Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

MATHEMATICS AND STATISTICS FOR DATA SCIENCE

Bài 3: *Principal Component Analysis*

Phòng LT & Mạng

https://csc.edu.vn/lap-trinh-va-csdl/Mathematics-and-Statistics-for-Data-Science_194

2019

Nội dung

Giới thiệu

- Một phương pháp machine learning quan trọng để giảm kích thước được gọi là Phân tích thành phần chính - Principal Component Analysis (PCA).

- Đây là một phương pháp sử dụng các phép toán ma trận đơn giản từ đại số tuyến tính và thống kê để tính toán một phép chiếu của dữ liệu gốc thành ma trận có cùng kích thước hoặc kích thước nhỏ hơn.

Giới thiệu

- Đặc điểm
  - Một trong những phương pháp hữu ích ứng dụng từ đại số tuyến tính
  - Cách không tham số (non-parametric) trích xuất thông tin có ý nghĩa từ các tập dữ liệu khó hiểu
  - Khám phá các cấu trúc ẩn, chiều thấp
  - Những cấu trúc này *trực quan hơn* và thường *dễ hiểu* đối với các chuyên gia.

Giới thiệu

- Tìm kiếm thuộc tính dư thừa
  - Một trong những điều quan trọng mà phân tích thành phần chính có thể làm là thu hẹp sự dư thừa trong tập dữ liệu. Trong biểu hiện đơn giản nhất của nó, sự dư thừa xảy ra khi hai biến tương quan.
  - Pearson correlation coefficient (hệ số tương quan Pearson) là một số nằm giữa -1 và 1. Các hệ số gần 0 chỉ ra hai biến là độc lập, trong khi các hệ số gần -1 hoặc 1 chỉ ra rằng hai biến có liên quan tuyến tính.

Nội dung

1. Giới thiệu
2. **Linear Algebra trong PCA**
3. Tính toán PCA
4. PCA với sklearn

Linear Algebra trong PCA

- Principal Component Analysis (PCA): là một phương pháp để giảm chiều của dữ liệu. Nó có thể được coi là một phương pháp chiếu trong đó dữ liệu với n-columns (features) được chiếu vào một không gian con có n hoặc ít cột hơn, trong khi vẫn giữ được bản chất của dữ liệu gốc.

- Phương pháp PCA có thể được mô tả và thực hiện bằng các công cụ của đại số tuyến tính.

Linear Algebra trong PCA

- PCA là một hoạt động được áp dụng cho một dataset, được biểu thị bằng ma trận A(n × m) A dẫn đến phép chiếu A mà chúng ta sẽ gọi B.

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{pmatrix}$$

$$B = PCA(A)$$

Linear Algebra trong PCA

❑ Các bước thực hiện như sau:

- Bước 1: Tính các giá trị trung bình của mỗi cột

$$M = \text{mean}(A)$$

- Bước 2: Tiếp theo, cần tính Center (ma trận Center C) của các giá trị trong mỗi cột bằng cách trừ đi giá trị cột trung bình.

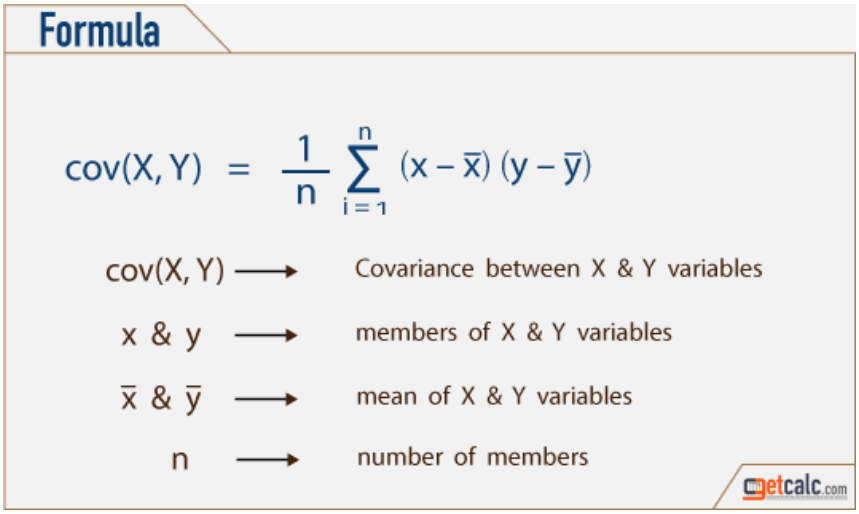
$$C = A - M$$

Linear Algebra trong PCA

- Bước 3: Tính covariance matrix (ma trận hiệp phương sai) của ma trận trung tâm C. Correlation (tương quan) là một phép đo chuẩn hóa của amount (số lượng) và direction (hướng) (dương hoặc âm) mà hai cột thay đổi cùng nhau. Covariance (hiệp phương sai) là một phiên bản tổng quát và không chuẩn hóa của correlation trên nhiều cột. Covariance matrix là phép tính covariance của một ma trận đã cho với điểm số covariance cho mỗi cột với mọi cột khác, kể cả chính nó

$$V = cov(C)$$

Linear Algebra trong PCA



Linear Algebra trong PCA

- Bước 4: Tính eigen-decomposition (phân rã eigen) của covariance matrix V (ma trận hiệp phương sai V) => có một danh sách các trị riêng (eigenvalue) và một danh sách các vector riêng (eigenvector).

