IN4320 Machine Learning

Final Assignment

Author:
Milan Niestijl, 4311728

July 9, 2017

Introduction

In this assignment, the task is to build a classifier that optimally predicts the test set of the "Human Activity Recognition" dataset. The goal is thus not to obtain a classifier that generalizes well to other problems, but one that is specifically designed for this six-class problem. Three of the classes correspond to highly active movements, whereas the other three correspond to relatively low activity. The available training (X_{trn}) and test (X_{tst}) set contain 5580 and 4719 instances, respectively, both having a 554-dimensional feature space. Lastly, it is known what subject the different training instances stem from, which may be used to improve the performance of the classifier somehow. In this report, it is discussed how the resulting classifier was obtained. Both the reasoning and the considerations that led to the final classifier are discussed. A final score on the test set of **TODO** was achieved.

Data analysis

It is important to get some understanding of the data. We therefore firstly use principal component analysis (PCA) to visualise the data. In figure 1, the singular values of the training data are shown. It can be seen that a large portion of the variance in the data is captured in the first few principal components, indicating that the data in the reduced space of the first three components is reasonably representative of the complete data. In figure 2, the resulting plot is shown.

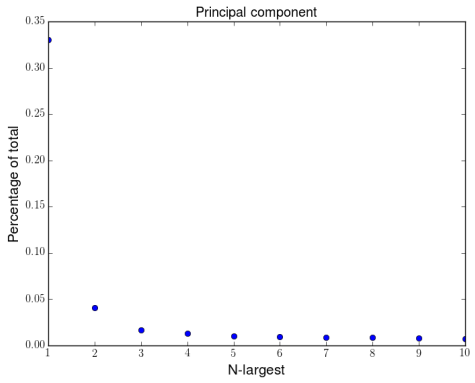

Figure 1: Percentage of the variance coming from the first ten principal components.

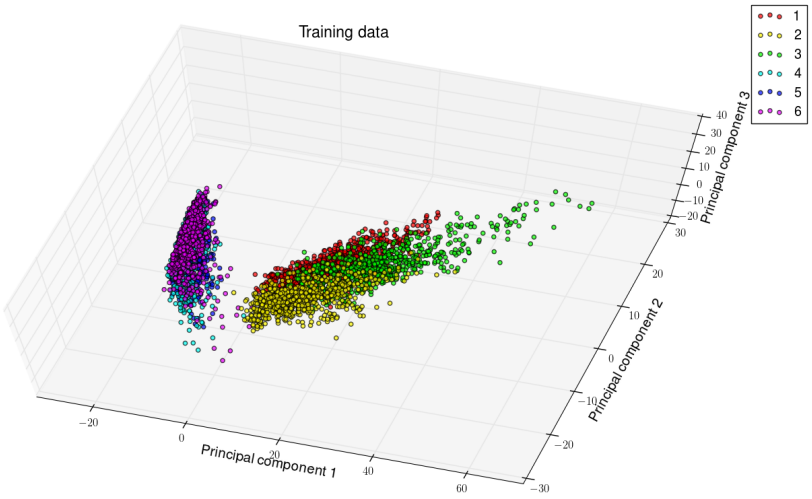

Figure 2: Plot of the training data in the reduced space of the first three principal components

We observe firstly that the data corresponding to the active movements seem separable from those of inactive nature. This becomes even more clear in figure 3, in which the activity type is plotted instead of the labels of the instances. The labelling of the activity types shown in figure 3 will be used to distinguish the two and specify which is meant, as it is not known which of the two corresponds to the high (or low) activities. Lastly, the data corresponding to the two activity types have a clearly different shape. It therefore makes sense to plot these separately. In figures 4 and 5, these separate plots are shown.

