教師なし学習

概要

- 教師なし学習（unsupervised learning）
  - 使える情報は $X_1, X_2, \ldots, X_p$ のみ
  - データの構造を分析
    - 教師あり学習のようにモデルを客観的に評価することは難しい
    - 人間がデータを理解するための重要なアプローチ

- 主な手法
  - 主成分分析
  - クラスタリング
    - K-means クラスタリング
    - 階層的クラスタリング

主成分分析

- 主成分分析（Principal Component Analysis, PCA）
  - 次元削減（dimension reduction）のための代表的な手法
  - なるべく情報を失うことなく高次元のデータを低次元のデータに変換

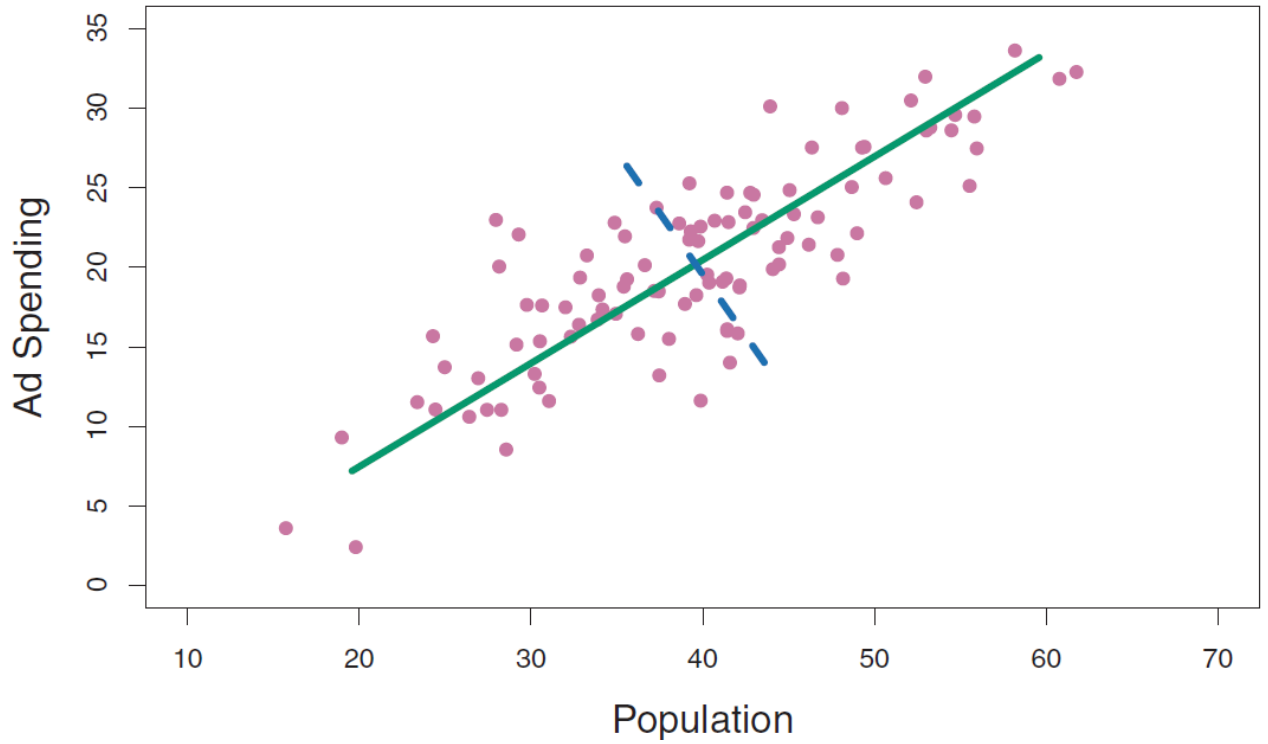
- 主な用途
  - 可視化（visualization）
  - 構造抽出
  - 過学習の回避
  - ノイズ除去

主成分分析

- 例）100都市の人口と広告費

緑：第1主成分方向 $\phi_1 = (0.839, 0.544)^T$

青：第2主成分方向 $\phi_2 = (0.544, -0.839)^T$



主成分分析

- データの前処理
  - 各次元に関して平均をゼロにする: $\sum_{i=1}^{n} x_{ij} = 0$

- 第1主成分
  - 分散が最大になる特徴量の線形結合

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{p1} X_p$$

  - 最適化問題

$$\underset{\phi_{11},\ldots,\phi_{p1}}{\text{maximize}} \sum_{i=1}^{n} z_{i1}^2$$

$$\text{subject to} \sum_{j=1}^{p} \phi_{j1}^2 = 1$$

第1主成分のスコア（score）

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \ldots + \phi_{p1} x_{ip}$$

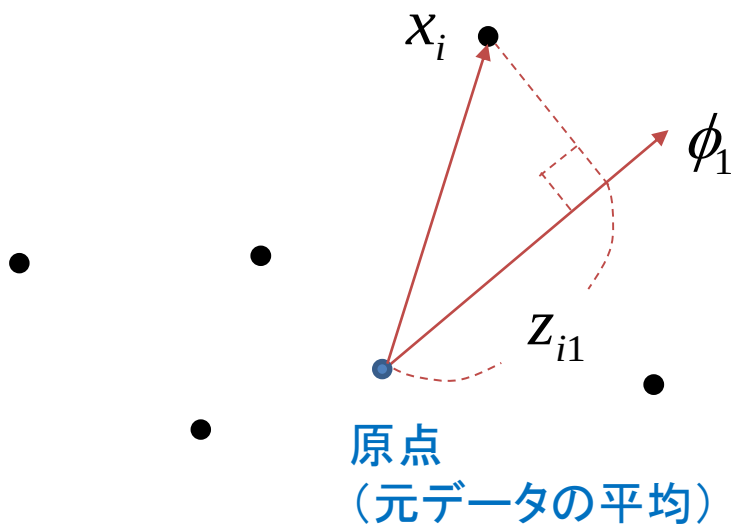
負荷量（loading）ベクトル

$$\phi_1 = \left(\phi_{11}, \phi_{21}, \ldots, \phi_{p1}\right)^T$$

参考

幾何的な解釈

- 正射影ベクトルの長さの2乗和が最大になるような ϕ_1



$$z_{i1} = \phi_1^T x_i$$
$$= \left|\phi_1\right| \left|x_i\right| \cos\theta_i$$
$$= \left|x_i\right| \cos\theta_i$$

$$z_{i1}^2 = \left|x_i\right|^2 \cos^2\theta_i$$
$$= \left|x_i\right|^2 \left(1 - \sin^2\theta_i\right)$$

主成分分析

- 第2主成分
  - Z_1と相関がなく、分散が最大になる特徴量の線形結合

$$Z_2 = \phi_{12} X_1 + \phi_{22} X_2 + \ldots + \phi_{p2} X_p$$

  - 負荷量ベクトル: $\phi_2 = \left(\phi_{12}, \phi_{22}, \cdots, \phi_{p2}\right)^T$
    - Z_1と無相関なので ϕ_1 と直交