$$values, vectors = eig(V)$$

  - Các vector riêng biểu thị các hướng hoặc các thành phần cho không gian con bị giảm B, trong khi các giá trị riêng biểu thị cường độ cho các hướng. Các vector riêng có thể được sắp xếp theo các trị riêng với thứ tự giảm dần để cung cấp rank (thứ hạng) của các thành phần hoặc trục của không gian con mới cho A.

Linear Algebra trong PCA

- Nếu tất cả các giá trị riêng đều có giá trị tương tự, có nghĩa là biểu diễn hiện tại có thể đã được nén và kết quả dự báo có thể ít.
- Nếu có giá trị riêng gần bằng 0, chúng đại diện cho các thành phần hoặc trục của B có thể bị loại bỏ.
- Tổng thành phần là n hoặc số thành phần ít hơn phải được chọn để có không gian con được chọn.
- Lý tưởng nhất là chọn k vector riêng, được gọi là thành phần chính, có các trị riêng (eigenvalue) lớn nhất k.

$$B = select(values, vectors)$$

Linear Algebra trong PCA

- **Phương pháp phân rã ma trận khác có thể được sử dụng như Singular-Value Decomposition - SVD. Như vậy, nhìn chung các giá trị được gọi là giá trị số đơn và vector của không gian con được gọi là các thành phần chính. Sau khi được chọn, dữ liệu có thể được chiếu vào không gian con thông qua phép nhân ma trận:**

$$P = B^T \cdot A$$

  - Trong đó: A là dữ liệu gốc muốn chiếu, B^T là chuyển vị của các thành phần chính được chọn và P là projection (hình chiếu) của A.

- **Đây được gọi là phương pháp hiệp phương sai để tính PCA, mặc dù có nhiều cách khác để tính toán**

Nội dung

1. Giới thiệu
2. Linear Algebra trong PCA
3. **Tính toán PCA**
4. PCA với sklearn

Tính toán PCA


```
from numpy import array
from numpy import mean
from numpy import cov
from numpy.linalg import eig
```

```
# define matrix
A = array([[1, 2],[3, 4],[5, 6]])
print(A)
```

```
[[1 2]
 [3 4]
 [5 6]]
```

```
# column means
M = mean(A.T, axis=1)
M
```

```
array([3., 4.])
```

```
# center columns by subtracting column means
C = A - M
C
```

```
array([[-2., -2.],
       [ 0.,  0.],
       [ 2.,  2.]])
```

```
# calculate covariance matrix of centered matrix
V = cov(C.T)
V
```

```
array([[4., 4.],
       [4., 4.]])
```

```
# factorize covariance matrix
values, vectors = eig(V)
print(vectors)
print(values)
```

```
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
[8. 0.]
```

```
# project data
P = vectors.T.dot(C.T)
print(P.T)
```

```
[[-2.82842712  0.        ]
 [ 0.          0.        ]
 [ 2.82842712  0.        ]]
```

Nội dung

1. Giới thiệu
2. Linear Algebra trong PCA
3. Tính toán PCA
4. PCA với sklearn

PCA với sklearn

- Sử dụng sklearn.decomposition.PCA trong thư viện sklearn

```
from numpy import array
from sklearn.decomposition import PCA
```

```
# define matrix
A = array([[1, 2, 3],[4, 5, 6],[7, 8, 9], [2, 3, 1], [6, 3, 4], [5, 2, 7]])
print(A)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]
 [2 3 1]
 [6 3 4]
 [5 2 7]]
```

```
# create the transform
pca = PCA(2)
# fit transform
pca.fit(A)
```

```
PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

PCA với sklearn

```
# access values and vectors
# components_ : array, shape (n_components, n_features)
# Các trục chính trong không gian feature, biểu thị
# các hướng của phương sai tối đa trong dữ liệu.
# explained_variance_ : array, shape (n_components,)
# Số lượng phương sai được giải thích bởi từng thành phần được chọn.
print(pca.components_)
print(pca.components_.shape)
print(pca.explained_variance_)
print(pca.explained_variance_.shape)
```

