Assignment 4

Computational Intelligence, SS2020

Team Members		
Last name	First name	Matriculation Number
Reindl	Hannes	01532129
Samer	Philip	01634718
Rösel	David	01634719

1 Linear SVM

For the linear SVM task we use the training set in the **data.json** file. The set consists of a binary classification problem with two input dimensions. The following figure shows the data points of the training set:

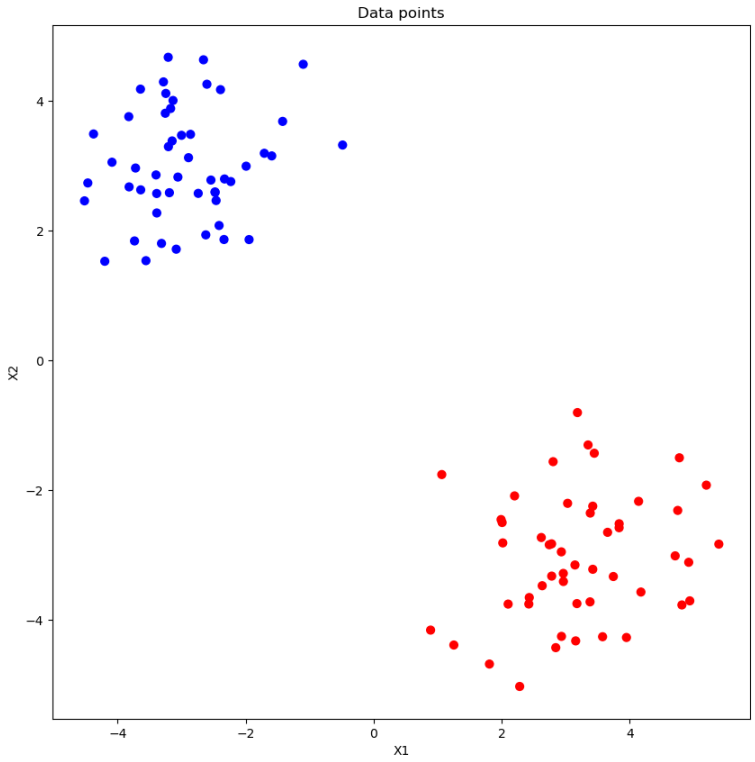

Figure 1: Data points

a)

In task a we want to train a linear SVM with the given training set. We use a instance of the SVM class from the sklearn library with a **linear** kernel. The followed figure shows the decision boundary of the trained SVM:

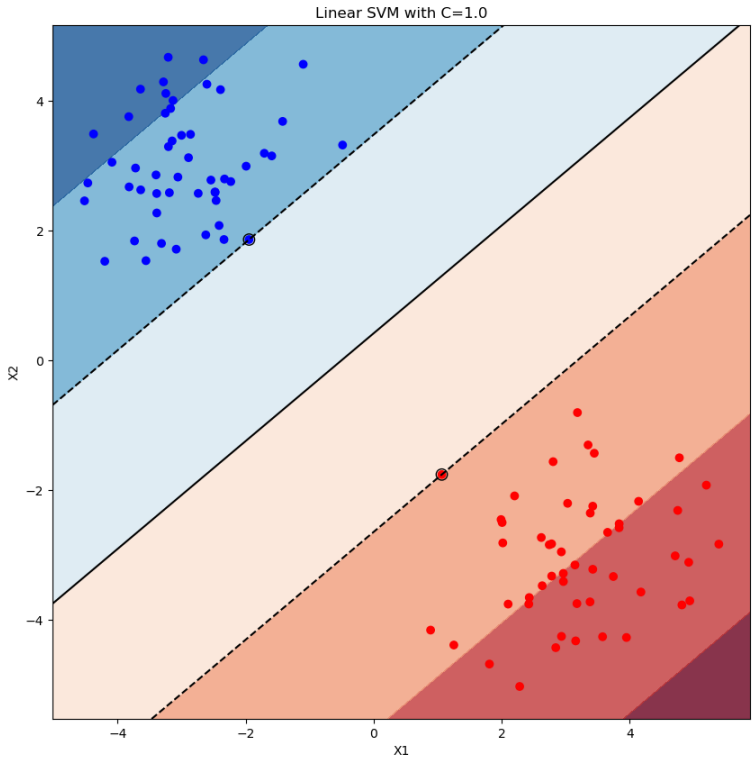

Figure 2: Trained SVM with a linear kernel

b)

In task b we add an additional point to our training set, (4,0) for x and label 1 for y. The added point is an outlier and therefore the whole decision boundary changes and rotated in such a way that all points are in the right decision boundary again to maximize the separation.