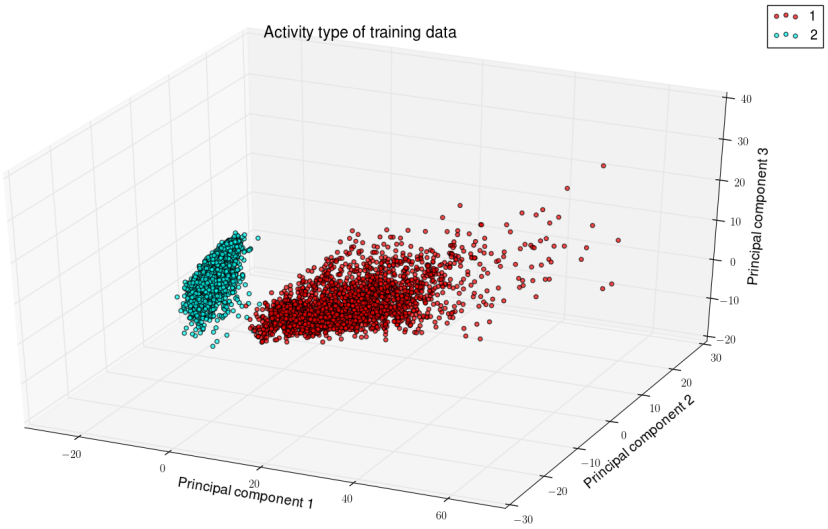

Figure 3: Plot of the activity types of the training data

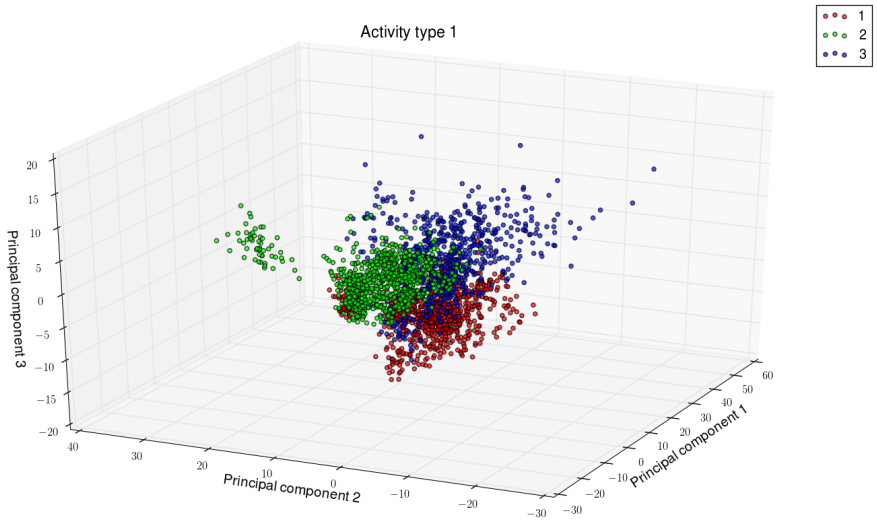

Figure 4: Plot of the training data corresponding to activity type 1 in the reduced space of the first three principal components

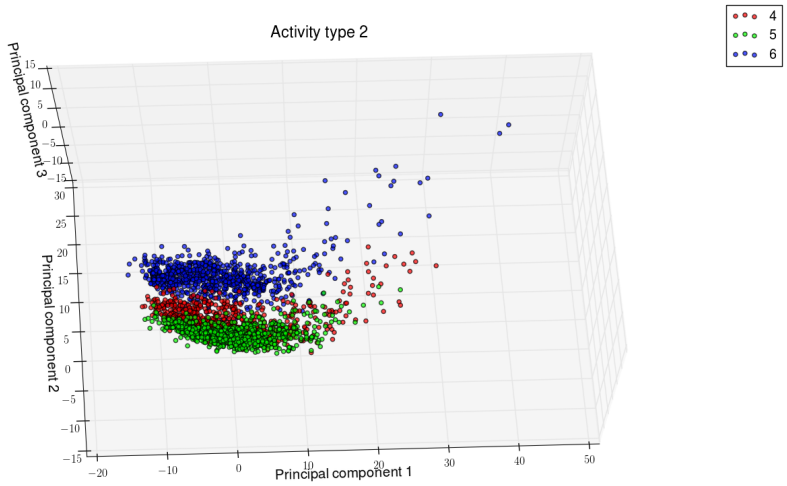

Figure 5: Plot of the training data corresponding to activity type 1 in the reduced space of the first three principal components

Based on these images, it seems that the labels corresponding to activity type 1 are not completely separable. On the other hand, the ones of activity type 2 seem separable by some non-linear manifold.

As the identifiers are also given, we are enabled to look at the data coming from a single subject for both activity types. This is shown for a few subjects in figure 6.

