# Which Device Knows Your Activity Better, Google Glass, Mobile Phones or Pebble Smartwatch?

Ben Zhang
UC Berkeley EECS
benzh@eecs.berkeley.edu

Yuxun Zhou
UC Berkeley EECS
yxzhou@eecs.berkeley.edu

## ABSTRACT

We focus on activity recognition problem on commercial mobile platforms including Android phones, Google Glass and Pebble Smartwatch. Because of their different positions on human body, their performances would differ; and this project aims to investigate 1) which device is better for which activity, 2) the quantitative differnece. We describe the whole process of system implementation, data collection, feature extraction and activity recognition, and present our findings of the performance for each device.

## Keywords

Activity Recognition; Google Glass; Android Phones; Pebble; Multinomial Logistic Regression; HMM

## 1. INTRODUCTION

Mobile phones nowadays no longer just a "phone". They are supporting a more diverse set of services including Internet access, gaming console, personal area hubs, photography, etc. Also more powerful sensors, such as GPS, accelerometer, gyroscope, compass, light sensor, proximity sensor, barometer, are integrated to mobile phones. What's more, the increasing computing power. According to NASA, *"your cell phone has more computing power than the computers used during the Apollo era"*. These changes have made mobile phones, the always-available devices, become the central hub for human to interact with the rest of the world. And we are seeing more development along the line. For example, Google's Project Tango[1] has designed a phone that comes with motion tracking camera, depth sensor, 2x computer vision processors to push the boundary of vision-based applications on mobile phones.

Among all the exciting new applications enabled by the advance of mobile phone platforms, there is a set of applications known as **Quantified Self (QS)**[5]. People are continuously acquiring data that can be used to assess a person's daily life. Examples of such data includes food consumed, quality of surrounding air, mood, daily activity, location history. Though we are still far from automatically collecting all the data (most applications require manual input for food consumed and mood), new technologies and data mining algorithms are being developed to leverage existing sensor data to infer meaningful events that human can understand. For example, logging the raw accelerometer data is not useful unless the data are parsed, analyzed and used in applications like games or activity logging. The the process of *parse, analyze, infer* is where many statistical inference algorithms can help in broadening the application scope. In this project, we are interested in the activity recognition application that recognizes a humna's physical activity from the sensors available on mobile phones. Previous work [3, 1, 2] have studied such topic and there exist many commercial applications (such as Moves[2]). One observation that we have is that mobile phones, given their pervasiveness, can be placed at arbitrary position and orientation throughout normal uses. On the other hand, the emerging wearable computing devices, such as Google Glass, Pebble Smartwatch, Samsung Gear, FitBit, Qualcomm Toq, are offering new opportunities for activity recognition because of their unique position on human body. Their positions add another dimension for the activity inference problem and we seek to make a comparison among them in this project and quantify the difference.

Though there are a wide choice of wearable computing platform to use, we choose Google Glass, Pebble Smartwatch as case study for this project because of their accessibility (the authors happen to have them in hand). We use another Android smart phone – Samsung Galaxy S II, for the mobile phone data collection. To proceed, we first wrote two data collection program for both Android platform (for Android phones and Google Glass) and Pebble platform (for Pebble Smartwatch). To be succinct and without introducing ambiguity, throughout this paper, we will we use *Phone* for mobile phones, and *Glass*, *Pebble* for Google Glass and Pebble Smartwatch, respetively.

We then perform a series of actions (including standing, walking, running, climbing upstairs, climbing downstairs) and use our application to collect traces of data. A video is taken to obtain the ground truth. For this report, we are using two datasets that were collected at Soda Hall, Berkeley on May 10th. One is used for training, and the other as test set. Exploratary data analysis (EDA) is performed first, followed by feature extraction and classification using

[1] https://www.google.com/atap/projecttango/

[2] Moves, http://www.moves-app.com/

CS281A Spring 2014 Class Project.

multinomial logistic regression (MLR) and hidden markov-model (HMM). We have found out that Glass has the best prediction among these devices; Phone is not reliable for cross-dataset test; Pebble's performance is acceptable and it's best for identifying `running` activity because of the huge arm movement.

In short, we made the following efforts in this report:

- We have implemented data collection applications on both Android platform and Pebble platform, and used them to collect two traces of accelerometer data for activity recognition task.
- We present the preliminary EDA on the raw data trace and investigate possible features that can be used for the activity classification.
- We use multinomial logistic regression and hidden markov model as two main algorithms to quantify the classification performance on our dataset.

# 2. PROBLEM FORMULATION

A supervised learning framework for activity recognition consists of data collection, transformation, and the design of adequate learning algorithm. In this section, we first briefly describe our data model (detailed system description and data collection is postponed to Section 3 for completeness). In Section 2.2, we show how raw data is transformed into feature space. In addition, the problem of feature selection for efficient learning is discussed. In Section 2.3, we formulate the learning problem, and propose two methods for a comparative study.

