Human movement predictions based on cell-phone data

Jakub Langr

March 19, 2013

Contents

1 Introduction

Current cellphones have the capacity to track enormous amounts of data: everything from rotation, acceleration and other types of movement to our location using GSM or GPS. This is particularly troublesome when they are relatively unprotected against tempering and there have been numerous successful attempts at breaking into the accelerometer and remotely tracking their movement–not just what the user is doing, but even what he is typing. (TEDxMidAtlantic, 2011) The focus of this paper is to determine, just using the data gathered by Samsung S-II and its accelerometer and gyroscope what the user is doing.

The ultimate goal is to determine correctly using the 561 separate variables that the Samsung tracks which of the following six activities is the user doing: walking, walking upstairs, walking downstairs, sitting, standing, laying. The Human Activity Recognition Using Smartphones Dataset consisted of "experiments [that] have been carried out with a group of 30 volunteers within an age bracket of 19-48 years. Each person performed six activities [...] wearing a Smartphone (Samsung Galaxy S II) on the waist (UCI, 2013)

2 Statistical methods

2.1 Data

The dataset originally comes from from UCI "Human Activity Recognition Using Smartphones Dataset". The data was downloaded from an external website in RDA format and read into `R (ver 2.15.1)` under the name of `SD` from the following URL:

`https://spark-public.s3.amazonaws.com/dataanalysis/samsungData.rda`

The data then had to be cleaned–merged columns with same names–and separated into testing and training set. This separation is done to avoid overfitting and to give us estimate of how well might the user perform if we were to track a person about whom we have no clue what he or she might be doing. The training set (SDL) included what was labeled as subject 1 to 26. The test set (SDT) included subjects 27 to 30.

Given the size of the dataset and number of columns this dataset was hard to understand. Especially given that it was normalized as to only have values [-1,1], it was hard to, just using exploratory analysis find any distinguishing features. Credit also goes to Dr. Jeff Leek of Johns Hopkins University in Baltimore, MD for cleaning the dataset.

The data itself consisted of 7352 observations with 561 variables and two identifiers (i.e. subject and activity). The data tracked can be summarized–the credit for this great summary goes to Kai Xin Thia–as follows:

- 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.
- Similarly, the acceleration signal was then separated into body and gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ).
- 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 was calculated using the Euclidean norm (tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag).

In addition, the set of features that were estimated from these signals were (UCI, 2013):

- 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.

2.2 Exploratory Analysis

Given the size of the dataset and its heterogeneity it was virtually impossible to explore this data visually. Right after splitting, the data needed to be cleaned and transformed, if necessary. Cleaning turned out to be a non-issue since no data was missing and there were no outliers, so Dr. Leek did a good job. One existing issue, however, was with R itself: given some of the names of the variables included special characters (such as `fBodyAccJerk-bandsEnergy()-41,48`), R refused to model-build and so this had to be rectified. Likewise, in order to use `randomForest` we had to transform the `activity` as a factor variable. These seemed to be all the necessary transformations.

Initially, I explored few of the variables by plotting just to get a sense of the data. But overall using `summary` and `svd` plots was probably the most sensible method. However, given that we knew what we are looking for (best predictors of activity) not much exploration was really necessary. So I moved on to statistical modeling.

2.3 Statistical Modeling: Recursive Partitioning

The first statistical method I tried was recursive partitioning: it was clear from the beginning that something with the elements of machine learning will be necessary to determine the activity. Given that I used the entire dataset (although only the training set for to build my model of course!), I had sufficient number of observations to get fairly decent results. With the transformations already done, I only let R recursively select the the best variables that would predict the activity. (Dalgaard, 2008) In other words, R tried one variable after another determining which ones have a high predictive ability, only selecting the valuable ones and then I had to check that the model is robust (i.e. is not overfitted) by two ways (a) check that only the genuinely useful variables are included and that including any fewer variables would result in significantly reduced accuracy. (b) that this model has high score in terms of predictions even when I apply the same predictive model on the training set.

