NTNU
Kunnskap for en bedre verden

TTT4275 - Estimation, Detection and Classification

Classification Project

Authors:
Anders Slåkvik
Nicolai Adil Øyen Aatif

April, 2024

Abstract

This project is divided into two distinct classification tasks, employing different datasets and methodologies to demonstrate the application of classification techniques in pattern recognition. The first task, known as the Iris task, utilizes the well-known Fisher-Iris dataset to develop a linear discriminant classifier aimed at distinguishing between three species of the Iris flower based on sepal and petal measurements. This task shows the efficiency of linear classification methods in a near-linearly separable feature space, where experiments were conducted to optimize model performance by adjusting step factors, removing feature, and evaluating the impact using metrics such as Mean Square Error (MSE), error rates, and confusion matrices.

The second task focuses on the classification of handwritten digits $(0-9)$ using the MNIST dataset, a benchmark for image recognition algorithms. This task implements the K-Nearest Neighbors (KNN) algorithm with Euclidean distance metric to classify the digits. Two approaches were tested: using the entire dataset of 60,000 images as templates, and a reduced template set obtained through clustering, aiming to decrease computational demands while maintaining accuracy. Performance was assessed based on error rates, runtime, and the analysis of confusion matrices, illustrating the trade-offs between computational efficiency and accuracy.

Contents

1 Introduction

Classification algorithms are essential tools in data analysis for assigning categories to objects based on their features. These algorithms range from simple decision trees to complex neural networks capable of handling vast, unstructured datasets. The primary aim is to accurately predict object categories from predefined options.

This project explores two classification tasks: The first uses the Iris dataset to differentiate between three species of the Iris flower using linear discriminant analysis. This task focuses on linear separability and the impact of feature selection. The second task classifies handwritten digits (0-9) from the MNIST dataset using the K-Nearest Neighbors (KNN) algorithm, examining methods to optimize computational efficiency and accuracy.

Both tasks are intended to demonstrate the practical application of classification algorithms in data prediction and the decision-making processes behind algorithm selection and implementation. The detailed project documentation and codebase are available on GitHub [1].

2 Theory

This chapter is meant to inform and give necessary theory to understand the mathematical tools and concepts used in this project.

2.1 The Linear Classifier

Linear classifiers are sufficient in linear separable problems. A type of linear classifier is the discriminant classifier, which is defined by

$$z_i(x) = w_i^\top x + w_{i0}, \quad i = 1, 2, \cdots, C \tag{1}$$

where w_i is the weight vector and w_{i0} is the bias, also known as threshold weight [5].
When there are more than two classes, it is beneficial to write (1) on the matrix form given by

$$\mathbf{z} = \mathbf{W}_{original}\ \mathbf{x}_{original} + \mathbf{w_0} \tag{2}$$

Equation (2) can be simplified with the use of the bias trick [5]:

$$\mathbf{z} = \underbrace{[\mathbf{W}_{original}\ \ \mathbf{w}_0]}_{\mathbf{W}}\ \underbrace{[\mathbf{x}_{original}^\top\ 1]^\top}_{\mathbf{x}} = \mathbf{W}\mathbf{x} \tag{3}$$

During the training phase, this matrix $\mathbf{W}$ is altered in order to get the optimal weights by minimizing a cost function, which is a formula to quantify the error between predicted outputs and ground truth. The MSE[1] is an example of such cost function, and is given by

$$MSE = \frac{1}{2}\sum_{k=1}^{N}(g_k - t_k)^\top (g_k - t_k) \tag{4}$$

t_k is the ground truth. Because this approach relies on labeled data[2], it falls under the category of supervised classification methods. g_k is an activation function which squashes the output, z_k, between zero and one. A type of activation function is the sigmoid function [5], which is given by

$$g_{ik} = \frac{1}{1 + \mathrm{e}^{-z_{ik}}}, \quad i = 1, 2, \cdots, C \tag{5}$$

[1]Mean Square Error.
[2]The ground truth, t_k.

Unfortunately, finding the minima of equation (4) with respect to the weights $\mathbf{W}$ does not have an explicit solution [5]. Gradient descent serves as a compensatory method, which gives

$$\nabla_{\mathbf{W}} MSE = \sum_{k=1}^{N} \nabla_{g_k} MSE \; \nabla_{z_k} g_k \nabla_{\mathbf{W}} z_k \tag{6}$$

where

$$\nabla_{g_k} MSE = g_k - t_k \tag{7a}$$

$$\nabla_{z_k} g = g_k \circ (1 - g_k) \tag{7b}$$

$$\nabla_{\mathbf{W}} z_k = x_k^{\top} \tag{7c}$$

The result of embedding equation (7a), (7b) and (7c) in equation (6), is given by

$$\nabla_{\mathbf{W}} MSE = \sum_{k=1}^{N} [(g_k - t_k) \circ g_k \circ (1 - g_k)] x_k^{\top} \tag{8}$$

Given that the objective is to minimize equation (4) with respect to $\mathbf{W}$, the updates are made by adjusting $\mathbf{W}$ in the direction opposite to the gradient, as specified in equation 6 [5]. The gradient points in the direction of steepest ascent, which is why the update rule for $\mathbf{W}$ is given by

$$\mathbf{W}(m) = \mathbf{W}(m-1) - \alpha \nabla_{\mathbf{W}} MSE \tag{9}$$

where α is the step factor.
The decision rule for the model is given by

$$x \in \omega_j \Leftrightarrow g_j(x) = \max_i \, g_i(x) \tag{10}$$

This states that the class assigned to the input object[3], x, is the class whose discriminant function yields the highest value.

