Classification Pipeline design. How it was chosen and why?

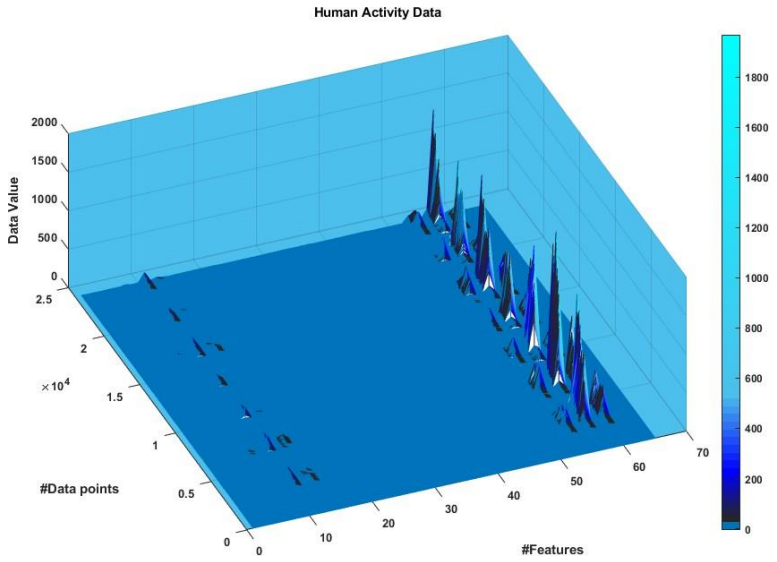

Figure 1 Raw data visualization. Image shows the number of data points, the number of features and the value of each feature of data points

Fig.1 represents the data with axes being the number of data points (24000), the number of features (64) and the value of each data point in the perpendicular axis. From Fig.1 it can be inferred that there is a pattern and that some features have very large values, such feature number 50~60.

The data provided is highly dimensional, therefore Naïve-Bayes classifier and the Multilayer perceptron classifier (MLP) will be considered instead of KNN or Binary classifier, as the former two classifiers are known to be effective with high dimensional data.

Down to two classifiers how to choose one? From Fig.1 it is not clear that the distribution of the data is normally distributed, so gaussian Naïve-Bayes classifier may not work. On the other hand, the MLP does not require to previously know the distribution of the data and with enough neurons, is able to classify non-linearly separable data, like when it acts as a XOR gate.

Therefore, the MLP classifier will be used instead of the Naïve-Bayes. The training method chosen to train MLP is backpropagation. The features will be inputs and as seen in Fig.1 there is a very large range, so for MLP to work with this data, the inputs will have to be normalized. It was also observed in the data given, that the classes were mostly grouped by rows, e.g. The first half of the data is almost all Class 1, this means that if the data were to be split 50/50 and set the 1st half as training data then the MLP will be trained almost exclusively for Class 1. Therefore, because the data was given in such way, the order of the rows will be randomized. The training data will be split again to obtain validation data to test different hyperparameters (learning rate, number of neurons, etc.) and find the best architecture for MLP. The training data will be used to train the MLP and the trained parameters (weights and biases) will be the input of the MLP classifier to predict unseen data. See Fig.2 for visual representation of this.

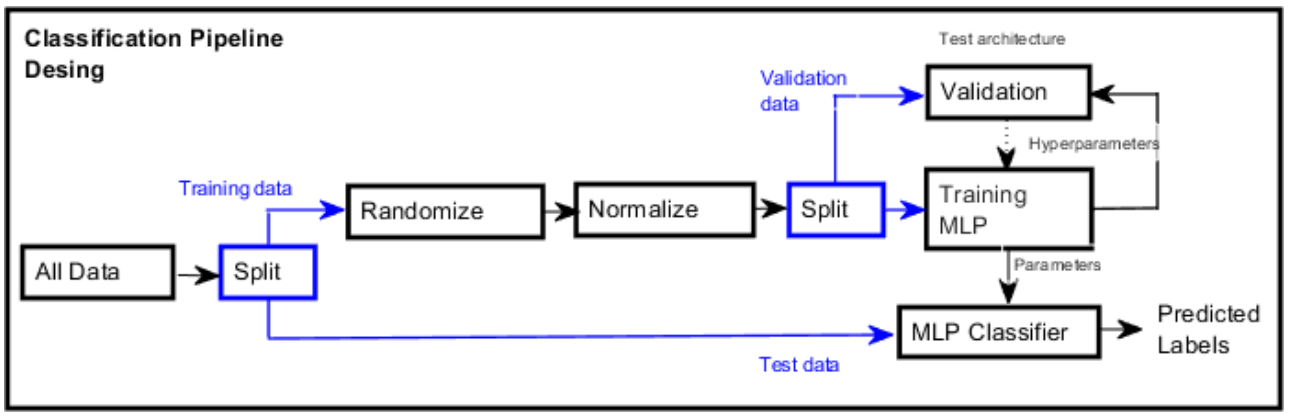

Figure 2 Visual representation of how the classifier will be designed

Implementation of code: File descriptions

Summary of ClassifyX.m

The algorithm used is the Feed forward algorithm with output neurons calculated as:

