Mô hình phân định tuyến tính (linear discriminant analysis)

@Lê Ngọc Thạch

https://ThachLN.github.io

Mô hình phân định tuyến tính

- Giới thiệu ý tưởng và mô hình
- Lí thuyết
- Ứng dụng

Phân tích phân định tuyến tính (LDA)

- Chúng ta có một nhóm biến tiên lượng (X), và 1 biến outcome Y mang tính phân nhóm (categorical)
- Chúng ta muốn tìm một hàm số để dùng X phân nhóm biến Y

Ứng dụng
phân tích phân định tuyến tính

- Chúng ta muốn biết một cá nhân nào đó bị ung thư. Biến outcome ở đây là có/không (hay 1/0)
- Các biến giúp chúng ta tiên lượng có thể là hút thuốc lá, số lần ho trong ngày, phơi nhiễm hoá chất độc hại, v.v.
- Chúng ta muốn tìm hàm số X để tiên lượng outcome.

Ý tưởng phân tích phân định tuyến tính

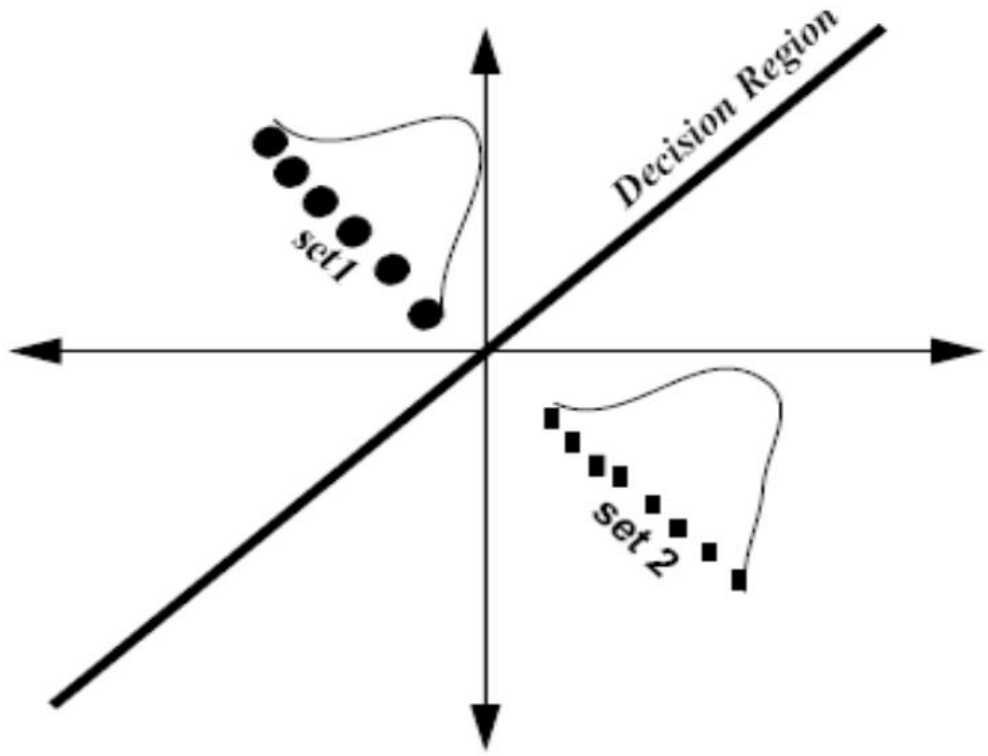
- Linear discriminant analysis (LDA) xây dựng một hệ thống phương trình phân định (discriminant equations) D_i để phân biệt outcome
- Hàm số phân định là:

$$D_i = b_0 + \sum_{k=1}^{p} b_k X_k$$

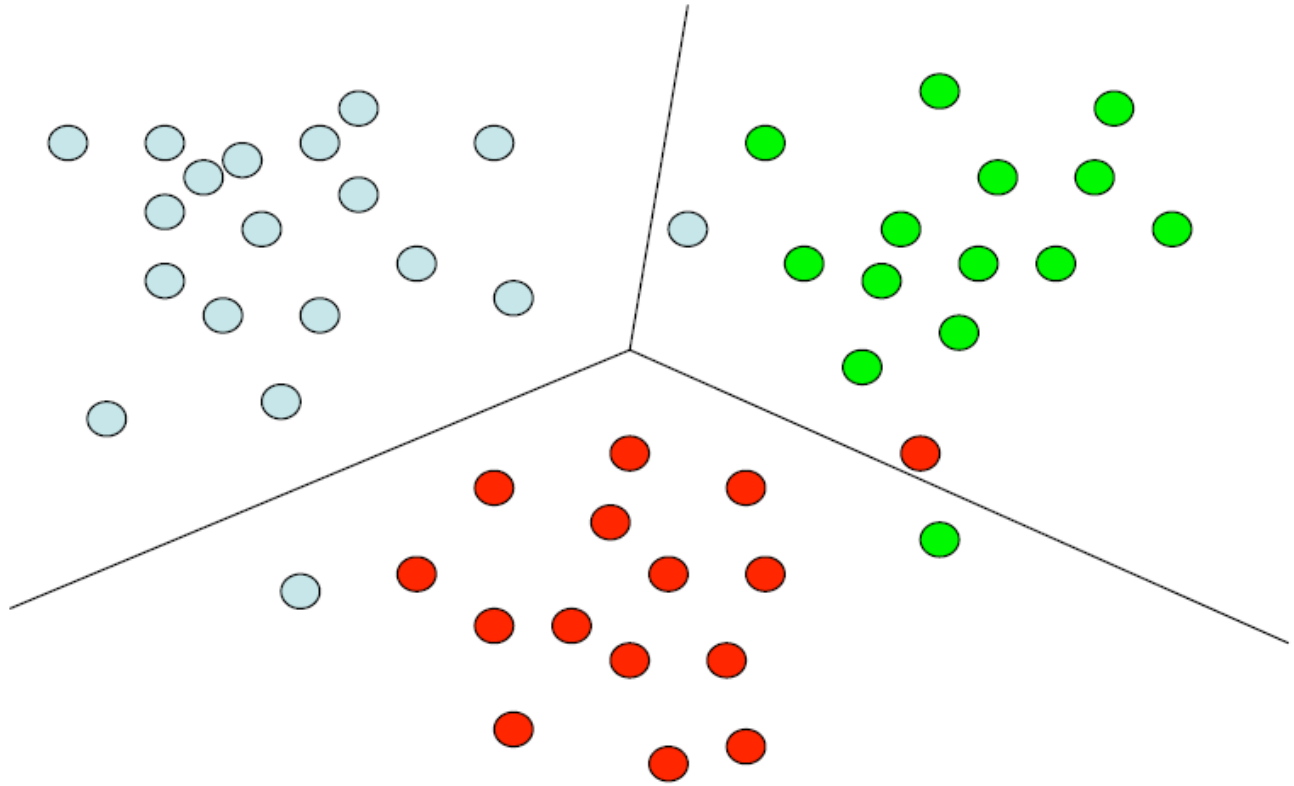
Cách tiếp cận

- Tìm trọng số của hàm số phân định sao cho **tỉ số** phương sai giữa các nhóm (between groups) trên phương sai trong mỗi nhóm (within group) là cao nhất
- Số hàm số phân định = min(số nhóm – 1, p)
- Chúng ta cần số liệu cho