2.2 The Template Based Classifier

A template based classifier matches an input x with one or several references, which is also referred to as templates [5]. These templates represent objects in a dataset where information about their respective class is either available or estimated. The simplest approach is to use a labeled dataset, often called a training dataset, as templates.
A variant of the template based classifier is using the nearest neighbors as the decision rule. The NN-based classifier finds the closest reference to the input x and categorizes x as the same class as the reference[5]. An example of NN is showed figure 1. Euclidean distance is a way of computing the distance between two objects in the feature space. The formula is given by

$$d(\mathbf{x}, \mu) = (\mathbf{x} - \mu)^{\top} (\mathbf{x} - \mu) \tag{11}$$

where $\mathbf{x}$ is the input and μ are the templates[5].
Another variant of the template based classifier is the KNN. In contrast to the NN-based classifier, the KNN is considering $K > 1$ nearest references in the feature space. The majority class within this K is the class assigned to the input x[5]. An example of KNN is showed in figure 2.
In order for both variants to find the closest neighbors of the input x, the calculation of the distance has to be done for all the data in the training set. The process is repeated for every new input x. As expected, this can be a high amount of processing. A solution is discussed in subsection 2.4.

[3]Named "object" to generalize an instance of the class.

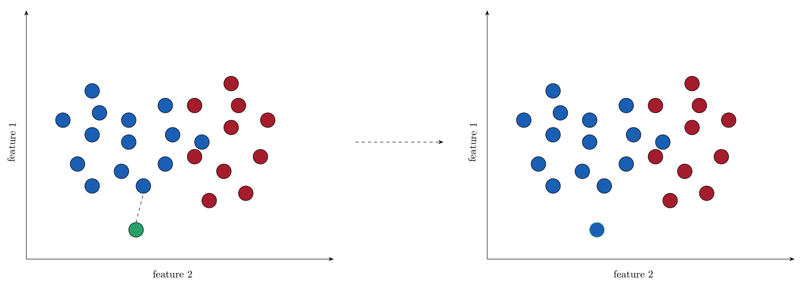

Figure 1: Illustration of $K = 1$, i.e the NN-based classifier. Input x is marked in green, while two different classes are marked in blue and red. x is assumed to be part of the class marked in blue, because the nearest reference to x belongs to this class. The dashed line symbolizes the distance between the two objects.

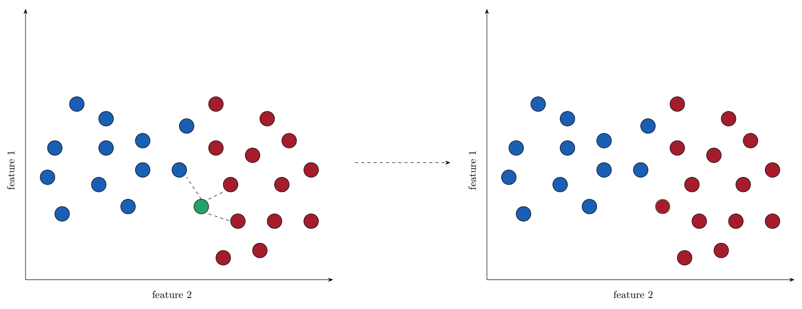

Figure 2: Illustration of $K = 3$, i.e the KNN-based classifier. The set of the three closest neighbors are found, which consists of one from class blue and two from class red. Since class red is the majority, x is assumed to be part of class red.

2.3 Overfitting and Underfitting

The goal when training a model is to make accurate predictions on unseen data. The training dataset is used to find fitted values for the parameters of the classifier [8]. In the context of the linear discriminant classifier discussed in subsection 2.1, the weights and the biases are examples of such parameters.
Overfitting is a result of when the model is trained for too many iterations over the same dataset, or the model is too complex. The model starts to learn the nonessential details in the training dataset [4], and a telltale sign is when the error rate[4], given in equation 12, for the training and test dataset start to deviate.

$$ERR_T = \frac{\#\text{Misclassified Objects}}{\#\text{All Objects}} \cdot 100\% \tag{12}$$

Underfitting occurs when the model is trained with an insufficient amount of iterations, or the model is too simple. The model fails to capture the important trends in the training dataset.
Both overfitting and underfitting result in a model that fails to generalize to unseen data [8]. This is captured in figure 3.

2.4 Clustering

Similar objects tend to be closer in the feature space, which is the foundation of clustering. Hence, the centers of a predefined amount of clusters can be calculated and further used as references for the template based classifiers discussed in subsection 2.2. This will reduce the amount of processing significantly, while still maintaining good coverage [5]. This clustering method is called K-means.

[4]The error rate is a metric to measure the performance of a model.

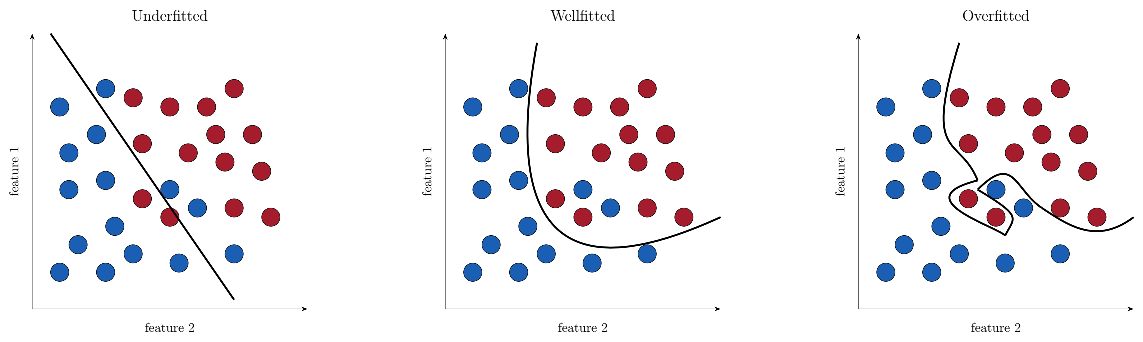

Figure 3: Illustration of underfitting, wellfitting and overfitting [2].

