Machine Learning

Digit Classification with Kernel Perceptron

Lukas Hermans

Università degli Studi di Milano
lukas.hermans@studenti.unimi.it

June 10, 2021

Abstract

The MNIST database of handwritten digits is a common benchmark for the performance of machine learning algorithms for image recognition. In the present work, the multiclass kernel perceptron alorithm - an expansion of Rosenblatt's famous perceptron algorithm from 1957 - is trained for the classification of handwritten digits in the MNIST database. The database is split in 60000 training images and 10000 test images. Three different realizations of the multiclass kernel perceptron algorithm lead to test error rates of 1.58 %, 1.55 %, and 1.54 %, respectively. A comparison with the literature reveals that the results are similar to the test error rates for other classical machine learning algorithms, especially the closely related support vector machine. The application of additonal data preprocessing techniques prior to the training phase as well as deep neural networks are reported to lead to significantly lower test error rates.

Contents

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work.
I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying.
This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

Milan, June 10, 2021

1 Introduction

Recently, the application of computer vision techniques has gained increasing importance in a broad variety of fields, e.g. for the development of autonomeous vehicles, for facial recognition, and for the detection of illnesses [1, 2, 3]. This process is especially driven by the growing field of machine learning.

A common dataset to benchmark and compare machine learning algorithms for image recognition as a subfield of computer vision is the MNIST database of handwritten digits. The dataset contains 60000 training examples and 10000 test examples of handwritten digits from 0 to 9. Originally, the dataset was published by Y. LeCun et al [4]. In this paper, in order to simplify the work with the images, a revision of the original dataset is adopted [5]. In the following, the revised dataset is simply refered to as MNIST dataset.

The objective of the present work is to train a multiclass kernel perceptron algorithm for the classification of handwritten digits using the training examples in the MNIST dataset. The performance measured by the test error rate is evaluated by using the test set.

In Section 2, the MNIST dataset is presented in more detail, and some basic definitions and notations are clarified. Then, in Section 3, the theoretical foundation of the multiclass kernel perceptron algorithm is summarized. There, three different approaches to retrieve a multiclass predictor from the multiclass kernel perceptron algorithm are distinguished. Section 4 describes the details of the implementation of the multiclass kernel perceptron algorithm in the present work and gives a list of the applied software. The results of the training of predictors using the multiclass kernel perceptron algorithm on the training part of the MNIST dataset are presented in Section 5. In doing so, the hyperparameters of the multiclass kernel perceptron algorithm are optimized, in order to obtain three multiclass predictors with the lowest error rates on the test part of the MNIST dataset. In Section 6, the main advantages as well as the drawbacks of the application of the multiclass kernel perceptron algorithm for the application on the MNIST dataset are discussed. In addition, the retrieved predictors are compared with the test error rates of different learning algorithms that are applied to the MNIST dataset in the literature. Finally, Section 7 summarizes the present work and gives an overview of current and future objects of research regarding the recognition of handwritten digits.

2 Dataset & Definitions

The MNIST dataset contains a total of 70000 images of handwritten digits labeled with a number from 0 to 9. A predifined division of the dataset distinguishes 60000 training examples and 10000 test examples. Each image contains $28 \times 28 = 784$ pixels, where each pixel encodes the brightness in 8 bit. The brightness of each pixel can be described by a variable $x_i \in \{0, 1, \ldots, 255\}$ for $i \in \{1, 2, \ldots, 784\}$. Hence, each image is completely defined by a vector of *features*

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{784} \end{pmatrix}. \tag{1}$$

The space of all possible images $\mathcal{X} = \{0, 1, \ldots, 255\}^{784}$ is called *feature space*. Each image $\vec{x}$ in the MNIST dataset has a label $y \in \{0, 1, \ldots, 9\}$ that represents the handwritten digit. The *label space* is thus given by $\mathcal{Y} = \{0, 1, \ldots, 9\}$. The labels were added by humans. In the present work, they are regarded as the *true labels* in contrast to labels $\hat{y}$ predicted by an algorithm. So, every *example* in the dataset has the form $(\vec{x}, y)$.

In Figure 1, an examplary ensemble of ten different handwritten digits from 0 to 9 in the dataset is shown. The decompistion of the 70000 examples into training and test examples can be described by two sets S (*training set*) and D (*test set*). The training set S contains all training examples, while the test set D contains all test examples. There is no intersection between the sets S and D, such that the test set contains images that are different from those in the training set. This allows the evaluation of the performance of a classification algorithm that was trained on the training set S.

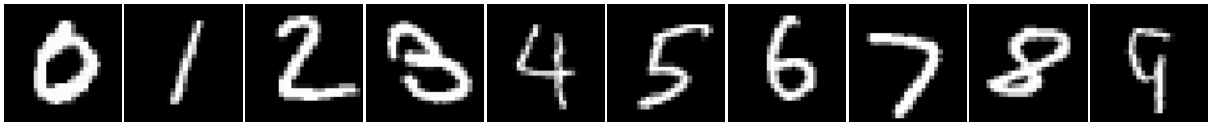
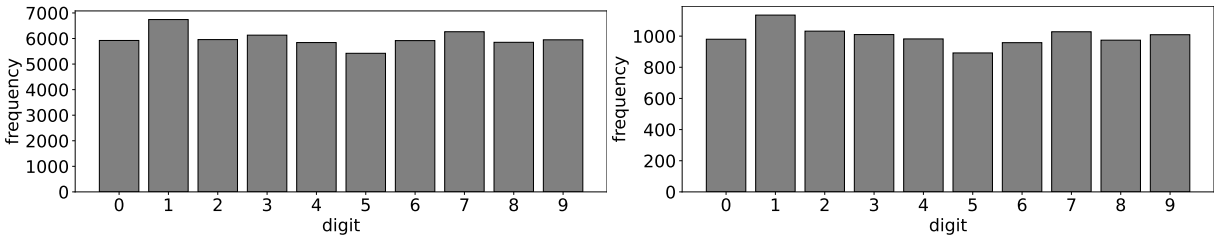


Figure 1: Examplary ensemble of ten different handwritten digits from 0 to 9 in the MNIST dataset. Each image can be described by a feature vector $\vec{x}$, and has a human-decided label y. The images consist of $28 \times 28 = 784$ pixels, where each pixel encodes an integer brightness level between 0 and 255.

Figure 2 shows histograms of the frequency, with which each digit occurs in the training and test set. Both the training and test set are balanced because the different digits appear with a similar frequency.



(a) training examples (b) test examples

Figure 2: The histograms show, how often each digit occurs in (a) the training set S and (b) the test set D. Both sets are balanced as all of the ten digits appear with a similar frequency.

3 Theory

With the definitions from Section 2 in mind, the problem in the present paper consists in finding a multiclass predictor $f_S : \mathcal{X} \to \mathcal{Y}, \vec{x} \mapsto \hat{y}$ that predicts a label $\hat{y} \in \mathcal{Y}$ for every possible image $\vec{x} \in \mathcal{X}$ using the multiclass kernel perceptron algorithm. The subscript S of f_S indicates that the classifier is trained on the training set S.

3.1 From Multiclass to Binary: One-vs-All Encoding

The basis of the multiclass kernel perceptron algorithm is the reduction of this multiclass classification problem to several binary classification problems - one for each of the ten digits - using the so-called *one-vs-all encoding*. Applying one-vs-all encoding, the initial label $y \in \{0, 1, \ldots, 9\}$ of each example in the training set S is transformed into a binary label $z \in \{-1, 1\}$ by fixing a digit $a \in \{0, 1, \ldots, 9\}$. z is only 1 for those examples for which $y = a$, otherwise z is set to -1. In doing so, a new binary training set $S^{(a)}$ with examples of the form $(\vec{x}, z)$ is generated. As can be seen immediately, the transformation only regards the labels, but leaves the images $\vec{x}$ themselves unchanged [6].