## 2.1 Data Model

We consider the sensor data model being a 3-dimensional accelerometer. Though gyroscope, magnetic sensor, rotation are also available and our data collection application supports them, for comparison with Pebble, we restrict the dataset to be accelerometer data. From a data analysis point of view, the obtained observations are a set of time series with fine time resolution. Each observation has 3-dimension $x, y, z$.

We also label the data from a video recording we did when we collect the data. We consider five different activities {standing, walking, downstairs, upstairs, running} (**for brevity, we will also use the number** $\{1, 2, ..., 5\}$ **to represent each activity**).

## 2.2 Data Transformation & Feature Selection

The raw data is a 3-dimensional time series exhibiting non-stationarity with fine time resolution. It is very difficult if not impossible to learn a good classifier that map the raw sensor readings directly to labels/classes. As most of the machine learning task, we face the challenge problem of feature extraction. In previous literature, it is widely acknowledged that a window frame based feature extraction could be adopted since most human activities are approximately periodic in nature. In this paper, we also consider feature extraction within a window frame, however, instead of merely focus on time and frequency domain feature, we proposed to add empirical distribution of frequencies as another categorical feature. This is because intuitively, different activities may induce very different frequency nature of the body. To be specific, we add binned distribution of Fast Fourier Transform (FFT) coefficients for each window as a $d$ dimensional feature, where $d$ is the number of bins. In addition, we calculate the empirical entropy as another feature candidate. As for time domain, we propose to add first order derivative (difference in time series) as an additional feature, the reason for this is the observation that different activities usually have distinguished "rate of change" in terms of body movement.

| | |
|---|---|
| Time Domain | $\bar{x}, \bar{y}, \bar{z}, var(x), var(y), var(z)$<br>$cov(x, y), cov(y, z), cov(z, x)$<br>$\sqrt{x^2 + y^2 + z^2}$<br>$x_{t+1} - x_t, y_{t+1} - y_t, z_{t+1} - z_t$ |
| Frequency Domain | $\sum_j w_j^2/N, -\sum_i p_i log(p_i)$ |

**Table 1: Extracted Features**

To sum up, we have the following features (also shown in Table 1):

- **Time Domain Features:** In each window, for one particular observation time series, we directly calculate Mean, Variance, and for each 3D streams of a particular sensor, we calculate the Covariance and the Magnitude. We also calculate the discrete first derivative and export the same statistics as features.
- **Frequency Domain Features:** We use Fast Fourier Transform to delve into frequency domain, and export energy and entropy of resulted FFT coefficients in each window.

Hence the transformed data set is a 20 dimensional tuple, with 19 features and 1 label indicating the ground truth of corresponding activity. Note that another information that are implicitly collected is the transition between activities. In fact, the consideration of adding this information or not is going to lead us to two entirely different probabilistic models. Namely, if dynamic transition information is ignored, the problem reduces to common classification problem, and the underlying assumption is that each observation is draw *iid* from some distribution. While if dynamic transitional information is incorporated in the model, because of the dependency of "future" and "past", usually local time series model should be considered for continuous measurement, and Markov Chains or semi Markov models should be used to model discrete hidden classes. Actually, the Hidden Markov Model (HMM) is one simple example of this kind.

People would argue that incorporating additional information should yield better classification results. However, it may not be true in practice. On one hand, a dynamic model like HMM is more complex to train and may has higher parameter estimation error. On the other hand, it is arguable if human activity really exhibits Markov transitions. Granted that transition tendency exist, the transitional matrix may (or for most people)NOT be stationary,which rend HMM unsuitable because it requires stationary transitions to propagate. Of cause non-stationary model could be considered, but we doubt it will be too complex to carry out real parameter estimation and inference work. That's why in our project, we tried both multinomial logistic regression and HMM as machine learning tools for the purpose of activity recognition.

Last but not least issue concerning the problem at hand is that the features we are using may not be all beneficial in terms of activity recognition. Since our previous feature selection is based mainly on empirical knowledge or intuition, the selected features may not be relevant or there may

be redundancy in the feature space. Principle component analysis (PCA) could be used to eliminate redundancy, but transformed features after PCA do not have a clear meaning. In this project, we consider regularized method for statistical feature selection. Particularly, we applied $L1$ regularized logistic regression and check the solution path.

## 2.3 Activity Recognition as Supervised Learning

As an abstraction of above description, our task is to find a function $f : R^{19} \times T \to \{1, 2, 3, 4, 5\}$. If we assume that transition information is not important and consider obtained samples as *iid* samples from certain distribution, our task is reduced to estimate $f : R^{19} \to \{1, 2, 3, 4, 5\}$. With this approximation, various classification methods are available, such as logistic regression (LR), support vector machine (SVM), linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), decision tree (DT), etc. We adopt Multinomial Logistic Regression (MLR) and Hidden Markove Model (HMM) in this project.

Let's briefly review some key formulation for multinomial logistic regression and hidden markov model. A comprehensive introduction of the theory can be found in Prof. Jordan's book.

