Midterm Report

Zachary Stence and Torry Johnson

6/22/2018

Purpose of the project

Wearable IoT devices often have no authentication techniques to verify which user is operating the device. If there are two people in a household and one happens to obtain a serious health issue while accidentally using another person's device, it would go unnoticed. This would cause the data collected by the health sensors to be misinterpreted and worsen the situation for all involved. The purpose of our project is to solve this problem by creating a framework for authenticating users based on detecting whether two accelerometer streams came from the same person or different people wearing the device.

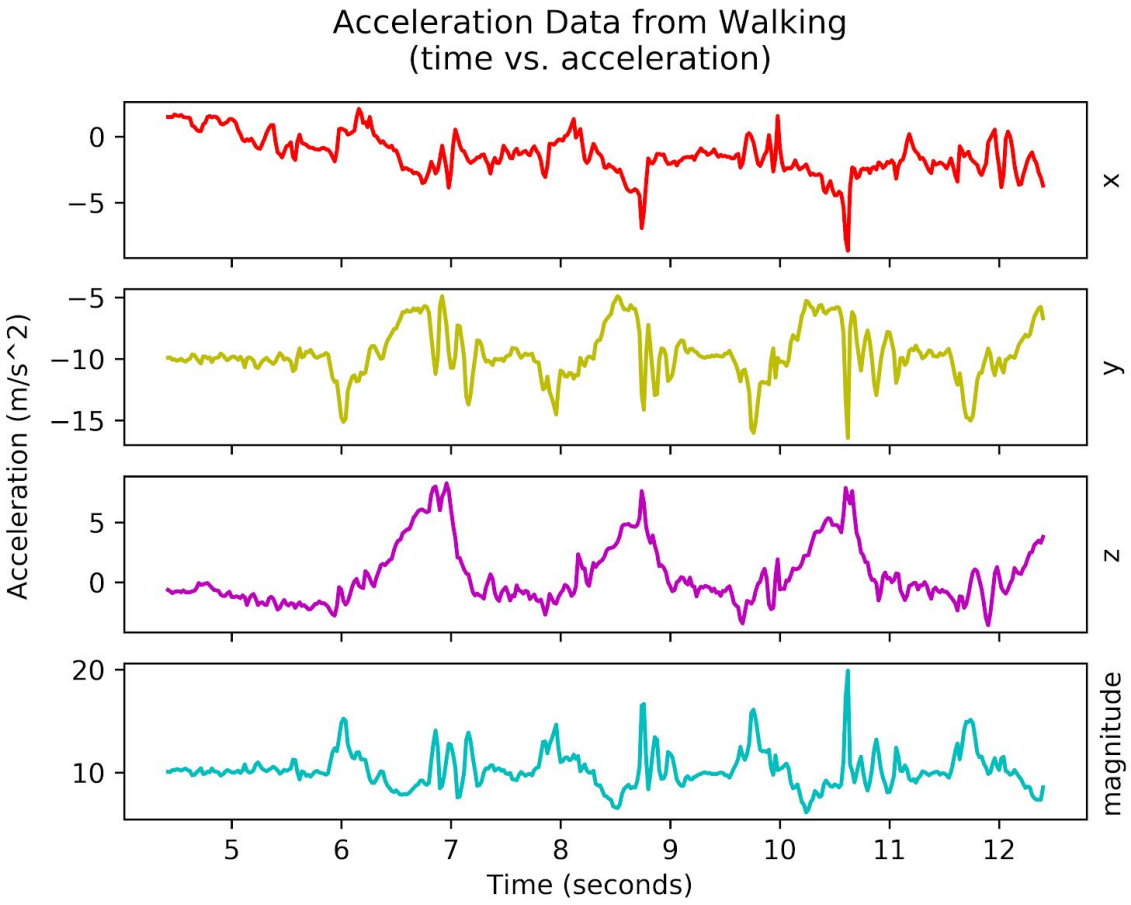
Description of technical approaches

We first experimented with the dataset and explored the format in order to understand what we needed to do to modify it for our needs. Once we had the data in a format we could use, we began working on extracting the necessary features from the data so we could train a machine learning classifier. The process is as follows:

1. Take one magnitude stream of acceleration data and partition it into equal windows of length w
2. From each window, extract a feature vector, F, comprising certain features (i.e. statistics), f, about the stream (e.g. mean, standard deviation, mean absolute deviation, etc)
3. Concatenate these feature vectors as the rows of a feature matrix, A, thus creating a mathematical representation of the stream in the form of a matrix
4. Repeat steps 1-3 with another magnitude stream to obtain a feature matrix, B
5. From feature matrices A and B, obtain a coherence matrix by calculating the normalized coherence of the same feature, f, from each matrix with a window length of several seconds
6. Label the coherence matrix with a 1 if the two original magnitude streams came from the same person wearing an accelerometer, and a 0 if from different people
7. Split the coherence matrix into rows and label them with the same label as the coherence matrix
8. Now, each of these rows has a length of 7 features and can be used as a collection of labelled data points with a machine learning classifier

Preliminary results

The dataset being used is from a study done regarding activity recognition [1]. The data consists of 30 people each performing 17 different activities, with 2 or 6 trials each. Below is an example of some of the data collected from one of the trials a person walking for a few seconds.



While the dataset was created with activity recognition in mind, we can use it for our purpose of correlating accelerometer streams and classifying them as from the same person or different people. Using the process described above, we have taken every possible combination of accelerometer streams to produce labelled data.

We have trained and tested several machine learning classifiers using scikit-learn [2] and are currently working on optimizing the results. We have been looking at several metrics to test how the classifiers are performing. Most important is accuracy (`test_score`), the number of correctly classified samples out of the whole set when asked to classify the test set. Similarly, `train_score` is the accuracy of the classifier when tested again on the same datapoints it was trained on. Also important are precision (the number of true positives out of all predicted positives) and F1 score ($\frac{2}{\frac{1}{Recall}+\frac{1}{Precision}}$, where recall is the total number of true positives out of condition positives). Last is `train_time`, which is the time in seconds the classifier took to train.

Initially, we were getting poor results because of the skewed nature of our dataset. Since there are more data points for different-body rather than same-body, the classifier was being trained to always guess false to achieve around a 98% accuracy rating, as seen below.

```
15 people, not balanced
--------------------------------------------------------------------------------
                     classifier  train_score  test_score  train_time
3             Nearest Neighbors     0.964450    0.967028    0.234375
4                 Random Forest     1.000000    0.967028   63.781250
0                    Neural Net     0.963545    0.966889   16.500000
1                    Linear SVM     0.963545    0.966889    3.125000
2           Logistic Regression     0.963545    0.966889    0.062500
5  Gradient Boosting Classifier     0.975790    0.962716   16.937500
6                   Naive Bayes     0.951892    0.955342    0.343750
7                 Decision Tree     1.000000    0.932248    0.390625
--------------------------------------------------------------------------------
```

While this is a high accuracy rate, this is obviously not what we set out to achieve and would prove useless in the real world. After trimming the dataset to have equal proportions of different- and same-body datapoints, the classifiers were getting an accuracy of just above 50% (i.e. not much better than guessing).

