EPART Lab 2 Report

Rustenis Tolpeznikas

November 4, 2024

Point 1

Task:

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:

train(idx, :) = [] ;

Results:

The class distribution was found as follows:

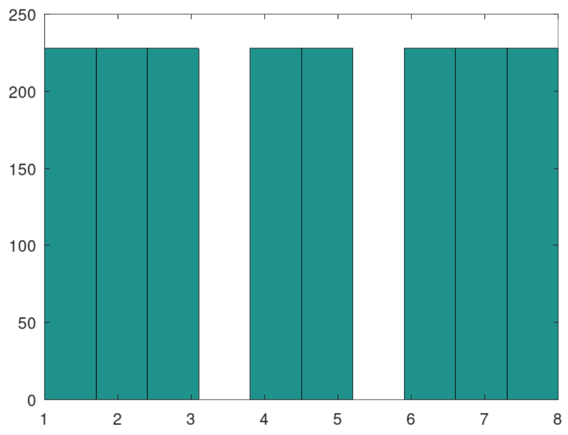
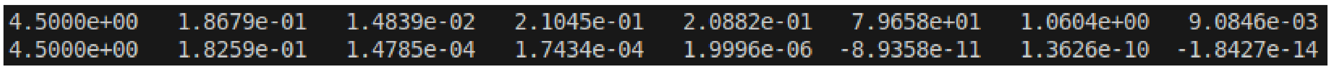


Figure 1: Class distribution histogram

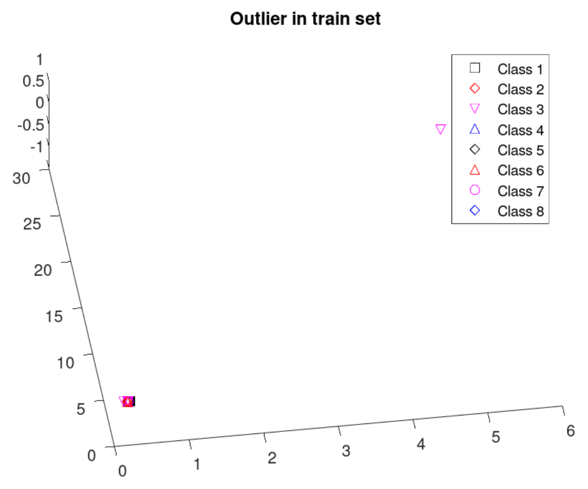
The classes appear to be uniformly distributed, which is preferable. Next, the mean and the median values of the training set were calculated:



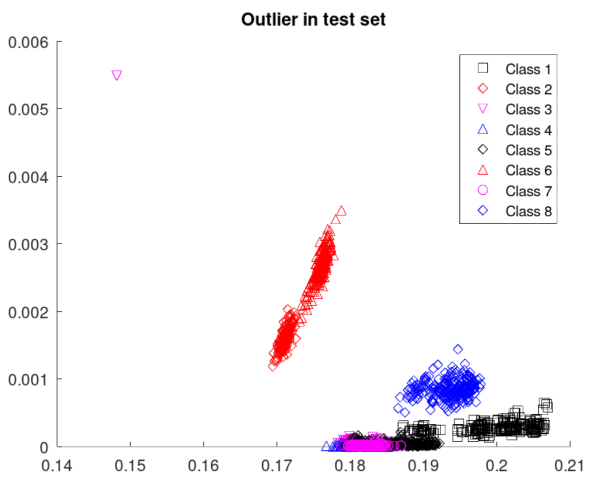
```
4.5000e+00   1.8679e-01   1.4839e-02   2.1045e-01   2.0882e-01   7.9658e+01   1.0604e+00   9.0846e-03
4.5000e+00   1.8259e-01   1.4785e-04   1.7434e-04   1.9996e-06  -8.9358e-11   1.3626e-10  -1.8427e-14
```

Figure 2: Train set mean(top) and median(bottom) values

Usually, the mean and the median values should not differ significantly, but we can see several orders of magnitude difference in several features. This indicates that there are outliers in the data. Therefore, an initial feature plot was created to identify the outliers:



(a) Outlier in top right



(b) Outlier in top left

Figure 3: Outliers found

While the plot for both training and testing sets clearly shows the outliers, it is difficult to pinpoint them graphically. Thus, using *min* and *max* functions, 2 outliers, namely elements 642 and 186, same in the both sets, were identified and removed. Further analysis did not reveal any other outliers.

