Activity recognition from time series using supervised classification algorithms

Alessandro Pomes and Simon Schmoll

October 23, 2017

Abstract

Activity Recognition is a highly important classification problem for a lot of different domains. With better classification of accelerometer data all smartphone devices or activity trackers (to name a few: smartwatches, sport gear, sport watches, etc.) can more precisely predict which activity a person performs. There is already a lot of research available and the different steps for processing the raw data and classifying activities is getting better throughout the years. The intend of this project is to classify different activities on the basis of x-, y-, z-axis accelerometer data.[1]

Contents

1 Description of the problem 1

2 Description of our approach 2
- 2.1 Research . . . 2
- 2.2 Preprocessing . . . 2
  - 2.2.1 Missing Data . . . 2
  - 2.2.2 Feature Extraction . . . 3
- 2.3 Classifiers . . . 4
- 2.4 Validation . . . 4

3 Implementation 5
- 3.1 Data Handling . . . 5
- 3.2 Missing Data . . . 5
- 3.3 Feature Extraction and Selection . . . 5
- 3.4 Classification . . . 5
- 3.5 Main file (Activity_Recognition) . . . 5

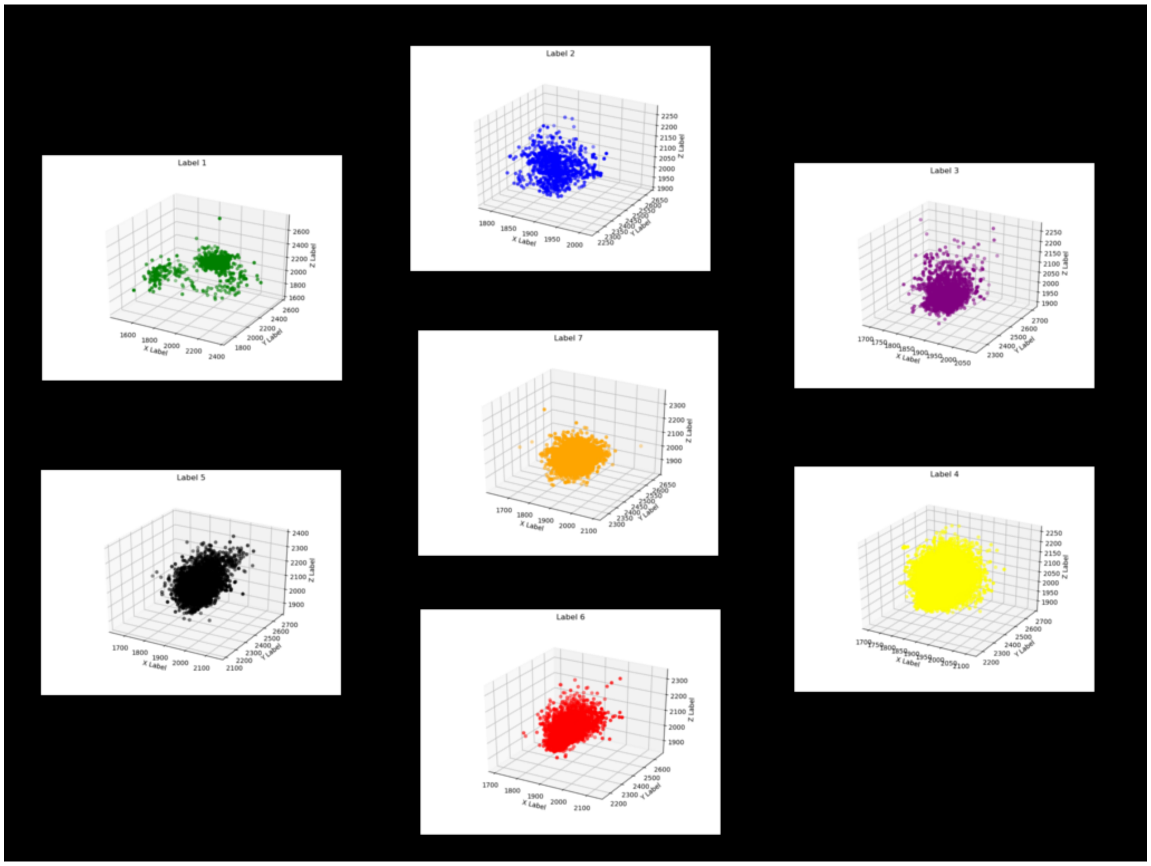
4 Results 6

5 Conclusions 7

[1]The dataset and additional information can be found here: https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer.