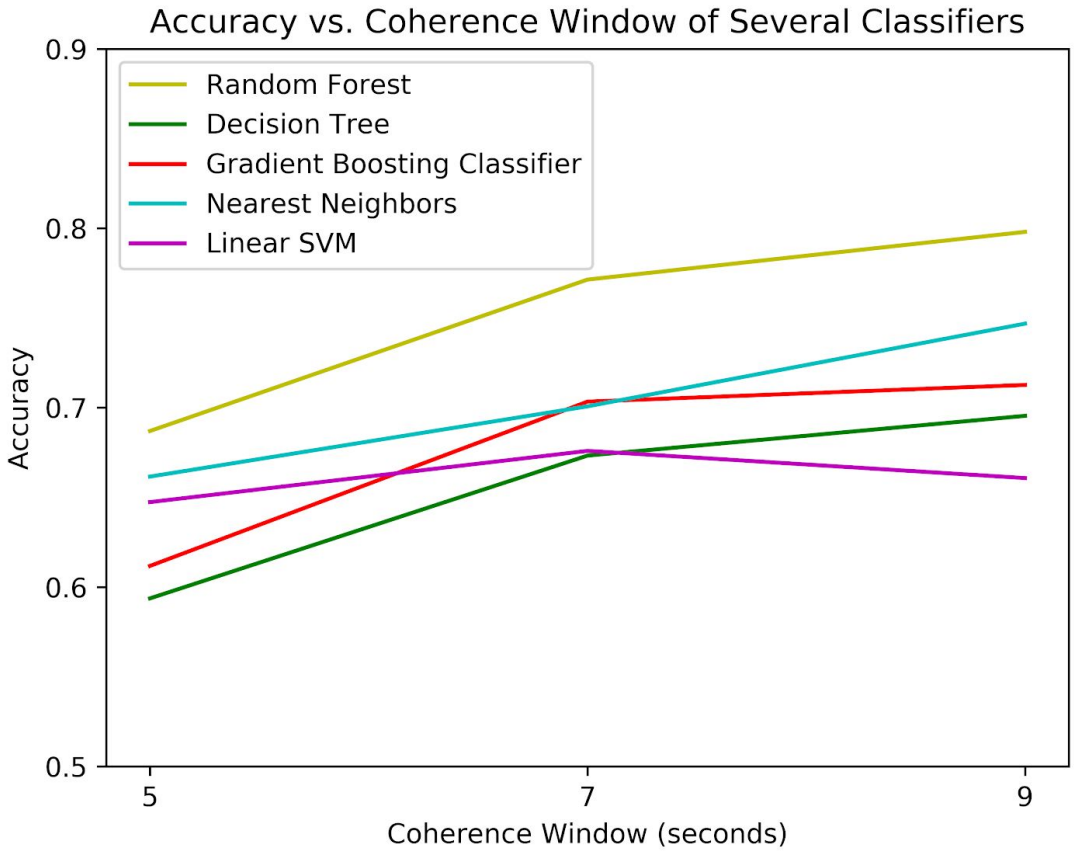
```
15 people, balanced
--------------------------------------------------------------------------------
                     classifier  train_score  test_score  train_time
6                 Random Forest     1.000000    0.565049    3.265625
5             Nearest Neighbors     0.732134    0.563107    0.000000
4           Logistic Regression     0.570248    0.547573    0.000000
3                   Naive Bayes     0.561983    0.541748    0.000000
7                    Neural Net     0.560525    0.541748    2.328125
2  Gradient Boosting Classifier     0.996597    0.537864    1.890625
0                    Linear SVM     0.561011    0.530097    0.218750
1                 Decision Tree     1.000000    0.497087    0.015625
--------------------------------------------------------------------------------

30 people, balanced
--------------------------------------------------------------------------------
                     classifier  train_score  test_score  train_time
2           Logistic Regression     0.576735    0.593750    0.109375
7                    Linear SVM     0.576735    0.588867    1.531250
3                 Random Forest     1.000000    0.576172    6.406250
5                    Neural Net     0.571114    0.576172    2.500000
1             Nearest Neighbors     0.717498    0.562500    0.000000
6                   Naive Bayes     0.563050    0.558594    0.000000
4  Gradient Boosting Classifier     0.938416    0.549805    2.218750
0                 Decision Tree     1.000000    0.521484    0.031250
--------------------------------------------------------------------------------
```

