ĐẠI HỌC BÁCH KHOA – ĐẠI HỌC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN

Môn học

Khoa Học Dữ liệu

Nội dung môn học

- Chương 1: **Tổng quan KHDL**
  - Cách mạng công nghiệp lần thứ tư
  - Khoa học dữ liệu là gì?
  - Nguyên lý và phương pháp của khoa học dữ liệu
- Chương 2: **Các công cụ toán học trong KHDL**
  - Phân tích dữ liệu khám phá
  - Phân bố dữ liệu và phân bố lấy mẫu
  - Kiểm định giả thiết
  - Suy diễn Bayes

Nội dung môn học

- Chương 3:**Lập trình Python cho KHDL (5 buổi)**
  - Các công cụ lập trình.
  - Cơ bản về Python
  - Xử lý dữ liệu vector và ma trận với Numpy
  - Thao tác dữ liệu với Pandas
  - Trực quan hoá dữ liệu với Matplotlib và Seaborn
- **Kiểm tra giữa kỳ**
- Chương 4:**Thu thập và lưu trữ dữ liệu**
  - Đọc và ghi dữ liệu từ file
  - Thu thập dữ liệu từ web
  - Sử dụng các API
  - Tương tác với cơ sở dữ liệu

Nội dung môn học

- Chương 5:Xử lý dữ liệu
  - Thống kê mô tả về dữ liệu
  - Làm sạch dữ liệu
  - Chuẩn hoá dữ liệu
- Bài tập Chương 4 và 5
- Chương 6: Phân tích dữ liệu chuỗi thời gian
- **Chương 7. Phân tích dữ liệu nhiều chiều**
- Chương 8. Học máy và ứng dụng
- Báo cáo Bài tập nhóm

Phân tích dữ liệu nhiều chiều

- Phân tích dữ liệu nhiều chiều (**Multivariate analysis**) là kỹ thuật xử lý dữ liệu có nhiều biến nhằm mục đích giải thích, dự đoán hay tạo các mối quan hệ giữa các biến dữ liệu với nhau.

Phân tích dữ liệu nhiều chiều

- Có 2 kỹ thuật chính:
  - Phân tích phụ thuộc (Analysis of dependence) : Trong đó một (hoặc nhiều) biến là biến phụ thuộc, được giải thích hoặc dự đoán bởi những biến khác.
    - Multiple regression, **Discriminant analysis**, Multivariate analysis of variance, Partial Least Square
  - Phân tích phụ thuộc lẫn nhau (Analysis of interdependence): Không có biến nào được coi là "phụ thuộc“ → Xem xét mối quan hệ giữa các biến.
    - **cluster analysis**, factor analysis, **principal component analysis**.

Phân tích dữ liệu nhiều chiều

- Việc lựa chọn kỹ thuật phân tích dựa trên:
  - Loại dữ liệu cần phân tích:
    - Dữ liệu định danh (Nominal data):
      - tập các nhãn dùng để mô tả, phân loại các đối tượng. Ví dụ: tên màu, mẫu nhân viên.
    - Dữ liệu thứ tự (Ordinary data):
      - tập các phần tử chỉ định một thứ tự được sắp. Ví dụ: xếp loại (kém, trung bình, khá, giỏi)
    - Dữ liệu nhị phân (Binary data):
      - một trường hợp đặc biệt của kiểu định danh, các dữ liệu thuộc kiểu này chỉ mang một trong hai giá trị. Ví dụ: kiểu boolean (true, false), giới tính (nam, nữ).
    - Dữ liệu Số nguyên (Integer Data)
    - Dữ liệu Khoảng (Interval data)
    - Dữ liệu Tỷ lệ (Ratio-scaled data)

Phân tích dữ liệu nhiều chiều

- Việc lựa chọn kỹ thuật phân tích dựa trên (tt):
  - Mục đích phân tích:
    - Mục đích giảm chiều dữ liệu
      - Principal component analysis (PCA)
      - Correspondence analysis (CA)
    - Mục đích phân loại dữ liệu
      - Cluster analysis
      - Discriminant analysis


Phân tích thành phần chính
Principal component analysis (PCA)

Phân tích thành phần chính

- Phân tích thành phần chính - Principal component analysis (PCA):
  - Là kỹ thuật sử dụng phép biến đổi trực giao để biến đổi một tập hợp dữ liệu từ một không gian nhiều chiều sang một không gian mới ít chiều hơn (2 hoặc 3 chiều) nhằm tối ưu hóa việc thể hiện sự biến thiên của dữ liệu.