Ultimately I arrived at the recursively created tree (see Figure 1 in the appendix) that had decent predictive ability. For more discussion see the relevant results section. However, I wanted to test whether there is not a better way of predicting the current activity, so I tried another statistical method: random forests.

2.4 Statistical Modeling: Random Forests

This is a statistical method that is much more computationally intensive that recursive partitioning, but usually rewards the extra CPU-cycles with higher accuracy, which turned out to be the case even in this scenario. My understanding is that random forests does in essence the same process, but goes one

level deeper and bootstraps its original sample. So the whole process goes as follows (simplified): (Lynch, 2007)

1. Establish a `for` loop to do this whole procedure several times (default number of attempts is 500)
2. Within this `for` loop, we subsample, with replacement–i.e. bootsrap–and split the set into a training and testing set; we use rest of the cases to eliminate the error and overfitting
3. Still as a part of the `for` loop, we fully grow the tree
4. After the `for` loop cycle is done, we combine the predictions and prune the trees so that only a robust model is outputted.
5. The collection of models (trees) then votes on which of the models is the best one.

This process is somewhat complicated in the sense that we subsample our sample–so treat it as the original population–and so we get a robust model, because it was already tried 500 times. Ultimately this leaves us with a better classifying tree (see Figure 2 in the appendix) that has a better accuracy on the test set. More discussion in the results section.

2.5 Benchmark

Given that we are trying to build a predictive model, we have to be able to predict more than if we guessed randomly, because a model that guesses worse than that is of no use, you might as well just randomly guess. Ergo, our accuracy has to be higher than $\frac{1}{6}$, which is our baseline model, to be of any good. Hence we expect both our models to do better than $\approx 16.66\%$, which they indeed do.

3 Results

This section will present the two results separately, as to keep clear which model we are currently discussing. Each section will only include some discussion of the accuracy and the confusion matrix – i.e. the matrix showing how many times was the algorithm right and how many times was it wrong. If the algorithm has a 100% prediction accuracy, all the numers will be on the diagonal, any divergence from the diagonal means accuracy lower than 100%; the actual number will also be reported with each model. More information, including visual representations and the code, will be presented in the appendix.

Given the fact that I have used all available data for one purpose (training) or another (testing), I have ended up with fairly good models in both cases. There is already substantial evidence that more data is better than better models.

However, in case of machine learning techniques, such as the ones used above, more data *means* better models. (Alon Halevy, 2010)

3.1 Recursive Partitioning

The model yielded a reasonable accuracy of 87% and with 95% confidence interval between $(0.8519, 0.8867)$. P-value was smaller than $2.2e-16$, so the model is definitely more significant than R can even account for. Additional statistics,as well as the tree-model itself, pertaining to accuracy can be found in the appendix.

Table1: Confusion matrix of Recursive Partitioning Model

*pred*1	*laying*	*sitting*	*standing*	*walk*	*walkdown*	*walkup*
laying	293	0	0	0	0	0
sitting	0	226	55	0	0	0
standing	0	38	228	0	0	0
walk	0	0	0	220	9	27
walkdown	0	0	0	0	158	22
walkup	0	0	0	9	33	167

Ergo the model is significantly better than random guessing and for many activities we had been able to distinguish more than 90% accurately. As is apparent from the table, there are essentially two groups of activities: one that has to do anything with walking and the rest. Very understandably the model has some problems distinguishing between them.

3.2 Random Forest

This model yielded even better accuracy of 95% with 95% confidence interval between $(0.9386, 0.9613)$, which is an amazing level of accuracy. P-value was again at the level of accuracy of R (i.e. $2.2e-16$ or below). More statistics can be found in the appendix, as well as the tree.

