Project #2 - Support Vector Machines

Optimization Methods for Machine Learning - Fall 2022

Nicolò Bertini
Business intelligence and analytics
bertini.1844162@studenti.uniroma1.it

Leonardo Chichiricc̀o
Modelli decisionali per l'ingegneria gestionale
chichiricc̀o.1853964@studenti.uniroma1.it

Fabio Ciccarelli
Modelli decisionali per l'ingegneria gestionale
ciccarelli.1835348@studenti.uniroma1.it

***Abstract*—In this paper, we are going to discuss and analyze the implementation of different optimization methods for training Support Vector Machines (SVM) classifiers.**

I. Standard soft-SVM

A. Building the kernel matrix

For the first part of the project, we analyzed the performances of both the Gaussian and the polynomial kernels applied to the optimization procedure linked to SVM classifier training.
Given two vectors x and z, kernel functions can be defined as follows:

$$\textit{Gaussian Kernel} \quad k(x,z) = e^{-\gamma\|x-z\|^2}; \quad \gamma \geq 0$$
$$\textit{Polynomial Kernel} \quad k(x,z) = (x^T z + 1)^\gamma; \quad \gamma \geq 1$$

Note that, from the efficiency point of view, computing the entire kernel matrix between all the points of two data sets X_1 and X_2 can be very expensive. Hence, we derived it in a vectorized form, which lowers a lot the computational effort.

Polynomial kernel:

$$K(X_1, X_2) = (\mathbb{1} + X_1 X_2^T)^\gamma$$

$$X_1 \in \mathbb{R}^{N_1 \times n}$$
$$X_2 \in \mathbb{R}^{N_2 \times n}$$
$$\mathbb{1} \in \mathbb{R}^{N_1 \times N_2}$$

Gaussian kernel:

$$\|x-y\|^2 = (x-y)^T(x-y) = x^T x - 2x^T y + y^T y =$$
$$= \|x\|^2 - 2x^T y + \|y\|^2$$

$$x \in X_1$$
$$y \in X_2$$

$$K(X_1, X_2) = D_1 - 2X_1 X_2^T + D_2$$

where:

$$D_1 \in \mathbb{R}^{N_1 \times N_2} \quad s.t. \quad d_{i,j} = \|x_i\|^2 \quad \forall\, j = 1, ..., N_2$$

$$D_2 \in \mathbb{R}^{N_1 \times N_2} \quad s.t. \quad d_{h,k} = \|y_k\|^2 \quad \forall\, h = 1, ..., N_1$$

B. Selecting the best values for the hyperparameters

In order to select the best values for the model hyperparameter C and the kernel hyperparameter γ, we carried on a grid search over all the possible combinations of the following values:

Polynomial kernel:

$C \in \{0.1,\ 1,\ 3,\ 5,\ 10,\ 20,\ 50,\ 100,\ 500,\ 10^3,\ 5 \cdot 10^3,\ 10^4,\ 10^5\}$
$\gamma \in [1, 5] \cap \mathbb{N}$

Gaussian kernel:

$C \in \{0.1,\ 1,\ 3,\ 5,\ 10,\ 20,\ 50,\ 100,\ 500,\ 10^3,\ 5 \cdot 10^3,\ 10^4,\ 10^5\}$
$\gamma \in \{10^{-4},\ 5 \cdot 10^{-4},\ 10^{-3},\ 5 \cdot 10^{-3},\ 10^{-2},\ 5 \cdot 10^{-2}\}$

Note that we made the grid search on different values of γ, depending on the kernel. In order to select the best combination, we evaluated the performance of the model for each pair of C and γ via a 5-folds cross-validation procedure. In table I the optimal parameters of C and γ for the part 1 of the project are shown.