Phân tích thành phần chính

- Ưu điểm:
  - Giảm số chiều của không gian chứa dữ liệu khi nó có số chiều lớn
  - Xây dựng những trục tọa độ mới, thay vì giữ lại các trục của không gian cũ, nhưng lại có khả năng biểu diễn dữ liệu tốt tương đương
  - Tạo điều kiện để các liên kết tiềm ẩn của dữ liệu có thể được khám phá trong không gian mới, mà nếu đặt trong không gian cũ thì khó phát hiện vì những liên kết này không thể hiện rõ
  - Đảm bảo các trục tọa độ trong không gian mới luôn trực giao đôi một với nhau

Phân tích thành phần chính

- Cho tập dữ liệu

$$T = \left\{ \boldsymbol{x}_i \,\middle|\, \boldsymbol{x}_i \in \boldsymbol{R}^d \right\}_{i=1}^{n}$$

- Với:
  - $\boldsymbol{x}_i$ là sample thứ i
  - d là số chiều dữ liệu
  - n là số lượng sample
- PCA tìm trục tọa độ **w** mới sao cho khi chiếu dữ liệu lên nó ($\mathbf{w}^{\mathrm{T}}\mathbf{x}_i$), phương sai sẽ trở nên lớn nhất

Phân tích thành phần chính

- Gọi $\mathbf{w} \in \mathbf{R}^d$ là vector đơn vị thể hiện trục tọa độ mà dữ liệu T chiếu lên. Bài toán tối ưu cần giải quyết là:

$$\max_{\mathbf{w}} \quad \frac{1}{n}\sum_{i}^{n}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - \mathbf{w}^{\mathrm{T}}\boldsymbol{\mu}\right)^2$$

$$\mathbf{s.t.} \qquad \|\mathbf{w}\|^2 = 1$$

- Với:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i}^{n}\mathbf{x}_i$$

Phân tích thành phần chính

- Sử dụng Phương pháp nhân tử Lagrange với $\lambda \geq 0$ ta cần cực đại hóa hàm Lagrange:

$$L(\mathbf{w}, \lambda) = \frac{1}{n}\sum_{i}^{n}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i - \mathbf{w}^{\mathrm{T}}\boldsymbol{\mu}\right)^2 - \lambda(\|\mathbf{w}\|^2 - 1)$$

- Từ

$$\frac{\delta L(\mathbf{w}, \lambda)}{\delta \mathbf{w}} = 0$$

- Ta có:

$$\frac{1}{n}\sum_{i}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}}\,\mathbf{w} = \lambda\mathbf{w}$$

Phân tích thành phần chính

- Để tìm $\mathbf{w}$ ta có thể giải bài toán tìm vector riêng:

$$\mathbf{Cw} = \lambda \mathbf{w}$$

- Với:

$$\mathbf{C} = \frac{1}{n}\sum_{i}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathrm{T}}$$

là ma trận hiệp phương sai của dữ liệu

PCA với Python

- Sử dụng thư viện prince
  - `pip install prince`

	Sepal length	Sepal width	Petal length	Petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

- Load dữ liệu iris:

```
import pandas as pd
import prince
from sklearn import datasets

X, y = datasets.load_iris(return_X_y=True)
X = pd.DataFrame(data=X, columns=['Sepal length', 'Sepal width', 'Petal lengt
h', 'Petal width'])
y = pd.Series(y).map({0: 'Setosa', 1: 'Versicolor', 2: 'Virginica'})
X.head()
```

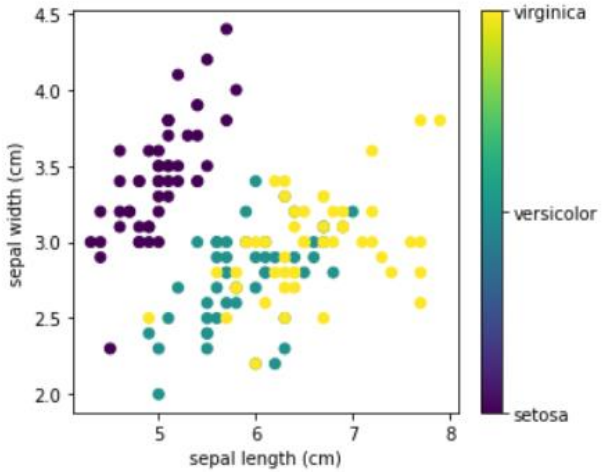
Dữ liệu trước và sau khi sử dụng PCA

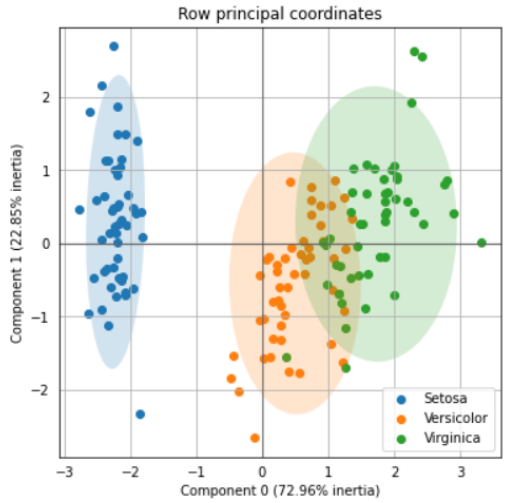
	Sepal length	Sepal width	Petal length	Petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

