KTH Royal Institute of Technology
M.Sc. Machine Learning
DD2380

Diogo Pinheiro
Jakob Lindén

October 5, 2020

Question 1

Move the clusters around and change their sizes to make it easier or harder for the classifier to find a decent boundary. Pay attention to when the optimizer (minimize function) is not able to find a solution at all.

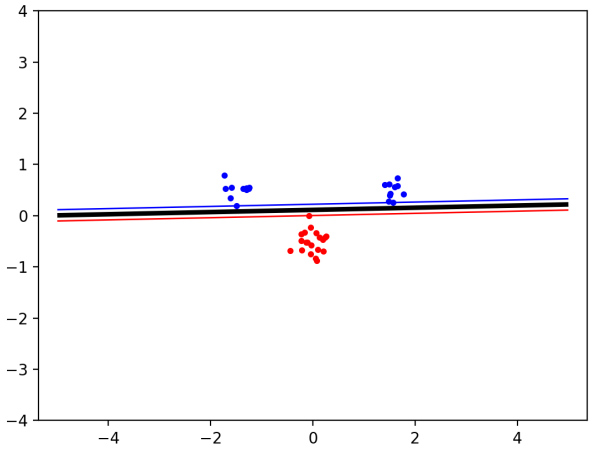

Figure 1: Question 1: Class A: Size (10,2) , Range [1.5, 0.5] and [-1.5, 0.5] , std = 0.2; Class B: Size (20,2), Range [0.0, -0.5] , std = 0.2 , C = 1000

With this easy to separate data the SVM does a good job of splitting it up and we manage to find a minimize solution.

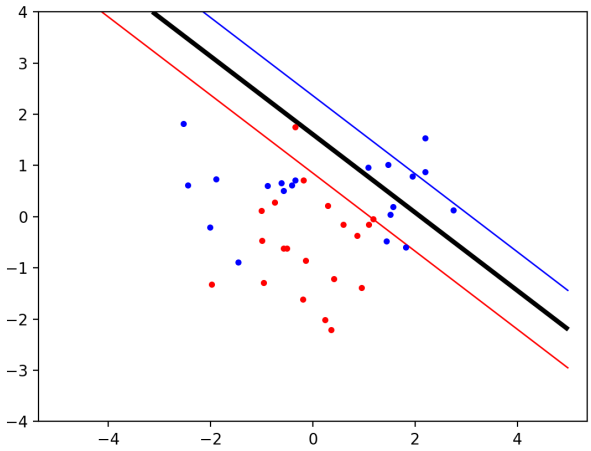

Figure 2: Question 1: Class A: Size (10,2) , Range [1.5, 0.5] and [-1.5, 0.5] , std = 0.9; Class B: Size (20,2), Range [0.0, -0.5] , std = 0.9 , C = 1000

With this more noisy data there is not an easy way to split the data linearly, therefore the minimize function is not able to find a minimal solution.

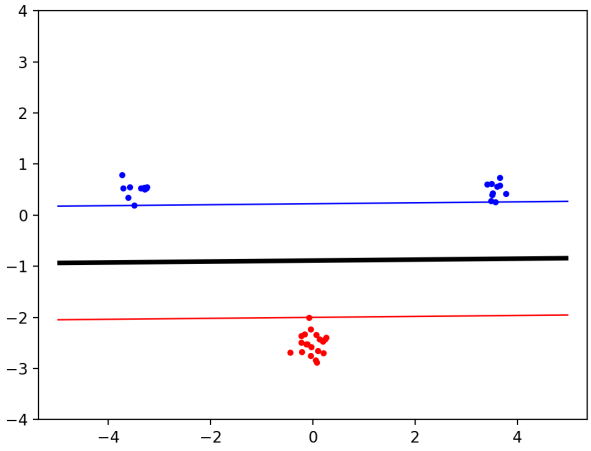

Figure 3: Question 1: Class A: Size (10,2) , Range [3.5, 0.5] and [-3.5, 0.5] , std = 0.2; Class B: Size (20,2), Range [0.0, -2.5] , std = 0.2 , C = 1000

Separating the clusters more gives us the following result.