K-means is an unsupervised classification method, meaning it does not require labels. In K-means, the following expression is minimized

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{n,k} ||x_n - \mu_k||^2 \tag{13}$$

where N is the amount of objects in the training dataset, K is the amount of clusters, μ_k is the center of the k'th cluster and x_n is an object in the training dataset.

$r_{n,k}$ is an indicator given by

$$r_{n,j} = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases} \tag{14}$$

which is active if x_n belongs to the k'th cluster [7].
To minimize equation (13), the Euclidean distance between each object x_n and belonging center μ_k must be as small as possible. K-means is an iterative algorithm. At initialisation, the centers of the clusters are chosen arbitrarily, shown in figure 4 part 1. The following steps are repeated:

- Identify the closest center and set the corresponding indicators for each sample x_n, which is shown in figure 4 part 2 and 4 [7].
- Compute the new centroid[5] of each cluster and update the center μ_k with the newly calculated centroid accordingly [7]. See figure 4 part 3 and 5.

2.5 Confusion Matrix

A confusion matrix is a table used in classification tasks to visualize the performance of an classifier by showing how classes overlap or are distinct within the feature space [5]. An example of a confusion matrix is shown in figure 10.

[5]"The centroid of a set of points is the mean point position, that is, the sum of all point coordinates divided by the number of points" [6].

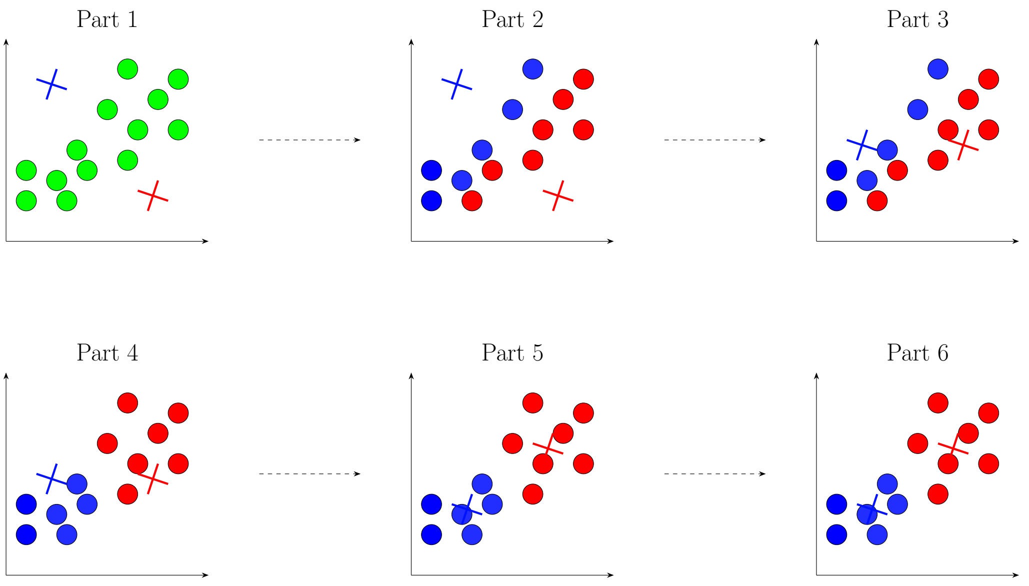

Figure 4: Illustration of K-means with two clusters. The algorithm stops when it converges, i.e the change in center of each cluster does not fulfill the minimum change [3]. This is shown in part 6.

3 The Task

This section will give a brief introduction to the classification tasks.

The Iris Task: The Iris classification task involves developing a linear discriminant classifier to distinguish between three variants of the Iris flower: Setosa, Versicolor, and Virginica. The task includes experiments aimed at optimizing performance, such as the removal of overlapping features. These classifications are based on the dimensions of the flower's sepals and petals.

Handwritten Numbers Task: The handwritten numbers classification task involved the development of two main strategies using a nearest neighbor approach with the MNIST dataset. This dataset consists of 60,000 training and 10,000 testing examples of handwritten digits ranging from 0 to 9, with each image being 28x28 pixels in size and preprocessed for centering and scaling. Some examples of the dataset are shown in figure 5, and the class distribution is shown in figure 15.

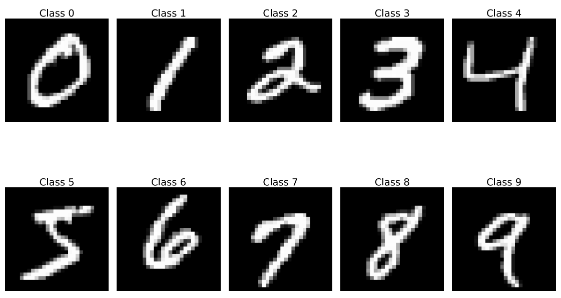

Figure 5: Examples of the 10 classes in the MNIST dataset.

4 Implementation and Results

Both the implementation of the Iris Task, and Handwritten Numbers Task can be found in their respective folders on GitHub [1], and was implemented in Python, utilizing libraries such as NumPy, Matplotlib, Seaborn, and Scikit-learn for data handling, processing, and visualization.

4.1 The Iris Task

Visualization of Data Characteristics: The dataset used for this classification consists of 150 samples, divided equally among the three Iris species. Each sample contains four features: sepal length, sepal width, petal length, and petal width. The feature space is shown in figure 6. Examining this, it can be visually confirmed that the dataset is near-linear separable, which confirms the use of a linear classifier.