	0	1	2	3
0	-2.264703	0.480027	-0.127706	-0.024168
1	-2.080961	-0.674134	-0.234609	-0.103007
2	-2.364229	-0.341908	0.044201	-0.028377
3	-2.299384	-0.597395	0.091290	0.065956
4	-2.389842	0.646835	0.015738	0.035923

```
pca = prince.PCA(
    n_components=2,
    n_iter=3,
    rescale_with_mean=True,
    rescale_with_std=True,
    copy=True,
    check_input=True,
    engine='auto',
    random_state=42
)
pca = pca.fit(X)
pca.transform(X).head()
```

Plot 2 trục chính






```
ax = pca.plot_row_coordinates(
    X,
    ax=None,
    figsize=(6, 6),
    x_component=0,
    y_component=1,
    labels=None,
    color_labels=y,
    ellipse_outline=False,
    ellipse_fill=True,
    show_points=True
)
ax.get_figure()
```

Sau khi áp dụng PCA, dữ liệu “có vẻ” được phân loại tốt hơn!!

Mức độ đóng góp của các trục chính

```
[55] pca.explained_inertia_
```

```
array([0.72962445, 0.22850762])
```

```
[56] pca.eigenvalues_
```

```
array([2.91849782, 0.91403047])
```

```
[57] pca.total_inertia_
```

```
4.000000000000001
```

```
pca.column_correlations(X)
```

	0	1
Petal length	0.991555	0.023415
Petal width	0.964979	0.064000
Sepal length	0.890169	0.360830
Sepal width	-0.460143	0.882716

```
[59] pca.row_contributions(X).head()
```

	0	1
0	1.757369	0.252098
1	1.483777	0.497200
2	1.915225	0.127896
3	1.811606	0.390447
4	1.956947	0.457748

```
pca.inverse_transform(pca.transform(X)).head()
```

	0	1	2	3
0	5.018949	3.514854	1.466013	0.251922
1	4.738463	3.030433	1.603913	0.272074
2	4.720130	3.196830	1.328961	0.167414
3	4.668436	3.086770	1.384170	0.182247
4	5.017093	3.596402	1.345411	0.206706


Phân tích tương ứng
Correspondence analysis (CA)

Phân tích tương ứng

- Phân tích tương ứng - Correspondence analysis (CA) là
  - phương pháp trực quan để phân tích dữ liệu được biểu diễn bằng các bảng dữ liệu 2 chiều. Kết quả của CA là bản đồ các điểm. Trong đó các điểm biểu diễn cho các dòng và các cột của bảng. Vị trí của các điểm trong bản đồ cho biết sự tương tự (similarities) giữa các dòng, sự tương tự giữa các cột và sự kết hợp giữa dòng và cột trong bảng.

Mục đích của CA

- Mục đích của CA giúp phát hiện các mối quan hệ tiềm ẩn trong các bảng dữ liệu lớn.

	Column Variable						
Row Variable	B_1	B_2	$\cdots$	B_j	$\cdots$	B_s	Row Total
A_1	n_{11}	n_{12}	$\cdots$	n_{1j}	$\cdots$	n_{1s}	n_{1+}
A_2	n_{21}	n_{22}	$\cdots$	n_{2j}	$\cdots$	n_{2s}	n_{2+}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
A_i	n_{i1}	n_{i2}	$\cdots$	n_{ij}	$\cdots$	n_{is}	n_{i+}
$\vdots$	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
A_r	n_{r1}	n_{r2}	$\cdots$	n_{rj}	$\cdots$	n_{rs}	n_{r+}
Column total	n_{+1}	n_{+2}	$\cdots$	n_{+j}	$\cdots$	n_{+s}	n

Ứng dụng khác của CA

- Cho một bảng gồm danh sách các đối tượng chứa các thuộc tính A và B, trong đó
  - Thuộc tính A có dạng "dữ liệu định danh":
    - $A_1, A_2, \ldots, A_s$
  - Thuộc tính B cũng có dạng "dữ liệu định danh":
    - B1, B2,…, B_r
  - Cần tìm mối liên hệ giữa A và B
- Ví dụ:
  - Cho Bảng dữ liệu về màu tóc và màu mắt của người.
  - Tìm mỗi liên hệ của màu tóc và màu mắt.