Point 2

Task:

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.

Results:

The following features were selected by plotting all possible combinations, and looking for the most distinct classes:

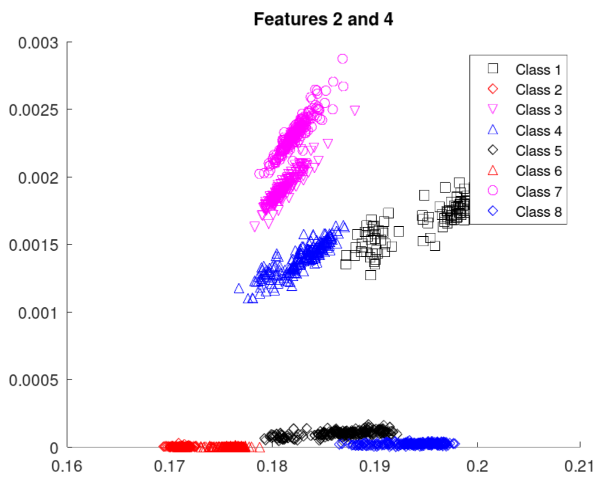
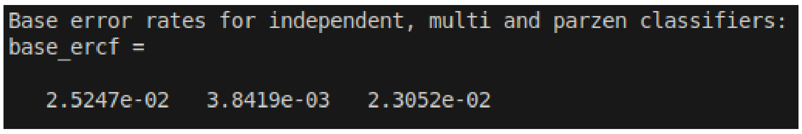


Figure 4: Features 2 and 4 shown the most distinct classes

While there is some overlap, the classes are quite distinct. This setup gave the following base error rates:



```
Base error rates for independent, multi and parzen classifiers:
base_ercf =

   2.5247e-02   3.8419e-03   2.3052e-02
```

Figure 5: Base error rates, with Parzen window width $h_1 = 0.001$

The error rates are quite low, which is expected, given the classes were quite distinct.

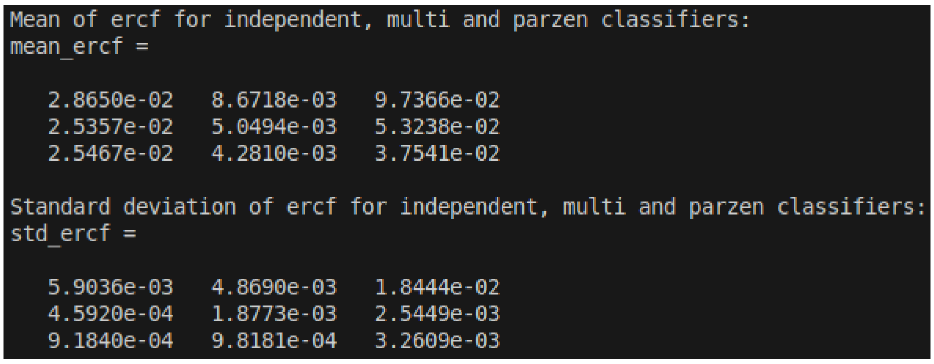
Point 3

Task:

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.

Results:

After implementing the *reduce* function, the 3 classifiers were tested with 0.1, 0.25, and 0.5 of the training set. General assumption would be that the more data is used, the lower the mean error rate and the standard deviation:



```
Mean of ercf for independent, multi and parzen classifiers:
mean_ercf =

   2.8650e-02   8.6718e-03   9.7366e-02
   2.5357e-02   5.0494e-03   5.3238e-02
   2.5467e-02   4.2810e-03   3.7541e-02

Standard deviation of ercf for independent, multi and parzen classifiers:
std_ercf =

   5.9036e-03   4.8690e-03   1.8444e-02
   4.5920e-04   1.8773e-03   2.5449e-03
   9.1840e-04   9.8181e-04   3.2609e-03
```

Figure 6: Mean error rates as well as the standard deviation over 5 runs

The assumption was correct, as with more data, the error rate decreased as well as the standard deviation.

Point 4

Task:

Check how width of the Parzen window h_1 influences the classification quality (note that this point has sense for Parzen classifier only).

