Lab5

August 8, 2020

[ ]:
```
feature_dict = {i:label for i,label in zip(
                range(4),
                  ('sepal length in cm',
                  'sepal width in cm',
                  'petal length in cm',
                  'petal width in cm', ))}
```

[26]:
```
from google.colab import drive
drive.mount("/content/drive")
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

[30]:
```
import pandas as pd
import numpy as np

df = pd.read_csv ('/content/drive/My Drive/ml_data/data_banknote_authentication.
 ↪csv')
df.dropna(how="all", inplace=True) # to drop the empty line at file-end

df.tail()

#df
```

[30]:

	variance	skewness	kurtosis	entropy	class
1367	0.40614	1.34920	-1.4501	-0.55949	1
1368	-1.38870	-4.87730	6.4774	0.34179	1
1369	-3.75030	-13.45860	17.5932	-2.77710	1
1370	-3.56370	-8.38270	12.3930	-1.28230	1
1371	-2.54190	-0.65804	2.6842	1.19520	1

[35]:
```
%matplotlib inline

from sklearn.preprocessing import LabelEncoder

X = df.iloc[:,[0,1,2,3]].values
y = df['class'].values

enc = LabelEncoder()
label_encoder = enc.fit(y)
y = label_encoder.transform(y) + 1

label_dict = {1: 'Not Accepted', 2: 'Accepted'}
```

[36]:
```
print(X)
print(y)
```

```
[[  3.6216   8.6661  -2.8073  -0.447 ]
 [  4.5459   8.1674  -2.4586  -1.4621]
 [  3.866   -2.6383   1.9242   0.1065]
 ...
 [ -3.7503 -13.4586  17.5932  -2.7771]
 [ -3.5637  -8.3827  12.393   -1.2823]
 [ -2.5419  -0.658    2.6842   1.1952]]
[1 1 1 ... 2 2 2]
```

[37]:
```
np.set_printoptions(precision=4)

mean_vectors = []
for cl in range(1,3):
    mean_vectors.append(np.mean(X[y==cl], axis=0))
    print('Mean Vector class %s: %s\n' %(cl, mean_vectors[cl-1]))
```

```
Mean Vector class 1: [ 2.2767  4.2566  0.7967 -1.1476]

Mean Vector class 2: [-1.8684 -0.9936  2.1483 -1.2466]
```

[38]:
```
S_W = np.zeros((4,4))
for cl,mv in zip(range(1,3), mean_vectors):
    class_sc_mat = np.zeros((4,4))                  # scatter matrix for every␣
 ↪class
    for row in X[y == cl]:
        row, mv = row.reshape(4,1), mv.reshape(4,1) # make column vectors
        class_sc_mat += (row-mv).dot((row-mv).T)
    S_W += class_sc_mat                             # sum class scatter␣
 ↪matrices
print('within-class Scatter Matrix:\n', S_W)
```

```
within-class Scatter Matrix:
 [[  5258.339   -1333.6414  -4499.5048   2127.6972]
 [ -1333.6414  37886.4429 -24885.8881  -9073.929 ]
 [ -4499.5048 -24885.8881  24849.3223   4003.7394]
 [  2127.6972  -9073.929    4003.7394   6048.6247]]
```

[39]:
```
overall_mean = np.mean(X, axis=0)

S_B = np.zeros((4,4))
for i,mean_vec in enumerate(mean_vectors):
    n = X[y==i+1,:].shape[0]
    mean_vec = mean_vec.reshape(4,1) # make column vector
    overall_mean = overall_mean.reshape(4,1) # make column vector
    S_B += n * (mean_vec - overall_mean).dot((mean_vec - overall_mean).T)

print('between-class Scatter Matrix:\n', S_B)
```

```
between-class Scatter Matrix:
 [[ 5.8211e+03  7.3730e+03 -1.8980e+03  1.3903e+02]
 [ 7.3730e+03  9.3386e+03 -2.4040e+03  1.7609e+02]
 [-1.8980e+03 -2.4040e+03  6.1887e+02 -4.5332e+01]
 [ 1.3903e+02  1.7609e+02 -4.5332e+01  3.3205e+00]]
```

[40]:
```
eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))

for i in range(len(eig_vals)):
    eigvec_sc = eig_vecs[:,i].reshape(4,1)
    print('\nEigenvector {}: \n{}'.format(i+1, eigvec_sc.real))
    print('Eigenvalue {:}: {:.2e}'.format(i+1, eig_vals[i].real))
```

