Chair for Theoretical Information Technology
RWTH AACHEN UNIVERSITY

Prof. Dr. Rudolf Mathar, Dr. Arash Behboodi, Emilio Balda

Exercise 5

Friday, December 22, 2017

Problem 1. *Discriminant Analysis for MNIST dataset (PyTorch)*

In this script, we apply Fisher's linear discriminant analysis for classification of MNIST dataset. The dataset is first loaded.

```
In [1]: %matplotlib inline
        import torch
        from torchvision import datasets
        from matplotlib.pyplot import imshow
        import matplotlib.pyplot as plt
        import numpy as np
        ## Loading the data
        DNLD=True
        trainingMNIST = datasets.MNIST('./MNIST', train=True, download=DNLD)
        testMNIST = datasets.MNIST('./MNIST', train=False, download=DNLD)
```

After loading the dataset, training and test sets, we choose N entry for the training and Ntest for the evaluation part. We would like to evaluate the effect of training set size and test set size on the performance.

```
In [2]: from torchvision import transforms

        ## Training set
        N=100
        Xtraining=trainingMNIST.train_data[0:N].numpy().reshape(N,28*28)
        Ytraining=trainingMNIST.train_labels[0:N].numpy()

        ## Test set
        Ntest=100
        Xtest=testMNIST.test_data[0:Ntest].numpy().reshape(Ntest,28*28)
        Ytest=testMNIST.test_labels[0:Ntest].numpy()
```

Here only the binary classification problem is considered. Two classes are chosen accordingly with labels C_i and C_j.

```
In [3]: Ci=2
        Cj=5

        ## Choosing respective classes from the training set
```

```
        Ind12=np.array([ind for ind in range(N)
                        if ((Ytraining[ind]==Ci) or (Ytraining[ind]==Cj))])
        X12=Xtraining[Ind12]
        Y12=Ytraining[Ind12]
        print("The size of the trainging set with two classes is given by:"
                        , len(Ind12))
        N12=len(Ind12)
        ## Choosing respective classes from the test set
        Ind12=np.array([ind for ind in range(Ntest)
                        if ((Ytest[ind]==Ci) or (Ytest[ind]==Cj))])
        Xtest12=Xtest[Ind12]
        Ytest12=Ytest[Ind12]
        print("The size of the test set with two classes is given by:"
                        , len(Ind12))
        N12test=len(Ind12)
```

```
The size of the trainging set with two classes is given by: 11
The size of the test set with two classes is given by: 15
```

Now we fit the linear discriminant analysis (LDA) to this model. First, LDA is applied to the training set. The training error reperesents how well LDA can separate two classes. The performance of LDA is evaluated on the test set. The test error shows the generalization property of LDA.

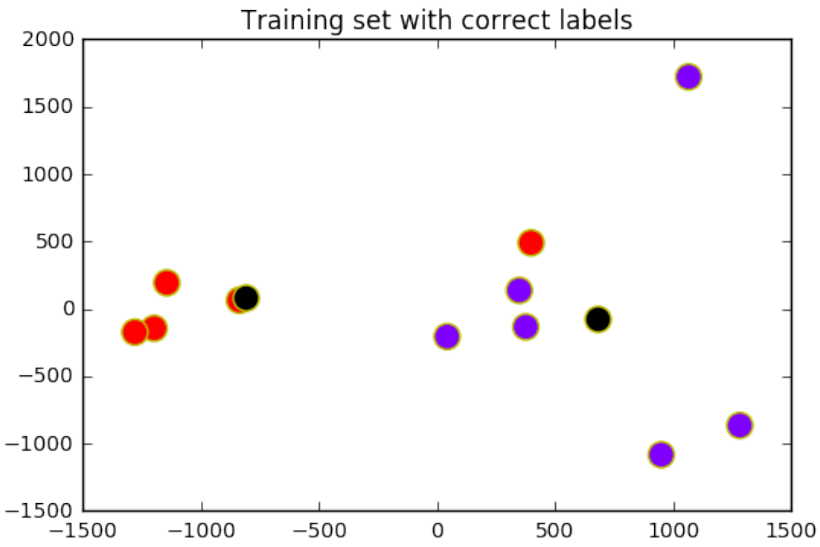
```
In [4]: from sklearn.discriminant_analysis import
                        LinearDiscriminantAnalysis as LDA
        model = LDA()
        model.fit(X12, Y12)
        TrainingError=np.count_nonzero(np.array(model.predict(X12))-Y12)/N12
        print("The misclassification error for the training set is given by:
                        ",TrainingError)
        TestError=np.count_nonzero(np.array(model.predict(Xtest12))-Ytest12)/N12test
        print("The misclassification error for the test set is given by:
                        ",TestError)
```

