COL341 ASSIGNMENT - 2

Brief description of each parts

Binary SVM

`fit(self, train_data_path:str) -> None:`
This function finds X and Y and calls `binary_classifier(self , X, Y)` function
We first find all the matrics - `P, q, G, h, A, b` and calculate α by using `alpha = np.array(qpsolvers.solve_qp(P, q, G, h, A, b, solver="cvxopt"))`
Using alpha the value of b is calculated. I have taken the avg of all the values we get from each support vector.

I have saved all the support vectors (α , X , Y) with the trainer

`predict(self, test_data_path:str)->np.ndarray:`
This function first extracts X_test from the given file and calls `binary_pred(self , X_test )` function. Using kernel we calculate the value of Y for each data point

Confusion matrix for validation part

	0	1
0	22	1
1	3	16

Virtically → actual value
Horizontally → predicted value

Best Hyperparameters

kernel → RBF
C → 1
gamma → 0.0001

Analysis

Linear kernel

C = 0.01	C = 0.1	C = 1	C = 10
89.74%	84.61%	84.61%	84.61%

Confusion matrices

C = 0.01

	0	1
0	21	2
1	3	16

C = 0.1

	0	1
0	19	4
1	3	16

C = 1

	0	1
0	19	4
1	3	16

C = 10

	0	1
0	19	4
1	3	16

RBF kernel

	C = 0.01	C = 0.1	C = 1	C = 10
Gamma = 0.1	53%	53%	53%	53%
Gamma = 0.01	53%	53%	79%	79%
Gamma =0.001	89%	92%	87%	87%

Multi-class Classification

OVO

First I have initiazed ${}^{n}C_{2}$Trainers using `_init_trainers(self)`

```
fit(self, train_data_path:str, max_iter=None) -> None
```

This function extracts X and Y from the csv file.
Now we iterated through the trainers. and train them
for trainer number [i,j] (where i iterate from 1 to n_classes and j iterate from i+1 to n_classes) we train them as follows.
We first seperate the data that belongs to class i+1 and j+1 only
We create a new Y_new such that Y_new[n] = 1 if Y[n] == i+1 else Y_new[n] = -1
we train them using the binary classifier

`predict(self, test_data_path:str) -> np.ndarray`

I run all the trained models on each data point and for (i,j) trainer, we add the weight of nth data point being in class i and we subtract the weight of that point being in class j.
Using the above voting algorith we then find the argmax for each data point and return the answer

Confusion matrix

[[2. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 2. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 5. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 2. 6. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 4. 1. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 6. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 6. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 0. 1. 2. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

Analyis

- gamma = 0.1 and C = 0.1
  Accuracy = 48.71%
  Confusion matrix -
  [[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 2. 0. 0. 0. 0. 0. 0. 0.]
  [0. 1. 5. 5. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 4. 1. 0. 0. 0. 0.]
  [2. 1. 0. 0. 0. 6. 4. 1. 3. 0.]
  [0. 0. 0. 0. 0. 0. 2. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

- gamma = 0.1 and C = 1.0
  Accuracy = 69.23%
  Confusion matrix -
  [[2. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 1. 5. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 2. 5. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 4. 1. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 6. 1. 1. 3. 0.]
  [0. 0. 0. 0. 0. 0. 4. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

OVA

First I have initiazed nTrainers using `_init_trainers(self)`

`fit(self, train_data_path:str, max_iter=None) -> None`

This function extracts X and Y from the csv file.
Now we iterated through the trainers. and train them
for trainer number i (where i iterate from 1 to n_classes) we train them as follows.
We create a new Y_new such that Y_new[n] = 1 if Y[n] == i+1 else Y_new[n] = -1
we train them using the binary classifier

`predict(self, test_data_path:str) -> np.ndarray`

I run all the trained models on each data point and ith trainer.
We just take the argmax of each trainer's answer on each data point and return that.

Confusion matrix

[[2. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 2. 1. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 4. 1. 0. 0. 0. 0. 0. 0.]
[0. 0. 2. 5. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 4. 1. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 6. 1. 1. 0. 0.]

[0. 0. 0. 0. 0. 0. 4. 1. 0. 0.]
[0. 0. 0. 0. 0. 0. 1. 0. 1. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 2. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

Analysis

- Gamma = 0.1 and C = 0.1
  Accuracy = 53.8%
  Confusion matrix -
  [[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 2. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 3. 5. 0. 0. 0. 1. 2. 0.]
  [0. 0. 0. 1. 4. 1. 0. 0. 0. 0.]
  [1. 2. 2. 0. 0. 6. 3. 0. 1. 0.]
  [0. 0. 0. 0. 0. 0. 3. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]

- Gamma = 0.1 and C = 1.0
  Accuracy = 74.35%
  Confusion matrix -
  [[2. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 2. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 5. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 2. 5. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 4. 1. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 6. 2. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 3. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 1. 0. 1. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 2. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]