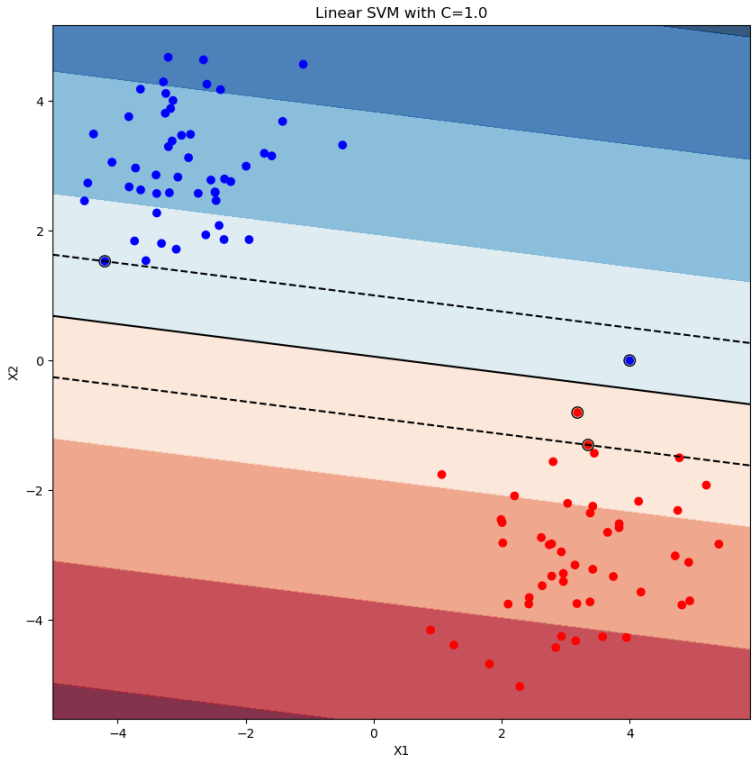

Figure 3: Trained SVM with a linear kernel with additional point

c)

In task c we take the modified training set from task b and vary the parameter C. The following figures show the decision boundary for the different parameter C.