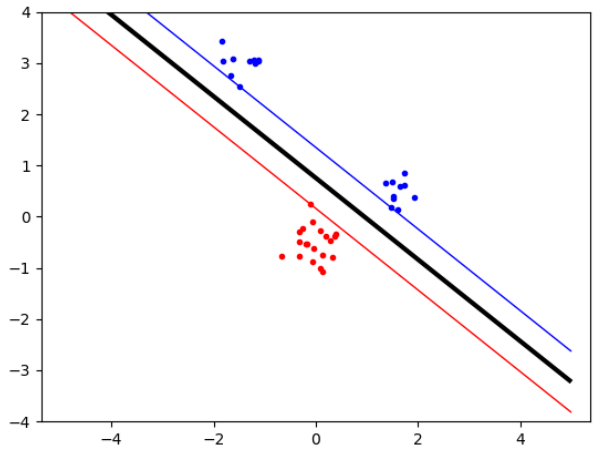

Figure 4: Question 1: Class A: Size (10,2) , Range [1.5, 0.5] and [-1.5, 3] , std = 0.3; Class B: Size (20,2), Range [0.0, -0.5] , std = 0.3 , C = 1000

Moving one of the clusters up gives us this result.

Question 2

Implement the two non-linear kernels. You should be able to classify very hard data sets with these.

```
# Polynomial Kernel
def polynomialKernel(x, y, p):
    return math.pow(np.dot(x, y) + 1, p)

# RBF Kernel
def rbfKernrel(x, y, sigma2):
    return math.exp(-math.pow(np.linalg.norm(np.subtract(x, y)), 2)/(2 * sigma2))
```

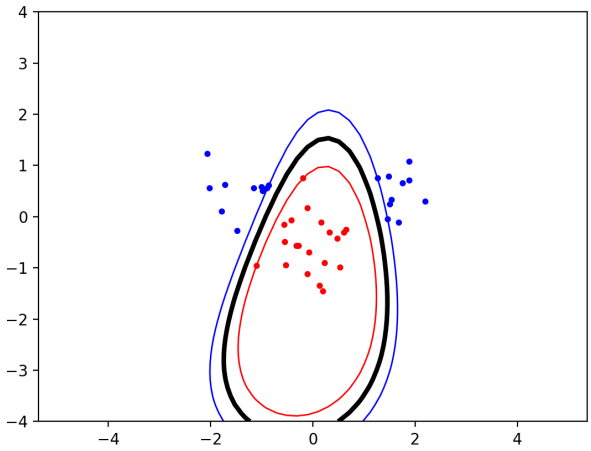

Figure 5: Question 2: RBF : std = 0.5 , $\sigma = 3$, C = 1000

The RBF kernel with a *sigma* of 3 gives us the following split.

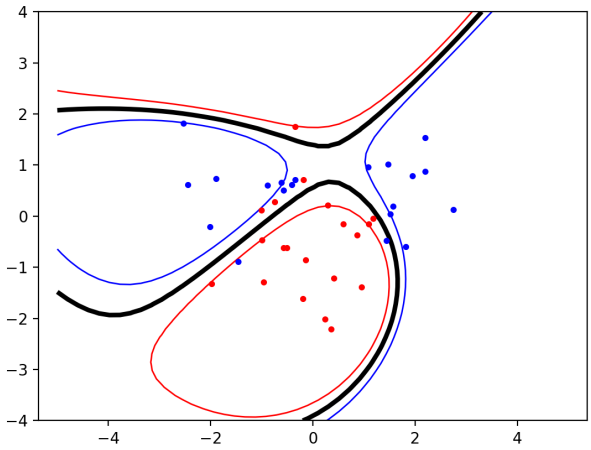

Figure 6: Question 2: RBF : std = 0.9 , $\sigma = 3$, C = 1000

The RBF kernel with a σ of 3 with more noisy data gives us this split.

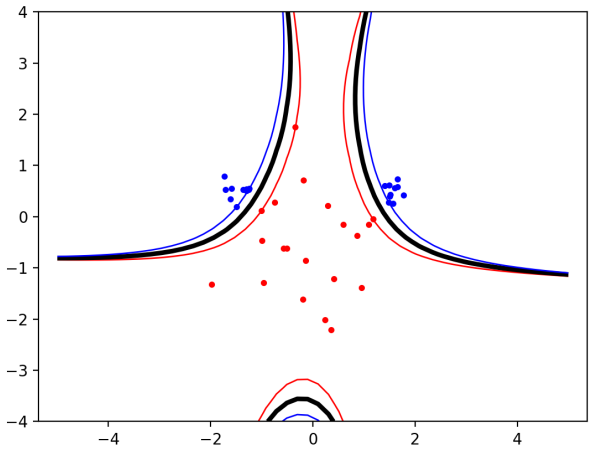

Figure 7: Question 2: Poly : std = 0.9 , p = 3 , C = 1000

The polynomial kernel with a power of 3 gives us the following split.