1 Description of the problem

- 15 datasets
- 3 attributes (x-, y-, z-axis values) with sequence numbering
- 7 classes: 1: Working at Computer, 2: Standing Up, Walking and Going up\down stairs, 3: Standing, 4: Walking, 5: Going Up\Down Stairs, 6: Walking and Talking with Someone, 7: Talking while Standing.
- 162500 instances for the first dataset.[2]
- The data is not completely balanced the instances for the 7 labels are[3]: 1: 33677, 2: 928, 3: 11179, 4: 26860, 5: 3191, 6: 2917, 7: 83748



2 Description of our approach

2.1 Research

To get a better overview of the scientific literature and the previous approaches to predict activity patterns the team chose to examine the relevant literature[1][2][3][4][5] provided by the lecturer. From

[2]The team analyzed the first of the 15 datasets

[3]Code for counting the instances can be found in the jupyter notebook

the research the team could extract valuable information which directed the implementation of the classification problem. Due to the time constraints of the project, the chosen approach was evaluated as the most promising one. As the team can learn from the mistakes and successful implementation of previous attempts of activity recognition. Especially the research by Akram Bayat et al.[2] and Pierluigi Casale et al.[3] was really helpful in understanding the problems which arise with classification of accelomerator data. After reading through the research the most promising work for our specific dataset was done by Pierluigi Casale et al.[3], as the data collection-chest mounted accelomerator- and the sampling frequency - 52 Hz- are identical to our classifcation problem.

2.2 Preprocessing

2.2.1 Missing Data

There are two important facts to consider when searching for missing data. First it is highly advisable to scan the dataset for missing data. There are several possibilities, how missing data is represented in the data set. For example, N/A, ? or the value 0 could all be used to represent missing data. Whereas the latter is highly dependent on the context.
In the description of the dataset there is no specific information about missing data, therefore the team was not aware if there is any missing data that could have some effect on our prediction.
TO find out if there are representation of missing data in the dataset the team searched for the specific representations and removed a single instance. A The terms *None* and *NaN* are commonly used to represent missing data in a Dataframe with the Panda library or in Python in general. So, as we have seen in the literature[6] , Panda treats *None* and *NaN* interchangeable for indicating missing or null values. To clean our dataset from missing values, after importing the data, we used several useful methods for detecting, removing, and replacing null values in Pandas Dataframe. For the final detection we used the method $isnull()$ that provide us with a mask where every *None* or *NaN* is replaced with 'True', vice versa if it is a valid representation of an value it is replaced with 'False'. Conclusively, the team just scanned through the output vector to find out if there were 'True' values in the DataFrame. Through this approach it was possible to see that no such representation of missing data occurred in the Data set. Oftentimes, missing or incorrect data can also be represented by values which are not in the value range of the example. After examining the different informations provided, we concluded that only the label could have incorrect values. Through searching for values which are not in the described value range (1-7), we could reason that there is indeed one instance with a '0' as label. Deleting instances is not always the best approach for incorrect data representations, but in our case it was a feasible approach as it only affected one instance of our dataset.

2.2.2 Feature Extraction

Feature Extraction is a crucial step in any classification problem. There are several possibilities, how to extract features from the dataset. As the current state of research presents different approaches to feature extraction, the team carefully evaluated the approaches and chose one technique for narrowing down the data and two mathematical models for extracting different features.

2.2.2.1 Windowing

As stated by Pierluigi Casale et al.[3] the most promising and most precise technique for starting the preprocessing of the different accelerometer data is to use 'windowing' with a 50 percent overlap between the windows. Concretely, this means dividing the dataset into different sections where we use the cutoff frequency of 1 Hz. Which is also inspired by Pierluigi Casale et al.[3]. As a consequence we get windows of 52 instances of data. But there are some limitations which are highly important to mention. It is not possible to create only consecutive windows of 52 samples each. This has to do with a inherent property of the data itself. As there are different labels for each sample set, it would be possible to get two different labels into one window. As this would destroy the purpose of this approach (which will be shown in the coming sections when the processing of the windows is demonstrated), the team decided to accept a few windows with a size smaller than 52 samples. As they occur only between two different activities the quantity and influence of those windows is negligible.

