MACHINE LEARNING: BÀI THỰC HÀNH SỐ 4 – Phần 2

PHÂN TÍCH PHÂN BIỆT TUYẾN TÍNH (LINEAR DISCRIMINANT ANALYSIS - LDA)

Trong phần này chúng ta xây dựng chương trình cho phương pháp LDA và ứng dụng trong việc giảm số chiều dữ liệu. Dữ liệu sau khi được giảm số chiều sẽ được sử dụng cho một số mô hình phân loại đã học (ví dụ Logistic Regression, SoftMax).

Tương tự PCA, LDA cần khá nhiều tính toán với ma trận, nên chúng ta sẽ sử dụng thư viện Sci-Kit Learn (sklearn) và/hoặc numpy để tận dụng việc tính toán kiểu vector hóa.

A. BINARY LABELED DATA

Ví dụ A.1. Trong ví dụ này chúng ta tạo 02 tập dữ liệu ngẫu nhiên 02 chiều với số điểm lần lượt là N1 và N2, phân bố chuẩn với hiệp phương sai và kỳ vọng khác nhau, ứng với 02 nhãn (label) là 1 và 2. Đoạn mã lệnh tương tự với các bài thực hành trước:

```
np.random.seed(12)

means = [[0, 3], [2, 0]]
cov1 = [[1, 0.3], [0.3, 1]]
cov2 = [[1, 0.2], [0.2, 1.5]]
N1 = 50
N2 = 40
N = N1 + N2
X1 = np.random.multivariate_normal(means[0], cov1, N1) # each row is a data point
X2 = np.random.multivariate_normal(means[1], cov2, N2)

# Combine classes and create labels
X = np.vstack((X1, X2))
y = np.hstack((np. ones(50), 2*np.ones(40)))
```

Đoạn mã dưới đây hiển thị trực quan dữ liệu cũng như đánh dấu tâm điểm (vector trung bình) của mỗi tập dữ liệu ứng với mỗi nhãn:

```
# Separate data by class to class1 (y=0) and class2 (y=1)
class1 = X[y == 1]
class2 = X[y == 2]

print("Number of samples in Class 1:", class1.shape[0])
print("Number of samples in Class 2:", class2.shape[0])

# Calculate class means
mean1 = np.mean(class1, axis=0)
mean2 = np.mean(class2, axis=0)

print("Mean of Class 1:", mean1)
print("Mean of Class 2:", mean2)

# Visualize class means
plt.scatter(class1[:, 0], class1[:, 1], label='Class 1')
plt.scatter(class2[:, 0], class2[:, 1], label='Class 2')
plt.scatter(mean1[0], mean1[1], color='red', s=200, marker='*', label='Mean Class 1')
plt.scatter(mean2[0], mean2[1], color='green', s=200, marker='*', label='Mean Class 2')
plt.legend()
plt.title('Class Means')
plt.show()
```

Tiếp theo ta tính Between-Class Variance Matrix $\mathbf{S_B = (m0 - m1)(m0 - m1)^T}$ và Within-Class Variance Matrix $\mathbf{S_W = \Sigma_{xn \in X1} (x_n - m1)(x_n - m1)^T + \Sigma_{xn \in X2} (x_n - m2)(x_n - m2)^T}$

```
# Build S_B
m1 = np.mean(X1.T, axis = 1, keepdims = True)
m2 = np.mean(X2.T, axis = 1, keepdims = True)

a = (m2 - m1)
S_B = a.dot(a.T)

# Build S_W
SW1 = X1.T - np.tile(m1, (1, N1))
SW2 = X2.T - np.tile(m2, (1, N2))

S_W = SW1.dot(SW1.T) + SW2.dot(SW2.T)
print('Between-class covariance matrix: S_B =\n', S_B)
print('Within-class covariance matrix: S_W =\n',S_W)
```

Tính giá trị riêng, vector riêng tương ứng của $(S_W)^{-1}(S_B)$. Chú ý các giá trị riêng tìm được theo thuật toán của eig sẽ sắp xếp theo thứ tự giảm dần, do đó trị riêng lớn nhất và vector riêng tương ứng của nó có chỉ số [0].

```
L, W = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))

w = W[:, 0]

print(w)
print(w.shape)
```

