Optimization Methods for Machine Learning

Group Name : Optimized Noobs
Alessio Barboni - 2027647
Nina Kaploukhaya - 1998079
Mehrdad Hassanzadeh - 1961575

December 30, 2022

Method comparison table

Ex	Hyperparameters			ML performance		Optimization performance			
	C	γ	q	Training accuracy	Test accuracy	# its	number fun evals	KKT viol	cpu time
Q1	0.1	1	-	0.9987	0.995	13	13	No	2.02
Q2	0.001	1	6	0.99	0.99	99	840	No	0.5
Q3	0.1	1	-	0.995	0.988	100	100	Yes	0.043
Q4	0.1	1	-	0.671	0.642	30	30	No	11.631

Question 1

Choosing hyperparameters and kernel. We used the function "find hyperparameters()" from "functions-1.py", that performs grid search to find the best hyperparameters for both the RBF and the Polynomial Kernel. The train dataset is split with K-Fold cross-validation (setting K = 5). The validation accuracy is calculated as the average of the model accuracies on the 5 different parts of the dataset during KFold.

For the RBF kernel, we tried out different values with a *gridsearch* approach ($C \in \{0.0001, 0.001, 0.001, 0.1, 0.5, 1, 5\}$, $\gamma \in \{0.0001, 0.001, 0.005, 0.01, 0.1, 1, 5\}$), and the best hyper-parameters were $\gamma = 0.1$, $C = 5$. The model performances with this setting reached: validation accuracy = 0.953125, train accuracy = 1.

For the polynomial kernel, we also used *gridsearch* ($C \in \{0.0001, 0.001, 0.001, 0.1, 0.5, 1, 5\}, \gamma \in \{1, 2, 3, 4, 5, 10\}$), and the best hyper-parameters turned out to be $C = 0.1, \gamma = 1$. The model performances with this setting achieved: validation accuracy= 0.994375, train accuracy = 0.9990625.

As we can see, the model using the RBF kernel and the chosen hyper-parameters overfits, reaching an accuracy on the training set of 1, whereas its valid accuracy is lower (but most importantly much smaller than the model that uses the polynomial kernel). The other model instead shows great results for both the train and the validation set. In the plots 1 and 2 is shown how the performance of the model that uses the polynomial kernel change with different values of C and γ. From 1 we can see that a large value of C can cause overfitting, the train accuracy is almost 1, while test accuracy is low. According to this discussion we chose:

- Chosen Kernel: Polynomial.
- Chosen hyperparameters for polynomial kernel: $C = 0.1$, $\gamma = 1$.

Optimization routine. We used the interior-point method of the cvxopt library for quadratic programming: cvxopt.solvers.qp.
Machine learning performances. Train accuracy: 0.9987, Test accuracy: 0.995, Validation accuracy: 0.994375.
Optimization performance: Initial value of dual objective function: -2.237288, number of iterations: 13, number of function evaluations: 13, violation of the KKT condition: "No".

Question 2

The **decomposition method** is one of the most commonly used methods for solving support vector machines. In this method, instead of updating the parameters of the SVM model by considering all of the datapoints, we will update them by selecting a **subset of the datapoints**. **Working sets** are the datapoints that have been chosen for updating the parameters. We have used the same kernel and the hyperparmeters chosen in Question 1. Here we have a report about or model:

- **Chosen kernel**: Polynomial kernel
- **Optimization routine**: cvxopt.qp() with the default solver, absolute tolerance = 1e-13, relative tolerance = 1e-13, feasibility tolerance = 1e-13
- **Machine learning performances:** 0.50% on the trainig set, 0.49% on the test set
- **Initial value of the objective function:** 0
- **Final value of the objective function:** -2.23
- **Number of the iterations:** 594
- **Number of the function evaluations:** 4662
- **KKT condition violation:** No

As we have seen a poor performance using the chosen parameters in the Question 1, we decided to run a hyperparameter tuning for this problem. We will put the performance of our best model in the final table.

Question 3

Among the various existing decomposition methods, *Sequential Minimal Optimization (SMO)* algorithms are by far the most commonly used (to our knowledge). These algorithms update two variables at a time ($q = 2$), ie. they select the most sparse direction possible (since 2 is the minimum number of variables that must be changed in order to maintain feasibility). So at each iteration the the problem reduces to a *convex quadratic programming* problem of two variables, in a form which allows its solution to be found **analytically**. This could somehow explain the interest in defining such kind of algorithms.
We decided to use same kernel and hyper-parameters of Question 1.

- **Chosen kernel**: Polynomial kernel
- **Chosen C**: 0.1
- **Chosen** γ: 1

Optimization routine. We implemented the *SMO-MVP* algorithm, and we computed the optimal value of α at each iteration solving the quadratic problem analytically (i.e., implementing "Algorithm 5.A.1").

