パターン認識と学習
統計的パターン認識（2）

管理工学科
篠沢佳久

資料の内容

- 統計的パターン認識(2)
  - 最尤推定
    - 混合正規分布
    - EMアルゴリズム

- 実習①(単純ベイズ決定則の例題)
- 実習②(混合正規分布の例題)

統計的パターン認識①

- 統計的パターン認識
  - 未知のパターンから特徴 x を求め，特徴の出現確率を利用し，どのクラスに属するのかを求めたい

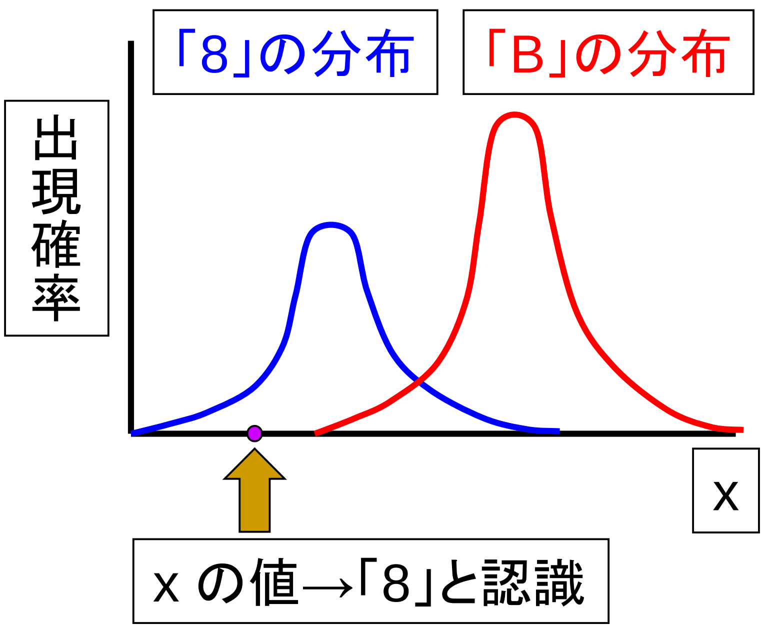
① 尤度を用いた認識（最尤法）

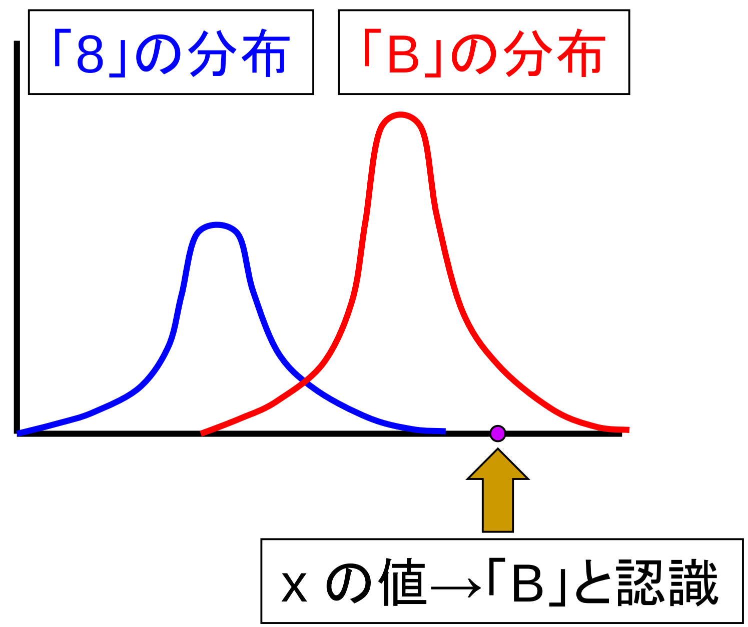
② ベイズ決定則

統計的パターン認識②

p(x|8) > p(x|B) の場合
→「8」と認識

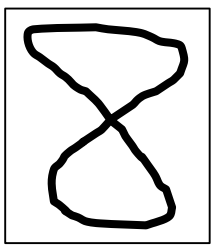
p(x|8) < p(x|B) の場合
→「B」と認識





統計的パターン認識③

- p(8|**x**), p(B|**x**)
  - 特徴ベクトル **x** を観測した後の生起確率（事後確率）



特徴ベクトル **x**

p(8|**x**), p(B|**x**) を求める

p(8|**x**) > p(B|**x**) の場合
→「8」と認識

p(8|**x**) < p(B|**x**) の場合
→「B」と認識

事後確率が最大になるクラスを結果とする（ベイズ決定則）

統計的パターン認識④

母集団

同時確率

$P(\mathbf{x},\omega)$

クラスω

事前確率

$P(\omega)$

$P(\mathbf{x},\omega)$
$=P(\mathbf{x}|\omega)P(\omega)$

パターン生成

条件付き確率

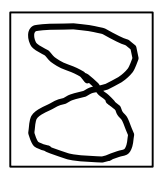
$P(\mathbf{x}|\omega)$

事後確率

$P(\omega|\mathbf{x})$

パターン認識

観測パターンx



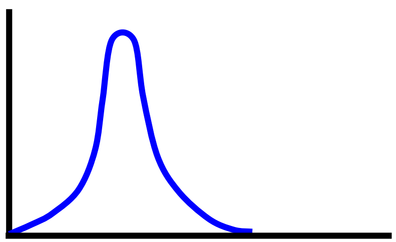
$P(\omega|\mathbf{x})$
$=P(\mathbf{x}|\omega)P(\omega)/P(\mathbf{x})$

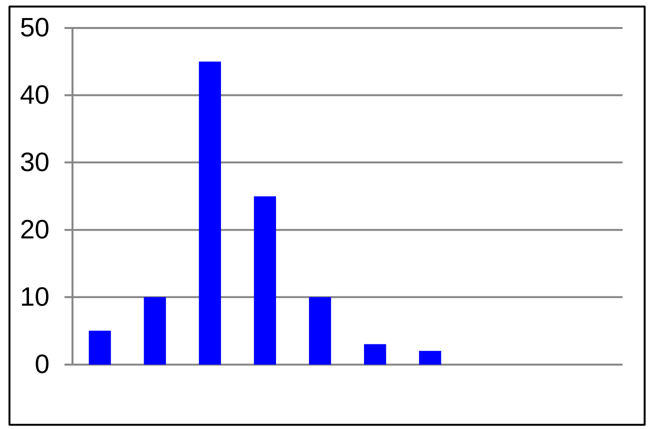
パラメータの推定方法

パラメータの推定方法①

母集団

特徴x
0.1, 0.4, 0.3, 0.5・・・





特徴xの出現する
真の分布

真の分布を推定
① 確率密度関数を仮定
② パラメータを推定

パラメータの推定方法②

- 確率密度関数は正規分布と仮定
- パラメータ（平均，分散共分散行列）はどのような値を用いればよいか

結論としては…

$$\hat{m}_8 = \frac{1}{n_8}\sum_{\mathbf{x}\in 8}\mathbf{x} \qquad \hat{m}_B = \frac{1}{n_B}\sum_{\mathbf{x}\in B}\mathbf{x}$$

$$\hat{\Sigma}_8 = \frac{1}{n_8}\sum_{\mathbf{x}\in B}(\mathbf{x}-\hat{\mathbf{m}}_8)(\mathbf{x}-\hat{\mathbf{m}}_8)^t \qquad \hat{\Sigma}_B = \frac{1}{n_B}\sum_{\mathbf{x}\in B}(\mathbf{x}-\hat{\mathbf{m}}_B)(\mathbf{x}-\hat{\mathbf{m}}_B)^t$$

最尤推定*①

- ω_i に属する n個のデータ $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$の集合を$\chi_i$
- $\boldsymbol{\theta}_i$ は推定したいパラメータ（$\mathbf{m}_i$, $\sum_i$）
- 確率密度関数 $p(\mathbf{x}| \omega_i; \boldsymbol{\theta}_i)$
- 各データ $\mathbf{x}_i$ は確率密度関数に従って独立に生起
- 集合χ_i が得られる確率 $p(\chi_i; \boldsymbol{\theta}_i)$

$$p(\chi_i;\boldsymbol{\theta}_i) = \prod_{k=1}^{n} p(\mathbf{x}_k | \omega_i;\boldsymbol{\theta}_i)$$

尤度（関数）

*参考書によっては最尤法とも呼ぶ場合もあります

最尤推定②

- 尤度を最大にする $\boldsymbol{\theta}_i$ を求める（最尤推定）

$$\max_{\boldsymbol{\theta}_i}\{\ p(\chi_i;\boldsymbol{\theta}_i)\} = p(\chi_i;\hat{\boldsymbol{\theta}}_i) = \prod_{k=1}^{n} p(\mathbf{x}_k|\omega_i;\hat{\boldsymbol{\theta}}_i)$$

$$\frac{\partial}{\partial\boldsymbol{\theta}_i} p(\chi_i;\boldsymbol{\theta}_i) = \mathbf{0}$$

対数をとった場合
（対数尤度）

$$\frac{\partial}{\partial\boldsymbol{\theta}_i} \log p(\chi_i;\boldsymbol{\theta}_i)$$

$$= \frac{\partial}{\partial\boldsymbol{\theta}_i} \log \prod_{k=1}^{n} p(\mathbf{x}_k \mid \omega_i;\boldsymbol{\theta}_i)$$

$$= \sum_{k=1}^{n} \frac{\partial}{\partial\boldsymbol{\theta}_i} \log p(\mathbf{x}_k \mid \omega_i;\boldsymbol{\theta}_i) = \mathbf{0}$$

最尤推定③

d次元の正規分布

$$p(\mathbf{x}_k \mid \omega_i;\mathbf{m}_i,\Sigma_i) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}_k-\mathbf{m}_i)^t \Sigma_i^{-1} (\mathbf{x}_k-\mathbf{m}_i)\right)$$

$$\sum_{k=1}^{n} \frac{\partial}{\partial \mathbf{m}_i} \log p(\mathbf{x}_k \mid \omega_i;\mathbf{m}_i,\Sigma_i) = \mathbf{0}$$

$$\sum_{k=1}^{n} \frac{\partial}{\partial \Sigma_i} \log p(\mathbf{x}_k \mid \omega_i;\mathbf{m}_i,\Sigma_i) = \mathbf{0}$$

$$\Downarrow$$

$$\hat{\mathbf{m}}_i = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$$

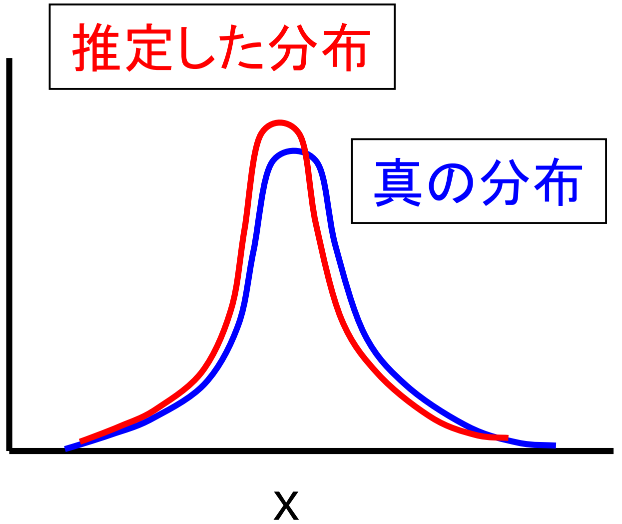
$$\hat{\Sigma}_i = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}-\hat{\mathbf{m}}_i)(\mathbf{x}-\hat{\mathbf{m}}_i)^t$$

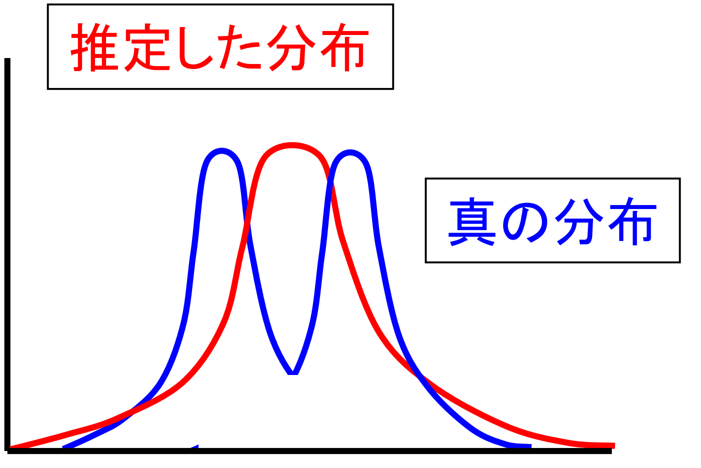
平均ベクトル，分散共分散行列は一般的に求める方法と同じ

EMアルゴリズム

混合正規分布のパラメータ推定

ここまでの確率密度関数





ここまで扱ってきた確率密度関数
「単峰性」を仮定

真の分布が「多峰性」の場合は
どうすればよいか

正規分布の場合

正規分布（d次元）

$$p(\mathbf{x}\,|\,\omega) = N(\mathbf{x}\,|\,\omega; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}\,|\,\Sigma\,|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu}) \right)$$

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

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t$$

平均ベクトル，分散共分散行列は一般的に求める方法と同じ

混合正規分布①

正規分布(d次元)

$$p(\mathbf{x}) = N(\mathbf{x};\boldsymbol{\mu},\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