Kernel	C	γ	Validation score
Gaussian	5000	0.001	99.44%
Polynomial	1000	2	99.62%

TABLE I
RESULTS OF A 5-FOLDS CROSS-VALIDATION

1) Evaluating the occurrence of over/under-fitting: The data at hand were almost perfectly separable even with a linear classifier. This means that we could not notice any significant behavior in terms of over/under-fitting, particularly when applying a polynomial kernel (more stable in terms of accuracy). Nevertheless, driving the C value below a certain threshold, we noticed slightly worse performances (which can be interpreted as a modest under-fitting) for the Gaussian kernel SVM (see figure [3]). An ulterior analysis can be done in the case in which specific choices of C and γ are done. For instance, when $C = 0.1$ and $\gamma = 10^{-4}$, the solver finds a solution α^* such that we have no free support vectors (meaning that all the support vectors are at bound). In this case, to find the b* value we followed the procedure illustrated by C.J. Burges and D. Crisp [1]. We go deeper into this topic in I-D.
It is worth to notice that, when all the SVs are at bound, usually the accuracy of the classifier is really low, emphasizing a case of under-fitting.

C. Optimization routine

In order to get a solution for the soft-SVM dual formulation, we leveraged the unified optimization API qp_solvers, which allows the user to define a quadratic programming formulation in an easy and immediate way, and to choose a QP-specific solver, depending on the peculiarities of the problem. In our case, for the point 1 of the project we decided to use the CVXOPT solver, since it is supposed to perform better on high-sized problem (with n $>$ 1000, see Figure [1]). The optimization routine takes as input the vectors and matrices defining the quadratic problem (having only linear constraints) and returns the optimal values for α^*, representing the Lagrangian multipliers of the dual soft-SVM problem.

D. *Defining the classification function*

Once we have the optimal values for the lagrangian multipliers, we can proceed in computing the decision function representing the hyper-surface which divides the data points. The procedure to get it is described both in the lectures' slides and in Pattern recognition and Machine Learning by C. Bishop [2].

$$f(x) = sign\left(\sum_{i=1}^{P} \alpha_i^* y_i K(x_i, x) + b^*\right)$$

$$b^* = \frac{1}{|SV_f|} \sum_{i \in SV_f} \left(y_i - \sum_{m=1}^{P} a_m y_m K\left(\mathbf{x}_i, \mathbf{x}_m\right)\right)$$

Where SV_f is the set of the free support vectors, i.e. those data-points for which $0 < \alpha_i < C$.
As already said, in the case in which all the support vectors are at bound, meaning that $SV_f = \emptyset$, we compute b^* as follows:

$$b^* = \begin{cases} max\{max_{i \in S_-}(-1 - w^T x_i), max_{i \in V_+}(1 - w^T x_i)\}, & |S_-| \geq |S_+| \\ min\{min_{i \in V_-}(-1 - w^T x_i), max_{i \in S_+}(1 - w^T x_i)\}, & |S_-| \leq |S_+| \end{cases}$$

where

$$S_+, (S_-) = \{i \;\; s.t. \;\; \alpha_i = C \;\; and \;\; y_i = +1, (-1)\}$$
$$V_+, (V_-) = \{i \;\; s.t. \;\; \alpha_i = 0 \;\; and \;\; y_i = +1, (-1)\}$$

	Train set	Validation set	Test set
Accuracy score (G)	100%	99.43%	98.75%

TABLE II
GAUSSIAN

	Train set	Validation set	Test set
Accuracy score (P)	100%	99.62%	99.25%

TABLE III
POLYNOMIAL

1) *Machine Learning performances:*

#iterations	$f(\alpha_0)$	$f(\alpha*)$	$m^* - M^*$
27	$-1.46 \cdot 10^{10}$	$-1.0963 \cdot 10^3$	$5.01 \cdot 10^{-14}$

TABLE IV
GAUSSIAN

#iterations	$f(\alpha_0)$	$f(\alpha^*)$	$m^* - M^*$
24	$-1.05 \cdot 10^7$	$-2.07 \cdot 10^{-2}$	$1.64 \cdot 10^{-14}$

TABLE V
POLYNOMIAL

2) *Optimization performances:*

In some cases, the SVM with polynomial kernel ensures more solid results (for instance, it is almost impossible having no free support vectors when using it, even with very low values of C). Nevertheless, we are going to show the obtained results for both the gaussian and the polynomial kernel SVM in the following sections as well, since, apart from the aforementioned extreme choices of the hyper-parameters, their performances are almost always comparable.

