# EPART - summary of the 3rd lab

Bartosz Latosek

March 2024

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

# 1 Perceptron function

The perceptron function was tested on two-dimensional data sets:

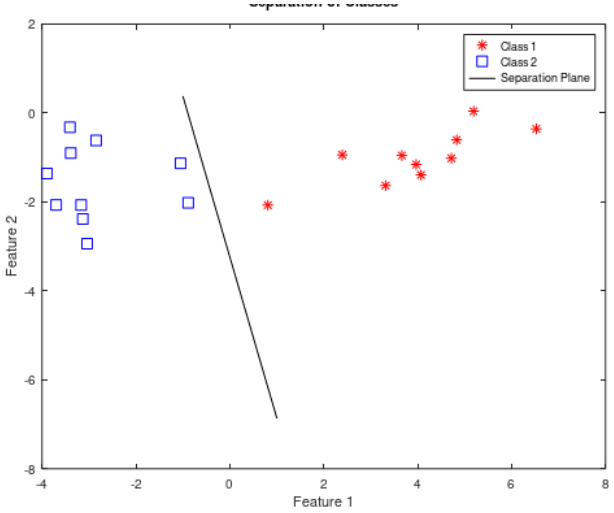


Figure 1: The 2-dimensional data set with separation plane

We can see, that it correctly separates the arbitrarly chosen classes, therefore we can assume that the *perceptron.m* module is complete.

# 2 Multidimensional digits data

For this part of the code, the performance of perceptron was tested on two cases:

- Easy case - *Zeros* and *Ones*
- Difficult case = *Fours* and *Nines*

For the two splits listed above, the average model precisions for positive and negative classes (first in the list is positive - 0 and 4) are as follows (It's worth noting that the accuracies are the average values from 100 repetitions):

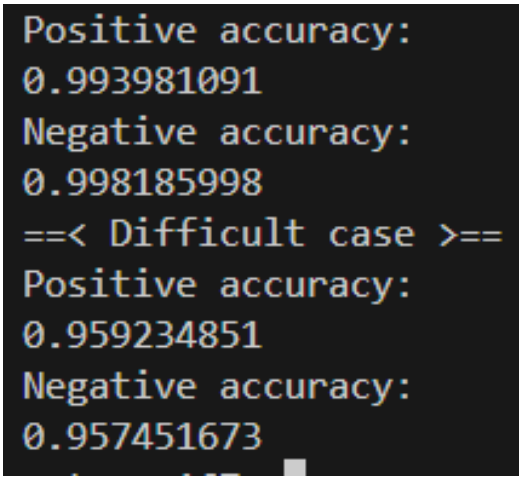


Figure 2: Average accuracy of model

We can clearly see, that the model performs good for both the easy and difficult case (Althrough it's slightly better for the easier case).

## 3 One vs One classification

Testing the *one v one* classification on the validation set returns the following confusion matrix:

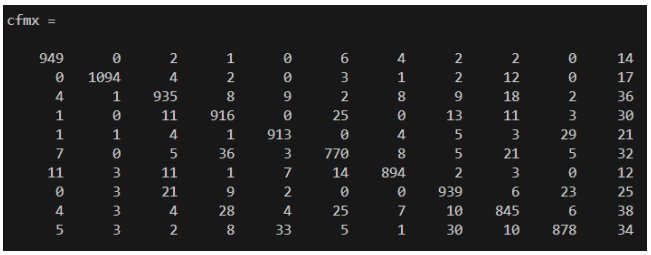


Figure 3: Confusion matrix

The classifier seems the most confused about classifying 7 as a 9 and 9 as a 4.
We can compare the errors for each individual One vs One classifier, looking at the table below:

Table 1: OvO Classifier Comparison

| First Class | Second Class | Error | First Class | Second Class | Error | First Class | Second Class | Error |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0.002447690486 | 1 | 8 | 0.023266894306 | 4 | 8 | 0.014709655349 |
| 0 | 2 | 0.014729399882 | 1 | 9 | 0.004964147821 | 4 | 9 | 0.041048257145 |
| 0 | 3 | 0.008461921354 | 2 | 3 | 0.032178013070 | 5 | 6 | 0.025222682776 |
| 0 | 4 | 0.005354866128 | 2 | 4 | 0.019237288136 | 5 | 7 | 0.011723429745 |
| 0 | 5 | 0.019393511989 | 2 | 5 | 0.027243167238 | 5 | 8 | 0.047285308730 |
| 0 | 6 | 0.010640993159 | 2 | 6 | 0.026860895925 | 5 | 9 | 0.019525065963 |
| 0 | 7 | 0.006317689531 | 2 | 7 | 0.019635114129 | 6 | 7 | 0.005417384881 |
| 0 | 8 | 0.010786478682 | 2 | 8 | 0.034380557202 | 6 | 8 | 0.015549324497 |
| 0 | 9 | 0.009097035040 | 2 | 9 | 0.016544889561 | 6 | 9 | 0.005393106935 |
| 1 | 2 | 0.011574803150 | 3 | 4 | 0.011442412094 | 7 | 8 | 0.014443710796 |
| 1 | 3 | 0.011341567622 | 3 | 5 | 0.050813711911 | 7 | 9 | 0.051170787621 |
| 1 | 4 | 0.005085823268 | 3 | 6 | 0.011370238194 | 8 | 9 | 0.024237288136 |