STT	Màu Mắt	Màu Tóc
1	Nâu	Đen
2	Xám	Xám
3	Xám	Vàng

Chuyển đổi dữ liệu

- Đối với thuộc tính là dữ liệu định danh, ta cần tạo các thuộc tính mới bằng cách biến nó thành các thuộc tính dạng dữ liệu nhị phân.

$$X_{ij} = \begin{cases} 1, & \text{if the } j\text{th individual belongs to } A_i \\ 0, & \text{otherwise} \end{cases}$$

$$Y_{ij} = \begin{cases} 1, & \text{if the } i\text{th individual belongs to } B_j \\ 0, & \text{otherwise} \end{cases}$$

STT	Màu Mắt	Màu Tóc
1	Nâu	Đen
2	Xám	Xám
3	Xám	Vàng

STT	Mắt Nâu	Mắt Xám
1	1	0
2	0	1
3	0	1

STT	Tóc Đen	Tóc Xám	Tóc Vàng
1	1	0	0
2	0	1	0
3	0	0	1

Bảng tầng suất

- Từ các thuộc tính mới

$$\overset{r\times n}{\mathcal{X}} = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix}$$

$$\overset{s\times n}{\mathcal{Y}} = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 & \cdots & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 1 & \cdots & 1 \end{pmatrix}$$

STT	Mắt Nâu	Mắt Xám
1	1	0
2	0	1
3	0	1

STT	Tóc Đen	Tóc Xám	Tóc Vàng
1	1	0	0
2	0	1	0
3	0	0	1

- Ta tính được tần suất quan hệ giữa 2 thuộc tính

$$\mathcal{X}\mathcal{Y}^{\tau} = \begin{pmatrix} n_{11} & n_{12} & \cdots & n_{1s} \\ n_{21} & n_{22} & \cdots & n_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ n_{r1} & n_{r2} & \cdots & n_{rs} \end{pmatrix} = \mathbf{N}$$

	Mắt Nâu	Mắt Xám
Tóc Đen	1	0
Tóc Xám	0	1
Tóc Vàng	0	1

Ma trận hiệp phương sai

- Xây dựng ma trận hiệp phương sai của cả 2 thuộc tính

$$\begin{pmatrix} \mathcal{X} \\ \mathcal{Y} \end{pmatrix} \begin{pmatrix} \mathcal{X} \\ \mathcal{Y} \end{pmatrix}^{\tau} = \begin{pmatrix} n\mathbf{D}_r & \mathbf{N} \\ \mathbf{N}^{\tau} & n\mathbf{D}_c \end{pmatrix},$$

$$\mathbf{D}_r = n^{-1}\mathcal{X}\mathcal{X} = \mathrm{diag}\{n_{1+}/n, \ldots, n_{r+}/n\},$$

$$\mathbf{D}_c = n^{-1}\mathcal{Y}\mathcal{Y}^{\tau} = \mathrm{diag}\{n_{+1}/n, \ldots, n_{+s}/n\}.$$

- Cuối cùng, quy về bài toán tìm vector riêng để giải tương tự PCA.

PCA với Python

- Cho trước dữ liệu:

	Hair Color					
Eye Color	Fair	Red	Medium	Dark	Black	Totals
Blue	326	38	241	110	3	718
Light	688	116	584	188	4	1,580
Medium	343	84	909	412	26	1,774
Dark	98	48	403	681	85	1,315
Totals	1,455	286	2,137	1,391	118	5,387

```
import pandas as pd
pd.set_option('display.float_format', lambda x: '{:.6f}'.format(x))
X = pd.DataFrame(
    data=[
            [326, 38, 241, 110, 3],
            [688, 116, 584, 188, 4],
            [343, 84, 909, 412, 26],
            [98, 48, 403, 681, 85]
    ],
    columns=pd.Series(['Fair', 'Red', 'Medium', 'Dark', 'Black']),
    index=pd.Series(['Blue', 'Light', 'Medium', 'Dark'])
)
```

Biến đổi dữ liệu

```
import prince
ca = prince.CA(
    n_components=2,
    n_iter=3,
    copy=True,
    check_input=True,
    engine='auto',
    random_state=42
)
X.columns.rename('Hair color', i
nplace=True)
X.index.rename('Eye color', inpl
ace=True)
ca = ca.fit(X)
```

```
ca.row_coordinates(X)
```