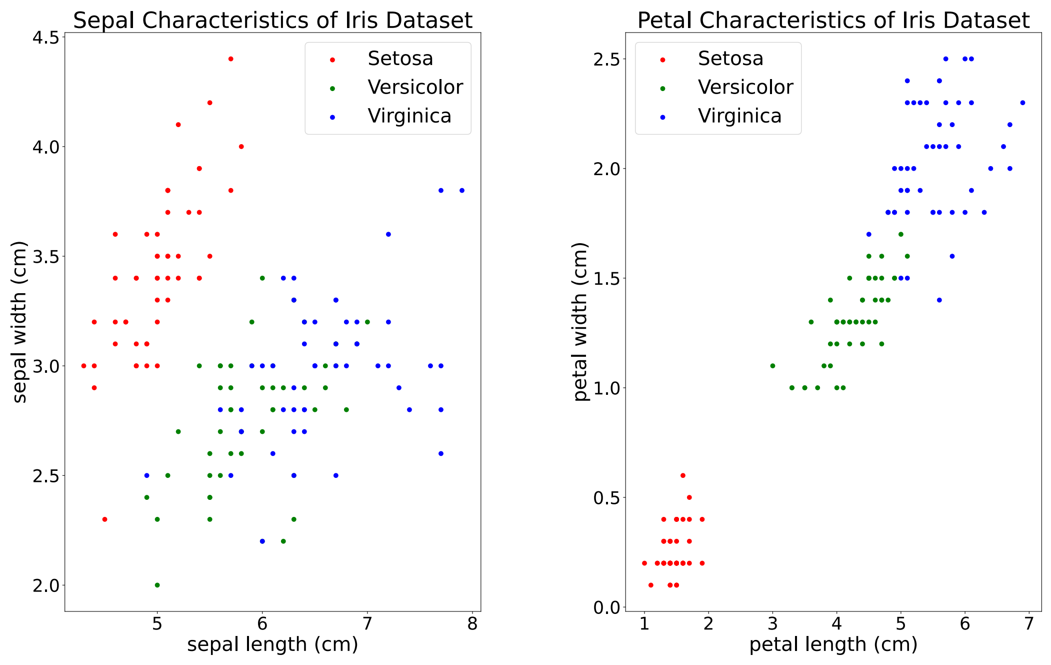

Figure 6: Scatter plot of the feature space of the iris dataset.

Model Development and Evaluation: The linear classification model was trained using MSE, which is defined in equation 4, as the cost function, and sigmoid, which is defined in equation 5, as the final and only activation function. This model architecture is illustrated in figure 9. Various training and testing splits were tested to determine the performance of the classifier:

- First, using the initial 30 samples of each species for training and the remaining 20 for testing.
- Then, the last 30 samples were used for training with the first 20 samples reserved for testing.

Each configuration was evaluated with varying step factors (α) to find the optimal rate that provides the best balance between convergence speed[6] and lowest MSE. It is apparent from figure 7 that $\alpha = 0.01$ has an oscillatory behavior, indicating that the step size is too big, and it is most likely stepping over the local minima. The best suited step size is $\alpha = 0.005$ due to fast convergence, and good performance.

Evaluation Using Error Rates and Confusion Matrices: The model's performance was further evaluated through error rates and confusion matrices. These evaluations were important in understanding the classifier's performance across different training and testing configurations. By using the weights obtained from training the model in the two cases discussed above with the specified step factor, results were achieved as depicted in the confusion matrices in figure 10, and corresponding error rates as shown in figure 8. The first test condition, using the first 30 samples for training, yields better results on unseen data than the second test condition, last 30 samples for training. This is most likely due to more varied data in the 30 first samples, making the model

[6]A convergence is decided if the difference in MSE between two subsequent iterations is smaller than 0.001.

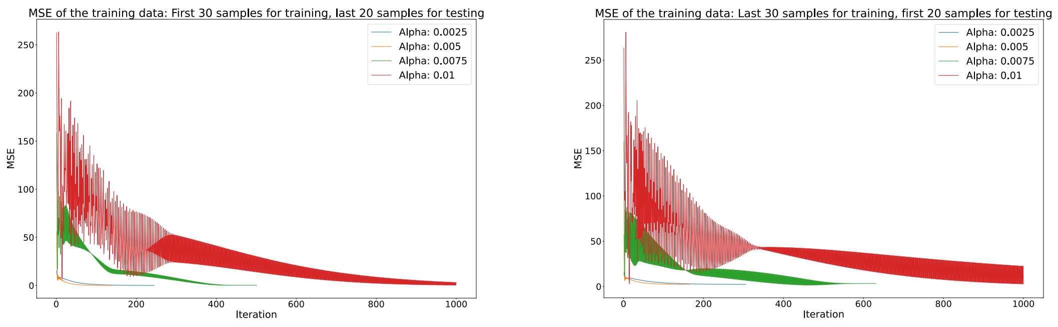

Figure 7: MSE with different step lengths, α.

more generalized. The first 30 samples for training, and the last 20 for testing will therefore be used for the rest of the iris task.

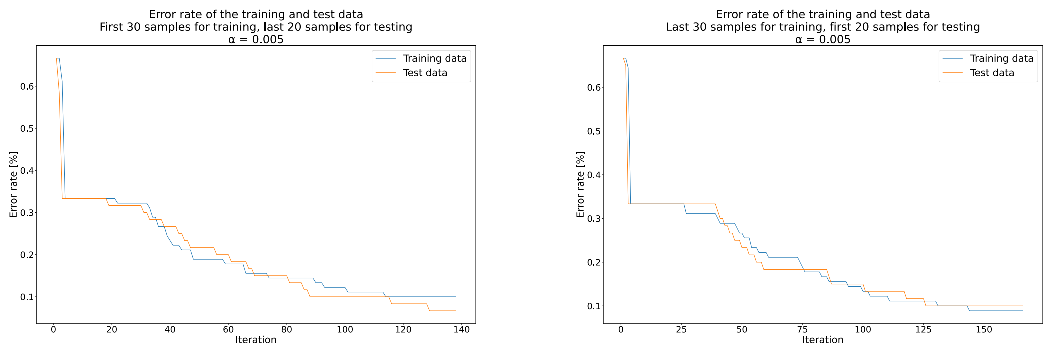

Figure 8: Error rates of different train and test splits.