3.2 Binary Kernel Perceptron Algorithm

Now, for each of the ten binary training sets $S^{(a)}$ obtained via one-vs-all encoding, the goal is to train a binary classifier that predicts $\hat{z} = 1$ when a given image $\vec{x}$ shows the digit a and $\hat{z} = -1$ when it shows another digit. For this purpose, the *binary kernel perceptron algorithm* - that was first presented in 1984 by Aizerman et al. - can be applied [7], see Algorithm 1.
The binary kernel perceptron algorithm is an online learning algorithm as the training examples are processed sequentially.
In the version of the binary kernel perceptron algorithm that is applied in the present work, the training examples are repeatedly processed for N_{epoch} epochs. Each epoch is a loop over a random permutation of the training examples in the training set $S^{(a)}$.

The binary kernel perceptron trains a predictor of the form $h_{S^{(a)}} : \mathcal{X} \to \mathcal{Z}, \vec{x} \mapsto \hat{z}$, where $\mathcal{Z} = \{-1, 1\}$ is

the binary label space. The predictor $h_{S^{(a)}}$ has the following form:

$$h_{S^{(a)}} = \text{sgn}\left(\sum_{s:\alpha_s \neq 0} \alpha_s z_s K_p(\vec{x_s}, \vec{x})\right). \tag{2}$$

The part inside the sgn-function will be called $g_{S^{(a)}}$. In Eq. 2, K_p is a polynomial kernel of degree p that has the functional form

$$K_p(\vec{x_i}, \vec{x_j}) = (1 + \vec{x_i} \cdot \vec{x_j})^p,$$

for all $\vec{x_i}, \vec{x_j} \in \mathcal{X}$. Other kernels are of course possible, but the present work focuses on polynomial kernels. The binary classifier in Eq. 2 corresponds to a decision surface of degree p in the feature space $\mathcal{X}$. For $p = 1$, the surface is a hyperplane that is adjusted using the binary training set $S^{(a)}$ in order to make predictions. This corresponds to the original perceptron algorithm first introduced by Rosenblatt in 1957 [8]. Note, that the predictor $h_{S^{(a)}}$ depends only on a vector $\vec{\alpha}$ that - as can be seen in the algorithm panel below - is updated by the binary kernel perceptron algorithm only if a training example is missclassified. Thus, $\vec{\alpha}$ counts the number of missclassifications during the training process for all examples in the training set $S^{(a)}$. The dependence of $h_{S^{(a)}}$ on the specific $\vec{\alpha}$ is not explicitly denoted in order to maintain readability. However, from the context the particular vector $\vec{\alpha}$ that specifies the predictor should be obvious.

The binary kernel perceptron algorithm stores the $\vec{\alpha}$ for all iterations of the algorithm in a set A and outputs these (technically, A is a multiset as it can and most likely will contain the same $\vec{\alpha}$ more than once, when the prediction for an iteration is correct). As every $\vec{\alpha}$ defines a binary classifier $h_{S^{(a)}}$, the result of the algorithm is a large set A of binary predictors. In the present paper, both the function $h_{S^{(a)}}$ and the corresponding vector $\vec{\alpha}$ are called binary predictor as both contain the same information.

Algorithm 1: Binary Kernel Perceptron

Input : N_{epoch}, p, $S^{(a)}$
let $A = \emptyset$, $\vec{\alpha} = \vec{0}$;
for *all* $1, \ldots, N_{epoch}$ **do**
  shuffle training examples in $S^{(a)}$ to obtain a new set $T^{(a)}$;
  for *all* $t = 1, \ldots, |T^{(a)}|$ **do**
    compute $\hat{z} = \text{sgn}\left(\sum_{s:\alpha_s \neq 0} \alpha_s z_s K_p(\vec{x_s}, \vec{x_t})\right)$;
    if $\hat{z} \neq z_t$ **then**
      $\alpha_i \leftarrow \alpha_i + 1$ where i is the index of example $(\vec{x_t}, z_t)$ in $S^{(a)}$;
    end
    store $\vec{\alpha}$ in A;
  end
end
Output: A

3.3 Choice of Binary Predictors

The remaining problem regards the choice of a predictor from the set A of $\vec{\alpha}$-vectors. A straightforward approach is to simply take the last predictor $\vec{\alpha}_{fin}$ after $N_{epoch} \times |S^{(a)}|$ iterations as in the original perceptron algorithm by Rosenblatt. In the present work, also two somewhat more sophisticated approaches are considered.

First, another possible choice is to compute the average $\langle\vec{\alpha}\rangle$ over all predictors in A. This leads to the following vector:

$$\langle\vec{\alpha}\rangle = \frac{1}{N_{epoch} \cdot |S^{(a)}|} \sum_{\vec{\alpha} \in A} \vec{\alpha}.$$

The second approach makes use of the binary training error $\ell_{S^{(a)}}$ of a particular predictor $\vec{\alpha}$ that is defined

as follows:

$$\hat{\ell}_{S^{(a)}}(\vec{\alpha}) = \frac{1}{|S^{(a)}|} \sum_{(\vec{x},z)\in S^{(a)}} \ell(h_{S^{(a)}}(\vec{x}), z).$$

Here, ℓ is the binary *zero-one loss*

$$\ell(\hat{z}, z) = \mathbb{1}(\hat{z} = z),$$

where $\mathbb{1}$ is the indicator function. The binary training error is the error rate of a particular predictor $\vec{\alpha}$ on the training set $S^{(a)}$. Now, the second predictor is extracted by computing the $\vec{\alpha}_{min}$ in A, for which the training error $\ell_{S^{(a)}}$ is minimized:

$$\vec{\alpha}_{min} = \underset{\vec{\alpha}\in A}{\operatorname{argmin}}\, \hat{\ell}_{S^{(a)}}(\vec{\alpha}).$$

In the present work, the predictors are refered to as final predictor $\vec{\alpha}_{fin}$, average predictor $\langle\vec{\alpha}\rangle$, and minimizing predictor $\vec{\alpha}_{min}$. Of course, they depend on the number of epochs N_{epoch} from which they are retrieved.

3.4 Multiclass Kernel Perceptron Algorithm

The binary kernel perceptron algorithm just presented can be applied for all of the ten digits $a \in \{0, 1, \ldots, 9\}$, using the corresponding binary training sets $S^{(a)}$. If a specific $\vec{\alpha}$-type is chosen, the result are ten binary predictors $h_{S^{(a)}}$ that predict $\hat{z} = 1$ when a given image $\vec{x}$ shows the handwritten digit a and $\hat{z} = -1$ when not.

Finally, the multiclass classifier f_S that predicts a digit from a given image $\vec{x}$ is the combination of all of the ten binary classifiers:

$$f_S(\vec{x}) = \underset{a\in\{0,1,\ldots,9\}}{\operatorname{argmax}}\, g_{S^{(a)}}(\vec{x}).$$

Note, that the functions $g_{S^{(a)}}$ inside the sgn-function in Eq. 2 enter the computation of f_S, and not the binary predictors $h_{S^{(a)}}$ themselves. f_S predicts the digit of the binary classifier that is most secure that the image $\vec{x}$ contains a certain digit by chosing the largest $g_{S^{(a)}} \in \mathbb{R}$. The algorithm is called *multiclass kernel perceptron algorithm* as it makes use of the kernelized perceptron algorithm and expands it to predictions beyond the binary case [6]. Depending on the choice of the used binary predictors $\vec{\alpha}$, three different realizations of the multiclass kernel perceptron algorithm are obtained. These are implemented in the present work.

