# パターン認識と学習
# 統計的パターン認識(1)

管理工学科
篠沢佳久

# 資料の内容

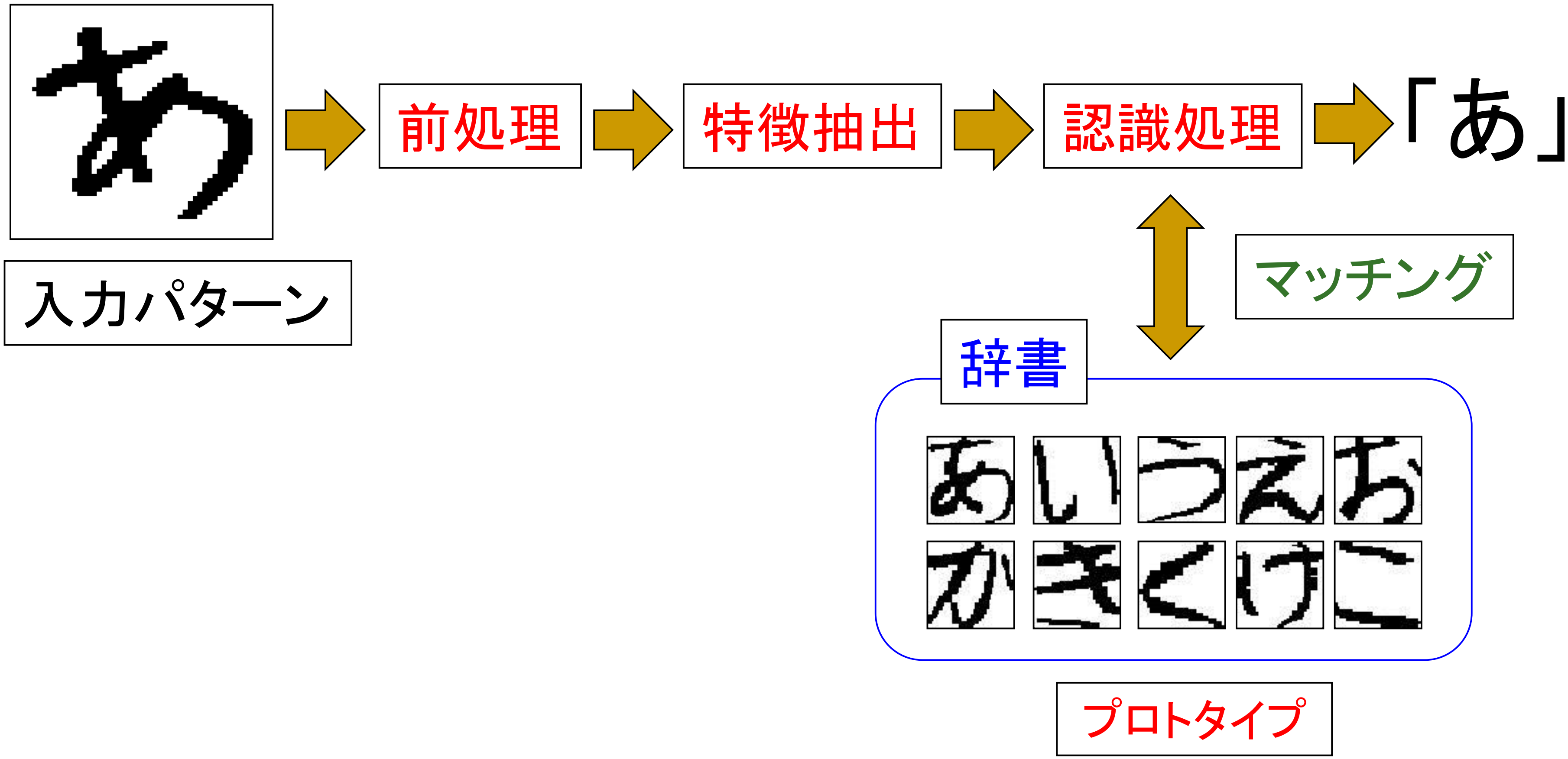
- 統計的パターン認識の基礎(1)
  - 特徴の分布
  - 特徴の出現確率を利用した認識方法
  - ベイズ決定則

- マハラノビス距離による認識の例題

# 統計的パターン認識の基礎

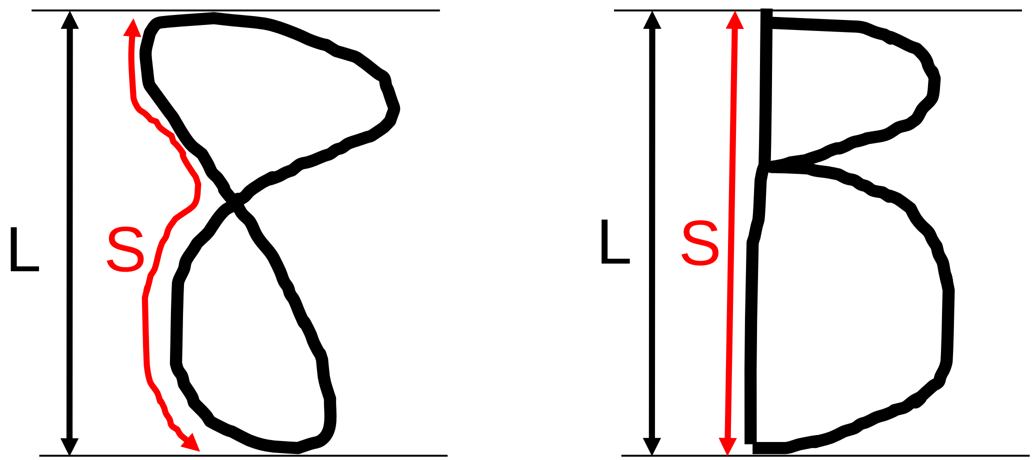
特徴の分布

# 最近傍法(復習)



# 特徴抽出の一例

- 「8」と「B」を区別する特徴*
  - 特徴量は一変数




直線を示す特徴

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

xが0に近い程 → 曲線的
xが1に近い程 → 直線的

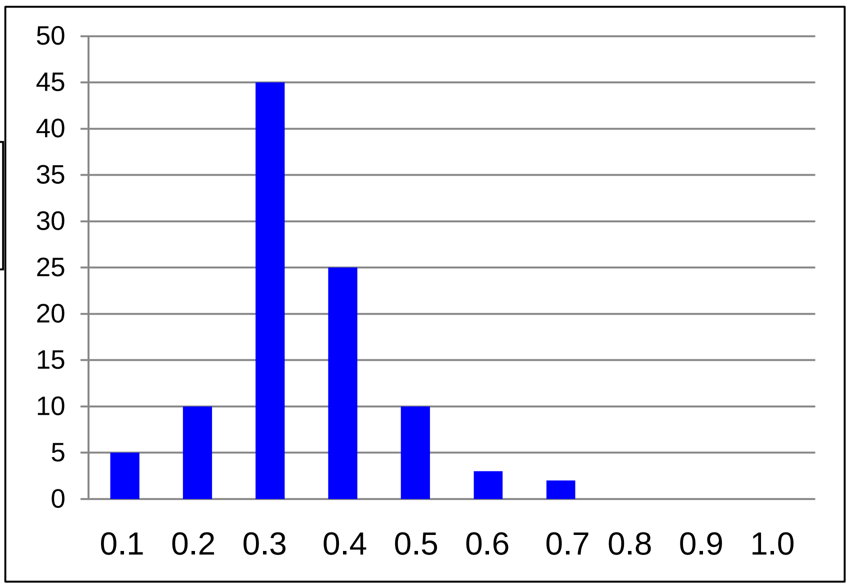
*実際に抽出するのは難しい特徴です

# 特徴の分布

「8」と「B」，それぞれ100個ずつのパターンについて
直線を示す特徴の抽出を行なった場合

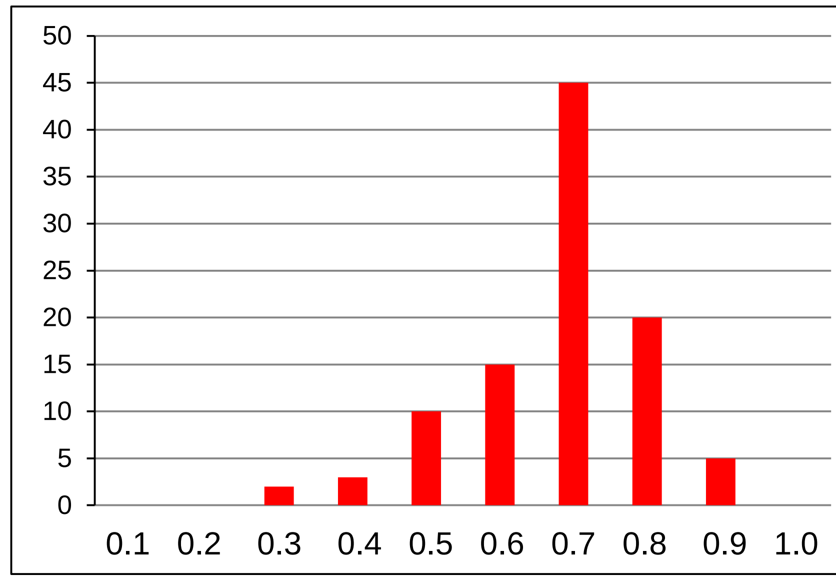
「8」の結果

度数




直線を示す特徴 x

「B」の結果



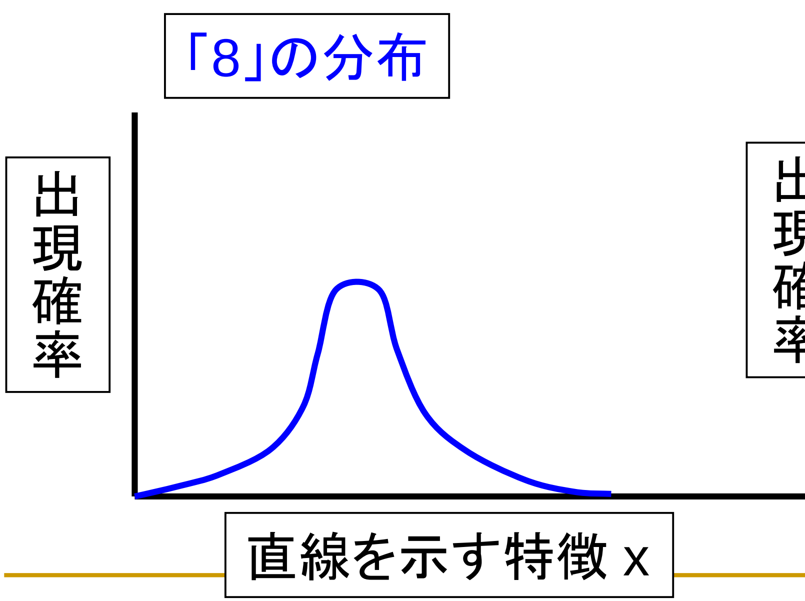

直線を示す特徴 x

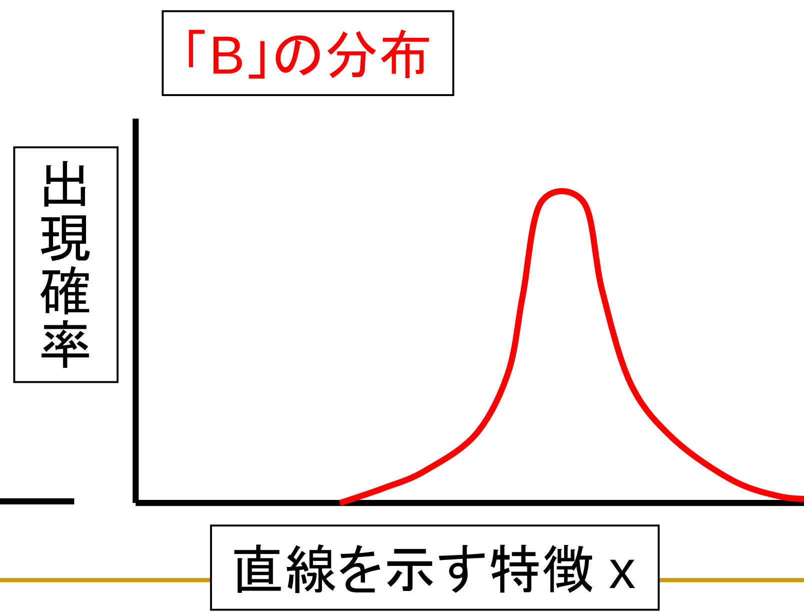
# 特徴の出現確率（確率密度関数）①

- p(x|B)
  - 「B」という条件のもとで，特徴 x=0.3 の出現する確率
  - （例） p(x=0.3|B) = 45/100
- p(x|8)
  - 「8」という条件のもとで，特徴 x=0.7 の出現する確率
  - （例） p(x=0.7|8) = 45/100

# 特徴の出現確率(確率密度関数)②

- 特徴の分布(確率密度関数)を知るためには？
  - 無限個のパターンから特徴をとらなければならない
  - 現実的には不可能なため，分布を仮定する
    - パラメータも推定する
  - 例えば正規分布