As a generalization of logistic regress for multiclass problems, MLR models the probability of each class $y$ as a Logit function of linear combinations of feature variables $x$. Specifically, assume there are $K$ classes, for $m = 1, 2, ..., K-1$

$$Pr(y_i = m) = \frac{e^{\beta_m^T x_i}}{1 + \sum_{k=1}^{K-1} \beta_k^T x_i} \tag{1}$$

and for $m = K$

$$Pr(y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} \beta_k^T x_i} \tag{2}$$

which ensures each probability has value between $[0, 1]$ and their sum equal to 1. The model parameters are vectors $\beta_1, \beta_2, ..., \beta_{k-1}$. Note that our project, from a machine learning viewpoint consists of two task, which are parameter estimation for training and inference for testing.

The parameter estimation for logistic regression is a widely studied topic. Usually, various algorithms are available to compute the ML estimation in non-regularized case and MAP estimation in $L2$ regularized case. In fact, for logistic regression without regularization, the ML problem just reads,

$$min_\beta \sum_i -logp(y_i|x_i, \beta) \tag{3}$$

and with $L2$ norm regularization, we simply add a $L2$ penalty for $beta$,

$$min_\beta \sum_i -logp(y_i|x_i, \beta) + \lambda \|\beta\|^2$$

the penalty term can also be thought of putting a prior for $\beta$, thus sometimes is called MAP estimation. From a vast pool of such algorithms we find generalized iterative scaling, Iteratively Reweighted Least Squares (IRLS), L-BFGS (gradient based nonlinear programming), and some specialized coordinate descent algorithms. However, $L2$ norm only regularize parameter "proportionally", and usually is depreciated when feature selection is our main purpose. As an alternative, $L1$ norm is used instead of $L2$ with advantages that it tend to push some parameters to be exactly zero, thus favors feature selection. Actually, the improvement here is very similar to the advantages of using LASSO. In our project, we perform both logistic regression without regularization, and with $L1$ regularization as an attempt for feature selection, i.e.

$$min_\beta \sum_i -logp(y_i|x_i, \beta) + \lambda \|\beta\|_1$$

| | |
|---|---|
| $\bar{x}, \bar{y}, \bar{z}\|y$ | multivariate Gaussian |
| $var(x), var(y), var(z)\|y$ | multivariate Gaussian |
| $cov(x, y), cov(y, z), cov(z, x)\|y$ | multivariate Gaussian |
| $\sqrt{x^2 + y^2 + z^2}\|y$ | Gamma |
| $x_{t+1} - x_t, y_{t+1} - y_t, z_{t+1} - z_t\|y$ | multivariate Gaussian |
| $\frac{\sum_j w_j^2}{N}\|y$ | Exponential |
| $-\sum_i p_i log(p_i)\|y$ | Gamma |

**Table 2: Extracted Features Distribution**

The inference problem, or in other words, the prediction problem aims at estimate class label $y$ given a new set of observations. In the case of logistic regression, this is simply done by calculating class probability $Pr(y_i =$ based on estimated $\hat{\beta}$, and then choose a proper cut-off probability to assign class labels.

As is mentioned in last section, a major drawback of common classification tools like Logistic Regression, SVM, QDA, Decision Tree make *iid* assumption for samples and ignore temporal dependence in the phenomenon. Although most of existing research on activity recognition choose to ignore time dependence, we argue that transitional information may be able to help in that it can "filter out" impossible transitions and thus improve accuracy. The challenge for using HMM in our case is the choice of emission distribution. Because we have a relatively high feature dimension, some EDA should be performed before concluding any form of distribution $Pr(x|y)$. We summarized our choice in Table 2 and EDA will be provided in later chapters. Note that the choice is only approximately correct, especially for $var(x), var(y), var(z)$, a more proper choice could be matrix Gamma distribution, but the parameter estimation is hard.

The parameter estimation for HMM in our case is fairly simple, in fact, since we are in a supervised learning case, while training the hidden states are actually observable. The log likelihood is already a "complete" likelihood. i.e. in the parameterization

$$l(q, y) = log \left\{ \pi_{q_0} \prod_{t=1}^{T-1} a_{q_t, q_{t+1}} \prod_{t=0}^{T} Pr(y_t|q_t, \eta) \right\}$$

both $q$ and $y$ are known. The ML estimation can be calculated directly by taking derivatives,

$$\hat{a}_{i,j} = \frac{m_{ij}}{\sum_{k=1}^{M} m_{ij}}$$

with $m_{ij}$ defined as in the class, and the parameter estimate for distribution in table (2) can be calculated trivially in each class. For Gaussian and exponential, the results are well known, as for Gamma distribution with parameter $\theta$ and $\alpha$,

$$\hat{\theta} = \frac{1}{\alpha N} \sum_{i=1}^{N} x_i \tag{4}$$

we can compute $\alpha$ iteratively

$$\alpha \leftarrow \alpha - \frac{\ln(\alpha) - \psi(\alpha) - s}{\frac{1}{\alpha} - \psi'(\alpha)} \tag{5}$$

with

$$s = \ln\left(\frac{1}{N}\sum_{i=1}^{N} x_i\right) - \frac{1}{N}\sum_{i=1}^{N}\ln(x_i)$$

