Kernel Perceptron for MNIST dataset prediction

Loris Bartesaghi

09/02/2022

Abstract

This paper presents Kernel Perceptron algorithm as a machine learning approach for digit recognition task by using as input set a vector, representing the pixel data of a digit image. This paper examines the performance of the polynomial kernel for online handwritten digit recognition, after the process of hyperparameters optimization. Confusion matrices, recall and precision metrics will be used to deepen the analysis on the performance of the predictor. Finally, an analysis on the test accuracy of the predictor will be made, varying the training set size.

Keywords : Digit Recognition, Kernel Perceptron, Mnist Dataset

1 Introduction

In machine learning, the Perceptron, introduced by Frank Rosenblatt in 1957, is an algorithm for supervised classification of an input into one of several possible non-binary outputs. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the input features. The learning algorithm for Perceptron is an online algorithm, i.e. it processes elements in the training set one at a time.
A prediction based on a linear function could be the cheapest way for solving a classification task, but it fails to capture non-linear decision boundaries and, in reality, most of the data available are non-linear.
To cope with this problem, kernel methods are introduced. These methods owe their name to the use of kernel functions, which enable them to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called the "kernel trick".
In the second chapter of the paper, a theoretical background is introduced to explain the difference between the classical Perceptron and the Kernel Perceptron. In the third chapter a preliminary analysis is carried out on the data, the process to tune the hyperparameters is performed and subsequently the performance of the obtained predictors is evaluated. In the last chapter the conclusion are written.

2 Theoretical background.

The Perceptron is a linear classifier, i.e. a function $h : \mathbb{R}^d \to \mathbb{R}$ such that $h(x) = sgn(\omega^T x - c)$ for some $c \in \mathbb{R}$, where the function sgn is the activation function for which $sgn(t) = +1$ if $t > 0$ and -1 otherwise. Geometrically, the Perceptron defines an hyperplane $H \subset \mathbb{R}^d$ such that, if H^+, H^- are the two halfsapces on each side pf the hyperplane, could be shown that

$$h(x) = \begin{cases} +1 \text{ if } x \in \mathbb{H}^+ \\ -1 \text{ if } x \in \mathbb{H}^- \end{cases}$$

The Perceptron algorithm, in the case of linearly separable data, defines an homogeneous separating hyperplane. The pseudo-code is defined as follow.

Input: Training set $(x_1, y_1), \cdots, (x_m, y_m)$
Inizialization $\omega = (0, \cdots, 0)$
Repeat

1. Read next training example (x_t, y_t)
2. If $y_t \omega^T x_t \leq 0$, then $\omega \leftarrow \omega + y_t x_t$

Until $\gamma(\omega) > 0$
Output ω

As said before, Perceptron defines an hyperplane in the space described by the features of the examples. Consequently, it can't captures non-linear relationship between features and the consequence is that, for the classical Perceptron, a large bias error arise.
Features expansion is a popular technique to reduce bias error, which introduce new parameters by constructing new features through nonlinear combination of the base features. It could be seen as a function $\phi : \mathbb{R}^d \to \mathcal{V}$, mapping each data point $x \in \mathbb{R}^d$ to an higher-dimensional space $\mathcal{V}$. Fixing such a $\phi : \mathbb{R}^d \to \mathbb{R}^N$ and considering a binary classifier $h : \mathbb{R}^d \to \{-1, 1\}$ defined by $h(x) = sgn(\omega^T \phi(x))$, the non-linear classifier in $\mathbb{R}^d$ correspond to a linear classifier in R^N.
The computation of ϕ, and thus the prediction, could become infeasible even for moderately large power degrees of the base features. Kernel trick could be helpful to cope with this computational problem for many of many algorithms for learning linear predictor. In particular, the kernel trick helps to find an efficiently computable kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that:

$$K(x, x') = \phi(x)^T \phi(x') \qquad \forall x, x' \in \mathbb{R}^d$$

So, given a kernel K, it is possible to write a non linear classifier generated by the Perceptron as

$$h_K(x) = sgn\left(\sum_{s \in S} y_s K(x_s, x)\right)$$

S is the set of indices s of the training examples (x_s, y_s) on which the Perceptron made an update.

