q5

April 5, 2021

1 5. Fisherfaces- Face classification using LDA (40 classes)

Use the following "face.csv" file to classify the faces of 40 different people.

[18]:
```
!pip install scikit-plot
```

```
Collecting scikit-plot
  Downloading https://files.pythonhosted.org/packages/7c/47/32520e259340c140a4ad
27c1b97050dd3254fdc517b1d59974d47037510e/scikit_plot-0.3.7-py3-none-any.whl
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.7/dist-
packages (from scikit-plot) (1.4.1)
Requirement already satisfied: matplotlib>=1.4.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-plot) (3.2.2)
Requirement already satisfied: scikit-learn>=0.18 in
/usr/local/lib/python3.7/dist-packages (from scikit-plot) (0.22.2.post1)
Requirement already satisfied: joblib>=0.10 in /usr/local/lib/python3.7/dist-
packages (from scikit-plot) (1.0.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-
packages (from scipy>=0.9->scikit-plot) (1.19.5)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib>=1.4.0->scikit-plot)
(1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib>=1.4.0->scikit-plot)
(2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib>=1.4.0->scikit-plot)
(2.4.7)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib>=1.4.0->scikit-plot) (0.10.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-
packages (from python-dateutil>=2.1->matplotlib>=1.4.0->scikit-plot) (1.15.0)
Installing collected packages: scikit-plot
Successfully installed scikit-plot-0.3.7
```

[52]:
```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB
```

```
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import scikitplot as skplt
```

2 Function for LDA

[86]:
```
def LDA(X,labels):

        d = X.shape[1]
        classes=np.unique(labels)
        c=len(classes)
        d_=c-1
        class_dict={}
        for i in range(len(classes)):
                class_dict[classes[i]]=i

        class_wise_data=[np.empty((0,)+X[0].shape,float) for i in classes]
        for i in range(len(X)):
                class_wise_data[class_dict[labels[i]]]=np.
 ↪append(class_wise_data[class_dict[labels[i]]], np.array([X[i],]),axis=0)

        means=[]
        for i in class_wise_data:
                means.append(np.mean(i,axis=0))

        Sw = np.zeros((d,d))
        for i,data in enumerate(class_wise_data):
                z=data-means[i]
                Sw+=(z.T @ z)
        Sw_inv=np.linalg.inv(Sw)

        overall_mean = np.mean(X,axis=0)
        Sb = np.zeros((d,d))
        for i, data in enumerate(means):
                Ni=len(class_wise_data[i])
                z=np.array([means[i]-overall_mean])
                Sb+=(Ni * (z.T @ z))

        M = Sw_inv @ Sb
        eigen_values , eigen_vectors = np.linalg.eigh(M)
        eigen_values , eigen_vectors = eigen_values.astype(np.float64) ,␣
 ↪eigen_vectors.astype(np.float64)
        sorted_index = np.argsort(eigen_values)[::-1]
        sorted_eigenvectors = eigen_vectors[:,sorted_index]
        sorted_eigenvalue = eigen_values[sorted_index]
```

```
        eigenvector_subset = sorted_eigenvectors[:,0:d_]

        plt.bar(list(range(1,eigen_vectors.shape[0]+1)),sorted_eigenvalue)
        plt.ylabel("eigen values")

        Y=X @ eigenvector_subset
        return Y,eigenvector_subset
```

3 Read data and split it to test and train

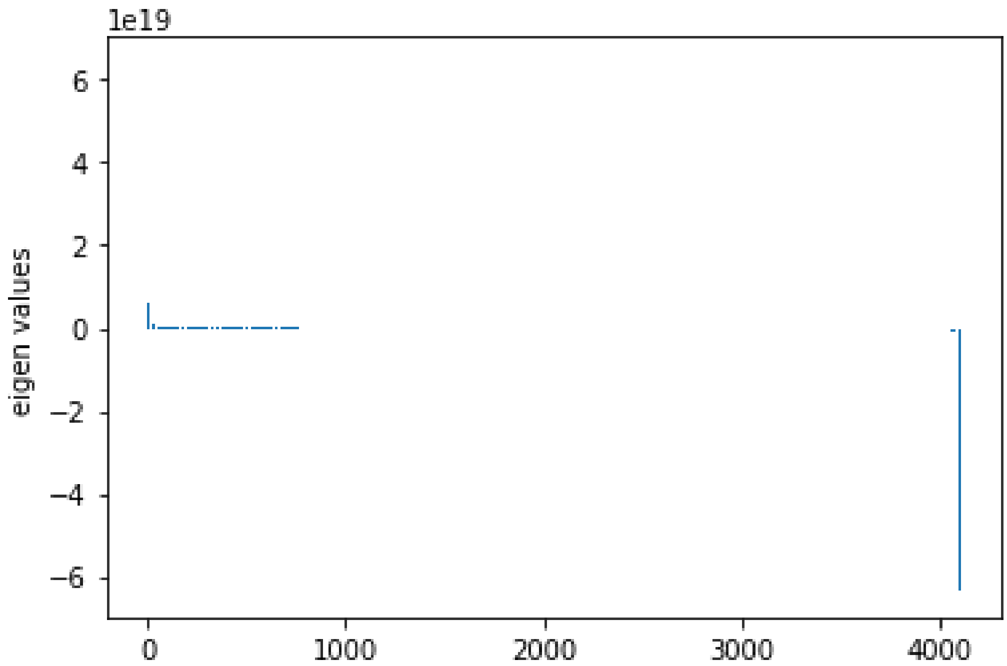
[87]:
```
data=pd.read_csv('face.csv')
```

