Department of Ingegneria Informatica, Automatica e Gestionale

Optimization Methods for Machine Learning

Professor:
L. Palagi

Students:
Francesco Sanetti
1933718
Francesco Ventura
1841504

Academic Year 2024/2025

Contents

0 Introduction

The project involves implementing and optimizing **Support Vector Machines** procedures (SVMs) to **classify handwritten digits** from the MNIST database (see Figure 1 in the plots), specifically distinguishing between the digits '1' and '5' for Q1,Q2 and Q3 whereas for Q4 considering also the digits '7', all the images have been converted and mapped as row vectors (1,784).
The 80% of the images was used as a **training set** while the remaining 20% as **testing** (random selection performed using *train_test_split* form SKlearn) set to evaluate the performances of the model.
For non-linearly separable data, soft-margin SVMs allow some classification errors, balancing margin maximization with error minimization. This approach is formulated as a **convex quadratic** programming problem, for which strong duality conditions apply, this means that the primal problem and its dual share the same optimal value function. **Optimizing the dual** problem often simplifies computations, especially when using the **kernel trick**, as it involves only the dot products between input vectors. Below the addressed dual optimization problem:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^p} \quad & \frac{1}{2}\alpha^T Q \alpha - e^T \alpha \\ \text{s.t.} \quad & \alpha^T y = 0 \\ & 0 \leq \alpha \leq C \end{aligned}$$

It is important to note that all variables have been initially **scaled** to improve the overall optimization performances using the **StandarScaler** function form Sklearn library.

1 Q1: Soft Support Vector Machine (SVM)

As stated in the introduction, the optimization routine is based on the solution of a **quadratic programming problem** thus it is possible to use the pre-built **CVXOPT solver library** (solvers.qp). The algorithm was implemented to allow also for an **easy switching** between polynomial and Gaussian kernel by defining a variable "kernel type" which can be either the string 'Polynomial' or 'Gaussian' in order to immediately switch the kernel inside the decision function, this design enabled performance comparisons across the two **different kernel types** (further discussed in the grid search section). For the polynomial kernel, to allow the execution of the grid search even on the set of parameters resulting in **zero free support vectors**, the approach proposed by Chang [1] has been implemented: if there is no α_i such that $0 < \alpha_i < C$, then r_1 satisfies

$$\max_{\alpha_i = C, y_i = 1} \nabla_i f(\alpha) \leq r_1 \leq \min_{\alpha_i = 0, y_i = 1} \nabla_i f(\alpha)$$

and it is selected as the middle point of this range, the same is done for r_2 (by considering $y_i = -1$) and b is then set as the mean between the computed values.
The initial value of α in this case is automatically set by the solver and it is different from $\alpha_0 = 0$. The tolerance of the model was set to 1×10^{-5} since higher values were associated with significantly higher KKT violations (m-M). The CVXOPT solver was set with a tolerance of 1×10^{-10}.

1.1 Hyperparameter, grid search and performances

The hyperparameters of the model are the upper bound of alphas (C), and the degree of the polynomial function or the coefficient of the degree of the Gaussian function (γ). The selection of hyperparameters was conducted through a **heuristic approach**: a grid search algorithm was implemented to determine the combination of hyperparameters that minimizes the **validation accuracy score**, assessed using a **cross-validation** procedure (with k set to 5). The grid search was performed using both a polynomial and a Gaussian kernel over the following sets:

- C using a logarithmic scale: from 10^{-4} to 10 in powers of 10
- γ for the polynomial kernel: all the integers from 1 to 21
- γ for the Gaussian kernel using a logarithmic scale: from 10^{-4} to 10 in powers of 10

The result using the best hyperparameters are displayed above:

kernel type	C	γ	validation accuracy	test accuracy
Gaussian	10	0.0001	0.990625	0.9825
Polynomial	0.0001	2	0.994375	0.99

Table 1: Hyperparameters perfomances for Q1

As shown the polynomial one tends to **perform better** than the Gaussian over the considered search intervals, leading to its selection for the implementation in Q2,Q3 and Q4. It is worth noting that the parameter C does not affect the validation accuracy of the polynomial kernel, as demonstrated in Figure 13. Consequently, the selection of C was made by considering **the number of iterations** associated with each value, with the chosen C being the one that minimizes the number of iterations. The plots of training and validation accuracy with respect to different hyperparameters, where the fixed hyperparameter corresponds to the optimal ones are presented in the section 6.2. The occurrence of **over-fitting** can be observed for higher values of gamma in both the polynomial ($\gamma > 5$) and Gaussian one ($\gamma > 10^{-2}$) while **under-fitting** can be observed in the Gaussian for lower values of C ($C < 10^{-1}$). The final performances obtained under the chosen hyperparameters are:

- Training accuracy: 100%
- Testing accuracy: 99%
- Final value of the dual objective function: -0.000629
- KKT violation: $7.5 \times 10^{-10} \approx 0$
- Number of solver.qp iterations: 14
- Time used for optimize: 1.63s

Since the number of function evaluations is NOT provided by the CVXOPT solver, and following the tutor's instructions, this metric will not be computed for ALL the different algorithms.

