パターン認識と学習
教師あり学習(1)

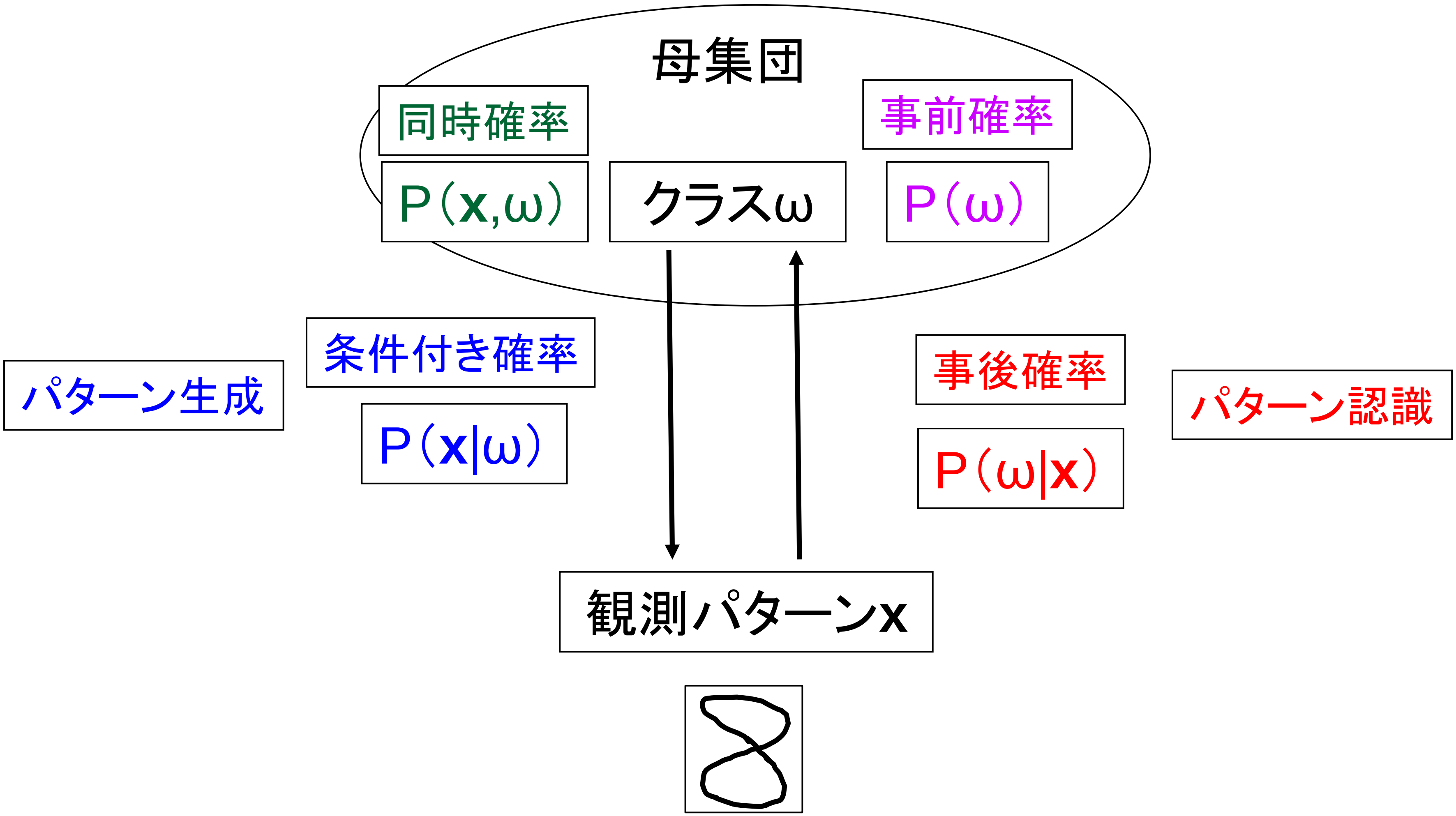
管理工学科
篠沢佳久

資料の内容

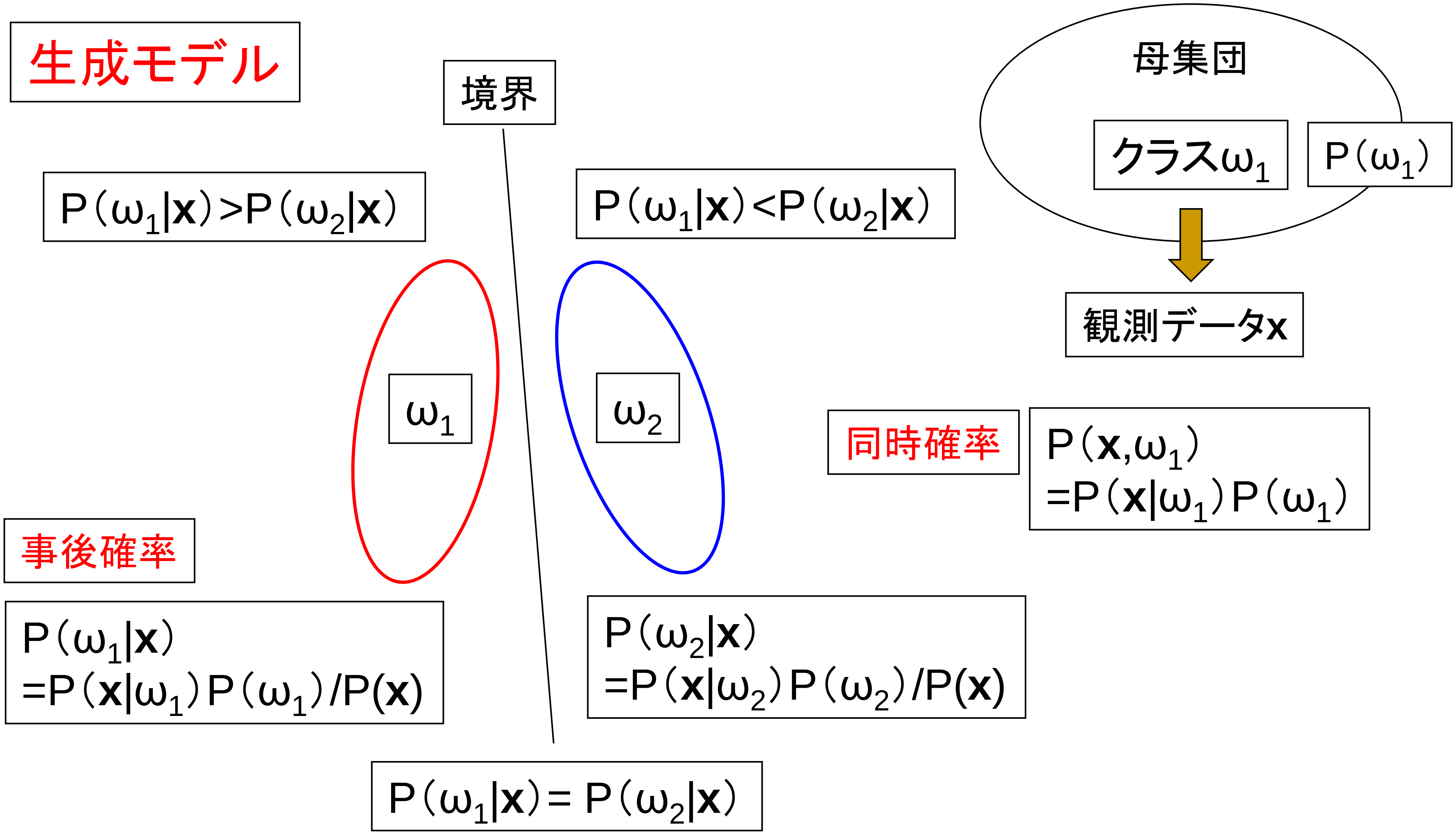
- 識別関数法
  - 線形識別関数
  - フィッシャーの線形判別
  - 特徴空間の変換

- 実習①（フィッシャーの線形判別）
- 実習②（固有顔）

生成モデルと識別モデル（復習）

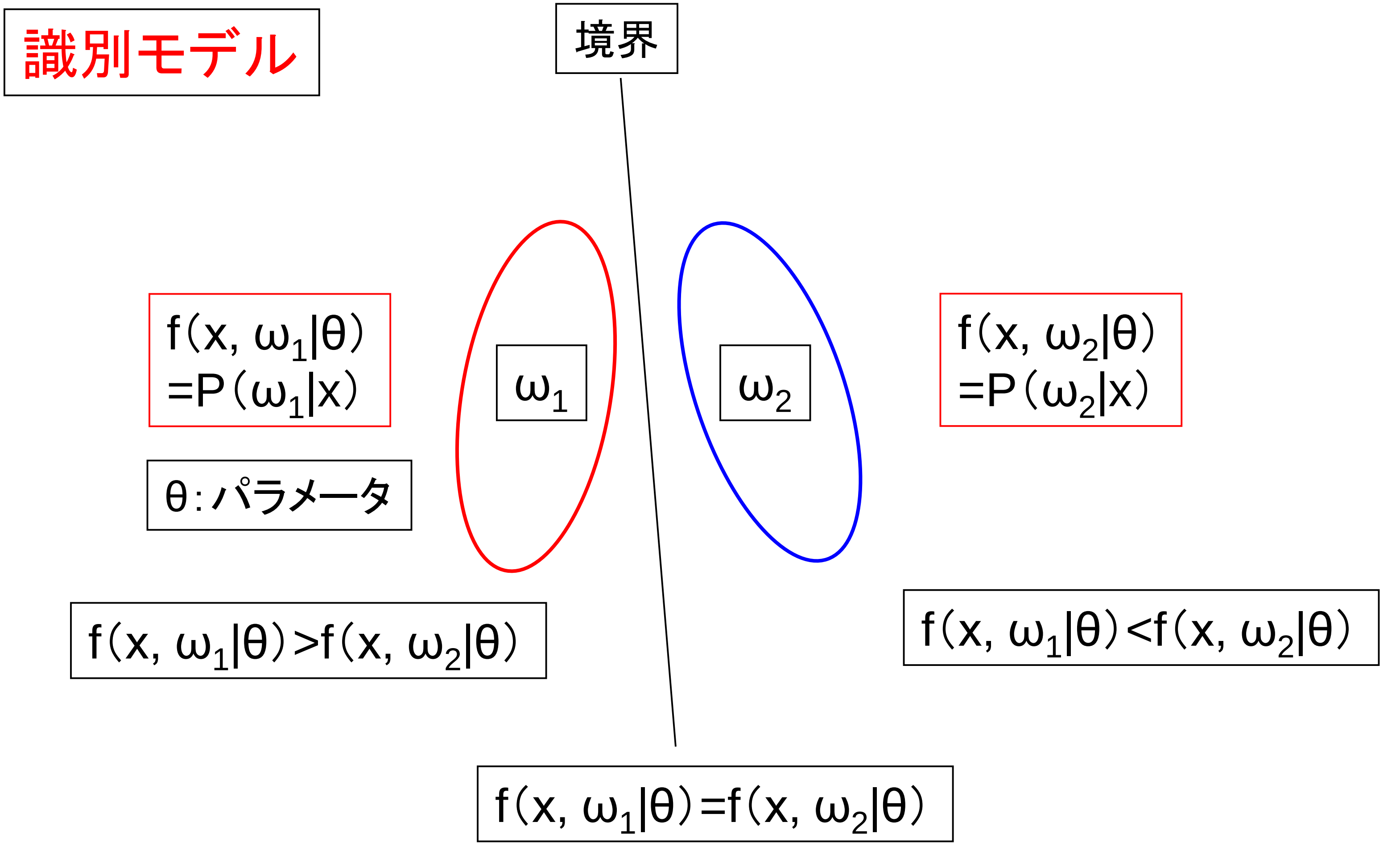


生成モデル



生成モデル
境界
母集団
クラスω1
P(ω1)
観測データx
P(ω1|x)>P(ω2|x)
P(ω1|x)<P(ω2|x)
ω1
ω2
同時確率
P(x,ω1)
=P(x|ω1)P(ω1)
事後確率
P(ω1|x)
=P(x|ω1)P(ω1)/P(x)
P(ω2|x)
=P(x|ω2)P(ω2)/P(x)
P(ω1|x)= P(ω2|x)

識別モデル



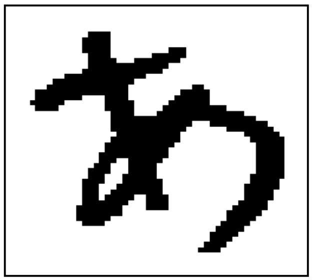
識別モデル
境界
$f(x, \omega_1|\theta)$
$=P(\omega_1|x)$
ω_1
ω_2
$f(x, \omega_2|\theta)$
$=P(\omega_2|x)$
θ:パラメータ
$f(x, \omega_1|\theta) > f(x, \omega_2|\theta)$
$f(x, \omega_1|\theta) < f(x, \omega_2|\theta)$
$f(x, \omega_1|\theta) = f(x, \omega_2|\theta)$

識別関数法

線形識別関数

パターン認識の基本的な流れ

前処理部　特徴抽出部　識別部



入力パターン

⇒ 前処理 ⇒ 特徴抽出 ⇒ 認識処理 ⇒「あ」

最近傍法（k近傍法）

最尤法
ベイズ決定則

識別関数法

識別関数法

- クラスω_i（i=1,2,･･･,c）
- 各クラスに対応した関数 g_i
- 特徴ベクトル$\mathbf{x}$（d次元）の属するクラスを調べる

$$\max_{i=1,2\cdots,c}\{g_i(\mathbf{x})\} = g_k(\mathbf{x}) \Rightarrow \mathbf{x} \in \omega_k$$

- 関数 g_i を識別関数と呼ぶ

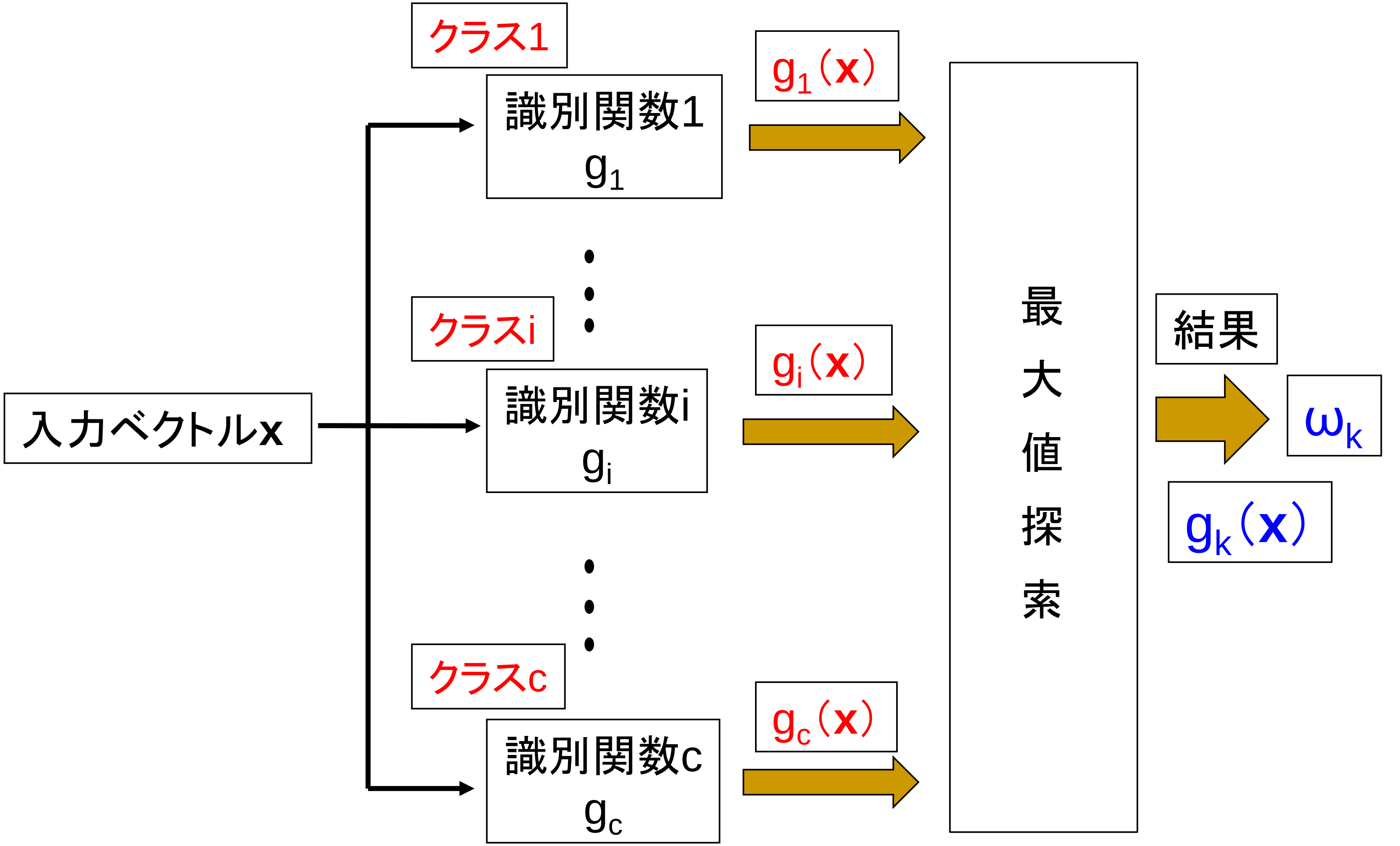
識別関数法のアルゴリズム①

```
max = －∞

for( i = 0 ; i < c ; i++ ) {
    val = gi(x)      識別関数の計算
    if( val > max ) {
        max = val
        answer = i
    }
}
クラス answer が認識結果
```

c ： クラスの総数
g_i ： クラス i の識別関数
x ： 調べたいパターンの特徴ベクトル

識別関数法のアルゴリズム②



ベイズ決定則(復習)①

- 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

$$
\begin{aligned}
\max_{i=1,2,\cdots,c}\{p(\omega_i\,|\,\mathbf{x})\} &= \max_{i=1,2,\cdots,c}\left\{\frac{p(\mathbf{x}\,|\,\omega_i)\,p(\omega_i)}{p(\mathbf{x})}\right\}\\
&= \max_{i=1,2,\cdots,c}\left\{p(\mathbf{x}\,|\,\omega_i)\,p(\omega_i)\right\}\\
&= p(\omega_k\,|\,\mathbf{x}) \Rightarrow \mathbf{x}\in\omega_k
\end{aligned}
$$

ベイズ決定則(復習)

- 確率密度関数に正規分布を仮定(d次元)

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

$\Downarrow$

識別関数

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

対数をとると

$$g_i(\mathbf{x}) = -\frac{1}{2} (\mathbf{x} - \mathbf{m}_i)^t \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i) - \frac{d}{2} \log 2\pi - \frac{1}{2} \log |\Sigma_i| + \log p(\omega_i)$$

$g_i(\mathbf{x})$が最大となるクラスω_iを認識結果とする

線形識別関数

- 識別関数
  - 重み係数 w_{i0}, w_{i1}, ･･･,w_{id}

$$g_i(\mathbf{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{id}x_d$$

$$= w_{i0} + \sum_{j=1}^{d} w_{ij}x_j \quad \boxed{\text{線形識別関数}}$$

  - ベクトル表記

$$\mathbf{x} = (x_1, \cdots, x_d)^t$$

$$\mathbf{w}_i = (w_{i1}, \cdots, w_{id}) \quad \boxed{\text{重みベクトル}}$$

$$g_i(\mathbf{x}) = w_{i0} + \mathbf{w}^t\mathbf{x}$$

線形識別関数

- 別に表記すると…

$$g_i(\mathbf{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{id}x_d$$

$$= w_{i0} + \sum_{j=1}^{d} w_{ij}x_j$$

$$\mathbf{x} = (1, x_1, \cdots x_d)^t$$

$$\mathbf{w}_i = (w_{i0}, w_{i1}, \cdots, w_{id})$$

$$g_i(\mathbf{x}) = \mathbf{w}_i^t\mathbf{x}$$

線形識別関数のアルゴリズム

```
max = －∞

for( i = 0 ; i < c ; i++ ) {
    val = 0
    for( j = 1 ; j <= d ; j++ ){
        val += w[ i ][ j ] * x[ j ]
    }
    val += w[ i ][ 0 ]
    if( val > max ) {
        max = val
        answer = i
    }
}
クラス answer が認識結果
```