After parameter estimation/training, we use the obtained model for inference/testing. The classification problem is equivalent to estimate hidden state $q$ give observation $y$ and the model parameters. The posterior probability of $q$ is defined as

$$p(q_t^i = 1|y, \eta) = \gamma_t^i$$

where backward and forward propagation can be performed to find $\gamma_t^i$

$$\gamma(q_t) = \sum_{q_{t+1}} \frac{\alpha(q_t)a_{q_t,q_{t+1}}}{\sum_{q_t}\alpha(q_t)a_{q_t,q_{t+1}}}\gamma(q_{t+1}) \tag{6}$$

and

$$\alpha(q_{t+1}) = \sum_{q_t}\alpha(q_t)a_{q_t,q_{t+1}}p(y_{t+1}|q_{t+1}) \tag{7}$$

Finally, we pick up a reasonable cut-off probability threshold, e.g. $[\frac{1}{5}, ..., \frac{1}{5}]$, to assign the class labels for each $q_t$.

# 3. SYSTEM AND DATA

In this section, we describe our system implementation – Android platform and Pebble platform, followed by a briefly cover of our the experiement setup.

## 3.1 Android Platform

Our Android application is adapted from BearLoc[3]. We built a continous sensor monitoring application on top of `BearLocService` which makes sensor data collection

Our initial application involves sampling all sorts of sensor data from Android platform, including acclerometer, gyroscope, magnetic field, light, GPS, WiFi signals, etc. However, through some preliminary study, we have found that mobile phones and Google Glass cannot easily handle the extensive sensing tasks, especially when we are trying to sample acceleration in a relative high frequency. As has discussed in Section 2.1, since we are interested in the comparison among these wearable devices and Pebble only has a 3D accelerometer, we then restricted our Android application only for sampling acclerometer data.

The sensor data is obtained through `onSensorChanged()` API provided by Android OS. The sampling frequency is resource adaptive – it turns out to be 100Hz for phones and 50Hz for Glass.

Once the data is measured, we record the time in millisecond accuracy by calling `System.currentTimeMillis()`. To extract the data, we logged the data locally by writing to a `csv` file and also post a HTTP request that saves the data to `MongoDB`.

[3]BearLoc is developed by Software Defined Buildings (SDB) group at Berkeley; the intial goal of that project is to provide an open-source implementation for indoor semantic localization service.

| Device | max range | resolution | sample frequency |
|---|---|---|---|
| Phone | 19.6g | 0.013g | 100Hz |
| Glass | 19.6g | 0.039g | 50Hz |
| Pebble | 4g | 0.008g | 25Hz |

**Table 3: Sensor information of the devices.**

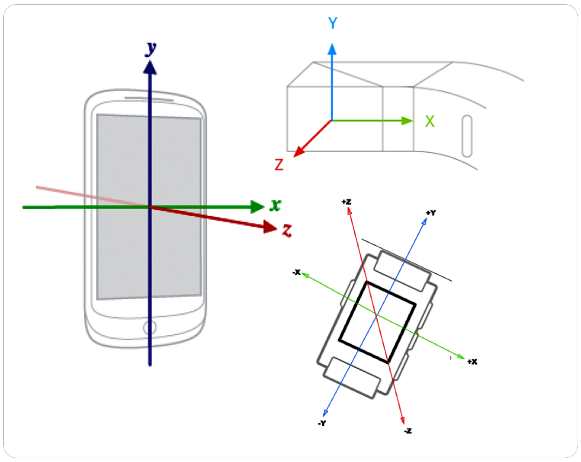


**Figure 1: 3D co-ordinate system of our hardware platforms.**

## 3.2 Pebble Platform

The Pebble accelerometer data collection application is based on `AccelerometerService` API provided by Pebble OS. We register a callback function as the handler whenever accelerometer data is available. The callback function operates on `AccelData` which already has a field of timestamp in millisecond accuracy.

The data storage and transmission is supported by `DataLogging` API which requires a customized Android application that uses Pebble SDK to retrieve logs by application universally unique identifier (UUID). We have also developed this accompany Android application for retrieving the data and writing them into a `csv` file for data analysis.

## 3.3 Accelerometer Data

On both Android and Pebble platform, the acclerometer coordinate system is defined relative to the device's screen (see Figure 1). When the user is facing the screen, the axes are:

- x: horizontal and points to the right.
- y: vertical and points up.
- z: points towards the outside of of the screen face.

However, the data differs in range, resolution and sample frequency. We use $g$ to denote the gravitational accelerometer constant (normal $9.81m/s^2$). In Table 3, we have listed the sensor information. Note that these values are extracted from our hardware sensor API so they are device dependent. Our window frame based feature extraction (see Section 2.2) is immune to the discrepancy of sampling frequency.

