Optimization Methods for Machine Learning - Fall 2017

Assignment # 2 - Support Vector Machines

Group 11: Conti Emmanuele and Rossini Valerio

1. About the data set

In this assignment we will implement training methods for Support Vector Machines applied to classification problems (our labels assume as values or +1 or 0). The data set is provided separately and corresponds to a health application. The shape of the data set is 466 rows and 28 columns: the last column represents the labels, while the remaining columns represent the features. First, using train_test_split from the library sklearn.model_selection, we divide our data set into training set (the 70% of our data) and test set (the remaining 30%). Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual feature do not more or less look like standard normally distributed data. So, about the features we have decide to standardize them with StandardScalar from sklearn.preprocessing; this estimator scales and translates each feature individually such that it is in the given range on the training set, i.e. between zero and one. We have also tried to normalize the data, instead of standardizing them, but we haven't chosen that approach because we obtained a similar result with the standardising. About the labels we have decided to change the 0 in -1 for a simple reason: when we will implement the decomposition method and the MVP algorithm, in the formula $-\frac{\nabla f(\alpha)_i}{y_i}$ if the labels assume values 0, we will have infinite as result since the 0 is present in the denominator.

2. Question 1

In this first question we need to write a program which implements the dual quadratic problem and use a standard QP algorithm for its solution.
Starting from:

$$\max_{\{\alpha \in R^P\}} -\frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P}\alpha_i\alpha_j y_i y_j k(x_i, x_j) + \sum_{i=1}^{P}\alpha_i$$

With:

$$\sum_{i=1}^{P}\alpha_i y_i = 0 \;\; and \; 0 \le \alpha_i \le C \quad with \; i = 1, \dots P$$

We have that the dual formulation gives a quadratic program (QP):

$$\begin{cases} \min_{\{\alpha \in R^P\}} \frac{1}{2}\,\alpha^T Q \alpha - I^T \alpha \\ with \; \alpha^T y = 0 \; and \; 0 \le \alpha_i \le C \end{cases}$$

Where:

$$Q_{ij} = y_i y_j k(x_i, x_j)$$

So, the first thing that we have done was the construction of the kernel and the matrix Q_{ij}
About the kernel we have two options:

- RBF kernel:

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

- polynomial kernel

$$k(x, y) = (x^T y + 1)^p$$

For the whole assignment we have decided to use the RBF kernel. We have done this choice because subsequently we have obtained a better result regarding the training and test error instead of the polynomial kernel.

For the choice of the optimum parameters, through a grid search we vary the parameters (C and γ) and we use a k-fold:

- C can assume one of the following values: 0.5, 1 and 10
- γ can assume one of the following values 0.0003, 0.0007, 0.003, 0.007, 0.03, 0.3, 0.5, 0.8 and 1

About the possibility of overfitting and underfitting, we can affirm that:

- Theoretically, by increasing the parameter p of the power of the kernel or gamma, we can shape a more complex non-linear boundary in the train set between negative and positive data points. This means that we should achieve overfitting of the train set as p or gamma increases, as we have a more flexible kernel.
- The value C is instead used to assess the training error in the primal problem. By increasing we make more relevant the penalty term in the primal objective function we are minimizing. This means that the more C increases, the more we will go in overfitting as we are forced to find a boundary able to classify correctly as many points as possible.

The figure that you can find at the end, highlight over/underfitting. The underfitting, in particular is possible to see its behaviour in the first case where C is equal to 0.50 where the training error is high due to the less complexity of the model. The other cases show instead the overfitting because where C and gamma increase, the training error goes to zero and the test error increases.

After writing the dual quadratic problem function and constraints, we use the python package scipy to run the SLSQP method (where SLSQP means Sequential Least Squares Programming). This method is a sequential least square programming algorithm which uses the Han-Powell quasi-Newton method.
We have also used a routine which uses the gradient of the objective function:

$$\nabla f(\alpha) = Q\alpha - e$$

About the SLSQP method, we have used the option "disp", that is present inside scipy.optimize.minimize, equal to True to print convergence messages ("Optimization terminated successfully" is printed in case of success) and we have also used the option "ftol" equal to 1e-6, i.e. the precision goal for the value of f in the stopping criterion.

