Activity Recognition Analysis for Discriminating Weight Lifting Exercises

Wenzhen Gong
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
wenzheng@sfu.ca

Tianhan Zhang
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
tianhanz@sfu.ca

Kathy Yan Shi
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
yshi2@sfu.ca

Dongyuan Liu
School of Computing Science
Simon Fraser University
Burnaby, BC, Canada V5A 1S6
dla126@sfu.ca

Abstract

Activity recognition has been traditionally focusing on identification of different activities. Our research is focusing on analyzing quality of executing certain activities. We gathered data from sensors while doing weight lifting exercises. We performed different learning algorithms to analyze the quality of activities.

1 Introduction

Physical exercises offer tremendous benefits to improve the qualities of peoples lives. However, improper activities may cause injuries and lead to long term sufferings. Especially in weight trainings, risk of injuries gets higher compare to other exercises [1]. Therefore, identifying the correct movement in each repetition of weight exercises helps to improve the quality of training and prevent the chances of injuries. However, the key challenge is discriminating the proper movements against those improper ones without a professional trainer. Our research applies machine learning classification techniques to automatically assess the quality of each bicep weight lifting activity.

Our dataset is from Weight Lifting Exercises monitored with Inertial Measurement Units Dataset in UCI Repository. The dataset contains on-body sensing data of six young healthy participants performing one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions, among which one is correct and the rest four is wrong.

During the training stage, we compared the performances between feeding the preprocessed data and semi raw data, and conducted different experiments on the following classification models: Variations of Random Assignment, Perception Method, Parzen-Window Density Estimation, K-Nearest Neighbour Algorithm, and Support Vector Machine. During the testing stage, we performed two types of testings: Testing new activities on existing subject, and testing new activities on new subjects. Figure 1 illustrates the workflow we conducted during our study of activity recognition.

Velloso *et al.* conducted activity recognition study on the same dataset in 2013 to distinguish the five classes of activities. Their work also intends to automatically assess the quality of weight lifting activities [2]. However, they focus on three key components of quality activity recognition, which are specifying the correct execution, detecting execution mistakes, and providing feedback on users [2]. The classification model they constructed was applied by leveraging Random Forest approach with Banging method [2].

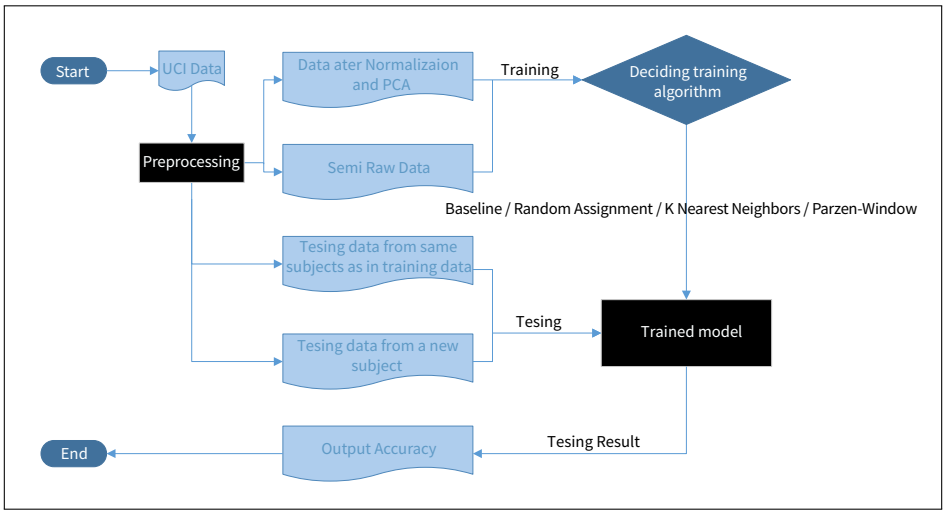

Figure 1: Workflow Diagram

Other than study the three key components of qualitative activity recognition, we conducted our experiment in a different perspective to explore the breadth of classification algorithms. Our study focuses on analyzing different classification results conducted by different classification models. Our ultimate goal is to be able to distinguish correct activities as well as different wrong activities.

2 Approach

2.1 Data Pre-processing

Our dataset contains 152 dimensions and 39422 instances. The data pre-processing stage includes transforming categorical data into numerical data, filling and removing sparse data cells, normalizing data, and reducing dimensions. Sparse cells were filled with their corresponding statistical data, such as min, max, mean, and variance. All the numerical data was normalized to have mean zero 0 and covariance 1. Moreover, feature exaction was handled by Principal Component Analysis (PCA) to reduce the dimensions. After we pre-processed the data, the data was reduced to 50 dimensions other than 152.