Feature Analysis: Another interesting result from the confusion matrices in figure 10 is that all misclassifications are between the two species Versicolor and Virginica. This is easy to see when plotting the feature space, as shown in figure 6, and figure 12.

Each of the four features of the iris species Versicolor and Virginica have some degree of overlap. To test the robustness and accuracy of the classifier, the most overlapping features were removed. The confusion matrices are shown in figure 11, and the error rates are shown in table 1. Removing the feature with the most overlap, the sepal width, had little to no effect on the model's performance. Due to the overlap, the sepal width is less prioritized than other features, and in this case the difference is so small that the performance is identical. However, when removing the sepal length as well, which has less overlap, a worsening in the model performance is seen, due to the feature space shrinking. Surprisingly, the model performance increases slightly on unseen data when removing the petal length, but way worse on the training data. This might be due to the petal length feature containing substantial noise or other non-representative patterns in the training dataset.

Removed Feature(s)	Error Rate Test Data (%)	Error Rate Train Data (%)
1. No features removed	10	6.67
2. Sepal width	10	6.67
2. Sepal width, sepal length	13.33	10
3. Sepal width, sepal length, petal length	18.89	8.33

Table 1: Error rates for test data with different test conditions

4.2 Handwritten Numbers Task

Model Design and Training: The template based classifier uses KNN as the decision rule where the distance is Euclidean distance between every pixel on the test image to the same relative pixel on the template image. This concept is shown through figure 2. Two different sets of templates were used to study the runtime and error rate:

- The whole training set, 60,000 images, as templates
- Clustering to reduce the amount of template to 64 per class, for a total of 640 templates. Examples of some clusters are shown in figure 13.

Data processing was executed in batches of maximum 1,000 images for both methods to optimize memory use and computational efficiency. The improvement in runtime can be seen in point 1 and 2 in table 2.

Test Condition	Clustering Runtime (s)	Classification Runtime (s)	Total Runtime (s)
1. K=1, no batching	0	661	661
2. K=1, with batching	0	640	640
3. K=7, with batching	0	646	646
4. K=1, with batching and clustering	56	1.56	57.76
5. K=7, with batching and clustering	56	1.91	57.91

Table 2: Runtime comparison[7] for different test conditions, with the entire training and test dataset, and a batch size of 1000.

Evaluation of Performance: Evaluation of the classifier was done with the same approach for as the the iris task, through error rates and confusion matrices. However it is also relevant to discuss the runtime as they are drastically different.

The error rates for the different test conditions are found in table 3. To get further insight of the performance, the confusion matrices for the two best models, and their respective test condition are plotted in figure 14. Even though clustering leads to a slightly worse error rate, the classification runtime is drastically reduced as seen in table 2. Furthermore, some of the correct, and incorrect classifications are plotted in figure 16.

Test Condition	Error Rate (%)	Correctly Classified	Misclassified
1. K=1, with batching	3.09	9691	309
2. K=7, with batching	3.06	9694	306
2. K=1, with batching and clustering	4.61	9539	461
3. K=7, with batching and clustering	6.49	9351	649

Table 3: Error rates for test data with different test conditions

5 Conclusion

5.1 The Iris Task

The experiments confirmed the efficiency of the linear classifier in handling the near-linearly separable Iris dataset, emphasizing the importance of feature selection and parameter optimization in enhancing model performance.

[7]Tested on CPU: 11th Gen Intel i7-1165G7 (8) @ 4.700GHz

5.2 Handwritten Numbers Task

The analysis of the Handwritten Numbers Task demonstrated that while the full template method offers slightly better accuracy, the clustered template approach provides a more practical solution for applications needing faster processing times. This task highlights the importance of considering application-specific requirements when choosing classifiers, especially in resource-constrained environments. The exploration also highlighted the robustness of simple algorithms like KNN in handling complex, high-dimensional datasets such as handwritten digits.

References

[1] Slåkvik, Anders Aatif, Nicolai A. *TTT4275 Classification Project.* URL: https://github.com/NicolaiAdil/ttt4275-classification_project (visited on 26th Apr. 2024).

[2] GeeksForGeeks. *ML — Underfitting and Overfitting.* URL: https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/ (visited on 29th Apr. 2024).

[3] IBM. *K-means cluster analysis: Convergence criteria.* 2021. URL: https://www.ibm.com/docs/en/spss-statistics/beta?topic=analysis-k-means-cluster-convergence-criteria (visited on 27th Apr. 2024).

[4] IBM. *What is overfitting?* URL: https://www.ibm.com/topics/overfitting (visited on 29th Apr. 2024).

[5] Magne H. Johnsen. *Classification.* NTNU, 2017.

[6] Eckersly, Robert J. King, Andrew P. *Statistics for Biomedical Engineers and Scientists.* 2019, pp. 2017–228.

[7] Pierluigi Salvo Rossi. *Estimation, Detection and Classification , Lecture 14-15.* PowerPoint presentation. Slide 54-58.

[8] Pierluigi Salvo Rossi. *Estimation, Detection and Classification , Lecture 18.* PowerPoint presentation. Slide 8-10.

A Iris Confusion Matrices and Figures

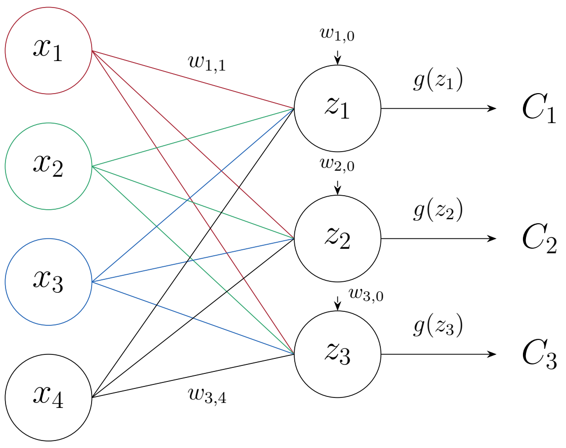

Figure 9: Architecture of the linear classifier for the iris dataset, where x_i are the features, $w_{j,i}$ are the weights, $g(\cdot)$ is the activation function, and C_j are the classes.

