Imperial College
London

MLNC – Machine Learning & Neural Computation – Dr Aldo Faisal

Assessed Coursework 2

To be returned via TurnItIn as indicated online.

The focus of this assessed coursework is to solve the classification problem of different human activities. In the first part of the coursework, you are are provided with a multilayer perceptron (MLP), of which you will optimize the parameters to perform the classification. In the second part, you will implement your own classification method and compare your results.

This coursework counts for $\frac{1}{3}$ of your final course mark. The coursework submission should comprise a report and two Matlab files. Your coursework report should contain on each page in the header: Your name, your ic.ac.uk email, your CID and your degree course (i.e. "UG", "MSc NT", "MSc HBR", "MRes NT") on the header of each page. Your text should provide *brief* explanations so that the GTA markers can understand what you did. Please use *succinct* answers to the questions. The final document should be submitted in **PDF** format (we recommend using Latex but Word or other word processors are also fine). Figures should be from Matlab or a drawing program and appropriately labeled, captioned and referenced in the main text. Poor, unclear or incomprehensible presentation may result in a loss of marks. Marks are shown next to each question. Note that the marks are only indicative.

Your coursework report should not be longer than 3 single sided A4 pages including all figures (excluding code appendices). You are encouraged to discuss with other students, but your answers should be *yours*, i.e., written by you, in your own words, showing your own understanding. You have to produce your own code and submit it where appropriate.

You will need to turn in two Matlab scripts. Please make sure to add your fully annotated source code that generated your results in the appendix of your PDF submission. In addition please provide the two files `TrainClassifierX.m`, `ClassifyX.m` in the zip file. For the implementation of your own classifier, you can choose any of the methods discussed in the course; however, you are not allowed to use any of the built-in Matlab functions in the Statistics and Machine Learning Toolbox. Your classifier should run the sanity check script provided on Blackboard; otherwise you may be deducted points for your implementation.

Download and open the coursework zip-file from Blackboard. All coursework has to be submitted individually and will be marked as such. You are welcome to discuss your work during the lab sessions with other students and your GTAs. All code files must contain a comment line identifying the Name and CID of the submitter.

How to submit: Your 3 submission files (`ClassifyX.m`, `TrainClassifierX.m`, and your report PDF) should be all placed in a new single folder. The folder name should be your CID without leading "0"s. Go into the parent directory and zip your submission folder into a file called "YourCID.zip". Double-check that when you unzip your "YourCID.zip" file a folder called "YourCID" is created and the files are within. The submission zip file should then be uploaded to the TurnItIn submission box on Blackboard. Adhering to the correct format is important so the GTAs can run auto-testing of your models and see how they compare against each other.

Human Activity Recognition

In the first part of this coursework, you will use the provided Multilayer Perceptron code to classify human activities dataset. In the second part, you will implement your own classifier to solve the problem at hand.

You are given a human activity dataset, which contains labeled recordings of four different activities collected from a smart phone accelerometer. The smart phone was worn in a trouser pocket (see Figure 1 for the axes of motion relative to the user).

The accelerometer data (x,y,z acceleration) was collected at 20 Hz sample rate. The time-series were averaged into *3 dimensional data points* that reflect the average accelerations over a 10 second-interval. Each data point has one of 4 possible classes: walking, jogging, walking upstairs, walking downstairs.

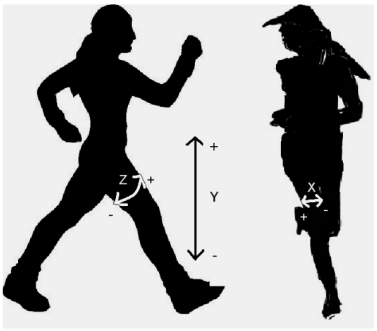

Figure 1: Axes of Motion Relative to the User

In your coursework zip directory you will find the following files:

- `Activities.mat` file contains 4 variables. The original dataset has 10,000 data points of averaged linear acceleration values (X,Y, Z). This dataset is split into half for training data, contained in `train_data` and the other half for testing data, provided in `test_data`. These datasets should be used to test the performance of both MLP and your own classification method. The columns of the variables `train_data` and `test_data` contain averaged linear acceleration values in the X,Y, Z axes respectively. The 4 recorded activities (classes) are walking, jogging, walking upstairs, walking downstairs with labels 1, 2, 3, 4 respectively. `train_labels` variable contains the labels for each data point in the `train_data` and `test_labels` variable contains the labels for each data point in the `test_data`.

- `run_MLP.m` – This is the script containing all the parameters of the multilayer perceptron and calls the `MLP.m` function that trains the network on the training data and returns the accuracy of the classification on the testing data. **You will only need to modify `run_MLP.m` code** for the questions. `run_MLP` (after running) yields the following variables in memory:

  - `accuracy` – a vector of percent classification accuracies in classifying the testing data for each epoch during the learning process.
  - `best_prediction` – a vector of predicted classes for each testing data point resulting from the epoch that yielded the highest percent accuracy in classifying the testing dataset.

- `MLP.m` – The Multilayer Perceptron (MLP) function. This function takes in the parameters you specify in `run_MLP.m` and trains the MLP on the training data and classifies the testing data in each epoch. This function does not need to be modified for the coursework, but you are welcome to familiarise yourself to see how an MLP can be implemented.

- `TrainClassifierX.m` – an empty function which should train your classifier on the training data and return the parameters for your classifier.

