Optimization Methods for Machine Learning

Assignment # 3

1536242 - Paolo Tamagnini

29 December 2016

1 About the data

I chose the DATA SET 2, where we have handwritten digits from the US Postal Service. More in detail I chose the digits 0 and 8. I merged the files of those 2 digits, then I added the y column by setting the 0s with $y = 1$ and the 8s with y = -1. Then I permuted the rows and split the data-set so that 70% of the rows are in the train (1215 digits) and the remaining 30% in the test (521 digits).

2 Question 1

- I chose the polynomial kernel.
- To find the parameters C and p I tried all the combinations with C from $\{0.01, 0.1, 1, 10\}$ and p from $\{2, 3, 4, .., 9\}$. The best test error achieved is given by $C = 10$ and $p = 2$. With such values we are able to achieve a 0.9 % error over the test and at the same time to classify correctly all the instances in the train set. The errors are computed by the following function:

$$E(\hat{y}) = \frac{1}{2N}\sum_{i}^{N}|y_i - \hat{y}_i| = \frac{\#\ incorrectly\ predicted\ instances}{\#\ instances}$$

Below a table displaying the errors given by the different combination of C and p:

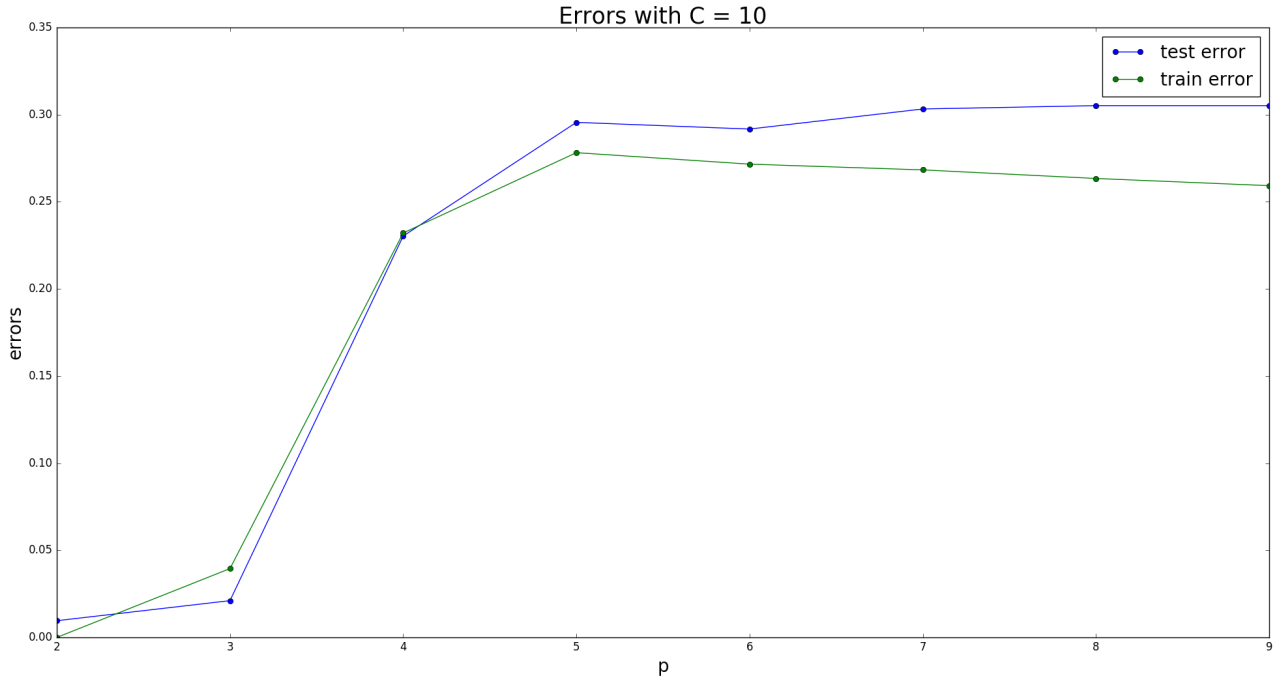
TRAIN ERROR:

	p: 2	**p: 3**	**p: 4**	**p: 5**	**p: 6**	**p: 7**	**p: 8**	**p: 9**
C: 0.01	0.000000	0.039506	0.046091	0.242798	0.270782	0.268313	0.263374	0.259259
C: 0.1	0.002469	0.039506	0.046091	0.274897	0.271605	0.268313	0.263374	0.259259
C: 1.0	0.000000	0.039506	0.213992	0.278189	0.271605	0.268313	0.263374	0.259259
C: 10.0	0.000000	0.039506	0.232099	0.278189	0.271605	0.268313	0.263374	0.259259

