機械学習
識別モデル（識別関数法）

管理工学科
篠沢佳久

資料の内容

- 識別モデル(識別関数法)
  - 線形識別関数の学習
  - デルタルール
  - パーセプトロン

- 実習
  - パーセプトロン(Breast cancer dataset)
  - デルタルール(Iris dataset)

統計的機械学習

データの生成

P(**x**,ω)
=P(**x**|ω)P(ω)

P(**x**,ω)が分かれば無限にデータを生成できる

母集団

同時確率

P(**x**,ω)

クラスω

事前確率

P(ω)

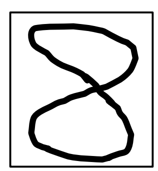
条件付き確率

P(**x**|ω)

クラスの特徴

P(**x**|ω)が分かればクラスωの特徴分布が分かる

観測データ **x**



事後確率

P(ω|**x**)

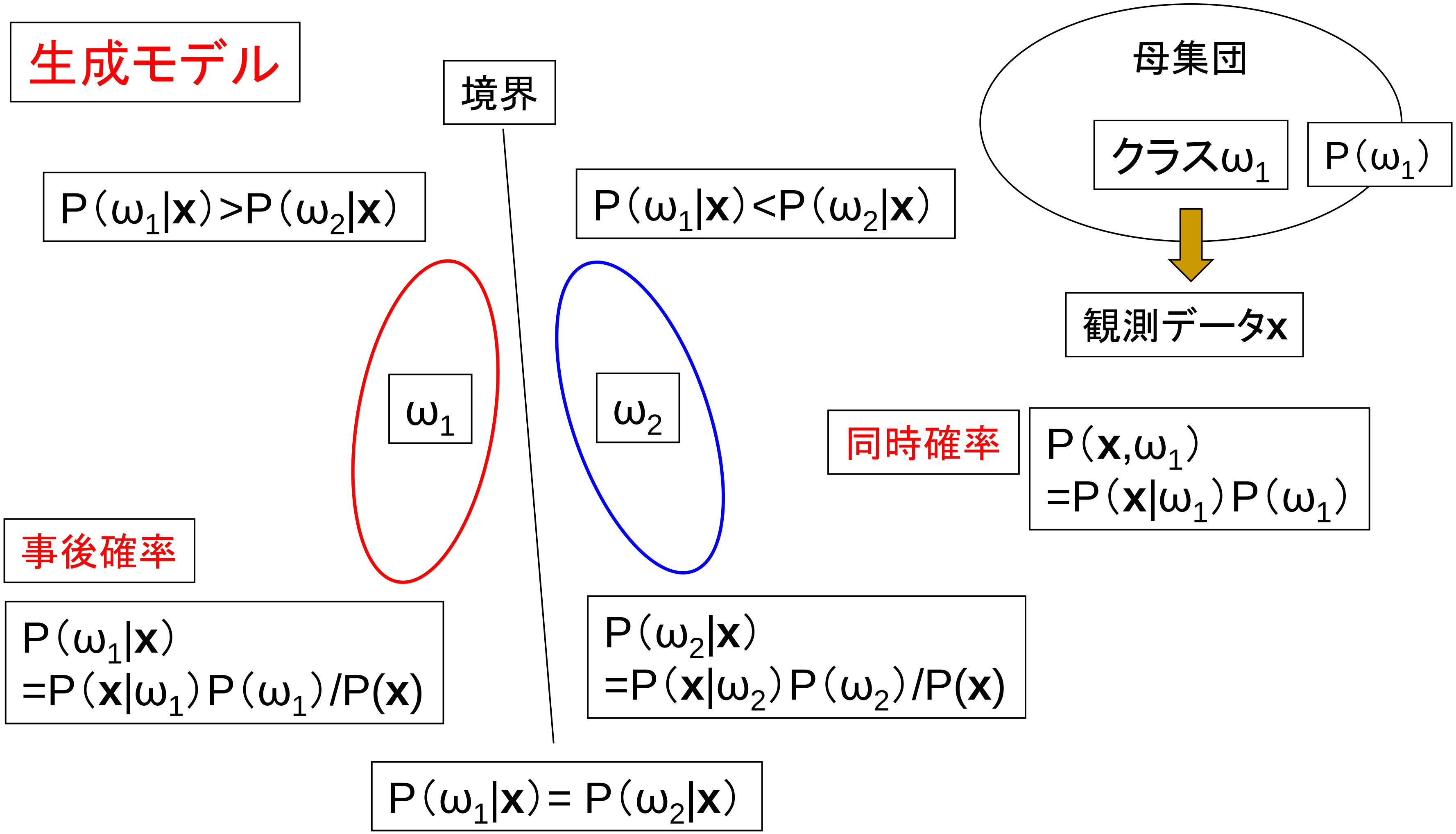
P(ω|**x**)が分かれば**x**の予測ができる

データの予測

P(ω|**x**)
=P(**x**|ω)P(ω)/P(**x**)

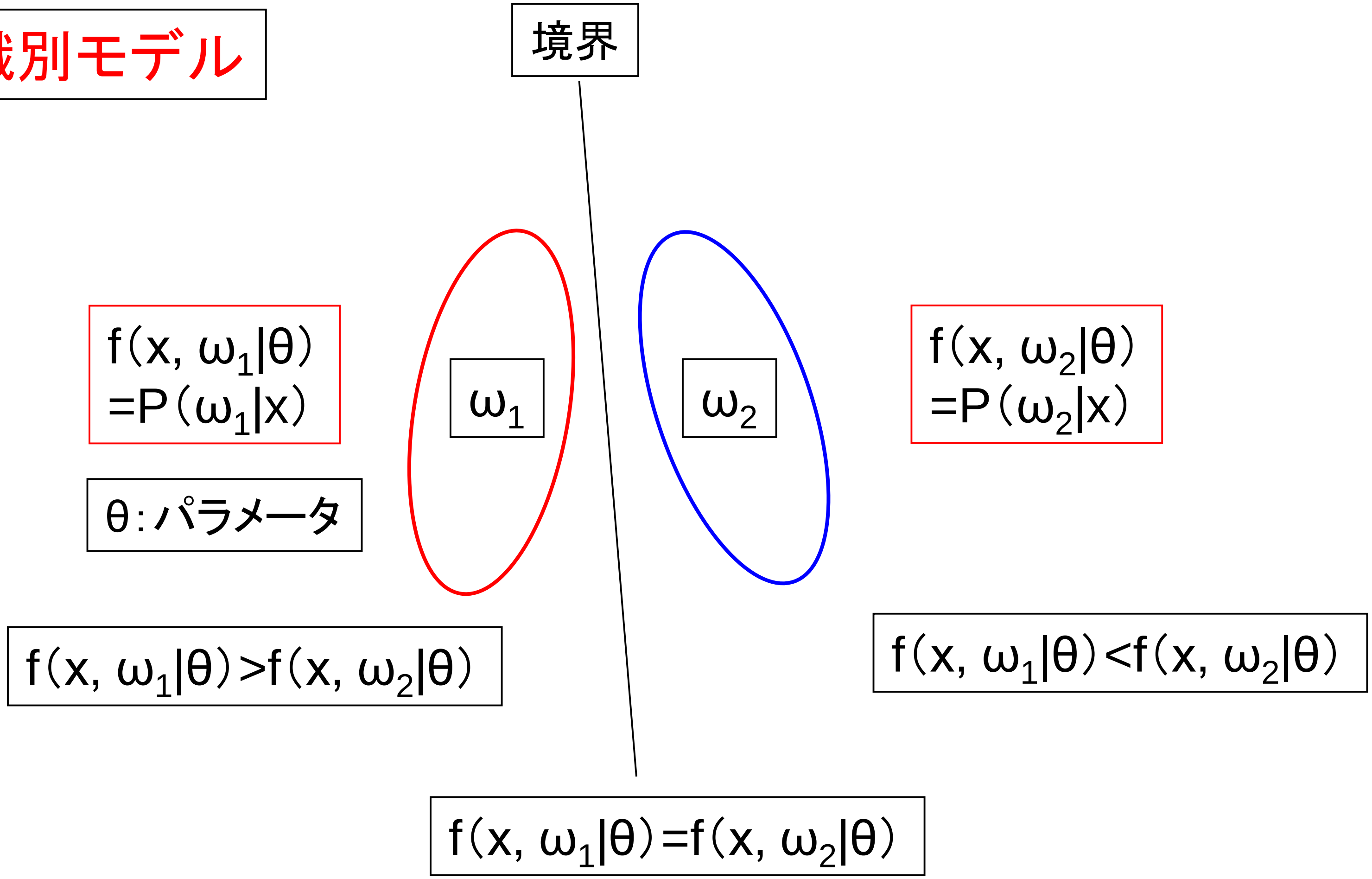
同時確率を間接的に用いて予測
(生成モデル)