Table2: Confusion matrix of Random Forest Model

predVal	*laying*	*sitting*	*standing*	*walk*	*walkdown*	*walkup*
laying	293	0	0	0	0	0
sitting	0	226	27	0	0	0
standing	0	38	256	0	0	0
walk	0	0	0	228	2	1
walkdown	0	0	0	0	194	0
walkup	0	0	0	1	4	215

Again, we can see that the same two groups appear and that the difficulty of distinguishing between different kinds of walking and other activities is not easy. Overall, this model performs at least as good as, and in many cases better than the recursive partitioning model. While more computationally intensive, this model is still more more useful and the accuracy difference of 8 % is worth the extra minute or two.

3.3 Confounders

Given that the model creation was, for the most part, automated, it is hard to give a qualified opinion on the confounders. In addition, I have little information about how the data was actually gathered (other than what was written on the website) so it is difficult to give a qualified opinion of what is actually going on. The margins can definitely be improved, but a much more in-depth analysis would be required to unravel and get rid of potential confounders, if there are any. The initial analysis suggests that there are none, given the fairly good performance of the random forest model.

4 Conclusion

In the end, the random forest model has proven itself to be much better than the recursive partitioning model. For the random forest model the error margins are quite understandable–the model overall is performing fairly well–and even though better models are almost certainly feasible, I doubt that we can improve beyond 97-98 % accuracy. With human activity and diversity of how we behave there are bound to be errors and I would be very leery of models with better accuracy, because they might just be overfitted. To improve the model, we would need perhaps a whole new approach to the prediction problem. Of course, this model might perform badly in cases the data was gathered in a completely different manner and as with all research, this model does not attain universal applicability, since this data might have been biased in a certain way at the point of gathering. Hence this model has preformed well and is somewhat robust, but more tests would be needed in order for it to attain more reliability.

5 References

References

Dalgaard, P. (2008): *Introductory Statistics with R*, Statistics and Computing. Springer.

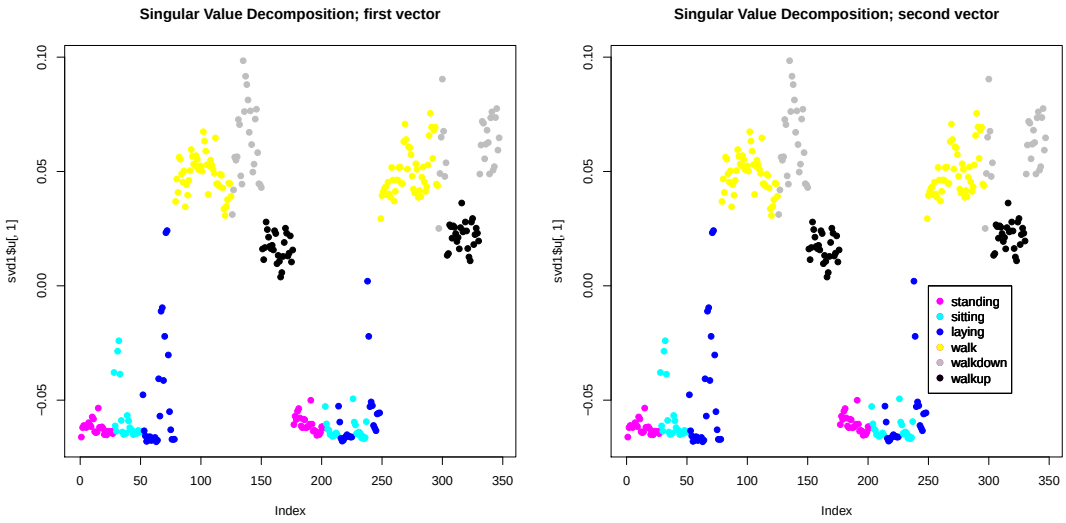
Halevy, A. (2010): "The Unreasonable Effectiveness of Data," in *West Virginia University: Department of Computer Science*.

Lynch, S. (2007): *Introduction to Applied Bayesian Statistics and Estimation for Social Scientists*, Statistics for social science and public policy. Springer Science+Business Media, LLC.

Rubin, A. (2011): "All your devices can be hacked," in *TEDxMidAtlantic*. TED.