2.2.2.2 Mean value

After windowing the data set into nearly even windows. The team extracted the mean value of each axis in each window. An example calculation for the extracted features would be as follows: the dataset has 162500 values after preprocessing. Divided by 26 (because of the 50 percent overlap), we receive 6246[4] windows. For every of these 6246 windows we get the mean values of the x-, y-, z-axis. To heighten the chance of a better classification we used another feature extraction method which will be explained in the following section.

2.2.2.3 Standard Deviation

The selection for feature creation with the mean value and standard deviation is not arbitrary. As can be seen from the research of Pierluigi Casale et al.[3] the first two feature engineering methods mentioned by the authors are mean value and standard deviation. This feature creation is itself based on the research of Mannini et al.[7]
Standard Deviation is a commonly used technique to extract features. The appliance of the standard deviation of the three axis is identical to the mean value extraction technique. In total it was possible to extract 6 features for every window. Which will lead to 6246 windows times 6 features. This leaves us with 37476 features to select from.

2.2.2.4 Feature Selection

After consultation with the team's advisor, it was decided that feature selection will not be used for this multi-label classification problem. As there are not as many features extracted to produce significant noise in the features.

2.3 Classifiers

In accordance with the research papers, the team choose some classifications algorithm that are relevant for the type of classification problem.
Especially the research of Bayat et al.[2] provides valuable information. The results show that Random Forest yields a high accuracy for this type of classification. Random Forest Classifier is an ensemble algorithm. Ensemble algorithms combine more than one algorithms of the same or

[4]This is not the exact result of the division because of the previously mentioned windows with less than 52 samples

different kind for classifying objects. For example, running predictions with Naive Bayes, SVM and Decision Tree and there after making a final consideration on which classification to use. The Random Forest classifier creates a set of Decision Trees from randomly selected subsets of the training set. Afterwards it aggregates the votes from the different Decision Trees to choose the final class of the test object. To have a comparison the team decided to provide also estimator scores for the Super Vector Machine. As we can see in Result it produced similar results, in general Random Decision Forest seems to perform better compared to SVM.

2.4 Validation

For validation two strategies were chosen. On the one hand Cross-validation and on the other hand splitting the dataset in two parts, a training and a test set. The team adopted this approach with the awareness of the differences of the two methods. The main problem of splitting "handmade" data is that there is no right measure of the estimators because the training and especially testing is correlated with the way the data is split. The solution to this problem is a procedure of cross-validation. In the basic approach, called k-fold Cross Validation, the training set is split into k smaller sets. For training and testing each other. The performance measures provided by k-fold cross-validation is then the average of all testing performance values.

3 Implementation

As stated early for implementation we used a modular approach as advised by our lecturer. Therefore, we separated components with different logic in regards to the data. The team decided to create files for Data handling, finding and handling missing data, feature extraction and selection, classification and one file to ensure the workflow of the code. This approach makes it easier to reuse logic and especially to use other datasets in the future.

3.1 Data Handling

This file consists of two functions: read and count_labels. Whereas the first one is essential for importing the data the second one is merely of informative value. As it provides us with the number of instances of each label. The former one reads one of the 15 datasets and the input parameter is the number of the dataset which should be imported (1-15). The function returns a Panda Dataframe which afterwards can be further processed.

3.2 Missing Data

As described in Section Missing Data, handling of missing data is an important step for classification. In this file the team created one function for eliminating labels with the value '0'.

3.3 Feature Extraction and Selection

In regards to feature extraction the team implemented two rather complex functions to firstly sequence the data and secondly extract the two feature types discussed in Section Feature Extraction. The former one takes an numpy array of data and sequences it into windows. One important algorithm sequence is the safety check that the labels of the sequences are the same at the beginning and the end of the sequence, If this is not the case the window will be a little bit smaller until the first and the last label are identical. The latter function extracts the mean value and standard deviation of all x-,y-,z- axis values from all windows.

3.4 Classification

As described in Section Classifiers we implemented two functions:

1. One function to classify the data with Random forest and SVM, but without Crossvalidaion. The dataset was split in two parts for training and testing. Furthermore the team provide the estimators of F1-Score, Accuracy and the Confusion Matrix.
2. One function to classify the data with Crossvalidation and for this also estimators of the Mean Accuracy and F1score with their Standard Deviation.

3.5 Main file (Activity_Recognition)

This file has the intent to call all the previously described functions with the right parameters in the right order to make an executable program. As mentioned earlier this modular approach makes re-usability easier.