生成モデル


生成モデル
境界
母集団
クラスω_1
$P(\omega_1)$
観測データ$\mathbf{x}$
$P(\omega_1|\mathbf{x})>P(\omega_2|\mathbf{x})$
$P(\omega_1|\mathbf{x})<P(\omega_2|\mathbf{x})$
ω_1
ω_2
同時確率
$P(\mathbf{x},\omega_1)$
$=P(\mathbf{x}|\omega_1)P(\omega_1)$
事後確率
$P(\omega_1|\mathbf{x})$
$=P(\mathbf{x}|\omega_1)P(\omega_1)/P(\mathbf{x})$
$P(\omega_2|\mathbf{x})$
$=P(\mathbf{x}|\omega_2)P(\omega_2)/P(\mathbf{x})$
$P(\omega_1|\mathbf{x})=P(\omega_2|\mathbf{x})$

識別モデル

識別モデル



*識別関数法とも呼ばれます

識別モデル（識別関数法）

線形識別関数の学習

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**x**（d次元）の属するクラスを予測

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

- 関数 g_i を識別関数と呼ぶ

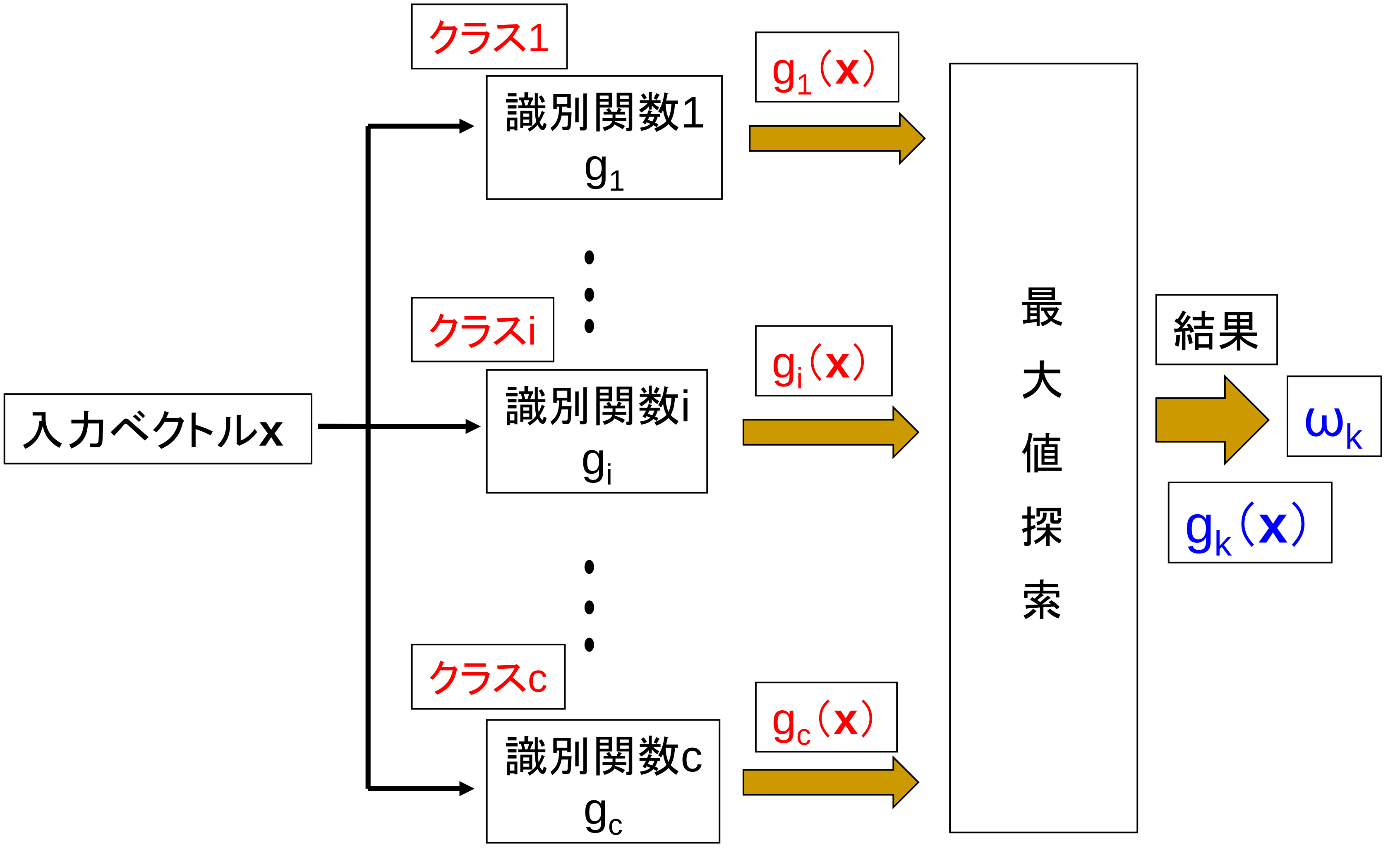
識別関数法のアルゴリズム①

```
max = −∞

for i in range( c ):
    val = g_i(x)
    if val > max:
        max = val
        answer = i

クラス answer が予測結果
```

識別関数の計算

c ： クラスの総数
g_i ： クラス i の識別関数
x ： 調べたいデータの特徴ベクトル

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
- 予測したいデータの特徴ベクトル $\mathbf{x}$（d次元）
  - 条件付き確率 $p(\mathbf{x}|\omega_i)$
  - 特徴ベクトル $\mathbf{x}$ はどのクラスに属するか

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

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

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

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）

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

$$\Downarrow$$

識別関数

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

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}

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

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

線形識別関数

  - ベクトル表記

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

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

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…

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

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

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

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

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

w_{i0}に対する入力データを1とする

線形識別関数のアルゴリズム

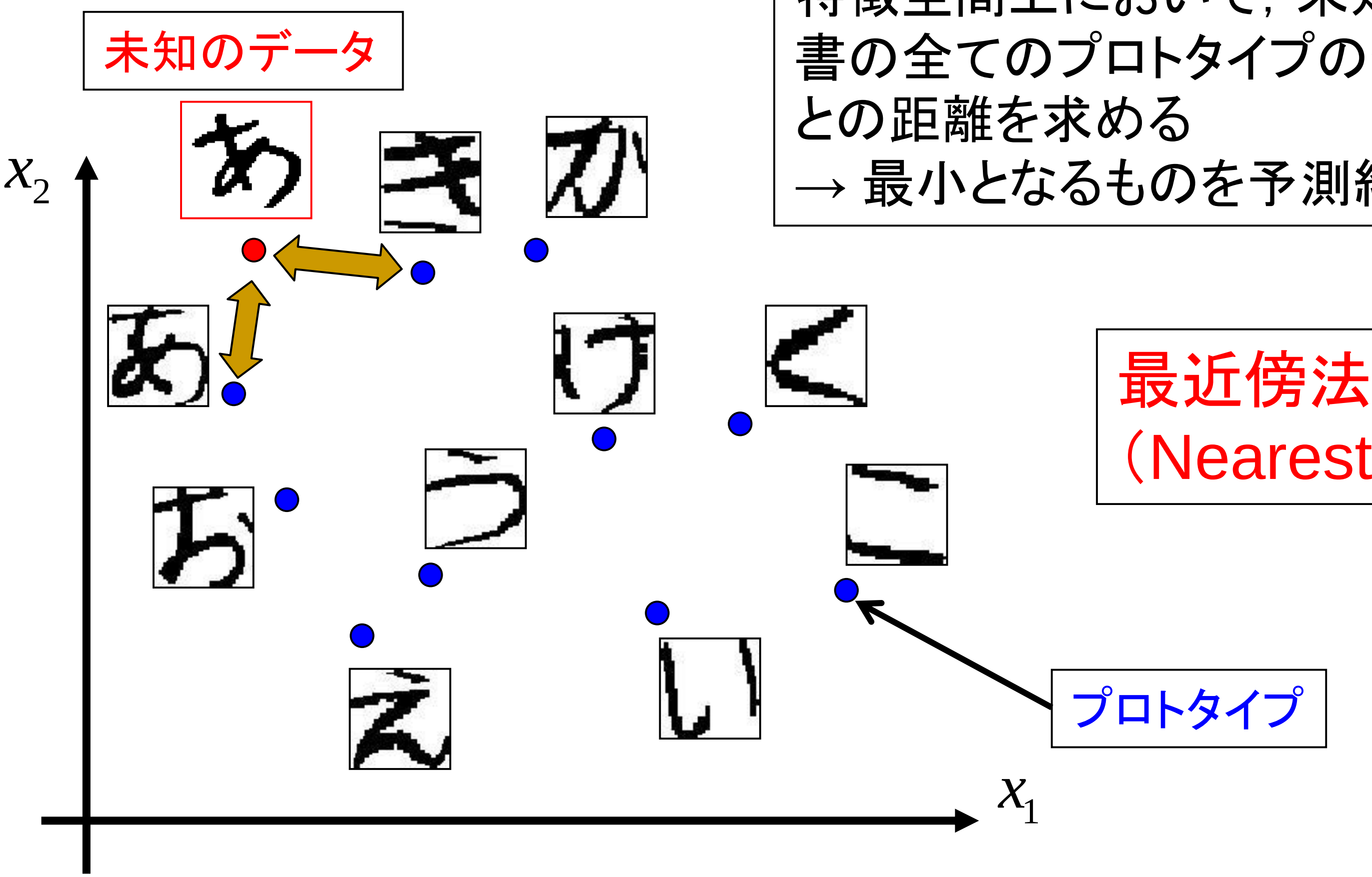
```
max = −∞

for i in range( c ):
    val = 0
    for j in range(1,d+1):
        val += w[ i ][ j ] * x[ j ]
    val += w[ i ][ 0 ]
    if  val > max:
        max = val
        answer = i

クラス answer が予測結果
```