Cuối cùng, chúng ta hiển thị trực quan dữ liệu và hình chiếu của chúng theo phương W vừa tìm được

```
# Visualize data
plt.plot(X1[:, 0], X1[:, 1], 'ro')
plt.plot(X2[:, 0], X2[:, 1], 'b^')
plt.axis([-5, 10, -5, 10])

#Compute projections of data points on to LDA's dimention
#
norm_W = w.T.dot(w)
X1_tmp = X1.dot(w)/norm_W
X2_tmp = X2.dot(w)/norm_W

print(X1_tmp.shape, w.shape)

X1_p = np.multiply(w.reshape(2,1), X1_tmp.T)
X2_p = np.multiply(w.reshape(2,1), X2_tmp.T)

print(X1_p.shape,X2_p.shape)

#Draw projected points
plt.plot(X1_p[0, :], X1_p[1, :], 'mo')
plt.plot(X2_p[0, :], X2_p[1, :], 'c^')

plt.show()
```

Sử dụng thư viện SKLEARN

Để sử dụng, chúng ta cần import các thư viện chưa công cụ LDA như đoạn lệnh dưới đây:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

Sau đó sử dụng các phương thức dưới đây để tìm hình chiếu lên không gian mới (tương tự PCA). Ở đây ta giả sử dữ liệu là **X** và nhãn của các mẫu là **y**

```
# Initialize LDA and fit the model
lda = LinearDiscriminantAnalysis(n_components=<số chiều giữ lại>)
```

```
X_lda = lda.fit_transform(X, y)
```

Ví dụ A.2 (Bài tập thực hành 1). Chúng ta sử dụng phương pháp LDA để giảm số chiều của tập dữ liệu bệnh nhân Parkinson, với 754 trường (thuộc tính) và 756 records. Trường 'class' chứa hai giá trị 0 và 1 xác định bệnh nhân có mắc Parkinson hay không (1 ~ có mắc) là đầu ra, và trường chỉ số mẫu, cần lấy khỏi phần dữ liệu X. Dữ liệu có trong tệp đính kèm (dạng CSV) hoặc tại link: https://www.kaggle.com/datasets/dipayanbiswas/parkinsons-disease-speech-signal-features .

Đoạn chương trình đọc dữ liệu có thể tham khảo từ bài thực hành phần PCA. Hãy giảm số chiều dữ liệu về 02 chiều bằng PCA và hiển thị để quan sát.

Yêu cầu tự thực hiện: Hãy sử dụng các bước và công cụ phù hợp để thực hiện được các công việc sau:

1) Giảm số chiều xuống còn 01 chiều bằng phương pháp LDA và 02 chiều bằng PCA nhằm phục vụ cho việc phân loại dữ liệu (theo class). Chia dữ liệu Train gồm 500 bản ghi và dữ liệu Test là phần còn lại. Sử dụng mô hình phân loại phù hợp để phân loại dữ liệu theo class. Dùng các công cụ đo độ chính xác để kiểm tra đánh giá mô hình ứng với một dữ liệu đã giảm chiều (theo LDA và PCA).
2) Sử dụng dữ liệu ban đầu, chia thành các tập Train- Test với tỷ lệ 3:1, sau đó áp dụng phương pháp Naïve Bayes phù hợp và phương pháp Hồi quy Logistic để thực hiện bài toán phân loại (theo class). Tiếp theo lại thực nghiệm các mô hình nói trên với dữ liệu đã giảm chiều ở ý 1), cùng tỷ lệ chia như trên. Hãy đánh giá và giải thích:
    a. 2 mô hình này, mô hình nào có độ chính xác thay đổi nhiều hơn?
    b. Thay đổi diễn ra với loại dữ liệu nào nhiều hơn. Tại sao?

B. LDA FOR MULTINOMIAL DATA

Tiếp theo chúng ta xây dựng chương trình cho phương pháp LDA với dữ liệu có nhiều nhãn – tức là C > 2. Ví dụ dưới đây sẽ sử dụng dữ liệu hoa Iris (có 03 nhãn đầu ra).