3 Experimental evaluation.

The approach used for solving multiclass classification is the "one-vs-rest"(OvR for short, also referred to as One-vs-All or OvA): it is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem.
In this paper, in order to extract a single binary predictor, the algorithm was run on the 75% of the training set (in the paper, the 75% of the training set will mentioned as S_{train}) for a given combination of epochs (1 to 10) and degree of the polynomial kernel (1 to 6). At the end of each epoch, binary predictors are saved, and from the ensemble the following two binary classifiers are extracted:

- the average of the binary predictors in the ensemble. For the sake of simplicity, in this paper, the multi-class predictor composed by the aggregation of the binary classifiers generated with this method will be called "Predictor_mean_binary_classifiers".
- the predictor achieving the smallest error on S_{train} among those in the ensemble. Again, for the sake of simplicity, in this paper, the multi-class predictor composed by the aggregation of the binary classifiers generated with this method will be called "Predictor_best_binary_classifiers".

To optimize the hyperparameters of the Kernel Perceptron algorithm, the resulting multi-class classifiers are tested on the remaining 25% of the training set, the validation set S_{dev}, and the best combination of hyperparameters is choosen. The learning algorithm has been re-run on the entire training set using the value of the hyperparameters corresponding to the combination achieving the largest accuracy on S_{dev}. Finally, the performance of the predictors was evaluated on the test set.

3.1 Data understanding and visualization.

The training set is composed by 8'000 examples, while the test set is composed by 2'500 observations. Each example is a vector of 784 number between 0 and 255, which represents an image of 28 $\times$ 28 pixels. "Figure 1" shows an image of the first example.

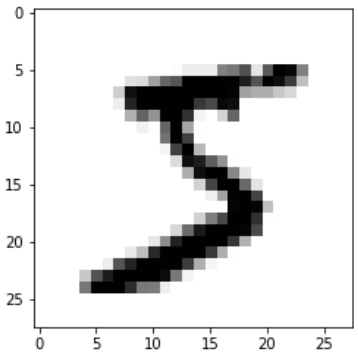


Figure 1: Digit image example.

"Figure 2" shows that the training set is quite balanced, which is an important characteristic of the dataset to generate a model that performs higher accuracy.

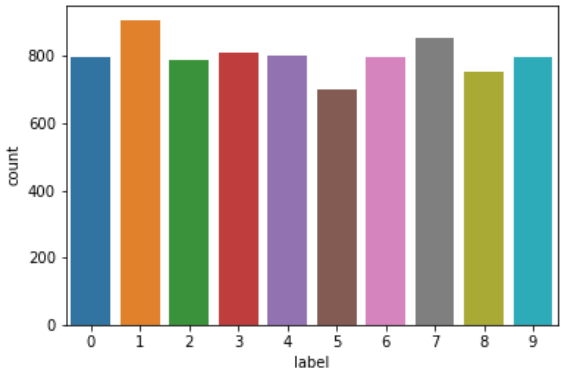


Figure 2: Frequency histogram of digits.

All the images were standardized by subtracting the mean over the training set from each pixel and dividing by their standard deviation. This is done to improve execution time: all these matrices multiplications would be faster with small numbers compared to very large number.

$$Z = \frac{X - \mu}{\sigma}$$

To have more comprehension about data, the mean of the vector of each example is computed. Then, grouping the examples by digit, the mean and standard deviation for each digit group were calculated. This analysis will be useful in analyzing the confusion matrix, allowing us to understand what are the similarities between the digits that lead to a predictor error. "Figure 3" graphically shows the mean and standard deviation for each digit group.