4 Results

The classification of the activities yielded an accuracy score of ~92 percent for dividing the features in training and test set. For the Crossvalidation with a 5-fold it produced an accuracy of ~80 percent. This is explainable through the more strict testing in the Crossvalidation as the average score of all the testing folds are used for the final accuracy score.
Result data:

Accuracy Random forest (one training and test set): 0.923200
Accuracy with SV (one training and test set: 0.868800

Confusion Matrix Random Forest:

$$\begin{vmatrix} 242 & 0 & 0 & 2 & 0 & 0 & 7 \\ 0 & 1 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 44 & 6 & 0 & 0 & 39 \\ 0 & 0 & 0 & 199 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 17 & 1 & 3 \\ 0 & 0 & 0 & 22 & 0 & 5 & 1 \\ 2 & 0 & 3 & 1 & 0 & 0 & 646 \end{vmatrix}$$

Confusion Matrix with SVM:

$$\begin{vmatrix} 243 & 0 & 0 & 1 & 0 & 1 & 6 \\ 0 & 0 & 0 & 3 & 1 & 0 & 1 \\ 0 & 0 & 1 & 6 & 0 & 0 & 82 \\ 0 & 0 & 2 & 190 & 7 & 1 & 3 \\ 0 & 0 & 0 & 2 & 18 & 0 & 2 \\ 0 & 0 & 0 & 24 & 0 & 2 & 2 \\ 9 & 1 & 6 & 3 & 1 & 0 & 632 \end{vmatrix}$$

Crossvalidation:
Accuracy Random Forest: 0.80 (+/- 0.31)
F1 Score Random Forest: 0.80 (+/- 0.31)
Accuracy SVM: 0.80 (+/- 0.31)
F1 Score SVM: 0.80 (+/- 0.31)

Scores for the test folds (Random Forest) [0.79456435 0.92246203 0.90472378 0.88701923 0.50521251]
Scores for the test folds (SVM) [0.80655476 0.90647482 0.89671737 0.90144231 0.50200481]

Especially the falsely classified label for Talking while Standing instead of the true label Standing has enormous effect on our accuracy score. (Random Forest only 44 instances were rightly classified as Standing and 39 were falsely classified as Talking while standing). The reason for this is that the difference between talking and standing and only standing are not really traceable in reality.

5 Conclusions

As could be seen in the results the activities were nearly all rightly classified. The only outliers are, as explained in the previous chapter, the two activities Standing while Talking and Standing. For further research it would be advisable to make clearer separation for the activities.
One limitation of the team's approach is to take more datasets into consideration, as a result the classifiers could be further trained and tested. Furthermore, a broader spectrum of features and a following feature selection could yield in higher accuracy.
To sum up the accuracy of the classification can be improved if the separation between the activities would be clearer as previously mentioned and more data and features would be used, in retrospective the team concentrated on the process of generating a good classification with neatly preprocessed data.

Notes

[1]Media Anugerah Ayu, Siti Aisyah Ismail, Ahmad Faridi Abdul Matin, and Teddy Mantoro. A comparison study of classifier algorithms for mobile-phone's accelerometer based activity recognition. Procedia Engineering, 41:224–229, 2012.

[2]Akram Bayat, Marc Pomplun, and Duc A Tran. A study on human activity recognition using accelerometer data from smartphones. Procedia Computer Science, 34:450–457, 2014.

[3]Pierluigi Casale, Oriol Pujol, and Petia Radeva. Human activity recognition from accelerometer data using a wearable device. Pattern Recognition and Image Analysis, pages 289–296, 2011

[4]Wenchao Jiang and Zhaozheng Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In Proceedings of the 23rd ACM international conference on Multimedia, pages 1307–1310. ACM, 2015.

[5]Adil Mehmood Khan, Ali Tufail, Asad Masood Khattak, and Teemu H Laine. Activity recognition on smartphones via sensor-fusion and kda-based svms. International Journal of Distributed Sensor Networks, 10(5):503291, 2014.

[6] Python Data Science Handbook Essential Tools for Working with DataBy Jake VanderPlasPublisher: O'Reilly MediaRelease Date: November 2016 Pages: 541

[7]Mannini, A., Sabatini, A.M.: Machine Learning Methods for Classifying Human Physical Activities from on-body sensors. Sensors 10, 1154–1175 (2010)