What Am I Doing?
Robust Human Activity Detection with Smartphones Athletics & Sensing Devices

Elias Wu*
Stanford University
eliwu@stanford.edu

Paa Adu†
Stanford University
paa@stanford.edu

1. Overview

One of the most interesting areas of research with regards to HCI has been the problem of detecting what people are doing at a specific time - for example, detecting whether the person is biking, walking, sleeping, etc. This could potentially help with several tasks such as detecting exercising routines a la Google Fit [1], automatically bringing up a "cycling" UI displaying relevant information for cyclists, or turning on silent mode automatically when sleep is detected. In our project, we ingest human activity data and experiment on several different models to ultimately determine the best model.

2. Motivation

The wide-scale prevalence of smartphones and smartwatches has greatly improved the possibilities of human activity detection. Today, nearly everyone carries around a smart devices with sensors capable of measuring their movement (or lack thereof) and position, most notably accelerometers, gyroscopes, and magnetometers. This sensor data is already being utilized for applications such as step counting and localization. The application of sensor data towards activity detection has many interesting implications towards making our smart devices smarter.

3. Related Work

Machine learning methods that have been previously utilized for human activity detection include Naive Bayes, Support Vector Machines, and Markov Chains. Anguita et al. [4] tested Multiclass Hardware Friendly Support Vector Machine (MC-HF-SVM), which leveraged computation efficiencies to be able to perform better on smartphones. The team achieved an accuracy of 89% when distinguishing between the following activities: walking, standing, sitting, laying, walking up stairs, and walking down stairs.

Recently, human activity detection research has been greatly influenced by deep architectures. This has enabled better accuracy detection over a wider array of human activities. State of the art architectures typically utilize recurrent neural networks (RNN) with long short-term memory (LSTM) cells or similar cells. Nils Y et al. [6] achieved 93.7% accuracy on sensor dataset composed of records from 9 participants performing household activities and various exercises. This accuracy was achieved using complex CNNs.

*Elias Wu is a student in CS 229. He worked on parsing the data, implementing the modified k-means algorithm, and implementing and iterating the RNN.

†Paa Adu is a student in CS 221 and CS 229. He worked on implementing logistic regression and SVM.

4. Method

We evaluate four different models for classifying the sensor data. These are modified K-means prediction, Logistic Regression, Support Vector Machines, and Recurrent Neural Networks. These models were programmed in Python 3.5, and the recurrent neural network built with PyTorch [3]. The dataset we use to test these methods on is the subset of the UCI Heterogeneity Activity Recognition Data Set [11], namely the dataset only using the Android-based LG Watch [2], ruling out different calibrations from different watches. However, we also ran the best model across the entire watch data (collected on several smartwatch devices) in order to see how much of the data could be classified correctly.

4.1. Dataset

The dataset that we used contained time series accelerometer and gyroscope readings collected on 4 smartwatches (2 LG watches, 2 Samsung Galxy Gears) and 8 smartphones (2 Samsung Galaxy S3 mini, 2 Samsung Galaxy S3, 2 LG Nexus 4, 2 Samsung Galaxy S+). Each of these sensor readings had a corresponding activity class attached: biking, sitting, standing, walking, stairs up, stairs down, and null.

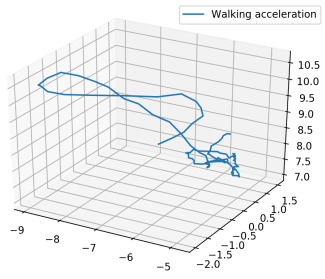

Figure 1. A plot of a sample of acceleration data gathered while walking.

In order to use the CSV data, we read the files line-by-line, and created datapoints consisting of a tuple of an array of n timesteps of x,y,z accelerometer data, and the corresponding class. Since the accelerometer sampled data around 166Hz. For K-means, logistic regression, and SVM, the x,y, and z accelerometer data is concatenated with each other, so the feature vector is of length $3n$. For the recurrent neural network, we shaped the data as a $(n * batchsize * 3)$ matrix. We tested several different values for n, and we decided on 80 timesteps (around 0.5 seconds) as a good interval for fast detection of a person's current physical activity. In total, we created 31,773 datapoints, of which 85% were used for training, 10% for validation, and 5% for testing.

The data was mostly distributed evenly, as can be seen below.2

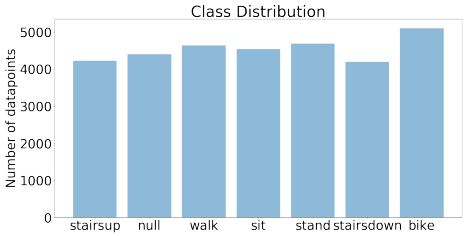

Figure 2. A plot of the distribution of the data classes.

