Same Body Detection Using Accelerometers

Zachary Stence
Texas State University
San Marcos, TX 78666
Email: zms22@txstate.edu

Torry Johnson
Central State University
Wilberforce, OH 45384
Email: tj45303@gmail.com

Bin Wang
Wright State University
Fairborn, OH 45324
Email: bin.wang@wright.edu

***Abstract*—Producers of wearable devices often skimp on security and authentication techniques in order to cut costs. With the increasing prevalence of wearable devices in our society, this has potential to cause great harm. This work aims to reliably determine whether two readings from accelerometer sensors came from the same person or different people wearing a device using machine learning classifiers provided by Pythons scikit-learn package. This work could later be used as a framework for providing easy authentication techniques for wearables using accelerometers, a very cheap sensor already present in most devices today.**

I. Introduction

In our society, wearable devices are becoming more useful, and thus are adopting more users. Current devices have wide capabilities ranging from simple activity tracking focused on heart rate and step counters [3], to wearables that can detect dangerous changes occurring in your body, such as the iTBra that detects symptoms of breast cancer [4]. Because of the small nature of these devices, producers often skimp on authentication to keep profile and costs low. However, when devices monitoring the medical status of a user have no means of authenticating the user, there is huge potential for problems to occur. For example, if there are multiple people in a household using similar medical trackers, a mix-up could cause both a false positive and false negative in identifying possibly serious medical conditions.

To solve this, wearable devices need a cheap and small, yet reliable system to authenticate the user of the device in a quick manner. We propose using accelerometers for this task as they are both cheap and small, and already used in most electronic devices today. Accelerometers sense how much the device is accelerating in each direction relative to the device. Because all humans are different and have different mannerisms inherent in how they move, their specific acceleration over a period of time can be used to identify and authenticate them, thus solving our problem.

This project aims to serve as groundwork for such an authentication system. As such, we will be exploring whether or not it is possible to detect whether two accelerometer readings were created from the same person or different people wearing a device while walking or performing a similar activity. We will first extract meaningful features from accelerometer streams, compare two at a time using coherence, then train a model to learn to classify them. We will use machine learning classification models in Pythons scikit-learn package [6] to accomplish this.

II. Related Work

This work was inspired by similar work completed in [1]. Our theory and method was largely taken directly from theirs, with a few adaptations. They managed to achieve a reliable accuracy of 85% when identifying pairs of accelerometer streams. Our goal was to replicate their results using different datasets and use Pythons scikit-learn package, a favorite among data scientists.

We also used a dataset published for the intent of activity recognition, discussed further in the next section.

III. Data

A. UniMiB SHAR

First, we used a dataset collected by researchers at the University of Milano Bicocca for activity recognition (UniMiB SHAR) [2]. While this dataset wasnt collected with same body detection in mind, it was almost a perfect dataset for us. It consisted of 30 different people, each performing 17 activities with 2 or 6 trials each. This allowed for many different possibilities of correlating two accelerometer streams and a wide variety of data to ensure our models did not over fit. However because they were only concerned with activity recognition, the streams they collected for each trial were only several seconds long. In all, the dataset contained roughly 60 seconds of data for each activity. This hindered our analysis because we needed longer streams in order to more effectively correlate two accelerometer streams. This led us to collect our own data.

Additionally, because there were only two trials for each person performing most activities (walking, running, going up/down stairs), when creating labelled data we encountered a problem with class weights. Out of the 60 trials we had at hand, 1,740 of all possible combinations were choosing two streams from different people, and only 30 of them from the same person. This caused our labelled dataset to be heavily skewed with around 98% of them being *False* classifications.

B. Our Data

We collected our own data using the PowerSense iPhone app [5]. We simply started the logging, waited a few seconds, put the phone in the pocket of our shorts, and walked down to the end of the hall. Each trial we completed resulted in a roughly 1-minute long accelerometer stream sampled at 50 Hz. The two of us did 6 trials in each pocket for a total of 24 accelerometer streams and around 24 minutes of total data.

In comparison to the UniMiB SHAR dataset, this gave us approximately 24 times as much data. However, we still only had less than 5% of the 13 hours of data that were collected for [1], unfortunately not made public.