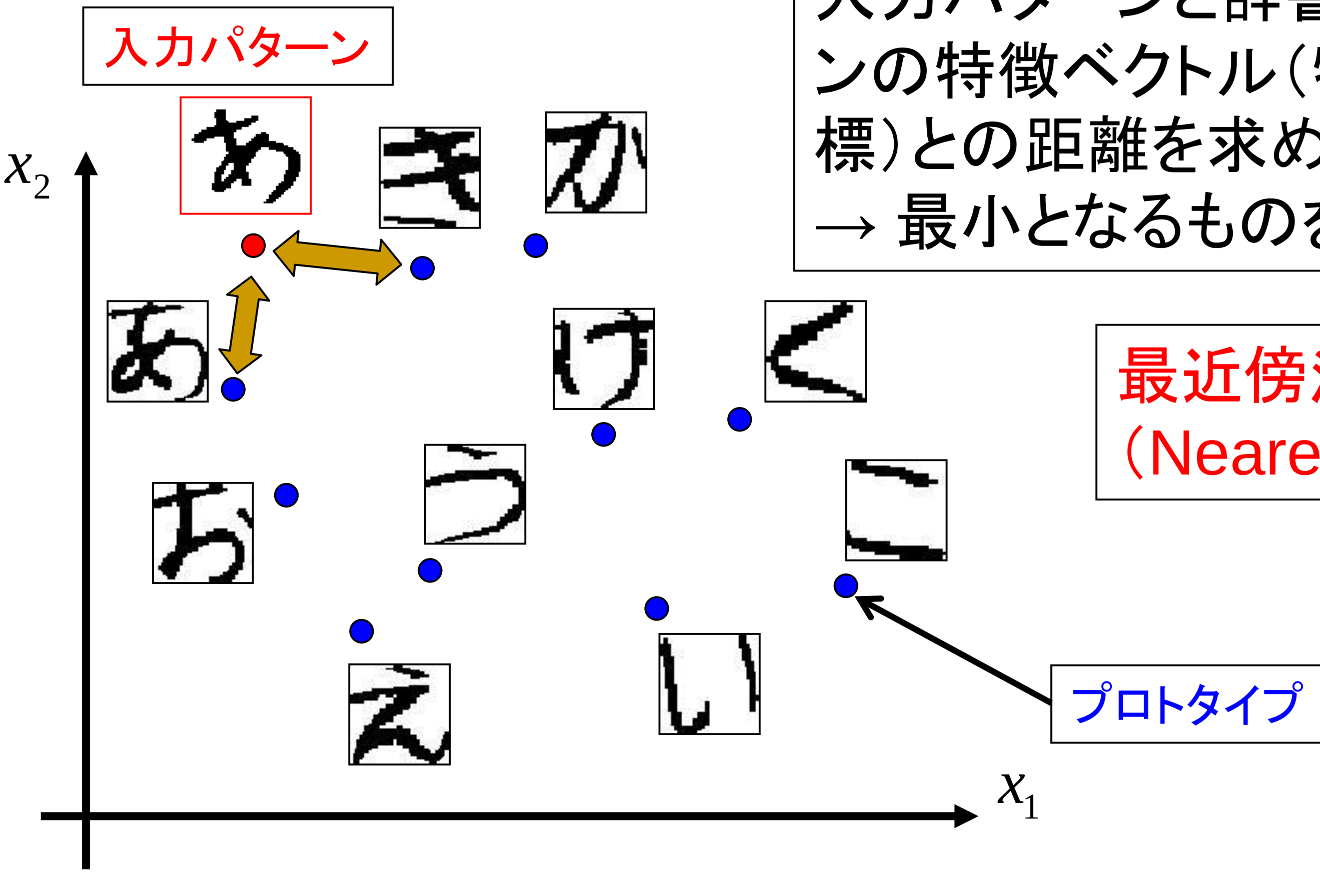
線形識別関数の計算

c ： クラスの総数
w[i][j] ： クラス i の識別関数の j番目の重み係数
x[j] ： 特徴ベクトルの j番目の要素

最近傍法（復習）①

特徴空間上において，
入力パターンと辞書の全てのパターンの特徴ベクトル（特徴空間上の座標）との距離を求める
→ 最小となるものを認識結果

最近傍法
（Nearest Neighbor）



最近傍法（復習）②

- クラス ω_i（i=1,2,･･･,c）
- 各クラスのプロトタイプ $\mathbf{p}_i$（d次元）
- 特徴ベクトル $\mathbf{x}$の属するクラスを調べる

$$\min_{i=1,2,\cdots,c} \left\| \mathbf{x} - \mathbf{p}_i \right\| = \left\| \mathbf{x} - \mathbf{p}_k \right\| \Rightarrow \mathbf{x} \in \omega_k$$

距離が最も近いプロトタイプを認識結果とする

最近傍法と線形識別関数

最近傍法

$$\min_{i=1,2,\cdots,c} \|\mathbf{x}-\mathbf{p}_i\| = \|\mathbf{x}-\mathbf{p}_k\| \Rightarrow \mathbf{x} \in \omega_k$$

$$\|\mathbf{x}-\mathbf{p}_i\|^2 = \|\mathbf{x}\|^2 - 2\mathbf{p}_i^t\mathbf{x} + \|\mathbf{p}_i\|^2$$

最小

||x||は全てのクラスで同じ

$$g_i(\mathbf{x}) = \mathbf{p}_i^t\mathbf{x} - \frac{1}{2}\|\mathbf{p}_i\|^2$$

最大

線形識別関数と等価

$$g_i(\mathbf{x}) = w_{i0} + \mathbf{w}^t\mathbf{x}$$

$$\mathbf{w}_i = \mathbf{p}_i$$

$$w_{i0} = -\frac{1}{2}\|\mathbf{p}_i\|^2$$

その他の線形識別関数

識別関数

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^t \Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i) - \frac{d}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_i| + \log p(\omega_i)$$

分散共分散行列を単位行列と仮定

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^t(\mathbf{x}-\mathbf{m}_i) + \log p(\omega_i)$$

事前確率は各クラスで同じ

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^t(\mathbf{x}-\mathbf{m}_i)$$

最小

$$g_i(\mathbf{x}) = \mathbf{m}_i^t\mathbf{x} - \frac{1}{2}\|\mathbf{m}_i\|$$

最大

線形識別関数と等価

$$g_i(\mathbf{x}) = w_{i0} + \mathbf{w}^t\mathbf{x}$$

$$\mathbf{w}_i = \mathbf{m}_i$$

$$w_{i0} = -\frac{1}{2}\|\mathbf{m}_i\|^2$$

重みベクトルの決め方

- 重みベクトル
  - 最近傍法

→ 各プロトタイプの特徴ベクトル

$$g_i(\mathbf{x}) = w_{i0} + \mathbf{w}^t\mathbf{x}$$
$$\mathbf{w}_i = \mathbf{p}_i$$
$$\mathbf{w}_{i0} = -\frac{1}{2}\|\mathbf{p}_i\|^2$$

  - 各クラスの平均ベクトル

$$g_i(\mathbf{x}) = w_{i0} + \mathbf{w}^t\mathbf{x}$$
$$\mathbf{w}_i = \mathbf{m}_i$$
$$\mathbf{w}_{i0} = -\frac{1}{2}\|\mathbf{m}_i\|^2$$

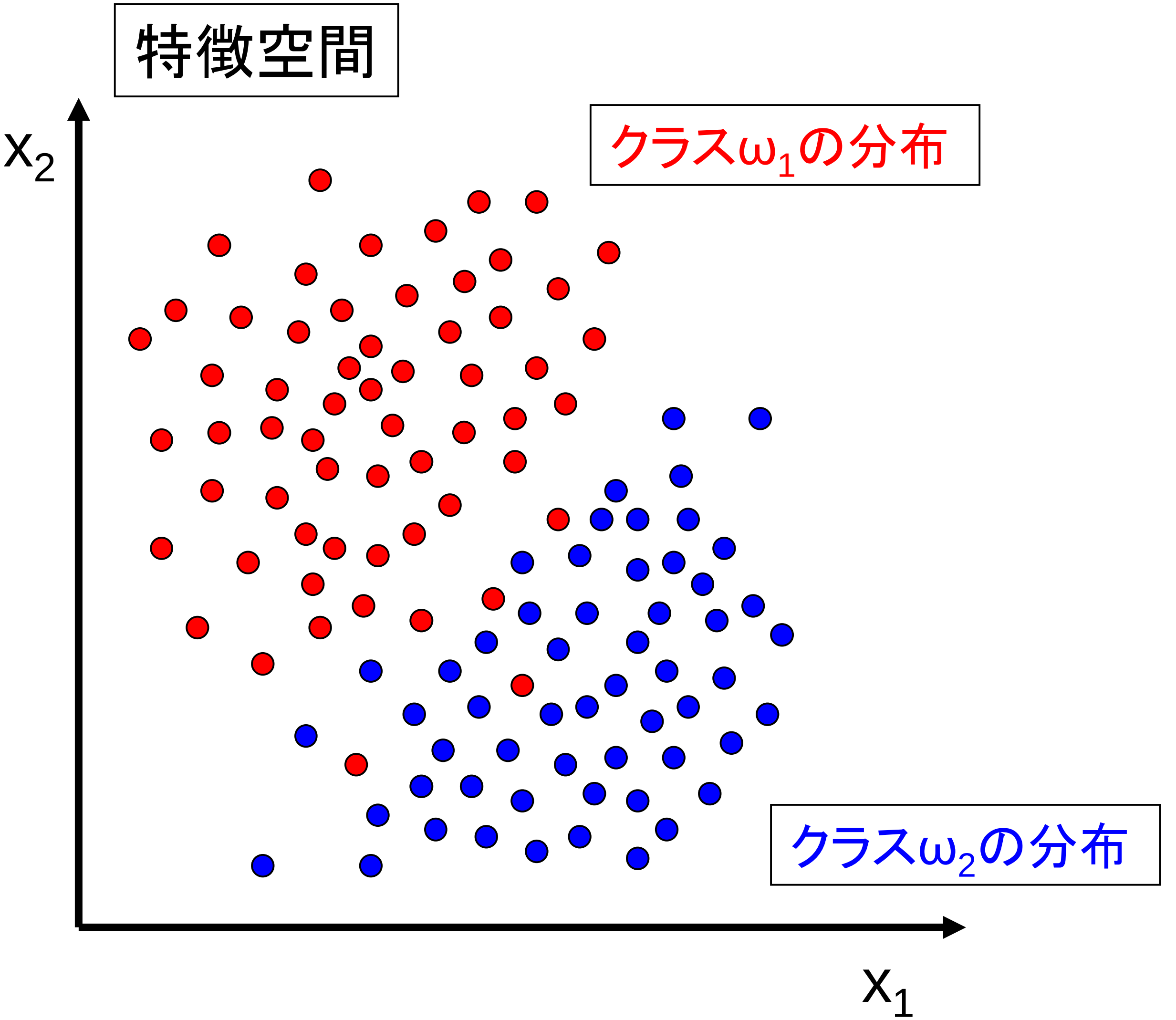
↓

- フィッシャーの線形判別（特徴空間の変換）
- 学習
  - デルタルール，パーセプトロン

フィッシャーの線形判別

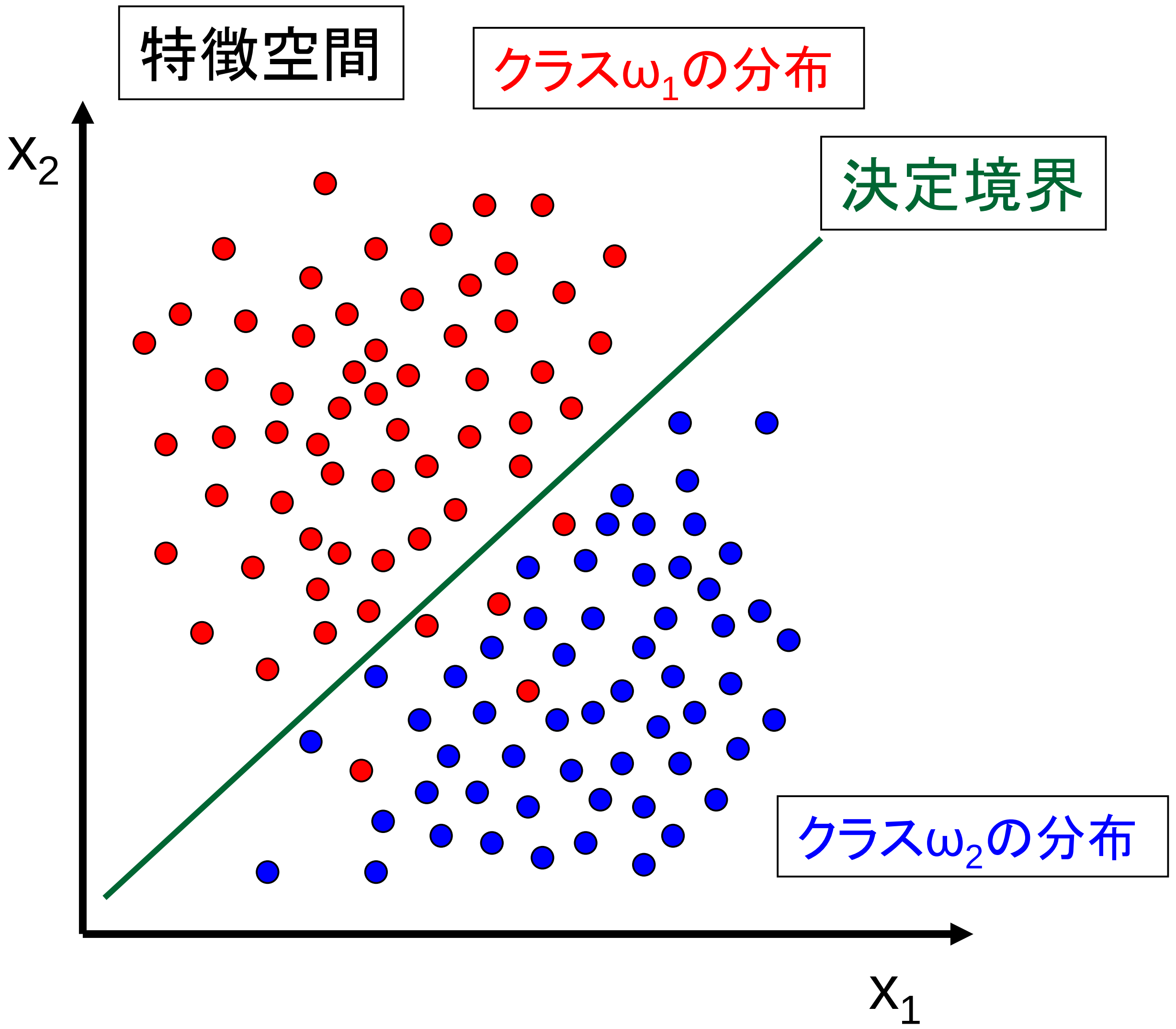
正準判別法

二群の判別



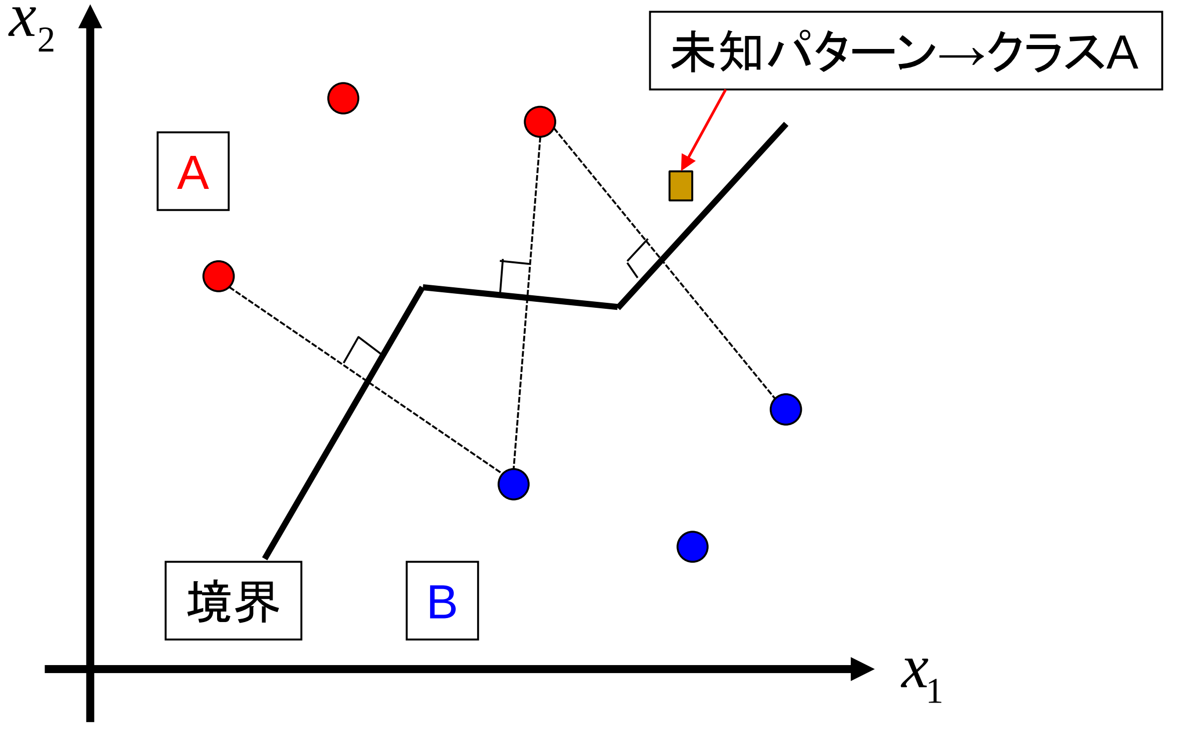
クラスω_1とクラスω_2を分離するためには？

決定境界

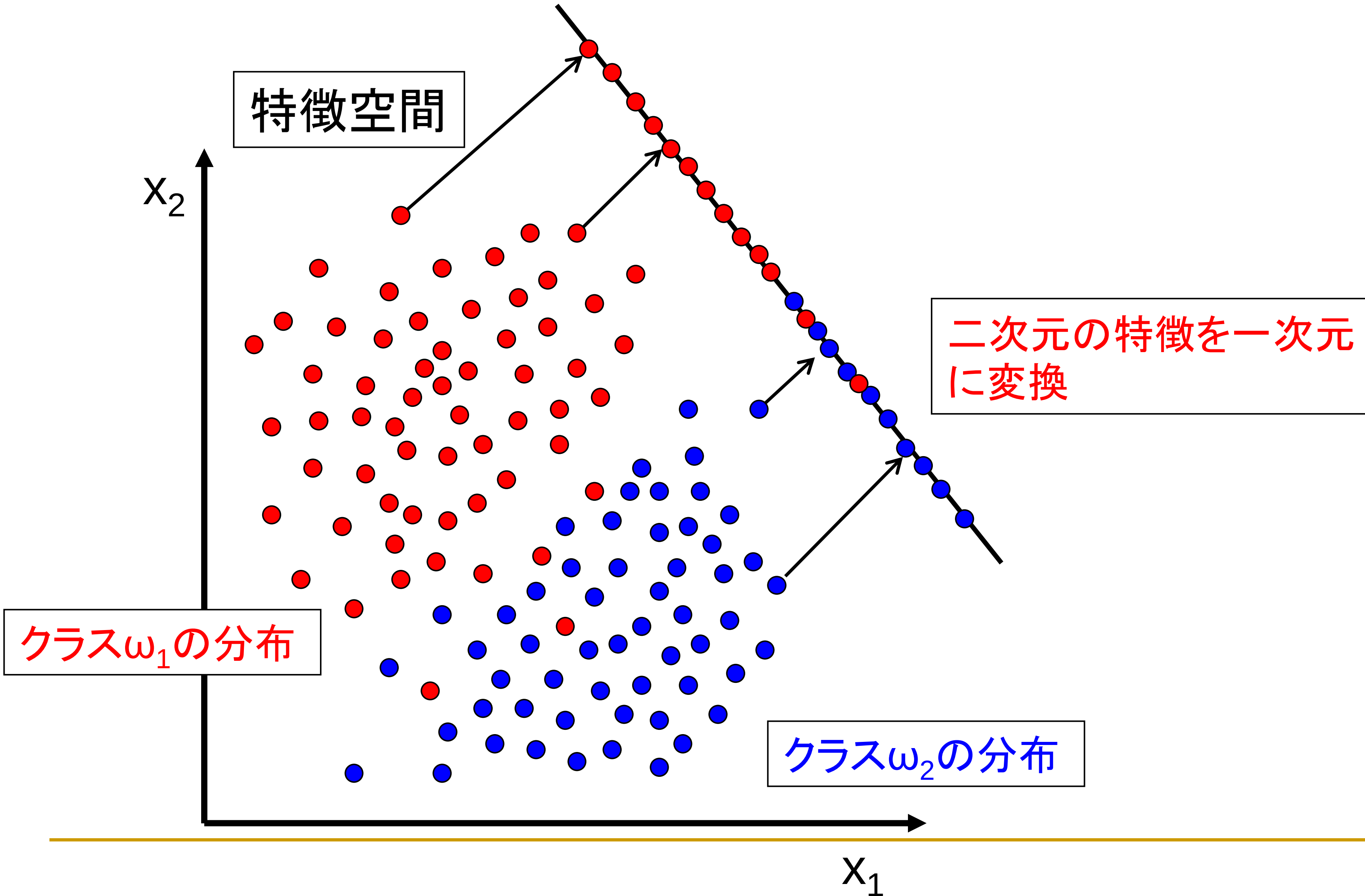


ボロノイ図(復習)

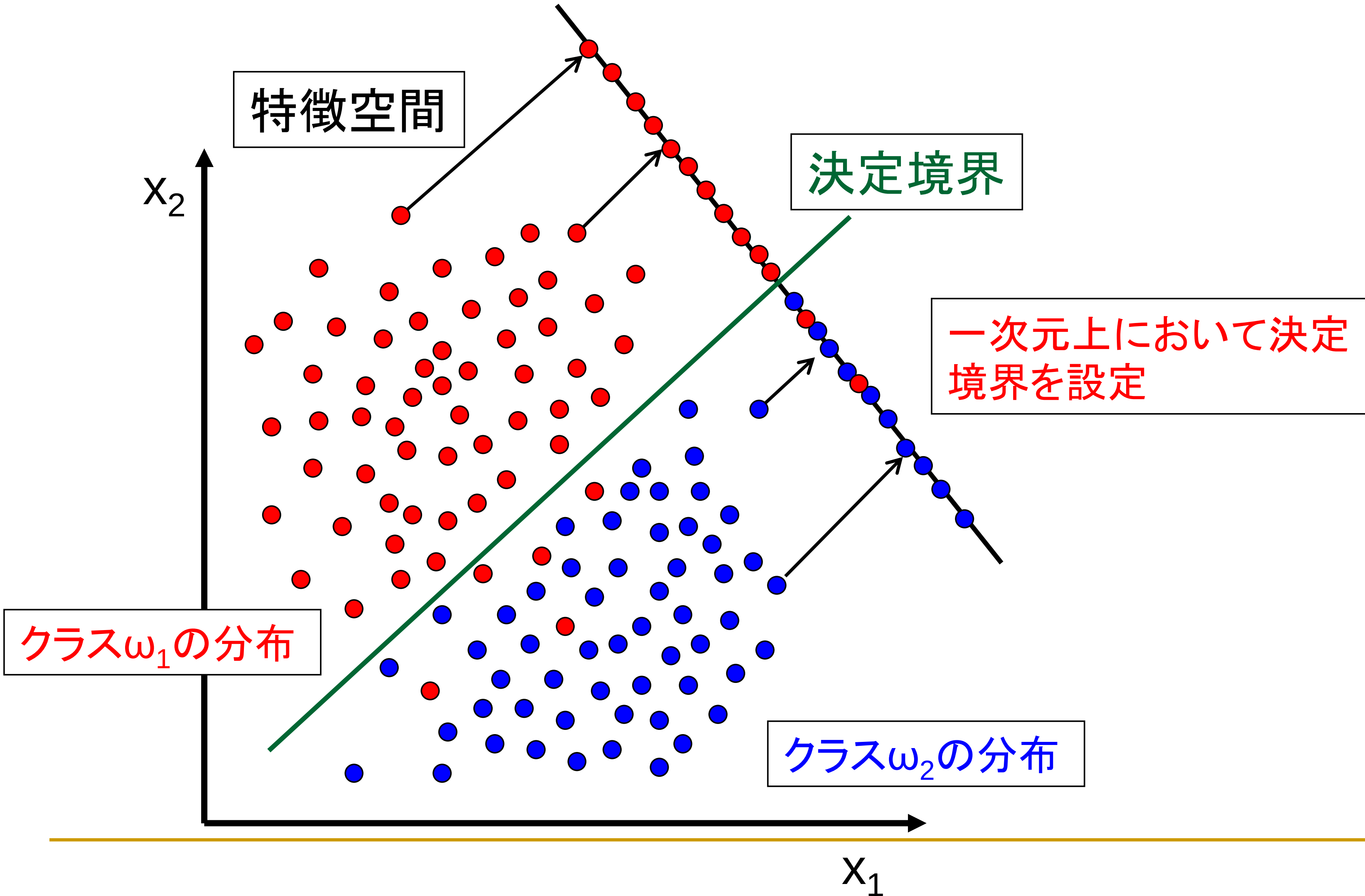
- 複数個のプロトタイプを利用
  →境界が複雑に



特徴空間の変換①



特徴空間の変換②


特徴空間
決定境界
一次元上において決定
境界を設定
x_2
x_1
クラスω_1の分布
クラスω_2の分布

フィッシャーの線形判別の原理

- 二クラスω_1, ω_2
- 各パターンの特徴ベクトル **x**（d次元）
- d次元のパターン **x** を一次元に変換

$$y = A^t \mathbf{x}$$

- Aはd×1の行列?（ベクトル）
- 変換後の値によって二つのクラスを分離

クラス内変動とクラス間変動

変動行列

d×d 行列

$$S_i = \sum_{\mathbf{x}\in\chi_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t$$

