EE660 Machine Learning from Signals

Course Project

Wearable Computing: Human Activity Recognition

Akash Mukesh Joshi

USC ID: 4703642421

Abstract

Human Activity Recognition has emerged as one of the key research areas in the past decade. We have seen HAR being used in the gaming industry a lot. Most of the animation we see its done using human activity recognition. Cameras and Video capturing have been done to do human activity recognition.

Based on this a new field of research emerged where hardware sensors where mounted at different locations on a human and this sensor data was then used in a machine learning algorithm to perform Human Activity Recognition. In this project we doing classification of body postures and movements by attaching four accelerometer data located at the ***waist***, ***left thigh, right ankle*** and ***upper right arm***. The classification of data set is done by using the accelerometer data in the x, y and z axis and other features such as gender, age, height, body mass index and weight. The data is taken in packets over 150 millisecond window. Each packet is then converted into a single data point and labeled into an activity. This is called supervised learning. These parameters help in classification of the dataset.

The dataset is classified into 5 activities which are ***sitting, siitingdown, standing, standingup*** and ***walking***. The dataset was first divided into 2 parts – "***training dataset***" and "***testing dataset***" keeping the percentage class distribution the same in comparison to the original dataset. Before the training data was every given to different models the data needed to be cleaned and pre-processed and various measures such as treatment of missing values, outlier detection and removal, categorical data merging, feature dimensionality expansion or reduction and normalization might help in preprocessing the dataset. We will see some of these methods of preprocessing used on the dataset. The pre-processed data was then sent to various types of classifiers to train them for the incoming testing dataset. It was found that ***K-Nearest Neighbors*** gave the best ***F1-score of 0.99*** on the testing dataset and we were able to correctly classify almost all the data correctly. This goes to show that using hardware sensors is the one of the best methods of performing human activity recognition.

Goals and Methodology

The primary goal of this project is to find a classification model which can accurately categorize incoming unknown data points with any overall accuracy of greater than 80 % and also identify other variable sensor positions to improve the classification by adding in more activities in the dataset. To achieve this goal meant that the data had to be pre-processed and cleaned thoroughly and then sent to suitable classifiers to model them for the testing dataset.

Dataset Generation & Pre-processing of Dataset

The data set had to be first divided into 2 parts to give us a training set and a testing set which would be used once the classifier is trained on the training set.

1. ***Training and Testing Set Dataseparation***

   The dataset is divided into two parts randomly with an even distribution of each class in each part using the ***cvpartition ()*** MATLAB inbuilt function using 2 fold split. This split the 165633 x 18 data set into 82817 x 18 data set - ***feature_train*** and 82816 x 20 data set - ***feature_test***. Column number 19 is changed from a string array to an integer array and also split according to its relative split of the rest of the 18 column thus leaving us with 82817 x 1 data set - ***label_train*** and 82816 x 1 data set ***label_test***.

   The training data set will be used for preprocessing and designing the pattern recognition system. The test data set will not be touch until the final classification system is not ready. Any kind of preprocessing applied to the training data set will directly apply to the testing data while testing the classifier. Data points in the test data set if found to be non-existent in the train set will not be classified to any class and will be deemed to lie in the unclassified region.

2. *Preprocessing of Training Data Set*

The training set is now sent to another MATLAB function – ***preprocessing.m*** where the training data set gets preprocessed such that the relevant data is sent to the classifier. Some of the techniques used to preprocess this data set are: recasting of the features into numeric form, removing/replacing of missing data with an appropriate representation of the data present in that column, removing sample points and perform feature dimensionality reduction of the data set, normalization of the data set if required.

(a) Recasting the representation of Feature

The training set is imported into MATLAB as a ***table*** which contains different data types. We need to firstly recast the data set into single datatype. ***Numeric*** columns/features are not changed throughout the entire data set. On the other hand, ***String*** data set is converted into ***Numeric*** form by allocating different numbers for a different string set in the column entry.

(b) Data Point Reduction

At this stage of pre-processing, there were some duplicate/irrelevant data entries in the dataset. These data samples are hence discarded.