UCI (2013): "Human Activity Recognition Using Smartphones Dataset.," .

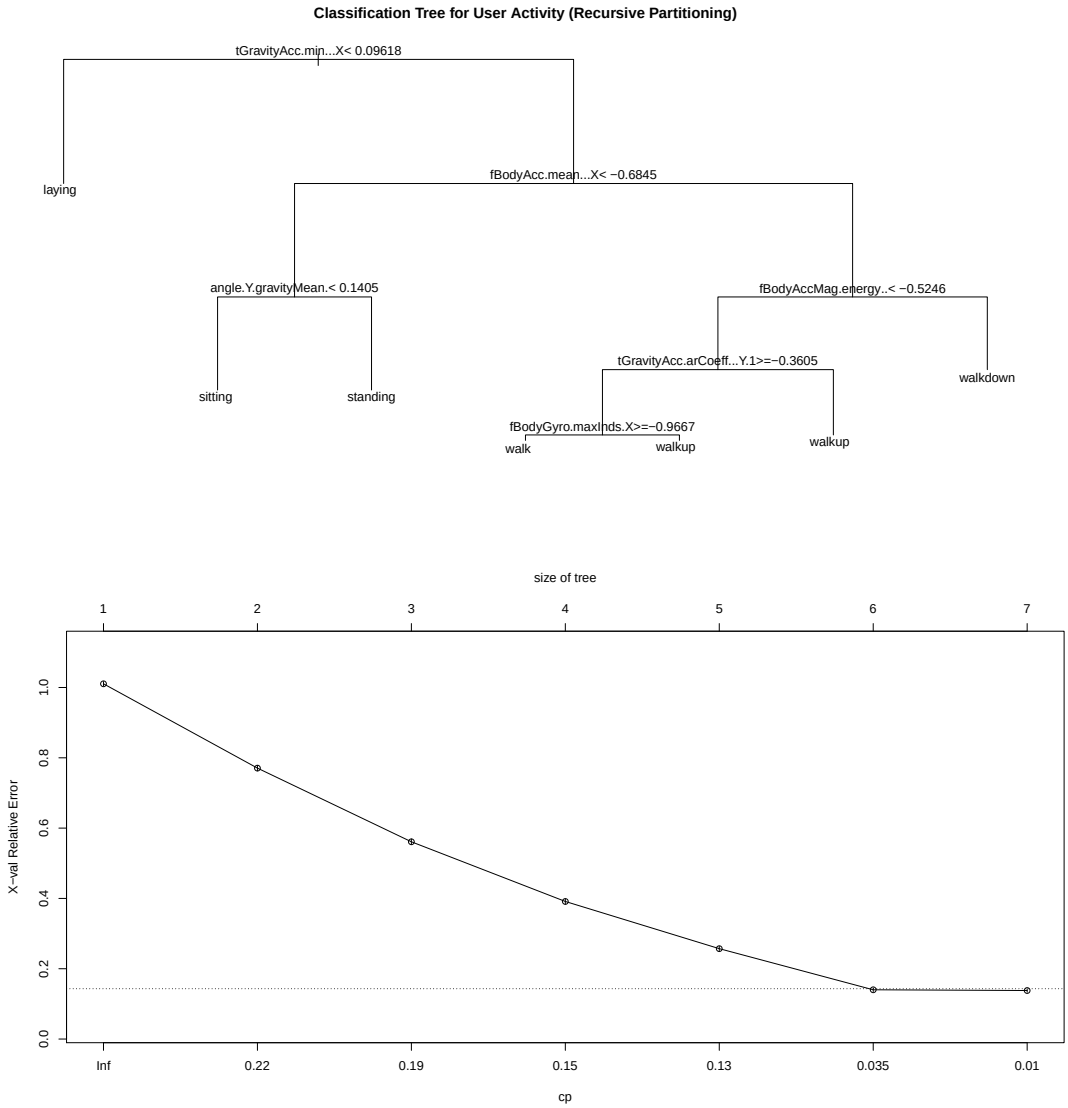
6 Appendix A: Figures



This is to demonstrate that even if we consider all the combinations of capturing as much data as we can by a single identifier (i.e. perform the SVD), we still cannot effectively–visually–separate different kinds of walking and the other three activities also stick together.