2 Q2: Soft SVM$^{\text{light}}$ decomposition method

For the second task, as requested, using the same kernel function and hyperparameters chosen in Q1, a decomposition method for the Support Vector Machine (SVM) was implemented: when the complexity of an optimization system arises from the high number of variables (which may be so large to become practically intractable), such systems can be approached through **block decomposition methods** that instead of directly solving the entire problem address subproblems of significantly smaller dimensions, optimizing only with respect to a **subset of variables** and in this way, these subproblems can sometimes even be solved **analytically** as in the case of q=2 in Q3. The subproblems solved at each iteration of the SVM$^{\text{light}}$ is still a convex quadratic problem (therefore it was solved using CVX.OPT library) and it's the following

$$\begin{aligned} \alpha_W^* = \arg\min_{\alpha_W} \ & f(\alpha_W, \alpha_{\bar W}^k) \\ \text{s.t.} \quad & y_W^T \alpha_W = -y_{\bar W}^T \alpha_{\bar W}^k \\ & 0 \le \alpha_W \le C. \end{aligned}$$

The initial value of *alpha* is set to $\alpha_0 = 0$ resulting in $\nabla f(\alpha_0) = -e$ and $f(\alpha_0) = 0$. The main stopping criterion for the algorithm is derived from the **KKT conditions** as $m(\alpha_k) - M(\alpha_k) \le \epsilon = 1 \times 10^{-10}$ but also a maximum iteration safeguard RULE was included with max_iter=1000. Below the pseudo code of the implemented algorithm:

```
**Data.** A feasible point $\alpha^0 = 0$.
**Initialization.** Set $k = 0$, $\nabla f(\alpha^0) = -e$.
**While** $\big((m(\alpha^k) - M(\alpha^k) > \epsilon) \wedge (k < \text{max\_iter})\big)$
  ① select the working set $W^k$ using the SVM$^{\text{light}}$ selection rule;
  ② set $W = W^k$ and compute a solution $\alpha_W^*$ applying solvers.qp to the subproblem;
  ③ set $\alpha_i^{k+1} = \begin{cases} \alpha_i^* & \text{for } i \in W \\ \alpha_i^k & \text{otherwise;} \end{cases}$
  ④ update $\nabla f(\alpha^{k+1}) = \nabla f(\alpha^k) + \sum_{i \in W^k} Q_i \left(\alpha_i^{k+1} - \alpha_i^k\right)$;
  ⑤ set $k = k + 1$.
**end while**
**Return** $\alpha^* = \alpha^k$
```

The solver was set with a tolerance of 1×10^{-10} (which is different from the tollerance of the KKT condition violation) and the chosen value for the **cardinality of the working set** was $q = 6$, this selection was made after testing several even values of q ($[4, 6, 8, 10, 20, 40, 60]$) with the final choice corresponding to the value that resulted in the **highest test accuracy** with a **low running time**.

2.1 Q2 Performances

The final results are shown below:

- Training accuracy: 100%
- Testing accuracy: 99.5%
- Initial value of the dual objective function: 0
- Final value of the dual objective function: -0.000562
- KKT violation: $8.0 \times 10^{-11} \approx 0$
- Number of external iterations: 183
- Time used for optimize: 1.41s

Note that the number of iterations considered is the one of the decomposition cycle and not the iteration of the CVXOPT solver.