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

線形識別関数のパラメータ（重み）はどう決めればよいのか

最近傍法（復習）①



特徴空間上において，未知データと辞書の全てのプロトタイプの特徴ベクトルとの距離を求める
→ 最小となるものを予測結果

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 **x**の属するクラスを調べる

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

距離が最も近いプロトタイプを予測結果とする

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

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

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

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

ベイズ決定則と線形識別関数

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

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

$\Updownarrow$

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

$\Rightarrow$

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

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

  - ベイズ決定則

  → 各クラスの平均ベクトル

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

↓

- 学習
  - デルタルール，パーセプトロン

線形識別関数の学習

デルタルール

線形識別関数

- 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**x**（d次元）の属するクラスを調べる

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

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

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

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

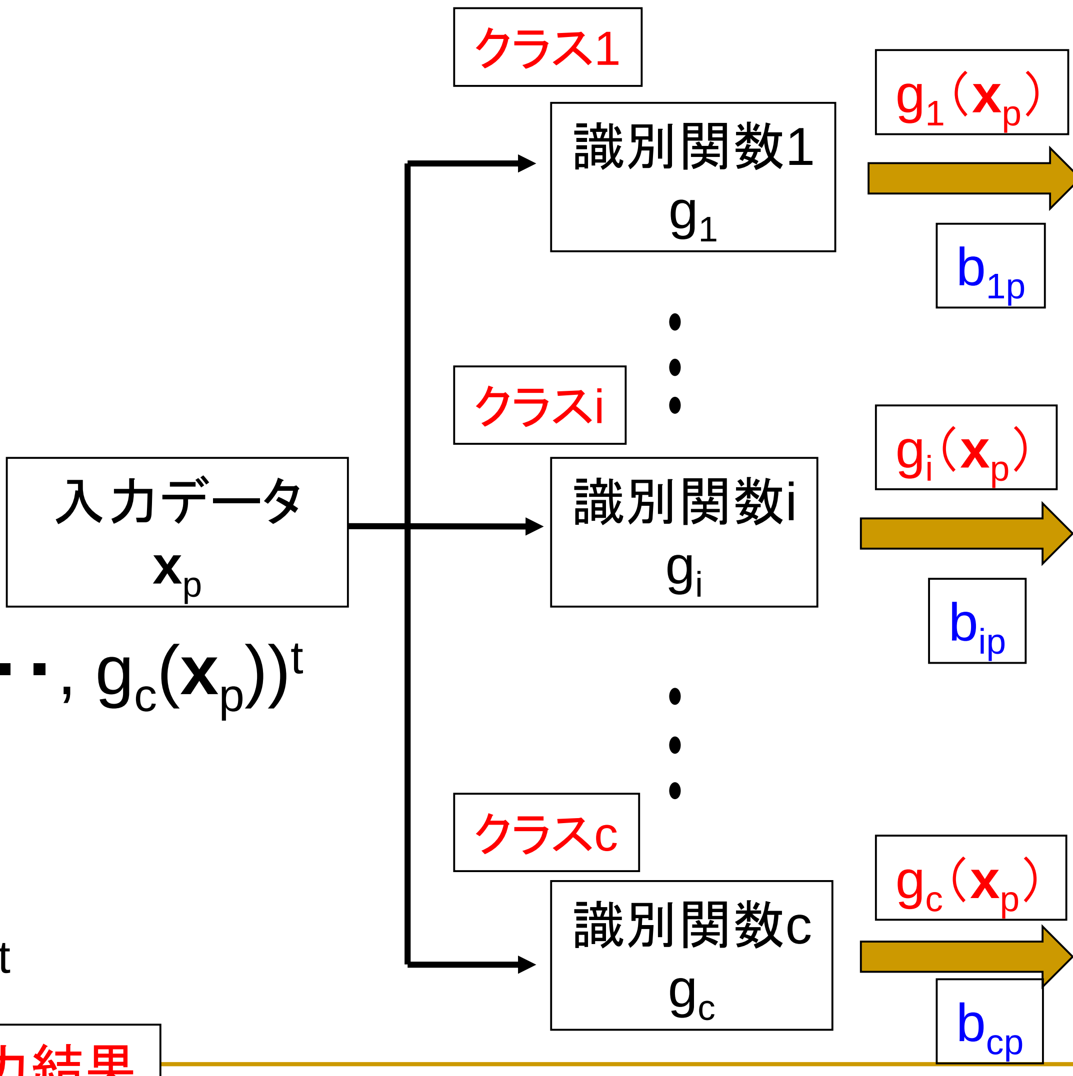
教師信号(ベクトル)①

- 学習データ
  - $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$
  - 入力データ $\mathbf{x}_p$

出力

  - $(g_1(\mathbf{x}_p), g_2(\mathbf{x}_p), \cdots, g_c(\mathbf{x}_p))^t$
  - $(b_{1p}, b_{2p}, \cdots, b_{cp})^t$

望ましい出力結果



教師信号（ベクトル）②

- 教師信号
  - 出力 $g_i(\mathbf{x}_p)$ に対して望ましい出力 b_{ip}

$$\mathbf{x}_p \in \omega_i \Rightarrow b_{ip} > b_{jp}$$

一般的には,
正例ラベル → 1
負例ラベル → 0

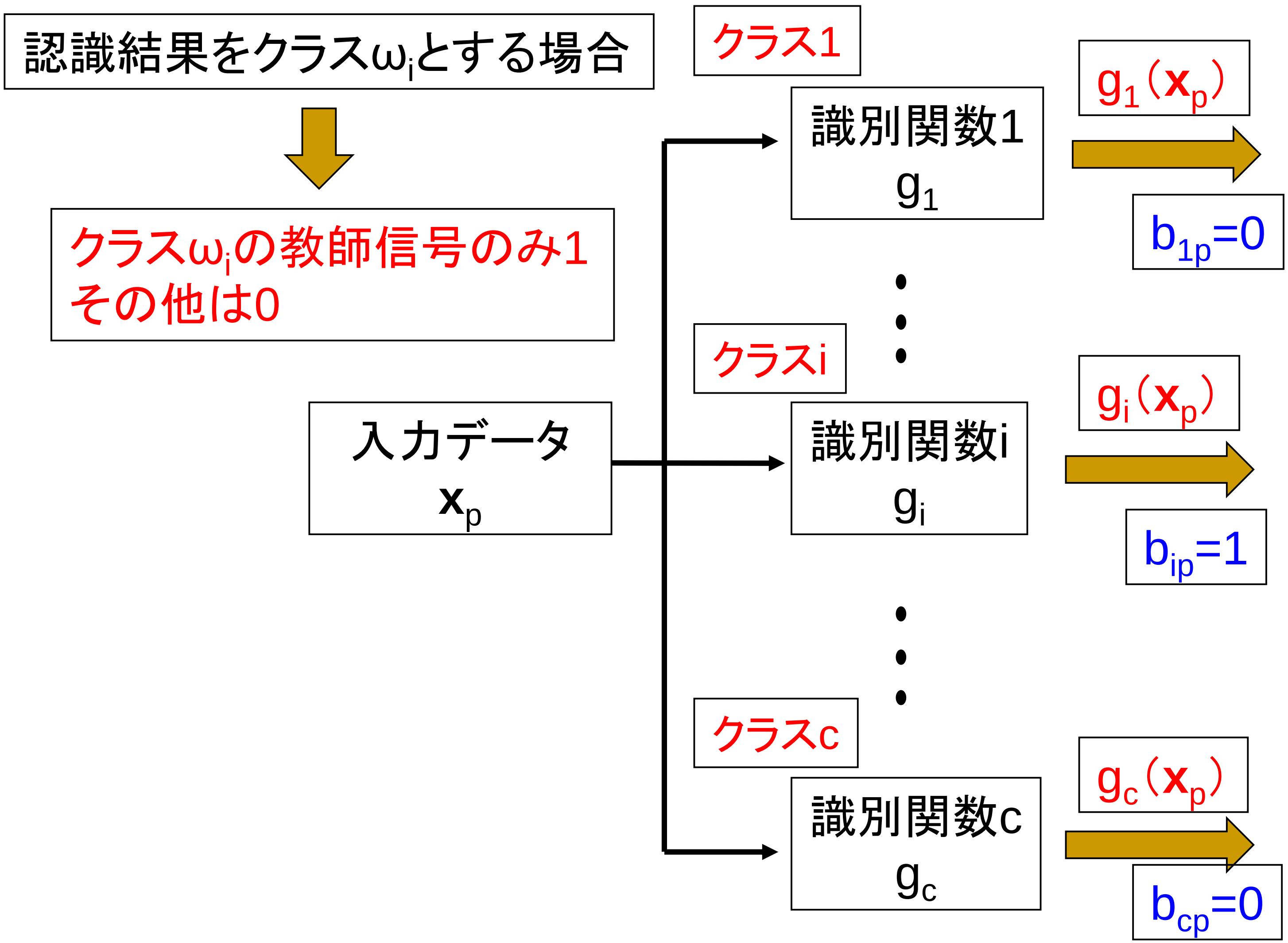
- 教師信号ベクトル
  - $(b_{1p}, b_{2p}, \cdots, b_{cp})^t$
  - （例）

