Martin Lundfall, Henri Bunting, Malte Siemers, Patrik Bey

Exercise sheet 09 - Machine Intelligence I

9.1

The primal problem for C-SVMs is solved by using the Karush-Kuhn-Tucker conditions, a generalization of Lagrange multipliers. We aim to minimize the following equation:

$$L_{(w,b,\{\lambda_\alpha\})} = f_0(w,b) + \sum_{\alpha=1}^{p} \lambda_\alpha f_\alpha(x) \tag{1}$$

Where f_0 is the function to be minimized, in our case:

$$f(w,b) = \frac{||w||^2}{2} + \frac{C}{p}\sum_{\alpha=1}^{p} \varphi_\alpha \tag{2}$$

and f_k are the inequality constraints, in our case:

$$f_\alpha(w,b) = (1-\varphi_\alpha) - y_T(w^T x + b) \tag{3}$$

$$f_\alpha(w,b) = -\varphi_\alpha \tag{4}$$

We minimize the KKT equation first with respect to w. Note that the slack variables φ_α are independent of w. This results in:

$$w = \sum_{\alpha=1}^{p} \lambda_\alpha y_T^{(\alpha)} x^{(\alpha)} \tag{5}$$

Then minimizing with respect to b we get the following constraint on λ_α:

$$0 = \sum_{\alpha=1}^{p} \lambda_\alpha y_T^{(\alpha)} \tag{6}$$

Putting these results back into the KKT equation, we attain:

$$L_{(x_k,\{\lambda_k\})} = \frac{1}{2}\sum_{\alpha,\beta=1}^{p} \lambda_\alpha \lambda_\beta y_T^{(\alpha)} y_T^{(\beta)} (x^{(\alpha)})^T x^{(\beta)} + \sum_{\alpha=1}^{p} \lambda_\alpha \tag{7}$$

We clearly see that minimizing this equation corresponds to maximizing the expression given in the exercise. The upper bound of the λ_α variables is given by the duality of the problem.

9.3 - C-SVM with standard parameters

a)

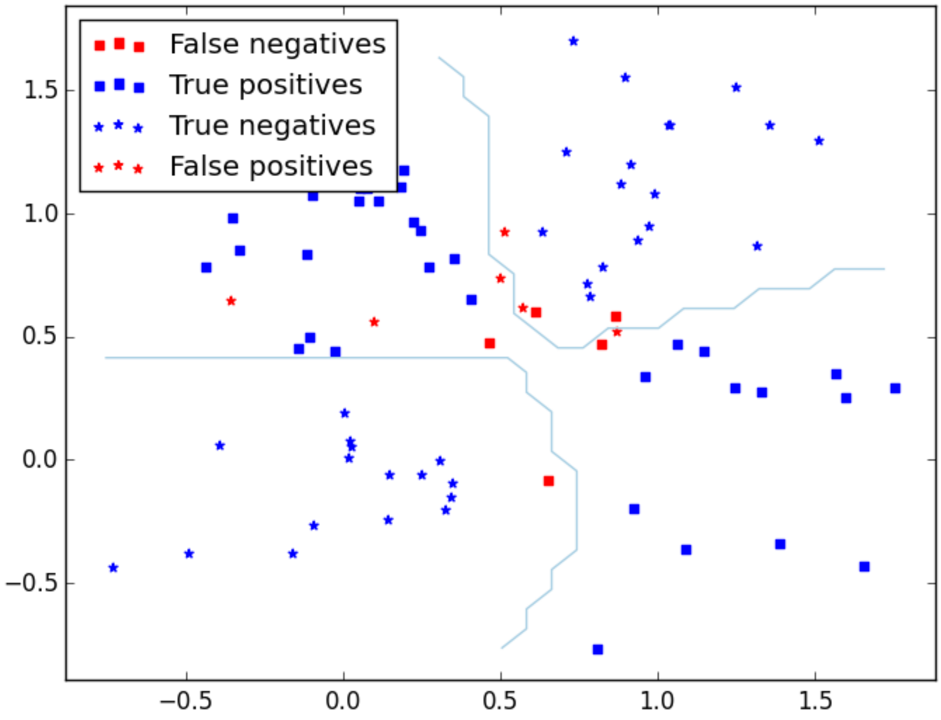
We used the scikit.learn python package to create and train a C-SVM as can be seen in the file 9.3.py.

b)

The mean error from the 0-1 loss function was 16.25%

c)

The plot of our classification with decision boundaries and false positives and false negatives highlighted:



9.4 - Parameter Optimization

(a)