χ_i ：クラスω_iの学習パターンの集合
$\mathbf{m}_i$ ：クラスω_iの平均ベクトル

クラス内変動行列

$$S_w = S_1 + S_2$$
$$= \sum_{i=1,2}\sum_{\mathbf{x}\in\chi_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^t$$

$\mathbf{m}$ ：全パターンの平均ベクトル
n_i ：クラスω_iの学習パターン数
n ： 全学習パターン数

クラス間変動行列

$$S_B = \sum_{i=1,2} n_i(\mathbf{m}_i-\mathbf{m})(\mathbf{m}_i-\mathbf{m})^t$$
$$= \frac{n_1 n_2}{n}(\mathbf{m}_1-\mathbf{m}_2)(\mathbf{m}_1-\mathbf{m}_2)^t$$

$$\mathbf{m} = \frac{n_1\mathbf{m}_1 + n_2\mathbf{m}_2}{n}$$
$$n = n_1 + n_2$$

変換後のクラス内変動とクラス間変動①

- d次元のパターンxを一次元に変換

$$y = A^t \mathbf{x}$$

変換後の平均ベクトル*

$$\tilde{m}_i = \frac{1}{n_i} \sum_{y \in Y_i} y = \frac{1}{n_i} \sum_{\mathbf{x} \in \chi_i} A^t \mathbf{x} = A^t \mathbf{m}_i$$

変換後の変動行列*

$$\tilde{S}_i = \sum_{y \in Y_i} (y - \tilde{m}_i)^2 = A^t S_i A$$

変換後のクラス内変動行列*

$$\tilde{S}_w = \tilde{S}_1 + \tilde{S}_2$$

$$= \sum_{i=1,2} \sum_{y \in Y_i} (y - \tilde{m}_i)^2 = A^t S_w A$$

変換後のクラス間変動行列*

$$\tilde{S}_B = \sum_{i=1,2} n_i (\tilde{m}_i - \tilde{m})^2 = A^t S_B A$$

変換後の全パターンの平均

$$\tilde{m} = \frac{1}{n} \sum_{y \in Y_1, Y_2} y = A^t \mathbf{m}$$

*この場合はスカラー

変換後のクラス内変動とクラス間変動②

変換後の変動行列

$$\tilde{S}_i = \sum_{y \in Y_i} (y - \tilde{m}_i)^2 = A^t S_i A$$

$$= n_i \tilde{\sigma}_i^2$$

変換後の分散 $\tilde{\sigma}^2$

$$\tilde{\sigma}_i^2 = \frac{1}{n_i} \sum_{y_i \in Y_i} (y_i - \tilde{m}_i)^2$$

変換後のクラス内変動行列

$$\tilde{S}_w = \tilde{S}_1 + \tilde{S}_2$$

$$= \sum_{i=1,2} \sum_{y \in Y_i} (y - \tilde{m}_i)^2 = A^t S_w A$$

$$= n_1 \tilde{\sigma}_1^2 + n_2 \tilde{\sigma}_2^2$$

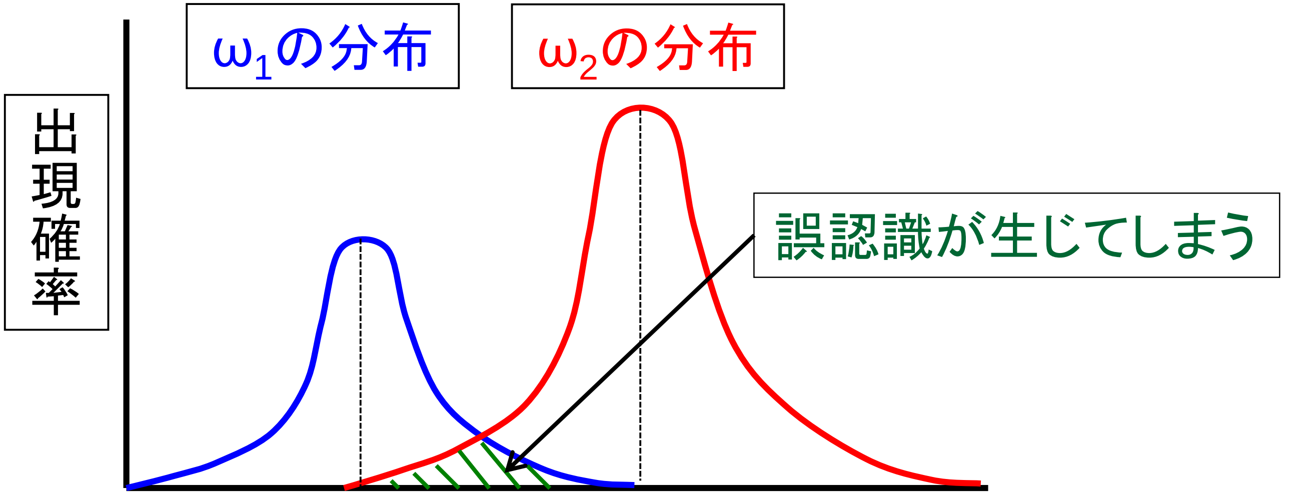
変換後のクラス間変動行列

$$\tilde{S}_B = \sum_{i=1,2} n_i (\tilde{m}_i - \tilde{m})^2 = A^t S_B A$$

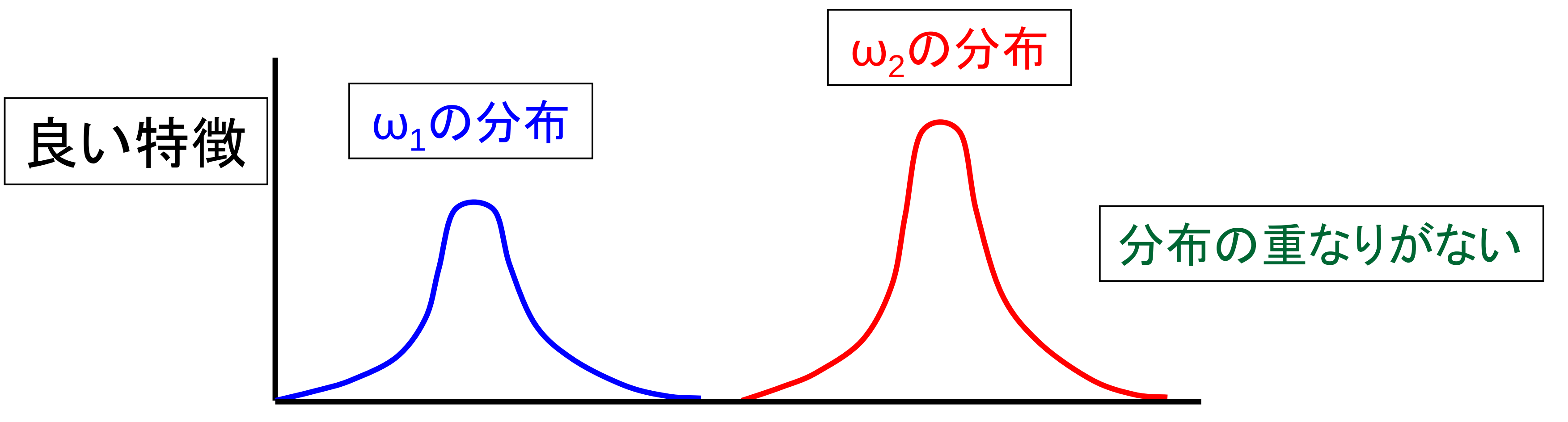
$$\tilde{m} = \frac{n_1 \tilde{m}_1 + n_2 \tilde{m}_2}{n}$$