(c) Missing Values Treatment

Dealing with missing values is really important as it can lead to the biased performance of the classification model on the training dataset. The missing data in the data set is represented as ***NaN*** values or strings as containing ***'?'***. Firstly, we need to find out which features have missing data and the amount of occurrence of the missing data. This can be done using the ***isnan()*** or ***strcmp(feature(:,:),{'?'})*** function in Matlab. We treat missing data by removing that data point in the training and testing dataset too.

(d) Feature Dimensionality Reduction

Many features in the data set contain irrelevant data and that leads to the poor classification of testing dataset if they are not discarded. This kind feature selection is done to reduce the dimensionality of the dataset and is carried out in two ways – finding the ***correlation*** between features and looking of near to ***zero value variance*** of a feature. Features that also have a large number of ***unique values*** that is the number of unique values is close to the number sample points those features are also eradicated from the training as well as the testing data set.

Correlation and Near Zero Variance

- This kind feature dimensionality reduction depends on the correlation among different features and if a particular ***feature is highly correlated*** with other features then that feature is ***completely removed*** from the training and testing data set. This is achieved using the Matlab function ***corrcoef(feature_train_pp,'rows','pairwise')***. This function checks the correlation between all the 49 features and results into a correlation matrix.
- Many features in the dataset have near to zero variance which means that those features very ***less differentiable data points*** and these data do not hold any importance when given to classifier to model the data set. This is done using the ***var()*** function and then finding out which column variances were near zero value (< 0.1).

(e) Outlier Detection and Removal

An outlier is a data point that statistically inconsistent with rest of the data set. In this project, we looked into ***univariate*** outliers only. Most of these outliers were dealt with by using the categorical feature expansion method. To find the remaining outliers we can look at the box and whisker plots. The outliers are only removed from features having continuous values as categorical features have been converted to indicator feature space. Hence, outlier detection was done on the basis of the ***modified Thompson tau technique***. In this kind of outlier detection and removal, only 1 data

point is considered at a time and the new means and standard deviations are calculated every single time an outlier is detected and removed.

(f) Normalization

All the data set were normalized such that the entire data set is within a predefined set range. Un-normalized data was also run on various classifiers and it was found that the classifiers were very sensitive to vast differences in the data set values or skewed data and hence needed to be normalized. The categorical features that were expanded were also normalized. The reason behind doing this was to adjust the dataset for the kNN classifier to prevent every data point with 0s and 1s to lie within the 16-dimensional hypersphere. Normalizing the features would result in the irregular distribution of the data points within and outside the hypersphere and hence would prevent any conflict arising during selecting the k nearest neighbors.

Classifiers

The project uses 2 classifiers: ***Least Squares,*** and ***KNN Model.*** The KNN classifier gives the best results in comparison to the other classifier. Many other classifiers like MSE, SVM, Decision Tree, Bagging and Adaptive Boosting were also tried out and the results were not satisfactory, hence the above 2 classifiers were chosen and are explained briefly below.

KNN Model with 5 Neighbors

The ***KNN Model*** using the classification toolbox gave the best accuracy.The toolbox was used to import the training set and an option of K-Fold cross-validation was also provided. Different values of K-Fold were chosen (10:10:50) during the classification modeling stage. It was seen that increasing the number of K for cross-validation would lead to increasing the accuracy on the testing set i.e. increase in F1 score.

To ***validate such a*** **classifier** a few tests were run to find the best:

- ***Number of K nearest neighbors***
- ***K-Fold cross-validation value*** which would give minimum classification error.

Number of K nearest Neighbors

This test was done on the training set itself. Here the ***fitckNN()*** function was used with 10 different values of the parameter ***'NumNeighbors'*** over the range of number of samples using the ***logspace()*** function. It was found that a value of 104 nearest neighbors resulted in minimum classification error as it can be seen in the graph below.

