EPART - summary of the 2nd lab

Bartosz Latosek

March 2024

All the code for each point is present in corresponding file included in source.

1 Data analysis

Check the data, esp. the training set. Outliers can change significantly computed distribution parameters, which can dramatically reduce recognition quality. You can try here to compare mean and median values, plot histogram of individual features (hist function).
To remove a sample with known index idx use expression:

$$\text{train(idx, :)} = [] \ ;$$

Firstly, class distribution has been analysed according to the following histogram:

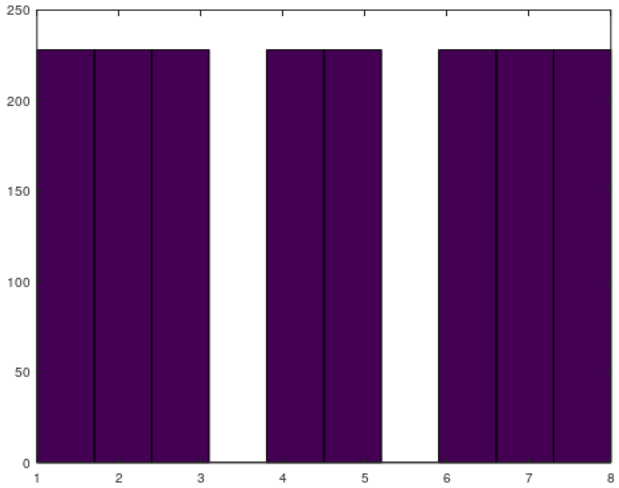

Figure 1: Class distribution

On the above diagram we can see that the class distribution is perfect, the same as in test set (which won't be shown here in order to save space and make the report more concise). Next, corelations between features were plotted to spot possible outlayers.

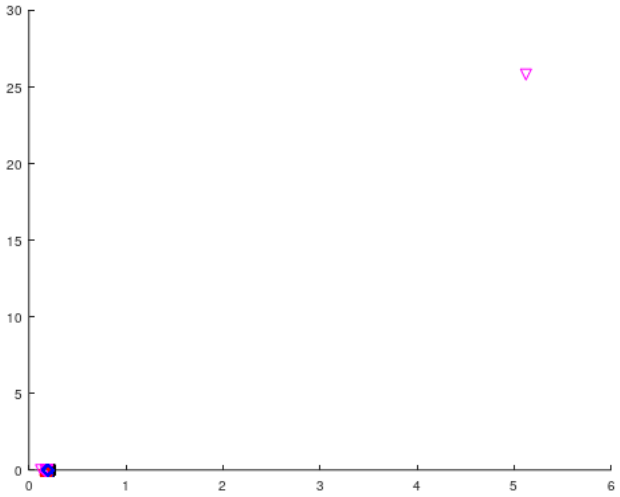

Figure 2: First outlayer

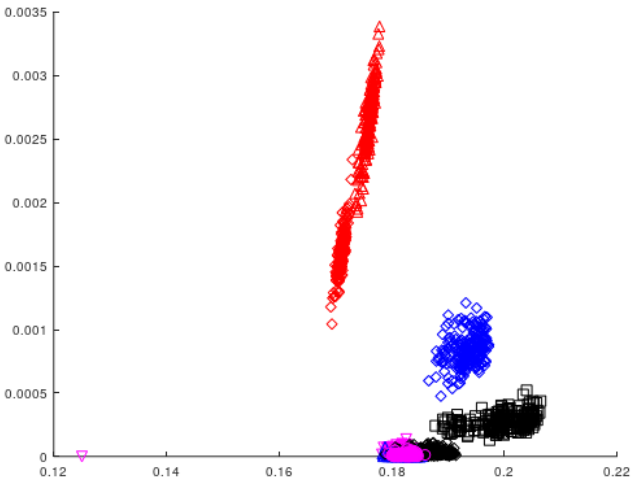

Figure 3: Second outlayer

They were both removed from training set, the rest of the data seemed to be averaged out, so the identified outlayers were [186, 642]. (It turned out to be the exact same for test set, once again the diagrams will be skipped).