# 特徴の分布からの認識

- 未知のパターンから特徴 $x$ を求め，「8」か「B」のどちらに属するのかを求めたい

↓

- 特徴の出現確率（$p(x|8)$, $p(x|B)$）を利用する

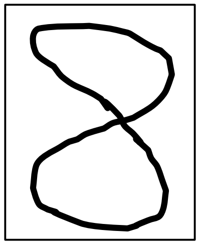
$\boxed{\text{統計的パターン認識}}$

# 出現確率を用いた認識方法

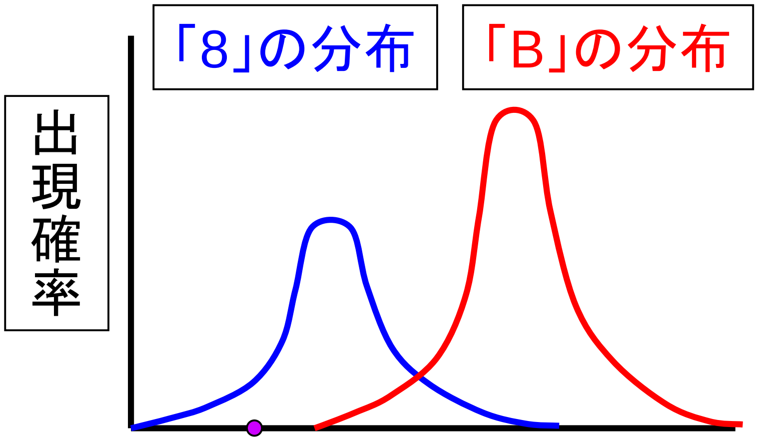
正規分布

マハラノビス距離

# 出現確率を用いた認識①



抽出した特徴xの値は 0.3

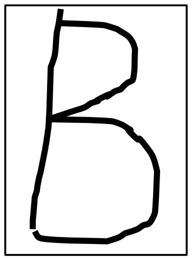



x の値は0.3

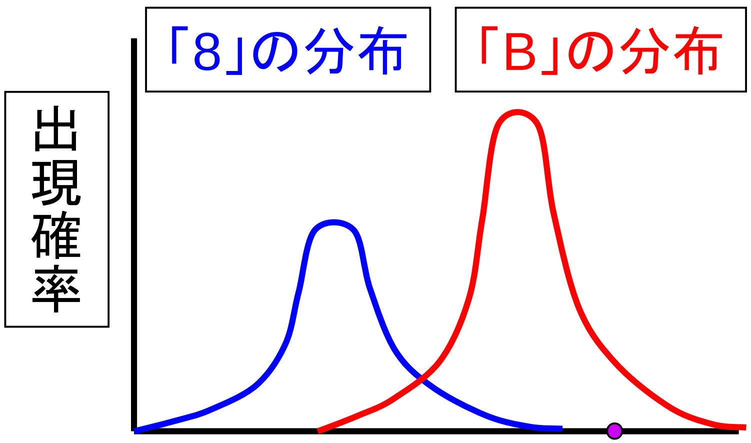
p(x|8) > p(x|B)
→「8」と認識

*少々，正しくない説明をしていますが，今は気にしないで下さい

# 出現確率を用いた認識②



抽出した特徴xの値は 0.8



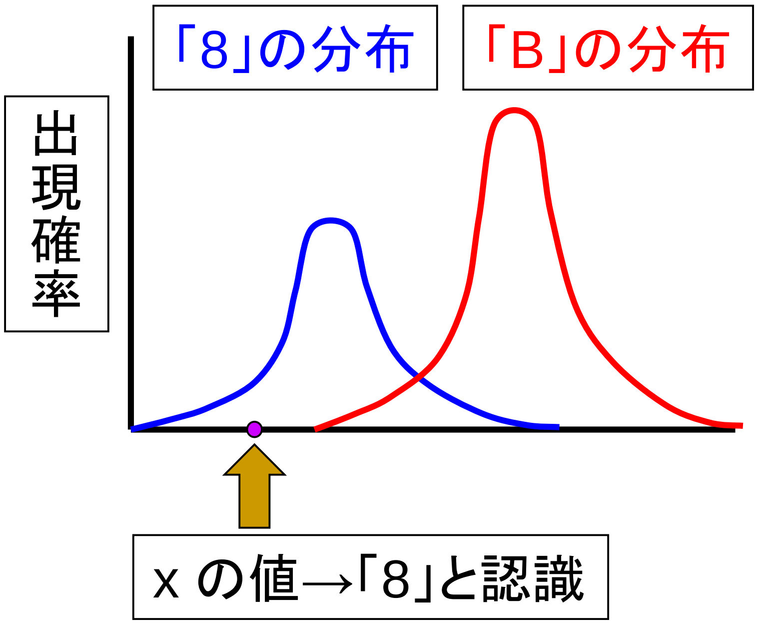

x の値は0.8

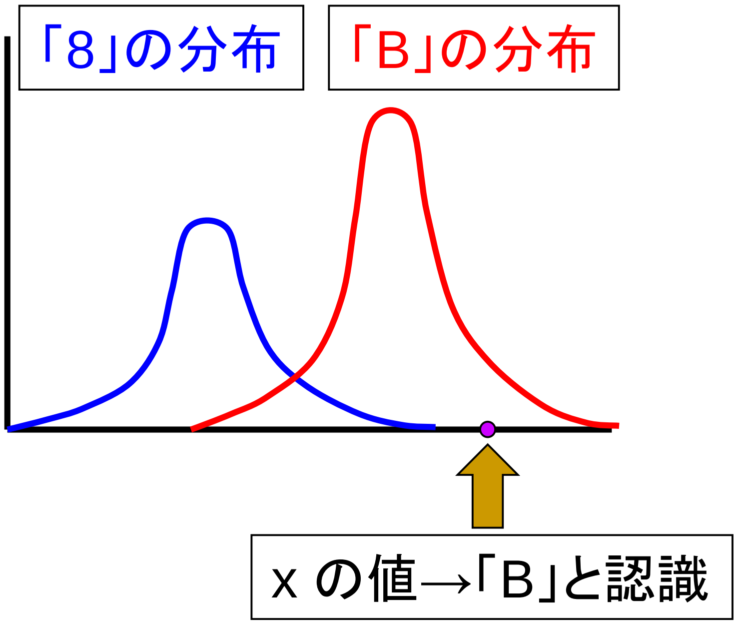
$p(x|8) < p(x|B)$
→「B」と認識

# 統計的パターン認識③

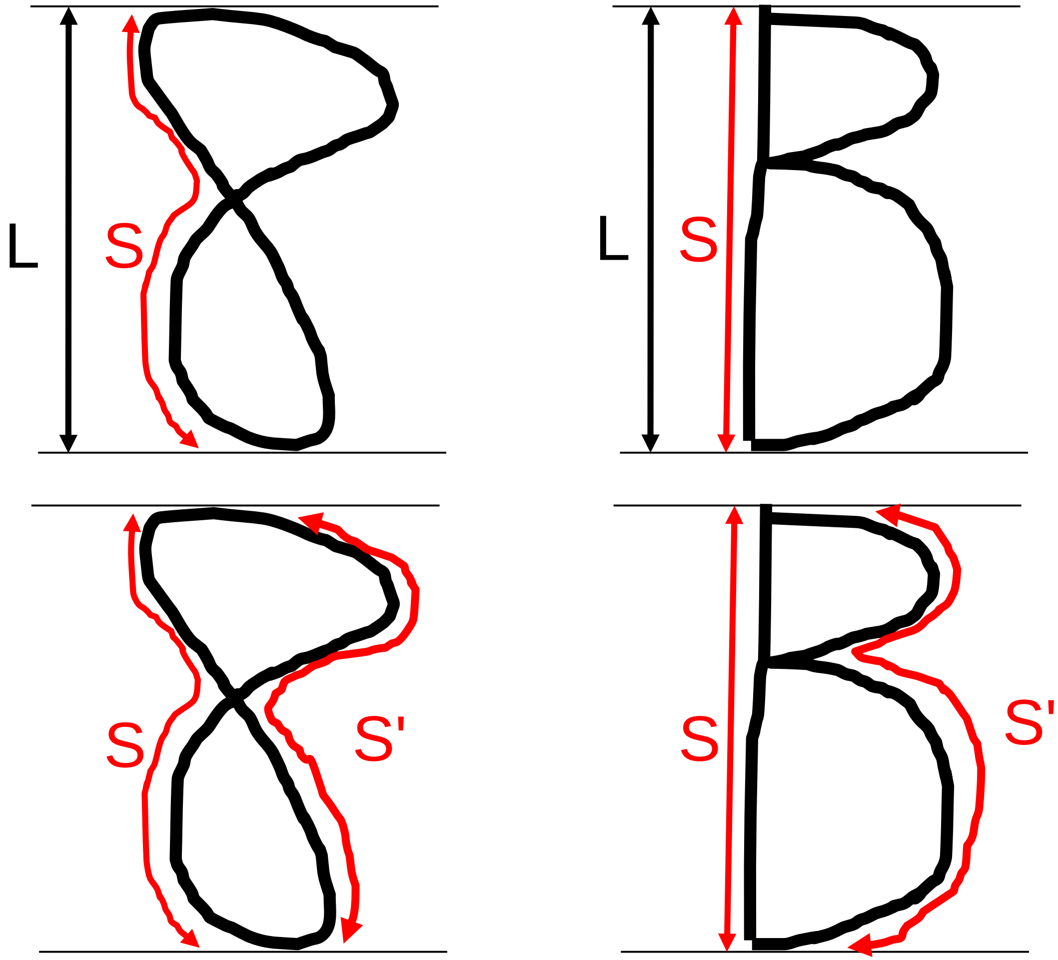
p(x|8) > p(x|B) の場合
→「8」と認識

p(x|8) < p(x|B) の場合
→「B」と認識



# 新しい特徴抽出



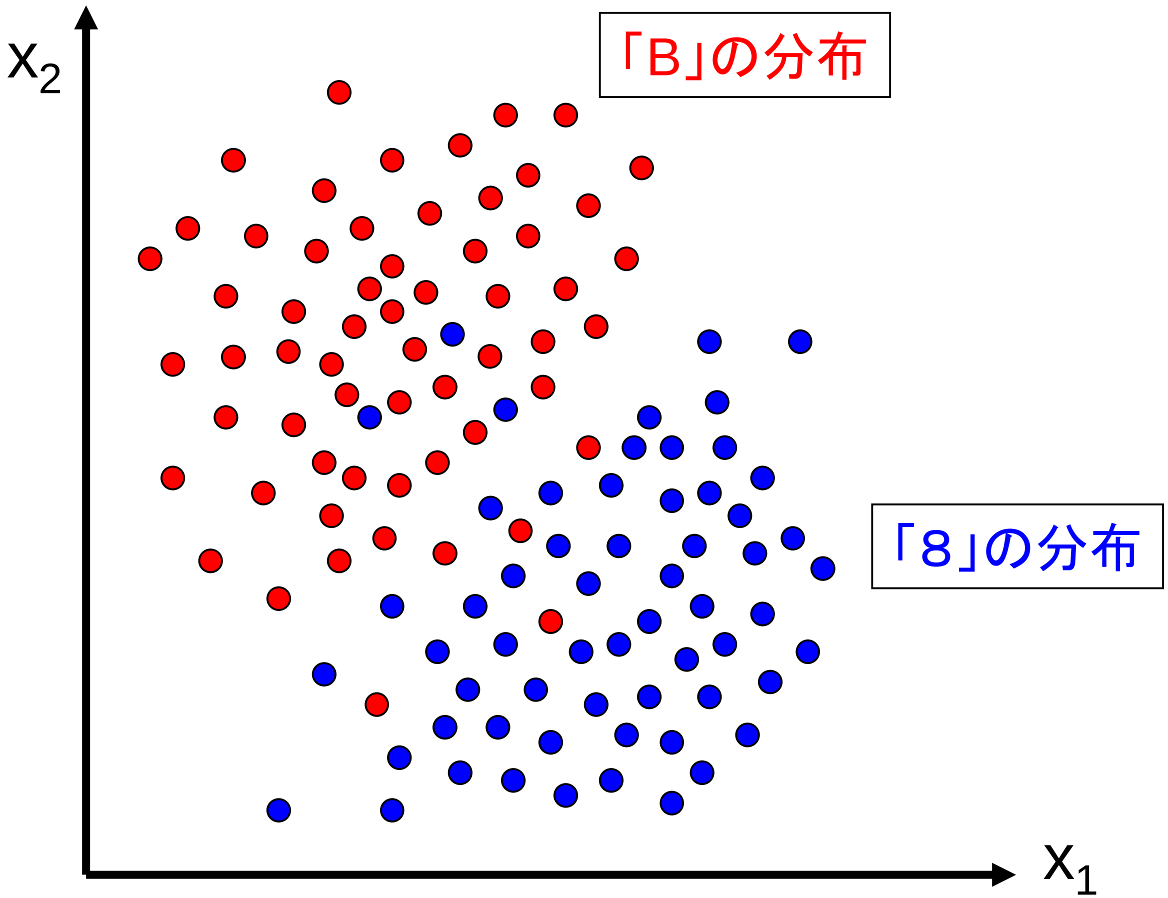

特徴量は二変数

$$x_1 = \frac{L}{S}$$

$$x_2 = \frac{S}{S'}$$

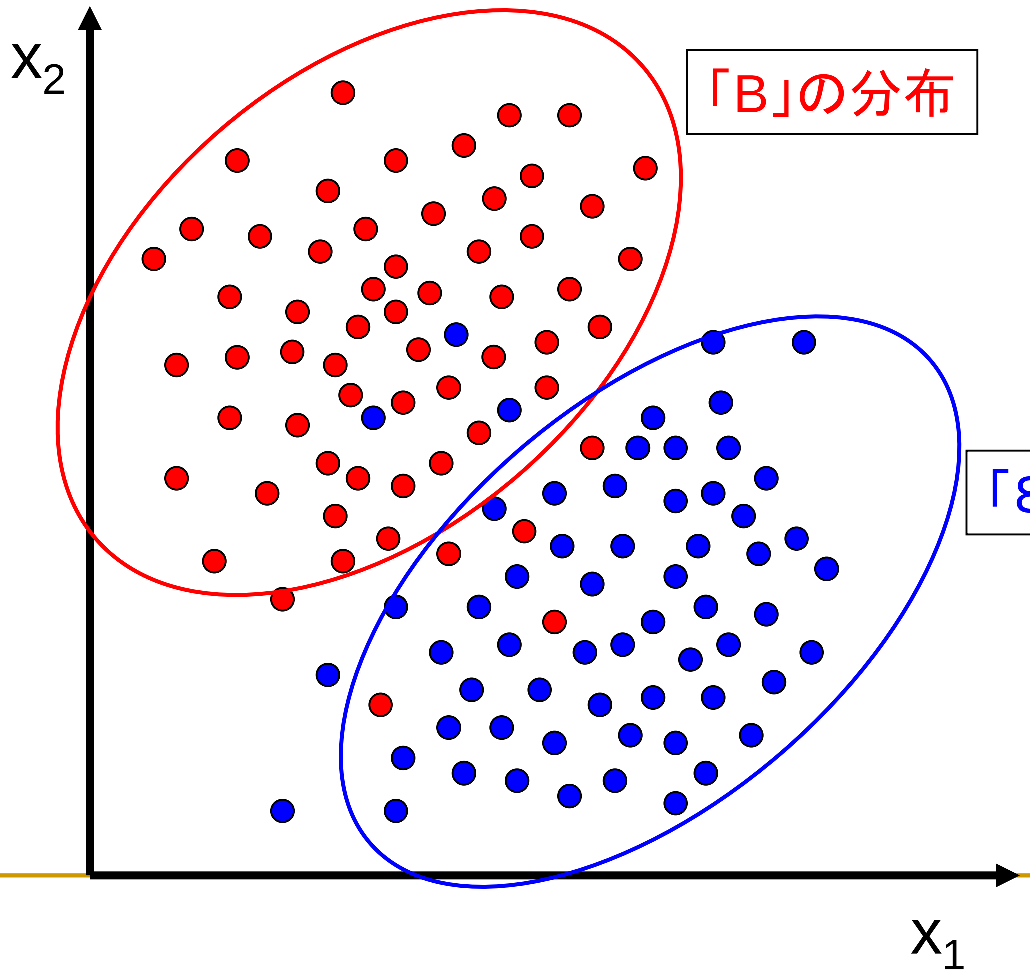
# 特徴空間(復習)

- 「8」と「B」の特徴ベクトル$(x_1, x_2)$を平面上に表現



# 特徴空間(復習)

- クラスター
  - 特徴空間上において，クラスごとにまとまって観測される塊



# 特徴の分布の仮定①

- 特徴の確率密度関数
  - 特徴 $x_i$（i=1,2）の出現確率は一般的には分からない
  - $x_i$（i=1,2）の出現確率は独立と仮定する

$$p(\mathbf{x}\,|\,8) = p(x_1, x_2\,|\,8) = p(x_1\,|\,8)\,p(x_2\,|\,8)$$

$$p(\mathbf{x}\,|\,B) = p(x_1, x_2\,|\,B) = p(x_1\,|\,B)\,p(x_2\,|\,B)$$

# 特徴の分布の仮定②

- 特徴の確率密度関数
  - さらに，p($x_i$|8) とp($x_i$|B) が平均*$m_i^8$と$m_i^B$，標準偏差σの正規分布に従うと仮定する（i=1,2）

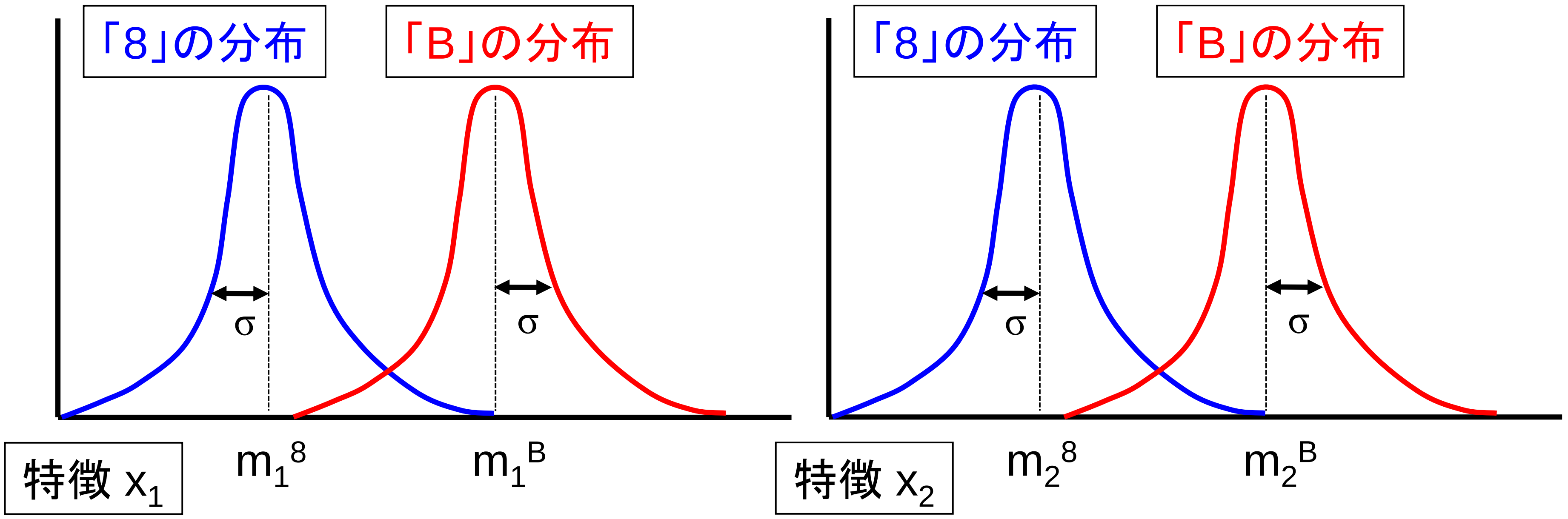
$$m_i^8 = \frac{1}{n_8} \sum_{\mathbf{x} \in \chi_8} x_i$$

$$p(x_i \mid 8) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[ -\frac{(x_i - m_i^8)^2}{2\sigma^2} \right]$$

$$p(\mathbf{x} \mid 8) = p(x_1 \mid 8)\, p(x_2 \mid 8) = \frac{1}{2\pi\sigma^2} \exp\left( -\frac{1}{2\sigma^2} \left\| \mathbf{x} - \mathbf{m}_8 \right\|^2 \right)$$

# 特徴の分布の仮定②

正規分布

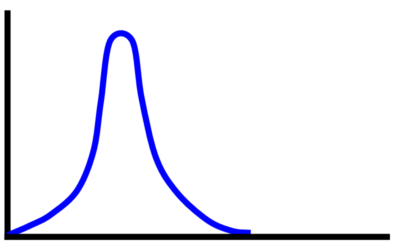


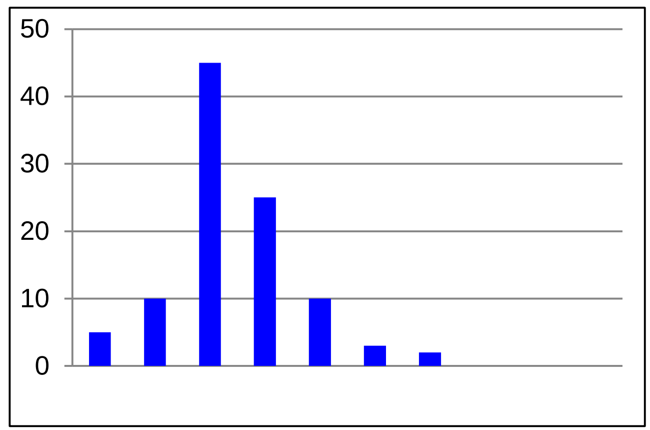
# 確率密度関数のパラメータの推定

確率密度関数を仮定した場合，パラメータ（平均，標準偏差）を推定しなければならない*

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

*次回説明します

# 標準偏差が同じという仮定での認識方法①

- 認識方法
  - p(**x**|8) > p(**x**|B) → 「8」と認識

$$\frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{m}_8\right\|^2\right) > \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{m}_B\right\|^2\right)$$

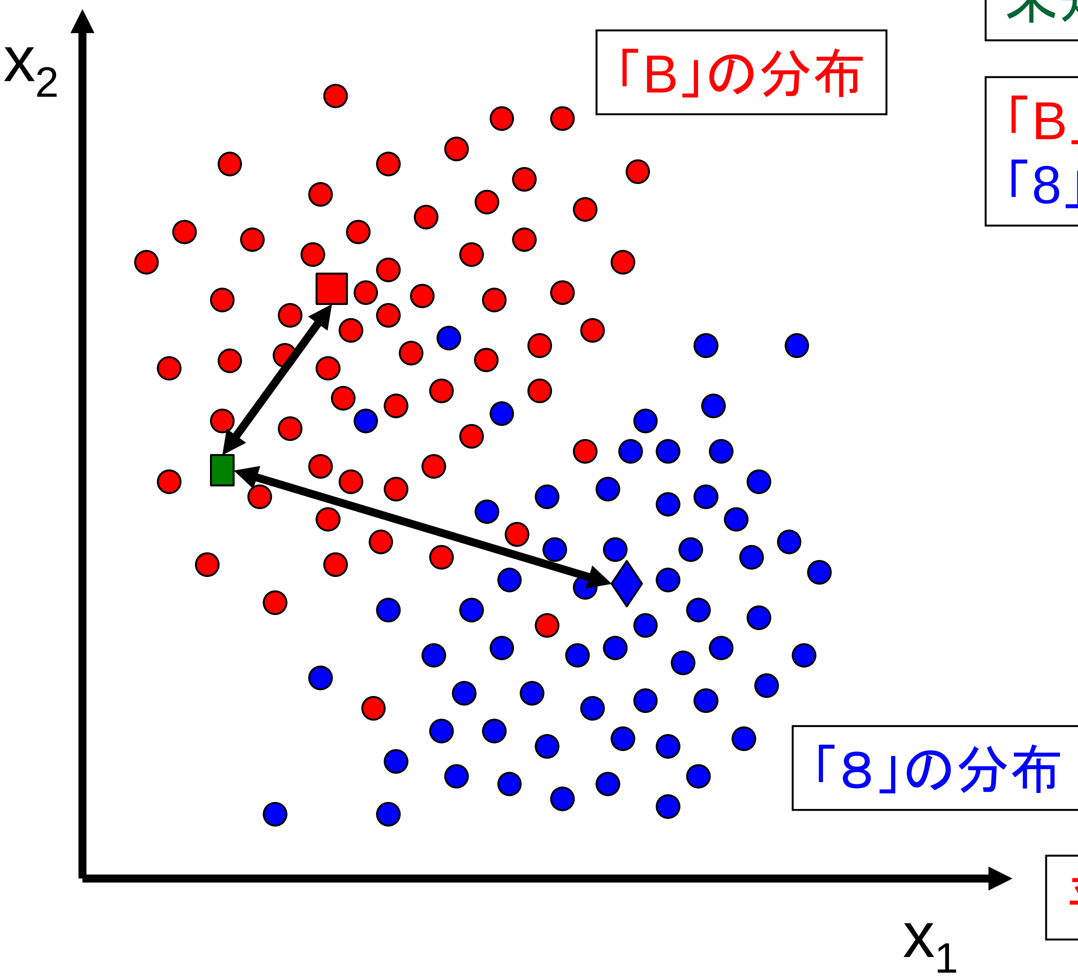
$$\Rightarrow \boxed{\left\|\mathbf{x}-\mathbf{m}_8\right\|^2 < \left\|\mathbf{x}-\mathbf{m}_B\right\|^2}$$

  - p(**x**|8) < p(**x**|B) → 「B」と認識

$$\frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{m}_8\right\|^2\right) < \frac{1}{2\pi\sigma^2}\exp\left(-\frac{1}{2\sigma^2}\left\|\mathbf{x}-\mathbf{m}_B\right\|^2\right)$$

$$\Rightarrow \boxed{\left\|\mathbf{x}-\mathbf{m}_8\right\|^2 > \left\|\mathbf{x}-\mathbf{m}_B\right\|^2}$$