混合正規分布
K個の正規分布が混合した確率密度関数

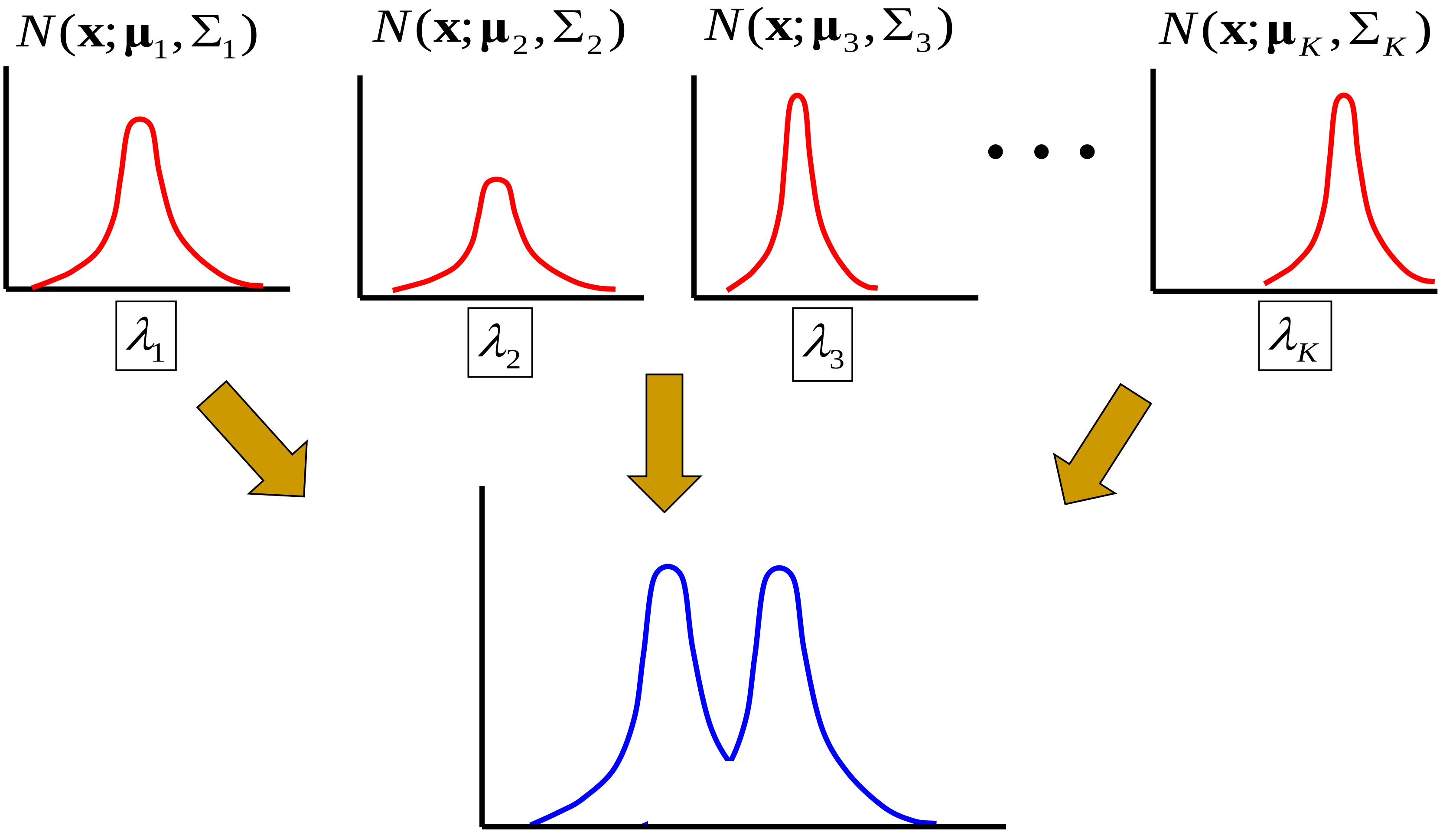
$$p(\mathbf{x}) = \sum_{k=1}^{K}\lambda_k N(\mathbf{x};\boldsymbol{\mu}_k,\Sigma_k) = \sum_{k=1}^{K}\lambda_k \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^t\Sigma_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

$$\sum_{k=1}^{K}\lambda_k = 1$$

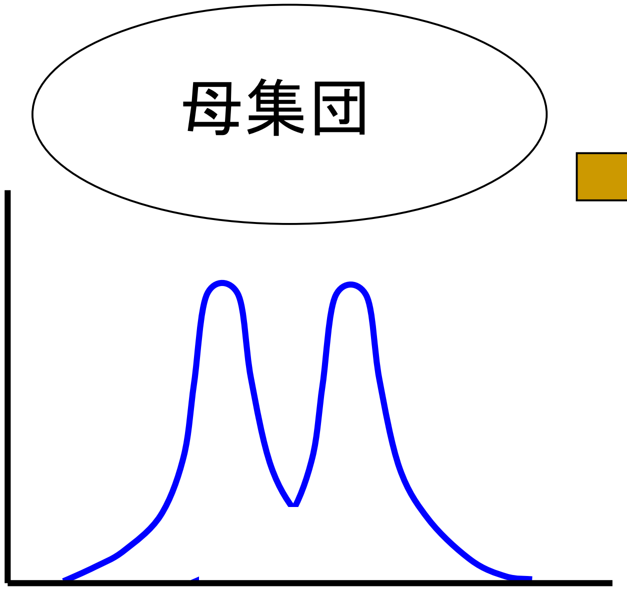
推定するパラメータ(k=1,2,･･･,K)
λ_k, $\boldsymbol{\mu}_k$, Σ_k

*p(**x**|ω) を p(**x**)と略す

混合正規分布②


$N(\mathbf{x};\boldsymbol{\mu}_1,\Sigma_1)$
$N(\mathbf{x};\boldsymbol{\mu}_2,\Sigma_2)$
$N(\mathbf{x};\boldsymbol{\mu}_3,\Sigma_3)$
・・・
$N(\mathbf{x};\boldsymbol{\mu}_K,\Sigma_K)$
λ_1
λ_2
λ_3
λ_K

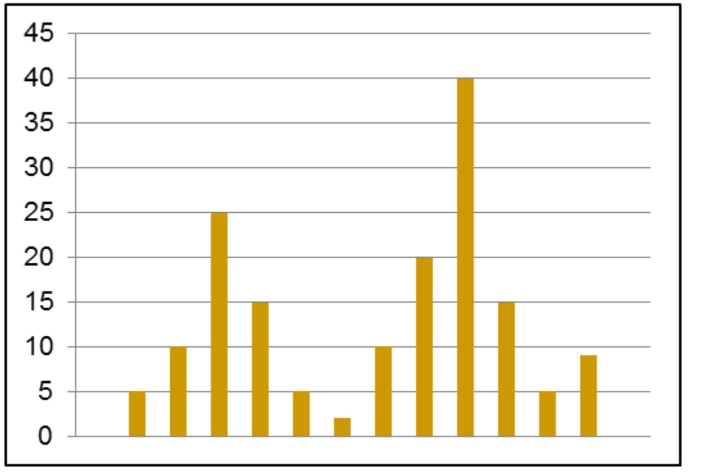
混合正規分布の推定



特徴xの出現する
真の分布

特徴x
0.1, 0.4, 0.3, 0.5 ･･･

度数



特徴 x

真の分布を推定
① 混合正規分布と仮定
② パラメータを推定

パラメータの推定方法①

- ωに属するn個のデータ $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$ の集合をχ
- $\boldsymbol{\theta}$は推定したいパラメータ（λ_k, $\boldsymbol{\mu}_k$, Σ_k, $k=1,2,\cdots K$）
- 確率密度関数 $p(\mathbf{x};\boldsymbol{\theta})$
- 各データ $\mathbf{x}_i$ は確率密度関数に従って独立に生起
- 集合χが得られる確率 $p(\chi;\boldsymbol{\theta})$

$$p(\chi;\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{x}_i;\boldsymbol{\theta})$$

尤度（関数）

パラメータの推定方法②

$$p(\chi;\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{x}_i;\boldsymbol{\theta})$$

$$= \prod_{i=1}^{n}\sum_{k=1}^{K}\lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)$$

$$= \prod_{i=1}^{n}\sum_{k=1}^{K}\lambda_k \frac{1}{(2\pi)^{d/2}\,|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_k)^t \Sigma_k^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_k)\right)$$

対数をとって

$$\log p(\chi;\boldsymbol{\theta}) = \log\prod_{i=1}^{n} p(\mathbf{x}_i;\boldsymbol{\theta})$$

$$= \sum_{i=1}^{n}\log\sum_{k=1}^{K}\lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)$$

最大化

最尤推定

$$\log p(\chi;\boldsymbol{\theta}) = \prod_{i=1}^{n} p(\mathbf{x}_i;\boldsymbol{\theta})$$

$$= \sum_{i=1}^{n} \log \sum_{k=1}^{K} \lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)$$

$$= \sum_{i=1}^{n} \log \sum_{k=1}^{K} Q_{ik} \frac{\lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)}{Q_{ik}}$$

$$\geq \sum_{i=1}^{n} \sum_{k=1}^{K} Q_{ik} \log \frac{\lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)}{Q_{ik}}$$

$$\sum_{k=1}^{K} Q_{ik} = 1$$

ジェンセンの不等式

$$\sum_{i=1}^{n} a_i = 1$$

$$\log(\sum_{i=1}^{n} a_i x_i) \geq \sum_{i=1}^{n} a_i \log(x_i)$$

Q_{ik}を下式とする

$$Q_{ik} = \frac{\lambda_k N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{k'=1}^{K} \lambda_{k'} N(\mathbf{x}_i; \boldsymbol{\mu}_{k'}, \Sigma_{k'})}$$

$$\frac{\lambda_k N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{Q_{ik}} = \sum_{k'=1}^{K} \lambda_{k'} N(\mathbf{x}_i; \boldsymbol{\mu}_{k'}, \Sigma_{k'})$$

$$\log p(\chi; \boldsymbol{\theta})$$

$$= \sum_{i=1}^{n} \log \sum_{k=1}^{K} Q_{ik} \frac{\lambda_k N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{Q_{ik}}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} Q_{ik} \log \frac{\lambda_k N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{Q_{ik}}$$

ジェンセンの不等式の等号が成立

$$\sum_{k'=1}^{K} \lambda_{k'} N(\mathbf{x}_i; \boldsymbol{\mu}_{k'}, \Sigma_{k'}) = \frac{\lambda_k N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{Q_{ik}}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{K} Q_{ik} \log \frac{\lambda_k N(\mathbf{x}_i; \boldsymbol{\mu}_k, \Sigma_k)}{Q_{ik}}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} Q_{ik} \log \sum_{k'=1}^{K} \lambda_{k'} N(\mathbf{x}_i; \boldsymbol{\mu}_{k'}, \Sigma_{k'})$$

$$= \sum_{i=1}^{n} \left(\sum_{k=1}^{K} Q_{ik}\right) \log \sum_{k'=1}^{K} \lambda_{k'} N(\mathbf{x}_i; \boldsymbol{\mu}_{k'}, \Sigma_{k'})$$

$$= \sum_{i=1}^{n} \log \sum_{k'=1}^{K} \lambda_{k'} N(\mathbf{x}_i; \boldsymbol{\mu}_{k'}, \Sigma_{k'})$$

$$= \log p(\chi; \boldsymbol{\theta})$$

$$\log p(\chi;\boldsymbol{\theta})$$

$$= \sum_{i=1}^{n}\sum_{k=1}^{K} Q_{ik} \log \frac{\lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)}{Q_{ik}}$$

最大化

$$\frac{\partial}{\partial \lambda_k} \log p(\chi;\boldsymbol{\theta}) = 0$$

$$\frac{\partial}{\partial \boldsymbol{\mu}_k} \log p(\chi;\boldsymbol{\theta}) = \mathbf{0}$$

$$\frac{\partial}{\partial \Sigma_k} \log p(\chi;\boldsymbol{\theta}) = 0$$

$$\lambda_k = \frac{1}{n}\sum_{i=1}^{n} \overline{Q}_{ik}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} \overline{Q}_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} \overline{Q}_{ik}}$$

$$\Sigma_k = \frac{\sum_{i=1}^{n} \overline{Q}_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{n} \overline{Q}_{ik}}$$

解法の参考書

新納浩幸，Rで学ぶクラスタ解析，オーム社，2007年

EMアルゴリズム①

パラメータθを初期化

E（Expectation）ステップ

$$\overline{Q}_{ik} = \frac{\lambda_k N(\mathbf{x}_i ; \boldsymbol{\mu}_k , \Sigma_k)}{\sum_{k'=1}^{K} \lambda_{k'} N(\mathbf{x}_i ; \boldsymbol{\mu}_{k'} , \Sigma_{k'})}$$

パラメータを用いて，Q_{ik}を更新
Mステップへ移る

$$N(\mathbf{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^t \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right)$$

EMアルゴリズム②

M（Maximization）ステップ

$$\lambda_k = \frac{1}{n}\sum_{i=1}^{n}\overline{Q}_{ik}$$

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n}\overline{Q}_{ik}\mathbf{x}_i}{\sum_{i=1}^{n}\overline{Q}_{ik}}$$

$$\Sigma_k = \frac{\sum_{i=1}^{n}\overline{Q}_{ik}(\mathbf{x}_i-\boldsymbol{\mu}_k)(\mathbf{x}_i-\boldsymbol{\mu}_k)^t}{\sum_{i=1}^{n}\overline{Q}_{ik}}$$

Eステップで推定したQ_{ik}を用いてパラメータを更新
Eステップに戻り，更新したパラメータを用いて，Q_{ik}を更新する

EMアルゴリズム
Eステップ，Mステップを交互に繰り返す

実習①
単純ベイズ決定則

実習(ベイズ決定則)

- 「頭痛」の日の判定
- 特徴
  - 天気 ・・・ 晴れ, 曇り, 雨
  - 気温 ・・・ 暑い, 適温, 寒い
  - 湿度 ・・・ 高い, 適当, 低い
  - 講義 ・・・ あり, なし

- 「頭痛」
  - 天気=晴れ, 気温=適温, 湿度=適当, 講義=あり
  - 天気=雨, 気温=寒い, 湿度=低い, 講義=あり
  - 頭痛はyes, no の判定

→「yes」と「no」のどちらのクラスに属するのか

数値データ例

	天気	気温	湿度	講義	頭痛
1	晴れ	寒い	低い	あり	yes
2	曇り	暑い	高い	なし	yes
3	晴れ	暑い	低い	なし	no
4	雨	適温	高い	あり	no
5	雨	寒い	高い	なし	no
6	晴れ	適温	適当	なし	yes
7	曇り	暑い	低い	あり	no
8	雨	寒い	高い	なし	no
9	曇り	適温	低い	あり	yes
10	曇り	適温	適当	なし	no
11	晴れ	暑い	適当	あり	yes
12	晴れ	適温	高い	なし	no
13	雨	暑い	低い	なし	no
14	雨	寒い	適当	あり	yes
15	曇り	適温	低い	あり	no
16	晴れ	暑い	適当	なし	yes
17	晴れ	寒い	高い	あり	yes
18	曇り	適温	高い	あり	no
19	曇り	適温	適当	なし	no
20	雨	寒い	適当	なし	yes

学習データ

Bayes-train.csv

テストデータ

Bayes-test.csv

天気=晴れ，気温=適温，湿度=適当，講義=あり→頭痛={yes,no}?