$$\tilde{S}_B = \frac{n_1 n_2}{n} (\tilde{m}_1 - \tilde{m}_2)^2$$

良い特徴とは①

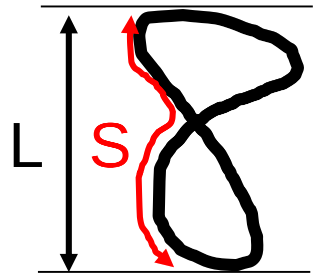


出現確率
ω_1の分布
ω_2の分布
誤認識が生じてしまう



良い特徴
ω_1の分布
ω_2の分布
分布の重なりがない

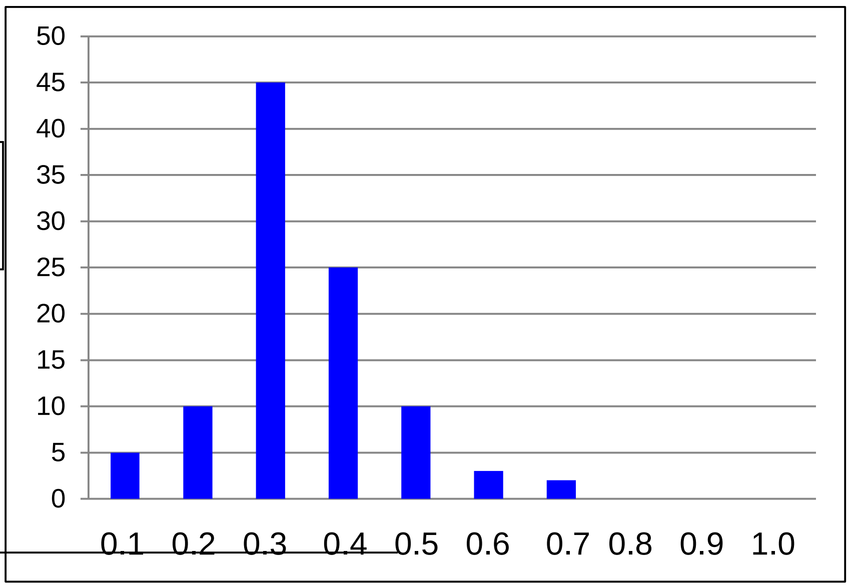
誤認識の可能性①



$$x = \frac{L}{S}$$

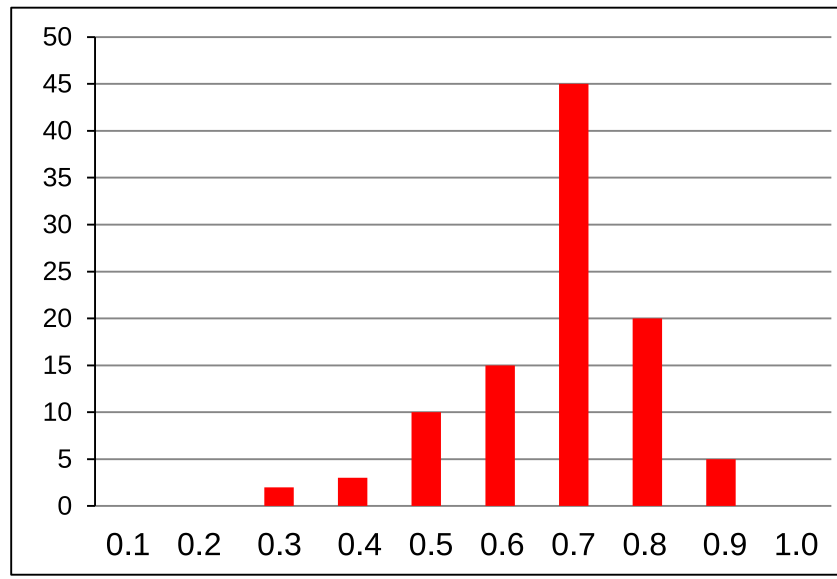
「8」の分布

度数



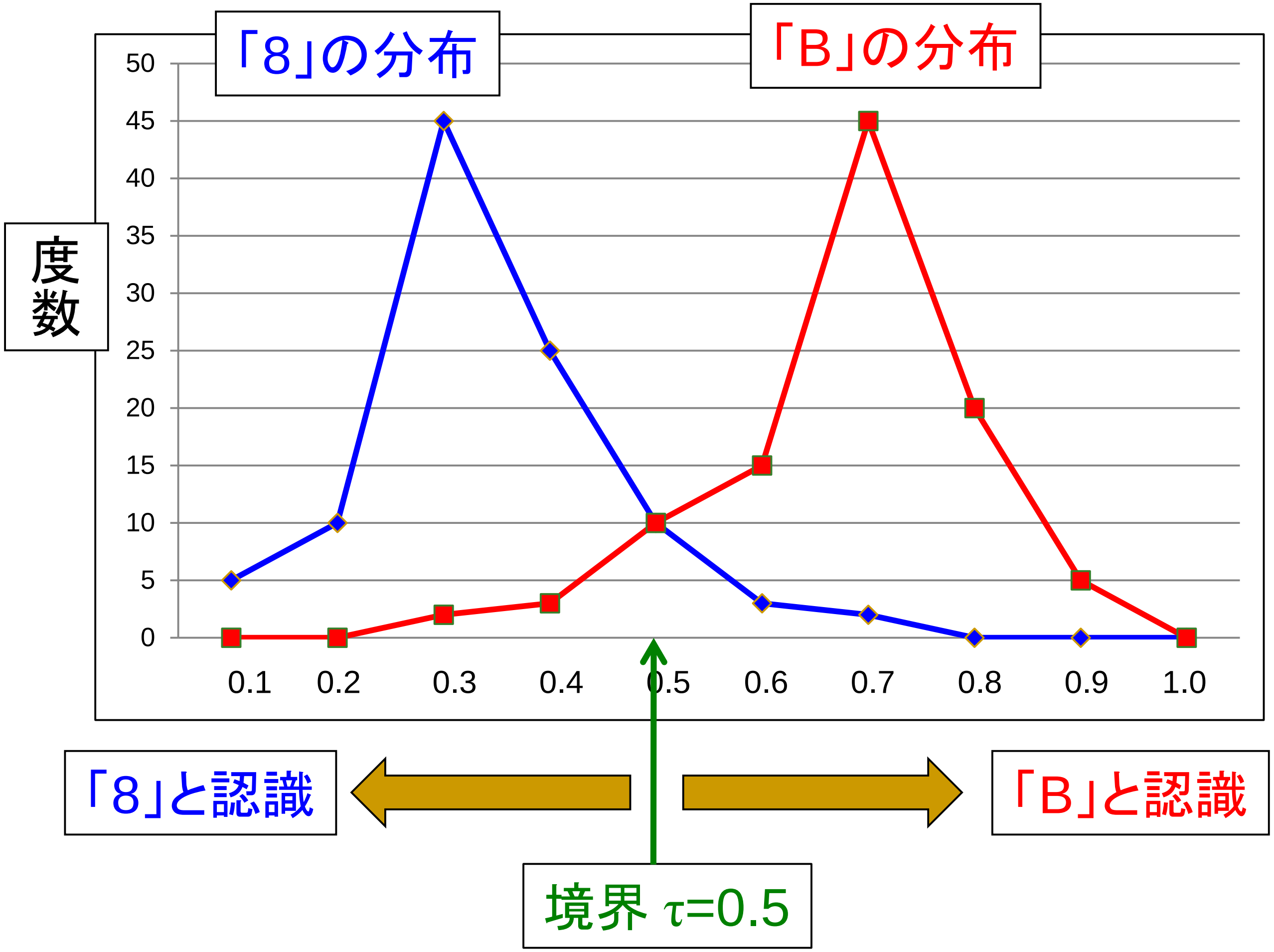
直線を示す特徴 x

「B」の分布



直線を示す特徴 x

誤認識の可能性②

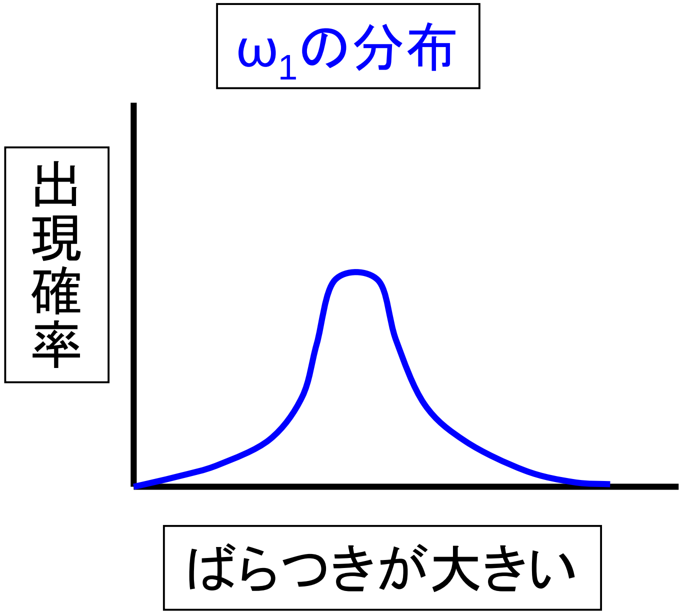


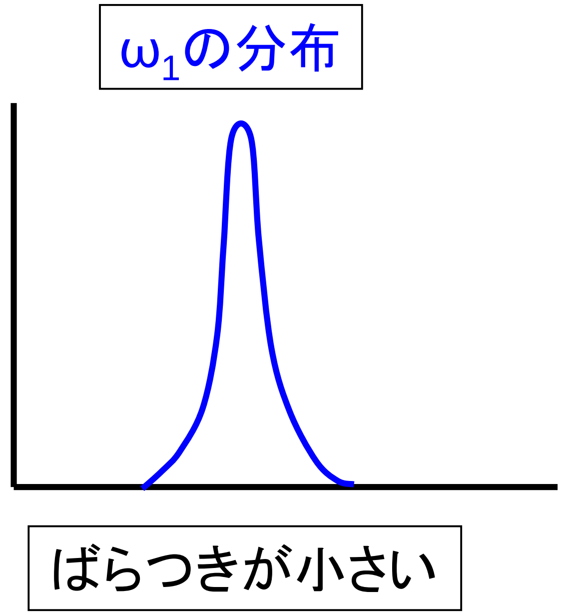
境界 τ=0.5 とした場合

「8」において，0.5以上の特徴が出現する確率は0.15
→ 0.15 の割合で「8」を「B」と誤認識する可能性がある

「B」において，0.5以下の特徴が出現する確率は0.15
→ 0.15 の割合で「B」を「8」と誤認識する可能性がある

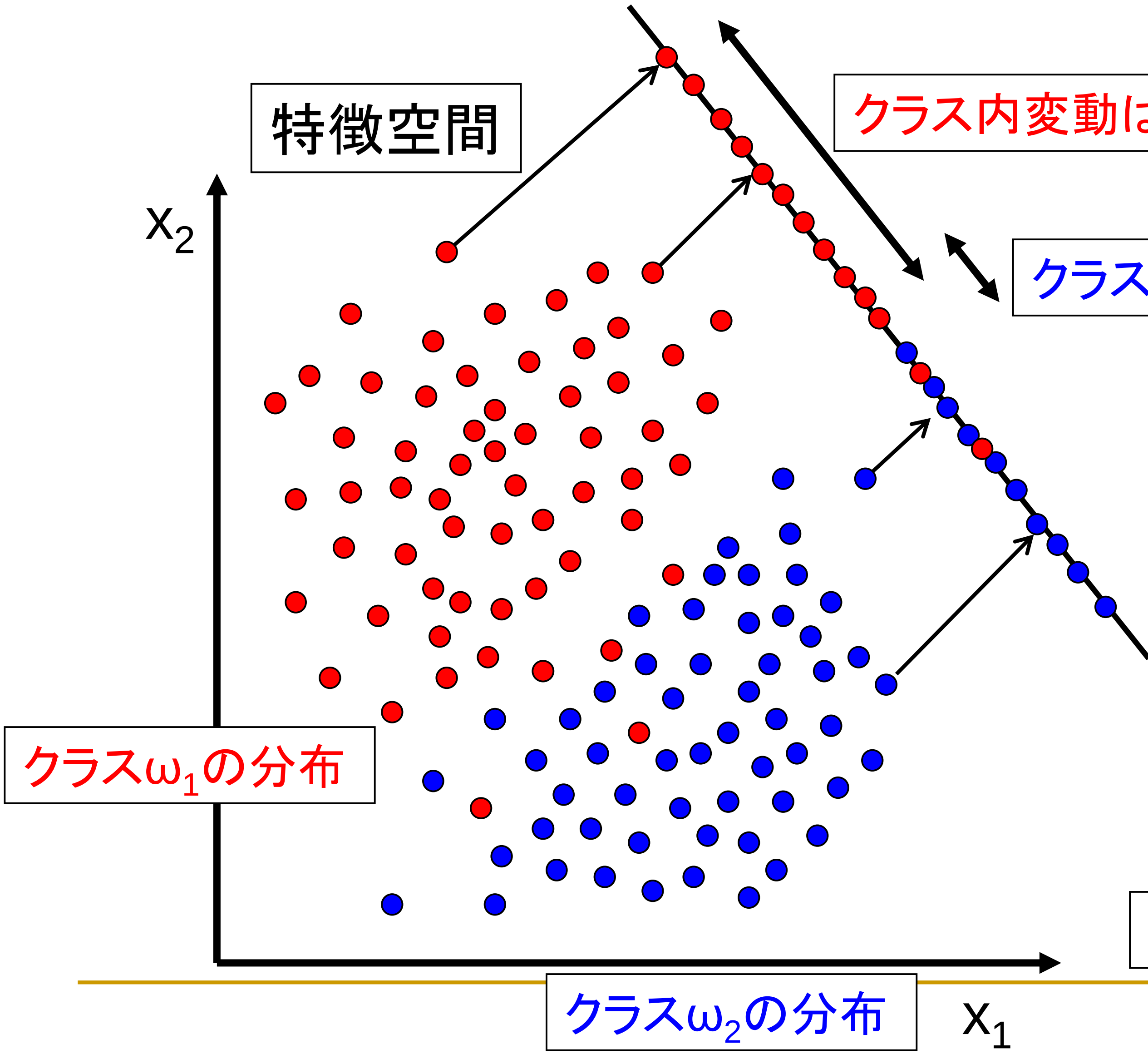
良い特徴とは②





- クラス内の変動
  - 平均値付近を中心にばらつきが小さい
- クラス間の変動
  - 分布が重なっていない

フィッシャーの評価基準



クラス内変動は小さく

クラス間変動は大きく

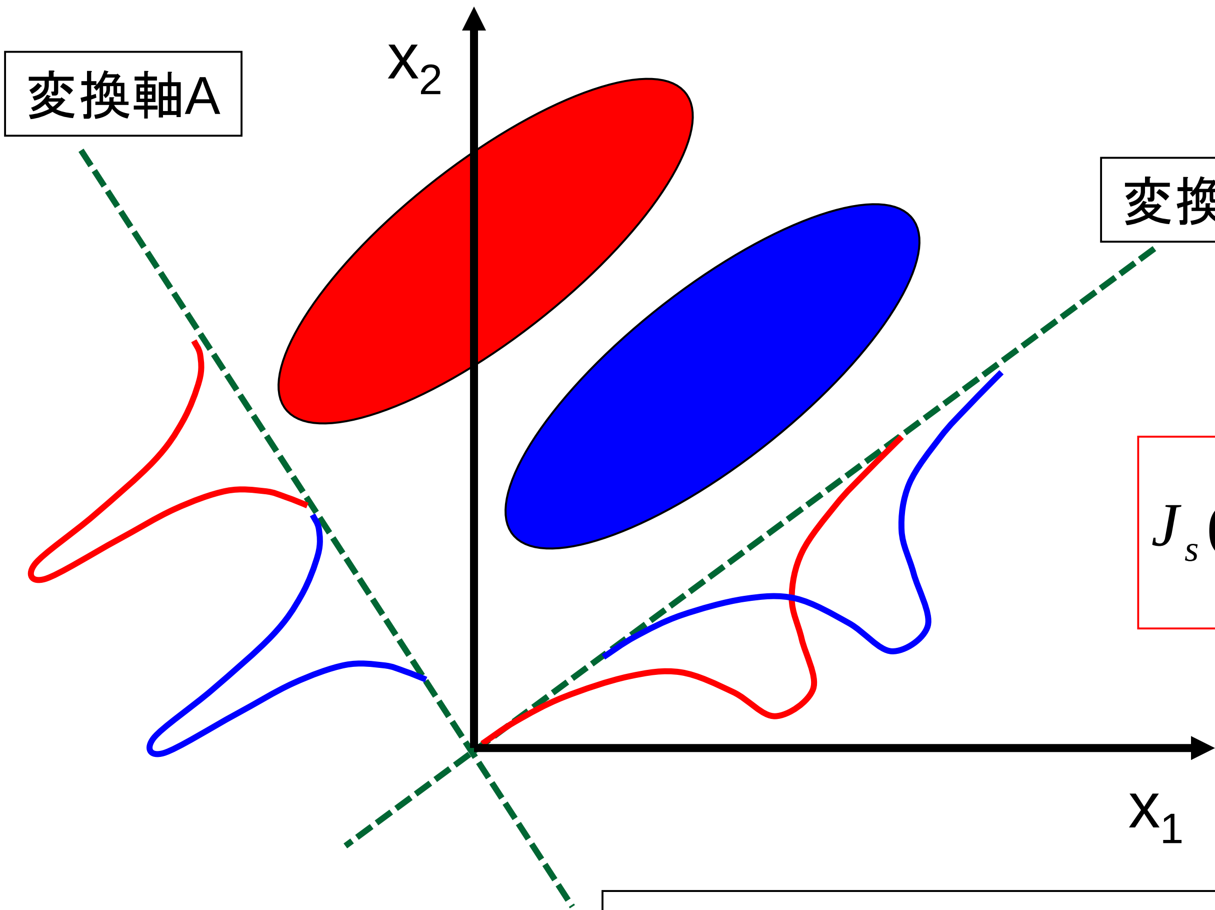
$$\tilde{S}_w = A^t S_w A = n_1 \tilde{\sigma}_1^2 + n_2 \tilde{\sigma}_2^2$$

$$\tilde{S}_B = A^t S_B A = \frac{n_1 n_2}{n}(\tilde{m}_1 - \tilde{m}_2)^2$$

$$J_s(A) = \frac{\tilde{S}_B}{\tilde{S}_w} = \frac{A^t S_B A}{A^t S_w A}$$

フィッシャーの評価基準

フィッシャーの評価基準の意味



クラス間変動は大きく

$$J_s(A) = \frac{\tilde{S}_B}{\tilde{S}_w} = \frac{A^t S_B A}{A^t S_w A}$$

クラス内変動は小さく

変換軸Aの方が評価基準値が大きい
→ A軸を変換軸として用いる

変換行列の求め方

$$J_s(A) = \frac{\tilde{S}_B}{\tilde{S}_w} = \frac{A^t S_B A}{A^t S_w A}$$

最大

最大 $\tilde{S}_B = A^t S_B A$

制約条件 $\tilde{S}_w = A^t S_w A = I$ （この場合はスカラー）

ラグランジュ乗数法

$$J(A) = A^t S_B A - \lambda (A^t S_w A - I)$$

変換行列の求め方②

$$\frac{\partial J(A)}{\partial A} = 2S_B A - 2\lambda S_w A = 0$$

$$S_B A = \lambda S_w A$$

$\boxed{\text{S}_\text{w}\text{に逆行列が存在すれば…}}$

$$S_w^{-1} S_B A = \lambda A$$

$\boxed{\text{固有値問題}}$

$$(S_w^{-1} S_B - \lambda I) A = \mathbf{0}$$

Aは$S_w^{-1}S_B$（d×d行列）の最大固有値に対応する固有ベクトル

もしくは…

$$\lambda S_w A = S_B A = \frac{n_1 n_2}{n_1} (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t A$$

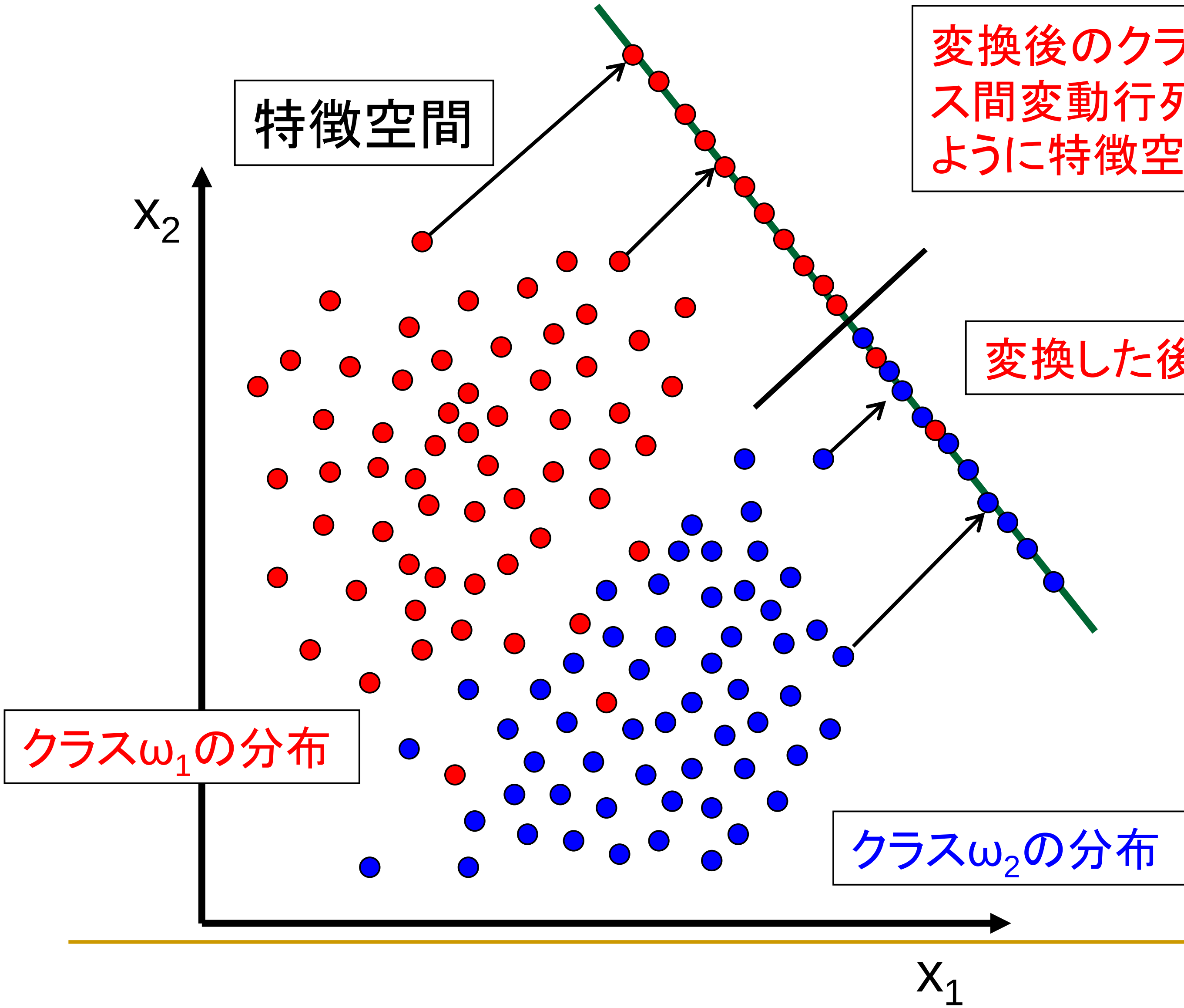
$$S_B = \sum_{i=1,2} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t = \frac{n_1 n_2}{n} (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t$$

$$A \propto S_w^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

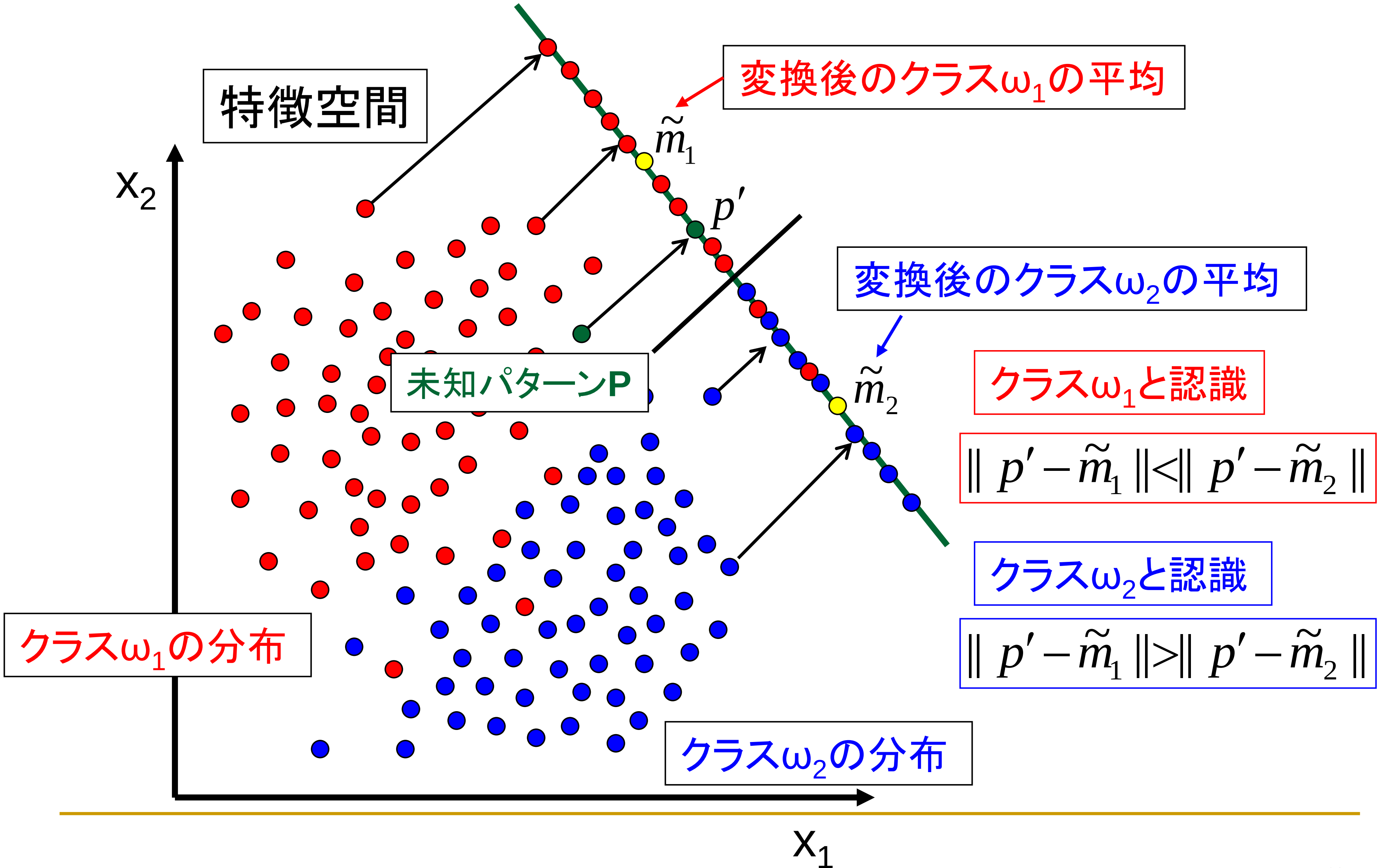
（注意）求められたのは変換軸のみ

変換後のクラス内変動行列とクラス間変動行列の比が最大になるように特徴空間を一次元に変換

変換した後，変換軸上で認識

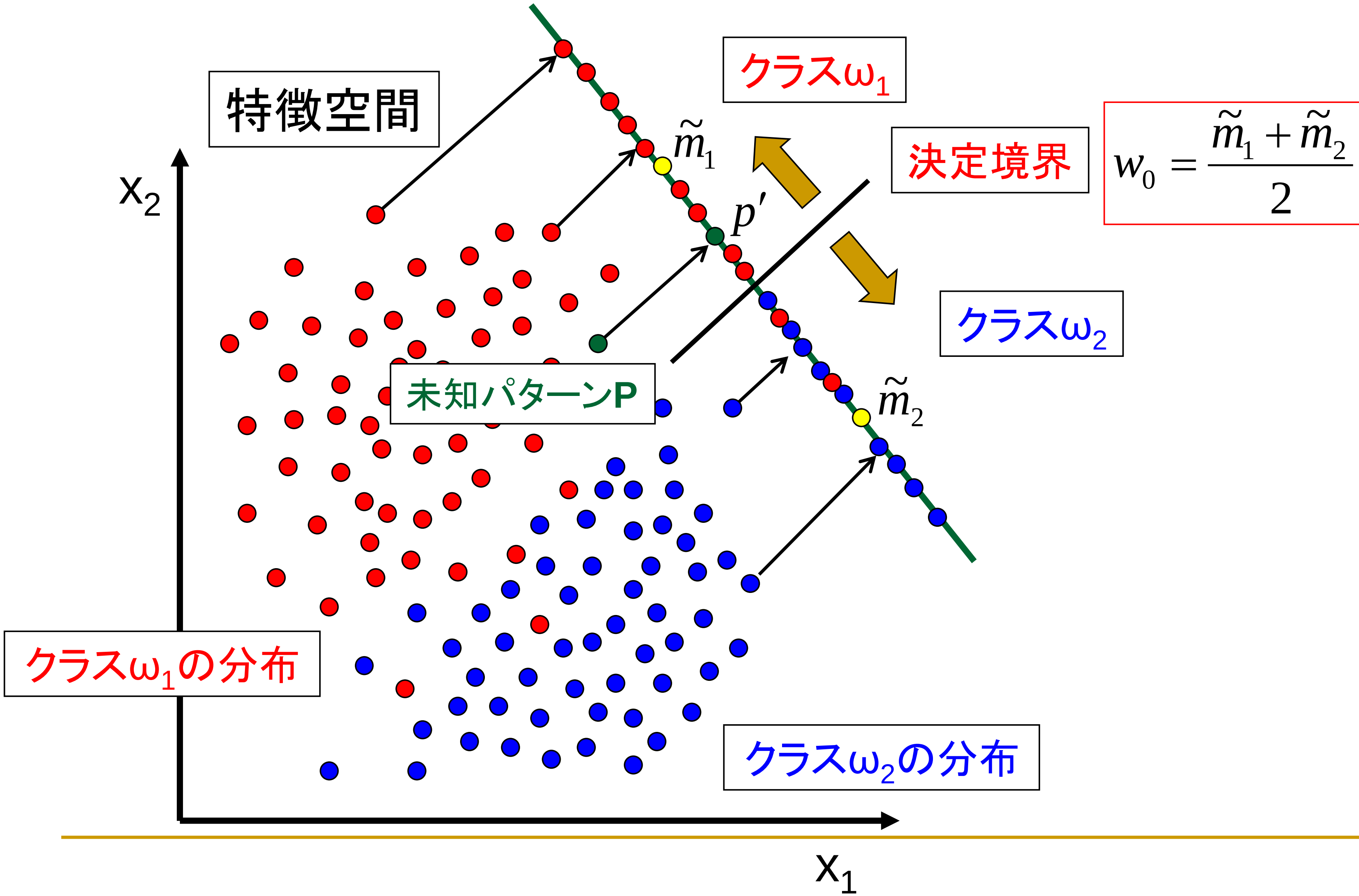


最近傍法による認識



特徴空間
変換後のクラスω1の平均
$\tilde{m}_1$
p'
変換後のクラスω2の平均
$\tilde{m}_2$
未知パターンP
クラスω1と認識
$\| p' - \tilde{m}_1 \| < \| p' - \tilde{m}_2 \|$
クラスω2と認識
$\| p' - \tilde{m}_1 \| > \| p' - \tilde{m}_2 \|$
クラスω1の分布
クラスω2の分布
x2
x1

決定境界による認識



特徴空間
クラスω1
決定境界
$w_0 = \frac{\tilde{m}_1 + \tilde{m}_2}{2}$
$\tilde{m}_1$
p'
クラスω2
$\tilde{m}_2$
未知パターンP
x2
クラスω1の分布
クラスω2の分布
x1

フィッシャーの線形判別のまとめ①

- クラスごとの平均ベクトル$\mathbf{m}_i$を求める
- クラス内変動行列，クラス間変動行列を求める

クラス内変動行列

$$S_w = S_1 + S_2$$
$$= \sum_{i=1,2} \sum_{\mathbf{x} \in \chi_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

クラス間変動行列

$$S_B = \sum_{i=1,2} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$$
$$= \frac{n_1 n_2}{n} (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^t$$

- $S_W^{-1}S_B$ の最大固有値に対応する固有ベクトルを求める
  - 変換行列（ベクトル）A = 固有ベクトル

フィッシャーの線形判別のまとめ②

- 変換後の各クラスの平均値，決定境界を求める

$$\boxed{\begin{aligned}\tilde{m}_i &= A^t\mathbf{m}_i\\ w_0 &= \frac{\tilde{m}_1+\tilde{m}_2}{2}\end{aligned}}$$

- 未知のパターン **p** を認識したい場合

$$p' = A^t\mathbf{p}$$

  - 最近傍法もしくは決定境界w_0によって判別

特徴空間の変換①

- d次元の特徴から，認識に有用な特徴に次元数を削減

$$\mathbf{y} = A^t \mathbf{x}$$

  - 変換行列Aはd×d'（d'<d）
  - **y** の次元数はd'

- これを特徴空間の変換と呼ぶ
  - フィッシャーの線形判別の場合，一次元（d'=1）に削減

特徴空間の変換②

- 次元数の削減には目的によって基準が異なる
- フィッシャーの線形判別
  - 変換後のクラス内変動行列とクラス間変動行列の比を最大
- KL展開（主成分分析）
  - 変換後のパターンの分布の分散を最大
  - 変換前後の誤差の自乗和を最小

多クラスの判別（正準判別法）①

- クラス数が2よりも大きい場合（c>2）

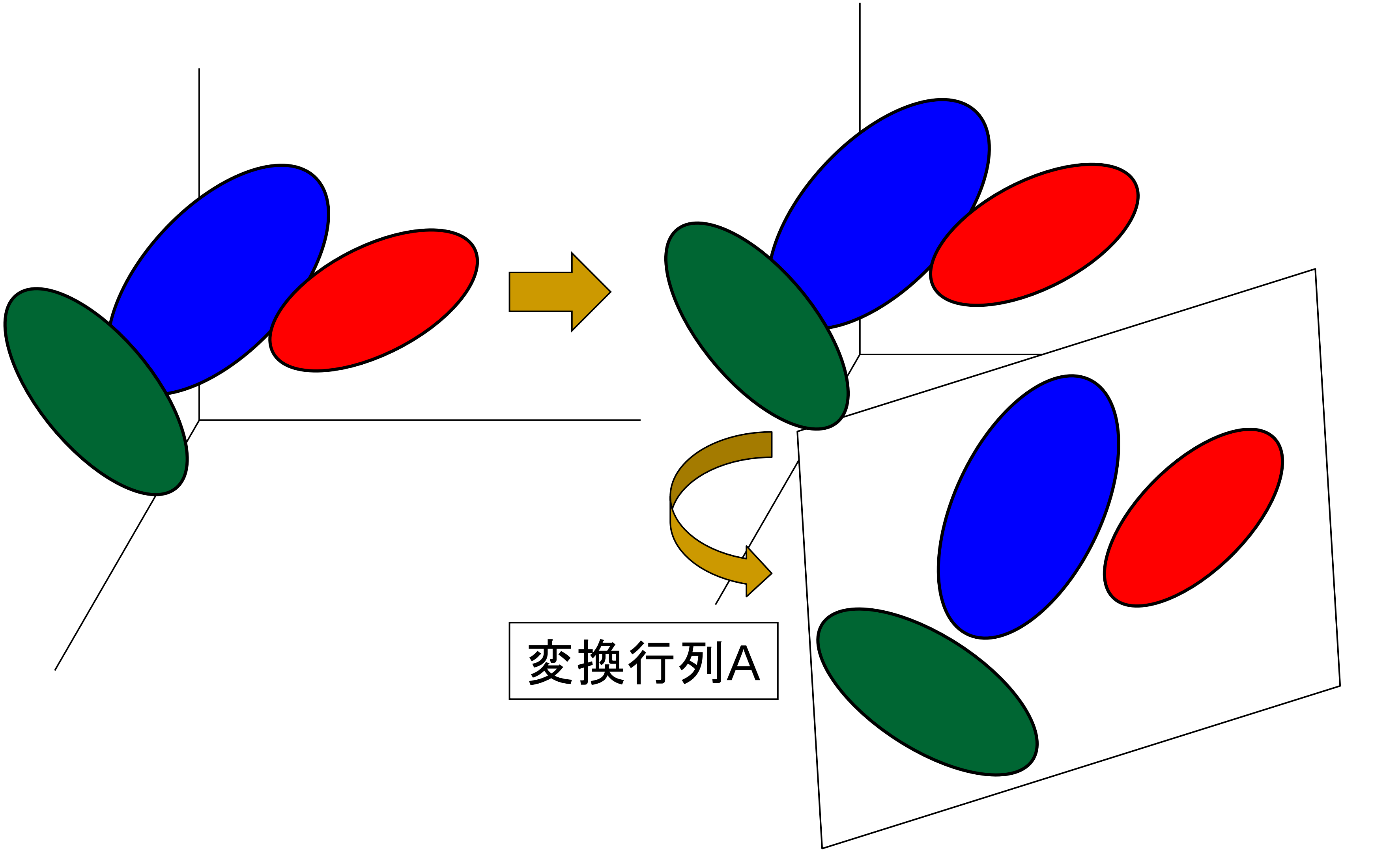
$$\mathbf{y} = A^t \mathbf{x}$$

  - 変換行列Aはd×d'（d'=c-1）

クラスω_iに属するパターンの分散共分散行列

$$\Sigma_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \chi_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