3 Q3: Soft SMO-MVP decomposition method

For the third task, the implementation of an **SMO algorithm** was requested, specifically utilizing the **Most Violating Pair** (MVP) strategy, which guarantees convergence of SMO decomposition method, and using the same kernel (Polynomial) and hyperparameters chosen in Q1. The SMO-MVP algorithm updates the gradient by computing the **largest feasible descent direction** at each iteration and determines the optimal step size through an **exact line search** procedure (no pre-built solver was used), the descent direction and the optimal stepsize are analitically computed as the following

$$\text{Select} \quad i \in I(\alpha_k) \text{ and } j \in J(\alpha_k), \text{ and set the descent direction d as:}$$

$$d_h^k = \begin{cases} y_i & \text{if } h = i, \quad i \in I(\alpha_k) \\ -y_j & \text{if } h = j, \quad j \in J(\alpha_k) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Find the optimal stepsize } t^k : \quad t_{\text{feas}}^{\max} = \min \left\{ \min_{i:d_i^k > 0} \frac{C - \alpha_i^k}{d_i^k}, \min_{i:d_i^k < 0} \frac{\alpha_i^k}{|d_i^k|} \right\}$$

To enhance the computational efficiency of SMO-MVP algorithm, a **caching system** was introduced: during each iteration, the algorithm checks if the columns of Q associated with the most violating pairs have already been computed (i.e. if they are in the cache) or else these columns are computed directly, avoiding the need to compute the **full Q matrix** at every iteration. Also in this case, the initial value of α is set to $\alpha_0 = 0$.

3.1 Q3 Performances

The final results are shown below:

- Training accuracy: 100%
- Testing accuracy: 99.25%
- Initial value of the dual objective function: 0
- Final value of the dual objective function: -0.000562
- KKT violation: $9.9 \times 10^{-11} \approx 0$
- Number of iterations: 613
- Time used for optimize: 0.36s

3.2 Performances comparison

All proposed algorithms achieve **high test accuracy** scores (≥ 0.9) while maintaining a training accuracy always of 100%; however, the SVM$^{\text{light}}$ algorithm demonstrates a slightly superior performance, yielding the highest test accuracy. In terms of computational efficiency, as anticipated, the **decomposition method** results in reduced execution times for Q2 and Q3 respect to Q1.
Among the evaluated algorithms, **SMO-MVP** exhibits the lowest computational time despite requiring a significantly higher number of iterations (thanks also to the implemented **caching mechanism**). It can also be noted that, since the SVM soft dual problem is (not strictly) convex, the optimal value function is expected to be **unique**: the optimal value of the objective function should remain consistent across the different optimal points. As observed, the difference between the final values of the objective function of different algorithms is minimal ($< 10^{-4}$).

4 Q4: SVM for multi-class classification

Due to the homogeneous size distribution across different classes and only 3 classes to manage, a **one-against-one** (OAO) approach was implemented to address the multi-class classification problem, using the **SMO-MVP** utilized in Q3 due to the smallest time to optimize respect to the other procedures but still excellent performance that helps to **efficiently** solve several problems (3) instead of only one.

The algorithm is structured around **two primary functions**:

- **OAO Optimization:** This function, for each possible pair of classes [(1,5);(1,7);(5,7)] solves the respective dual soft quadratic problem using the SMO-MVP approach and returns a dictionary containing all classes pairs associated with the corresponding optimal values of α.
- **OAO Classification:** This function implements a **voting** procedure as follows: a matrix $fclass$ of size ($P_{test} \times 3$) is defined, for each pair of classes (m, h), the classification (f_h) is computed, and the corresponding columns are updated according to the following rules:

$$fclass_m = fclass_m + 0.5(-f_h + 1)$$

$$fclass_h = fclass_h + 0.5(f_h + 1)$$

  Finally, each test sample is assigned to the class in which it appears most frequently by determining the arg max (column index) of the respective row in $fclass$. This process generates the prediction vector containing the final class assignments.

4.1 Q4 Performances

The final results are shown below:

- Training accuracy: 100%
- Testing accuracy: 99.34%
- Initial value of the dual objective function for each pair: 0
- Final value of the dual objective function for each pair: (1,5): -0.000463; (1,7): -0.000830; (5,7): -0.000481
- KKT violation for each pair: (1,5): 9.2×10^{-11}; (1,7): 9.8×10^{-11}; (5,7): 9.6×10^{-11}
- Number of total iterations: 1600
- Time used for optimize: 1.21s

5 Summary Table

	Hyperparameters (C, γ)	**q**	**Training accuracy**	**Test accuracy**	**Number of iterations**	**KKT violations**	**CPU time**
Q1	0.0001, 2	#	100 %	99 %	14	7.5×10^{-10}	1.63s
Q2	As in Q1	6	100 %	99.5 %	183	8.0×10^{-11}	1.41s
Q3	As in Q1	2	100 %	99.25 %	613	9.9×10^{-11}	0.36s
Q4	As in Q1	2	100 %	99.34 %	1600	9.2×10^{-11};9.8×10^{-11}; 9.6×10^{-11}	1.21s

