Human Activity Recognition on Smartphones Using Various Classifiers

Haotian He

University of Washington
haotianh@u.washington.edu

June 13, 2013

Abstract

The final project attemps to search and apply effective methods to analyze and recognize human activties by the nertial signal data collected from the sensors in smartphones. This report discusses various possible classifiers, and compare them to seek the best performing one. Meanwhile, it explores how dimension reduction affect the classifiers' performance. Moreover, we try to find another more effective possible algorithm, SVM with different kernels, and find SVM with linear kernel is able to provide more precise performance.

I. Introduction

Smartphones grow and influence our daily lives rapidly and boardly within over 30 years. Besides the basic function of telephony, many other features in smartphones, such as multi-tasking and the deployment of various sensors, are currently incorporated to current mobile devices. It is envisioned that with those features, we are able to use smartphones to keep track of our activities, learn from them, and subsequently assist us to make better decisions regarding our future actions. Particularly, thanks to those sensors embedded by default, we are able to process inertial body signals through a supervised Machine Learning algorithm for hardware with limited resources and use them to classift and recognize a set of physical activities.

In this report, we employ various algorithms to analyze those data of accelerometer and gyroscope (the sensor signals), and then predict and recognize six human activities (walking, walking upstairs, walking downstairs, sitting, standing, and laying).

The report is structured in the following way: The description of the dataset is given in Section 2. The comparison of the analyses results employming Gaussian Naïve Bayes, LDA, QDA, Logistic Regression, K-Nearest Neighbors, Decision Trees, and Random Forest classifiers is presented in Section 3. The evaluation of the performance of each of the methods given the reduced dimension features is presented in Section 4. The analysis by employing SVM with different kernels classifier is presented in Section 5. The explanation of the best performance to distinguish between the rest state and other states using SVM with linear kernel classifier is presented in Section 6. Finally, conclusions of this research project are described in Section 7.

II. Dataset

The dataset includes all triaxial linear acceleration and triaxial angular velocity at a constant rate of 50 Hz captured by the accelerometer and gyroscope embedded in the smartphones, from which we get 561 features. These smartphones are worn on the waist of 30 volunteers with an age range of 19–48 years, during their experimented six activities (walking, walking upstairs, walking downstairs, sitting, standing, and laying). These six activities as six classes are corresponding to each data point with 561 features.

III. Multiclass Classification

We've learned several of the algorithms which are capable of multiclass prediction, namely Gaussian Naïve Bayes, LDA, QDA, Logistic Regression, K-Nearest Neighbors, Decision Trees, and Random Forest classifiers. We apply those algorithms to do the multiclass prediction for the dataset, and compare the confusion matrix of those classification results on the test data using the diffrent training classifiers, presented by confusion matrices.

The Gaussian Naïve Bayes classifier is the first classifier implemented. The theory behind it is to assume a simple probabilistic classifier based on applying Bayes' theorem with naive independence assumptions (each feature is independent to each other). Moreover, the continuous values associated with each class are distributed according to a Gaussian distribution.

To be more detailed, based on the theory Bayes' theorem,

$$P(Y = y_k | X_1, ..., X_n) = \frac{P(Y = y_k)P(X_1, ..., X_n | Y = y_k)}{\sum_j P(Y = y_j)P(X_1, ..., X_n | Y = y_j)}$$

Then, as each feature is mutually independent, we assume conditional independence among X_i's:

$$P(Y = y_k | X_1, ..., X_n) = \frac{P(Y = y_k)\prod_i P(X_i | Y = y_k)}{\sum_j P(Y = y_j)\prod_i P(X_i | Y = y_k)}$$

Then, we assume that the distribution of value of each feature of the data point follows the normal distribution with respect to each possible class of k classes, we can get the model,

$$P(X_i = x | Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}}\, e^{-\frac{1}{2}(\frac{x-\mu_{ik}}{\sigma_{ik}})^2}$$

Therefore, during the prediction, we can get each feature of the current data point, and calculate $P(Y = y_k | X_i = x)P(X = x)$ ($P(X = x)$ can be ignored) for each possible class, and we have,

$$Y \leftarrow \arg\max_{y_k} P(Y = y_k)\prod_i \mathfrak{N}(X; \mu_{ik}, \sigma_{ik})$$

Now we use the Gaussian Naïve Bayes classifier, and other classifiers implemented in scikit-learn work out-of-the box for multiclass classification. The classification results of all the classifiers for the test data are depicted by means of a confusion matrix in Tables 1-7.