- 第3主成分
  - Z_1, Z_2 と相関がなく、分散が最大になる特徴量の線形結合

$$Z_3 = \phi_{13} X_1 + \phi_{23} X_2 + \ldots + \phi_{p3} X_p$$

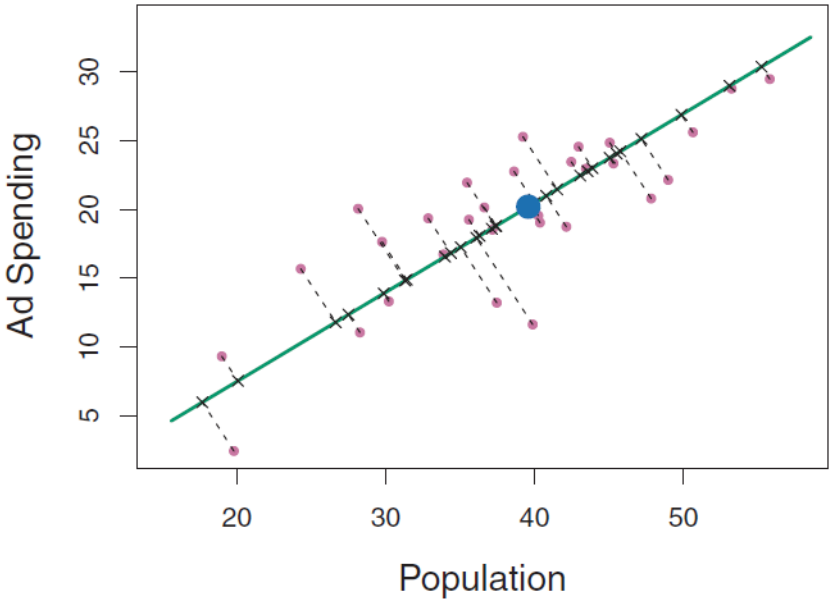
  - …

⇩ 以下同様

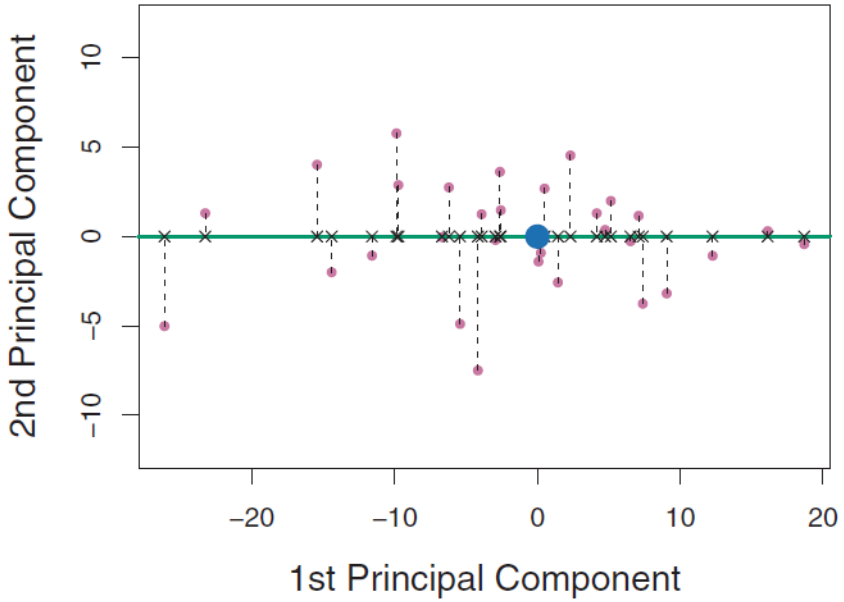
主成分分析

- 例

青点：平均
緑線：第１主成分方向



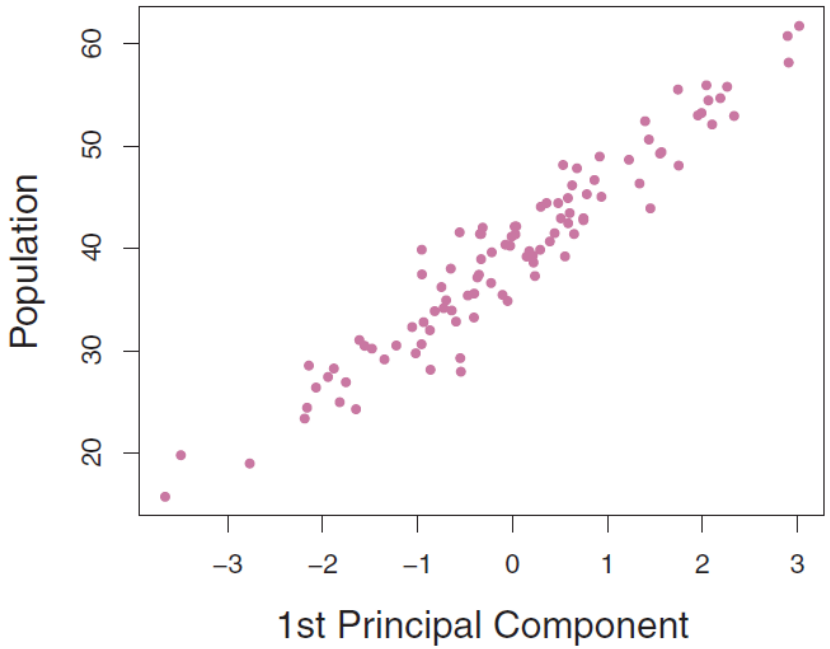
第１主成分と第２主成分

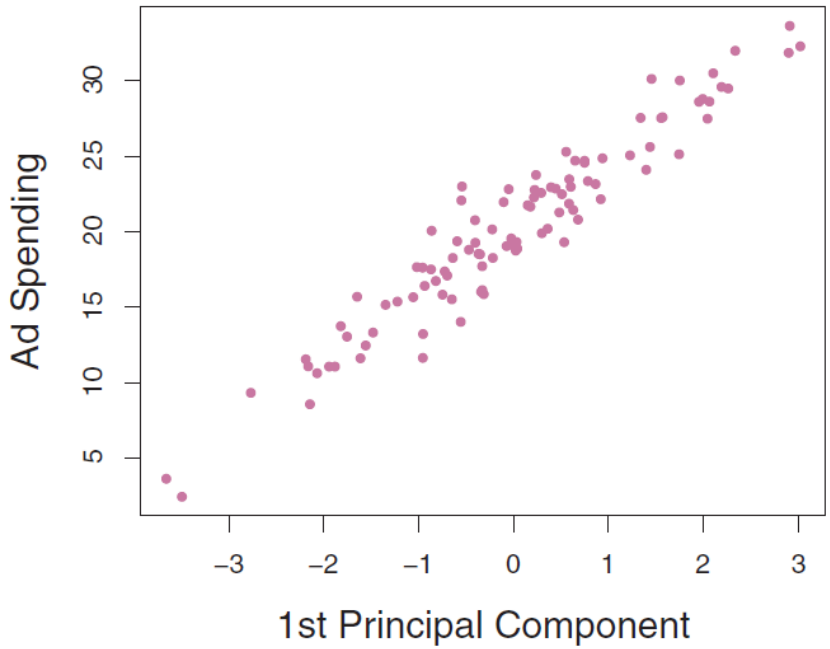


各データ点までの距離の二乗和が
最小になる直線になっている

主成分分析

- 第1主成分と特徴量

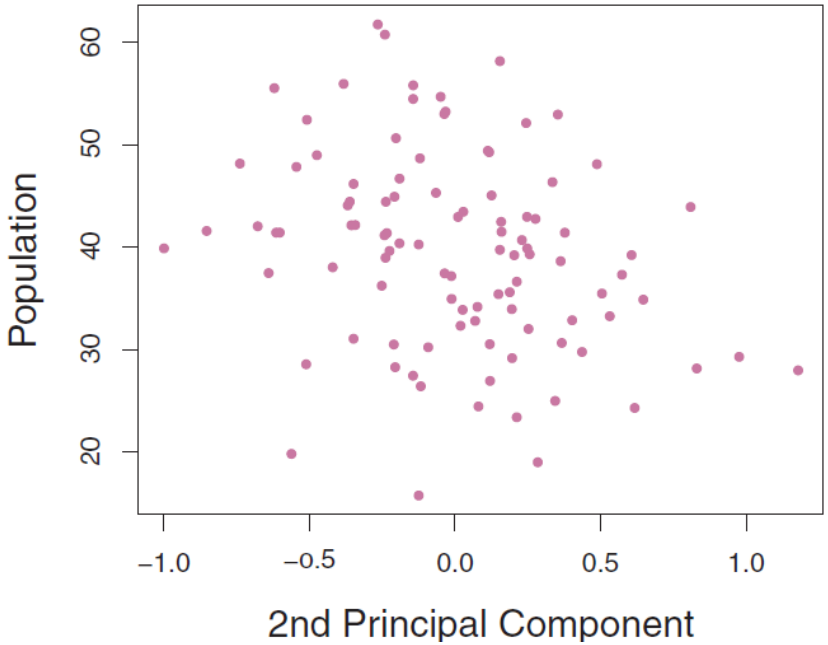


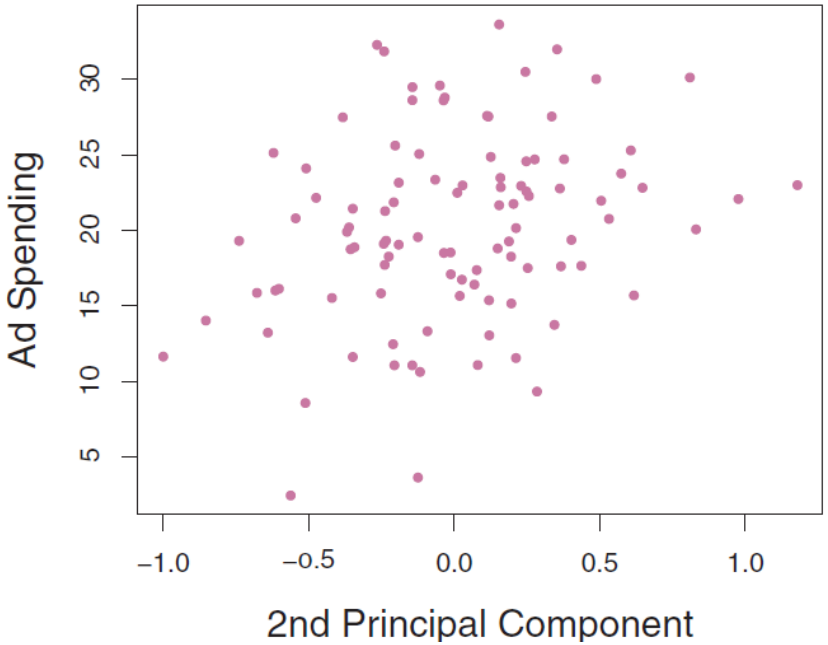


第1主成分が Population と Ad Spending の多くの情報を捉えている

主成分分析

- 第2主成分と特徴量

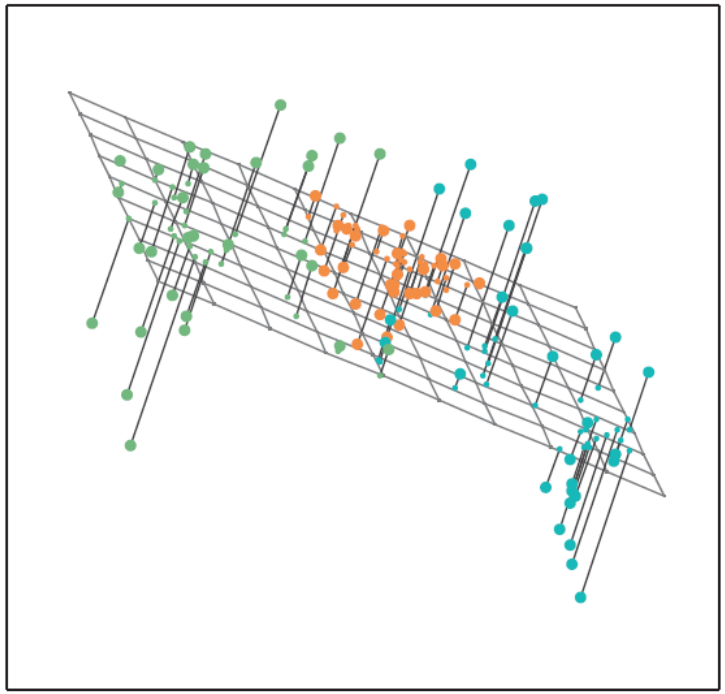


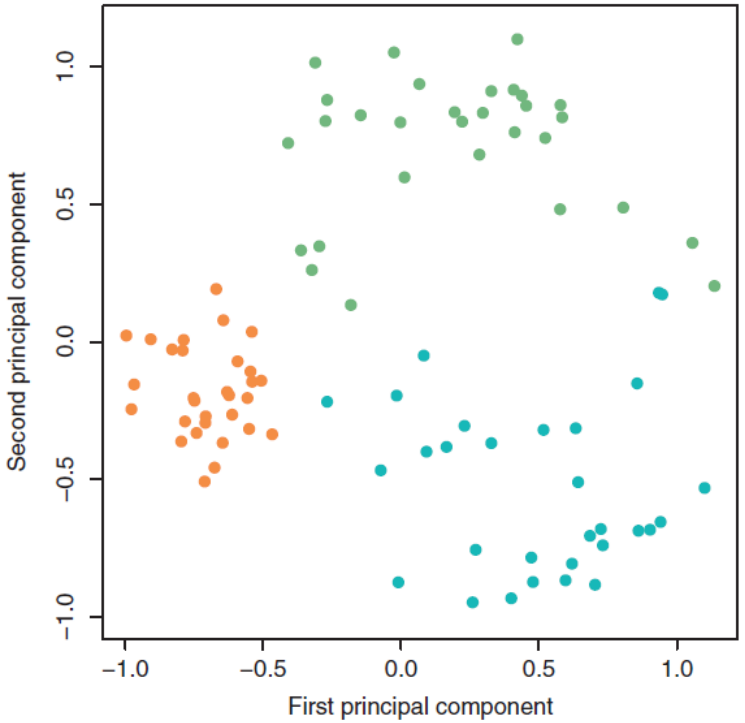


主成分分析

- 3次元データの主成分分析

第1主成分方向と第2主成分方向
で張られる平面





各事例までの距離の2乗和が最小
になる平面になっている

$$x_i \approx z_{i1}\phi_1 + z_{i2}\phi_2$$

一般には

$$x_i \approx \sum_{m=1}^{M} z_{im}\phi_m$$

主成分分析

- 寄与率（Proportion of variance explained, PVE）
  - 元のデータの情報をどれだけ説明できているか
    - 元のデータの分散

$$\sum_{j=1}^{p} \mathrm{Var}\left(X_j\right) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

    - 第 m 主成分によって説明される分散