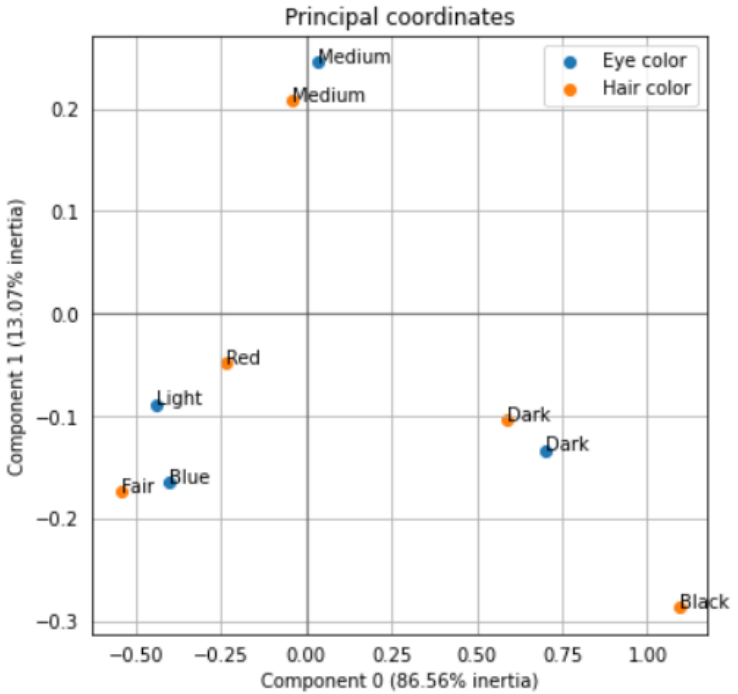
	0	1
Blue	-0.400300	-0.165411
Light	-0.440708	-0.088463
Medium	0.033614	0.245002
Dark	0.702739	-0.133914

```
ca.column_coordinates(X)
```

	0	1
Fair	-0.543995	-0.173844
Red	-0.233261	-0.048279
Medium	-0.042024	0.208304
Dark	0.588709	-0.103950
Black	1.094388	-0.286437

Hiển thị mối tương quan

```
ax = ca.plot_coordinates(
    X=X,
    ax=None,
    figsize=(6, 6),
    x_component=0,
    y_component=1,
    show_row_labels=True,
    show_col_labels=True
)
ax.get_figure().savefig('ca_coordinates.svg')
```