[88]:
```
x = data.iloc[:,:-1]
target = data.iloc[:,-1]
```

[89]:
```
train_data = pd.concat([data.iloc[i*10+2:(i+1)*10] for i in range(40)])
test_data = pd.concat([data.iloc[i*10:i*10+2] for i in range(40)])
train_data.reset_index(drop=True, inplace=True)
test_data.reset_index(drop=True, inplace=True)
```

4 Use LDA function to reduce dim

[90]:
```
reduced,eigenvector_subset = LDA(np.array(train_data.iloc[:,:
 ↪-1]),list(train_data['target']))
```



[91]:
```python
reduced = pd.DataFrame(reduced)
```

5 Classify test data and find accuracy

[92]:
```python
model = GaussianNB()
model.fit(reduced,train_data["target"])
```

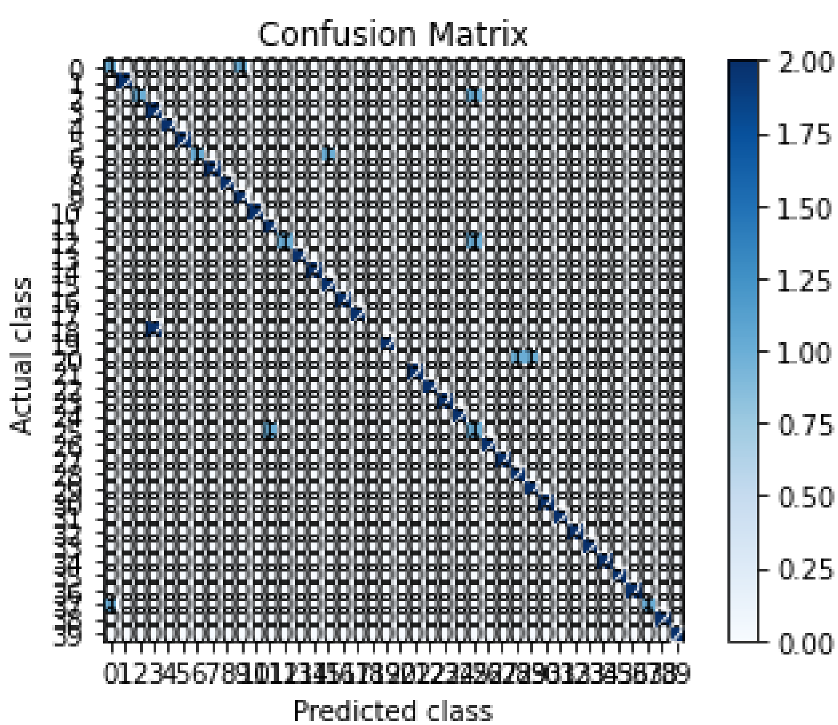
[92]: GaussianNB(priors=None, var_smoothing=1e-09)

[93]:
```python
test_reduced=(test_data.iloc[:,:-1]).dot(eigenvector_subset)
predicted= model.predict(test_reduced)
test_reduced['target']=test_data['target']
```

[94]:
```python
test_reduced['predicted'] = predicted
correctness=[]
for i in test_reduced.index:
        if test_reduced['target'][i] == test_reduced['predicted'][i]:
                correctness.append("correct")
        else:
                correctness.append("wrong")

test_reduced["correctness"]=correctness
print(test_reduced)
```

```python
x=accuracy_score(test_reduced["target"], predicted)
print(f"Accuracy ={x*100}%")

skplt.metrics.plot_confusion_matrix(test_reduced["target"], predicted)#, normalize=True)
plt.xlabel('Predicted class')
plt.ylabel('Actual class')
plt.show()
```

```
              0          1          2  …  target  predicted  correctness
0  -14.907826 -4.569430 -0.827318  …       0          0      correct
1  -12.912718 -2.557044  0.112835  …       0          9        wrong
2  -14.155274 -4.687639 -1.662370  …       1          1      correct
3  -13.903795 -4.280577 -1.522141  …       1          1      correct
4  -12.324845 -2.570033 -1.145355  …       2         25        wrong
..        …         …         …  …     …          …
75 -13.024525 -2.618749 -0.415230  …      37         37      correct
76  -9.124297 -3.698598 -0.485431  …      38         38      correct
77  -8.225609 -2.829540 -0.746084  …      38         38      correct
78 -11.922242 -1.713910 -1.664928  …      39         39      correct
79 -14.177076 -3.706217 -1.264375  …      39         39      correct

[80 rows x 42 columns]
Accuracy =87.5%
```



```
[ ]:
```