Sapienza
Università di Roma

Department of Computer, Control, and Management Engineering

PROJECT 2

Course of Optimization Methods for Machine Learning

Project by Group AWS:

Buzzichini Davide 1895533 and Mazza Manuel 1819451

Academic Year 2024/2025

Introduction

We choose to adopt the polynomial kernel into all of our implementations, so that the returned results would have been consistently comparable:

$$K(x, y) = (x^\top y + 1)^\gamma$$

The entire dataset has always been randomly split into a training set(80%) and a test set(20%), using as seed one of our matricolas, and scaled using appropriate functions provided by the scikit-learn library. Also, each program has been divided into an effective training function and a "printing routine", that computes and returns all the metrics of interest for the problem's resolution. The goal of this approach is reducing the time to compute each iteration when running computationally expensive functions as the grid search.

1 Question 1

The selection of the hyperparameters C and γ has been made based on a grid search algorithm implementing the 5-fold cross validation. This allowed us to find the most performing combination of the two, choosing from the following ranges:

- C: *1, 2, 3, 4, 5, 10, 15, 20, 25, 30, 40, 50, 60, 75, 90, 100, 250, 500, 750, 1000*
- γ: *1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19*

Such algorithm returned C= 1 and γ= 2 to be our optimal hyperparameters' values.

A deeper analysis to address the possible presence of under/over fitting has been conducted on the basis of the accuracy error's fluctuations in response to the different values of each hyperparameter. As shown in *Figure 2*, the accuracy on both the training and the test set remain unchanged as C increases. Specifically, we notice that the test accuracy reaches its optimal value at C=1, as further increasing it does not lead to any improvements. That being said, we can't report any sign of overfitting or underfitting. On the other hand, the grid search results for γ, shown in *Figure 3*, illustrate how this parameter actually influences the test accuracy, whose optimal value is reached when γ=2. The figure also displays that as γ continues to increase, the accuracy starts to swing inside a somewhat descending trend. For this reason, it's evident how for increasing values of γ, the model might falls into overfitting due to the training error staying equal to 1.

After fixing the hyperparameters, our concern was the selection of the value of ϵ, for which we simply tested manually some of its values in the context of 10^{-x}, keeping an eye also on the number of *support vectors* each ϵ produced. With that in mind, the most suitable value turned out to be ϵ= 10^{-9}, which would be kept the same along all questions to ensure comparability.

To solve the quadratic problems, we exploited the "*solvers.qp*" optimization routine from the *cvxopt* python package, in which we set all the tolerances (abstol, reltol, feastol) to 10^{-15}, to best balance out convergence time and accuracy rate.

All the results of this implementation are visible in the *Table 1*, while the resulting confusion matrix is displayed in the *Figures* section.

Solver	**Train Accuracy**	**Test Accuracy**	**Final O.F. value**	**N° it**	**Max KKT viol.**	**Run time**
cvoxpt	1	0.995	-5.9e-4	19	3e-10	6.3s

Table 1: Summary table for Q1 with C=1 and gamma=2

2 Question 2

As requested, we kept the same hyperparameters identified for the previous question to implement the decomposed quadratic programming algorithm. For such algorithm, SVM light served as the decomposition method. With respect to question 1, we also kept the same data manipulation concept, optimization routine and ϵ value, as all were still suitable for this implementation as well.

The dimension of the working set has been identified using, once again, a 5-fold cross-validation, which returned the value of q (selected among the range *[4,100]*) associated with the highest average accuracy on the test sets or, for similar results, with the lowest optimization time. Actually, this approach showed how the dimension of q almost negligibly influenced the accuracy score on the test set, while it mainly impacted the run time of the algorithm, with higher values of q providing much more efficient execution. However, the algorithm still appointed q=66 to be the slight best performing value accuracy wise, also providing a good enough time efficiency due to it being in the upper side of q's range.

Instead of building the entire matrix Q and extracting from it the sub-matrix in exam, we selected at each iteration only the columns associated with the violating pairs among those in the working set, allowing us to get a more efficient optimization without losing any relevant information. This approach was developed to face and overcome some memory issues emerged during the execution of a first implementation attempt, such as *"Execute failed: MemoryError: Unable to allocate [...] for an array with shape [...] and data type [...]"*.

The cycle ascribed to solve the optimization was bounded by a stopping criterion on the optimality conditions, which would consider the optimization done whenever the difference between m(α) and M(α) is greater or equal than a tolerance variable, that has been set to 10^{-10}. The selection of this tolerance value has been made with the goal of balancing out the maximum KKT violation allowed and the execution time of the algorithm. From this perspective, we identified tol=10^{-10} to be the best value for our program, as it granted an optimally low maximum kkt violation and achieved very good performances. From our testing, we noticed that our program has no issues to achieve a max kkt of even smaller magnitude, but at the cost of increasing the execution time. That, from our point of view, would invalidate the goal of developing a decomposition method and it is also unnecessary in our application due to the already low enough maximum kkt violation attained.