Table 2: Comparison of Output Values for all the exercises

6 Plots

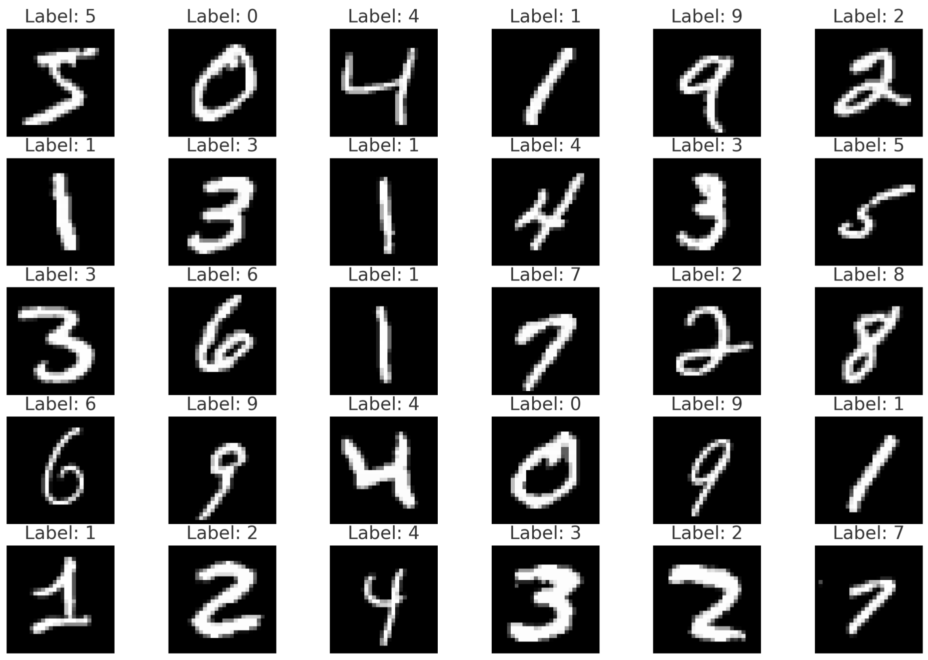

Figure 1: first 30 images with the associated labels

.

