**AIDL_B_AS01 - Signal Processing, Pattern Recognition and Machine Learning**

## Assignment #1 - Features Extraction for Classification

A set of training and test examples of human activity sensor readings is provided at: *https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones*

The goal is to recognize the correct human activity (out of six) for each test example based on the training data.

1. ***Data inspection and acquisition.*** Inspect the data and the accompanying files and read the related published paper at

https://www.esann.org/sites/default/files/proceedings/legacy/es2013-84.pdf

The data of interest for this assignment are located in the "Inertial Signals" train and test folders. Specifically, we will focus on the "body_acc_x", "total_acc_x", and "body_gyro_acc_x" train and test data files which contain one-dimensional body sensor readings, of length 128 samples, across the x-axis for 6 different human activities. The training set contains 7352 instances while the test set contains 2947 instances, i.e. you should get 3 matrices of size 7352x128 for the training data and 3 matrices of size 2947x128 for the testing data.

2. ***Yule–Walker linear prediction***. Perform linear prediction via the Yule-Walker equations of the current sample based on the $p$ previous samples (i.e. order $p$), for 10 randomly chosen data vectors of 128 samples of the "body_acc_x_train.txt" file. To find the best order $p$, calculate the error between the predicted vector and the original vector and take the average error over all 10 cases. Try several values, starting from $p$=2 and find the one that minimizes the average error. Plot the original data and the predicted data on the same plot for one of the 10 vectors for the best order $p$. What happens if $p$ is too high?

Theoretical question: How is linear prediction related to AR modelling and specifically when is the output of a linear predictor an AR process?

3. ***KNN algorithm.*** You will be testing the effect of features design on the classification performance of the KNN (K-Nearest-Neighbors) classifier. This simple classifier will provide a scoring mechanism for the assessment of your selected features. A good tutorial on the KNN algorithm can be found at

https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/

4. ***KNN benchmark***. Perform KNN classification on the testing set of step 1, given the training set of step 1. You will use the Euclidean distance metric for KNN throughout

this assignment. The performance metric of KNN will be the total classification accuracy, defined as

*total accuracy = # of correctly classified instances/# of instances examined (%)*

As a first step, perform classification on the raw data in order to acquire a benchmark for comparison with the configurations that you will implement next. Set *K*=3 and report the total accuracy.

5. ***KNN data pre-processing.*** Perform KNN classification after you have applied normalization as a pre-processing step. Set again *K*=3. Report the new classification accuracy and discuss how it compares to the benchmark accuracy. Various normalization methods exist, such as normalization to zero mean and unit standard deviation, range normalization to [0, 1], subtraction of global mean and division by global standard deviation.

6. ***Selecting the most important training set.*** Select the two most important (out of the three) training data files (e.g. combine "body_acc_x_train.txt" and "total_acc_x_train.txt") and justify your selection. This will be your new training set (adjust accordingly the testing set).

7. ***KNN features design.*** Perform classification with different features derived from the training and testing sets. The goal is to find the features that yield the highest classification accuracy among several feature configurations. The file features_info.txt contains some ideas on possible features, such as min, max, kurtosis, skewness etc. In order to constrain the high number of available features, the maximum length of the derived feature vector is set to 32, i.e. 1/8 of the original data vector length. You will design at least 3 different feature configurations and report on their effect on total classification accuracy. Try not to use pure luck but instead apply logical reasoning and intuition in order to arrive at the best features configuration.

As a starting point, experiment with the following 3 feature designs and then combine them to achieve the best accuracy:

a) Only LPC (linear prediction coding) coefficients as calculated in step 2

b) Only time domain features (e.g. mean, variance, zero-crossing rate)

c) Only frequency domain features (FFT basically)

Related question: Is step c) related to a spectrogram and how?

8. ***KNN fine-tuning.*** The KNN algorithm contains only one hyperparameter, namely the number of neighbors *K.* Try different values of *K* on the best features design derived in the previous step and report on possible improvements in terms of classification accuracy. What happens when *K* increases or decreases and why? How is it related to model underfitting/overfitting? Why would we select odd values of *K*?

Remarks:

- The easiest way to initially visualize and explore the data is through Excel (Matlab and Python can also be used). Be careful with the arithmetic format of the data as the dot (.) is used for the decimals instead of a comma (,).
- You will write your own code for this task, excluding basic operations/functions such as mean, standard deviation, FFT, Yule-Walker etc. whose existing code you may use freely. You will have to write your own code for the KNN algorithm.
- You should submit:

  **A.** One pdf file with your answers, together with formulas and plots where necessary. It should have the form of a report and CONTAIN NO CODE.

  **B.** One .ipynb file or .m file with your code and some comments to explain the main tasks of the code.
- The answers to the above problems should be analytical with supporting formulas and plots where necessary. Attention to detail and the ability for rigorous reasoning will be assessed and evaluated.
- The code should be easily executable as it will be examined and run by the tutors. Matlab or Python may be used. Specifically for python files, they should run smoothly on **JupyterLab (Anaconda is recommended)**.