イメージとしては…



多クラスの判別（正準判別法）②

クラス内分散共分散行列

$$\Sigma_W = \sum_{i=1}^{c} P(\omega_i)\Sigma_i$$

$$\boxed{\mathbf{y} = A^t\mathbf{x}}$$

変換後のクラス内分散共分散行列

$$\tilde{\Sigma}_W = A^t\Sigma_w A$$

クラス間分散共分散行列

$$\Sigma_B = \sum_{i=1}^{c} P(\omega_i)(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$$

変換後のクラス間分散共分散行列

$$\tilde{\Sigma}_B = A^t\Sigma_B A$$

p（ω_i）はクラスω_iの事前確率

変換後のクラス間分散共分散行列 → 最大？
変換後のクラス内分散共分散行列 → 最小？

多クラスの判別（正準判別法）③

評価基準*

$$J(A)=\frac{tr(\tilde{\Sigma}_B)}{tr(\tilde{\Sigma}_W)}=\frac{tr(A^t\Sigma_B A)}{tr(A^t\Sigma_W A)}$$

最大 $tr(\tilde{\Sigma}_B)=tr(A^t\Sigma_B A)$

制約条件 $\tilde{\Sigma}_w=A^t\Sigma_w A=I$

固有値問題

$$\Sigma_B A=\Sigma_W A\Lambda$$

$$\Sigma_W^{-1}\Sigma_B A=A\Lambda$$

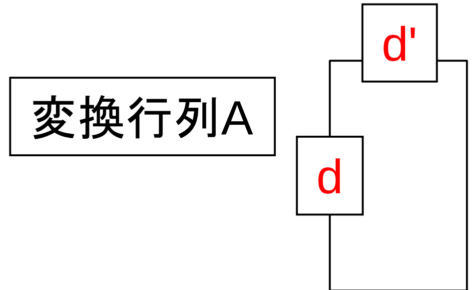
Λはd'×d'行列

$\Sigma_w^{-1}\Sigma_B$ の固有値の大きい順にd'個の固有値に対応する固有ベクトル $\mathbf{u}_1$, $\mathbf{u}_2$, ･･･, $\mathbf{u}_{d'}$

*他にも多数考案されています

多クラスの判別(正準判別法)④

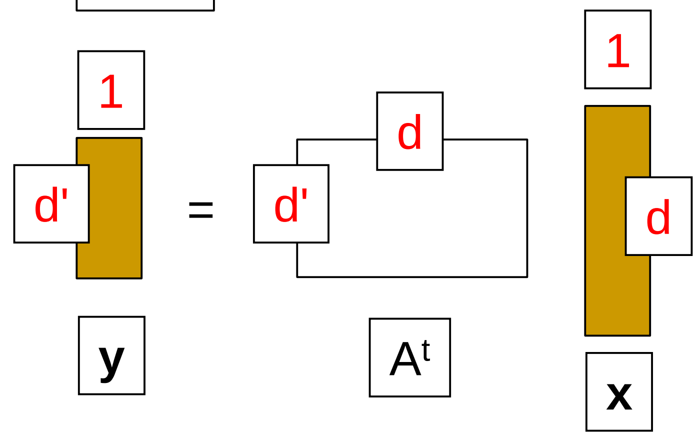
- 変換行列



$$\mathbf{y} = A^t\mathbf{x}$$

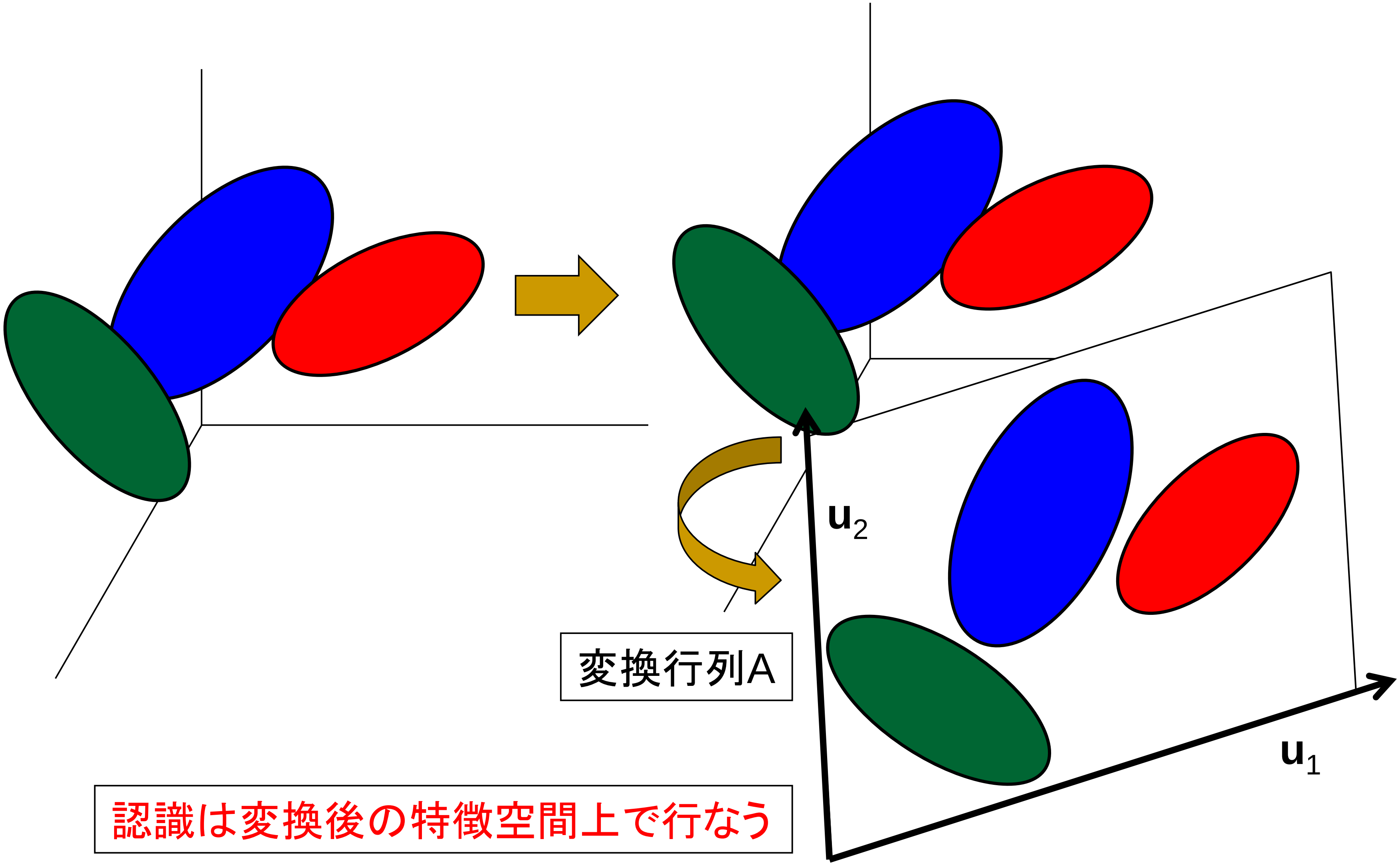
$$A = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{d'})$$

$\mathbf{u}_i$はd×1



- 変換行列Aによって，変換後のクラス内分散共分散行列とクラス間分散共分散行列の比(?)が最大になるように，特徴空間をd'次元に削減

多クラスの判別（正準判別法）⑤



認識は変換後の特徴空間上で行なう

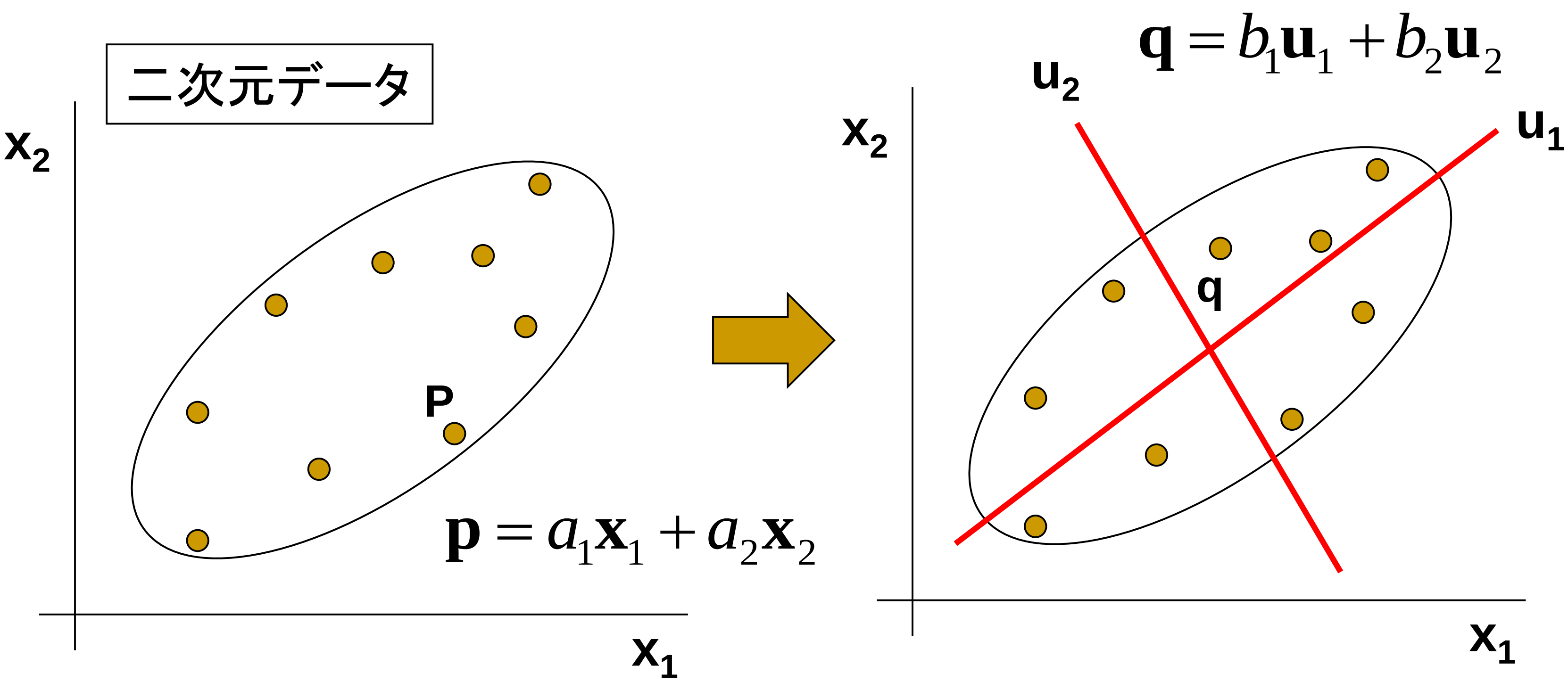
特徴空間の変換

KL展開

固有顔

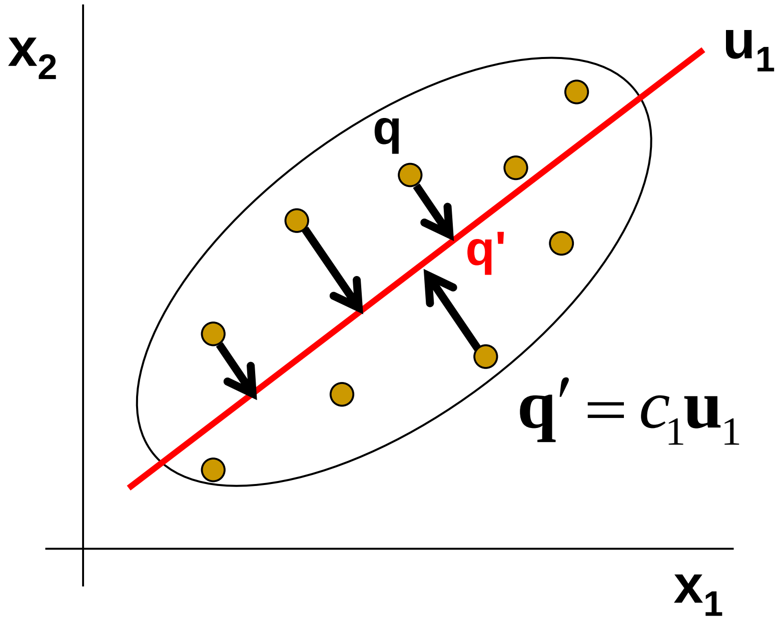
特徴空間の変換①

- KL展開*（主成分分析）
  - 特徴空間の変換（次元圧縮）



*Karhunen-Loeve展開

特徴空間の変換②



$$\mathbf{q}' = c_1\mathbf{u}_1$$

二次元データを一次元データに変換

$\mathbf{x}_1$, $\mathbf{x}_2$ 上でのデータ
→ $\mathbf{u}_1$ 上でのデータ

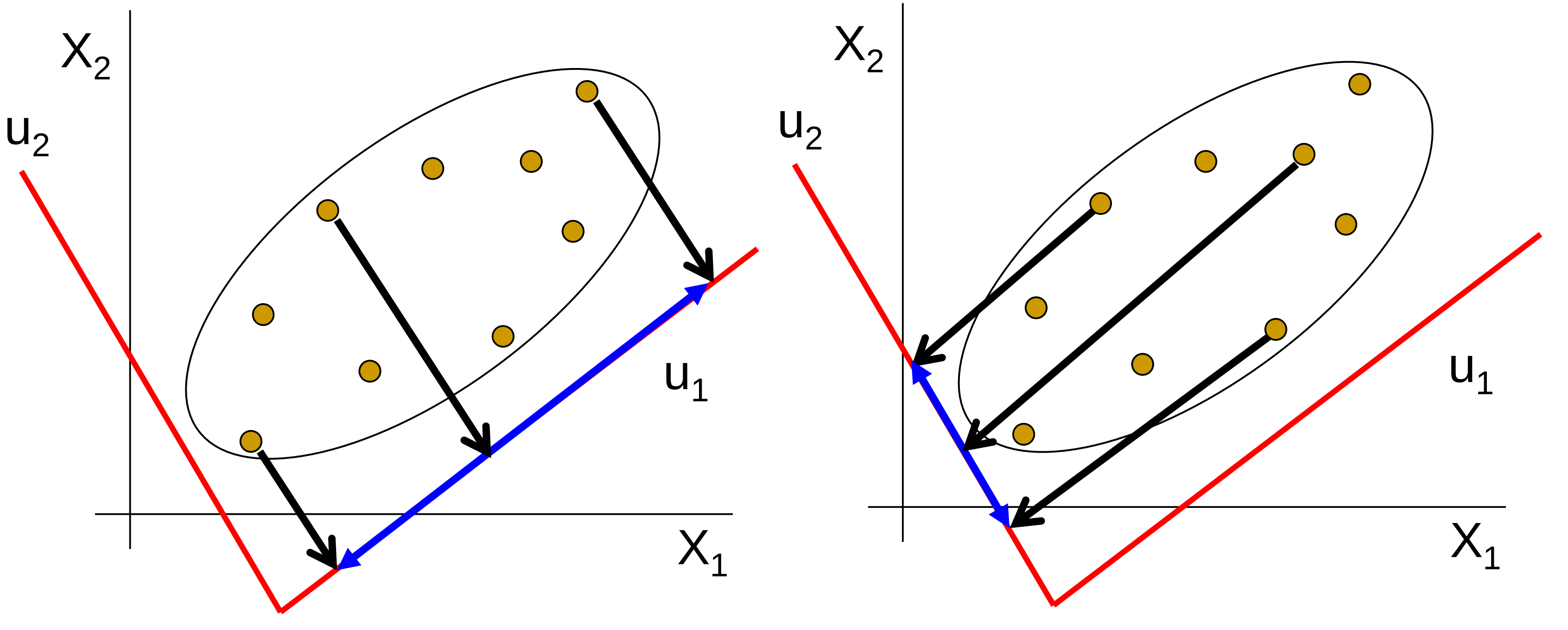
特徴空間の変換（次元削減）

特徴空間の変換の役割

- d次元の特徴から，認識に有用な特徴に次元数を削減 → フィッシャーの線形判別
- その性質を明らかにできる特徴を抽出するため次元数を削減　→　KL展開（主成分分析）

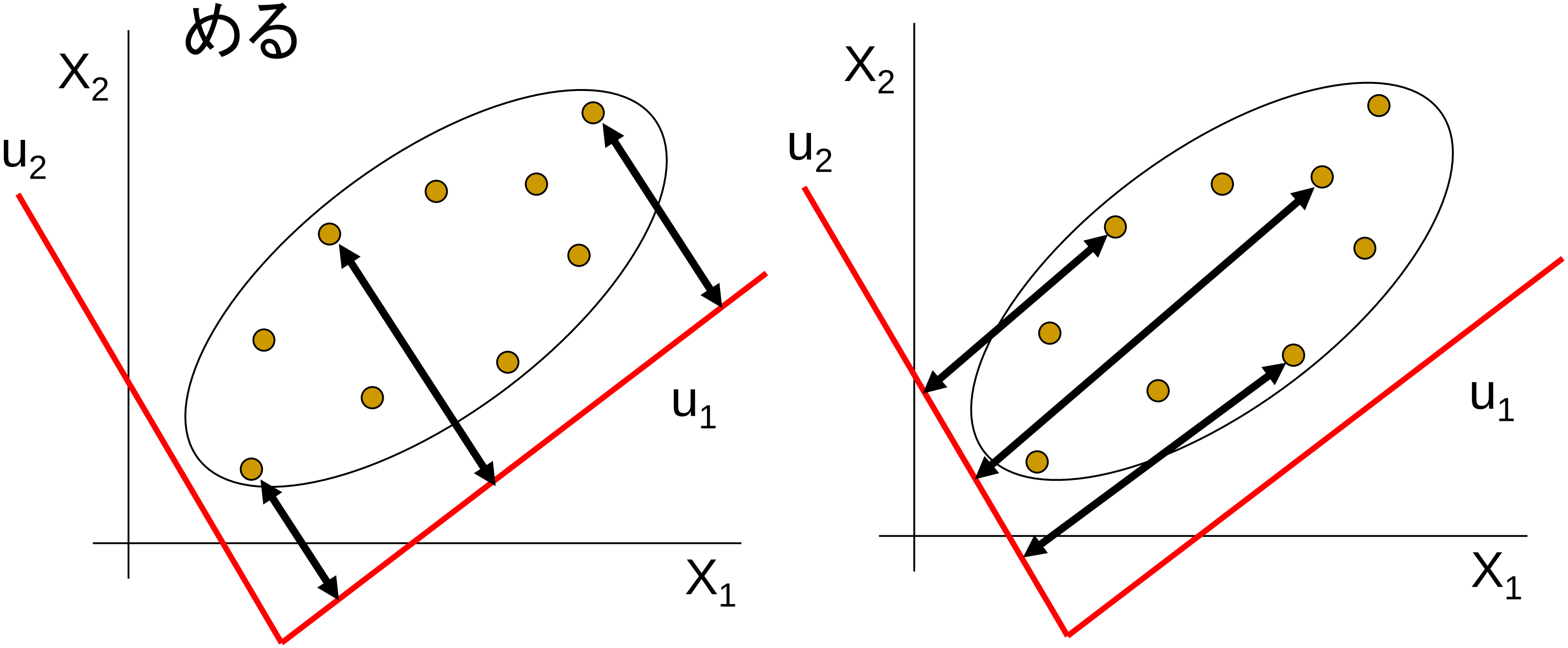
KL展開

- 分散最大基準
  - 変換後の特徴の分散が最大となる軸を求める



KL展開

- 平均自乗誤差最小基準
  - 変換後の特徴との誤差の自乗平均が最小と軸を求める



KL展開の手順①

- N個のデータ $\mathbf{x}_i$（d次元）（i=1,2,･･･,N）

平均

$$\mathbf{m} = \frac{1}{N}\sum_i \mathbf{x}_i$$

分散最大基準の場合

分散共分散行列

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

平均自乗誤差最小基準の場合

相関行列

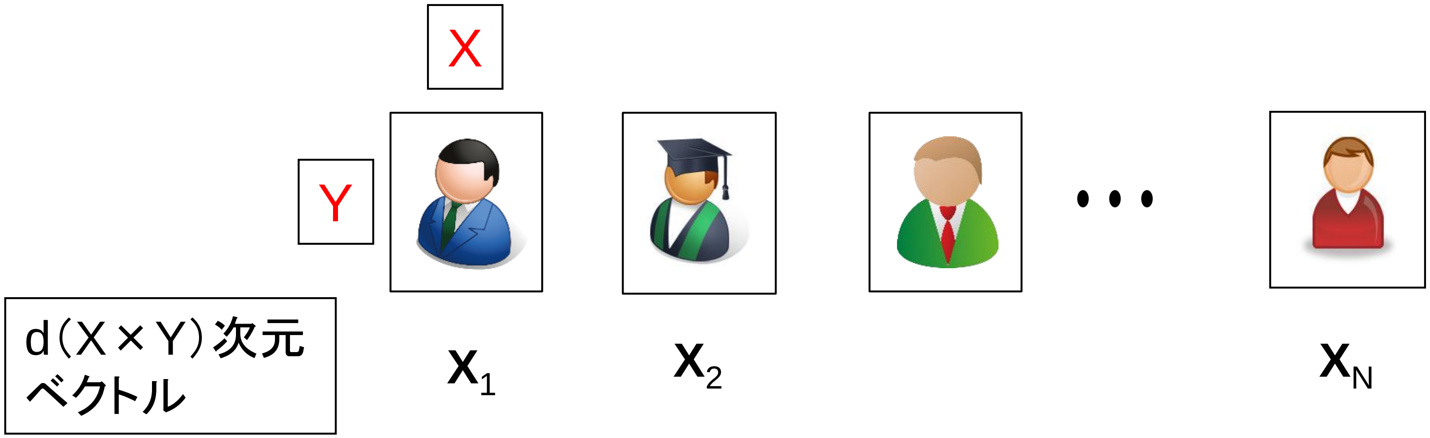
$$\Sigma' = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^t$$