i番目の要素

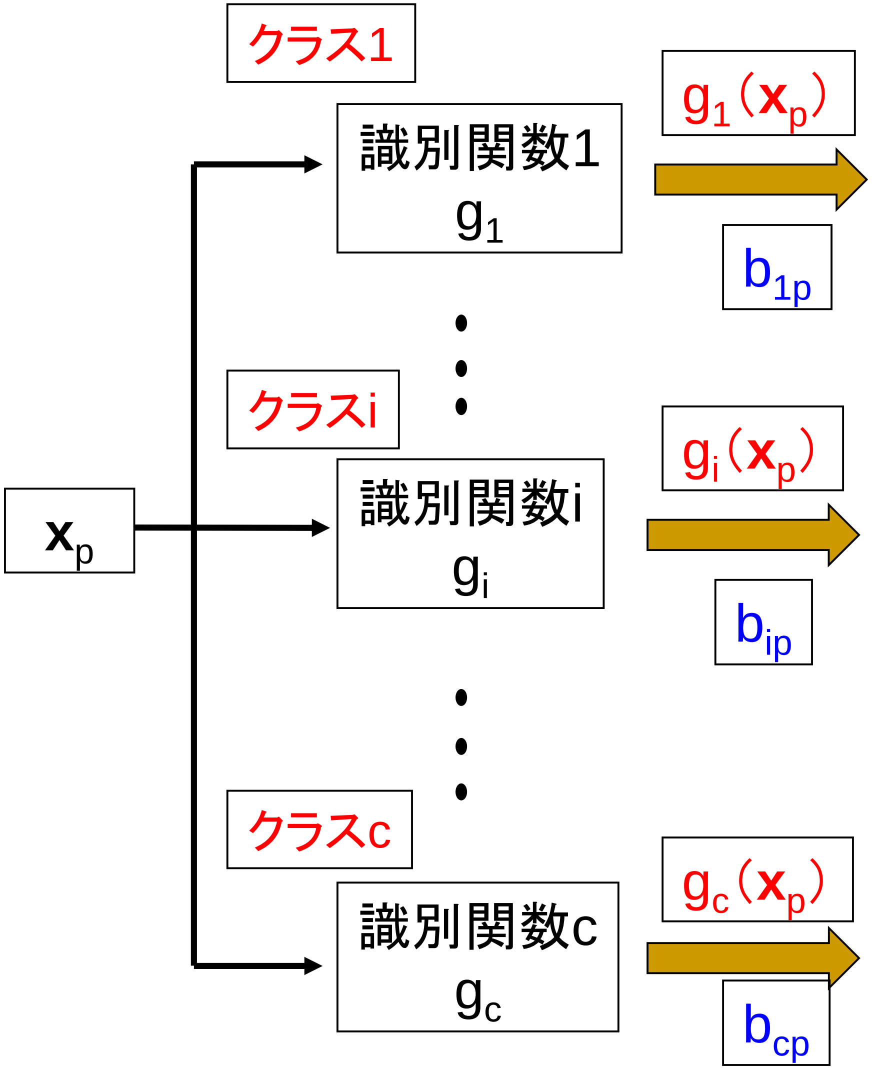
$$\mathbf{x}_p \in \omega_i \Rightarrow (0,0,\ldots,0,1,0,\cdots,0)^t$$

  - $b_{ip}=1$，他は0とする

教師信号(ベクトル)の一例



重みの決め方①



全ての学習データにおいて，出力値が教師信号と一致することが望ましい

学習データ$\mathbf{x}_p$に対する各識別関数での教師信号との誤差

$$\varepsilon_{ip} = g_i(\mathbf{x}_p) - b_{ip}$$

全クラスでの誤差二乗和

$$J_p = \sum_{i=1}^{c} \varepsilon_{ip}^2$$

誤差関数

全ての学習データの誤差二乗和

$$J = \sum_{p=1}^{n} J_p = \sum_{p=1}^{n} \sum_{i=1}^{c} \varepsilon_{ip}^2$$

重みの決定②

- 全ての学習データの誤差二乗和が最小となるように重み係数を決定

$$J = \sum_{p=1}^{n}\sum_{i=1}^{c}\varepsilon_{ip}^{2} = \sum_{p=1}^{n}\sum_{i=1}^{c}(g_i(\mathbf{x}_p) - b_{ip})^2$$

最小

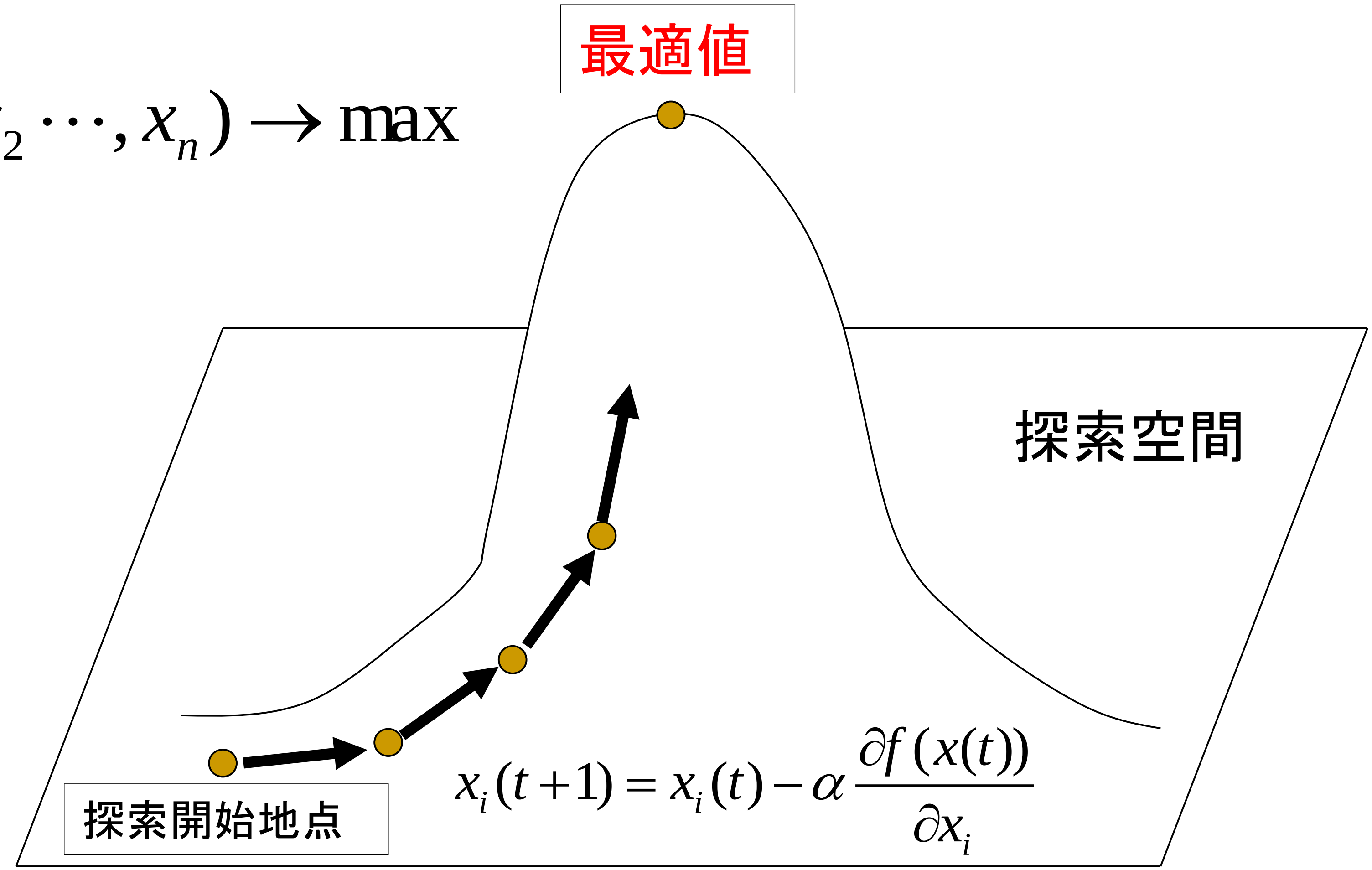
↓ データごとの誤差二乗和

$$J_p = \sum_{i=1}^{c}\varepsilon_{ip}^{2} = \sum_{i=1}^{c}(g_i(\mathbf{x}_p) - b_{ip})^2$$