6.1 Confusion matrices

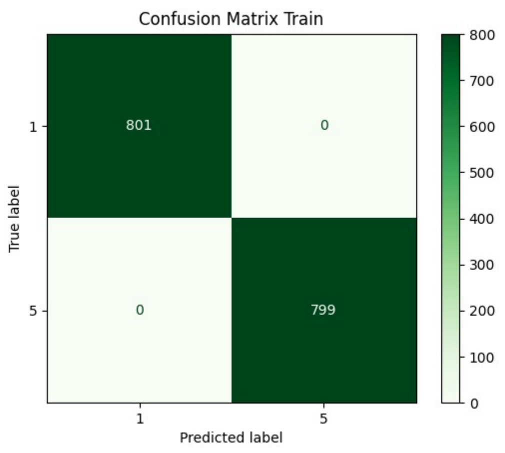

Figure 2: Confusion Matrix Training set Q1

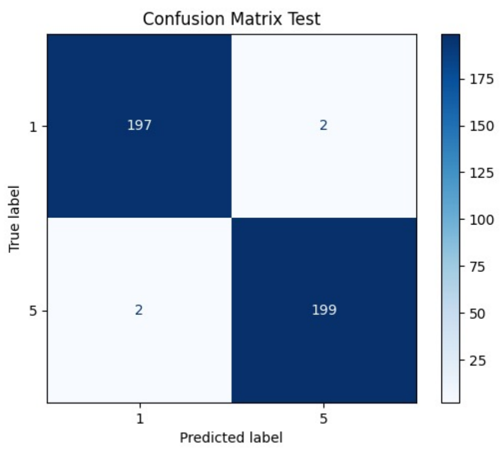

Figure 3: Confusion Matrix Test set Q1

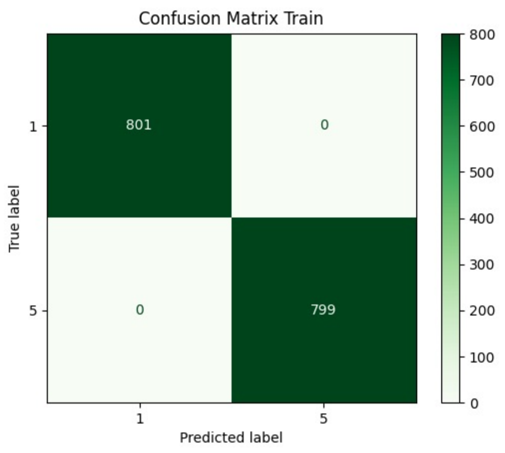

Figure 4: Confusion Matrix Training set Q2

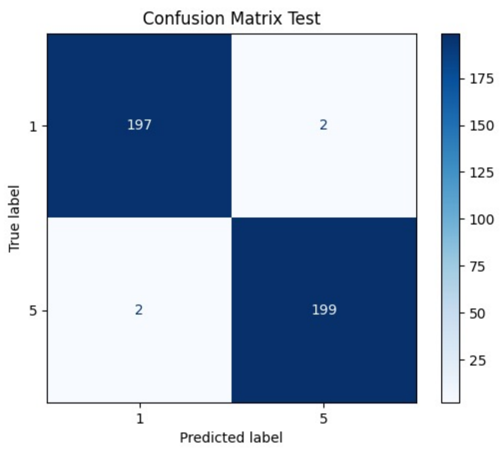

Figure 5: Confusion Matrix Test set Q2

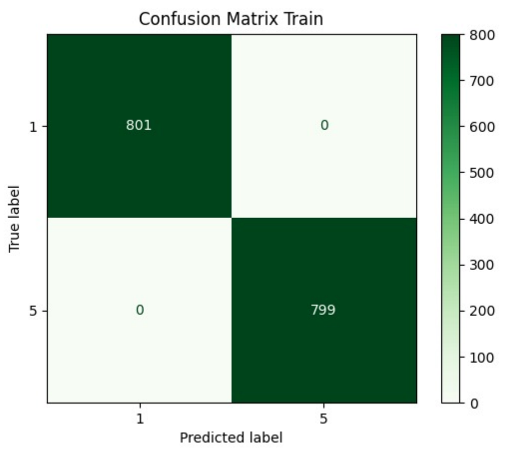

Figure 6: Confusion Matrix Training set Q3

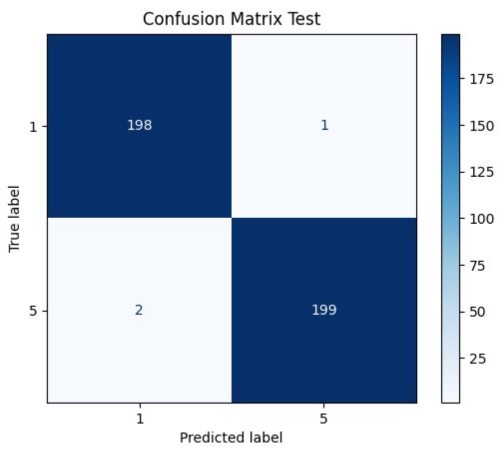

Figure 7: Confusion Matrix Test set Q3

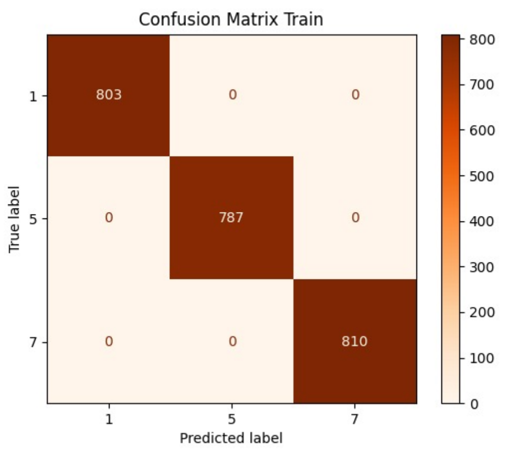

Figure 8: Confusion Matrix Training set Q4

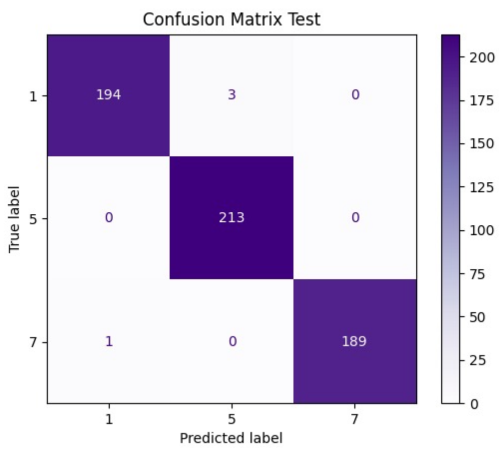

Figure 9: Confusion Matrix Test set Q4

6.2 Hyperparameters plot

Note that all hyperparameters for the Gaussian kernel on the x-axis are displayed in logarithmic scale to produce clearer plots.