IV. Method

As discussed above in Related Work, our method was heavily based off of [1]. In order to convert the discrete time signals captured by the accelerometers into usable mathematical representations, some processing needed to be done. First, rather than using the x-, y-, and z-axis signals, the magnitude of the signal was calculated at each point in the signal.

$$m_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$$

This addressed problems associated with varying orientation of the accelerometer in the device or in the users pocket. Next, the magnitude streams were split into windows of length w, and various statistics were extracted from each window known as features. We chose to use seven features, modeled off of [1]: mean, standard deviation, variance, mean absolute deviation, interquartile range, power, and energy. Once a feature vector was extracted from each window, they were concatenated into a feature matrix, serving as a mathematical representation of that accelerometer stream. Next, we needed a way to compare two feature matrices and evaluate how well they correlate with one another. To do this, we used coherence, "a measure of how well two signals correlate in the frequency domain" [1].

$$C_{xy}(f) = \frac{|G_{xy}(f)|^2}{G_{xx}(f) \cdot G_{yy}(f)}$$

Since not all humans move at the same frequency, it is necessary to consider many frequencies. This was done by summing each value in the coherence up until a maximum frequency of 10 Hz, and then dividing by that maximum frequency; the normalized coherence.

$$N(x, y) = \frac{1}{f_{\text{max}}} \cdot \int_0^f C_{xy}(f) df$$

Thus, the better two signals correlate across multiple frequencies, the higher this sum will be, and the higher the coherence will be.

Since we now have a way to get a mathematical representation of a signal (feature matrix) and a way to correlate two signals (coherence), now we only need to apply coherence to the feature matrices to obtain a representation of how well the original signals correlate in terms of the features we extracted. To do this we calculated the normalized coherence of each feature from the two feature matrices A and B across a window c. This mathematically represents how well each feature from A and B correlate with each other.

$$N^{AB} = \begin{bmatrix} N(A^1_{1\ldots1+c}, B^1_{1\ldots1+c}) & N(A^2_{1\ldots1+c}, B^2_{1\ldots1+c}) & \cdots \\ N(A^1_{2\ldots2+c}, B^1_{2\ldots2+c}) & N(A^2_{2\ldots2+c}, B^2_{2\ldots2+c}) & \cdots \\ \vdots & \vdots & \end{bmatrix}$$

Where $A^x_{y\ldots y+c}$ represents the vector of c feature values from the xth feature column of A starting in row y.

This coherence matrix can then be labelled with a 0 or 1 corresponding to whether A and B came from the same person wearing the device or not. Then, the matrix is split into its rows and each row is given the same label as the matrix. These rows of 7 coherence values and a label are then used as the inputs to various machine learning classifiers.

V. Results

We trained and tested several classifiers with different parameters and hyperparameters used in order to attempt to find an optimization. We found that keeping the feature window low and coherence window high generally yielded the best result, with an accuracy of 70-80%.

First, we generated some preliminary datasets and tried to narrow in on which classification model would work the best with our problem and dataset. We considered Support Vector Machine (SVM, as was used in [1]), k-Nearest Neighbors (kNN), Random Forest (RF), Logistic Regression, Naive Bayes, Neural Network, Gradient Boosting Classifier, and Decision Tree. We ran some preliminary tests with a smaller dataset of the first 15 people from UniMiB SHAR (feature window of 5 samples, coherence window of 3 seconds). Since we were relatively new to working with scikit-learns models, we utilized some code from a blog detailing how to test many classifiers at once and print their results [7]. We found that all of the classifiers were achieving an accuracy of 93-98% (Table I).

Classifier	Train Score	Test Score	Train Time
Nearest Neighbors	0.964450	0.967028	0.234375
Random Forest	1.000000	0.967028	63.781250
Neural Net	0.963545	0.966889	16.500000
Linear SVM	0.963545	0.966889	3.125000
Logistic Regression	0.963545	0.966889	0.062500
Gradient Boosting Clf.	0.975790	0.962716	16.937500
Naive Bayes	0.951892	0.955342	0.343750
Decision Tree	1.000000	0.932248	0.390625

Table I. Results from testing many classifiers on 15 people from UniMiB SHAR, unbalanced classes

Unfortunately, upon closer inspection of the confusion matrices, the classifiers were simply learning to always guess *False* because of the skewed nature of the UniMiB SHAR data, as discussed above. This led us to balance the data by keeping a random sample of the *False* classifications equal in size to the *True* classifications. With class weights of 50/50, our results changed dramatically (Table II).

