Homework 2 – Machine Learning (2018/19)

Angelo Laudani s253177

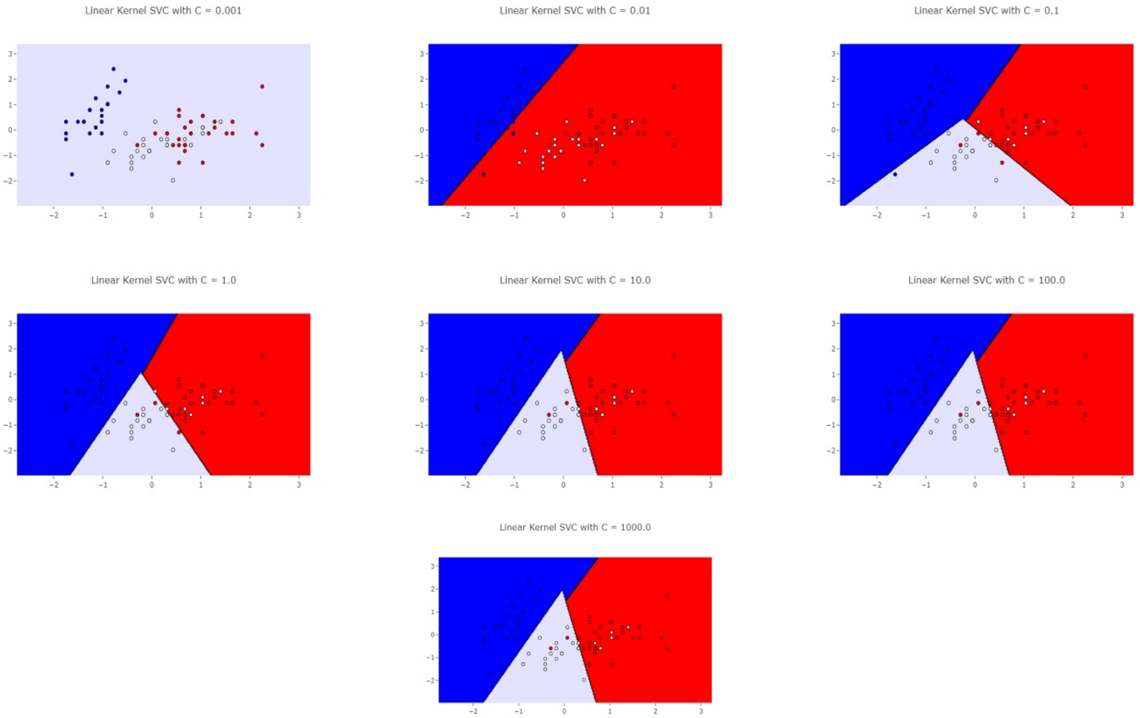
SVM

Introduction

Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. We plot each data item as a point in n-dimensional space (*with n features*) with the value of each feature being the value of a particular coordinate. The classification is performed by finding the hyper-plane that differentiate the classes very well.

1 – Linear SVM

Our goal is to correctly classify the “Iris Dataset” using a SVM classifier. After proper data standardization and splitting into training, validation and test set (with proportions 5:2:3) we train a linear SVM on the training set.