# 標準偏差が同じという仮定での認識方法②




未知のパターンの特徴ベクトル **x**

「B」の平均ベクトル $\mathbf{m}_b$

「8」の平均ベクトル $\mathbf{m}_8$

$$\|\mathbf{x}-\mathbf{m}_8\| > \|\mathbf{x}-\mathbf{m}_B\|$$

→「B」と認識

$$\|\mathbf{x}-\mathbf{m}_8\| < \|\mathbf{x}-\mathbf{m}_B\|$$

→「8」と認識

平均ベクトルとのユークリッド距離

# 標準偏差が同じという仮定での認識方法③

- 標準偏差が同じという仮定での認識方法

↓

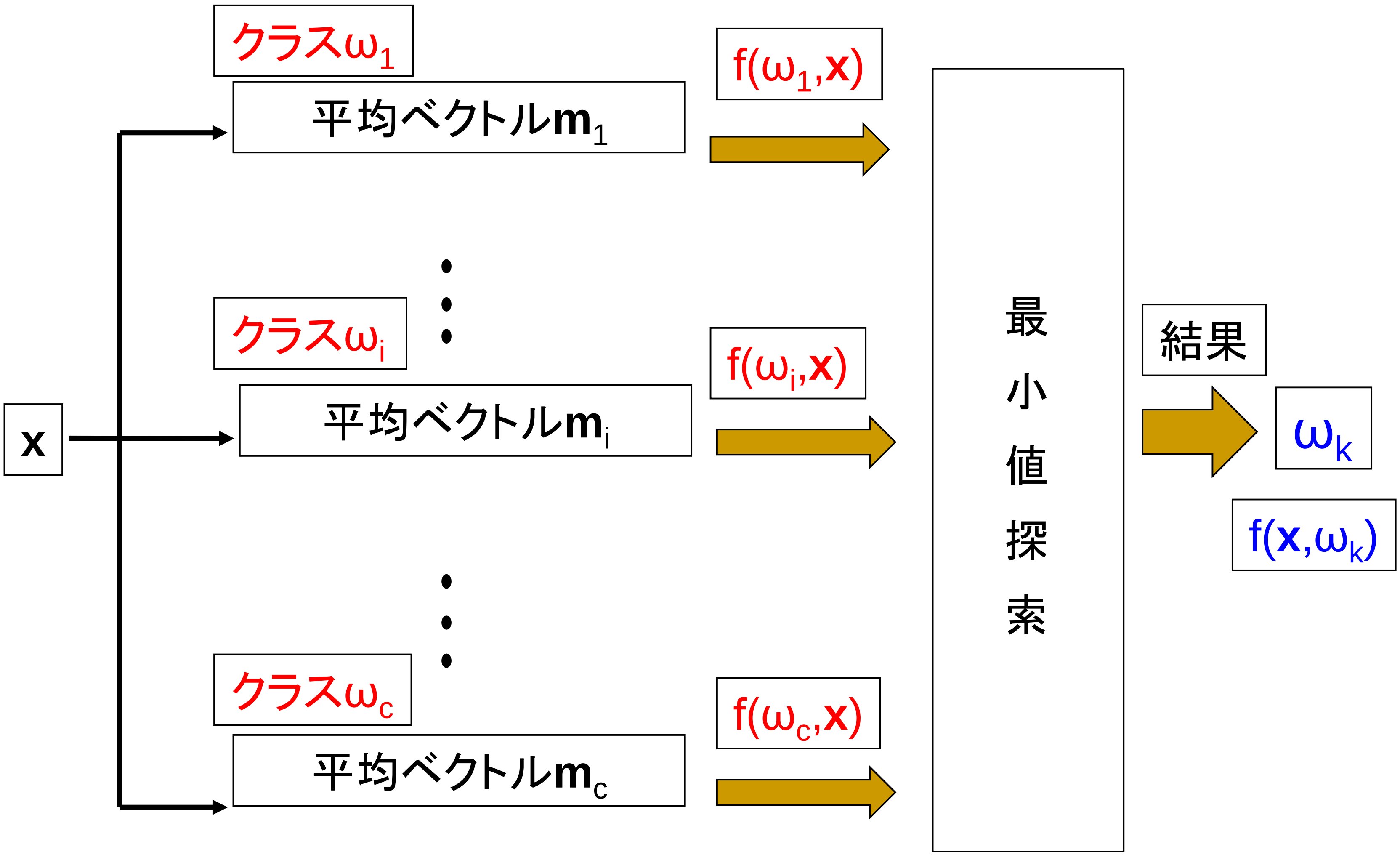
- 各クラスのプロトタイプの特徴ベクトルを平均ベクトルとした最近傍法と等価

# 多クラスへの拡張（標準偏差が同じという仮定）

- c個のクラス $\omega_i$（i=1,2,･･･,c)
  - クラス $\omega_i$ の平均ベクトル $\mathbf{m}_i$（d次元）
- 未知のパターンの特徴ベクトル **x**
  - 特徴ベクトル **x** はどのクラスに属するか

$$\boxed{\begin{aligned}&\min_{i=1,2,\cdots,c} \|\mathbf{x}-\mathbf{m}_i\| \\ &= \|\mathbf{x}-\mathbf{m}_k\| \Rightarrow \mathbf{x} \in \omega_k\end{aligned}}$$

# 多クラスへの拡張（標準偏差が同じという仮定）




$$f(\omega_i, \mathbf{x}) = \| \mathbf{x} - \mathbf{m}_i \|$$

# 特徴の分布の仮定③

- 特徴の確率密度関数
  - 「8」の特徴ベクトル **x** は，平均$\mathbf{m}_8$，分散共分散行列$\Sigma_8$の正規分布に従う

$$\Sigma_8 = \frac{1}{n_8}\sum_{\mathbf{x}\in\chi_8}(\mathbf{x}-\mathbf{m}_8)(\mathbf{x}-\mathbf{m}_8)^t$$

$n_8$:「8」のパターン数

$$p(\mathbf{x}\,|\,8) = \frac{1}{2\pi\,|\,\Sigma_8\,|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\,\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)\right)$$

  - 「B」の特徴ベクトル **x** は，平均$\mathbf{m}_B$，分散共分散行列$\Sigma_B$の正規分布に従う

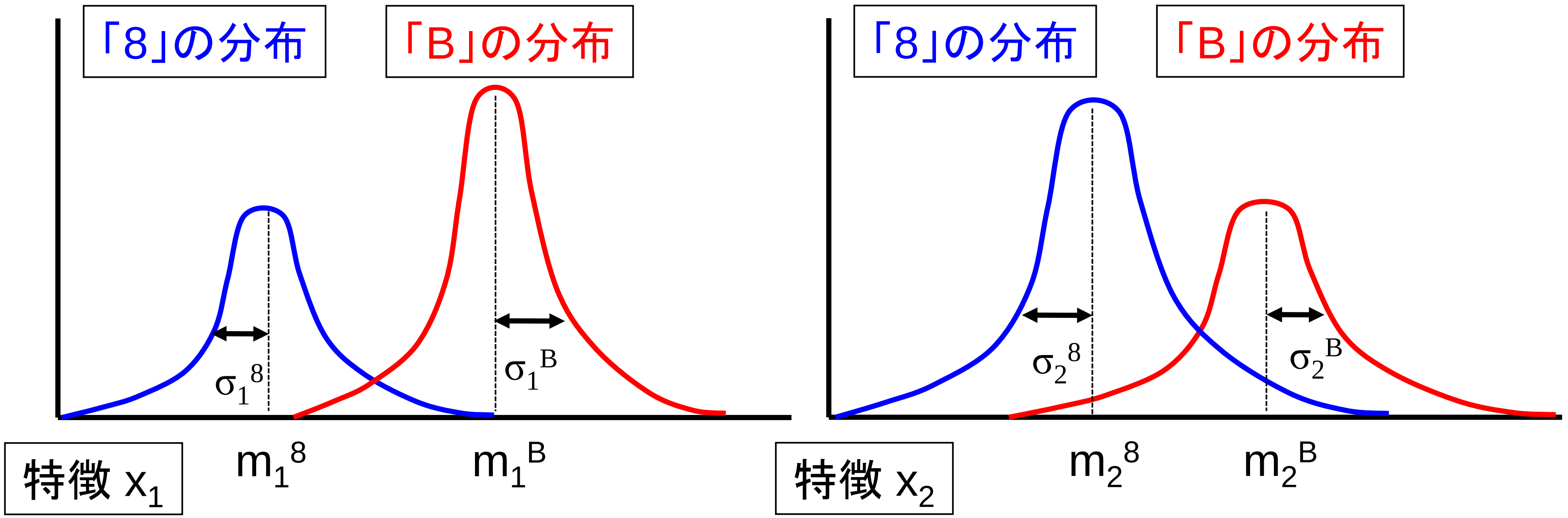
$$\Sigma_B = \frac{1}{n_B}\sum_{\mathbf{x}\in\chi_B}(\mathbf{x}-\mathbf{m}_B)(\mathbf{x}-\mathbf{m}_B)^t$$

$n_B$:「B」のパターン数

$$p(\mathbf{x}\,|\,B) = \frac{1}{2\pi\,|\,\Sigma_B\,|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\,\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)\right)$$

# 特徴の分布の仮定③

正規分布



# パラメータの推定

- （繰り返しですが）分布を仮定した場合，パラメータも未知なため推定しなければならない

平均ベクトル

$$\mathbf{m}^{8} = \frac{1}{n_8} \sum_{\mathbf{x} \in \chi_8} \mathbf{x}$$

分散共分散行列

$$\Sigma_8 = \frac{1}{n_8} \sum_{\mathbf{x} \in \chi_8} (\mathbf{x} - \mathbf{m}_8)(\mathbf{x} - \mathbf{m}_8)^t$$

# 正規分布に従うという仮定での認識方法①

- 認識方法
  - p(**x**|8) > p(**x**|B) → 「8」と認識

$$\frac{1}{2\pi|\Sigma_8|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)\right) > \frac{1}{2\pi|\Sigma_B|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)\right)$$

両辺，対数をとると

$$-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)-\log 2\pi-\frac{1}{2}\log|\Sigma_8| > -\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)-\log 2\pi-\frac{1}{2}\log|\Sigma_B|$$

$$\Downarrow$$

$$\boxed{(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)+\log|\Sigma_8| < (\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)+\log|\Sigma_B|}$$

# 正規分布に従うという仮定での認識方法②

- 認識方法
  - p(**x**|8) < p(**x**|B) → 「B」と認識

$$\frac{1}{2\pi|\Sigma_8|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)\right)<\frac{1}{2\pi|\Sigma_B|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)\right)$$

両辺，対数をとると

$$-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)-\log 2\pi-\frac{1}{2}\log|\Sigma_8|<-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)-\log 2\pi-\frac{1}{2}\log|\Sigma_B|$$

$$\Downarrow$$

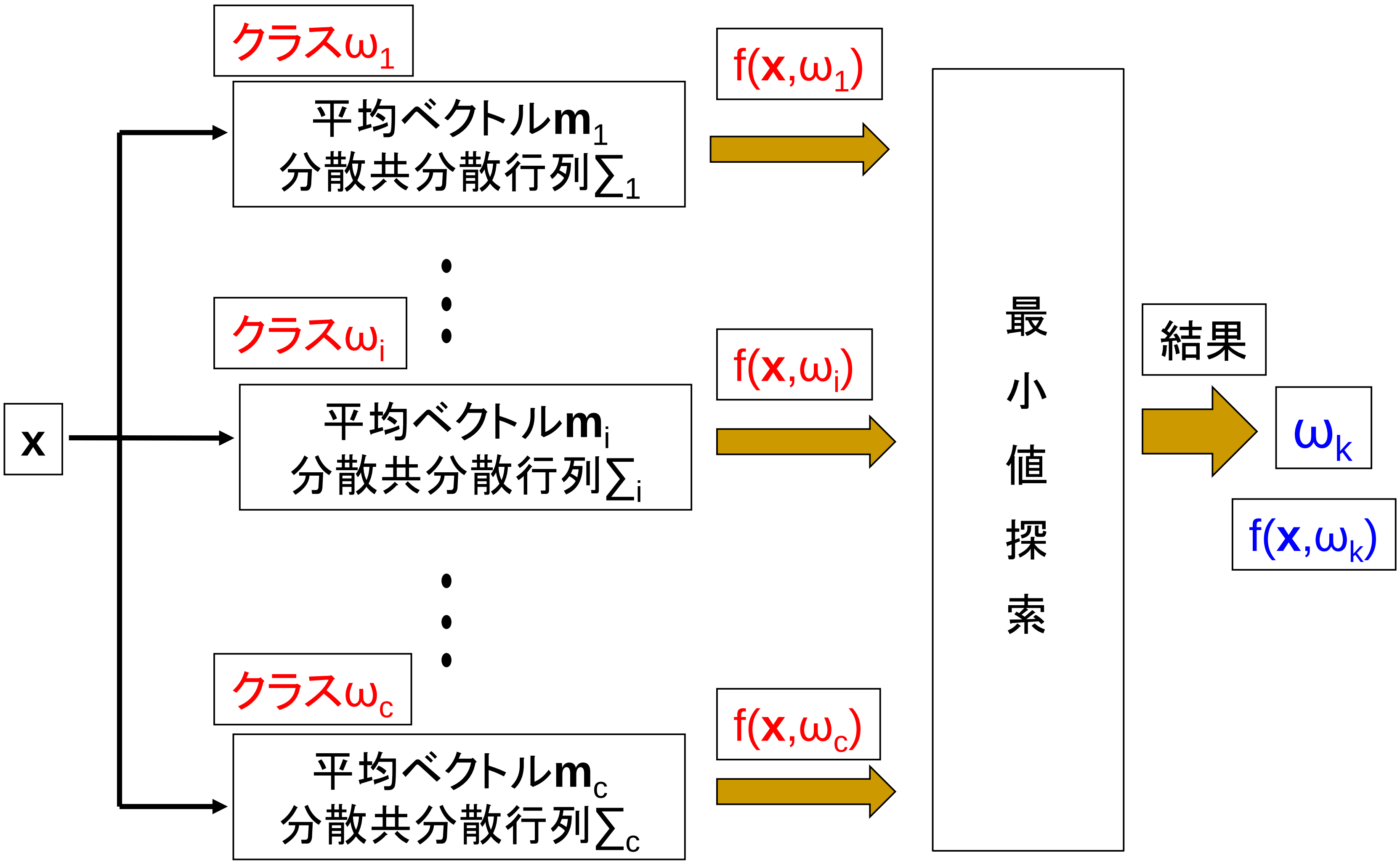
$$\boxed{(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)+\log|\Sigma_8|>(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)+\log|\Sigma_B|}$$

# 多クラスへの拡張(正規分布に従うという仮定)①

- c個のクラス $\omega_i$(i=1,2,・・・,c)
  - クラス $\omega_i$ の平均ベクトル $\mathbf{m}_i$(d次元)
  - 分散共分散行列 $\Sigma_i$
- 未知のパターンの特徴ベクトル **x**
  - 特徴ベクトル **x** はどのクラスに属するか

$$\min_{i=1,2,\cdots,c} (\mathbf{x}-\mathbf{m}_i)^t \Sigma_i^{-1} (\mathbf{x}-\mathbf{m}_i) + \log|\Sigma_i|$$
$$= (\mathbf{x}-\mathbf{m}_k)^t \Sigma_k^{-1} (\mathbf{x}-\mathbf{m}_k) + \log|\Sigma_k| \Rightarrow \mathbf{x} \in \omega_k$$

# 多クラスへの拡張（正規分布に従うという仮定）②




$$f(\mathbf{x},\omega_i)=(\mathbf{x}-\mathbf{m}_i)^t\Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)+\log|\Sigma_i|$$

# マハラノビス距離による認識方法①

- 認識方法

マハラノビス距離

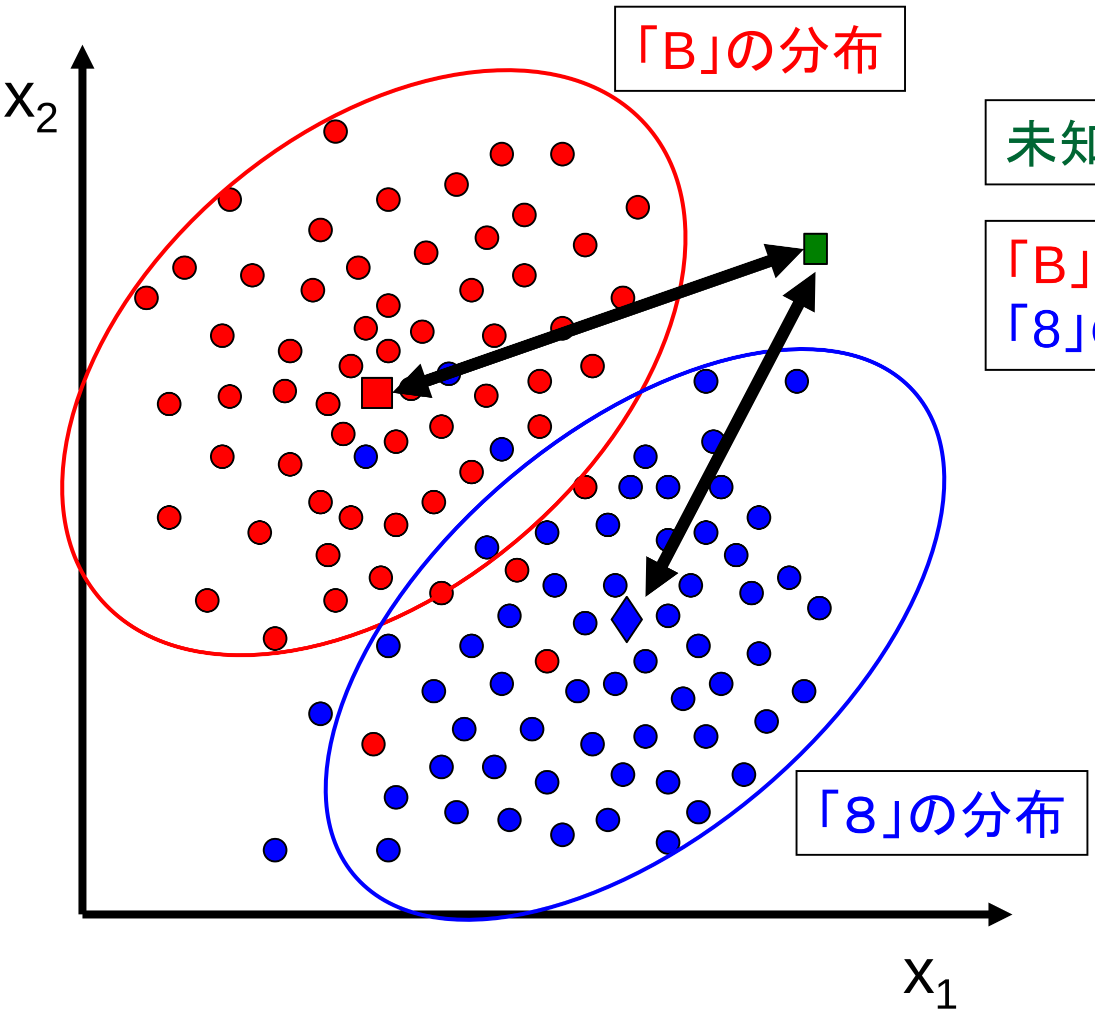
$$(\mathbf{x}-\mathbf{m}_8)^t \Sigma_8^{-1} (\mathbf{x}-\mathbf{m}_8) < (\mathbf{x}-\mathbf{m}_B)^t \Sigma_B^{-1} (\mathbf{x}-\mathbf{m}_B)$$

⇨ 「8」と認識

$$(\mathbf{x}-\mathbf{m}_8)^t \Sigma_8^{-1} (\mathbf{x}-\mathbf{m}_8) > (\mathbf{x}-\mathbf{m}_B)^t \Sigma_B^{-1} (\mathbf{x}-\mathbf{m}_B)$$