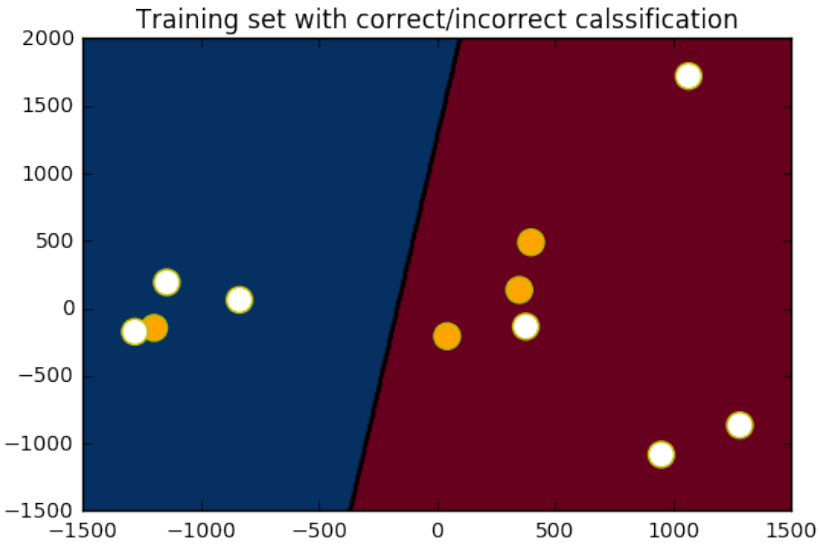
```
The misclassification error for the training set is given by: 0.36363636363636365
The misclassification error for the test set is given by: 0.3333333333333333
```

1 Visualizing LDA

Using PCA, we project the data in two dimensional space and then visualize the performance of LDA. Since the classifier is linear the output of PCA in two dimensional space is a line.

In [5]:
```
##############################################
#### Loading PCA
from sklearn.decomposition import PCA
## Fitting the model to the data
Xpca = PCA(n_components=2).fit_transform(X12)
pca=PCA(n_components=2)
Xpca=pca.fit_transform(X12)
##############################################
#### Plotting the output
import matplotlib.pyplot as plt
fig = plt.figure()
## Plotting the training data with the correct labels
plt.scatter(Xpca[:, 0], Xpca[:, 1],  s=150, c=Y12,
                cmap=plt.cm.rainbow, edgecolor='y')
## Plotting the mean values per each class
Xmean=pca.transform(model.means_)
plt.scatter(Xmean[:, 0], Xmean[:, 1],  s=150, c='black',
                cmap=plt.cm.rainbow, edgecolor='y')
plt.title("Training set with correct labels")
x_min, x_max = plt.xlim()
y_min, y_max = plt.ylim()
plt.show()
##############################################
#### Classification partition of the space
nx, ny = 400, 200
xx, yy = np.meshgrid(np.linspace(x_min, x_max, nx),
                np.linspace(y_min, y_max, ny))
Xmesh=pca.inverse_transform(np.c_[xx.ravel(), yy.ravel()])
Ymesh=model.predict(Xmesh).reshape(xx.shape)
plt.pcolormesh(xx, yy, Ymesh, cmap=plt.cm.RdBu)
plt.contour(xx, yy, Ymesh, [0.5*Ci+0.5*Cj], linewidths=2., colors='k')
##############################################
#### Error for the training set
ErrorColor=['white' if (i==0)  else 'orange'
                for i in np.absolute(model.predict(X12)-Y12)]
plt.scatter(Xpca[:, 0], Xpca[:, 1],  s=150, c=ErrorColor,
                cmap=plt.cm.rainbow, edgecolor='y')
plt.title("Training set with correct/incorrect calssification")
# plt.axis('tight')
plt.xlim(x_min,x_max)
plt.ylim(y_min,y_max)
plt.show()
```





