Mobile Activity Detection

Alejandro Carbonara
CMU CSD
GHC-6507
plothole@gmail.com

Pulkit Bhuwalka
MS-IIS, LTI, SCS, CMU
GHC-5411
pulkit.bosco05@gmail.com

Abstract

Our project aims to be able to categorize human movements using the sensors typically found on a phone. We first worked with a dataset collected by another project. Using these sets, we recreated the functions they used to generate feature vectors, and tested various learning mechanisms. We have created a small application to collect mobile sensor data from our phones, and plan to use that to create our own dataset. We found that we could get good classification using an SVM and a Gaussian Hidden Markov Model.

1 Introduction

Our project is about detecting user activity using mobile sensor data. Most mobile devices come equipped with an accelerometer and a gyroscope. When a person does their daily activities, their movements depend on what actions they take. The movements made while walking are subtly different from the ones made while walking up stairs, which are also quite different from the movements made while sitting. We aim to categorize these various types of motion, using only accelerometer and gyroscope readings.

1.1 Motivation

We think that this project can create value in Context Aware Computing and for measuring human performance. For instance, if a system is aware of a user's current activity, it can tailor itself to match the user's needs better. The system can understand that a user is sitting at home or at work. Based on such information, the system can make necessary adjustments such as changing the light settings in the room or bringing up a task list. For instance, a cell phone can start playing a user's favorite playlist and start a running app once the user starts to run. A cell phone may want to switch between vibrate and a ringtone depending on whether the user is sitting or walking.

Our system can also be used on athletes and physio patients to measure their performance. If we are able to successfully record the movements of a healthy runner and an injured runner, we may be able to detect injuries before they worsen. Similarly, the system can also be applied to assistive systems for the disabled. We can notify caretakers if an elderly person is having trouble climbing stairs.

1.2 Background

There have been various attempts at Human Activity recognition in research. Approaches have ranged from using ambient light and infra-red sensors to computer vision[3]. Some of these sensors are body worn, while some are external[4] to the user. There has also been some work [5][6] which proves that body pose and orientation can be estimated using acceleration measurements on the rigid parts of a body.

Since smartphones have become ubiquitous in our daily life, there has been significant research in utilizing data from its sensors. [2] provides a more comprehensive overview of the current state of the field. It also compares various classification algorithms, and stresses more focus on Hidden Markov Models. There has also been some work in figuring out the best approach for feature selection. [7] Approaches involve breaking the acceleration signal into high-frequency AC components for dynamic motion and low frequency DC components for static posture estimation[8]. Both, statistical measures such as variance, signal entropy etc. and frequency domain features such as Short Time Frequency Transform (STFT) and Direct Wavelet Transform (DFT)[9−10] have been used. [1] talks about a novel way to use SVM with fixed-point arithmetic for classification so as to reduce the computational cost on resource limited cell phones. It also hints at some methods of feature extraction.

1.3 Our Approach

We first tested our learning algorithms over some clean prepared data. We found several datasets online (UCI Machine Learning Repo and Kaggle) which have data from sensors annotated with activity, along with some precomputed feature vectors. After some deliberation, we chose to focus on the UCI dataset. We experimented running a support vector machine over the data. This dataset should serve as a good starting point, as it tested whether our code works on older data. We didn't want bugs in our collection to give us trouble at the start of our project.

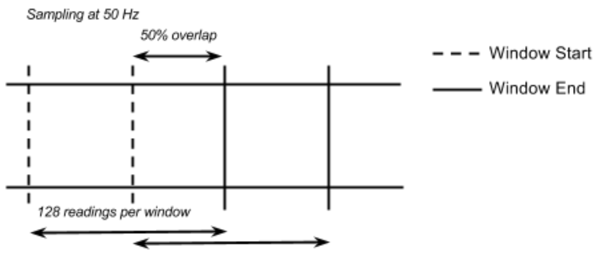
We modelled the time series information by taking fixed-width duration readings. Each portion gives us 128 readings in the X, Y, and Z direction for our accelerometer and gyroscope. These readings are used to compute a large number of features, creating a feature vector with 561 elements (simulating the generated feature vectors that were included with the UCI data).

This feature vector is used to train a SVM which can classify multiple classes using the 1v1 trick. We were able to get a good classification by working with a linear soft margin SVM.

While the SVM does a good job of classifying the data, we wanted to further increase the accuracy. One of the intuitions behind it was using past state to determine the current state. We thus modeled the data as an HMM with multiple Gaussian outputs which increased our accuracy by another 4%.

2 Feature Engineering

For our first tests, we worked with the UCI Human Activity Recognition Using Smartphone Data data set. This dataset was collected in a series of experiments over 30 volunteers. These volunteers wore waist mounted smartphones which had accelerometers and gyroscopes. They were instructed to perform several activities of daily living (WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING). The accelerometers and gyroscopes captured linear acceleration and angular velocity for all three dimensions at a constant rate of 50 HZ.