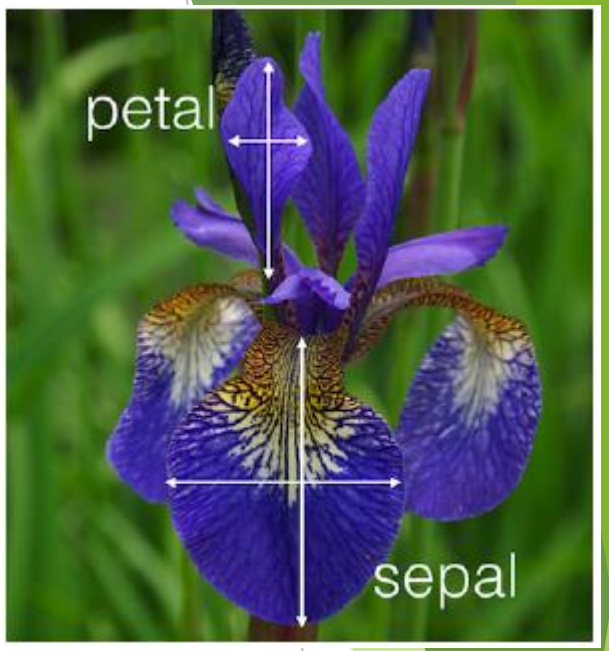
Trường hợp có 2 nhóm



Trường hợp 3 nhóm



Ví dụ nổi tiếng – iris data

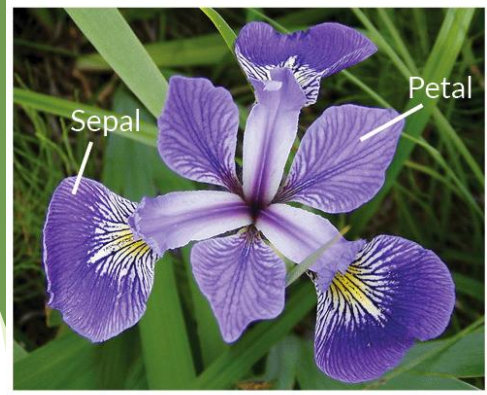


- Dữ liệu về hoa iris cung cấp đo lường liên quan đến chiều dài (sepal length, petal length), bề rộng (width)
  - của 50 loại hoa
  - từ 3 giống (setosa, versicolor, virginica)

Python

```
import pandas as pd
file_path = 'https://thachln.github.io/datasets/iris-data.csv'
df = pd.read_csv(file_path)
df.head()
```

	S.Length	S.Width	P.Length	P.Width	Species
0	5.1	3.5	1.4	0.2	I.setosa
1	4.9	3.0	1.4	0.2	I.setosa
2	4.7	3.2	1.3	0.2	I.setosa
3	4.6	3.1	1.5	0.2	I.setosa
4	5.0	3.6	1.4	0.2	I.setosa



Iris Versicolor


Iris Setosa


Iris Virginica

Phân tích

- Mục tiêu: dùng đo lường để phân biệt 3 loài hoa
- Chúng ta bắt đầu bằng LDA – mô hình phân định tuyến tính

Cảm nhận vài dữ liệu

```
# Xem 4 cột dữ liệu đầu tiên từ 0 đến 3
X = df.iloc[:, 0:4].values
X
```

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       ...
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

```
df.head()
```

	S.Length	S.Width	P.Length	P.Width	Species
0	5.1	3.5	1.4	0.2	I.setosa
1	4.9	3.0	1.4	0.2	I.setosa
2	4.7	3.2	1.3	0.2	I.setosa
3	4.6	3.1	1.5	0.2	I.setosa
4	5.0	3.6	1.4	0.2	I.setosa

Cảm nhận vài dữ liệu

```
# Xem cột dữ liệu thứ 4 (tính từ 0)
Y = df.iloc[:, 4].values
Y
```

```
array(['I.setosa', 'I.setosa', 'I.setosa', 'I.setosa', 'I.setosa',
       'I.setosa', 'I.setosa', 'I.setosa', 'I.setosa', 'I.setosa',
       'I.setosa', 'I.setosa', 'I.setosa', 'I.setosa', 'I.setosa',
       'I.setosa', 'I.setosa', 'I.setosa', 'I.setosa', 'I.setosa',
...
 'I.virginica', 'I.virginica', 'I.virginica', 'I.virginica',
       'I.virginica', 'I.virginica', 'I.virginica', 'I.virginica',
       'I.virginica', 'I.virginica', 'I.virginica', 'I.virginica'],
      dtype=object)
```

Chuyển nhãn từ tên sang số

```
from sklearn.preprocessing import LabelEncoder
class_le = LabelEncoder()
Y_num = class_le.fit_transform(Y.values)
Y_num
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Cách lấy dữ liệu nhóm theo nhãn