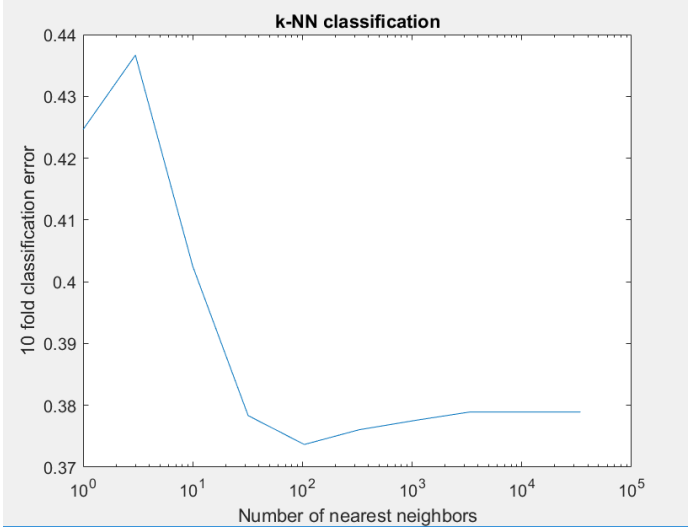


Figure 4: Best number of nearest neighbors

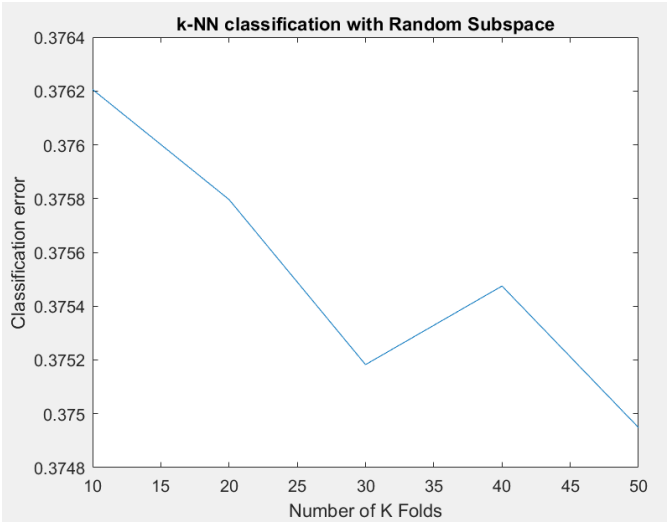


Figure 5: Best Number of K Folds

K-Fold Cross-Validation

Using the above parameters the model was run again for different values of K Folds ranging from 10 to 50 over a step of 10. It can be seen from the graph below that a 50 Fold cross-validation yields the best results. These 2 parameters were chosen and fed to the "***classification toolbox***". These predicted labels were tested with the true labels and an ***F1 score of 0.99*** and an ***accuracy of 99.38%*** was achieved.

Least Squares

The least squares classifier is distribution free classifier and is modeled using the ***multiclass()*** function provided in the classification toolbox. The model was run using the ***one vs all ('all-pairs)*** algorithm to classify data between these 3 classes. The maximum number of iterations used to achieve the optimum weight vector was ***1000 iterations***. The LS classifier is not a good classifier in comparison to the above classifier as it does not take into consideration the class distributions. This classifier gave the best ***F1 score of 0.73*** and an ***accuracy of 74.48%*** on the testing data set.

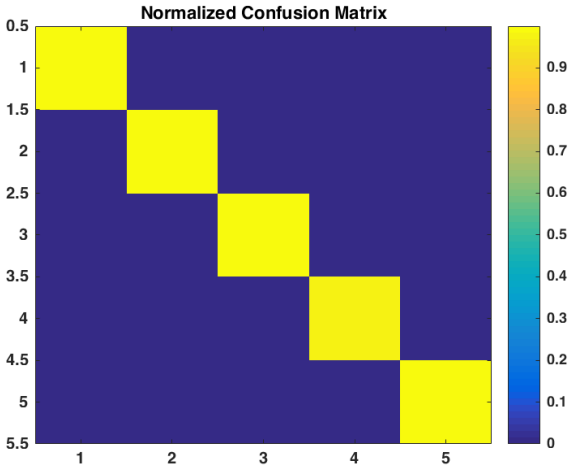
Performance Evaluation Techniques

The performance evaluation was done on the testing data set that which would go through the same pre-processing steps as the training data did. The testing data was then fed into different classification models and predicted labels were obtained from testing the test data on the trained classifiers. This predicted data was then sent to a function ***classification_report()*** to test the predicted labels with the true test labels. This function resulted in a ***confusion matrix*** and an ***F1 score*** for the testing dataset. A confusion matrix is a table that is used to describe the performance of classification model on a set of testing data set whose true label test is known. The classification performance of each classifier is shown below.