3.5 Evaluation: Training and Test Error Rate

To evaluate the performance of a multiclass predictor f_S, two error rates have to be distinguished.
One the one hand, there is the *training error rate* that is given by:

$$\hat{\ell}_S(f_S) = \frac{1}{|S|} \sum_{(\vec{x},y)\in S} \ell(f_S(\vec{x}), y). \tag{3}$$

Here, ℓ is the binary zero-one loss

$$\ell(f_S(\vec{x})) = \mathbb{1}(\hat{y} = y).$$

The training error rate is the relative number of images that the multiclass predictor f_S (that of course depends on the binary predictors choosen previously) missclassifies among all the images $\vec{x}$ in the training set S. It is the multiclass counterpart of the binary training error $\hat{\ell}_{S^{(a)}}$.
On the other hand, the *test error rate*

$$\hat{\ell}_D(f_S) = \frac{1}{|D|} \sum_{(\vec{x},y)\in D} \ell(f_S(\vec{x}), y) \tag{4}$$

is the relative number of missclassified images $\vec{x}$ in the test set D. Note, that these images do not enter the training of f_S. Similarly to the binary training error $\hat{\ell}_{S^{(a)}}$, there exists also a binary test error $\hat{\ell}_{D^{(a)}}$.

4 Implementation & Software

For the present work, the multiclass kernel perceptron algorithm is implemented in the programming language Python (version 3.8.6). On top of vanilla Python, the following software is used:

- pip (version 21.1.2): managing of Python modules
- matplotlib (version 3.4.2) & seaborn (version 0.11.1): modules for visualization & plots
- pandas (version 1.2.4): module for loading data
- numpy (version 1.20.3): module for array computing (such as dot products)

All of the Python code used in this paper can be found in the following GitHub repository: `https://github.com/lukher98/digit-classification`. To ensure the reproducability of the numerical results, seeds for the random number generator of the numpy module are set whenever random numbers enter the computation. In contrast to Algorithm 1, the predictors of each iteration of the binary kernel perceptron algorithm are not collected but the average and minimizing predictors are computed on the fly, in order to obtain higher efficiency and less memory usage. In addition to the Python code, the repository contains the whole MNIST dataset (using Git Large File Storage), as well as the LaTeX code of this paper.
The Python code was executed on a Linux machine, but it should also work on a Mac machine. For Windows, the user might have to change some paths inside the code.

5 Results

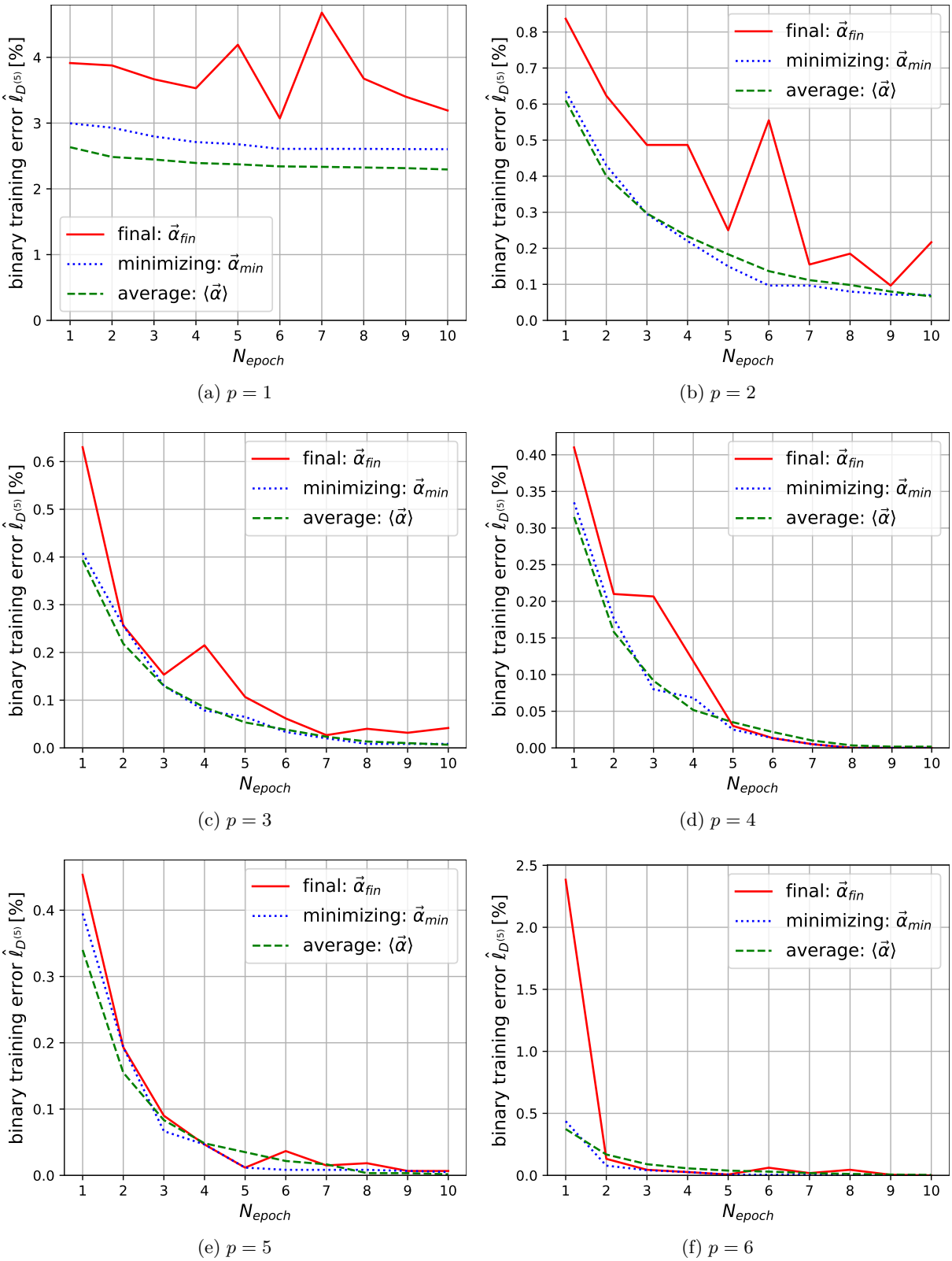
In this Section, the multiclass kernel perceptron algorithm presented in the theory part above is trained on the training set S of the MNIST dataset for several settings. The multiclass predictors are trained using the three different realizations of the multiclass kernel perceptron algorithm. The performance of the trained multiclass predictors is evaluated on the test set D. The test error rates of the three realizations of the multiclass kernel perceptron algortihm are compared for different choices of the degree of the polynomial kernel p. In the present work, degrees from $p = 1$ up to $p = 6$ are applied. Each predictor is trained for a total of $N_{epoch} = 10$ cycles over random permutations of the training examples in the training set S. The final result are three multiclass predictors - one for each realization of the multiclass kernel perceptron algorithm - that minimize the test error rate on the test set D.

5.1 Binary Predictors

Before presenting the obtained multiclass predictors, it is instructive to take a look at what happens at the heart of the multiclass kernel perceptron algorithm when performing a training on the training set S. As explained in the theory part, the multiclass kernel perceptron algorithm is a simple combination of ten binary predictors - one for each digit a. Here, exemplarily, the binary predictor $h_{S^{(5)}}$ for the digit $a = 5$ is considered. Nonetheless, the key statements in this Subsection are valid for all of the ten digits.

Figure 3 shows the binary training error rates $\hat{\ell}_{S^{(5)}}$ as a function of the number of epochs N_{epoch} for the six different degrees p of the polynomial kernel. In each panel, the binary training error rates $\hat{\ell}_{S^{(5)}}$ for the final $\vec{\alpha}_{fin}$, the minimizing $\vec{\alpha}_{min}$, and the average predictor $\langle\vec{\alpha}\rangle$ are displayed. The test error rates $\hat{\ell}_{D^{(5)}}$ for the same binary predictors are reported in Fig. 4.

