Human Activity Recognition using Wearable Devices Sensor Data

Zhongyan Wu zhowu@stanford.edu Shutong Zhang zhangst@stanford.edu
Chenying Zhang czhang3@stanford.edu

Abstract

Wearable devices are getting increasingly popular nowadays as the technology products become smaller, more energy efficient and as more sensors are available on our wrist. By wearing these devices everyday, we could easily collect mega-bytes of data each day. In spite of the abundance of available data from these sensors, there isn't too much information we can tell from these raw data about what we have done each day. In this project, our goal is to recognize patterns from these raw data, and extract useful information about the user's daily activities. We used feature extraction and selection techniques to process the raw data, and then applied various learning algorithms. The results are quite impressive as compared to previous work, as we have significantly reduced the number of features required and therefore made the process more practical on mobile devices, where energy consumption is a paramount concern.

1. Introduction

Wearable devices are getting more and more popular recently, which presents a convenient and portable way to record physiological data from users. Thus it is more possible to collect physical-related data and perform further analysis. All these data will be generated and made available to better understand the activities users are performing in and could be used to monitor health or recreational activities.

However, the data directly come from these devices are raw data and couldn't provide much information about human's activities. In this project, we use data come from smartphones' built-in sensors (accelerometer and gyroscope) of 30 users and evaluate several machine learning algorithms to recognize human activities such as walking, sitting, standing, etc. Thus, we use two strategies to gain information from raw data:

1. Using feature selection strategy to select from given features.
2. Directly extracting physical features from raw data.

An accurate prediction of human's activity through physical-related data could be used to create mobile applications based on different movement and provide useful suggestions for users. In addition, people can use these results to keep record of self-activity.

2. Related Work

There has been a variety of works trying to identify human activities based on different kinds of data source [5]. Some data comes form environmental sensors, which may not give much information about human activity as body-worn sensors [2]. Some data comes from specialized devices, which are expensive, uncomfortable for users and become a burden to the user during regular activity [3].
Thus more and more researches are focusing on using wearable devices' data to classify human activity[4] [1]. However, usually they only pay attention to activities and ignore postural transitions, the process between two activities. In this project, we used a dataset obtained from smartphones containing data related to both activities and postural transitions.

3. Problem Formulation

In this part, we will formally described our problem formulation here. Our problem includes two essiential parts:

1. **Extract useful features F from Raw sensor data R for training examples $X_{train} = \{x_1, x_2, ..., x_m\}$.**

   We use the notation $R_{train} = \{R_1, R_2, ..., R_m\}$ to denote the raw data, where we directly collected from the wearable sensors and m is the number of training data. Then, we can extract useful information from the raw data R_{train}, and project them into a new feature space to obtain the features $F_{train} = \{F_1, F_2, ..., F_m\}$ for each training example $x_i, i = 1, 2, ..., m$.

2. **Use the features F to predict the labels for test data X_{test}.**

Note that we use the same feature space for X_{train} and X_{test}, thus we could compute the features for all the test data $F_{test} = \{F_1, F_2, ..., F_{m_{test}}\}$. Based on the extracted features we get on the test examples, we are able to build classification models to classify them. We will use $Y_{predict} = \{y'_1, y'_2, ..., y'_{m_{test}}\}$ to denote the predict labels for all the test data.

4. Data Preprocessing

4.1. Dataset

We obtained our dataset from UC Irvine Machine Learning Repository [4]. This dataset was collected from participants wearing a smartphone (Samsung Galaxy S II) on the waist during the experiment execution.
This dataset involves 30 volunteers from 19 to 48 years. Data related to six basic activities including three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs) were recorded, so were data about six kinds of postural transitions between the three static postures. To use these data for machine learning, 70% of the volunteers were selected for generating the training data and 30% for the test data randomly.