⇨ 「B」と認識

# マハラノビス距離による認識方法②




未知のパターンの特徴ベクトル **x**

「B」の平均ベクトル $\mathbf{m}_b$

「8」の平均ベクトル $\mathbf{m}_8$

マハラノビス距離
分布との距離を示す

# マハラノビス距離①

- 分散共分散行列Σ

$$\Sigma\Phi = \lambda\Phi$$

固有値分解

$$\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_d)$$

固有値

$$\Phi = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_d)$$

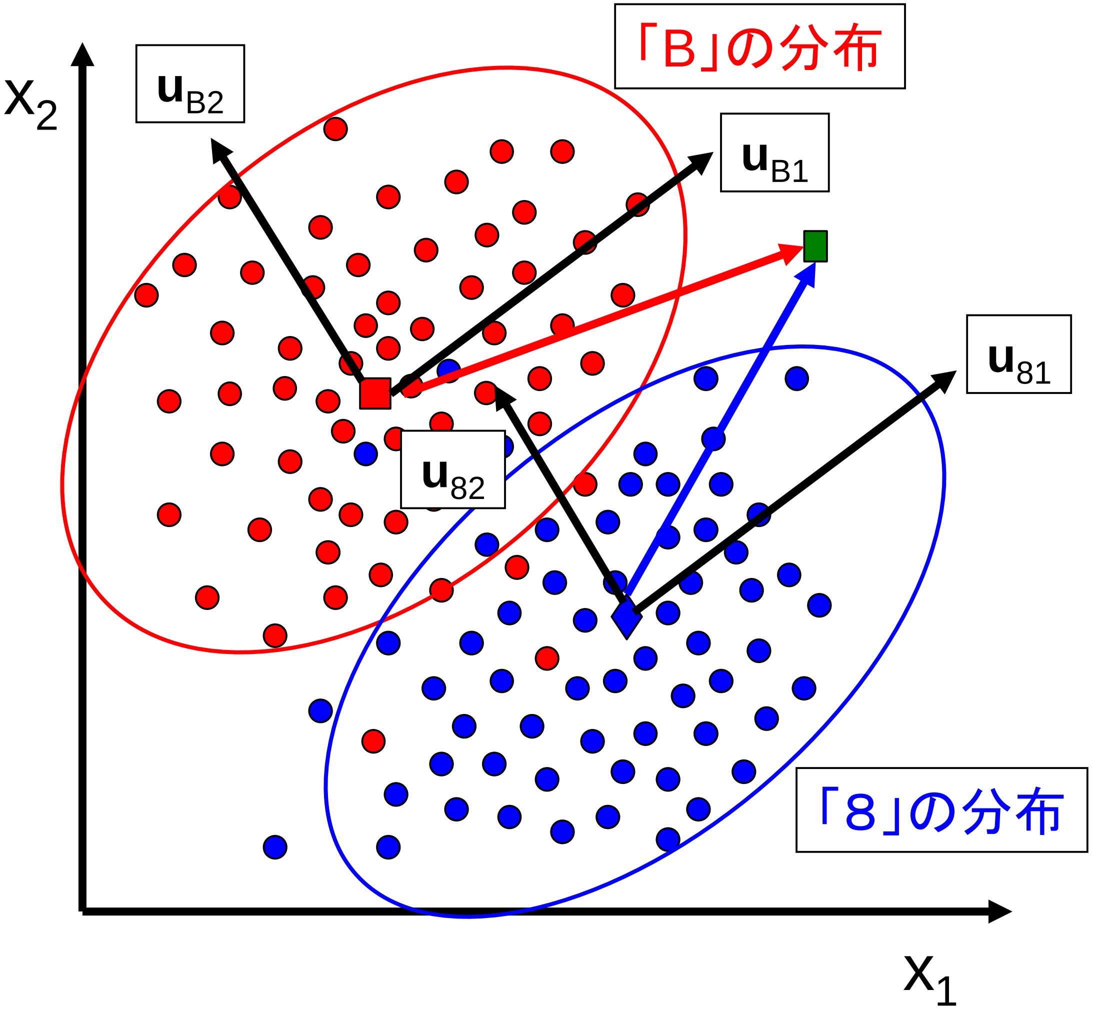
固有ベクトル

$$\Sigma = \sum_{i=1}^{d} \lambda_i \mathbf{u}_i \mathbf{u}_i^t$$

$$\Sigma^{-1} = \sum_{i=1}^{d} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^t$$

$$(\mathbf{x}-\mathbf{m})^t \Sigma^{-1} (\mathbf{x}-\mathbf{m}) = \sum_{i=1}^{d} \frac{(\mathbf{u}_i^t(\mathbf{x}-\mathbf{m}))^2}{\lambda_i}$$

# マハラノビス距離②




$$(\mathbf{x}-\mathbf{m})^t \Sigma^{-1} (\mathbf{x}-\mathbf{m}) = \sum_{i=1}^{d} \frac{(\mathbf{u}_i^t (\mathbf{x}-\mathbf{m}))^2}{\lambda_i}$$

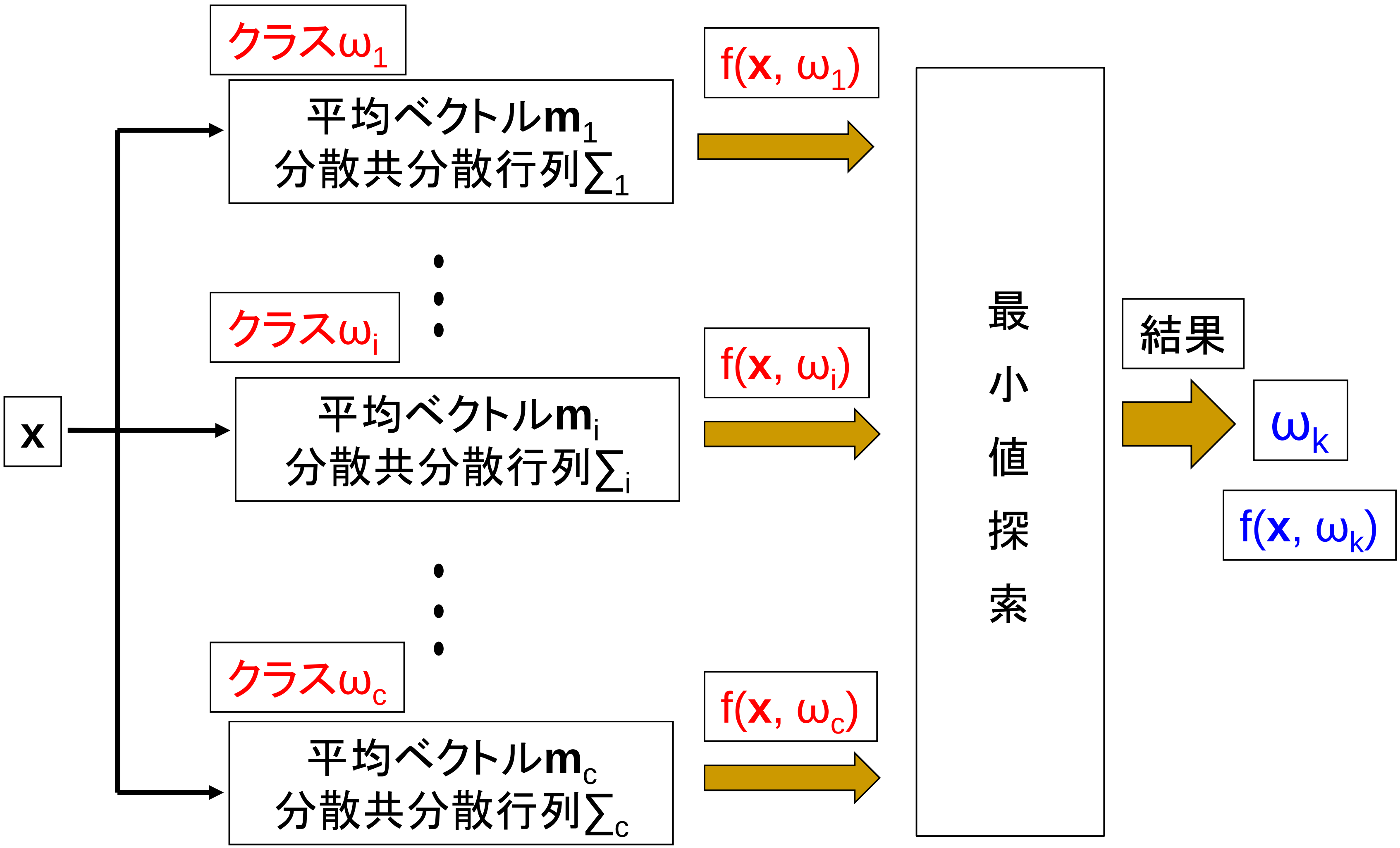
# マハラノビス距離による認識方法③

- c個のクラス $\omega_i$ (i=1,2,・・・,c)
  - クラス $\omega_i$ の平均ベクトル $\mathbf{m}_i$ (d次元)
  - 分散共分散行列 $\sum_i$
- 未知のパターンの特徴ベクトル **x**
  - 特徴ベクトル **x** はどのクラスに属するか

$$\min_{i=1,2,\cdots,c} (\mathbf{x}-\mathbf{m}_i)^t \Sigma_i^{-1} (\mathbf{x}-\mathbf{m}_i) = (\mathbf{x}-\mathbf{m}_k)^t \Sigma_k^{-1} (\mathbf{x}-\mathbf{m}_k) \Rightarrow \mathbf{x} \in \omega_k$$

分布とのマハラノビス距離が最小となるクラス

# マハラノビス距離による認識方法④

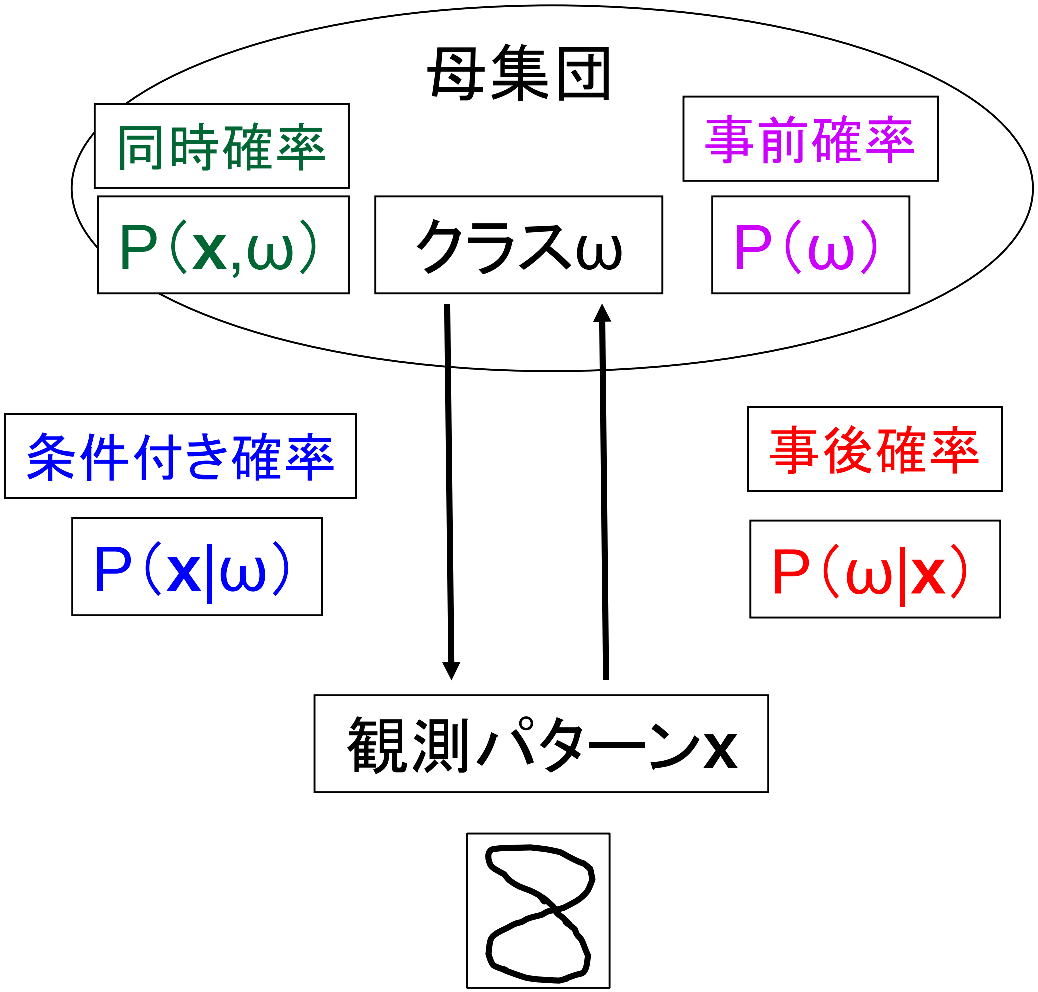



マハラノビス距離 $f(\mathbf{x}, \omega_i) = (\mathbf{x} - \mathbf{m}_i)^t \Sigma_i^{-1} (\mathbf{x} - \mathbf{m}_i)$

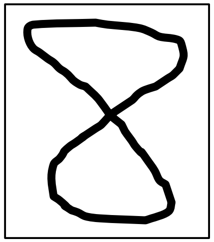
# ベイズ決定則

生成モデルと識別モデル

# 統計的パターン認識で用いる確率



# パターン認識の手順①



特徴は…
○が上下に二つ並んでいる

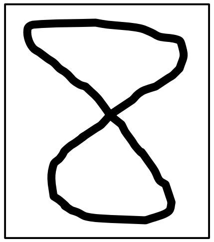
数字の「8」と認識

特徴は…
髪が短い

「男性」と認識

観測パターンを観測，特徴を抽出後，認識を行なう

# パターン認識の手順②



特徴は…
○が上下に二つ並んでいる

数字の「8」と
認識

「○が上下に二つ並んでいる場合，8である確率」
＞「○が上下に二つ並んでいる場合，Bである確率」

特徴は…
髪が短い

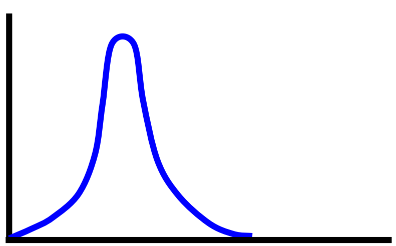
「男性」と認識

「髪が短い場合，男性である確率」
＞「髪が短い場合，女性である確率」

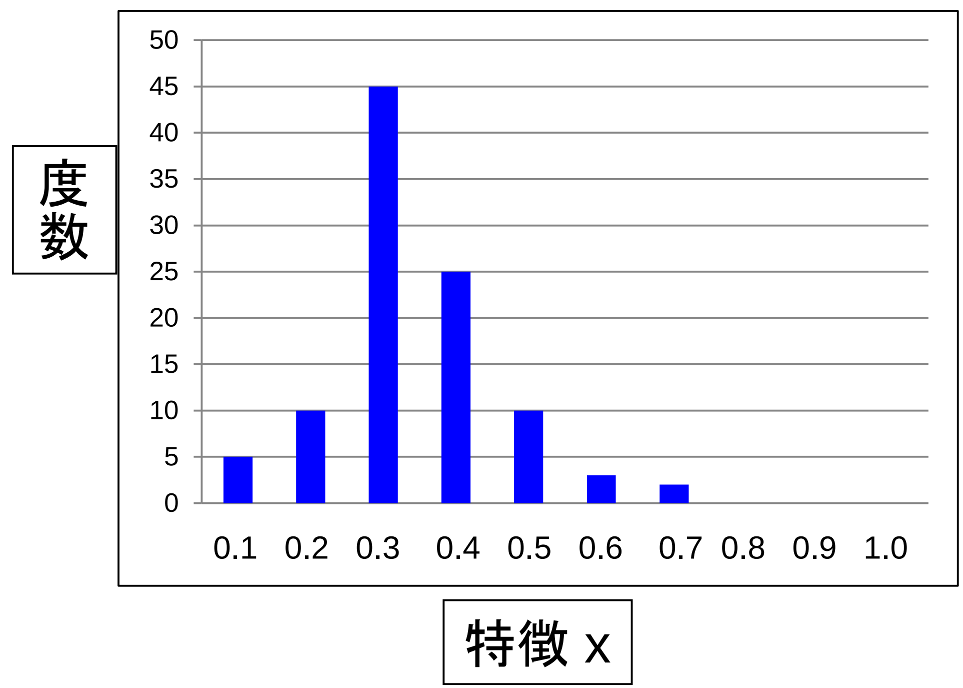
# これまで認識に利用していた確率密度関数

- p(**x**|8)
  - 「8」というクラスが与えられた場合，特徴xが出現する確率（条件つき確率）

「8」の母集団



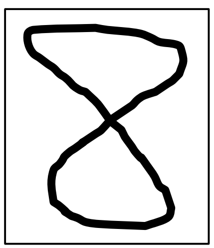
特徴xの出現する
真の分布



# パターン認識に必要な確率密度関数

- p(8|**x**)
  - x という特徴が出現した場合，「8」が出現する確率
  - 事後確率と呼ぶ

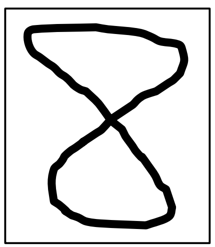
認識に必要な確率密度関数



→ 観測，特徴抽出 → 特徴ベクトル **x** → p(8|x) = 0.8
p(B|x) = 0.2 → p(8|x)>p(B|x) → 数字の「8」と認識

# ベイズ決定則

- 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）

# 事後確率の求め方

- ベイズの定理

p(8), p(B)
事前確率と呼ぶ

$$p(8|\mathbf{x}) = \frac{p(\mathbf{x}|8)\,p(8)}{p(\mathbf{x})}$$

$$p(B|\mathbf{x}) = \frac{p(\mathbf{x}|B)\,p(B)}{p(\mathbf{x})}$$

$$p(8|\mathbf{x}) + p(B|\mathbf{x}) = 1$$

$$p(8) + p(B) = 1$$

$$p(\mathbf{x}) = p(\mathbf{x}|8)\,p(8) + p(\mathbf{x}|B)\,p(B)$$

$p(\mathbf{x})$ は同じなので,

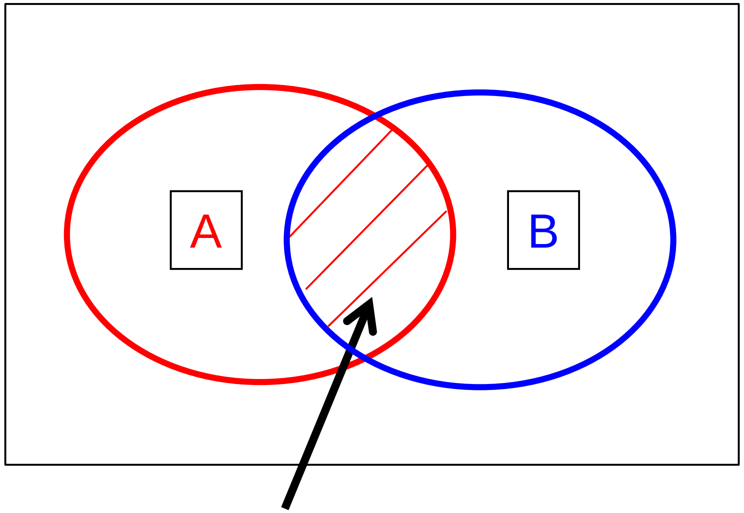
$$p(\mathbf{x}|8)\,p(8) > p(\mathbf{x}|B)\,p(B)$$

