Neural networks: assignment 1

March 12, 2017

Introduction

This reports describes several approaches that were undertaken for classifying images of handwritten digits. The data set consists of 2707 images, which were split in a training set of 1707 images and a test set of 1000 images. Each image is represented by a vector with 256 dimensions, representing a 16x16-sized pixel map. This report summarizes the methods and results for completing 5 particular tasks, which are described in the following sections.

1 Distance analysis of center clouds

Each image in the training data has one of 10 class labels d, where d $= 0, 1, ..., 9$. Table 1 summarizes the number of images representing a specific class.

Class	**Number of images**	**Radius** C_d
0	319	15.89
1	252	9.48
2	202	14.17
3	131	14.74
4	122	14.53
5	88	14.45
6	151	14.03
7	166	14.91
8	144	13.71
9	132	16.14

Table 1: Number of images representing a specific digit and the radii of the center clouds for the corresponding digit.

In this first task, we created 10 clouds of points in 256 dimensions, C_d, for d $= 0, 1, ..., 9$. For each cloud C_d, we used all training images representing the corresponding class d for calculating the mean value in each dimension. The clouds C_d thus represent the center of each digit, and can be used as simple classifiers for classification of new images: calculate

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	**0.0**	14.45	9.33	9.14	10.77	7.52	8.15	11.86	9.91	11.49
1	14.45	**0.0**	10.13	11.73	10.17	11.12	10.61	10.74	10.09	9.93
2	9.33	10.13	**0.0**	8.18	7.93	7.91	7.33	8.87	7.08	8.89
3	9.14	11.73	8.18	**0.0**	9.09	6.12	9.3	8.92	7.02	8.35
4	10.77	10.17	7.93	9.09	**0.0**	8.0	8.78	7.58	7.38	6.01
5	7.52	11.12	7.91	6.12	8.0	**0.0**	6.7	9.21	6.97	8.26
6	8.15	10.61	7.33	9.3	8.78	6.7	**0.0**	10.89	8.59	10.44
7	11.86	10.74	8.87	8.92	7.58	9.21	10.89	**0.0**	8.47	5.43
8	9.91	10.09	7.08	7.02	7.38	6.97	8.59	8.47	**0.0**	6.4
9	11.49	9.93	8.89	8.35	6.01	8.26	10.44	5.43	6.4	**0.0**

Table 2: Pairwise distances between center clouds.

the distance from the image to each center C_d, and select as a label the class of the closest center cloud. The results for this approach are discussed in section 2.

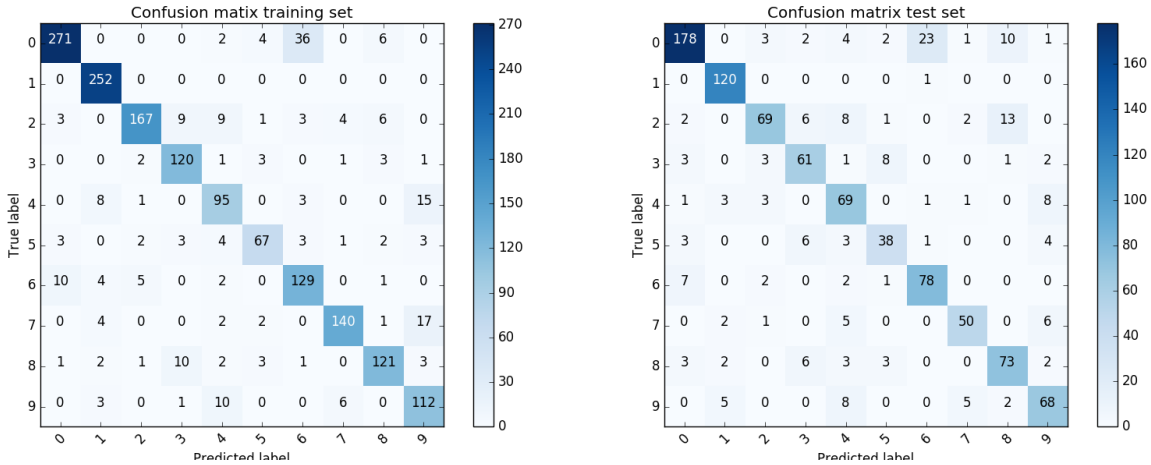
By comparing the cloud centers with each other we can see beforehand which digits are the easiest and the hardest to compare. These distances can be found in table 2. Images representing digits 0 and 1 have the highest distance and are most unlikely to be confused with each other. Other remarkably high distance values occur for instance between 0 and 9, 0 and 7 and between 6 and 7. Pairs of distance that are hardest to separate are 7 and 9, 4 and 9 and 3 and 5.