For the binary predictors with a linear kernel ($p = 1$), the binary training error rate $\hat{\ell}_{S^{(5)}}$ remains constant roughly between 2 % and 4 % with respect to the number of epochs N_{epoch}. Especially, the binary kernel perceptron algorithm does not terminate, i.e. it does not reach a training error rate of 0 %. It is well-known that the binary perceptron algorithm presented by Rosenblatt is only guaranteed to terminate on linearly separable training sets. Given the complexity of the MNIST dataset and the obtained training error rates $\hat{\ell}_{S^{(5)}}$, it is concluded that the binary training set $S^{(5)}$ is not linearly separable. For higher degrees p, a convergence of the training error rate $\hat{\ell}_{S^{(5)}}$ to lower values, and even to 0 % is observed. This is especially

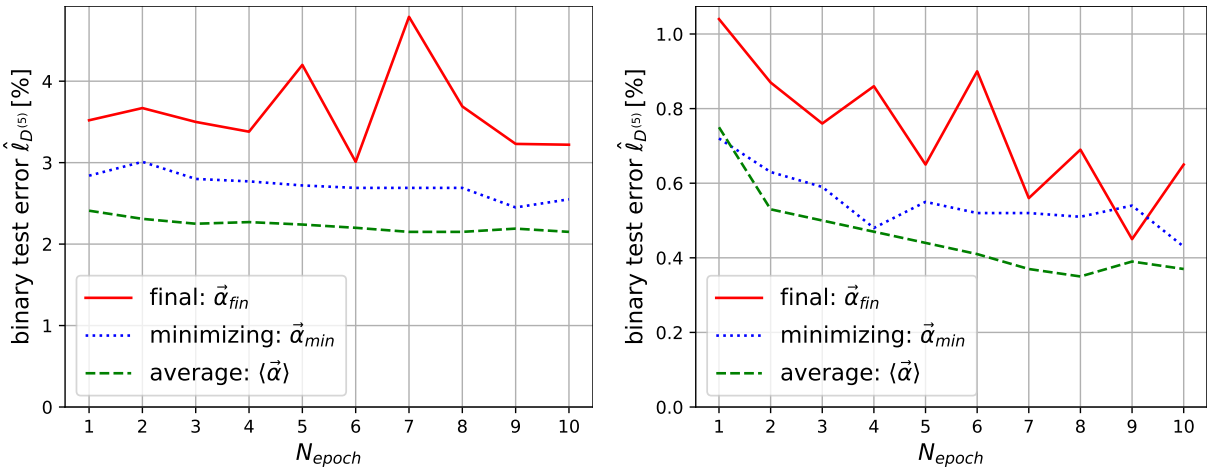


(a) $p = 1$ (b) $p = 2$

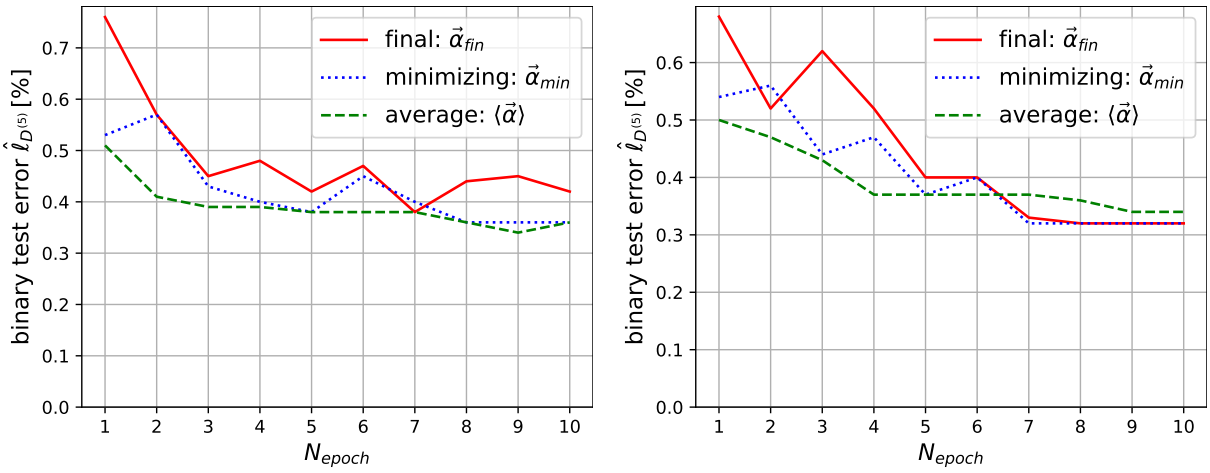
(c) $p = 3$ (d) $p = 4$

(e) $p = 5$ (f) $p = 6$

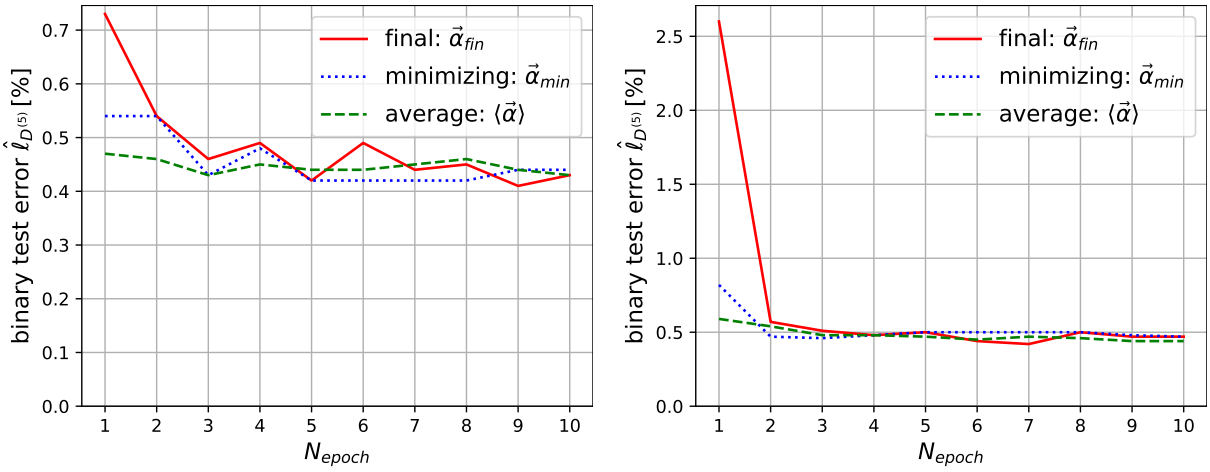
Figure 3: Binary training error rates $\hat{\ell}_{S^{(5)}}$ as a function of the number of epochs N_{epoch} for the training of binary predictors that classify, whether an image shows the digit $a = 5$ or not. Each panel displays the binary training error rates $\hat{\ell}_{S^{(5)}}$ for a different degree p of the polynomial kernel for the three approaches to retrieve binary predictors from the binary kernel perceptron algorithm. Note, the different scales of the vertical axes. The binary test error $\hat{\ell}_{D^{(5)}}$ rates for the same binary predictors are displayed in Fig 4.