最小

最急降下法

$$f(x_1, x_2 \cdots, x_n) \rightarrow \max$$



α ：学習係数

最急降下法による重みの決定

- 確率的勾配降下法（オンライン学習）
  - 各学習データ$\mathbf{x}_p$が示される度に重みを修正

$$J_p = \sum_{i=1}^{c} \varepsilon_{ip}^2 = \sum_{i=1}^{c} (g_i(\mathbf{x}_p) - b_{ip})^2$$

$$\boxed{\frac{\partial J_p}{\partial \mathbf{w}_i} = \frac{\partial J_p}{\partial g_i(\mathbf{x}_p)} \frac{\partial g_i(\mathbf{x}_p)}{\partial \mathbf{w}_i}}$$

$$\mathbf{w}_i^{'} = \mathbf{w}_i - \alpha \frac{\partial J_p}{\partial \mathbf{w}_i}$$

$$\frac{\partial J_p}{\partial g_i(\mathbf{x}_p)} = 2(g_i(\mathbf{x}_p) - b_{ip})$$

$$\Downarrow$$

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

$$\mathbf{w}_i^{'} = \mathbf{w}_i - \alpha (g_i(\mathbf{x}_p) - b_{ip}) \mathbf{x}_p$$

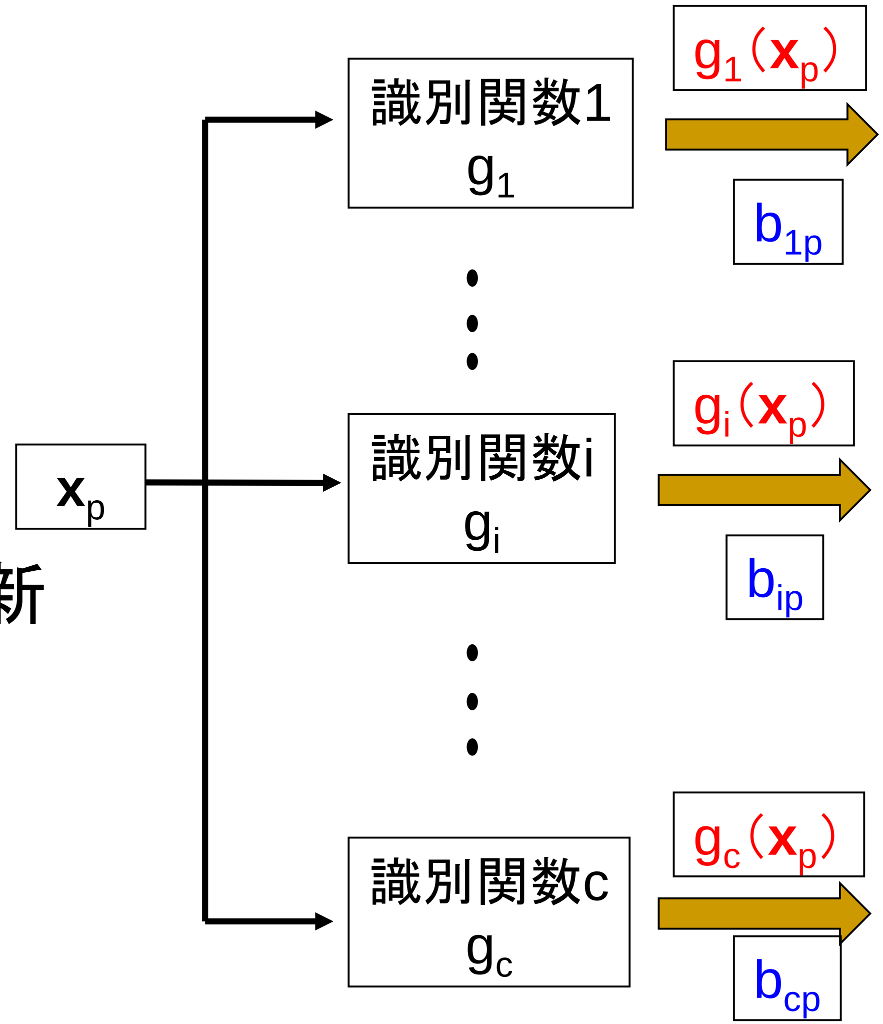
$$= \mathbf{w}_i - \alpha \underbrace{(\mathbf{w}_i^t \mathbf{x}_p - b_{ip})}_{\boxed{\text{間違えた量}}} \underbrace{\mathbf{x}_p}_{\boxed{\text{入力データ}}}$$

デルタルール*による重みの決定

- 重みの更新方法

$$\mathbf{w}_i' = \mathbf{w}_i - \alpha(g_i(\mathbf{x}_p) - b_{ip})\mathbf{x}_p$$
$$= \mathbf{w}_i - \alpha(\mathbf{w}_i^t\mathbf{x}_p - b_{ip})\mathbf{x}_p$$

  - 全ての$\mathbf{w}_i$において更新
  - （i=1,2,･･･,c）



*Widrow-Hoffの学習規則（ADALINEと呼ばれるニューラルネットワークの学習規則）とも呼ばれる

デルタールールのアルゴリズム

while True:
　Error = 0
　for p in range(N):
　　for i in range(c):
　　　$g_i(\mathbf{x}_p)$ の計算
　　　$e_{ip} = (g_i(\mathbf{x}_p) – b_{ip})$
　　　for j in range(d):
　　　　w_{ij} -= alpha * e_{ip} * x_{pj}
　　　Error += e_{ip} * e_{ip}

　if Error < ε : break

重みベクトルは乱数によって
事前に初期化しておく

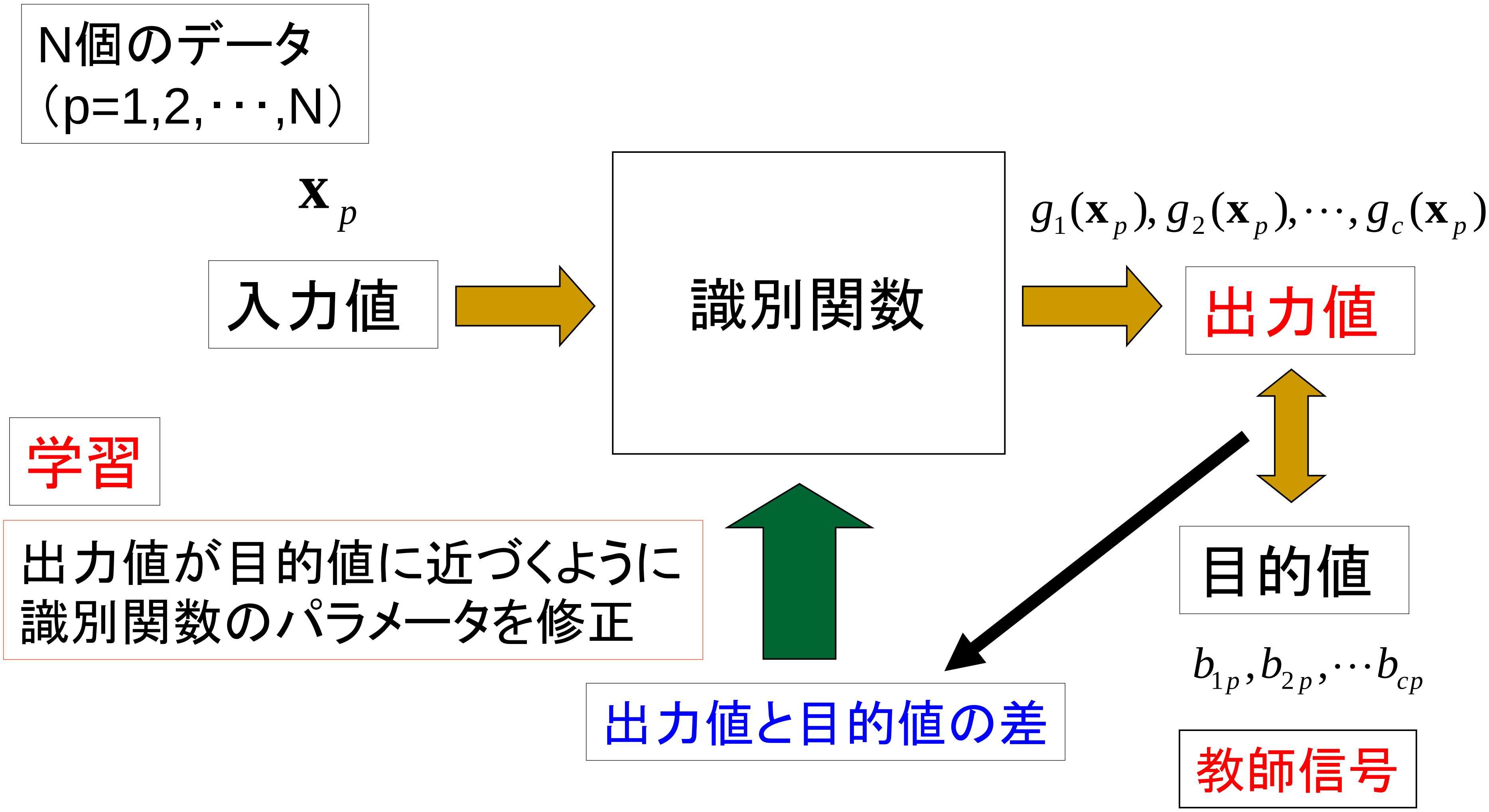
誤差の計算

重みの修正

誤差二乗和が一定値
以下となったら停止

c ： クラスの総数　　　d ： 次元数
w_{ij} ： クラス i の識別関数の j番目の重み係数
x_{pj} ： 特徴ベクトル$\mathbf{x}_p$の j番目の要素