We can see that we can achieve above 16.04% accuracy by guessing "bike" every time. Thus, our models will need to achieve higher than this accuracy to prove that they work.

5. Experiments

5.1. Modified K-Means

The modified K-Means assumes that there are already "clusters" created - training data of the same class is clustered together. Then, one step of the k-means algorithm is run, creating new centroids for all the clusters. Once this is done, predictions are made on new data by clustering it to its closest centroid. Our performance metric for K-means is its ability to cluster new data to its corresponding cluster. To this end, we split the data into training, validation, and testing sets. Since this algorithm does not have an iterative process, the validation set is used as a second test set.

5.2. Logistic Regression

We utilized Scikit-Learn to run a regularized logistic regression algorithm that minimized the following cost function:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} log(exp(-y_i(X_i^T w + c)) + 1)$$

In order to make predictions over seven classes present in the dataset, we trained seven different logistic regression models rather than a single multinomial logistic regression model. For each class, we created a model that predicted whether an example was or wasn't the given class. Probability estimates were calculated for each class and the highest scoring class was chosen to be the final prediction.

Figure 3. A plot of the learning curve for the standing vs. not standing logistic regression classifier.

5.3. SVM with Polynomial Kernel

We utilized the SVM algorithm using a third degree polynomial kernel. Similar to logistic regression, in order to make predictions over seven classes present in the dataset, we trained seven different SVM models, and picked the highest scoring model for prediction.

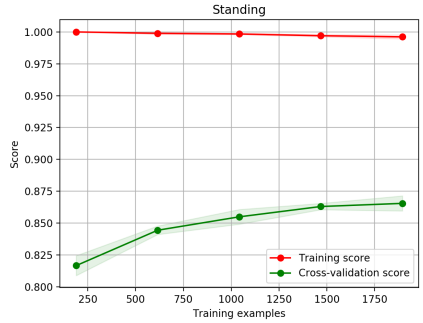

Figure 4. A plot of the learning curve for the standing vs. not standing SVM classifier.

5.4. Recurrent Neural Network

We tested several different Recurrent Neural Network architectures, specifically networks using Gated Recurrent Units (GRU) [5] or Long-Short Term Memory (LSTM) units [7]. We hypothesize that these networks are better suited for this task as they were designed for this purpose. These networks are able to encode a notion of "history" that the other methods may not be able to.

We first started with a single LSTM layer, with input size 3, using 40 timesteps, and a hidden size of 7. We iterated from this model to our final LSTM model, and switched to GRU as it was able to train faster and obtained results that seemed better than that of the LSTM-based models.

Our best performing architecture utilizes two stacked GRU units, with an input size of three (x, y, z) acceleration values for each timestep. Its output is of size 32, which is fed to a fully-connected layer with output size 16. The output is fed to a ReLU layer [10], which is then fed to another fully-connected layer with output size 7, for the seven different classes. This output is then fed to a softmax cross-entropy loss function.

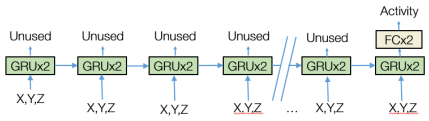

Figure 5. The best performing RNN architecture.

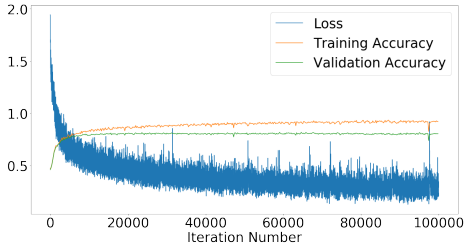

Figure 6. A plot of the learning curve for the RNN.

6. Results

Overall, the simpler models seemed to perform poorly, only obtaining accuracies slightly higher than if the model were to guess "bike" every time. The recurrent neural net architectures produced significantly more accurate predictions. A full table of our results is shown below.

Model	Train	Val	Test
Modified K-means	N/A	27.74%	27.22%
Logistic Regression	46.88%	34.42%	33.2%
SVM	65.23%	56.3%	55.6%
2 GRU, 2 FC, Dropout	**90.86%**	**81.83%**	**80.79%**
2 LSTM, 2 FC	72.14%	70.76%	69.83%
Best Class	16.04%	16.04%	16.04%

Table 1. A table summarizing our results with the different models. the GRU model was by far the most accurate on all datasets.

6.1. Modified K-Means

The modified K-Means achieved 27.22% accuracy on the test set. This simple model scored poorly. Although it does do better than just guessing the most common category, it does so only marginally. This shows that the data's similarity to its corresponding class is not represented properly by Euclidian distance, as used in the distance metric of K-means.

6.2. Logistic Regression

Logistic regression achieved 33.2% accuracy on the test set after using the best predictors from 7 individual models. This model also performed poorly, only slightly outperforming the modified K-means algorithm. Moreover, it overfits on the training set. This poor accuracy on both train and test accuracy shows that the data is likely not linearly separable, or at least not easily linearly separable.