(a) $p = 1$ (b) $p = 2$



(c) $p = 3$ (d) $p = 4$



(e) $p = 5$ (f) $p = 6$

Figure 4: Binary test error rates $\hat{\ell}_{D^{(5)}}$ as a function of the number of epochs N_{epoch} for the training of binary predictors that classify, whether an image shows the digit $a = 5$ or not. Each panel displays the binary test error rates $\hat{\ell}_{D^{(5)}}$ for a different degree p of the polynomial kernel for the three approaches to retrieve binary predictors from the binary kernel perceptron algorithm. Note, the different scales of the vertical axes. The binary training error rates $\hat{\ell}_{S^{(5)}}$ for the same binary predictors are displayed in Fig 3.

true for kernels with a degree of $p = 3$ or more. The number of epochs N_{epoch} required for the convergence of the training error rate $\hat{\ell}_{S^{(5)}}$ decreases with the degree p of the polynomial kernel.
Regarding the three different binary predictor types $\vec{\alpha}_{fin}$, $\vec{\alpha}_{min}$, and $\langle\vec{\alpha}\rangle$, the final predictor $\vec{\alpha}_{fin}$ reveals the less smooth curve. This observation is caused by the fact that the final predictor $\vec{\alpha}_{fin}$ is highly biased by the latest training examples in each epoch. Typically, the minimizing and the average predictors, $\vec{\alpha}_{min}$ and $\langle\vec{\alpha}\rangle$, have a lower training error rate $\hat{\ell}_{S^{(5)}}$ than the final predictor $\vec{\alpha}_{fin}$, given a certain number of epochs N_{epoch} and a degree p of the polynomial kernel.

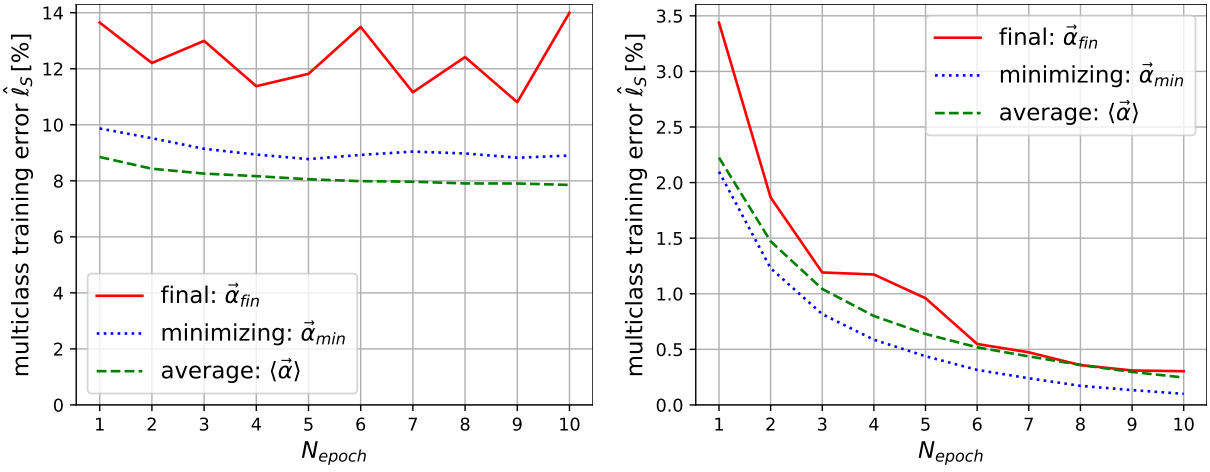
Interestingly, the training error rate $\hat{\ell}_{S^{(5)}}$ for the average predictor $\langle\vec{\alpha}\rangle$ can be lower than the training error rate $\hat{\ell}_{S^{(5)}}$ of the minimizing predictor $\vec{\alpha}_{min}$. The reason for this observation is that the minimizing predictor $\vec{\alpha}_{min}$ searches for the predictor that minimizes the training error rate $\hat{\ell}_{S^{(5)}}$ among the predictors $\vec{\alpha}$ that are produced during the training phase of the binary kernel perceptron algorithm, compare Alg. 1. The average predictor $\langle\vec{\alpha}\rangle$ is not included in this set, and can thus have a lower training error rate $\hat{\ell}_{S^{(5)}}$. However, the training error rate $\hat{\ell}_{S^{(5)}}$ of the minimizing predictor $\vec{\alpha}_{min}$ is always lower than the one of the final predictor $\vec{\alpha}_{fin}$. In general, the dependence of the training error rate $\hat{\ell}_{S^{(5)}}$ for the minimizing and the average predictor, $\vec{\alpha}_{min}$ and $\langle\vec{\alpha}\rangle$, are comparable, though the average predictor leads to a slightly smoother curve.

A closer look at the corresponding binary test error rates in Fig. 4 reveals that the test error rate $\hat{\ell}_{D^{(5)}}$ saturates at values significantly larger than 0 % for all degrees p of the polynomial kernel. For the lowest four degrees p of the polynomial kernel, a notably decrease of the binary test error rates $\hat{\ell}_{D^{(5)}}$ is observed. For the degrees $p \geq 5$, the test error rate for all the three predictor types does not drop below roughly 0.5 %. Here, the polynomial kernel overfits, such that an excellent performance on the training set $S^{(5)}$, but a rather poor performance on the test set $D^{(5)}$ is obtained. In general, the saturation of the test error rate $\hat{\ell}_{D^{(5)}}$ as a function of N_{epoch} implies that the application of the binary kernel perceptron algorithm using a polynomial kernel does not lead to arbitrary accurate binary predictors, as suggested by the training error rate $\hat{\ell}_{S^{(5)}}$.

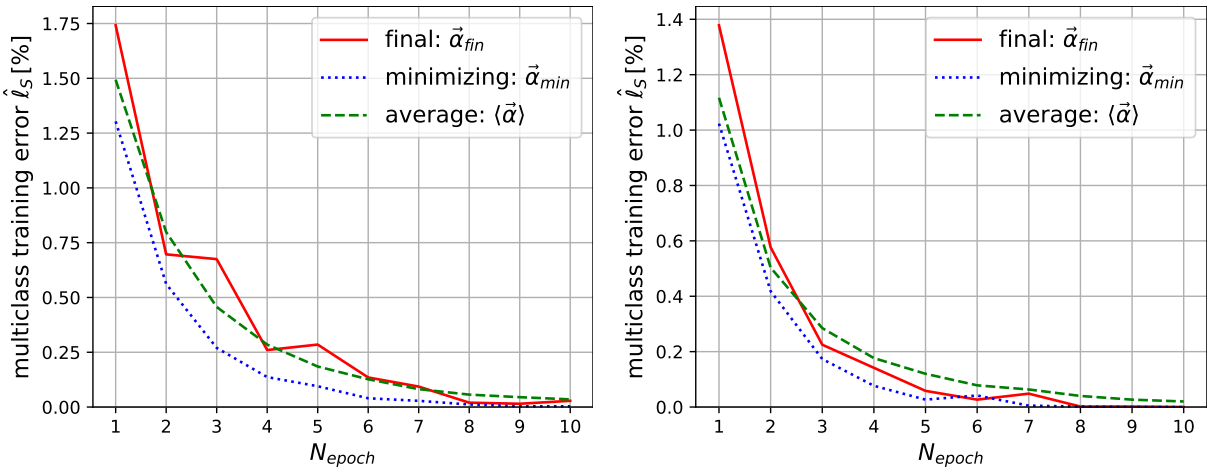
5.2 Multiclass Predictors

In order to obtain multiclass predictors from the multiclass kernel perceptron algorithm, the ten binary predictors for each digit a are combined. Fig. 5 reports the multiclass training error rates $\hat{\ell}_S$ as a function of the number of epochs N_{epoch} for the three different predictor types. Each panel corresponds to a different degree p of the polynomial kernel that is applied to train the multiclass kernel perceptron algorithm. The test error rate $\hat{\ell}_D$ of the same multiclass predictors can be found in Fig. 6.