As you can see, all quadrants of the confusion matrices have roughly the same number of datapoints, meaning the machine is not learning, and more likely just guessing.

```
trained Linear SVM in 0.86 s
           predicted
            F     T
actual  F   337     172
        T   277     238

trained Naive Bayes in 0.02 s
           predicted
            F     T
actual  F   339     170
        T   281     234

trained Logistic Regression in 0.02 s
           predicted
            F     T
actual  F   296     213
        T   245     270

trained Decision Tree in 0.36 s
           predicted
            F     T
actual  F   273     236
        T   245     270

            classifier  train_score  test_score  train_time
0           Linear SVM     0.579912    0.561523    0.859375
3          Naive Bayes     0.565494    0.559570    0.015625
1  Logistic Regression     0.582600    0.552734    0.015625
2        Decision Tree     1.000000    0.530273    0.359375
```

While this may look like a step in the wrong direction, it was eye opening and got us back onto the right path because we realized that the classifiers were not doing as well as we had thought. We were not sure why this was the case, until we thought about changing some of the parameters that controlled how our datapoints were created from the raw dataset.

We decided to try generating new data points with a much higher coherence window length (from 3 seconds up to 9 seconds). While this does greatly reduce the quantity of data we have, it increases the quality because the coherence calculations are able to much better capture the periodic nature of the accelerometer streams. After doing so, we were able to achieve much better accuracy of 65-70% with all the models.



This encouraged us that we were working in the right direction, and that now we should begin working with the best models so far to try and tune their parameters.

We are now in arguably the most difficult and time consuming part of the project, tuning the hyperparameters of each model to see how much we can improve the results. Luckily, `scikitlearn` makes this much easier with two functions `RandomizedSearchCV` and `GridSearchCV` which allow you to specify a set of parameters and a model, and the function will evaluate the model with every combination of parameters and report the set that achieves the highest score (user defined). After doing this process for the best performing models, we have achieved much better accuracies.

```
Decision Tree Tuned Hyperparameters
Average accuracy:  0.6910126582278482
Average precision: 0.631180179467885
Average F1 Score:  0.6807788558361927

SVM Tuned Hyperparameters
Average accuracy:  0.6734177215189874
Average precision: 0.7468354430379749
Average F1 Score:  0.6465753424657537
```

```
Random Forest Tuned Hyperparameters
Average accuracy:  0.8035443037974688
Average precision: 0.7959695725467468
Average F1 Score:  0.8079489235148458

Nearest Neighbors Tuned Hyperparameters
Average accuracy:  0.8202531645569622
Average precision: 0.7869565217391303
Average F1 Score:  0.8360277136258658
```

As you can see, Random Forest (RF) and Nearest Neighbors (kNN) are both performing well with accuracy, precision, and F1 scores around and above 80%. kNN performs slightly better than RF, and also trains much faster (a fraction of a second compared to several seconds).

In order to further improve our results, we have a few steps we can take. We hypothesize that with more data, our classifiers would perform better. Thus, we plan to first create new datapoints from another activity containing more trials such as running (will give us 15 times more data). We also plan to see if it is possible to achieve a meaningful classification when mixing two different activities, which would give us much more data. Lastly, we will collect our own data to see if we can get similar results as we did with the UniMiB-SHAR dataset.

Tasks completed

- Visualized dataset and features (line plot, boxplots, etc) to gain insight into how classification will take place
- Constructed data into correct format (features) ready for use with scikit-learn
- Outline what our Python code will accomplish
- Code preliminary program in Python for testing one machine learning algorithm
- Train and test our data with at least one algorithm
- Determine which classification algorithm will work the best with the UniMiB-SHAR dataset

Tasks Remaining

1. Tweaking dataset/algorithms used to improve performance
2. Find best classification algorithm(s)
3. Obtain statistics about fully tuned algorithms
    a. Confusion matrix
    b. How much time is needed for authentication?
    c. How reliable is authentication?
    d. How do different algorithms perform?
4. Try using different activities in analysis
5. Collect our own data

References

[1] 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.

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