Results:

The initial assumption would be that too small or too large window width would result in a higher error rate, due to overfitting or underfitting:

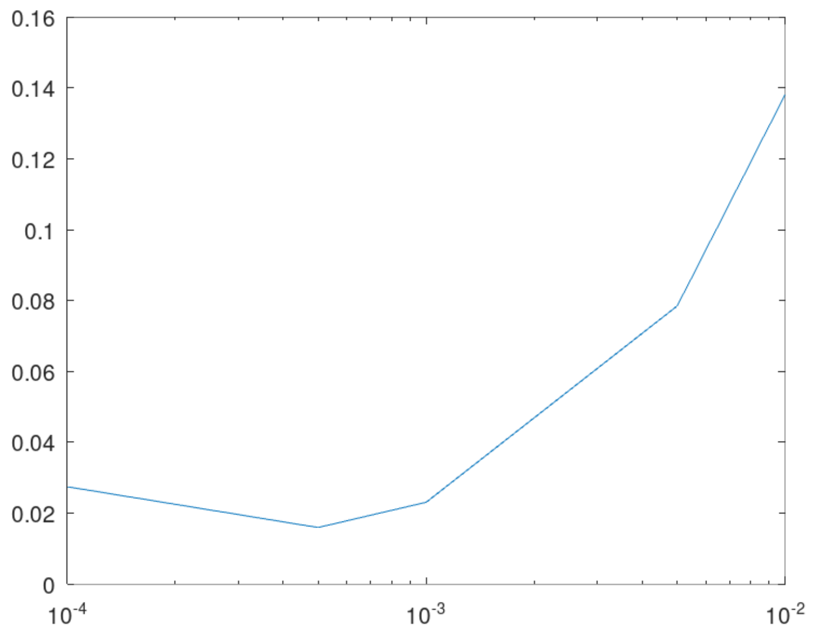


Figure 7: Parzen resolution, given $h_1 = [0.0001, 0.0005, 0.001, 0.005, 0.01]$

Contrary to the previously used $h_1 = 0.001$, the $h_1 = 0.0005$ performed slightly better. However, the initial assumption was correct, as the smallest and largest window widths performed worse.

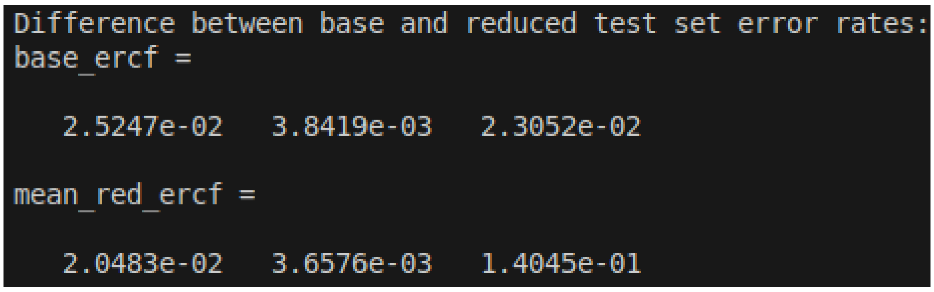
Point 5

Task:

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)?
Note that in this case you should reduce number of red suits in the **testing set** only!

Results:

Given the random nature of the reduce function, the error rate was averaged over 5 runs. These were the results:



```
Difference between base and reduced test set error rates:
base_ercf =

   2.5247e-02   3.8419e-03   2.3052e-02

mean_red_ercf =

   2.0483e-02   3.6576e-03   1.4045e-01
```

Figure 8: Difference between the base error rate and the reduced red suits one

The results were mixed. While the error rate for the independent and the multivariate classifiers decreased, the Parzen classifier performed worse. This could be due to the fact that the Parzen classifier is more sensitive to the changes in the *priori* probabilities, or perhaps the window width was not optimal.

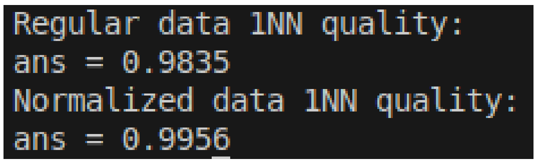
Point 6

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.

Task:

Results:

Since the datasets were small enough to quickly perform the 1-NN classification, both the normalized and the non-normalized data were tested. The normalization method used was standardization, here are the results:



```
Regular data 1NN quality:
ans = 0.9835
Normalized data 1NN quality:
ans = 0.9956
```

Figure 9: 1-NN classification quality

While the normalized data performed slightly better, the difference was in a single percentage point.