Homework 4: SVM
class in “Machine Learning”, Fall 2016/17

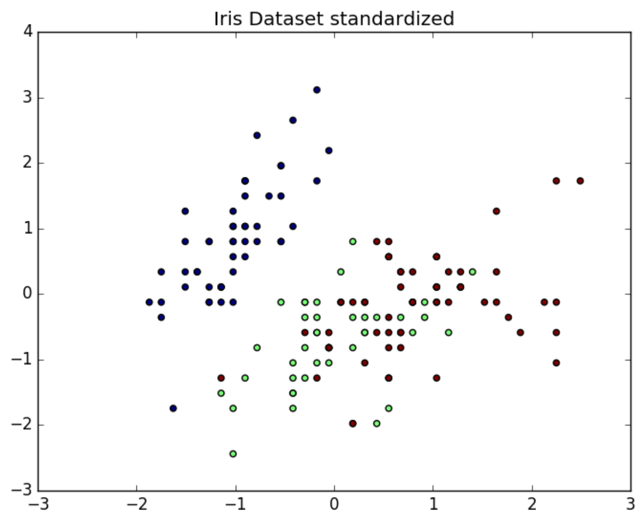
Marco Favorito
Master of Science in Engineering in Computer Science
Department of Computer, Control, and Management Engineering
University of Rome “La Sapienza’
favorito.1609890@studenti.uniroma1.it

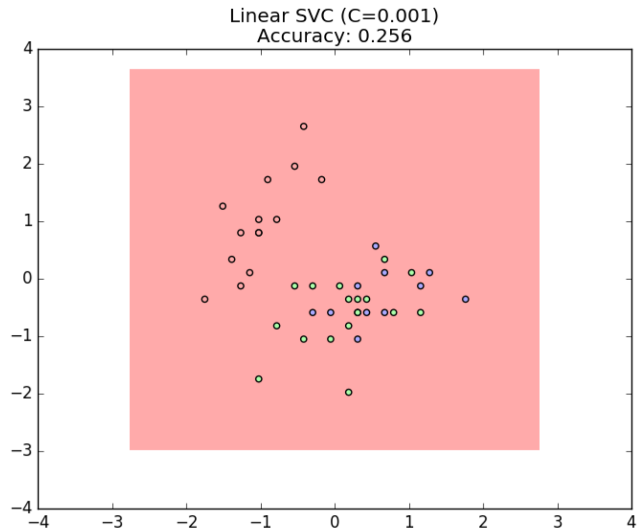
14 November 2016

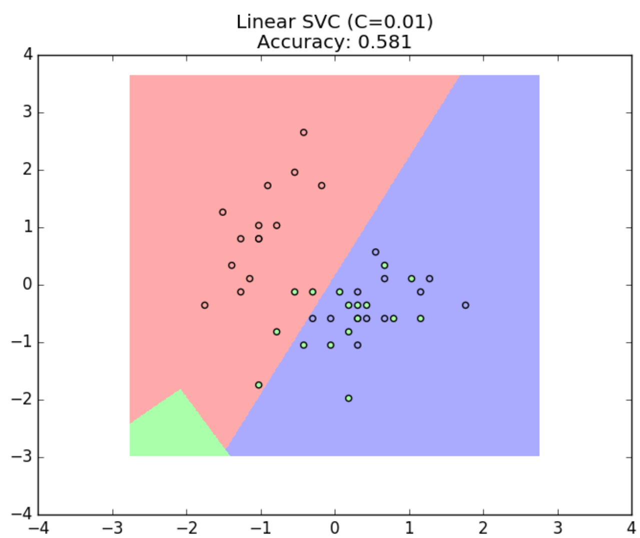
Contents

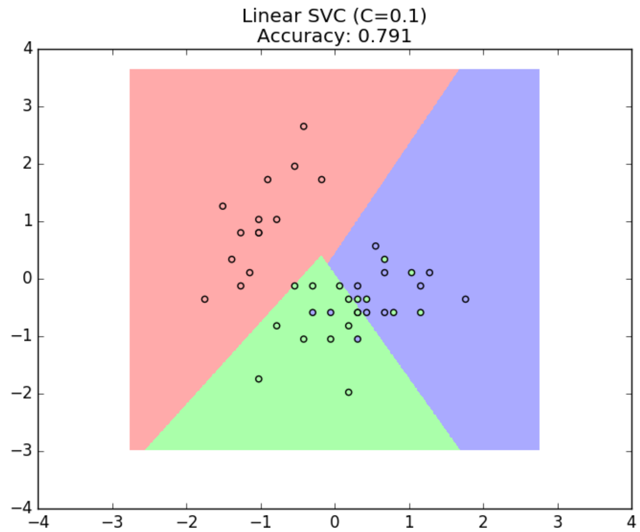
1 LinearSVM

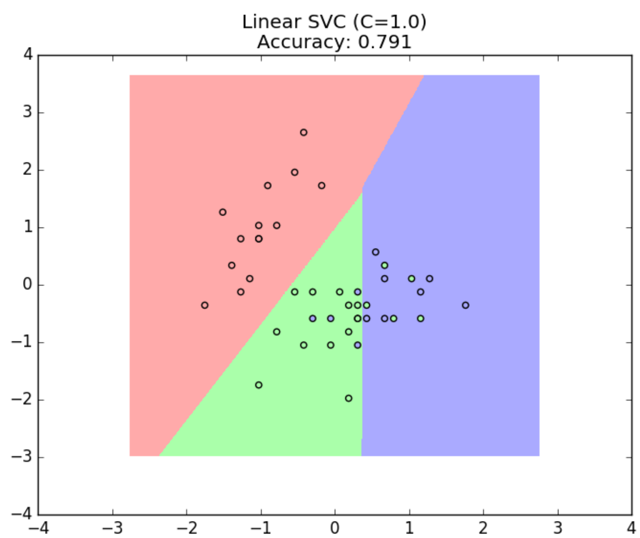
First of all I standardized data and splitted in train, validation and test set. Then I trained, on standardized data, the model with values of C from 10^{-3} to 10^{3} and for each of them, I reported plot of decision boundaries on validation set and its accuracy.