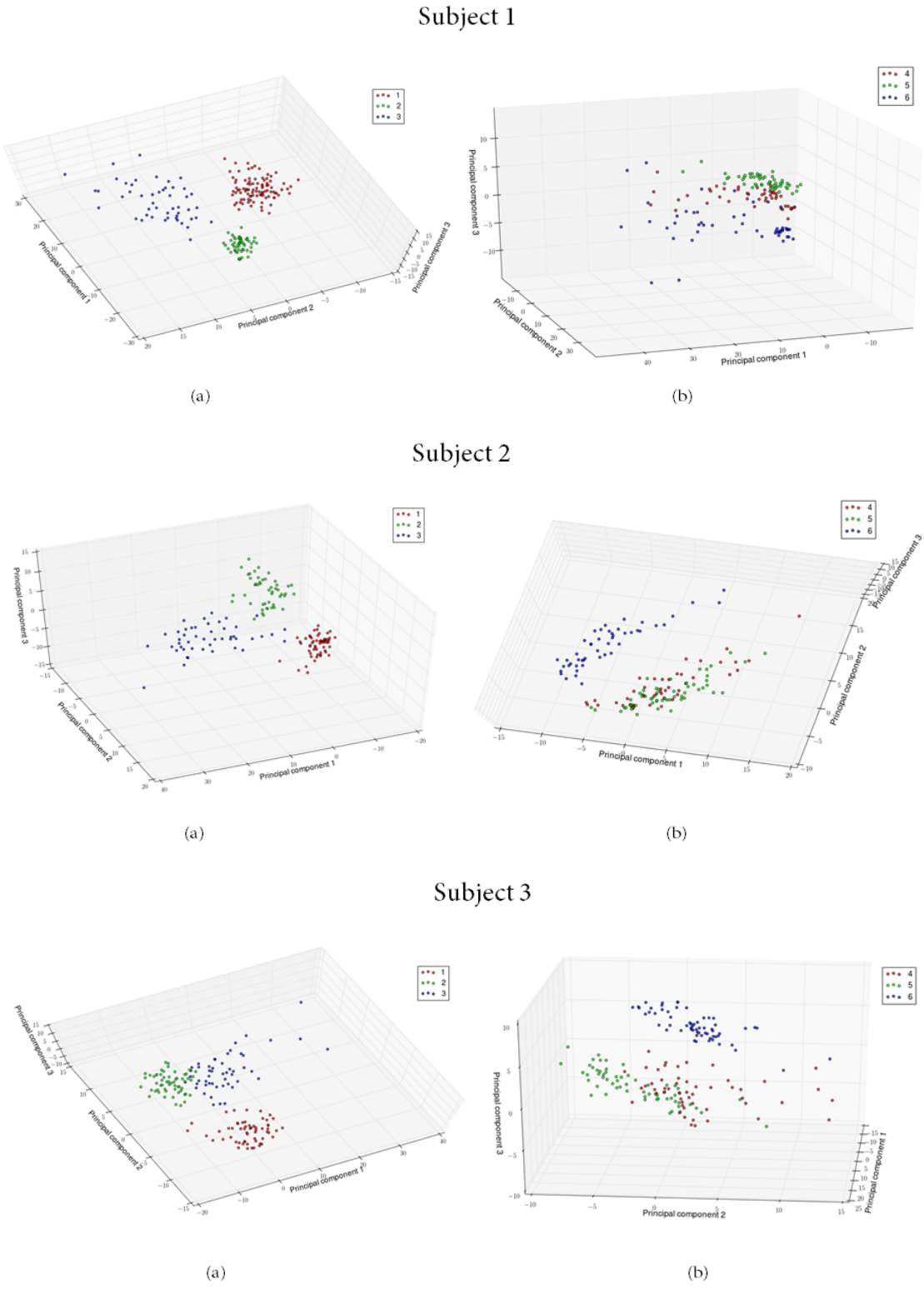

Figure 6: Plot the data coming from one of the subjects. Both the instances of activity type 1 (a) and type 2 (b) are shown.

Firstly, we observe that per subject, the labels of activity type 1 form (separate) clusters. Secondly, class 6 seems separable from class 4 and 5, but the latter two do not seem separable. This also corresponds with figure 5.

Finally, we plot both the training and the test set in a single plot in figure 7, as to verify that the training

set is representative for the test set, for if this is not the case, one could not expect the classifier trained on the training set to perform well on the test set. It may be that some correction has to be applied, such as in the covariate shift setting. However, in this case it does seem that both datasets are drawn the same distribution, or at least very similar ones.

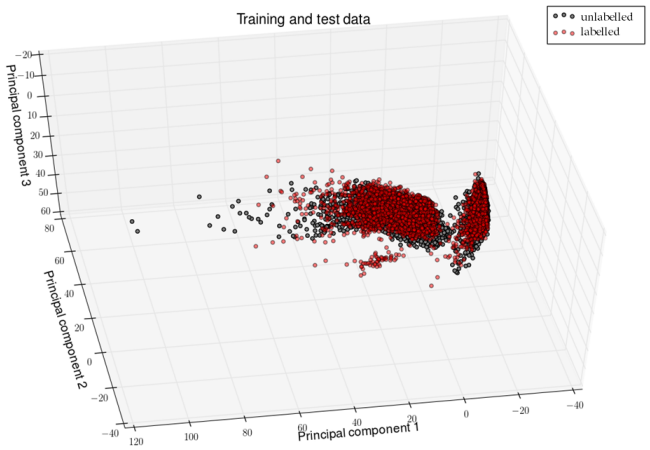

Figure 7: Plot of both the training data (labelled) and the test data (unlabelled).

