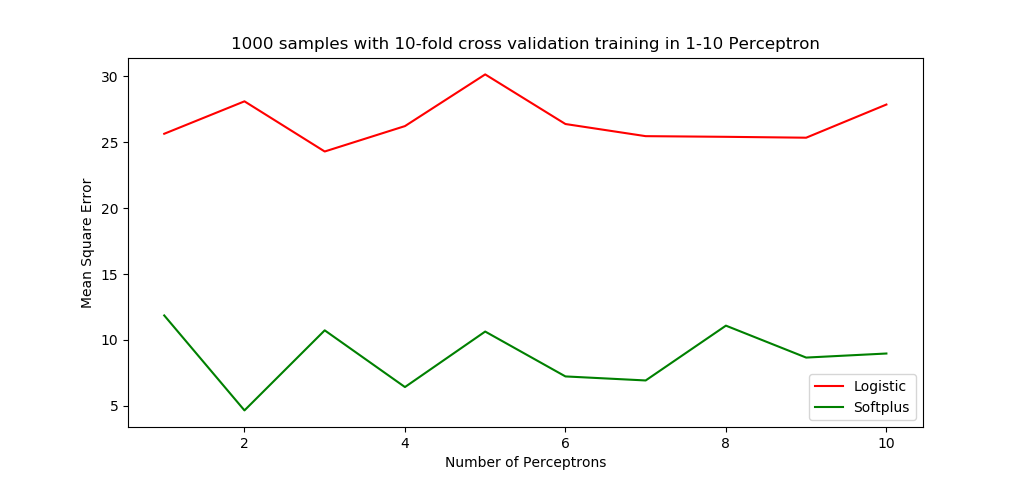
Tried the following epoch to observe that is the final prediction is getting like the original testing data