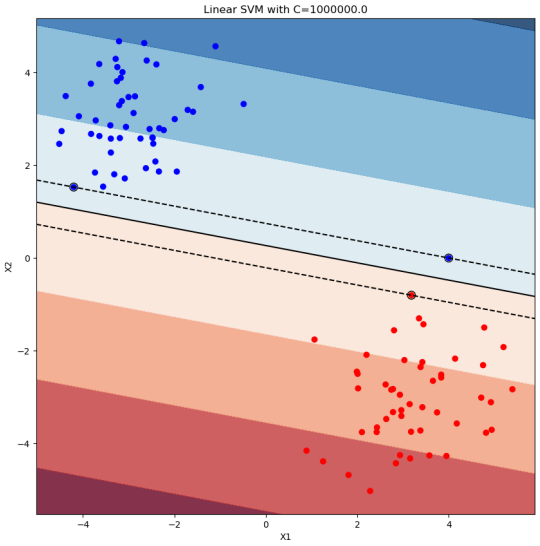

Figure 4: SVM with C = 1000000

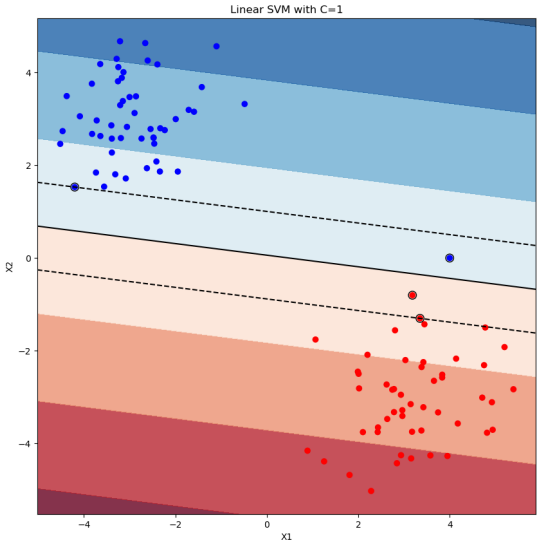

Figure 5: SVM with C = 1

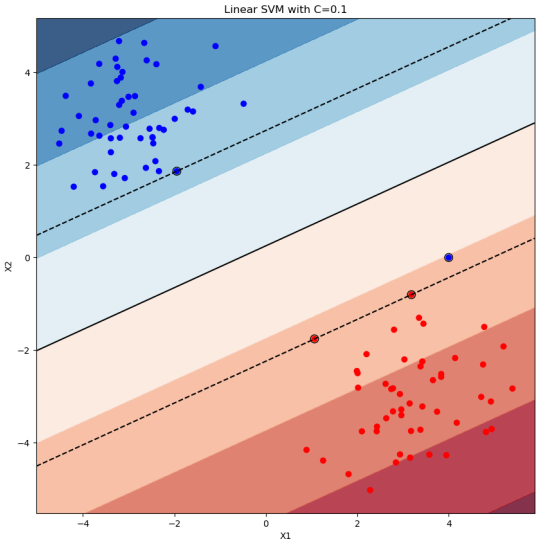

Figure 6: SVM with C = 0.1

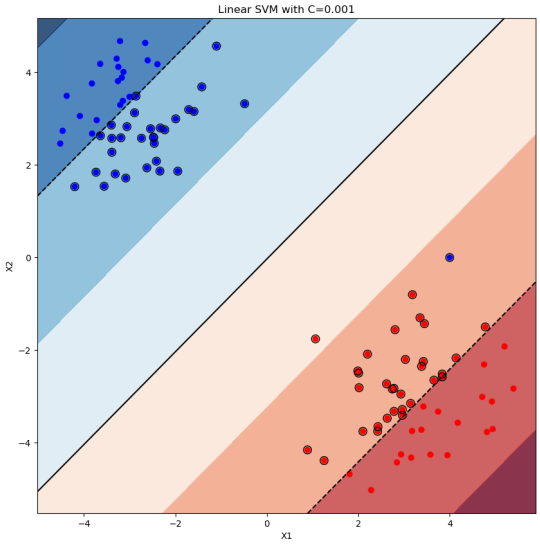

Figure 7: SVM with C = 0.001

In the previous task the parameter C was set to its default value = 1. For a high C outlier are considerate more important for the decision boundary and less support vectors are found. For small C more support vectors are found and the decision boundary is not much impacted by outlier.

2 Nonlinear (kernel) SVM

For the nonlinear SVM task we use the training set and the test set in **data_nl.json**. The set consists of a binary classification problem with two input dimensions. In this task we want to use different kernel to solve this nonlinear classification problem. The following figure show the data points of the training set:

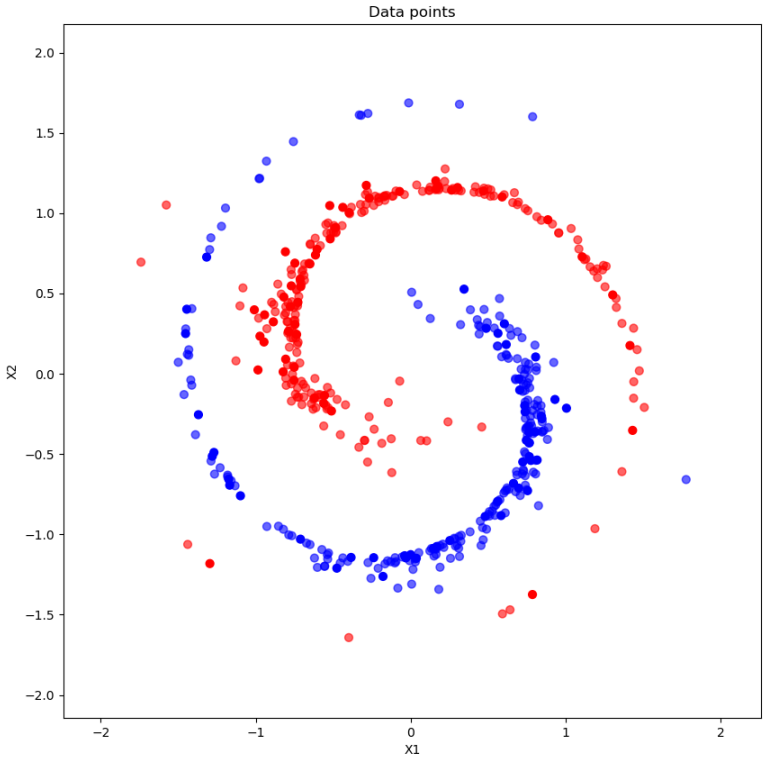

Figure 8: Data points

a)

In task a we use a linear kernel like in exercise 1. A linear kernel is not enough to solve the nonlinear problem of the new data set. The following figure show the decision boundary for a SVM with a linear kernel. The normal dots show the training set and the dots with a black border are from the test set.

