SUMMER INTERNSHIP PROGRAM 2019

20th May, 2019 - 5th July, 2019

Human Activity Recognition Using Smartphone

Final Report

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

ROBOTICS AND MACHINE ANALYTICS (RAMAN) LAB

Submitted by:

Mukul Dhaka	**Kshitij Abhay**	**Abhishek Kumar Pandey**
Govt. Engineering College Ajmer	Govt. Engineering College Ajmer	Cochin University of Science and Technology
mukuldhaka5@gmail.com	kshitijabhay@gmail.com	akumarpandey686@gmail.com

Under the supervision of

Prof.(Dr.) Rajesh Kumar

Department of Electrical Engineering
Malaviya National Institute of Technology
Jaipur, Rajasthan, India

Contents

1 Introduction

The experiments have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities (WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS, SITTING, STANDING, LAYING) wearing a smartphone (Samsung Galaxy S II) on the waist. [1]

Using its embedded accelerometer and gyroscope, 3-axial linear acceleration and 3-axial angular velocity has been captured at a constant rate of 50Hz. The experiments have been video-recorded to label the data manually.

The obtained dataset has been randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window).
The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity.

The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.

There are 7352 records in training set and 2947 records in testing set.

2 Problem Development

Activity recognition data set built from the recordings of 30 subjects performing basic activities and postural transitions while carrying a waist-mounted smartphone with embedded inertial sensors. [2]

2.1 Information regarding Features

The features selected for this database come from the accelerometer and gyroscope 3-axial raw signals tAcc-XYZ and tGyro-XYZ. These time domain signals (prefix 't' to denote time) were captured at a constant rate of 50 Hz. Then they were filtered using a median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz to remove noise. Similarly, the acceleration signal was then separated into body and gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ) using another low pass Butterworth filter with a corner frequency of 0.3 Hz.

Subsequently, the body linear acceleration and angular velocity were derived in time to obtain Jerk signals (tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ). Also the magnitude of these three-dimensional signals were calculated using the Euclidean norm (tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag).
Finally a Fast Fourier Transform (FFT) was applied to some of these signals producing fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag. (Note the 'f' to indicate frequency domain signals).

2.2 List of Features in dataset

These signals were used to estimate variables of the feature vector for each pattern:
'-XYZ' is used to denote 3-axial signals in the X, Y and Z directions.

- tBodyAcc-XYZ
- tGravityAcc-XYZ
- tBodyAccJerk-XYZ
- tBodyGyro-XYZ
- tBodyGyroJerk-XYZ
- tBodyAccMag
- tGravityAccMag
- tBodyAccJerkMag
- tBodyGyroMag
- tBodyGyroJerkMag
- fBodyAcc-XYZ
- fBodyAccJerk-XYZ
- fBodyGyro-XYZ

- fBodyAccMag
- fBodyAccJerkMag
- fBodyGyroMag
- fBodyGyroJerkMag

The set of variables that were estimated from these signals are:

- mean(): Mean value
- std(): Standard deviation
- mad(): Median absolute deviation
- max(): Largest value in array
- min(): Smallest value in array
- sma(): Signal magnitude area
- energy(): Energy measure. Sum of the squares divided by the number of values.
- iqr(): Interquartile range
- entropy(): Signal entropy
- arCoeff(): Autorregresion coefficients with Burg order equal to 4
- correlation(): correlation coefficient between two signals
- maxInds(): index of the frequency component with largest magnitude
- meanFreq(): Weighted average of the frequency components to obtain a mean frequency
- skewness(): skewness of the frequency domain signal
- kurtosis(): kurtosis of the frequency domain signal
- bandsEnergy(): Energy of a frequency interval within the 64 bins of the FFT of each window.
- angle(): Angle between to vectors.

Additional vectors obtained by averaging the signals in a signal window sample. These are used on the angle() variable:

- gravityMean
- tBodyAccMean
- tBodyAccJerkMean
- tBodyGyroMean
- tBodyGyroJerkMean

3 Methodology

3.1 Logistic Regression

Logistic Regression uses Sigmoid function.

An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is a Sigmoid function, which takes any real value between zero and one. It is defined as

$$\sigma(t) = \frac{e^t}{(e^t + 1)} = \frac{1}{(1 + e^{-t})} \tag{1}$$

And if we plot it, the graph will be S curve,

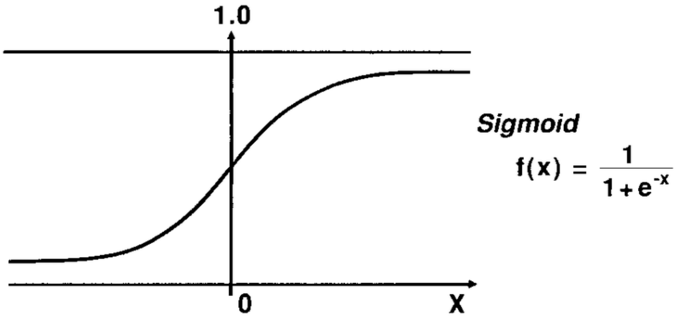

Figure 1: Sigmod Function [4]

Multinomial Logistic Regression

Since we are having more than 2 classes therefore we are using Multinomial Logistic Regression.

