Project #2 - Support Vector Machines

Group gli Ottimizza-TORI
Francesco Elvio Buti, Luca Bigi, Matteo Martinovich

December 2024

Table 1: Summary of Results

Question	C	γ	q	Training accuracy	Test accuracy	Number of iterations	KKT violations	CPU time	Opt. f value
Q1	15	$8e^{-5}$	-	99,81%	99,25%	18	$4{,}9e^{-12}$	5,78s	-774,52336
Q2	15	$8e^{-5}$	90	99,81%	99,25%	150	$6{,}1e^{-10}$	0,61s	-774,52336
Q3	15	$8e^{-5}$	2	99,81%	99,25%	2452	$9{,}47e^{-9}$	0,79s	-774,52336
Q4	15	$8e^{-5}$	2	99,54%	98,33%	9146	$9{,}69e^{-9}$	3,11s	-777,80477

Disclaimer: The results in Table 1 for Question 4 represent averages of the objective function values and KKT violations from the individual binary classifiers in the One-Against-One (OAO) strategy. Detailed, non-aggregated values are provided in the appendix with the Python output (see Figure 5). The results for the 1vs5 classification, shown in Figure 5, differ from previous questions due to the larger dataset and its randomization with a fixed seed before splitting into training and test sets, causing variations in the training and test sets for the 1vs5 pair and leading to differences in the objective function.

Project Overview

In this project, we develop Support Vector Machines (SVM) to classify handwritten digits from the MNIST dataset, which consists of 28x28 pixel grayscale images. In the first three questions, the focus was on binary classification between two digits, 1 and 5, in the fourth question, the problem was extended to multiclass classification by including the digit 7, requiring additional strategies to adapt the SVM framework to handle multiple classes efficiently. To address the challenges of non-linear data separations, both the Radial Basis Function (RBF) kernel and the polynomial kernel were implemented. While both kernels demonstrated comparable accuracy in the first question, the RBF kernel is widely regarded as more versatile in capturing intricate decision boundaries.
For this reason, and to ensure scalability to the multiclass classification task, the RBF kernel was chosen for the rest of the project.

$$k(x, x') = e^{-\gamma \|x - x'\|^2}$$

Data Preparation

To ensure the model was trained on balanced and representative data, we selected 1,000 samples for each class. The dataset was then split into a training set (80%) and a test set (20%) using the train_test_split function. To maintain class balance in both subsets, we enabled the stratify option and set a seed for the random reshuffling to ensure reproducibility. Standardization was applied to the training set features and the same transformation was then applied to the test set using the statistics computed from the training set, ensuring consistency and avoiding data leakage.

Question 1: Solving the Dual SVM Problem

A pivotal aspect of this project was solving the dual form of the soft SVM, that is a quadratic programming problem and it's formulation is reported below:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{\alpha}^\top Q\boldsymbol{\alpha} - \mathbf{e}^\top\boldsymbol{\alpha} \\ \text{subject to:} \quad & 0 \leq \alpha_i \leq C \quad \forall i \in \{1, 2, \ldots, P\} \\ & \mathbf{Y}^\top\boldsymbol{\alpha} = 0 \end{aligned}$$

where:

- $\boldsymbol{\alpha} \in \mathbb{R}^P$ is the vector of dual variables, with P being the number of training samples
- $\mathbf{Y} \in \mathbb{R}^P$ is the vector of class labels ($+1$ or -1)
- $Q = \text{diag}(\mathbf{Y})K(\mathbf{X}, \mathbf{X})\text{diag}(\mathbf{Y}) \in \mathbb{R}^{P\times P}$, with $K(\mathbf{X}, \mathbf{X}) \in \mathbb{R}^{P\times P}$ representing the kernel matrix
- $\text{diag}(\mathbf{Y}) \in \mathbb{R}^{P\times P}$ is a diagonal matrix with the elements of the label vector Y on its diagonal
- $\mathbf{X} \in \mathbb{R}^{P\times d}$ is the matrix representing the training data, where d is the number of features per sample
- $\mathbf{e} \in \mathbb{R}^P$ is a vector of all ones
- $C \in \mathbb{R}$ is the regularization parameter

To address this in Question 1, we utilized the cvxopt library, which is specifically designed for efficiently solving quadratic programming problems. Precise tuning of the SVM hyperparameters C and γ was accomplished through a grid search integrated with k-fold cross-validation. For each fold, the training set was split into 80% for training the model and 20% for validating its performance. This rigorous approach helped us avoid overfitting and ensured that the SVM could generalize effectively from training to unseen data. The optimal hyperparameters identified were then consistently applied to subsequent questions.

