ENGR421

HW6

Doğa Demirtürk
68859

In this homework, we implement a one-versus-all support vector classification algorithm in Python.

First, I started by reading the data into memory and dividing it into two sets: first 1000 data points assigned to training set and remaining 4000 data points assigned to test set.

After that I started investigating the lab 8 to better understand SVM implementation. I also get help from my lecture notes to understand the steps of the algorithm.

I first implemented an algorithm with use of lab 8 called Gaussian_kernel to calculate kernel matrix. Then I implemented a function called SVM that takes parameters Y and C since we change C in the last step of the homework, and we need to adjust y values for each class in one-versus-all approach.

To implement one-versus-all approach, we need to have K support vector machine results as alphas and w0s since we need to set y to 1 for each class separately while setting all the rest to -1. In the Figure 1, the implementation of K different SVM parameters is shown.

```
# calculate Gaussian kernel
K_training = gaussian_kernel(x_training, x_training, s)

SVM_alphas = []
SVM_w0s = []
for i in range (K):
    y_adjusted = np.ones(y_training.shape[0])
    y_adjusted[y_training != i+1] = -1
    alpha, w0 = SVM(y_adjusted, C)
    SVM_alphas.append(alpha)
    SVM_w0s.append(w0)
SVM_alphas = np.array(SVM_alphas)
SVM_w0s = np.array(SVM_w0s)
```

Figure 1: K different SVM values

Then I used those alpha and w0 values to predict y labels for training set. Again, I got help from lab 8 to predict y values. I predict for each SVM and take the class of the maximum

value for prediction. Then I created the confusion matrix and results were the same as given in the pdf.

y_train	1	2	3	4	5
y_predicted					
1	207	1	0	9	0
2	2	199	1	1	0
3	0	1	204	6	0
4	0	1	4	185	1
5	0	0	0	0	178

Figure 2: Confusion matrix for training data set

Predicting the test data set was harder since there was no implementation of it as an example in the lab 8. I looked from the book to understand where to use which value while predicting. It was challenging to adjust vector and matrix dimensions. Finally, I got a correct result.

y_test	1	2	3	4	5
y_predicted					
1	641	23	3	137	9
2	43	714	27	40	4
3	4	39	666	90	10
4	100	32	69	541	16
5	12	2	6	15	757

Figure 3: Confusion matrix for test data set

For the last part of the assignment, I merged the codes I used for the previous parts and defined a new function called calc_accuracy which predicts y values with given C value for both training and test data sets and returns the accuracy of the predictions. Then I used those accuracy values to draw the desired graph.

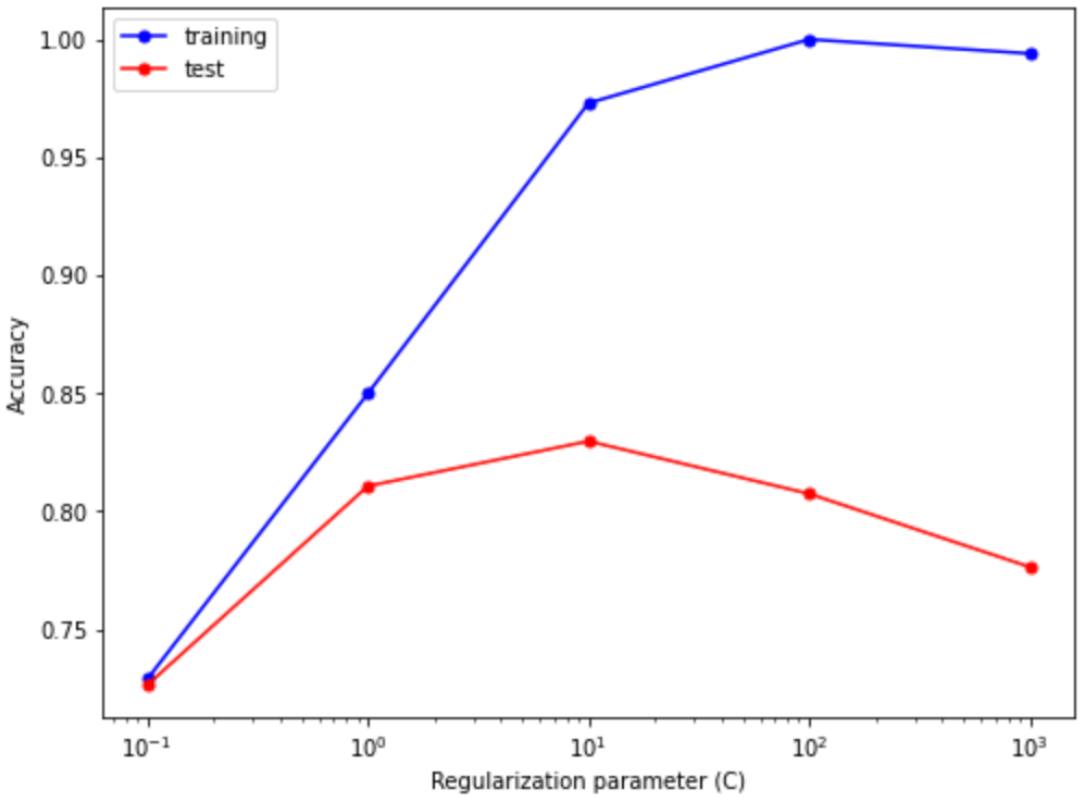

Figure 4: Regularization parameter - Accuracy Graph