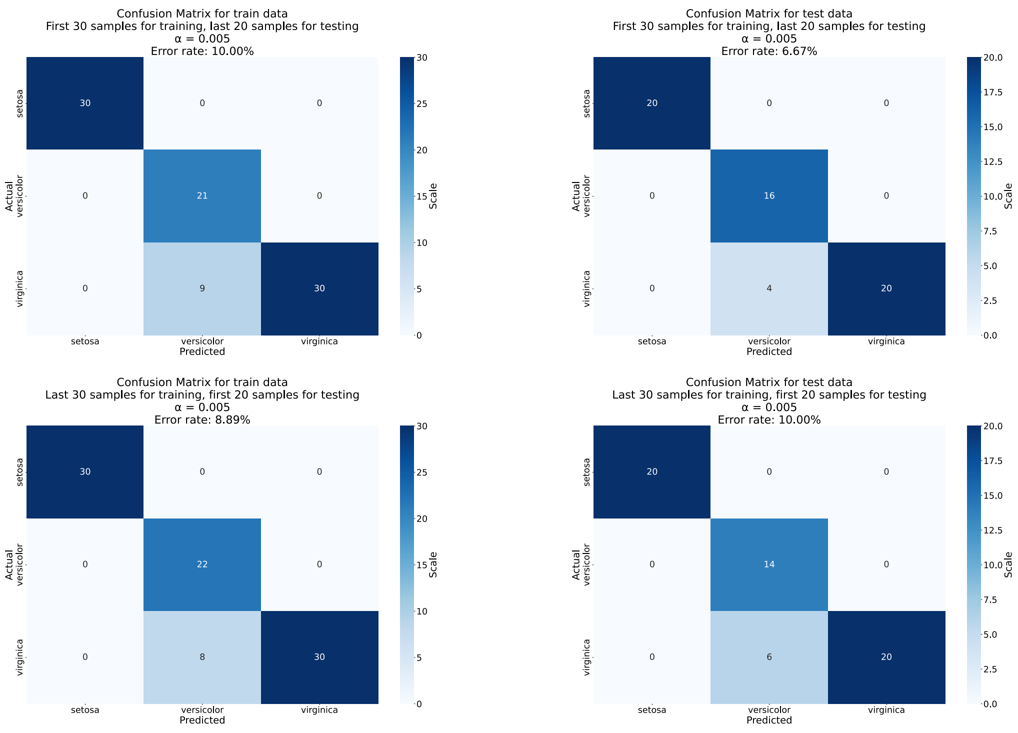

Figure 10: Confusion matrices of different train and test splits for the iris dataset.

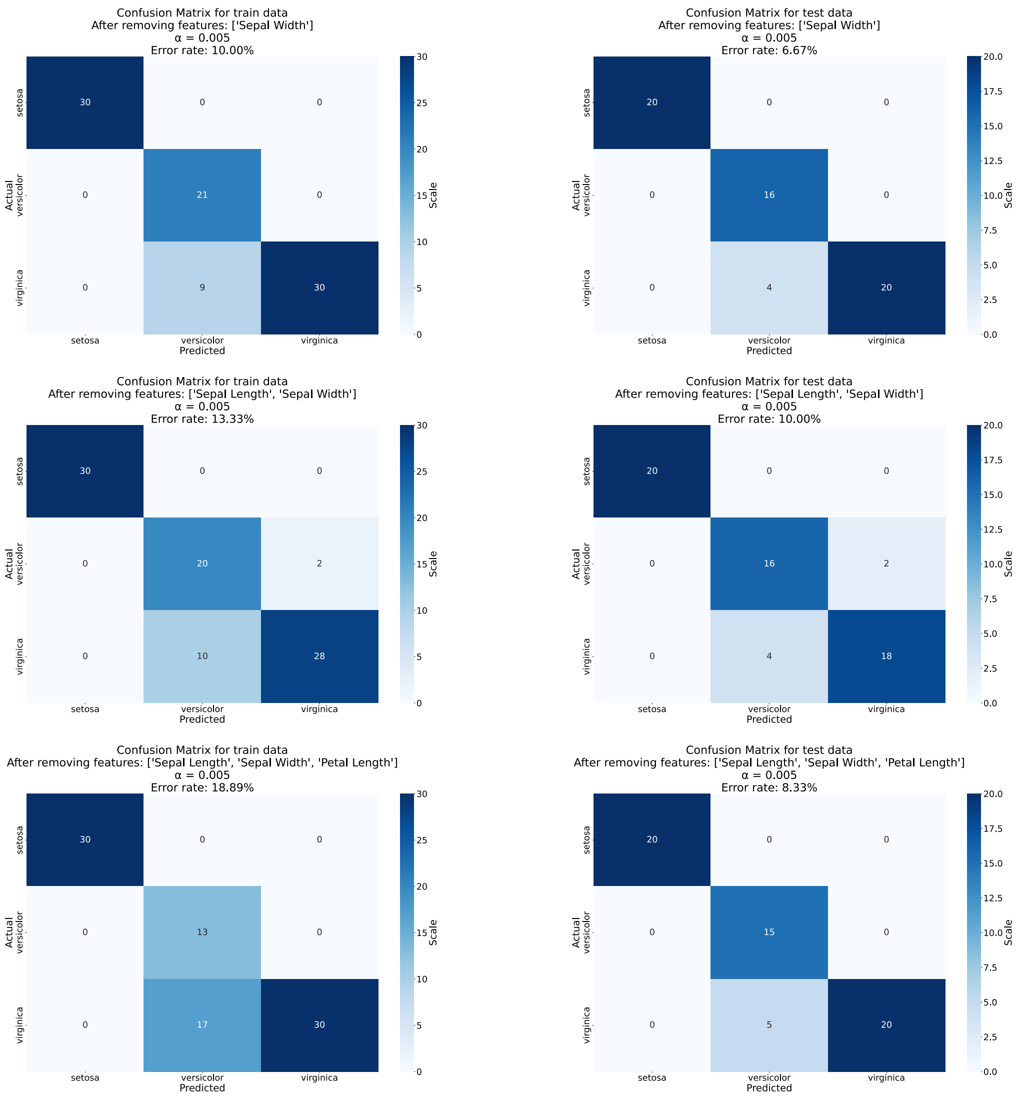

Figure 11: Confusion matrices after removing most overlapping features for the iris dataset.