## 3.4 Data Collection

With our data collection platform, we have performed a series of activities and collected the acclerometer data. Note that institutional review board (IRB) is a necessity for this kind of experiment if more volunteers are involved, thus for

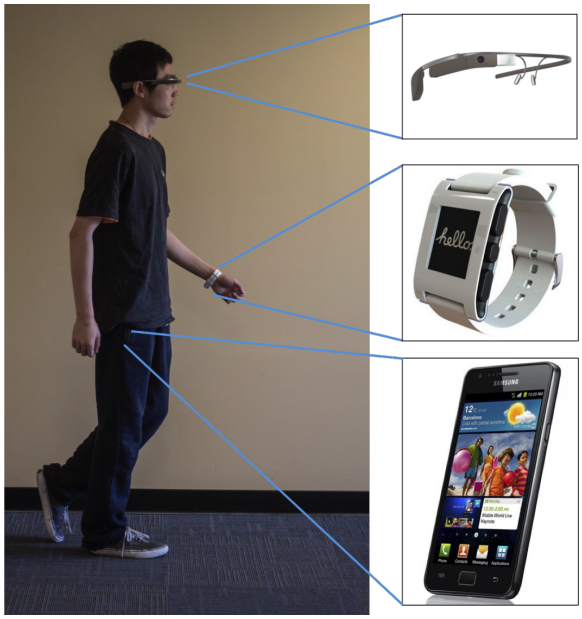

**Figure 2: Data collection process of simutaneously wearing three different devices and an illustration of their relative positions on the author's body.**

now the experiment is conducted by the authors. Figure 2 shows how these devices are worn on the author and the experiement is being conducted. To obtain the ground truth, we use an iPhone 5 to take videos continuously during the study. The video is manually processed to label the collected acclerometer data.

Two traces that we are using for this report were collected on May 10th. The author first walks out of a conference room, and then walk upstairs to 7th floor. He then pauses a while (standing) and then walk downstairs. After comning back to 4th floor, he passed the hallway by running. In such way we are able to collect all possible activities in one run. Notice that each activity duration is relatively random (some are quite short) which can be a challenge for a precise inference; however such randomness makes the data more realistic to people's normal life.

# 4. EVALUATION

In this section, we present our data and the results of our algorithm.

## 4.1 Raw Data and Extracted Features: EDA

The raw data sets are shown in Figure 3 and 4 with ground truth labels. A quick glance tell us that the three devices we are using generate measurement of different quality as far as classification is concerned. For example, in all axis $\{x, y, z\}$, Phone yields measurement with little noise.

The empirical distribution for some selected features are shown in Figure 5. We choose proper family of distributions for our HMM model by inspecting these empirical distributions, for example, we have chosen *Gamma* distribution for Entropy, and multivariate Gaussian for $\bar{x}, \bar{y}, \bar{z}$.

In order to check the relation among features, we look at the pairwise scatter plot, and one example of these plots containing frequency domain feature is shown in Figure 6. From

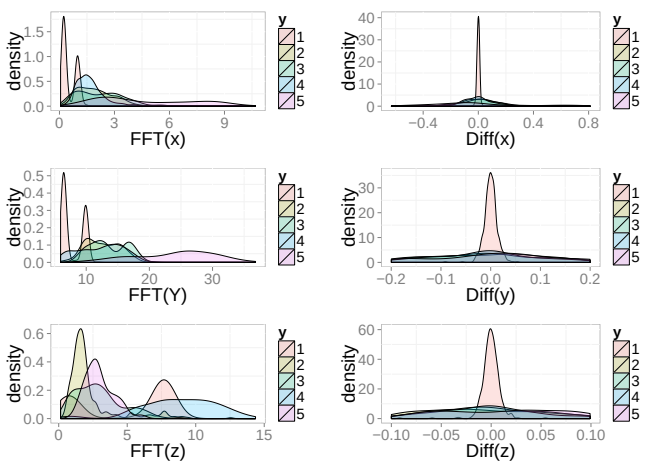


**Figure 5: Empirical distribution for some selected features**

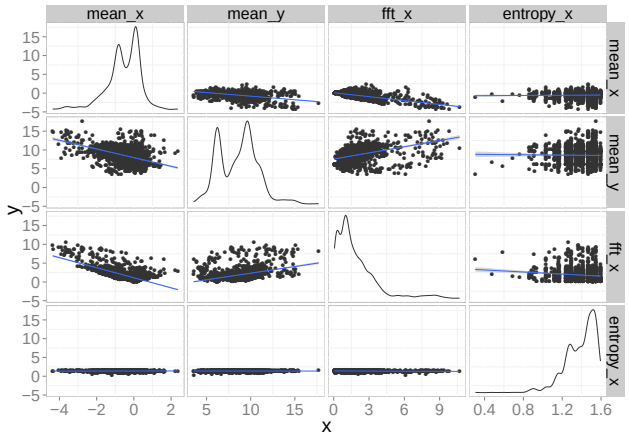


**Figure 6: Pairwise scatter plot for some features**

this figure we can further justify the conditional dependence and independence of the features for HMM. For example, we use multivariate Gaussian instead of single Gaussian for $\bar{x}, \bar{y}, \bar{z}$, since they exhibit a strong correlation when conditioning on a specific activity.

## 4.2 Classification Results

Both model fitting and testing errors are summarized in terms of overall accuracy and cross error table to indicate the performance of activity recognition with three different devices and two different learning methods. Let's examine the results in a device order for easy comparison.