```
Eigenvector 1:
[[0.7434]
 [0.4082]
 [0.5298]
 [0.0042]]
Eigenvalue 1: 6.40e+00

Eigenvector 2:
[[-0.4578]
 [ 0.5426]
 [ 0.7042]
 [ 0.0055]]
Eigenvalue 2: -4.44e-16

Eigenvector 3:
[[-0.0937]
```

```
 [ 0.3162]
 [ 0.9434]
 [ 0.0357]]
Eigenvalue 3: 6.10e-16

Eigenvector 4:
[[ 0.4423]
 [-0.5292]
 [-0.6812]
 [ 0.2457]]
Eigenvalue 4: 1.63e-19
```

[41]:
```
# Make a list of (eigenvalue, eigenvector) tuples
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)

# Visually confirm that the list is correctly sorted by decreasing eigenvalues

print('Eigenvalues in decreasing order:\n')
for i in eig_pairs:
    print(i[0])
```

```
Eigenvalues in decreasing order:

6.3993202815265695
6.10094740551881e-16
4.440892098500626e-16
1.6286122920302746e-19
```

[42]:
```
print('Variance explained:\n')
eigv_sum = sum(eig_vals)
for i,j in enumerate(eig_pairs):
    print('eigenvalue {0:}: {1:.2%}'.format(i+1, (j[0]/eigv_sum).real))
```

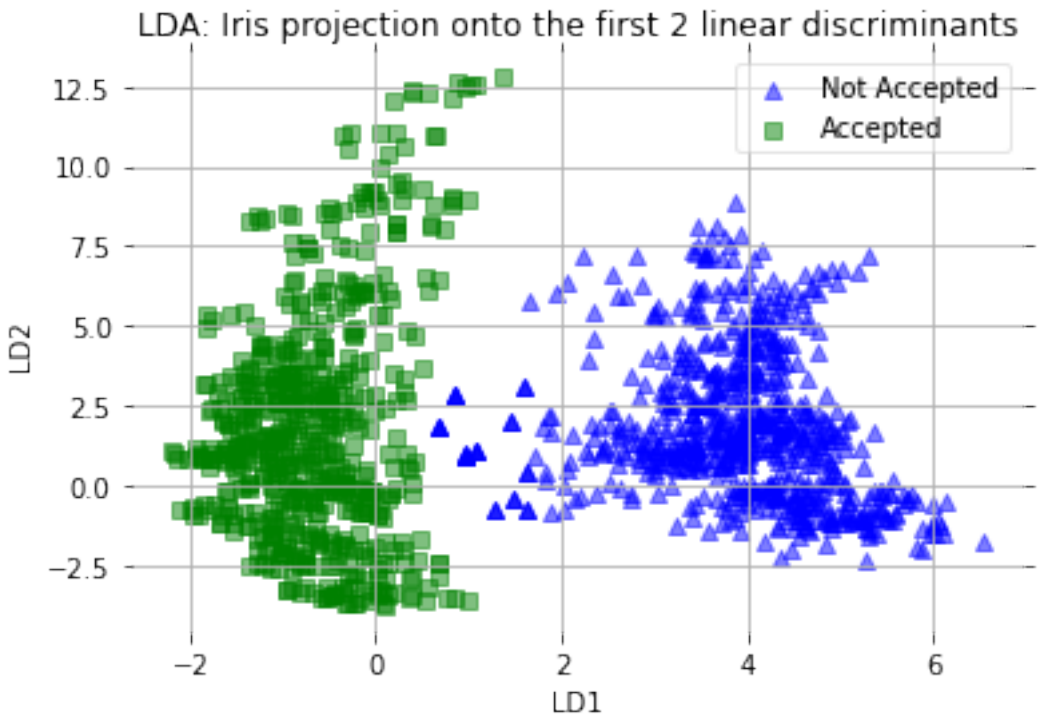
```
Variance explained:

eigenvalue 1: 100.00%
eigenvalue 2: 0.00%
eigenvalue 3: 0.00%
eigenvalue 4: 0.00%
```

[43]:
```
W = np.hstack((eig_pairs[0][1].reshape(4,1), eig_pairs[1][1].reshape(4,1)))
print('Matrix W:\n', W.real)
```

```
Matrix W:
 [[ 0.7434 -0.0937]
 [ 0.4082  0.3162]
 [ 0.5298  0.9434]
 [ 0.0042  0.0357]]
```

[44]:
```
X_lda = X.dot(W)
assert X_lda.shape == (df.shape[0],2), "The matrix is not 1372x2 dimensional."
```

[45]:
```
from matplotlib import pyplot as plt

def plot_step_lda():

    ax = plt.subplot(111)
    for label,marker,color in zip(
        range(1,3),('^', 's'),('blue', 'green')):

        plt.scatter(x=X_lda[:,0].real[y == label],
                y=X_lda[:,1].real[y == label],
                marker=marker,
                color=color,
                alpha=0.5,
                label=label_dict[label]
                )

    plt.xlabel('LD1')
    plt.ylabel('LD2')

    leg = plt.legend(loc='upper right', fancybox=True)
    leg.get_frame().set_alpha(0.5)
    plt.title('LDA: Iris projection onto the first 2 linear discriminants')

    # hide axis ticks
    plt.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

    plt.grid()
    plt.tight_layout
    plt.show()

plot_step_lda()
```



1 USING THE LIBRARY FOR LDA

[ ]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from matplotlib import pyplot as plt



feature_dict = {i:label for i,label in zip(
                range(4),
                  ('sepal length in cm',
                  'sepal width in cm',
                  'petal length in cm',
                  'petal width in cm', ))}



# Reading in the dataset

df = pd.io.parsers.read_csv(
```

```
    filepath_or_buffer='https://archive.ics.uci.edu/ml/
 ↪machine-learning-databases/iris/iris.data',
    header=None,
    sep=',',
    )
df.columns = [l for i,l in sorted(feature_dict.items())] + ['class label']
df.dropna(how="all", inplace=True) # to drop the empty line at file-end


# use the LabelEncode from the scikit-learn library to convert the class labels␣
 ↪into numbers: 1, 2, and 3

X = df.iloc[:,[0,1,2,3]].values
y = df['class label'].values

enc = LabelEncoder()
label_encoder = enc.fit(y)
y = label_encoder.transform(y) + 1

label_dict = {1: 'Setosa', 2: 'Versicolor', 3:'Virginica'}


# LDA
sklearn_lda = LDA(n_components=2)
X_lda_sklearn = sklearn_lda.fit_transform(X, y)

def plot_scikit_lda(X, title):

    ax = plt.subplot(111)
    for label,marker,color in zip(
        range(1,4),('^', 's', 'o'),('blue', 'red', 'green')):

        plt.scatter(x=X[:,0][y == label],
                    y=X[:,1][y == label] * -1, # flip the figure
                    marker=marker,
                    color=color,
                    alpha=0.5,
                    label=label_dict[label])

    plt.xlabel('LD1')
    plt.ylabel('LD2')

    leg = plt.legend(loc='upper right', fancybox=True)
    leg.get_frame().set_alpha(0.5)
    plt.title(title)

    # hide axis ticks
```

```
    plt.tick_params(axis="both", which="both", bottom="off", top="off",
            labelbottom="on", left="off", right="off", labelleft="on")

    # remove axis spines
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
    ax.spines["bottom"].set_visible(False)
    ax.spines["left"].set_visible(False)

    plt.grid()
    plt.tight_layout
    plt.show()

plot_scikit_lda(X_lda_sklearn, title='Default LDA via scikit-learn')
```