4.2. Data Preprocessing

4.2.1 HAPT Feature

Function	Description
mean	Mean value
std	Standard deviation
mad	Median absolute value
max	Largest values in array
min	Smallest value in array
sma	Signal magnitude area
energy	Average sum of the squares
iqr	Interquartile range
entropy	Signal Entropy
arCoeff	Autorregresion coefficients
correlation	Correlation coefficient
maxFreqInd	Largest frequency component
meanFreq	Frequency signal weighted average
skewness	Frequency signal Skewness
kurtosis	Frequency signal Kurtosis
energyBand	Energy of a frequency interval
angle	Angle between two vectors

Table 1. List of measures for computing feature vectors

Raw data contains triaxial linear acceleration and angular velocity signals collected through the smartphone accelerometer and gyroscope at a sampling rate of 50Hz of each volunteer. The time signals were sampled in fixed-width sliding windows of 2.56 seconds. A vector of 561 features provided by [4] was mapped using the functions in **Table 1** for a single sampling window.

4.2.2 Feature Selection

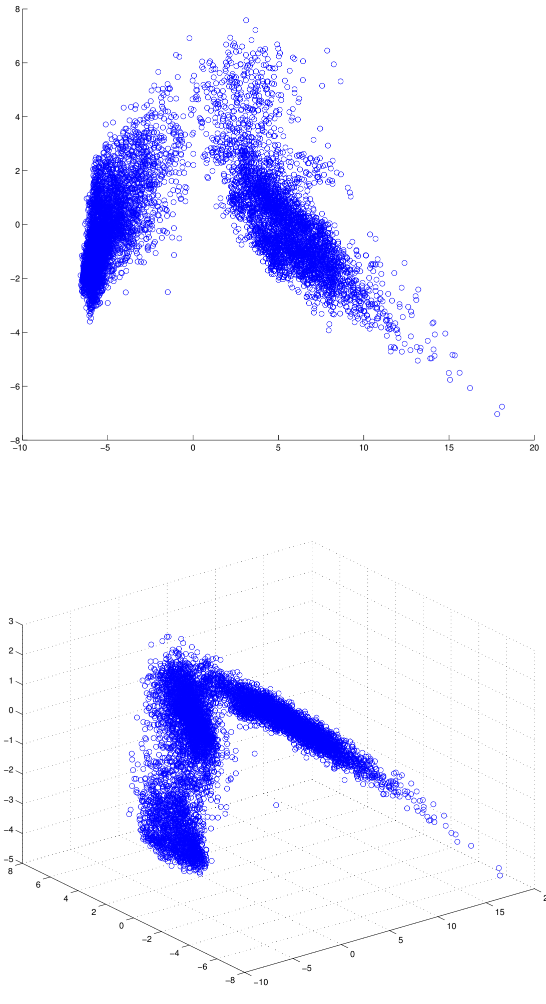

Figure 1. 2D and 3D PCA Component Visualization

These 561 features were given by UC Irvine Machine Learning Repository dataset together with the raw data. These extracted data were used on several models (see the Method Section) and they performed very well (see the Result Section), but we thought there may be too many features and there may be some redundant information, thus principal components analysis (PCA) is performed on the train and test matrix X to make feature selections. PCA generates a new set of variables, called principal components. Each principal component is a linear combination of the original variables. All the principal components are orthogonal to each other, so there is no redundant information. The full set of principal components is as large as the original set of variables (thus there are totally 561 variables). But it is commonplace for the sum of the variances of the first few principal components to exceed 80% of the total variance of the original data. In our case, the first 30 components contribute 88.18% of the total variance. Therefore, 561 features are projected to a 30-dimension space and these new selected features were used to perform activity classification. However, the result is less than satisfactory (see the Result Section) and we decided to generate the same amount of features from the raw data ourselves, which is described later. In addition, to develop a deeper understanding of the driving forces that generated the original data and visualize data, we drew the plots of the first two and three new variables generated by PCA respectively.

Since we have 12 classes in total, it's hard to set color to these points in the above plots based on their classifiers. However, we can see clearly that there are no 12 distinct clusters and these data points are mixed together, which we thought may be the reason why the results obtained by using PCA feature selection data did not give an accuracy estimate.