All the results of this implementation are visible in the *Table 2*, while the resulting confusion matrix is displayed in the *Figures* section.

Solver	**Train Accuracy**	**Test Accuracy**	**Final O.F. value**	**N° it**	**Max KKT viol.**	**Run time**
cvoxpt	1	0.97	-1.8e-2	587	6.8e-11	1.8s

Table 2: Summary table for Q2 with C=1, gamma=2 and q=66

3 Question 3

This problem basically requires coding the "Sequential Minimal Optimization", in which the working set of fixed dimension (q=2) is constructed using the "Most Violating Pair" (MVP) rule. To do so, at each iteration the algorithm built the 2-column sub-matrix using the same approach used in question 2. This computation was followed by the definition of the feasible descent direction and, consequently, of the values of the maximum t feasible and of the optimal t. After selecting the smallest among these last two variables, the SMO algorithm imposes applying the update rule on α, and start a new iteration whenever the KKT violation's magnitude is too relevant. Such stopping criterion has been kept the same as in the previous question, with a new iteration starting until $m(\alpha) - M(\alpha) >= 10^{-10}$. All the discussions about the value of the tolerance made for the previous question, also hold for this one.

The results of this implementation are visible in the *Table 3*, while the resulting confusion matrix is displayed in the *Figures* section.

Train Accuracy	**Test Accuracy**	**Final O.F. value**	**N° it**	**Max KKT viol.**	**Run time**
1	0.97	-1.8e-2	2295	9.5e-11	1.0s

Table 3: Summary table for Q3 with C=1, gamma=2 and q=2

4 Question 4

Among the two possible methods for SVM multi-class problems, we opted for the One-Against-All approach to classify the numbers 1,5 and 7. This choice led us to solve 3 subproblems, in which we compared each class with the all the others, so: 1vsALL, 5vsALL and 7vsALL. For this implementation, we also included a grid search algorithm to identify the best set of hyperparameters. However, due to its long execution time (about 1 minute per-fold, so 5 minutes for each [C,γ] pair) we decided to stick with the hyperparameters we already used throughout all the other questions.

Each sub-problem produced the results displayed in the following corresponding tables, whereas the resulting confusion matrix is displayed in the *Figures* section.

Solver	**Train Accuracy**	**Test Accuracy**	**Final O.F. value**	**N° it**	**Max KKT viol.**	**Run time**
cvoxpt	1	0.985	-9.9e-4	20	1e-10	18.8s

Table 4: Summary table for Q4: 1vsALL with C=1 and gamma=2

Solver	**Train Accuracy**	**Test Accuracy**	**Final O.F. value**	**N° it**	**Max KKT viol.**	**Run time**
cvoxpt	1	0.985	-6.6e-4	20	9.0e-11	19.6s

Table 5: Summary table for Q4: 5vsALL with C=1 and gamma=2

Solver	**Train Accuracy**	**Test Accuracy**	**Final O.F. value**	**N° it**	**Max KKT viol.**	**Run time**
cvoxpt	1	0.985	-1e-3	19	5.4e-9	20.3s

Table 6: Summary table for Q4: 7vsALL with C=1 and gamma=2

5 Summary table and final considerations

	hyperparameters			ML peformance		optimization performance			
question	C	γ	q	Training accuracy	Test accuracy	number its	objective fun values	KKT viol	CPU time
Q1	1	2	/	1	0.995	19	-5.9e-4	3e-10	6.3s
Q2	1	2	66	1	0.97	587	-1.8e-2	6.8e-11	1.8s
Q3	1	2	2	1	0.97	2295	-1.8e-2	9.5e-11	1.0s
Q4	1	2	/	1	0.985	59	/	1.8e-9	58.7s

Table 7: Summary table

We can say that the classic dual SVM quadratic problem algorithm (Q1) turns out to be the most accurate among all the methods dealing with the same type of problem. On the other hand, both decomposition methods (Q2 and Q3) achieve a much higher efficiency time wise while losing a slim percentage in the accuracy measurement. Furthermore, we can stress how the SMO with MVP decomposition algorithm results to be a bit faster than the SVM decomposition method, while attaining the same accuracy results. To conclude, it's possible to notice how the multi-class SVM strategy allows to solve a much more complex problem, accomplishing great precision at the expense of having a much heftier execution time.