Following the procedure for grid search as described in the guide we performed leave-one-out cross validation (LOOCV) on the training data using the following parameter range:

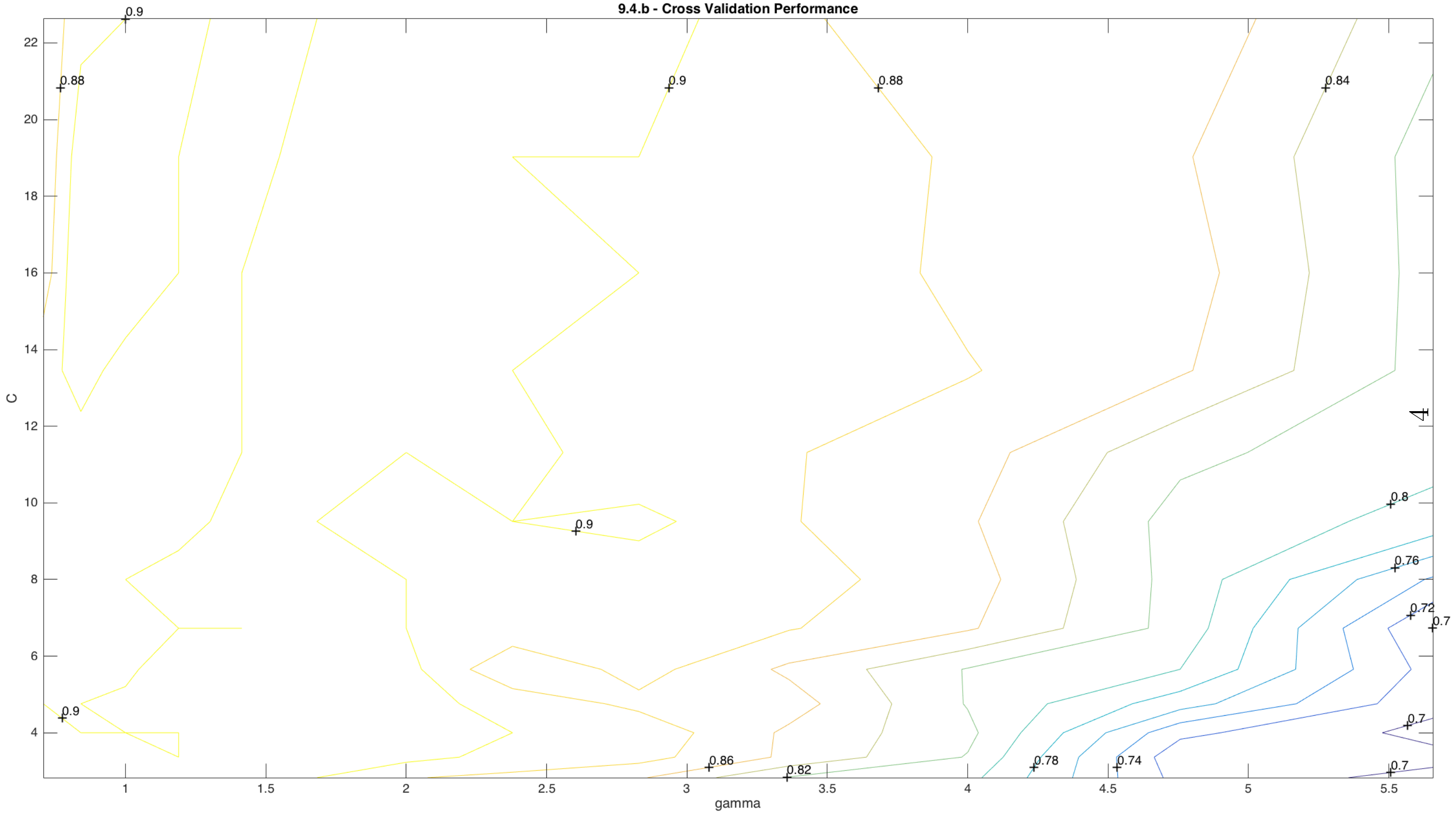
$\gamma = [2, 2^{-1}, 2^{-3}, 2^{-5}, 2^{-7}, 2^{-9}, 2^{-13}, 2^{-15}]$, $C = [2^{-5}, 2^{-3}, 2^{-1}, 2, 2^{3}, 2^{5}, 2^{7}, 2^{9}]$
After that we refined the grid to:
$C = [2^{1.5}, 2^{1.75}, 2^{2}, 2^{2.25}, 2^{2.5}, 2^{2.75}, 2^{3}, 2^{3.25}, 2^{3.5}, 2^{3.75}, 2^{4}, 2^{4.25}, 2^{4.5}]$
$\gamma = [2^{-.5}, 2^{-.25}, 2^{0}, 2^{.25}, 2^{.5}, 2^{.75}, 2^{1}, 2^{1.25}, 2^{1.5}, 2^{1.75}, 2^{2}, 2^{2.25}, 2^{2.5}]$
The parameter we obtained from the second grid search were $\gamma = 0.8408$ and $C = 16$. On the following pages the cross validation performance and the mean training classification for those parameters are shown, as well as the test data with the classification boundaries.

