James Fan

Patrick Wang

Professor Musco

CS-UY 4563 Final Project Report

Predicting Human Action Based on Smartphone Data

Introduction

In this project, our goal is trying to predict human action based on smartphone data. There are six human actions: standing, sitting, lying, walking, walking downstairs, and walking upstairs. We are trying to accomplish a classifying problem in which one of the six actions is predicted as an output based on smartphone movement data such as triaxial acceleration, gyroscope, etc.

Data Description

The database we are using is:

http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions

The database is an updated version of a similar study performed before, but we decided to use the updated version and ditch the original version instead.

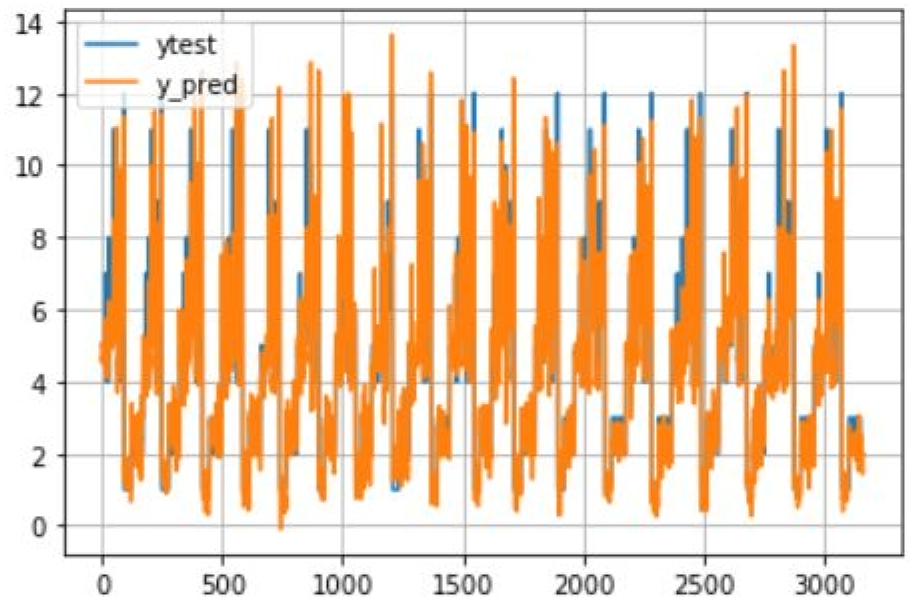
In the experiment, 30 subjects were asked to perform different actions, and smartphone data and action labels were recorded. The raw data was then calculated and converted into 561 different features. For a detailed description of all the features, please refer to features_info.txt. For a list of all the features, please refer to features.txt.

Approaches & Goal

There is a paper associated with the dataset, which achieved ~96% accuracy on predicting the actions, and this is our ultimate goal to beat. For a baseline method, however, we decided to use a simple multi-linear regression, and apply other methods learned throughout the course. There is also another method used, which is separating all the participants and train/test on each participant, as different people might have different movement patterns when performing certain actions. All codes for the methods can be found in the Data folder, and their results will be shown in the report.

Multiple Linear Regression

Even though this is a classification problem, our team takes multiple linear regression as the baseline method to see how well the data predict human actions. We simply use the linear regression method to train the training dataset to get a vector of beta. Next, we compute the predict y values using our beta and testing dataset, and compare to the actual y values. Y has a range of values from 1 to 12 which represent different human actions.



The above graph shows the comparison between the predicted human actions and the actual human action. From the graph, we can see that the regression lines are coincided which means the multiple linear regression does a good job on predicting human actions. This method also has an accuracy around 90% which supports the graph.