The data was then cut into 2.56 second portions with 50% overlap. Using a Butterworth low pass filter, they then split up the accelerometer data into its gravity component and a body motion component. They assume that the gravity will have a low frequency component, so a filter with .3 Hz cutoff would suffice. They then compute a large number of features over these time slots, creating a feature vector with 561 elements. 70% of the data was then put into our training data set, and the remaining 30% was put into our test data set. For all of our following tests, we trained over the data they labeled as training data, and tested over the data they classified as testing data.

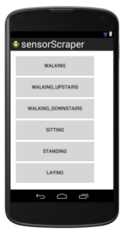
Our next step was to create a feature generation function ϕ that could give us similar classification. The UCI Dataset described all 561 features they generated in the some of the files they included with their dataset. As described earlier, the raw vectors we worked with were the gyroscope readings in 3 dimensions (tGyro-XYZ), and the accelerometer readings broken up into the parts caused by gravity (tGravityAcc-XYZ) and the parts caused by human motion (tBodyAcc-XYZ). We took the total magnitude of tGyro and tBodyAcc using the euclidean norm, creating two more signals. We then derived each of these vectors in time to create jerk signals. Finally, we took fourier transformations of some of these signals. In the end, we had generated a total 17 signals.

To convert these signals into vectors, we generated variables using the following functions (as taken from the UCI dataset):

- 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: Autoregression 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 two vectors.

Once each of these features were computed they were then normalized to be between -1 and 1. We created functions for every one of these features. We found that some of the values we generated were different from the values in the UCI featurized vectors. However, we were still able to classify the data pretty well using a subset of these features. Since we wanted to be able to work with any real world data, we created the entire pipeline of feature extraction from raw signal data.

3 Android Phone Data



While we have had success generating and testing various models on data from repositories such as UCI Machine Learning, we have to remember that these datasets are partly pre-processed and may be specific to the condition of collection. We want to ensure that the models generated by us work on real world data collected from our phones. To that end, we developed a simple Android application which allows us to extract raw data from the phones sensors. The application measures the exact same readings as the UCI dataset with the same frequency of 50Hz and a 2.56 second fixed window.

Unfortunately, due to differences in sensor types, the data we collected was not able to be classified by our mechanisms. Our readings were differently scaled than the readings from the UCI set, and we were not able to figure out how to translate them into a a classifiable format. Furthermore, we were unable to build up a sufficiently large dataset to learn on our own data. In the future, we would like to spend more time on eliminating noisy data points from the android dataset by using filters and thresholding the parameters.

4 Support Vector Machines

Our first approach to classifying our data was to simply run a support vector machine. We used the LIBSVM library, mainly because it had many features built in that were not in the matlab svm implementation. One major feature it had was that it had built in support for SVMs with multiple category classification. The comparison is done using ”1-against-1” classification. For this mechanism, we first solve the pairwise classification problem using a linear soft margin SVM.

For this mechanism, we first solve the pairwise classification problem. For every pair of classes i and j, we solve the optimization:

$$Min_{w^{ij},b^{ij},\xi^{ij}} \frac{1}{2}(w^{ij})^T w^{ij} + C \sum_t \xi_t^{ij}$$

$$(w^{ij})^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} \; \textit{if} \, y_t = i$$

$$(w^{ij})^T \phi(x_t) + b^{ij} < 1 - \xi_t^{ij} \; \textit{if} \, y_t = j$$

$$\xi_t^{ij} \geq 0$$

Where $\phi(x_t)$ is a function that generates a set of feature vectors. Once we have computed each of these pairwise classifications, we use the voting strategy to pick a class for each point. In this strategy, the class selected is the one that is chosen by the most pairwise classifications.

$$Max_i \sum_{j \neq i} sign((w^{ij})^T \phi(x_t) + b^{ij})$$

Using the support vector machines implemented in the library, we were able to get a pretty good classification on the UCI data. For the sake of comparison, we tested various learning systems. We ran five-fold cross validation to check whether we selected the best mechanism to model our data.

System	Parameters	Test Err
K-NN	(euclidean, k=10)	96.91
K-NN	(correlation, n=10)	96.93
Naive Bayes		73.29
1v1-SVM	(C =1)	90.37
1v1-SVM	(C =100)	93.94
1v1-SVM	(C =1000)	93.96
1v1-SVM	(C =10000)	93.91

Even though K-NN performed best, we were interested in working with SVMs. We also tested various kernels, but found that none of them improved our classification. Based upon our results, we decided that we would get our best results working with a SVM with C = 1000.

4.1 Results

Running this on our hold out data, we got an accuracy of 93.38%. The confusion matrix for our classification was:

490	11	10	0	0	0	WALKING
1	460	34	2	0	0	WALKING_UPSTAIRS
5	0	376	0	0	0	WALKING_DOWNSTAIRS
0	0	0	396	39	0	SITTING
0	0	0	93	493	0	STANDING
0	0	0	0	0	537	LAYING

The (i,j)th position is the number of items classified i that were really j. The six labels in order are: WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING. From this, we can see that the various walking motions were frequently confused, and standing was often confused with sitting.