For the prediction of the labels, we need first to compute w^* and b^*. About w^* we have that:

$$w^* = \sum_{i=1}^{P} y_i \alpha_i k(x, x_i)$$

Given w^* and any $0 \leq \alpha_i \leq C$, the scalar b^*can be determined using the complementarity conditions:

$$y_i \left(\sum_{j=1}^{P} y_j \alpha_j k(x_i, x_j) + b^* \right) - 1 = 0 \xrightarrow[isolating\ b^*]{} b^* = \frac{1}{y_i} - \sum_{j=1}^{P} y_j \alpha_j k(x_i, x_j)$$

We have created two functions to compute them: "compute_w" and "compute_b". Then, we have determined the non-linear decision function:

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

Also in this case we have created a function to the non-linear decision function, called “compute_pred”, that we will use to compute the error that assume the following form:

$$error = \frac{\text{number of incorrectly predicted instances}}{number\ of\ total\ instances}$$

The best values of the parameters C and γ, which allow us to have the minimum test error, are:

- C equal to 1
- γ equal to 0.1

With these two best parameters, we have also obtained the current function value, the iterations, the function evaluations, the gradient evaluations and the computational time that are displayed below:

```
---------------------(C = 1 and free parameter = 0.1 )---------------------

Optimization terminated successfully.    (Exit mode 0)
            Current function value: -95.9109871035
            Iterations: 26
            Function evaluations: 31
            Gradient evaluations: 26
Optimization terminated successfully.


------TEST-----------
Error on test 0.0857142857143
Accurancy on test 0.914285714286
------TRAIN-----------
Error on train 0.00306748466258
Accurancy on train 0.996932515337
--------------------------------
Elapsed time: 4.317441701889038

Lambda (first 10) founded with Gaussian Kernel
[ 1.          0.72261307  0.          0.          1.          1.          1.
  0.02381039  1.          0.18926439]
```

Regarding the performance from machine learning perspective the values of the accuracy on the training and test set are the following:

- accuracy training set equal to 0.996
- accuracy test set equal to 0.914

The accuracy was computed as:

$$accuracy = 1 - error = 1 - \frac{number\ of\ incorrectly\ predicted\ instances}{number\ of\ total\ instances}$$

3. Question 2

In this second point we need to:

- write a program which implements a decomposition method for the dual quadratic problem
- define the selection rule of the working set
- construct the subproblem at each iteration
- use a standard QP algorithm for its solutions

We have decided to implement the SVM^{light} algorithm.

If the dimension of the working set q is equal to 2, in a generic decomposition algorithm we need to resolve, for each iteration, a subproblem that assumes the following form:

$$\min q(\alpha_i, \alpha_j) = \frac{1}{2}\begin{pmatrix}\alpha_i & \alpha_j\end{pmatrix}^T \begin{pmatrix}q_{ii} & q_{ij}\\ q_{ji} & q_{jj}\end{pmatrix}\begin{pmatrix}\alpha_i\\ \alpha_j\end{pmatrix} - \alpha_1 - \alpha_2$$

Such that:

$$y_i\alpha_i + y_j\alpha_j = 0 \; and \; 0 \leq \alpha_h \leq C \qquad with \; h = i, j$$

For solving the quadratic minimization subproblem, we have used, as before, the python package scipy to run the SLSQP method.

For each iteration, we have given the subproblem matrix Q_{W_k} and the recomputed gradient of the subproblem given by:

$$\nabla_{\alpha_k} f(\alpha) = Q_{W_k}\,\alpha_{W_k} - e$$

The stopping rule that we have decided to use is the following:

- if $m \leq M + \varepsilon$, where ε is a constant greater or equal to zero, we decided to stop the algorithm printing the following message: "Optimality reached!!".

We have implemented the algorithm having the possibility to choose the value q (knowing that q must assume even values). At the end, we have chosen a q equal to 6.