```
group_0 = X[Y_num==0]
group_0
# The transposed array.
group_0.T
```

```
group_0
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
...
 [5.3, 3.7, 1.5, 0.2],
       [5. , 3.3, 1.4, 0.2]])
```

```
group_0.T

rray([[5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.6, 5. , 4.4, 4.9, 5.4, 4.8, 4.8,
       4.3, 5.8, 5.7, 5.4, 5.1, 5.7, 5.1, 5.4, 5.1, 4.6, 5.1, 4.8, 5. ,
       5. , 5.2, 5.2, 4.7, 4.8, 5.4, 5.2, 5.5, 4.9, 5. , 5.5, 4.9, 4.4,
       5.1, 5. , 4.5, 4.4, 5. , 5.1, 4.8, 5.1, 4.6, 5.3, 5. ],
...
 [0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.2, 0.1,
       0.1, 0.2, 0.4, 0.4, 0.3, 0.3, 0.3, 0.2, 0.4, 0.2, 0.5, 0.2, 0.2,
       0.4, 0.2, 0.2, 0.2, 0.2, 0.4, 0.1, 0.2, 0.2, 0.2, 0.2, 0.1, 0.2,
       0.2, 0.3, 0.3, 0.2, 0.6, 0.4, 0.3, 0.2, 0.2, 0.2, 0.2]])
```

Chuẩn bị ma trận 4x4

```
import numpy as np
s_w = np.zeros((4,4))
s_w
```

```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

Tính hệ số tương quan trong mỗi nhóm

```
# Ma trận tương quan của 4 cột dữ liệu trong tập X
for i in range(3):
    s_w += np.cov(X[Y_num==i].T)


s_w
```

```
array([[0.795 , 0.2782, 0.5025, 0.1152],
       [0.2782, 0.3462, 0.1657, 0.0981],
       [0.5025, 0.1657, 0.5556, 0.128 ],
       [0.1152, 0.0981, 0.128 , 0.1256]])
```

Tính means trong từng nhóm

```
# Construct between-class scatter matrix s_b
N = np.bincount(Y_num) # number of samples for given class
vecs=[]
[vecs.append(np.mean(X[Y_num==i], axis=0)) for i in range(3)] # class means
vecs
```

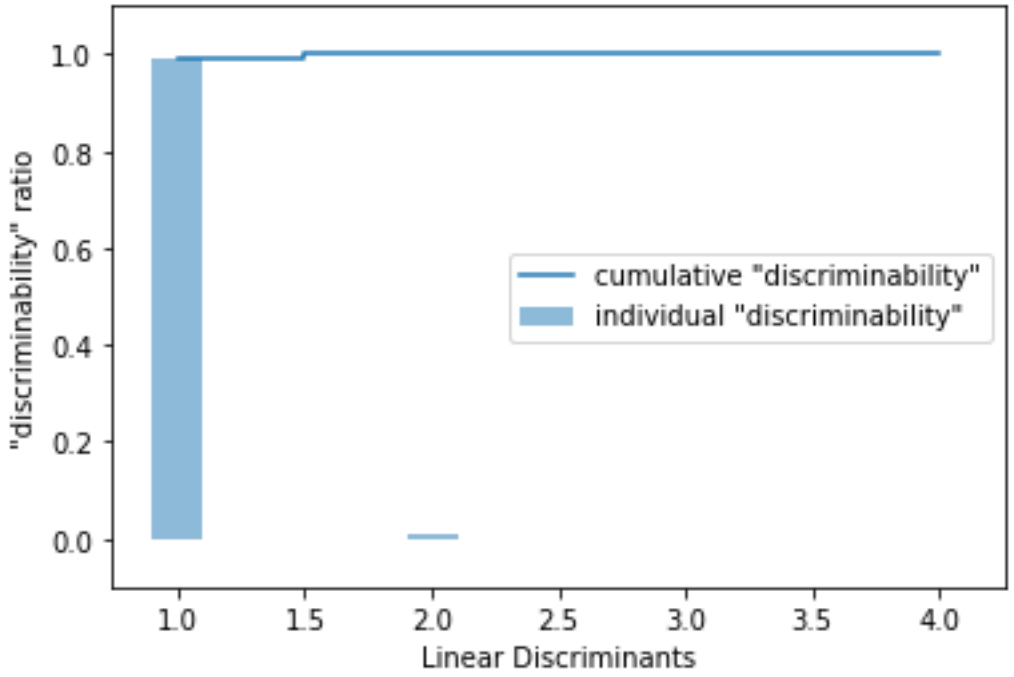
```
[array([5.006, 3.428, 1.462, 0.246]),
 array([5.936, 2.77 , 4.26 , 1.326]),
 array([6.588, 2.974, 5.552, 2.026])]
```