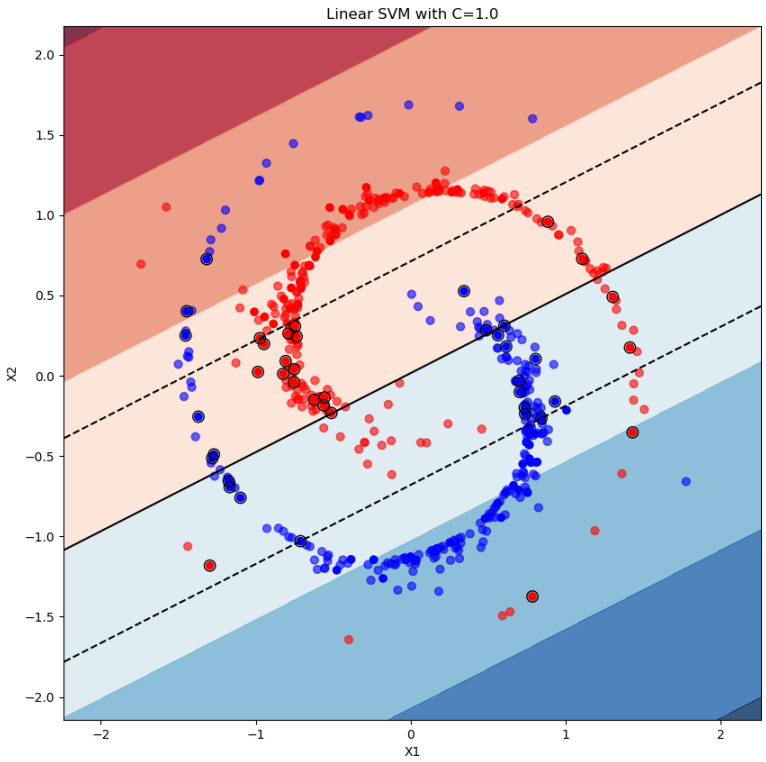

Figure 9: SVM with linear kernel

The test score for the linear kernel is 0.8125 for the test set and 0.87 for the training set. The score is not as bad as expected, but the figure show that the decision boundary doesn't really follow the data points.

b)

In task b we change the linear kernel with a polynomial kernel. We calculated every degree between 1 and 20 degree for our polynomial **d** of our SVM, to find the best fitting degree for our problem. The following figure shows the scores each degree archived:

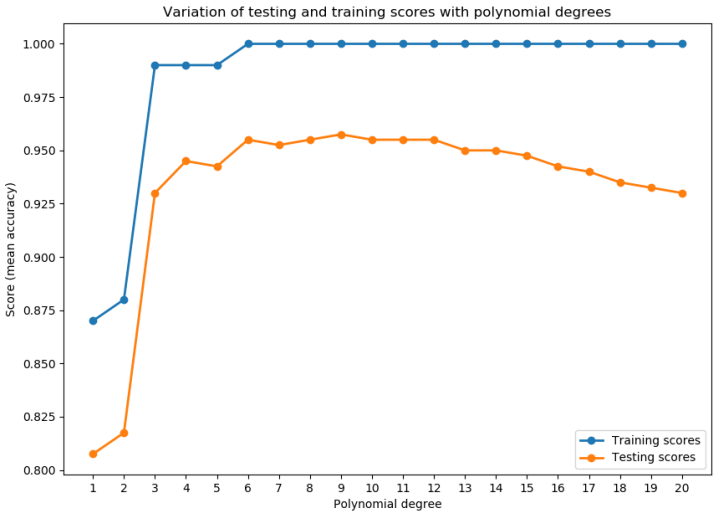

Figure 10: Scores for different degrees

The best score was archived for a polynomial with a degree of 9, with a test score of 0.9575 and a train score of 1.0.The following figure shows the decision boundary for the best score:

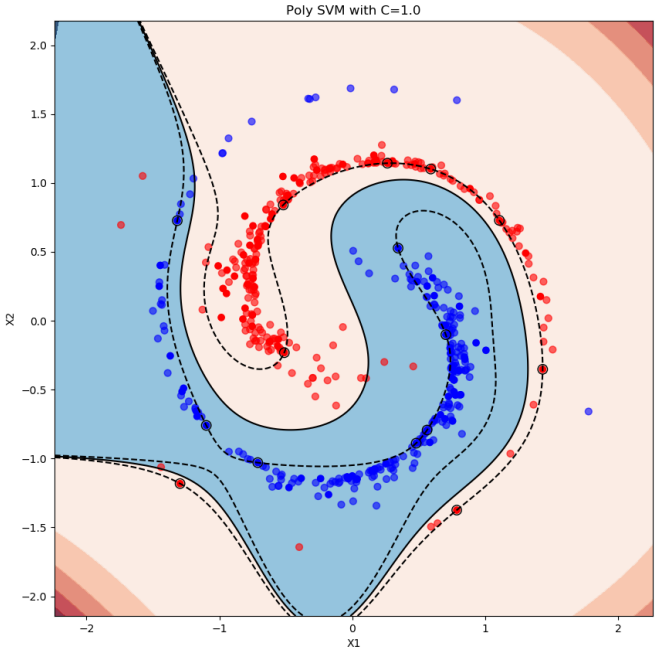

Figure 11: SVM with polynomial kernel and degree 9 with the best score

c)

For the last task c we used a RBF kernel. Like before in task b we search the best γ to maximize our score in a range between $\gamma = [0.01...1.99]$. The following figure shows the score depending on γ:

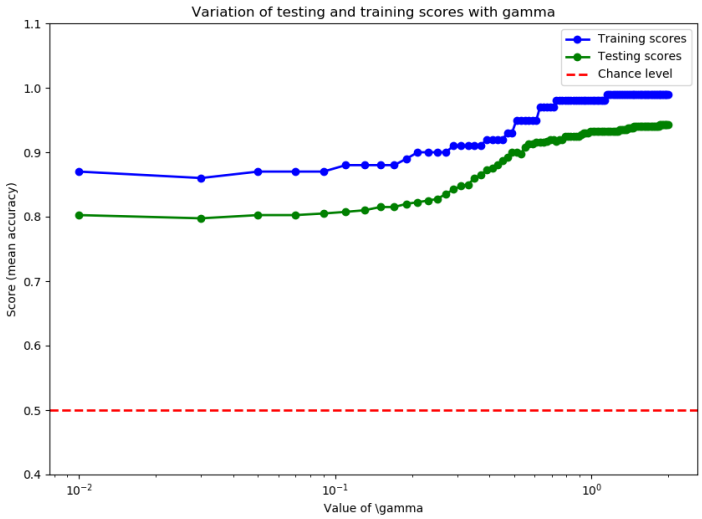

Figure 12: Scores for different gammas

The figure show, that the score increases with an increase in γ. The best score is archived with the highest γ and a test score of 0.9425 and a train score of 0.99. The following figure shows the best result:

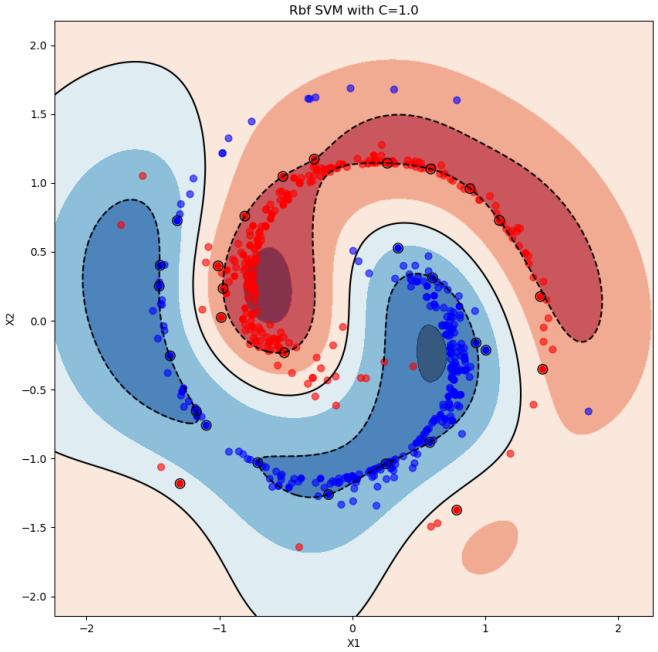

Figure 13: SVM with RBF kernel and gamma with the best score

comparison

The best kernel was the polynomial kernel with a degree of 9. The RBF kernel was increasing its score with an increase in γ an can be better then the polynomial kernel but for the given range it was slightly worse. The linear and the RBF kernel use almost the same amount of support vector and the polynomial kernel use only a small number. The RBF has the highest complexity and the linear linear kernel the lowest. The polynomial kernel generalizes best for the given dataset.