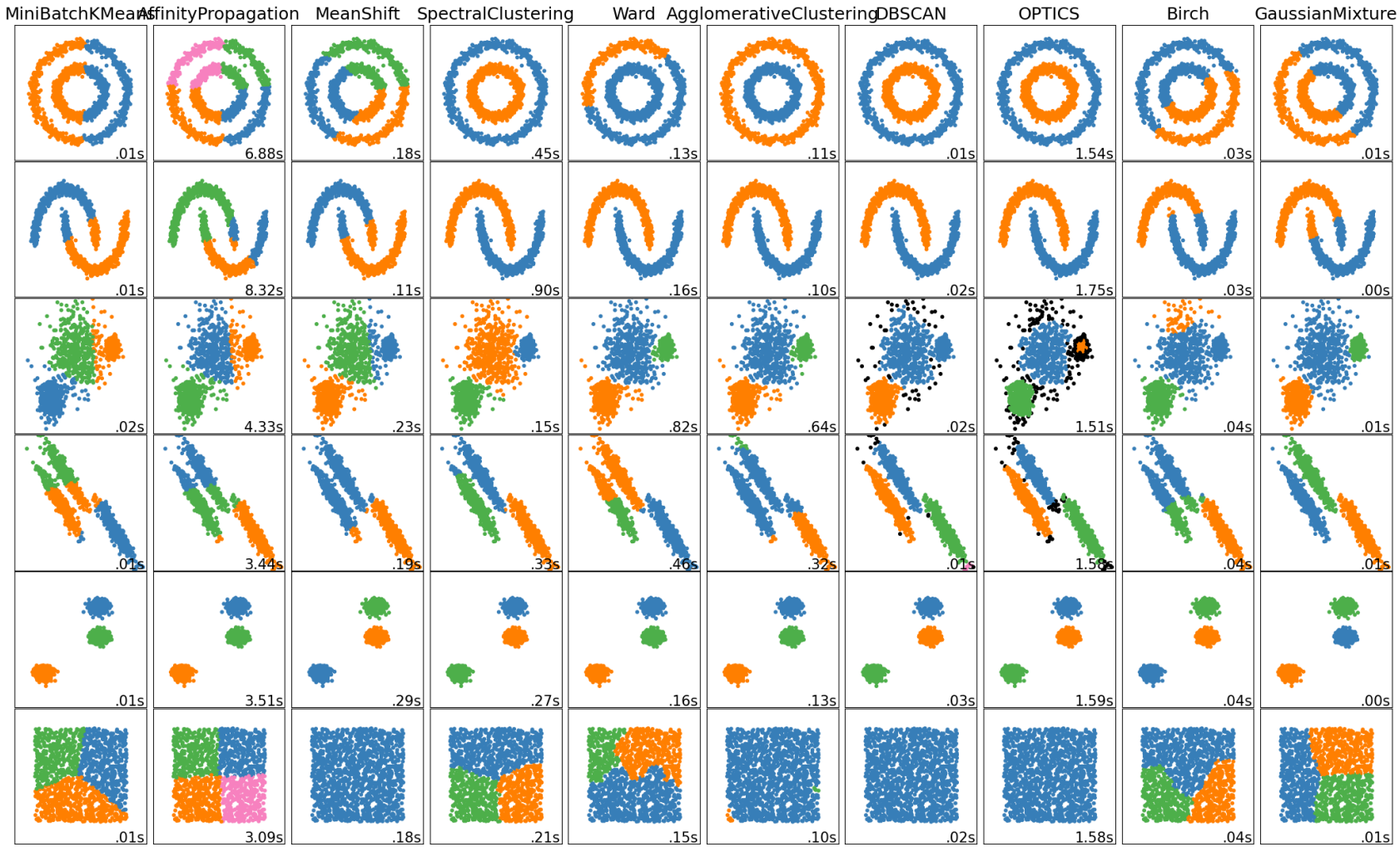



Phân tích cụm
Cluster analysis

Phân tích cụm - cluster analysis

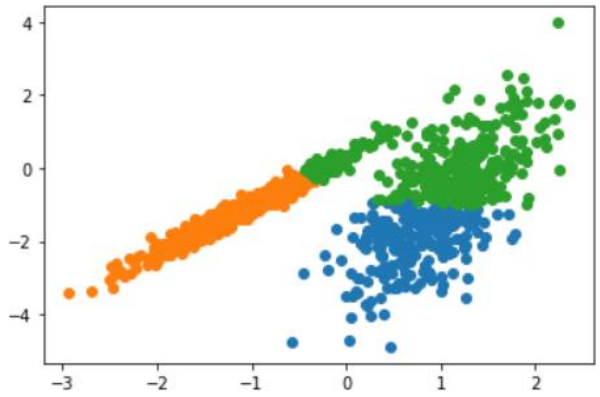
- Phân tích cụm (cluster analysis) là một tác vụ gom nhóm một tập các đối tượng theo cách các đối tượng cùng nhóm (gọi là cụm, cluster) sẽ có tính giống nhau (theo các đặc tính nào đó) hơn so với các đối tượng ngoài nhóm hoặc thuộc các nhóm khác.
- Phân tích cụm có nguồn gốc ở lĩnh vực nhân chủng học do Driver và Kroeber đề xuất năm 1932

Các phương pháp phân cụm



Sử dụng thư viện sciket-learn

- Cài đặt thư viện:
  - pip install scikit-learn
- Ví dụ, phân cụm với thuật toán k-means:
- Tham khảo thêm:
  - https://machinelearningmastery.com/clustering-algorithms-with-python/




```
# k-means clustering
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import KMeans
from matplotlib import pyplot
# define dataset
X, _ = make_classification(n_samples=1000, n_featur
es=2, n_informative=2, n_redundant=0, n_clusters_pe
r_class=1, random_state=4)
# define the model
model = KMeans(n_clusters=3)
# fit the model
model.fit(X)
# assign a cluster to each example
yhat = model.predict(X)
# retrieve unique clusters
clusters = unique(yhat)
# create scatter plot for samples from each cluster
for cluster in clusters:
  # get row indexes for samples with this cluster
  row_ix = where(yhat == cluster)
  # create scatter of these samples
  pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
```