4.2.3 Feature Extraction

The data set provided by HAPT includes 561 features, in the previous section we used PCA to select 30 features from those features and achieved a correctness rate of 40.8%, which is less than satisfactory. Therefore, we performed our own feature extraction process to extract a same amount of features and achieved a correctness rate of 84.93%.

The features selected for this database come from the accelerometer and gyroscope 3-axial raw signals tAcc and tGyro. These time domain signals were captured at a constant rate of 50 Hz. These signals were used to estimate variables of the feature vector for each pattern, the set of variables that were estimated from these signals are: max(tAcc/tGyro), min(tAcc/tGyro), mean(tAcc/tGyro), std(tAcc/tGyro), mad(tAcc/tGyro)

The functions we used was performed on each dimension of the triaxial linear acceleration and angular velocity signals. Therefore, we have a total of 30 features ((3 linear + 3 angular) * 5 functions).

The training and testing datasets were randomly selected with a 70/30 percentage from the entire raw dataset.

	Selected Features
max(tacc)	Largest value of accelerometer
min(tacc)	Smallest value
mean(tacc)	Mean value
std(tacc)	Standard deviation
mad(tacc)	Median absolute deviation
max(tgyro)	Largest value in gyro
mean(tgyro)	Mean value
std(tgyro)	Standard deviation
mad(tgyro)	Standard deviation

Table 2. List of measures for computing feature vectors

5. Methods

We conducted five different multi-classification models to perform human activity recognition.

1. **Naive Bayes**
   We chose Naive Bayes as the baseline model to get a rough idea of how is the training process going.
   In a bayes classifier, we assign a class label $\hat{y} = C_k$ for some k as follows:
   $$\hat{y} = \text{argmax}_{k\in\{1,\dots,K\}} p(C_k) \prod_{i=1}^{n} p(x_i|C_k)$$

2. **KNN**
   Given a positive integer K and a test observation x_0, the KNN classifier first identifies the neighbors K points in the training data that are closest to x_0, represented by N_0. It then estimates the conditional probability for class j as the fraction of points in N_0 whose response values equal j:
   $$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i\in N_0} I(y_i = j)$$

3. **SVM**
   Support Vector Machines [1] is a supervised learning algorithm that classifies objects based on the support vectors of a dataset or points lie closest to the decision boundary. SVM maximize the distance between support vectors and the decision boundary.
   The objective function is:
   $$\begin{aligned} \min_{\gamma,w,b} \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, i = 1, 2, ..., m \\ & \xi_i \geq 0, i = 1, 2, ..., m \end{aligned}$$

4. **Softmax**
   In Softmax model, the probability that an input vector

x is a member of a class C_k is defined as followed:

$$P(Y = C_k|x, W, b) = \frac{e^{W_{C_k}x+b_{C_k}}}{\sum_j e^{W_j x + b_j}}$$

The prediction label $\hat{y}$ should be:

$$\hat{y} = \text{argmax}_{C_k} P(Y = C_k|x, W, b)$$

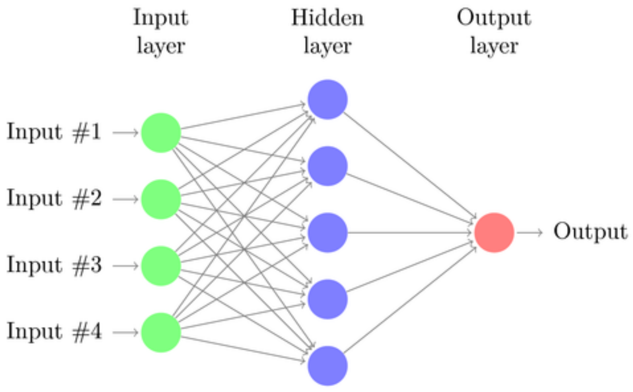

Figure 2. Structure of MLP

5. **Multi-Layer Perceptron**
Multi-Layer Perceptron can be seen as a fully-connected neural network with one input layer, one hidden layer and one output layer. A one-hidden-layer MLP can be formalized as follow:
It is a function f, $R^D \to R^L$, where D is the size of input vector x and L is the size of the output vector $f(x)$, such that, in matrix notation:

$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x)))$$

with bias vectors $b^{(1)}, b^{(2)}$; weight matrices $W^{(1)}, W^{(2)}$ and activation functions G and s.