Ví dụ B.1. Trong ví dụ này, chúng ta xây dựng chương trình cho phương pháp LDA giảm số chiều của bộ dữ liệu hoa IRIS về 02 chiều. Dữ liệu gồm có 150 mẫu hoa phân đều trong 03 loại (class), mỗi mẫu có 04 chiều đã cho trong các bài trước. Dưới đây ta chỉ tập trung vào thực hiện phương pháp LDA xây dựng từ thư viện NUMPY.

Code minh họa trong Python:

Đoạn chương trình đọc tệp dữ liệu, truy xuất một số thông tin thống kê trên dữ liệu. Khai báo thư viện và đường dẫn đến tệp dữ liệu. Cần sửa đường dẫn này đến vị trí đặt dữ liệu cụ thể.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Load Iris dataset
iris = load_iris()
X_iris = iris.data
y_iris = iris.target

label_dict = {0:'Setosa', 1:'Versicolor', 2:'Virginica'}
```

Tính toán d-dimensional mean vectors của các class dữ liệu

```
np.set_printoptions(precision = 4)

mean_vectors = []
for cl in range(0,3):
    mean_vectors.append(np.mean(X[Y==cl], axis=0))
```

```
    print('Mean vector of class', cl+1, mean_vectors[cl],'.T')
```

Tính toán các ma trận within-class scatter matrix và $S_W = \sum_{i=1}^{c} S_i$ với $S_i = \sum_{\boldsymbol{x} \in D_i}^{n} (\boldsymbol{x} - \boldsymbol{m}_i)\,(\boldsymbol{x} - \boldsymbol{m}_i)^T$

```
S_W = np.zeros((4,4))
for cl, mv in zip(range(0, 3), mean_vectors):
    class_sc_mat = np.zeros((4,4))
    for row in X[Y == cl]:
        row = row.reshape(4, 1)
        mv  = mv.reshape(4, 1)
        class_sc_mat += (row - mv).dot((row -mv).T)
    S_W += class_sc_mat

print(S_W)
```

và between-class scatter matrix $S_B = \sum_{i=1}^{c} N_i (\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$

```
#the calculate the overall mean with a simple function
overall_mean = np.mean(X, axis = 0)

#initialize zeros mnatrix
S_B = np.zeros((4,4))
#For every iteration in the mean vectors
for cl,mean_vec in enumerate(mean_vectors):
    #Extracting the number of samples per class, in this case, 50
    n = X[Y==cl,:].shape[0]
    #Reshaping matrices from 1x4 to 4x1 for better manipulation
    mean_vec = mean_vec.reshape(4,1)
    overall_mean = overall_mean.reshape(4,1)
    #Applying the equation
    S_B += n * (mean_vec - overall_mean) .dot((mean_vec - overall_mean).T)

print(S_B)
```

Tính ma trận $S_W^{-1} S_B$ và hệ riêng của nó:

```
eigenvalues, eigenvectors = np.linalg.eig(np.linalg.inv(S_W).dot(S_B))

for i in range(len(eigenvalues)):
    eigenvectors_sc = eigenvectors[:,i].reshape(4,1)
    print("\nEigenvector {}: \n{}".format(i+1, eigenvectors_sc.real))
    print("Eigenvalue {:}: {:.2e}".format(i+1, eigenvalues[i].real))
```

Đoạn chương trình dưới đây hiển thị giá trị riêng theo thứ tự giảm dần và phương sai giải thích tương ứng

```
#List of every eigenvector and its corresponding eigenvalue
eigen_pairs = [(np.abs(eigenvalues[i]),eigenvectors[:,i]) for i in range(len(eigenvalues))]

#Sorting in descending order
eigen_pairs = sorted(eigen_pairs, key=lambda k: k[0], reverse = True)

print("Eigenvalues in decreasing order: \n")
for i in eigen_pairs:
    print (i[0])

print("Variance Explained: \n")
eigenvalues_sum = sum(eigenvalues)
for i,j in enumerate(eigen_pairs):
    print("Eigenvalue {0:}: {1: .2%}".format(i+1, (j[0]/eigenvalues_sum).real))
```

In ra ma trận W (ma trận chiếu)

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

Chiếu dữ liệu trong không gian mới thành lập và hiển thị:

```
X_lda= X.dot (W)
#will throw an error in case the dot product has made an error assert
X_lda.shape == (150,2), "The matrix is not 150x2 dimensional"

def plot_step_lda():
    ax = plt.subplot(111)
    for label, marker, color in zip(range(0,3), ("^", "s", "o"), ("blue", "red", "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 left", fancybox = True)
    leg.get_frame().set_alpha (0.5)
    plt.title("LDA: Iris Projection onto the first 2 linear discriminants")
    plt.tick_params (axis="both", which = "both", bottom = "off", top = "off",
                     labelbottom = "on", left = "off", right= "off", labelleft = "on")
    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()
```

Chúng ta có thể sử dụng thư viện sklearn để thực hiện yêu cầu trên. Toàn bộ đoạn lệnh như sau:

```
import matplotlib.pyplot as plt

from sklearn import datasets
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names

pca = PCA(n_components=2)
X_r = pca.fit(X).transform(X)

lda = LinearDiscriminantAnalysis(n_components=2)
X_r2 = lda.fit(X, y).transform(X)

# Percentage of variance explained for each components
print(
    "explained variance ratio (first two components): %s"
    % str(pca.explained_variance_ratio_)
```

```
)

plt.figure()
colors = ["navy", "turquoise", "darkorange"]
lw = 2

for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(
        X_r[y == i, 0], X_r[y == i, 1], color=color, alpha=0.8, lw=lw, label=target_name
    )
plt.legend(loc="best", shadow=False, scatterpoints=1)
plt.title("PCA of IRIS dataset")

plt.figure()
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(
        X_r2[y == i, 0], X_r2[y == i, 1], alpha=0.8, color=color, label=target_name
    )
plt.legend(loc="best", shadow=False, scatterpoints=1)
plt.title("LDA of IRIS dataset")

