Support Vector Machine

Δεύτερη Εργασία
Νευρωνικά δίκτυα

Γρήγορης Παντζαρτζής ΑΕΜ: 3785

Ο αλγόριθμος:

```
from sklearn.neighbors import NearestNeighbors
import numpy as np
import math
from sklearn import neighbors
from sklearn.neighbors import NearestCentroid
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import LinearSVC
from sklearn.decomposition import PCA
import time
from sklearn.svm import SVC

scaler = MinMaxScaler()

#load Data
TrainMnist = np.loadtxt('mnist_train.csv',delimiter=",", dtype=np.float32, skiprows=1)
TestMnist = np.loadtxt('mnist_test.csv',delimiter=",", dtype=np.float32, skiprows=1)


labels = TrainMnist[:,0:1]
TrainMnist = TrainMnist[:,1:]

labelst = TestMnist[:,0:1]
TestMnist = TestMnist[:,1:]

for i in range(60000):
    if (labels[i] % 2 == 0):
        labels[i] = 0  # Evens
    else:
        labels[i] = 1  # Odds

for i in range(10000):
    if (labelst[i] % 2 == 0):
        labelst[i] = 0  # Evens
    else:
        labelst[i] = 1  # Odds

y_train=np.ravel(labels)
x_train=np.array(TrainMnist)

y_test=np.ravel(labelst)
x_test=np.array(TestMnist)

#x_train = scaler.fit_transform(x_train)
#x_test = scaler.transform(x_test)

main_pca = PCA(n_components=90)
x_train = main_pca.fit_transform(x_train)

main_pca = PCA(n_components=90)
x_test = main_pca.fit_transform(x_test)

x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

-Κάνουμε import τις κατάλληλες Βιβλιοθήκες για την λύση της άσκησης

-Κάνουμε Load τα δεδομένα μας και έπειτα με μια for ξεχωρίζουμε τους μονούς και τους αρτίους αριθμούς

-Χρησιμοποιούμε Pca για να μειώσουμε τις διαστάσεις

-Στην συνέχεια με την βοήθεια της βιβλιοθήκης της sklearn χρησιμοποιούμε τις εντολές για την υλοποίηση SVM

```
#time1=time.time()

#svm = LinearSVC(loss='hinge', multi_class='ovr', C=1)
#svm.fit(x_train, y_train)
#print("Train, test acc:", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))

#time2=time.time()
#print('Time Passed: {:.3f}sec'.format(time2-time1))

#C=10
#time1=time.time()
#svm = LinearSVC(loss='hinge',C=C, multi_class='ovr', max_iter=1000000)
#svm.fit(x_train, y_train)
#print("C = ", C, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
#time2=time.time()
#print('Time Passed: {:.3f}sec'.format(time2-time1))


#C=10
#time1=time.time()
#svm = SVC(C=C, kernel='linear', decision_function_shape='ovr', max_iter=100000)
#svm.fit(x_train, y_train)
#print("C = ", C, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
#time2=time.time()
#print('Time Passed: {:.3f}sec'.format(time2-time1))


#C=10
#time1=time.time()
#svm = SVC(C=C, kernel='poly', degree=5, decision_function_shape='ovr', max_iter=100000)
#svm.fit(x_train, y_train)
#print("C = ", C, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
#time2=time.time()
#print('Time Passed: {:.3f}sec'.format(time2-time1))

gamma=10
time1=time.time()
svm = SVC(C=1, kernel='rbf', gamma=gamma, decision_function_shape='ovr', max_iter=100000)
svm.fit(x_train, y_train)
print("gamma = ", gamma, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
time2=time.time()
print('Time Passed: {:.3f}sec'.format(time2-time1))
```

Αποτελέσματα:

```
time1=time.time()
svm = LinearSVC(loss='hinge',C=C, multi_class='ovr', max_iter=1000000)
svm.fit(x_train, y_train)
print("C = ", C, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
time2=time.time()
print('Time Passed: {:.3f}sec'.format(time2-time1))
```

Αλλάζουμε την μεταβλητή C και βλέπουμε τα εξής αποτελέσματα:

	C=0.01	C=0.1	C=1	C=10
Train	87.64	88.50	88.39	88.51
Test	74.19	73.44	73.24	72.77
Time Pass	0.390	1.060	7.194	19.532

Συμπεράσματα: Το accuracy του train όσο ανεβαίνει το c ανεβαίνει ενώ του test μικραίνει. Όσο πιο μεγάλο είναι το c τόσο πιο πολύ ώρα τρέχει ο αλγόριθμος

```
time1=time.time()
svm = SVC(C=C, kernel='linear', decision_function_shape='ovr', max_iter=100000)
svm.fit(x_train, y_train)
print("C = ", C, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
time2=time.time()
print('Time Passed: {:.3f}sec'.format(time2-time1))
```

Βάζοντας στην μεταβλητή kernel =linear και αλλάζοντας την μεταβλητή C έχουμε τα εξής αποτελέσματα:

	C=0.01	C=0.1	C=1	C=10
Train	87.51	88.37	88.48	88.55
Test	74.15	73.49	72.63	73.1
Time Pass	304.595	210.648	196.180	554.207

Συμπεράσματα: Το accuracy του train όσο ανεβαίνει το c ανεβαίνει ενώ του test μικραίνει ο χρόνος στην αρχή μειώνεται όμως στην συνέχεια αυξάνεται για c=10

```
time1=time.time()
svm = SVC(C=C, kernel='poly', degree=5, decision_function_shape='ovr', max_iter=100000)
svm.fit(x_train, y_train)
print("C = ", C, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
time2=time.time()
print('Time Passed: {:.3f}sec'.format(time2-time1))
```

Βάζοντας στην μεταβλητή kernel =poly και αλλάζοντας την μεταβλητή C και το degree έχουμε τα εξής αποτελέσματα:

Degree=3	C=0.01	C=0.1	C=1	C=10
Train	98.85	99.58	99.97	99.99
Test	75.44	72.95	70.1	68.63
Time Pass	169.528	308.590	422.010	405.819

Συμπεράσματα: Το accuracy του train όσο ανεβαίνει το c ανεβαίνει ενώ του test μικραίνει ο χρόνος στην αρχή αυξάνεται ενώ για c=10 βλέπουμε να κατεβαίνει

Degree=5	C=0.01	C=0.1	C=1	C=10
Train	100	100	100	100
Test	69.62	69.19	68.45	69.23
Time Pass	240.577	245.037	249.539	230.157

Συμπεράσματα: Το accuracy του train έχει φτάσει στο 100% ενώ του test στην αρχή μικραίνει ενώ έπειτα μεγαλώνει .ο χρόνος στην αρχή αυξάνεται ενώ για c=10 βλέπουμε να κατεβαίνει

```
gamma=10
time1=time.time()
svm = SVC(C=1, kernel='rbf', gamma=gamma, decision_function_shape='ovr', max_iter=100000)
svm.fit(x_train, y_train)
print("gamma = ", gamma, "Train, test acc: ", 100*svm.score(x_train, y_train), 100*svm.score(x_test, y_test))
time2=time.time()
print('Time Passed: {:.3f}sec'.format(time2-time1))
```

Βάζοντας στην μεταβλητή kernel =rbf και αλλάζοντας την μεταβλητή gamma έχουμε τα εξής αποτελέσματα:

	gamma=0.01	gamma=0.1	gamma=1	gamma=10
Train	88.26	95.95	99.64	100
Test	73.89	75.24	74.26	49.72
Time Pass	392.490	265.854	243.993	2009.839

Συμπεράσματα: Το accuracy του train όσο πάει μεγαλώνει ενώ του test στην αρχή μεγαλώνει ενώ έπειτα μικραίνει και για c=10 έχουμε μεγάλη μείωση .ο χρόνος στην αρχή μειώνεται ενώ για c=10 βλέπουμε ότι το πρόγραμμα έχει μεγάλο χρόνο εκτέλεσης