### *4.2.1 Glass Results*

We summarize the results **with Multinomial Logistic Regression** in Table 4, 5 and **with HMM** in Table 6 and 7.

As expected, the testing error is much higher than training error for both Logistic Regression and HMM, while on testing set our system and algorithm still gives above 70% accuracy.

More interesting results can be revealed by looking at

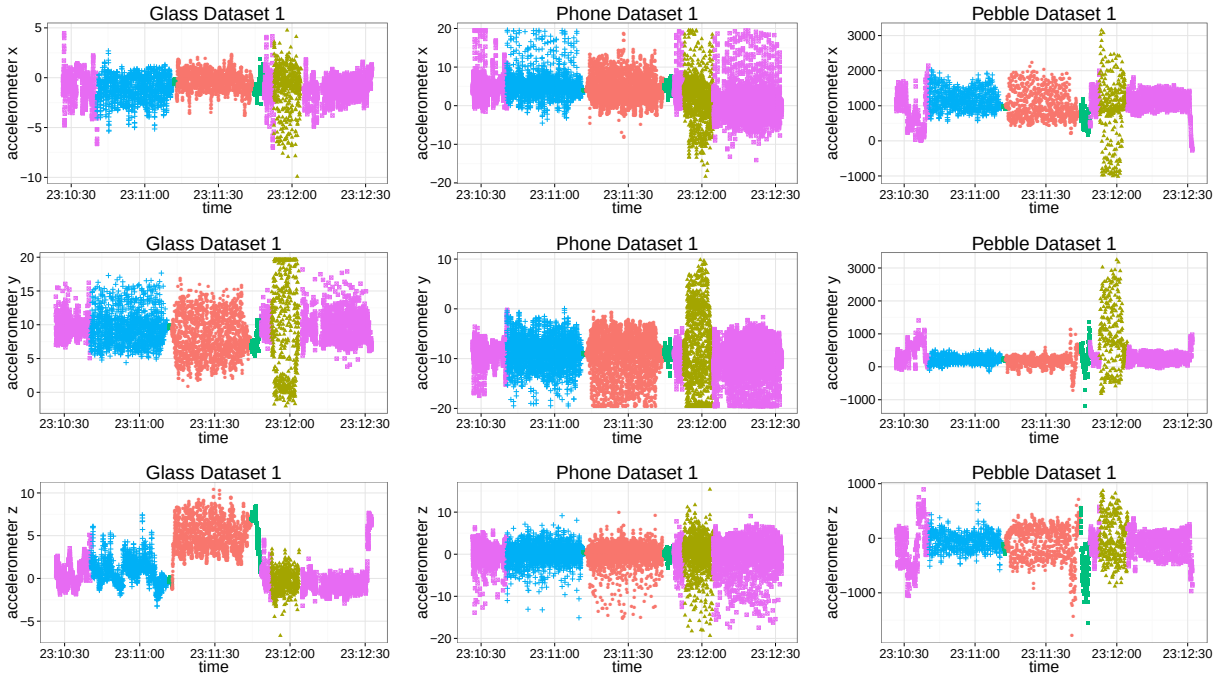


**Figure 3: The first raw data set visualization with groundtruth labeled.**

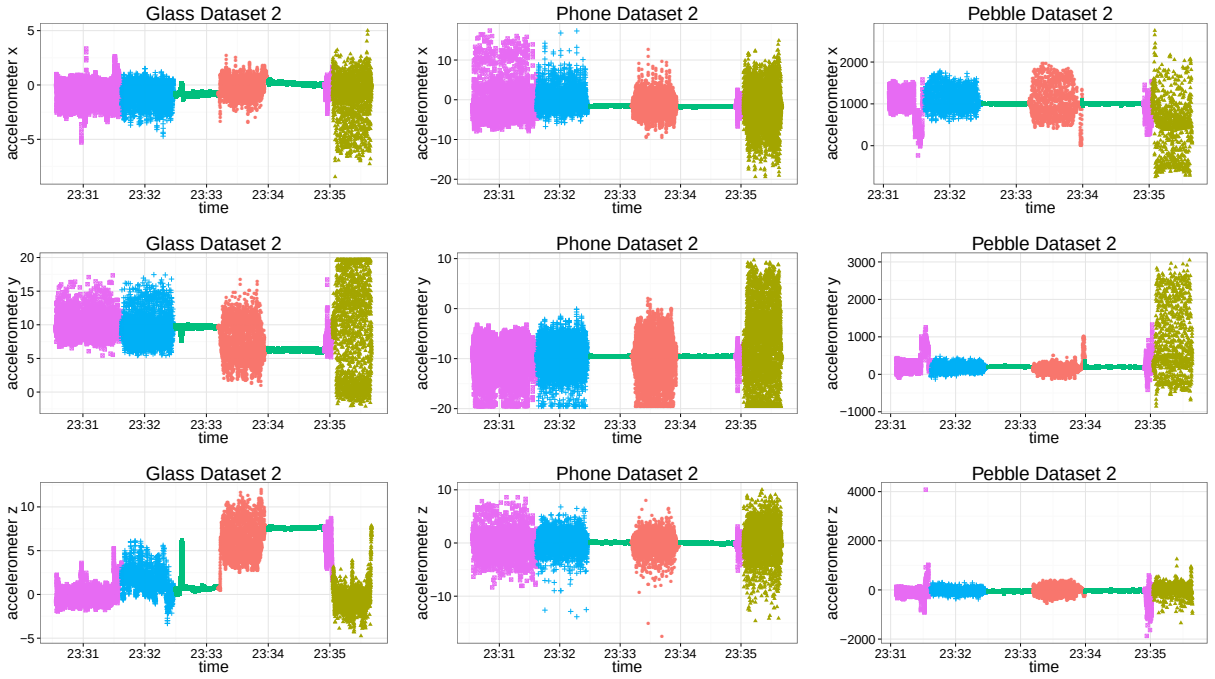


**Figure 4: The second raw data set visualization with groundtruth labeled.**

| Training Accuracy | Testing Accuracy |
|---|---|
| 0.9036635 | 0.7469244 |

**Table 4: Glass with LR: overall accuracy**

| P/T | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.625 | 0.032 | 0.020 | 0.057 | 0.000 |
| 2 | 0.125 | 0.727 | 0.188 | 0.057 | 0.107 |
| 3 | 0.250 | 0.199 | 0.651 | 0.007 | 0.071 |
| 4 | 0.000 | 0.041 | 0.114 | 0.878 | 0.053 |
| 5 | 0.000 | 0.000 | 0.026 | 0.000 | 0.767 |

**Table 5: Glass with LR: Cross Error Table**

| Training Accuracy | Testing Accuracy |
|---|---|
| 0.9599729 | 0.7188049 |

**Table 6: Glass with HMM: overall accuracy**

cross error table. Note that the number $\{1, 2, ..., 5\}$ represent {standing, walking, downstairs, upstairs, running}. The rows of the table are ground truth and the columns are predicted value, thus the $\{i, j\}^{th}$ element of the table is interpreted as *predicted value is i, while the true value should be j*. From this interpretation we see the diagonal terms in the table are actually accuracy in each class, and cross terms are different classification errors[4].

We observe that in the case of using Google glass, the logistic regression method performs a little better than with HMM. Specifically, in both algorithms, Google glass distinguished very well the *upstairs* state. Intuitively, this could be explained by the fact that when people are going upstairs, their head usually look up and maintains a distinct position, while in other states, people tend to move their head occasionally, which produce additional noise to Google glass measurement and harms the classification. This phenomena can also been seen especially from the fact that, for Google glass data with both algorithm, the standing state yields the most error.