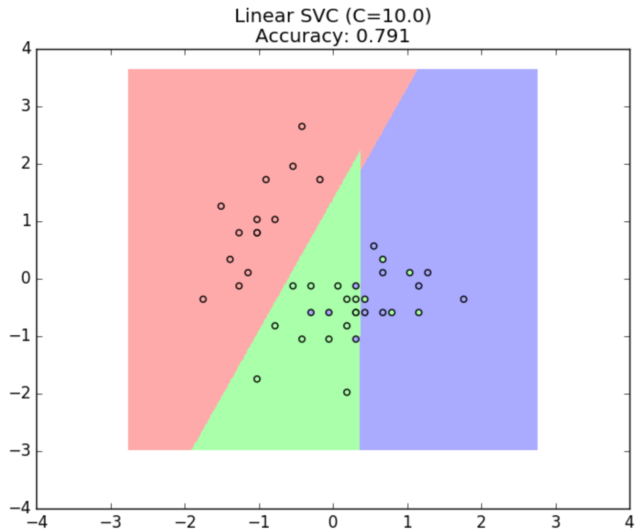


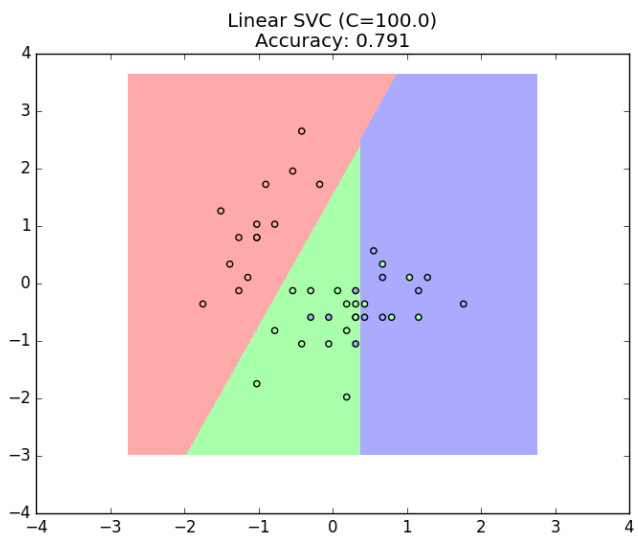

Linear SVC (C=0.001)
Accuracy: 0.256

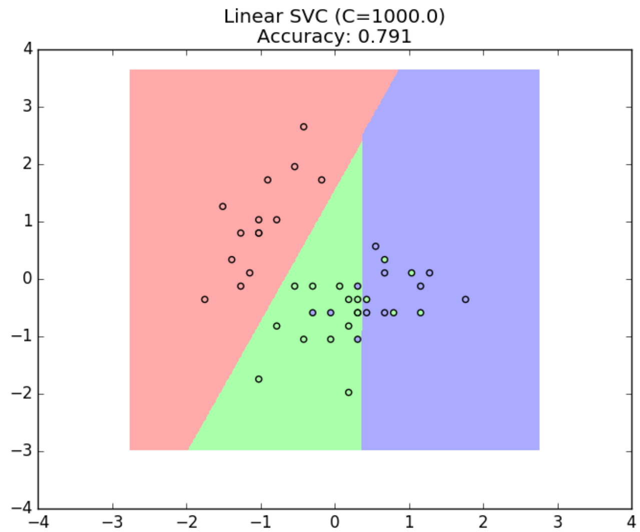

Linear SVC (C=0.01)
Accuracy: 0.581


Linear SVC (C=0.1)
Accuracy: 0.791

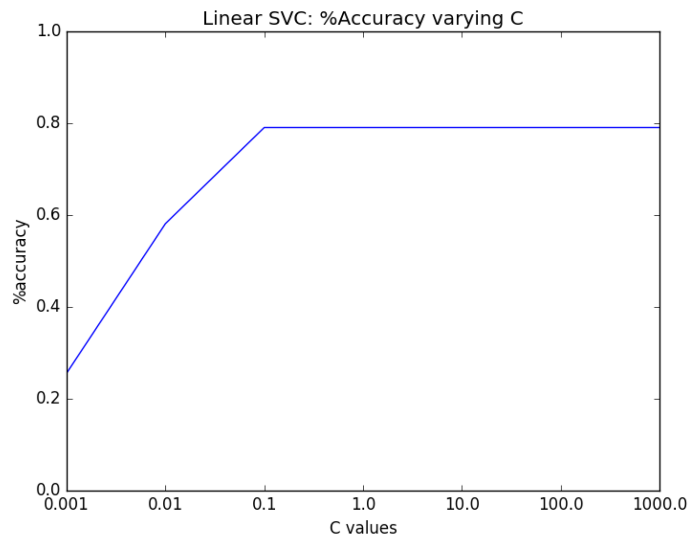

Linear SVC (C=1.0)
Accuracy: 0.791


Linear SVC (C=10.0)
Accuracy: 0.791
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

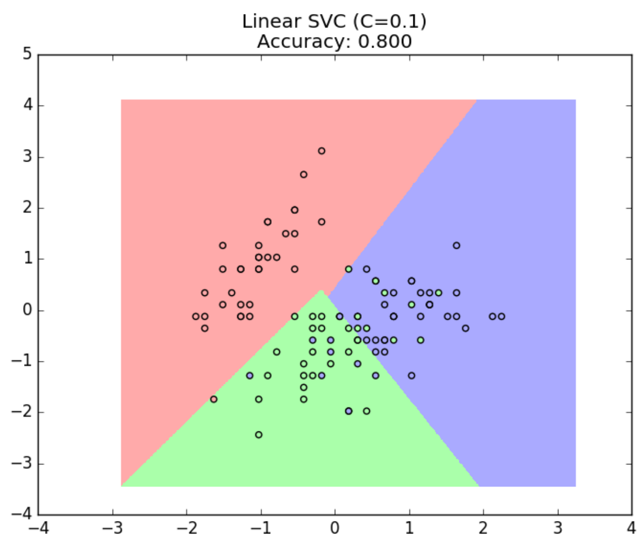

Linear SVC (C=100.0)
Accuracy: 0.791
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4


Linear SVC (C=1000.0)
Accuracy: 0.791
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

The following plot shows how accuracy change on varying C.



And this is the plot with the best C value (in this case, 0.1) on test set, with an accuracy of 80% :

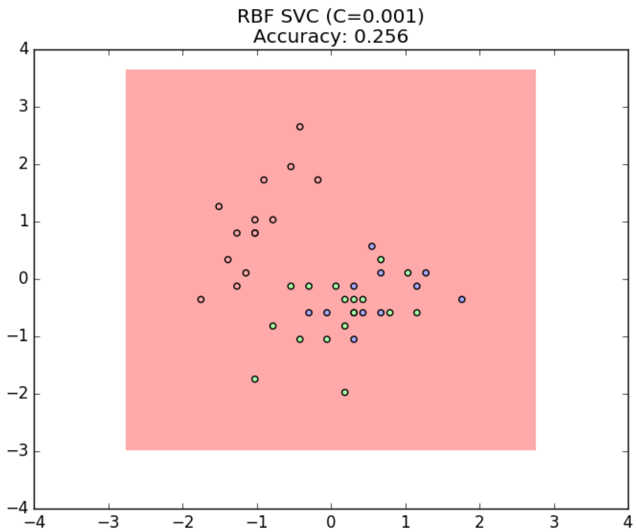

Linear SVC (C=0.1)
Accuracy: 0.800

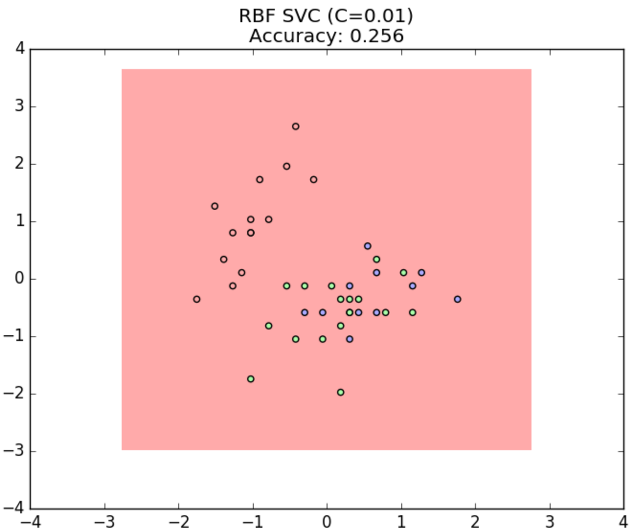
On these plots, we can notice how decision boundaries change in function of C. Indeed, C is a regularization parameter that controls the trade off between biasing (underfitting) and variance (overfitting). For $C = 0.001$ we have a very bad classifier: it classifies all useful sample space to one class (red class). As C increases, decision boundaries become quite good in separation of sample groups; in other words, give more importance to data.

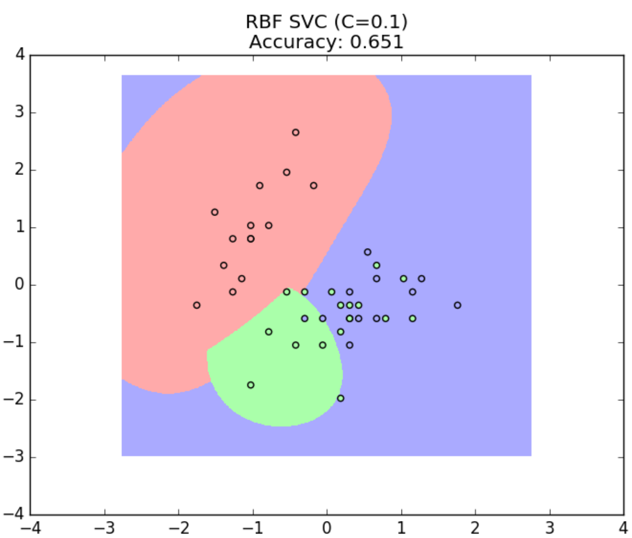
2 RBF Kernel

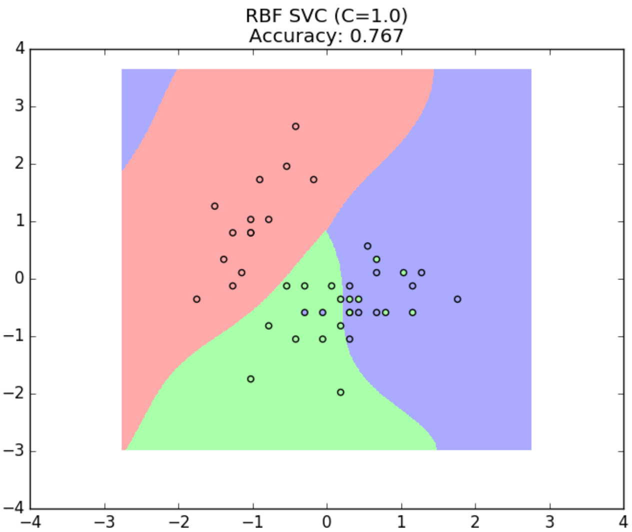
2.1 Varying C parameter on RBF

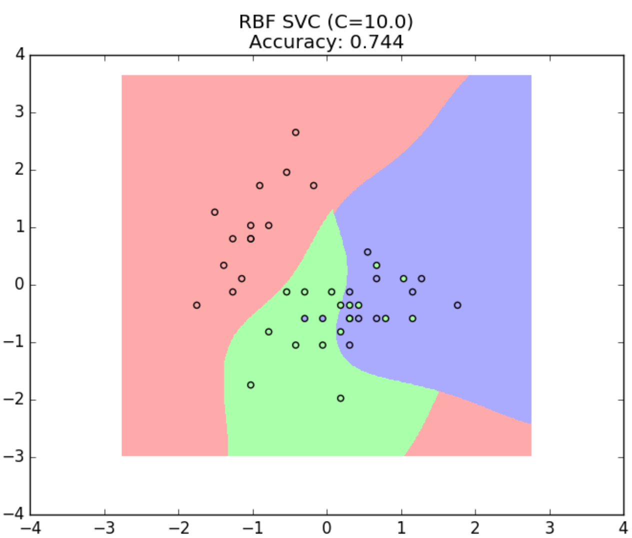
I repeated the same operations, I varied C and plotted decision boundaries:

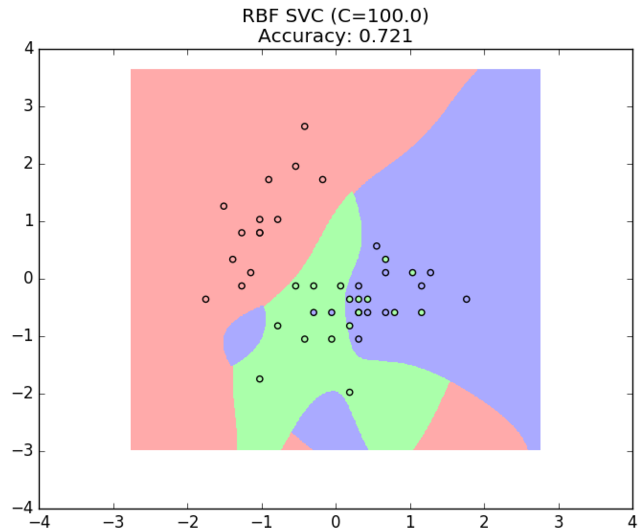


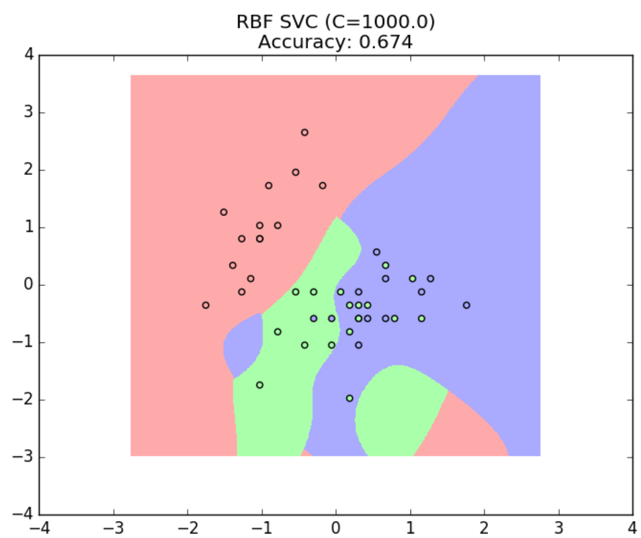

RBF SVC (C=0.01)
Accuracy: 0.256


RBF SVC (C=0.1)
Accuracy: 0.651

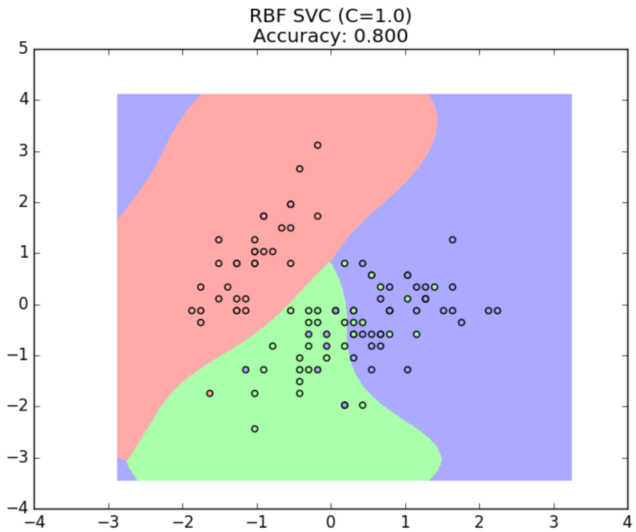

RBF SVC (C=1.0)
Accuracy: 0.767


RBF SVC (C=10.0)
Accuracy: 0.744

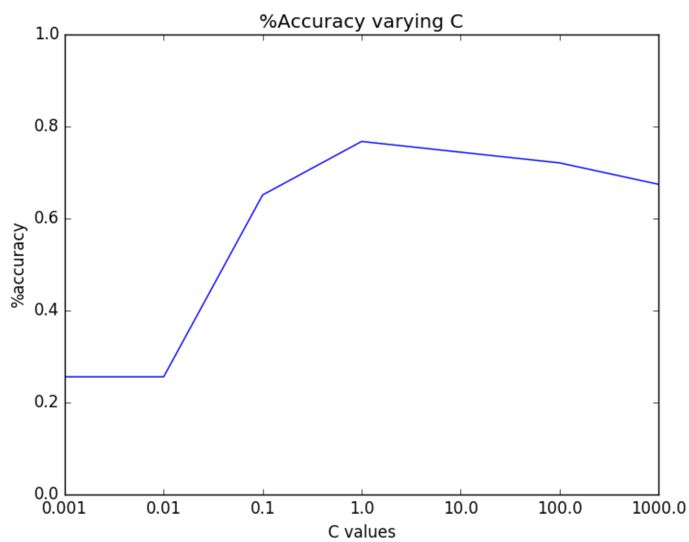

RBF SVC (C=100.0)
Accuracy: 0.721


RBF SVC (C=1000.0)
Accuracy: 0.674

And this is the best model found, with $C = 1$ and accuracy equal to 80%:



There are a lot of differences in decision boundaries, with reference to linear kernel.Due to its mathematical form, RBF kernel can find non-linear curves to define class decision region. RBF is the best in cases where there is no knowledge on the data distribution.
Parameter selection is crucial, since RBF is subjected to overfit data, as we can see where $C = 100$ and $C = 1000$. Here we can see how varies accuracy over C:

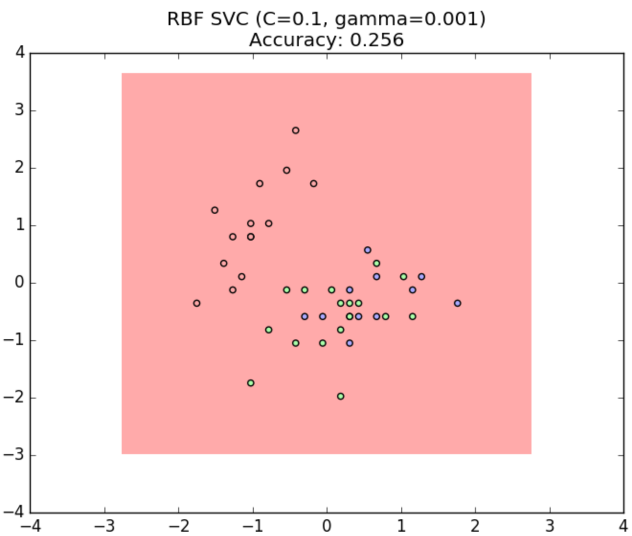

%Accuracy varying C
%accuracy
C values
1.0
0.8
0.6
0.4
0.2
0.0
0.001
0.01
0.1
1.0
10.0
100.0
1000.0

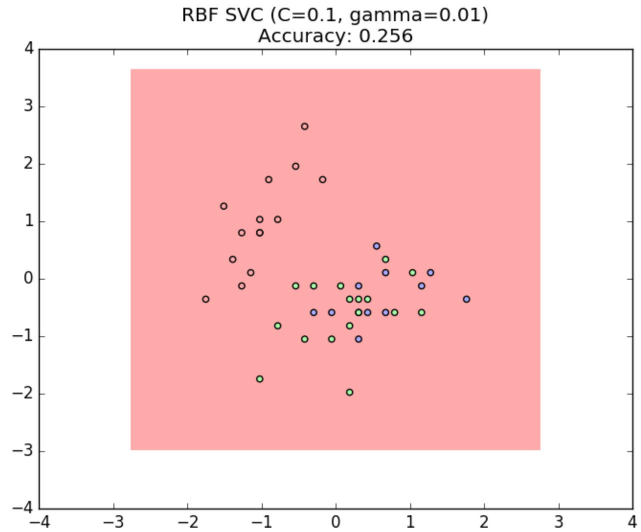
2.2 Grid search on C and γ

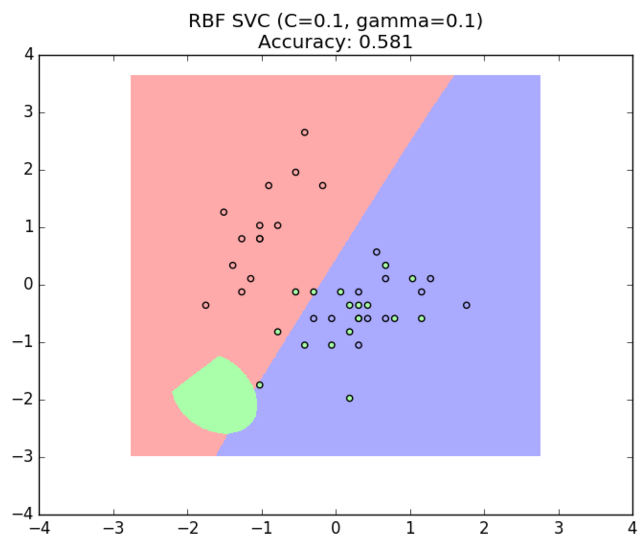
I performed grid search on the following range of values:

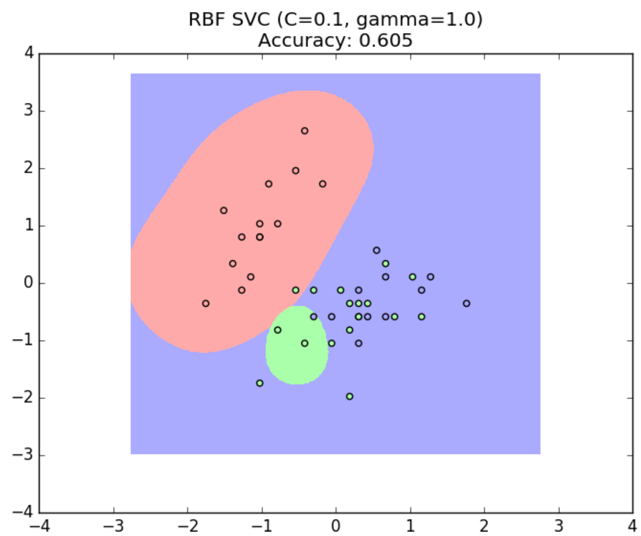
- for C: $\{10^{-1}, 10^{0}, \ldots, 10^{3}\}$
- for γ: $\{10^{-3}, 10^{-2}, \ldots, 10^{3}\}$

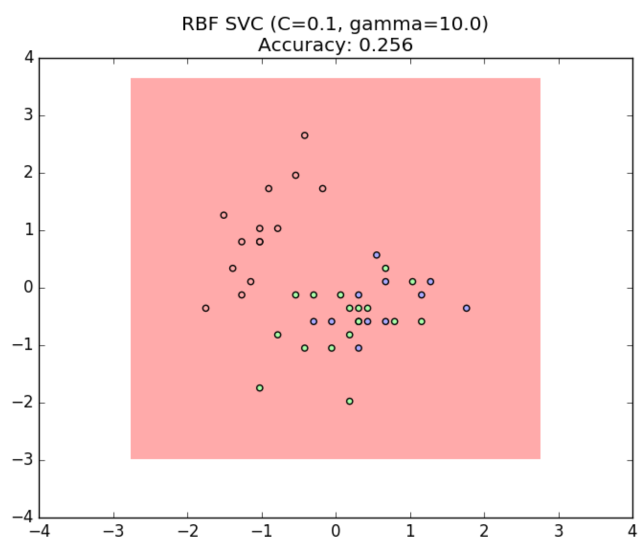
I reported all plots:

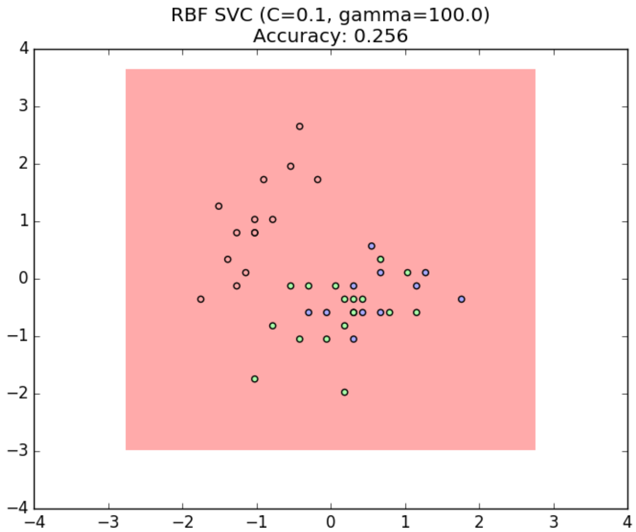


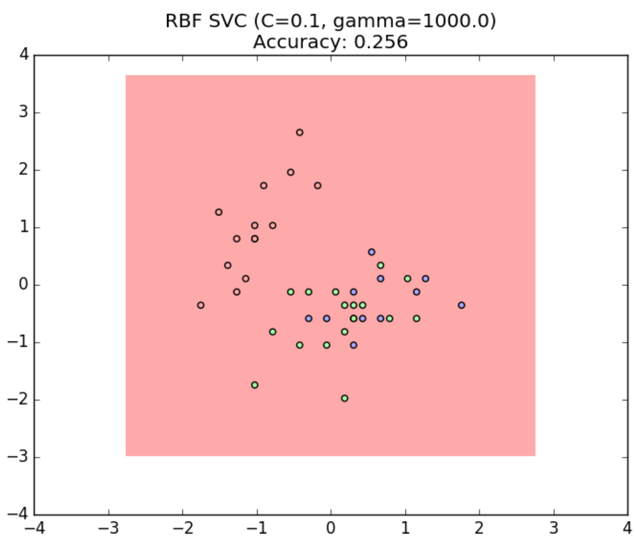

RBF SVC (C=0.1, gamma=0.01)
Accuracy: 0.256
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4


RBF SVC (C=0.1, gamma=0.1)
Accuracy: 0.581
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

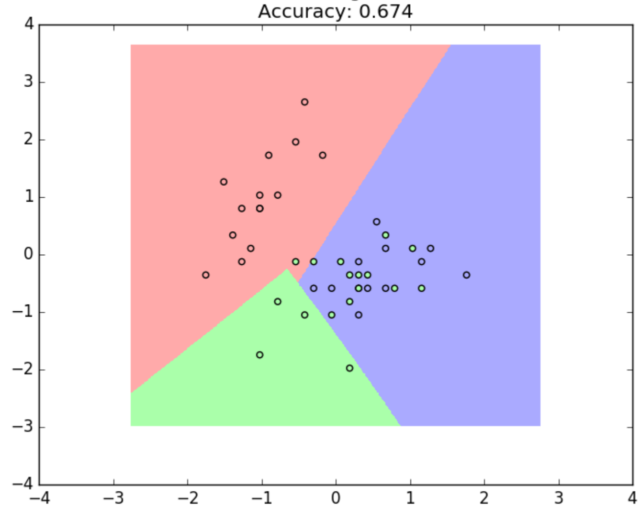

RBF SVC (C=0.1, gamma=1.0)
Accuracy: 0.605
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4


RBF SVC (C=0.1, gamma=10.0)
Accuracy: 0.256
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

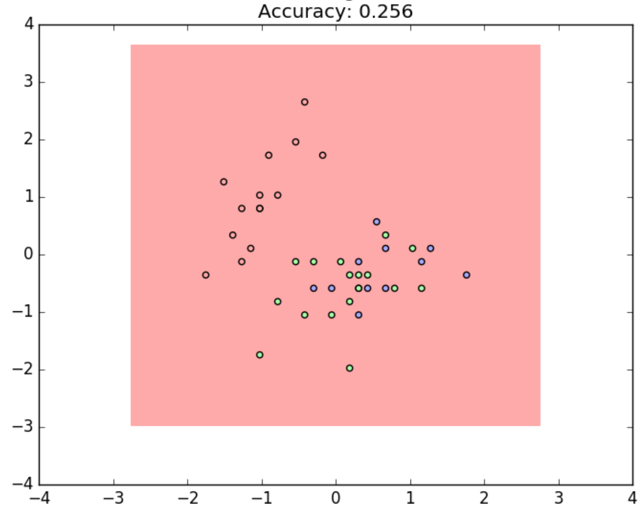

RBF SVC (C=0.1, gamma=100.0)
Accuracy: 0.256

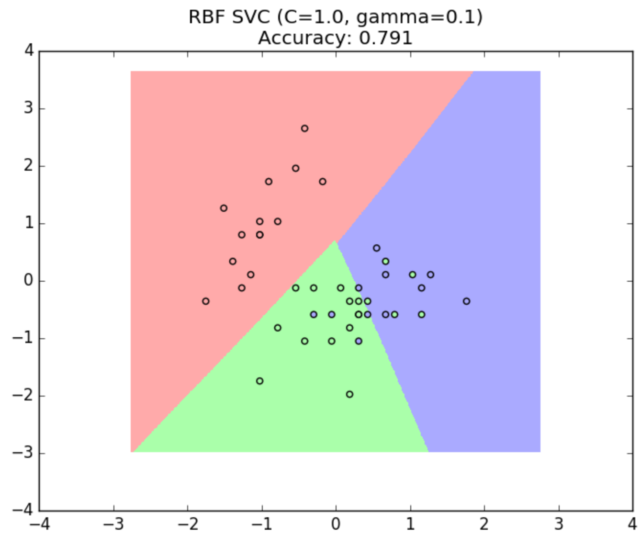

RBF SVC (C=0.1, gamma=1000.0)
Accuracy: 0.256

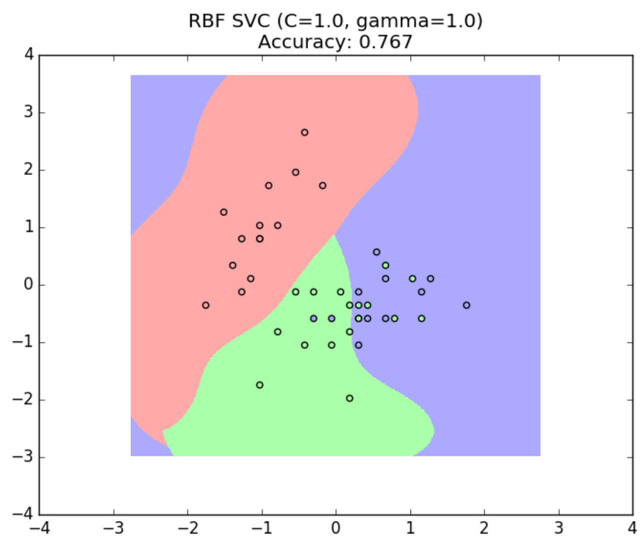
RBF SVC (C=1.0, gamma=0.01)
Accuracy: 0.674