Grid Search Analysis

Identifying Patterns of Overfitting and Underfitting

The first grid search explored a broad range of values for C and γ, revealing distinct patterns of underfitting and overfitting (Figure 1A). Large values of γ led to overfitting, as the mean training error rate remained at zero while the mean validation error rate increased significantly (Figure 1C), reaching approximately 23% at $\gamma = 0.1$. Conversely, small values of C caused underfitting, as the model penalized misclassifications too weakly, resulting in higher training and validation error rates.
This behavior is further detailed in Figure 1B, where γ is fixed at 1×10^{-4}, corresponding to the minimum mean validation error rate in this grid search. As C increases from small values, the mean validation error rate decreases sharply before stabilizing for $C > 0.8$.

Refining Hyperparameter Selection

Insights from this grid search motivated a more focused and refined exploration of the hyperparameter space. In the second grid search, values of $C > 50$ were excluded from the 3D plot (Figure 2A) to improve clarity by reducing point density. The refined plot reveals that increasing C beyond moderate values does not improve performance: the mean validation error rate begins to stabilize around $C = 8$, reaching a local minimum at $C = 25$, while the mean training error rate continues to drop to zero (Figure 2B). The 2D analysis (Figure 2C), with C fixed at 25, shows that the mean validation error rate reaches its minimum around $\gamma = 1.5 \times 10^{-4}$.

Beyond this value, the validation error progressively increases, signaling the onset of overfitting, consistent with trends observed in the first grid search. Based on these results, we selected $\gamma = 1 \times 10^{-4}$ to prioritize reduced model complexity and mitigate the risk of overfitting. Similarly, $C = 15$ was chosen as a balanced value, as no significant signs of overfitting were observed for higher C values within the analyzed range.

Reducing Complexity to Mitigate Overfitting Risk

To further minimize the complexity of the SVM, we conducted two additional analyses. The first, with C fixed at 15 (Figure 3A), demonstrated that lowering γ to 8×10^{-5} resulted in strong **mean validation accuracy of 99.06%** while reducing the risk of overfitting. The second analysis, with γ fixed at 8×10^{-5} (Figure 3B), confirmed that selecting $C = 15$ balances regularization and performance without introducing overfitting. **Taking these findings into account, the final hyperparameters were determined as $\gamma = 8 \times 10^{-5}$ and $C = 15$. This configuration ensures a low risk of overfitting while maintaining robust generalization and high accuracy**. The final results of this Question are presented in Table 1, while the corresponding confusion matrix is shown in Figure 6.

Question 2: Decomposition Algorithm

As required in Question 2, we implemented a decomposition method to solve the dual soft SVM problem by breaking it into smaller subproblems. To efficiently solve these subproblems, we used the cvxopt library, as in Question 1, and further refined the solver tolerances to boost computational efficiency and reduce runtime while maintaining high prediction accuracy.

The decomposition algorithm solves the dual soft SVM problem by iteratively optimizing a subset of q dual variables α, known as the working set W. To construct the working set W, the algorithm identifies two groups of indices, R and S, which represent distinct subsets of the training data. It is important to note that R and S are not necessarily disjoint and in most cases have overlapping indices. The selection process for indices in R and S is based on evaluating conditions on the dual variables α and their corresponding labels. For each index in R and S, the product between the gradient of the objective function and the corresponding training label is calculated. Indices in R are then sorted by the lowest values of this product, while indices in S are sorted by the highest values. To ensure convergence, the first $q/2$ indices from both R and S, after sorting, are selected to form the working set W for each iteration. The working set size was chosen to be $q = 90$, reflecting a careful trade-off designed to optimize the decomposition algorithm's performance (Figure 4). A smaller q results in solving many subproblems, requiring a higher number of iterations overall, while a larger q reduces the number of subproblems but increases the computational cost of solving each one. Additionally, excessively large q values, particularly those exceeding the number of free support vectors expected at the optimal solution, can lead to convergence issues or inefficiencies in the optimization process. However, in our case, the dataset size was manageable, allowing $q = 90$ to strike a practical balance between computational cost and the number of subproblems to solve in order to reach optimality, while ensuring it remained below the number of free support vectors expected at the optimal solution.