(c)

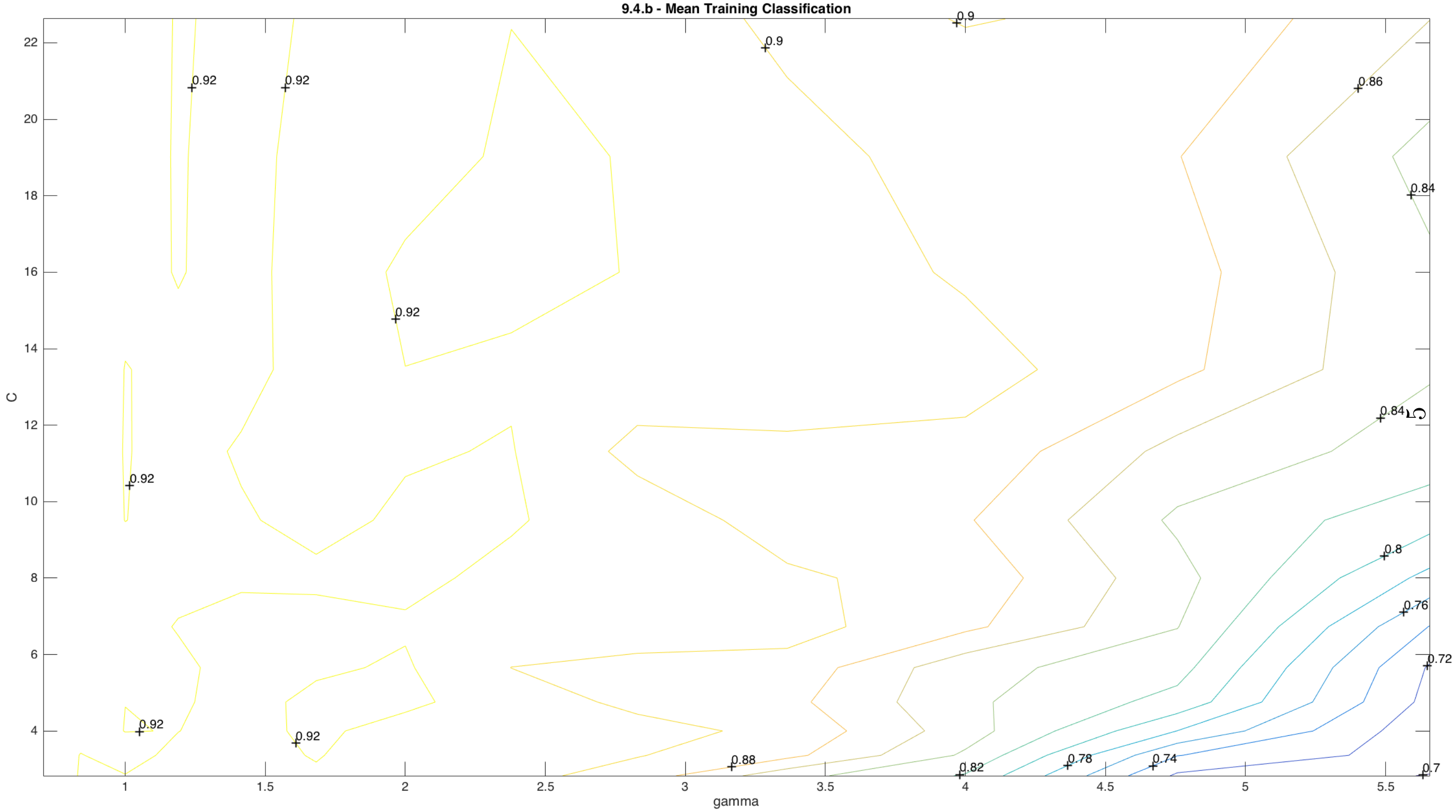
The classification error was then 16.25% and the CSVM used 42 supporting vectors.

