DMG2 Assignment : Problem 4

Bayesian Classifier

In [1]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn import preprocessing
from sklearn.neighbors import KernelDensity
from sklearn.mixture import BayesianGaussianMixture
```

In [2]:
```
DATA_DIR = '/home/jishnu/Documents/ISB/Term3/dmg2/assignments/hw_assignment
1/dmg2/datasets/mnist'
```

In [3]:
```
train = pd.DataFrame(columns=['V{}'.format(i) for i in range(1,785)] + ['lab
el'])
test = pd.DataFrame(columns=['V{}'.format(i) for i in range(1,785)] + ['labe
l'])
for num in range(10):
    # Consolidating training data
    temp_train =
pd.read_csv(os.path.join(DATA_DIR,'train{0}.csv'.format(num)),usecols=
['V{}'.format(i) for i in range(1,785)])
    temp_train['label'] = num
    train = train.append(temp_train,ignore_index=True)
    # Consolidating test data
    temp_test =
pd.read_csv(os.path.join(DATA_DIR,'test{0}.csv'.format(num)),usecols=
['V{}'.format(i) for i in range(1,785)])
    temp_test['label'] = num
    test = test.append(temp_test,ignore_index=True)
```

In [4]:
```
train.shape
```

Out[4]:
```
(36470, 785)
```

In [5]:
```
test.shape
```

Out[5]:
```
(24190, 785)
```

In [6]:
```
train[train.isnull().any(axis=1)].groupby(by='label')
['label'].value_counts()
```

Out[6]:
```
label  label
4      4         46
5      5        299
6      6          1
8      8         42
Name: label, dtype: int64
```

In [7]:
```
test[test.isnull().any(axis=1)].groupby(by='label')['label'].value_counts()
```

Out[7]:
```
label  label
4      4         35
5      5        203
6      6          4
8      8         30
Name: label, dtype: int64
```

There are missing values in both the training and test data. Shown above is the count of rows with missing values, and the associated labels.

In [8]:
```
train.groupby(by='label')['label'].value_counts()
```

Out[8]:
```
label  label
0      0        3567
1      1        4034
2      2        3582
3      3        3677
4      4        3567
5      5        3567
6      6        3567
7      7        3763
8      8        3567
9      9        3579
Name: label, dtype: int64
```

In [9]:
```
test.groupby(by='label')['label'].value_counts()
```

Out[9]:
```
label  label
0      0        2356
1      1        2708
2      2        2376
3      3        2454
4      4        2356
5      5        2356
6      6        2356
7      7        2502
8      8        2356
9      9        2370
Name: label, dtype: int64
```

Considering the number of complete data for each label, we can safely remove the rows with missing values for our analysis.

In [10]:
```
train = train.dropna()
test = test.dropna()
```

In [11]:
```
train.isnull().values.any()
```

Out[11]:
```
False
```

In [12]:
```
test.isnull().values.any()
```

Out[12]:
```
False
```

There are no missing values in the training and test data now

In [13]:
```
X_train = train.iloc[:,:784]
Y_train = train.iloc[:,784]

X_test = test.iloc[:,:784]
Y_test = test.iloc[:,784]
```

In [14]:
```
# Standardizing feature values
X_train = StandardScaler().fit_transform(X_train)
X_test = StandardScaler().fit_transform(X_test)
```

Applying PCA

In [15]:
```
# Applying PCA
pc = PCA(n_components=9).fit_transform(X_train)
d1_train = pd.DataFrame(data=pc,columns=['pc{0}'.format(i) for i in
range(1,10)])
d1_train['label'] = Y_train.values
d1_train.head(5)

pc = PCA(n_components=9).fit_transform(X_test)
d1_test = pd.DataFrame(data=pc,columns=['pc{0}'.format(i) for i in range(1,1
0)])
d1_test['label'] = Y_test.values
d1_test.head(5)
```

Out[15]:

	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8	p
0	1.751830	-6.389664	-2.021165	-2.694659	-6.429366	1.019104	-0.542573	5.256936	3.7678
1	5.884343	-7.690853	-2.390978	0.261106	-4.924829	-0.391050	0.309557	6.492449	3.0926
2	16.381096	5.663253	-1.865696	-3.367635	4.355100	-3.101171	-5.638945	-3.853234	2.5162
3	12.814321	-7.638624	-4.434649	-7.658103	-0.552335	-1.798447	-0.070458	1.757393	0.3404
4	11.123280	7.252469	5.007046	-0.332320	15.016163	-0.064696	-0.998728	-2.343500	0.0363

Applying Fisher LDA

In [16]:
```
fisher = LinearDiscriminantAnalysis(n_components=9).fit_transform(X_train,Y_
train.astype('int'))
d2_train = pd.DataFrame(data=fisher,columns=['f{0}'.format(i) for i in
range(1,10)])
d2_train['label'] = Y_train.values
d2_train.head(5)

fisher = LinearDiscriminantAnalysis(n_components=9).fit_transform(X_test,Y_t
est.astype('int'))
d2_test = pd.DataFrame(data=fisher,columns=['f{0}'.format(i) for i in
range(1,10)])
d2_test['label'] = Y_test.values
d2_test.head(5)
```

