Machine Learning Laboratory

(410302)

BE Sem I Honors in AI/ML

Academic Year: 2021-22

Lab Assignment No. 6

Name: Aboli Marathe

Roll Number: 41301

Branch: Department of Computer Engineering

Problem Statement:

Write a program to solve a problem using SVM in python

Objective

Train SVM classifier using sklearn digits dataset

(i.e. from sklearn.datasets import load_digits) and then,

1. Measure accuracy of your model using different kernels such as rbf and linear.
2. Tune your model further using regularization and gamma parameters and try to come up with highest accuracy score.
3. Use 80% of samples as training data size

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
import sklearn
```

```
from sklearn.datasets import load_digits
```

```
digit = load_digits()
digit
```

```
      'data': array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
              [ 0.,  0.,  0., ..., 10.,  0.,  0.],
              [ 0.,  0.,  0., ..., 16.,  9.,  0.],
              [ 0.,  0.,  0., ..., 10.,  9.,  0.],
              ...,
              [ 0.,  0.,  1., ...,  6.,  0.,  0.],
              [ 0.,  0.,  2., ..., 12.,  0.,  0.],
              [ 0.,  0., 10., ..., 12.,  1.,  0.]]),
      'images': array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
              [ 0.,  0., 13., ..., 15.,  5.,  0.],
              [ 0.,  3., 15., ..., 11.,  8.,  0.],
              ...,
              [ 0.,  4., 11., ..., 12.,  7.,  0.],
              [ 0.,  2., 14., ..., 12.,  0.,  0.],
              [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

              [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
              [ 0.,  0.,  0., ...,  9.,  0.,  0.],
              [ 0.,  0.,  3., ...,  6.,  0.,  0.],
              ...,
              [ 0.,  0.,  1., ...,  6.,  0.,  0.],
              [ 0.,  0.,  1., ...,  6.,  0.,  0.],
              [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

              [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
              [ 0.,  0.,  3., ..., 14.,  0.,  0.],
              [ 0.,  0.,  8., ..., 16.,  0.,  0.],
              ...,
              [ 0.,  9., 16., ...,  0.,  0.,  0.],
              [ 0.,  3., 13., ..., 11.,  5.,  0.],
              [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

              ...,

              [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
              [ 0.,  0., 13., ...,  2.,  1.,  0.],
              [ 0.,  0., 16., ..., 16.,  5.,  0.],
              ...,
              [ 0.,  0., 16., ..., 15.,  0.,  0.],
              [ 0.,  0., 15., ..., 16.,  0.,  0.],
              [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

              [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
              [ 0.,  0., 14., ..., 15.,  1.,  0.],
              [ 0.,  4., 16., ..., 16.,  7.,  0.],
              ...,
              [ 0.,  0.,  0., ..., 16.,  2.,  0.],
              [ 0.,  0.,  4., ..., 16.,  2.,  0.],
              [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