KL展開の手順②

- 分散共分散行列*∑の固有値λ_i, 固有ベクトル$\mathbf{u}_i$を求める（i=1,2,･･･,d）
- 固有値の上位d'（d>d'）個に対応する固有ベクトルを基底ベクトル$\mathbf{u}_i$として用いる（i=1,2,･･･,d'）

*相関行列の場合も同じです

顔画像での例



平均ベクトル(平均顔)

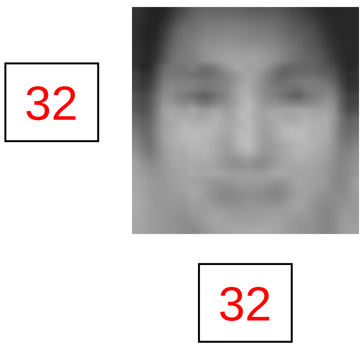
$$\mathbf{m} = \frac{1}{N}\sum_{i}\mathbf{x}_i$$

分散共分散行列

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

平均顔

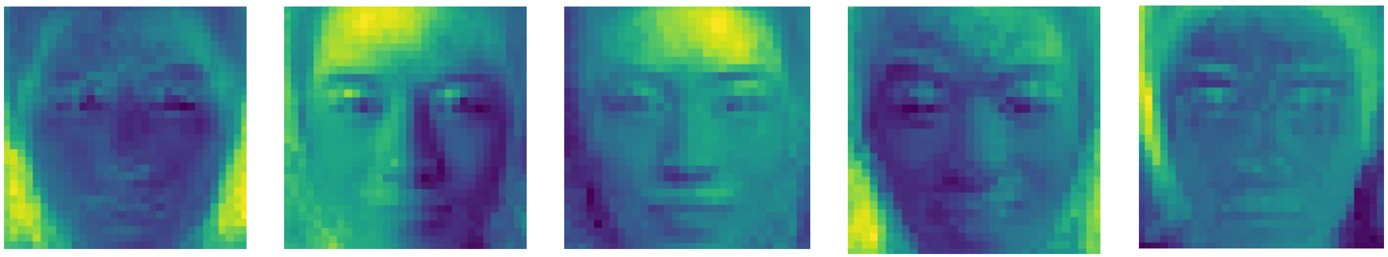
- 男性(100枚)，女性(100枚)の平均顔



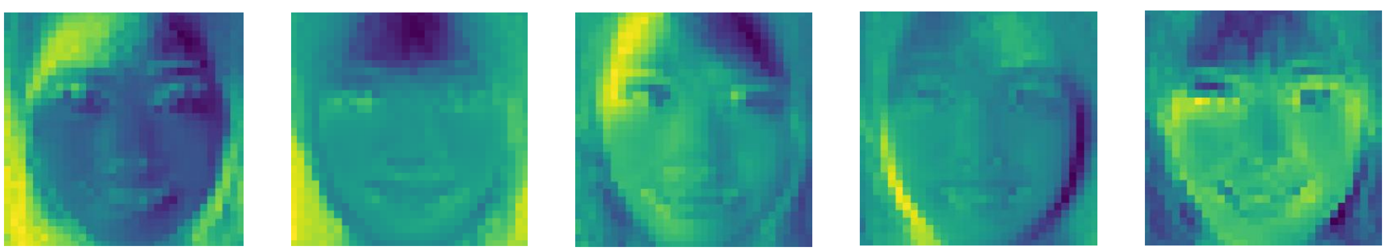
- 男性，女性(合わせて200枚)の平均顔

固有顔（Eigen Face）

100枚の男性画像から抽出した固有ベクトル



100枚の女性画像から抽出した固有ベクトル



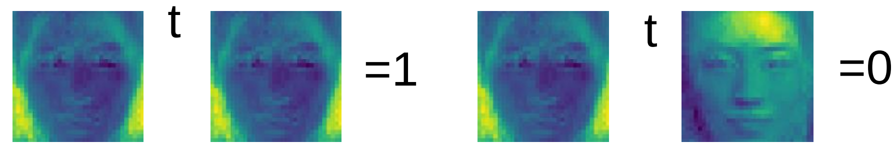
固有ベクトルを**固有顔**と呼ぶ

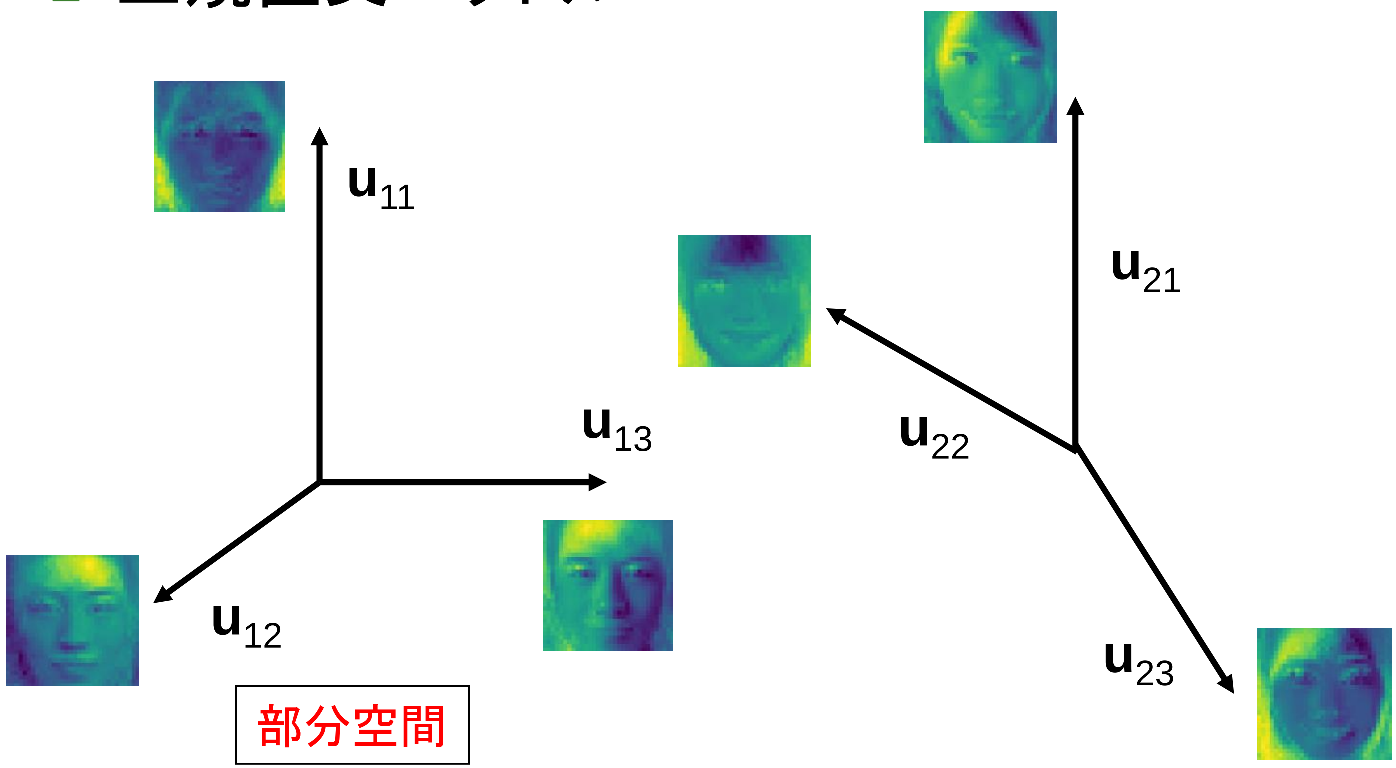
部分空間

$\|\mathbf{u}_{1i}\| = 1$
$\mathbf{u}_{1i}{}^{t}\,\mathbf{u}_{1j} = 0\ (i \neq j)$

$\|\mathbf{u}_{2i}\| = 1$
$\mathbf{u}_{2i}{}^{t}\,\mathbf{u}_{2j} = 0\ (i \neq j)$

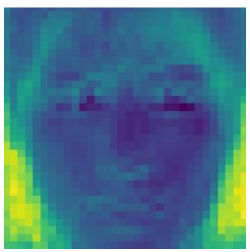
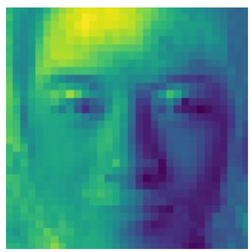
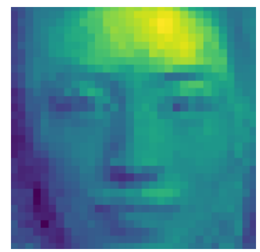
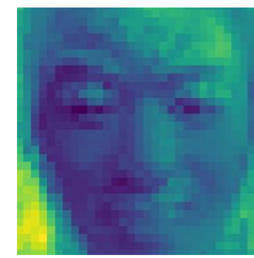
- 固有顔（固有ベクトル）
  - 正規直交ベクトル





固有顔による顔画像の復元①

元画像 s
（d次元ベクトル）

c_1  + c_2  + c_3  ・・・ + $c_{d'}$ 

元画像は固有顔の線形和で表現可能

特徴は（c_1, c_2, c_3, ・・・,$c_{d'}$）のd'次元ベクトル（d>d'）で表現可能（次元圧縮）

固有顔による顔画像の復元②

- 固有ベクトル　$A = (\mathbf{u}_1, \mathbf{u}_1, \cdots, \mathbf{u}_d)$
- 元画像 p　$\mathbf{p} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_d\mathbf{u}_d + \bar{\mathbf{x}}$
- 係数 c_i　$c_i = \mathbf{u}_i^T(\mathbf{p} - \bar{\mathbf{x}})$
- 復元画像 s　$\mathbf{s} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_{d'}\mathbf{u}_{d'} + \bar{\mathbf{x}}$

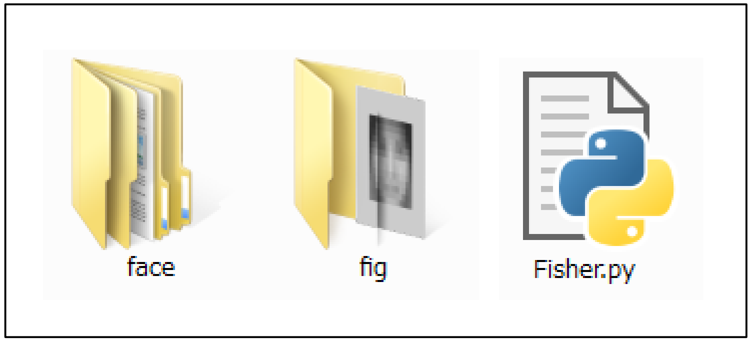
$$d > d'$$

実習①（フィッシャーの線形判別）

性別判定

フィッシャーの線形判別

- Fisher.py
  - 顔画像（性別判定）



faceを同じフォルダーに入れて下さい

figという名前のフォルダーを作成して下さい

- 実行方法
  - > python Fisher.py

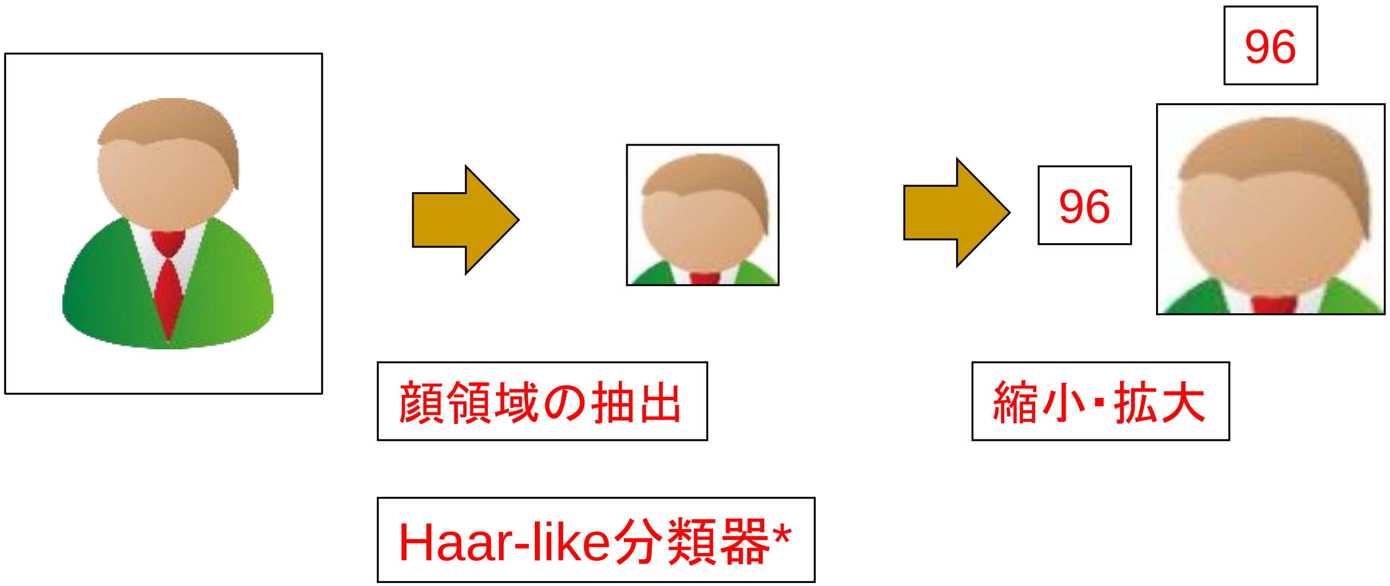
顔画像①

- Face以下のフォルダー
  - グレースケール画像
  - Male 男性画像（100枚）
  - Female 女性画像（100枚）
  - この講義のみの利用で，二次利用はしないで下さい

  - 諏訪瑛くん（2009年度卒）がYahoo!から収集
  - あつく御礼申し上げます

顔画像②

- 顔画像の抽出



*Paul Viola，Michael J. Jones，Robust real-time face detection，International Journal of Computer Vision (IJCV)，Vol.57，No.2，pp.137-154, 2004

変数の定義①

```
# クラス数
class_num = 2

# 画像の大きさ
size = 8

# 学習データ
train_num = 80

# テストデータ
test_num = 100-train_num

# 学習データ，平均ベクトル（クラス），平均ベクトル（全データ）
train_vec = np.zeros((class_num,train_num,size*size), dtype=np.float64)
ave_vec = np.zeros((class_num,size*size), dtype=np.float64)
all_ave_vec = np.zeros((size*size), dtype=np.float64)
```

変数の定義②

```
# クラス内分散共分散行列，クラス間分散共分散行列
Sw = np.zeros((size*size,size*size), dtype=np.float64)
Sb = np.zeros((size*size,size*size), dtype=np.float64)

# 固有値，固有ベクトル
lamda = np.zeros(size*size, dtype=np.float64)
eig_vec = np.zeros((size*size,size*size), dtype=np.float64)

# fig以下の画像を削除（MS-Windows）
os.system("del /Q fig¥*")
```

MacOS，UNIXの場合
"rm fig/*"

使用する変数

- 学習データ train_vec： class_num，train_num，（size×size）
- 平均ベクトル（各クラス） ave_vec：class_num，（size×size）
- 平均ベクトル（全データ） all_ave_vec：（size×size）

- クラス内分散共分散行列 Sw：（size×size），（size×size）
- クラス間分散共分散行列 Sb：（size×size），（size×size）

- 固有値 lamda：（size×size）
- 固有ベクトル eig_vec：（size×size），（size×size）

学習データの読み込み①

```
# 学習データの読み込み
dir = [ "Male" , "Female" ]
for i in range(class_num):
    for j in range(1,train_num+1):
        train_file = "face/" + dir[i] + "/" + str(j) + ".png"
        work_img = Image.open(train_file).convert('L')

        resize_img = work_img.resize((size, size))
        train_vec[i][j-1] =
                np.asarray(resize_img).astype(np.float64).flatten()
```

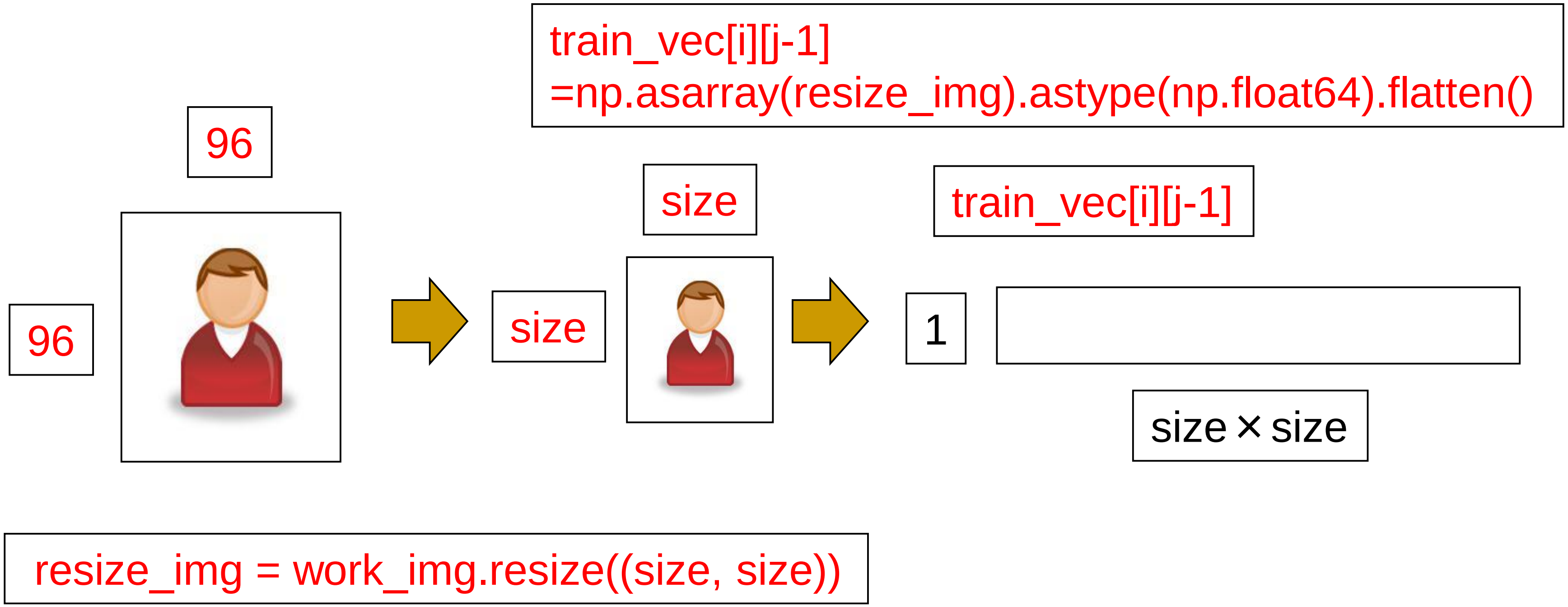
読み込む画像のファイル名

グレースケール画像として読み込み

（size×size）に縮小

numpyに変換→ベクトル化

学習データの読み込み②



学習データの読み込み③

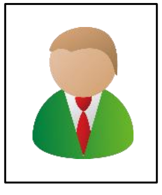
男性画像

size×size

train_vec[0][0]

train_vec[0][1]

⋮

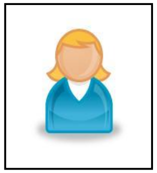


train_vec[0][train_num-1]

学習データの読み込み④

女性画像

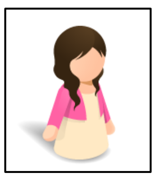
size×size



train_vec[1][0]

train_vec[1][1]

・
・
・



train_vec[1][train_num-1]

平均ベクトル（平均顔）①

```
# 平均ベクトル（各クラス）
for i in range(class_num):
    ave_vec[i] = np.mean( train_vec[i] , axis=0 )

    # 平均顔の保存（各クラス）
    ave_img = Image.fromarray(np.uint8(np.reshape(ave_vec[i],(size,size))))
    ave_file = "fig/" + str(dir[i]) + "-ave.png"
    ave_img.save(ave_file)
```

列方向に平均を求める（axis=0）

（size×size）に変換し，画像化

保存先のファイル名

画像として保存

平均顔
（32×32）

男性

女性

平均ベクトル（平均顔）②

列方向に平均を求める（axis=0）

```
# 平均ベクトル（全データ）
all_ave_vec = np.mean( ave_vec , axis=0 )

# 平均顔の保存（全データ）
ave_img = 
Image.fromarray(np.uint8(np.reshape(all_ave_vec,(size,size))))
ave_file = "fig/all-ave.png"
ave_img.save(ave_file)
```