epochs = 1

epochs = 30

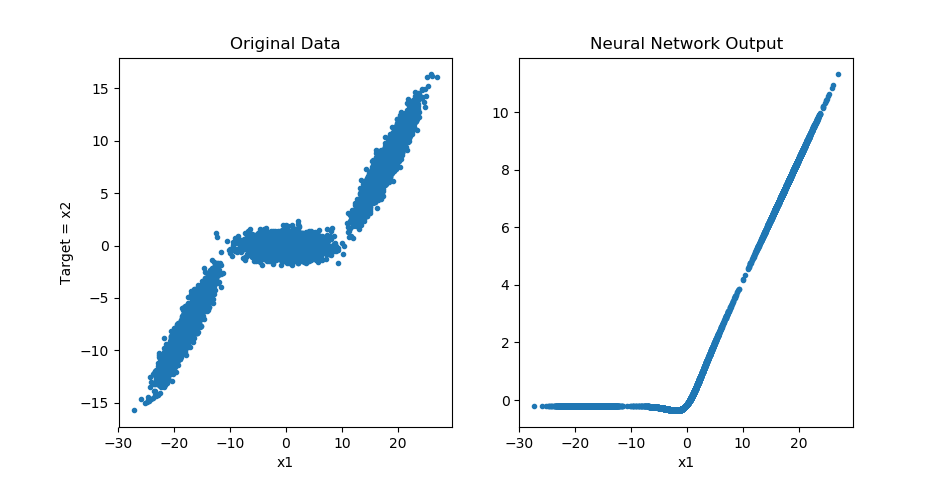
epochs = 50

epochs = 100

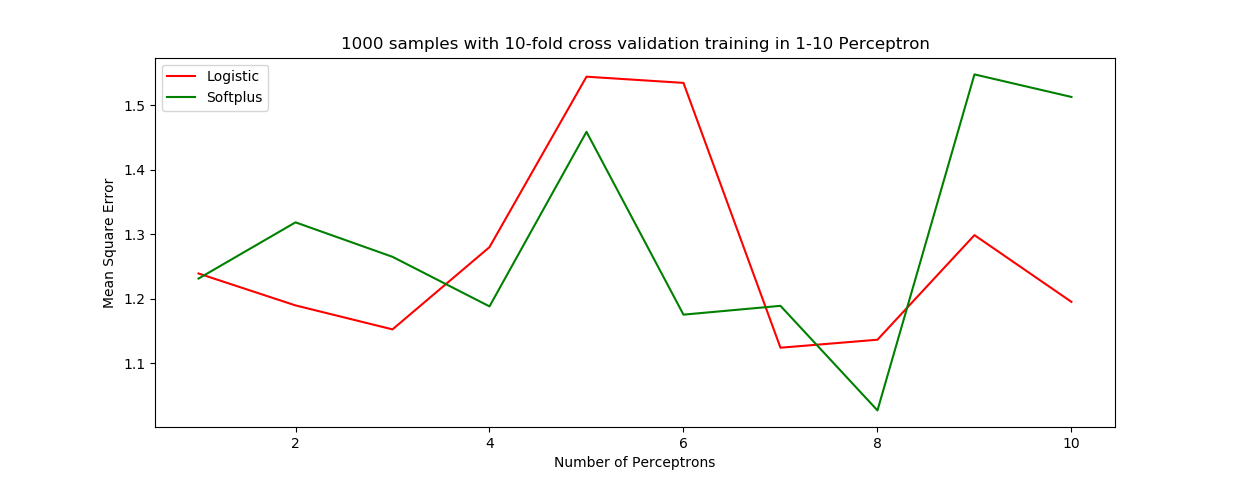


This result shows the NN training only with epochs = 1, both logistic and softplus have large MSE