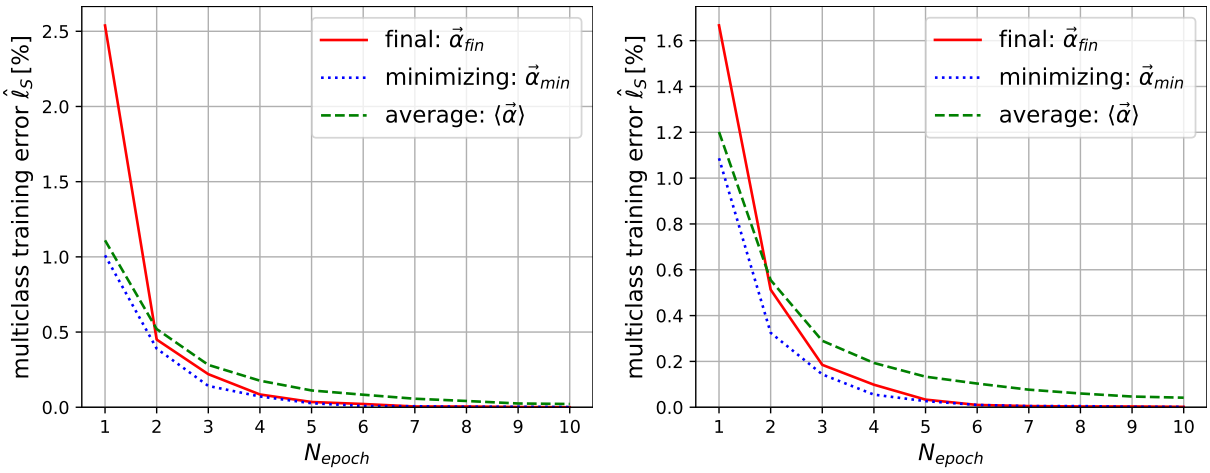
Unsurprisingly, the results for the multiclass predictors here reveal similar features as the binary training and test error rates for the binary predictors, reported in the last subsection. The polynomial kernel with less than $p = 3$ degrees leads to slow convergence of the multiclass training error rate $\hat{\ell}_S$ as a function of N_{epoch} as well as to comparably high test error rates $\hat{\ell}_D$. Especially for $p = 1$, the multiclass predictor underfits the data clearly, as both the training and test error rates are comparably large, and do not decrease with the number of epochs N_{epoch}. Polynomial kernels with a degree of at least $p = 3$ do not improve the test error rate, but also do not reduce the performance notably, i.e., for the applied degrees p, the polynomial kernel does not overfit the data strongly. In contrast to the binary predictors, here, the training error rate $\hat{\ell}_S$ of the minimizing predictor $\vec{\alpha}_{min}$ is lower than the one of the average predictor $\langle\vec{\alpha}\rangle$. For the test error rate $\hat{\ell}_D$, the final predictor $\vec{\alpha}_{min}$ performance typically worse than the other two realizations of the multiclass kernel perceptron algorithm.



(a) $p = 1$ (b) $p = 2$

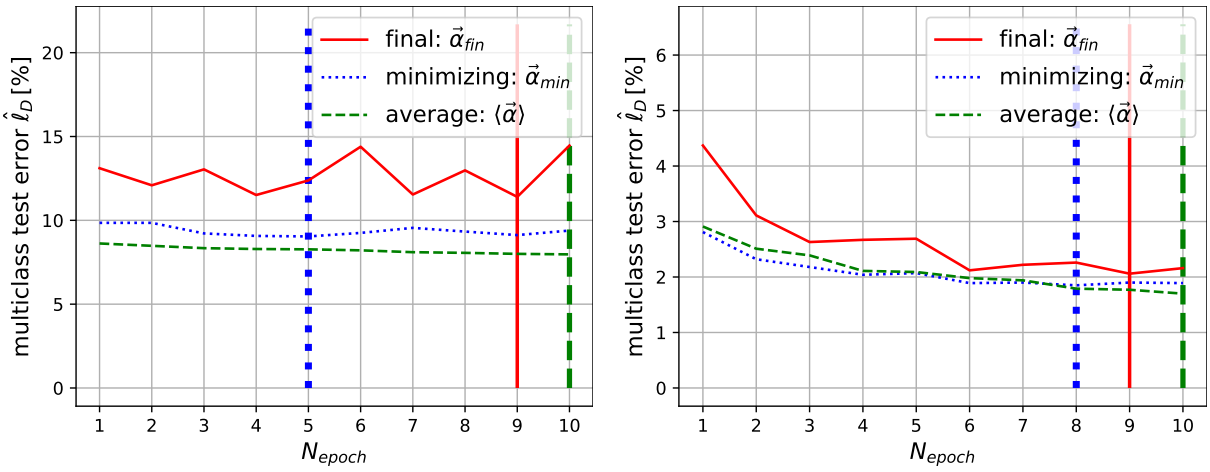


(c) $p = 3$ (d) $p = 4$

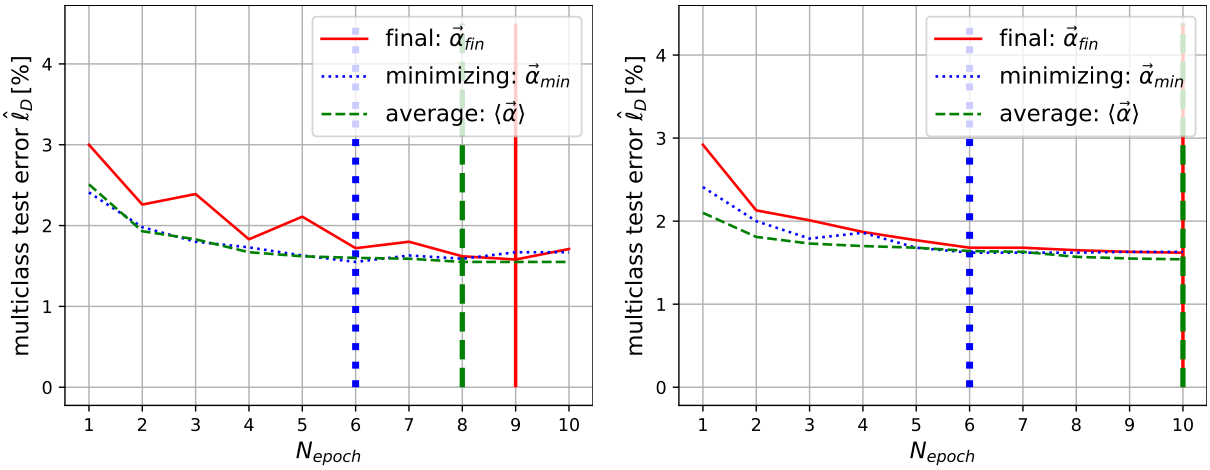


(e) $p = 5$ (f) $p = 6$

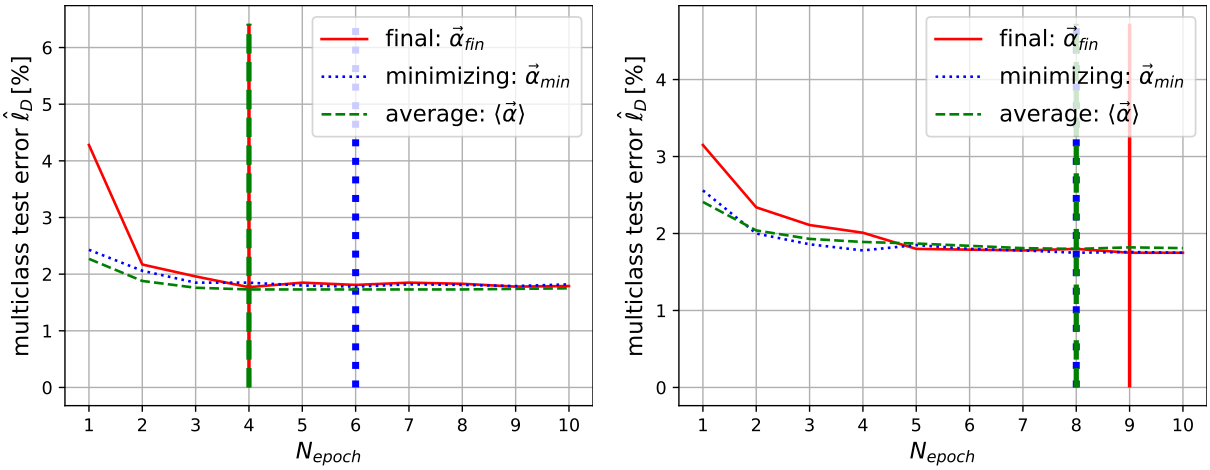
Figure 5: Multiclass training error rates $\hat{\ell}_S$ as a function of the number of epochs N_{epoch} for the training of multiclass predictors on the MNIST training set S. The multiclass predictors classify given images of handwritten digits with a label between 0 and 9. Each panel displays the multiclass training error rates $\hat{\ell}_S$ for a different degree p of the polynomial kernel for the three approaches to retrieve binary predictors inside the multiclass kernel perceptron algorithm. Note, the different scales of the vertical axes. The multiclass test error rates $\hat{\ell}_D$ for the same multiclass predictors are displayed in Fig 6.