II. GENERAL DECOMPOSITION ALGORITHM (SVM^{light})

A. *Optimization routine*

The SVM^{light} is a training procedure which requires the solution of multiple q-dimensional sub problems (where q is an hyper-parameter that can be tuned to get the beste performances in terss of computational effort and accuracy of the classifier). It is, therefore, a decomposition method used to lower the complexity of the standard soft-SVM "batch" algorithm (in which a single fully dimensional quadratic problem is solved). The q-dimensional formulation, as well as the criterion for the choice of the working set at each iteration, are described both in L. Palagi, M. Sciandrone (2005) and in the lectures' slides. They can be briefly summarized as follows:

Sub-problem

$$\begin{aligned} &\min_{\alpha_W} \frac{1}{2}\alpha_W^T Q_{WW}\alpha_W - (e - Q_{W\overline{W}}\alpha_{\overline{W}}^k)^T\alpha_W \\ \text{s.t.} \quad &y_W^T\alpha_W = -y_{\overline{W}}^T\alpha_{\overline{W}}^k \\ &0 \leq \alpha_W \leq Ce_W \end{aligned}$$

Selection of the working set

$$R(\alpha^k) = \{i: \; \alpha_i > 0 \;\; , y_i = +1\} \;\; \cup \;\; \{i: \; \alpha_i < C \;\; , y_i = -1\}$$
$$S(\alpha^k) = \{j: \; \alpha_j > 0 \;\; , y_j = -1\} \;\; \cup \;\; \{j: \; \alpha_j < C \;\; , y_j = +1\}$$

$$I(\alpha^k) = \left\{i \in R(\alpha^k) \;\; : \;\; -\frac{\nabla f(\alpha^k)_{i^1(k)}}{y_{i^1(k)}} \geq .... \geq -\frac{\nabla f(\alpha^k)_{i^q1\,(k)}}{y_{i^q1\,(k)}}\right\}$$

$$J(\alpha^k) = \left\{j \in S(\alpha^k) \;\; : \;\; -\frac{\nabla f(\alpha^k)_{j^1(k)}}{y_{j^1(k)}} \leq .... \leq -\frac{\nabla f(\alpha^k)_{j^q2\,(k)}}{y_{j^q2\,(k)}}\right\}$$

where $q_1 = q_2 = \frac{q}{2}$

$$W = \{I(\alpha^k), J(\alpha^k)\}$$

For solving the q-dimensional quadratic problem at each iteration, we used the quadprog solver, since it is the most efficient solver provided by the qp_solvers API for low-sized problems (n $<$ 200, see figure 1).

B. *PPD modification to the standard SVM^{light} algorithm*

One of the main issues we had to face while defining the optimization routine related to the second part of the project is the fact that the chosen solver required the Q_{WW} matrix to be positive definite, which means that its minimum eigenvalue must be higher than 0. The higher is q, the more likely is to have that condition violated, and therefore to prematurely interrupt the routine without having a good solution. Moreover, the convergence of the standard SVM^{light} algorithm requires the exact same hypothesis. On the other hand, CVXOPT does not require these same properties, but it may return very poor solutions if a starting point for the optimization is given and it is way slower than quadprog for this kind of problems (it takes almost ten times more time than the alternative solver). For this reason, we decided to apply the Proximal Point Modification proposed by L. Palagi and M. Sciandrone in [3], only in those case in which the standard SVM^{light} failed.
The PPD-SVM^{light} formulation is the following one:

$$\begin{aligned} &\min_{\alpha_W} \frac{1}{2}\alpha_W^T(Q_{WW} + w\tau I_W)\alpha_W - (e_W - Q_{W\overline{W}}\alpha_{\overline{W}}^k - 2\tau\alpha_W^k)^T\alpha_W \\ \text{s.t.} \quad &y_W^T\alpha_W = -y_{\overline{W}}^T\alpha_{\overline{W}}^k \\ &0 \leq \alpha_W \leq Ce_W \end{aligned}$$

The PPD modification allowed us to get good results for larger values of q as well, and even better performances could have been achieved if a proper tuning of τ had been done. For cleanness sake, and since $\tau = 1$ seemed to behave correctly for our purposes, we did not dig deeper into this topic.

