Optimization method for machine learning : Homework 2

Guadagnino Tiziano, Paoli Andrea, Alfano Domenico

January 8, 2018

Question 1 : Dual quadratic problem using standard Quadratic Programmimg algorithm.

In the first question we deal with the dual SVM training problem that can be expressed by the convex quadratic problem :

$$\begin{aligned} \underset{\alpha}{\text{minimize}} \quad & \alpha^T \cdot Y \cdot K \cdot Y \cdot \alpha - e^T \cdot \alpha \\ \text{subject to} \quad & y^T \cdot \alpha = 0 \\ & 0 \leq \alpha \leq C \cdot e \end{aligned} \tag{1}$$

Where $Y = \text{diag}(y)$, e is a vector of ones and K is the kernel matrix which elements are determined as[1] :

$$K_{ij} = k(x_i, x_j) \tag{2}$$

And in particular we choose as kernel function the RBF kernel:

$$k(x, z) = e^{-\gamma ||x-z||^2} \tag{3}$$

The aim of the problem is to determine the non-linear decision function:

$$y(x) = sign(\sum_{p=1}^{P} \alpha_p^* \cdot y^{(p)} \cdot k(x^{(p)}, x) + b) \tag{4}$$

[1] In particular in the following $Q = Y \cdot K \cdot Y$

In the optimization problem we have two hyperparameters, C that controls the trade-off between the maximization of the margin and the number of classification errors allowed on the training set and γ that is refered as the bandwidth of the RBF kernel. For the tuning of this parameters we choose to use a simple grid search procedure using the set of values :

$$\gamma = \{\ 0.001,\ 0.003,\ 0.01,\ 0.03,\ 0.1,\ 0.3,\ 0.5,\ 0.7\} \tag{5}$$

$$C = \{\ 0.001,\ 0.01,\ 0.1,\ 1,\ 10,\ 100\} \tag{6}$$

Evaluating for each couple of value the objective and the test accuracy.

For solving the convex quadratic problem (1) we choose to use the **cvxopt** library. This choice is made for two main reason, first the library is optimized for convex problem and internally use the interior point algorithm, that are prove to be very effective for this class of problems both from the computational point of view (need to solve a set of linear equation) and from the level of accuracy reached. Second the optimization routine of the library doesn't require any initial guess from the user and give us a constant result where instead **scipy.optimize**[2] give totally different numbers depending on the starting point that we give to the algorithm.

Results

The final setting of the parameters is $C = 1$ and $\gamma = 0.3$. With this values the algorithm converge after 6 iterations with a value of the objective of -118.467. In particular this value is influenced from both the value of the hyperparameters and the preprocessing phase that we do when we import the dataset that consist in subtracting the mean and dividing by the standard deviation each feature[3]. This phase is important because usually the features "lives" on different scales and this can have an influence in the outcome of a learning algorithm favoring feature with larger values.

From a machine learning point of view we obtain an accuracy of 0.893 on the test set and 1.0 on the training set. This result may seems to be an overfitting but we verify that with this setting we obtain the higher level of accuracy on the test set (that we interpretate as an higher capability of generalize), so we conclude that the training data are exactly separable by the non-linear SVM.

[2] In particular we refer to the Sequential Linear Least Square method, the only suitable for this class of problems

[3] We compute the mean and the standard deviation just for the training set and then apply the scaling also to the test set.

Question 2 : Decomposition Method

In this section we address the dual formulation of the SVM training problem using a decomposition method. As general schema this type of algorithm select at each iteration a subset of the variables, identified by the set of indices W, using some criteria and then solve the (reduced) optimization problem :

$$\begin{aligned} \underset{\alpha_W}{\text{minimize}} \quad & \alpha_W^T \cdot Q_{WW} \cdot \alpha_W + (\alpha_{\tilde{W}}^T \cdot Q_{\tilde{W}W} - e_W^T) \cdot \alpha_W \\ \text{subject to} \quad & y_W^T \cdot \alpha_W = -y_{\tilde{W}}^T \cdot \alpha_{\tilde{W}} \\ & 0 \leq \alpha_W \leq C \cdot e_W \end{aligned} \tag{7}$$

Where $\tilde{W}$ represent the set of indices that correspond to the variables that are not selected in that specific iteration. In particular the number of variables selected at each iteration $q = |W|$ and the selection criteria are crucial for determine the convergence of the algorithm and the number of iterations required for reaching the optimum.
In particular in our implementation we choose to use the SVM^{light} algorithm with $q = 10$ where we "modify" the selection rule as follow. We compute the value of

$$-\frac{\nabla f(\alpha^{(k)})_i}{y_i} \tag{8}$$

For every index in the set $R(\alpha^{(k)})$ and separately for $S(\alpha^{(k)})$, defined as:

$$R(\alpha) = \{i : \alpha_i = C, \quad y_i < 0\} \cup \{i : \alpha_i = 0, \quad y_i > 0\} \cup \{i : 0 < \alpha_i < C\} \tag{9}$$

$$S(\alpha) = \{i : \alpha_i = C, \quad y_i > 0\} \cup \{i : \alpha_i = 0, \quad y_i < 0\} \cup \{i : 0 < \alpha_i < C\} \tag{10}$$