Now it was clear that the models were not learning at all, with accuracies of just better than 50%, and were merely guessing *True* or *False*. We discovered that the problem was with our choice of coherence window. We chose 3 seconds initially because in [1] they mentioned "[b]ecause the typical walk cycle is on the order of seconds, it is advisable to chose [sic] a coherence window on the order of several seconds." However, upon reading further, they used a much longer coherence window nearer to 10 seconds, so we began creating

Classifier	Train Score	Test Score	Train Time
Nearest Neighbors	1.000000	0.565049	3.265625
Random Forest	0.732134	0.563107	0.000000
Neural Net	0.570248	0.547573	0.000000
Linear SVM	0.561983	0.541748	0.000000
Logistic Regression	0.560525	0.541748	2.328125
Gradient Boosting Clf.	0.996597	0.537864	1.890625
Naive Bayes	0.561011	0.530097	0.218750
Decision Tree	1.000000	0.497087	0.015625

Table II. Results from testing many classifiers on 15 people from UniMiB SHAR, balanced classes

new datasets to find what coherence window worked best. We also began narrowing down which classifiers we were considering, as k-Nearest Neighbors, Random Forest, Decision Tree, Gradient Boosting Classifier, and SVM were the best balance between performance and speed.

To optimize parameters involved in creating the datasets (feature window, coherence window, number of people, etc) we did all experimenting by hand. Since each time we wanted to try a new combination of feature window and coherence window we had to generate a new dataset, this process was time consuming. When testing with the UniMiB SHAR dataset, we found that a feature window of 4 samples and a coherence window of 9 seconds yielded the best results (Figure I). Because each stream of data was only seconds long, we could not increase the coherence window any further, but hypothesized that this would increase performance even more.

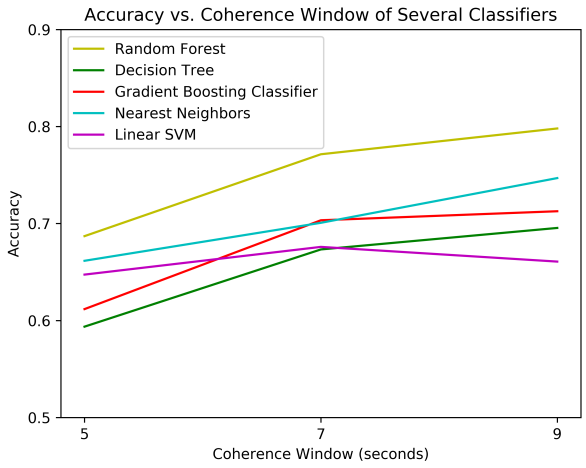

Fig. 1. A graph showing how coherence window affects the accuracy of different classifiers. Feature window was kept at a constant 4 samples. Classifiers were trained and tested 10 times each and results were averaged.

We chose to continune with a feature window of 4 samples and a coherence window of 9 seconds, as this was yielding the best results. In hindsight however, we may have obtained a local optimization without finding the best. This is an opportunity where future work can be done, but we moved into optimizing hyperparameters of each classifier to get their performance as high as possible. We employed scikit-learn's functions `RandomizedSearchCV` and `GridSearchCV` to accomplish this task. They proved invaluable. Results obtained using these optimization functions are shown in Tables III & IV, and the parameters for each can be found in the raw results on GitHub.

Statistic	Value
Accuracy	0.71646
Precision	0.67755
F1 Score	0.74775
Train Time (s)	0.00375
Test Time (s)	0.00313

Table III. Optimal performance for kNN on UniMiB SHAR dataset, feature window 4, coherence window 9. 10-fold cross validation when finding the best hyperparameters, 25 trials when computing performance statistics.

Statistic	Value
Accuracy	0.77190
Precision	0.74808
F1 Score	0.76246
Train Time (s)	7.74063
Test Time (s)	0.2

Table IV. Optimal performance for RF on UniMiB SHAR dataset, feature window 4, coherence window 9. 10-fold cross validation when finding the best hyperparameters, 10 trials when computing performance statistics.

Although RF performs slightly better than kNN overall, it took roughly 2,000 times longer to train, and 64 times longer to test, proving kNN is the better option for these datasets.