2 Distance-based image classification

Here we present the results of the distance-based classification of images as described in the previous section. The clouds of image centers were calculated from the training set and the classifier was applied to both the training and test set. The confusion matrices for both sets can be found in figure 1.

Table 3 shows for each digit the percentage of correctly predicted classes in both the training and test set. Additionally, the last column shows the average distance to the center of all other digits (by averaging the pairwise distances in table 2). From these results we see that digit *5* has both the lowest average pairwise distance and achieves the lowest accuracy on both sets. On the other extreme, digit *1* achieves the highest accuracies and has the highest average pairwise distance.

The Euclidean distance measure was used in the above mentioned results. We also used the cosine distance as a measure. When using the cosine distance, the confusion matrices look as in figure 2.Table 4 shows for each digit the percentage of correctly predicted classes in both the training and test set. The accuracies are very similar to those of the Euclidean distance-based classifier.



(a) Training set

(b) Test set

Figure 1: Confusion matrix plots for the Euclidean distance-based classifier.

Class	Acc. train (%)	Acc. test (%)	Avg. dist.
0	85	79	9.26
1	100	99	9.9
2	83	68	7.56
3	92	77	7.78
4	78	80	7.57
5	76	69	7.18
6	85	87	8.08
7	84	78	8.2
8	84	79	7.19
9	85	77	7.52

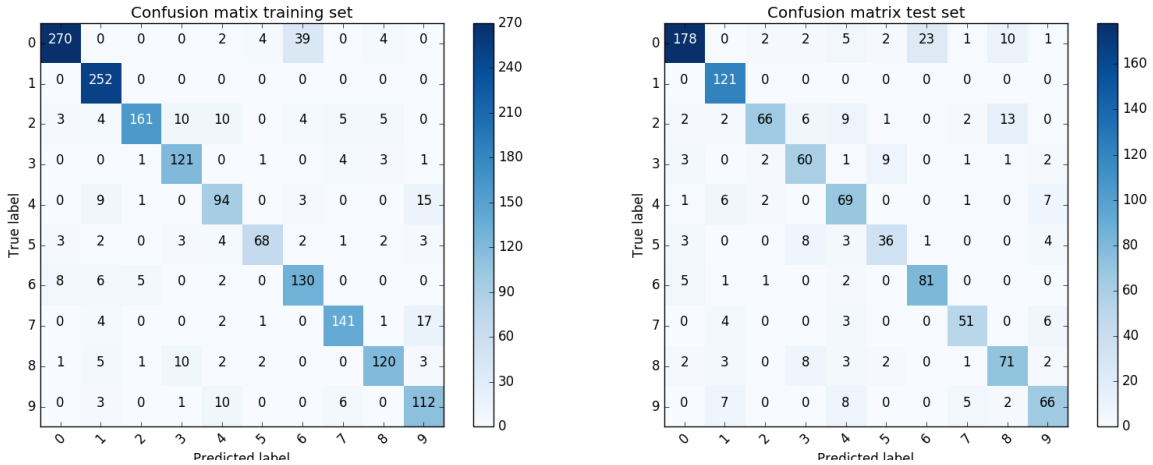
Table 3: Percentages of correctly classified images in the training and test sets. The last column shows the average pairwise distance for each digit, computed from table 1.

3 Bayes rule classifier

For this task we chose the Euclidean distance to each digit-cloud center as the feature to disceriminate. Here we chose the digits 0 and 6 as the two classes because in task 2 these two digits showed a high possiblity to be misclassfied as the other.