Tính means tổng thể

```
mean_overall = np.mean(X, axis=0) # overall mean
mean_overall
```

array([5.8433, 3.0573, 3.758 , 1.1993])

Vẽ biểu đồ

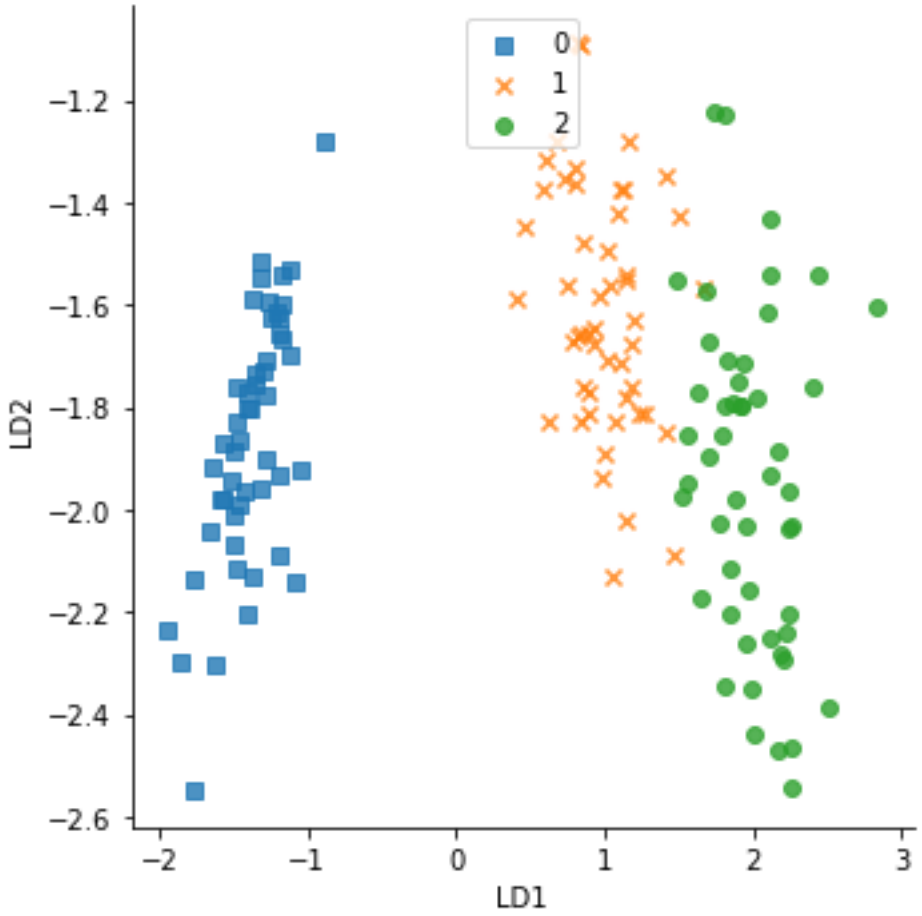



```
# Plot main LDA components
import matplotlib.pyplot as plt
tot = sum(eigen_vals.real)
discr = [(i / tot) for i in sorted(eigen_vals.real, reverse=True)]
cum_discr = np.cumsum(discr)
plt.bar(range(1, 5), discr, width=0.2,alpha=0.5, align='center',label='individual "discriminability"')
plt.step(range(1, 5), cum_discr, where='mid',label='cumulative "discriminability"')
plt.ylabel('"discriminability" ratio')
plt.xlabel('Linear Discriminants')
plt.ylim([-0.1, 1.1])
plt.legend(loc='best')
plt.show()
```