2.2 Training Algorithms

Our training approaches are based on sklearn library written in Python. In this section we will discuss different training algorithms we applied in our project. The following table should provide a general overview about our approaches.

2.2.1 Random Assignment

In this approach, we assign a data instance to a random class. Every class has equal probability to be assigned to. Since we have five classes in total, this means every class has 20% probability to be assigned to. However, as we already know that the distributions of different classes in our data set are different, we do not expect high performance from it. Instead, we will only use it as a benchmark for comparison.

2.2.2 Random Assignment I

In our second randomized approach, we also assign a data instance to a class randomly. However, unlike the first random approach, we take the original distribution of different classes in our training data set into considerations. That means different classes have different probabilities to be assigned

to. For example, Class 0 is 27.9% of training data, therefore, for a testing data instance, it has 27.9% possibility to be assigned to Class 0. The rest four classes possibilities follow the same idea.

2.2.3 Random Assignment II

Our third randomized approach is rather simple comparing the previous two. In this approach, we blindly assign all the testing instances to Class 0, which has the highest frequency in the entire data set. Hence, the performance of this approach is within a predictable range. With the test cases we randomly selected for existing subjects, both the semi raw data and the normalized data have 28.63% accuracy. For new subject, the accuracy rate is 33.56%.

2.2.4 Baseline

In baseline approach, we applied an algorithm which borrows the idea of k nearest neighbour. We use all of our training data to generate five centers of five classes. For the testing instances, we assignment them to the class with the closest center to them.

2.2.5 K Nearest Neighbours

In this training algorithm, we assign a testing instance to a class based on its K nearest neighbours. Among the K nearest neighbours, we find the class, which most of the training instances belong to, then assign the testing instance to that class. In terms of choice of K, we have tried multiple values, among which we found that the square root of total amount of training data gives the best outcome. This specific comes from one of our team members previous project experience.

2.2.6 Parzen-Window

Parzen-Window approach has very similar idea to K nearest neighbours approach, but also slightly different from it. In parzen-window approach, we focus more on the density of data instances around the testing data. In this case, we exam an area instead of a certain number of neighbours. After we decide the radius of the area we should be looking at, we pick the class that has majority number of instances within that area and assign our testing data to that class. That being said, with different settings of radius R, assignment of classes could be quite different. It is also possible that certain data instance does not have any training data instance within R. Therefore, we will have outliers. In this project and for this approach, we set all outliers to be Class 0, which is the major class in our training data.

2.2.7 Perceptron and SVM

In this project, we also included basic machine learning algorithms, such as perceptron and SVM. Since the training phase is based on sklearn package in Python, we used the librarys perceptron and SVM functions. For perceptron, instead of using default 5 times of passes over the training data, we set n_iter to 1000 and left the rest arguments as default. For SVM, we set the kernel argument as linear instead of the default rbf setting.

3 Experiments

We used 8 different approaches for experiments, namely Random Assignment (denoted as `RA`), Random Assignment I (denoted as `RA1`), Random Assignment II (denoted as `RA2`), Baseline (denoted as `baseline`), K Nearest Neighbours (denoted as `knn`), Parzen-Window (denoted as `par`), Perceptron (denoted as `percep`), and SVM (denoted as `svm`).

We tried different data preprocessing techniques and approaches. As being said before, we tried to pre-process the data using PCA. We did tests on both datasets with and without being processed by PCA.

3.1 Existing Subject

In the first set of experiments, we randomly selected 1/10 of data as testing data. Therefore, this dataset was to simulate an existing test subject, of which we have already learnt from.

Table 1: Accuracy for Existing Subject

Method	Accuracy Rate	Accuracy Rate (PCA)
RA	19.95%	19.84%
RA1	20.04%	21.20%
RA2	28.63%	28.63%
baseline	55.34%	67.29%
percep	91.20%	62.54%
par	98.15%	98.30%
knn	85.86%	85.73%
svm	N/A	83.45%

As can be seen in the table, the best result we have for existing subject is 98.30%.