6.1 Recursive Partitioning

FIGURE 1 This is the model derived by recursive partitioning.



This is to show error with recursive partitioning using different number of splits in the tree.

6.2 Random Forest

7 Appendix B: Additional statistical information

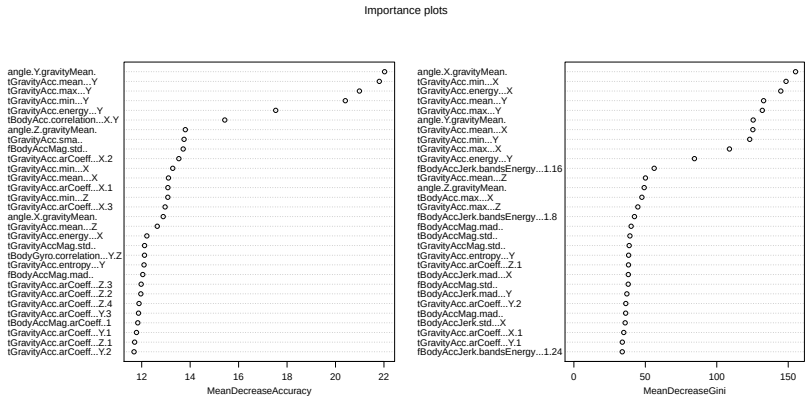
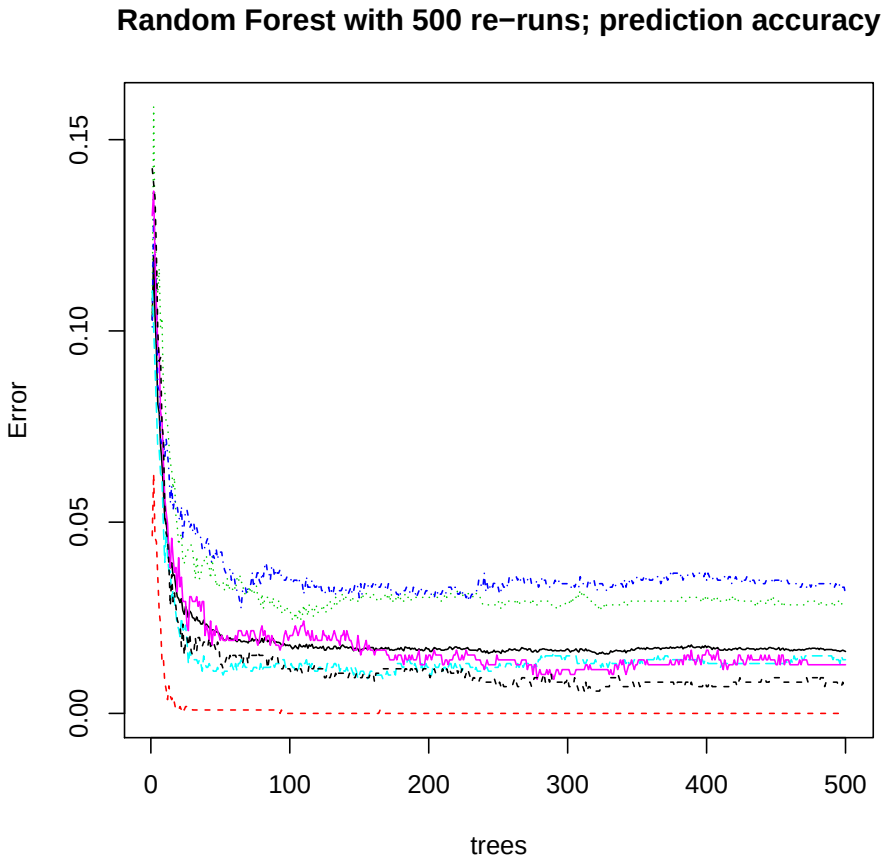


FIGURE 2
This are the relative strengths of contributions of different variables to the random forest model; as you can see they are not that much different, but nevertheless random forest model yields 95 % accuracy.