C. *Choosing the best value for q*

In order to select the best value for the dimension of the sub-problems, both for the polynomial and for the gaussian kernel SVM, we evaluated the performance of the model depending on q, fixing C and γ to the best values found in the previous section, via a 5-folds cross-validation procedure. Note that, in this case, the validation accuracy score was almost always the same, independently from the value of q chosen. For this reason, we decided to evaluate the best dimension of the sub problems comparing the computational effort (in terms of run time) made by the solver when changing the parameter.
In figures [2] and [4] we plotted the convergence time of the optimization routines depending on q. As it can be noticed, after a certain threshold both the gaussian and the polynomial kernel SVM have a spike, due to the fact that, increasing the size of the sub-problems, it is more likely to incur the aforementioned problem related to the positive definiteness of the Q_{WW} matrix, and therefore to aplly the PPD-modification.
In tables VI the optimal values of q for part 2 of the project are shown.

Kernel	q	Time for convergence
Gaussian	70	$\simeq$ 0.5s
Polynomial	100	$\simeq$ 0.2s

TABLE VI
THE OPTIMAL VALUES OF Q

*1) **Machine Learning performance**:* In the table below

	Train set	Validation set	Test set
Accuracy score (G)	100%	99.44%	98.75%

TABLE VII
GAUSSIAN

	Train set	Validation set	Test set
Accuracy score (P)	100%	99.625%	99.25%

TABLE VIII
POLYNOMIAL

*2) **Optimization performance**:* In the table below are displayed

#iterations	$f(\alpha_0)$	$f(\alpha*)$	$m^* - M^*$
36	0	$-1.0963 \cdot 10^3$	$4.54 \cdot 10^{-13}$

TABLE IX
GAUSSIAN

#iterations	$f(\alpha_0)$	$f(\alpha^*)$	$m^* - M^*$
31	0	- 2.07 $\cdot 10^{-2}$	5.55$\cdot 10^{-15}$

TABLE X
POLYNOMIAL

III. SEQUENTIAL MINIMAL OPTIMIZATION

The SMO algorithm requires, at each step, the solution of a bi-dimensional quadratic problem, which can be solved in a closed analytical form. We are now going to briefly describe some of the steps of the algorithm. Nevertheless, in the lectures' slides they are explained in a more exhaustive way. At first, we select the bi-dimensional working set W^k such that:

$$W^k = \{ \hat{i} , \hat{j} \}$$

where

$$\hat{i} \in \{i \quad : \quad i \in arg \max_{i \in R(\alpha^k)} \{-(\nabla f(\alpha^k))_i y_i\}\}$$
$$\hat{j} \in \{j \quad : \quad i \in arg \min_{j \in S(\alpha^k)} \{-(\nabla f(\alpha^k))_j y_j\}\}$$

We can now define the direction of the step in the following way:

$$d_h = \begin{cases} y_{\hat{i}} & if \quad h=\hat{i} \\ -y_{\hat{j}} & if \quad h=\hat{j} \\ 0 & otherwise \end{cases} \qquad h=1,\dots,P$$

The step to move along d is then chosen as the minimum value between t^{max}_{feas} and t^*, which are defined as follows:

$$t^{max}_{feas} = min \left\{ \min_{i:d^k_i>0} d^k_i (C - \alpha^K_i) \quad , \min_{i:d^k_i<0} |d^k_i| \alpha^k_i \right\}$$

$$t^* = - \frac{\nabla f(\alpha^k)^t d^k}{(d^k)^T Q d^k}$$

Once we have found a descent, feasible direction and the relative optimal step, we can update both the gradient and the value of α and starting again from the first step (until we do not reach a certain sopping criterion such as a maximum number of iterations or the satisfaction of the KKT conditions).
In the following subsection the performances of the SMO algorithm applied to our case are displayed.
*1) **Machine Learning performances**:* In the table below

	Train set	Test set
Accuracy score	100%	98.75%

TABLE XI
GAUSSIAN

	Train set	Test set
Accuracy score	100%	99.25%

TABLE XII
POLYNOMIAL

*2) **Optimization performances**:* In the table below

#iterations	$f(\alpha_0)$	$f(\alpha*)$	$m^* - M^*$
5129	0	$-1.0963 \cdot 10^3$	$1.99 \cdot 10^{-13}$

TABLE XIII
GAUSSIAN

#iterations	$f(\alpha_0)$	$f(\alpha^*)$	$m^* - M^*$
3271	0	$-2.07 \cdot 10^{-2}$	$1.15 \cdot 10^{-14}$

