Assignment 3

Computational Intelligence, SS2018

Team Members		
STRUGER	Patrick	01530664
BÖCK	Manfred	01530598
HAUPT	Anna	01432018

Contents

1 Linear SVM

The file data.json contains the training set (X, Y) of a binary classification problem with two input dimensions (X is the set of 2-dimensional points, Y are labels). Your task is to use a linear soft-margin SVM to solve this classification problem and to inspect how its parameter C, the positive cost factor that penalizes misclassified training examples (slack variables), influences the decision boundary found by SVM. In function *ex_1* in file svm_main.py the data points are first loaded from the data file and plotted so that we can observe the structure of the data. Do the following:

(a) Fill in function ex_1_a in file svm.py to train an SVM with a linear kernel with the provided training data $'x'$ and $'y'$. Plot the training points, decision boundary and support vectors using function plot_svm_decision_boundary in file svm_plot.py (passing in only $'x'$ *and* $'y'$ as parameters $'x\_train'$ and $'y\_train'$).

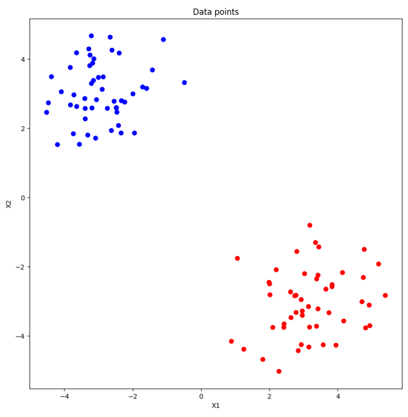

Figure 1: Data structure of the data points.

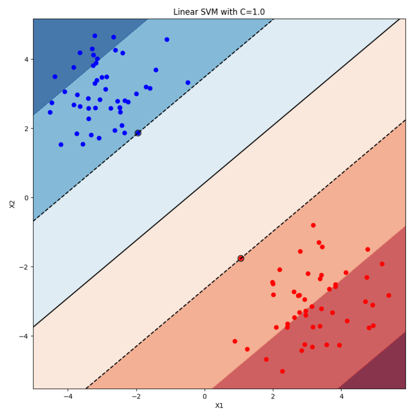

Figure 2: The decision boundary and support vectors.

(b) In function ex_1_b in file svm.py add an additional data point (4,0) with label 1 to the data. Train a linear SVM again on this extended data. Plot results using plot_svm_decision_boundary.

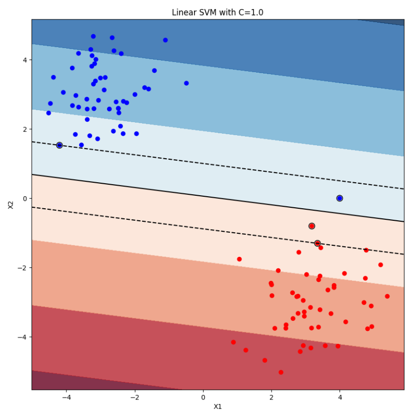

Figure 3: The decision boundary and support vectors after adding additional data point.

For task b), discuss how and why the decision boundary changed when the new point was added.
After adding a new data point at *(4, 0)* with *label1* to the data set, the decision boundary gets more horizontal because the SVM tries to find a decision boundary to separate the classes best.

(c) In function ex_1_c in file svm.py, again extend the dataset as above, and train linear SVMs with different values of the parameter C : 10^6, 10^0, 10^{-1}, and 10^{-3}. Plot the decision boundaries and support vectors for each of these values of C.

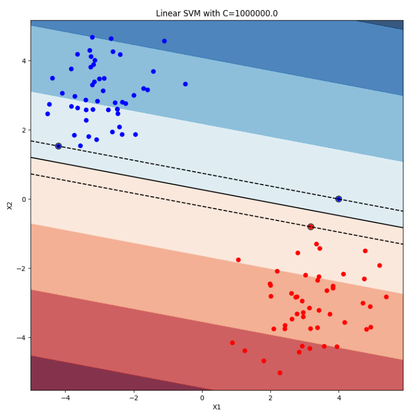

Figure 4: The decision boundary and support vectors with parameter $C = 10^6$.

