P22 - Activity recognition from time series using supervised classification algorithms

Alex Beltran and Oihane Cantero

Abstract

Activity recognition from time series is a difficult yet really important machine learning problem. From recognising simple actions like raising a hand to more complex uses like complex activity recognition, the ability of taking data from a number of sensors (usually accelerometers) and precisely predict the actions of the individual wearing it have been proven useful in a lot of industries and/or sectors. Of course, we cannot ignore the influence of new smartphones, more advanced than ever, already equipped with increasingly amount of sensors like gyroscopes, accelerometers...

Our particular intent with this project is to give an use to that data collected by an accelerometer to classify actions made by humans.

Contents

1 Description of the problem

In this project, we receive data from 15 different users and we want to know what kind of activity they are practising. Data is collected from a wearable accelerometer mounted on the chest. Each user's data forms a data-set and each one has 3 attributes (x, y, and z axis' acceleration), apart from a label that identifies the action being done.

We have to classify the data into 7 different classes that are the activity they are practising:

- 1: Working at Computer.
- 2: Standing Up, Walking and Going up or down stairs.
- 3: Standing.
- 4: Walking.
- 5: Going Up and Down Stairs.
- 6: Walking and Talking with Someone.
- 7: Talking while Standing.

Each database (One for each user, so it ends up being 15 databases), has a different amount of instances (for example, the first user has 162500, and the second user 138000). Also the distribution of the data is different for each user, so we cannot assume the data-set is balanced for any of the users. As a curiosity, the data-set of the first user is clearly unbalanced, as it has 928 instances for the second class (Standing Up, Walking and Going up or down stairs), where the rest of the classes have thousands more of examples.

2 Description of our approach

2.1 Research

Before starting to work on these data-sets, we had to do a deep research into the topic and the own data-sets to see what we were working on. Research on the data-set was done by simple computing, taking a look at how the data-sets really were, and thanks to it we found out really important characteristics of the data-sets that would be useful later (How the labels looked to be in order, a missing value at the end of every data-set labeled 0, how the label names made no practical sense and the frequency of the accelerator of 52Hz), mainly in the preprocessing steps.

On a more theoretical level, we had to resort to both the papers given by the promoter of the project and some internet research for more "specific" matters (Like low-pass / high-pass filters and some wave theory). Research made by Andrea Mannini and Angelo Maria Sabatini has proven key at the time of creating good and useful statistical features on the Feature Extraction steps. Likewise, the findings by Media Anugerah Ayu, Siti Aisyah Ismail, Ahmad Faridi Abdul Matin and Teddy Mantoro were plenty of help at the time of choosing classifiers for the problem. Lastly, the works of Pierluigi Casale, Oriol Pujol, and Petia Radeva have been key at the time of creating the theoretically most important features during the Feature Engineering steps.

2.2 Preprocessing

2.2.1 Data columns renaming

As mentioned before, during the analysis of the database we noticed that the columns of the database had names that didn't follow any sort of logic that we could guess, so we decided to use Panda's features to rename every column to their correct meaning.

We renamed every column in the following way:

- Column 1: sequential_number.
- Column 2: x_acceleration.

- Column 3: y_acceleration.
- Column 4: z_acceleration.
- Column 5: Label

With these changes, the coding was both simpler and cleaner, and made the database more visual.

2.2.2 Missing Data

During our database analysis, we found out that the last instance of each one of our databases contained a "0" labeled instance that made no sense on the context of the problem. This "0" label didn't appear anywhere else on any of the databases, so we decided that the way of solving this inconvenience was simply deleting the tail of every data-set. We could also know that there were no more "0" labeled instances, since the instances where ordered using the labels (from 1 to 7).

Of course, deleting instances is not always the best approach for incorrect data representations, but since it was a common factor and only 15 instances between all databases (every tail of every data-set), this was a feasible solution for the problem.

2.2.3 Feature Extraction

Feature Extraction is usually a really important step on every classification task / problem. Its usually even more important on these activity recognition problems since we work with a really reduced number of features (in our case, 4 not including the class label).

2.2.3.1 ***Separating gravity from user acceleration, AC/DC components.*** Although the raw accelerometer data is useful, it doesn't give us any real information about the real work tasks the users are doing. We can try to work mathematically around this problem, trying to engineer new features from our current data that will allow us to find those features that tell us more about whats going on in real life.

For this, we can separate the gravity from the user acceleration. We take each of these accelerations over time (in samples, of course) and filter them, so we differentiate between their AC and DC components. The AC component will be the high frequency component and is mostly related to the dynamic motion the subject (aka: walking, running, etc). On the other hand, the low-frequency component of the acceleration signal, the DC component, is tied to the influence of gravity on the lectures.

We separate both of these using high-pass filters and low-pass filters. As their names mention, they separate high frequencies and low frequencies of a signal. Windowing can be applied to these methods depending on the way of applying the filters. We decided not to, since the AC DC component differentiation already gives us really good results.