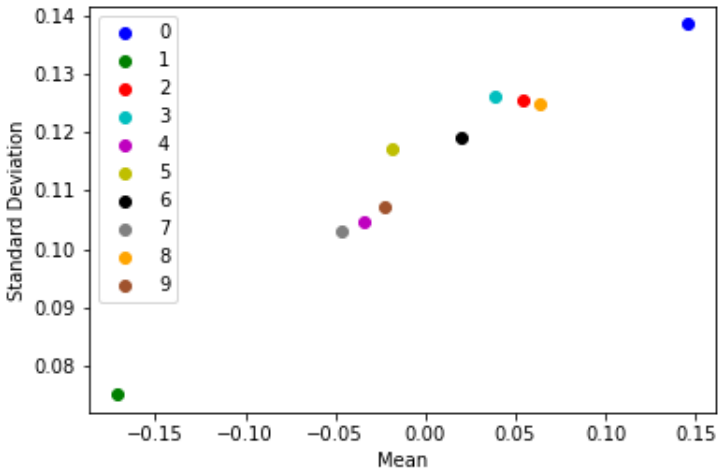


Figure 3: Averages and standard deviations for each digit group.

3.2 Hyperparameters optimization.

A training set of 8'000 examples was considered and, subsequently, it has been divided into S_{train}(75% of the training set) and S_{dev}(25% of the training set), which are necessary for the hyperparameter optimization process. The learning algorithm is run on S_{train} for each combination of epochs and polynomial kernel grades. The resulting predictors are tested on S_{dev}. The final predictor is obtained by running the learning algorithm on the 8,000 examples using the value of the hyperparameters corresponding with the predictor with the highest accuracy on the validation set. The hyperparameters tuning is performed on both the predictors generated by the two different methods ("Predictor_best_binary_classifiers" and "Predictor_mean_binary_classifiers").

"Figure 4" shows the accuracy on S_{train} and on S_{dev} of *Predictor_best_binary_classifiers*.

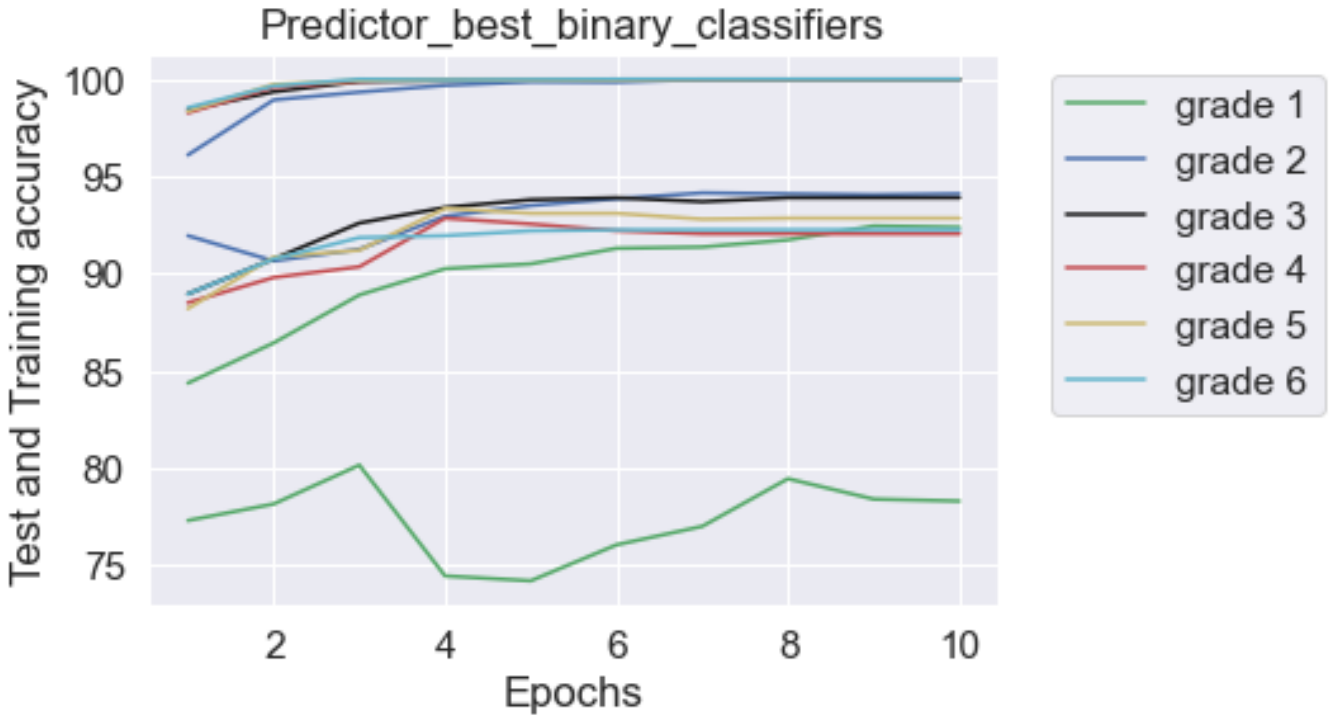


Figure 4: Accuracy on training and development set of Predictor_best_binary_classifier