Linear Kernel SVC with C = 0.001

Linear Kernel SVC with C = 0.01

Linear Kernel SVC with C = 0.1

Linear Kernel SVC with C = 1.0

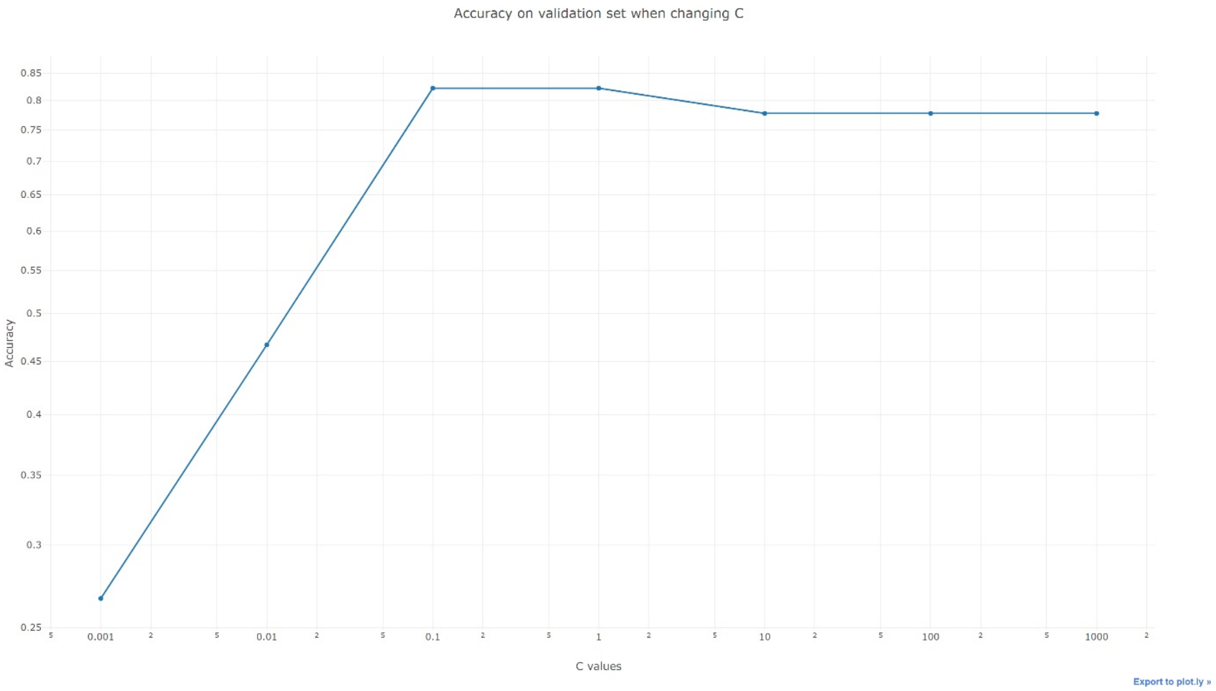
Linear Kernel SVC with C = 10.0

Linear Kernel SVC with C = 100.0

Linear Kernel SVC with C = 1000.0

C is the parameter for the soft margin cost function, introduced in the field of SVM by Cortes and Vapnik in 1995. The C parameter tells the SVM optimization how much we want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane, for a very small value of C the optimizer will to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