→ 「8」と認識

$$p(\mathbf{x}|8)\,p(8) < p(\mathbf{x}|B)\,p(B)$$

→ 「B」と認識

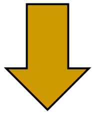
# ベイズの定理




$p(A \cap B)$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$



$$p(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

# ベイズ決定則による認識方法①

$$p(\omega_i) = \frac{n_i}{\sum_{i=1}^{c} n_i}$$

$n_i$ クラス$\omega_i$のデータ数

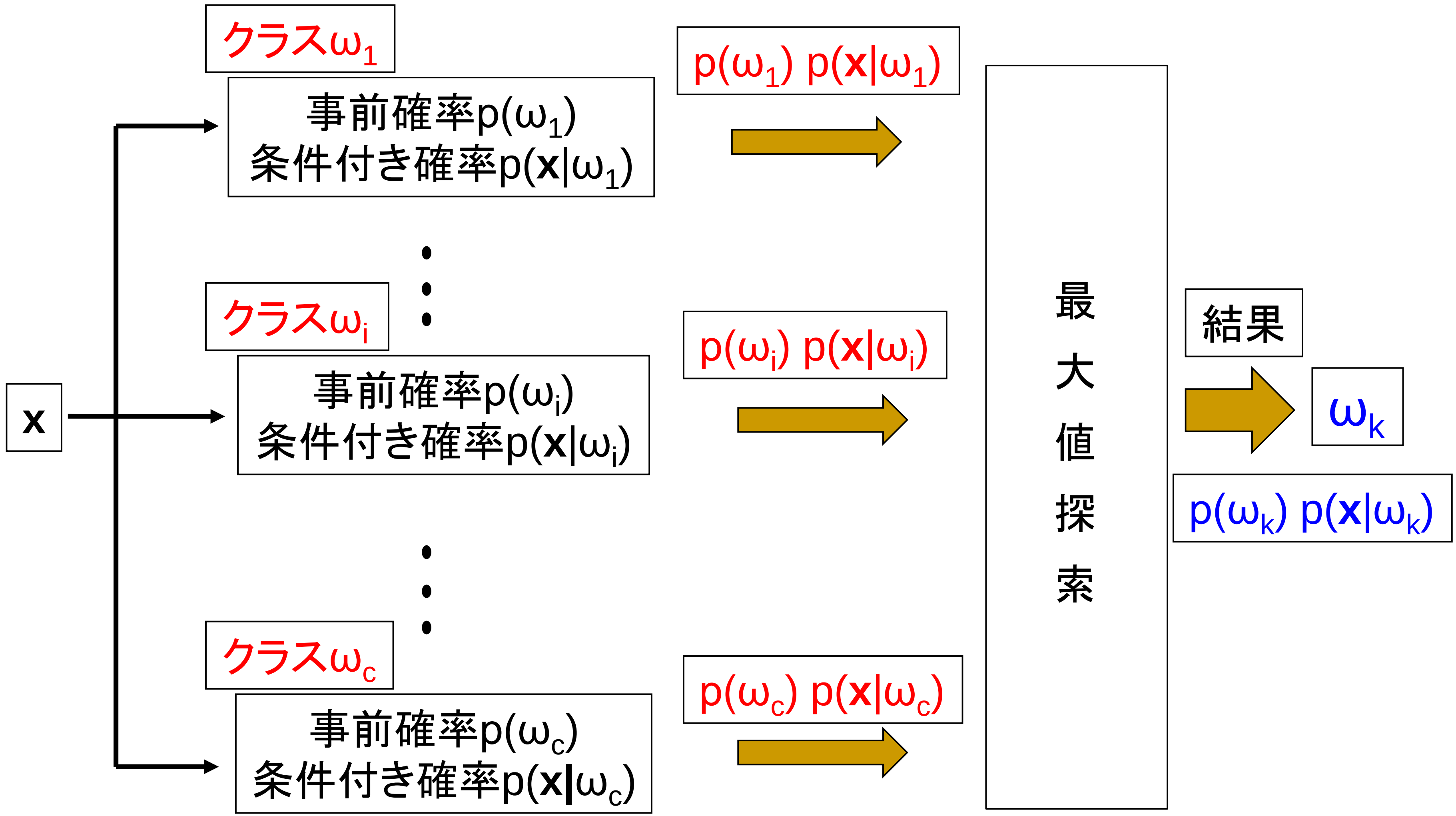
- c個のクラス$\omega_i$（i=1,2,・・・,c)
  - 事前確率 $p(\omega_i)$ は既知
- 未知のパターンの特徴ベクトル **x**（d次元）
  - 各クラスごとの条件付き確率 $p(\mathbf{x}|\omega_i)$
  - 特徴ベクトル **x** はどのクラスに属するか

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

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

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

# ベイズ決定則による認識方法②



# 事後確率の計算

- 事後確率を計算するには？
  - 事前確率 $p(\omega_i)$ は既知

$$p(\omega_i) = \frac{n_i}{\sum_{i=1}^{c} n_i}$$

$n_i$ クラス$\omega_i$のデータ数

- 条件付き確率 $p(\mathbf{x}|\omega_i)$ を求める必要がある
  - 確率密度関数は仮定する
  - パラメータの推定が必要

# 確率密度関数の仮定

- 特徴の確率密度関数
  - 「8」の特徴ベクトル **x** (d次元)は，独立*でかつ平均$\mathbf{m}_8$，分散共分散行列$\Sigma_8$の正規分布に従う

$$p(\mathbf{x}|8) = p(x_1, x_2, \cdots, x_d|8) = p(x_1|8)\,p(x_2|8)\cdots p(x_d|8)$$

$$= \frac{1}{(2\pi)^{d/2}|\Sigma_8|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)\right)$$

d次元の正規分布

  - 「B」の特徴ベクトル **x** は，独立でかつ平均$\mathbf{m}_B$，分散共分散行列$\Sigma_B$の正規分布に従う

$$p(\mathbf{x}|B) = \frac{1}{(2\pi)^{d/2}|\Sigma_B|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)\right)$$

*独立性を仮定したベイズ決定則を単純ベイズ（ナイーブベイズ）と呼ぶ

# 正規分布の仮定でのベイズ決定則①

- p(**x**|8)p(8) > p(**x**|B)p(B) ⟶「8」と認識

$n_8$:「8」のパターン数　$n_B$:「B」のパターン数

$$\underbrace{\frac{1}{(2\pi)^{d/2}|\Sigma_8|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)\right)}_{p(\mathbf{x}|8)}\times\underbrace{\frac{n_8}{n_8+n_B}}_{p(8)} > \underbrace{\frac{1}{(2\pi)^{d/2}|\Sigma_B|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)\right)}_{p(\mathbf{x}|B)}\times\underbrace{\frac{n_B}{n_8+n_B}}_{p(B)}$$

両辺，対数をとると

$$-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)-\frac{d}{2}\log 2\pi-\frac{1}{2}\log|\Sigma_8|+\log(\frac{n_8}{n_8+n_B}) > -\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)-\frac{d}{2}\log 2\pi-\frac{1}{2}\log|\Sigma_B|+\log(\frac{n_B}{n_8+n_B})$$

$$\Downarrow$$

$$\boxed{(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)+\log|\Sigma_8|-2\log(\frac{n_8}{n_8+n_B}) < (\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)+\log|\Sigma_B|-2\log(\frac{n_B}{n_8+n_B})}$$

# 正規分布の仮定でのベイズ決定則②

- p(**x**|8)p(8) < p(**x**|B)p(B) → 「B」と認識

$$\frac{1}{(2\pi)^{d/2}|\Sigma_8|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)\right)\times\frac{n_8}{n_8+n_B}<\frac{1}{(2\pi)^{d/2}|\Sigma_B|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)\right)\times\frac{n_B}{n_8+n_B}$$

p(**x**|8)　p(8)　p(**x**|B)　p(B)

両辺，対数をとると

$$-\frac{1}{2}(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)-\frac{d}{2}\log 2\pi-\frac{1}{2}\log|\Sigma_8|+\log(\frac{n_8}{n_8+n_B})<-\frac{1}{2}(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)-\frac{d}{2}\log 2\pi-\frac{1}{2}\log|\Sigma_B|+\log(\frac{n_B}{n_8+n_B})$$

$$\Downarrow$$

$$\boxed{(\mathbf{x}-\mathbf{m}_8)^t\Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)+\log|\Sigma_8|-2\log(\frac{n_8}{n_8+n_B})>(\mathbf{x}-\mathbf{m}_B)^t\Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)+\log|\Sigma_B|-2\log(\frac{n_B}{n_8+n_B})}$$

# 単純ベイズ決定則

- c個のクラス$\omega_i$（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）
  - 各クラスごとの条件付き確率 $p(\mathbf{x}|\omega_i)$
  - 特徴ベクトル **x** はどのクラスに属するか

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

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

$$= \max_{i=1,2,\cdots,c}\left\{p(x_1,x_2,\cdots,x_d|\omega_i)p(\omega_i)\right\} = \max_{i=1,2,\cdots,c}\left\{\prod_{j=1}^{d}p(x_j|\omega_i)p(\omega_i)\right\}$$

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

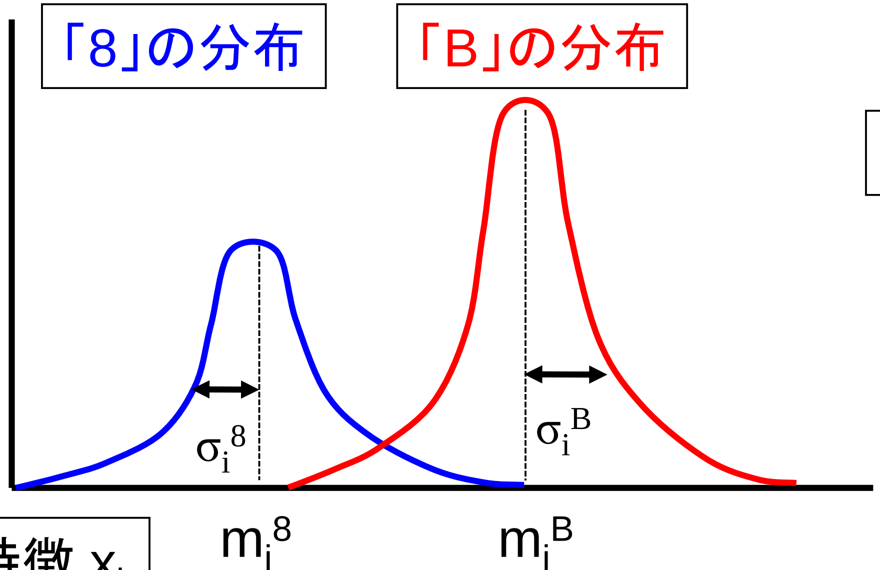
独立性を仮定

# ベイズ決定則のまとめ①

条件付き確率 $p(x_i|8)$ $p(x_i|B)$

「8」の分布

「B」の分布

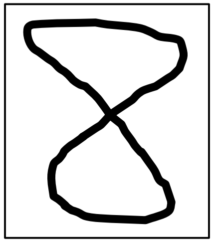



特徴 $x_i$

正規分布を仮定
平均ベクトル（$\mathbf{m}_8$, $\mathbf{m}_B$）
分散共分散行列（$\Sigma_8$, $\Sigma_B$）を推定

事前確率 p(8) p(B) は既知

未知文字を認識したい場合



特徴ベクトル $\mathbf{x}$

事後確率を計算
$p(8|\mathbf{x})$ $p(B|\mathbf{x})$

# ベイズ決定則のまとめ②

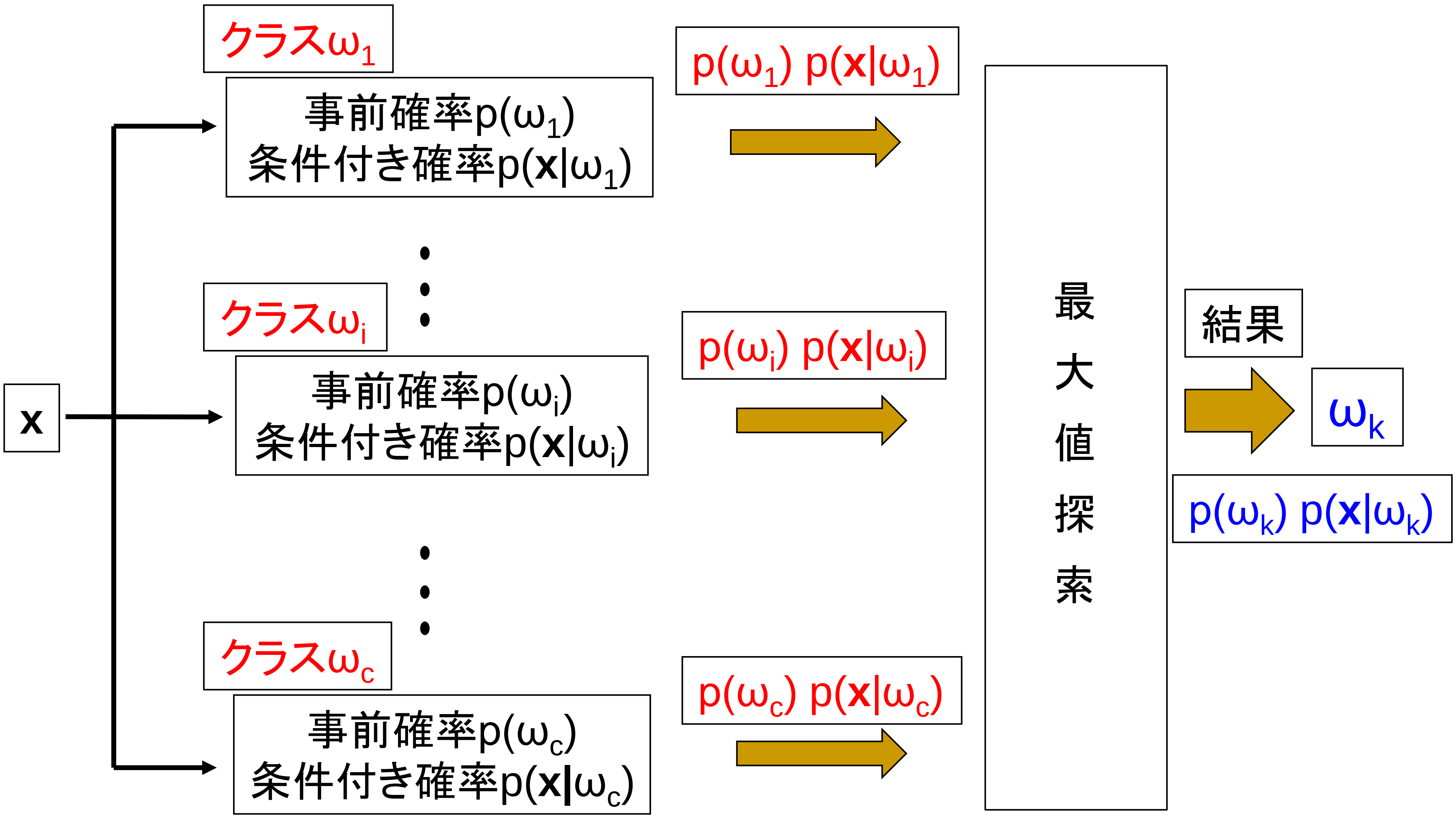
- p(**x**|8)p(8) > p(**x**|B)p(B) → 「8」と認識

$$\boxed{(\mathbf{x}-\mathbf{m}_8)^t \Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)+\log|\Sigma_8|-2\log(\frac{n_8}{n_8+n_B}) < (\mathbf{x}-\mathbf{m}_B)^t \Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)+\log|\Sigma_B|-2\log(\frac{n_B}{n_8+n_B})}$$

- p(**x**|8)p(8) < p(**x**|B)p(B) → 「B」と認識

$$\boxed{(\mathbf{x}-\mathbf{m}_8)^t \Sigma_8^{-1}(\mathbf{x}-\mathbf{m}_8)+\log|\Sigma_8|-2\log(\frac{n_8}{n_8+n_B}) > (\mathbf{x}-\mathbf{m}_B)^t \Sigma_B^{-1}(\mathbf{x}-\mathbf{m}_B)+\log|\Sigma_B|-2\log(\frac{n_B}{n_8+n_B})}$$

# ベイズ決定則のまとめ③（多クラスの場合）




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

# 特徴の出現確率を用いた認識とは？①

- ベイズ決定則

$$p(8\,|\,\mathbf{x}) > p(B\,|\,\mathbf{x})$$
$$p(\mathbf{x}\,|\,8)\,p(8) > p(\mathbf{x}\,|\,B)\,p(B)$$

⇒ 「8」と認識

$$p(8\,|\,\mathbf{x}) > p(B\,|\,\mathbf{x})$$
$$p(\mathbf{x}\,|\,8)\,p(8) < p(\mathbf{x}\,|\,B)\,p(B)$$

⇒ 「B」と認識

$p(8) = p(B)$ の場合

$$p(\mathbf{x}\,|\,8) > p(\mathbf{x}\,|\,B)$$

⇒ 「8」と認識

$$p(\mathbf{x}\,|\,8) < p(\mathbf{x}\,|\,B)$$

⇒ 「B」と認識

# 特徴の出現確率を用いた認識とは？②

- ベイズ決定則において，事前確率 p(8)，p(B)（$n_8$と$n_B$）が同じ場合，特徴の出現確率のみを用いて認識が可能

- ただし，
  - 「p(**x**|8)+p(**x**|B)=1」は成立せず，「確率」の大小によって認識しているとは言い難い
  - この場合，p(**x**|8)，p(**x**|B) を尤度と呼び，この手法を最尤法*と呼ぶ

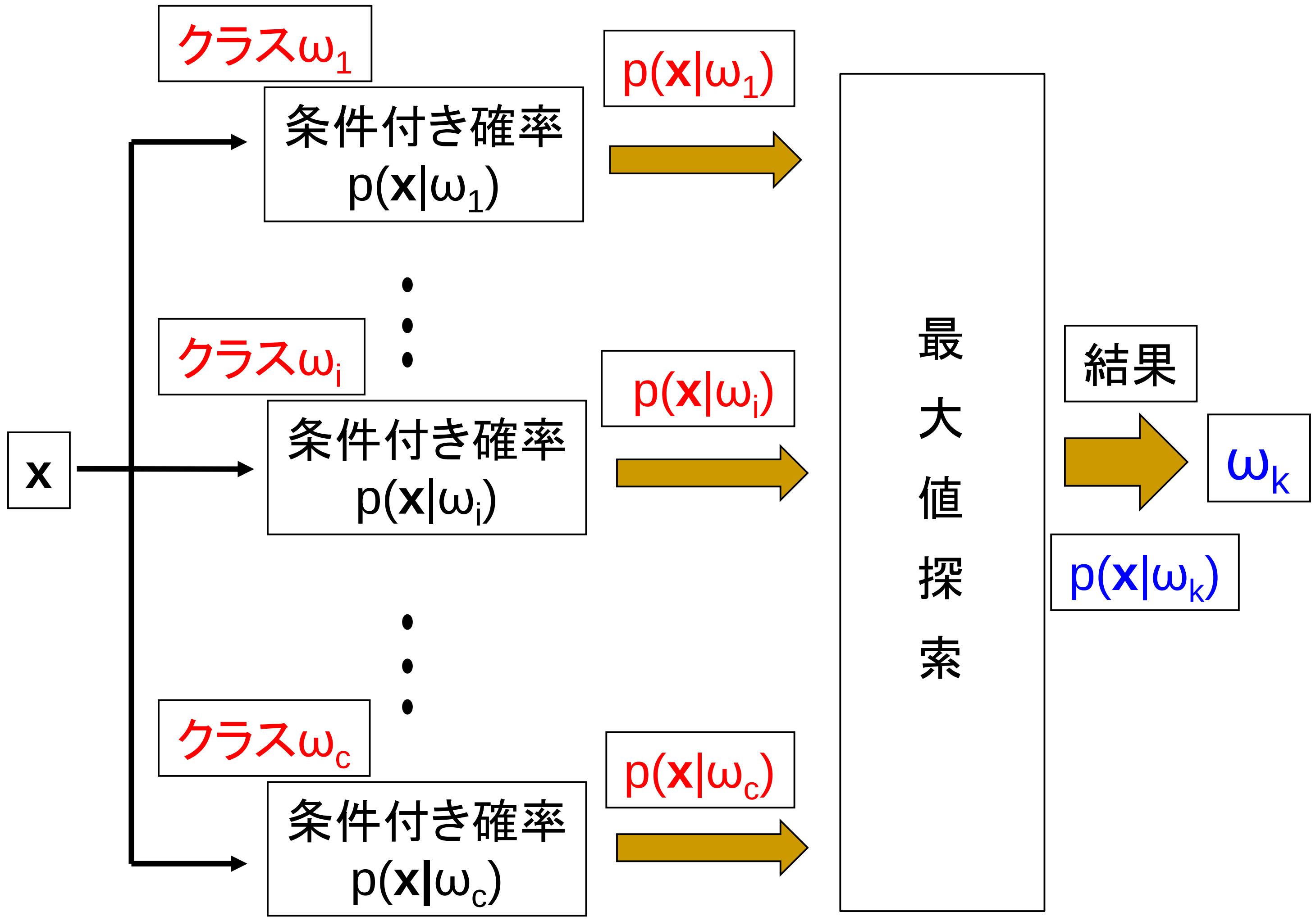
*後の説明で出てくる尤度関数，最尤推定とは違うことに注意して下さい

# 最尤法による認識①

- c個のクラス$\omega_i$（i=1,2,・・・,c)
  - 事前確率 $p(\omega_i)$ は全て同じ( $p(\omega_i) = 1/c$）
- 未知のパターンの特徴ベクトル **x**（d次元）
  - 各クラスごとの条件付き確率（尤度） $p(\mathbf{x}|\omega_i)$
  - 特徴ベクトル **x** はどのクラスに属するか