From "Figure 4" it is possible to understand that for all kernel degrees greater than two, the accuracy on S_{train} reaches almost 100%. Furthermore, it is possible to note that the development set accuracy performances of the predictors are better for kernel grades of 2 or 3. Higher kernel grades could incur in overfitting. For all predictors generated by different kernel grades, increasing the number of epochs beyond the sixth does not lead to significant performance improvements.

"Figure 5" shows the accuracy on S_{train} and on S_{dev} of the *Predictor_mean_binary_classifiers*.

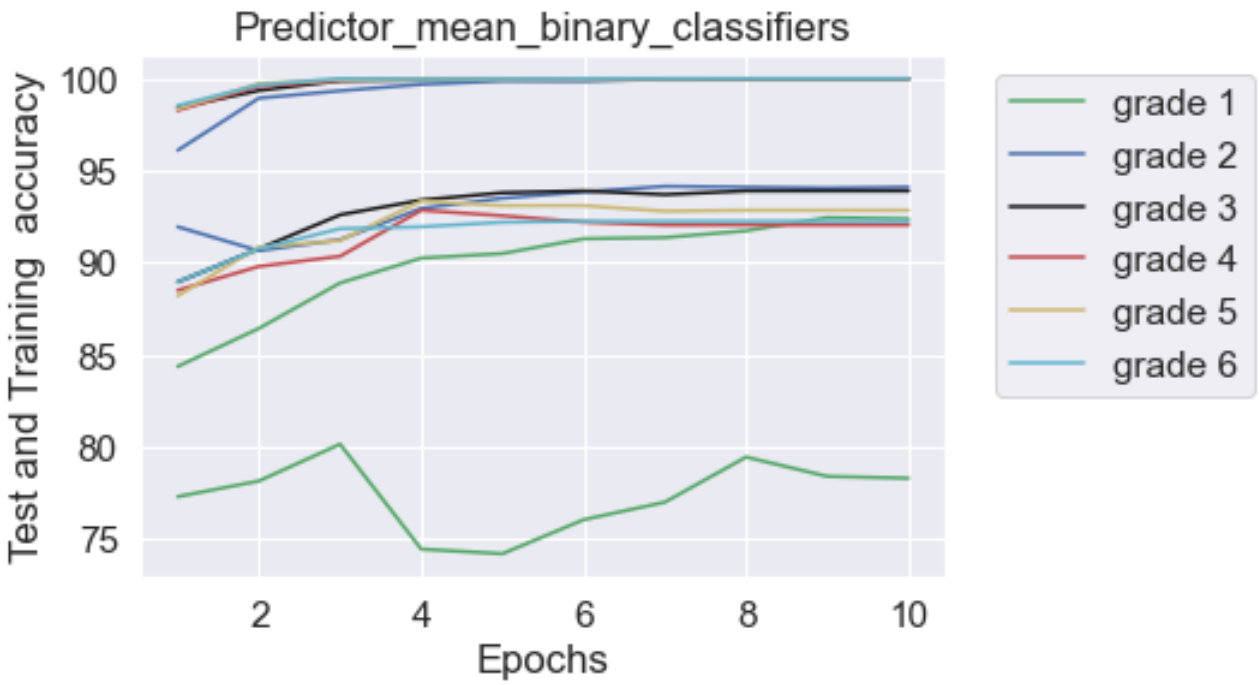


Figure 5: Accuracy on training and development test of Predictor_mean_binary_classifier

The same considerations made for the *Predictor_best_binary_classifiers* plot can be made by considering this figure. Furthermore, in both figures, it is clearly possible to notice a much lower performance for the first degree of the polynomial kernel.
For both predictors, generated with different methods, the best number of epochs is 7 while the degree of the polynomial kernel is 2.