(a) $p = 1$ (b) $p = 2$



(c) $p = 3$ (d) $p = 4$



(e) $p = 5$ (f) $p = 6$

Figure 6: Multiclass test error rates $\hat{\ell}_D$ as a function of the number of epochs N_{epoch} for the training of multiclass predictors on the MNIST training set S. The multiclass predictors classify given images of handwritten digits with a label between 0 and 9. Each panel displays the multiclass test error rates $\hat{\ell}_D$ for a different degree p of the polynomial kernel for the three approaches to retrieve binary predictors inside the multiclass kernel perceptron algorithm. Note, the different scales of the vertical axes. The multiclass training error rates $\hat{\ell}_S$ for the same multiclass predictors are displayed in Fig 5. In each panel, the vertical lines indicate the number of epochs N_{epoch} with the smallest test error rate for all three predictor types. The predictors obtaining overall the lowest test error rates are reported in Tab. 1.

5.3 Minimizing the Test Error Rate

In the previous subsection, the multiclass test error rates $\hat{\ell}_D$ of several multiclass predictors trained on S using the multiclass kernel perceptron algorithm were presented. In Tab. 1, for each of the three predictor types, the degree p of the polynomial kernel as well as the absolved number of epochs N_{epoch} for which the test error rate $\hat{\ell}_D$ is minimized is reported. For completeness, also the obtained training error rate $\hat{\ell}_S$ is stated. These three classifiers reveal the highest performance for the identification of handwritten digits in the MNIST dataset, and are the final result of the present work. As already observed, the final predictor $\vec{\alpha}_{fin}$ has a larger test error rate $\hat{\ell}_D$ than the minimizing and average predictor, $\vec{\alpha}_{min}$ and $\langle\vec{\alpha}\rangle$.

	$\vec{\alpha}_{fin}$	$\vec{\alpha}_{min}$	$\langle\vec{\alpha}\rangle$
$\hat{\ell}_D$ [%]	**1.58**	**1.55**	**1.54**
$\hat{\ell}_S$ [%]	0.0150	0.127	0.0200
p	3	3	4
N_{epoch}	9	6	10

Table 1: Multiclass predictors that obtain the lowest test error rates $\hat{\ell}_D$ on the test set D of the MNIST database of handwritten digits applying the multiclass kernel perceptron algorithm. In the table, the degree p of the polynomial kernel as well as the number of epochs N_{epoch} for which the test error rate $\hat{\ell}_D$ is minimized are stated for each of the three predictor types $\vec{\alpha}_{fin}$, $\vec{\alpha}_{min}$, and $\langle\vec{\alpha}\rangle$. In the present work, kernel degrees from $p = 1$ to $p = 6$ for up to $N_{epoch} = 10$ training cycles over random permuations in the training set S were considered. For completeness, the corresponding training error rate $\hat{\ell}_S$ is reported.

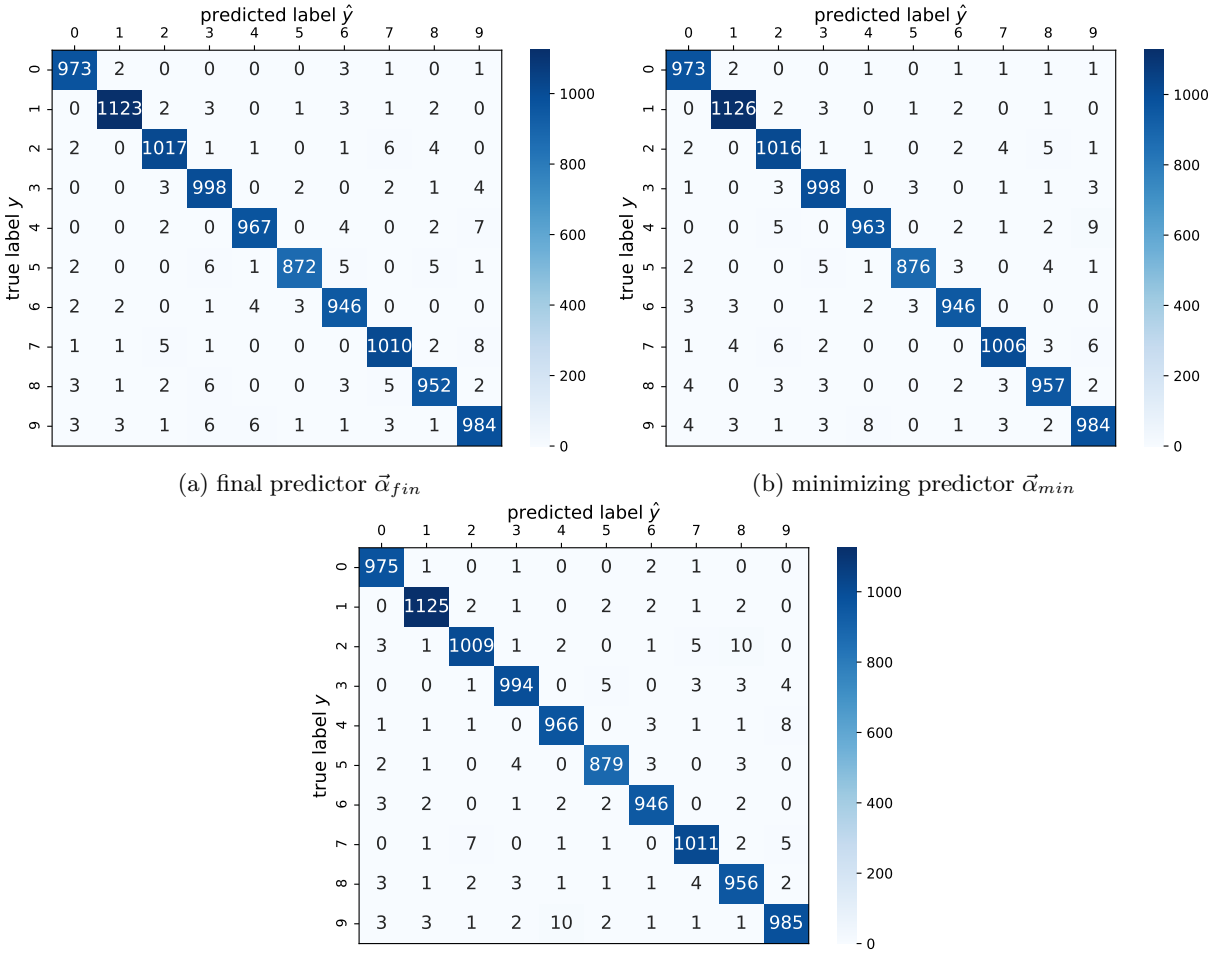
Finally, in Fig. 7, the confusion matrix for the three multiclass predictors with the lowest test error rates $\hat{\ell}_D$ are reported. Clearly, all the predictors perform well on the test set D. There is no particular digit for which a particular large number of missclassifications is observed. This refers to the observation in Sec. 2 that the training set S is balanced. Therefore, there is no bias towards particular digits, and for all ten digits roughly the same amount of information is available.