Figure 4: Errors for individual classifiers

The above table shows the results for a One vs One ensemble classifier with unanimity voting. The smallest

error rate seems to be assigned to the 0 vs 1 classifier - **0.2%**, whilst the largest one is assigned to the 3 vs 5 classifier - **5.1%**. The total number of classifiers in this case is **45**.

The overall statistics for the ensemble are:

- Proper classification - **91.33%**
- Error rate - **6.08%**
- Reject decision coefficients - **2.6%**

# 4 Preparing one-versus-rest ensemble of classifiers

For this part of the task, new functions had to be developed to train the different classifiers. In addition - new voting system was written - *OVRvoting.m* and *OVRvotingcast.m* files. The process can be summarised:

- **Initialization**:
  - The function `OVRvoting` initializes the labels by extracting unique positive class labels from the voting committee matrix `clsmx`. It also sets up a label for rejecting the classification if there is no unanimity among the classifiers.
- **Casting Votes**:
  - The function `OVRvotingcast` casts votes by iterating over each individual classifier defined in the voting committee matrix `clsmx`.
  - For each classifier, it computes the response for all samples using the separating hyperplane coefficients.
  - If the response is non-negative, it increases the number of votes for the positive class of the classifier.
- **Vote Counting**:
  - After casting votes for all classifiers, `OVRvoting` counts the votes for each class by summing the votes across all classifiers for each sample.
  - It also checks if there is unanimity among the classifiers for each sample (i.e., if only one classifier voted positively).
  - If there is unanimity, it selects the class with the most votes as the classification result for that sample.
  - If there is no unanimity, it rejects the classification.

# 5 OVR ensemble validation

The error rates for individual classes are visible in the table below:

Table 1: Error Rates for Each Class

| Class | Error Rate |
|---|---|
| 0 | 0.01113333333 |
| 1 | 0.01011666667 |
| 2 | 0.02736666667 |
| 3 | 0.03153333333 |
| 4 | 0.02473333333 |
| 5 | 0.04678333333 |
| 6 | 0.01640000000 |
| 7 | 0.01955000000 |
| 8 | 0.04986666667 |
| 9 | 0.04471666667 |

We can see the error values are generaly higher than those we observed in the OVO ensemble. The confusion matrix for the test set looks as follows:

```
cfmx =

   949     0     2     1     0     6     4     2     2     0    14
     0  1094     4     2     0     3     1     2    12     0    17
     4     1   935     8     9     2     8     9    18     2    36
     1     0    11   916     0    25     0    13    11     3    30
     1     1     4     1   913     0     4     5     3    29    21
     7     0     5    36     3   770     8     5    21     5    32
    11     3    11     1     7    14   894     2     3     0    12
     0     3    21     9     2     0     0   939     6    23    25
     4     3     4    28     4    25     7    10   845     6    38
     5     3     2     8    33     5     1    30    10   878    34
```

Figure 5: Confusion matrix

The overall statistics for the ensemble are:

- Proper classification - **77.5%**
- Error rate - **3.5%**
- Reject decision coefficients - **18.92%**

The reject decision coefficient is very high for this ensemble, compared to the OvO. This is due to the voting method, where every form of draw is being rejected. We have more rejections than errors, but since the coefficient is high, it's not satisfying.

# 6 Expanding the features of data set

We can combine the available features in pairs and expand the current feature set to 40 (old) + 780(new) features (860 in total). We can now validate the performance of both the OVO and OVR ensembles with the new features.

# 7 Ensembles validation with extended feature set

Code was modified and the feature expansion was added to both OvO and OvR training code.