To enhance efficiency and reduce optimization time, we used a dictionary to store precomputed columns of the Q matrix. This method is especially beneficial when q is moderate, as it reduces a lot the need for repeated computations. However, even with our larger choice of $q = 90$, the use of the dictionary led to a 40% reduction in the overall runtime of the algorithm. The final performance of the algorithm is summarized in Table 1, with the confusion matrix provided in Figure 7.

Question 3: SMO-MVP

For Question 3, we implemented the Most Violating Pair decomposition method with a working set size of $q = 2$. This method focuses on selecting the two most influential indices to form the working set W, which is crucial for identifying and adjusting the most significantly margin-violating support vectors to improve the SVM's accuracy. Unlike in Question 1 and 2, where the optimization relied on the cvxopt library, here the optimization process was implemented manually. At each iteration, we dynamically determined the steepest feasible descent direction and the step size for the selected working set. By strategically selecting direction using the MVP indices, we ensured convergence for each subproblem in one single iteration, significantly improving the overall optimization time. A key innovation to improve the run time was the use of a dictionary to store precomputed columns of the Q matrix. This approach played a pivotal role by enabling rapid access to the necessary Q matrix values, ensuring that these critical indices could be processed efficiently.
The results obtained are listed in Table 1, and the confusion matrix is reported in Figure 8.

Question 4: SVM Strategy for Multiclass Classification

To handle the requirements of Question 4, the SVM was extended to a multiclass classification problem involving three classes: 1, 5, and 7. For this purpose, we implemented and tested both the One-Against-One (OAO) and One-Against-All (OAA) strategies, comparing their computational efficiency and predictive accuracy. Since we had only three classes both the OAO and OAA strategy involved training only three binary classifiers. However, the OAO approach proved to be more efficient since it requires processing fewer data points per classifier (each classifier is trained on only two-thirds of the data compared to OAA), while achieving comparable accuracy. The OAA strategy, while conceptually simpler, required each classifier to process the full dataset and distinguish one class against all others. Given these advantages, we chose to use OAO for the final implementation.
For the binary SVM models, we applied the Most Violating Pair (MVP) decomposition method with $q = 2$, as used in Question 3. This approach allowed for efficient training of the binary SVMs by focusing on a working set of indices strategically selected at each iteration, ensuring computational efficiency and high accuracy.
After the training phase, each binary classifier outputs its predictions, which are then converted from the +1 and -1 values into their corresponding class labels. This crucial step generates a prediction matrix with three rows, each row corresponding to one binary classifier used in the OAO approach, with the values representing true class labels instead of mere +1 and -1 predictions.
The final predictions were then combined using a voting system: for each test sample (column of the prediction matrix), the class with the most occurrences was selected as the predicted class. This method effectively used binary SVM models to solve the multiclass problem while keeping computational costs low.
The performance of this method is detailed in Table 1 , with the confusion matrix displayed in Figure 9.