3 Multiclass classification

In this example we try to recognize written numbers from 1 to 5. The training samples are shown in figure 14 below. A picture consists of 28 by 28 Pixel.
one-versus-one: for N classes $\frac{N(N-1)}{2}$ classifiers.

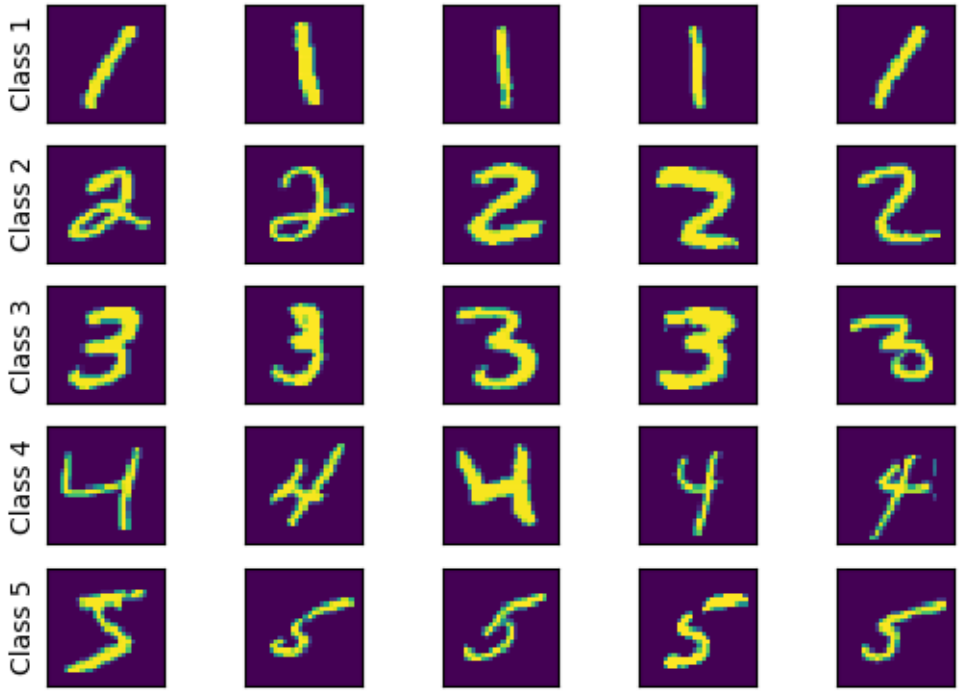

Figure 14: Trainingset

one-versus-rest: for N classes N classifiers.
The figure 15 below shows the results for the linear and RBF Kernel in respect to gamma. As one can see the Linear Kernel gives very good results, to understand why we briefly discuss what a Kernel does.
Basically it constructs a higher dimensional space where the different classes can be separated for example with a linear function. A linear Kernel works especially good for cases with a large number of features as is the case with images. The reason the linear Kernel has pretty solid values even when compared with an RBF Kernel is, that the result doesn't change much by adding even higher dimensions as is the case with RBF Kernels.
An example where a RBF is needed and a linear is no longer enough is, if there is a non linear connection between the classifiable and attributes. However even in this case it has to be noted, that for a certain gamma the RBF performs better than the linear.
We can also observe an overfitting like behavior for a higher value of gamma, were it perfectly fits the training sample but gets worse results for the test set.