```
[[ 0.52432669  0.48166461  0.70219707]
 [ 0.33007983 -0.87513362  0.35381981]]
(2, 3)
[15.30094455  2.36819285]
(2,)
```

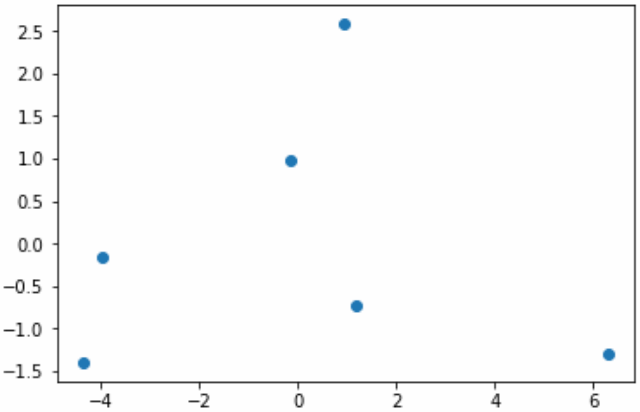
```
# transform data
B = pca.transform(A)
print(B)
```

```
[[-3.94781377 -0.14848076]
 [ 1.17675134 -0.72218272]
 [ 6.30131644 -1.29588469]
 [-4.34621662 -1.40117416]
 [-0.14231865  0.98060456]
 [ 0.95828127  2.58711777]]
```


PCA với sklearn

- Sử dụng sklearn.decomposition.PCA trong thư viện sklearn