学習による識別関数の構築



ロジスティック回帰(復習)

- ロジスティック回帰

予測値 $$y_i = \frac{1}{1+\exp(-(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_N x_{iN}))}$$

$y_i > 0.5 \rightarrow$ 正例

$y_i < 0.5 \rightarrow$ 負例

- 誤差関数

クロスエントロピー

$$E = -\sum_{i=1}^{P} (t_i \log y_i + (1-t_i)\log(1-y_i))$$

ロジスティック回帰（復習）

i番目のデータに対する修正
（オンライン学習）

$$\beta_j \leftarrow \beta_j - \alpha \frac{\partial E_i}{\partial \beta_j}$$

$$\frac{\partial E_i}{\partial \beta_j} = \underbrace{(y_i - t_i)}_{\text{予測値と正解値の差}} \underbrace{x_{ij}}_{\text{i番目のデータ}}$$

予測値と正解値の差

i番目のデータ

全てのデータに対する修正
（バッチ学習）

$$\beta_j \leftarrow \beta_j - \alpha \frac{\partial E}{\partial \beta_j}$$

$$\frac{\partial E}{\partial \beta_j} = \sum_{i=1}^{P} (y_i - t_i) x_{ij}$$

デルタルールと同じ学習アルゴリズム

パーセプトロン

線形分離可能

線形分離

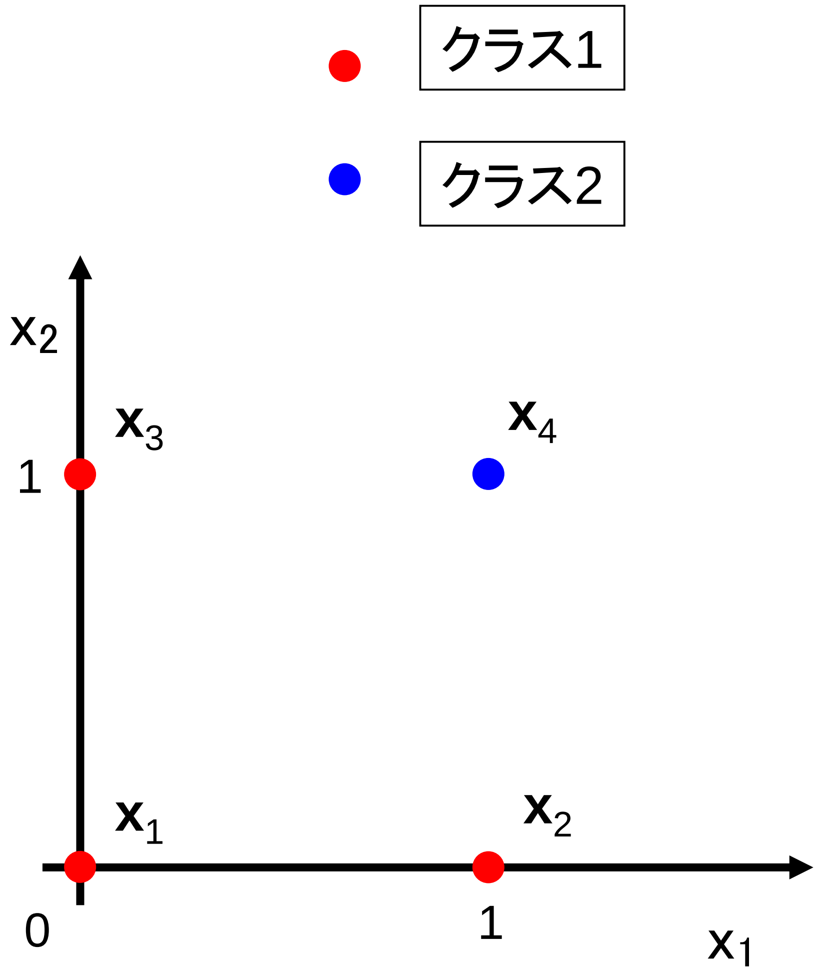
- クラスω_i（i=1,2,・・・,c）
- クラスω_iの学習パターンの集合χ_i
- χ_i に属する全ての学習データ**x**

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad (j = 1,2,\cdots,c \quad j \neq i)$$

- 上記の式を満たす重みベクトル**w**が存在する場合，線形分離可能と呼ぶ

線形分離可能

	x_1	x_2	
$\mathbf{x}_1$	0	0	クラス1
$\mathbf{x}_2$	1	0	クラス1
$\mathbf{x}_3$	0	1	クラス1
$\mathbf{x}_4$	1	1	クラス2



数値例①

	x_1	x_2	
$\mathbf{x}_1$	0	0	クラス1
$\mathbf{x}_2$	1	0	クラス1
$\mathbf{x}_3$	0	1	クラス1
$\mathbf{x}_4$	1	1	クラス2

識別関数1
g_1

$w_{10}=1.25$
$w_{11}=-0.50$
$w_{12}=-0.50$

識別関数2
g_2

$w_{10}=-0.25$
$w_{11}=0.50$
$w_{12}=0.50$

識別関数

$$g_i(\mathbf{x}) = w_{i0} + \sum_{j=1}^{2} w_{ij} x_j$$

$\mathbf{x}_1=(0,0)^t$

$g_1(\mathbf{x}_1)=1.25 > g_2(\mathbf{x}_1)=-0.25$

$\mathbf{x}_2=(1,0)^t$

$g_1(\mathbf{x}_2)=0.75 > g_2(\mathbf{x}_2)=0.25$

数値例②

	x_1	x_2	
$\mathbf{x}_1$	0	0	クラス1
$\mathbf{x}_2$	1	0	クラス1
$\mathbf{x}_3$	0	1	クラス1
$\mathbf{x}_4$	1	1	クラス2

識別関数1
g_1

$w_{10}=1.25$
$w_{11}=-0.50$
$w_{12}=-0.50$

識別関数2
g_2

$w_{10}=-0.25$
$w_{11}=0.50$
$w_{12}=0.50$

識別関数

$$g_i(\mathbf{x}) = w_{i0} + \sum_{j=1}^{2} w_{ij} x_j$$

$\mathbf{x}_3=(0,1)^t$

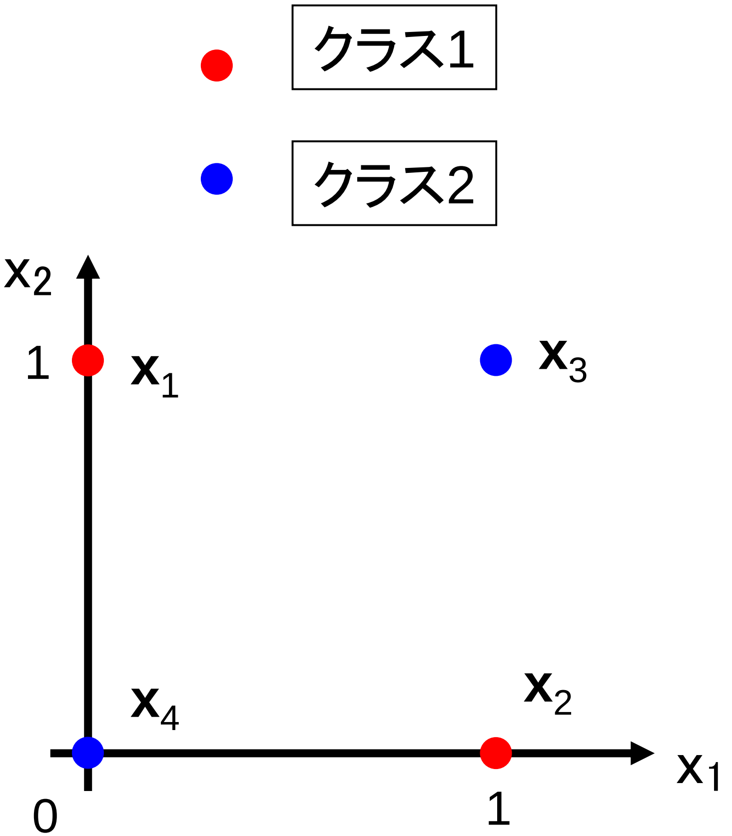
$g_1(\mathbf{x}_3)=0.75 > g_2(\mathbf{x}_3)=0.25$