Note also that we did not see improvement by incorporating transition (by using HMM), however, it does not exclude the benefit of transitional information. Actually, since our data set only measures a relatively short period, and the transitions of states are not statistically sufficient, we assume that for longer data with more complicated transitions, HMM method should show its advantages.

### 4.2.2 Phone Results

We summarize the results **with Multinomial Logistic Regression** in Table 8 and 9, and **with HMM** in Table 10 and 11.

We observe that both algorithm performed terribly on Phone data set. In effect, from previous part of EDA we already have a sense that Phone data set is more noisy and indistinguishable. The reason for that is simple: we put the Phone in the trousers pocket, and in all states except for standing, people's legs move periodically with a similar pattern. Hence we see that with Phone data set only the standing state is well classified, even with 1 accuracy with HMM, other states are mixed together and both classifiers only performs slightly better than a random guess.

[4]We normalize the error in each class, however different classes may have different sample size.

| P/T | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.400 | 0.057 | 0.006 | 0.029 | 0.000 |
| 2 | 0.066 | 0.692 | 0.240 | 0.067 | 0.155 |
| 3 | 0.133 | 0.211 | 0.629 | 0.022 | 0.017 |
| 4 | 0.400 | 0.033 | 0.110 | 0.880 | 0.068 |
| 5 | 0.000 | 0.005 | 0.013 | 0.000 | 0.759 |

**Table 7: Glass with HMM: Cross Error Table**

| Training Accuracy | Testing Accuracy |
|---|---|
| 0.748297 | 0.3341772 |

**Table 8: Phone with LR: overall accuracy**

| P/T | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.593 | 0.125 | 0.071 | 0.017 | 0.000 |
| 2 | 0.196 | 0.125 | 0.313 | 0.366 | 0.318 |
| 3 | 0.125 | 0.000 | 0.306 | 0.241 | 0.181 |
| 4 | 0.129 | 0.125 | 0.282 | 0.294 | 0.272 |
| 5 | 0.062 | 0.625 | 0.026 | 0.080 | 0.227 |

**Table 9: Phone with LR: Cross Error Table**

| Training Error | Testing Error |
|---|---|
| 0.9591281 | 0.3578059 |

**Table 10: Phone with HMM: overall accuracy**

| P/T | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 1 | 0.014 | 0.064 | 0.006 | 0.000 |
| 2 | 0 | 0.542 | 0.303 | 0.635 | 0.158 |
| 3 | 0 | 0.136 | 0.335 | 0.155 | 0.123 |
| 4 | 0 | 0.188 | 0.271 | 0.162 | 0.341 |
| 5 | 0 | 0.117 | 0.026 | 0.040 | 0.376 |