天気=雨，気温=寒い，湿度=低い，講義=あり→頭痛={yes,no}?

学習データ中に存在しない

単純ベイズ①

ベイズ決定則

- c個のクラスω_i（i=1,2,･･･,c)
  - 事前確率 $p(\omega_i)$ は既知
- 未知のパターンの特徴ベクトル **x**（d次元）
  - 条件付き確率 $p(\mathbf{x}|\omega_i)$
  - 特徴ベクトル **x** はどのクラスに属するか

$$\max_{i=1,2,\cdots,c}\{p(\omega_i\,|\,\mathbf{x})\}=\max_{i=1,2,\cdots,c}\left\{\frac{p(\mathbf{x}\,|\,\omega_i)\,p(\omega_i)}{p(\mathbf{x})}\right\}$$

$$=\max_{i=1,2,\cdots,c}\left\{p(\mathbf{x}\,|\,\omega_i)\,p(\omega_i)\right\}$$

$$=p(\omega_k\,|\,\mathbf{x})\Rightarrow\mathbf{x}\in\omega_k$$

単純ベイズ②

- 特徴の独立性を仮定

$$
\begin{aligned}
&\max_{i=1,2,\cdots,c}\{p(\omega_i \mid \mathbf{x})\} = \max_{i=1,2,\cdots,c}\left\{\frac{p(\mathbf{x} \mid \omega_i)\,p(\omega_i)}{p(\mathbf{x})}\right\} \\
&= \max_{i=1,2,\cdots,c}\left\{p(\mathbf{x} \mid \omega_i)\,p(\omega_i)\right\} = \max_{i=1,2,\cdots,c}\left\{p(x_1, x_2, \cdots, x_d \mid \omega_i)\,p(\omega_i)\right\} \\
&= \max_{i=1,2,\cdots,c}\left\{p(x_1 \mid \omega_i) \times p(x_1 \mid \omega_i) \cdots \times p(x_d \mid \omega_i) \times p(\omega_i)\right\} \\
&= \max_{i=1,2,\cdots,c}\left\{\prod_{j=1}^{d} p(x_j \mid \omega_i) \times p(\omega_i)\right\}
\end{aligned}
$$

単純ベイズ③

- 事後確率の求め方
  - p(頭痛=yes | 天気=晴れ，気温=適温，湿度=適当，講義=あり)
  - p(天気=晴れ|頭痛=yes)×
    p(気温=適温|頭痛=yes)×
    p(湿度=適当|頭痛=yes)×
    p(講義=あり|頭痛=yes)×p(頭痛=yes)
  - 現実的にはこの方法でも，いづれかの特徴の条件付き確率が0となり計算できない場合も多々ある

単純ベイズ④

- 事後確率の求め方
  - p(頭痛=no | 天気=晴れ，気温=適温，湿度=適当，講義=あり)
  - p(天気=晴れ|頭痛=no)×
    p(気温=適温|頭痛=no)×
    p(湿度=適当|頭痛=no)×
    p(講義=あり|頭痛=no)×p(頭痛=no)
- 事後確率の大小によって判定

単純ベイズのプログラム

- プログラム
  - Bayes.py

「Bayes」というフォルダー中にあります

- 学習データ
  - Bayes-train.csv
- テストデータ
  - Bayes-test.csv
- 実行方法
  - > python Bayes.py

変数の初期化

```
import sys
import os
import numpy as np

# データ
data = []
```

読み込む学習データ

```
answer = [ "yes" , "no" ]

# 事前確率
pp = { "yes":0 , "no":0 }
```

P(頭痛=yes) P(頭痛=no)

P(天気=晴れ|頭痛=yes) P(天気=晴れ|頭痛=no)

```
# 条件付き確率
cp = { "晴れ｜yes":0 , "晴れ｜no":0 ,
       "曇り｜yes":0 , "曇り｜no":0 ,
       "雨｜yes":0 , "雨｜no":0 ,
       "暑い｜yes":0 , "暑い｜no":0 ,
       "適温｜yes":0 , "適温｜no":0 ,
       "寒い｜yes":0 , "寒い｜no":0 ,
       "高い｜yes":0 , "高い｜no":0 ,
       "適当｜yes":0 , "適当｜no":0 ,
       "低い｜yes":0 , "低い｜no":0 ,
       "あり｜yes":0 , "あり｜no":0 ,
       "なし｜yes":0 , "なし｜no":0
       }
```

P(講義=なし|頭痛=yes) P(講義=なし|頭痛=no)

事前確率，条件付き確率の変数

- 事前確率　辞書　キー
  - P（頭痛=yes） pp["yes"]
  - P（頭痛=no） pp["no"]
- 条件付き確率　キー
  - P（天気=晴れ|頭痛=yes） cp["晴れ|yes"]
- 初期値は全て0

学習データの読み込み

```
# 学習データの読み込み
f = open( "Bayes-train.csv" , "r" )
count = 0
for line in f:
    # 先頭行を無視する
    if count == 0:
        count += 1
        continue

    data.append( line.rstrip().split( "," ) )
    count += 1
f.close()
```

「Bayes-train.csv」をオープン

line←晴れ,寒い,低い,あり,yes（改行）

rstrip() → 改行を削除
split(",") → 「,」で区切る

↓

dataに追加

ファイルを閉じる

読み込まれた学習データ

二次元配列data

	0	1	2	3	4
0	晴れ	寒い	低い	あり	yes
1	曇り	暑い	高い	なし	yes
2	晴れ	暑い	低い	なし	no
3	雨	適温	高い	あり	no
4	雨	寒い	高い	なし	no
5	晴れ	適温	適当	なし	yes
6	曇り	暑い	低い	あり	no
7	雨	寒い	高い	なし	no
8	曇り	適温	低い	あり	yes
9	曇り	適温	適当	なし	no
10	晴れ	暑い	適当	あり	yes
11	晴れ	適温	高い	なし	no
12	雨	暑い	低い	なし	no
13	雨	寒い	適当	あり	yes
14	曇り	適温	低い	あり	no
15	晴れ	暑い	適当	なし	yes
16	晴れ	寒い	高い	あり	yes
17	曇り	適温	高い	あり	no
18	曇り	適温	適当	なし	no
19	雨	寒い	適当	なし	yes

data[0]

data[19]

事前確率の計算

（例）s = ['晴れ', '寒い', '低い', 'あり', 'yes']

P(頭痛=yes)
=yesの合計/データ数

"yes" "no" の頻度

```
for s in data:
    pp[s[4]] +=1
```

pp["yes"]

pp["no"]

条件付き確率

```
for s in data:
    for i in range(4):
        work = s[i] + " | " +s[4]
        cp[ work ] += 1 / pp[ s[4] ]
```

辞書のキー（yes,no）を取り出す

事前確率

```
for i in pp.keys():
    pp[ i ] = pp[i] / len(data)
```

len(配列)
配列の長さ

	0	1	2	3	4
0	晴れ	寒い	低い	あり	yes
1	曇り	暑い	高い	なし	yes
2	[illegible]	[illegible]	[illegible]	[illegible]	no
3	[illegible]	[illegible]	[illegible]	[illegible]	no
4	[illegible]	[illegible]	[illegible]	[illegible]	no
5	晴れ	適温	適当	なし	yes
6	曇り	暑い	低い	あり	no
7	雨	寒い	高い	なし	no
8	曇り	適温	低い	あり	yes
9	曇り	適温	適当	なし	no
10	晴れ	暑い	適当	あり	yes
11	晴れ	適温	高い	なし	no
12	雨	暑い	低い	なし	no
13	雨	寒い	適当	あり	yes
14	曇り	適温	低い	あり	no
15	晴れ	暑い	適当	なし	yes
16	晴れ	寒い	高い	あり	yes
17	曇り	適温	高い	あり	no
18	曇り	適温	適当	なし	no
19	雨	寒い	適当	なし	yes

P(頭痛=no)
=noの合計/データ数

条件付き確率の計算

s = ['晴れ', '寒い', '低い', 'あり', 'yes'] の場合

```
# "yes" "no" の頻度
for s in data:
    pp[s[4]] +=1

# 条件付き確率
for s in data:
  for i in range(4):
      work = s[i] + " | " +s[4]
      cp[ work ] += 1 / pp[ s[4] ]

# 事前確率
for i in pp.keys():
    pp[ i ] = pp[i] / len(data)
```

cp["晴れ | yes"]

```
# 天気
work = s[0] + " | " +s[4]
cp[ work ] += 1 / pp[ s[4] ]

# 気温
work = s[1] + " | " +s[4]
cp[ work ] += 1 / pp[ s[4] ]

# 湿度
work = s[2] + " | " +s[4]
cp[ work ] += 1 / pp[ s[4] ]

# 講義
work = s[3] + " | " +s[4]
cp[ work ] += 1 / pp[ s[4] ]
```

work = "晴れ | yes"

work = "寒い | yes"

work = "低い | yes"

work = "あり | yes"

pp["yes"]
yesの回数

条件付き確率，事前確率の出力

```
print( " [ 事前確率 ]" )
for i in pp.keys():
    print( "{:^10} : {:6.4f}".format( i , pp[i] ) )

# 条件付き確率
print( "¥n [ 条件付き確率 ]" )
for i in cp.keys():
    print( "{:^10} : {:6.4f}".format( i , cp[i] ) )
```

6桁，小数点以下4桁

10文字，中央寄せ

テストデータの読み込み

```
# テストデータの読み込み
f = open( "Bayes-test.csv" , "r" )
count = 0
print( "¥n [ 事後確率 ]" )
for line in f:
    # 先頭行を無視する
    if count == 0:
        count += 1
        continue

    work = line.rstrip().split( "," )
```

「Bayes-test.csv」をオープン

line：晴れ,適温,適当,あり

rstrip() → 改行を削除
split(",") → 「,」で区切る

↓

work=['晴れ','適温','適当','あり']

事後確率の計算①

p(天気=晴れ|頭痛=yes)×p(気温=適温|頭痛=yes)×p(湿度=適当|頭=yes)
×p(講義=あり|頭痛=yes)×p(頭痛=yes)

work=['晴れ','適温','適当','あり']

i=0

answer[0]→yes

```
# 事後確率を求める
predict = np.ones( 2 , np.float32 )
for i in range(2):
    for j in range(4):
        temp = work[j] + "｜" + answer[i]
        predict[i] = predict[i] * cp[ temp ]
    predict[i] = predict[i] * pp[answer[i]]
```

j=0
temp="晴れ｜yes"

j=1
temp="適温｜yes"

j=2
temp="適当｜yes"

j=3
temp="あり｜yes"」

cp["晴れ｜yes"]
×
cp["適温｜yes"]
×
cp["適当｜yes"]
×
cp["あり｜yes"]
×
pp["yes"]

predict[0]

事後確率の計算②

p(天気=晴れ|頭痛=no)×p(気温=適温|頭痛=no)×p(湿度=適当|頭=no)×p(講義=あり|頭痛=no)×p(頭痛=no)

work=['晴れ','適温','適当','あり']

```
# 事後確率を求める
predict = np.ones( 2 , np.float32 )
for i in range(2):
    for j in range(4):
        temp = work[j] + "｜" + answer[i]
        predict[i] = predict[i] * cp[ temp ]
    predict[i] = predict[i] * pp[answer[i]]
```

i=1

answer[1]→no

j=0 temp="晴れ｜no"

j=1 temp="適温｜no"

j=2 temp="適当｜no"

j=3 temp="あり｜no"」

cp["晴れ｜no"]

×

cp["適温｜no"]

×

cp["適当｜no"]

×

cp["あり｜no"]

×

pp["no"]

predict[1]

事後確率が最大の結果の出力

```
# 事後確率の出力
s = ""
for i in range(2):
    s = answer[ i ] + " | "
    for j in range(4):
        s += work[j] + " "
    print( "{:>18}: {:10.8f}".format( s , predict[ i ] ) )

# 事後確率最大の結果の出力
ans = np.argmax( predict )
print( answer[ ans ] )
count+=1
f.close()
```

18桁，右詰め

10桁，小数点以下8桁

np.argmax（配列）
配列の最大値の要素番号を返す

predictの最大値の要素番号を返す

ファイルを閉じる

出力結果

C:¥Windows¥system32¥cmd.exe

```
C:¥home¥shino¥prml-2018¥10-22¥program>python3
 [ 事前確率 ]
   yes      : 0.4500
    no      : 0.5500

 [ 条件付き確率 ]
  晴れ | yes   : 0.5556
  晴れ | no    : 0.1818
  曇り | yes   : 0.2222
  曇り | no    : 0.4545
  雨 | yes    : 0.2222
   雨 | no    : 0.3636
  暑い | yes   : 0.3333
  暑い | no    : 0.2727
  適温 | yes   : 0.2222
  適温 | no    : 0.5455
  寒い | yes   : 0.4444
  寒い | no    : 0.1818
  高い | yes   : 0.2222
  高い | no    : 0.4545
  適当 | yes   : 0.5556
  適当 | no    : 0.1818
  低い | yes   : 0.2222
  低い | no    : 0.3636
  あり | yes   : 0.5556
  あり | no    : 0.3636
  なし | yes   : 0.4444
  なし | no    : 0.6364
```

