Exercise 4

Pattern Recognition, Fall 2021

Nico Aebischer
Max Jappert

Theory/Coding questions

1. In linear SVM, an optimal hyperplane can only directly be found, when the vectors are separable and the hyperplane has to fulfill the following two criteria:

   - Separates the two classes.
   - Maximizes the separation (miminum margin) between the two classes.

   More generally, the SVM relies on the hyperplane. The decision surface, defined by the separating hyperplane, should not be affected by values that are far away from the hyperplane. In contrast, only the vectors closest to the hyperplane should affect the actual separation.

2. The linear toy plots for `C = None` and `C = 10` respectively:

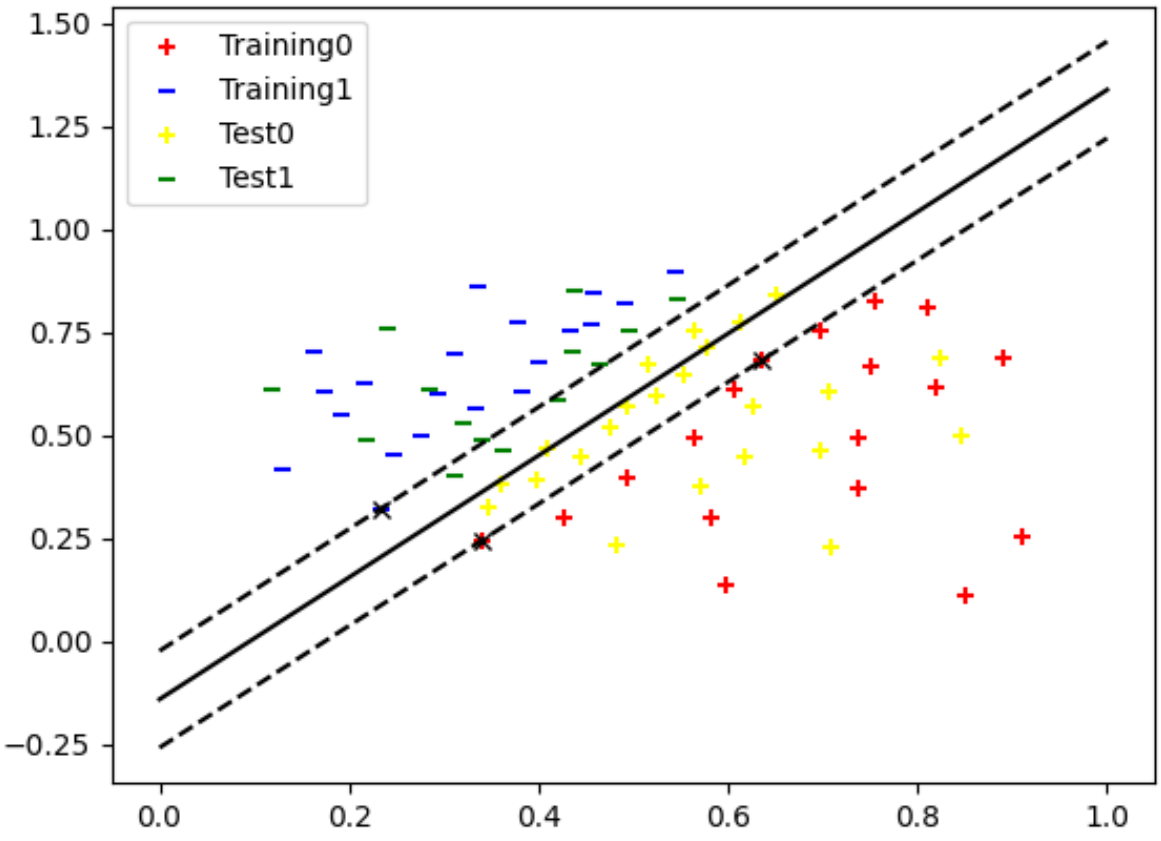


Figure 1: `C = None`

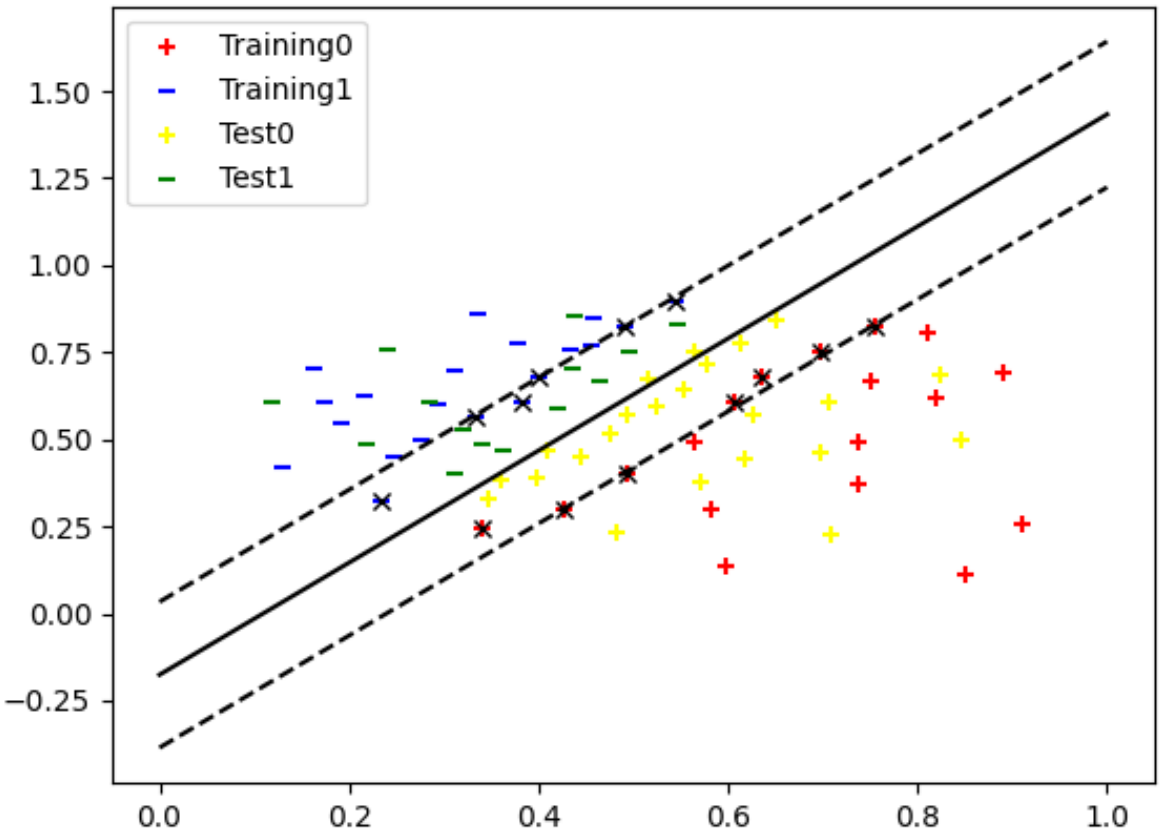


Figure 2: `C = 10`

3. A support vector is a member of the set of datapoints from the training data which lie closest to the dividing hyperplane. They are being classified as critical to the SVM, because if they were removed, the position of the hyperplane would be altered as well.

   In the code we compute them by first computing the λ_i parameters by means of quadratic programming with the `cvx` library. The support vectors are then those datapoints x_i with a corresponding λ_i, such that $\lambda_i > 0$ holds.

4. For `C = None` we have training error of 0% and a total error of 13.51%. For `C = 10` we have the same training error with a total error of 5.41%.

5. With the help of the C parameter, one can adjust, how much training errors should be accepted or in other words, we introduce a soft margin for cases where the classes are not entirely separable. In SVM, one has to choose how much attention should be put on two things:

   - Yielding a hyperplane with a large minimum margin.
   - Yielding a hyperplane, that correctly classifies as many data points on the training set as possible.

   Those two aspects do not usually complement each other.
   Larger values for C produce fewer training errors, which can be good, but that also results in a lower generalization performance and fewer support vectors. With lower values of C, one allows more

misclassified data points on the training set (e.g. neglecting outliers).
Depending on the data that one wants to predict in the future, either larger or lower values of C suit the problem better. C is usually adjusted by a trial and error process on a validation data set.

6. The main difference between SVM and the perceptron classifier is the condition on which the algorithms stop. The perceptron aims to classify all data correctly. The SVM on the other hand aims to find the hyperplane with the largest minimum margin, in other words, the maximum distance between data points of both classes. To get to this point, some outliers might be neglected. This aims at correctly classifying future data points, at which the SVM usually does a better job than the perceptron.
   Also, the perceptron can be used when the training examples are linearly separable. In SVM one does not have to rely on linear separability, since with the help of soft margins and the kernel functions, the SVM can also work with linearly non-separable data.