TABLE XIV
POLYNOMIAL

IV. COMPARISON AMONG ALL THE IMPLEMENTED METHODS

Analyzing the behavior of the various methods described so far, the SVM^{light} seems to perform better than the other methods in terms of computational effort (it requires from 2 to 5 times less time of the other algorithms to find a solution). On the other hand, it can be way slower in the case in which Q_{WW} is not positive definite, since it has to start a second optimization routine (the one based on the PPD modification) from scratch.
The one conjugating in the best way speed of convergence and solidity in terms of asymptotic convergence is the SMO algorithm, which reaches satisfying results in a relatively short amount of time (with respect to the "batch" dual soft-SVM seen in section I). As a matter of fact, the slowest option is the first one, which does not leverages any kind of decomposition methods. For these reasons, we decided to apply a SVM^{light} procedure to train the multi-label classifier of the last part of the project, as explained in the following section.
In TABLE XV an overview of the results of each method implemented is provided, both in the case of Gaussian kernel and for the polynomial one.

	Hyperparameters			ML performance		Optimization performance	
Question	C	Gamma	q	Training Accuracy	Test Accuracy	KKT violation	Computational time
Q1	0.001	5000	1600	100%	98.75%	$2.27 \cdot 10^{-13}$	$\simeq 6s$
Q2	0.001	5000	70	100%	98.75%	$4.54 \cdot 10^{-13}$	$\simeq 0.32s$
Q3	0.001	5000	2	100%	98.75%	$1.9 \cdot 10^{-13}$	$\simeq 1s$

TABLE XV
FINAL GAUSSIAN KERNEL SVM RESULTS

	Hyperparameters			ML performance		Optimization performance	
Question	C	Gamma	q	Training Accuracy	Test Accuracy	KKT violation	Computational time
Q1	2	1000	1600	100%	99.25%	$1.60 \cdot 10^{-14}$	$\simeq 5s$
Q2	2	1000	100	100%	99.25%	$5.55 \cdot 10^{-15}$	$\simeq 0.32s$
Q3	2	1000	2	100%	99.25%	$1.15 \cdot 10^{-14}$	$\simeq 1s$
Q4	2	1000	100	100%	99.5%	n.d.	$\simeq 5s$

TABLE XVI
FINAL POLYNOMIAL KERNEL SVM RESULTS

*These results can slightly vary depending on the random seed which determines the division of the data set into training and test set.

V. MULTI-LABEL CLASSIFICATION

In the last section of the project we implemented a multi-label classifier, applying a One VS All (OVA) criterion. We chose to follow a One VS All instead of a One VS One approach since the second one requires a higher number of comparisons than the first one (in our specific case, the number of comparisons was exactly the same for both the methods, since the labels were 3, but we built a generalized version of a multi-label classifier, which can be applied to a generically large number of classes). Hence, we trained three different polynomial kernel SVM classifiers (having as C and γ the best hyperparameters found in Section I), which in this case, taking as input a certain pattern $x^{(i)}$, returned a value $p_l^{(i)} \in \mathbb{R}$ (obtained removing the sign() function from the standard -1, 1 classifier). We then selected the label l belonging to the labels set $\mathcal{M}$ having the maximum value of the prediction.

$$\hat{l}^{(i)} = argmax\{p_l^{(i)} \ , \ l \in \mathcal{M}\}$$

In figure [5] the confusion matrix on the test set for the multi-label classification is shown.