SVM

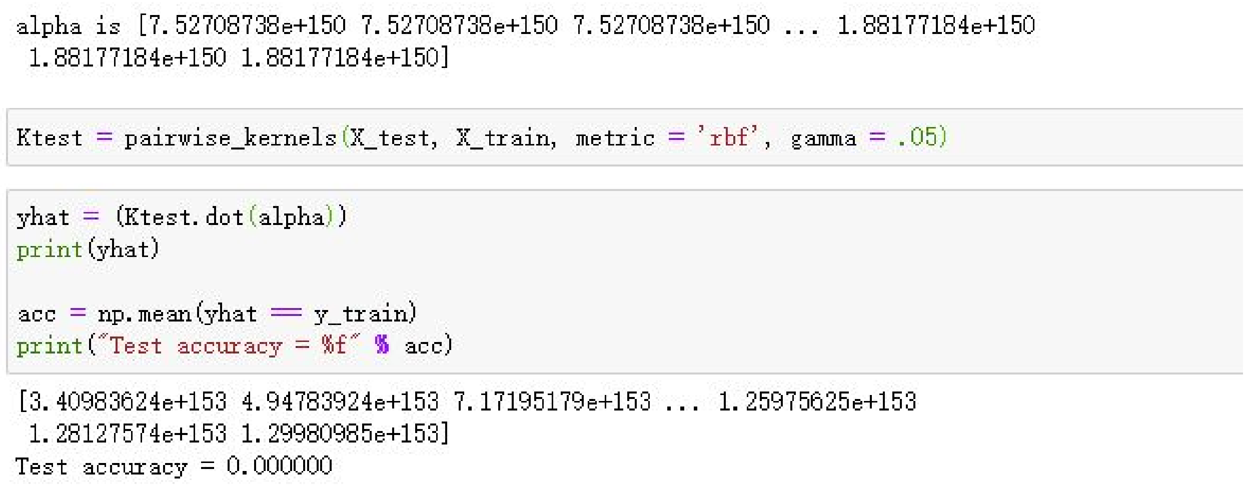
For the support vector machine method, our team first starts with a basic parameter of kernel="rbf", C=10, gamma=.05,verbose=10. And obtain an accuracy of 94%. We believe by changing the parameters we can get better results on precision and recall.

Therefore we used a grid-search algorithm to find the best combination of parameters using this method:

For the entire result, please refer to Data\svm_grid_search_results.txt. The accuracy score was given as a range (0.855 (+/-0.085)), and the best upper bound score is .94, which is not at all a significant increase. However, by doing the grid search, we discovered that the classification results differ greatly by action. For some actions we were able to get 1.0 accuracy and precision, but activities 9 to 12 have low accuracy compared to others. Class 9, 10, 11, 12 are all activities involving lie down. This is an interesting discovery regarding action data for actions related to lying down, but we were not able to pinpoint which of the features caused this phenomenon.

Kernel

Our team tried the kernel method, but the results are bad. Alpha values are very huge, and accuracy is 0. Minimum values in the kernel matrix are all close to 0. Our team thinks that's the reason why the kernel method doesn't work.



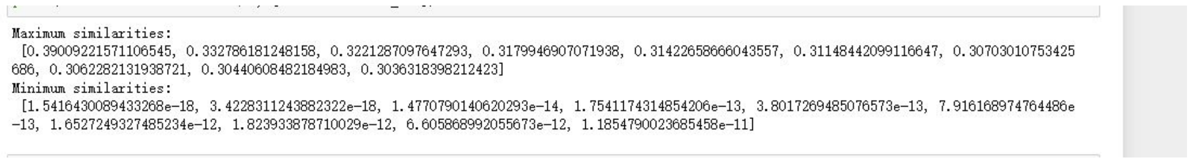
```
alpha is [7.52708738e+150 7.52708738e+150 7.52708738e+150 ... 1.88177184e+150
 1.88177184e+150 1.88177184e+150]
```

```
Ktest = pairwise_kernels(X_test, X_train, metric = 'rbf', gamma = .05)
```

```
yhat = (Ktest.dot(alpha))
print(yhat)

acc = np.mean(yhat == y_train)
print("Test accuracy = %f" % acc)
```

```
[3.40983624e+153 4.94783924e+153 7.17195179e+153 ... 1.25975625e+153
 1.28127574e+153 1.29980985e+153]
Test accuracy = 0.000000
```