Multinomial Logistic Regression is the regression analysis to conduct when the dependent variable is nominal with more than two levels. Similar to multiple linear regression, the multinomial regression is a predictive analysis. Multinomial regression is used to explain the relationship between one nominal dependent variable and one or more independent variables.[3]

3.2 Support vector machine

A Support Vector Machine is a supervised machine learning algorithm which can be used for both classification and regression problems. It follows a technique called the kernel trick to transform the data and based on these transformations, it finds an optimal boundary between the possible outputs. The main idea is to identify the optimal separating hyperplane which maximizes the margin of the training data.

3.3 Random forest

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees. In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracyresults.

4 Simulations and Results

4.1 Using Logistic Regression

```
In [20]: cli = [10,1,0.5,0.1,0.01,0.003]

In [21]: for j in cli:
             clas = LogisticRegression(C=j)
             clas.fit(x_train,y_train)
             y_pred_train = clas.predict(x_train)
             y_pred_test = clas.predict(x_test)
             trainac = accuracy_score(y_pred_train,y_train)
             print("For value of c =",j,"Training Accuracy score is :",trainac)
             testac = accuracy_score(y_pred_test,y_test)
             print("For value of c =",j,"Testing Accuracy score is :",testac)

         For value of c = 10 Training Accuracy score is : 0.9955114254624592
         For value of c = 10 Testing Accuracy score is : 0.9609772650152698
         For value of c = 1 Training Accuracy score is : 0.9911588683351469
         For value of c = 1 Testing Accuracy score is : 0.9613165931455717
         For value of c = 0.5 Training Accuracy score is : 0.9895266594124048
         For value of c = 0.5 Testing Accuracy score is : 0.9623345775364778
         For value of c = 0.1 Training Accuracy score is : 0.9816376496191512
         For value of c = 0.1 Testing Accuracy score is : 0.9575839837122497
         For value of c = 0.01 Training Accuracy score is : 0.9479053318824809
         For value of c = 0.01 Testing Accuracy score is : 0.9409569053274517
         For value of c = 0.003 Training Accuracy score is : 0.9177094668117519
         For value of c = 0.003 Testing Accuracy score is : 0.9138106549032915
```

Figure 2: Logistic Regression

Here, we are tunning the parameter(C) using multiple values of C and we have seen that at C=0.5, we are getting the highest testing accuracy of approximately 96.23%.

4.2 Using SVM

```
In [24]: clf = SVC()
         clf.fit(x_train, y_train)
         predsvm = clf.predict(x_train)
         a = accuracy_score(y_train,predsvm)
         predsvm1 = clf.predict(x_test)
         b = accuracy_score(y_test,predsvm1)
         print("Trainging Accuracy score:",a)
         print("Testing accuracy score:",b)

         Trainging Accuracy score: 0.9579706202393906
         Testing accuracy score: 0.9307770614183916
```

Figure 3: SVM

We have come across the equation of a straight line as $y = mx + c$, where m is the slope and c is the y-intercept of the line and m and c are the parameter for the model and we have kept them as default. So we are getting the testing accuracy of 93.07%

4.3 Using Random Forest

```
In [27]: rf_model = RandomForestClassifier(n_jobs=2,random_state=0)

In [28]: rf_model.fit(x_train,y_train)

Out[28]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                     max_depth=None, max_features='auto', max_leaf_nodes=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=2,
                     oob_score=False, random_state=0, verbose=0, warm_start=False)

In [29]: pred_train = rf_model.predict(x_train)

In [30]: pred_test = rf_model.predict(x_test)

In [31]: c = accuracy_score(y_train,pred_train)
         d = accuracy_score(y_test,pred_test)
         print("Training Accuracy Score:",c)
         print("Testing Accuracy Score:",d)

         Training Accuracy Score: 0.999727965179543
         Testing Accuracy Score: 0.9070240922972514
```

Figure 4: Random Forest

To perform the prediction using the trained random forest algorithm we have passed the test data through the rules of each randomly created trees.
We got the testing accuracy of about

Table 1: Accuracy table for Training Session

Training Method	Accuracy
Using SVM	95.79%
Using Random Forest	99.97%
Using Logistic Regression	98.95%

Table 2: Accuracy table for Testing Session

Testing Method	Accuracy
Using SVM	93.07%
Using Random Forest	90.70%
Using Logistic Regression	96.23%

5 Conclusions and Future work

We achieved,
Training accuracy using Logistic Regression is : 98.95%
Testing Accuracy using Logistic Regression is : 96.23%
Logistic Regression gives the best performance and stability, hence, it is promising to run on mobile devices.
In this work, we analyzed the role of accelerometer and gyroscope sensor in activity recognition using logistic regression. Based on the experiment, accelerometer and gyroscope sensors can be used to recognize human activities individual. Combining both sensors performed better than using them individually, however, using multiple sensors can create serious challenge due to mobile phone battery limitations- low battery capacity. Activity recognition needs continuous sensing from the mobile phone. In future, we will use accelerometer sensor data to implement real time human activity recognition using smartphone. [6]

Bibliography

[1] https://github.com/gokulramanaa/Human-Activity-Recognition-with-Smartphones

[2] https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

[3] https://www.statisticssolutions.com/mlr/

[4] https://www.researchgate.net/profile/Knut_Kvaal/publication/239269767/figure/fig2/AS:643520205
illustration-of-the-signal-processing-in-a-sigmoid-function.png

[5] https://static1.squarespace.com/static/597d2753be6594cef6a34840/597d2c4715d5dbc171e4723f/597d
06-01100-g002.png?format=1000w

[6] https://www.google.com/url?q=https://dl.acm.org/citation.cfm?id