5 Continuous Hidden Markov Models (with Gaussian outputs)

A Hidden Markov Model (HMM) is a natural fit for our project given that sensor output is time series data. Users generate a stream of sensor acceleration readings when they switch between different states. We consider the actual activities that a user is performing (WALKING, STANDING etc.) to be the hidden states of the model and the observed sensor readings to be output that is emmited by the model. Each 561 dimension feature vector that we observe is considered to be the output created by the one hidden activity. We assume a gaussian distribution over each one of these 561 features. This means that we learn a mean and variance of each feature dimension conditional on each hidden state.

The model is composed of the following (Q = 6 states/activities, D = 561 features)

1. π (Q X 1) - Prior probability of each of the states.
2. TPM (Q X Q) - The transition probability matrix.
3. Mean μ (D X Q) - The mean of each feature dimension, conditional on the state or class.
4. Variance Σ (D X D X Q) - The variance of the each feature dimension, conditional on the state or class.

Our steps to generate the model are as follows:

1. The prior probability of the model is given a random value and assigned across the 6 states.
2. The model is trained using the observed emissions (Xtrain) and hidden states (Ytrain) that are known to us. We train the model by directly computing the MLEs from the observations as is common in HMMs.
3. Once we have learned the model, the sequence of the test data is predicted by using the Viterbi path algorithm.

6 HMM Results

The algorithm is run over the test data using various sets of observations together. We notice that the HMM helps increase the accuracy up to 97.8% With no background information (each test point individually) the accuracy is only 94.3% but it increases by 2-4% when considered together.

Accuracy (in %)	Observations per sequence
94.3	1
94.67	5
96.2	10
97.12	20
97.05	50
97.29	100
97.79	All (3k)

Confusion Matrix

487	1	8	0	0	0	WALKING
0	470	1	0	0	0	WALKING_UPSTAIRS
5	42	373	0	0	0	WALKING_DOWNSTAIRS
0	0	0	463	27	1	SITTING
0	0	0	0	532	0	STANDING
0	0	0	0	0	537	LAYING

The (i,j)th position is the number of items classified i that were really j. Similar to the SVM case, the various walking motions were frequently confused, and standing was often confused with sitting.

7 Conclusion

We were able to get good classification of our data using both SVM and HMM models. Not only was the HMM model was able to give slightly better accuracy without taking temporal data into account, but it gave a significant boost in accuracy by about 4% when it had a sequence of information it could learn from. Another very interesting find was that we only need a sequence of 20 inputs to provide classification almost as good as taking the entire test data together. This has valuable practical implications since classification can be done in real-time using a small stream of data.

We were unable to run our models on our personal phones, due to the conditions of collection being too different to train off of the UCI data.

References

[1] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra and Jorge L. Reyes-Ortiz. Human Activity Recognition on Smartphones using a Multiclass Hardware-Friendly Support Vector Machine. International Workshop of Ambient Assisted Living (IWAAL 2012). Vitoria-Gasteiz, Spain. Dec 2012

[2] Mannini, A.; Sabatini, A.M. Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers. Sensors 2010, 10, 1154-1175

[3] Poppe, R. Vision-based human motion analysis: an overview. Comput. Vis. Image Underst. 2007, 108, 418

[4] Su, M.C.; Chen, Y.Y.; Wang, K.H.; Tew, C.Y.; Huang, H. 3D arm movement recognition using syntactic pattern recognition. Artif. Intell. Eng. 2000, 14, 113118

[5] Padgaonkar, A.J.; Krieger, K.W.; King, A.I. Measurement of angular acceleration of a rigid body using linear accelerometers. ASME J. Appl. Mech. 1975, 42, 552556

[6] Cappa, P.; Masia, L.; Patane, F. Numerical validation of linear accelerometer systems for the measurement of head kinematics. J. Biomech. Eng. 2005, 127, 919928

[7] Sabatini, A.M. Inertial sensing in biomechanics: a survey of computational techniques bridging motion analysis and personal navigation. In Computational Intelligence for Movement Sciences: Neural Networks and Other Emerging Techniques; Begg, R.; Palaniswami, M., Eds.; Idea Group Publishing: Hershey, PA, USA, 2006; pp. 70100

[8] Veltink,P.H.;Bussmann,H.J.;deVries,W.;Martens,W.J.;VanLummel,R.C.Detection of static and dynamic activities using uniaxial accelerometers. IEEE Trans. Rehab. Eng. 1996, 4, 375385.

[9] Bao, L.; Intille, S.S. Activity recognition from user-annotated acceleration data. In Pervasive Computing; Springer Berlin/Heidelberg: Berlin, Germany, 2004; pp. 117

[10] Mantyjarvi, J.; Himberg, J.; Seppanen, T. Recognizing human motion with multiple acceleration sensors. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Tucson, AZ, USA, October 710, 2001; pp. 747752.