Building the classifier

On the training data

We now start to optimize a classifier for this specific problem. We first try some basic classifiers on the training data to find out which suit the problem well and to establish some baseline on the accuracy. In table 1, the accuracy of a few classifiers on the training set is shown, evaluated using 10-fold cross-validation. For the support vector machines (SVM), a trade-off parameter of $C = 10$ was used, but it was found that different values did not influence the performance that much.

Table 1: Accuracy of standard classifiers on the training data, evaluated using 10-fold cross-validation. A random forest classifier with N estimators and criterion A is denoted by RFC(N=N, criterion=A). The criteria 'gini', for the Gini impurity and 'entropy' for the information gain are used.

Classifier	Accuracy
KNN(K=3)	0.881 ± 0.031
KNN(K=5)	0.889 ± 0.027
KNN(K=11)	0.894 ± 0.030
LDA	0.964 ± 0.031
SVC(C=10, kernel='rbf')	0.950 ± 0.036
SVC(C=10, kernel='linear')	0.951 ± 0.032
SVC(C=10, kernel='poly', degree=2)	0.944 ± 0.037
SVC(C=10, kernel='poly', degree=3)	0.955 ± 0.032
SVC(C=10, kernel='poly', degree=5)	0.957 ± 0.032
RFC(N=100, criterion='gini')	0.927 ± 0.051
RFC(N=100, criterion='entropy')	0.931 ± 0.042

Furthermore, on the full training set, the SVM with a polynomial kernel of degree 3 was able to score an accuracy of 1 on the training set. It therefore immediately becomes clear that the training data is separable, contrary to what might be expected from figure 2. This is not necessarily unexpected since the plots only show the data in reduced dimensions. Secondly, it appears that both LDA and support vector machines suit the problem very well.

As the low and high activity instances appear to be separable, an idea is to first train a classifier to predict the activity type, i.e., type 1 or 2, in order to reduce the problem to two separate ones and then train different classifiers on the two sub-problems. Especially since the distributions of data from the

two activity types seem to be different, one might also expect different classifiers to perform better on the two sub-problems. The obtained classifier thus uses three different classifiers. It will be denoted by SP(preClassifier, classifier1, classifier2) (for 'splitting classifier'). In order to find out which classifiers work well on the two sub-problems, the training data is split based on the activity type, yielding X_{trn1} and X_{trn2}, after which once again a 10-fold cross-validation scheme was followed to evaluate the classifiers. The result is shown in table 2

Table 2: Accuracy of standard classifiers on the training data, evaluated using 10-fold cross-validation. A random forest classifier with N estimators and criterion A is denoted by RFC(N=N, criterion=A). The criteria 'gini', for the Gini impurity and 'entropy' for the information gain are used.

Classifier	Accuracy on X_{trn1}	Accuracy on X_{trn2}
KNN(K=3)	0.921 ± 0.041	0.845 ± 0.049
KNN(K=5)	0.922 ± 0.033	0.858 ± 0.050
KNN(K=11)	0.918 ± 0.032	0.872 ± 0.060
LDA	0.982 ± 0.023	0.937 ± 0.043
SVC(C=10, kernel='rbf')	0.962 ± 0.041	0.942 ± 0.040
SVC(C=10, kernel='linear')	0.968 ± 0.028	0.937 ± 0.043
SVC(C=10, kernel='poly', degree=2)	0.960 ± 0.041	0.931 ± 0.043
SVC(C=10, kernel='poly', degree=3)	0.970 ± 0.030	0.943 ± 0.041
SVC(C=10, kernel='poly', degree=5)	0.970 ± 0.030	0.946 ± 0.042
RFC(N=100, criterion='gini')	0.932 ± 0.048	0.925 ± 0.062
RFC(N=100, criterion='entropy')	0.943 ± 0.036	0.925 ± 0.059

It can be seen that classifiers generally perform better on X_{trn1} than on X_{trn2}, indicating that the latter is less separable. Taking further into consideration figures 5 and 6, it seems that especially labels 4 and 5 are problematic.
It also becomes clear that LDA is extremely well-fit for X_{trn1}. This may be explained by figure 6, in which it becomes apparent that the different clusters resemble Gaussian distributions. The distribution of X_{trn1} is a weighted sum of the distributions of the data from the various subjects, which seem well approximated by Gaussian distributions. As a weighted sum of Gaussians is once again Gaussian, X_{trn1} is expected to meet the assumptions of LDA, explaining the good performance of LDA on X_{trn1}.
However, both LDA and a linear SVM do not perform well on X_{trn2}, which is readily explained by figure 5, in which it can be seen that X_{trn2} is not linearly but rather non-linearly separable. The reason behind the superior performance of the polynomial support vector machines of various degrees is therefore clear.

next, we evaluate a few classifiers to find out which separates the two sub-problems best. The result is summarized in table 3.

