CSE4088 Intro. to Machine Learning

Homework 4 Report

Muhammed Fatih Öztel

150119907

SVM with Soft margins

To solve this homework I used scikit-learn and numpy libraries. scikit-learn has functions to solve SVM question.

Question 2

First I open the related train and test data given for homework using numpy. We will use these data for all questions. To solve this question and following questions I created a function that return the X of given n agains the all dataset. This function takes the number and the dataset. First it finds the class with n then sets the related label of this class as 1 and others as -1. Finally return the data and the label. As question wants for 0 to 8 even numbers versus all, we will iterate for all even numbers and for each iteration we will create an SVM model using scikit learn with given related parameters C=0.01, kernel as poly, degree as 2 and gamma as 1.0. Then we will get the related x and y labels from that function we created, we give train data here. then we fit this into our model. After fitting we ge the predicted Ys using predict function of our model with x. To find E_{in}, we compare the predictions with the real labels and get the mean of it. Question wants us to find the highest E_{in} and in this example, 0 has the highest E_{in} as seen in the picture below. The answer is 0, A

```
print(max(final_ein2))
# 0 has the maximun
```

```
(0.10588396653408312, 0)
```

Output 1

Question 3

Question 3 is the same as question 2 we just look for the lowest E_{in} and for different numbers. so we iterate through odd numbers and do same process. The answer is 1, A

```
# 1 has the minimum
print(min(final_ein3))
```

```
(0.014401316691811822, 1)
```

Output 2

Question 4

For question 4 we use the answers we get from questions 2 and 3. So we will iterate over 0 and 1 only. I do same process but dont evaluate E_{in} since we dont need it in this question rather I put the related models of 0 and 1 into a list to store. To find the difference between the number of support vectors of these 2 classifiers, I use model object and get number of vectors. scikit learn library provides a functinality that returns the list of support vectors which is support_. So for each calssifier I get the difference of length of these lists.Answer is close to 1800, C

```
len(clfs[0].support_) - len(clfs[1].support_) # get the number of vector and
# which is close to 1800
```

```
: 1793
```

Output 3

Question 5

For question 5 I created another function that takes 2 numbers for class number and the dataset. This function is similar to that function we used for previous questions. Different is it only labels the class for the given number m.

As question wants we will test for different C numbers, I created a list of C that we will iterate through it. First, we get the related data and the labels for the train and test dataset from a function that we created. then we iterate over C list, so for each iteration, we change the C parameter of our classifier. For iteration, we create a classifier with related parameters then fit the train data that we get from our function. Finally, we find a number of support vectors, E_{in} and E_{out}. For E_{out} we only compare the test data with predicted data. and we print the c, a number of vectors, E_{in} and E_{out} to see and analyze the situation. The answer is D. Maximum C achieves the lowest E_{in}.

```
| C | Number of vectors | E in | E out |
0.001 76 0.004484304932735426 0.01650943396226415
0.01 34 0.004484304932735426 0.018867924528301886
0.1 24 0.004484304932735426 0.018867924528301886
1 24 0.0032030749519538757 0.018867924528301886
```

Output 4

Question 6

Question 6 is similar to 5, but we look for model degree 2 and 5 for different Cs. First we get data and labels for train and test sets. ten while iterating for Cs, do 2 classifiers. One for degree 2 and the other for degree 5 as the question wants. and find related number of support vectors, E_{in} and E_{out} of this 2 classifiers for each iteration and print the related numbers to analyse and get the answer. So the answer is B,When C = 0.001, the number of support vectors is lower at Q = 5.

```
# When C = 0.001, the number of support vectors is lower at Q = 5.
```

```
c:0.0001 v2:236  Ein2:0.008968609865470852 Eout2:0.01650943396226415
        v5:26  Ein5:0.004484304932735426  Eout5:0.018867924528301886
--------------------------------------------------------------------------------
c:0.001 v2:76  Ein2:0.004484304932735426 Eout2:0.01650943396226415
        v5:25  Ein5:0.004484304932735426  Eout5:0.02122641509433962
--------------------------------------------------------------------------------
c:0.01 v2:34  Ein2:0.004484304932735426 Eout2:0.018867924528301886
        v5:23  Ein5:0.003843689942344651  Eout5:0.02122641509433962
--------------------------------------------------------------------------------
c:1 v2:24  Ein2:0.0032030749519538757 Eout2:0.018867924528301886
        v5:21  Ein5:0.0032030749519538757  Eout5:0.02122641509433962
--------------------------------------------------------------------------------
```

Output 5

Question 7

In question 7, we need a dictionary to store a number of selected c values from folds and a list of best cross-validation errors from every iteration. Since the question wants us to run 100 times we will split data 100 times. Thankfully sci-kit learn has a method so we can easily do split data into 10 and shuffle it. Then for every iteration, we iterate over the C values list to find cross-validation. We do the same process as we did in previous questions while finding errors, we just give the related fold data index to find that fold error. Then we fit it and get the validation error. We add this error to the related cross-validation error dictionary of that run with the related C value key. After 10 fold, we check for which C value has the lowest value and we get this as the best to increment the number of occurrences at the public dictionary that we created at the beginning. Also, we have to add the best c value by taking its average to the winning list to solve question 10. After 100 runs we check which C value selected most. So the answer is 0.001, B.

```
print(max(selected, key=selected.get))
# 0.001 is selected most often, Answer B
```

```
0.001
```

Output 6

Question 8

We just get the average of the winning cross-validation errors. So the answer is close to 0.005, C

```
print(np.mean(winning_E_cvs))
# Answer is close to 0.005, C
```

```
0.0044526375959496965
```

Output 7

Question 9

For question 9 have a list of C values for classifiers and create a classifier in every iteration with different c values with kernel RBF. In sci-kit learn we just set the parameter kernel=”rbf” is enough and check for E_{in} and E_{out} with the same way we did for previous questions and add to a list to find the lowest E_{in} and E_{out}. So the answer is E, $C = 10^6$

```
min(all_ein)
#10^6 has the lowest e in so C,
```

```
(0.0006406149903907751, 1000000)
```

Output 8

Question 10

Since we add E_{in} and E_{out} to the list in question 9, we just get the lowest E_{out} from that list. So the answer is C, C = 100

```
#  100 is the lowest for e out
min(all_eout)
```

```
(0.018867924528301886, 100)
```

Output 9