TEST ERROR:

	p: 2	**p: 3**	**p: 4**	**p: 5**	**p: 6**	**p: 7**	**p: 8**	**p: 9**
C: 0.01	0.013436	0.021113	0.026871	0.249520	0.291747	0.303263	0.305182	0.305182
C: 0.1	0.013436	0.021113	0.026871	0.293666	0.291747	0.303263	0.305182	0.305182
C: 1.0	0.013436	0.021113	0.207294	0.295585	0.291747	0.303263	0.305182	0.305182
C: 10.0	0.009597	0.021113	0.230326	0.295585	0.291747	0.303263	0.305182	0.305182

It follows a plot of the different values of the error fixing $C = 10$ and varying p.



Theoretically, by increasing the parameter p of the power of the kernel, 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 over-fitting of the train set as p increases, as we have a more flexible kernel. Such a kernel should be able to find a plane $w^\intercal x + b$ that perfectly separates the data points. This theory is not proved by what we have because we are already able to fit perfectly our train set and almost test set for $p = 2$.
The value C is instead used to asses the training error in the primal problem. By increasing we make more relevant the penalty term $C \sum \xi_i$ in the primal objective function we are minimizing. This means that for $0 < C < 1$ we take less into account the increase of the function caused by misclassified points. For $C > 1$ we instead amplify the penalty given by misclassified point. Theoretically this means that the more C increases, the more we will go in over-fitting as we are forced to find a boundary able to classify correctly as many points as possible, even the outliers within the train set. Like before, it is hard to see this in practice, as we are able to classify correctly all the digits in the train set without losing accuracy in the test set.

- After I acquired the dual quadratic problem function and constraints I could use the python package *scipy* to run the method SLSQP (Sequential Least Squares Programming). This method is a sequential least squares programming algorithm which uses the Han–Powell quasi–Newton method. I had to provide the problem in the following form:

$$\begin{aligned} min_\lambda \quad & \frac{1}{2} \ \lambda^\intercal Q \lambda - c^\intercal \lambda \\ & A\lambda \leq b \\ & Z\lambda = h \end{aligned}$$

$$q_{i,j} = y^i y^j k(x^i, x^j) \qquad c = \begin{pmatrix} -1 \\ -1 \\ .. \\ -1 \end{pmatrix}$$

$$A = \begin{pmatrix} -1 & 0 & .. & 0 \\ 0 & -1 & .. & 0 \\ .. & .. & .. & .. \\ 0 & .. & 0 & -1 \\ 1 & 0 & .. & 0 \\ 0 & 1 & .. & 0 \\ .. & .. & .. & .. \\ 0 & .. & 0 & 1 \end{pmatrix} \quad b = \begin{pmatrix} 0 \\ 0 \\ .. \\ 0 \\ C \\ C \\ .. \\ C \end{pmatrix}$$

$$Z = \begin{pmatrix} y_1 & 0 & .. & 0 \\ 0 & y_2 & .. & 0 \\ .. & .. & .. & .. \\ 0 & .. & 0 & y_P \end{pmatrix} \quad h = \begin{pmatrix} 0 \\ 0 \\ .. \\ 0 \end{pmatrix}$$

The optimization method was also provided of the gradient function:

$$\nabla f = Q\lambda + c$$

- The test error and train error achieved are the following:

```
Error on test: 0.00959692898273
Error on train: 0.0
```

- The infos regarding the oprimization are the following:

```
Elapsed time: 810.75 s
objective function minimum value: -0.000948474912636238
message: 'Optimization terminated successfully.'
# of function evaluations: 134
# of iterations: 37
# of gradient evaluations: 37
```