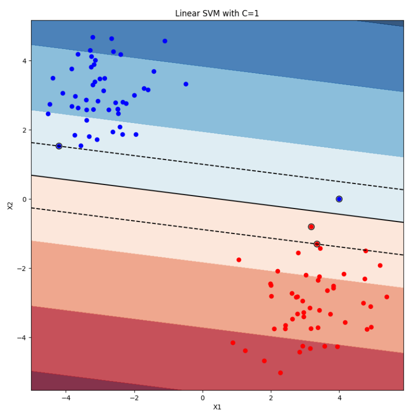

Figure 5: The decision boundary and support vectors with parameter $C = 10^0$.

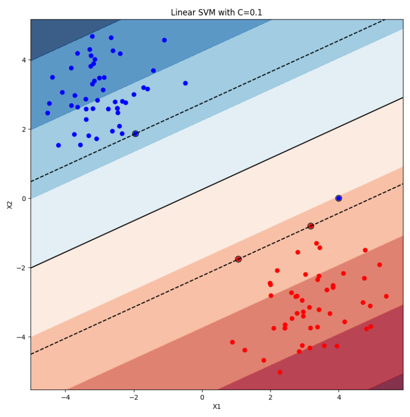

Figure 6: The decision boundary and support vectors with parameter $C = 10^{-1}$.

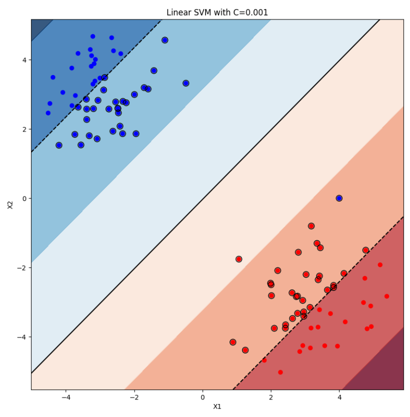

Figure 7: The decision boundary and support vectors with parameter $C = 10^{-3}$.

For task c):

- **Report how the parameter C influences the decision boundary found by the SVM.** By reducing the cost factor C, the SVM tries to find decision boundaries where all data points of the same class are included. If the cost factor C gets smaller, the SVM tries to focus on the main occurrence of a data point class. As shown in 7 the SVM focuses on the main occurrence of the red data points and ignores the additional blue one.

- **How does the number of support vectors found by the SVM change with the value of C? Why?** The number of support vectors decreases by an increase of the cost factor C. If the value of C increases, so the SVM tries to find a decision boundary to include data points of same class, which turns out that the support vectors are only the closest vectors to the decision boundary. While decreasing the cost factor C the number of data points not belonging to the same class increases and so the number of support vectors increases as well.

2 Nonlinear (kernel) SVM

The file data_nl.json contains the training (X, Y) and test (XT, YT) sets of a binary classification problem with two input dimensions. The goal of this task is to use different kernels (linear, polynomial and RBF) in combination with SVM in order to solve this nonlinear classification problem and to inspect the influence of kernel parameters on the classification error.

In function ex_2 in file svm_main.py the data points are first loaded from the data file and plotted so that we can observe the non-linear structure of the data (The training points are plotted in dark colors, and the testing points in lighter colors).
We consider the following kernels:

- Linear kernel: $K(x, y) = x^T y$
- Polynomial (non-homogeneous) kernel: $K(x, y) = (x^T y + r)^d$
- RBF kernel: $K(x, y) = exp(-\gamma \|x - y\|^2)$ where $\gamma = \frac{1}{2\sigma^2}$

Do the following:

(a) Fill in function ex_2_a in file svm.py to train and test an SVM with a linear kernel with the provided training data *'x_train'* and *'y_train'*. Plot the training points, decision boundary and support vectors using function plot_svm_decision_boundary in file svm_plot.py. Calculate the test score.

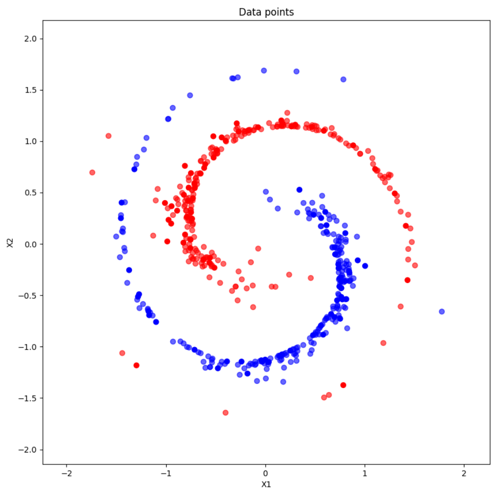

Figure 8: Data structure of the data points.