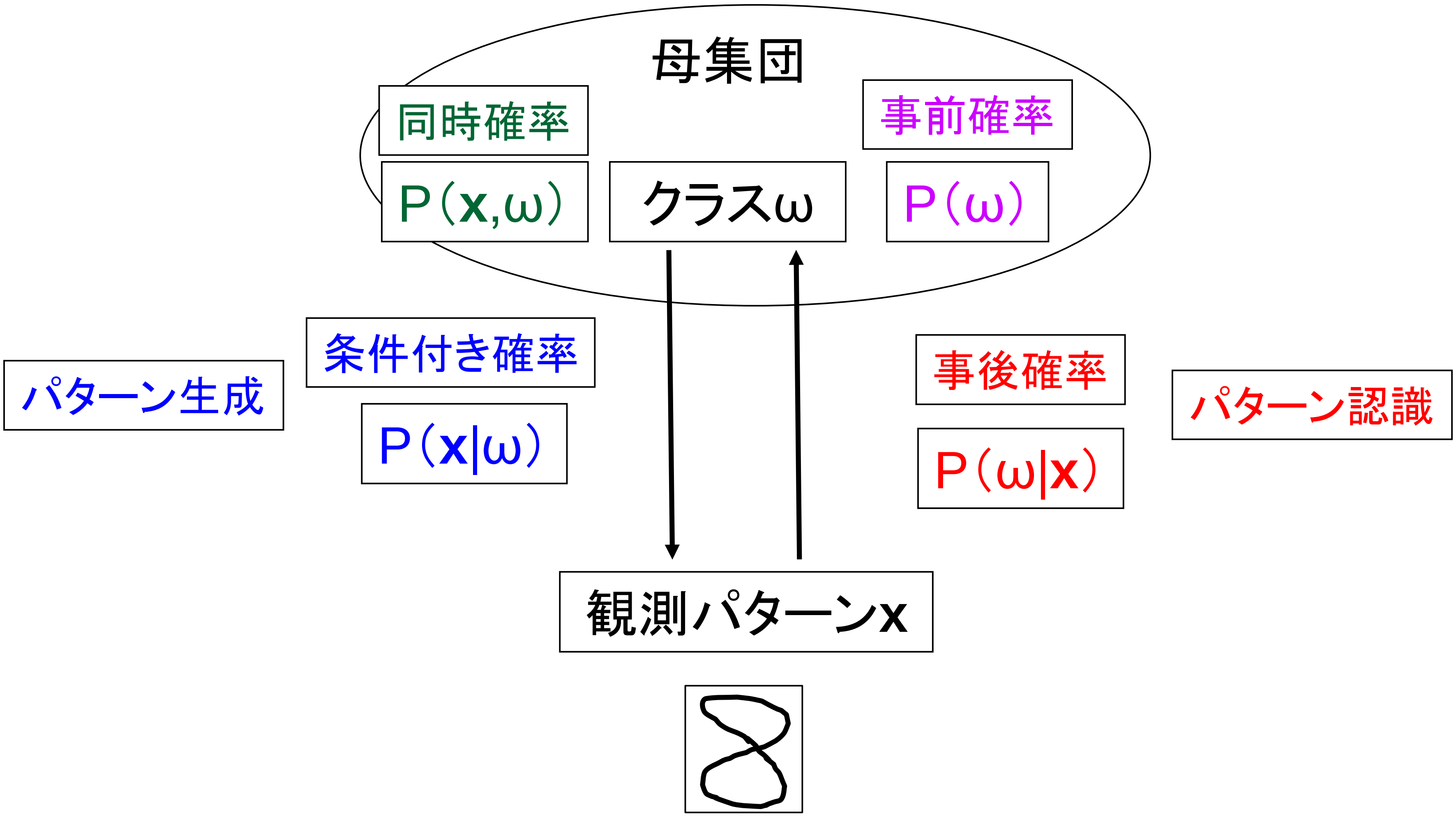
$$
\boxed{\begin{aligned}
&\max_{i=1,2,\cdots,c}\left\{p(\mathbf{x}\,|\,\omega_i)\right\}\\
&= p(\omega_k\,|\,\mathbf{x}) \Rightarrow \mathbf{x} \in \omega_k
\end{aligned}}
$$

# 最尤法による認識②



クラス$\omega_1$
条件付き確率
$p(\mathbf{x}|\omega_1)$
$p(\mathbf{x}|\omega_1)$
クラス$\omega_i$
条件付き確率
$p(\mathbf{x}|\omega_i)$
$p(\mathbf{x}|\omega_i)$
クラス$\omega_c$
条件付き確率
$p(\mathbf{x}|\omega_c)$
$p(\mathbf{x}|\omega_c)$
$\mathbf{x}$
最大値探索
結果
$\omega_k$
$p(\mathbf{x}|\omega_k)$

# パターン生成とパターン認識



# 生成モデルと識別モデル①

- パターン認識
  - 事後確率 $P(\omega|\mathbf{x})$を用いて認識

- ベイズ決定則
  - 事後確率$P(\omega|\mathbf{x})$を直接用いない
  - 条件付き確率 $P(\mathbf{x}|\omega)$を求め，事後確率を推定し，認識
  - 同時確率$P(\mathbf{x},\omega)=P(\mathbf{x}|\omega)P(\omega)$を求めることと等価
  - 生成モデルと呼ぶ

- 識別モデル
  - 事後確率$P(\omega|\mathbf{x})$を直接推定し，認識

# 生成モデルと識別モデル②

生成モデル

境界

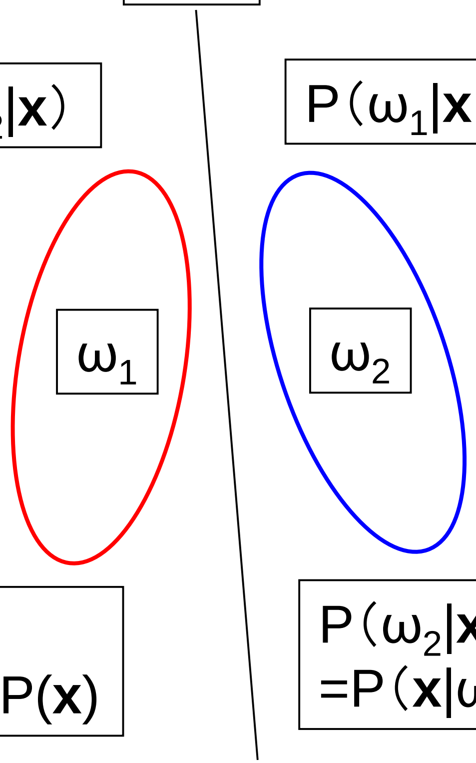
$P(\omega_1|\mathbf{x}) > P(\omega_2|\mathbf{x})$

$P(\omega_1|\mathbf{x}) < P(\omega_2|\mathbf{x})$

母集団

クラス$\omega_1$　$P(\omega_1)$

観測データ$\mathbf{x}$




同時確率

$P(\mathbf{x},\omega_1)$
$=P(\mathbf{x}|\omega_1)P(\omega_1)$

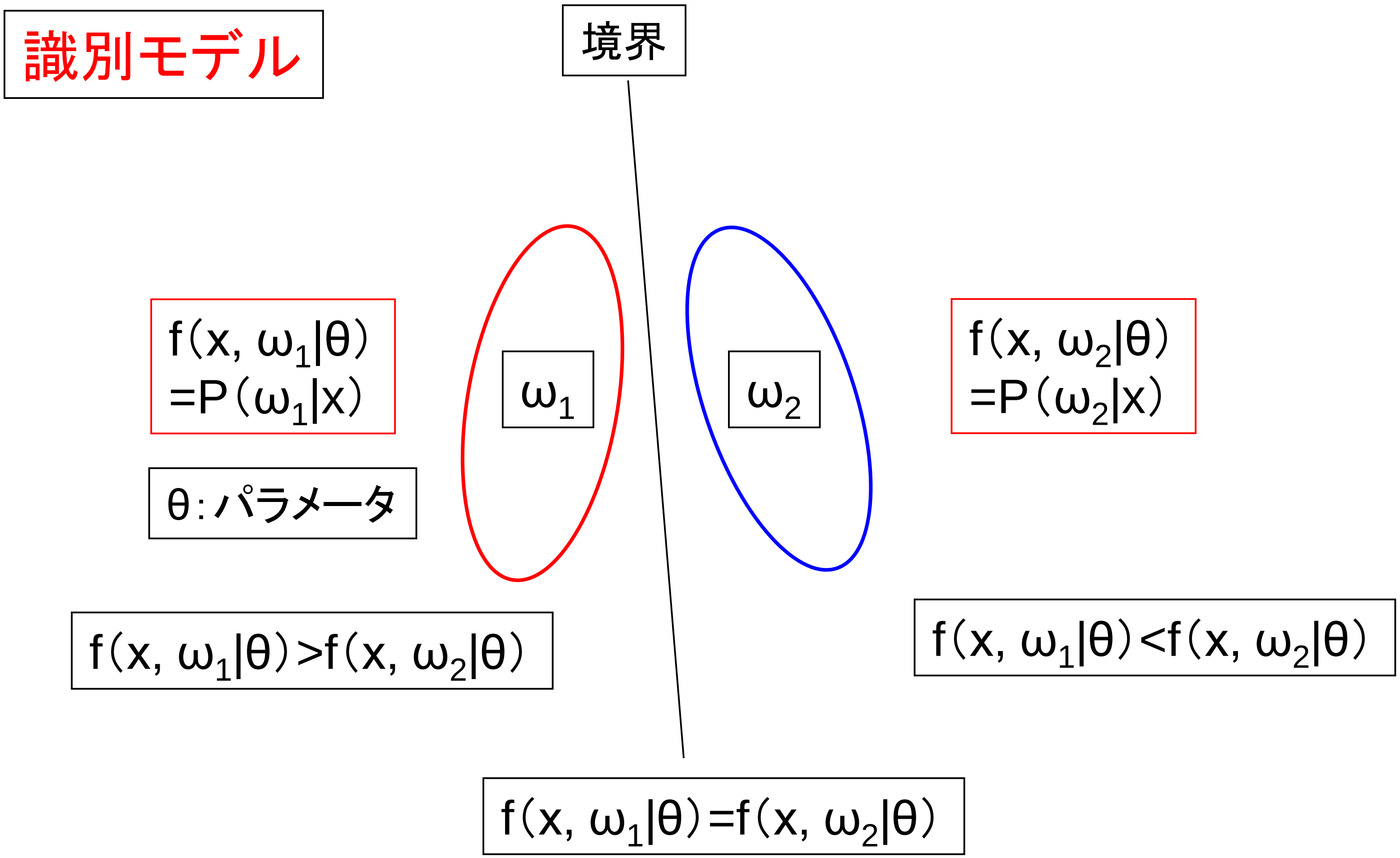
事後確率

$P(\omega_1|\mathbf{x})$
$=P(\mathbf{x}|\omega_1)P(\omega_1)/P(\mathbf{x})$

$P(\omega_2|\mathbf{x})$
$=P(\mathbf{x}|\omega_2)P(\omega_2)/P(\mathbf{x})$

$P(\omega_1|\mathbf{x}) = P(\omega_2|\mathbf{x})$

# 生成モデルと識別モデル②



# ベイズ決定則の他の解釈

- 損失 $l(\omega_j \mid \omega_i)$
  - クラス数はc個
  - クラス$\omega_i$ を $\omega_j$ と認識した場合の損失

  - 特徴ベクトル **x** が与えられた場合，**x** を $\omega_j$ と認識した場合の損失（期待損失）

$$L(\omega_j \mid \mathbf{x}) = \sum_{i=1}^{c} l(\omega_j \mid \omega_i) p(\omega_i \mid \mathbf{x})$$

  - 期待損失が最小になるように認識する

# 期待損失最小化

- 0-1損失基準

$$l(\omega_j \mid \omega_i) = \begin{cases} 0 & \text{if } j = i \\ 1 & \text{otherwise} \end{cases}$$

間違えなかった場合は0

間違えた場合は1

- 期待損失

$$L(\omega_j \mid \mathbf{x}) = \sum_{i=1}^{c} l(\omega_j \mid \omega_i) p(\omega_i \mid \mathbf{x})$$

$$= p(\omega_1 \mid \mathbf{x}) + p(\omega_2 \mid \mathbf{x}) + \cdots + p(\omega_{j-1} \mid \mathbf{x}) + p(\omega_{j+1} \mid \mathbf{x}) + \cdots + p(\omega_c \mid \mathbf{x})$$

$$= \sum_{i \neq j} p(\omega_i \mid \mathbf{x}) = 1 - p(\omega_j \mid \mathbf{x}) \longrightarrow \text{最小}$$

$$\Downarrow$$

$$p(\omega_j \mid \mathbf{x}) \longrightarrow \text{最大}$$

事後確率の最大化と同等

# 統計的パターン認識のまとめ

- 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 特徴の出現確率（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の例題

マハラノビス距離による認識

# 統計的パターン認識の例題

- マハラノビス距離による認識

- 実習①
  - 人工データ(正規分布によるデータ)の分類
  - Mahalanobis-1.py

- 実習②
  - MNISTの数字画像認識
  - Mahalanobis-2.py

# 実行方法

- Mahalanobis-1.py
  - > python Mahalanobis-1.py

- Mahalanobis-2.py
  - > python Mahalanobis-2.py
  - matplotlibが必要です（後述）
  - mnistのデーターがあるフォルダーにプログラムを置いて下さい

# Mahalanobis-1.py（変数の定義）

```
# クラス数，学習データ数，テストデータ数，特徴数
class_num = 4
train_num = 100
test_num = 100
size = 5

# 学習データ，テストデータ，平均ベクトル，分散共分散行列，行列式
train_vec = np.zeros((class_num,train_num,size), dtype=np.float64)
test_vec = np.zeros((class_num,test_num,size), dtype=np.float64)
ave_vec = np.zeros((class_num,size), dtype=np.float64)
var = np.zeros((class_num,size,size), dtype=np.float64)
det = np.zeros(class_num, dtype=np.float64)
```

倍精度

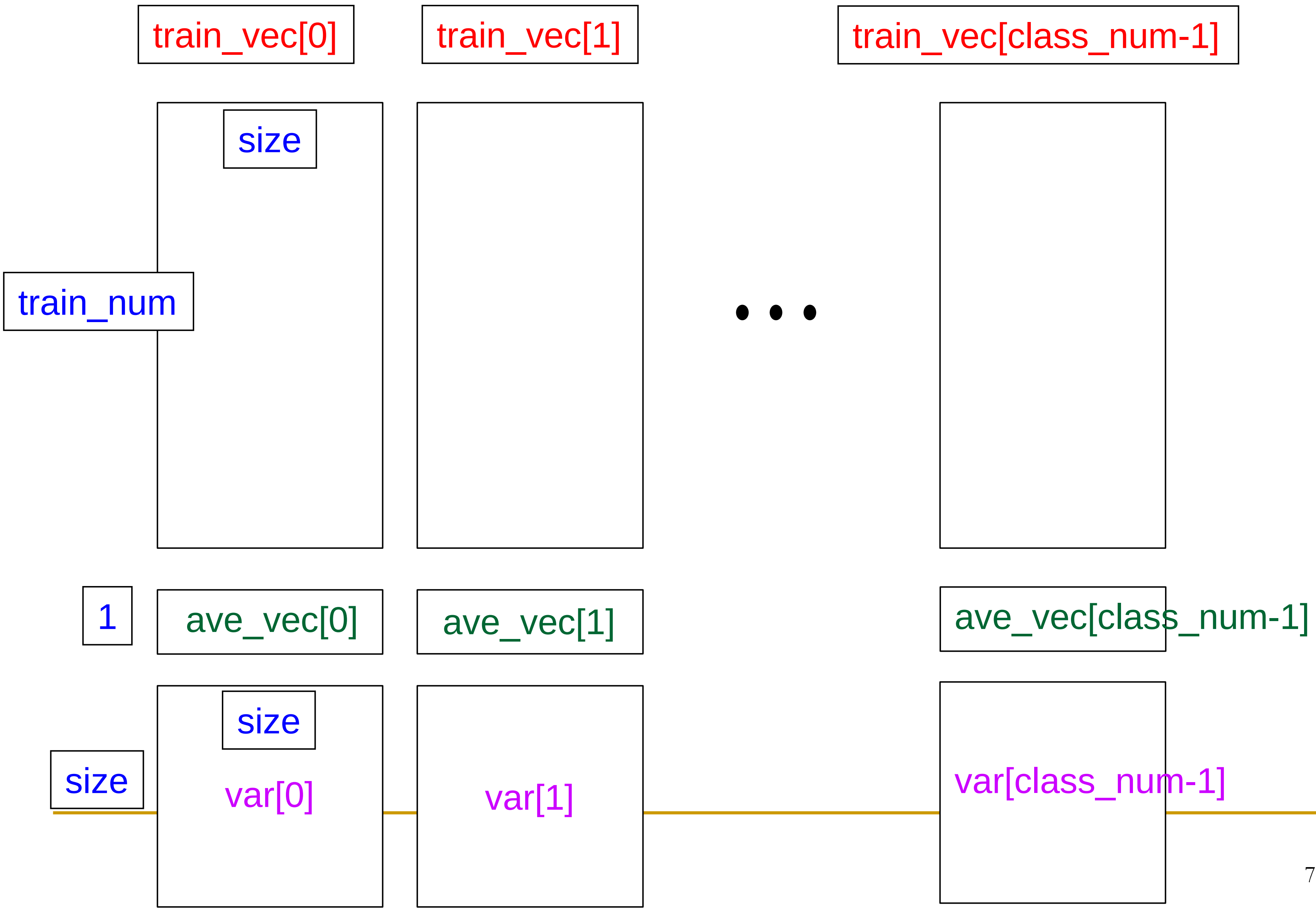
# 変数名

- クラス数 class_num
- 特徴数　size
- 学習データ数 train_num
- テストデータ数 test_num

クラス数 class_num個，特徴数 size個の正規分布のデータを生成

- 学習データ train_vec（class_num×train_num×size）
- テストデータ test_vec（class_num×train_num×size）
- 平均ベクトル ave_vec（class_num×size）
- 分散共分散行列の逆行列 var（class_num×size×size）
- 行列式 det（class_num）

# 配列の要素


train_vec[0]
train_vec[1]
train_vec[class_num-1]
size
train_num
・・・
1
ave_vec[0]
ave_vec[1]
ave_vec[class_num-1]
size
size
var[0]
var[1]
var[class_num-1]

# データの生成①

```
# 平均値の設定
d = np.zeros((class_num,size), dtype=np.int32)
s = np.zeros((class_num,size), dtype=np.float64)
for i in range(class_num):
    for j in range(size):
        d[i][j] = random.randint(1,5)
        s[i][j] = random.uniform(0,2)
```