3.3 Performance Analysis.

In the following table are expressed the values of training accuracy and the accuracy on the test set for the two different predictor trained using the best values for the hyperparameters, found in the previous paragraph.

	Predictor_mean_binary_classifiers	Predictor_best_binary_classifiers
Training accuracy	99.84	99.94
Test accuracy	91.76	90.2

From the table, it is possible to understand that *Predictor_mean_binary_classifiers* has a worse training accuracy than *Predictor_best_binary_classifiers*, but a better performance on the test set. This result can be a clue of overfitting for *Predictor_best_binary_classifiers*.

In "Figure 6" and in "Figure 7" the confusion matrices are shown graphically, which allow to make a more accurate analysis of the predictor's prediction errors. The digit forecasted by the predictor is indicated on the x-axis, while on the y-axis it is possible to notice the real value of the digit.

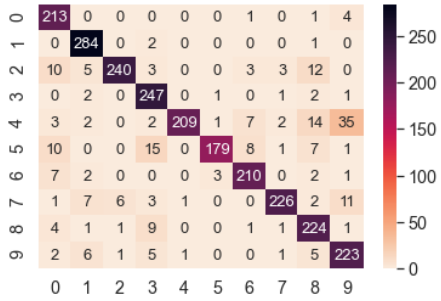


Figure 6: Confusion Matrix Predictor_best_binary_classifiers

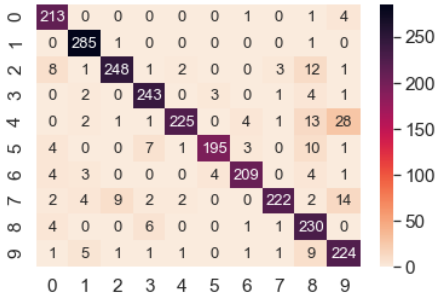


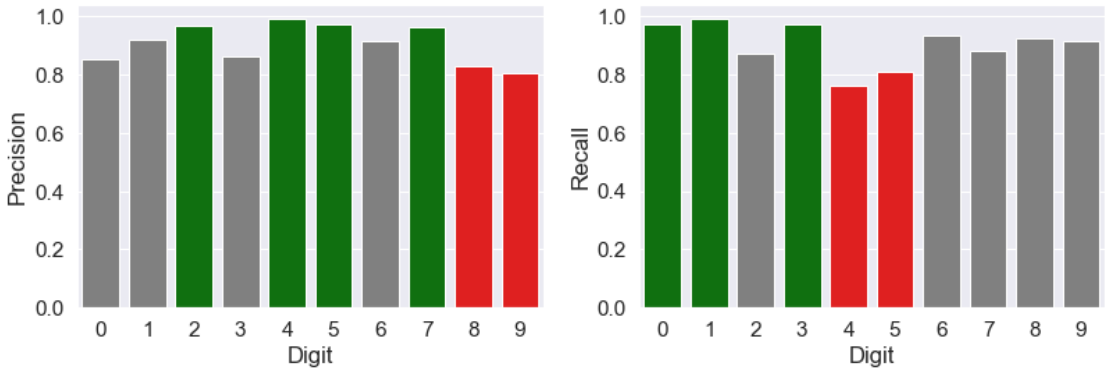
Figure 7: Confusion Matrix Predictor_mean_binary_classifiers

Thanks to the information given by confusion matrices, it is possible to calculate measures such as precision and recall. Precision is defined as $\frac{TP}{TP+FP}$, and it allows us to understand, for all the numbers, which portion of digit prediction is truly correct. While recall is defined as $\frac{TP}{TP+FN}$, and allows us to understand, for all the numbers, which portion of digit has been correctly identified.

The following table shows the precision and recall values for each digit, considering the prediction of Predictor_best_binary_classifiers.

	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
Precision	0.852	0.919	0.968	0.864	0.991	0.973	0.913	0.962	0.830	0.805
Recall	0.973	0.990	0.870	0.972	0.760	0.810	0.933	0.879	0.926	0.914

The next two plots allow to visualize the precision and recall values of Predictor_best_binary_classifiers. The bars of the digits are colored green if the precision/recall values are higher than 0.95 and, on the other hand, they are colored red if the precision/recall values are lower than 0.825. The coloring was done in order to better visualize the best and worst performances for each digit for this predictor.