Question 3

The non-linear kernels have parameters; explore how they influence the decision boundary. Reason about this in terms of the bias- variance trade-off.

The σ parameter for the RBF kernel defines how far the influence of a single data point reaches, low values meaning far away, and high values meaning close. Therefore by increasing σ we lower variance by fitting more loosely to the data, but increasing bias and vice versa.

For the polynomial kernel increasing the power parameter p the complexity of the curve increases, therefore the variance is increased and bias decreased with each higher order polynomial.

In terms of overfitting, we overfit the data when the bias is low with high variance. Therefore how to choose these variables are problem dependent.

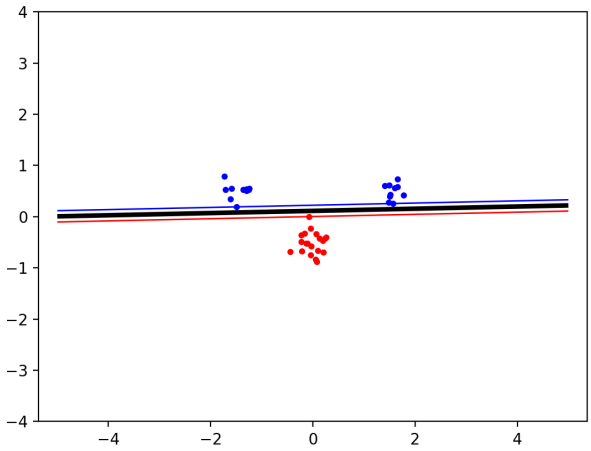

Figure 8: Question 3: Poly : p = 1

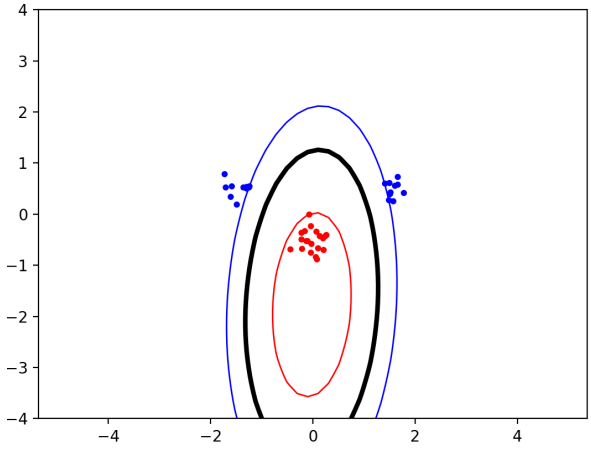

Figure 9: Question 3: Poly : p = 2