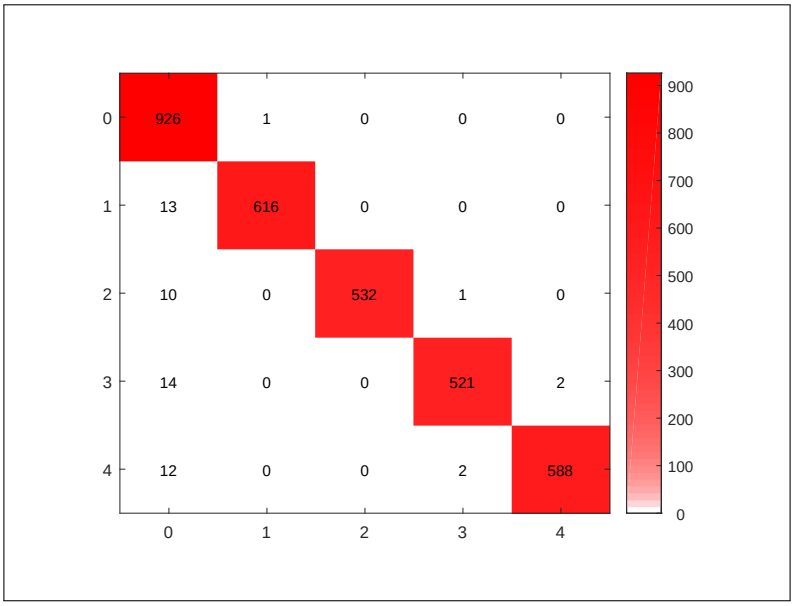

Figure 2: Confusion matrix for existing subject

3.2 New Subject

In real life scenario, we may not have any knowledge about the test subject. To gather more realistic results, we removed all the data for a certain subject in the training dataset. We used the data for that certain subject as testing data.

Table 2: Accuracy for New Subject

Method	Accuracy Rate	Accuracy Rate (PCA)
RA	20.66%	19.77%
RA1	21.11%	19.79%
RA2	33.56%	33.56%
baseline	51.71%	69.54%
percep	30.38%	42.80%
par	48.94%	50.77%
knn	53.25%	51.36%
svm	N/A	40.36%

As can be seen in the table, the best result we have for new subject is 69.45%.

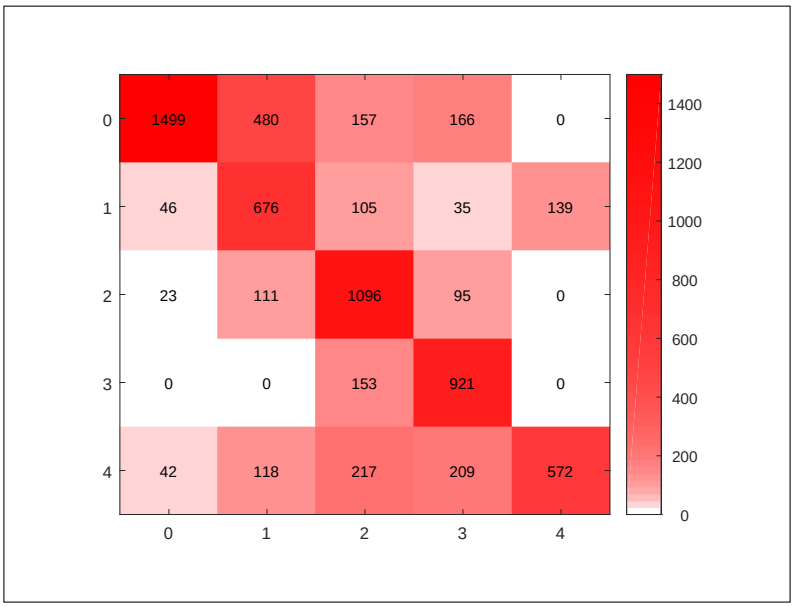

Figure 3: Confusion matrix for new subject

4 Conclusion

Based on our experiment, the maximum accuracy rate we could get for existing subject is 98.30% and for new subject is 69.45%, which is really good given the fact that the model does not need any knowledge about the new subject before hand.

In real life scenario, since health is one of the largest focuses in wearable smart device industry, companies like Fitbit or Jawbone could train such a model beforehand and provide their users with further services such as workout quality evaluations and wrong gesture warnings to encourage good exercise as well as to prevent injuries.

Contributions

Wenzhen Gong (`wenzheng`) implemented the most part of the Python program. Tianhan Zhang (`tianhanz`) and Dongyuan Liu (`dla126`) improved the program, fine-tuned the parameters and gathered data. Yan Shi (`yshi2`) mainly worked on data pre-processing. Each project members participated in choosing the learning algorithms and worked on the project report.

References

[1] M. Gallagher. Ten most common causes of training injury. *Muscle & Fitness*, June 1996.

[2] Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. *Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13).* Stuttgart, Germany: ACM SIGCHI, 2013.