（size×size）に変換し，画像化

保存先のファイル名

画像として保存

男性，女性合わせた
平均顔（32×32）

分散共分散行列，逆行列

```
# クラス内分散共分散
for i in range(class_num):
    Sw += np.cov( train_vec[i] , rowvar=0 , bias=1 )
```

$$\Sigma_W = \sum_{i=1}^{c} P(\omega_i)\Sigma_i$$

rowvar=0

データ1
データ2
:
データn

bias=1
標本分散

```
# クラス間分散共分散
for i in range(class_num):
    a = np.reshape( ave_vec[i] - all_ave_vec , (size*size,1) )
    Sb += np.dot( a , a.T )
```

$$\Sigma_B = \sum_{i=1}^{c} P(\omega_i)(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$$

```
# クラス内分散共分散の逆行列
Sw_1 = np.linalg.inv( Sw )

# Sw-1 Sb
V = np.dot( Sw_1 , Sb )
print( " Rank -> " , np.linalg.matrix_rank(V) )
```

$$\Sigma_W^{-1}\Sigma_B$$

固有値分解

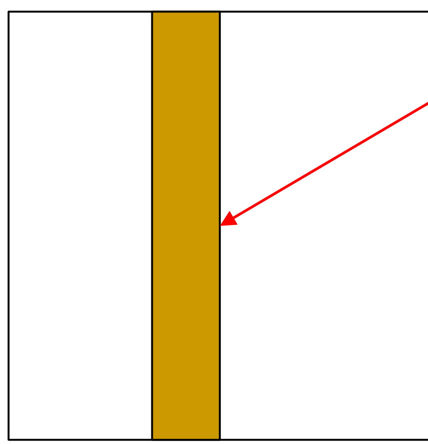
```
# 固有値分解
lamda , eig_vec = np.linalg.eig( V )
```

固有値（ベクトル）
大きい順にソート済み

固有ベクトル（固有行列）
固有値と対応済み

eig_vec

size*size

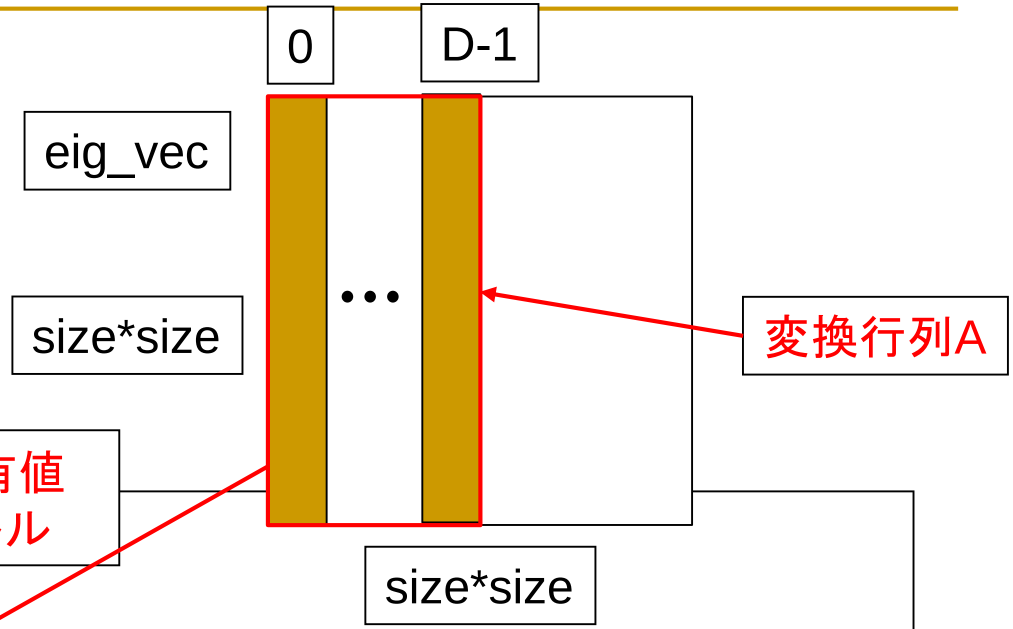


（注意）
固有ベクトルは列方向

j番目の固有値　lamda[j]
固有ベクトル eig_vec[:,j]

size*size

変換行列



この場合は，最大固有値に対応する固有ベクトル

```
# 変換行列
D = 1
A = np.reshape( eig_vec[:,0:D].real , (size*size,D) )
```

実数部

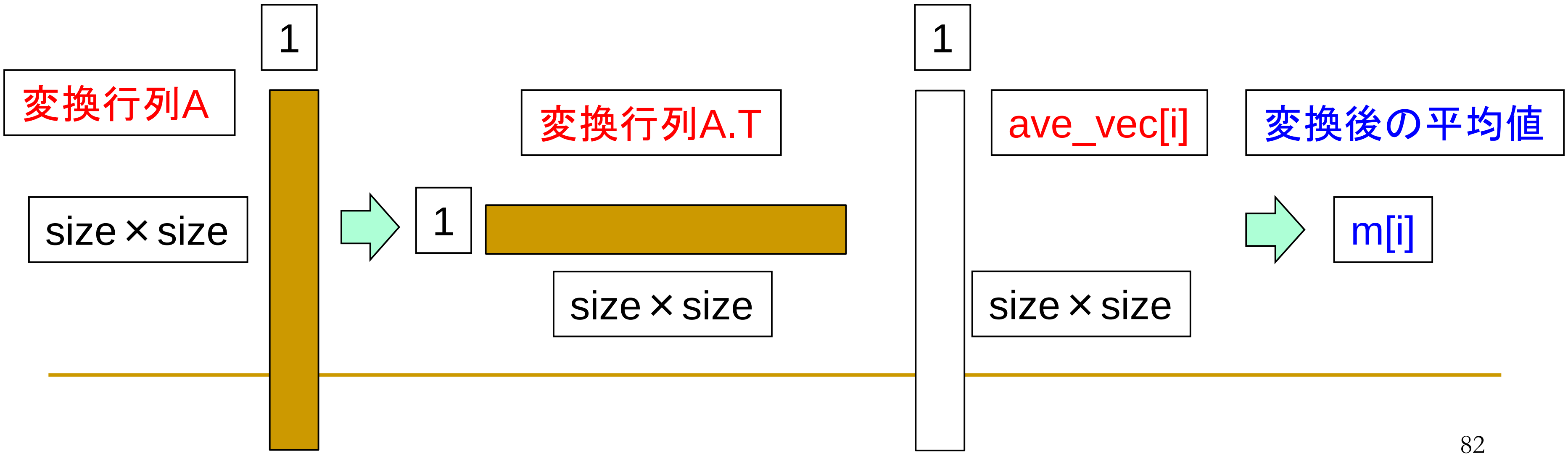
```
# 固有値分解を行わない場合（コメントしている）
a = np.reshape( ave_vec[0] - ave_vec[1] , (size*size,1) )
A = np.dot( Sw_1 , a )
```

$$\boxed{A \propto \Sigma_w^{-1}(\mathbf{m}_1 - \mathbf{m}_2)}$$

変換後の各クラスの平均値

```
# 変換後の各クラスの平均値
m = np.zeros((class_num,D), dtype=np.float64)
for i in range(class_num):
    a = np.reshape( ave_vec[i] , (size*size,1) )
    m[i] = np.dot( A.T , a ).flatten()
print( m )
```

変換行列によって変換



テストデータの読み込み

```
# 混合行列
result = np.zeros((class_num,class_num), dtype=np.int32)

for i in range(class_num):
    for j in range(train_num,101):
        # テストデータの読み込み
        pat_file = "face/" + dir[i] + "/" + str(j) + ".png"
        work_img = Image.open(pat_file).convert('L')
        resize_img = work_img.resize((size, size))
        pat_vec =
            np.reshape( np.asarray(resize_img).astype(np.float64) , (size*size,1) )
```

読み込む画像のファイル名

グレースケール画像として読み込み

（size×size）の画像に大きさを変更

numpyに変換→ベクトルに変形

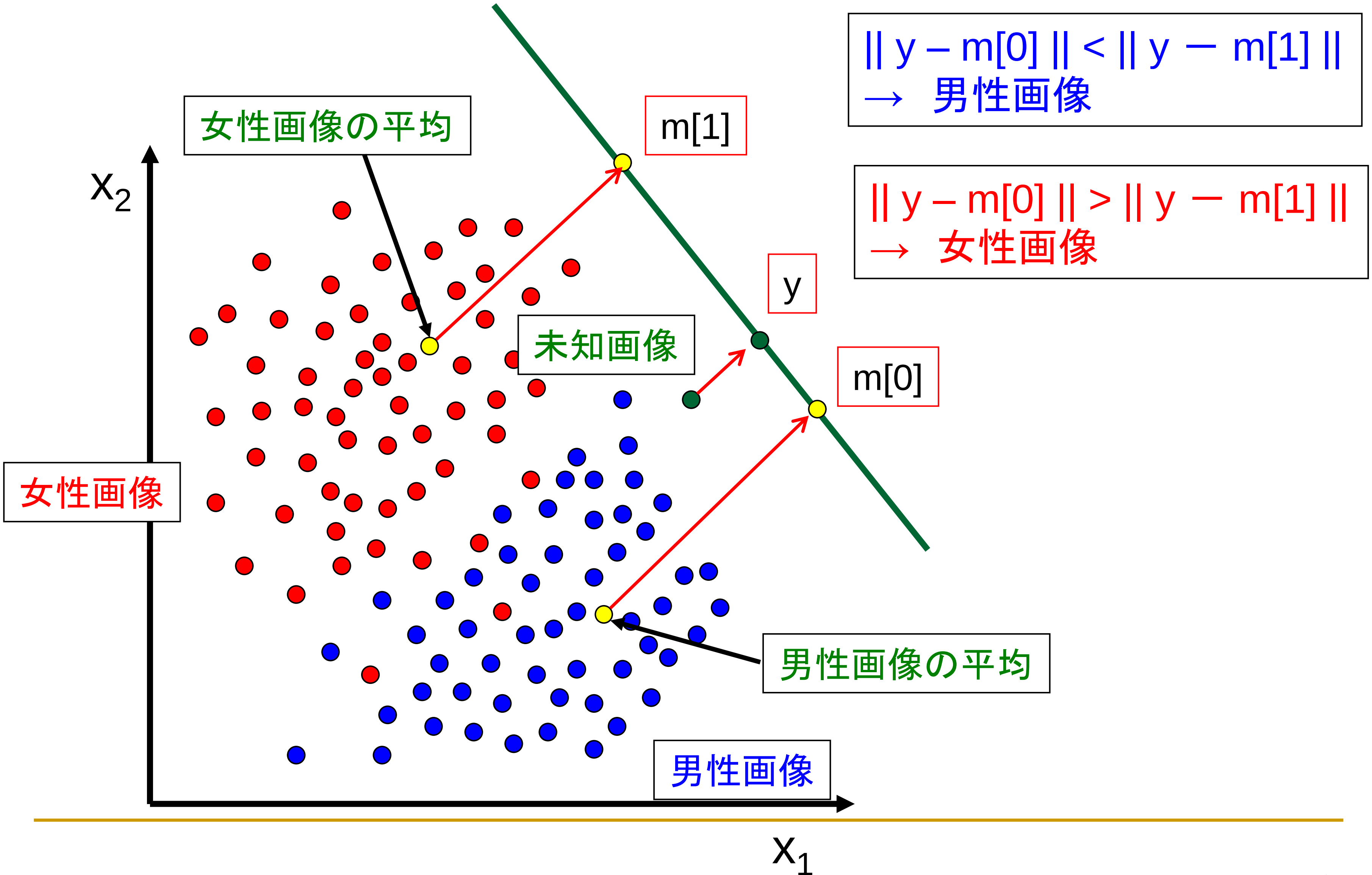
最近傍法による判別①

```
# 変換行列によって変換
y = np.dot( A.T , pat_vec ).flatten()

min_val = float('inf')
ans = 0
for k in range(class_num):
    dist = np.dot( (y-m[k]).T , y-m[k] )   最近傍法

    if dist < min_val:                  最小値の探索
        min_val = dist
        ans = k
                                        混合行列に予測値を代入
result[i][ans] +=1
print( i , j , "->" , ans )
```

最近傍法による判別②



|| y – m[0] || < || y − m[1] ||
→　男性画像
|| y – m[0] || > || y − m[1] ||
→　女性画像
女性画像の平均
m[1]
y
未知画像
m[0]
女性画像
男性画像の平均
男性画像
x2
x1

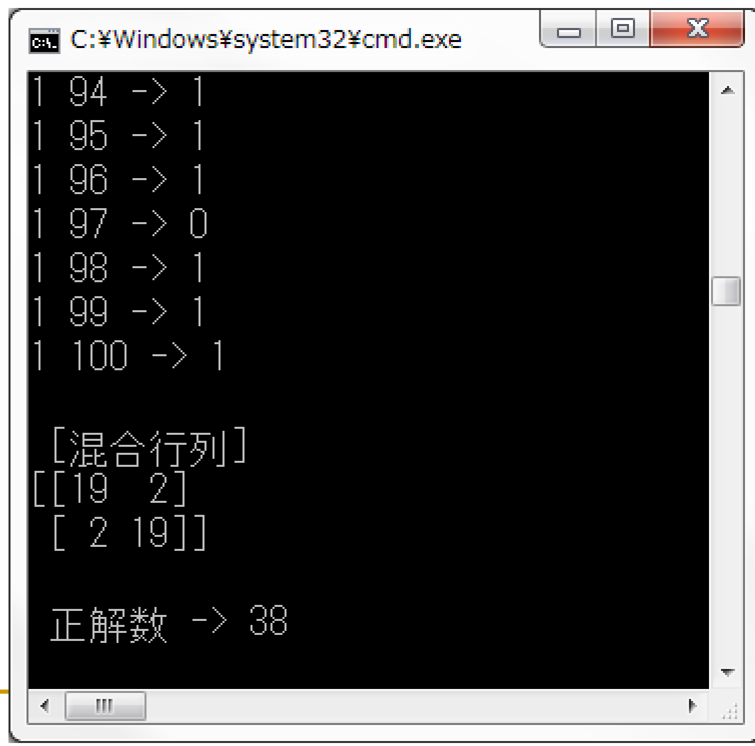
混合行列の表示

```
print( "¥n [混合行列]" )
print( result )
print( "¥n 正解数 ->" ,  np.trace(result) )
```

混合行列の出力

正解数の出力

出力結果



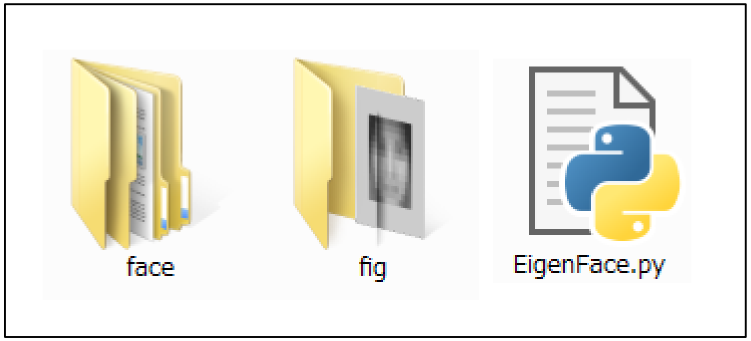
実習②（固有顔）

画像の復元

次元圧縮→最近傍法による認識

固有顔（EigenFace.py）

- EigenFace.py



faceを同じフォルダーに入れて下さい

figという名前のフォルダーを作成して下さい

- 実行方法
  - > python EigenFace.py

固有顔（EigenFace.py）

- 学習データを対象に，固有顔を求める
- （固有顔は性別ごとに求める）

- 固有顔を用いて，テストデータを再現する係数ベクトル（d次元）を求める

- d'（d>d'）個の係数のみを用いて，テストデータを復元

変数の定義①

Fisher.pyと同じです

```
# クラス数
class_num = 2

# 画像の大きさ
size = 32

# 学習データ
train_num = 90

# テストデータ
test_num = 100-train_num

# 学習データ，平均ベクトル
train_vec = np.zeros((class_num,train_num,size*size), dtype=np.float64)
ave_vec = np.zeros((class_num,size*size), dtype=np.float64)
```

変数の定義②

```
# 分散共分散行列，固有値，固有ベクトル
Sw = np.zeros((class_num,size*size,size*size), dtype=np.float64)
lamda = np.zeros((class_num,size*size), dtype=np.float64)
eig_vec = np.zeros((class_num,size*size,size*size), dtype=np.float64)

# 係数の個数の入力
D = int( input( " D? > " ) )
```

画像復元に用いる係数の個数

```
# fig以下の画像を削除（MS-Windows）
os.system("del /Q fig¥*")
```

MacOS，UNIXの場合
"rm fig/*"

データの読み込み

Fisher.pyと（ほぼ）同じです

```
# データの読み込み
dir = [ "Male" , "Female" ]
for i in range(class_num):
    for j in range(1,train_num+1):
        train_file = "face/" + dir[i] + "/" + str(j) + ".png"
        work_img = Image.open(train_file).convert('L')

        resize_img = work_img.resize((size, size))
        train_vec[i][j-1] = np.asarray(resize_img).astype(np.float64).flatten()
```

読み込む画像のファイル名

グレースケール画像として読み込み

（size×size）に縮小

numpyに変換→ベクトル化

平均顔

```
# 平均ベクトル
for i in range(class_num):
    ave_vec[i] = np.mean( train_vec[i] , axis=0 )

    # 平均顔の保存
    ave_img =
    Image.fromarray(np.uint8(np.reshape(ave_vec[i],(size,size))))
    ave_file = "fig/" + str(dir[i]) + "-ave.png"
    ave_img.save(ave_file)
```

列方向に平均を求める（axis=0）

（size×size）に変換し，画像化

保存先のファイル名

画像として保存

固有値分解

```
for i in range(class_num):
    # クラス内分散共分散
    Sw[i] = np.cov( train_vec[i] , rowvar=0 , bias=1 )

    # 固有値分解
    lamda[i] , eig_vec[i] = np.linalg.eig( Sw[i] )

    # 固有ベクトルの表示
    for j in range(5):
        a = np.reshape( eig_vec[i][:,j].real , (size,size) )
        plt.imshow(a , interpolation='nearest')
        plt.colorbar()
        file = "fig/eigen-" + str(i) + "-" + str(j) + ".png"
        plt.savefig(file)
        plt.clf()
```

rowvar=0

データ1
データ2
:
データn

bias=1
標本分散

上位5個の固有ベクトルを表示

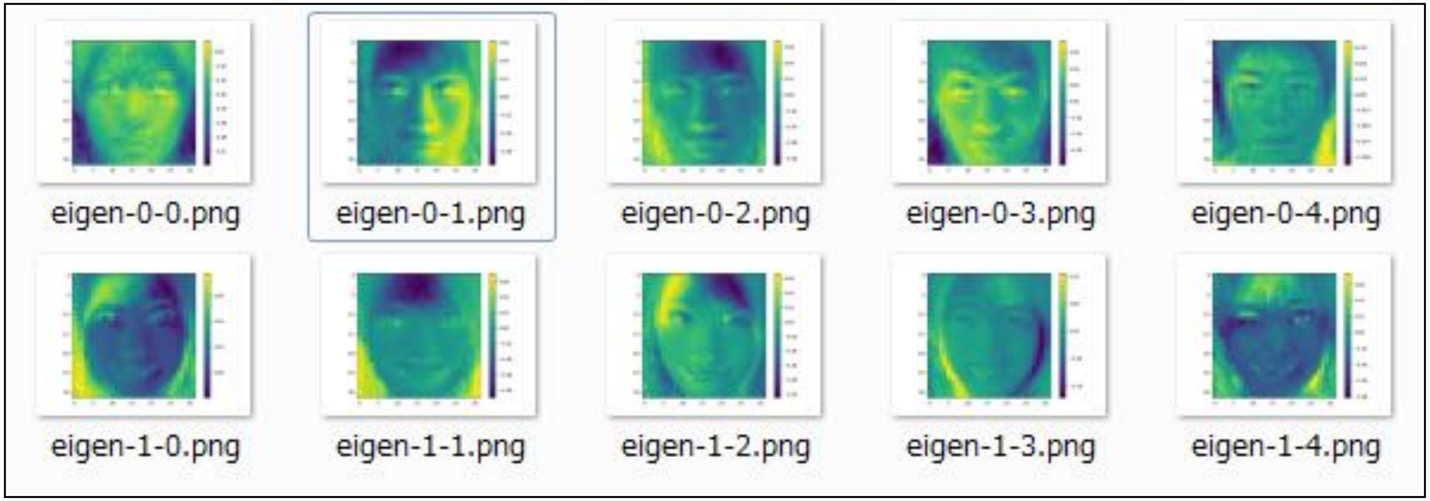
（size×size）に変換

書き込むファイル名

ファイルの保存→閉じる

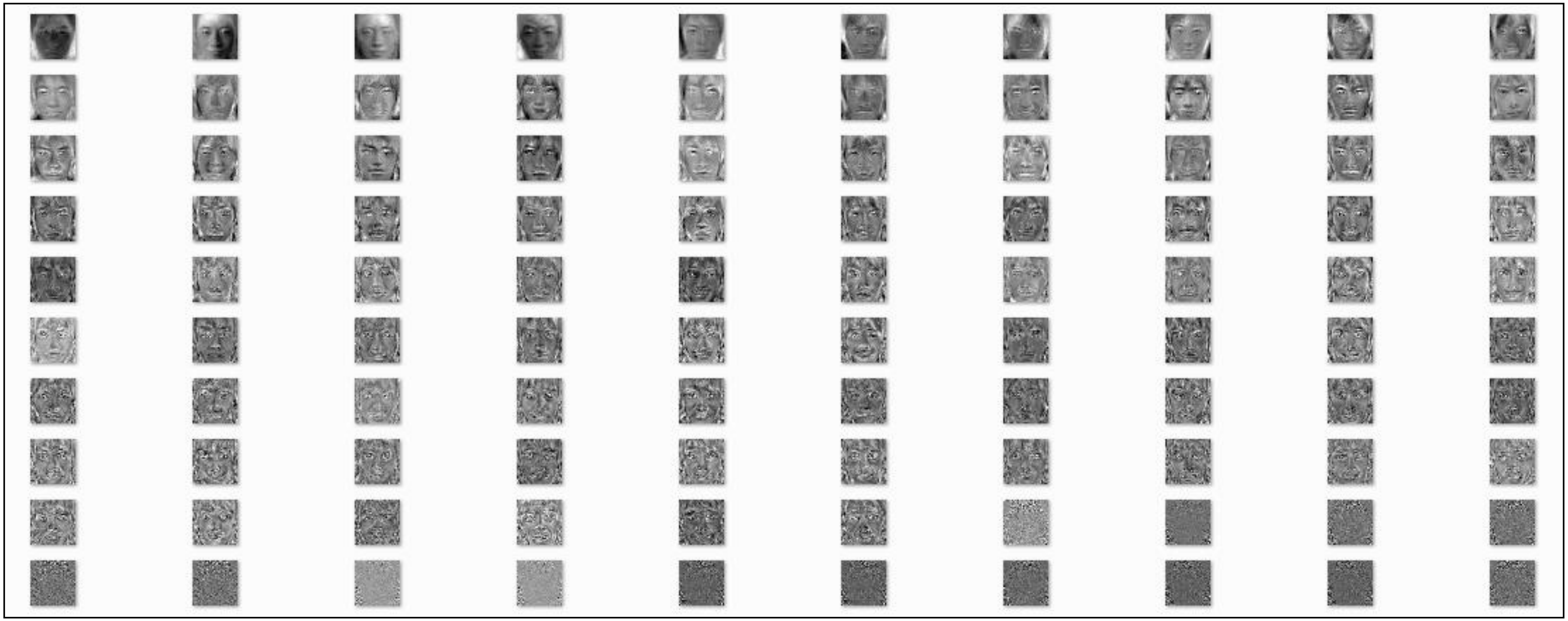
固有ベクトルの表示①

固有顔



固有ベクトルの表示②

男性画像（固有値の大きい順番に100個）



テストデータの読み込み

Fisher.pyと同じです

```
for j in range(train_num,101):
    # テストデータの読み込み
    pat_file = "face/" + dir[i] + "/" + str(j) + ".png"
    work_img = Image.open(pat_file).convert('L')

    resize_img = work_img.resize((size, size))
    src_vec =
  np.reshape( np.asarray(resize_img).astype(np.float64) , (size*size,1) )
```