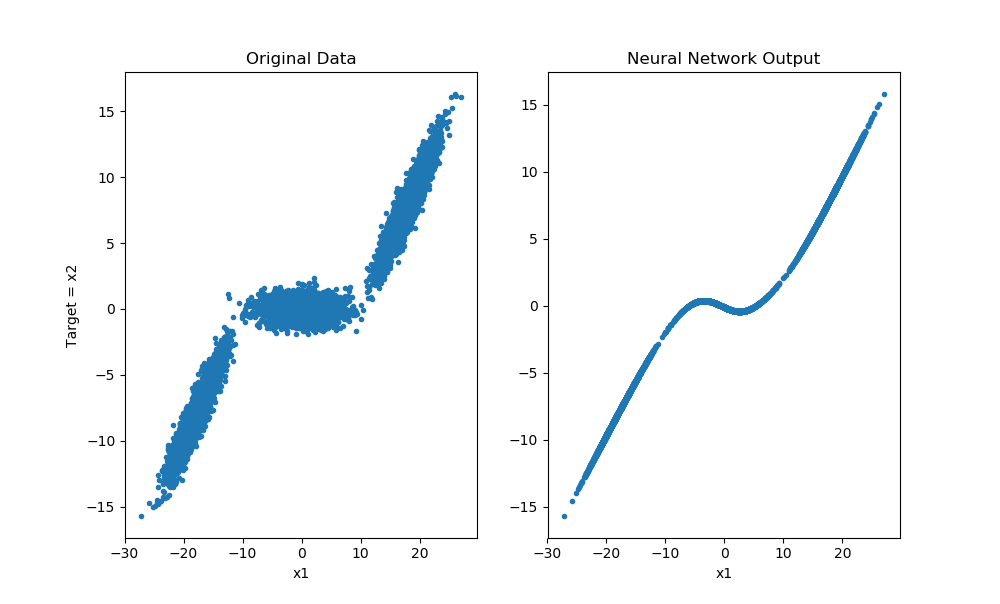
The result could be bad as below

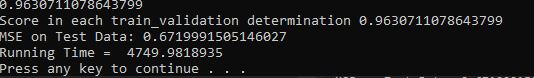


Now we increase the number of epochs to 50 see that if it is better



With 50 Epochs, the MSE become smaller. Thus, we might have better estimation

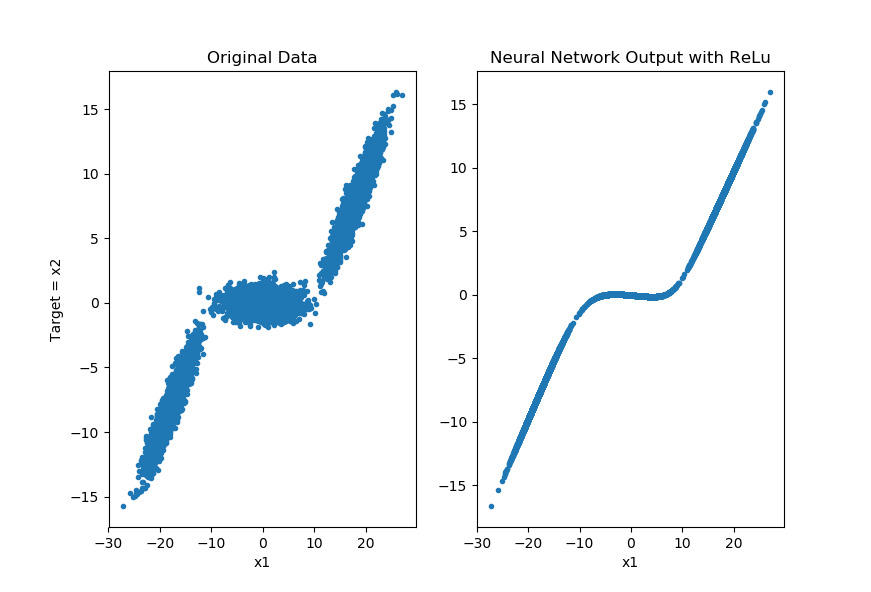




These plot shows better than the former one. But the middle data show two curves (not sharp turn) because the data distribution of Target (x2) is around 0 with Gaussian distribution. The MSE performance = 0.6720

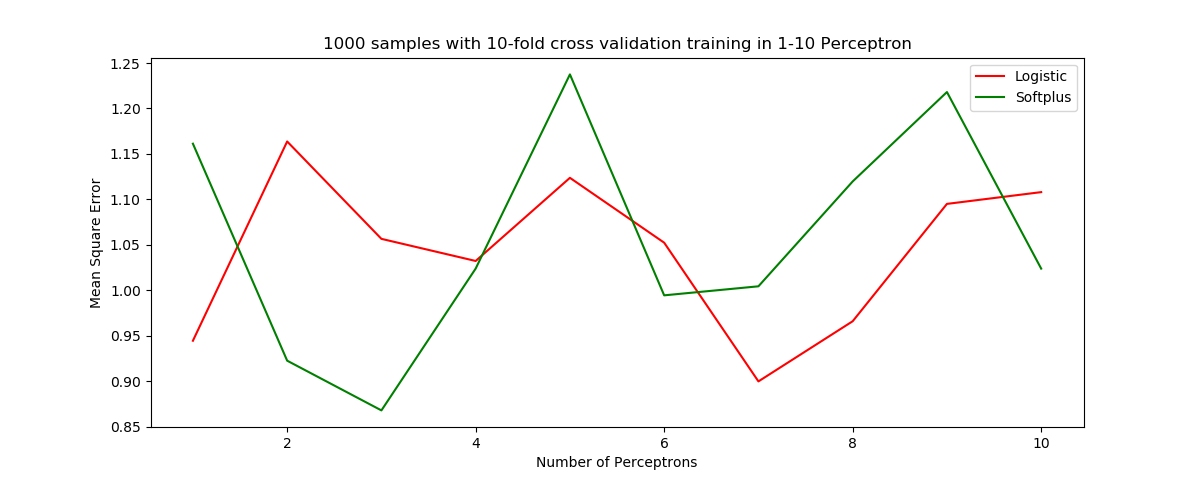
The Neural Network training is suppose to be like linear combination of two activation function. The expect result will be like SmoothReLu.