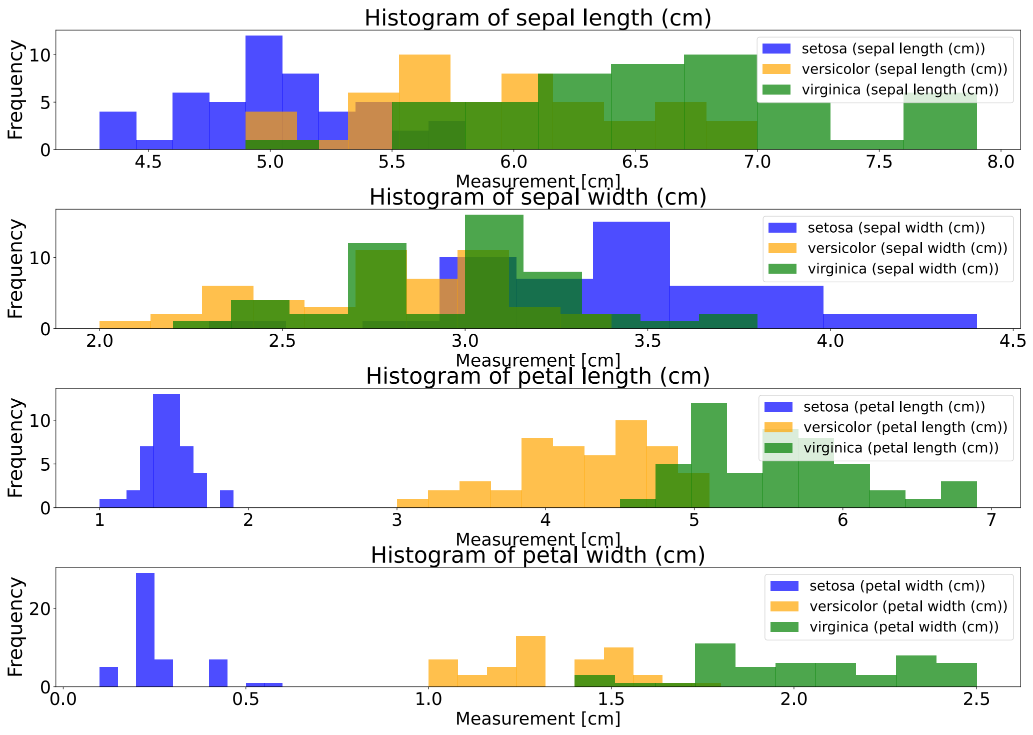

Figure 12: Histogram of the feature space of the iris dataset.

B MNIST Confusion Matrices and Figures

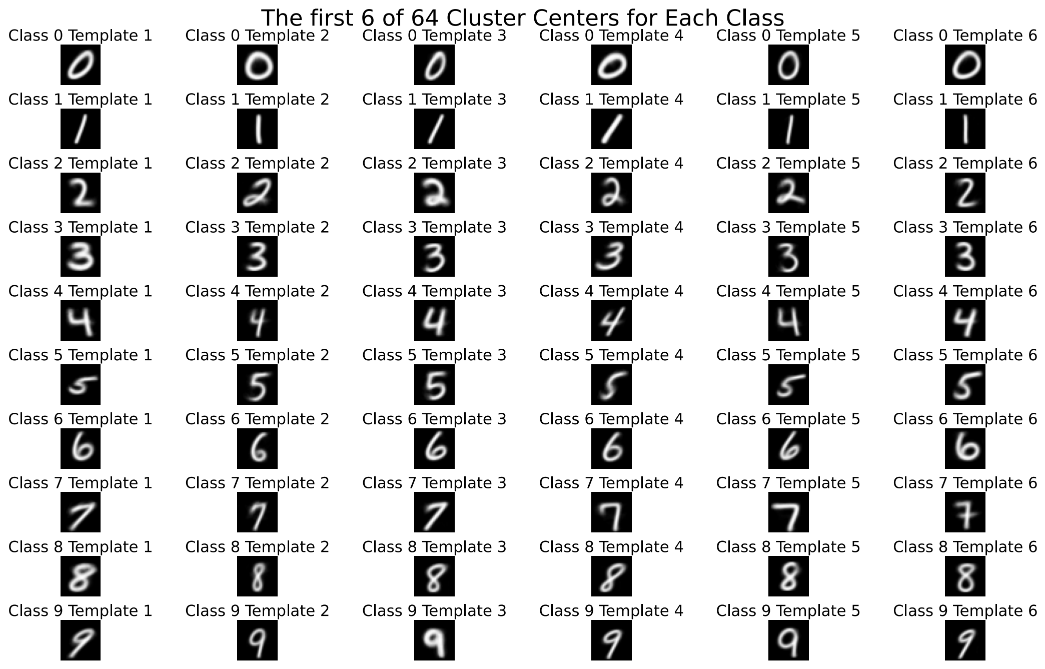

Figure 13: The first 6 of 64 cluster centers for each class.