6.3. SVM with Polynomial Kernel

The SVM model with a third degree polynomial kernel achieved 55.6% accuracy on the test set. This result was better than initially anticipated, as we got very poor results with the previous two models. Training each individual model appeared to overfit the data. For example, the SVM model trained to classify walking and not walking data had 100 % accuracy over the training set. Similar observations were made for the SVM model trained to classify biking and stair climbing data.

6.4. Recurrent Neural Networks

Our best performing RNN (Using two stacked GRU cells with two fully-connected layers on the last hidden output), we achieved 80.79% accuracy on the test set. This model exhibited overfitting, but was still able to classify more examples correctly than other models, including our best LSTM-based model. Although the results table shows that the LSTM model has much lower accuracy than the GRU-based model, this is partially due to the fact that we did not continue experimenting with LSTM-based models after the one listed in the results table. This model utilized only 40 timesteps (0.25 seconds), and did not use dropout to curb overfitting. It was also only partially trained, as the model would no longer achieve higher validation set accuracies after it reached around 70%. Left alone, the model would overfit with 99.5% accuracy on the training set. We decided to use GRU units as they trained faster, and iterated on GRU-based models.

To better visualize the results, we present a confusion matrix in Figure 7.

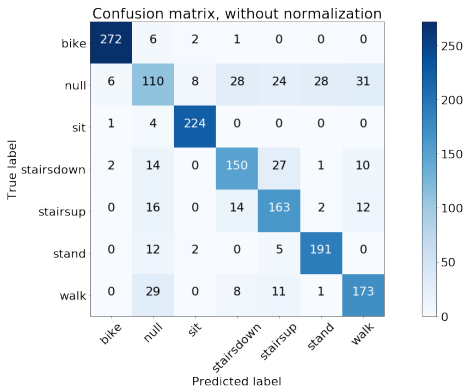

Figure 7. A confusion matrix of the RNN results.

We see that the null class is difficult for the model to classify; this may be due to the null class data looking like a combination of all the other classes. Interestingly, walking seems to be confused with null, suggesting that whatever action was classified as null in the dataset may be similar to walking (for example, slow pacing).

7. Future Work

Future work that may result in immediate improvements include training for longer, and iterating on different models for the recurrent neural networks, and using different recurrent cells such as Simple Recurrent Units (SRU) [9]. However, we also identify three key areas we would like to pursue.

7.1. Deployment

We would like to implement the best-performing model on a simple Android app for real word testing. The app will ingest accelerometer and gyroscope data from a phone, and output the predicted activity. To this end, we will have the device pipe data to the Google Cloud Compute instance, and retrieve the results from the instance to be used displayed in the application. We believe this would be a tangible, real-world experiment to visualize our project.

7.2. Rich features

All of the implemented models only utilized the accelerometer data (three features). The dataset also includes gyroscope data. Furthermore, pose information may be generated from combining both accelerometer data and gyroscope data with an algorithm such as complementary filtering or Kalman filtering [8]. These additional features may allow our models to produce better predictions

7.3. Longer time intervals

Each datapoint in our experiments only consisted of accelerometer data that was collected over 0.48 seconds. This granularity was chosen in hope of avoiding noise in the measurements that would be present over shorter sampling periods, as well as allow the model to predict what the the activity quickly when the data is live-streamed. Increasing the time interval for each datapoint to values such as 0.75 or 1 second could yield better classification accuracy at the cost of more delay in detection. For applications such as detecting workout time, a longer delay may not be detrimental.

8. Conclusion

Robust human activity detection has the potential to greatly improve smartphone user experiences. Our results indicated that it is difficult to achieve good activity classification accuracy using only accelerometer data and basic machine learning models. Although our best model did not perform as well as state of the art deep learning models, it was able to get within 15%. We believe that with more time iterating models, adding features, and utilizing more timesteps, we would be able to achieve or exceed state-of-the-art accuracies.

9. Acknowledgements

We would like to thank the staff of CS229 for offering a in-depth and interesting course. We especially enjoyed the poster session and the feedback we received.

References

[1] Google fit, 2017.

[2] Lg g watch, 2017.

[3] Pytorch, 2017.

[4] 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 *Proceedings of the 4th International Conference on Ambient Assisted Living and Home Care*, IWAAL'12, pages 216–223, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[6] N. Y. Hammerla, S. Halloran, and T. Ploetz. Deep, convolutional, and recurrent models for human activity recognition using wearables. *CoRR*, abs/1604.08880, 2016.

[7] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[9] T. Lei and Y. Zhang. Training rnns as fast as cnns. *CoRR*, abs/1709.02755, 2017.

[10] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[11] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pages 127–140, New York, NY, USA, 2015. ACM.