$$\frac{1}{n} \sum_{i=1}^{n} z_{im}^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2$$

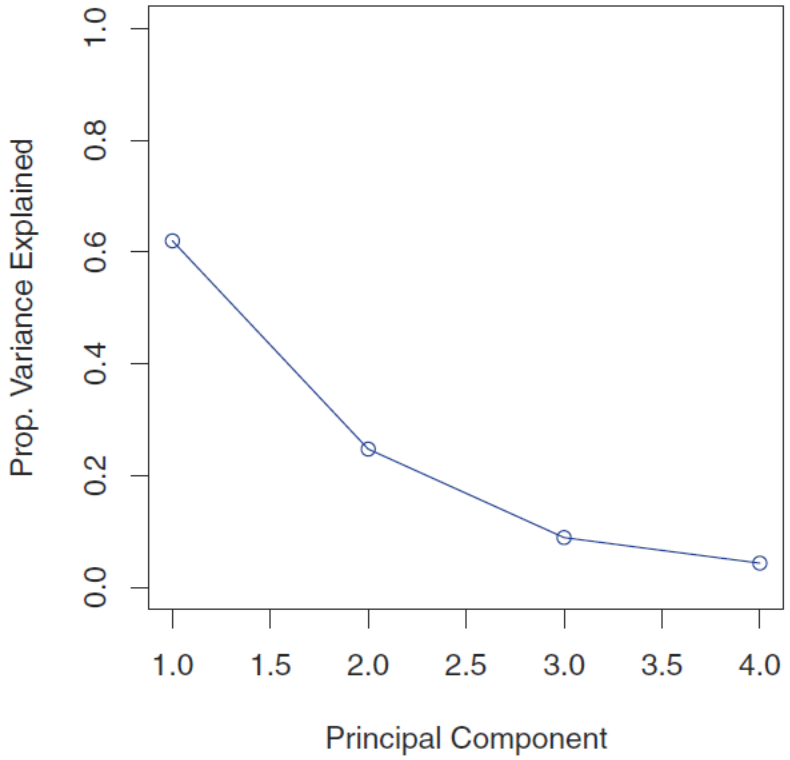
第m主成分のPVE

$$\frac{\sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{jm} x_{ij} \right)^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2}$$

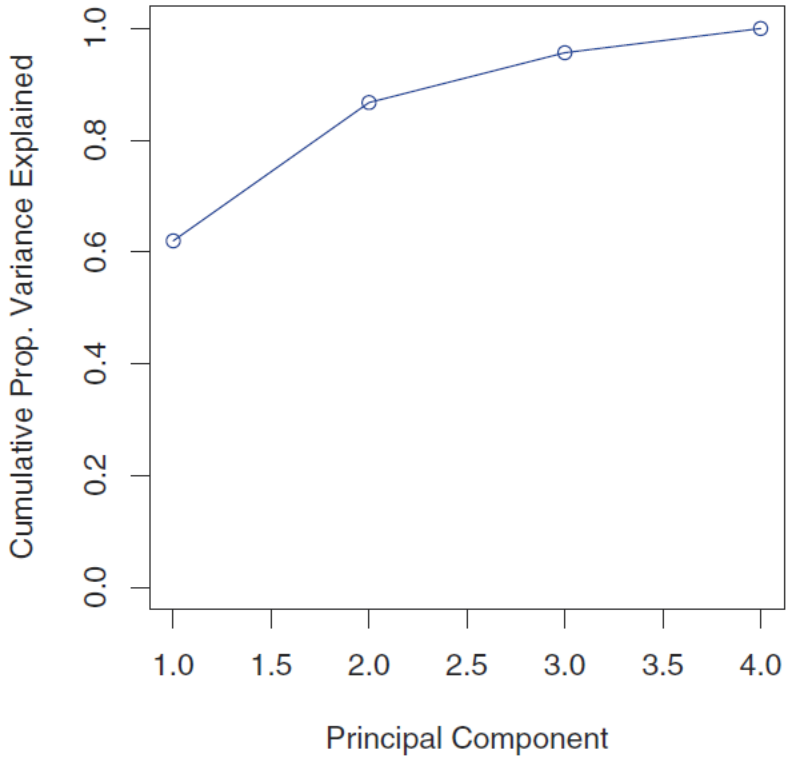
主成分分析

- 例　4次元データ

scree plot



累積寄与率



第2主成分までで87%説明できている

Python実習

- データ準備
  - NCI60 cancer cell line microarray data
    - 64個の細胞株（cell line）の遺伝子発現データ
  - ダウンロード
    - https://www.logos.t.u-tokyo.ac.jp/~tsuruoka/lecture/sml/NCI60_data.csv
    - https://www.logos.t.u-tokyo.ac.jp/~tsuruoka/lecture/sml/NCI60_labs.csv

```
>>> import pandas as pd
>>> import numpy as np
>>> import os
>>> os.chdir(os.path.expanduser("~"))
>>> nc_data = pd.read_csv('NCI60_data.csv', index_col = 0)
>>> nc_lab = pd.read_csv('NCI60_labs.csv', index_col = 0)
>>> nc_lab.labs.value_counts()
>>> from sklearn.preprocessing import scale
>>> nc_data_scaled = pd.DataFrame(scale(nc_data), columns =
nc_data.columns)
```