各クラスの各特徴の平均値を1から5の乱数（整数）で設定

各クラスの各特徴の標準偏差を0から2の乱数（小数）で設定

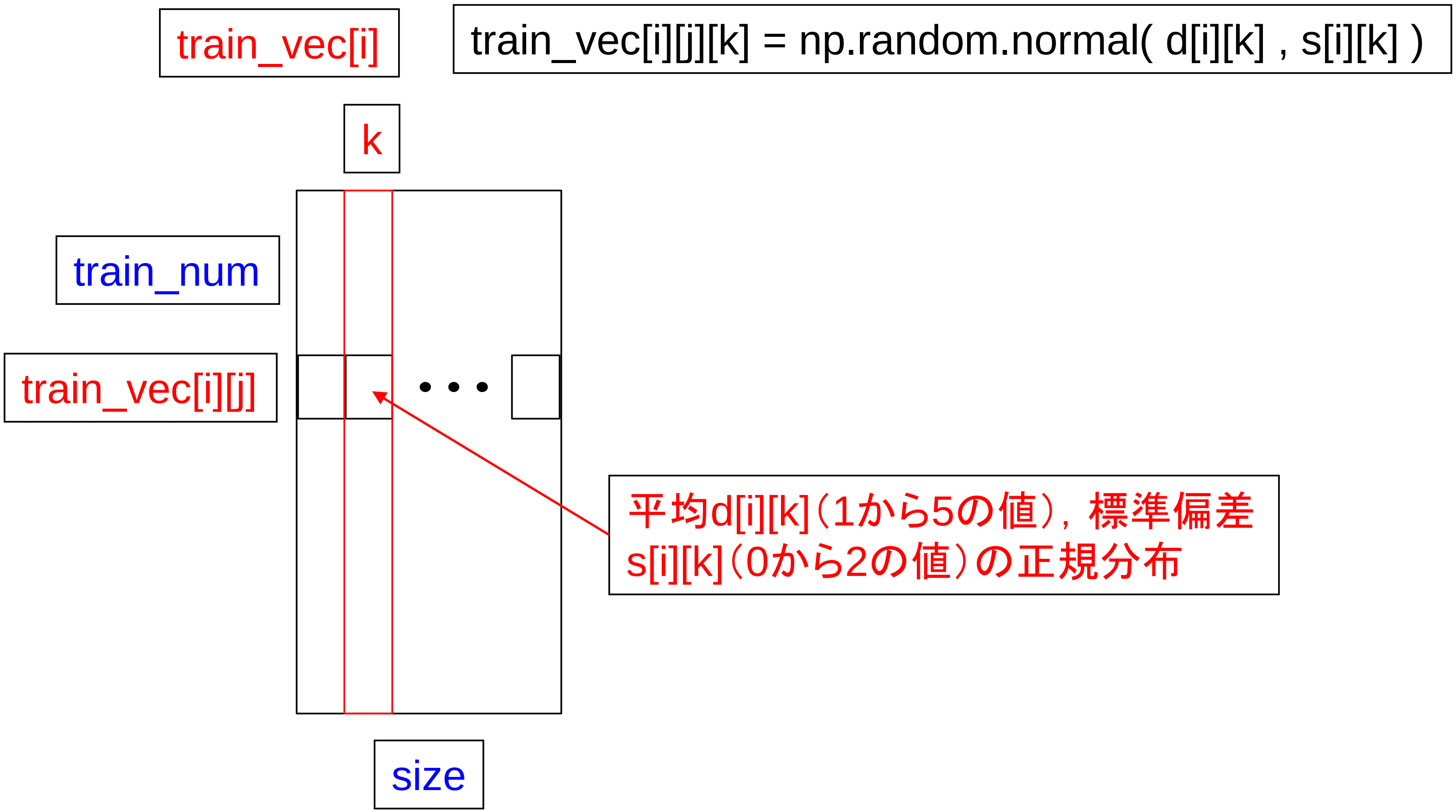
```
# 学習データの生成
for i in range(class_num):
    for j in range( train_num ):
        for k in range( size ):
            train_vec[i][j][k] = np.random.normal( d[i][k] , s[i][k] )
```

np.random.normal(ave,sd)
平均ave，標準偏差sdの正規分布に従う乱数を生成

d[i][k]：1から5の値

s[i][k]：0から2の値

# データの生成②


train_vec[i]
train_vec[i][j][k] = np.random.normal( d[i][k] , s[i][k] )
k
train_num
train_vec[i][j]
・・・
平均d[i][k]（1から5の値），標準偏差
s[i][k]（0から2の値）の正規分布
size

# データの生成③

```
# テストデータの生成
for i in range(class_num):
    for j in range( test_num ):
        for k in range( size ):
            test_vec[i][j][k] = np.random.normal( d[i][k] , s[i][k] )
```

d[i][k]：1から5の値

s[i][k]：0から2の値

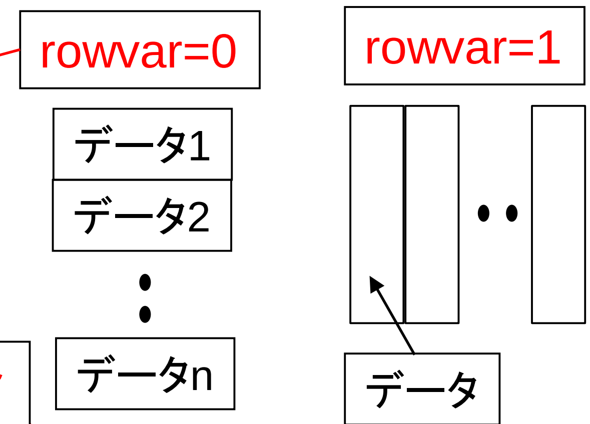
# 平均ベクトル，分散共分散行列，逆行列，行列式

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

for i in range(class_num):
    # 分散共分散行列
    v = np.cov( train_vec[i] , rowvar=0 , bias=1 )

    det[i] = np.linalg.det(v)
    if det[i] != 0.0:
        var[i] = np.linalg.inv(v)
        print( "¥n [ 検算 ]" )
        print( np.dot( var[i] , v ) )
    else:
        I = np.identity(size)
        var[i]=I
```

列方向に平均を求める（axis=0）




bias=0：不偏分散
bias=1：標本分散

行列式

行列式が0でない場合
→逆行列を求める

単位行列になるか検算

行列式が0の場合
→単位行列

# ユークリッド距離による認識①

ユークリッド距離の結果を格納する配列

マハラノビス距離の結果を格納する配列

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

for i in range(class_num):
    for j in range(0,test_num):
        # ユークリッド距離（SSD）
        min_val = float('inf')
        ans = 0
        for k in range(class_num):
            aT = np.reshape( (test_vec[i][j]-ave_vec[k]) , (1,size) )
            a = np.reshape( (test_vec[i][j]-ave_vec[k]) , (size,1) )
            dist = np.dot( aT , a )
```

（1×size）のベクトルに変形

（size×1）のベクトルに変形

SSDの計算

# ユークリッド距離による認識②

```
            if dist < min_val:
                min_val = dist
                ans = k
        result_e[i][ans] +=1

print( "¥n [混合行列（ユークリッド距離）]" )
print( result_e )
print( "¥n 正解数 ->" ,  np.trace(result_e) )
```

最小値の探索

混合行列に予測値を代入

混合行列の出力

正解数の出力

# マハラノビス距離による認識

```
for i in range(class_num):
    for j in range(0,test_num):
        # マハラノビス距離
        min_val = float('inf')
        ans = 0
        for k in range(class_num):
            aT = np.reshape( (test_vec[i][j]-ave_vec[k]) , (1,size) )
            a = np.reshape( (test_vec[i][j]-ave_vec[k]) , (size,1) )
            dist = np.dot( np.dot( aT , var[k] ) , a )

            if dist < min_val:
                min_val = dist
                ans = k
        result_m[i][ans] +=1
```

（1×size）のベクトルに変形

（size×1）のベクトルに変形

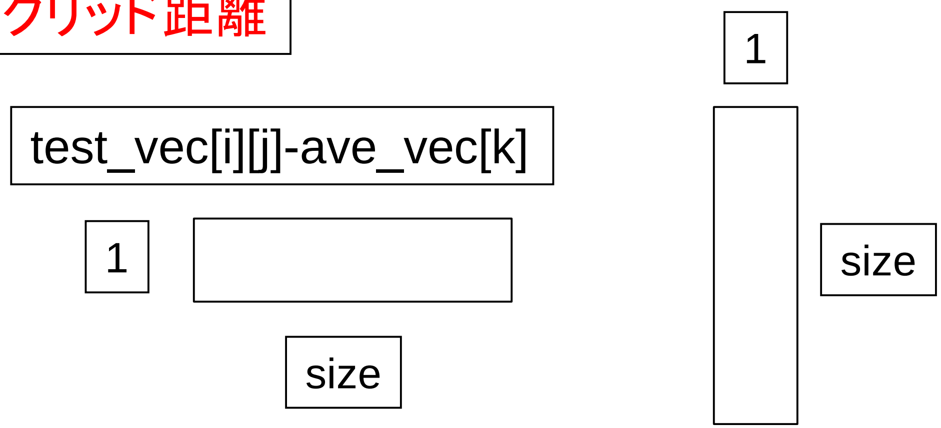
マハラノビス距離の計算

最小値の探索

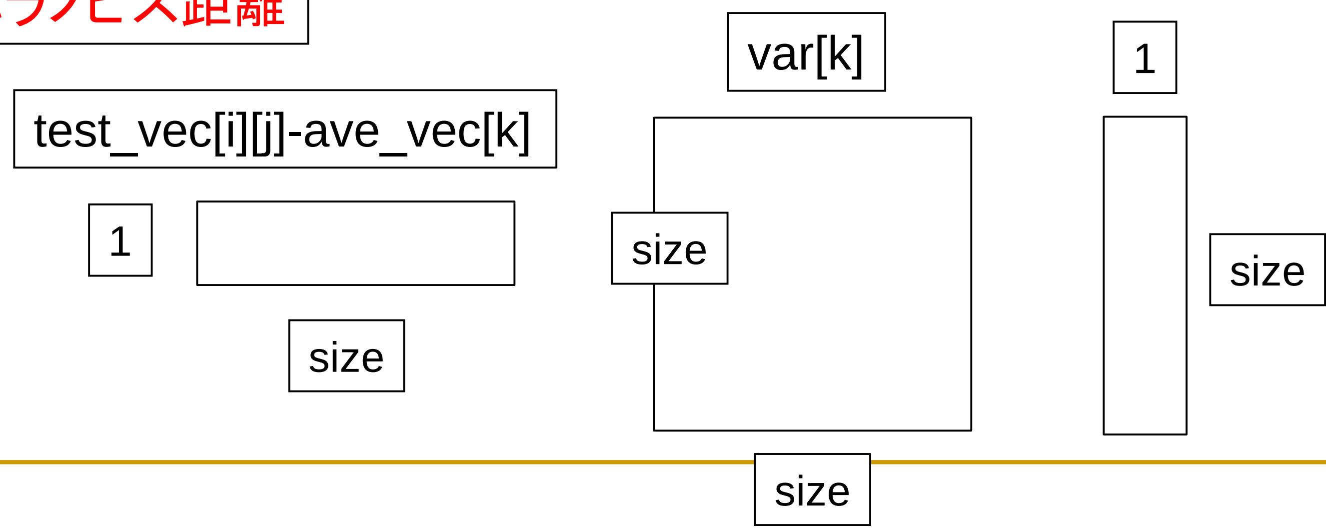
混合行列に予測値を代入

# 距離の計算

## ユークリッド距離




## マハラノビス距離



# 実行結果①

class_num=3, size=4の実行結果

```
[ 平均ベクトル ]
[[5.05795279 4.03443304 3.18264222 3.01876352]
 [4.782653   5.22828359 3.90579653 5.23915413]
 [1.00375903 1.02518734 4.99347005 2.94825368]]

 [ 検算 ]
[[ 1.00000000e+00 -1.73472348e-18 -4.07660017e-17  6.93889390e-18]
 [ 5.20417043e-18  1.00000000e+00  7.80625564e-18  1.38777878e-17]
 [ 4.94396191e-17  8.67361738e-18  1.00000000e+00  6.93889390e-18]
 [ 2.77555756e-17  2.77555756e-17 -1.38777878e-17  1.00000000e+00]]
```

クラス1の平均は(5,4,3,3)

クラス2の平均は(4,5,4,5)

クラス3の平均は(1,1,5,3)

逆行列と元の行列の積は(ほぼ)単位行列

# 実行結果②

```
[混合行列(ユークリッド距離)]
[[ 67  25   8]
 [ 27  64   9]
 [  0   0 100]]
 正解数 -> 231

 [混合行列(マハラノビス距離)]
[[60 40  0]
 [18 82  0]
 [ 4  4 92]]
 正解数 -> 234
```

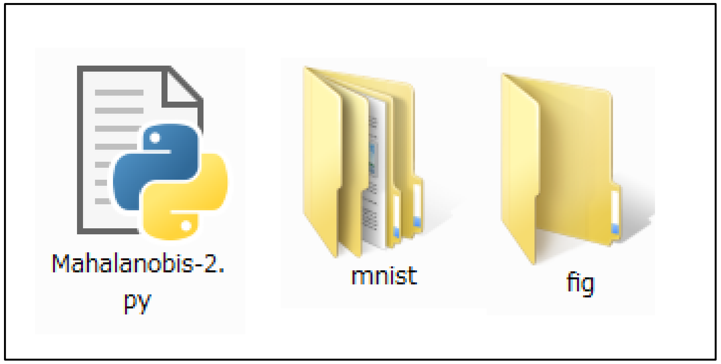
ユークリッド距離による結果

マハラノビス距離による結果

# Maharanobis-2.py

- MNISTの数字画像認識

- MNISTのデータがあるフォルダーにプログラムは置いて下さい
- 「fig」という名前のフォルダーを作成して下さい




- 必要なパッケージ
  - matplotlib
  - > pip install matplotlib
  - > pip install matplotlib==2.2.3

理工学ITCでは最新版がインストールできませんでした

# Maharanobis-2.py（変数の定義）

```
import sys
import os
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

ライブラリイのインポート

```
size = 14
```

画像の大きさ

```
train_num = 100
```

データ数

学習データ

```
train_vec = np.zeros((10,train_num,size*size), dtype=np.float64)
ave_vec = np.zeros((10,size*size), dtype=np.float64)
```

平均ベクトル

```
lamda = np.zeros((10,size*size), dtype=np.float64)
eig_vec = np.zeros((10,size*size,size*size), dtype=np.float64)
```

固有値ベクトル，固有行列

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

MS-Windowsのみ（UNIX，Macは注意）
→ "rm fig/*"

# 変数名

- 画像の縮小後の縦，横の大きさ　size
- 画像の画素数　size×size
- 学習データ数 train_num

- 学習データ train_vec（10×train_num×（size×size））
- 平均ベクトル ave_vec（10×（size×size））
- 固有値（固有値ベクトル） lamda（10×（size×size））
- 固有ベクトル（固有行列）
  eig_vec（10×（size×size）×（size×size））

# MNISTデータの読み込み①

```
# 学習データの読み込み
for i in range(10):
    for j in range(1,train_num+1):
        train_file = "mnist/train/" + str(i) + "/" + str(i) + "_" + str(j) + ".jpg"
        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