Appendix

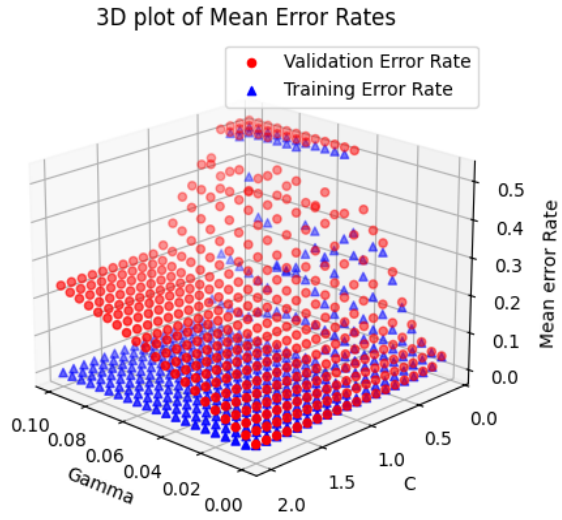

Figure 1A

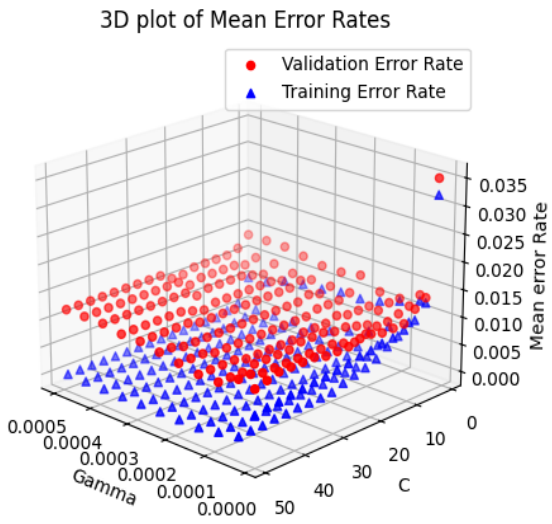

Figure 2A

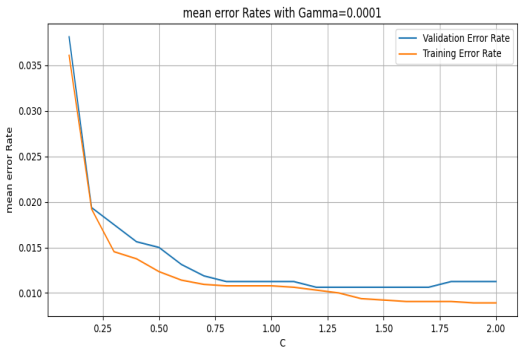

Figure 1B

Figure 2B

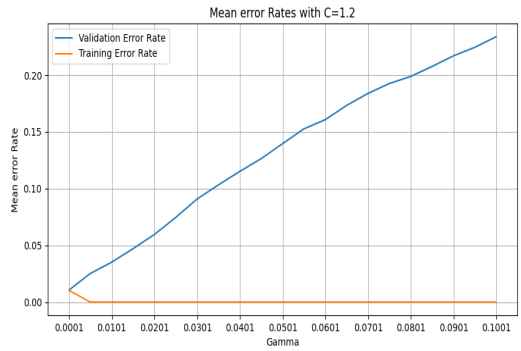

Figure 1C

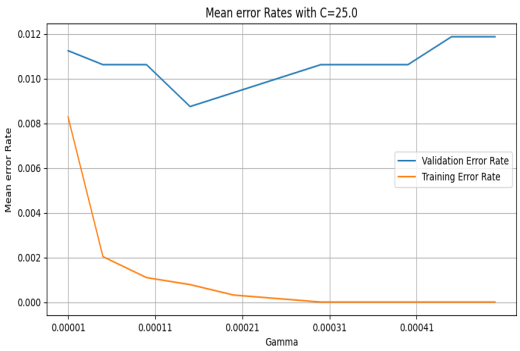

Figure 2C

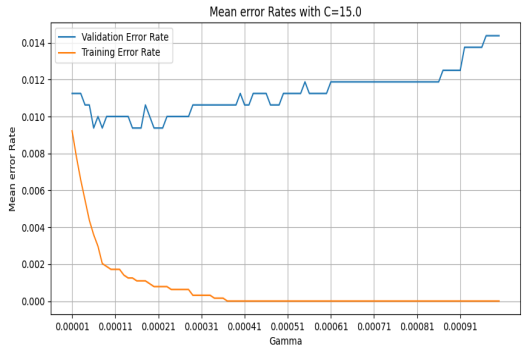

Figure 3A

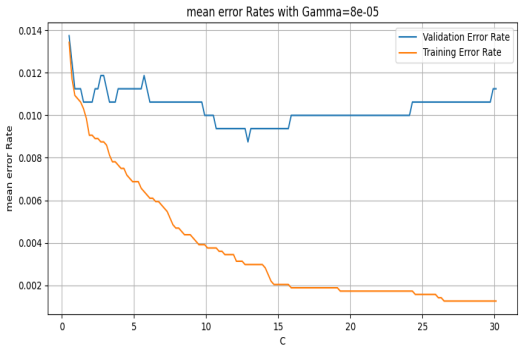

Figure 3B

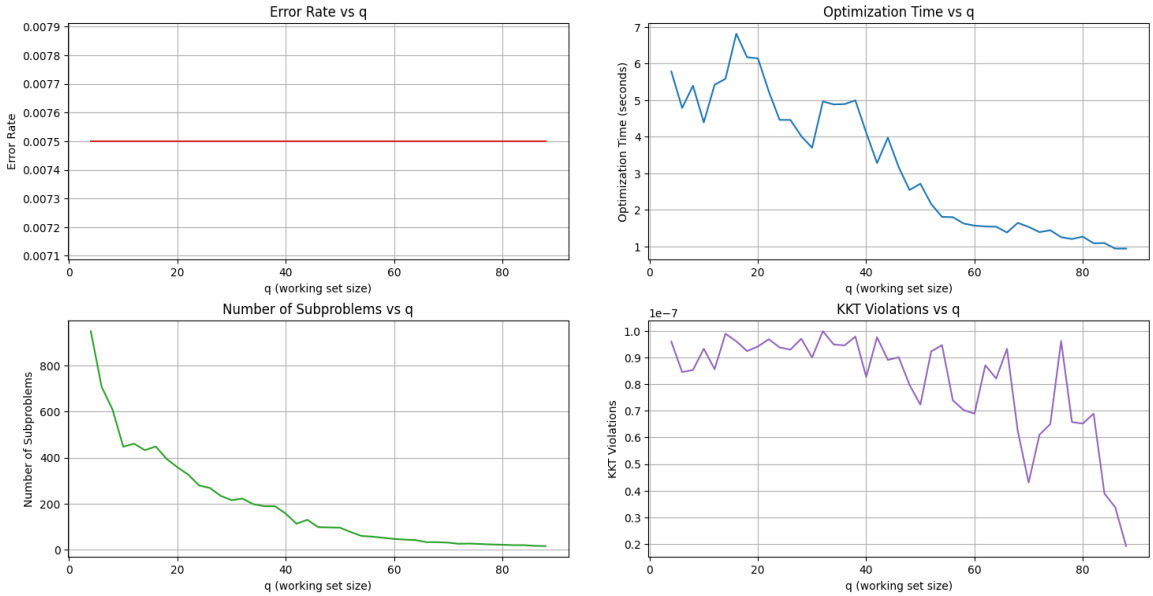

Figure 4

	Q1 Values
1. C	15
2. Gamma	8e-05
3. Accuracy on training set	0.998125
4. Accuracy on test set	0.9925
5. Optimization Time (seconds)	5.787700653076172
6. Iterations	18
7. KKT violations (m-M)	4.907407813448117e-12
8. Optimal f value	-774.5233671284755
9. Solver status	optimal

	Q3 Values
1. C	15
2. Gamma	8e-05
3. Accuracy on training set	0.998125
4. Accuracy on test set	0.9925
5. Run Time (seconds)	0.7945461273193359
6. Number of Q columns computed	254
7. Iterations	2452
8. KKT violations (m-M)	9.472422290990323e-09
9. Optimal f value	-774.52336712849

	Q2 Values
1. C	15
2. Gamma	8e-05
3. q	90
4. Accuracy on training set	0.998125
5. Accuracy on test set	0.9925
6. Run Time (seconds)	0.6133365631103516
7. Total number iterations	150
8. KKT violations (m-M)	6.107353511808356e-10
9. Optimal f value	-774.5233671264039
10. Distinct Solver State History	['optimal']