Method	Gaussian Naïve Bayes implemented in HW 7						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**416**	38	42	0	0	0	*83.87*
Upstairs	9	**451**	11	0	0	0	*95.75*
Downstairs	80	83	**257**	0	0	0	*61.19*
Sitting	0	7	0	**368**	111	5	*74.95*
Standing	0	15	0	54	**455**	8	*85.53*
Laying	0	3	0	211	0	**323**	*60.15*
Precision %	*82.38*	*75.54*	*82.9*	*58.14*	*80.39*	*96.13*	**77.03**

Table 1. Confusion Matrix of the classification results on the test data using the training Gaussian Naïve Bayes classifier. Rows represent the actual class and columns the predicted class. The diagonal entries (in bold) show the number of test samples correctly classified.

Method	LDA implemented in scikit-learn						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**490**	6	0	0	0	0	*98.79*
Upstairs	11	**460**	0	0	0	0	*97.66*
Downstairs	1	14	**405**	0	0	0	*96.43*
Sitting	0	1	0	**434**	56	0	*88.39*
Standing	0	0	0	22	**510**	0	*95.86*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*97.61*	*95.63*	*100.0*	*95.18*	*90.11*	*100.0*	**96.23**

Table 2. Confusion Matrix of the classification results on the test data using the training LDA.

Method	QDA implemented in scikit-learn						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**325**	1	170	0	0	0	*65.52*
Upstairs	10	**316**	145	0	0	0	*67.09*
Downstairs	0	4	**416**	0	0	0	*99.05*
Sitting	1	7	6	**377**	96	4	*76.78*
Standing	4	5	28	34	**461**	0	*86.65*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*95.59*	*94.89*	*54.38*	*91.73*	*82.76*	*99.26*	**82.52**

Table 3. Confusion Matrix of the classification results on the test data using the training QDA.

Method	Logistic Regression implemented in scikit-learn						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**494**	0	2	0	0	0	*99.6*
Upstairs	23	**448**	0	0	0	0	*95.12*
Downstairs	4	9	**407**	0	0	0	*96.9*
Sitting	0	4	0	**432**	55	0	*87.98*
Standing	2	0	0	13	**517**	0	*97.18*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*94.46*	*97.18*	*99.51*	*97.08*	*90.38*	*100.0*	**96.2**

Table 4. Confusion Matrix of the classification results on the test data using the training Logistic Regression.

Method	K-Neighbors Classifier implemented in scikit-learn						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**486**	1	9	0	0	0	*97.98*
Upstairs	42	**426**	3	0	0	0	*90.45*
Downstairs	51	42	**327**	0	0	0	*77.86*
Sitting	0	4	0	**420**	67	0	*85.54*
Standing	0	0	0	51	**481**	0	*90.41*
Laying	0	0	0	2	1	**534**	*99.44*
Precision %	*83.94*	*90.06*	*96.46*	*88.79*	*87.61*	*100.0*	**90.74**

Table 5. Confusion Matrix of the classification results on the test data using the training K-Neighbors Classifier.

Method	Decision Tree Classifier implemented in scikit-learn						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**474**	7	15	0	0	0	*95.56*
Upstairs	62	**380**	29	0	0	0	*80.68*
Downstairs	22	47	**351**	0	0	0	*83.57*
Sitting	0	0	0	**371**	120	0	*75.56*
Standing	0	0	0	61	**471**	0	*88.53*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*84.95*	*87.56*	*88.86*	*85.88*	*79.7*	*100.0*	**87.68**

Table 6. Confusion Matrix of the classification results on the test data using the training Decision

Tree Classifier.

Method	Random Forest Classifier implemented in scikit-learn						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**476**	15	5	0	0	0	*95.97*
Upstairs	60	**399**	12	0	0	0	*84.71*
Downstairs	22	56	**342**	0	0	0	*81.43*
Sitting	0	0	0	**435**	56	0	*88.59*
Standing	0	0	0	45	**487**	0	*91.54*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*85.3*	*84.89*	*95.26*	*90.62*	*89.69*	*100.0*	**90.8**

Table 7. Confusion Matrix of the classification results on the test data using the training Random Forest Classifier.