2 Feature selection

Select two features (note that you have plot2features function supplied) and build three Bayes classifiers with different probability density computations (according to points 1-3 above). You should use equal a priori probabilities of 0.125.

For this, features 2 and 4 were selected, because on the diagram below we can see that they perfectly split the samples into clusters.

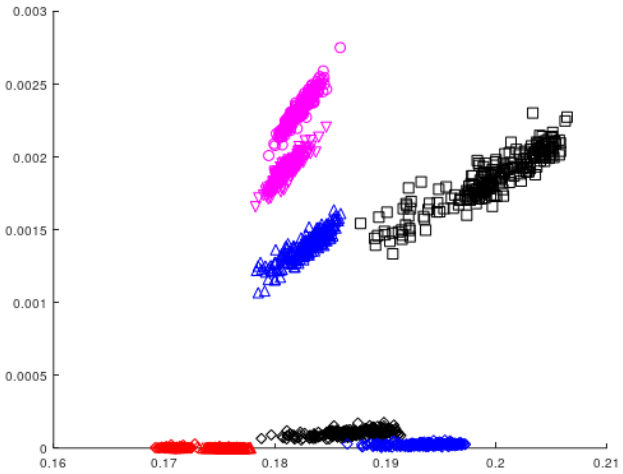

Figure 4: 2 and 4 feature diagram

The received errors can be seen in the table below:

Classifier	**Error**
`pdfindep_para`	2.5796×10^{-2}
`pdfmulti_para`	4.3908×10^{-3}
`pdfparzen_para`	2.3600×10^{-2}

Table 1: Error values for different classifiers.

The received erros are very low, thus we can assume that the two features are enough to create a decent classifier.

3 Number of samples vs classifier quality

Check how the number of samples in the training set influences the classification quality (you can take for example 10%, 25%, 50% of the whole training set).
Note: an appropriate part of the samples from the training set should be drawn independently from each class; because we introduce a random element, the experiment must be repeated (minimum 5 times) and report should contain averaged results (good practice is to include not only mean value but also a standard deviation). Here you should implement reduce function, which leaves the appropriate part of each class. At this point, the reduction applies only to the training set.

It's worth making an assumption, before analysing the results: Usually, the more data - the better. Therefore the expected outcome of gradually increasing the partition size is for the errors to lower accordingly. Let's take a look at the results (the code available in *task3.m* file.)

```
Overall Mean Errors:
   1.8112e-02   2.0527e-02   5.4116e-02
   1.8332e-02   1.6575e-02   2.8101e-02
   1.1196e-02   8.3425e-03   1.2623e-02

Overall Standard Deviations:
   5.2644e-03   3.0950e-03   2.3666e-02
   3.7747e-03   3.0162e-03   2.6150e-03
   2.3797e-03   2.6719e-03   2.6606e-03
```

Figure 5: Errors for different methods

In the picture above, the columns represent different methods used (from left to right: pdfindep_para, pdfmulti_para and pdfparzen_para) and the rows represent different partitions (top to bottom: 10%, 25%, 50%). We can clearly see, that the assumptions made earlier match the given results. Increasing the number of samples in training set, gradually decreases the error values (both the average and standard deviation).

4 Parzen window's width effect

Check how width of the Parzen window h1 influences the classification quality (note that this point has sense for Parzen classifier only).

To assess the impact of the Parzen window width (h1) on classification quality, we shall experiment with different values of h1 and evaluate the corresponding changes in classification performance. The expected outcome is that there should be optimal range of values where the classifier's accuracy is the highest.

Overall: The trend typically observed with Parzen window classifiers is that a larger window size (h1) leads to smoother decision boundaries but may oversmooth the data, resulting in lower accuracy.