事前確率

条件付き確率

P(yes|晴れ，適温，適当，あり)
> P(no|晴れ，適温，適当，あり)
→「yes」と認識

事後確率

C:¥Windows¥system32¥cm

```
 [ 事後確率 ]
  yes | 晴れ 適温 適当 あり : 0.01714678
   no | 晴れ 適温 適当 あり : 0.00360631
yes
   yes | 雨 寒い 低い あり : 0.00548697
    no | 雨 寒い 低い あり : 0.00480842
yes
```

P(yes|雨，寒い，低い，あり)
> P(no|雨，寒い，低い，あり)
→「yes」と認識

宿題④

- 「頭痛」の日の判定
- 特徴
  - 天気・・・晴れ，曇り，雨
  - 気温・・・暑い，適温，寒い
  - 湿度・・・高い，適当，低い
  - 睡眠・・・寝過ぎ，普通，寝不足
  - 講義・・・あり，なし
- 「頭痛」
  - 頭痛はyes，no

学習データ(25個)

Prob-4-train.csv

天気	気温	湿度	睡眠	講義	頭痛
晴れ	寒い	低い	寝過ぎ	あり	yes
曇り	暑い	高い	普通	なし	yes
晴れ	暑い	低い	寝過ぎ	なし	no
雨	適温	高い	寝不足	あり	no
雨	寒い	高い	普通	なし	no
晴れ	適温	適当	普通	あり	yes
曇り	暑い	低い	寝不足	あり	no
雨	寒い	高い	寝過ぎ	なし	no
曇り	適温	低い	寝過ぎ	あり	yes
曇り	適温	適当	寝不足	なし	no
晴れ	暑い	適当	普通	あり	yes
晴れ	適温	高い	普通	なし	no
雨	暑い	低い	寝過ぎ	なし	no
雨	暑い	適当	寝過ぎ	あり	yes
曇り	適温	低い	普通	あり	no
晴れ	暑い	適当	寝不足	なし	yes
晴れ	寒い	高い	寝過ぎ	あり	yes
曇り	適温	高い	普通	あり	no
曇り	適温	適当	普通	なし	no
雨	寒い	適当	普通	なし	yes
晴れ	寒い	低い	普通	なし	no
雨	適温	高い	寝不足	あり	yes
曇り	暑い	適当	普通	あり	no
晴れ	暑い	高い	寝過ぎ	なし	yes
晴れ	適温	適当	寝過ぎ	なし	yes

宿題④

- 学習データ（Prob-4-train.csv，25個）を用いて，事前確率，条件付き確率を推定し，テストデータ（Prob-4-test.csv，5個）の条件の場合，頭痛がyesか，noか，をベイズ決定則にて推定しなさい．

テストデータ（5個）

Prob-4-test.csv

天気	気温	湿度	睡眠	講義	頭痛
雨	寒い	高い	寝過ぎ	あり	
晴れ	寒い	高い	寝過ぎ	なし	
晴れ	暑い	適当	寝不足	あり	
曇り	暑い	低い	普通	なし	
曇り	適温	適当	普通	あり	

実習②
混合正規分布の例題

EMアルゴリズム

混合正規分布のプログラム

「EM」というフォルダー中にあります

- GM.py
  - 混合正規分布の表示プログラム
- EM.py
  - EMアルゴリズムを用いて，混合正規分布(一変量)のパラーメータを推定するプログラム
- EM-1.py
  - EMアルゴリズムを用いて，混合正規分布(二変量)のパラーメータを推定するプログラム*

*途中までです

実行方法

- 必要なライブラリイ
  - > pip install scipy

Anacondaにはインストール済み

- GM.py
  - > python GM.py
- EM.py
  - > python EM.py
- EM-1.py
  - > python EM-1.py

途中までです（宿題用）

混合正規分布の表示プログラム（GM.py）

```
import sys
import os
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as st

# 合成する正規分布の個数
K = 3

# データ数
N = 100
line = np.linspace(-10, 10, N)

# 平均，標準偏差，重みを格納する配列
average = []
sigma = []
lamda =[]
```

ライブラリイのインポート
scipy が必要

K個の正規分布を合成した
混合正規分布

－10から10までN（=100）等分

普通のリストです

正規分布のパラメータの設定

```
# 平均，標準偏差，重みの設定
for k in range(K):
    a = (np.random.rand() - 0.5) * 10.0
    s = np.random.rand() * 3.0
    l = np.random.randint(1,10)
    average = np.append( average , a )
    sigma = np.append( sigma , s )
    lamda = np.append( lamda , l )

sum_l = np.sum( lamda )
for k in range(K):
    lamda[k] = lamda[k] / sum_l
print( average , sigma , lamda )
```

平均：ー5から5

標準偏差：0から3

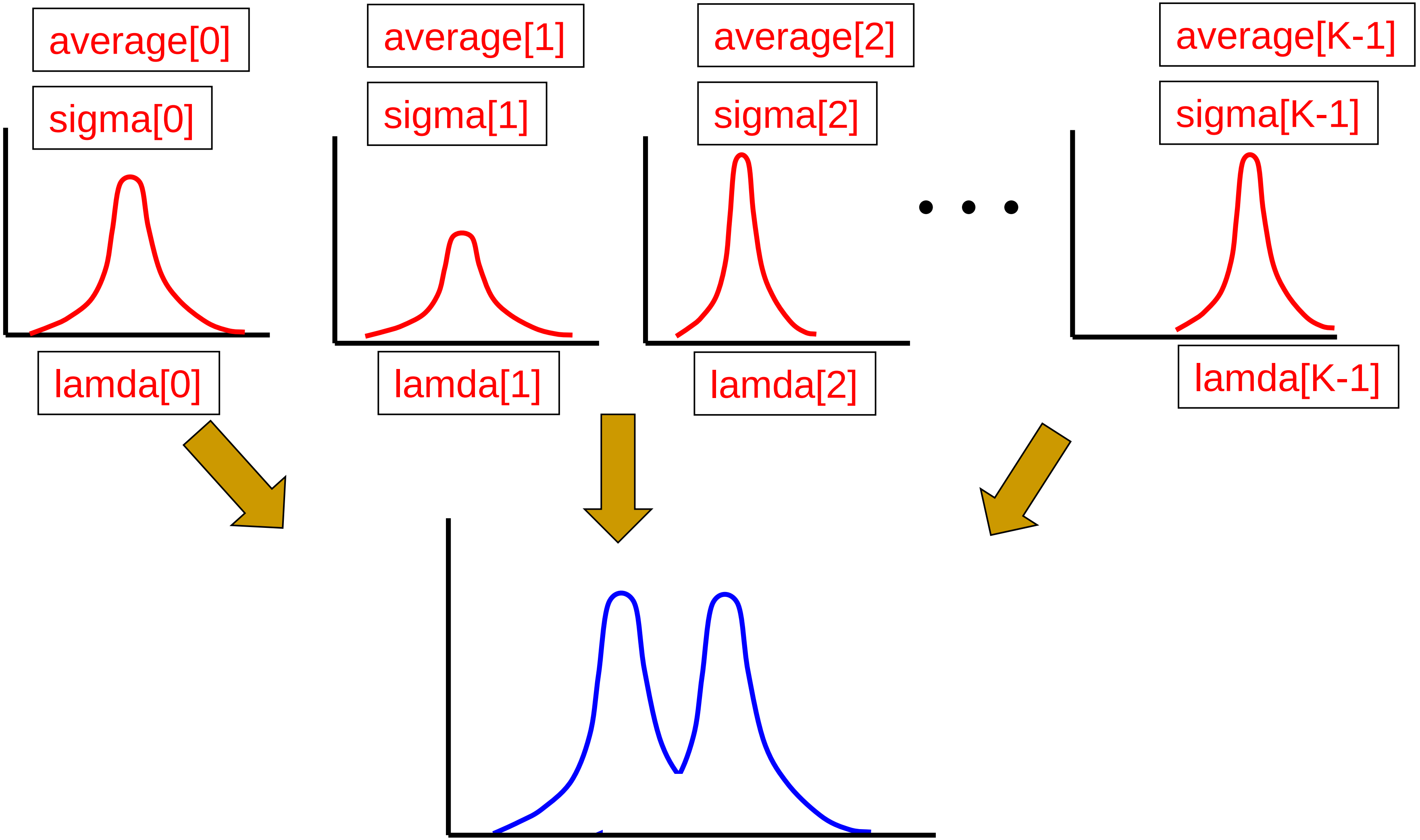
平均がaverage[k]，標準偏差sigma[k]の正規分布をK個設定

重み：lamda[k]

$$\lambda_i = \frac{x_i}{\sum_{k=1}^{K} x_i}$$

x_i：1〜10

混合正規分布(一変量)の生成



混合正規分布

gaussian：(K×N)個

```
# 正規分布を格納する配列
gaussian = np.zeros((K, N),dtype=np.float64)

# 混合正規分布を格納する配列
gaussian_mixture = np.zeros(N,dtype=np.float64)
for k in range(K):
    for n in range(N):
        # 正規分布
        gaussian[k][n] =  st.norm.pdf(line[n], average[k], sigma[k])

        # 混合正規分布
        gaussian_mixture[n] += lamda[k] * gaussian[k][n]
```

gaussian_mixture：(K×N)個

st.norm.pdf(x,a,s)
値がx，平均a，標準偏差sの正規分布の値を求める

－10から10までN等分した間隔ごとで求める

重みlamda[k]でK個の正規分布を合成
→混合正規分布

グラフの表示

```
# 表示
plt.figure()
plt.subplot(2,1,1)
for k in range(K):
    plot_t = "lamda={:6.4f}".format( lamda[k] )
    plt.plot(line, gaussian[k],label=plot_t)
    plt.legend()
    plt.title("Gaussian")

plt.subplot(2,1,2)
plt.plot(line, gaussian_mixture )
plt.title("Gaussian Mixture")
plt.tight_layout()
plt.show()
plt.clf()
```

plt.subplot（行数，列数，番号）

二行一列の一番目のグラフ表示

ラベル名の生成

K個の正規分布の表示

凡例の表示

タイトルの表示

二行一列の二番目のグラフ表示

混合正規分布の表示

位置の調整

ヒストグラムを表示→終了

plt.subplot(2,1,1)
plt.subplot(2,1,2)

実行結果

K=3, N=100

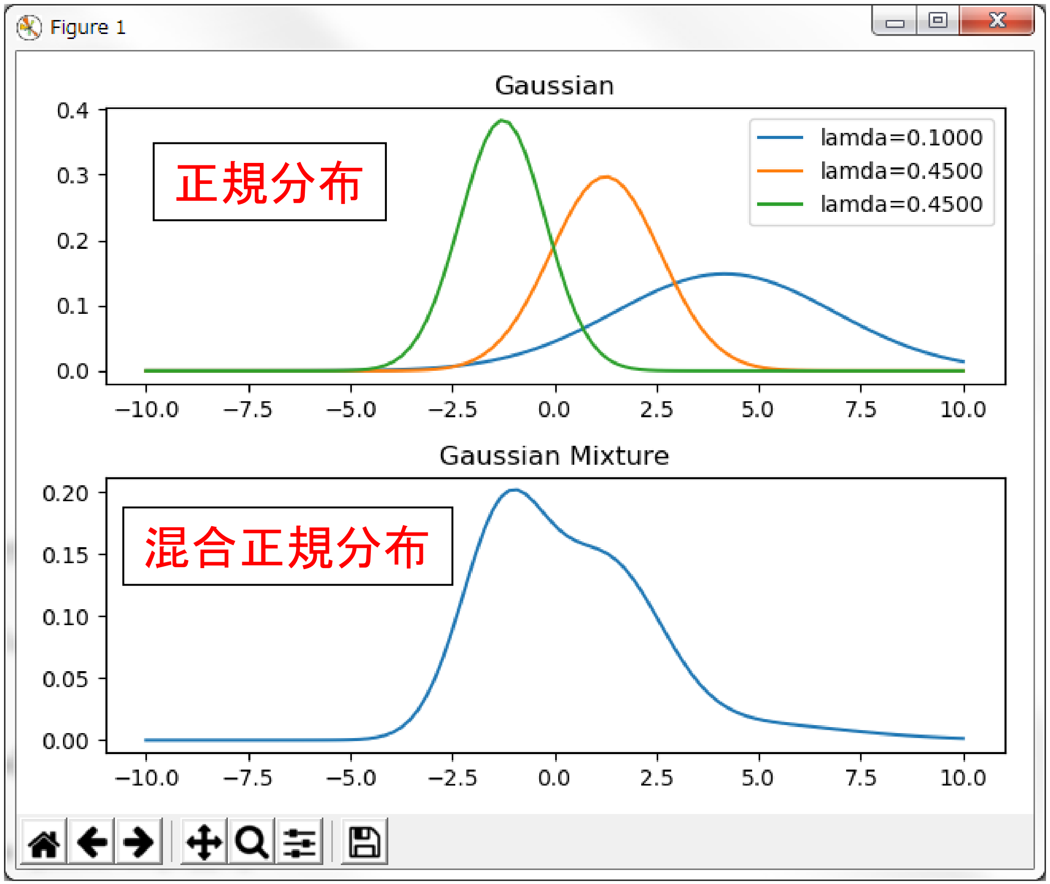
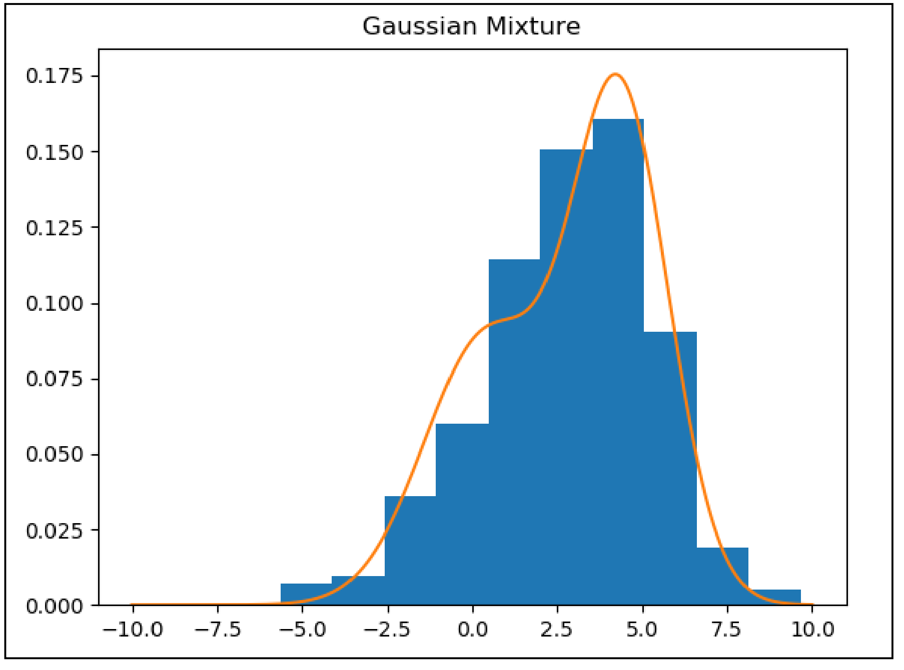