(a) Precision (b) Recall

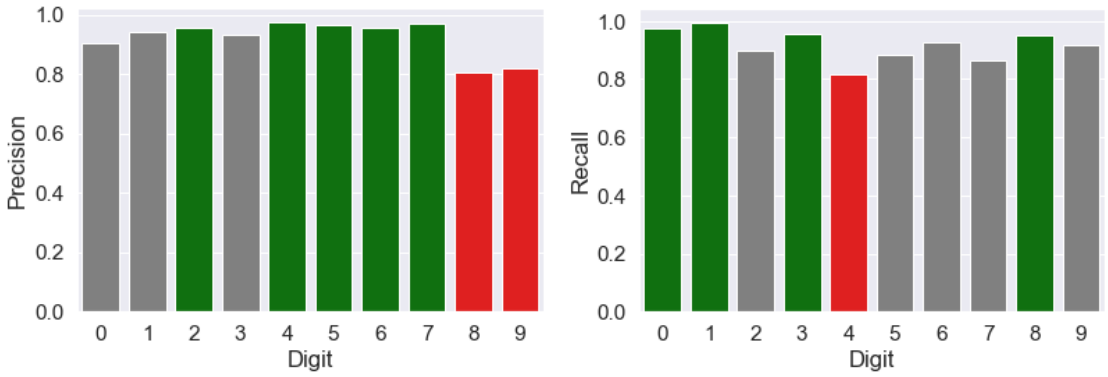
From the graph it is possible to understand that the predictor returns an excessive number of digits 'eight' and 'nine'. The confusion matrix can help to determine which numbers are most confused for 'eight' and 'nine'. It is possible to notice that for 35 times the true digit 'four' was predicted as 'nine' (indeed, it is also possible to notice a particularly low recall value for digit 'four') and 11 times the digit 'seven' was recognized as 'nine'.
From "Figure 3", it is possible to notice that the digits 'four', 'seven' and 'nine' form a cluster in the plot. This characteristic could explain a higher number of errors of the classifier despite the numbers taken into consideration are graphically different.
Instead, the number 'eight' is predicted 14 times as 'four' and 12 times as 'two'. As before, 'eight' and 'two' occupy a very similar position in "Figure 3".
It is also possible to notice a low recall value for digit five, which is incorrectly predicted as 'zero', 'three', 'six' and 'eight' respectively 10,15, 5 and 7. They are all digits represented in the upper right part of "Figure.3". An additional reason could be that the number 'five' is, for the number of examples, the least present in the training set.

The following table shows the precision and recall values for each digit, considering the prediction of Predictor_mean_binary_classifiers.

	Zero	One	Two	Three	Four	Five	Six	Seven	Eight	Nine
Precision	0.903	0.944	0.954	0.931	0.974	0.965	0.954	0.969	0.804	0.818
Recall	0.973	0.993	0.899	0.957	0.818	0.882	0.929	0.864	0.950	0.918

The next two plots allow to visualize the precision and recall values. Again, the bars of the digits are colored green if the precision/recall values are higher than 0.95 and, on the other hand, they are colored red if the precision/recall values are lower than 0.825.

From the plots it is possible to understand that the predictor returns an excessive number of digits 'eight' and 'nine'. The confusion matrix can help to determine which numbers are most confused for 'eight' and 'nine'.
It is possible to notice that for 26 times the true digit 'four' was predicted as 'nine' (indeed, it is also possible to notice a particularly low recall value for digit 'four') by the predictor and 14 times the 'seven' was recognised as digit 'nine'.



(c) Precision

(d) Recall

From "Figure 3", you can see that the values 'four', 'seven' and 'nine' form a cluster in the plot. This characteristic could explain a higher number of errors of the classifier despite the numbers taken into consideration are graphically different.
Instead, the number 'eight' is predicted 13 times as 'four' and 12 times as 'two'. As before, 'eight' and 'two' occupy a very similar position in "Figure 3".