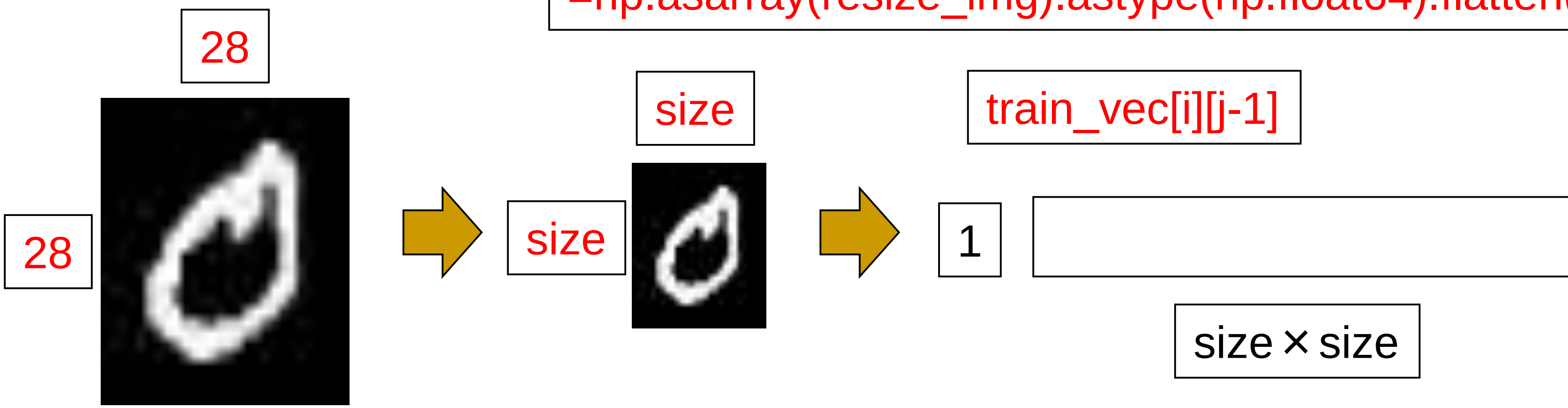
グレースケール画像として読み込み

（size×size）の画像に大きさを変更

numpyに変換→ベクトルに変形

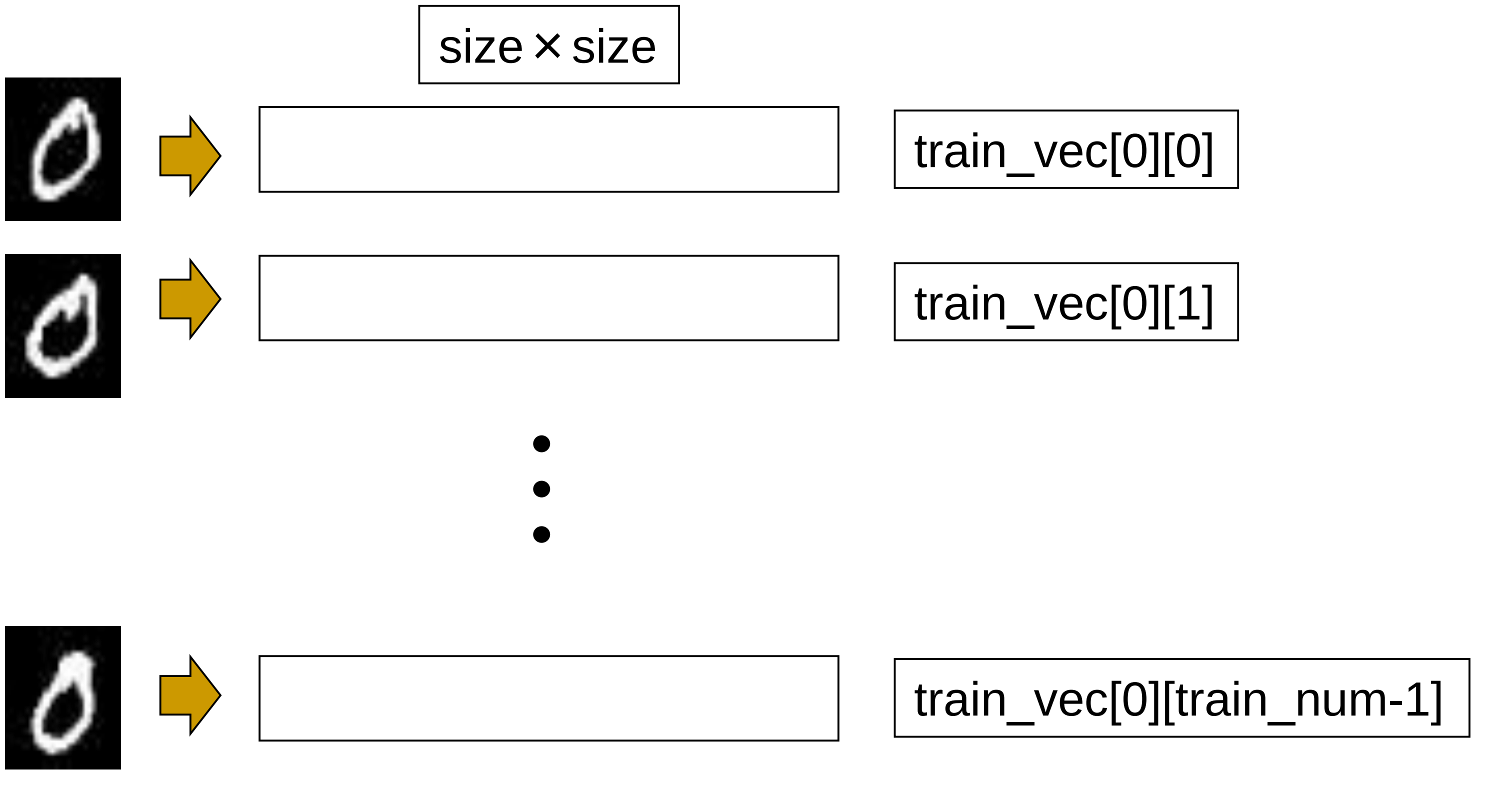
# MNISTデータの読み込み②

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

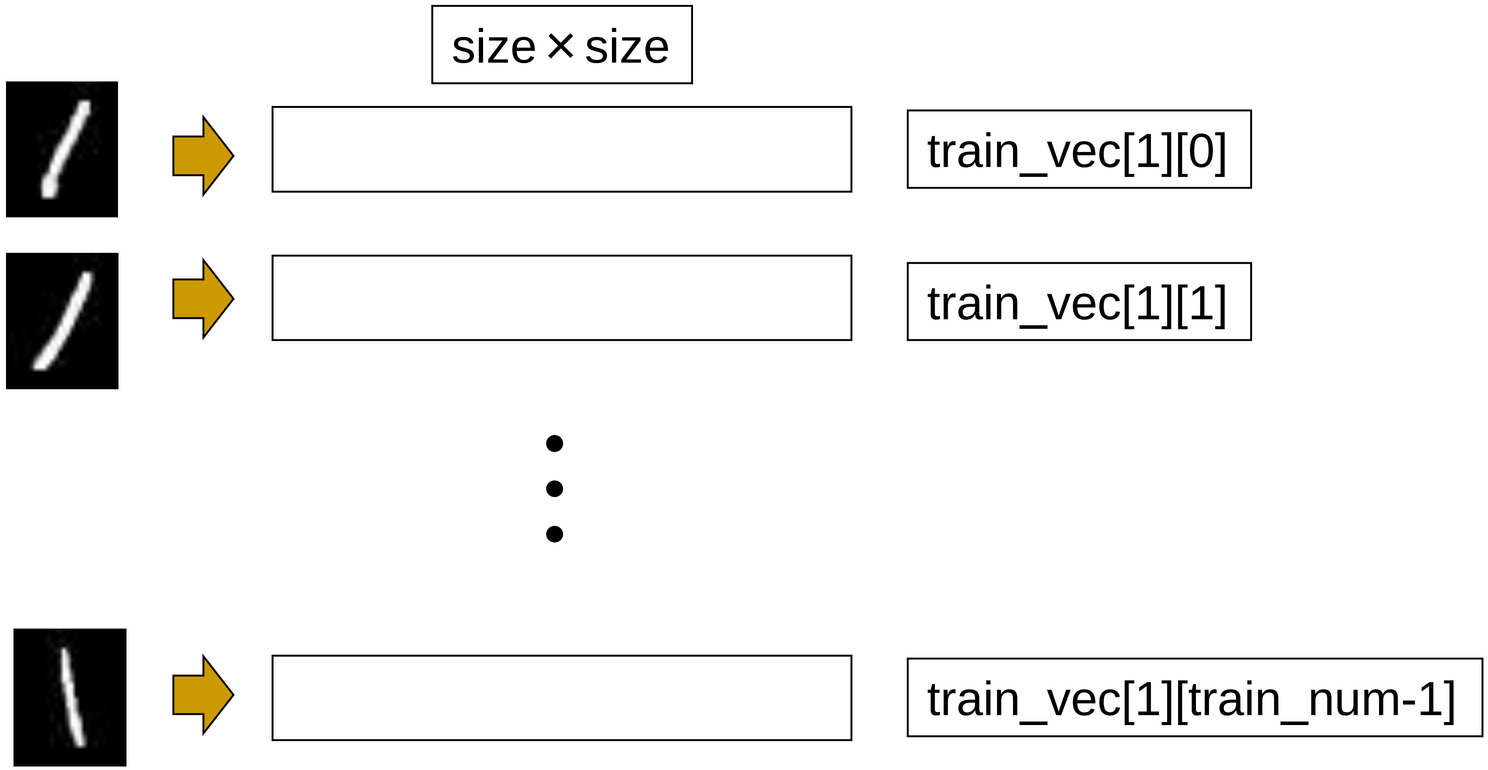



```
resize_img = work_img.resize((size, size))
```

# MNISTデータの読み込み③



# MNISTデータの読み込み④



# 平均ベクトル(平均画像)

```
# 平均ベクトル
for i in range(10):
    ave_vec[i] = np.mean( train_vec[i] , axis=0 )

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

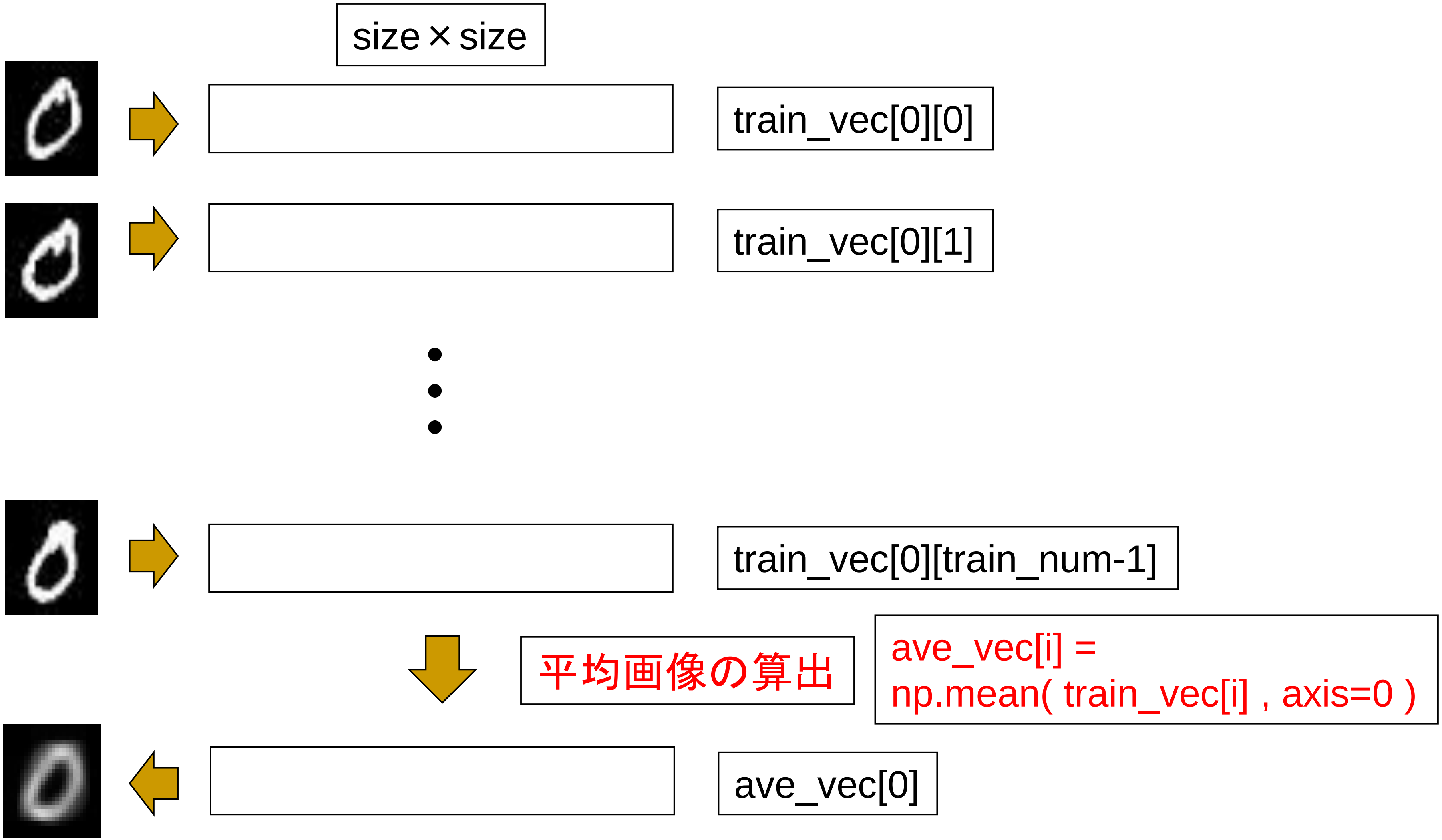
列方向に平均を求める(axis=0)

(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

# 平均ベクトル(平均画像)の算出




ave_img =Image.fromarray(np.uint8(np.reshape(ave_vec[i],(size,size))))

# 近似式によるマハラノビス距離の求め方

- 分散共分散行列Σ

$$\Sigma\Phi = \lambda\Phi$$

固有値分解

$$\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_d)$$

固有値

$$\Phi = (\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_d)$$

固有ベクトル

$$\Sigma = \sum_{i=1}^{d} \lambda_i \mathbf{u}_i \mathbf{u}_i^t$$

$$\Sigma^{-1} = \sum_{i=1}^{d} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^t$$

$$(\mathbf{x}-\mathbf{m})^t \Sigma^{-1} (\mathbf{x}-\mathbf{m}) = \sum_{i=1}^{d} \frac{(\mathbf{u}_i^t(\mathbf{x}-\mathbf{m}))^2}{\lambda_i} = \sum_{i=1}^{d'} \frac{(\mathbf{u}_i^t(\mathbf{x}-\mathbf{m}))^2}{\lambda_i}$$

$$(d > d')$$

# 分散共分散行列，固有値分解

```
for i in range(10):
    # 分散共分散
    cov = np.cov( train_vec[i] , rowvar=0 , bias=1 )

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

    # 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')
        file = "fig/eigen-" + str(i) + " " + str(j) + ".png"
        plt.savefig(file)

    # 検算
    print( np.dot( eig_vec[i][:,0] , eig_vec[i][:,1]) )
    print( np.dot( eig_vec[i][:,1] , eig_vec[i][:,1] ) )
```

rowvar=0

データ1
データ2
:
データn

bias=1
標本分散

# 分散共分散行列，固有値分解

```
for i in range(10):
    # 分散共分散
    cov = np.cov( train_vec[i] , rowvar=0 , bias=1 )

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

rowvar=0

データ1
データ2
:
データn

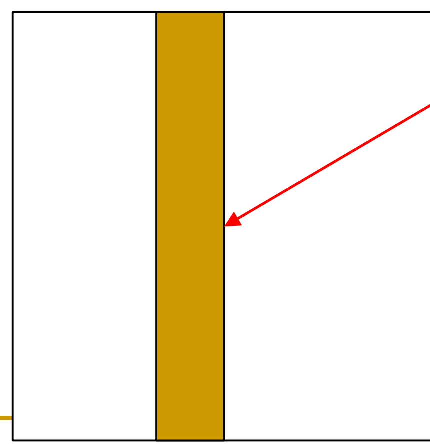
bias=1
標本分散

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[i]

size*size



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[i][:,j]

# （参考）固有ベクトルの表示*

```
# 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()
```

実数部

（size×size）に変換

二次元マップで表示（matplotlibの機能）→保存

```
# 検算
print( np.dot( eig_vec[i][:,0] , eig_vec[i][:,1]) )
print( np.dot( eig_vec[i][:,1] , eig_vec[i][:,1] ) )
```

$\boxed{\mathbf{u}_i^t \mathbf{u}_j = 0}$

$\boxed{\mathbf{u}_i^t \mathbf{u}_i = 1}$

*詳細は線形識別関数（特徴空間の変換）にて説明します

# 固有ベクトルの表示

内積は1
内積は0
eigen-0-0.png
eigen-0-1.png
eigen-0-2.png
eigen-0-3.png
eigen-0-4.png
eigen-1-0.png
eigen-1-1.png
eigen-1-2.png
eigen-1-3.png
eigen-1-4.png
eigen-2-0.png
eigen-2-1.png
eigen-2-2.png
eigen-2-3.png
eigen-2-4.png
eigen-3-0.png
eigen-3-1.png
eigen-3-2.png
eigen-3-3.png
eigen-3-4.png
eigen-4-0.png
eigen-4-1.png
eigen-4-2.png
eigen-4-3.png
eigen-4-4.png

# テストデータの読み込み

```
# 混合行列
result = np.zeros((10,10), dtype=np.int32)
D= 60
for i in range(10):
    for j in range(1,train_num+1):
        # テストデータの読み込み
        pat_file = "mnist/test/" + str(i) + "/" + str(i) + "_" + str(j) + ".jpg"
        work_img = Image.open(pat_file).convert('L')
        resize_img = work_img.resize((size, size))
        pat_vec = np.asarray(resize_img).astype(np.float64).flatten()
```

上位D個の固有値に対応する固有ベクトルを用いる

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

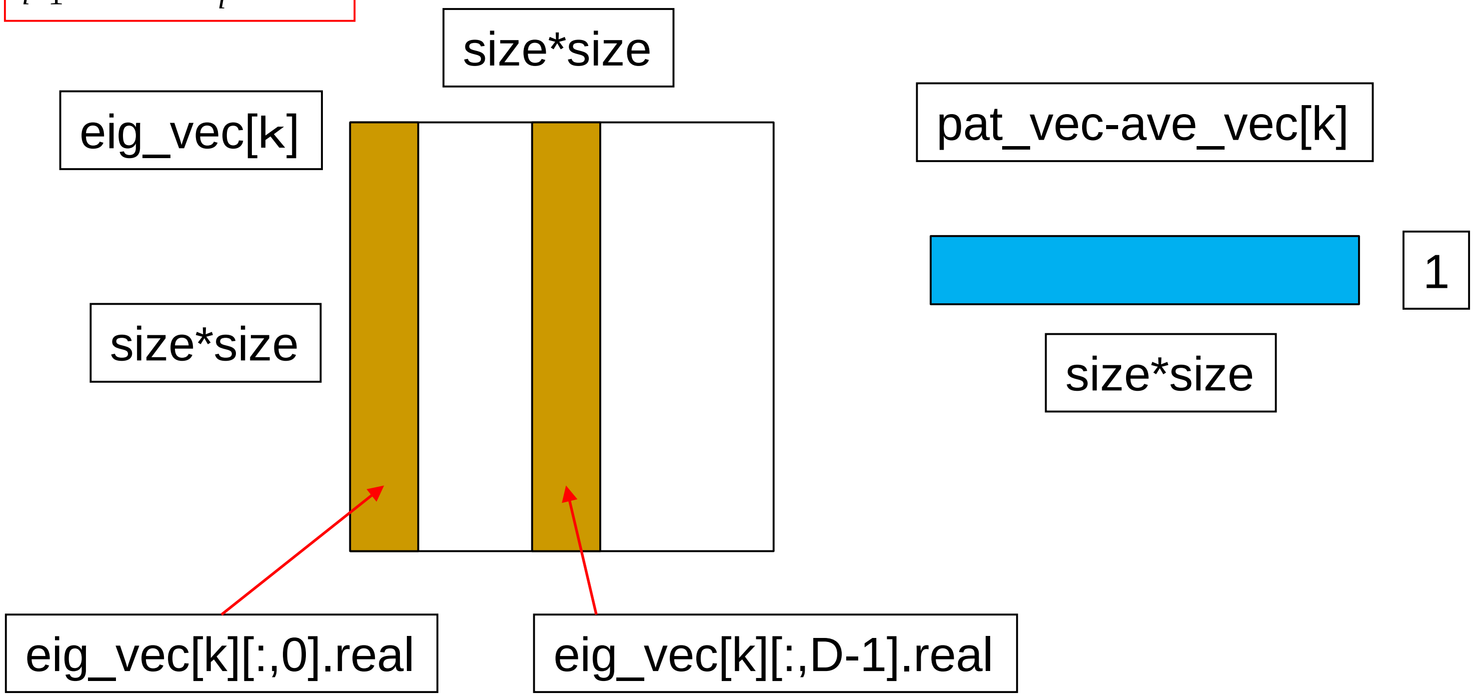
# マハラノビス距離(近似)

```
min_val = float('inf')
ans = 0
for k in range(10):
    dist = 0
    # マハラノビス距離(近似)
    for l in range(D):
        e = eig_vec[k][:,l].real
        a = pat_vec-ave_vec[k]
        dist += ( np.dot( e , a ) ) ** 2 / lamda[k][l]
```

$$\sum_{i=1}^{D} \frac{(\mathbf{u}_i^t(\mathbf{x}-\mathbf{m}))^2}{\lambda_i}$$

# 近似によるマハラノビス距離

$$\sum_{i=1}^{D} \frac{(\mathbf{u}_i^t(\mathbf{x}-\mathbf{m}))^2}{\lambda_i}$$



# 混合行列の表示

```
        if dist < min_val:
            min_val = dist
            ans = k
        result[i][ans] +=1
        print( i , j , "->" , ans )

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

最小値の探索

混合行列に予測値を代入

混合行列の出力

正解数の出力

# 実行結果

混合行列

C:¥Windows¥system32¥cmd.exe

```
 [混合行列]
[[85  0  9  2  0  4  0  0  0  0]
 [ 0 31 57  1  0  1  0  3  7  0]
 [ 0  0 96  0  1  2  0  0  1  0]
 [ 0  0 14 80  0  5  0  0  1  0]
 [ 0  0 12  1 85  1  1  0  0  0]
 [ 2  0  8 17  0 72  0  0  1  0]
 [ 3  0 22  8  8 12 45  1  1  0]
 [ 0  0 45  7  3  1  0 42  2  0]
 [ 0  0 25 13  2  4  1  0 55  0]
 [ 0  0 23  9 24  3  0  1  3 37]]

 正解数 -> 628

C:¥home¥shino¥prml-2018¥10-15¥program>
```

# 宿題③

- 同じ文字種につきK個のプロトタイプを作成し，認識することにします．
- プロトタイプは100個ある学習データからランダムにN個選択し，その平均画像を用います．
- 例えば，K=5，N＝30の場合，30個ランダムに各文字種を選択し，その平均画像をプロトタイプとすることになります．

30個の画像から作成した5個のプロトタイプ（平均画像）

# 宿題③

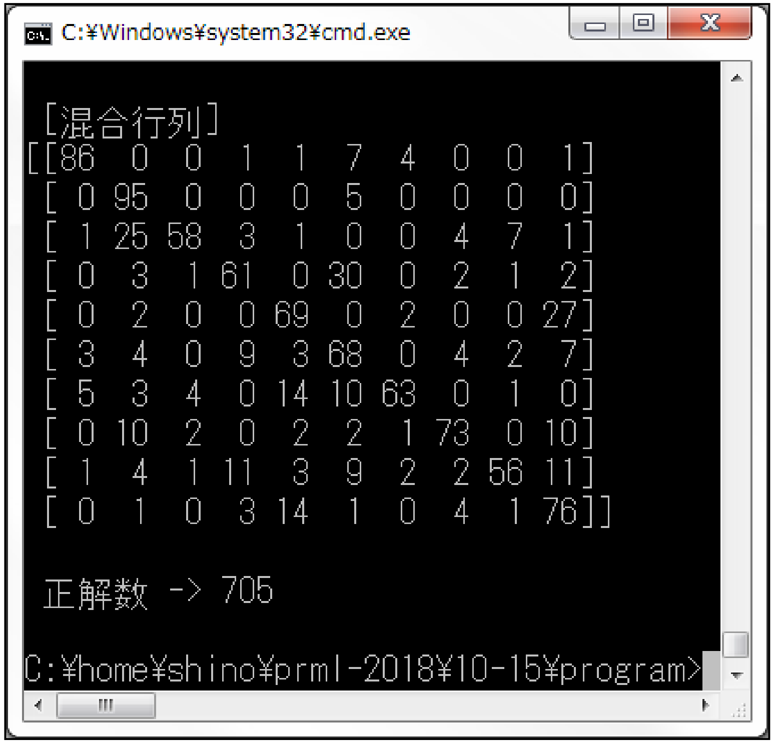
- 以上の過程で作成したプロトタイプを用いて最近傍法により認識を行ないなさい.

- 距離はユークリッド距離でかまいません.

（マハラノビス距離は難しいです）

- K，Nの値をいろいろと変え，K=1，N=100（プロトタイプが一字種につき1個）の場合の最近傍法よりも高い精度がでるか確認しなさい.

# 宿題③




K=1, N=100


K=5, N=30

# （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）
- 出口光一郎：画像認識論講義，昭晃堂（2002）
- 浜本義彦：統計的パターン認識入門，森北出版（2009）
- 平井有三：はじめてのパターン認識，森北出版（2012）