$$Neuron\ output = \sum_{i=0}^{n}(Wi\ Xi\ ) + Bias$$

Where: W=weight, X=input

The activation function used in ClassifyX.m is the sigmoid function:

$$Sigmoid = \frac{1}{1 + e^{-x}}$$

Where: "x" would be the output of a neuron.

The output layer of the MLP contains 5 neurons representing the labels from top to bottom as labels 1 to 5. The highest value output neuron from the 5 output neurons is selected as the predicted label.

Summary of TrainClassifierX.m

The classifier being trained is a fully connected multilayer perceptron classifier with 3x Layers:

- 64 input neurons
- 120 hidden neurons
- 5 output neurons

The trained parameters are:

- Weights
  - Weight from Input to Hidden neurons (64x120 matrix)
  - Weights from Hidden to Output neurons (120x5 matrix)
- Biases
  - Biases from Input to Hidden Neurons (120x1 matrix)
  - Biases from Hidden to Output Neuron (5x1 matrix)

Also, the true labels are given as single integers, and the architecture of the MLP has 5 outputs, therefore the code includes a function to represent the single integer label as a 5x1 vector, with 0s everywhere else apart from the class. e.g. Label 4 will be represented as [0 0 0 1 0]' and Label 1 = [1 0 0 0 0]'.

The algorithm use to train the classifier is the Back-Propagation algorithm where error is calculated at each layer backwards from the true label and the predicted one for every neuron involved a weight and a bias until the input neurons where the original weights and biases are updated by adding the difference (See code for detailed step by step math)

Other files

The hyperparameters for the training for Weights and Biases are depending on the number of hidden neurons, as the input and the output are set but hidden neurons are flexible, other hyperparameters include the learning rate and number of epochs to train. These flexible values are tested over the file called Hyperparameters.m

Another file was created apart form the main two, it is called Test_functions.m where the data is split into training data and test data, and MLP classifier is trained and then tested with unseen data along with the computation of the accuracy of the classifier on unseen data and the confusion matrix.

Test and validate. Methodology to verify performance. How hyperparameters were chosen? Includes accuracy and confusion matrix.

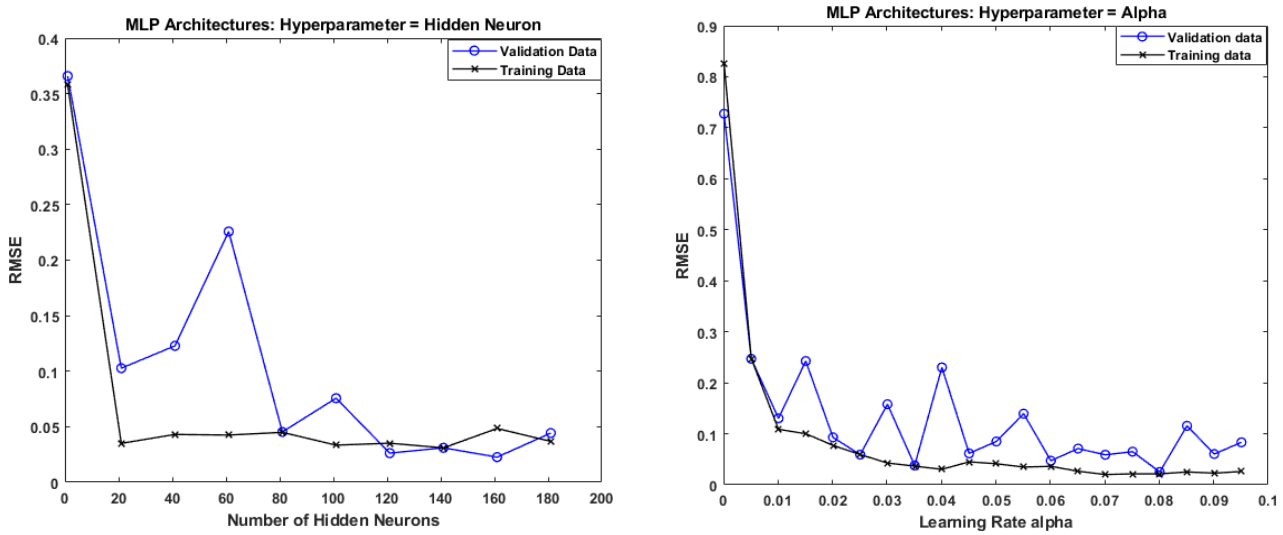

Figure 3 Testing different MLP architectures, a) shows RMSE for different hidden neurons, b) shows RMSE for different learning rates alpha.