IV. Dimension Reduction

Since this dataset is fairly high dimensional, we wonder whether it is possible to project it down to less dimensions while still preserving the accuracy or not. Thus, we use PCA to project the data down to 20 dimensions and to 50 dimensions, and then compare the performance of each of the methods of the different classifiers used in the previous section. If we only focus on the overall accuracy of each classifier, we have a comparison form as Table 8, showing the different performances.

Dimension / Classifier	Not Reduced	Reduced to 50	Reduced to 20
Gaussian Naïve Bayes	77.03	87.07	84.7
LDA	96.23	91.04	86.9
QDA	82.52	92.03	87.92
Logistic Regression	96.2	92.64	89.18
K-Neighbors Classifier	90.74	88.9	86.29
Decision Tree Classifier	87.68	83.03	81.07
Random Forest Classifier	91.48	86.9	85.99

Table 8. Comparison of the overall accuracies on the test data using various classifiers.

As Table 8 shows, the accuracies change corresponding to their changed dimensions. If their dimensions are not reduced, we find the accuracy order as $LDA > Logistic\ Regression > Random\ Forest\ Classifier > K-Neighbors\ Classifier > Decision\ Tree\ Classifier > QDA > Gaussian\ Naive\ Bayes$; if their dimensions are reduced to 50, we find the accuracy order as $Logistic\ Regression > QDA > LDA > K-Neighbors\ Classifier > Gaussian\ Naive\ Bayes > Random\ Forest\ Classifier > Decision\ Tree\ Classifier$; if their dimensions are reduced to 20, we find the accuracy order as $Logistic\ Regression > QDA > LDA > K-Neighbors\ Classifier >$

Random Forest Classifier > Gaussian Naive Bayes > Decision Tree Classifier.

Generally, except Gaussian Naïve Bayes and QDA, we find that the less dimension the classifier is reduced to, the lower accuracy it has. Here one thing we didn't expect is the performances of Gaussian Naïve Bayes and QDA. Even if their reduced-to-50-dimension accuracies are higher than those of reduced-to-20-dimension, which follows the trend we find in other classifier, but we find their non-reduced-dimension accuracies are lower than reduced-dimension accuracies. As for Gaussian Naïve Bayes, it cannot consider the mutual relationship between each feature. Thus, it probably get the statistical noise, which however PCA ignores. That's probably why they perform better when their dimension is reduced.

In order to further study their performance of dimension reduction, here we present the confusion matrices of both the surprisingly well-worked classifier Logistic Regression and the badly-worked one Decision Tree Classifier in Tables 9-12 below.

Method	Logistic Regression with Dimension Reduced to 50						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**484**	5	7	0	0	0	*97.58*
Upstairs	48	**403**	20	0	0	0	*85.56*
Downstairs	8	24	**388**	0	0	0	*92.38*
Sitting	1	2	0	**419**	69	0	*85.34*
Standing	2	2	0	29	**499**	0	*93.8*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*89.13*	*92.43*	*93.49*	*93.53*	*87.85*	*100.0*	**92.64**

Table 9. Confusion Matrix of the classification results on the test data using the training Logistic Regression with Dimension Reduced to 50.

Method	Logistic Regression with Dimension Reduced to 20						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**477**	2	17	0	0	0	*96.17*
Upstairs	22	**430**	19	0	0	0	*91.3*
Downstairs	26	38	**356**	0	0	0	*84.76*
Sitting	1	5	0	**383**	101	1	*78.0*
Standing	3	3	0	75	**451**	0	*84.77*
Laying	0	0	0	6	0	**531**	*98.88*
Precision %	*90.17*	*89.96*	*90.82*	*82.54*	*81.7*	*99.81*	**89.18**

Table 10. Confusion Matrix of the classification results on the test data using the training Logistic Regression with Dimension Reduced to 20.

Method	Decision Tree Classifier with Dimension Reduced to 50						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**444**	26	26	0	0	0	*89.52*
Upstairs	47	**379**	45	0	0	0	*80.47*
Downstairs	61	44	**315**	0	0	0	*75.0*
Sitting	0	0	0	**364**	124	3	*74.13*
Standing	0	0	0	96	**435**	1	*81.77*
Laying	0	0	0	20	7	**510**	*94.97*
Precision %	*80.43*	*84.41*	*81.61*	*75.83*	*76.86*	*99.22*	**83.03**

Table 11. Confusion Matrix of the classification results on the test data using the training Decision Tree Classifier with Dimension Reduced to 50.