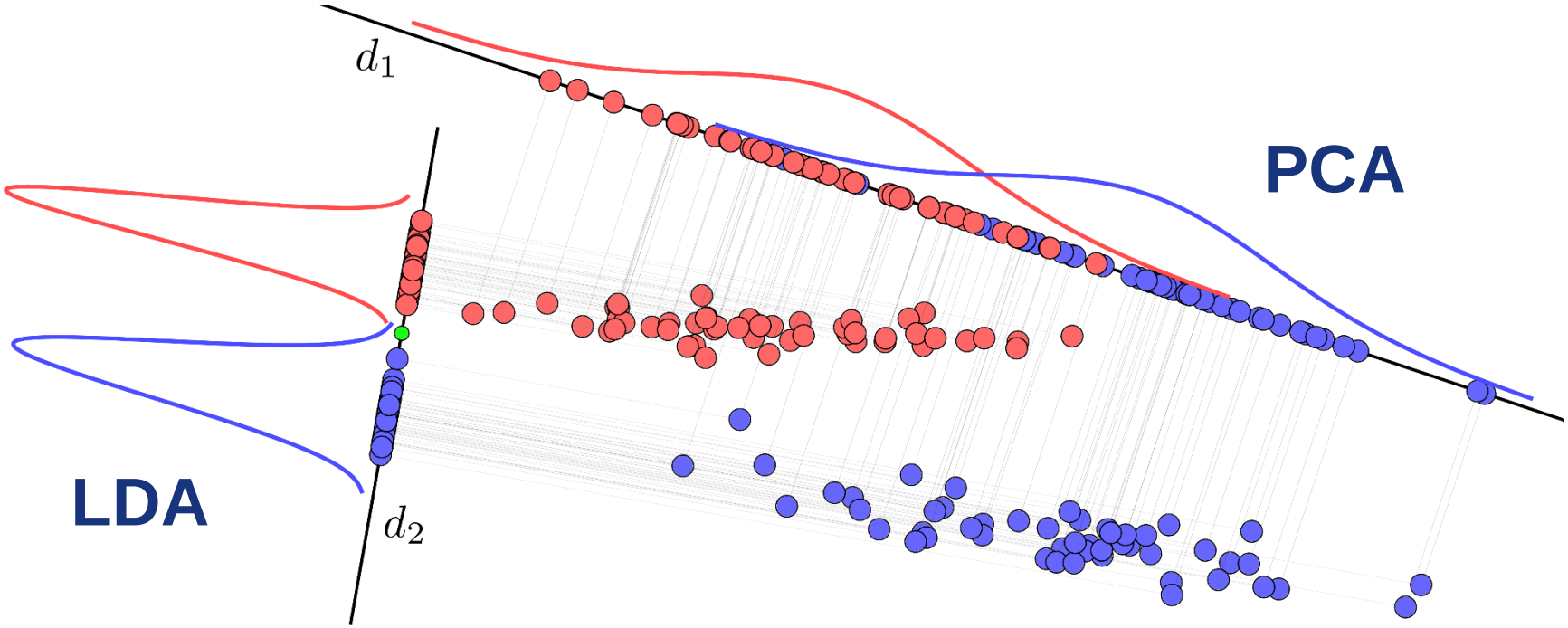

I T F

Phân tích biệt thức tuyến tính Linear Discriminant Analysis (LDA)

Linear Discriminant analysis

- Phân tích biệt thức tuyến tính - Linear Discriminant analysis (LDA) là phương pháp giảm chiều dữ liệu cho bài toán classification.
- LDA có thể được coi là một phương pháp giảm chiều dữ liệu (dimensionality reduction), và cũng có thể được coi là một phương pháp phân lớp (classification), và cũng có thể được áp dụng đồng thời cho cả hai, tức giảm chiều dữ liệu sao cho việc phân lớp hiệu quả nhất.

So sánh ý tưởng giữa PCA và LDA



Phân tích biệt thức tuyến tính

- Cho tập dữ liệu

$$T = \left\{ \boldsymbol{x}_i, y_i \,\middle|\, \boldsymbol{x}_i \in \boldsymbol{R}^d, y_i \in [1, \ldots, l] \right\}_{i=1}^{n}$$

- Với:
  - $\boldsymbol{x}_i$, y_i lần lượt là sample và nhãn của dữ liệu thứ i
  - d là số chiều dữ liệu, l là số loại nhãn của dữ liệu
  - n là số lượng sample
- LDA tìm trục tọa độ **w** mới sao cho khi chiếu dữ liệu lên nó ($\mathbf{w}^T\mathbf{x}_i$), độ ***phân ly*** lớn nhất.

Độ phân ly

- Độ phân ly của dữ liệu huấn luyện khi chiếu trên trục vector $\boldsymbol{w}$

$$\frac{\boldsymbol{w}^{\mathrm{T}} \mathrm{S}_{\mathrm{b}} \boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}} \mathrm{S}_{\mathrm{w}} \boldsymbol{w}}$$

- Trong đó :
  - S_{b} là ma trận phân tán liên hợp (within class scatter matrix)
  - S_{w} là ma trận phân tán nội lớp (within-class scatter matrix):

Ma trận phân tán liên hợp

- Ma trận phân tán liên hợp S_b (within class scatter matrix):

$$S_b = \sum_{c=1}^{l} \boldsymbol{n_c} (\boldsymbol{\mu_c} - \boldsymbol{\mu})(\boldsymbol{\mu_c} - \boldsymbol{\mu})^T \in \boldsymbol{R}^{d \times d}$$

- Trong đó:
  - $\mu_c = \frac{1}{n_c} \sum_{y_i = c} \boldsymbol{x}_i$ là trọng tâm của lớp c
  - $\mu = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i$ là trọng tâm của toàn bộ dữ liệu
  - n_c là số lượng dữ liệu thuộc lớp c

Ma trận phân tán nội lớp

- Ma trận phân tán nội lớp S_w (within-class scatter matrix):

$$S_w = \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{\mu}_{y_i})(\boldsymbol{x}_i - \boldsymbol{\mu}_{y_i})^T \in \boldsymbol{R}^{d \times d}$$

- Trong đó:
  - $\boldsymbol{\mu}_{y_i}$ là là trọng tâm của lớp có nhãn là y_i
  - $\boldsymbol{x}_i$ là vector thể hiện dữ liệu thứ *i*

