HW2_cm3700

February 22, 2018

1 5241_HW2_cm3700

1. LDA on the original 256 dimensional space.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn
from matplotlib import colors
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
df3 = np.loadtxt("/Users/chi/Desktop/5241/hw/hw2/train_3.txt",delimiter=',')
df5 = np.loadtxt("/Users/chi/Desktop/5241/hw/hw2/train_5.txt",delimiter=',')
df8 = np.loadtxt("/Users/chi/Desktop/5241/hw/hw2/train_8.txt",delimiter=',')
dft = np.loadtxt("/Users/chi/Desktop/5241/hw/hw2/zip_test.txt",delimiter=' ')

#train data prep

xtrain = np.vstack([df3,df5,df8])
yarray = np.array([3,5,8])
ytrain = np.repeat(yarray,[df3.shape[0],df5.shape[0],df8.shape[0]],axis=0)

#test data prep
ytest = dft[:,0]
xtest = dft[:,1:]
xtest = xtest[(ytest == 3)|(ytest == 5)|(ytest == 8),:]
ytest = ytest[(ytest == 3)|(ytest == 5)|(ytest == 8)]
```

In [3]:
```
#train dataset error rate
lda = LinearDiscriminantAnalysis(solver = "svd")
lda.fit(xtrain,ytrain)
sum(lda.predict(xtrain)!=ytrain)/ytrain.size
```

Out[3]: 0.015945330296127564

```
In [4]: #test dataset error rate
        sum(lda.predict(xtest)!=ytest)/ytest.size

Out[4]: 0.087398373983739841
```

2. LDA on the leading 49 principle components of the features.

```
In [5]: #PCA n=49
        pca = PCA(n_components = 49,svd_solver = "full")
        xxtrain = scale(xtrain,with_std = False)
        xxtest = scale(xtest,with_std = False)
        pca.fit(xxtrain)

        #transform data
        train49 = pca.transform(xxtrain)
        test49 = pca.transform(xxtest)

        #explained variance ratio
        sum(pca.explained_variance_ratio_)

Out[5]: 0.87737140126149915

In [6]: #LDA on the leading 49 components for train data
        lda.fit(train49,ytrain)

        #Training accuracy
        sum(lda.predict(train49)!=ytrain)/ytrain.size

Out[6]: 0.043849658314350795

In [7]: #test dataset error rate

        sum(lda.predict(test49)!=ytest)/ytest.size

Out[7]: 0.091463414634146339
```

3. LDA when you filter the data as follows. Each non-overlapping 2 Œ 2 pixel block is replaced by its average.

```
In [8]: #Function to filter data
        def filterdigit(x):
            matrix = np.reshape(x.values,(16,16))
            i,j = (2,2)
            m,n = matrix.shape
            b = matrix.reshape(m//i,i,n//j,j).mean((1,3),keepdims = 1)
            average_m = np.repeat(np.repeat(b,(i),axis = (1)),(j),axis = 3).reshape(matrix.sha
            average_r = average_m.flatten()
            return(average_r)

In [9]: df_train = pd.DataFrame(xtrain)
        df_test = pd.DataFrame(xtest)
```

```
In [10]: filter_xtrain = df_train.apply(filterdigit,1).values
         filter_xtest = df_test.apply(filterdigit,1).values

In [11]: lda.fit(filter_xtrain,ytrain)
         #train set error rate
         sum(lda.predict(filter_xtrain)!=ytrain)/ytrain.size

/Users/chi/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWarn
  warnings.warn("Variables are collinear.")


Out[11]: 0.033599088838268794

In [12]: #testset error rate
         sum(lda.predict(filter_xtest)!=ytest)/ytest.size

Out[12]: 0.075203252032520332
```

Multiple linear logistic regression using the same filtered data as in the previous question

```
In [13]: Log = LogisticRegression(multi_class = "multinomial",solver = "newton-cg")

         Log.fit(filter_xtrain,ytrain)

Out[13]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='multinomial',
                   n_jobs=1, penalty='l2', random_state=None, solver='newton-cg',
                   tol=0.0001, verbose=0, warm_start=False)

In [14]: sum(Log.predict(filter_xtrain)!=ytrain)/ytrain.size

Out[14]: 0.015375854214123007

In [15]: sum(Log.predict(filter_xtest)!=ytest)/ytest.size

Out[15]: 0.07926829268292683

In [16]: ##                                          Trainning    Test
         ## LDA with 256 features                   0.01594533    0.08739837
         ## LDA with 49 components                  0.04384966    0.09146341
         ## LDA with average 2x2 pixel blocks      0.03359909    0.07520325
         ## Multiple Logistic Regression           0.01537584    0.079268292

In [17]: ## LDA with 256 features has an overfitting problem since it has the lowerst trainnin
         ## LDA with 49 components has the worst result which means perform PCA before LDA is
         ## LDA wirh averafe 2x2 has the best result.
```