6. Experiments and Evaluation

6.1. Evaluation Matrices

For evaluation, we use accuracy to represent the performance of the models. Also, we have compute the precision, recall and F-score for each class, which can show us the performance of our classifier on every activity category.

$$Accracy(X_{test}) = \sum_{i=1}^{m_{test}} 1\{y_i == y_i\prime\}$$

6.2. Experiments Results

We use different percentage 50%, 60%, 70%, 80%, 90%,100% of the training dataset to train Naive Bayes, KNN, SVM, Softmax and Multi-Layer Perceptron model. And we show the error of test data below to compare the influence of the number of training data to the performance of the model.

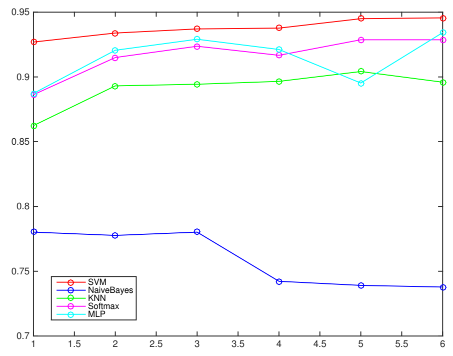

Figure 3. Classification Accuracy for Naive Bayes, KNN, SVM, Softmax and Multi-Layer Perceptron.

Table 3. Classification Accuracy Percentages for different classification models.

Model	50%	60%	70%	80%	90%	100%
Bayes	0.7802	0.7777	0.7802	0.7423	0.7391	0.7378
KNN	0.8624	0.8931	0.8933	0.8966	0.9042	0.8960
SVM	0.9269	0.9339	0.9371	0.9377	0.9450	0.9456
Softmax	0.8863	0.9150	0.9237	0.9167	0.9287	0.9287
MLP	0.8873	0.9206	0.9291	0.9212	0.8953	0.9342

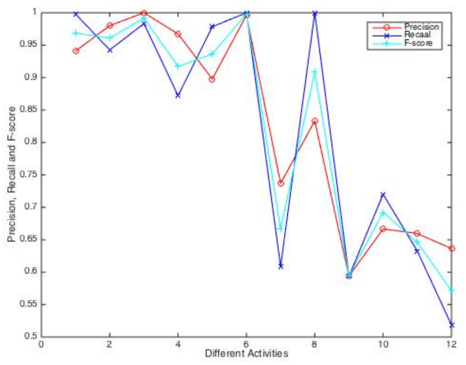

Figure 4. Precision, Recall and F-score for the existing 12 activities catagories using SVM.

From Figure 3, we can see that KNN, SVM, Softmax and Multi-Layer Perceptron performs much better than

Naive Bayes, the former achieve and accuracy around the 90th percentile where Naive Bayes was only able to get to the 70th percentile. Among all the classifiers , multi-class SVM reaches the best performance, which reduce the error to about 6 percentile. Meanwhile, the accuracy of Naive Bayes decreases when the percentage of training data is more than 70%. The decrement here may illustrates that there is some noise in the dataset, which may cause the overfitting problem.

The results of SVM is rather satisfatory, and by looking at Figure 4 we can see its Precision, Recall, and F-force shared a similar pattern and was quite high for activities 1-6. However, the results for activities from 6-12 has higher deviation, where the highest point is for activity 8 of 80% to 95%, and the lowest was activity 12 where the percentile is only about 55%. Note that activities 1-6 is pure activity, while activities 6-12 is the transition of different activities. Thus, maybe we should consider using more than one classifiers for these 12 activities and to see if the results can be improved. Besides using more than classifiers, what we can try is to change the features for activities 6-12. Since they are transition activities, i.e. from sitting to walking, we could change the feature to be the difference between the original features.