Figure 1
Gaussian
正規分布
lamda=0.1000
lamda=0.4500
lamda=0.4500
0.4
0.3
0.2
0.1
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
Gaussian Mixture
混合正規分布
0.20
0.15
0.10
0.05
0.00
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

混合正規分布のパラメータの推定(EM.py)

- K個の正規分布(一変量)からサンプリングしたデータを混合正規分布としてそのパラメータを推定



パラメータの設定

```
# 正規分布の個数
K = 2

# 正規分布一つあたりのデータ数
P = 500

# 全データ数
N = K*P

# 繰り返し回数
LOOP = 10

# データ，平均，標準偏差
data = []
average = []
sigma = []
```

K個の正規分布から，一つ当たりP個のデータを生成
（全データ数はK×P個）
データは一次元
→N（=K×P）個のデータを混合正規分布としてパラメータを推定

正規分布の生成①

```
# 平均，標準偏差の設定
for k in range(K):
    a = (np.random.rand() - 0.5) * 10.0
    s = np.random.rand() * 3.0
    average = np.append( average , a )
    sigma = np.append( sigma , s )
    print( average , sigma )

# K個の正規分布を生成
for k in range(K):
    data = np.append( data , np.random.normal(average[k],sigma[k],P) )
```

平均：ー5から5

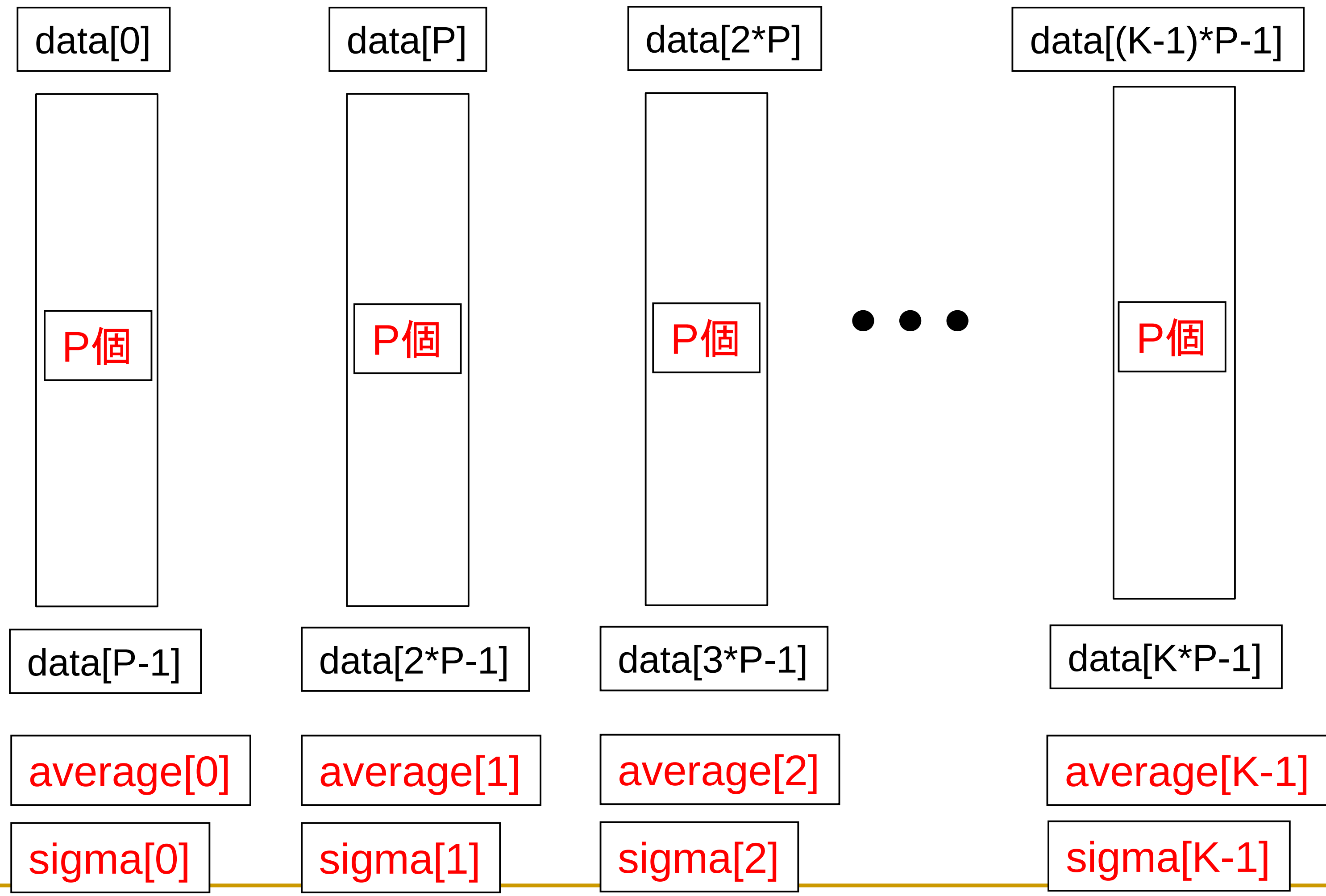
標準偏差：0から3

平均がaverage[k]，標準偏差sigma[k]の正規分布をK個設定

正規分布の乱数
np.random.normal(a,s,n)
a：平均
s：標準偏差
n：生成する個数

平均がaverage[k]，標準偏差sigma[k]の正規分布に従う乱数をP個生成

正規分布の生成②


data[0]
data[P]
data[2*P]
data[(K-1)*P-1]
P個
P個
P個
・・・
P個
data[P-1]
data[2*P-1]
data[3*P-1]
data[K*P-1]
average[0]
average[1]
average[2]
average[K-1]
sigma[0]
sigma[1]
sigma[2]
sigma[K-1]

ヒストグラムの作成

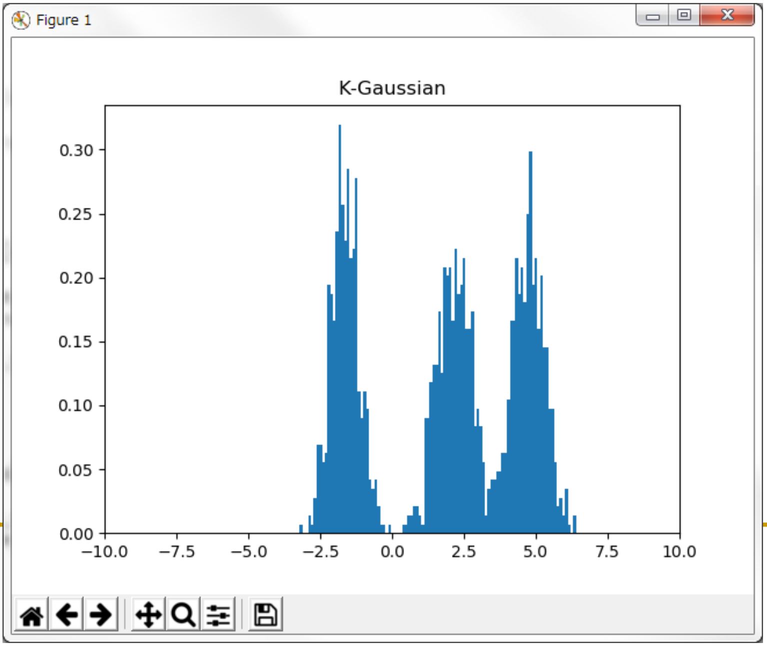
```
# ヒストグラムの表示
plt.figure()
plt.hist(data,bins=100,normed=True)
plt.title("K-Gaussian")
plt.xlim([-10,10])
plt.show()
plt.clf()
```

ヒストグラムの表示
bins：ビン数（100）
normed：True→正規化（合計が1）

タイトルの表示

x軸の範囲は－10から10

ヒストグラムを表示→終了



変数の初期化

```
# 初期化（Q値，平均，標準偏差，重み）
Q = np.zeros((K, N),dtype=np.float64)
e_average = np.zeros(K,dtype=np.float64)
e_sigma = np.zeros(K,dtype=np.float64)
e_lamda = np.zeros(K,dtype=np.float64)
for k in range(K):
    e_average[k] = random.choice(data)
    e_sigma[k] = np.sqrt(np.var(data))
    e_lamda[k] = 1/K
```

Q値：（K×N）個→初期値は0

推定する平均値：K個

推定する標準偏差：K個

推定する重み：K個

平均値：dataの中からランダムに選択

標準偏差：全データの標準偏差

重み：1/K

Eステップ

Q値の計算

$$\overline{Q}_{ik} = \frac{\lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)}{\sum_{k'=1}^{K}\lambda_{k'} N(\mathbf{x}_i;\boldsymbol{\mu}_{k'},\Sigma_{k'})}$$

LOOP回繰り返す

```
for loop in range(LOOP):

  for i in range(N):
    temp = np.zeros(K,dtype=np.float64)
    sum_t = 0
    for k in range(K):
      temp[k] = st.norm.pdf( data[i] , e_average[k] , e_sigma[k] )
      sum_t += ( temp[k] * e_lamda[k] )

    for k in range(K):
      Q[i][k] = ( temp[k] * e_lamda[k] ) / sum_t
```

$N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)$

st.norm.pdf(x,a,s)
値がx，平均a，標準偏差s
の正規分布の値を求める

$$\sum_{k'=1}^{K}\lambda_{k'} N(\mathbf{x}_i;\boldsymbol{\mu}_{k'},\Sigma_{k'})$$

$$\frac{\lambda_k N(\mathbf{x}_i;\boldsymbol{\mu}_k,\Sigma_k)}{\sum_{k'=1}^{K}\lambda_{k'} N(\mathbf{x}_i;\boldsymbol{\mu}_{k'},\Sigma_{k'})}$$

Mステップ（重みの更新）

$$\boxed{\lambda_k = \frac{1}{n}\sum_{i=1}^{n}\overline{Q}_{ik}}$$

```
for k in range(K):
        sum_Q = 0
        for i in range(N):
            sum_Q += Q[i][k]
        e_lamda[k] = sum_Q / N
    print( loop , ":lamda -> " , e_lamda )
```

$$\boxed{\sum_{i=1}^{n}\overline{Q}_{ik}}$$

Mステップ（平均値の更新）

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} \overline{Q}_{ik} \mathbf{x}_i}{\sum_{i=1}^{n} \overline{Q}_{ik}}$$

```
new_average = np.zeros(K,dtype=np.float32)
    for k in range(K):
        sum_q = 0        分子
        sum_q1 = 0       分母
        for i in range(N):
            sum_q += Q[i][k] * data[i]
            sum_q1 += Q[i][k]

        new_average[k] = sum_q / sum_q1
    print( loop , ":average -> " , new_average )
```

$\sum_{i=1}^{n} \overline{Q}_{ik} \mathbf{x}_i$

$\sum_{i=1}^{n} \overline{Q}_{ik}$

$\frac{\sum_{i=1}^{n} \overline{Q}_{ik} \mathbf{x}_i}{\sum_{i=1}^{n} \overline{Q}_{ik}}$

Mステップ（標準偏差の更新）

$$\Sigma_k = \frac{\sum_{i=1}^{n} \overline{Q}_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{n} \overline{Q}_{ik}}$$

```
new_sigma = np.zeros(K,dtype=np.float32)
    for k in range(K):
        sum_q = 0    分子
        sum_q1 = 0   分母
        for i in range(N):
            sum_q += Q[i][k] * (data[i]-e_average[k])**2
            sum_q1 += Q[i][k]
        new_sigma[k] = np.sqrt( sum_q / sum_q1 )
    print( loop , ":sigma -> " , new_sigma )
```

$$\sum_{i=1}^{n} \overline{Q}_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^t$$

$$\sum_{i=1}^{n} \overline{Q}_{ik}$$

$$\frac{\sum_{i=1}^{n} \overline{Q}_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^t}{\sum_{i=1}^{n} \overline{Q}_{ik}}$$