3.4 Predictors' performances based on number of training examples.

In machine learning, to improve predictors' performances, the amount of examples included in the training set are fundamental. "Figure 8" shows an example of the training and test accuracy behaviour, changing the number of examples included in the analysis. The test accuracy is computed on a test set of 2'000 example.

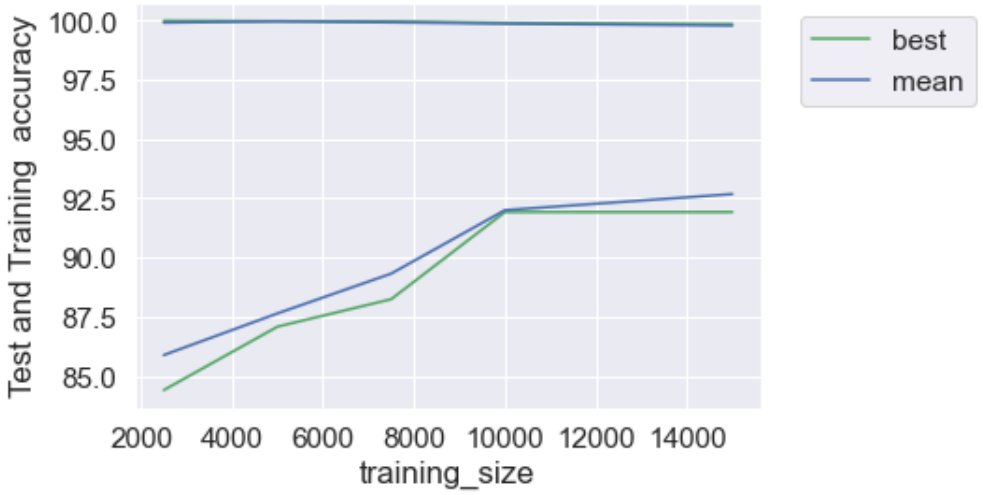


Figure 8: Training and test accuracy increasize the number of example of the final predictor.

In the table are shown the values of test accuracy, given the number of examples.

	2'500	5'000	7'500	10'000	15'000
Mean classifier	0.8588	0.8764	0.8932	0.92	0.9267
Best classifier	0.8439	0.8708	0.8824	0.9192	0.9192

In "Figure 8" the expectations deriving from the theory are respected. By increasing the size of the training set, test accuracy improves. Furthermore, it is very interesting to note that Predictor_best_binary_classifiers, after 10'000 examples, doesn't improve its performance in terms of test accuracy. Considering Predictor_mean_binary_classifiers, it is possible to notice a slight improvement in performance from 10'000 examples to 15'000.

4 Conclusion.

The experimental evaluation of the Kernel Perceptron algorithm on the MNIST dataset made it possible to understand that an excessive number of degrees of the polynomial kernel and of the epochs does not lead to an increase in the performance of the predictor. In fact, the degrees of the polynomial beyond the third degree lead to a decrease in the performance on the development set, a symptom of the overfitting phenomenon.
Thanks to these results, it is possible to state that the Kenrel Perceptron with hyperparameters of 2 for the degree of the kernel polynomial and 7 as the number of epochs, generates the best predictor for the digit recognition problem.
The confusion matrices allowed us to understand the digits that the two predictors misclassified in the prediction. It was important to understand that the numbers were not confused by the shape of the digit, but by the position on the "Figure 3", where digit groups' means and standard deviations are shown.
Furthermore, in paragraph 3.4. was confirmed that by increasing the number of examples in the training set, for the Predictor_mean_binary_classifiers, it was possible to reduce the test error. Result that respects the expectations deriving from the theory. In contrast to this conclusion, the Predictor_best_binary_predictor's performance on the test set doesn't improve the performance after the inclusion of more than 10,000 examples in the training set. Probably the composition of this predictor, formed by the binary classifiers with the best performance in the respective digits, introduce overfitting.
Further analysis could include comparing the performance between the Perceptron Kernel and the classic Perceptron, trying to understand the real performance improvements that the introduction of the Kernel functions. Furthermore, it could be interesting to approach the problem of digit recognition, considering the digit groups' averages and the relative standard deviations, in order to reduce the problem to a two-dimensional problem.

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.