$\mathbf{x}_4=(1,1)^t$

$g_1(\mathbf{x}_4)=0.25 < g_2(\mathbf{x}_4)=0.75$

線形分離不可能

	x_1	x_2	
$\mathbf{x}_1$	0	1	クラス1
$\mathbf{x}_2$	1	0	クラス1
$\mathbf{x}_3$	1	1	クラス2
$\mathbf{x}_4$	0	0	クラス2



クラス1とクラス2を識別できる重みは存在しない
→ 線形分離不可能

二クラスの線形分離

- 線形識別関数g_1, g_2
  - $g_1(\mathbf{x}) > g_2(\mathbf{x}) \rightarrow \mathbf{x}$ はクラス1
  - $g_1(\mathbf{x}) < g_2(\mathbf{x}) \rightarrow \mathbf{x}$ はクラス2

線形識別関数 g

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x}) = \mathbf{w}_1{}^t\mathbf{x} - \mathbf{w}_2{}^t\mathbf{x}$$

$$= (\mathbf{w}_1 - \mathbf{w}_2)^t\mathbf{x} = \mathbf{w}^t\mathbf{x}$$

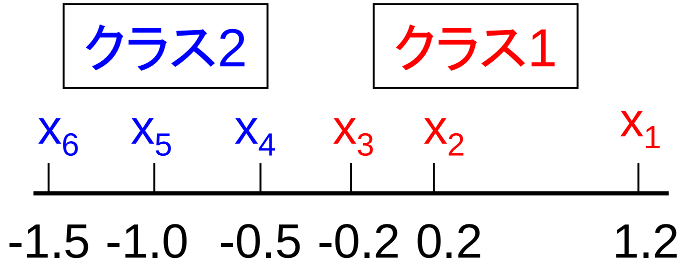
$$\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_2$$

$g(\mathbf{x}) > 0 \rightarrow$ クラス1
$g(\mathbf{x}) < 0 \rightarrow$ クラス2
$g(\mathbf{x}) = 0$ が決定境界

数値例

クラス1とクラス2を判別できる
重みベクトル**w**を求める

- クラス1
  - $x_1 = 1.2,\ x_2 = 0.2,\ x_3 = -0.2$
- クラス2
  - $x_4 = -0.5,\ x_5 = -1.0,\ x_6 = -1.5$

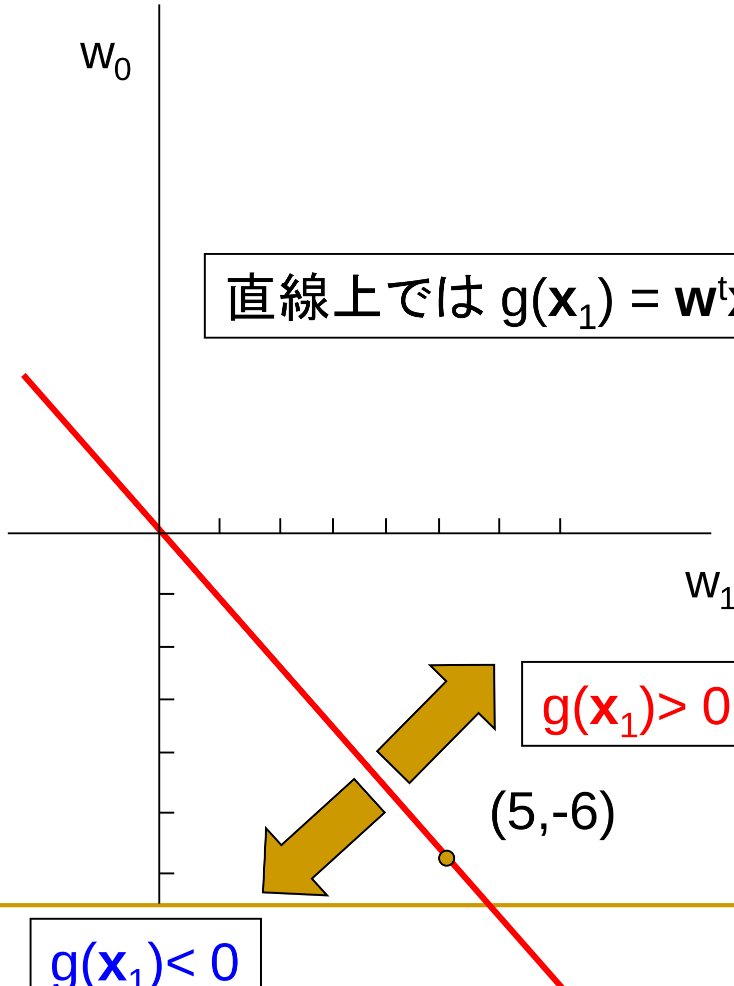


$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} = w_0 + w_1 x_1$$
$$\mathbf{w} = (w_0, w_1)^t$$
$$\mathbf{x} = (1, x_1)^t$$

- クラス1
  - $\mathbf{x}_1 = (1,1.2)^t,\ \mathbf{x}_2 = (1,0.2)^t,\ \mathbf{x}_3 = (1,-0.2)^t$
- クラス2
  - $\mathbf{x}_4 = (1,-0.5)^t,\ \mathbf{x}_5 = (1,-1.0)^t,\ \mathbf{x}_6 = (1,-1.5)^t$

重み空間*①

$\mathbf{x}_1 = (1,1.2)^t$ において$g(\mathbf{x}_1) > 0$ が成り立つ領域は?



直線上では $g(\mathbf{x}_1) = \mathbf{w}^t\mathbf{x}_1 = 0$ が成立

$$w_0 = -\frac{6}{5} w_1$$

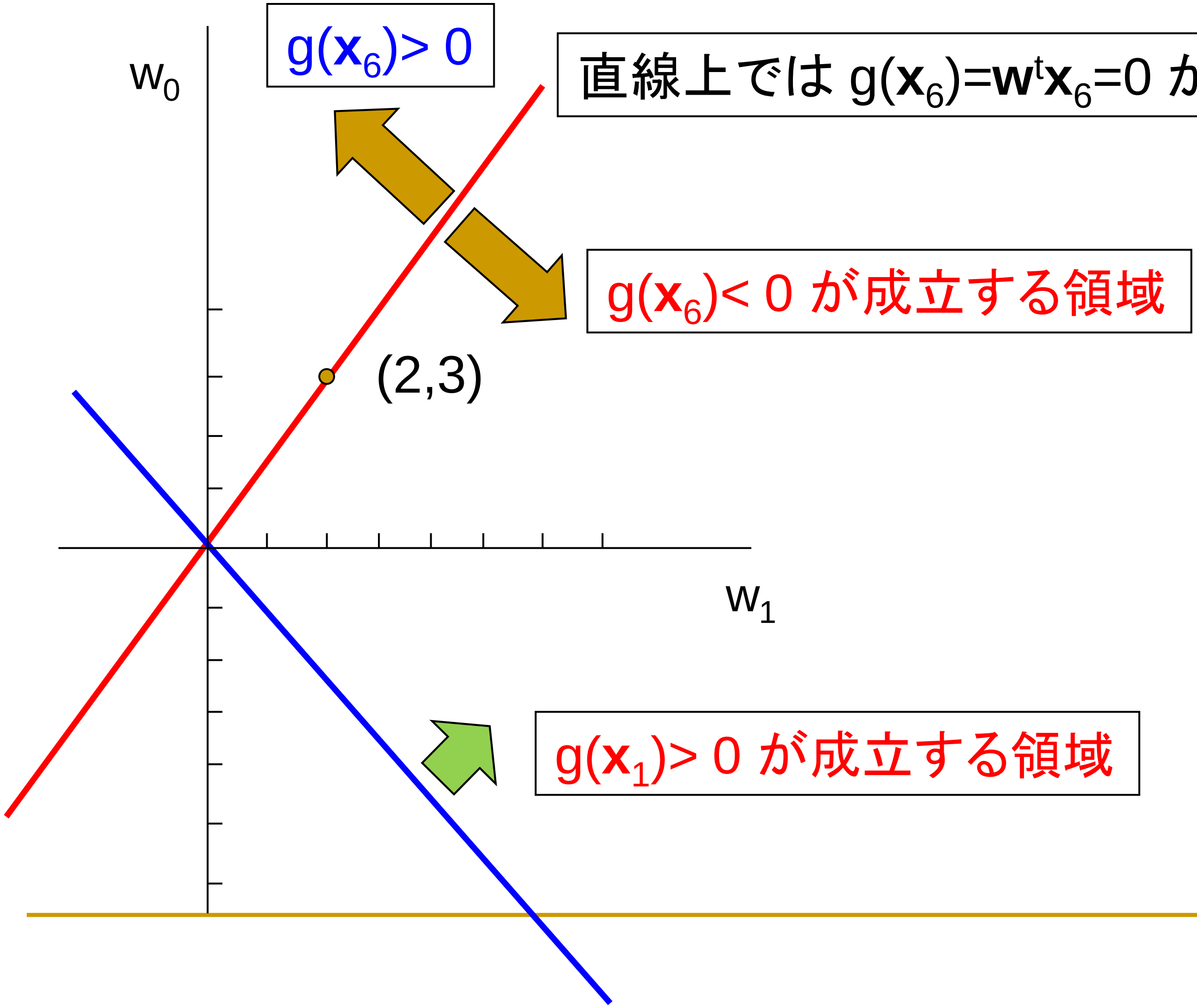
$g(\mathbf{x}_1) > 0$ が成立する領域 → 解が存在する領域