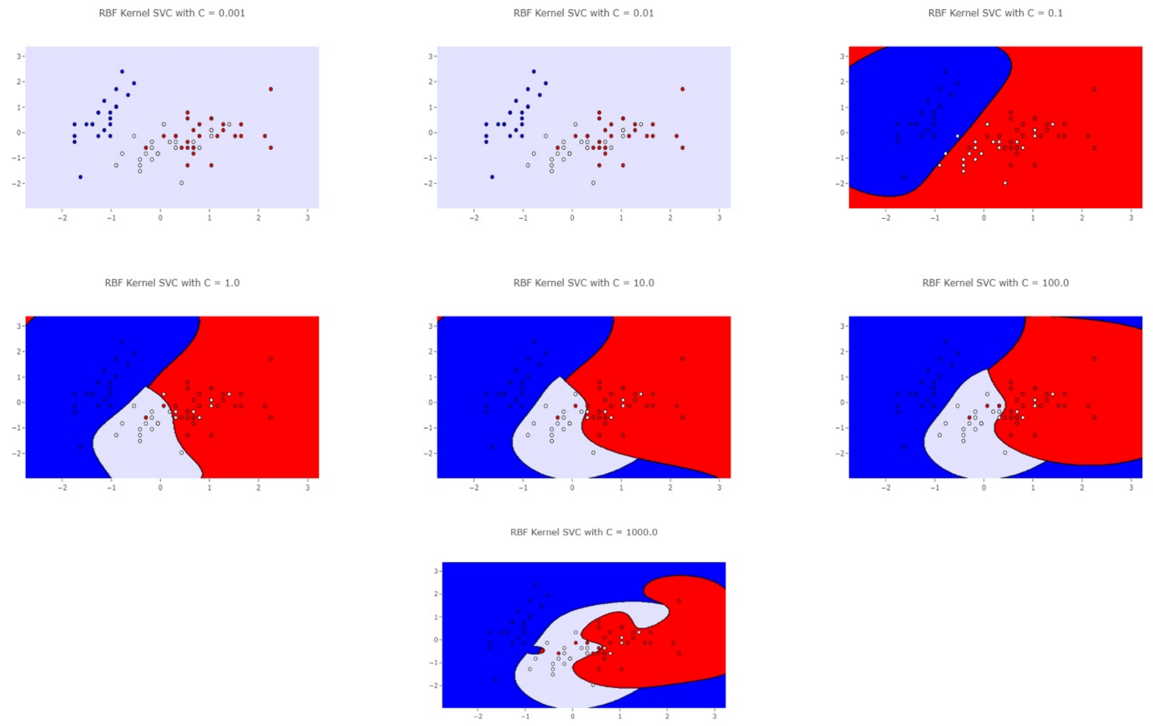
We can find the best C evaluating our classifier on the validation set:



With a C = 0.1 we have an accuracy on the validation set of **82%**, using this parameter on the test set will score a **80%** in accuracy, slightly less than on the validation set (which has less features), but still a good result.

2 – RBF Kernel SVM

Using the “Kernel Trick” we can transform the data into another dimension that has a clear dividing margin between classes of data. Kernel functions enable the classifiers to operate in a implicit high-dimensional feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. One of the kernels that we can use is the RBF, *radial basis function kernel*.