**Table 11: Phone with HMM: Cross Error Table**

### 4.2.3 Pebble Results

We summarize the results **with Multinomial Logistic Regression** in Table 12 and 13, and **with HMM** in Table 14 and 15.

| Training Accuracy | Testing Accuracy |
|---|---|
| 0.8658537 | 0.6677966 |

**Table 12: Pebble with LR: overall accuracy**

The results for Pebble data could be analyzed similarly. The performance of our classifier on this data set is in the middle of the performance on last two data sets. One interesting phenomenon is that the running state is very well classified, which could also be explained by intuition. When people are running, their forearm usually move with much higher frequency than in any other states. The classifica-

| P/T | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.166 | 0.065 | 0.000 | 0.013 | 0.000 |
| 2 | 0.166 | 0.631 | 0.274 | 0.083 | 0.242 |
| 3 | 0.000 | 0.163 | 0.709 | 0.111 | 0.060 |
| 4 | 0.166 | 0.131 | 0.000 | 0.763 | 0.090 |
| 5 | 0.500 | 0.0082 | 0.016 | 0.027 | 0.606 |

**Table 13: Pebble with LR: Cross Error Table**

| Training Accuracy | Testing Accuracy |
|---|---|
| 0.9318902 | 0.6237288 |

**Table 14: Pebble with HMM: overall accuracy**

| P/T | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.5 | 0.063 | 0.000 | 0.019 | 0.000 |
| 2 | 0.0 | 0.618 | 0.370 | 0.182 | 0.080 |
| 3 | 0.5 | 0.218 | 0.555 | 0.183 | 0.000 |
| 4 | 0.0 | 0.081 | 0.074 | 0.596 | 0.000 |
| 5 | 0.0 | 0.018 | 0.000 | 0.019 | 0.920 |

**Table 15: Pebble with HMM: Cross Error Table**

tion of running versus other activities are relatively simple because in our feature selection procedure, we also include frequency domain features.

# 5. RELATED WORK

In this section we briefly review a few literature on the activity recognition problem. Our work is built with their lessons in mind and mainly focuses on the comparison study fof three different devices.

In [1], the authors have collected a large set of cell phone sensor data with labeled activities. A decision tree algorithm[5] is mainly used for classification. However, according to our experiment and discussion, pocket phone accelerometer data is usually very noisy in that human leg movement is similar for some activities. We suggest that activity recognition can be greatly improved by considering various device measurements such as Google Glass for head movement and Pebble watch for forearm movement.

[2, 4] focus on improving the recognition accuracy by building more complicated statistical models. For example, in [4], a Two-Tier classifier is studied and in [2], a Hierarchical Hidden Markov Model is considered. Nonetheless, these models suffers from complicated training step, and may not be tractable if more features from diverse devices are included.

[6] discusses the energy consumption issues when using accelerometers in mobile devices for activity learning. They proposed a framework to adaptively choose sampling frequency when different activities are performed. This is important especially when measuring is conducted by consistently running application on cell phone, Google Glass, or smart watches. The choice of sampling rate also constitutes our future works.

# 6. CONCLUSION

To sum up, we have observed that measurements of different devices can help to distinguish different activities. As is mentioned above, this specialization comes from the fact that three devices are actually measuring the movement of different parts of human body. Therefore we propose that a combined sensoring system could greatly improve the classification performance of activity recognition. Major part of our future work will be devoted to this direction.

# 7. ACKNOWLEDGMENTS

We deliver our sincere thanks to Prof. Michael Jordan for his excellent teaching and explanation of a wide spectrum of statstical inference techniques. With the help from Kaifei Chen, we are able to quickly prototype the Android application using his `BearLoc` project. Also Prof. Björn Hartmann is generous enough to lend us his Google Glass and Qualcomm Toq[6] for the study. Moreover, many people are involved in the discussion of this project and how the study should be carried; without naming them one by one, we express our gratitude here.

# 8. REFERENCES

[1] J. R. Kwapisz, G. M. Weiss, and S. A. Moore. Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, 12(2):74–82, 2011.

[2] Y.-S. Lee and S.-B. Cho. Activity recognition using hierarchical hidden markov models on a smartphone with 3d accelerometer. In *Hybrid Artificial Intelligent Systems*, pages 460–467. Springer, 2011.

[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] V. Srinivasan and T. Phan. An accurate two-tier classifier for efficient duty-cycling of smartphone activity recognition systems. In *Proceedings of the Third International Workshop on Sensing Applications on Mobile Phones*, page 11. ACM, 2012.

[5] G. Wolf, A. Carmichael, and K. Kelly. The quantified self. *TED http://www.ted.com/talks/gary_wolf_the_quantified_self.html*, 2010.

[6] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *Wearable Computers (ISWC), 2012 16th International Symposium on*, pages 17–24. Ieee, 2012.

[5] J48 of Weka package

[6] Toq is not as friendly for programming as Pebble is. So we made the decision of using Pebble instead of Toq.