6 Figures

6.1 Q1

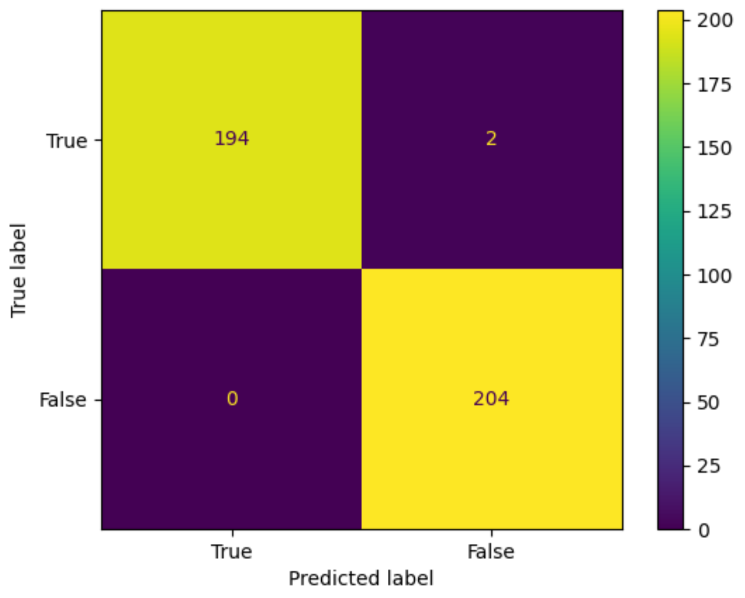

Figure 1: Confusion matrix Q.1

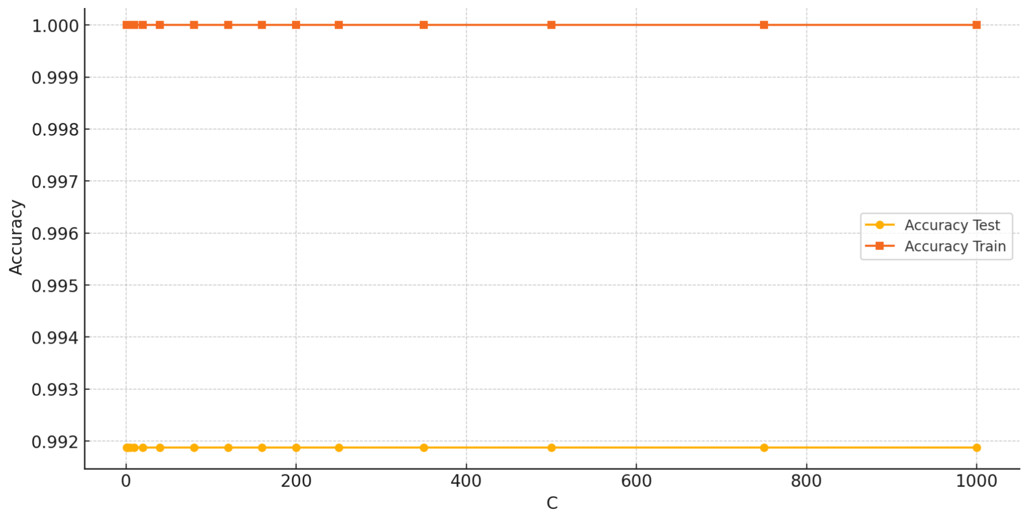

Figure 2: Evolution of accuracy's values with respect to C

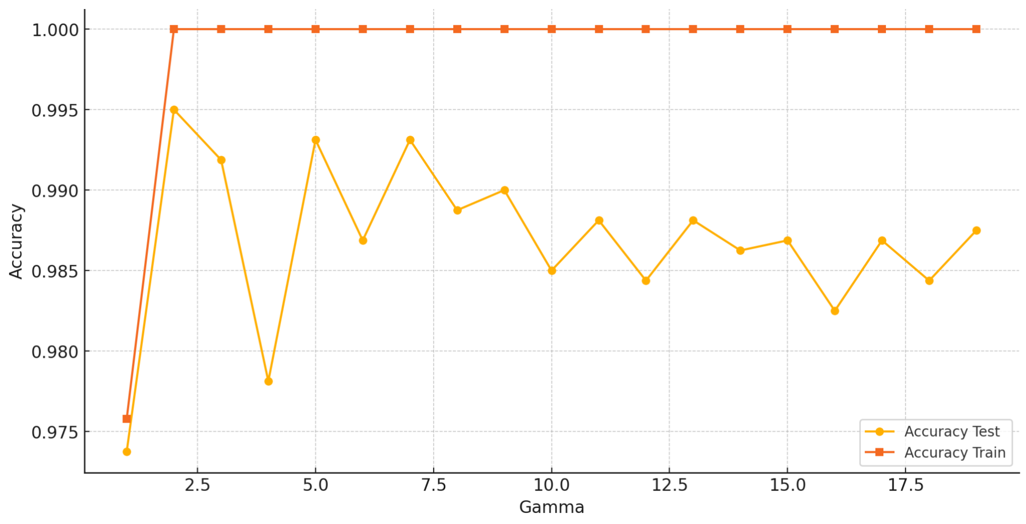

Figure 3: Evolution of accuracy's values with respect to Gamma