Results

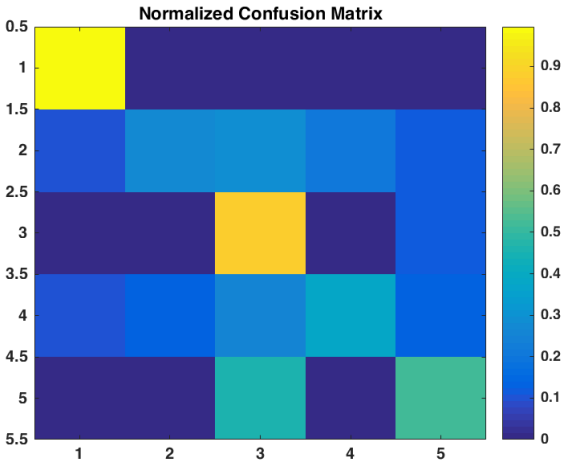
The results of various classifiers are shown below using the ***classification_report()*** function provided to get the ***confusion matrix and the F1 score.***

KNN Model – BEST RESULT



Precision	Recall	F1-score	Support
1.00	1.00	1.00	25315
0.98	0.99	0.98	5914
0.99	1.00	0.99	23685
0.98	0.97	0.98	6206
1.00	0.99	0.99	21695
0.99	0.99	0.99	82815

Least Squares Classification



Precision	Recall	F1-score	Support
0.94	1.00	0.97	25311
0.62	0.27	0.38	5831
0.61	0.88	0.72	23173
0.62	0.38	0.47	6013
0.73	0.52	0.61	21488
0.75	0.74	0.73	81816

Conclusion and Future Works

- Pre-processing of data is the most important part of the entire project on improving the accuracy of the classification models. It has been inferred that some features contain more relevant data as compared to others and the pre-processing steps helped in identifying such features.
- Important factors that lead to readmission of patients were : Sensor on the Belt: discretization of the module of acceleration vector, variance of pitch, and variance of roll; Sensor on the left thigh: module of acceleration vector, discretization, and variance of pitch; Sensor on the right ankle: variance of pitch, and variance of roll; Sensor on the right upper-arm: discretization of the module of acceleration vector; From all sensors: average acceleration and standard deviation of acceleration.
- It was also seen that testing data sets in small bundles would results in better classification on the testing dataset and this led to choosing ***10 to 50 fold cross-validation*** techniques to prevent data overfitting. Using cross-validation ***increased the performance*** of the KNN classifier giving an ***F1 score of 0.99***
- The next step is to use 14 sensors at different body locations such that we can have more number of classes and user specific performance recognition on a single class.
- Introduction of frequency domain features during feature engineering step of preprocessing will help us in making new features which might carry some new information as we changing the domain from time to frequency.

References

- ***Large-Scale Support Vector Machines: Algorithms and Theory*** - Aditya Krishna Menon https://cseweb.ucsd.edu/~akmenon/ResearchExam.pdf
- ***Random Subspace Classification*** https://www.mathworks.com/examples/statistics/mw/stats-ex98983766-random-subspace-classification
- ***Nearest Neighbor Ensemble*** - Carlotta Domeniconi, Bojun Ya, Information & Software Engg Dept, George Mason University https://cs.gmu.edu/~carlotta/publications/NNensemble.pdf
- ***A Comprehensive Guide to Data Exploration –*** Sunil Ray http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#three
- ***Outliers -*** John M. Cimbala, Penn State University http://www.mne.psu.edu/cimbala/me345/Lectures/Outliers.pdf
- ***Remove Outliers –*** M Sohrabinia http://www.mathworks.com/matlabcentral/fileexchange/37211-remove-outliers
- http://groupware.les.inf.puc-rio.br/har
- http://groupware.les.inf.puc-rio.br/public/papers/2012.Ugulino.WearableComputing.HAR.Classifier.RIBBON.pdf