Xác định vector w

- Vector $\boldsymbol{w}$ được xác định bởi:

$$\boldsymbol{w} = \underset{\boldsymbol{u}}{\operatorname{argmax}} \frac{\boldsymbol{u}^{\mathrm{T}} \mathrm{S}_{\mathrm{b}} \boldsymbol{u}}{\boldsymbol{u}^{\mathrm{T}} \mathrm{S}_{\mathrm{w}} \boldsymbol{u}}$$

- Ta có thể tìm được $\boldsymbol{w}$ bằng bài toán tìm giá trị riêng tổng quát:

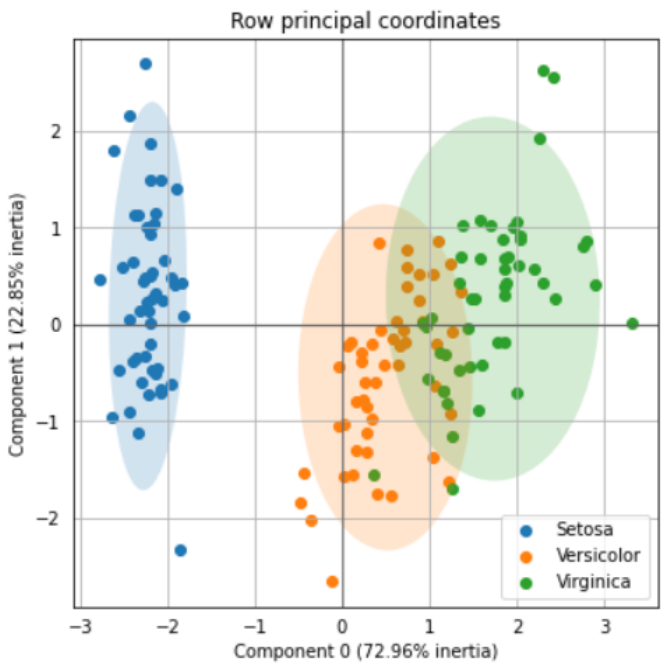
$$\mathrm{S}_{\mathrm{b}} \boldsymbol{w} = \lambda \mathrm{S}_{\mathrm{w}} \boldsymbol{w}$$

LDA với Python

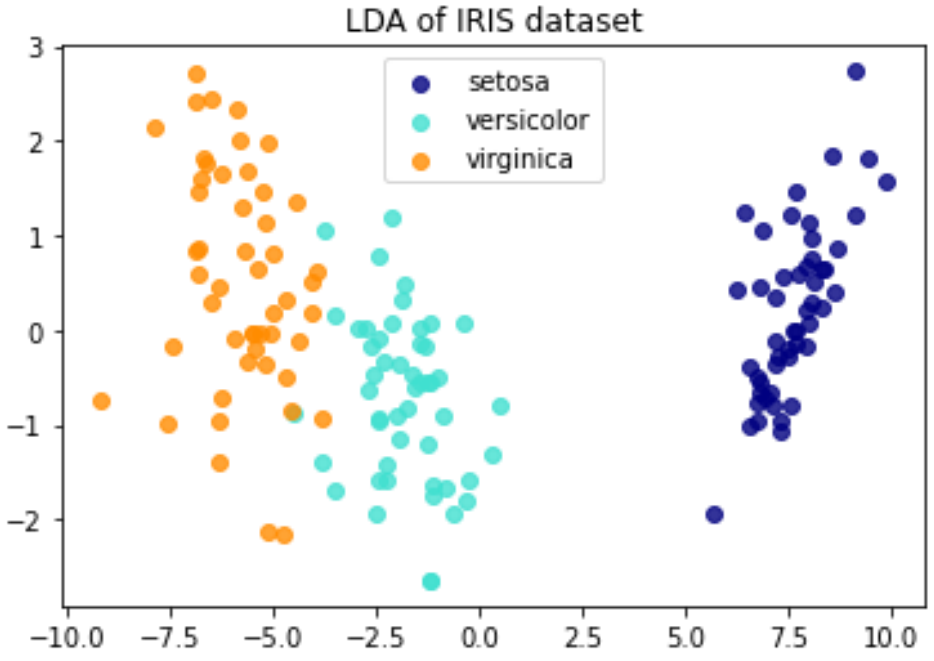
```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

iris = datasets.load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names
lda = LinearDiscriminantAnalysis(n_components=2)
X_r2 = lda.fit(X, y).transform(X)
plt.figure()
colors = ['navy', 'turquoise', 'darkorange']
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=.8, color=co
lor, label=target_name)
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('LDA of IRIS dataset')
plt.show()
```

So sánh PCA với LDA



PCA



LDA

Kết luận

- Xử lý dữ liệu nhiều chiều bằng các phương pháp:
  - Principal component analysis (PCA)
  - Correspondence analysis (CA)
  - Cluster analysis
  - Discriminant analysis
- Tùy thuộc vào mục đích sử dụng mà ta lựa chọn phương pháp cho phù hợp