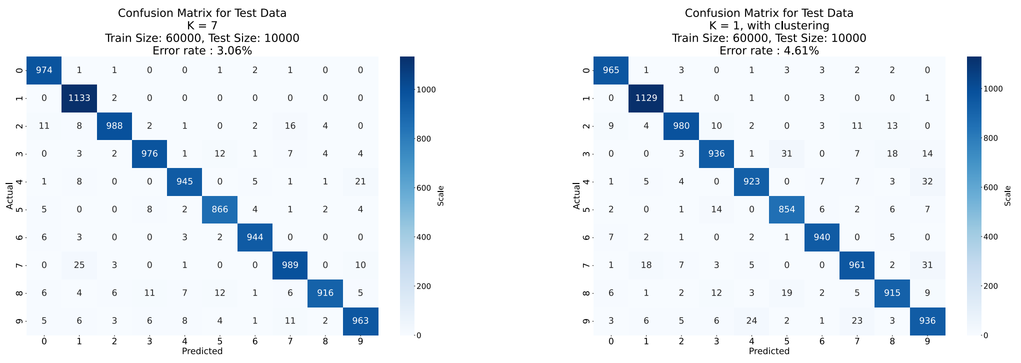

Figure 14: Confusion matrices of the best performing classifier with and without clustering.

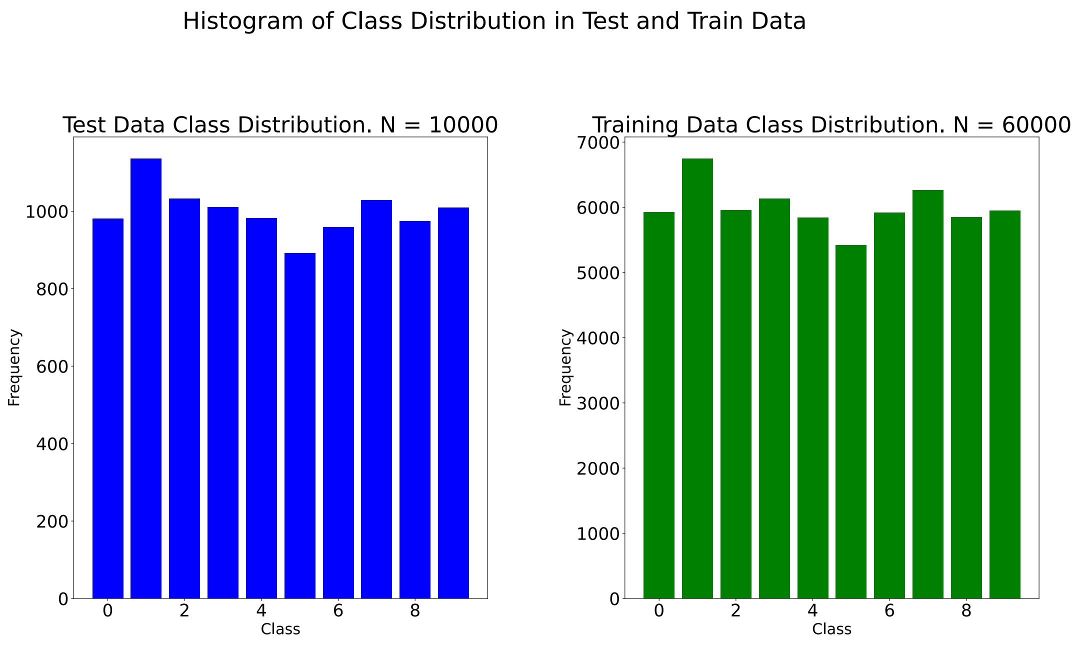

Figure 15: Class distribution of the MNIST dataset.

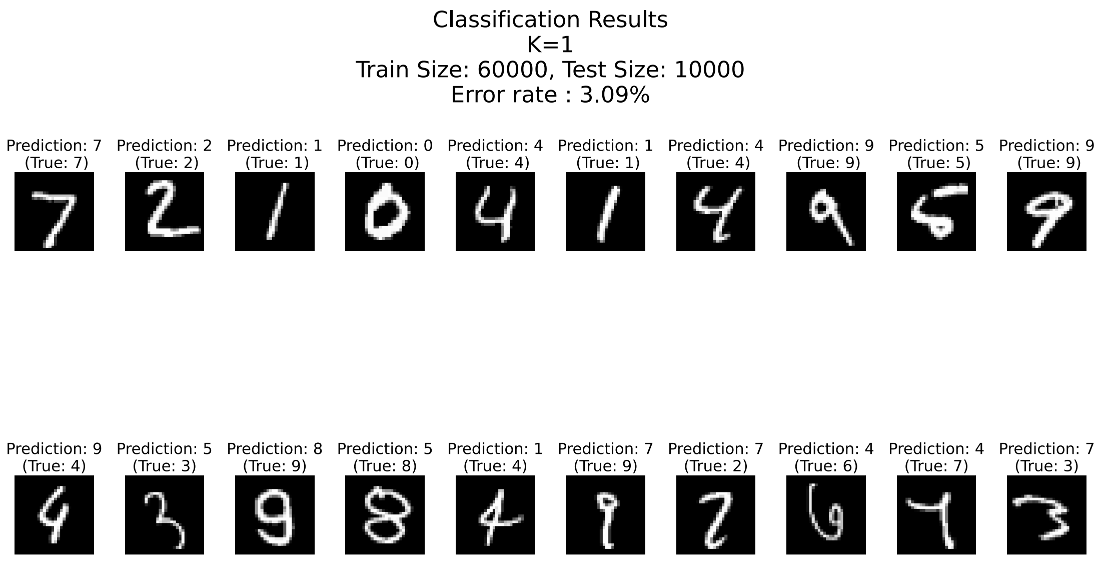

Figure 16: Some correctly and incorrectly classified images, no clustering.