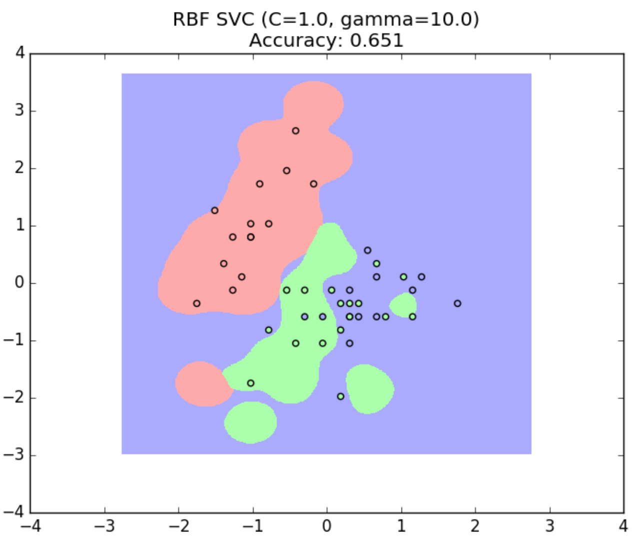


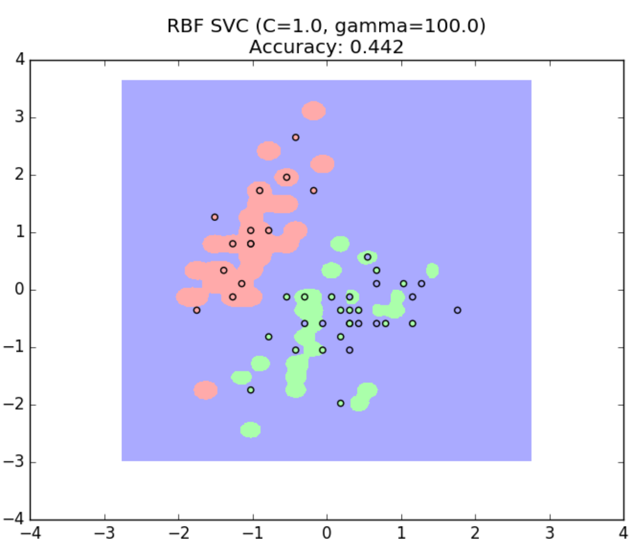
RBF SVC (C=1.0, gamma=0.001)
Accuracy: 0.256

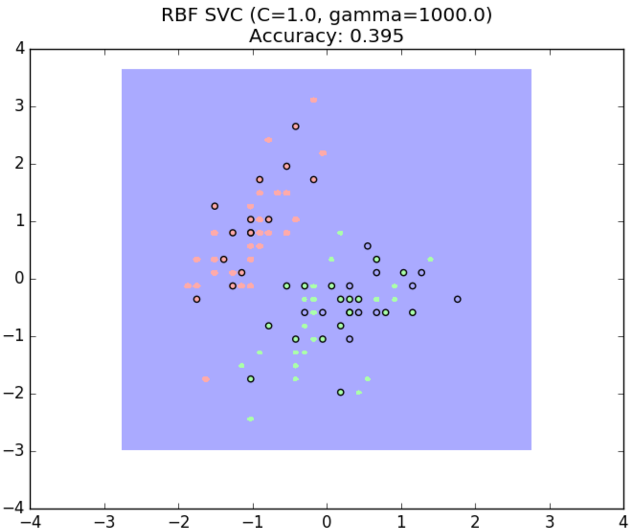



RBF SVC (C=1.0, gamma=0.1)
Accuracy: 0.791
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

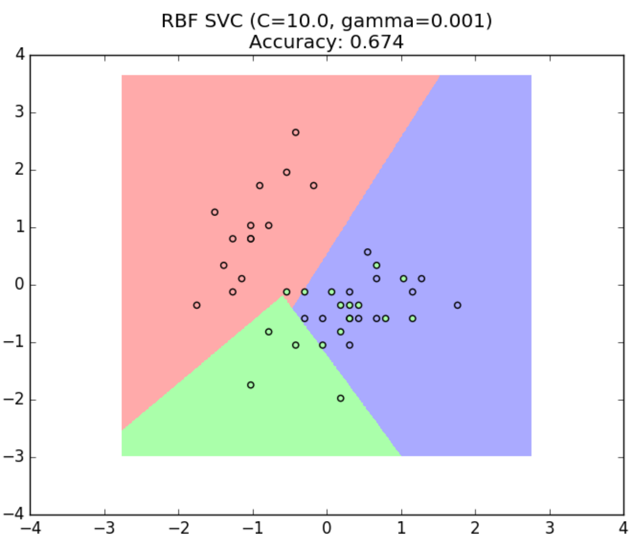

RBF SVC (C=1.0, gamma=1.0)
Accuracy: 0.767
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4


RBF SVC (C=1.0, gamma=10.0)
Accuracy: 0.651
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

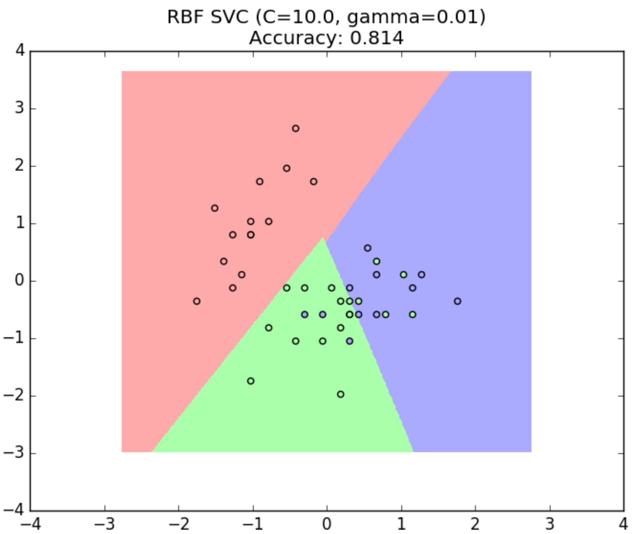

RBF SVC (C=1.0, gamma=100.0)
Accuracy: 0.442
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

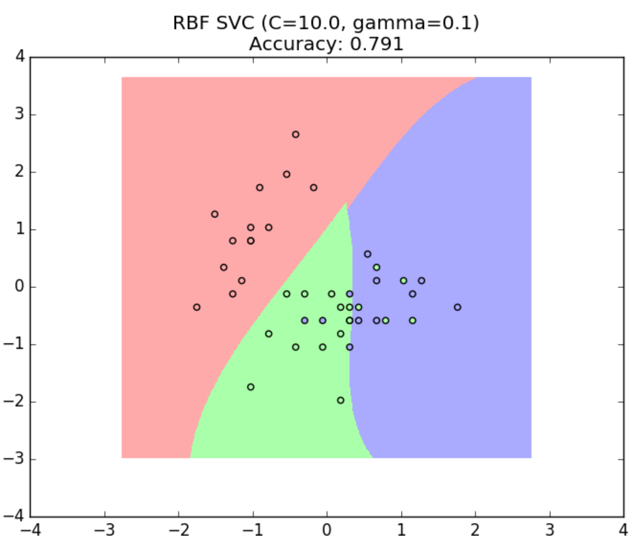


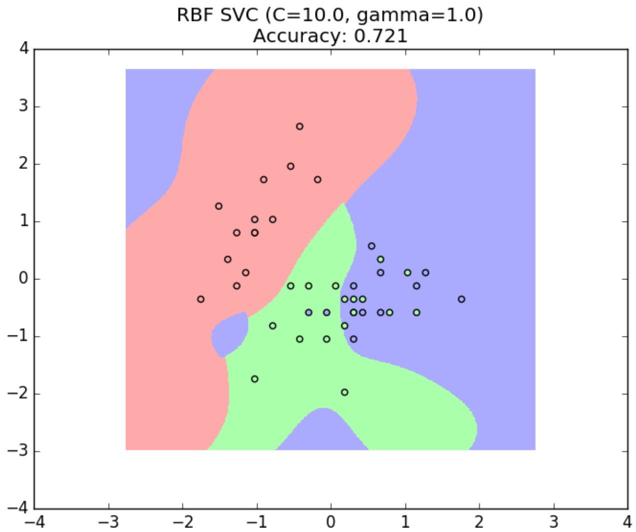
RBF SVC (C=1.0, gamma=1000.0)
Accuracy: 0.395