# 8 Ensembles with and without feature extension comparison

## 8.1 OvO

First of all, the time it takes to train the OvO ensemble has increased drastically. The new error rates are presented in the table below:

Table 2: OvO Classifier Comparison

| First Class | Second Class | Error | First Class | Second Class | Error | First Class | Second Class | Error |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0.002289774970 | 1 | 8 | 0.009132057492 | 4 | 5 | 0.003107520199 |
| 0 | 2 | 0.004965911960 | 1 | 9 | 0.005121739816 | 4 | 6 | 0.005187074830 |
| 0 | 3 | 0.006139040982 | 2 | 3 | 0.010091819009 | 4 | 7 | 0.006938134963 |
| 0 | 4 | 0.005269868253 | 2 | 4 | 0.003898305085 | 4 | 8 | 0.005900966390 |
| 0 | 5 | 0.005200987306 | 2 | 5 | 0.002548554355 | 4 | 9 | 0.012975998643 |
| 0 | 6 | 0.008360780339 | 2 | 6 | 0.003873358033 | 5 | 6 | 0.009083693447 |
| 0 | 7 | 0.003035772891 | 2 | 7 | 0.010144808967 | 5 | 7 | 0.004107479035 |
| 0 | 8 | 0.005180907083 | 2 | 8 | 0.009060885765 | 5 | 8 | 0.008871540099 |
| 0 | 9 | 0.004043126685 | 2 | 9 | 0.005794910557 | 5 | 9 | 0.005101143360 |
| 1 | 2 | 0.008818897638 | 3 | 4 | 0.002672680197 | 6 | 7 | 0.001395387015 |
| 1 | 3 | 0.008467334732 | 3 | 5 | 0.012378808864 | 6 | 8 | 0.006372673974 |
| 1 | 4 | 0.004450095359 | 3 | 6 | 0.004066727529 | 6 | 9 | 0.002359484284 |
| 1 | 5 | 0.004028611362 | 3 | 7 | 0.008712487899 | 7 | 8 | 0.006520303731 |
| 1 | 6 | 0.005529225908 | 3 | 8 | 0.014187948590 | 7 | 9 | 0.016292778778 |
| 1 | 7 | 0.007841931268 | 3 | 9 | 0.010182119205 | 8 | 9 | 0.009406779661 |

Figure 6: Error rates for OvO with feature expansion

We can see that the previously worst case error rate ( 3 v 5 ) has decreased from **5.1%** to **1.2%**. Overall, the quality of the whole ensemble had increased.

The overall new statistics for the ensemble are:

- Proper classification - **95.4%**
- Error rate - **3.4%**
- Reject decision coefficients - **1.16%**

We can see, that the proper classification rate increased by 4% and the error rate and reject decision coefficient decreased by half. This is very promising result and shows the value of feature extension.

## 8.2 OvR

Once again we can observe the increased training time of the OvR ensemble. The new error rates are presented in the table below (in comparison with the previous ones):

Table 2: Error Rates for Each Class

| Class | Error Rate - no feature expansion | Error rate - feature expansion |
|---|---|---|
| 0 | 0.01113333333 | 0.006316666667 |
| 1 | 0.01011666667 | 0.007133333333 |
| 2 | 0.02736666667 | 0.009783333333 |
| 3 | 0.03153333333 | 0.015750000000 |
| 4 | 0.02473333333 | 0.012600000000 |
| 5 | 0.04678333333 | 0.014116666667 |
| 6 | 0.01640000000 | 0.007633333333 |
| 7 | 0.01955000000 | 0.011850000000 |
| 8 | 0.04986666667 | 0.016566666667 |
| 9 | 0.04471666667 | 0.018583333333 |

We can see, that the error rates for indivudual classes decreased further. The new statistics for test set look as follows:

- Proper classification - **90.3%** (prev. *77.5%*)
- Error rate - **2.21%** (prev. *3.5%*)
- Reject decision coefficients - **7.47%** (prev. *18.92%*)

The accuracy increased by roughly *13%* while the most important factor - Reject decision rate decreased by over *10%*.

# 9 Improvements

## 9.1 OvR

One improvement for the OvR ensemble came to my mind. Right now the negative class set is roughly 8 times bigger than the positive class set. That is because the negative class set consists of all the numbers besides the positive one. That way the training set is imbalanced and can behave poorly. The first thing to do, before any further improvements would be to reduce the number of negative class samples. The solution was developed in *reduced_trainOVRensemble.m* and *reduce.m* functions, included in the source code.

The following results are related to *0.7* reduction factor. I am aware that this should be throughly tested and optimized, but for the sake of this task, I run the function with a couple of values and choose the one giving the best results.