1.1 Three-class classification

In this part, we repeat the experiments for three different classes. The linear discriminant analysis should naturally extend to multi-class cases.

```
In [6]: Ck=7
        ## Choosing respective classes from the training set
        Ind12=np.array([ind for ind in range(N) if ((Ytraining[ind]==Ci) or
                          (Ytraining[ind]==Cj) or (Ytraining[ind]==Ck))])
        X12=Xtraining[Ind12]
```

```
        Y12=Ytraining[Ind12]
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
        print("The size of the trainging set with three classes is given by:",
                          len(Ind12))
        N12=len(Ind12)
        ## Choosing respective classes from the test set
        Ind12=np.array([ind for ind in range(Ntest)if ((Ytest[ind]==Ci)
                          or (Ytest[ind]==Cj) or (Ytest[ind]==Ck))])
        Xtest12=Xtest[Ind12]
        Ytest12=Ytest[Ind12]
        print("The size of the test set with three classes is given by:",
                          len(Ind12))
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
        N12test=len(Ind12)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
The size of the trainging set with three classes is given by: 21
The size of the test set with three classes is given by: 30
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

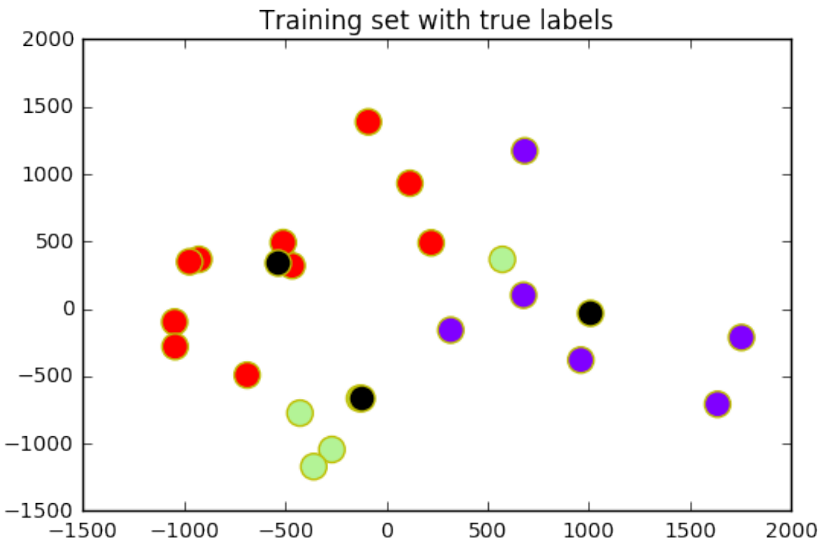
We load LDA model once again and fit it to the new data. The performance is measured afterward.

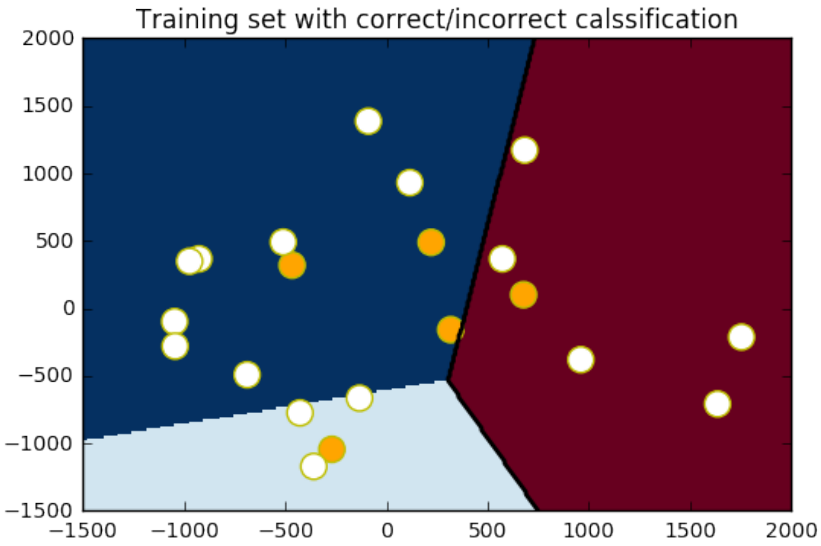
```
In [7]: modelII = LDA()
        modelII.fit(X12, Y12)
        TrainingError=np.count_nonzero(np.array(modelII.predict(X12))-Y12)/N12
        TestError=np.count_nonzero(np.array(modelII.predict(Xtest12))-Ytest12)/
                        N12test
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
        print("The misclassification error for the training set is given by:",
                        TrainingError)
        print("The misclassification error for the test set is given by:",
                        TestError)
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
        print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
The misclassification error for the training set is given by: 0.23809523809523808
The misclassification error for the test set is given by: 0.26666666666666666
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Finally the output is again represented in two-dimensional space using PCA.

```
In [8]: Xpca = PCA(n_components=2).fit_transform(X12)
        pca=PCA(n_components=2)
        Xpca=pca.fit_transform(X12)
        ######################################################################
        ######################################################################
        ### Plotting
        import matplotlib.pyplot as plt
        fig = plt.figure()
        ###############################
        # Plot 1
        ###############################
        plt.scatter(Xpca[:, 0], Xpca[:, 1],  s=150, c=Y12,
                            cmap=plt.cm.rainbow, edgecolor='y')
        plt.title("Training set with true labels")
        Xmean=pca.transform(modelII.means_)
        plt.scatter(Xmean[:, 0], Xmean[:, 1],  s=150, c='black',
                            cmap=plt.cm.rainbow, edgecolor='y')
        x_min, x_max = plt.xlim()
        y_min, y_max = plt.ylim()
        plt.show()
        ###############################
        # Plot 2
        ###############################
        ####################
        ## Creating the mesh
        nx, ny = 400, 200
        xx, yy = np.meshgrid(np.linspace(x_min, x_max, nx),
                            np.linspace(y_min, y_max, ny))
        Xmesh=pca.inverse_transform(np.c_[xx.ravel(), yy.ravel()])
        Ymesh=modelII.predict(Xmesh).reshape(xx.shape)
        plt.pcolormesh(xx, yy, Ymesh, cmap=plt.cm.RdBu)
        plt.contour(xx, yy, Ymesh, [0.5*Ci+0.5*Cj], linewidths=2., colors='k')
        ####################
        ErrorColor=['white' if (i==0)  else 'orange'
                            for i in np.absolute(modelII.predict(X12)-Y12)]
        plt.scatter(Xpca[:, 0], Xpca[:, 1],  s=150, c=ErrorColor,
                            cmap=plt.cm.rainbow, edgecolor='y')
        plt.title("Training set with correct/incorrect calssification")
        plt.xlim(x_min,x_max)
        plt.ylim(y_min,y_max)
        ######################################################################
        ######################################################################
        plt.show()
```