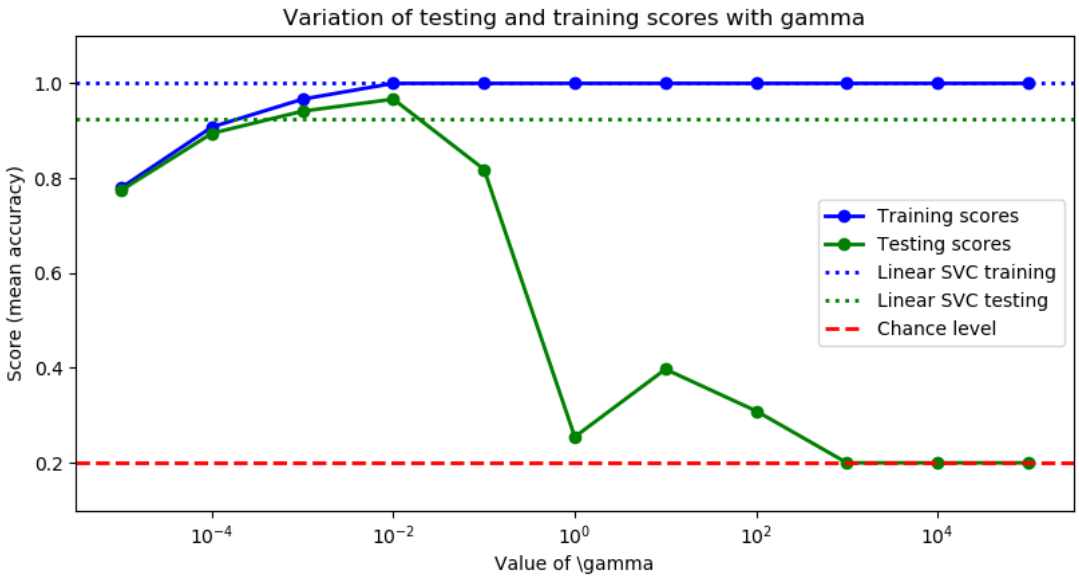

Figure 15: Comparison RBF and linear Kernel with different Gamma

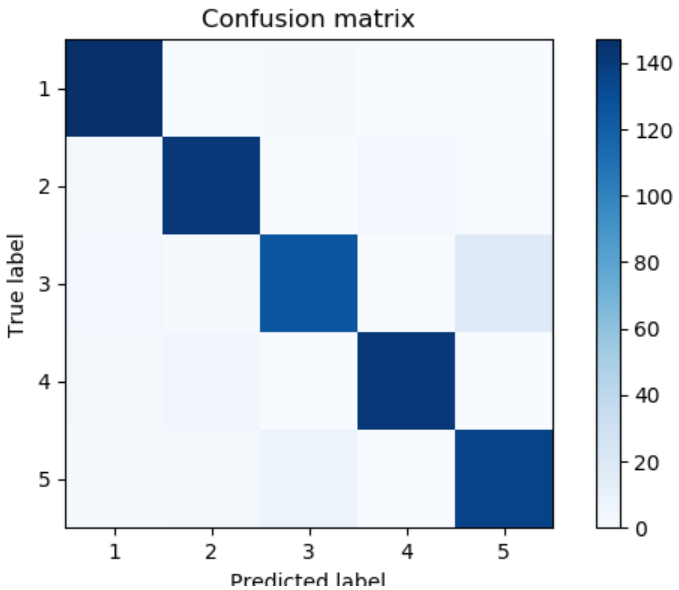

Figure 16: Confusion Matrix

The figure 16 shows the confusion matrix. For the number 3 it has the most difficulty recognizing, since the number in the main diagonal is the lowest for number 3.
The numbers for this row are as follows:

| 4 | 2 | 126 | 0 | 18 |

This basically means that number 3 was 4 times mistaken for number 1, 2 times mistaken for number 2, recognized correctly 126 times, and mistaken for number 5, 18 times.

Figure 17 shows the first 10 falsely classified images for number 3. As one can see 5 is very often represented, since the lower half of a 5 looks like a 3. Also some 1s were also falsely classified as 3.

Figure 17: First 10 false classified images for number 3

4 SVM with Gradient Descent

Report the optimal parameter $\theta^* = (\mathbf{w}, b)$ found and the final value of the cost function.

The learning rate was set to $\eta = 0.08$ and the maximum amount of iterations to max_iter = 20. Using bigger learning rates caused the cost function, evaluated at each iteration, to jump the 'hill' up down and it did not truly converge. The penalty Term was set to $C = 1$, as given in the assignment sheet. This resulted in the parameters given below.

$$w_{opt} = \begin{bmatrix} -0.2261 \\ 0.2171 \end{bmatrix} \qquad b_{opt} = -0.01400 \tag{1}$$

$$J(\theta^*) = 0.06105137 \tag{2}$$

Report the accuracy obtained on the test set.

Equation 3 can be used to classify the test data.

$$\hat{y}_i = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x}_i + b \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

Using this on the non-seen test set, the accuracy evaluates to

$$\text{acc} = 100.0\% \tag{4}$$

Use the function plot_decision_function in svm_plot.py to plot the decision function of your trained support vector machine. Compare it to the plot from Task 1 .

