CS353 Machine Learning Lab

Image classification & SVM (12/03/21)

Shumbul Arifa (181CO152)

TASK

Write a program to demonstrate Image Recognition. Classify the data using svm and try to identify the images present in the data set.

Dataset

We are using digits dataset from sklearn.datasets

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html

We've implemented 3 types of kernels:

1. Linear Kernel
2. Polynomial Kernel
3. Radial Basis Function (RBF) kernel

Imports

In [1]:
```
import numpy as np
import pandas as pd
from sklearn import datasets, svm, metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score, r2_sc
ore
from sklearn.metrics import confusion_matrix, accuracy_score,classification_report
```

Loading dataset

In [2]:
```
digits = datasets.load_digits()
images = digits.images

print('Digits dataset keys \n{}'.format(digits.keys()))
```

```
Digits dataset keys
dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])
```

In [3]:
```
print('shape of datset: {} \nand target: {}'.format(digits.data.shape, digits.target.shape))
print('shape of the images: {}'.format(digits.images.shape))
```

```
shape of datset: (1797, 64)
and target: (1797,)
shape of the images: (1797, 8, 8)
```

Splitting data

In [4]:
```
n_samples = len(digits.images)
data_images = digits.images.reshape((n_samples, -1))

X_train, X_test, y_train, y_test = train_test_split(data_images,digits.target)
```

In [5]:
```
print('Training data and target sizes: \n{}, {}'.format(X_train.shape,y_train.shape))
print('Test data and target sizes: \n{}, {}'.format(X_test.shape,y_test.shape))
```

```
Training data and target sizes:
(1347, 64), (1347,)
Test data and target sizes:
(450, 64), (450,)
```

Implementing SVM

SVM: Support Vector Machine is a supervised classification algorithm where we draw a line between two different categories to differentiate between them.

Linear Kernel

In [6]:
```
clf = svm.SVC(kernel="linear")
clf.fit(X_train,y_train)
```

Out[6]:
```
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='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [7]:
```
y_pred = clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

```
0.9822222222222222
```

Polynomial Kernel

In [8]:
```
clf_poly = svm.SVC(kernel="poly", degree = 5)
clf_poly.fit(X_train,y_train)
```

Out[8]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=5, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [9]:
```
y_pred2 = clf_poly.predict(X_test)
print(accuracy_score(y_test, y_pred2))
```

```
0.9844444444444445
```

RBG Kernel

In [10]:
```
clf_rbf = svm.SVC(kernel="rbf", gamma=0.001)
clf_rbf.fit(X_train,y_train)
```

Out[10]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [11]:
```
y_pred3 = clf_rbf.predict(X_test)
print(accuracy_score(y_test, y_pred3))
```

```
0.9888888888888889
```

RESULTS

In [12]:
```
result = pd.DataFrame({'original' : y_test,'predicted' : y_pred})
result
```

Out[12]:

	original	predicted
0	3	3
1	4	4
2	0	0
3	0	0
4	8	8
...	...	...
445	0	0
446	2	2
447	7	7
448	6	6
449	5	9

450 rows × 2 columns

In [13]:
```
print("Accuracy =  {} %".format(accuracy_score(y_test, y_pred)*100))
```

```
Accuracy =  98.22222222222223 %
```

In [14]:
```
print(metrics.confusion_matrix(y_test, y_pred))
```

```
[[48  0  0  0  0  0  0  0  0  0]
 [ 0 48  0  0  0  0  0  0  0  0]
 [ 0  0 44  0  0  0  0  0  0  0]
 [ 0  0  0 46  0  1  0  0  0  0]
 [ 0  0  0  0 54  0  0  0  0  0]
 [ 0  0  0  0  0 43  0  0  0  2]
 [ 0  1  0  0  0  0 44  0  0  0]
 [ 0  0  0  0  0  0  0 44  0  1]
 [ 0  1  0  0  0  0  0  0 35  0]
 [ 0  0  0  0  0  0  0  1  1 36]]
```

In [15]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        48
           1       0.96      1.00      0.98        48
           2       1.00      1.00      1.00        44
           3       1.00      0.98      0.99        47
           4       1.00      1.00      1.00        54
           5       0.98      0.96      0.97        45
           6       1.00      0.98      0.99        45
           7       0.98      0.98      0.98        45
           8       0.97      0.97      0.97        36
           9       0.92      0.95      0.94        38

    accuracy                           0.98       450
   macro avg       0.98      0.98      0.98       450
weighted avg       0.98      0.98      0.98       450
```