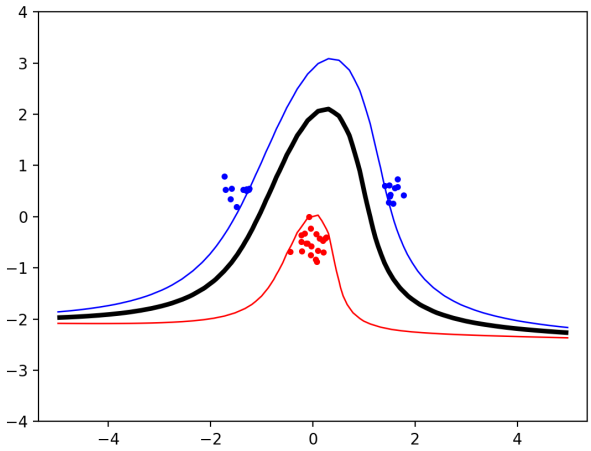

Figure 10: Question 3: Poly : p = 3

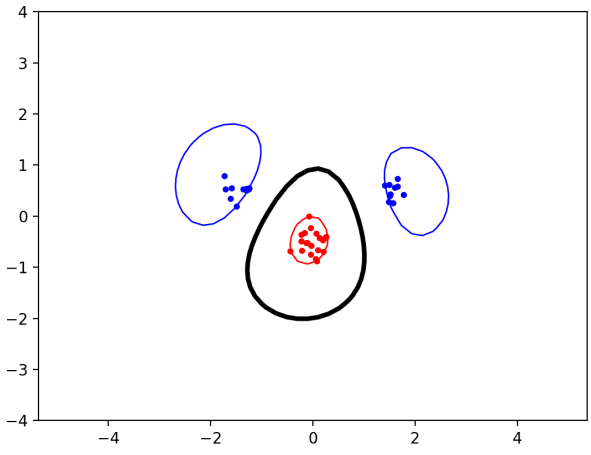

Figure 11: Question 3: RBF : $\sigma = 1$

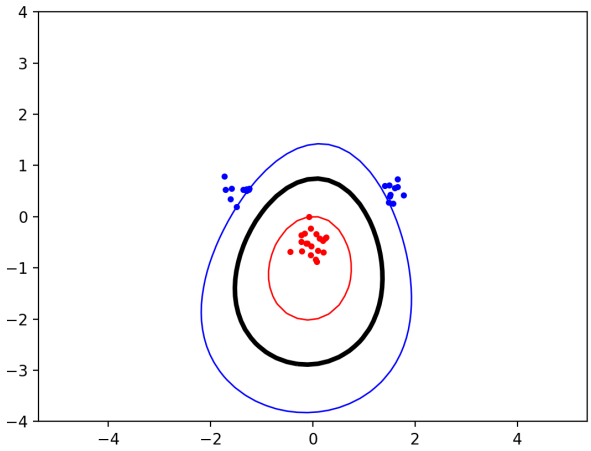

Figure 12: Question 3: RBF : $\sigma = 2$

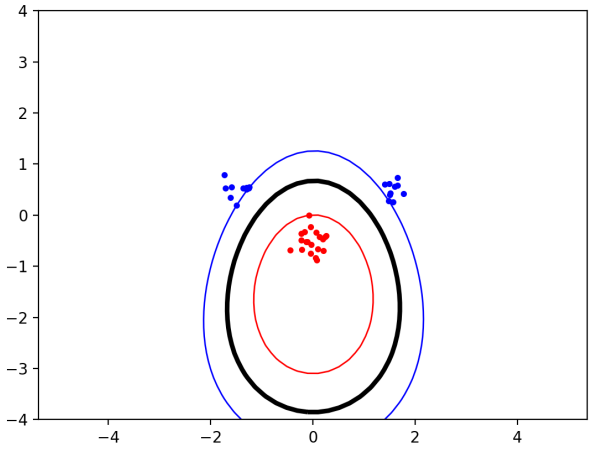

Figure 13: Question 3: RBF : $\sigma = 3$

Question 4

Explore the role of the slack parameter C. What happens for very large/small values?

With the C parameter we can control how much misclassification of each training example we want in the optimization. So, for large values, the optimization will choose a smaller margin hyperplane if it classifies the points correctly. In contrast, with smaller C values the optimization will try to use a wider margin hyperplane, even if this results in more misclassification.

Therefore using the slack parameter is most important when some data points we consider noise prevents the classification split of the underlying model.