(e)

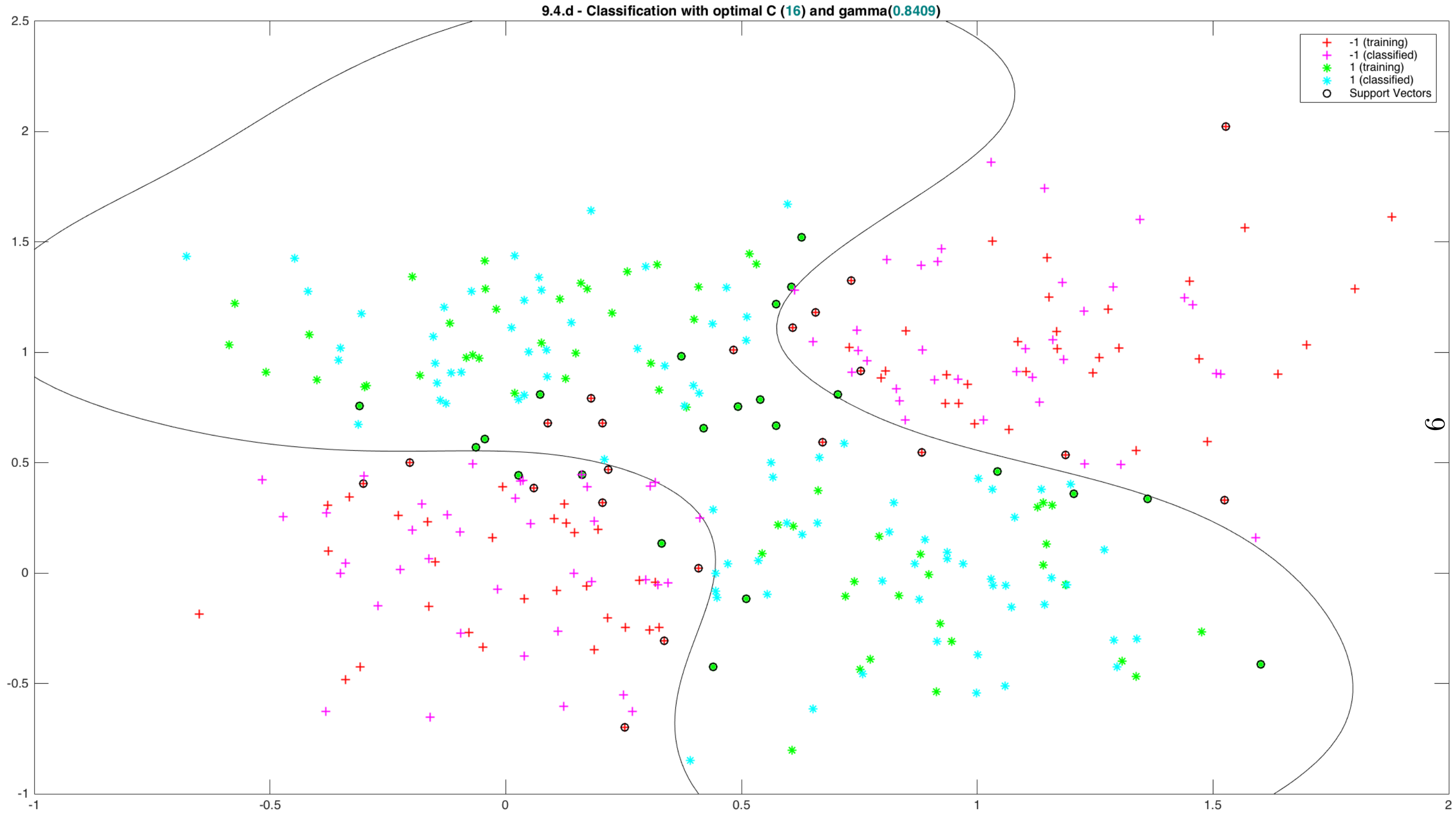
The classification error is the same as in our results with the standard parameters from the python package. The python package had to use 58 supporting vectors. Visually, the cross validated results show as well as the standard parameter results no signs of overfitting. We couldnt find a set of parameters, that significantly improved the results.



9.4.b - Cross Validation Performance



9.4.b - Mean Training Classification


9.4.d - Classification with optimal C (16) and gamma(0.8409)
-1 (training)
-1 (classified)
1 (training)
1 (classified)
Support Vectors
2.5
2
1.5
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1
1.5
2