RBF Kernel SVC with C = 0.001

RBF Kernel SVC with C = 0.01

RBF Kernel SVC with C = 0.1

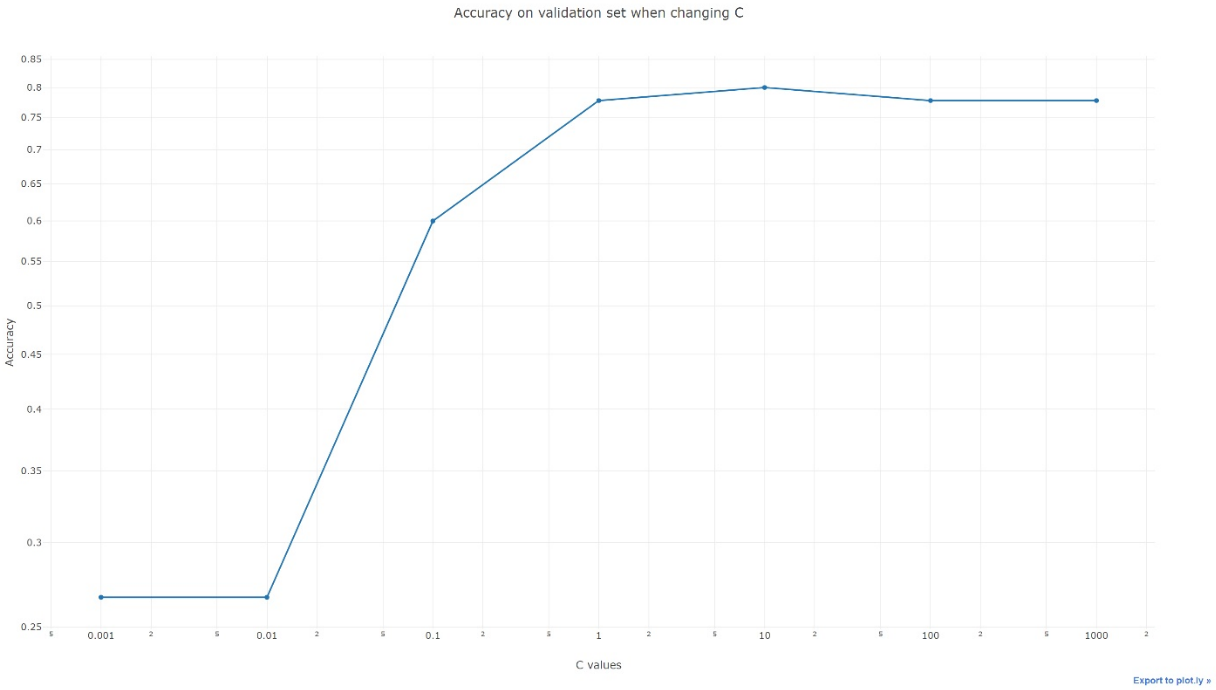
RBF Kernel SVC with C = 1.0

RBF Kernel SVC with C = 10.0

RBF Kernel SVC with C = 100.0

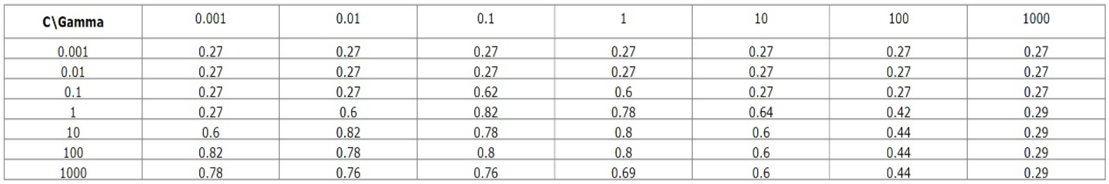
RBF Kernel SVC with C = 1000.0

This kernel makes curving the margins around the features possible, this is translated into a much better accuracy for problems with a strong non-linear nature. The classifier, with C = 10 has its best accuracy at **80%** on the validation set, and the same accuracy on the test set with the specified C parameter.



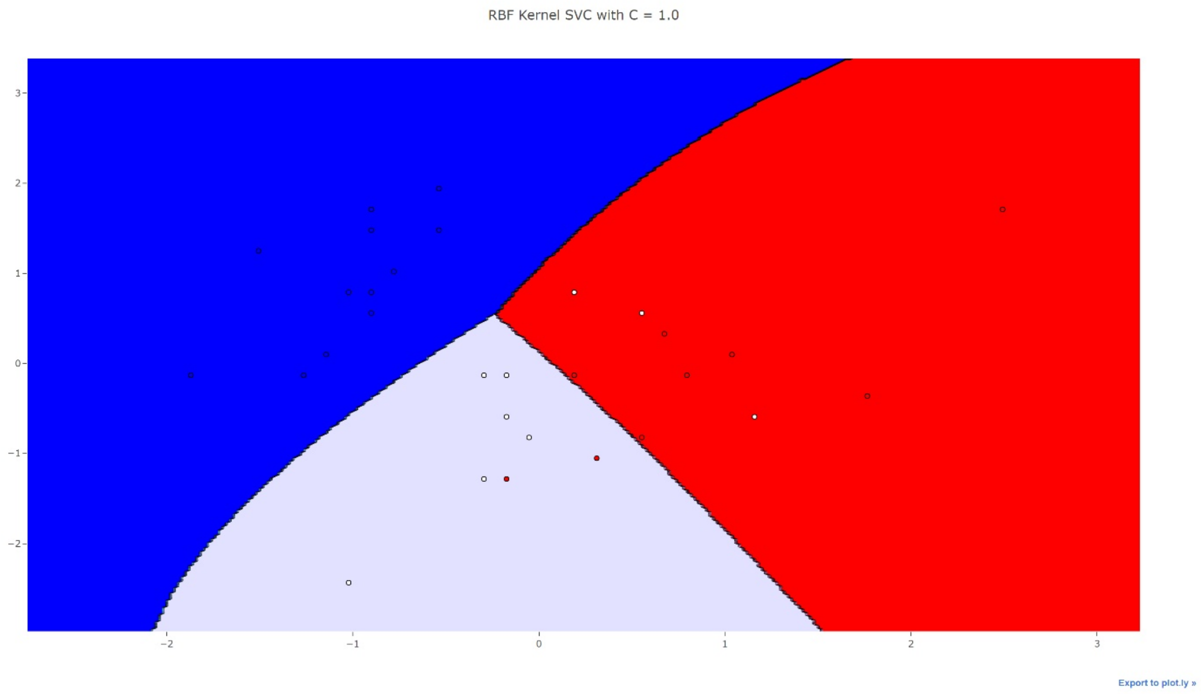
Accuracy on validation set when changing C