Method	Decision Tree Classifier with Dimension Reduced to 20						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**437**	31	28	0	0	0	*88.1*
Upstairs	61	**376**	34	0	0	0	*79.83*
Downstairs	67	62	**291**	0	0	0	*69.29*
Sitting	0	0	0	**361**	127	3	*73.52*
Standing	0	0	0	109	**423**	0	*79.51*
Laying	0	0	0	33	3	**501**	*93.3*
Precision %	*77.35*	*80.17*	*82.44*	*71.77*	*76.49*	*99.4*	**81.07**

Table 12. Confusion Matrix of the classification results on the test data using the training Decision Tree Classifier with Dimension Reduced to 20.

As those tables shown above, we find Logistic Regression surprisingly works well on different reduced dimension, while Decision Tree Classifier works badly in this experiment. This is actually another part what I didn't expect before, as I find Logistic Regression works not that outstandingly before. One of the possible reasons may be that it depends on different dataset and data patterns.

V. Different Kernels for Support Vector Machine

Support Vector Machine classifier does not have prior knowledge of the problem but learns about it during training. The major advantange of SVM is its generalization capability. Thus feature makes it better than most of the other classifiers. SVM works equally well for both linearly separable data as well as non-linearly separable data. It is able to classify unknown data points with high accuracy as it works on the concept of maximum margin hyperplane.

For Support Vector Machine classifier, if the kernel changes, we assume that its performance probably would change as well. Thus, we look at several different kernels to figure out what the accuracies of this classifier are. As we know, we have the kernels as "linear", "polynomial", and "rbf" (we here will not discuss about other kernel functions), and "linear" tends to work generally well with linearly separable data, while "polynomial" tends to work generally well with non-linearly or non-separable data. Furthermore, "rbf" tends to perform well with more generalized data.

Based on the knowledge above, we first decompose the test data to 2-dimension in order to plot the classification showing as Figure 1 below. From Figure 1, we can easily figure out that the dataset is more linear separable, and we assume that SVM with linear kernel function will get the highest accuracy among other two functions.

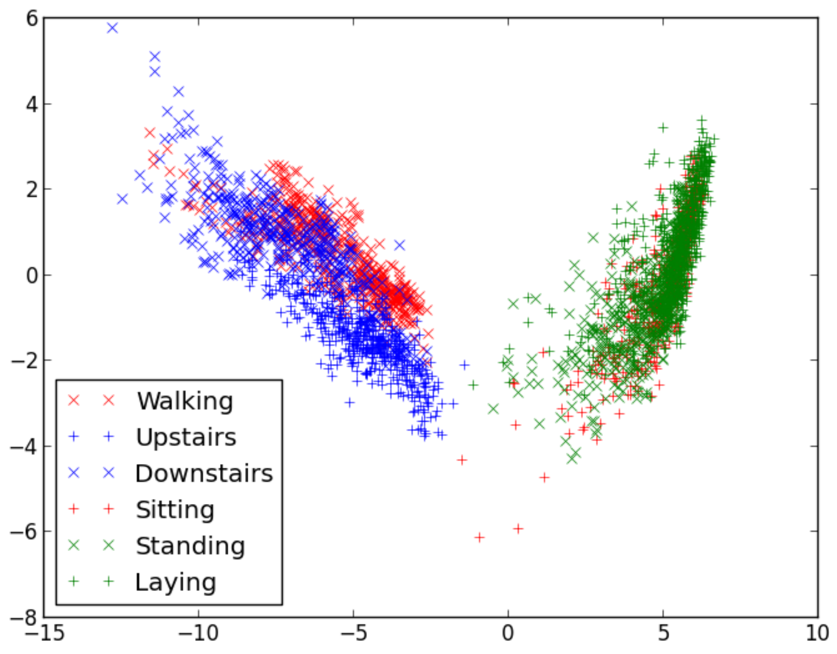

Figure 1. The scatter plot of the classification results on the test data reduced to 2 dimension.

We get the confusion matrices of the classification results as Tables 13-15 showing below. As the results show, it proves our assumption that the linear kernel function work the best and gives the highest accuracy among all the kernel functions.