Machine learning performances. Train accuracy: 0.995, Test accuracy: 0.988.

Optimization performance: Initial value of dual objective function: 0, Final value of dual objective function: 0.017, the number of iterations: 100, number of function evaluations: 100, violation of the KKT condition: Yes.

Question 4

We used one-against-all method to perform multiclass classification. The SVM hyperparameters were the same as defined in the first question (Polynomial, $C = 0.1, \gamma = 1$).

Optimization routine. We used interior-point method of cvxopt library for quadtatic programming: cvxopt.solvers.qp.

Machine learning performances. Train accuracy: 0.671, Test accuracy: 0.642, Validation accuracy: 0.994375.

Optimization performance: Initial value of dual objective function(average among 3 models): -1.98, the number of iterations: 30, the number of function evaluations: 30, the violation of the KKT condition: "No".

Appendix

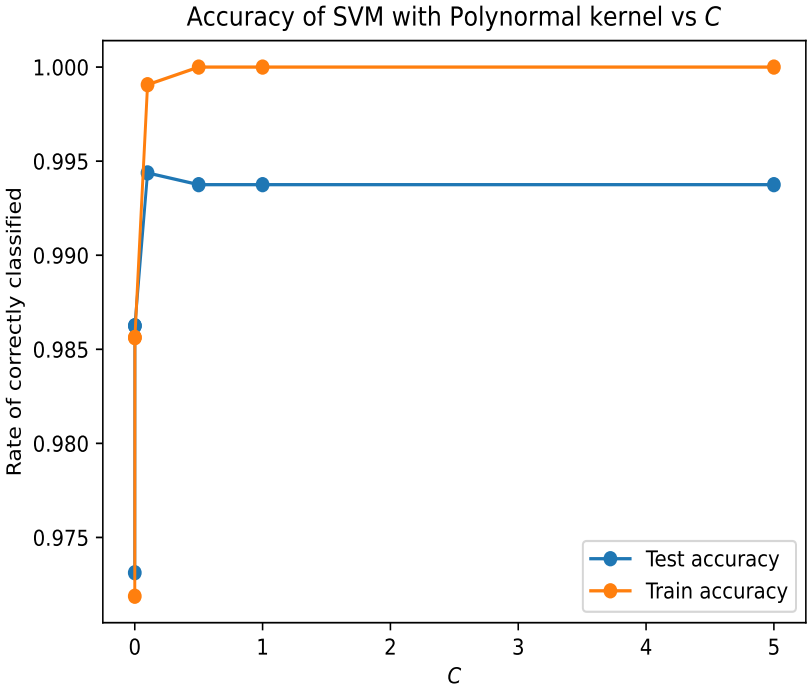

Figure 1: SVM performance with poly kernel, $\gamma = 1$ with different C

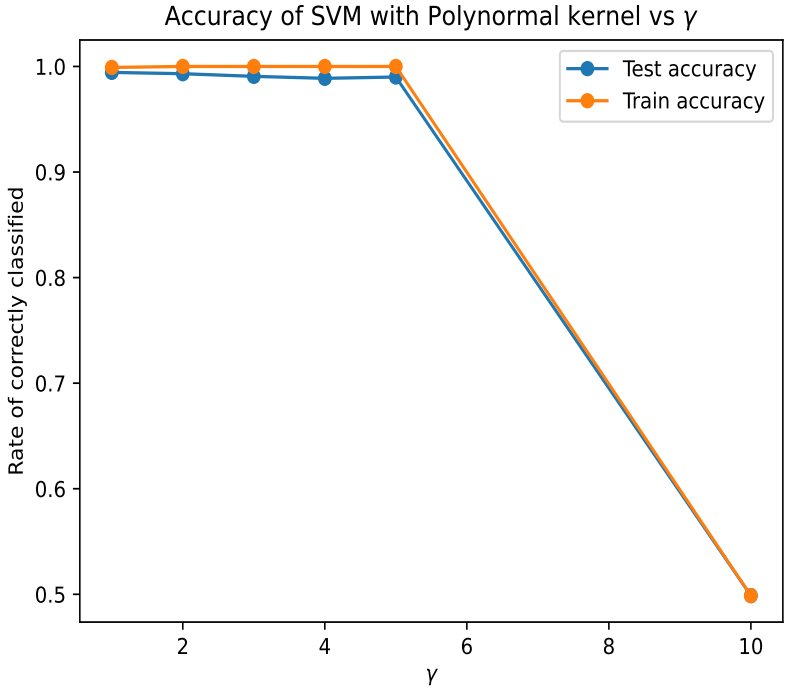

Figure 2: SVM performance with poly kernel, $C = 0.1$ with different γ