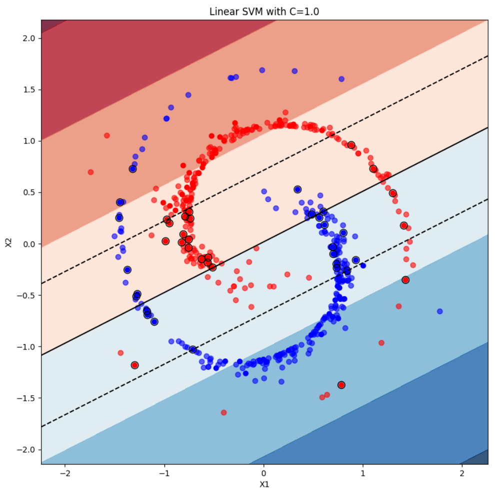

Figure 9: The training points, decision boundary and support vectors using linear kernel.

(b) In function ex_2_b in file svm.py train and test SVMs with a polynomial kernel for different degrees of the polynomial $d = 1, 2, ..., 19, 20$ (Setting the value of r to 1). Plot the variation of training and testing score with polynomial degree using function plot_score_vs_degree in file svm_plot.py. Plot the decision boundary and support vectors for the kernel with the highest test score using plot_svm_decision_boundary.

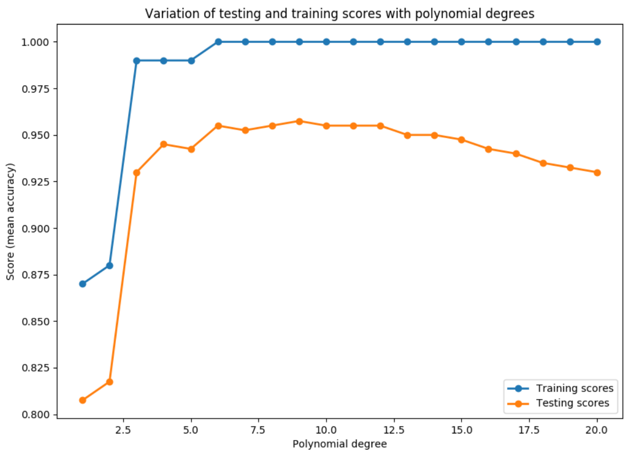

Figure 10: Variation of training and testing score with polynomial degree.

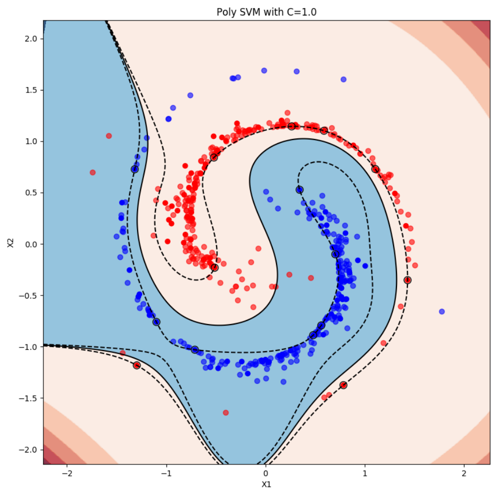

Figure 11: Decision boundary and support vectors for the kernel with the highest test score.

For task b) which degree of the polynomial produces the highest test score (accuracy)? Report this test score.
Degree 9 produces the Highest Test Score 0.9575

(c) Fill in function ex_2_c in file svm.py to repeat the above steps for an RBF kernel with values of $\gamma = 0.01, 0.03, ..., 1.97, 1.99$. Plot the variation of training and testing score with the value of γ using the function plot_score_vs_gamma in file svm_plot.py. Plot the decision boundary and support vectors for the kernel with the highest test score using plot_svm_decision_boundary.