平均値，標準偏差の更新

```
e_average = new_average.copy()
e_sigma = new_sigma.copy()
```

（×）

```
e_average = new_average
e_sigma = new_sigma
```

グラフの表示①

result_gaussian：(K×N)個

```
# 推定した正規分布
result_gaussian = np.zeros((K, N),dtype=np.float64)

# 推定した混合正規分布
result_gaussian_mixture = np.zeros(N,dtype=np.float64)
line = np.linspace(-10,10,N)
for k in range(K):
    for i in range(N):
        result_gaussian[k][i] = st.norm.pdf(line[i], e_average[k], e_sigma[k])
        result_gaussian_mixture[i] +=e_lamda[k] *  result_gaussian[k][i]
```

result_gaussian_mixture：N個

st.norm.pdf(x,a,s)
値がx，平均a，標準偏差sの正規分布の値を求める

予測した平均，標準偏差を用いて，－10から10までN等分した間隔ごとに正規分布の値を求める

混合正規分布の値を求める

グラフの表示②

ヒストグラムの表示
bins：ビン数（100）
normed：True→正規化（合計が1）

```
plt.figure()
plt.hist(data,bins=100,normed=True)
plt.plot(line, result_gaussian_mixture )
plt.title("Gaussian Mixture")
plt.show()
plt.clf()
```

元のデータのヒストグラムの表示

予測した混合正規分布の表示

タイトルの表示

グラフの表示→終了

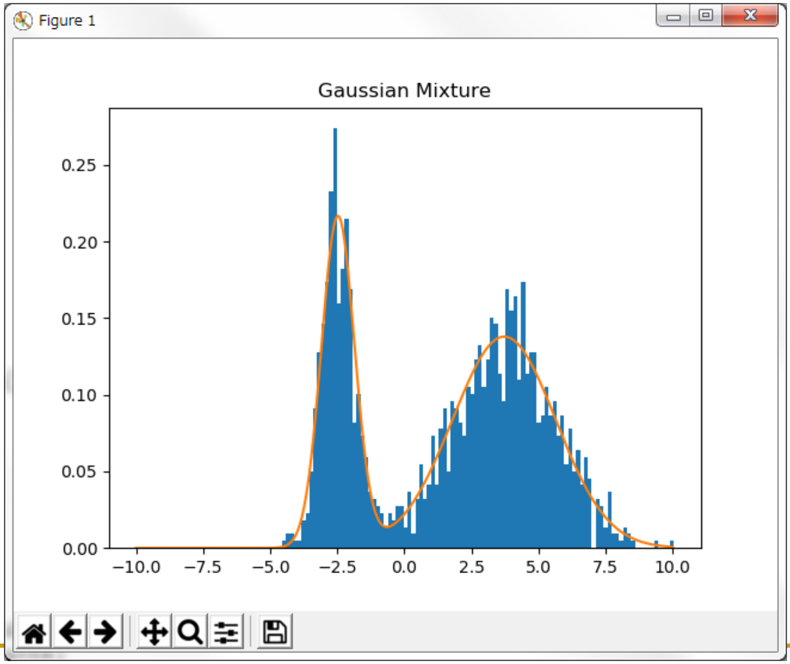
実行結果

```
-----
9 :lamda ->  [0.31817888 0.33290116 0.34891996]
9 :average ->  [ 3.5802479 -2.4763138  3.817239 ]
9 :sigma ->  [2.0286736  0.61575615 1.8420494 ]
-----
```

推定した重み

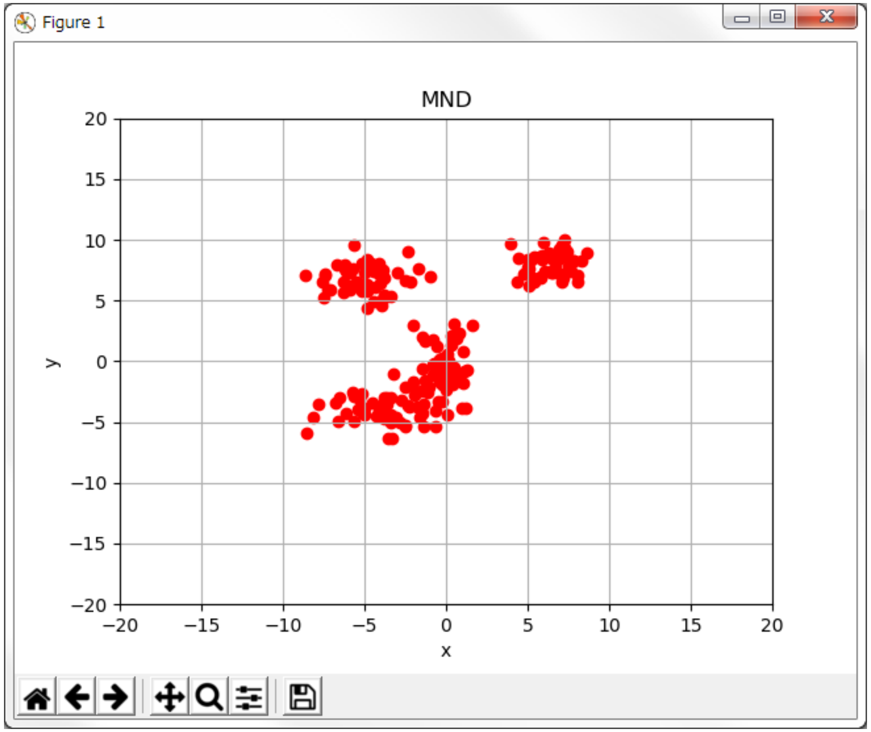
推定した平均

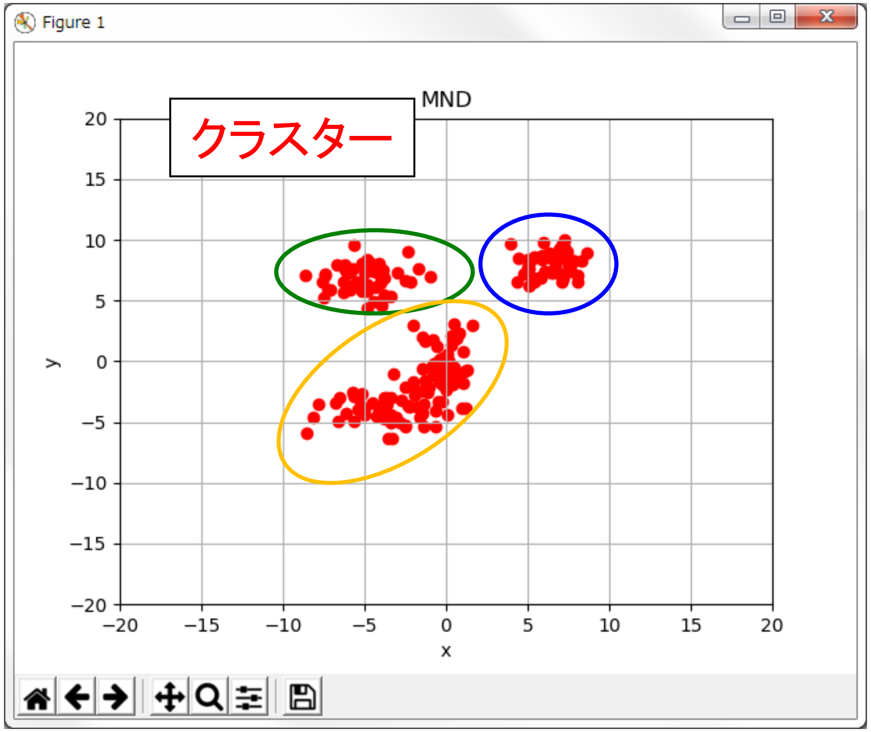
推定した標準偏差



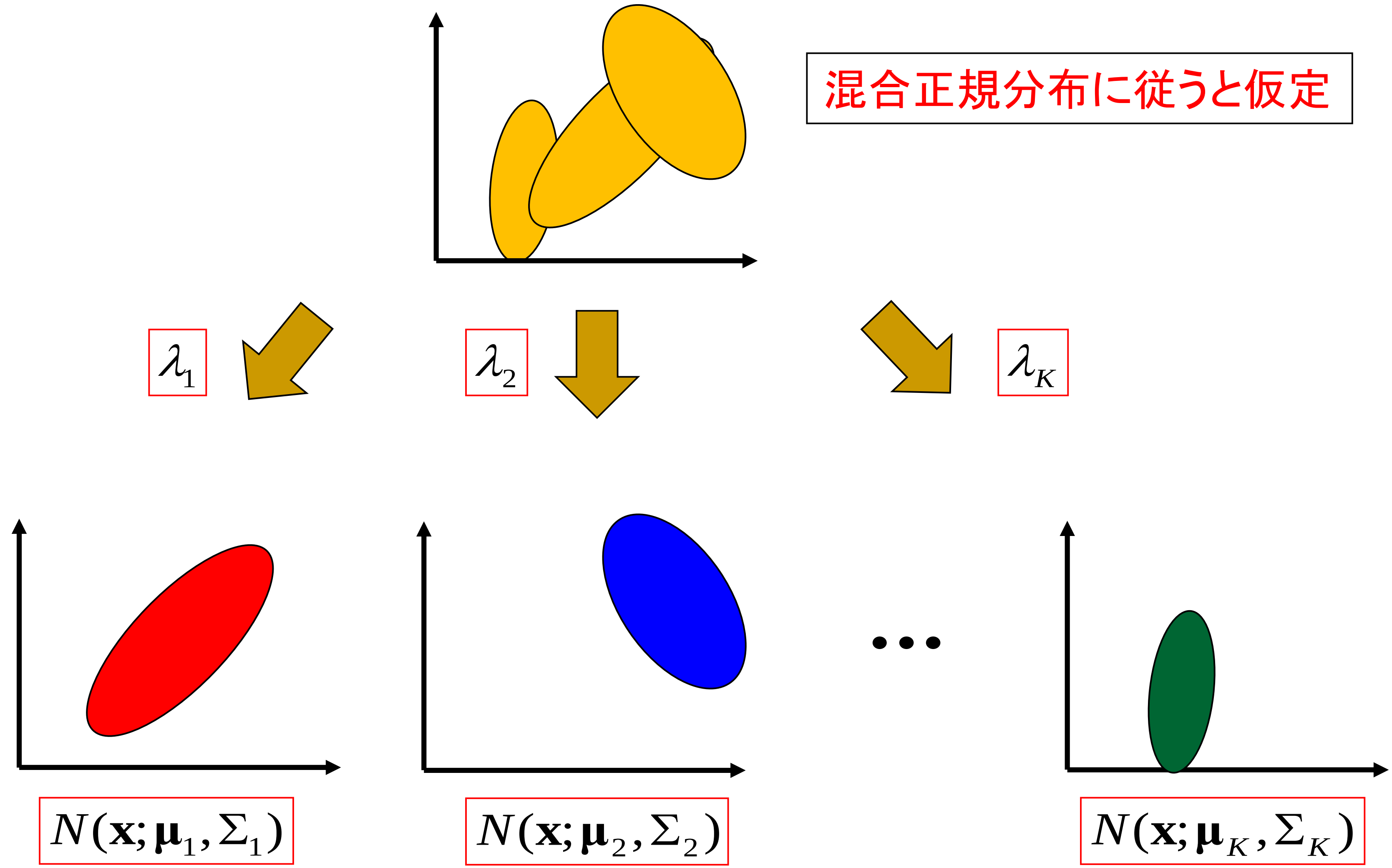
予測した混合正規分布

クラスタリングへの応用





クラスタリングへの応用



```
import sys
import os
import numpy as np
import random
import matplotlib.pyplot as plt

# 正規分布の個数
K = 4

# 特徴数（このプログラムはD=2のみ，D>2は表示ができないので注意）
D = 2

# 正規分布一つあたりのデータ数
P = 100

# 全データ数
N = K*P
```

ライブラリイのインポート

多次元正規分布の確率密度関数

$$N(\mathbf{x};\boldsymbol{\mu},\Sigma)=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

```
# 正規分布(多変量)
def mnd(x, ave, cov):
    a = np.sqrt( np.linalg.det(cov)*(2*np.pi)**cov.ndim)
    b = np.reshape( x-ave , (D,1) )

    bT = np.reshape( x-ave , (1,D) )
    c = -0.5 * np.dot( np.dot( bT , np.linalg.inv(cov) ) , b )
    return np.exp(c)/a
```

$$(2\pi)^{d/2}|\Sigma|^{1/2}$$

$$(\mathbf{x}-\boldsymbol{\mu})$$

$$(\mathbf{x}-\boldsymbol{\mu})^t$$

$$\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

平均ベクトル，分散共分散行列の設定

```
#平均ベクトル，分散共分散行列
average = np.zeros( (K,D) , dtype=np.float32 )
cov = np.zeros( (K,D,D) , dtype=np.float32 )

# 平均，標準偏差の設定
for k in range(K):
    for d in range(D):
        a = (np.random.rand() - 0.5) * 20.0
        average[k][d] = a

    for i in range(D):
        for j in range(D):
            s = np.random.randint( 1 , 5 )
            if i == j:
                cov[k][i][j] = s
```