              [[ 0.,  0., 10., ...,  1.,  0.,  0.],
              [ 0.,  2., 16., ...,  1.,  0.,  0.],
              [ 0.,  0., 15., ..., 15.,  0.,  0.],
              ...,
              [ 0.,  4., 16., ..., 16.,  6.,  0.],
              [ 0.,  8., 16., ..., 16.,  8.,  0.],
              [ 0.,  1.,  8., ..., 12.,  1.,  0.]]]),
      'target': array([0, 1, 2, ..., 8, 9, 8]),
      'target_names': array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])}
```

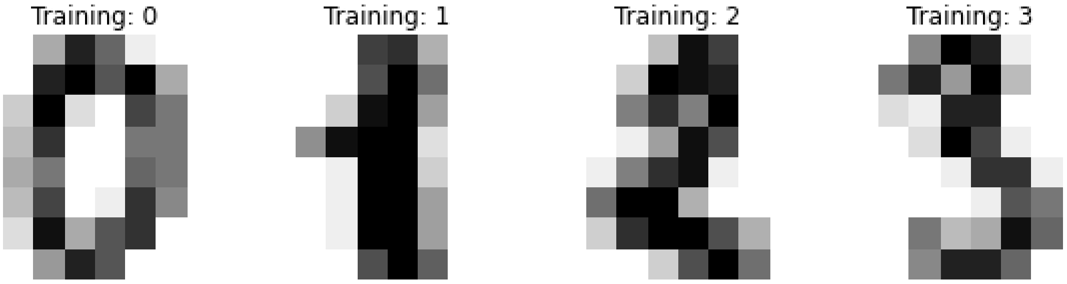
```
digit_df = pd.DataFrame(digit.data)
digit_df.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	13.0	15.0	10.0	15.0	5.0	0.0	0.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	0.0	11.0	16.0	9.0	0.0	0.0	0.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	3.0	16.0	15.0	14.0	0.0	0.0	0.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	13.0	6.0	15.0	4.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0	7.0	8.0	0.0	0.0	0.0	0.0

```
digit.images
```

```
array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ..., 15.,  5.,  0.],
        [ 0.,  3., 15., ..., 11.,  8.,  0.],
        ...,
        [ 0.,  4., 11., ..., 12.,  7.,  0.],
        [ 0.,  2., 14., ..., 12.,  0.,  0.],
        [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

       [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
        [ 0.,  0.,  0., ...,  9.,  0.,  0.],
        [ 0.,  0.,  3., ...,  6.,  0.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

       [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
        [ 0.,  0.,  3., ..., 14.,  0.,  0.],
        [ 0.,  0.,  8., ..., 16.,  0.,  0.],
        ...,
        [ 0.,  9., 16., ...,  0.,  0.,  0.],
        [ 0.,  3., 13., ..., 11.,  5.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

       ...,

       [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ...,  2.,  1.,  0.],
        [ 0.,  0., 16., ..., 16.,  5.,  0.],
        ...,
        [ 0.,  0., 16., ..., 15.,  0.,  0.],
        [ 0.,  0., 15., ..., 16.,  0.,  0.],
        [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

       [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
        [ 0.,  0., 14., ..., 15.,  1.,  0.],
        [ 0.,  4., 16., ..., 16.,  7.,  0.],
        ...,
        [ 0.,  0.,  0., ..., 16.,  2.,  0.],
        [ 0.,  0.,  4., ..., 16.,  2.,  0.],
        [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

       [[ 0.,  0., 10., ...,  1.,  0.,  0.],
        [ 0.,  2., 16., ...,  1.,  0.,  0.],
        [ 0.,  0., 15., ..., 15.,  0.,  0.],
        ...,
        [ 0.,  4., 16., ..., 16.,  6.,  0.],
        [ 0.,  8., 16., ..., 16.,  8.,  0.],
        [ 0.,  1.,  8., ..., 12.,  1.,  0.]]])
```

```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, label in zip(axes, digit.images, digit.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('Training: %i' % label)
```



```
n_samples = len(digit.images)
n_samples
x = digit.images.reshape((n_samples, -1))
print(x)
y= digit.target
print(y)
```

```
[[ 0.  0.  5. ...  0.  0.  0.]
 [ 0.  0.  0. ... 10.  0.  0.]
 [ 0.  0.  0. ... 16.  9.  0.]
 ...
 [ 0.  0.  1. ...  6.  0.  0.]
 [ 0.  0.  2. ... 12.  0.  0.]
 [ 0.  0. 10. ... 12.  1.  0.]]
[0 1 2 ... 8 9 8]
```

```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 4)
print("Shape of Xtrain data:", xtrain.shape)
print("Shape of Ytrain data:", ytrain.shape)
print("Shape of Xtest data:", xtest.shape)
print("Shape of Xtest data:", ytest.shape)
```

```
Shape of Xtrain data: (1437, 64)
Shape of Ytrain data: (1437,)
Shape of Xtest data: (360, 64)
Shape of Xtest data: (360,)
```

```
from sklearn.svm import SVC

model = SVC(kernel = "rbf")
print("The model:\n", model)
```

```
     The model:
      SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
         decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
         max_iter=-1, probability=False, random_state=None, shrinking=True,
         tol=0.001, verbose=False)
```

```
model.fit(xtrain, ytrain)
ypred = model.predict(xtest)
print("Prediction given by model\n", ypred)
```

```
