Bonn-Rhein-Sieg University of Applied Sciences

Scientific Experimentation and Evaluation - Assignment 09 - Handwritten Data Recognition

Mazin Eltayeb, Bastian Lang

June 20, 2016

Contents

1 Abstract 2

2 The Data 2

3 Approach 2

4 Results 3

5 Comparison 5

6 Conclusion 5

1 Abstract

This report describes how we applied a machine learning technique to the task of recognizing handwritten digits. It presents our approach, analyses the results and compares them to the results of others that have applied some ML techniques to this task as well.

2 The Data

The MNIST dataset is a collection of 70000 images representing handwritten digits ranging from 0 to 9. The dataset is divided into a training and a test set. The training set consists of 60000 single images and the test set of 10000 images. Each image is stored as a 784 row vector, where each vector represents 28x28 pixels. An example can be seen in figure 2.1.

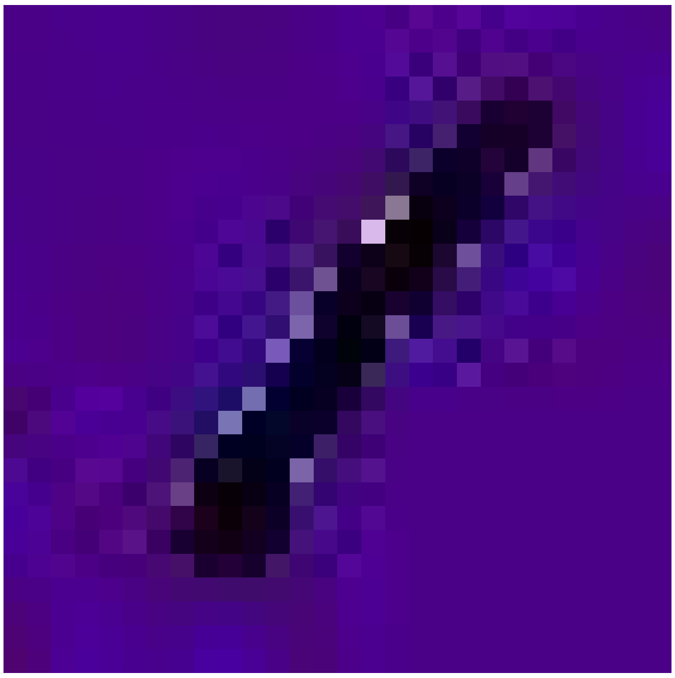

Figure 2.1: Sample Image from MNIST dataset

3 Approach

To classify the handwritten data we used a multilayer perceptron and trained it using the backpropagation of error algorithm. We used the implementation of David Stutz (davidstutz.de) and adjusted the parameters to our needs. The MLP uses 784 inputs, 700 hidden neurons in its hidden layer and ten outputs - one for each digit. As the activation function we used a locistic sigmoid:

$$y = \frac{1}{1 + exp(-x)}$$

We set the learning rate to 0.1. The training was done for 500 epochs, where in every epoch we presented 100 randomly chosen samples from the training set and computed the error over the whole batch to adjust the weights.
We trained MLP with this settings 50 times and computed its mean and best performance. The results can be seen in section 4.

4 Results

On average the MLP was able to classify 91.91% correctly. In the best run the performance was about 92.5%. Figure 4.1 visualizes the average performance of the network. The mean error progression while training can be seen in figure 4.2. The gradient of the error curve in the figure suggests that continuing the training could possibly lead to an even smaller error.