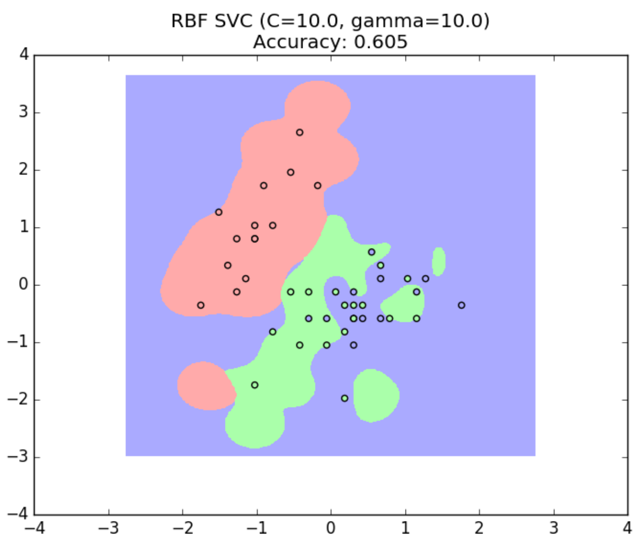


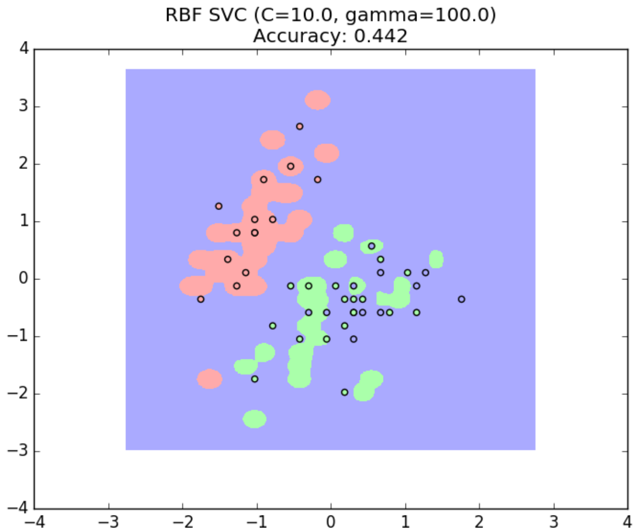
RBF SVC (C=10.0, gamma=0.001)
Accuracy: 0.674

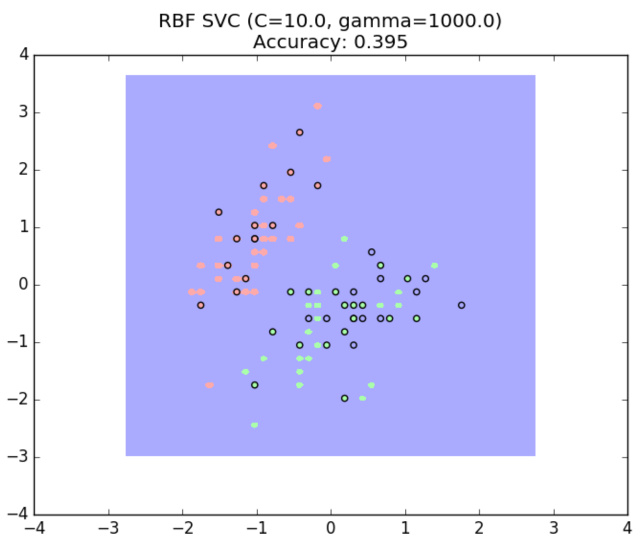

RBF SVC (C=10.0, gamma=0.01)
Accuracy: 0.814
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

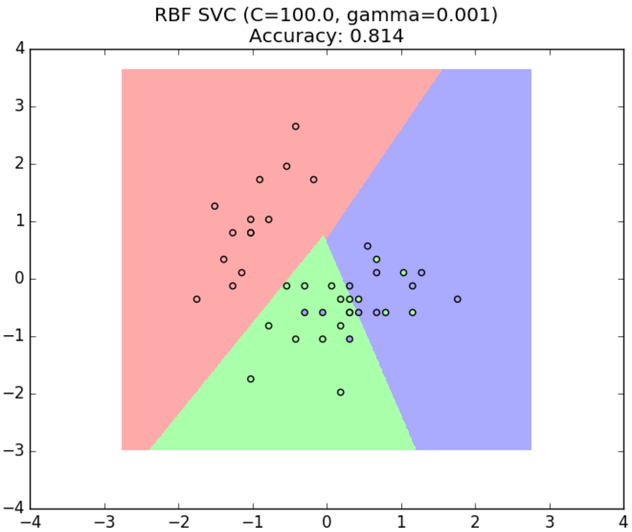

RBF SVC (C=10.0, gamma=0.1)
Accuracy: 0.791
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

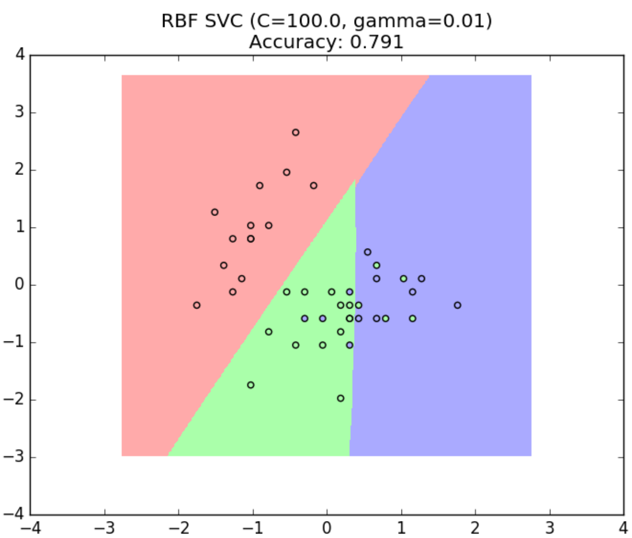

RBF SVC (C=10.0, gamma=1.0)
Accuracy: 0.721

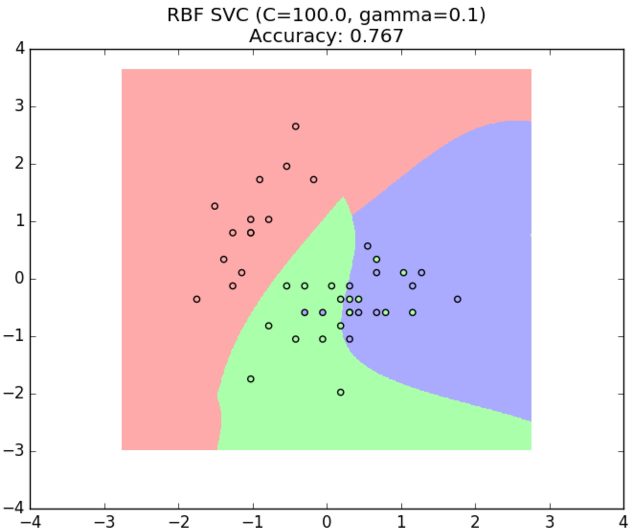

RBF SVC (C=10.0, gamma=10.0)
Accuracy: 0.605

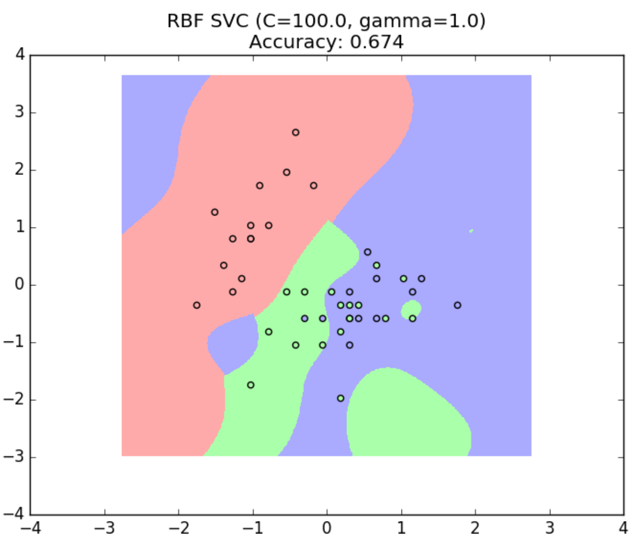

RBF SVC (C=10.0, gamma=100.0)
Accuracy: 0.442
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