Table 3: Accuracy of standard classifiers on the activity type of the training data, evaluated using 10-fold cross-validation.

Classifier	Accuracy
LDA	1 ± 0
SVC(C=10, kernel='rbf')	0.9994 ± 0.0014
SVC(C=10, kernel='linear')	0.9998 ± 0.0006
SVC(C=10, kernel='poly', degree=2)	0.9998 ± 0.0006
SVC(C=10, kernel='poly', degree=3)	0.9996 ± 0.0017
SVC(C=10, kernel='poly', degree=5)	0.9996 ± 0.0017

It is thus expected that using the splitting classifier in combination with LDA as both the activity-type and X_{trn1} predicting classifier and a SVM with polynomial kernel of degree 5 performs best on the training data. To verify this, a few combinations are evaluated using 10-fold cross-validation. The result is shown in table 4, which is as expected.

Table 4: Accuracy of the splitting classifier in combination with various classifiers, evaluated using 10-fold cross-validation.

Pre-classifier	Classifier1	Classsifier2	Accuracy
LDA	LDA	SVC(kernel='rbf'))	0.962 ± 0.03
LDA	LDA	SVC(kernel='poly', degree=3))	0.962 ± 0.03
LDA	LDA	SVC(kernel='poly', degree=5))	0.964 ± 0.03
LDA	SVC(kernel='poly', degree=3)	SVC(kernel='poly', degree=5))	0.958 ± 0.03
SVC(kernel='poly', degree=2)	LDA	SVC(kernel='poly', degree=5))	0.964 ± 0.03

On the test data

After having evaluated some classifiers on the training data, we continue to optimize the classifier on the test data.
From now on, let:

CL A := SP(LDA, LDA, SVC(kernel='poly', degree=3))
CL B := SP(LDA, LDA, SVC(kernel='poly', degree=5))
CL C := SP(SVC(kernel='poly', degree=2), LDA, SVC(kernel='poly', degree=3))
CL D := SP(SVC(kernel='poly', degree=2), LDA, SVC(kernel='poly', degree=5))

A basic SVM with polynomial kernel of degree 3 yields a score of 0.94588 and and CL C scores 0.96774. It is thus verified that the classifiers generalize well on the test-data and achieve quite a high accuracy.

As the problem is of transductive/semi-supervised nature, the test data can be used in the training process to increase the performance. Naturally, there are multiple ways of doing this. As many of the test instances were be predicted with high confidence, self-learning seems very promising. A label-propagation approach was also considered, but after training on the full dataset, this yielded a poor score on the training data and was thus discarded. The self-learning is implemented in a manner in which test instances are added to the labelled data if they are predicted with a confidence higher than a certain threshold, which decreases per iteration, from 0.99 to 0.6 in 200 iterations. Furthermore, test instances that are added to the labelled data during the i^{th} iteration are given a weight of γ^i for some $\gamma \in (0, 1)$, which takes into account that labels that are predicted during a later stage have a higher uncertainty, both due to the unstable nature of self-learning (an error early may grow in further iterations), and the lower threshold in a later iteration.

Lastly, the identifiers can potentially be used to boost the performance of the classifier. This is attempted by varying the weight of the samples based on the variance of the data $X_{s,l}$ with some label l belonging to a certain subject s. The idea is that a larger variance in this data corresponds to more uncertainty, so that these instances are given a lower weight. The weight is determined by first fitting a Gaussian distribution on $X_{s,l}$ for each subject and each label, yielding a covariance matrix Σ. We next define $\sigma_{s,l} = tr(\Sigma)$, where $tr(.)$ stands for the trace of a matrix. Thus, $\sigma_{s,l}$ represents the total variance of $X_{s,l}$.
BAGGING / ENSEMBLES / COMBINING

Final thoughts

Wat er nog beter kon: misschien Strn beter gebruiken, transductive SVM zou goed kunnen werken. Vergeet niet code toe te voegen waar nodig!!