Thus, try the relu function in Keras to show the case

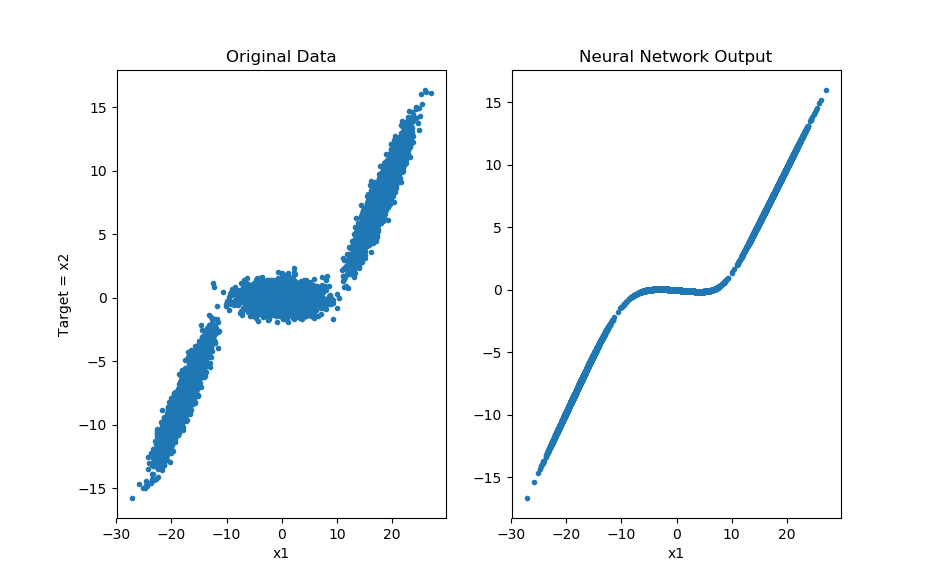


ReLu shows the expected result of Neural Network training result

Now we try to increase the epochs = 100 to see if it is getting similar to ReLu to check the neural network is create correctly

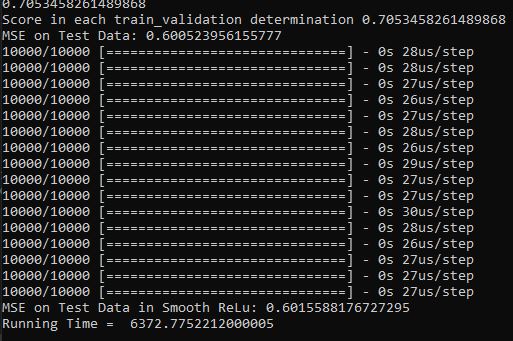


The plot show the logistic and sofplus activation function’s MSE performance in each perceptron



The 100 epochs result seems better than the 50-epochs’ one. This indicate the neural network is created correctly.

Here the performance MSE = 0.6016



Conclusion of Question 2:

The more training epochs, the more similar to the desire result. Also the Mean Square Error will be come smaller.

Reference:

<https://keras.io/backend/>

<https://github.com/keras-team/keras/tree/master/examples>

<https://www.tensorflow.org/>

<https://www.tensorflow.org/tutorials/keras/classification>

<https://scikit-learn.org/stable/model_selection.html#model-selection>

<https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html>

<https://www.mathworks.com/help/thingspeak/create-and-train-a-feedforward-neural-network.html>

<https://www.mathworks.com/help/deeplearning/ref/network.html>

<https://www.mathworks.com/help/deeplearning/ref/mse.html>

<https://www.mathworks.com/help/deeplearning/ug/create-and-train-custom-neural-network-architectures.html>

<https://www.mathworks.com/help/deeplearning/ref/train.html#namevaluepairarguments>

Code Resource:

<https://github.com/MakiseYuki/EECE5644-Machine-Learning/tree/master/Exam%202>