```
plt.scatter(B[:,0], B[:,1])
plt.show()
```



B3. PCA

Bổ sung thêm cho bài giảng

2019

Nội dung bổ sung

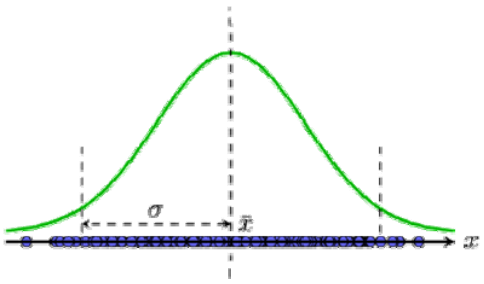
1. **Ma trận hiệp phương sai**
2. **Principal Component Analysis**

1. Ma trận hiệp phương sai

- **Kỳ vọng** (*expectation*)

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

trung bình cộng (*mean*)



- **Phương sai** (*variance*) **và độ lệch chuẩn** (*standard deviation*)

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

trung bình khoảng cách đến kỳ vọng

- σ: độ lệch chuẩn
- phương sai càng NHỎ thì các điểm dữ liệu càng gần kỳ vọng
- phương sai càng LỚN thì các điểm dữ liệu càng phân tán

1. Ma trận hiệp phương sai (tt.)

- **Vectơ cột $x_1, x_2, \ldots, x_n \in R^m$, ma trận $X = (x_1\ x_2 \ldots x_n) \in R_{m,n}$**
  - ma trận trung tâm (*center matrix* ≠ *centering matrix*) $\hat{X} \in R_{m,n}$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{x}_{ij} = (x_{ij} - \bar{x})$$

có thể tính trung bình trên mỗi cột

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij} \Rightarrow \hat{x}_{ij} = (x_{ij} - \bar{x}_j)$$

$$\hat{X} = (X - \bar{x}) = \left((x_1 - \bar{x}) \quad (x_2 - \bar{x}) \quad \cdots \quad (x_n - \bar{x})\right)$$

  - ma trận hiệp phương sai (*covariance matrix*) **của X**

$$V(X) = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^T . (x_i - \bar{x}) = \frac{1}{n}\hat{X}^T . \hat{X} \equiv V$$

1. Ma trận hiệp phương sai (tt.)

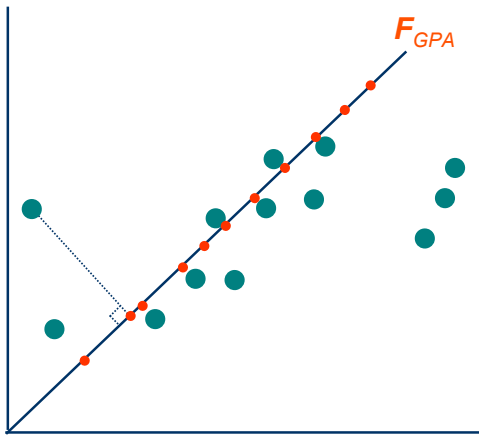
- **Một số tính chất của ma trận hiệp phương sai V**
  - ma trận đối xứng
  - ma trận nửa xác định dương
  - hệ số không âm trên đường chéo: phương sai trên từng chiều
  - hiệp phương sai v_{ij} ($i \neq j$): mối tương quan giữa x_i và x_j
  - nếu V là ma trận đường chéo ⇒ hoàn toàn không tương quan

Nội dung bổ sung

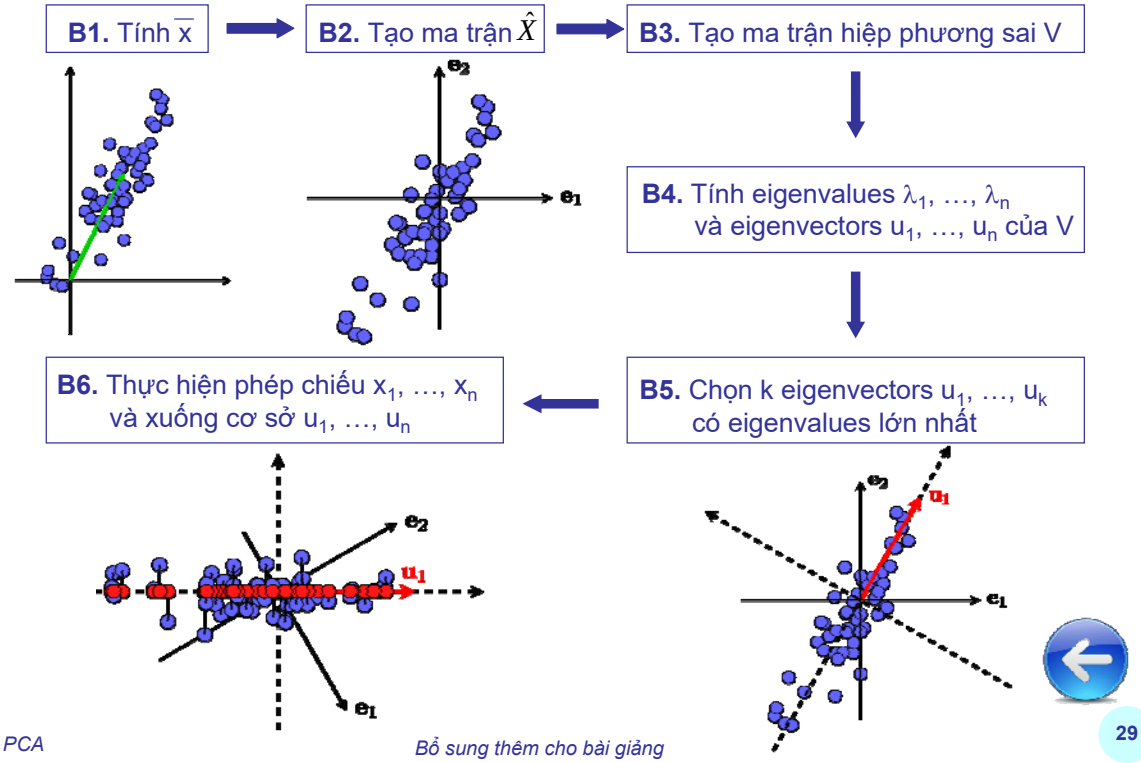
1. Ma trận hiệp phương sai
2. Principal Component Analysis

2. Principal Component Analysis (tt.)

- Tìm <u>không gian đặc trưng</u> mới ***F'*** tạo phân hoạch trên items tốt hơn không gian đặc trưng ban đầu ***F***



2. Principal Component Analysis



2. Principal Component Analysis (tt.)

❑ **Hệ cơ sở – Tọa độ trong không gian vectơ V**

Cơ sở "có thứ tự" B gồm các vectơ độc lập tuyến tính:

$B = \{ u_1, u_2, \ldots, u_n \}$

$\forall v \in V$: $v = \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_n u_n \quad \alpha_i \in \}$

Ma trận cơ sở của không gian V: $B = \begin{pmatrix} u_1^T & u_2^T & \cdots & u_n^T \end{pmatrix}$

Tọa độ của v theo B:

$$[v]_B = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}$$

2. Principal Component Analysis (tt.)

- **Hệ cơ sở – Tọa độ trong không gian vectơ V**

  Cơ sở "có thứ tự" B' gồm các vectơ độc lập tuyến tính:

  $$B = \{ u'_1, u'_2, \ldots, u'_n \}$$

  Ma trận chuyển đổi cơ sở từ B sang B':

  $$(B \to B') = \left([u'_1]_B \quad [u'_2]_B \quad \cdots \quad [u'_n]_B\right)$$

  $(B \to B')$ khả nghịch

  Công thức chuyển đổi tọa độ:

  $$[v]_{B'} = (B \to B')^{-1}[v]_B$$

  $$[v]_B = (B \to B')[v]_{B'}$$

Tài liệu tham khảo

Vũ Hữu Tiệp, *Machine Learning cơ bản*, 2018