Method	SVM with kernel = 'rbf'						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**492**	0	4	0	0	0	*99.19*
Upstairs	17	**452**	2	0	0	0	*95.97*
Downstairs	13	29	**378**	0	0	0	*90.0*
Sitting	0	2	0	**424**	65	0	*86.35*
Standing	0	0	0	44	**488**	0	*91.73*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*94.25*	*93.58*	*98.44*	*90.6*	*88.25*	*100.0*	**94.03**

Table 13. Confusion Matrix of the classification results on the test data using the training SVM with kernel = 'rbf'.

Method	SVM with kernel = 'linear'						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**492**	1	3	0	0	0	*99.19*
Upstairs	18	**451**	2	0	0	0	*95.75*
Downstairs	4	6	**410**	0	0	0	*97.62*
Sitting	0	2	0	**435**	54	0	*88.59*
Standing	0	0	0	16	**516**	0	*96.99*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*95.72*	*98.04*	*98.8*	*96.45*	*90.53*	*100.0*	**96.4**

Table 14. Confusion Matrix of the classification results on the test data using the training SVM with kernel = 'linear'.

Method	SVM with kernel = 'poly'						
Activity	Walking	Upstairs	Downstairs	Sitting	Standing	Laying	*Recall %*
Walking	**491**	0	5	0	0	0	*98.99*
Upstairs	38	**424**	9	0	0	0	*90.02*
Downstairs	54	44	**322**	0	0	0	*76.67*
Sitting	0	4	0	**410**	77	0	*83.5*
Standing	0	1	0	41	**490**	0	*92.11*
Laying	0	0	0	0	0	**537**	*100.0*
Precision %	*95.53*	*92.17*	*77.59*	*90.91*	*85.96*	*100.0*	**90.74**

Table 15. Confusion Matrix of the classification results on the test data using the training SVM

with kernel = 'poly'.

VI. Best Binary Classifier for Movement

According to all the classifiers we've learned and experimented till now, we are able get the form for the comparison of (1) the overall accuracies, (2) the overall error rates, and (3) the error rates of classifying 6, which is shown in Table 16 below.

Classifier	Overall Accurary	Overall Error Rate	Error Rate of 6
SVM with kernel = 'linear'	96.4	3.60	0.0
Gaussian Naïve Bayes	77.03	22.97	7.26
LDA	96.23	3.77	0.0
QDA	82.52	17.48	0.0
Logistic Regression	96.2	3.80	0.0
K-Neighbors Classifier	90.74	9.26	0.10
Decision Tree Classifier	87.68	12.32	0.0
Random Forest Classifier	91.48	9.09	0.0

Table 16. Comparison table of the classification results on the test data using the training related classifiers.

As Table 16 shows above, we can find that among all the classifiers we use in the previous sections, the SVM with linear kernel function has the best performance for all the accuracies and error rates. There are some reasons for this performance. First, SVM with linear kernel is flexible and is able to reduce overfitting. Second, actually due to the dataset we have, SVM with linear kernel performs surprisingly well for this dataset as linear separable data. However, probably for other quite different data, it would not have such good performance.

VII. Conclusion

In this report, we discusses and examine various classifiers (Gaussian Naïve Bayes, LDA, QDA, Logistic Regression, K-Nearest Neighbors, Decision Trees, and Random Forest classifiers) and their performance in classifying human activities using smartphones, and find that their accuracies are quite vary from each other.

We also examine how dimension reduction affect the classifiers' performance, and find that for most classifiers, the less dimension they are reduced to, the lower accuracies they have. At the same time, we surprisingly find Gaussian Naïve Bayes has a quite different trend from others, and conclude its different theory and feature lead to this different results.

Furthermore, we test one more classifier, SVM with different kernels, and surprisingly find SVM with linear kernel provides a more precise accuracy than all the other classifiers.

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. (2012). Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support

Vector Machine. *International Workshop of Ambient Assisted Living (IWAAL 2012)*. Victoria-Gasteriz, Spain. Dec. 2012.

[2] Ratnanjali Sood, Student Member SSI and Satish Kumar, Life Member SSI. (2008). The Effect of Kernel Function on Classification. *XXXII National Systems Conference (NSC 2008)*. Roorkee, India. December 17-19, 2008.