In a RBF Kernel SVM, C is not the only tuneable parameter. Gamma (γ) is the free parameter of the Gaussian radial basis function. Changing both those parameters on the validation set will produce the following results:



C\Gamma	0.001	0.01	0.1	1	10	100	1000
0.001	0.27	0.27	0.27	0.27	0.27	0.27	0.27
0.01	0.27	0.27	0.27	0.27	0.27	0.27	0.27
0.1	0.27	0.27	0.62	0.6	0.27	0.27	0.27
1	0.27	0.6	0.82	0.78	0.64	0.42	0.29
10	0.6	0.82	0.78	0.8	0.6	0.44	0.29
100	0.82	0.78	0.8	0.8	0.6	0.44	0.29
1000	0.78	0.76	0.76	0.69	0.6	0.44	0.29

We the best result on the validation set with C = 1 and γ = 0.1, with an accuracy of **82%** on the validation set and one of **83%** on the test set with the following boundaries:

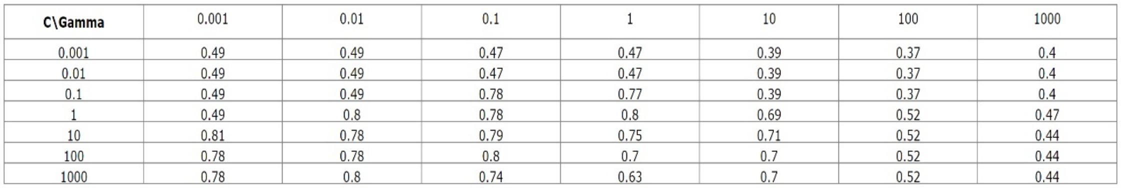


3 – K-Fold

After merging the validation and test set, we perform a new grid search for the best parameters of our RBF Kernel SVM, but this time using a K-fold cross validation.

This approach randomly divides the training set into k groups, and for each group takes the group as a hold out, trains the model on the remaining groups and tests it on the one left out. The score is retained, and the next group is selected as a hold out.

Using cross validation for the grid search of C and γ produces the following results:



C\Gamma	0.001	0.01	0.1	1	10	100	1000
0.001	0.49	0.49	0.47	0.47	0.39	0.37	0.4
0.01	0.49	0.49	0.47	0.47	0.39	0.37	0.4
0.1	0.49	0.49	0.78	0.77	0.39	0.37	0.4
1	0.49	0.8	0.78	0.8	0.69	0.52	0.47
10	0.81	0.78	0.79	0.75	0.71	0.52	0.44
100	0.78	0.78	0.8	0.7	0.7	0.52	0.44
1000	0.78	0.8	0.74	0.63	0.7	0.52	0.44

With the best C = 10 and γ = 0.01 with a score in accuracy of **81%** and an accuracy of **77%** on the test set. We can notice how there is a general increase in accuracy values for each combination on C and γ, but the high values of the previous results without cross validation are lower, which does not mean necessarily "worst", the results are now probably more accurate and with less overfitting in the training phase.