     Prediction given by model
      [6 7 0 5 3 5 1 3 1 8 2 7 8 4 7 7 8 3 0 6 9 7 1 0 8 6 8 1 0 0 0 2 7 1 1 7 6
      3 1 3 4 2 9 5 2 0 0 7 3 3 2 9 7 6 1 8 5 8 6 7 5 6 9 3 1 4 1 9 7 8 4 4 2 4
      1 6 6 7 8 1 2 6 9 1 7 4 2 6 7 3 7 5 4 8 5 1 5 6 7 1 2 5 5 2 0 8 5 2 2 3 0
      4 5 6 9 3 9 5 7 4 7 8 9 4 9 7 9 7 9 4 3 0 5 4 9 2 3 2 9 6 2 6 0 5 5 8 9 2
      4 3 4 4 2 0 9 8 4 3 6 6 2 9 7 1 5 7 6 0 5 3 2 3 1 3 2 6 6 0 8 2 5 7 6 8 4
      6 2 2 0 4 0 3 0 4 0 1 5 6 4 7 1 5 4 5 5 3 4 4 6 3 7 1 1 3 5 7 5 0 1 9 5 0
      8 7 4 0 6 6 5 0 2 4 2 9 4 0 6 2 9 1 9 6 3 9 0 8 3 1 2 1 3 2 0 9 0 7 5 9 1
      8 6 9 6 8 8 6 2 4 5 9 9 1 5 2 8 4 7 9 8 8 0 1 7 3 2 2 1 0 3 2 3 9 7 2 0 0
      1 2 6 0 9 9 7 8 5 4 0 0 1 5 7 1 0 3 9 8 5 4 7 0 4 9 5 6 0 8 2 0 5 2 3 2 2
      4 2 8 7 5 8 8 6 9 2 6 4 5 9 5 4 1 7 1 7 3 4 8 5 4 3 7]
```

```
from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(ytest, ypred)
print("Confusion Matrix is:\n", matrix)
```

```
     Confusion Matrix is:
      [[38  0  0  0  0  0  0  0  0  0]
      [ 0 32  0  0  0  0  0  0  0  0]
      [ 0  0 41  0  0  0  0  0  0  0]
      [ 0  0  0 32  0  1  0  0  0  0]
      [ 0  0  0  0 37  0  0  0  0  0]
      [ 0  0  0  0  0 38  0  0  0  0]
      [ 0  0  0  0  0  0 35  0  0  0]
      [ 0  0  0  0  0  0  0 38  0  0]
      [ 0  2  0  0  0  0  0  0 31  0]
      [ 0  0  0  0  0  0  0  0  0 35]]
```

```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(ytest, ypred)
print("Accuracy of model: {}%".format(accuracy*100))
```

```
     Accuracy of model: 99.16666666666667%
```

```
# Linear kernel

model_linear_kernel = SVC(kernel = "linear")
model_linear_kernel.fit(xtrain, ytrain)
print("Accuracy of model for linear kernel: {}%".format(model_linear_kernel.score(xtest, y
```

```
     Accuracy of model for linear kernel: 98.05555555555556%
```

```
# Experimenting with regularization and gamma parameters

C_2d_range = [1e-2, 1, 1e2]
gamma_2d_range = [1e-1, 1, 1e1]
classifiers = []
for C in C_2d_range:
    for gamma in gamma_2d_range:
        clf = SVC(kernel="linear",C=C, gamma=gamma)
        clf.fit(xtrain,ytrain)
        ypred = clf.predict(xtest)
        accuracy = accuracy_score(ytest, ypred)
        print("----------------------------------------------------------------")
        print("Regularization parameter lambda = " , C)
        print("Gamma parameter = " , gamma)
        print("Test accuracy= " , str(accuracy*100))
```

```
----------------------------------------------------------------
Regularization parameter lambda =  0.01
Gamma parameter =  0.1
Test accuracy=  97.77777777777777
----------------------------------------------------------------
Regularization parameter lambda =  0.01
Gamma parameter =  1
Test accuracy=  97.77777777777777
----------------------------------------------------------------
Regularization parameter lambda =  0.01
Gamma parameter =  10.0
Test accuracy=  97.77777777777777
----------------------------------------------------------------
Regularization parameter lambda =  1
Gamma parameter =  0.1
Test accuracy=  98.05555555555556
----------------------------------------------------------------
Regularization parameter lambda =  1
Gamma parameter =  1
Test accuracy=  98.05555555555556
----------------------------------------------------------------
Regularization parameter lambda =  1
Gamma parameter =  10.0
Test accuracy=  98.05555555555556
----------------------------------------------------------------
Regularization parameter lambda =  100.0
Gamma parameter =  0.1
Test accuracy=  98.05555555555556
----------------------------------------------------------------
Regularization parameter lambda =  100.0
Gamma parameter =  1
Test accuracy=  98.05555555555556
----------------------------------------------------------------
Regularization parameter lambda =  100.0
Gamma parameter =  10.0
Test accuracy=  98.05555555555556
```

Conclusion

Results of rbf kernel SVC: 99.16%

```
Result accuracy of linear kernel SVC: 98.055%

Hyperparameter tuning accuracy of linear kernel SVC: 98.055%
```

✓ 0s completed at 4:46 PM ● ×