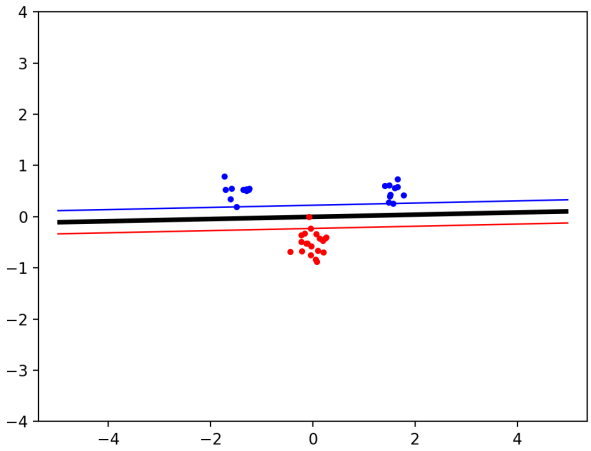

Figure 14: Question 4 : C=10

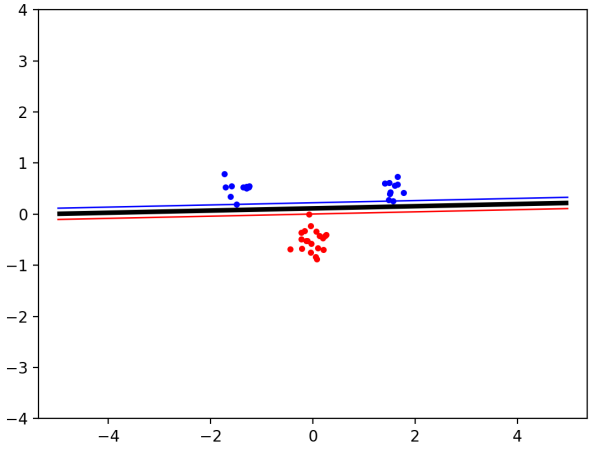

Figure 15: Question 4 : C=100

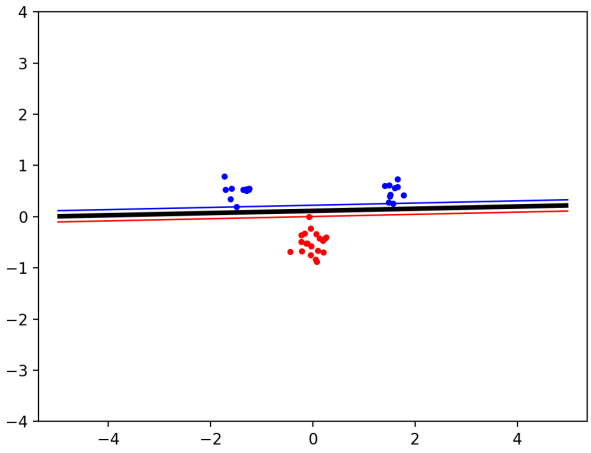

Figure 16: Question 4 : C = 100000000; Really small margin, due to having an extremely high C value.

Question 5

Imagine that you are given data that is not easily separable. When should you opt for more slack rather than going for a more complex model (kernel) and vice versa?

The decisions of whether choosing a different Kernel or simply just changing the slack parameter depends on the type of position of the data points, that is, how both classes (if we consider a binary classification problem) are positioned in the "map".

Let's consider, as an example, a case where class 1 is surrounded on all sides by class 2 (similar example can be seen in Figure 17). In this situation, changing the slack would not be much of an improvement if we have a linear kernel. So, changing the kernel type to RBF would allows to have more flexibility on the separation. However, this has some disadvantages, such as the risk of overfitting and the time it takes to learn, which in this case would be higher than other kernels because of the choice of using RBF.

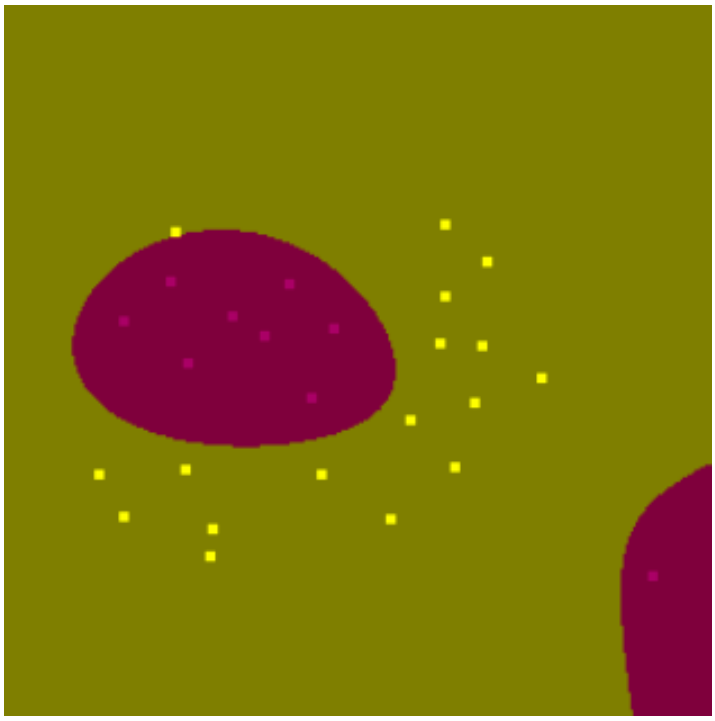

Figure 17: Example of a case where we should choose to change the kernel.

On the other hand, in cases where the data are not easily separable but choosing other kernel types would not convey a significant improvement in separation or if for that specific data type the disadvantages would be bigger than the advantages (underfitting/overfitting, learning time, number of hyperparameters) we would have to adjust the slack. Additionally, we also need to take in consideration that not all kernel types fit all kinds of data, linear kernels are more flexible in that sense, while polynomial and RBF are not.

There is also a case for keeping the amount of support vectors relatively low compared to the amount of data points. If the amount of support vectors increase in for example a more advanced model, the more likely overfitting becomes. In this case using slack instead of a more complex model is more desireable.