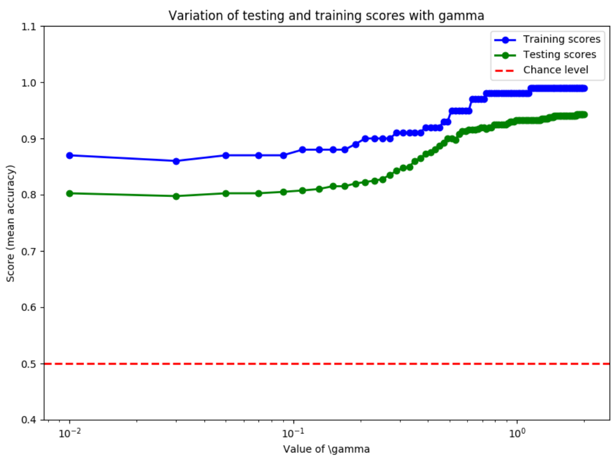

Figure 12: Variation of training and testing score with RBF kernel.

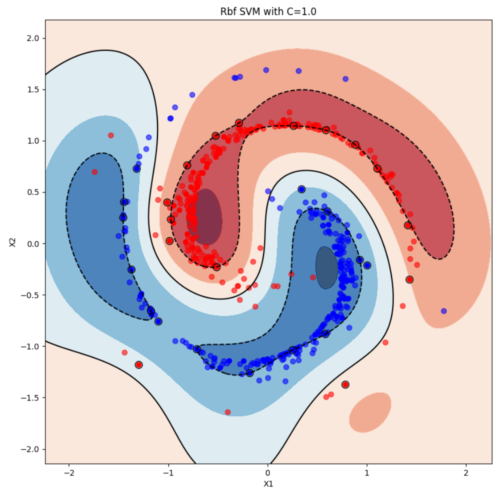

Figure 13: Decision boundary and support vectors for the kernel with the highest test score.

- **For task c) which value of gamma produces the highest test score (accuracy)? Report this test score.**
  A Gamma value of 1.85 produces the Highest Test Score 0.9425
- **Compare results obtained by each of these three kernels:**
  (a) State the maximum test score achieved for each of these kernels and the kernel parameter for which that was achieved.

Kernel Type	**Kernel Parameter**	**kernel**
linear	/	0.8125
poly	Degree: 9	0.9575
rbf	γ: 1.85	0.9425

Table 1: Highest Test Score for different Kernel Types

  (b) **Which of the considered kernels performs best and why?**
  Of the three kernel types, the kernel type *poly* scored the highest score because it focuses on classifying data points corresponding to the same class as possible. Which performs better in the general way while the rbf kernel focuses on classifying data points on the most occurrence of data points of the same class. So this is better for striking features.

  (c) **Compare the complexity of decision boundaries and the number of Support Vectors found.**
  The decision boundary for the *linear* kernel is the less complex than the other kernels, while the *rbf* kernel is the most complex one. The *rbf* kernel got the most support vectors while the number of the support vectors of the other kernels is the same.

  (d) **Which kernel generalizes best for the given dataset?**
  For the given dataset the poly Kernel generalizes best.

3 Multiclass classification

For this task you will use a linear SVM as a binary classifier to solve a multiclass problem. For this purpose use the data file data_mnist.json which contains training (X, Y) and test (XT, YT) sets of handwritten digits. To simplify the problem we consider only the first 5 digits: $1, ..., 5$. Each data sample in X (a handwritten digit) is an image of 28×28 pixels. Note that here the labels are not binary, but rather each data sample belongs to one of the 5 classes. Load the data file and plot several data samples to see handwritten digits (use plot_mnist). The goal of this task is to use the one-vs-all approach with SVM for multiclass classification in order to recognize digits. In all simulations of this task use the parameter $C = 10$. In function ex_3 in file svm_main.py the data points are first loaded from the data file and plotted so that we can observe the structure of the MNIST dataset.

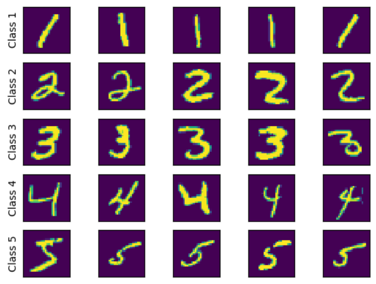

Figure 14: Dataset with handwritten digits.