Adapted from http://pmlwe.blogspot.com/2017/09/1061-pca-on-nci60-data.html

Python実習

- 主成分分析
  - PCA()

```
>>> from sklearn.decomposition import PCA
>>> pca = PCA()
>>> pca.fit(nc_data_scaled)
>>> nc_data_pca = pca.fit_transform(nc_data_scaled)
```

- プロット

```
>>> import matplotlib.pyplot as plt
>>> plt.figure()
>>> plt.scatter(nc_data_pca[:, 0], nc_data_pca[:, 1], c =
pd.factorize(nc_lab.labs)[0], cmap=plt.cm.Dark2)
>>> plt.xlabel('Z1')
>>> plt.ylabel('Z2')
>>> plt.show()
```

Python実習

- PVE (Proportion of variance explained)

```
>>> pca.explained_variance_ratio_[:5]
>>> pca.explained_variance_ratio_[:5].sum()
>>> plt.figure()
>>> plt.plot(pca.explained_variance_ratio_, '-o')
>>> plt.xlabel('Pricipal Components')
>>> plt.ylabel('Explained Variance')
>>> plt.figure()
>>> plt.plot(np.cumsum(pca.explained_variance_ratio_), '-o')
>>> plt.xlabel('Pricipal Components')
>>> plt.ylabel('Cumulative Explained Variance')
>>> plt.show()
```

クラスタリング

- クラスタリング（clustering）
  - データをサブグループ（クラスタ）に分ける
    - 類似の事例（距離が近い事例）が同一クラスタに属するように分割
    - 事例間の距離を定義する必要
      - ユークリッド距離など

- 主な用途
  - データ分析
  - 半教師あり学習

K-Meansクラスタリング

- *K*-meansクラスタリング（*K*-means clustering）
  - データを異なる *K* 個のクラスタに分ける
  - $C_1, C_2, \ldots, C_K$: 各クラスタに属する事例のインデックスの集合

$$C_1 \cup C_2 \cup \cdots \cup C_K = \{1, \ldots, n\}$$

$$C_k \cap C_{k'} = \{\} \quad \text{for all } k \neq k'$$

  - 最適化問題

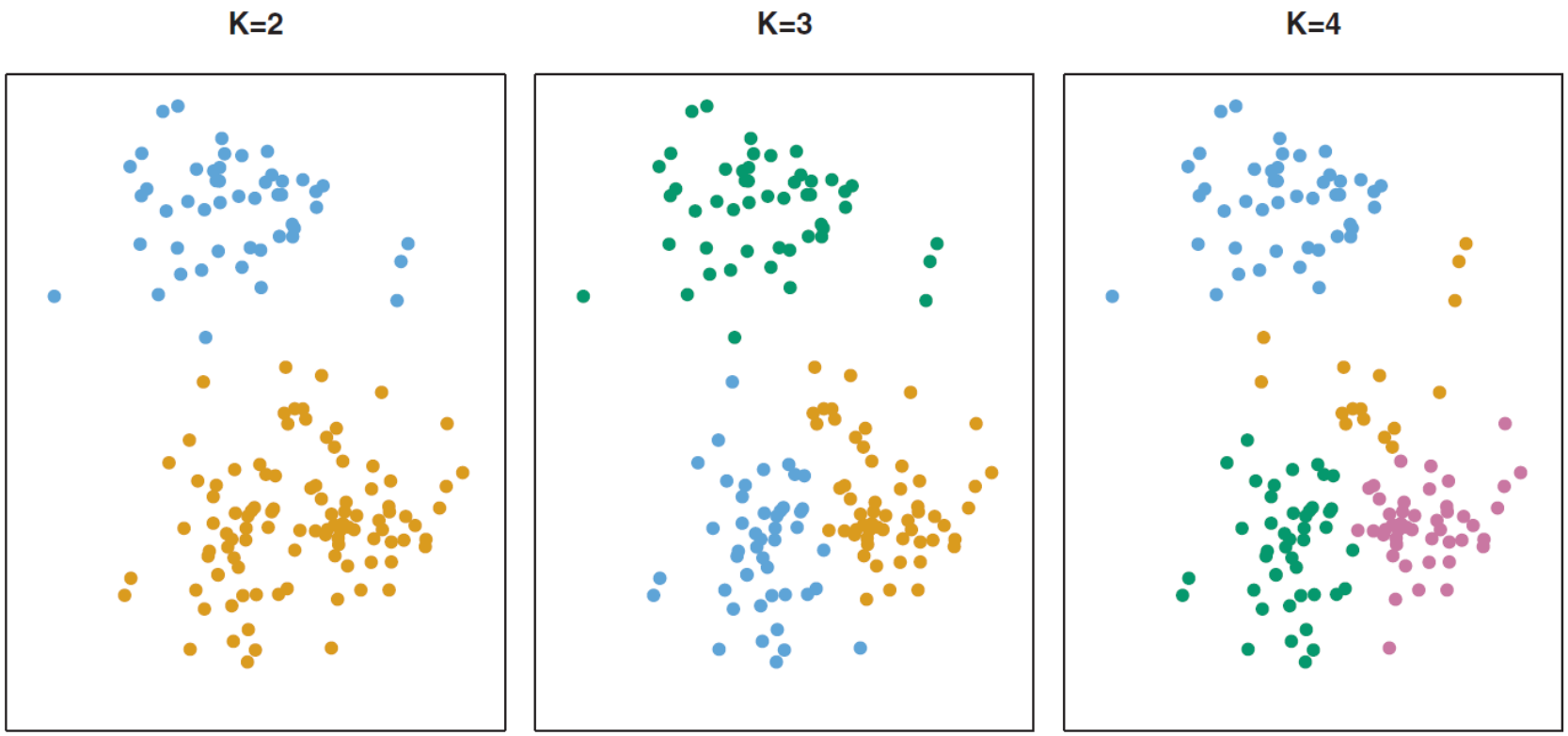
$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} W(C_k) \right\}$$

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$$

各クラスタ内の広がりが最小になるように分けたい
（厳密に解くのは計算量的に無理）

K-Meansクラスタリング

- クラスタ数 *K* による違い



K-Meansクラスタリング

- 手順
  - Step 1: 各事例をランダムにクラスタに割り当てる
  - 収束するまで以下を繰り返す
    - Step 2a: 各クラスタの重心を計算