Vẽ biểu đồ

```
# Plot transformed features in LDA subspace
data=pd.DataFrame(X_train_lda)
data['class']=Y_num
data.columns=["LD1","LD2","class"]
data.head()


markers = ['s', 'x','o']
import seaborn as sns
sns.lmplot(x="LD1", y="LD2", data=data, markers=markers,fit_reg=False,
hue='class', legend=False)
plt.legend(loc='upper center')
plt.show()
```

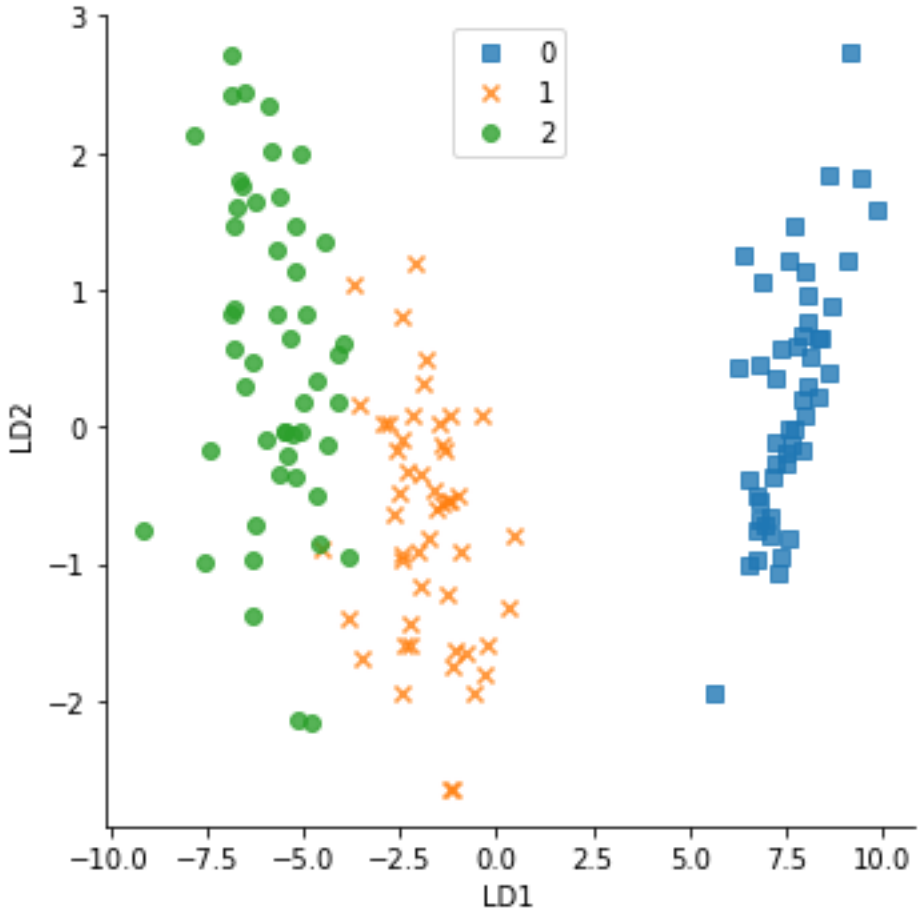


Vẽ biểu đồ

```
# LDA implementation using scikit-learn
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=2)
X_train_lda = lda.fit_transform(X, Y_num)

data = pd.DataFrame(X_train_lda)
data['class'] = Y_num
data.columns=["LD1","LD2","class"]
data.head()

markers = ['s', 'x','o']
colors = ['r', 'b','g']
sns.lmplot(x="LD1", y="LD2", data=data, hue='class',
markers=markers,fit_reg=False,legend=False)
plt.legend(loc='upper center')
plt.show()
```



Tham khảo

https://ThachLN.github.io