Do the following:

(a) In function ex_3_a in the file svm.py, use the scikit learn implementation of one-versus-rest multiclass classification. This is done by specifying the argument $decision\_function\_shape =' ovr'$ when creating the classifier object. Train the classifier on the MNIST dataset for a LINEAR kernel and a RBF kernel with gamma ranging from 10^5 to 10^{-5} (plot this for 10 values of gammas in between). Plot the results with plot_score_vs_gamma, and specify the score obtained with a linear kernel using the optional argument lin_score_train. Note that the chance level has changed for this example.

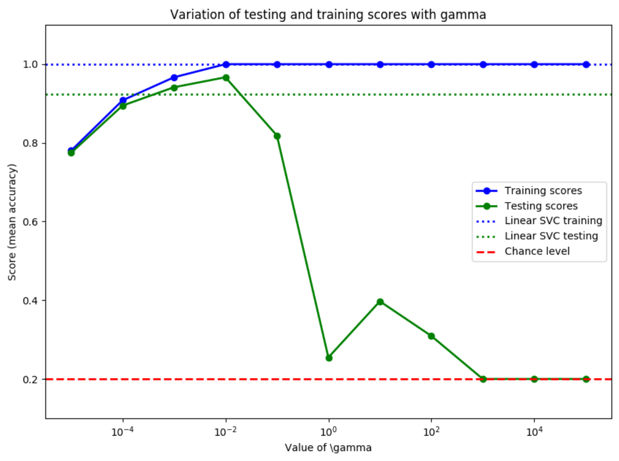

Figure 15: Classifier on the MNIST dataset for a LINEAR kernel and a RBF kernel with gamma ranging from 10^5 to 10^{-5}

In the results of variation between the *rbf* kernel and the *linear* kernel the *linear* kernel performs better than the *rbf* one because it just differentiate between lighter and darker colors.

(b) In function ex_3_b in the file svm.py, train a multi-class SVC with a linear kernel. Use the function confusion_matrix from scikit-learn to get the confusion matrix. Plot the confusion matrix with plot_confusion_matrix. Plot the first 10 images from the test set of the most misclassified digit using plot_mnist. (For example, if digit 3 is the most misclassified digit, plot the first 10 images miss-classified as 3 in the test set)

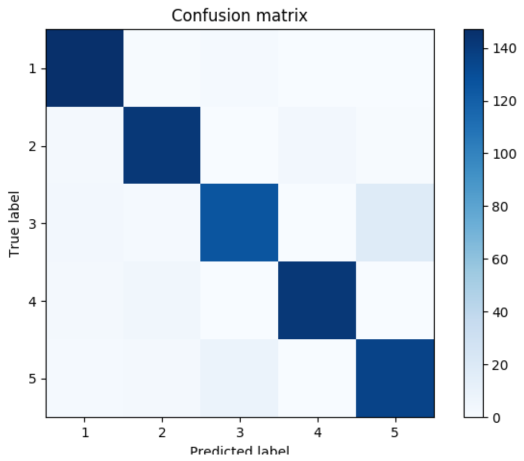

Figure 16: Confusion matrix.

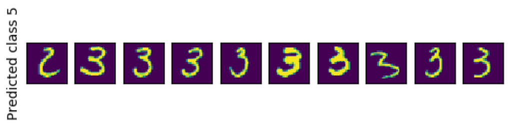

Figure 17: 10 images from the test set of the most misclassified digit.

In your report:

- **Recall the algorithms 'One-versus-Rest' (or versus-all) and 'One-versus-one' multi-class classification procedures. How many binary classifiers need to be trained in both cases?**

  The **one-versus-all** algorithm fits with one classifier per class. For each classifier the class is fitted against all other classes and is computational efficient because it only has to train n classes classifiers $\Rightarrow O(n)$.

  The **one-versus-one** algorithm concentrates on fitting one classifier per class pair. It is less computational efficient than the The one-versus-all because it has to train $nclasses * (nclasses - 1)/2$ classifiers $\Rightarrow O(n^2)$.

- **Find the digit class for which you get the highest error rate.** Digit class 5 produces the highest error rate.

- **With the help of these two plots, discuss why the classifier is doing these mistakes.** The digit 3 is the most misclassified image. This might appear because some of the handwritten 1 and 3 digits are similar to the digit 5 and so the most misclassified class is 5.