Imperial College London

MLNC – Machine Learning & Neural Computation – Dr Aldo Faisal

Assessed Coursework 2

Update 4/12/2018: Clarification on submission of files and tests to be performed following Piazza input.

The coursework return deadline is the first work day of term:
Monday 7th January 2019 at 4pm.
To be returned via TurnItIn as indicated online.

The focus of this assessed coursework is to solve the classification problem of different human activities. 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. Going over the page limit may incur a loss of points. 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, 12 pt font and 2 cm margins, 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 at least two Matlab scripts. Please provide the two files `TrainClassifierX.m`, `ClassifyX.m` in the zip file. To be clear: when we copy these two `.m` files into a different directory (the one used by our autotester) these two functions should be able to run without needing to refer to other code of yours. 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 higher built-in Matlab functions in the Statistics and Machine Learning Toolbox. You can test your classifier by running the sanity check script. Code that does not conform to these coding standards may be deducted points for your implementation. By the way, these type of coding standards are the same as in any real-world setting where software development requires the input from many different software developers – learning to be able to conform and work along such standards is an integral part of your training and actual deployability of your skills.

Please make sure to add your fully annotated source code that generated all your results (i.e. all plots and numbers reported) in the appendix of your PDF submission as well as to the .zip (not .7z or .rar) file submission. This is so the GTAs can reproduce or retrace your steps and thus help them in their assessment of your work.

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 submission should contain at least 3 files (`ClassifyX.m`, `TrainClassifierX.m`, and your report PDF) should be all placed in a new single folder. The folder name should be your `Firstname_Lastname-CID` without leading "0"s in the CID. Go into the parent directory and zip your submission folder into a file called `Firstname_Lastname-CID`.zip (no whitespaces or blanks in your name please). Double-check that when you unzip your `Firstname_Lastname-CID`.zip file a folder called `Firstname_Lastname-CID` is created and all 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. If the GTAs cannot run your code or extract the relevant files this may result in a loss of points.

Human Activity Recognition

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

- `data.mat` file contains 1 variable, called `data`. The dataset has 24,000 data points (rows) and 65 columns. The first column is the true activity label: The 5 recorded activities (classes) are sitting, standing, walking, jogging, martial arts with labels 1, 2, 3, 4, 5, respectively. The last 64 columns are features extracted from the data, a raw accelerometer and gyroscope data collected from wearable sensors were preprocessed.
- `TrainClassifierX.m` – an empty function which should train your classifier on the training data and return the parameters for your classifier. The function takes as input the variable `train_data` which is the training data input, i.e. a matrix where each row is an observation and the columns are the 64 dimensions of the input data. The second input is a variable `label` that is a column vector where each row is the label (1,2,3,...) of the corresponding observation in `train_data`. We have written the specification of the function into the file and give a code example.
- `ClassifyX.m` – an empty function script that implements your classifier decision. This function should take in testing data point (i.e. an observation) `input` and your classifier's parameters in the variable `parameter` and should return the predicted class label (in our case 1,2,3,...) as output in `label`. We have written the specification of the function into the file and provided an example random output.
- `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.
- 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. Note, that our automated testing code will read these two files only. The code should be able to run (test it with `SanityCheck.m`) as the GTAs will not able to fix any code that does not run because it does not conform to our coding standards.

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 and adjust the settings to make the clearly readable. Untidy or "grainy" figures will not be considered.

It is always worth to first explore a new dataset and e.g. visualize the data. E.g. are different activities (classes) clearly grouped together and separated from each otheror are they overlapping and intersecting? Consider these challenges when you try to judge how a good is a "good" classification accuracy.

Question 1

Develop a classification pipeline that allows you to classify the data. Describe how and why you choose your specific pipeline, including any reasoning behind preprocessing and your classification method. (20 points)

Question 2

Implement your classifier described above 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. Use `SanityCheck.m` to make sure your implementation produces parameters and labels that match our specifications. (40 points)

Question 3

Test and validate your classifier's performance. Briefly describe your methodology to verify the performance and generalisation capability of your classifier and how any relevant hyperparameters were chosen. Document these decisions with figures or tables as needed so as to efficiently and effectively communicate the robustness of your approach. Your report should provide the classification accuracy and the confusion matrix for your classifier. Discuss the advantages and disadvantages of your chosen method and relate them to your design choices. (30 points)

Question 4

Your code (which is why it needs to conform to the above laid out coding standards) will be used to perform automated testing for us to evaluate your classifier on our own test data. The best performing classifiers will be awarded up to 10 points.