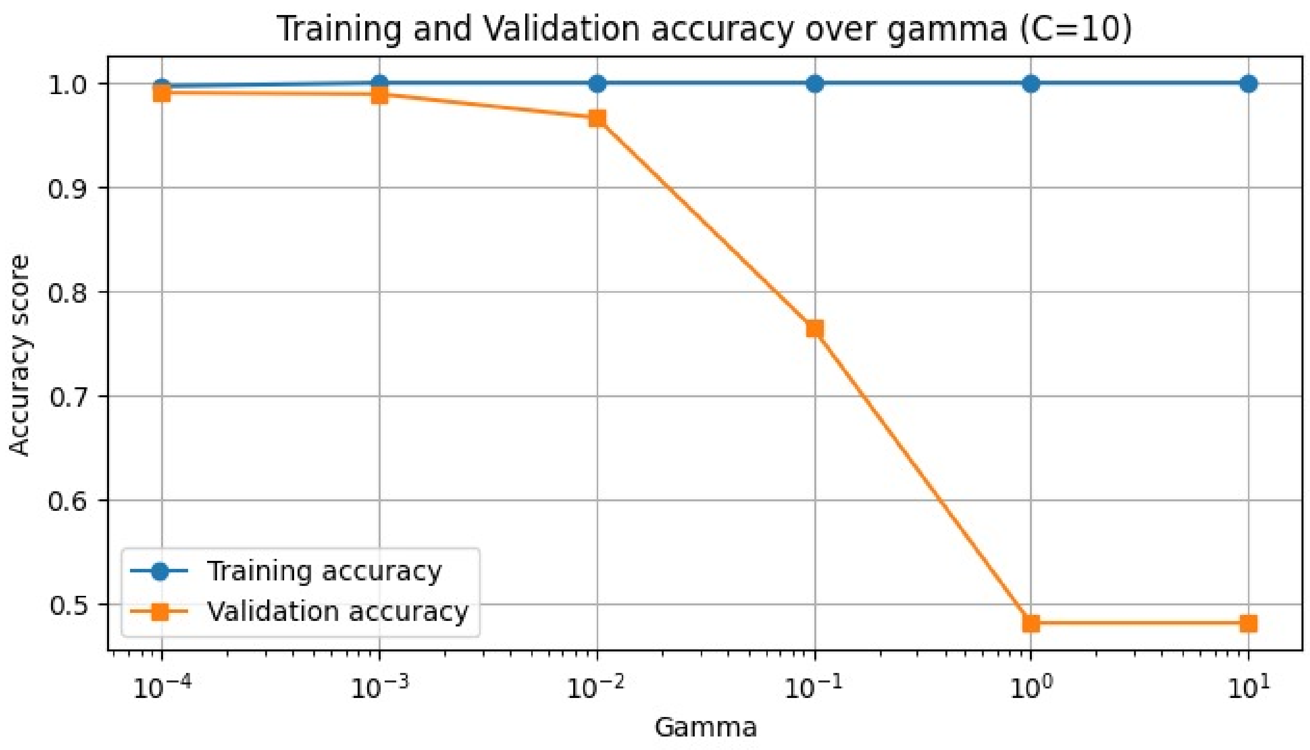

Figure 10: Evolution of the accuracy w.r.t gamma considering a Gaussian kernel

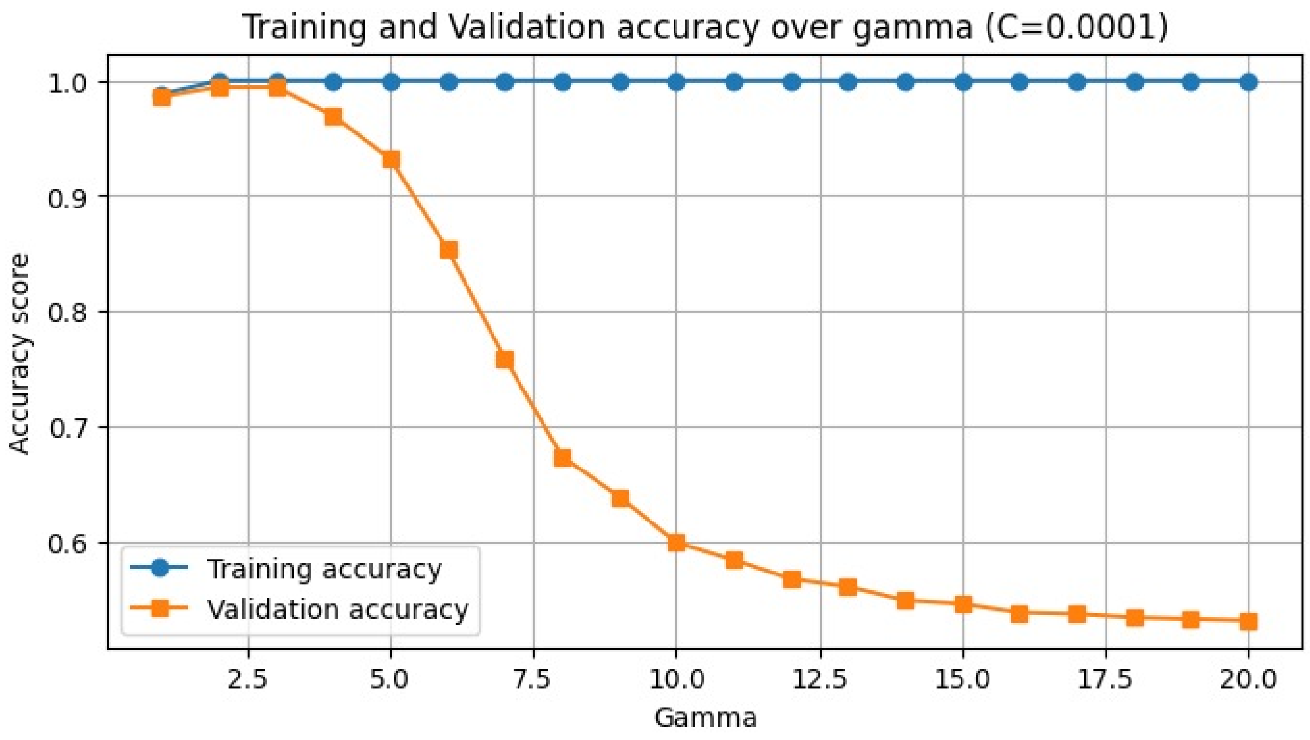

Figure 11: Evolution of the accuracy w.r.t gamma considering a polynomial kernel

