8/3/2019

Human Activity Recognition with Smartphones

PROJECT REPORT

Team Members:

- Akash Kumar Agile Mohan (axa175730)
- Amith Kumar Matapady (axm180029)
- Hitesh Gupta Tumsi Ramesh (hxg170230)
- Rushikesh Shirish Kulkarni (rsk180001)

Table of Contents

Introduction and problem description :

In the era where information is fuel, Mobile devices are one of the major contributer. Mobile devices have various sensors and the information generated by them is not used to its fullest. Among them are Accelerometer and Gyoscope.Their primary use is to detect the orientation of the phone , linear acceleration of movement and angular rotational velocity. This information can also be used to profile the user activity in areas such as Security, Surveillance and in understanding bevahioural patterns of humans.

Our aim is to use the information generated by smartphones , process them and then classify the human activity into six categories. The categories are WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LAYING.

Related work

The original experiment was performed by the team **Anguita et al.** they focused on applying a support vector machine adapted for multiclass classification.

Bao et al. developed algorithms to detect physical activities from everyday tasks. They observed that while some activities are classified more accurately with subject-independent training data, others require subject-specific training data.

There are several teams and interested individuals who have worked on same problem and their method and results are available online for reference.

Dataset description

We are using the following test and train dataset for our project:

https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones/downloads/test.csv/1 (Test dataset)
https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones/downloads/train.csv/1 (Train dataset)

The dataset provided was 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.

The dataset contains over 560 features, of which the following are the main features :

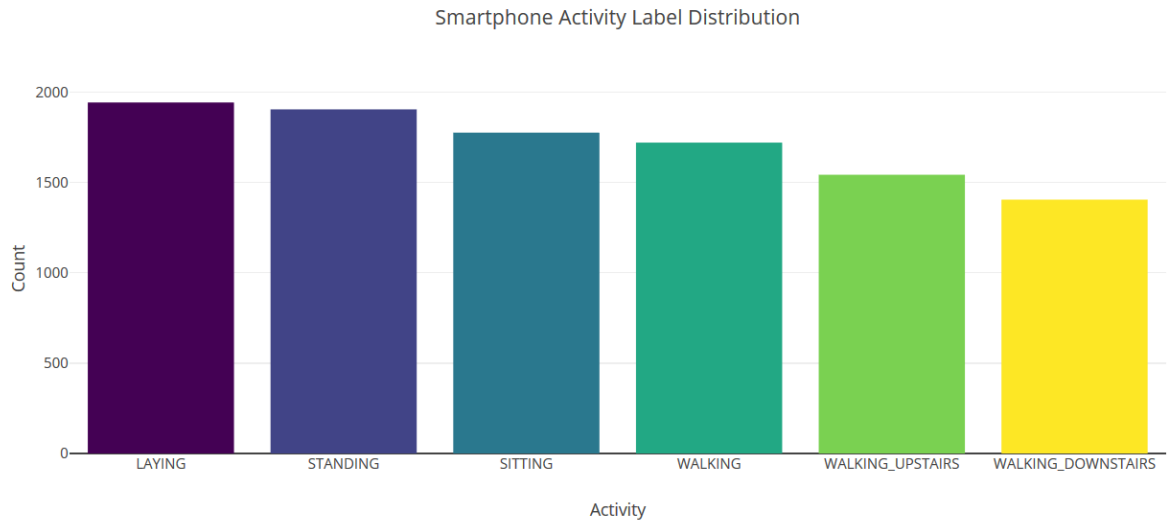
1. Subject ID
2. angle
3. fBodyAcc
4. fBodyGyro
5. fBodyAccJerk
6. tGravityAcc
7. tBodyAcc
8. tBodyGyroJerk
9. tBodyGyro
10. tBodyAccJerk
11. tBodyAccMag
12. tGravityAccMag
13. tBodyAccJerkMag
14. tBodyGyroMag
15. tBodyGyroJerkMag
16. fBodyAccMag
17. fBodyBodyAccJerkMag
18. fBodyBodyGyroMag
19. fBodyBodyGyroJerkMag

The Classes to which the activities are classified are:

1. Laying
2. Standing
3. Sitting
4. Walking
5. Walking upstairs
6. Walking downstairs

Data Distribution

The distribution of data points across these classes are as below:



TECHNIQUES

PRE-PROCESSING

Our dataset includes the above-mentioned main features. But these main features have multiple sub-features too.The initial pre processing was **NULL check and NA check.** With over 560 features, we have used dimensionality reduction (**PCA**) to reduce the number of features. The category labels were string, we used the **StringIndexer** to convert them to integer labels (0-5). Used **Vector assembler** and **Feature hasher** to convert all the feature to a feature vector. In particular for Naïve Bayes, we had to scale the dataset to ensure non-negative feature values using **Min-Max Scaler.**

MODEL DESIGN

We created pipelines for the following known standard multiclass classification models:

- Random Forest
- Naïve Bayes
- Decision Tree
- Logistic Regression

Decision Tree:

A decision tree is a tree-like structure in which each internal node represents an attribute, each branch represents the outcome of the decision, and each leaf node represents a class label. The paths from root to leaf represent classification rules.

In our Decision tree pipeline model, by trail and error, we found that PCA with K = 200 gave the best accuracy of 77.29%. So we went on to ensemble methods.

Databricks notebook link

Random Forest classifiers:

Random forests classifiers or random decision forests are an ensemble learning method for classification, that operates by constructing a multitude of decision trees at training time and outputting the class that is the modeof the classes of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

In our Random forest classifier pipeline, by trail and error , we found PCA with K = 100, num of trees = 1000 and feature subsetstratergies as ‘auto’ gave the best accuracy of 85.15 %. We went on to probalistic modeling.

Databricks notebook link

Naïve Bayes

Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on Bayes' theorem with naïve assumption that features are independent.

In our Naïve Bayes classifier pipeline, after scaling all the features to positive values, we found that the models accuracy of 83.61 %. Because the naïve assumption doesn't hold for our dataset, we decided to try out Logistic Regression.

Databricks notebook link

Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a dependent variable, although many more complex extensionsexist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression).

In our Logistic Regression pipleline, we used K- fold cross validator with K = 10 and we processed the entire dataset for modelling on train data and found the best accracy to be 93.79 % on test data, we cross checked for overfitting and found negative.

Databricks notebook link

The results are as follows :

```
Summary Statistics Accuracy = 0.9379029521547336
Precision(0.0) = 0.9906890130353817 Precision(1.0) = 0.9229323308270677
Precision(2.0) = 0.9124236252545825 Precision(3.0) = 0.9596774193548387
Precision(4.0) = 0.9087048832271762 Precision(5.0) = 0.9261904761904762
Recall(0.0) = 0.9962546816479401 Recall(1.0) = 0.9160447761194029
Recall(2.0) = 0.9142857142857143 Recall(3.0) = 0.9463220675944334
Recall(4.0) = 0.9087048832271762 Recall(5.0) = 0.9418886198547215
F1-Score(0.0) = 0.9934640522875817 F1-Score(1.0) = 0.9194756554307116
F1-Score(2.0) = 0.9133537206931702 F1-Score(3.0) = 0.952952952952953
F1-Score(4.0) = 0.9087048832271762 F1-Score(5.0) = 0.9339735894357742
Weighted false positive rate: 0.012265120320143711
```

Conclusion

We tested various Multi-Class Classifiers and successfully classified human activity using given dataset. Among our list of classifiers Logistic Regression gave highest accuracy and hence we conclude that it is best model for given dataset. We observed that data pre-processing techniques like principal component analysis, data scaling improved accuracy substantially. Model is trained using k fold cross-validation to get best results and avoid overfitting.

This work can be further extended by using more sensors like heart rate sensor to recognize high-level activities such as talking , eating, drinking. This will open up many prospects towards understanding and modelling human actions for various real-life applications.

Contribution of team members

Each team member contributed towards building the pipeline for each of the models above.

Akash Kumar Agile Mohan (axa175730) – Decision Tree.
Amith Kumar Matapady (axm180029) – Random Forest Classifier.
Hitesh Gupta Tumsi Ramesh (hxg170230) – Logistic Regression.
Rushikesh Shirish Kulkarni (rsk180001) – Naïve Bayes.

References

1. https://www.kaggle.com/uciml/human-activity-recognition-with-smartphones
2. https://spark.apache.org/docs/latest/ml-guide.html
3. https://spark.apache.org/docs/2.2.0/mllib-dimensionality-reduction.html
4. https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html
5. https://databricks.com/blog/2014/09/29/scalable-decision-trees-in-mllib.html
6. https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html
7. https://docs.scala-lang.org/tutorials/scala-for-java-programmers.html
8. https://machinelearningmastery.com/naive-bayes-for-machine-learning
9. https://docs.scala-lang.org/getting-started/intellij-track/building-a-scala-project-with-intellij-and-sbt.html