Since the UniMiB SHAR dataset offered such a limited amount of data (around 400 labelled datapoints) we concluded that collecting fresh data would increase performance simply with the availability of more datapoints. As discussed in the Data section, we collected 24 minutes of data, providing approximately 50,000 datapoints (125 times more than UniMiB SHAR). Due to the fact that RF is slow to train and test, we focused our efforts towards the end of the project on kNN. We used the same feature window and coherence windows as with the UniMiB SHAR dataset, then tried increasing the coherence window (Tables V–VII).

Statistic	Value
Accuracy	0.72903
Precision	0.72496
F1 Score	0.72997
Train Time (s)	0.08188
Test Time (s)	0.19313

Table V. Optimal performance for kNN on our dataset, feature window 4, coherence window 9. 3-fold cross validation when finding the best hyperparameters, 25 trials when computing performance statistics.

Our results mirror those from [1] "as the coherence window increases accuracy, precision and recall briefly climb but eventually settle."

Statistic	Value
Accuracy	0.73224
Precision	0.72449
F1 Score	0.73871
Train Time (s)	0.05625
Test Time (s)	0.14375

Table VI. Optimal performance for kNN on our dataset, feature window 4, coherence window 10. 3-fold cross validation when finding the best hyperparameters, 10 trials when computing performance statistics.

Statistic	Value
Accuracy	0.74407
Precision	0.73828
F1 Score	0.74726
Train Time (s)	0.05813
Test Time (s)	0.30625

Table VII. Optimal performance for kNN on our dataset, feature window 4, coherence window 15. 10-fold cross validation when finding the best hyperparameters, 25 trials when computing performance statistics.

Overall, we consider the results directly above to be our best and most reliable from the whole project. kNN offered the best results with reasonable performance and great speed even with our larger dataset, where other models offered poor performance and/or slow operation.

VI. Future Work

Based on the work done in this project, this system seems very promising in accomplishing the task at hand. However, several opportunities for future study became apparent during the project.

- Further investigate the effects of feature window and coherence window length to find better optimizations.
- Rather than using the magnitude to correct for orientation, make sure the accelerometer is positioned the same, or similarly every time. This would allow for full use of all of the x-, y-, and z-acceleration data providing more information about the movement of the user.
- Use this work's theory with other movement sensors commonly available in electronic devices such as gyroscopes, orientation sensors, linear acceleration sensors.
- Use other sensors in addition to movement sensors, to authenticate the user (GPS WiFi, magnetic, etc).

VII. Conclusion

There are countless ways that wearable devices will be used in the future of our society, medical or otherwise. The ability to gain realtime information about yourself or loved ones is invaluable, but only with the confirmation that the data came from who you expect. Regarding direct applications of this work, some more advancements need to be made. Classification rates of roughly 72% are good, but need to be much better before this method is used in mainstream devices.

All of the source code and data used for the project can be found on GitHub at:
github.com/zachstence/SameBodyDetection.

References

[1] Cory T Cornelius, David F Kotz. Recognizing Whether Sensors Are on the Same Body. Pervasive and Mobile Computing, vol. 8, no. 6, Dec. 2012, pp. 822836. ScienceDirect, doi:10.1016/j.pmcj.2012.06.005.

[2] Daniela Micucci, Marco Mobilio, Paolo Napoletano. UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones. Applied Sciences, vol. 7, no. 10, 24 Oct. 2017. Applied Sciences, doi:10.3390/app7101101.

[3] FitBit. https://www.fitbit.com/versa.

[4] iTBra. http://cyrcadiahealth.com/.

[5] PowerSense. Xu Shiyan, 2017. Vers. 2.6.3. Apple App Store, https://itunes.apple.com/us/app/powersense-motion-sensor-data-logging-tool/id1050491381?mt=8.

[6] Scikit-Learn: Machine Learning in Python. Scikit-Learn, Oct. 2017, scikit-learn.org/stable/index.html.

[7] Taspinar, Ahmet. Classification with Scikit-Learn. Ahmet Taspinar, 1 Mar. 2018, ataspinar.com/2017/05/26/classification-with-scikit-learn/.

[8] Xu, Weitao. Gait-based authentication system on smart wearable devices. Online video clip. YouTube. YouTube, 22 Aug. 2016. Web. 21 May 2018.