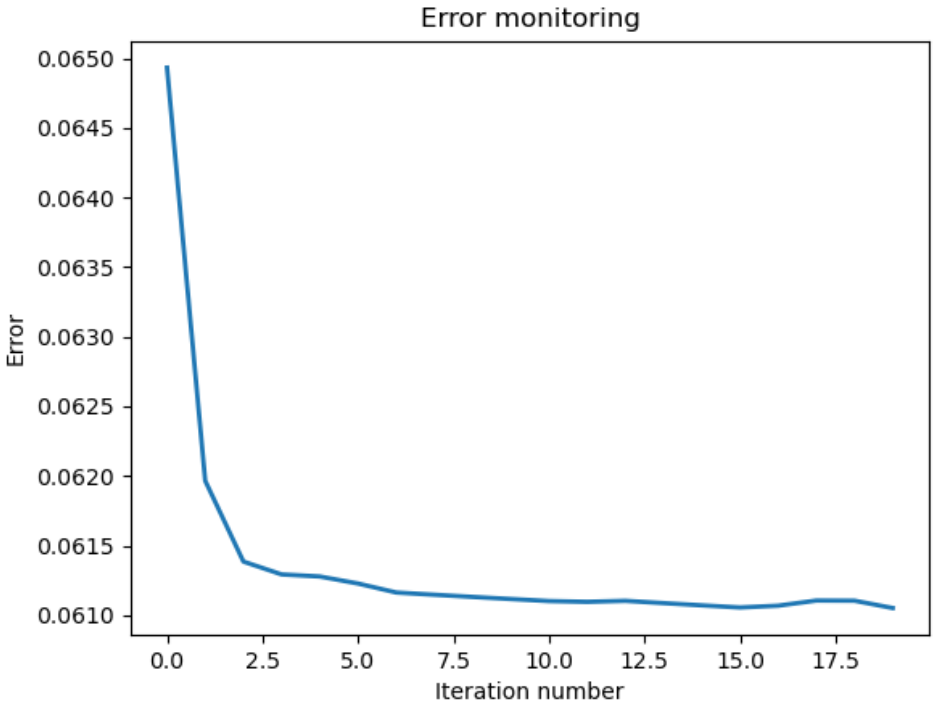

Figure 18: cost function over iterations

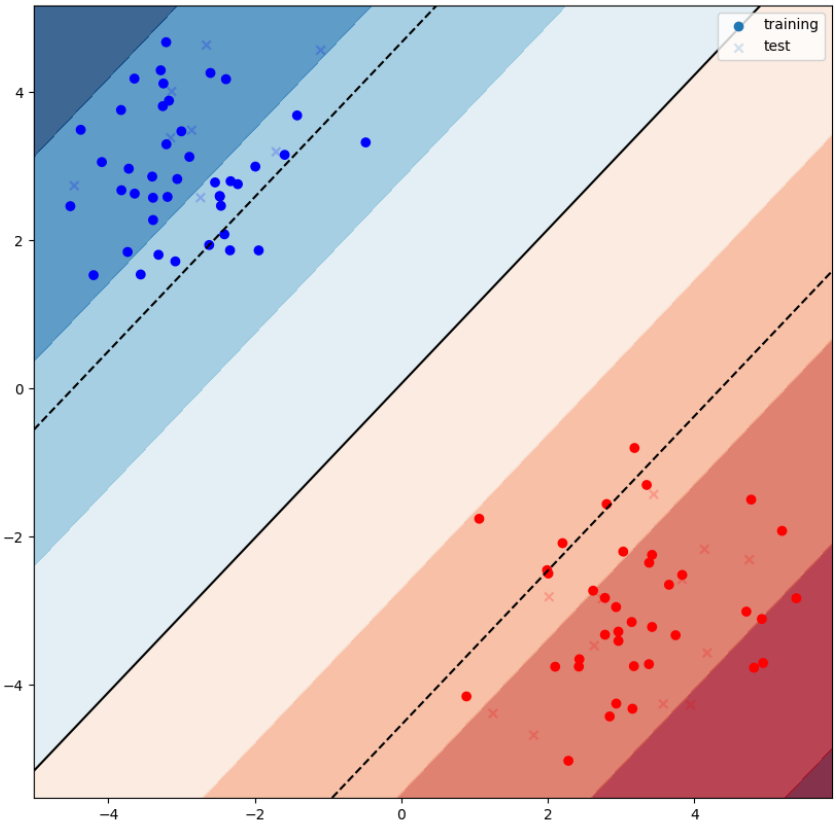

Figure 19: trained decision function

Unlike in Task 1, the support vector lines don't match with the most 'innerst' training sample, although in both cases the penalty factor is $C = 1$. Due to this, the decision boundary is not in the same spot as in task 1.

What is the main drawback of minimizing the support vector machine given in the form of Equation 5?

$$\min_{\mathbf{w},b} J(\mathbf{w},b) = \min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + \frac{C}{m}\sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b)) \tag{5}$$

Using the above formula the feature space cant be transformed into a higher space, thus not allowing to classify non-linear data in fashioned manner.