読み込む画像のファイル名

グレースケール画像として読み込み

（size×size）の画像に大きさを変更

numpyに変換→ベクトルに変形

係数ベクトル→復元

```
# 係数ベクトル
c = np.zeros((size*size), dtype=np.float64)
for k in range(size*size):
    a = np.resize( ave_vec[i] , (size*size,1) )
    c[k] = np.dot( eig_vec[i][:,k].real.T , ( src_vec - a ))
```

$$c_k = \mathbf{u}_k^T (\mathbf{s} - \bar{\mathbf{x}})$$

$\mathbf{u}_k^T$　　$(\mathbf{s} - \bar{\mathbf{x}})$

```
# 復元
restore_vec = np.zeros((size*size), dtype=np.float64)
for k in range(0,D):
    restore_vec += c[k] * eig_vec[i][:,k].real
restore_vec += ave_vec[i]
```

$$\mathbf{s} = c_1\mathbf{u}_1 + c_2\mathbf{u}_2 + \cdots + c_D\mathbf{u}_D + \bar{\mathbf{x}}$$

D個の係数を用いて復元

元画像と復元画像の表示①

```
# 画像の描画
plt.figure()

# 元画像の表示
plt.subplot(1,2,1)
plt.imshow(np.asarray(resize_img).astype(np.float64),cmap='gray')
plt.title( "Original Image" )

# 復元画像の表示
plt.subplot(1,2,2)
plt.imshow(np.reshape(restore_vec,(size,size)),cmap='gray')
```

plt.subplot(1,2,1)	plt.subplot(1,2,2)
元画像	復元画像

元画像と復元画像の表示②

保存する画像ファイル名

```
# 画像の保存
file = "fig/" + dir[i] + "-" + str(j) + "-result.png"
plt.title( "Restore Image( " + str(D) + ")" )
plt.savefig(file)
plt.close()
```

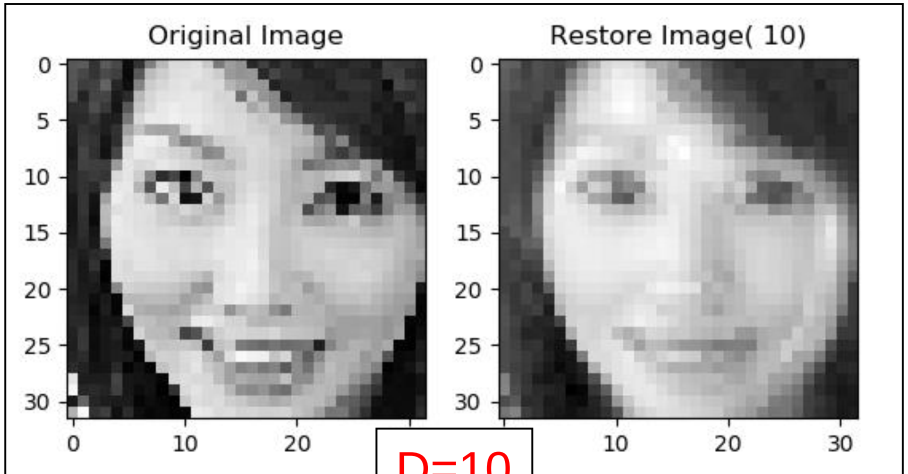
タイトル

保存→閉じる

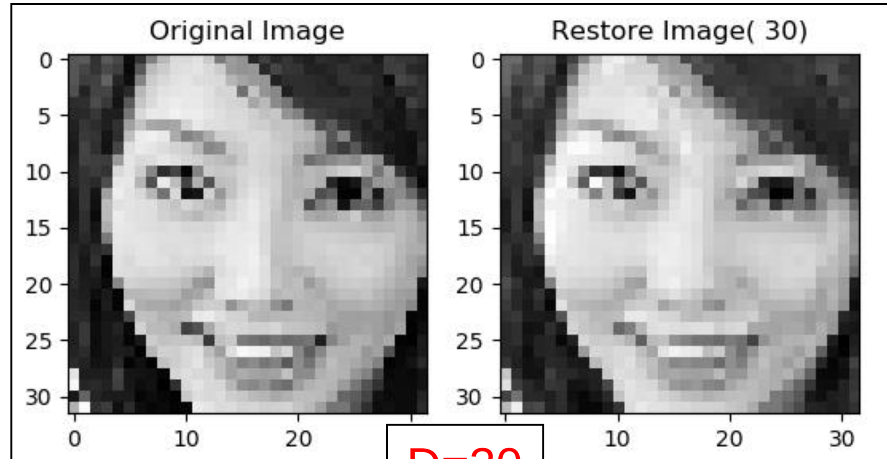
元画像と復元画像

元画像

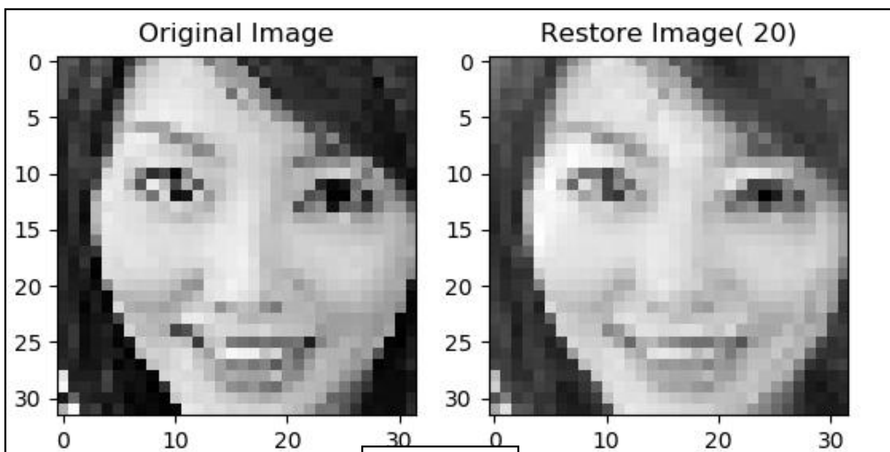
復元画像



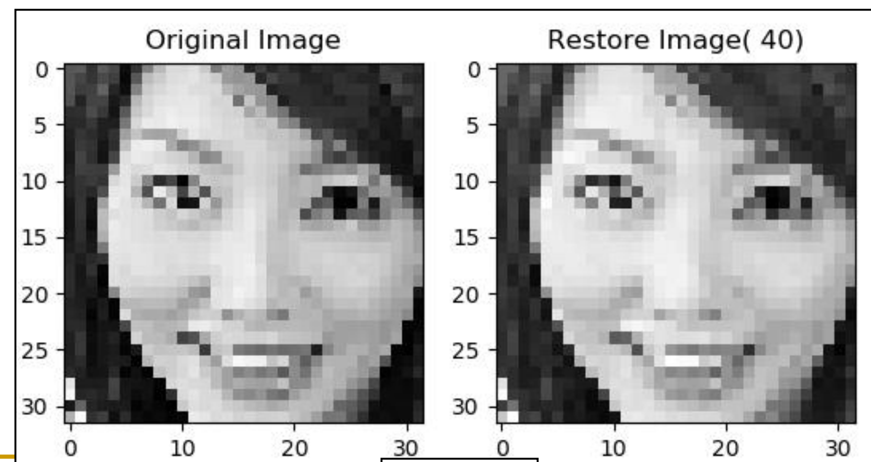
D=10



D=30



D=20



D=40

EigenFace-NN.py

- 全ての学習データ（男女合わせて）を対象に，固有顔を求める（固有顔は性別ごとに求めない）

- 固有顔を用いて，全ての学習データを再現する係数ベクトル（d次元）をそれぞれ求める

- テストデータを認識するため，固有顔を用いて，テストデータを再現する係数ベクトル（d次元）を求める

- d'（d>d'）個のみの係数を用いて，最近傍法を行なう

変数の定義①

```
# クラス数
class_num = 2

# 画像の大きさ
size = 32

# 学習データ
train_num = 80

# テストデータ
test_num = 100-train_num

# 学習データ，平均ベクトル
train_vec = np.zeros((train_num*class_num,size*size), dtype=np.float64)
ave_vec = np.zeros(size*size, dtype=np.float64)
```

性別ごとに固有顔は求めないことに注意

変数の定義②

```
# 分散共分散行列，固有値，固有ベクトル
Sw = np.zeros((size*size,size*size), dtype=np.float64)
lamda = np.zeros((size*size), dtype=np.float64)
eig_vec = np.zeros((size*size,size*size), dtype=np.float64)

# 係数の個数の入力
D = int( input( " D? > " ) )
```

性別推定に用いる係数の個数

```
# 係数ベクトル
c = np.zeros((train_num*2,size*size), dtype=np.float64)
```

学習データの読み込み

```
# fig以下の画像を削除（MS-Windows）
os.system("del /Q fig¥*")

# 学習データの読み込み
dir = [ "Male" , "Female" ]
for i in range(class_num):
    for j in range(1,train_num+1):
        train_file = "face/" + dir[i] + "/" + str(j) + ".png"
        work_img = Image.open(train_file).convert('L')
        resize_img = work_img.resize((size, size))
        train_vec[i*train_num+j-1] =
        np.asarray(resize_img).astype(np.float64).flatten()
```

MacOS，UNIXの場合
"rm fig/*"

読み込む画像のファイル名

グレースケール画像
として読み込み

（size×size）に縮小

numpyに変換→ベクトル化

平均顔

```
# 平均ベクトル
ave_vec = np.mean( train_vec , axis=0 )

# 平均顔の保存
ave_img = Image.fromarray(np.uint8(np.reshape(ave_vec,(size,size))))
ave_file = "fig/ave.png"
ave_img.save(ave_file)

# 分散共分散
Sw = np.cov( train_vec , rowvar=0 , bias=1 )
```

列方向に平均を求める（axis=0）

（size×size）に変換し，画像化

保存先のファイル名

画像として保存

rowvar=0

データ1
データ2
:
データn

bias=1
標本分散

固有顔

```
# 固有値分解
lamda , eig_vec = np.linalg.eig( Sw )

# 固有ベクトルの表示
for j in range(10):
    a = np.reshape( eig_vec[:,j].real , (size,size) )
    plt.imshow(a , interpolation='nearest')
    plt.colorbar()
    file = "fig/eigen-" + str(j) + ".png"
    plt.savefig(file)
    plt.close()
```

上位10個の固有ベクトルを表示

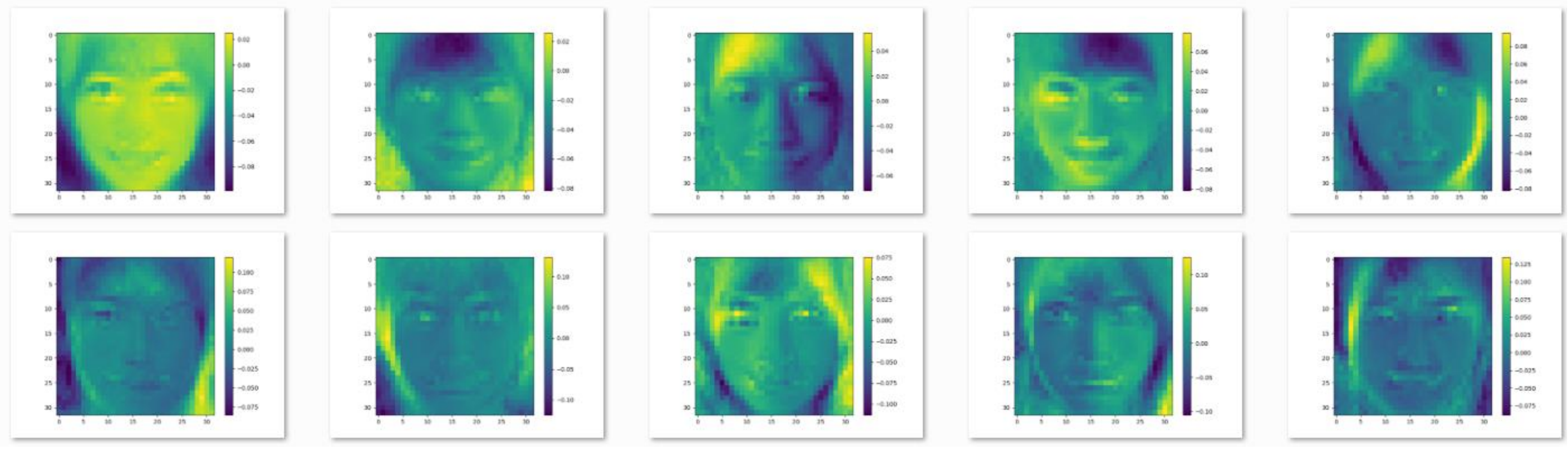
(size×size)に変換

書き込むファイル名

ファイルの保存→閉じる

固有顔

固有値の上位10個に対応した固有ベクトル



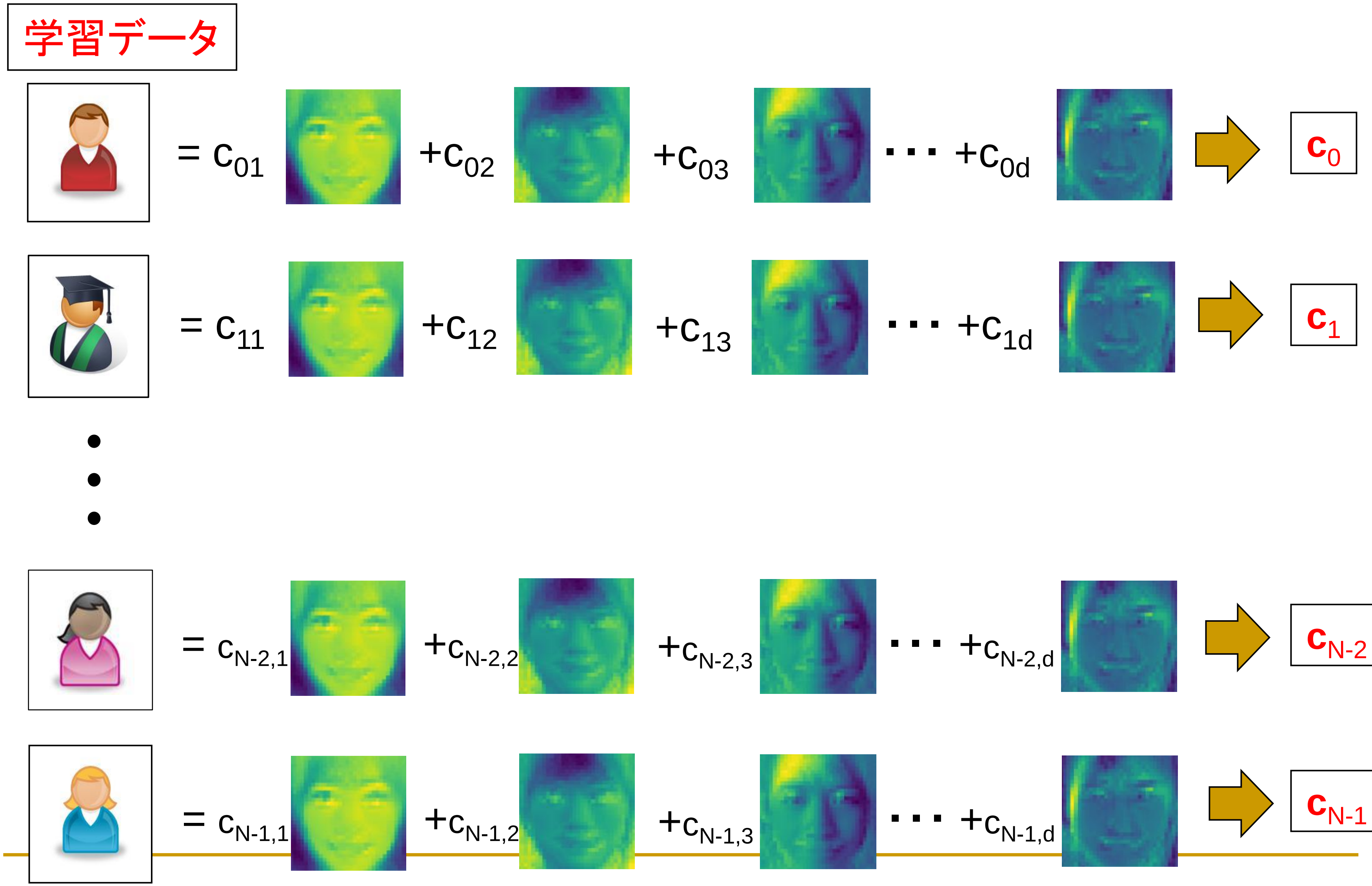
宿題⑥

- 固有顔を用いて，全ての学習データを再現する係数ベクトル（d次元=size×size）をそれぞれ求めなさい．

- 固有顔を用いて，テストデータを再現する係数ベクトル（d次元）を求めなさい．

- D（d>D）個のみの係数を用いて，最近傍法によりテストデータを認識しなさい．
  - Dはキーボードより入力

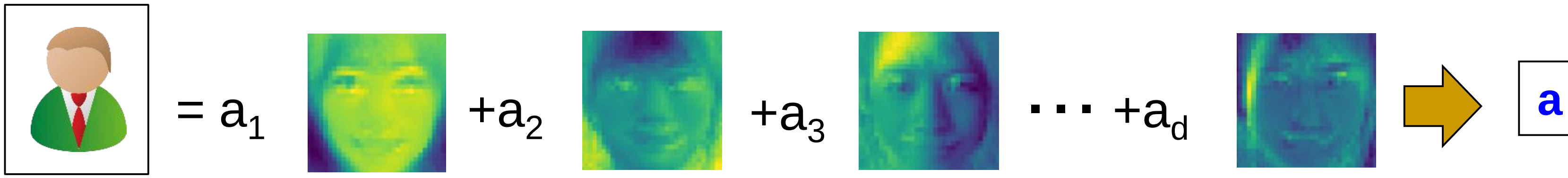
係数ベクトルを求める(学習データ)

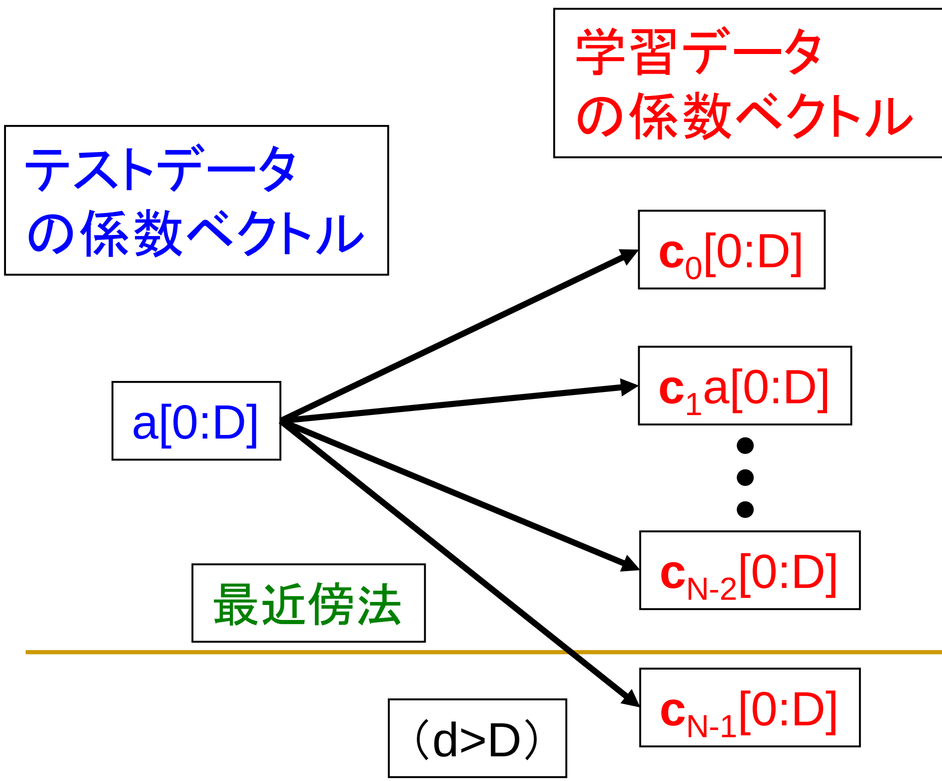


N : 学習データ数

最近傍法による判別①

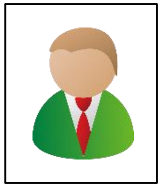
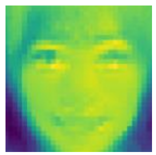
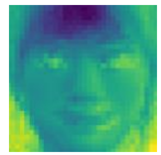
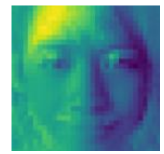
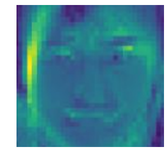
テストデータ





最近傍法による判別②

テストデータ

 $= a_1$  $+a_2$  $+a_3$  $\cdots +a_d$  → **a**

学習データ
の係数ベクトル

(d>d')

テストデータ
の係数ベクトル

$\mathbf{c}_0$[0:D]

⋮

$\mathbf{c}_{N/2-1}$a[0:D]

N/2

男性の平均
$\mathbf{m}_0$[0:D]

a[0:D]

女性の平均
$\mathbf{m}_1$[0:D]

最近傍法

$\mathbf{c}_{N/2}$[0:D]

⋮

$\mathbf{c}_{N-1}$[0:D]

N/2

（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）
- 酒井幸市：画像処理とパターン認識入門，森北出版（2008）
- 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）