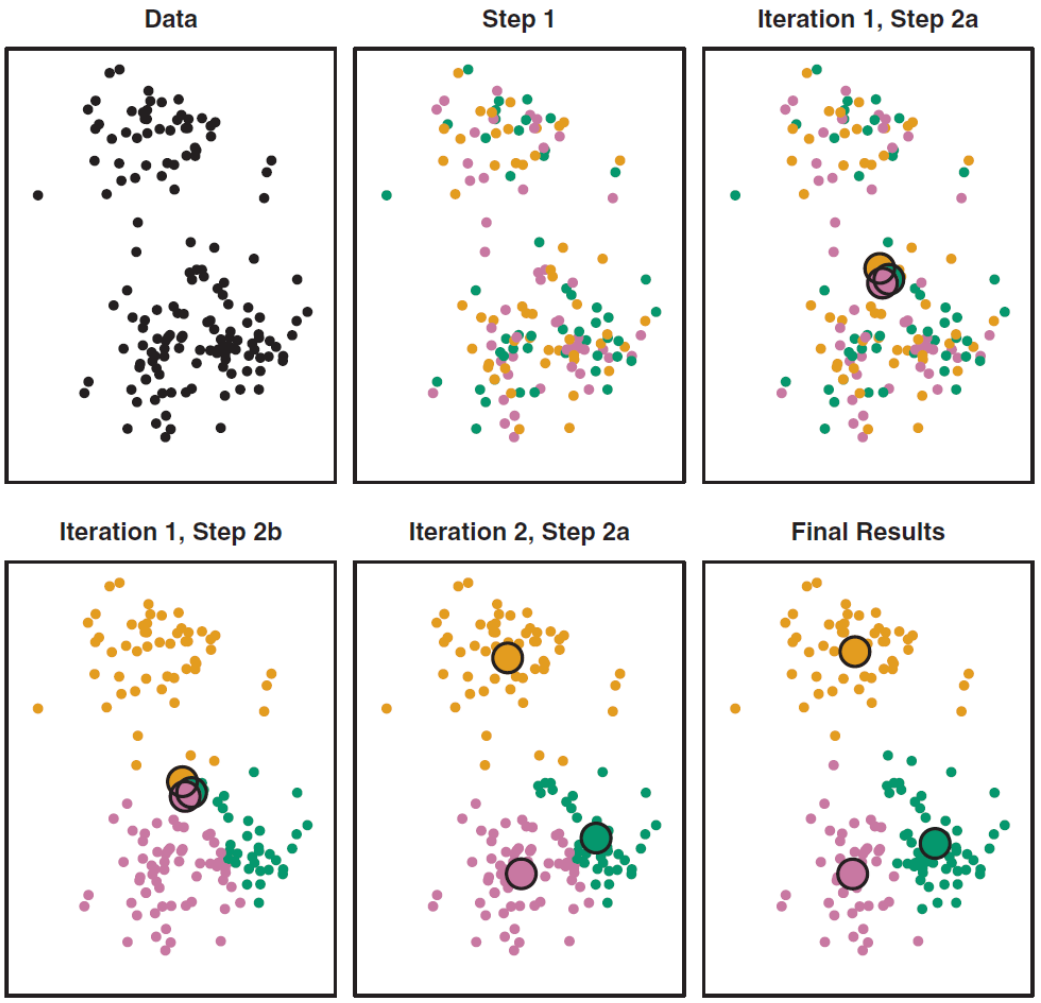
$$\bar{x}_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i$$

    - Step 2b: 各事例を重心が最も近いクラスタに割り当てる

- 注意点
  - 得られるのは局所最適解（local optimum）
  - 初期値を変えて複数回実行し、もっとも評価値が良い解を選ぶ

K-Meansクラスタリング

- 例



K-Meansクラスタリング

- 初期値による違い

Python実習

- データ準備
  - 人工データ
    - 中心 (0,0) の正規分布: 25個
    - 中心 (3,-4) の正規分布: 25個

```
>>> np.random.seed(123)
>>> X = np.random.randn(50,2)
>>> X[0:25, 0] = X[0:25, 0] + 3
>>> X[0:25, 1] = X[0:25, 1] - 4
>>> f, ax = plt.subplots(figsize=(6, 5))
>>> ax.scatter(X[:,0], X[:,1], s=50)
>>> ax.set_xlabel('X0')
>>> ax.set_ylabel('X1')
>>> plt.show()
```

Adapted from http://www.science.smith.edu/~jcrouser/SDS293/labs/lab16-py.html

Python実習

- K-means クラスタリング
  - KMeans()
    - K-means クラスタリングを実行
    - n_clusters: クラスタ数
    - n_init:異なる初期値で n_init 回実行し、最も良い結果を選ぶ

```
>>> from sklearn.cluster import KMeans
>>> kmeans = KMeans(n_clusters = 2, n_init = 20).fit(X)
>>> kmeans.labels_
>>> kmeans.inertia_                    ← クラスタ内の点間の距離の2乗和
```

- プロット

```
>>> plt.figure(figsize=(6,5))
>>> plt.scatter(X[:,0], X[:,1], s = 50, c = kmeans.labels_, cmap = plt.cm.bwr)
>>> plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
marker = '*', s = 150, color = 'cyan', label = 'Centers')
>>> plt.legend(loc = 'best')
>>> plt.xlabel('X0')
>>> plt.ylabel('X1')
>>> plt.show()
```

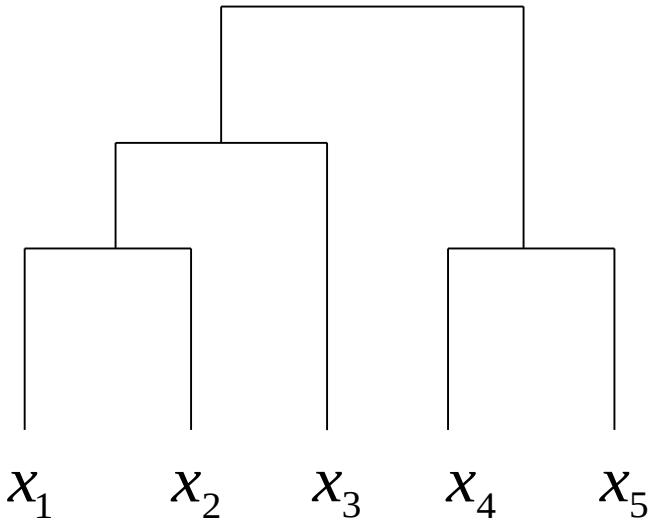
Python実習

- K-means クラスタリング
  - 通常、正しいクラスタ数は未知
  - クラスタ数を3にした場合

```
>>> kmeans = KMeans(n_clusters = 3, n_init = 20).fit(X)
>>> kmeans.inertia_
>>> plt.figure(figsize=(6,5))
>>> plt.scatter(X[:,0], X[:,1], s = 50, c = kmeans.labels_, cmap=plt.cm.prism)
>>> plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
marker = '*', s = 150, color = 'cyan', label = 'Centers')
>>> plt.legend(loc = 'best')
>>> plt.xlabel('X0')
>>> plt.ylabel('X1')
>>> plt.show()
```

階層的クラスタリング

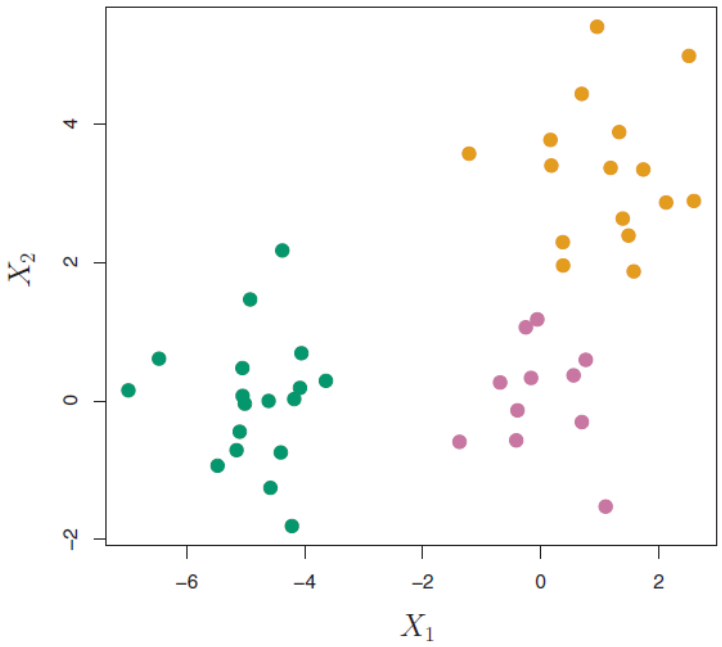
- 階層的クラスタリング（hierarchical clustering）
  - 事例をボトムアップにまとめあげていく
  - クラスタ数をあらかじめ決めておく必要はない
  - 結果が樹状図（dendrogram）として得られる