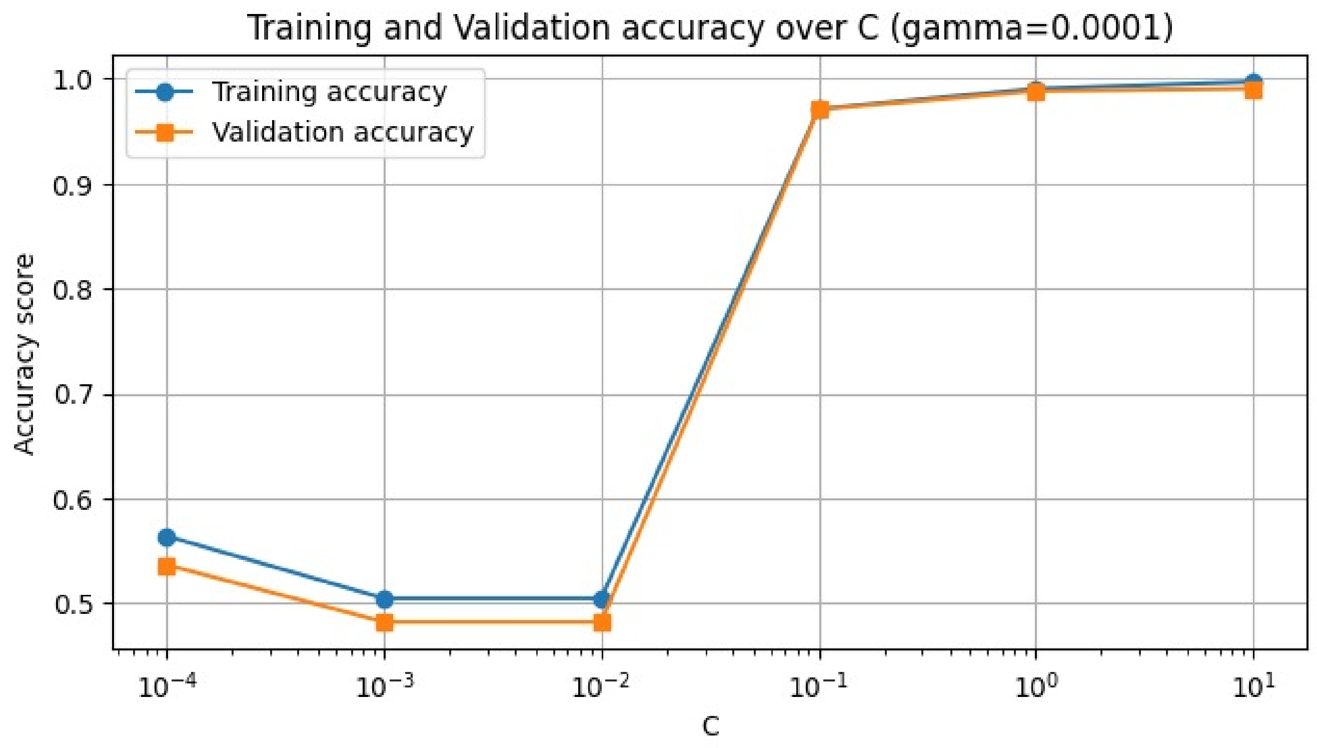

Figure 12: Evolution of the accuracy w.r.t C considering a Gaussian kernel

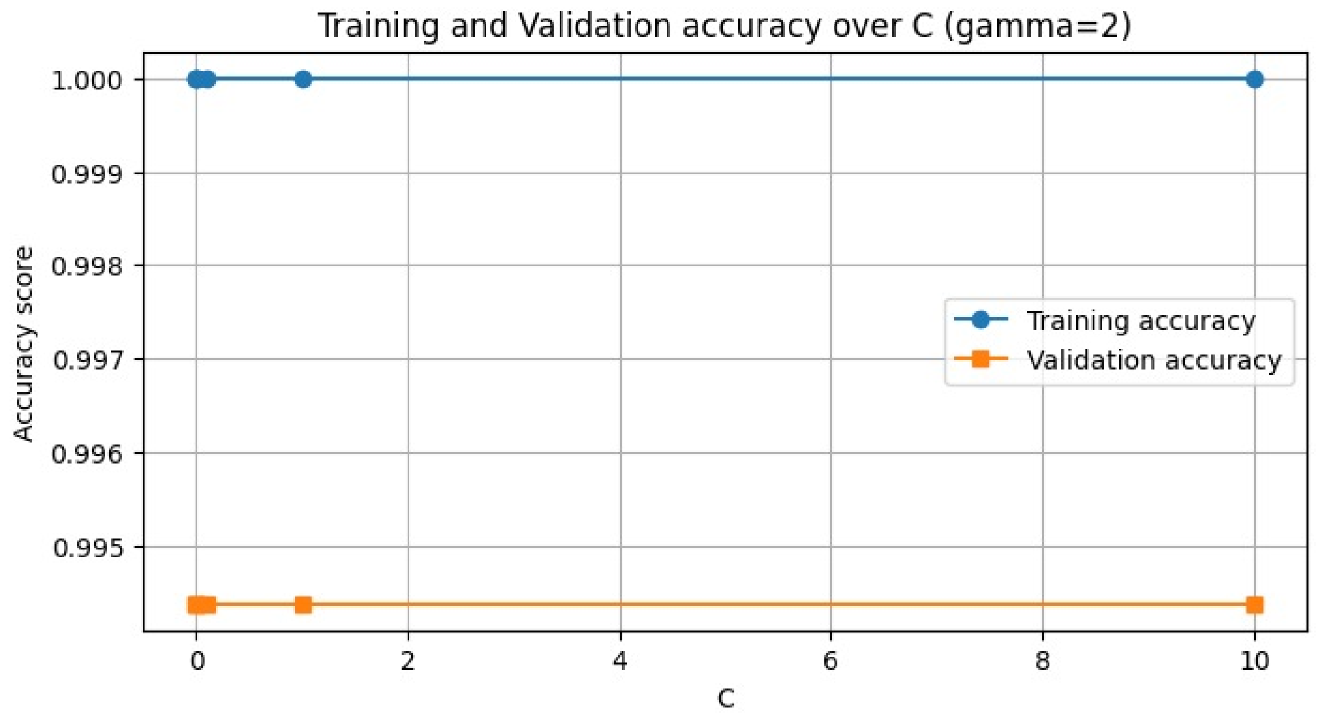

Figure 13: Evolution of the accuracy w.r.t C considering a polynomial kernel

References

[1] C.-C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines*, Department of Computer Science, National Taiwan University, Taipei, Taiwan, 2022. Available at: `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`