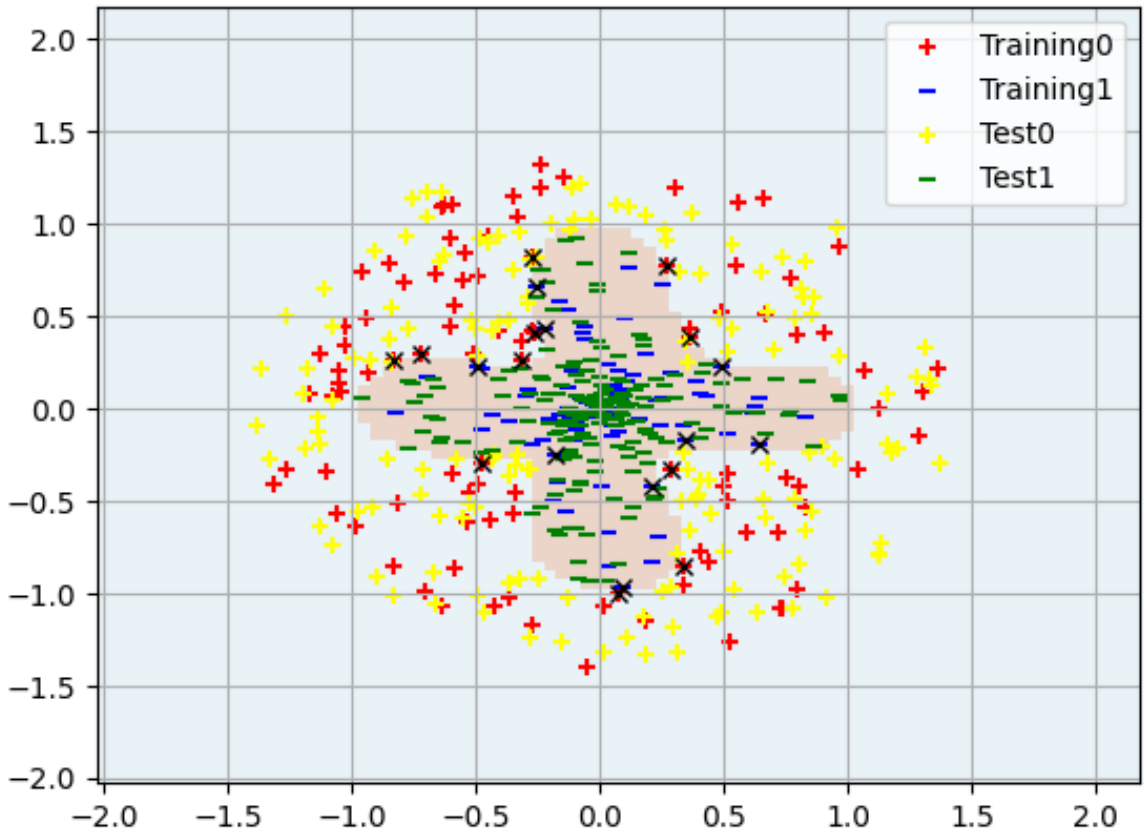
   By applying the least squares classifier, data points which lie far away from the hyperplane also have an influence on it and thus alter its location, not just the support vectors which lie closest. This results in worse classifying results on future data, since outliers which should be neglected in training actually led to a different decision surface.

7. The Sigma parameter in the RBF in the RBF kernel influences the smoothness of the classification. For large Sigmas, the decision surface is smooth, but it tends to make wrong classifications. This avoids overfitting. Smaller Sigmas on the other hand lead to a sharper decision boundary where overfitting can occur.
   Thus, we would more frequently observe islands around individual data points as sigma decreases. Consider the kernel function of the RBF:

$$K(x, x') = exp^{-\frac{||x-x'||^2}{\sigma^2}}$$

   Sigma acts as an amplifier for the distance between two points. Small values of Sigma result in small evaluations of the kernel function and thus only the points within a certain distance (very close) can affect the predicting point. Large values of Sigma on the other hand let more points influence the prediction of a certain point, thus the resulting smooth decision surface.

8. The decision surface of the toy kernel dataset with the required parameters:



9. We did this by running each case 1000 times and measuring the time before and after the iterations. The average time for each iteration is computed by dividing the execution time by 1000. We get the following results:

```
Linear SVM timing:
0.0144085050 over 1000 runs
SVM with linear kernel timing:
0.0147617774 over 1000 runs
Linear is 1.0245183271240443 times faster
```

10. Although, for some reason unknown to us, the classification times end up being nearly equal in our computation, it would make sense that the kernel classifier takes longer. The linear classification function involves one loop, iterating n times for n data points and a simple dot product calculation for each. The kernel classification, on the other hand, has a nested loop over the lambdas and corresponding support vectors within the loop over all n data points. In addition, per iteration, the classification function involves a computation with a kernel, which is computationally expensive.

    In that regard, it seems mysterious to us that our classifications require an equal amount of time.

11. - ZIP13: polynomial kernel, C=100, kernelpar = 0.3. This results in a training error of 0% and a test error of 0.7%.

- ZIP38: polynomial kernel, C=10 kernelpar = 0.2. This results in a training error of 0% and a test error of 2.71%.
- Ship: linear kernel, C=1, kernelpar = 0.5. This results in a training error of 0% and a test error of 25.94%.