2.2.3.2 ***Magnitude, Standard Deviation and Mean value.*** After obtaining these new AC, DC components, we decided to compute new statistical features that have been showed as beneficial for the predicting process (as shown on the research made by Andrea Mannini and Angelo Maria Sabatini).

Simple statistical descriptors, such as the variance, are widely used; the variance is computed by taking the average of the squared data samples within each frame, or in our case, one instance of the data.

Again, since we are working instance wise, and not window wise (Working with windows means we have to make windows of data and work on those windows), we calculate everything in a vectorial way.

We added the standard deviation and mean of the accelerations (instance wise). We also obtained a new feature (as done in several other papers) by rooting the sum of the squared elements, creating a "magnitude acceleration" (the magnitude of acceleration across all directions).

These magnitudes are calculated using the AC components only, because we want the user movement and not the gravity affecting it.

2.2.3.3 ***Data shuffling:*** As we mentioned on the early research of the database, we found out that the labels seemed to be in order, so that makes the instances follow the numerical order of the target labels (the numbers that define the action or target variables). To solve this, we decided to shuffle the data randomly (after adding all the new features, of course).

2.2.3.4 ***Feature Selection:*** Since we are still working with a limited amount of feature vectors (or columns in Pandas), we decided that feature selection will not be used for this problem since there are not enough features extracted to produce significant noise within the entire group of features.

2.3 Classifiers

In accordance with the research papers, the we choose some classification algorithms that are relevant for this kinds of problems. As mentioned on the research section, we have based our choices of classifiers on the paper made by Media Anugerah Ayu, Siti Aisyah Ismail, Ahmad Faridi Abdul Matin and Teddy Mantoro. Note that we have decided to test several classifiers on every user, since we have no a priori information that can suggest one classifier is gonna work better than others on an X databse (user).

The classifiers whe have chosen are:

- Gaussian Naive Bayes.
- Decision Tree.
- Random Forest.
- KNN.

2.4 Validation

To validate our classifiers and see their performances, we decided to use both accuracy score and the mean accuracy value of a 100 fold cross-validation, so the training set is split into 99 small train splits and 1 small test split. Outside the cross validation, in order to separate both train and test split, we stepped away from the normal splits used in the literature, by doing a 50/50 split of each database. This is because, since we noticed that the instances were probably ordered, even if we shuffled them in the preprocessing step, we decided to not take chances. The 50/50 split was done by sending odd instances to one set and the rest to the other set. The accuracy score is calculated out of the classical predicting method, via first fitting and then consequently predicting with each one of the classifiers. We also printed the confusion matrix for this method.

As a note, for the Random forest, we decided not to apply the cross-validation process, since it ended up being really costly computationally, making the time to execute our classifiers and validation process on each one of the 15 databases rise exponentially.

3 Results

In the following tables we can see the results obtained by each classifier for the 15 users:

First User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0,834048257256268	0,838327481948462
Decision Tree	0,99985951090646	0,999892853624948
Random Forest	0,988804460655492	(-Not computed-)
KNN	0,999408522276353	0,99961859336063

Table 1: Table of the results of the first user.

Second User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8666502413078451	0.8673323349202843
Decision Tree	0.9999565211090016	0.999927536231884
Random Forest	0.9938839693328889	(-Not computed-)
KNN	0.999492746271685	0,9996159367741799

Table 2: Table of the results of the second user.

Third User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.905294221110438	0.9078063809720183
Decision Tree	0.9999022845863706	0.9999218272696726
Random Forest	0.9990619320291583	(-Not computed-)
KNN	0.9992378197736911	0,9996482561247558

Table 3: Table of the results of the third user.

Fourth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.7921897248727475	0.791225902571607
Decision Tree	0.9999181656000916	0.9999345736439067
Random Forest	0.9876593724938215	(-Not computed-)
KNN	0.9995089936005499	0,9996890405357769

Table 4: Table of the results of the fourth user.

Fifth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8794359929499119	0.8942859987260566
Decision Tree	0.9994374929686621	0.9996687496286216
Random Forest	0.997062463280791	(-Not computed-)
KNN	0.9997749971874649	0,99986249589582

Table 5: Table of the results of the fifth user.

Sixth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.9999006373404874	0.8531032018738938
Decision Tree	0.9994374929686621	0.9999289922303146
Random Forest	0.9847691237632897	(-Not computed-)
KNN	0.9995315760337266	0,999602589159528

Table 6: Table of the results of the sixth user.

Seventh User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8774831593025681	0.8797963636549488
Decision Tree	0.9999509196431858	0.999950923946874
Random Forest	0.9947606719100848	(-Not computed-)
KNN	0.9994601160750438	0,9995889153001754

Table 7: Table of the results of the seventh user.

Eighth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.6656908071131465	0.6801419357091835
Decision Tree	0.9998550703633386	0.9999275203532322
Random Forest	0.9910143625269932	(-Not computed-)
KNN	0.9994492673806866	0,9996519946554665

Table 8: Table of the results of the eighth user.

Ninth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8407801460974703	0.8437981577407041
Decision Tree	0.9999040410704219	0.9999460358908169
Random Forest	0.9926711367534695	(-Not computed-)
KNN	0.9995202053521093	0,9997361711044361

Table 9: Table of the results of the ninth user.