Comparison of OvR ensemble results:

Table 3: Error Rates Comparison

| Class | OVR Ensemble | OVR Ensemble with NCR | OVR Ensemble with FE | OVR Ensemble with FE + NCR |
|---|---|---|---|---|
| 0 | 0.01113 | 0.01309 | 0.00631 | 0.00694 |
| 1 | 0.01011 | 0.01201 | 0.00713 | 0.00692 |
| 2 | 0.02736 | 0.03144 | 0.00978 | 0.01164 |
| 3 | 0.03153 | 0.03550 | 0.01575 | 0.01801 |
| 4 | 0.02473 | 0.02725 | 0.01260 | 0.01030 |
| 5 | 0.04678 | 0.05345 | 0.01411 | 0.01045 |
| 6 | 0.01640 | 0.01884 | 0.00763 | 0.00845 |
| 7 | 0.01955 | 0.02182 | 0.01185 | 0.01157 |
| 8 | 0.04986 | 0.05655 | 0.01656 | 0.01844 |
| 9 | 0.04471 | 0.04968 | 0.01858 | 0.01991 |

For some of the classes, the error rates decreased slightly, and for the others there's been a increase in value. Let's take a look on how the OvR ensemble behaves on all the classes in those cases:

Table 4: Error Rates Comparison

| Metric | OVR Ensemble | OVR Ensemble NCR | OVR Ensemble FE | OVR Ensemble FE + NCR |
|---|---|---|---|---|
| Accuracy | 77.5 | 77.35 | 90.3 | 90.9 |
| Error rate | 3.5 | 3.85 | 2.21 | 2.05 |
| Reject coef. | 18.92 | 18.8 | 7.47 | 7.06 |

OvR performs the best with both negative class reduction and feature extraction. This matches the made assumptions and is a predictable outcome.

## 9.2 General improvement

We can try and combine both the OvR and OvR ensembles into one classifier. We can for example try and classify the OvR's rejections by OvO. Theoritically we should see improvement in performance. The functionality was implemented in the *improvedVoting.m* function and utilizes both OvR voting and OvO voting. The results for

The comparasion of ensembles performance can be seen below:

Table 5: Error Rates Comparison

| dataset | Accuracy | Error Rate | Reject Coeff. |
|---|---|---|---|
| Training set | 90.6% | 6.65% | 2.61% |
| Test set | 91.09% | 6.24% | 2.67% |
| Training set FE | 96.03% | 2.95% | 1% |
| Test set FE | 95.51% | 3.27% | 1.22% |

We can note quite the improvement in accuracy for the sets with feature extension. The overall statistics seem better than those for the OvO classifier on dataset with feature extraction, thus we can count it as a win.

# 10 Conclusion

We can take a look and compare best approaches for OvO, OvR and custom improved solution in the table below (metrics for test sets):

Table 6: Error Rates Comparison

| approach | Accuracy | Error Rate | Reject Coeff. |
|---|---|---|---|
| OvO FE | 95.4% | 3.4% | 1.16% |
| OvR FE + NCR | 90.3% | 2.21% | 7.47% |
| improved (OvO + OvR) FE | 95.51% | 3.27% | 1.22% |

From there we can conclude that the custom solution beats the best achieveable solutions for separate OvR and OvO ensembles.

The confusion matrices for the test sets for OvO FE and improved (OvO + OvR) FE are below:

```
cfmx =

   959     0     2     1     0     3     6     1     2     0     6
     0  1110     3     1     4     0     2     1     8     0     6
     5     1   973     9     7     0     8     8     4     1    16
     0     1     4   967     0     9     0     7    10     3     9
     2     5     3     2   932     0     9     1     1    17    10
     3     2     2    12     1   840     4     3     6     4    15
    11     4     1     0     6     4   921     0     5     0     6
     0     8    11     2     1     0     0   969     1    15    21
     3     1     6    14     4     6     1     7   912     2    18
     2     5     2     7    16     3     2     8     4   942    18
```

Figure 7: OvO FE confusion matrix

```
cfmx2 =

   964     0     1     2     0     2     5     1     2     0     3
     0  1117     3     1     1     0     3     0     5     0     5
     7     1   977     4     6     1     6     9    12     0     9
     0     2     4   966     0    10     0    10     6     4     8
     3     4     2     1   932     0     7     1     2    18    12
     5     1     2    13     2   830     5     6     5     4    19
    13     3     1     0     8     3   918     0     3     0     9
     0    11    11     1     1     0     0   969     5    16    14
     6     1     4    14     1     7     1     8   908     2    22
     2     5     3     9    18     3     2    10     2   946     9
```

Figure 8: improved (OvO + OvR) FE confusion matrix