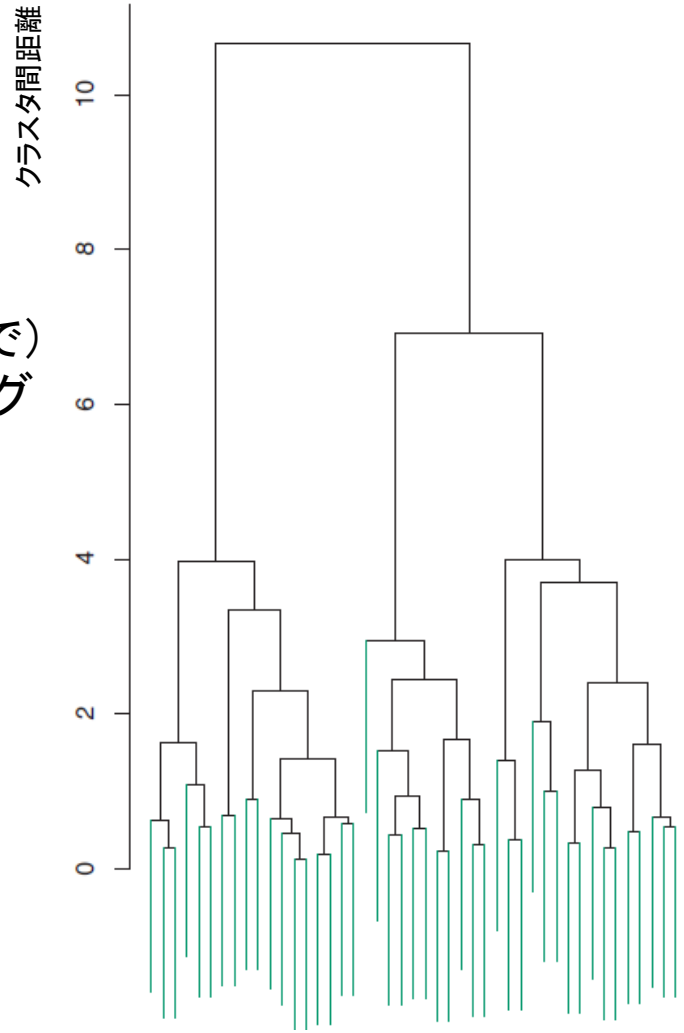
階層的クラスタリング

- 例

3つのクラスからなるデータ



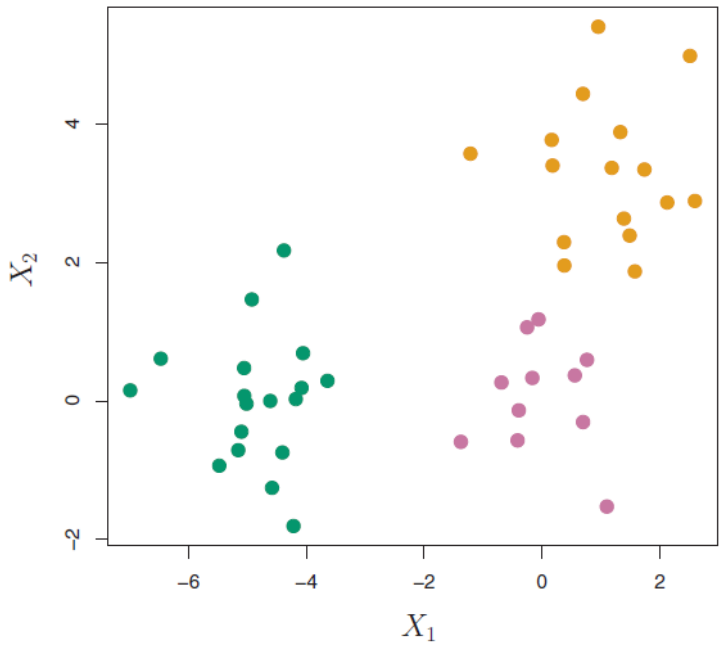
（クラスの情報なしで）
階層的クラスタリング

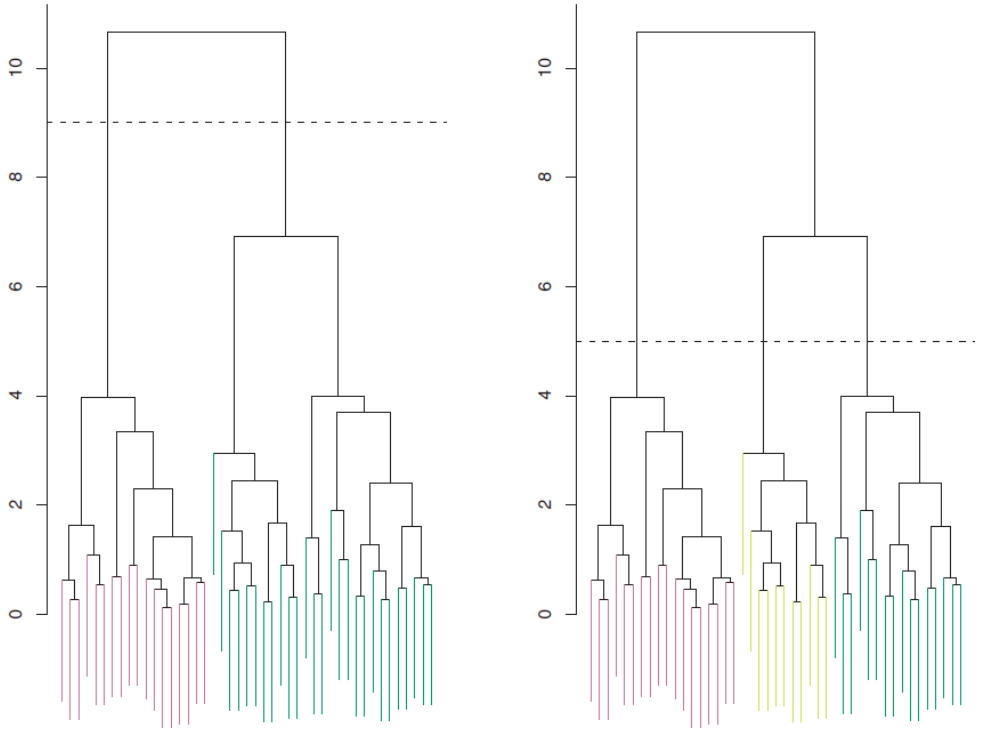


階層的クラスタリング

- 例

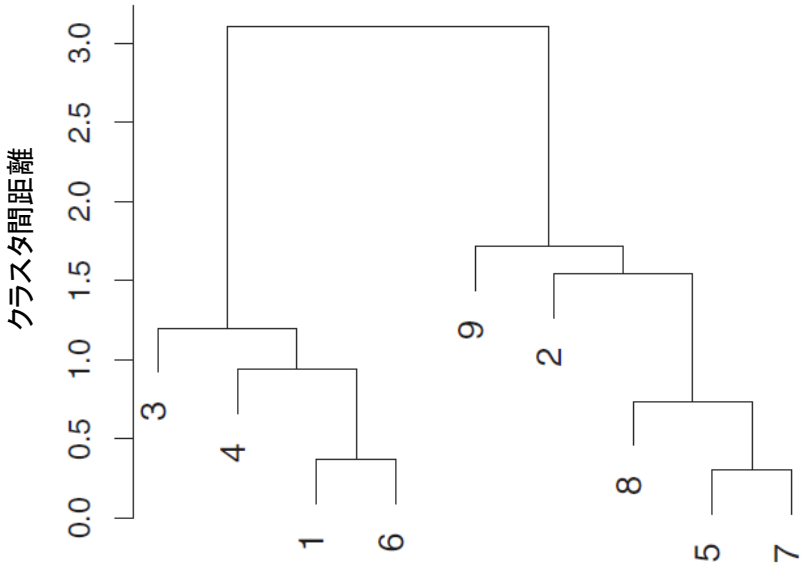
距離の閾値とクラスタリング結果

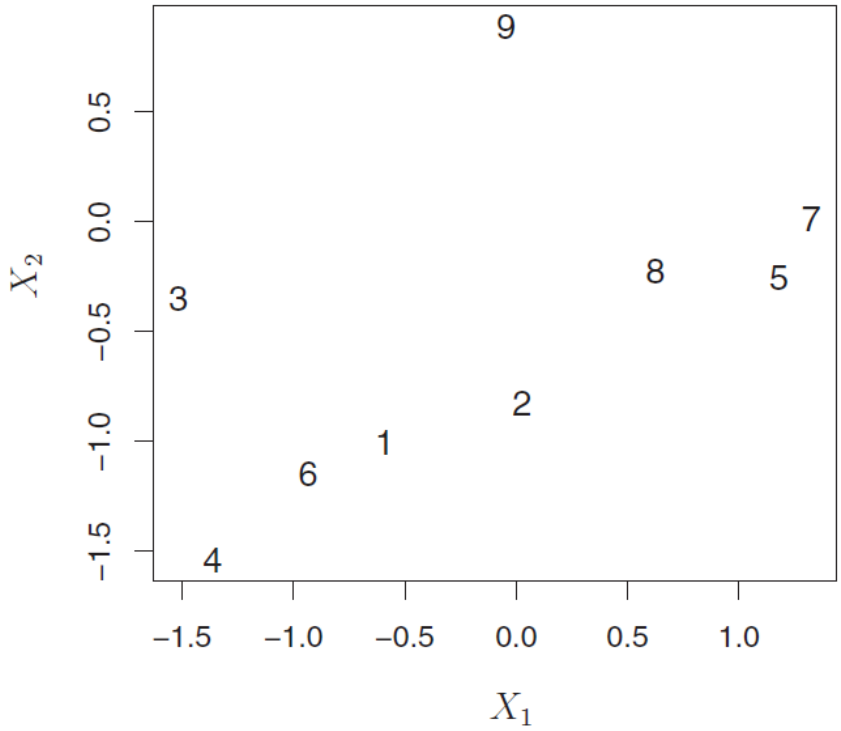




階層的クラスタリング

- 例





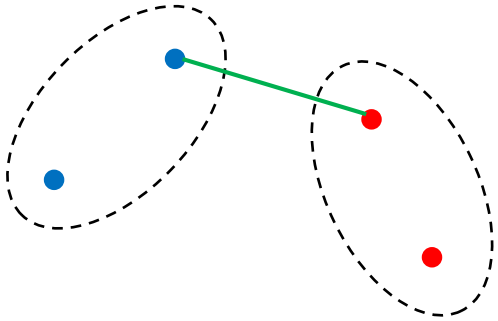
事例2と事例9が離れていることに注意

階層的クラスタリング

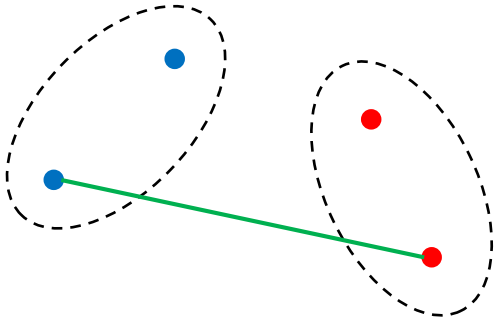
- 手順
  - 各事例を単一の事例から成るクラスタとする
  - For $i = n$ to 2
    - 距離が最も近いクラスタのペアを探し、それらを統合
    - クラスタ間距離を更新