1.2 Multi-class classification

In this part, we just apply Fisher's linear discriminant analysis to multi-class classification. We choose the whole training set for training phase and the algorithm is evaluated on the whole MNIST test set. One can achieve 12.86% error on the training set and 12.7% error on the test set.

```
In [9]: ################################################################
        ## Training set
        N=60000
```

```
Xtraining=trainingMNIST.train_data[0:N].numpy().reshape(N,28*28)
Ytraining=trainingMNIST.train_labels[0:N].numpy()

## Test set
Ntest=10000
Xtest=testMNIST.test_data[0:Ntest].numpy().reshape(Ntest,28*28)
Ytest=testMNIST.test_labels[0:Ntest].numpy()
#################################################################
#################################################################
modelIII = LDA()
modelIII.fit(Xtraining, Ytraining)
TrainingError=
    np.count_nonzero(np.array(modelIII.predict(Xtraining))-Ytraining)/N
TestError=np.count_nonzero(np.array(modelIII.predict(Xtest))-Ytest)/Ntest
print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
print("The misclassification error for the training set is given by:",
                    TrainingError)
print("The misclassification error for the test set is given by:",
                    TestError)
print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
print("%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%")
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
The misclassification error for the training set is given by: 0.12851666666666667
The misclassification error for the test set is given by: 0.127
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```