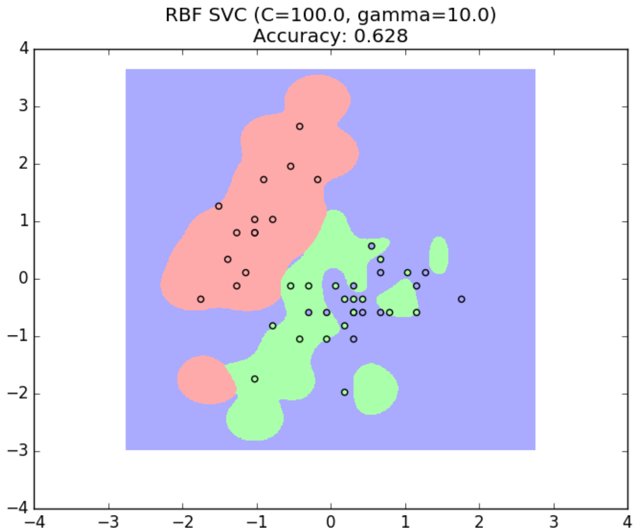

RBF SVC (C=10.0, gamma=1000.0)
Accuracy: 0.395
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

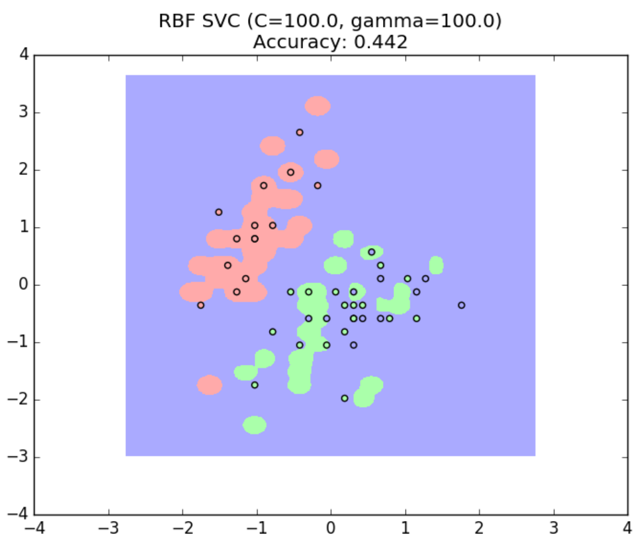

RBF SVC (C=100.0, gamma=0.001)
Accuracy: 0.814
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

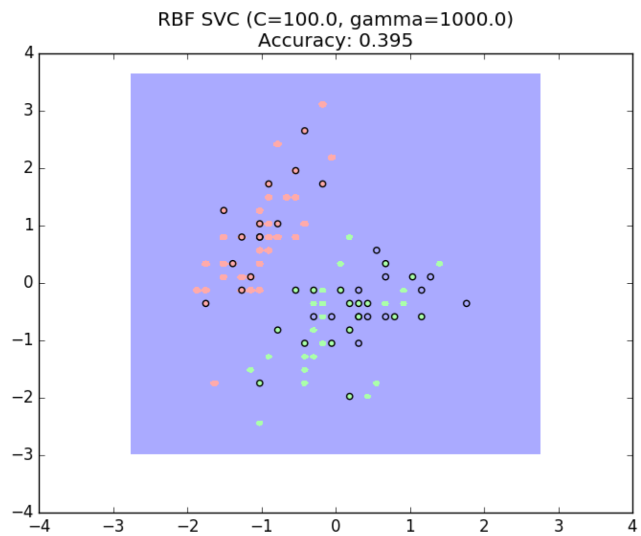

RBF SVC (C=100.0, gamma=0.01)
Accuracy: 0.791
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

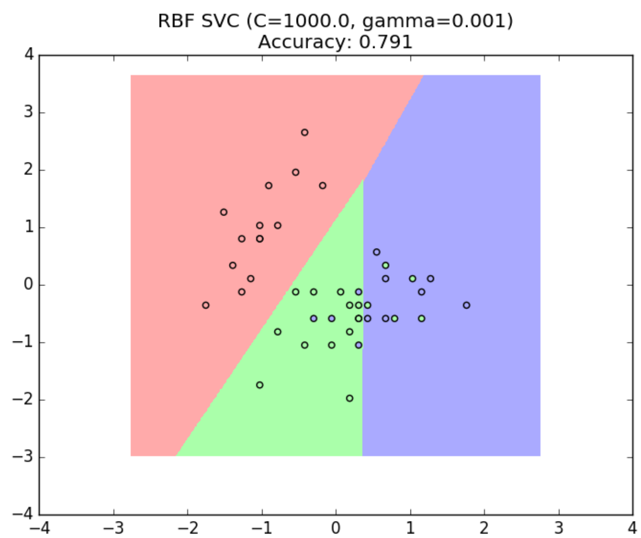

RBF SVC (C=100.0, gamma=0.1)
Accuracy: 0.767

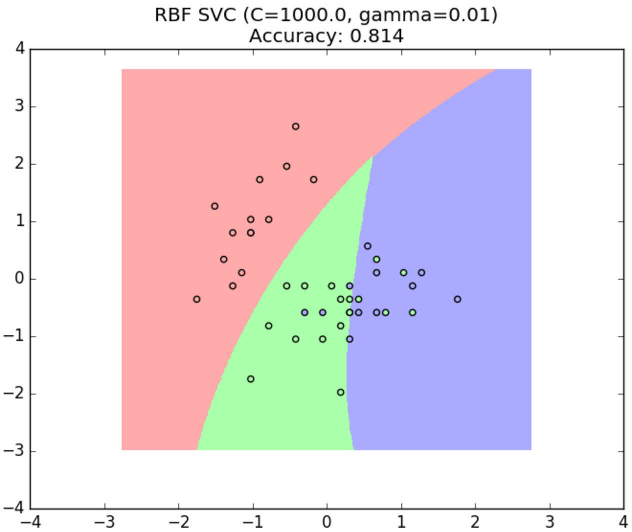

RBF SVC (C=100.0, gamma=1.0)
Accuracy: 0.674

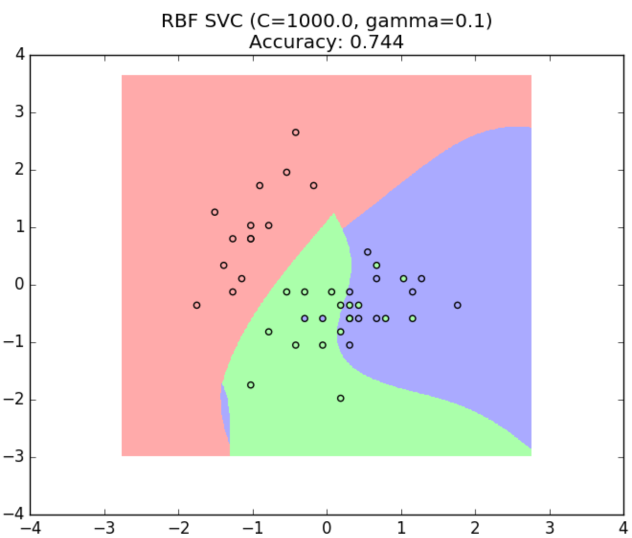

RBF SVC (C=100.0, gamma=10.0)
Accuracy: 0.628
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4


RBF SVC (C=100.0, gamma=100.0)
Accuracy: 0.442
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

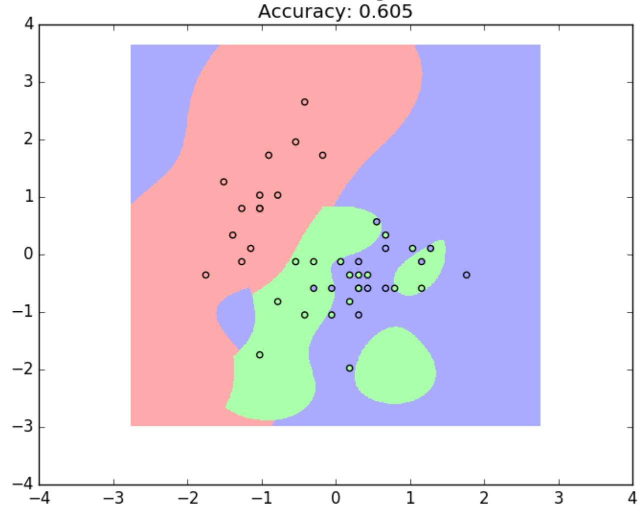

RBF SVC (C=100.0, gamma=1000.0)
Accuracy: 0.395


RBF SVC (C=1000.0, gamma=0.001)
Accuracy: 0.791

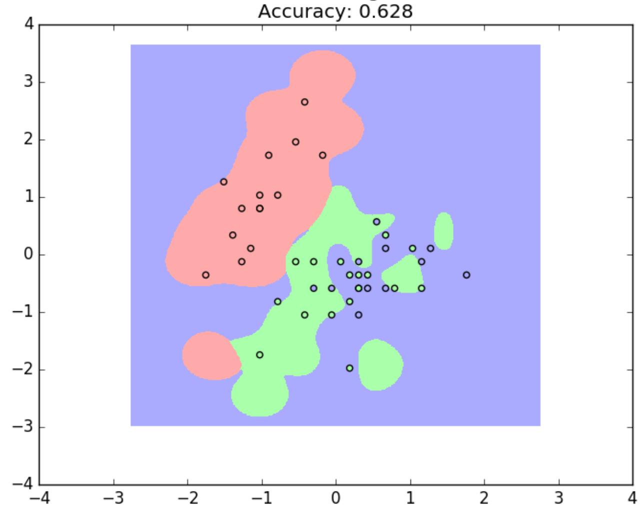

RBF SVC (C=1000.0, gamma=0.01)
Accuracy: 0.814
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4