About the choice of the working set, for each iteration, we have taken one index i inside the set I and one index j inside the set J where:

$$I(\alpha) = \left\{i: i = argmax_{\{h \in R(\alpha)\}} -\frac{(\nabla f(\alpha))_h}{y_h}\right\}$$

And

$$J(\alpha) = \left\{j: j = argmin_{\{h \in R(\alpha)\}} -\frac{(\nabla f(\alpha))_h}{y_h}\right\}$$

The pair of indices i and j, for each iteration, constitute the working set, i.e.: $W = \{i, j\}$

Regarding the performance from machine learning perspective the values of the accuracy on the training and test set are the following:

- accuracy training set equal to 0.996
- accuracy test set equal to 0.907

The current function value, the iterations, the function evaluations, the gradient evaluations, the computational time and other information are displayed below:

```
# iter:  87
index i selected:  [27, 134, 235] index j selected:  [211, 293, 143]
Optimization terminated successfully.    (Exit mode 0)
            Current function value: -0.486169560449
            Iterations: 5
            Function evaluations: 5
            Gradient evaluations: 5
Optimization terminated successfully.
Value of m:  0.487104227618 Value of M:  0.397893682549
------------------------------------------------------------

Optimality reached!!

Elapsed time total: 2.3982486724853516
Function evaluation total: 361
Gradient evaluation total: 348
```

About the lambda founded with SVM^{light}, the training error and the test error, we have the following results:

```
Lambda (first 10) founded with analytic solution of the SVM light
[ 1.          0.80308558  0.          0.          1.          1.          1.
  0.0622948   1.          0.15914737]

------TRAIN-----------
Error on train 0.00306748466258
Accurancy on train 0.996932515337
------TEST-----------
Error on test 0.0928571428571
Accurancy on test 0.907142857143
```

4. Question 3

In the last point, fixing q = 2, we need to implement a most violating pair (MVP) decomposition method which uses the analytic solution of the subproblems.
We need to state:

- performance from machine learning perspective
- optimization performance in comparison with the question 2.

Among violating pairs, it is possible to find a Most Violating Pair namely those indices (i, j) which define the steepest descent, namely such that:

$$\min_{\{st \in R(\alpha^k) x S(\alpha^k)\}} \nabla f(\alpha^k)^T d^{st}$$

The MVP rule consists in choosing a pair $(i^k, j^k) \in I(\alpha^k)\, x\, J(\alpha^k)$ which corresponds to choose a direction $d^{i,j}$ such that:

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

About the performance from machine learning perspective, we have:

```
# iter:  231
index i selected:  181 index j selected:  250
Indexes 181 250 selected are MVP
beta_star founded is:  0.126310873665
alpha_star founded is:  [ 0.69971936  0.88092057]
Elapsed time: 0.015626192092895508

-------------------------------------------------------------

# iter:  232
index i selected:  316 index j selected:  93
Optimality reached!!

Elapsed time total: 5.2995898723602295
```

About the lambda founded with analytic solution of MVP, the training error and the test error, we have the following results:

```
Lambda (first 10) founded with analytic solution of the MVP
[ 0.94579251  0.64429909  0.          0.          1.          1.          1.
  0.          1.          0.        ]

------TRAIN-----------
Error on train 0.00306748466258
Accuracy on train 0.996932515337
------TEST-----------
Error on test 0.0928571428571
Accuracy on test 0.907142857143
```

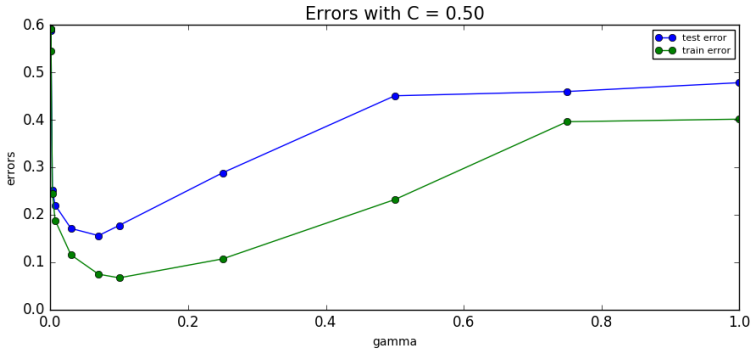
Regarding the performance from machine learning perspective the values of the accuracy on the training and test set are the following:

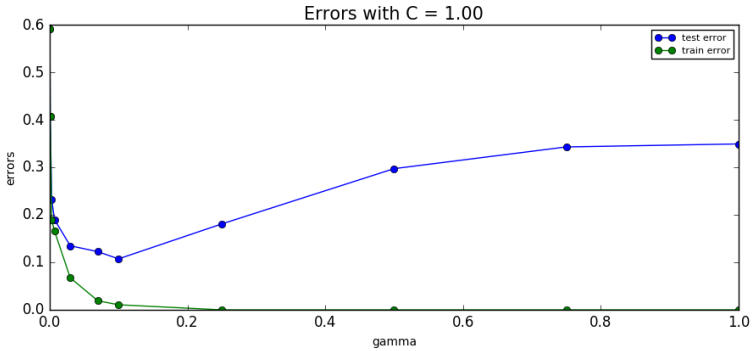
- accuracy training set equal to 0.996
- accuracy test set equal to 0.907

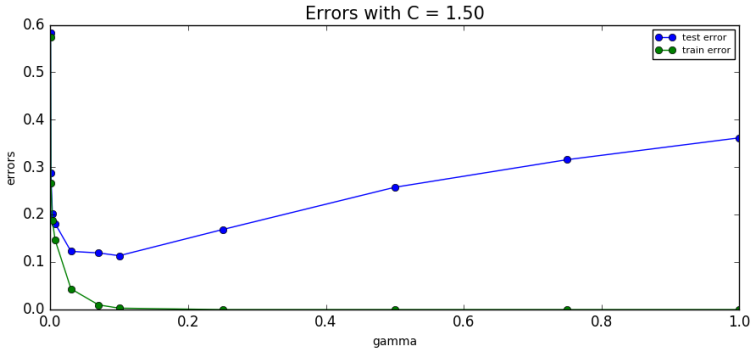
5. Table of the results

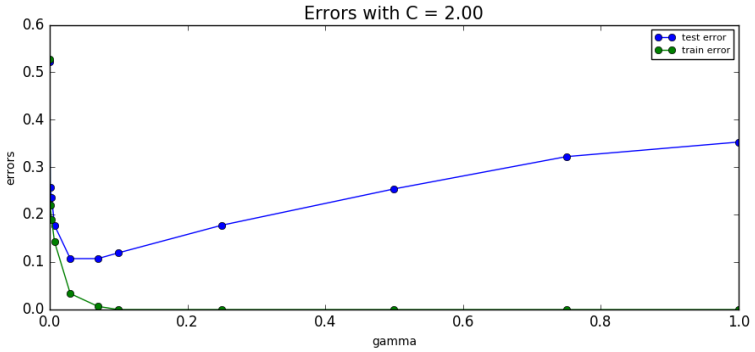
Ex	Method	Settings		Training error	Test error	Optimization time
Q1	Full QP	C = 5e-05	γ = 0.1	0.004	0.086	0.335
Q2	Decomposed QP	C = 5e-05	γ = 0.1	0.004	0.093	0.2.398
Q3	MVP	C = 5e-05	γ = 0.1	0.004	0.093	5.229

6. Figures


Errors with C = 0.50
test error
train error
errors
gamma
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
0.2
0.4
0.6
0.8
1.0


Errors with C = 1.00
test error
train error
errors
gamma
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
0.2
0.4
0.6
0.8
1.0


Errors with C = 1.50
test error
train error
errors
gamma
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
0.2
0.4
0.6
0.8
1.0


Errors with C = 2.00
test error
train error
errors
gamma
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
0.2
0.4
0.6
0.8
1.0