	Q4 Values
1. C	15
2. Gamma	8e-05
3. Implemented multiclass strategy	OAO
4. Accuracy on training set	0.9954166666666666
5. Accuracy on test set	0.9833333333333333
6. Run Time (seconds)	3.116934061050415
7. Number of iterations	9146
8. KKT violations (m-M) 1vs7	9.71765073964903e-09
9. KKT violations (m-M) 5vs1	9.685901747324976e-09
10. KKT violations (m-M) 7vs5	9.686260404873082e-09
11. Optimal f value 1vs7	-827.999149391529
12. Optimal f value 5vs1	-784.6593973713758
13. Optimal f value 7vs5	-690.7557728544858

Figure 5

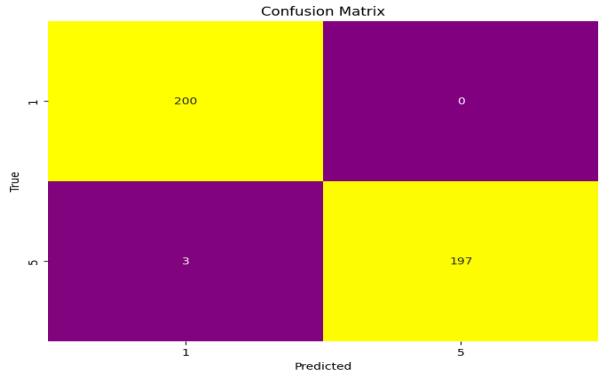

Figure 6: Q1 Confusion Matrix

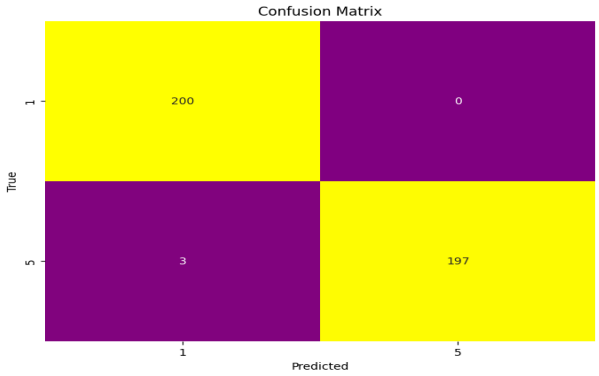

Figure 7: Q2 Confusion Matrix

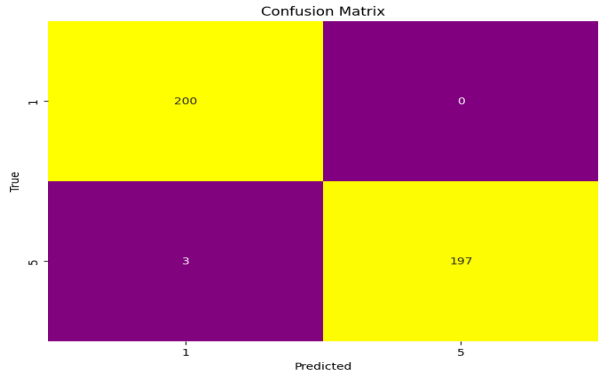

Figure 8: Q3 Confusion Matrix

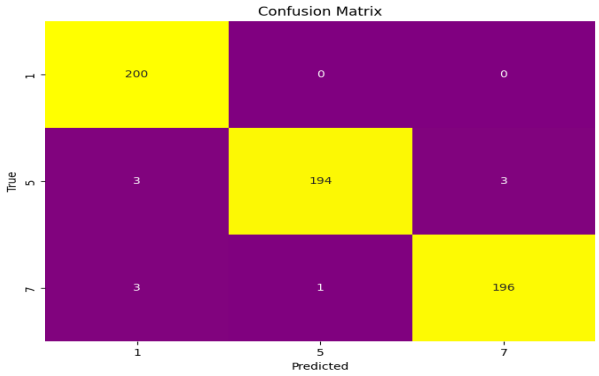

Figure 9: Q4 Confusion Matrix