- クラスタ間の距離を定義する方法
  - 最長距離法（complete-linkage）: 最も遠い事例間の距離
  - 最短距離法（single-linkage）: 最も近い事例間の距離
  - 群平均法（average-linkage）: 事例間距離の平均
  - 重心法（centroid-linkage）: 重心間の距離

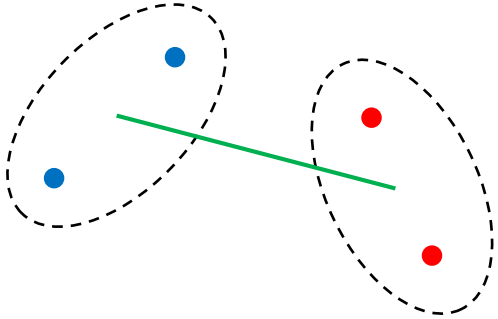
クラスタ間の距離の定義



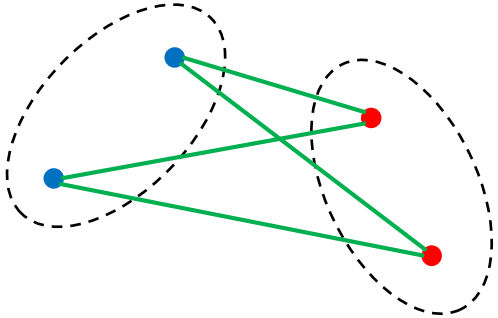
最短距離法



最長距離法



重心法

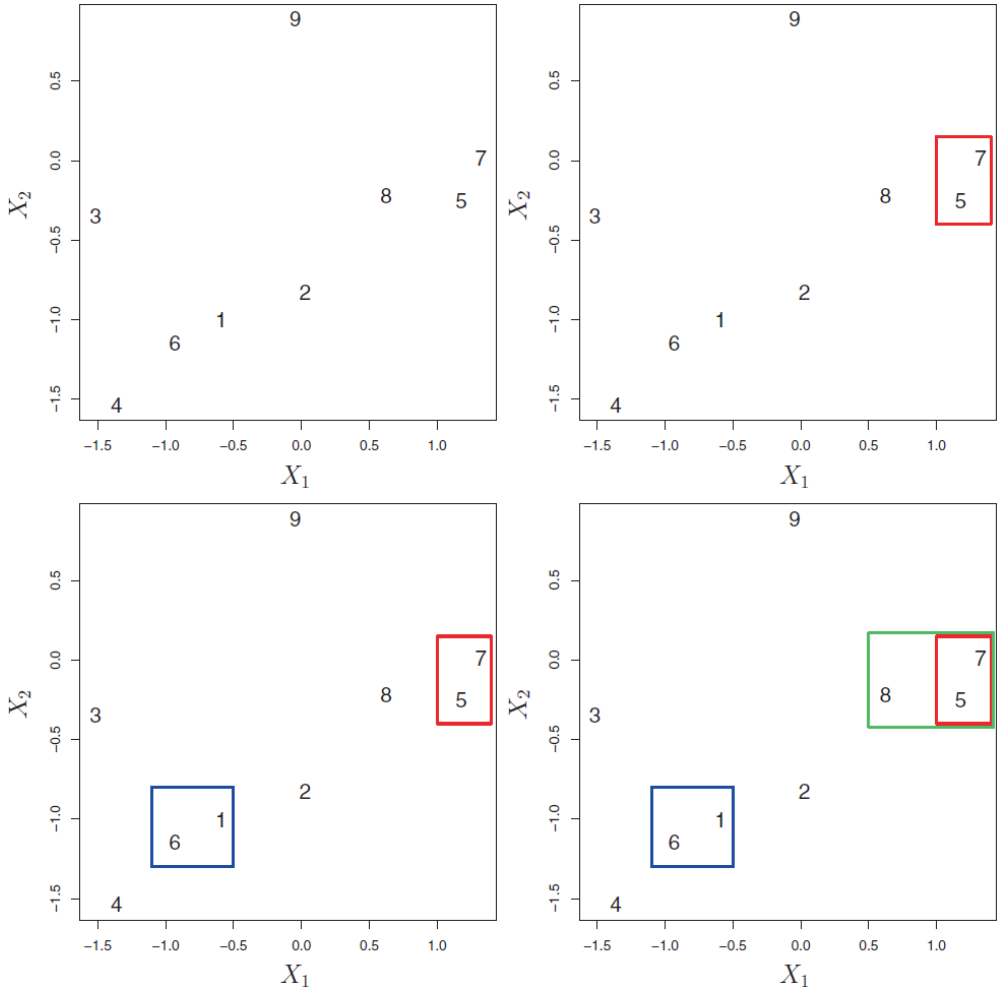


群平均法

階層的クラスタリング

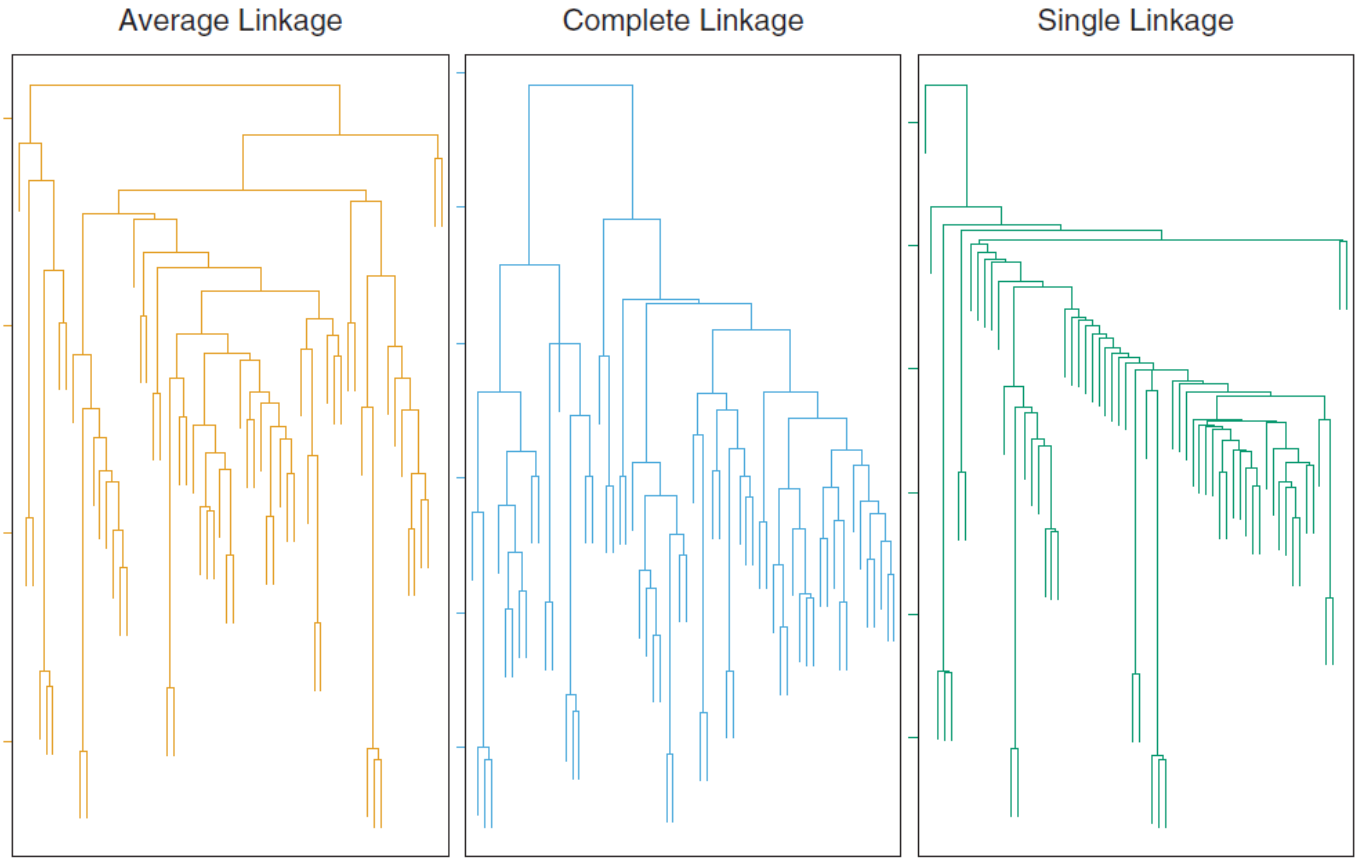
- 例

  最長距離法



階層的クラスタリング

- クラスタ間距離の定義による違い



平均法と最長距離法の方が最短距離法よりもバランスの取れた樹状図になりやすい

Python実習

- 階層的クラスタリング
  - `linkage()`

```
>>> from scipy.cluster.hierarchy import linkage
>>> hc_complete = linkage(X, "complete")
>>> hc_average = linkage(X, "average")
>>> hc_single = linkage(X, "single")
```

Python実習

- プロット

```
>>> from scipy.cluster.hierarchy import dendrogram
>>> plt.figure(figsize=(25, 10))
>>> plt.title('Hierarchical Clustering Dendrogram')
>>> plt.xlabel('sample index')