This shows how the error of the random forest changes with increasing number of runs.

7.1 Recursive Partitioning

```
                     Class: laying Class: sitting Class: standing
Sensitivity                 1.0000         0.8561          0.8057
Specificity                 1.0000         0.9550          0.9684
Pos Pred Value              1.0000         0.8043          0.8571
Neg Pred Value              1.0000         0.9684          0.9549
Prevalence                  0.1973         0.1778          0.1906
Detection Rate              0.1973         0.1522          0.1535
Detection Prevalence        0.1973         0.1892          0.1791

                     Class: walk Class: walkdown Class: walkup
```

Sensitivity	0.9607	0.7900	0.7731
Specificity	0.9713	0.9829	0.9669
Pos Pred Value	0.8594	0.8778	0.7990
Neg Pred Value	0.9927	0.9678	0.9616
Prevalence	0.1542	0.1347	0.1455
Detection Rate	0.1481	0.1064	0.1125
Detection Prevalence	0.1724	0.1212	0.1407

7.2 Random Forest

	Class: laying	Class: sitting	Class: standing
Sensitivity	1.0000	0.8561	0.9046
Specificity	1.0000	0.9779	0.9684
Pos Pred Value	1.0000	0.8933	0.8707
Neg Pred Value	1.0000	0.9692	0.9773
Prevalence	0.1973	0.1778	0.1906
Detection Rate	0.1973	0.1522	0.1724
Detection Prevalence	0.1973	0.1704	0.1980
	Class: walk	Class: walkdown	Class: walkup
Sensitivity	0.9956	0.9700	0.9954
Specificity	0.9976	1.0000	0.9961
Pos Pred Value	0.9870	1.0000	0.9773
Neg Pred Value	0.9992	0.9954	0.9992
Prevalence	0.1542	0.1347	0.1455
Detection Rate	0.1535	0.1306	0.1448
Detection Prevalence	0.1556	0.1306	0.1481

8 Appendix C: code

```
library(caret)
library(rpart)
library(randomForest)
library(boot)


# Split the entire dataset into
SDL = SD[SD$subject<27,]
SDT = SD[SD$subject>26,]

# This gets rid of the duplicate col-names problem
colnames(SDL)
```

```
SDL <- data.frame(SDL)
colnames(SDL)

colnames(SDT)
SDT <- data.frame(SDT)
colnames(SDT)


# Automated model-building; recursive partitioning
ST = rpart(activity~.,data=SDL)
table(SDL$activity,predict(ST,type="class"))

pred1 = predict(ST,SDT,type='class')
##For comparsion; error on default data
##conTable = confusionMatrix(table(pred1,SDL$activity))
conTable = confusionMatrix(table(pred1,SDT$activity))

# Alterantive method for comparsion;
# start by factoring the outcome variable to allow randomForest
SDL$activity <- as.factor(SDL$activity)

# Trying randomForest with default ntry=500
SDF = randomForest(activity~.,data=SDL,proximity=T,importance=T)

# Comaparison
predVal <- predict(SDF, SDT, type="response")
confTable = confusionMatrix(table(predVal,SDT$activity))

#plot
svd1 = svd(scale(samsungData[samsungData$subject==1,-c(562,563)]))
 plot(svd1$u[,1],col=numericActivity,pch=19,main="Singular Value
Decomposition; first vector")
plot(svd1$u[,1],col=numericActivity,pch=19,main="Singular Value
Decomposition; second vector")
colors = (as.numeric(as.factor(unique(SD$activity)))+3)
legend(250,0,legend=unique(SD$activity),col=colors,pch=19)
```