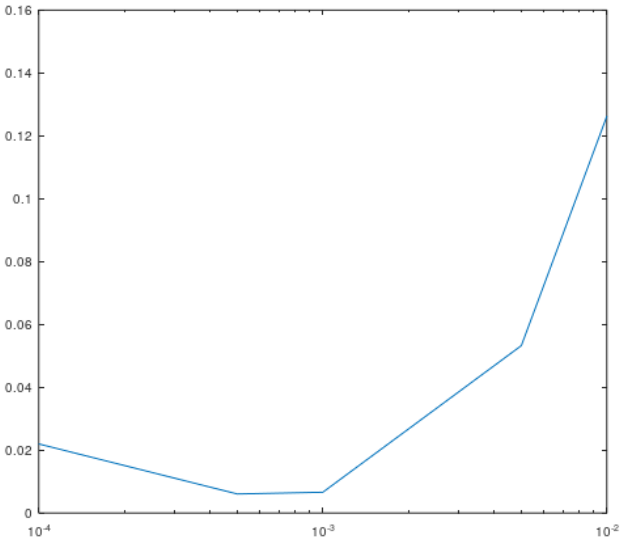

Figure 6: Errors for different h1 values

The above diagram confirms made assumptions. There is an optimal value for the Parzen window's width - arround 0.001. Increasing the window size above that value results in lower classifier's accuracy.

5 Effect of a priori probability change

How will the classification results change if the a priori probability will be two times higher for black suits, i.e. (0.165, 0.085, 0.085, 0.165, 0.165, 0.085, 0.085, 0.165)?

The task states that we should decrease the number of red suits in the TEST data only. If it were the case for the TRAIN data as well, the classifier itself might have become biased towards the black suites (because there would be more of them in the training set) and thus would prefer to classify data as "black suites", decreasing the accuracy of the whole classifier in effect. I am not sure what the outcome should be for decreasing the number of certain classes in the test data - the result may only slightly differ from the default one.

```
Errors without reduction: 0.008782 0.004391 0.006586
Errors with reduction: 0.007315 0.003658 0.005852
```

Figure 7: Result errors

In the picture above, the columns represent different methods used (from left to right: pdfindep_para, pdfmulti_para and pdfparzen_para). We can see, that changing the a priori probability for black suits slightly increased the performance of the classifier (the errors were smaller), but again - it doesn't affect the classifier's performance in any way.

6 Normalising the data

What is the classification quality of the 1-NN classifier (cls1nn.m) for these data? Don't use in this case leave-one-out method, you have large enough testing set at your disposal. Think about data normalization. If there is big difference in standard deviations between features you should normalize data before classification.

The accuracy for the 1-NN classifier for these data is **99.4%** (*task6.m*). That seems like a really good outcome, especially regarding the fact that 1-NN classifier should work poorly on unnormalised dataset.

In theory, we should expect the classifier's performance to improve, when working with normalised data, but it's worth keeping in mind that the two features selected in 2nd point were enough to create a decent classifier, thus the performance may not be affected much by the data normalisation.

Let's take a look at the standard deviations for each of the features of data in the training set:

8.8445e-03 — 9.2062e-04 — 9.5129e-04 — 2.6261e-05 — 1.1688e-09 — 7.6164e-07 — 3.8270e-10

The magnitudes of these values vary significantly, ranging from $10\hat{}(-10)$ to $10\hat{}(-3)$. Such variations in scale among features could impact the performance of the classifiers.

```
Errors without normalization: 0.008782 0.004391 0.006586
Errors with normalization: 0.008782 0.004391 0.762349
```

Figure 8: Result errors

In the picture above, the columns represent different methods used (from left to right: pdfindep_para, pdfmulti_para and pdfparzen_para). Something interesting happened. The first two methods seem unaffected by the data normalisation. It may indicate, that they are not that sensitive when it comes to feature values' ranges. On the other hand - the Parzen method seems to have been affected significantly. This may indicate, that this method is very sensitive to feature value's ranges, and normalisation may not be the best option here.