As you can clearly see it required quite a long time to optimize. Anyway, by changing C and p and by increasing the error of just around 1% (like in the case for $p = 3$ and $C = 10$), you can achieve a result almost as good in just 7.8 seconds. Below you can see a table that displays the elapsed time in seconds for all the different combination of C and p.

	p: 2	**p: 3**	**p: 4**	**p: 5**	**p: 6**	**p: 7**	**p: 8**	**p: 9**
C: 0.01	396.761	14.846	14.947	16.743	16.636	16.544	15.900	16.662
C: 0.1	330.024	15.378	15.009	15.506	15.578	15.321	15.864	14.898
C: 1.0	271.174	15.308	14.561	15.383	15.398	15.822	15.379	16.103
C: 10.0	810.750	7.841	7.368	7.603	7.660	7.465	7.678	7.670

- The problem is not 2-dimensional, therefore I cannot plot any boundary.

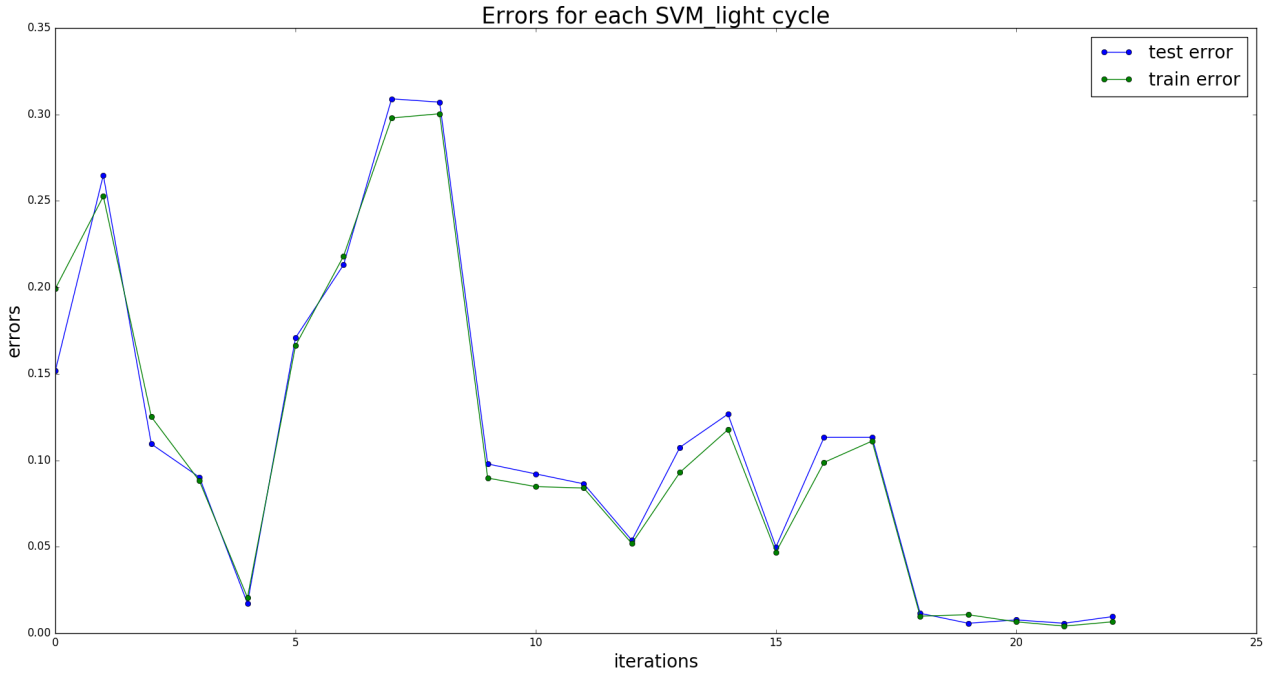
3 Question 2

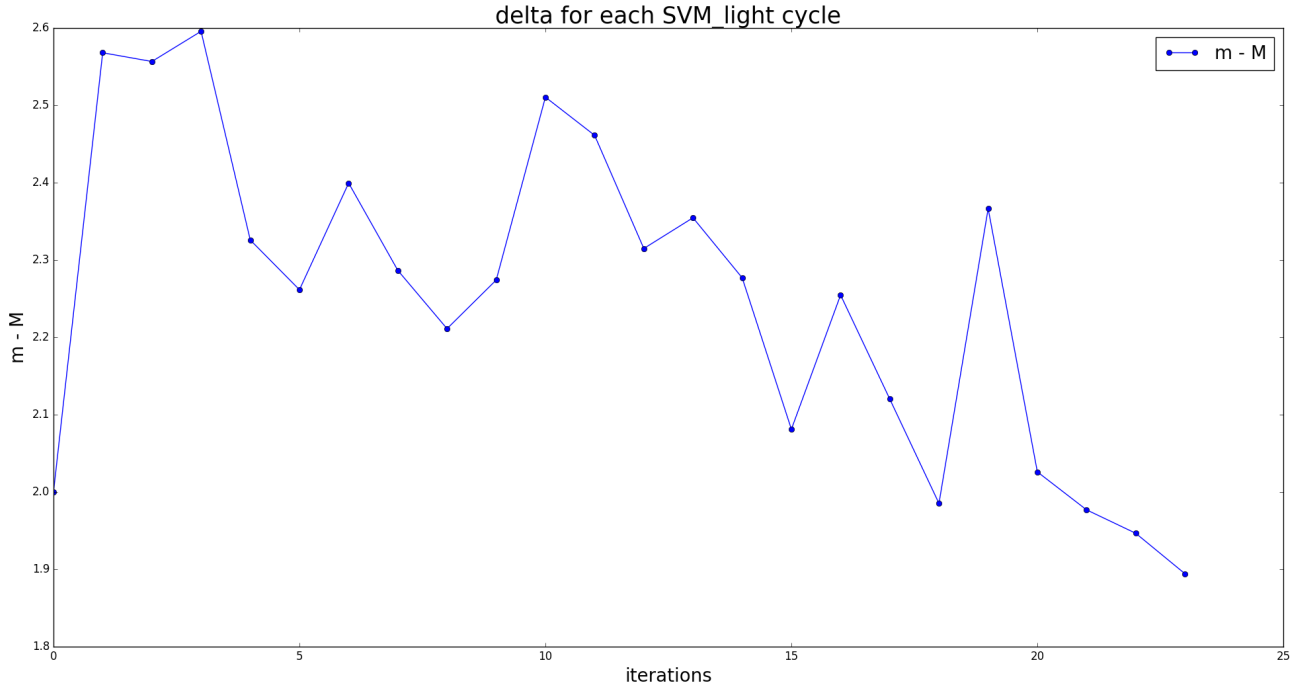
- Within the decomposition SVM_{light} of the dual problem with q = 2, I solved the single quadratic sub-problems always with the method SLSQP, using the python package *scipy*. For every optimization, I was giving the sub-problem matrix Q_{W_k}, Z_{W_k} and A_{W_k} and the vectors c_{W_k}, b_{W_k} and h_{W_k}. I also gave for each optimization

the recomputed gradient of the sub-problem given by $Q_{W_k}\lambda_{W_k} + c_{W_k}$. Every time, almost all those elements had to be recomputed given the current working pair W_k and, of course, instead of having the dimension given by the length of the train set, they had just dimension q=2. To find the working pair W_k I used the procedure of SVM_{light}, computing each time the gradient of the overall dual problem in the current and complete variable λ_k.

- The overall result errors, elapsed times, iterations and so on are displayed below, with also some plots that show the progress of such values for each iterations. An interesting situation is described by a not too small ϵ between m and M. Even though, in the last iteration, $m - M = \epsilon = 1.94 > 0$, I decided to stop the optimization because the test error is already really small (we are classifying incorrectly less that 1% of the instances in the test set). As you can see the test error is identical to the optimization in the question 1. It is quite good we are able now to achieve the same result in just 6.2 seconds.

```
Elapsed time: 6.24300003052 s
objective function minimum value: -0.000483191330675
Error on test set: 0.00959692898273
Error on train set: 0.00658436213992
# iterations: 23
# of gradient evaluations: 23
```





The following plot shows the function of the general dual problem evaluated in the different λ updated by the solutions of the sub-problems. As we can see there is a good decrease, sub-problem after sub-problem.