Then we take $q/2$ indices from $R(\alpha^{(k)})$ which corresponds to the highest values of (8) and $q/2$ from $S(\alpha^{(k)})$ that correspond to the smallest ones. Essentially with this small modification we take the $q/2$ "most violating pairs" of the set.
We implement the selection criteria using just **numpy** and doing the optimization using **cvxopt** with a stopping criteria for the outer iterations defined as:

$$m(\alpha^{(k)}) - M(\alpha^{(k)}) \leq \epsilon \tag{11}$$

Where

$$m(\alpha^{(k)}) = \max_{i \in R(\alpha^{(k)})} -\frac{\nabla f(\alpha^{(k)})_i}{y_i} \tag{12}$$

$$M(\alpha^{(k)}) = \min_{j \in S(\alpha^{(k)})} -\frac{\nabla f(\alpha^{(k)})_j}{y_j} \tag{13}$$

And $\epsilon = 10^{-3}$

Results

With our implementation and stopping criteria the algorithm converge in 128 outer iterations with a total number of gradient evaluation of 665 reaching a value of the objective that is more or less identical to the one of the first question (≈ -118.4669)[4]. The similarity of the two results is confirmed by the fact that we obtain the exact same level of accuracy in both the train (1.0) and the test set (0.893).

[4]This is obviously due to the fact the problem is convex quadratic

Question 3 : Most Violating Pair decomposition method

In the third part we face the problem presented in (1) using the most violating pair algorithm, that is a particular decomposition method with $q = 2$. This means that our working set W^k is built only by two indices, $\{i^k, j^k\}$.
For the fact that in this case each subproblem have dimension 2 we can find the solution of (7) analytically without the needing an iterative solver. As consequence, as we will see in the results, this approach allows us to have a more efficient computation inside each iteration. On the other hand, for the fact that we optimize just two variable at the time, many more (outer) iterations are needed to reach the optimum.

The MVP (Most Violating Pair) is a decomposition method that, at each iteration, select the couple of indices $(i, j) \in R(\alpha^k) \times S(\alpha^k)$ defining the most quickly downhill direction between only two non-zero component, or that:

$$\min_{(s,t)\in R(\alpha^k)\times S(\alpha^k)} \nabla f(\alpha^k)^T d^{s,t} \tag{14}$$

In every iteration we define two sets, which contain respectively the maximum and the minimum of (8).

$$I(\alpha) = \left\{ i : i = \arg \max_{h\in R(\alpha^k)} -y_h(\nabla f(\alpha))_h \right\} \tag{15}$$

$$J(\alpha) = \left\{ j : j = \arg \min_{h\in S(\alpha^k)} -y_h(\nabla f(\alpha))_h \right\} \tag{16}$$

We can define the optimal solution as the value of α^k that satisfy the following condition:

$$m(\alpha^k) = \max_{i\in R(\alpha^k)} \{-y_i(\nabla f(\alpha))_i\} \leq \min_{j\in S(\alpha^k)} \{-y_j(\nabla f(\alpha))_j\} = M(\alpha^k) \tag{17}$$

The MVP rule consist in choosing a couple $(i, j) \in I(\alpha^k) \times J(\alpha^k)$ that define an admissible direction $d^{i,j}$ such that:

$$\nabla f(\alpha^k)^T d^{i,j} \leq \nabla f(\alpha^k)^T d^{s,t}, \quad \forall (s,t) \in R(\alpha^k) \times S(\alpha^k) \tag{18}$$

where $y_i \in \{-1, 1\}$ and $d^{i,j}$:

$$d_h^{i,j} = \begin{cases} \frac{1}{y_i} & if \quad h = i \\ -\frac{1}{y_j} & if \quad h = j \end{cases} \tag{19}$$

For a convex function, $d^{i,j}$ is a descent direction if and only if equation (18) is satisfied. Furthermore the indices i and j selected corresponds to the pair that

mostly violate the $K.K.T.$ conditions.
From an implementational point of view we use just **numpy** for performing both the selection and the optimization with a stopping criteria that is identical to the one used in the previous question.

Results

The optimum is reached in 647 iterations with a value of the objective equal (up to the 5 decimal) to the one of the first question (-118.467). We can see that the MVP required about 5 times the number of iterations needed in the SVM^{light} implementation (647 vs 128), even so the final computational time required by the MVP is 30% less (0.199 vs 0.287). As we said this is do to the fact that we can solve the subproblem in the MVP with an exact linesearch where instead we need to use the interior point algorithm of **cvxopt** in the decomposition method.
As overall result we can say that the full QP algorithm results to be the best performing algorithm for this particular dataset, both in terms of number of gradient evaluation and time[5]. This is not a general result because with a bigger dataset the computational effort for solving directly the whole problem doesn't scale well. The main bottleneck is the computation of the kernel matrix K, we notice that even with our small dataset this computation require a time comparable to the training time itself. Probably with a larger dataset a decomposition method, that doesn't require the full knowledge of K, has to be preferred. As a matter of fact all three method give us the same results in terms of objective value and training/test accuracy so they can to be considered equivalent from this point of view.

Comparison

Ex		settings	Training error	Test error	Time
Q1.1	Full QP	$C = 1\ \gamma = 0.3$	0.0	0.107	0.0505 s
Q1.2	Decomposed QP	$C = 1\ \gamma = 0.3$	0.0	0.107	0.2872 s
Q1.3	MVP	$C = 1\ \gamma = 0.3$	0.0	0.107	0.1989 s

Table 1: Comparison table between three implemented methods.

[5]This is also due to the fact that we implement from scratch part of the second algorithm and the whole third