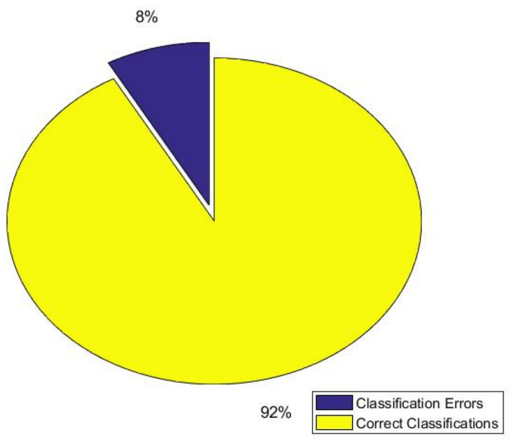

Figure 4.1: Pie Chart of the Quality of the Classification

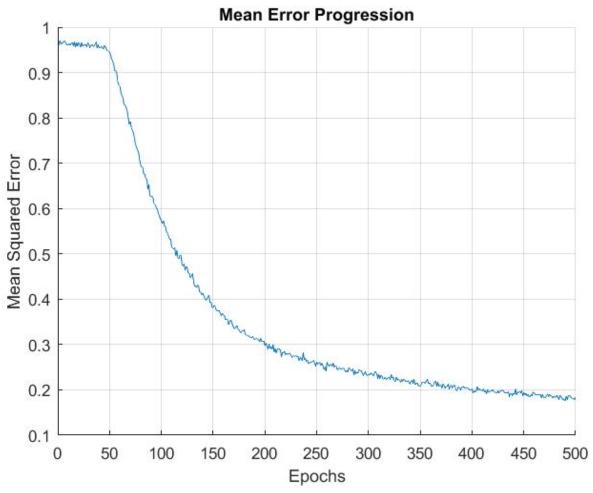

Figure 4.2: Error Progression While Training

In table 4.1 and figure 4.3 the confusion matrix of one of the nets can be seen. About 63 times an 8 was mistaken for a 10. About 40 times a 6 for a 9, a 10 for a 5 or a 6 for a 4.

Table 4.1: Confusion Matrix

classified\real	0	1	2	3	4	5	6	7	8	9
0	945	0	4	1	1	8	10	3	4	6
1	0	1102	0	0	3	2	3	18	5	7
2	2	2	952	21	5	2	4	29	4	0
3	2	6	9	923	0	37	0	4	12	8
4	0	1	13	1	916	9	11	14	11	44
5	13	2	3	21	0	761	11	1	14	15
6	12	5	12	4	11	18	914	0	11	1
7	1	1	7	4	1	6	0	894	2	3
8	5	16	28	24	9	41	5	2	900	15
9	0	0	4	11	36	8	0	63	11	910

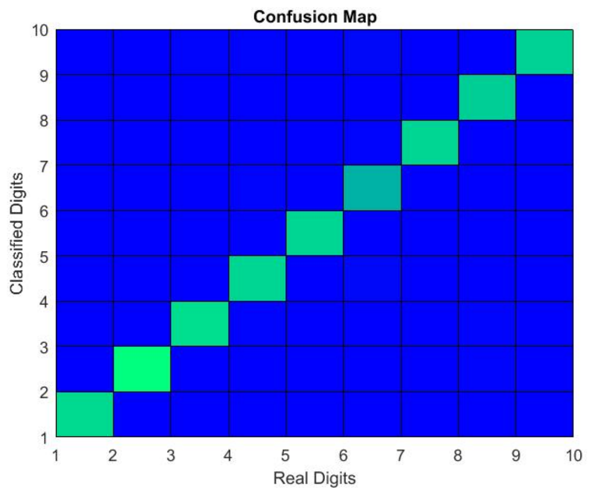

Figure 4.3: Graphical Confusion Matrix of Classifications

5 Comparison

We compared the accuracies of the MLP approach with two other approaches. One approach was to use PCA to reduce the dimensionality of the problem and then use an SVM to classify the images. The other approach also used an SVM for classification, but it used a method named *Sparse Coding* for preprocessing.
The results can be seen in table 5.1. Using PCA in combination with SVM performed best and significantly outperformed the other methods. The MLP achieved a slighty better performance than the sparse coding approach.

Table 5.1: Accuracies of Different Approaches

Method	MLP	Sparse Coding + SVM	PCA + SVM
Average Accuracy	91.91%	89.92%	98%

6 Conclusion

Using PCA in combination with SVM seems to be the best method to use here. On the other hand some more information are missing to be sure about it. No runtime or memory usage analysis could be done due to lack of information.
For the PCA/SVM approach no information about the number of performed experiments were available, so it may be possible that this result is an outlier. For the Sparse Coding approach the average accuracy is based on three runs only, which is also not enough to make a statistically sound statement.