- `ClassifyX.m` – an empty function script which should take in testing data and your parameters and return the predicted class for each testing data point.
- `SanityCheck.m` – This script checks if your two functions (`ClassifyX` , `TrainClassifierX.m`) match the specifications we use. Note: If your functions do not pass the automatic tests performed by this function we will be unable to automatically evaluate their performance and may deduct marks.

Your overall goal is to investigate how parameters influence the classification performance of the multilayer perceptron and compare the performance and efficiency of training with another classifier (General Rule: You should always compare your classifier to another classifier performing the same task). Please be aware that the run-time will take up significant portion of your time as the neural networks are computationally heavy. In the following questions, you will be asked to generate plots from data or your results. The figures should be exported so that lines and data points are clearly visible when exported and any text in the figures is clearly readable. Use "export figure" to export the figure from Matlab and store it as PNG. Untidy or "grainy" figures will not be considered.

It is always worth to first explore a new dataset and e.g. visualize the accelerometer readings of all four activities on a single 3D scatter plot where the 3D dimensions correspond to X,Y,Z accelerations. You could plot each class in a different colour (have a look at `scatter3`). Are different activities (classes) clearly grouped together and separated from each other in this 3D space or are they overlapping and intersecting. Consider these challenges when you try to judge how a good is a "good" classification accuracy.

Questions

Multi-Layer Perceptron

1. Explore different architectural designs and configurations of a multilayer perceptron (MLP). ~~For all sub-questions, you should plot the learning curve,~~ i.e. the classification accuracy obtained for each epoch (Y-axis classification accuracy, X-axis number of epochs) for each of the tested configuration of the MLP. You are encouraged to investigate influence of different number of training epochs on classification accuracy, but keep in mind that it heavily influences code's runtime. ~~For this question, you should use 500 training epochs by default as already specified in the provided code to produce your plots~~.

   Hint: You will have to balance the speed of the computer and the time it takes to run these evaluations if they involve many or large networks. Just as in real-life where resources are finite, you will need to specify what is "good enough" and explain your cut-off. For the sake of computational efficiency the data was split already in training and test data, there is no need for cross-validation in this coursework.

   (a) Investigate how MLPs depend on their learning rate: ~~Analyse the effect of the learning rate on the learning behaviour of the MLP network setup of your choice.~~ We start with an MLP that just has 5 neurons in a single hidden layer. ~~Plot the learning curves for 500 training epochs obtained with different learning rates.~~ Learning rates can be static (fixed over all epochs) or dynamic (changing over the epochs). ~~Discuss how learning rate strategy and settings affect MLP training speed and performance.~~ (15 points)

   (b) Investigate how MLPs depend on their network geometry: ~~Consider MLPs with different numbers of hidden layers~~. ~~Consider MLPs with different numbers of neurons in each layer~~. ~~Discuss what systematic changes in the learning curve and performance of the MLPs you observe when you modify the number of neurons per layer and the number of layers~~. Support your answers with figures. (15 points)

   (c) Based on your findings from the previous questions, choose an MLP configuration (number of layers, number of neurons in each layer and learning rate) that yielded the best classification accuracy (write out your choice of parameters in a small table). ~~Explain your selection~~ and ~~calculate how~~

many weights the chosen neural network have in total (show your calculation). Using these parameters, calculate a confusion matrix. The confusion matrix of a classifier has as many rows and columns as classes, here 4. Each column of the matrix represents a predicted class, each row an actual class. The entry in the corresponding row and column is the probability or frequency that class "row" is classified as class "column". (15 points)

2. From the four classes, choose the two classes that you think will be the easiest to distinguish assuming that you only perform binary classification between these two classes. Using the parameters for your best-performing MLP you proposed in question 1.c, run the MLP now for binary classification of your two chosen activities from the previous question. Does the same network architecture perform better on the two-way classification than the 4-way classification? Report your achieved accuracy over 500 training epochs and produce the confusion matrix for the binary classification. (10 points)

Your classifier

3. Implement any of the classifiers covered in the course (except for neural networks) to classify the two classes you selected in question 2. Implement your classifier in the function `ClassifyX` that takes in arguments `input`, containing a column matrix of testing data points to be classified (each row is a data point) and `parameters` obtained from your `TrainClassifierX`. The function should return a column vector of predicted classes. Implement your training in the `TrainClassifierX`. The function should take in two arguments: `input` containing your training data points (each row one data point, the columns correspond to the features of the data point) and the training class labels (desired class labels) as a column vector of numbers. This function should return a `parameters` to be fed to your `ClassifyX` function. (15 points)

4. Test and validate your classifier's performance. Briefly describe your classification methodology and any relevant hyperparameters, report your method's accuracy and provide the confusion matrix for discussed binary classification solved using your method. Explain your choice of classifier and calculate how many parameters your chosen classifier has in total (show your calculation). (20 points)

   - Use `SanityCheck.m` to make sure your implementation produces parameters and labels that match our specifications.
   - Make sure to turn in the `ClassifyX` and `TrainClassifierX` files in your zip file along with your report. Please include additional scripts in the appendix of your report.

5. Compare the performance of the multilayer perceptron and your classifier based on their accuracies and confusion matrices for the binary classification. Discuss the advantages and disadvantages of each method. (10 points)

Hints:

1. More is not always better, while it is relatively easy to specify huge networks, these take much longer to train. Choose wisely.

2. If you have questions about the coursework please make use of the labs or Piazza. You will understand that GTAs cannot provide you with answers that directly solve the coursework, they will be delighted to help you with conceptual, programming, mathematics questions underlying your coursework.