Tenth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.7974573731446868	0.7952926911894213
Decision Tree	0.9998895881638512	0.999889676499485
Random Forest	0.9914509692581902	(-Not computed-)
KNN	0.9994952601776054	0,9995819926465159

Table 10: Table of the results of the tenth user.

Eleventh User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8292738970588235	0.8398722237953479
Decision Tree	0.9999234068627451	0.9999234262405831
Random Forest	0.9903684129901961	(-Not computed-)
KNN	0.9992723651960784	0,999454138636867

Table 11: Table of the results of the eleventh user.

Twelfth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8942284219703575	0.8968354474578011
Decision Tree	0.9999825632083696	0.99991282331842
Random Forest	0.9862772449869224	(-Not computed-)
KNN	0.9992502179598953	0,9995902872068193

Table 12: Table of the results of the twelfth user.

Thirteenth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.7982497634815515	0.813936263579161
Decision Tree	0.9999408703878903	0.9998815683667612
Random Forest	0.9707308420056765	(-Not computed-)
KNN	0.9991426206244087	0,9994825291536905

Table 13: Table of the results of the thirteenth user.

Fourteenth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8632706851108546	0.8545316348697917
Decision Tree	0.999896639046323	0.9999138746404076
Random Forest	0.9907836482971283	(-Not computed-)
KNN	0.9994315147547761	0,9996640451524658

Table 14: Table of the results of the fourteenth user.

Fifteenth User		
Classifier:	Accuracy:	Mean 100-Fold cross-validation result:
Gaussian Naive Bayes	0.8099480183191946	0.8183075894634885
Decision Tree	0.9999420278652728	0.9999227425824213
Random Forest	0.9768304701540126	(-Not computed-)
KNN	0.9993043343832732	0,999526467208758

Table 15: Table of the results of the fifteenth user.

Small Note: Sadly, to stay between the page constraints of the requirements, we cannot show the confusion-matrix for the classifiers. This confusion matrix can be found by executing the jupyter notebook that goes alongside this paper or, if preferred, view in the result files attached alongside this paper.

4 Conclusions

From these results we can infer a couple of thoughts / conclusions:

- The best classifier overall for every user was the decision tree, getting an almost perfect score on every data-set (user). This is specially surprising since the Ensemble of decision trees (the random forest), although it did exceptionally well, couldn't match the simpler version of itself that is the decision tree. This proves how the more complex models are not always the best, since for this task, the decision tree got better results and faster, since the Random forest takes more time to compute.
- The second most surprising feature we can get from these results were the really good results given by the KNN model. Being the simplest model on the table, it had results that could perfectly substitute the decision tree. At the start we decided to include this simpler model to see how it managed against the more "expected to do good" models. Once again, also being the fastest one to compute from them all, it shows how more complexity does not necessarily mean better results.
- The Naive Bayes was a bit of a two edged sword. Even tho it got decent results on some users, we can see how in users like the eighth user (see: Table 8), the model becomes borderline poor. One conclusion we can take out of this is how this model is way to variable depending on the user.

Once more, the decision tree is the best model we have found for this problem, and we think that its the best available model for when working with several users. But, seeing the good KNN results for this problem, we can also advice that, when not a lot of computational room is available, KNN can be expected to give really good results no matter the user.

It was also surprising that we expected different classifiers to be the best ones on each user, when this wasn't the case.

On new advancements that could be made, it would be interesting to mix the windowed feature selection approach usually used in the literature alongside the vectorial approach showed here, and see the results of both combined. Also it would be interesting to work on more standardized databases in order to avoid having to create separate databases for users. Maybe an interesting approach would be to create a new feature that specifies the user, but that's outside the scope of this paper.

Finally, we think that the overall accuracy could be improved by a cleaner separation of the data, as we think that the tasks of standing pp, walking and going up or down stairs and the task of standing are way too similar.

5 Bibliography

- Wikipedia: Low-pass filters.
- Wikipedia: High-pass filters.
- *Machine Learning Methods for Classifying Human Physical Activity from On-Body Accelerometers* by Andrea Mannini and Angelo Maria Sabatini.
- *A Study on Human Activity Recognition Using Accelerometer Data from Smartphone* by Akram Bayat, Marc Pomplun, Duc A.Tran
- *A comparison study of classifier algorithms for mobile-phone's accelerometer based activity recognition. Procedia Engineering, 41:224–229, 2012.* by Media Anugerah Ayu, Siti Aisyah Ismail, Ahmad Faridi Abdul Matin, and Teddy Mantoro.
- *Human activity recognition from accelerometer data using a wearable device. Pattern Recognition and Image Analysis, pages 289–296, 2011.* by Pierluigi Casale, Oriol Pujol, and Petia Radeva.
- *Activity recognition on smartphones via sensor-fusion and kda-based svms. International Journal of Distributed Sensor Networks, 10(5):503291, 2014.* by Adil Mehmood Khan, Ali Tufail, Asad Masood Khattak, and Teemu H Laine.