6.2 Q2

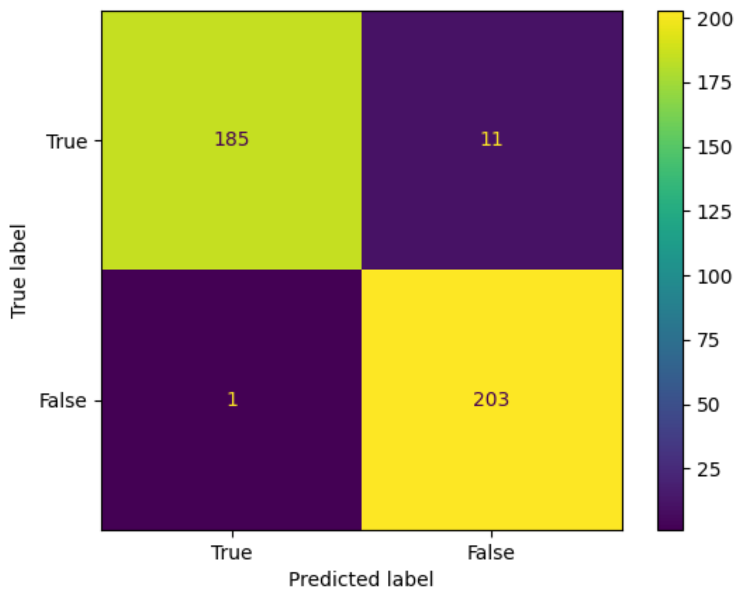

Figure 4: Confusion matrix Q.2

6.3 Q3

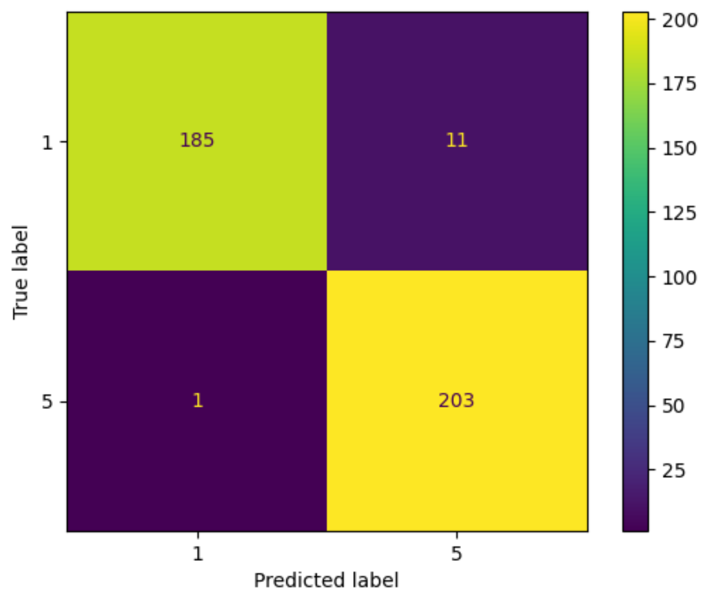

Figure 5: Confusion matrix Q.3

6.4 Q4

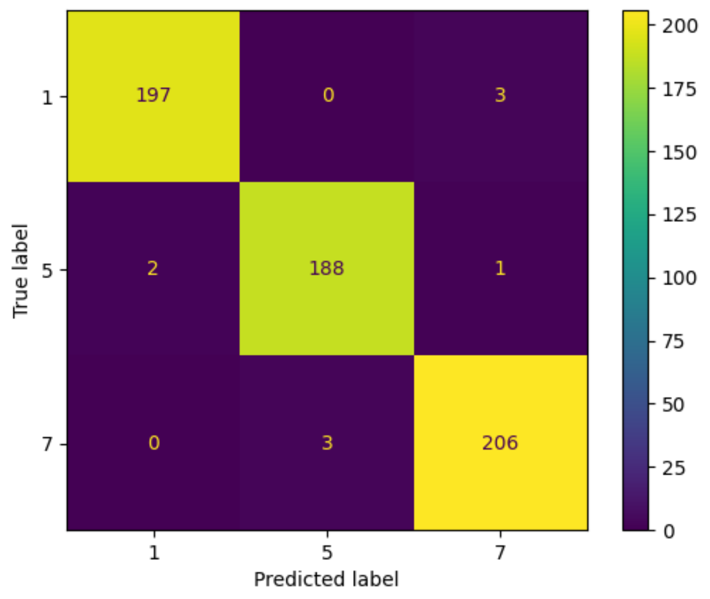

Figure 6: Confusion matrix Q.4