6 Discussion

The result of the implemention of the multiclass kernel perceptron algorithm for the identification of handwritten digits in the training set D of the MNIST database lead to test error rates between 1.58 % and 1.54 %, as reported in the previous Section. As the MNIST database is a common benchmark dataset for machine learning algorithms for the recognition of images, there is plenty of literature regarding the test error rates obtained with other learning algortihms. In particular, the websites `http://yann.lecun.com/exdb/mnist/` and `https://paperswithcode.com/sota/image-classification-on-mnist` offer insightful overviews of the performance of different image recognition techniques on the MNIST database.

Tab. 2 compares the test error rates that were obtained by applying the multiclass kernel perceptron algorithm in the present work with results from other learning algorithms in the literature. The performance of the multiclass kernel perceptron algorithm is comparable to the test error rate of 1.4 % obtained with the closely related support vector machine (SVM) with a gaussian kernel. The K-nearest-neighbors algorithm with a euclidean norm leads to a test error rate of 5.0 %. The results for the SVM as well as for the KNN algorithm improve signficantly when applying various image preprocessing techniques. The lowest test error rate that the author of the present paper aware of is 0.16 % for the application of a rather complex ensemble of several convolutional neural networks in combination with different image preprocessing techniques. The reduction of the test error rate on the test set D of the MNIST database is an ongoing field of research, where improved learning algorithms are developed and presented continually.

The main advantage of the application of the multiclass kernel perceptron algorithm for the recognition of handwritten digits is that it is a sequential algorithm. In practice, this is important when the database of training images S is growing as it might be the case for many applications, e.g. for self-driving vehicles. A major drawback of the multiclass kernel perceptron algorithm is that the obtained prediction for the label $\hat{y}$ of a given image $\vec{x}$ is outputed without any uncertainty estimation. This might be important for applications where decisions must be made on basis of a prediction. One possibilty to overcome this issue is to train the multiclass kernel perceptron (for the same hyperparameters N_{epoch} and p) several times for



(a) final predictor $\vec{\alpha}_{fin}$

(b) minimizing predictor $\vec{\alpha}_{min}$

(c) average predictor $\langle\vec{\alpha}\rangle$

Figure 7: Confusion matrix for the multiclass predictor that minimizes the test error rate $\hat{\ell}_D$ on the test set D of the MNIST database, for (a) the final predictor $\vec{\alpha}_{fin}$, (b) the minimizing predictor $\vec{\alpha}_{min}$, and (c) the average predictor $\langle\vec{\alpha}\rangle$, as reported in Tab. 1.

learning algorithm	image preprocessing	test error rate [%]
KNN, euclidean norm (L2)	none	5.0
KNN, euclidean norm (L2)	deskewing	2.4
KNN, euclidean norm (L2)	deskewing, noise removal, blurring	2.4
multiclass kernel perceptron	**none**	**1.54 to 1.58**
SVM, gaussian kernel	none	1.4
SVM, polynomial kernel ($p = 4$)	deskewing	1.1
ensemble of CNNs	rotation, translation, random erasure	0.16

Table 2: Comparison of the test error rate on the MNIST test set D obtained by training the multiclass kernel perceptron algorithm with other image recognition techniques in the literature.

different random permutations over the training examples in the training set S, such that an ensemble of multiclass predictors is obtained. Then, a prediction for the image $\vec{x}$ consists in fact of a distribution of predictions that give insights on the uncertainty of the prediction.
In the same way, uncertainties for the reported test error rates could be obtained. The uncertainties would allow a more meaningful comparisons between the multiclass predictors obatined for different realizations of the multiclass kernel perceptron algorithm as well as with other image recognition techniques.

The determination of prediction uncertainties in machine learning is a subject of current research [9][10]. The multiclass kernel perceptron algorithm might not be the best choice if predictions with reliable uncertainties are desired because the computational complexity of this algorithm increases significantly as the algorithm has to be trained completely several times. Other algorithms such as bayesian neural networks might be more suitable as the implement intrinsically a probabilistic component [11].

7 Conclusion & Outlook

In the present work, three realizations of the multiclass kernel perceptron algorithm lead to test error rates of 1.54 %, 1.55 %, and 1.58 % on the test set of the MNIST database. This result compares well with other classical machine learning approaches such as support vector machines or K-nearest-neighbor algortihms. In general, complex neural networks have a better performance. The obtained test error rate of the multiclass kernel perceptron algorithm might further decrease, when preprocessing (e.g. rotating, blurring) the images prior to the training of the multiclass predictors.
It has been seen that the multiclass kernel perceptron algorithm has the major advantage that it is an online learning algorithm that handles growing databases well. The main drawback is a missing quantification of the uncertainties on the predictions that might be critical in some areas of application.

Further research regards the expansion of the predictors beyond the case of the MNIST database, e.g. to include letters from A to Z [12] It might also be of interest to include a class “invalid input” for images that can not be classified as a certain digit. This could reduce the number of missclassified digits in some applications.

References

[1] Arien P. Sligar. "Machine Learning-Based Radar Perception for Autonomous Vehicles Using Full Physics Simulation." In: *IEEE Access* 8 (2020), pp. 51470–51476. DOI: 10.1109/ACCESS.2020.2977922.

[2] Deepak Gupta Anvita Saxena Ashish Khanna. "Emotion Recognition and Detection Methods: A Comprehensive Survey." In: *Journal of Artificial Intelligence and Systems* 2 (2020), 53–79. DOI: https://doi.org/10.1038/s41746-020-00376-2.

[3] A. Esteva, K. Chou, and S. Yeung. "Deep learning-enabled medical computer vision." In: *npj Digit. Med.* 4.5 (2021). DOI: https://doi.org/10.1038/s41746-020-00376-2.

[4] Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. *The MNIST database of handwritten digits*. 2021. URL: http://yann.lecun.com/exdb/mnist/ (visited on 06/02/2021).

[5] Dariel Dato-on (User) on Kaggle. *MNIST in CSV*. 2021. URL: https://www.kaggle.com/oddrationale/mnist-in-csv (visited on 06/02/2021).

[6] Jianhua Xu and Xuegong Zhang. "A multiclass kernel perceptron algorithm." In: *2005 International Conference on Neural Networks and Brain*. Vol. 2. 2005, pp. 717–721. DOI: 10.1109/ICNNB.2005.1614728.

[7] M. A. Aizerman, E. A. Braverman, and L. Rozonoer. "Theoretical foundations of the potential function method in pattern recognition learning." In: *Avtomat. i Telemekh.* 25.6 (1967), pp. 917–936.

[8] Rosenblatt. "The Perceptron: A Perceiving and Recognizing Automaton." In: *Conrell Aeronautical Laboratory* 85-460-1 (1957).

[9] Moloud Abdar et al. "A Review of Uncertainty Quantification in Deep Learning: Techniques, Applications and Challenges." In: *CoRR* abs/2011.06225 (2020). arXiv: 2011.06225. URL: https://arxiv.org/abs/2011.06225.

[10] Barbara Hammer and Thomas Villmann. "How to process uncertainty in machine learning?" In: Jan. 2007, pp. 79–90.

[11] Yongchan Kwon et al. "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation." In: *Computational Statistics & Data Analysis* 142 (2020), p. 106816. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2019.106816. URL: https://www.sciencedirect.com/science/article/pii/S016794731930163X.

[12] Gregory Cohen et al. *EMNIST: an extension of MNIST to handwritten letters*. 2017. arXiv: 1702.05373 [cs.CV].