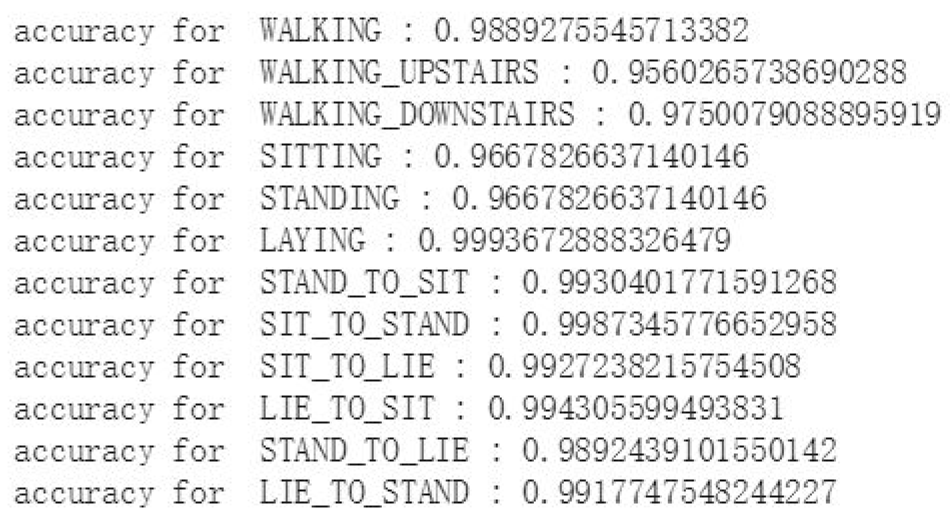


```
Maximum similarities:
 [0.39009221571106545, 0.332786181248158, 0.3221287097647293, 0.3179946907071938, 0.31422658666043557, 0.31148442099116647, 0.30703010753425
686, 0.3062282131938721, 0.30440608482184983, 0.3036318398212423]
Minimum similarities:
 [1.5416430089433268e-18, 3.4228311243882322e-18, 1.4770790140620293e-14, 1.7541174314854206e-13, 3.8017269485076573e-13, 7.916168974764486e
-13, 1.6527249327485234e-12, 1.823933878710029e-12, 6.605868992055673e-12, 1.1854790023685458e-11]
```

Gradient Descent

Our team chose to use one vs all method to do the classification problem with gradient descent. There are 12 different human actions, so it takes too much time to do one vs one classification. We believe one vs all would still give us accurate results. Each time we choose an action and make its y values to 1 and other actions' y values to 0. Then we find an optimal beta through gradient descent method and test its accuracy.

Here is the accuracy for predicting 12 human actions using gradient descent:


```
accuracy for  WALKING : 0.9889275545713382
accuracy for  WALKING_UPSTAIRS : 0.9560265738690288
accuracy for  WALKING_DOWNSTAIRS : 0.9750079088895919
accuracy for  SITTING : 0.9667826637140146
accuracy for  STANDING : 0.9667826637140146
accuracy for  LAYING : 0.9993672888326479
accuracy for  STAND_TO_SIT : 0.9930401771591268
accuracy for  SIT_TO_STAND : 0.9987345776652958
accuracy for  SIT_TO_LIE : 0.9927238215754508
accuracy for  LIE_TO_SIT : 0.994305599493831
accuracy for  STAND_TO_LIE : 0.9892439101550142
accuracy for  LIE_TO_STAND : 0.9917747548244227
```

All accuracy are above 95% and some of them are even 99% which means the data predict human actions very precisely. And this is a huge improvement from the accuracy of multiple linear regression methods. When using gradient descent, the accuracy for lying actions are actually higher than the other methods, which is also an interesting find.

Per-participant Training

In this method, we tried to separate the participants and perform a simple svm classifier along with a 5-fold validation on each of them. The result can be found at Data\per_participant_training_results.txt. Since each participant only gets 300-400 rows of data, we did not expect the result to be extremely good. And the result indeed shows quite a bit of fluctuation: within the 5-fold, all of the participant gets up to 1.0 accuracy, but the lowest accuracy was as low as .43, which greatly hurt the overall accuracy. However, this was an interesting method to try because this was something not learned in class, and it really showed that different and creative methods could be used on the dataset in order to get a good result.

Conclusion

In conclusion, we were able to beat the paper and achieve up to 99.9% accuracy on predicting the actions using gradient descent. But all the other methods and their results were fun to implement and see. We hope that this project (the dataset, the code and the results) can be looked at as significant. We definitely could have started earlier and started working independently when the circumstance prevented us from efficient and effective communication. Given more time, we might try the methods with more combinations of parameters as well as more methods such as feature extraction, regularization, and keep exploring how the dataset affect training results, and ideally generate an interesting find such as the one mentioned above with the lying down actions.