>>> plt.ylabel('distance')
>>> dendrogram(hc_complete, leaf_rotation=90, leaf_font_size=8)
>>> plt.show()
```

- クラスタリング
  - `cut_tree()`
    - ある高さでカットしてクラスタに分割
    - オプションとしてクラスタ数を指定

```
>>> from scipy.cluster.hierarchy import cut_tree
>>> cut_tree(hc_complete, n_clusters = 2).T
```

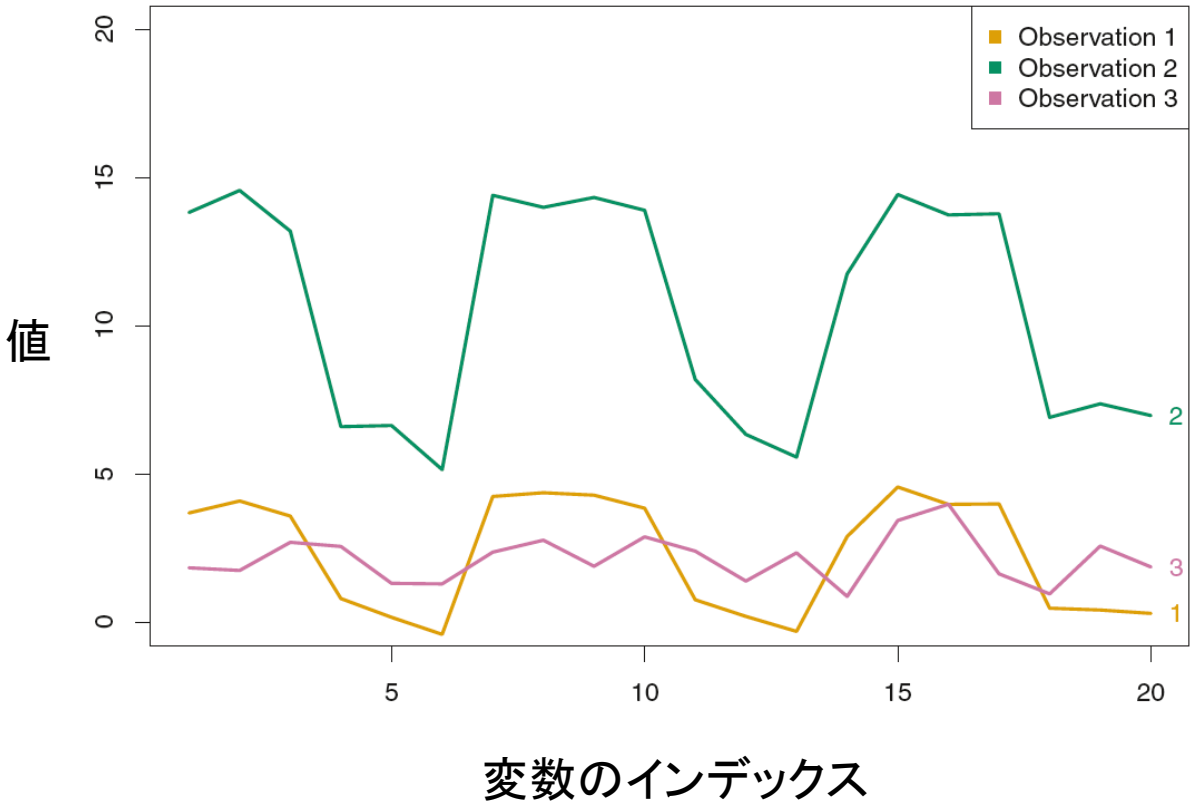
事例間の距離の定義

- 事例間の距離の定義
  - クラスタリングの結果に大きな影響
  - これまで用いてきたのはデータ点（事例）間のユークリッド距離
  - タスクによってはより適切な尺度があることも

- ユークリッド距離以外の例
  - 相関に基づく距離（correlation-based distance）
    - 例）顧客を購買履歴によってクラスタリング

相関に基づく距離

- 例



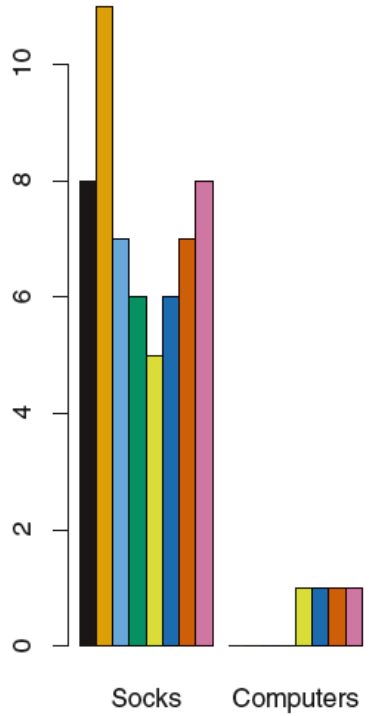
  - 事例1と事例3はユークリッド距離は近い
  - 事例1と事例2は相関に基づく距離は近い

スケーリング

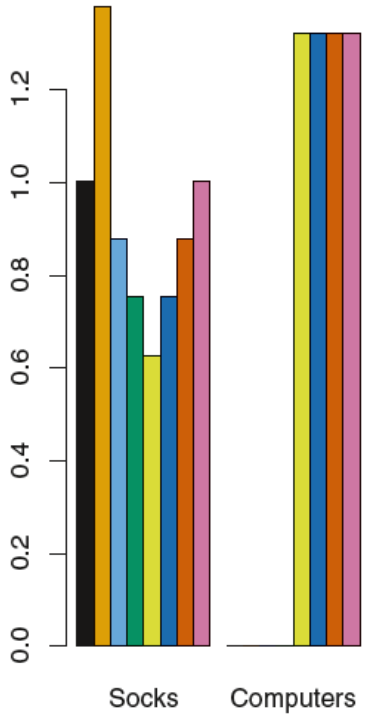
- 各変数のとりうる値が大きく違う場合
  - 距離尺度としてユークリッド距離を用いた場合、分散の大きな変数がクラスタリングに支配的に影響
- 対策
  - 各変数の分散が1になるように値を正規化

スケーリング

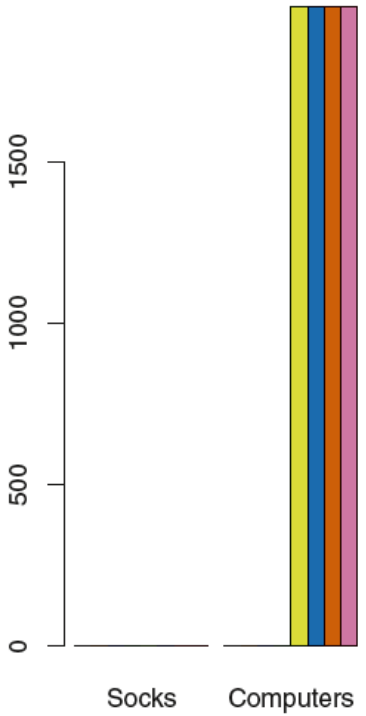
- 例）顧客８人の購買履歴



購入した個数



分散を1に正規化



購入金額