平均ベクトル：（K×D）個

分散共分散行列：（K×D×D）個
→初期値を0とする

平均：ー10～10

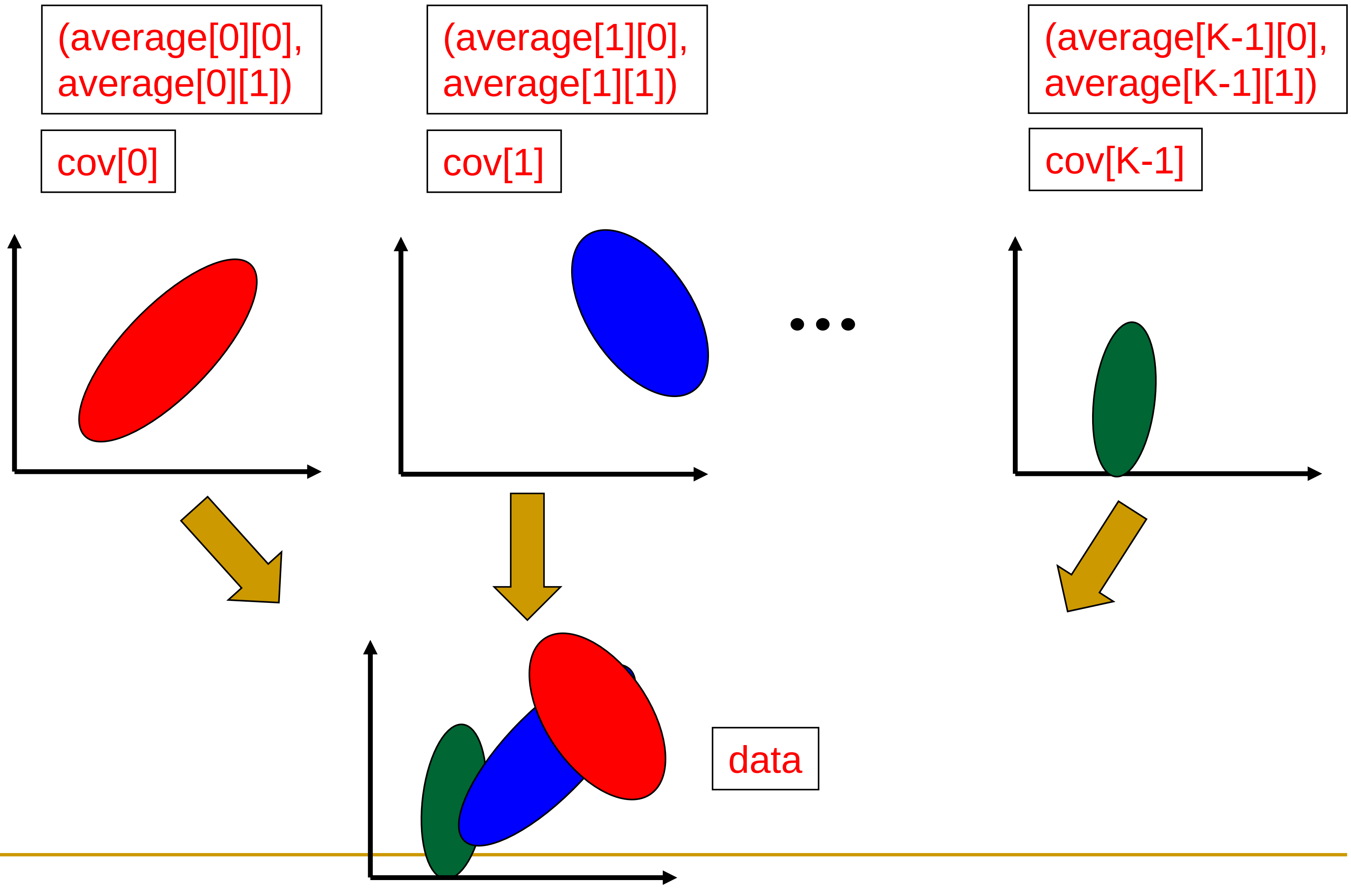
average[k][d]（平均値）をー10から10に設定

分散：1から5

共分散：0

cov[k][d][d]
対角成分（分散）は1から5，それ以外（共分散）は0

データの生成①


(average[0][0],
average[0][1])
cov[0]
(average[1][0],
average[1][1])
cov[1]
(average[K-1][0],
average[K-1][1])
cov[K-1]
data

データの生成②

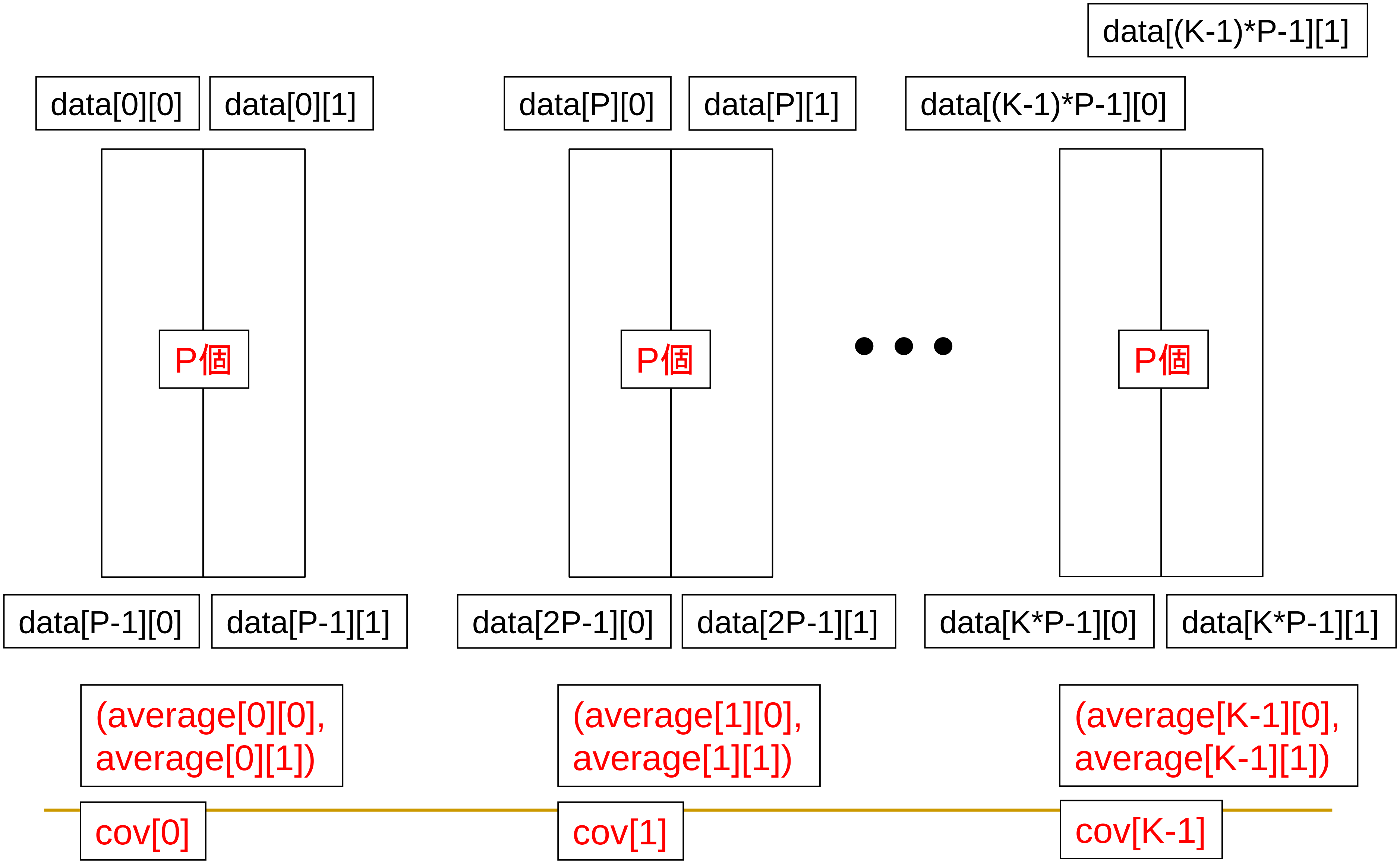
data：（N×D）個

```
# データ（全データ数，特徴数）
data = np.zeros( (N,D) , dtype=np.float32 )

# データの生成
for k in range(K):
    for p in range(P):
        data[k*P+p][0],data[k*P+p][1] =
                    np.random.multivariate_normal(average[k],cov[k]).T
```

np. random.multivariate_normal
平均ベクトルaverage[k]，分散共分散行列cov[k]
の多次元正規分布に従う乱数を生成

データの生成③


data[(K-1)*P-1][1]
data[0][0]
data[0][1]
data[P][0]
data[P][1]
data[(K-1)*P-1][0]
P個
P個
• • •
P個
data[P-1][0]
data[P-1][1]
data[2P-1][0]
data[2P-1][1]
data[K*P-1][0]
data[K*P-1][1]
(average[0][0], average[0][1])
(average[1][0], average[1][1])
(average[K-1][0], average[K-1][1])
cov[0]
cov[1]
cov[K-1]

グラフの表示①

```
# グラフの描画
plt.figure()

# 散布図をプロットする
for n in range(N):
    plt.scatter(data[n][0],data[n][1],color="red")

# ラベル
plt.xlabel("x",size=10)
plt.ylabel("y",size=10)
plt.title( "MND" )
```

plt.scatter（x,y）
散布図（x,y）の表示

散布図
x軸：data[n][0] （n=1,2,・・・,N）
y軸：data[n][1]

プロットの色（color）は赤（red）

x軸, y軸のラベル名の設定
フォントの大きさ（size）は10

タイトルの設定

グラフの表示②

X軸の範囲（－20～20）

Y軸の範囲（－20～20）

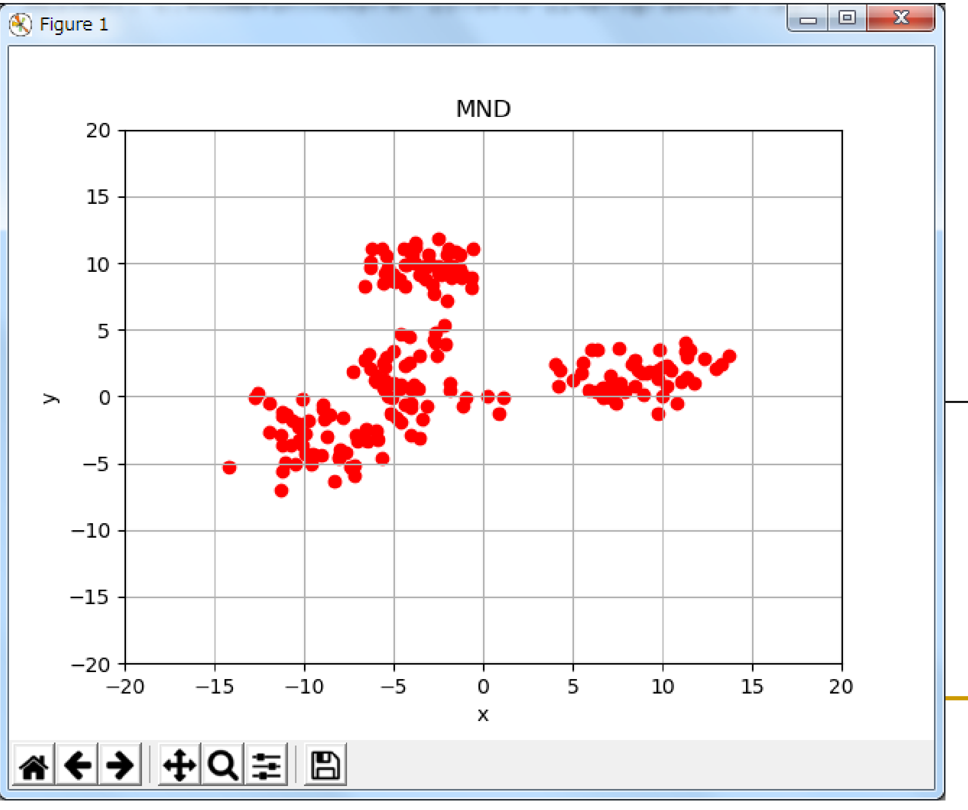
```
# 軸
plt.axis([-20.0,20.0,-20.0,20.0])
plt.grid(True)
```

グリッド線の表示

```
# グラフの表示
plt.show()
plt.clf()
```

表示→終了

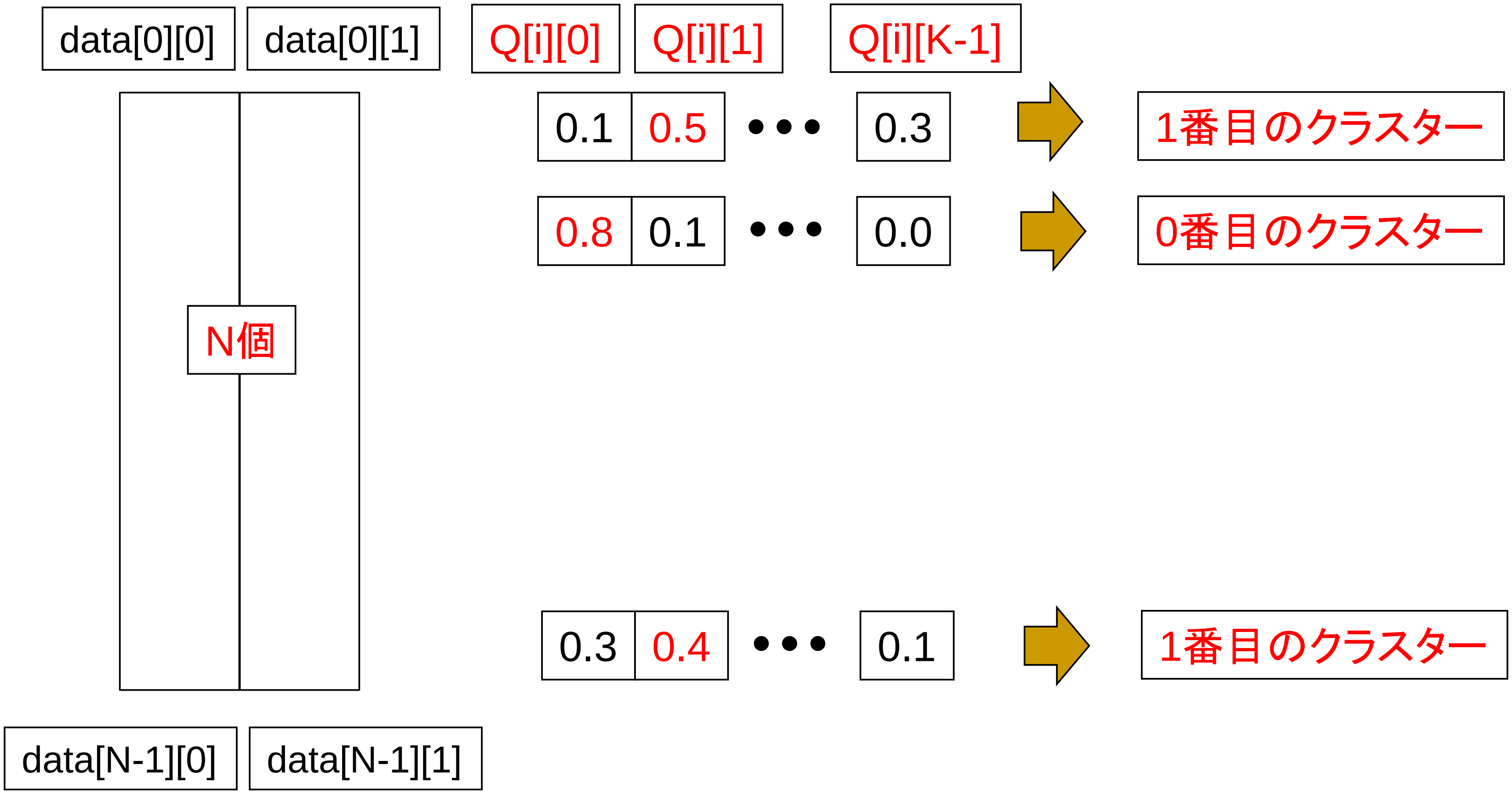
K=4，P=50で生成したデータ



宿題⑤

- N個のdata（2次元データ）は混合正規分布から生成されたデータと仮定します.
- EMアルゴリズムにより，K個の正規分布のパラメータ（重み，平均，標準偏差）を求めなさい.
- そしてQ_{ik}の値は，data[i]がk番目のクラスターに属する確率と考えます．従ってQ_{ik}の値が最大となるクラスターに各データは属すると考えることができます.
- すなわち，各データは，K個のクラスターの中でどれに属するかが決定できます（クラスタリング）