RBF SVC (C=1000.0, gamma=0.1)
Accuracy: 0.744
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

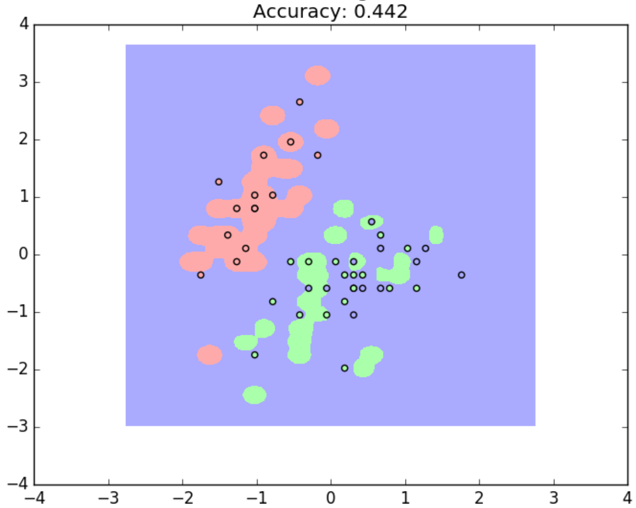
RBF SVC (C=1000.0, gamma=1.0)
Accuracy: 0.605



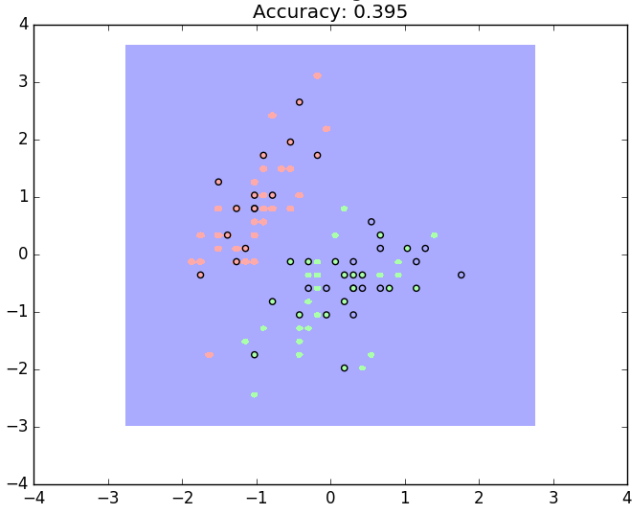
RBF SVC (C=1000.0, gamma=10.0)
Accuracy: 0.628



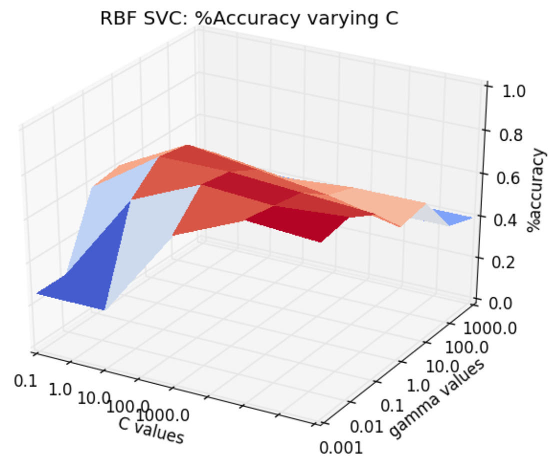
RBF SVC (C=1000.0, gamma=100.0)
Accuracy: 0.442



RBF SVC (C=1000.0, gamma=1000.0)
Accuracy: 0.395

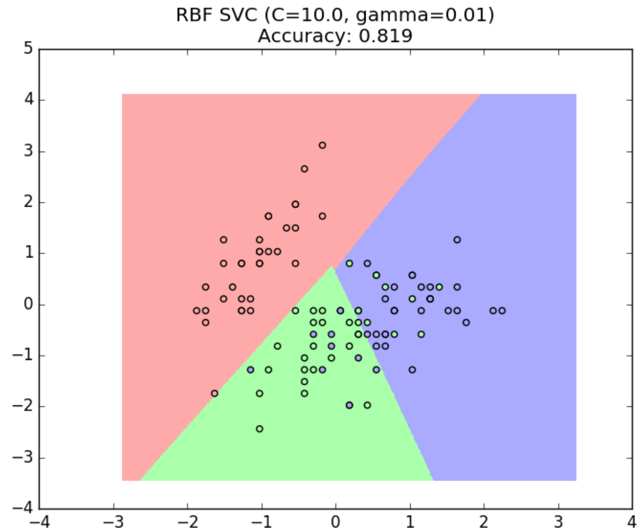


We can observe that, in general, for low value of γ and C we have underfitting, and for higher value of them we have overfitting. Where these parameters are almost in the same order of magnitude, it is evident that we have almost a linear model.
In this plot I show how vary the accuracy of prediction on validation set in function of C and γ, first on a 3d plot and second on a table:



	0.001	0.01	0.1	1.0	10.0	100.0	1000.0
0.1	0.255813953488	0.255813953488	0.581395348837	0.604651162791	0.255813953488	0.255813953488	0.255813953488
1.0	0.255813953488	0.674418604651	0.790697674419	0.767441860465	0.651162790698	0.441860465116	0.395348837209
10.0	0.674418604651	0.813953488372	0.790697674419	0.720930232558	0.604651162791	0.441860465116	0.395348837209
100.0	0.813953488372	0.790697674419	0.767441860465	0.674418604651	0.627906976744	0.441860465116	0.395348837209
1000.0	0.790697674419	0.813953488372	0.744186046512	0.604651162791	0.627906976744	0.441860465116	0.395348837209

The best combination found is $C = 10$ and $\gamma = 0.01$. Now I plot the decision boundaries with the best choice of parameters. The accuracy on test set is 81,9%

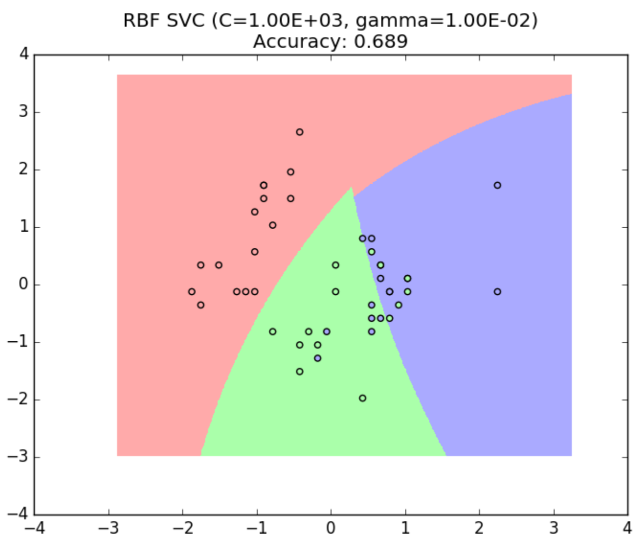


3 K-Fold

Now I merged train and validation set and I performed the same grid search on C and γ. I have not reported all the plots, because they were a lot. The program yields that the best values of the search are:

- $C = 1000$
- $\gamma = 0.01$

With the max validation accuracy at 81,9%. The final model is this:


RBF SVC (C=1.00E+03, gamma=1.00E-02)
Accuracy: 0.689
4
3
2
1
0
−1
−2
−3
−4
−4
−3
−2
−1
0
1
2
3
4

And the grid search yields:

	0.001	0.01	0.1	1.0	10.0	100.0	1000.0
0.1	0.247619047619	0.247619047619	0.628571428571	0.647619047619	0.247619047619	0.228571428571	0.228571428571
1.0	0.247619047619	0.685714285714	0.809523809524	0.8	0.638095238095	0.438095238095	0.352380952381
10.0	0.685714285714	0.809523809524	0.8	0.761904761905	0.6	0.447619047619	0.352380952381
100.0	0.809523809524	0.8	0.819047619048	0.72380952381	0.590476190476	0.447619047619	0.352380952381
1000.0	0.8	0.819047619048	0.780952380952	0.647619047619	0.590476190476	0.447619047619	0.352380952381

The final score is different because the algorithm performs train and validation on different data set. In our case, it performs worse than the algorithm with no corss-validation. It depends on how data are splitted in train-validation-test set (in my program randomly, indeed at each run I found different values). With more data, probably, results will improve.