```
/home/jishnu/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_anal
ysis.py:388: UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
/home/jishnu/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_anal
ysis.py:442: UserWarning: The priors do not sum to 1. Renormalizing
  UserWarning)
/home/jishnu/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_anal
ysis.py:388: UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
```

Out[16]:

	f1	f2	f3	f4	f5	f6	f7	f8	
0	-2.833687	-1.079133	-0.845838	-0.764987	-0.484096	0.313129	-0.955681	-0.997196	1.0761
1	-3.849851	-3.435085	-1.935596	0.478846	-2.420500	-0.475621	0.572627	-0.989022	0.7790
2	-3.350865	-4.500847	-3.900589	-1.086583	-2.654538	-1.739049	-1.565408	-0.213139	-0.0240
3	-3.283216	-2.503597	-3.138834	-0.103059	-1.582149	-0.295842	1.330582	-3.239352	1.4733
4	-2.231976	-1.552025	-3.889057	-0.017441	-0.577388	-2.299944	-0.684894	0.945457	0.4041

Building Bayesian Classifier on D1 Dataset

In [17]:
```
d1_train_X = d1_train.iloc[:,:9]
d1_train_Y = d1_train.iloc[:,9].astype('int')

d1_test_X = d1_test.iloc[:,:9]
d1_test_Y = d1_test.iloc[:,9].astype('int')

d2_train_X = d2_train.iloc[:,:9]
d2_train_Y = d2_train.iloc[:,9].astype('int')

d2_test_X = d2_test.iloc[:,:9]
d2_test_Y = d2_test.iloc[:,9].astype('int')
```

In [18]:
```
bayes_clf_acc = pd.DataFrame(columns=['Dataset','Covariance','Test
Accuracy'])
```

In [19]:
```
bayesian_d1_diag =
BayesianGaussianMixture(n_components=10,covariance_type='diag',max_iter=1000
00)
bayesian_d1_diag = bayesian_d1_diag.fit(d1_train_X,d1_train_Y)
bayes_d1_diag_df = pd.DataFrame({'actual':d1_test_Y,'predicted':bayesian_d1_
diag.predict(d1_test_X)})
bayes_clf_acc = bayes_clf_acc.append({'Dataset':'D1','Covariance':'diag','Te
st Accuracy':np.round(bayes_d1_diag_df.loc[bayes_d1_diag_df['actual'] == bay
es_d1_diag_df['predicted']].shape[0]/bayes_d1_diag_df.shape[0],4)},ignore_in
dex=True)
```

In [20]:
```
bayesian_d1_full =
BayesianGaussianMixture(n_components=10,covariance_type='full',max_iter=1000
0)
bayesian_d1_full = bayesian_d1_full.fit(d1_train_X,d1_train_Y)
bayes_d1_full_df = pd.DataFrame({'actual':d1_test_Y,'predicted':bayesian_d1_
full.predict(d1_test_X)})
bayes_clf_acc = bayes_clf_acc.append({'Dataset':'D1','Covariance':'full','Te
st Accuracy':np.round(bayes_d1_full_df.loc[bayes_d1_full_df['actual'] == bay
es_d1_full_df['predicted']].shape[0]/bayes_d1_full_df.shape[0],4)},ignore_in
dex=True)
```

In [21]:
```
bayesian_d2_diag =
BayesianGaussianMixture(n_components=10,covariance_type='diag',max_iter=1000
00)
bayesian_d2_diag = bayesian_d2_diag.fit(d2_train_X,d2_train_Y)
bayes_d2_diag_df = pd.DataFrame({'actual':d2_test_Y,'predicted':bayesian_d2_
diag.predict(d2_test_X)})
bayes_clf_acc = bayes_clf_acc.append({'Dataset':'D2','Covariance':'diag','Te
st Accuracy':np.round(bayes_d2_diag_df.loc[bayes_d2_diag_df['actual'] == bay
es_d2_diag_df['predicted']].shape[0]/bayes_d2_diag_df.shape[0],4)},ignore_in
dex=True)
```

In [22]:
```
bayesian_d2_full =
BayesianGaussianMixture(n_components=10,covariance_type='full',max_iter=1000
00)
bayesian_d2_full = bayesian_d2_full.fit(d2_train_X,d2_train_Y)
bayes_d2_full_df = pd.DataFrame({'actual':d2_test_Y,'predicted':bayesian_d2_
full.predict(d2_test_X)})
bayes_clf_acc = bayes_clf_acc.append({'Dataset':'D2','Covariance':'full','Te
st Accuracy':np.round(bayes_d2_full_df.loc[bayes_d2_full_df['actual'] == bay
es_d2_full_df['predicted']].shape[0]/bayes_d2_full_df.shape[0],4)},ignore_in
dex=True)
```

In [23]:
```
bayes_clf_acc
```

Out[23]:

	Dataset	Covariance	Test Accuracy
0	D1	diag	0.1456
1	D1	full	0.0186
2	D2	diag	0.3316
3	D2	full	0.1964

The test accuracies for the four classifier show that, classifiers using Fisher projections dataset has higher accuracies when compared to the PCA projected dataset.

It is also seen that classifiers with diagonal covariance matrix have better accuracies when compared to those full covariance matrix, for this dataset.