宿題⑤


data[0][0]
data[0][1]
Q[i][0]
Q[i][1]
Q[i][K-1]
0.1
0.5
•••
0.3
1番目のクラスター
0.8
0.1
•••
0.0
0番目のクラスター
N個
0.3
0.4
•••
0.1
1番目のクラスター
data[N-1][0]
data[N-1][1]

変数の初期化

```
# 初期化(Q値, 平均, 分散共分散, 重み)
Q = np.zeros((K, N),dtype=np.float64)
e_average = np.zeros((K,D),dtype=np.float64)
e_cov = np.zeros((K,D,D),dtype=np.float64)
e_lamda = np.zeros(K,dtype=np.float64)

for k in range(K):
    e_average[k] = random.choice(data)
    e_cov[k] = (np.cov(data , rowvar=0 , bias=1) )
    e_lamda[k] = 1/K

print( "平均 -> " , e_average )
print( "分散 -> " , e_cov )
print( "重み -> " , e_lamda )
```

Q値:(K×N)個

平均ベクトル:(K×D)個

分散共分散行列:(K×D×D)個

重み:K個

平均ベクトル:ランダムにdataから選択

分散共分散行列:
全データの分散共分散行列

重み:1/K

宿題⑤

EMアルゴリズムにより，パラメータQ値（Q），平均ベクトル（e_average），分散共分散行列（e_cov），重み（e_lamda）を推定しなさい

Q値が最大のクラスターを表示

```
for i in range(N):
    ans = np.argmax( Q[i,:] )
    print( i , ans )
```

np.argmax（配列）
配列の最大値の要素番号を返す

Q[i]の最大値の要素番号を返す

クラスタリングの結果の表示①

```
# グラフの描画
plt.figure()
plt.subplot(2,1,1)

# 散布図をプロットする（K=5まで対応）
plot_c = [ "red" , "orange" , "blue" , "green" , "pink" ]
plot_l = [ "class-1" , "class-2" , "class-3" , "class-4" , "clss-5" ]
for k in range(K):
    plt.scatter(data[k*P:(k+1)*P,0],y=data[k*P:(k+1)*P,1],
                color=plot_c[k],label=plot_l[k])

for k in range(K):
    plt.figtext((average[k][0]+20)/40,(average[k][1]+20)/40/2+0.5,
            plot_l[k],size=10)
```

plt.subplot(2,1,1)
plt.subplot(2,1,2)

正解（？）のクラスターの表示

color：プロットの色
label：プロットのラベル（凡例）

クラスタリングの結果の表示②

```
# ラベルの設定
plt.xlabel("x",size=10)
plt.ylabel("y",size=10)

# 軸の設定
plt.axis([-20.0,20.0,-20.0,20.0])
plt.grid(True)
plt.legend()
```

X軸のラベルの設定

Y軸のラベルの設定

X軸の範囲（ー20～20）

Y軸の範囲（ー20～20）

グリッド線の表示

凡例の表示

クラスタリングの結果の表示③

```
plt.subplot(2,1,2)

# 散布図をプロットする
for i in range(N):
    ans = np.argmax( Q[i,:] )
    plt.scatter(data[i][0],data[i][1],color=plot_c[ans])

for k in range(K):
     plt.figtext((e_average[k][0]+20)/40,(e_average[k][1]+20)/40/2,
                      plot_l[k],size=10)
```

Q[n]の最大値の要素番号
→所属するクラスターの番号

クラスタリングの結果の表示

クラスタリングの結果の表示④

```
# ラベルの設定
plt.xlabel('x',size=10)
plt.ylabel('y',size=10)

# 軸の設定
plt.axis([-20.0,20.0,-20.0,20.0])
plt.grid(True)

# 保存
plt.savefig("EM-result.png")
plt.show()
plt.clf()
```

X軸のラベルの設定

Y軸のラベルの設定

X軸の範囲（－20～20）

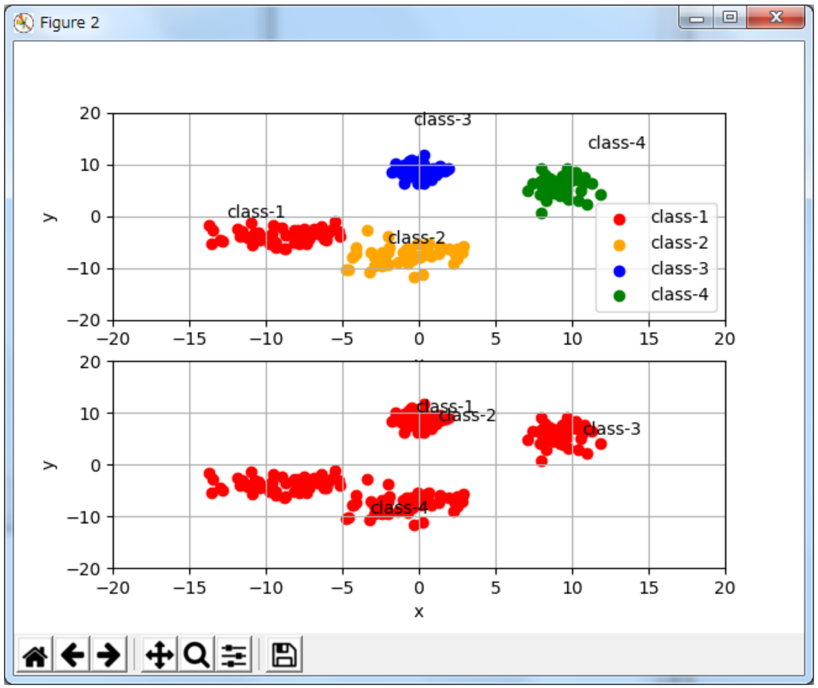
Y軸の範囲（－20～20）

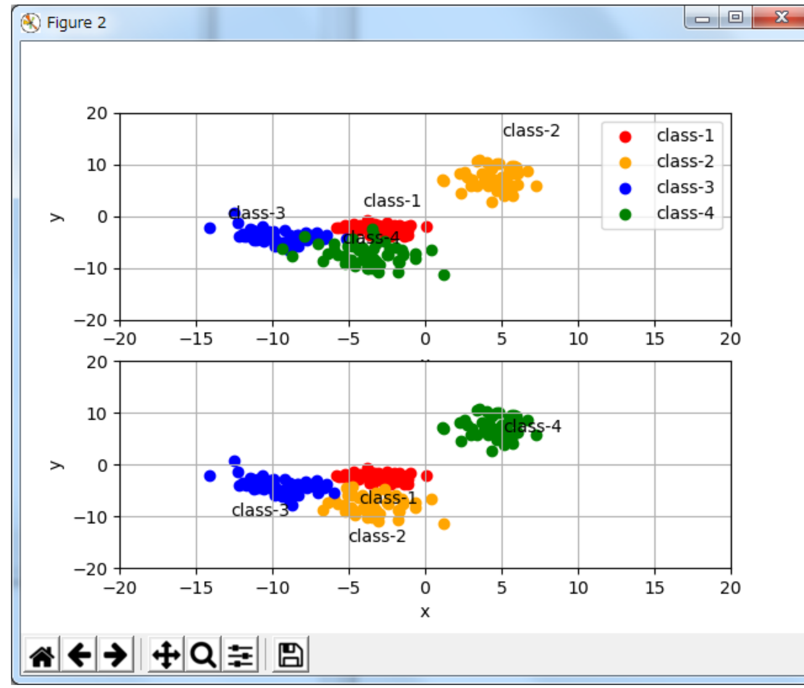
グリッド線の表示

「EM-result.png」に保存

表示→終了

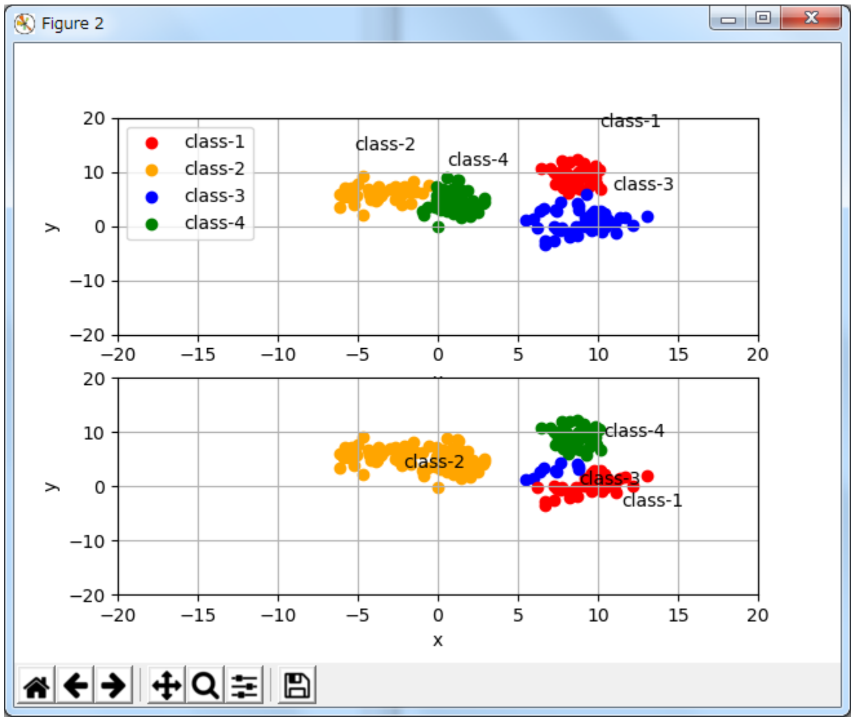
Q値が求まった場合

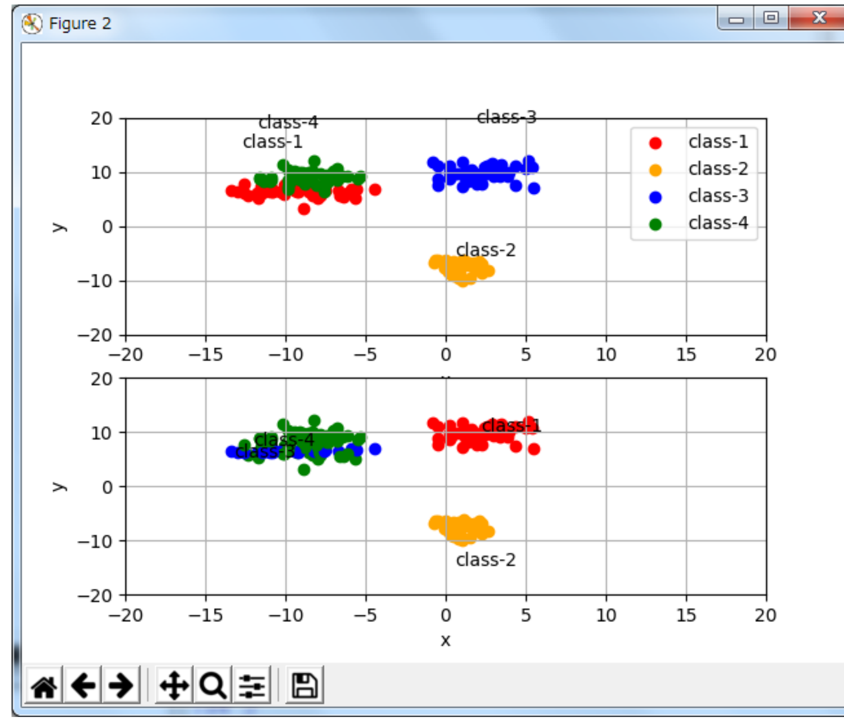




クラスターごとに色分けされる

クラスタリングの結果の例





EMアルゴリズムによるクラスタリング

- 宿題としておきますが，教師なし学習の講義の時に再度，説明します．試してみて下さい．

（本日の）参考文献

- 舟久保登：パターン認識，共立出版（1991）
- 石井健一郎他：わかりやすいパターン認識，オーム社（1998）
- 新納浩幸，Rで学ぶクラスタ解析，オーム社（2007）
- 杉山将：統計的機械学習，オーム社（2009）
- 浜本義彦：統計的パターン認識入門，森北出版（2009）