As there are only two classes ($C1$ as digit 0 and $C2$ as digit 6), we use function $y(x) = y_1(x) - y_2(x)$ as the discriminant. $y_1(x)$ represents the Euclidean distance between the being classified point to the center of cloud 0, $y_2(x)$ is to the cloud 6. Which means if it's closer to the center of digit 0 the result of the discriminat function would be negative, then possitive stands for it closer to center of digit 6.

The histogram figure 3(a) shows the distribution of the handwriten digits 0 and 6 ac-



(a) Training set (b) Test set

Figure 2: Confusion matrix plots for the cosine distance-based classifier.

Class	Acc. train (%)	Acc. test (%)
0	85	79
1	100	100
2	80	65
3	92	76
4	77	80
5	77	65
6	86	90
7	85	80
8	83	77
9	85	75

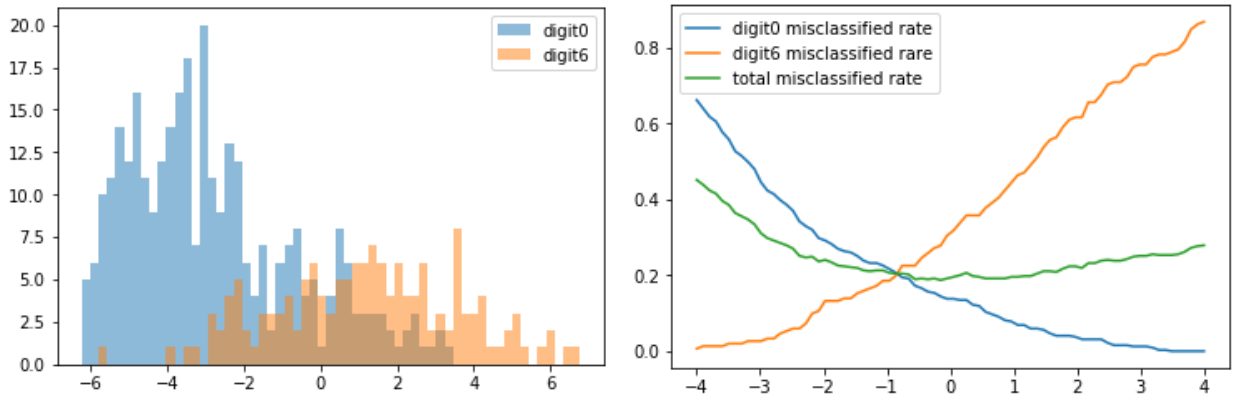
Table 4: Percentages of correctly classified images in the training and test sets when using the cosine distance measure.

cording to the discriminant function $y(x)$.

In our case $P(C1) = \frac{319}{470} = 0.679$, $P(C2) = \frac{151}{470} = 0.321$. The distribution of misclassfied rate according to the choice of discriminant has been shown in figure 3(b). When $y_{opt} = -0.152$ the function researched the optimal. We chose the optimal y_{opt} as the Bayes clssifier. Which means if the $y(x) < y_{opt}$ (case $X1$) we classfy it as the digit 0, otherwise it's case $X2$ and classfied as digit 6. With the optimal classifier we reached $P(X1|C1) = 0.856$, $P(X2|C2) = 0.722$, $P(X1) = 0.670$, $P(X2) = 0.330$. Use the Bayes Theorem:

$$P(C_k|X) = \frac{P(X|C_k)P(C_k)}{P(X)}$$

We could calculate the accuracy of our Bayes classfier in training dataset is digit 0: $P(C1|X) = 0.867$ and digit 6: $P(C2|X) = 0.702$. And overall accuracy 0.813.



(a) histogram

(b) error rate on trainning data

Figure 3: task 3

For the testing dataset we reached the accuracy digit 0: $P(C1|X) = 0.973$, digit 6: $P(C2|X) = 0.589$ and overall accuracy is 0.863.[1]

4 Multi-class linear perceptron

For this task we implemented a multi-class linear perceptron algorithm, as described at slide 44 of the lecture on linear models. The perceptron takes image vectors of length 256 plus a bias term as the input, which are connected to 10 output nodes, corresponding to all 10 image labels. An image is classified as digit d if the corresponding d^{th} output node of the perceptron is activated most (i.e. outputs the highest value).