The architecture of the MLP was validated by testing different values for the hyperparameters of the classifier: the learning rate, the number of hidden neurons and number of epochs. To choose the best hyperparameters the training data was further split into validation data and training data.

The first hyperparameter tested is the number of hidden neurons. In a separate file called Hyperparameters.m I looped the train classifier and calculated the Root Square Mean Error (RSME) and plotted it against the different architectures. This was done by setting the other hyperparameters constant and obtaining the RSME changing number hidden neurons each time. This was done in order to find the best number of hidden neurons to use in the MLP classifier. From Fig.3.a) it is concluded that the best number of hidden neurons is around 120 Hidden neurons where the RMSE of validation data was actually even lower than the RMSE of the training data.

The other hyperparameter tested is the value for the Learning Rate, alpha. In the same separate file Hyperparameters.m different learning rates were tested and calculated the RSME at each learning rate for training and for validation set. It was concluded that the best learning rate for this specific MLP classifier is 0.08 as seen in Fig.3.b), because from this point the RMSE for validation data starts to increase suggesting overfitting.

Lastly the last hyperparameter to choose is the number of epochs. The classifier presented in this coursework is trained via backpropagation, and the parameters, weights and biases. Instead of updating the weights and biases in batches they are updated for each point i.e. stochastic training. Therefore in 1 epoch, the classifier would have been updated, as many times as the number of training points, and takes around 7 s/epoch. In appendix it can be seen how with this large number of training

The performance of the classifier was obtained by looking at the confusion matrix and the accuracy of the classifier. The data given was split into 2 data sets, training and test data. The classifier was tested over the latter, "unseen data"

```
Testing with unseen data

matrixconfused =                    True Class

                1185           0           0           0           0
                   0        1242           2           0           0
Predicted          0           4        1057           0           0
                   0           0           5         726          44
                   0           1           7         157         371

                                                    Confusion
                                                      Area
accuracy =
                Accuracy
   95.4176
```

The columns of the confusion matrix are the true classes and the rows are the predicted classes. From this matrix it can be seen that the area of confusion is around classes 4&5. Predicting class4 as class5, 157 times out of 726 in that trial. There lies the confusion area of this MLP classifier.

With regards to accuracy, this was calculated by obtaining the sum of the diagonal of the confusion matrix over the total number of samples tested. And it can be seen that 95% accuracy was obtained on unseen data.

In summary the architecture of the MLP was validated by choosing hyperparameters values before overfitting by monitoring validation data, and the MLP was finally tested on unseen test data and performs with 95% accuracy.

APPENDIX

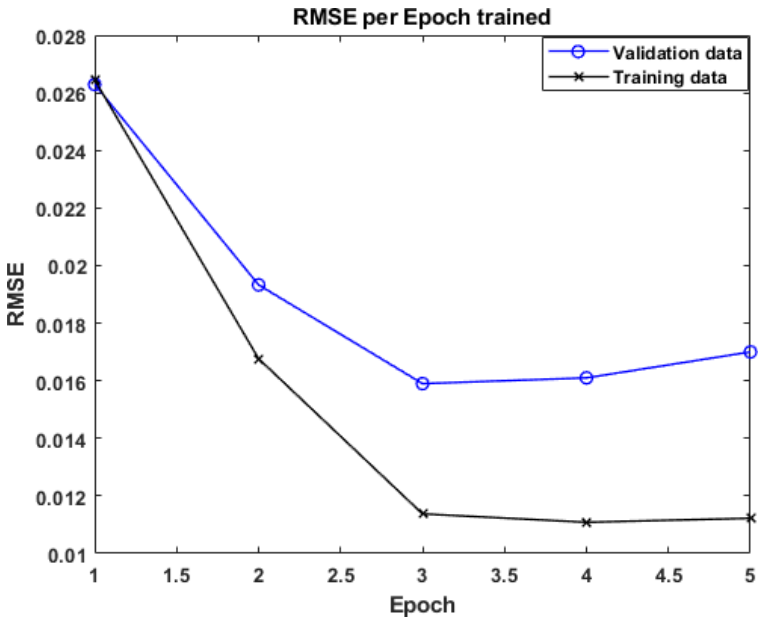

Figure 4 Training over different number of epoch