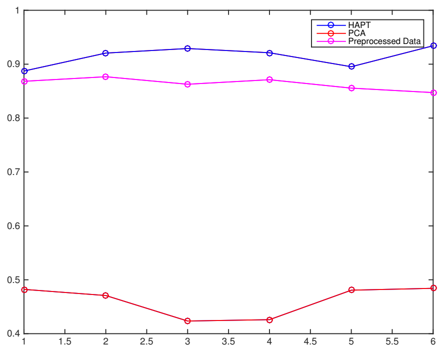

Figure 5. Classification Accuracy for HAPT features, PCA features and our preprocessed Data using MLP

Table 4. Classification Accuracy Percentages for different classification models using KNN.

Model	50%	60%	70%	80%	90%	100%
HAPT	0.8624	0.8931	0.8943	0.8965	0.9041	0.8959
PCA	0.4079	0.4029	0.4073	0.4098	0.4089	0.4092
Preprocessed	0.8493	0.8301	0.8246	0.8438	0.8438	0.8328

Figure 5 and Figure 6 illustrates the effectiveness of different features, including the original HAPT features, the features after using PCA to select and the features we extract ourselves. Comparing different methods of processing the input data, we can tell from Figure 5 that

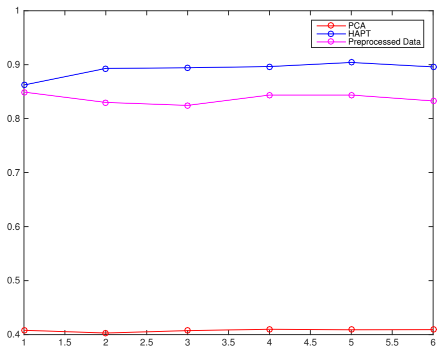

Figure 6. Classification Accuracy for HAPT features, PCA features and our preprocessed Data using KNN

we were able to achieve a very good result by performing the 10 selected function on raw data. It was able to achieve around 84% accuracy with only 30 features. In contrast to the 561 features introduced by HATP and this was a very satisfying result, and it perform outstandingly as compared to the 30 features selected by PCA from the 561 HAPT features. Of course, the original HAPT features have the best performance among these three features, but with the fact that these features has 561 dimensions, we are able to gain good performance with only 30 dimensional features.

By comparing Figure 5 and 6, we can draw the conclusion that result of KNN and MLP achieved a similar accuracy on our pre-processed data.

7. Conclusion and Future Work

We have worked on feature extraction, feature seletion and classifiacion methods on the problem of human activity recognition. After conducting experiments on real-world the sensor data from smart phone, the results have shown the effectiveness of our chosen methods.

In our future work, we can explore the following aspects:

1. We need to try more methods to improve our result's accuracy.

2. We can collect data ourselves, using different kinds of smartphones or sensors on different parts of the body, such as chest, wrist, waist and ankle

3. We can try to classify more kinds of motion pattern, such as jumping, swimming and playing basketball. We only trained models based on 12 basic motion patterns right now.

4. We can apply our results of motion recognition to health monitoring and weight control.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *Ambient assisted living and home care*, pages 216–223. Springer, 2012.

[2] R. Poppe. Vision-based human motion analysis: An overview. *Computer vision and image understanding*, 108(1):4–18, 2007.

[3] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *AAAI*, volume 5, pages 1541–1546, 2005.

[4] J.-L. Reyes-Ortiz, L. Oneto, A. Ghio, A. Samá, D. Anguita, and X. Parra. Human activity recognition on smartphones with awareness of basic activities and postural transitions. In *Artificial Neural Networks and Machine Learning–ICANN 2014*, pages 177–184. Springer, 2014.

[5] C.-C. Yang and Y.-L. Hsu. A review of accelerometry-based wearable motion detectors for physical activity monitoring. *Sensors*, 10(8):7772–7788, 2010.