Initialization The bias term we added to each image vector was exactly 1. All weights were initialized randomly, with numbers drawn from a uniform distribution between 0 and 1.

Updating the weights If an input vector x from the training set with corresponding output label c was misclassified, the weights of the perceptron's output node c were updated as follows:

$$w = w + \eta x$$

The weights of the output node that (incorrectly) gave the highest activation value were updated in the opposite way:

$$w = w - \eta x$$

[1]Loss Matrix was not being considered for all the cases. In other words, we treated the risk of misclassifying both digits the same.

This training proces was repeated for 60 iterations, which proved to be more than sufficient to obtain 100% accuracy on the training set (see figure 3). Furthermore, we used an adaptive learning rate that decreased with each of these 60 iterations as: $\eta = 9^{\frac{-i}{20}+1}$, where i is the iteration.

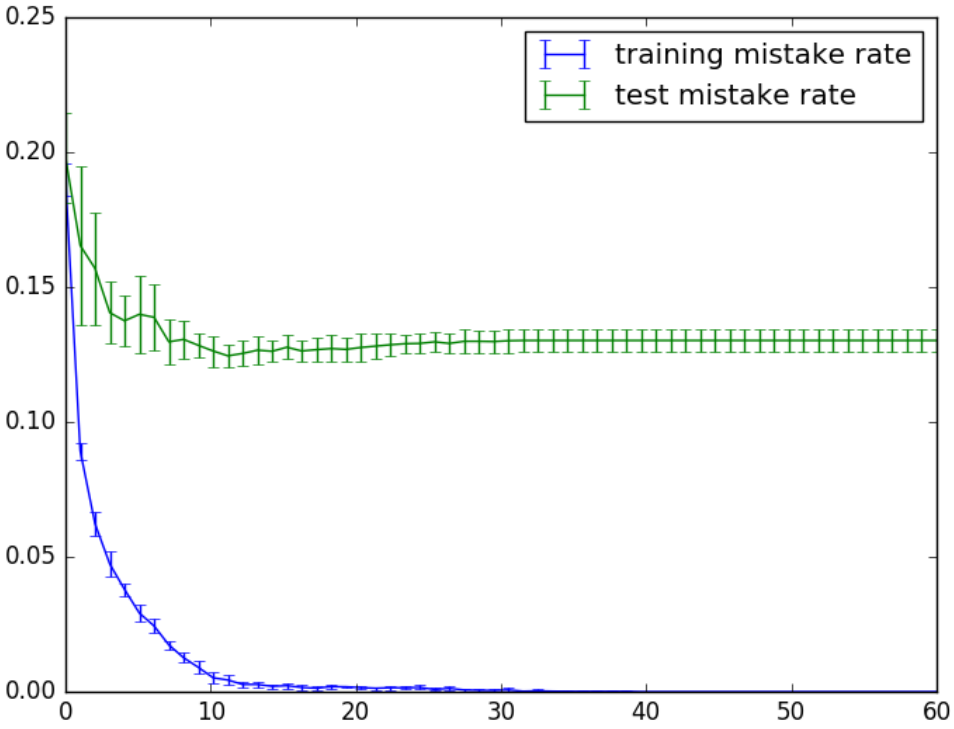


Figure 4: Average mistake rates for 10 different runs of the multi-class perceptron on both the training and test set.

Averaging and results Training of the perceptron was done for 10 different runs, each with a different (random) initialization of the weights. Figure 3 shows the average mistake rates for these runs. After full training of the perceptron (60 iterations) the average error on the test set was 13 % with a standard deviation of 0.0042.

5 Neural network with one hidden layer

For this fifth and final task a more sophisticated algorithm was used for classifying the images. A neural network with one hidden layer was used, largely based on software developed by Michael Nielsen (available from https://github.com/mnielsen/neural-networks-and-deep-learning.git).

References

[1] A. C. Melissinos and J. Napolitano, *Experiments in Modern Physics*, (Academic Press, New York, 2003).

[2] N. Cyr, M. Têtu, and M. Breton, IEEE Trans. Instrum. Meas. **42**, 640 (1993).

[3] *Expected value*, available at `http://en.wikipedia.org/wiki/Expected_value`.