plt.show()
```

Yêu cầu thực hành

1) Sử dụng đoạn code chọn số chiều chính trong ví dụ 2, đưa tập dữ liệu đã đọc về còn 2 chiều, sau đó hiển thị lên màn hình để xem quan hệ giữa các lớp dữ liệu.

2) Với đoạn chương trình đọc dữ liệu đã có, hãy chạy lại ví dụ này với các thư viện của gói linear_model, lớp LogisticRegression và so sánh kết quả, chia Train:Test = 4:1 (theo từng loại hoa để tránh phân bố các loại hoa trong tập train và tập test mất cân bằng), tương ứng với 02 trường hợp:

    a. Chạy với dữ liệu nguyên bản, lưu lại độ chính xác, ma trận nhầm lẫn trong trường hợp này;

    b. Chạy với dữ liệu giảm còn 02 chiều, hãy xử lý theo 02 quy trình dưới đây:

        i. Sử dụng PCA. Lưu lại các kết quả.

        ii. Sử dụng LDA. Lưu lại kết quả

    Hãy so sánh kết quả trong 2 trường hợp i) và ii) để tìm ra quy trình phù hợp khi xử lý giảm chiều dữ liệu để phục vụ bài toán phân loại (tương tự với hồi quy). Hãy giải thích. Sau đó so sánh kết quả trong trường hợp dữ liệu nguyên bản và dữ liệu giảm chiều.

Ví dụ B2 (Bài tập thực hành 2): Hãy sử dụng đoạn mã lệnh giảm số chiều bằng phương pháp phân tích thành phần chính đã học và tham khảo phần đọc, hiển thị dữ liệu hình ảnh chữ số viết tay đã có trong bài thực hành phần Multinomial Logistic Regression, để áp dụng vào dữ liệu ảnh chữ số viết tay đã được cung cấp trong bài thực hành tuần trước:

1) Đọc dữ liệu ảnh, lấy tập dữ liệu 5000 ảnh bất kỳ; giảm số chiều dữ liệu xuống còn 100 chiều (từ 28x28 = 784 chiều ban đầu).

2) Áp dụng phương pháp phân loại nhiều lớp Multinomial Logistic Regression (tham khảo bài trước) để phân loại để phân loại tập dữ liệu đã chọn trong ý 1), tỷ lệ train:validation là 0.7:0.3. Hãy so sánh kết quả (độ chính xác và thời gian) chạy mô hình phân loại trong hai trường hợp:

    a) Dữ liệu nguyên bản (giữ nguyên 28x28 = 784 chiều)

b) Dữ liệu đã qua giảm chiều: PCA còn 100 chiều và LDA còn 8 chiều. So sánh và cho biết cách nào phù hợp hơn.

Ví dụ B.4. (Bài tập thực hành 3): Trong phần thực hành này, chúng ta sẽ dùng LDA và PCA để giảm số chiều của một vài tập dữ liệu thực tế và áp dụng mô hình phân loại đã học để phân lớp.

Bài tập tự thực hành

Hãy sử dụng đoạn mã lệnh giảm số chiều bằng phương pháp phân tích thành phần chính đã học (tham khảo mã lệnh áp dụng cho dữ liệu ảnh đã được gửi):

Giải nén tệp face_data.zip đính kèm sẽ có 165 ảnh các khuôn mặt của 15 người, mỗi người chụp ở 11 trạng thái khác nhau, dưới định dạng tệp png. Có một số thư viện Python hỗ trợ việc đọc ảnh và trả về mảng số ứng với các điểm ảnh (xem lại khái niệm ảnh số trong phần đọc dữ liệu chữ số viết tay). Tên ảnh được ghép bởi prefix 'subject', chỉ số của người tương ứng ghi theo kiểu 01, 02 … đến 15 và dấu chấm "." . Tiếp theo là các trạng thái, gồm 11 trạng thái

['centerlight', 'glasses', 'happy', 'leftlight', 'noglasses', 'normal', 'rightlight', 'sad', 'sleepy', 'surprised', 'wink']

Cuối cùng là phần mở rộng trong tên tệp, .png.

Đoạn chương trình dưới đây cho phép sử dụng phương thức của thư viện OpenCV (cv2) để đọc tệp ảnh và lấy ra ma trận ứng với các điểm ảnh, sau đó duỗi thẳng thành 1 vector với số chiều D = height

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

# path to the database - change it if needed
path = 'D:\\Teach_n_Train\\Machine Learning\\code\\yale\\yalefaces_data\\'

ids = range(1, 16) # 15 people
states = ['centerlight', 'glasses', 'happy', 'leftlight',
          'noglasses', 'normal', 'rightlight','sad',
          'sleepy', 'surprised', 'wink' ]
prefix = 'subject'
surfix = '.png' #file extension is png

# open one picture to get the image's size
fn = prefix + '01.' + states[0] + surfix
im = cv2.imread(path + fn, 0)

h = im.shape[0] # hight
w = im.shape[1] # width

D = h * w
N = len(states)*15
print(N, D, h, w)

X = np.zeros((D, N))

# collect all data
count = 0

# there are 15 people
for person_id in range(1, 16):
    for state in states:

        # get name of each image file
        fn = path + prefix + str(person_id).zfill(2) + '.' + state + surfix
```

```
        # open the file and read as grey image
        tmp = cv2.imread(fn, cv2.IMREAD_GRAYSCALE)

        # then add image to dataset X
        X[:, cnt] = tmp.reshape(D)
        count += 1
```

Áp dụng cho dữ liệu ảnh khuôn mặt (của 15 người, mỗi người có 11 trạng thái):

a. Giảm số chiều dữ liệu xuống còn 135.
b. Áp dụng các phương pháp phân loại nhiều lớp: Multinomial Logistic Regression và Naïve Bayes (đã có code) để phân loại, tỷ lệ train:test là 0.7:0.3
c. Coi toàn bộ 165 ảnh đầu vào là train, tìm 5 ảnh chân dung (tùy ý) sau đó đưa về cùng kích thước như trong ảnh dữ liệu (H = 320; W = 243), với phần chân dung lệch sang tay phải theo hướng người nhìn vào. Thử dụng mô hình đã huấn luyện, chạy test xem 5 ảnh dữ liệu mới sẽ thuộc nhóm nào.