VI. Figures

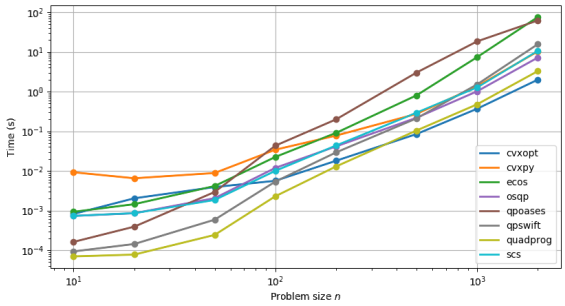

Fig. 1. Performances of qp_ solvers alternative solvers

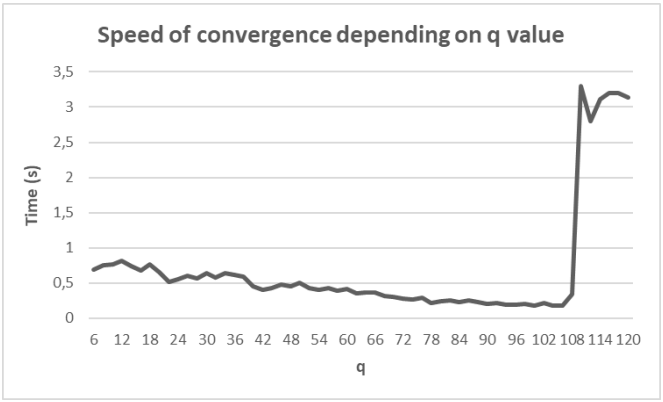

Fig. 2. Speed of convergence for Polynomial kernel SVM^{light}

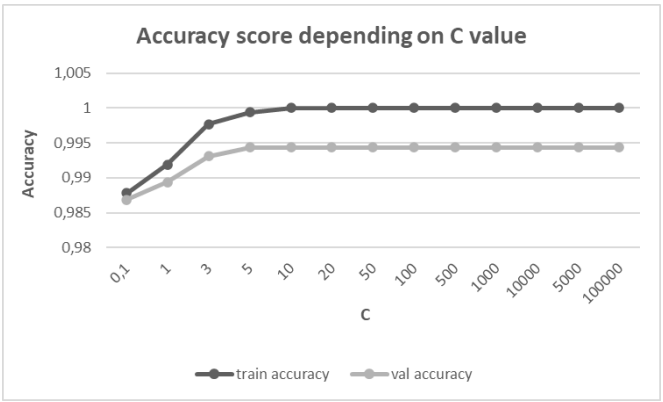

Fig. 3. Validation accuracy score depending on C (Gaussian Kernel)

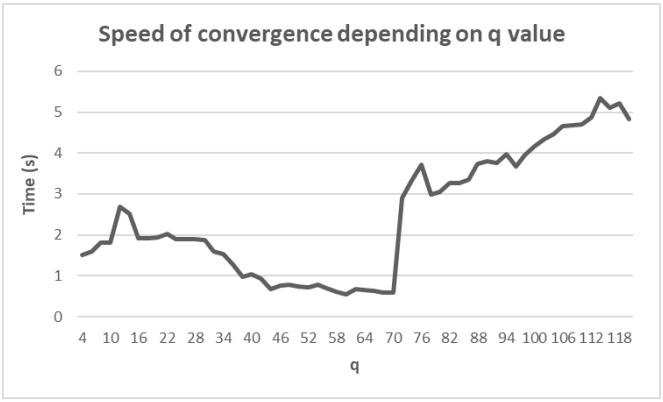

Fig. 4. Speed of convergence for Gaussian kernel SVM^{light}

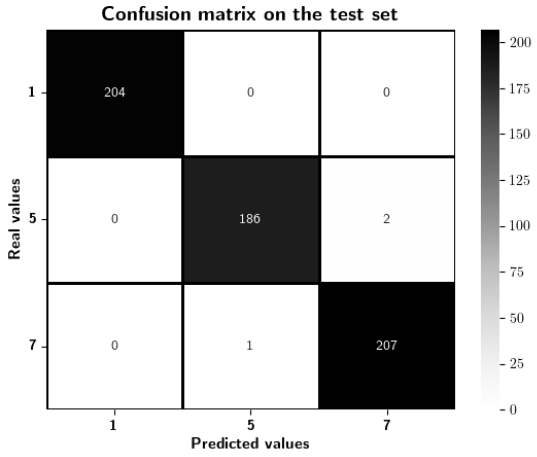

Fig. 5. Confusion matrix test

References

[1] C. J. C. Burges and D. J. Crisp, "Uniqueness of the svm solution," in *NIPS*, 1999.

[2] C. M. Bishop, *Pattern recognition and machine learning*. Springer, no. 7.

[3] L. Palagi and M. Sciandrone, "On the convergence of a modified version of svm light algorithm," *Optimization Methods and Software*, vol. 20, no. 2-3, pp. 317–334, 2005. [Online]. Available: https://doi.org/10.1080/10556780512331318209

[4] L. Palagi, "Lecture's slides," Fall 2022.