$g(\mathbf{x}_1) < 0$

*$w_1 > 0$の領域のみ考える

重み空間②

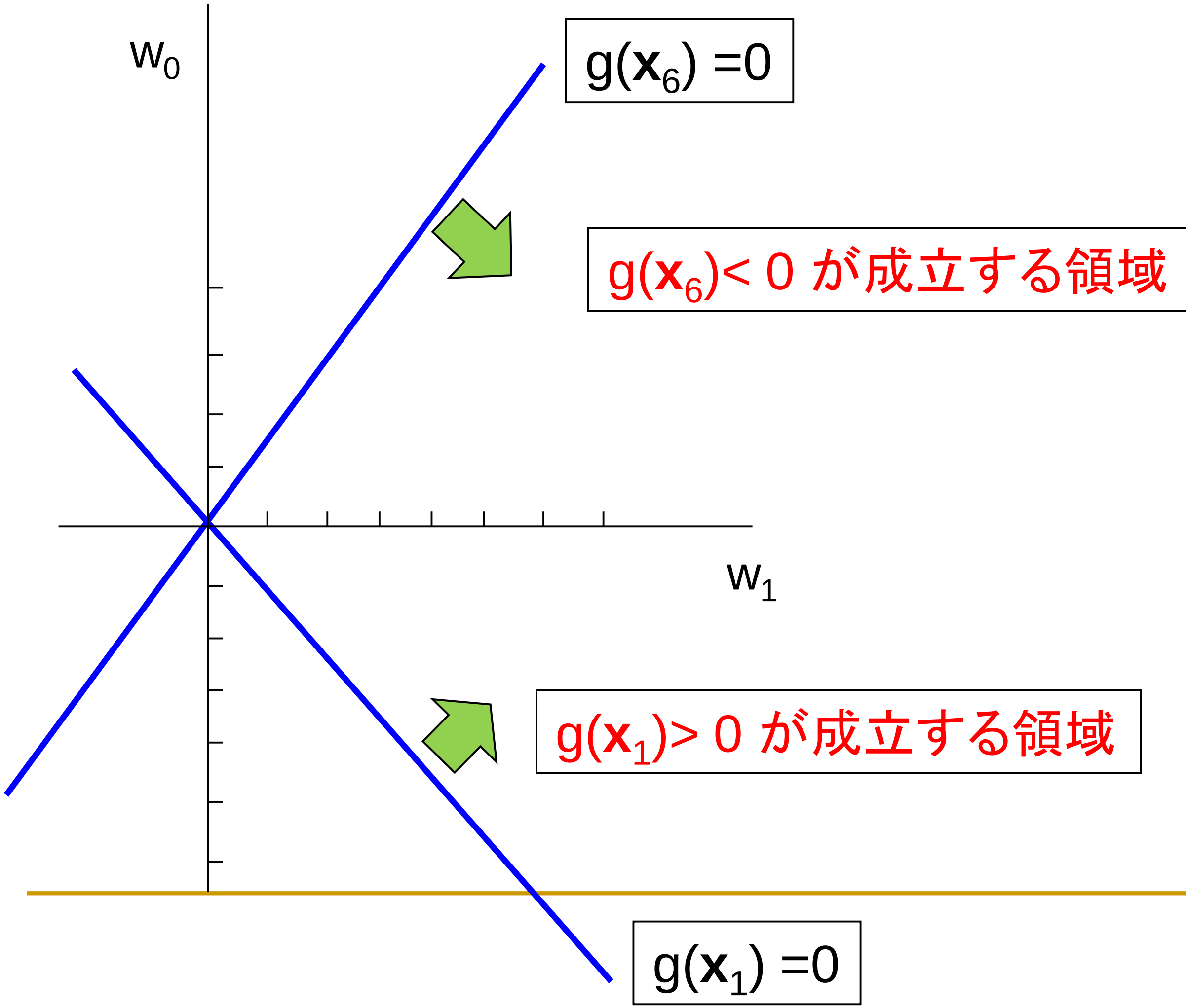
$\mathbf{x}_6 = (1,-1.5)^t$ において$g(\mathbf{x}_6) < 0$ が成り立つ領域は?



$$w_0 = -\frac{3}{2} w_1$$

重み空間③

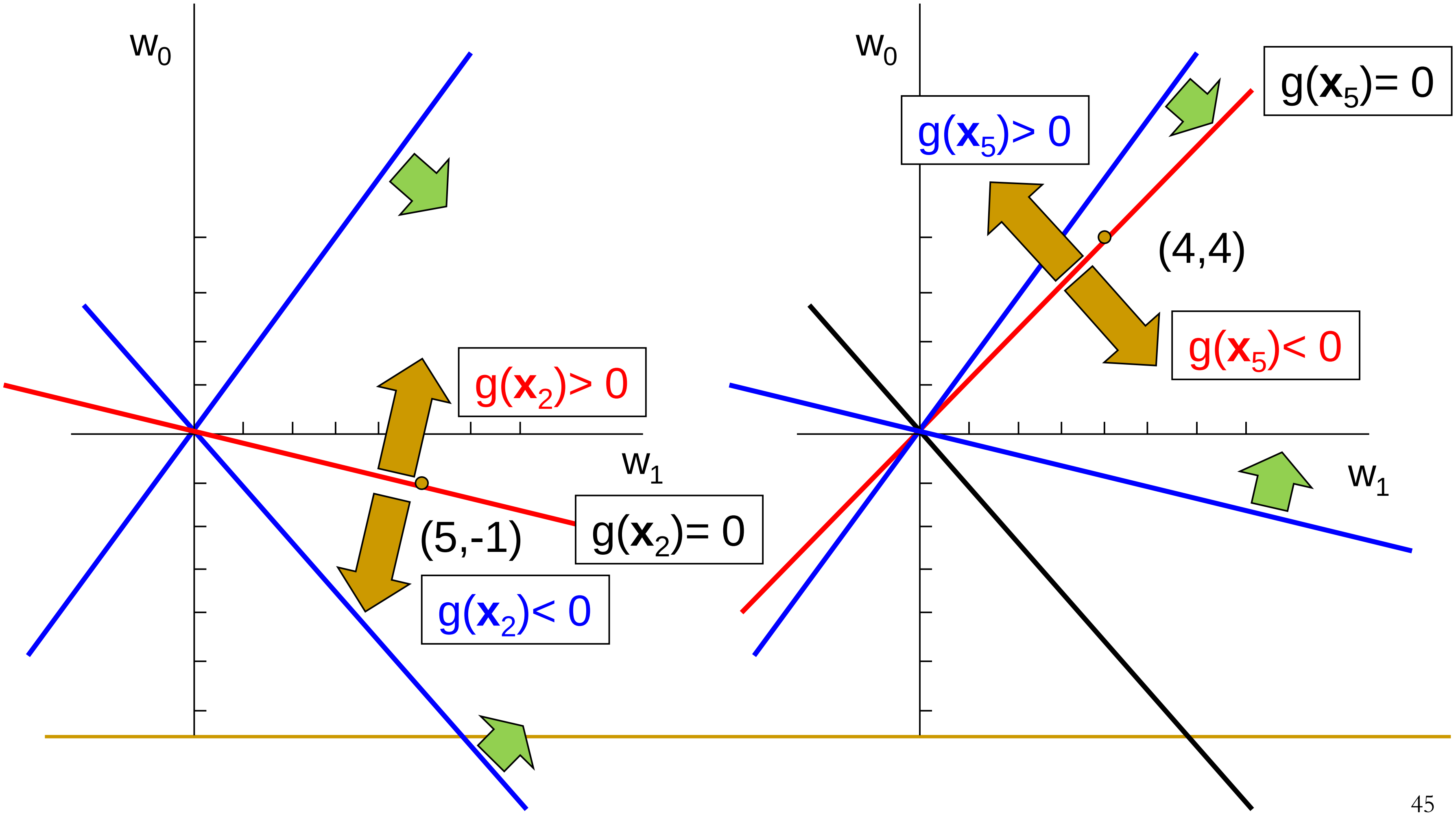
g($\mathbf{x}_1$)> 0 かつ g($\mathbf{x}_6$)< 0 が成り立つ領域は?



重み空間④

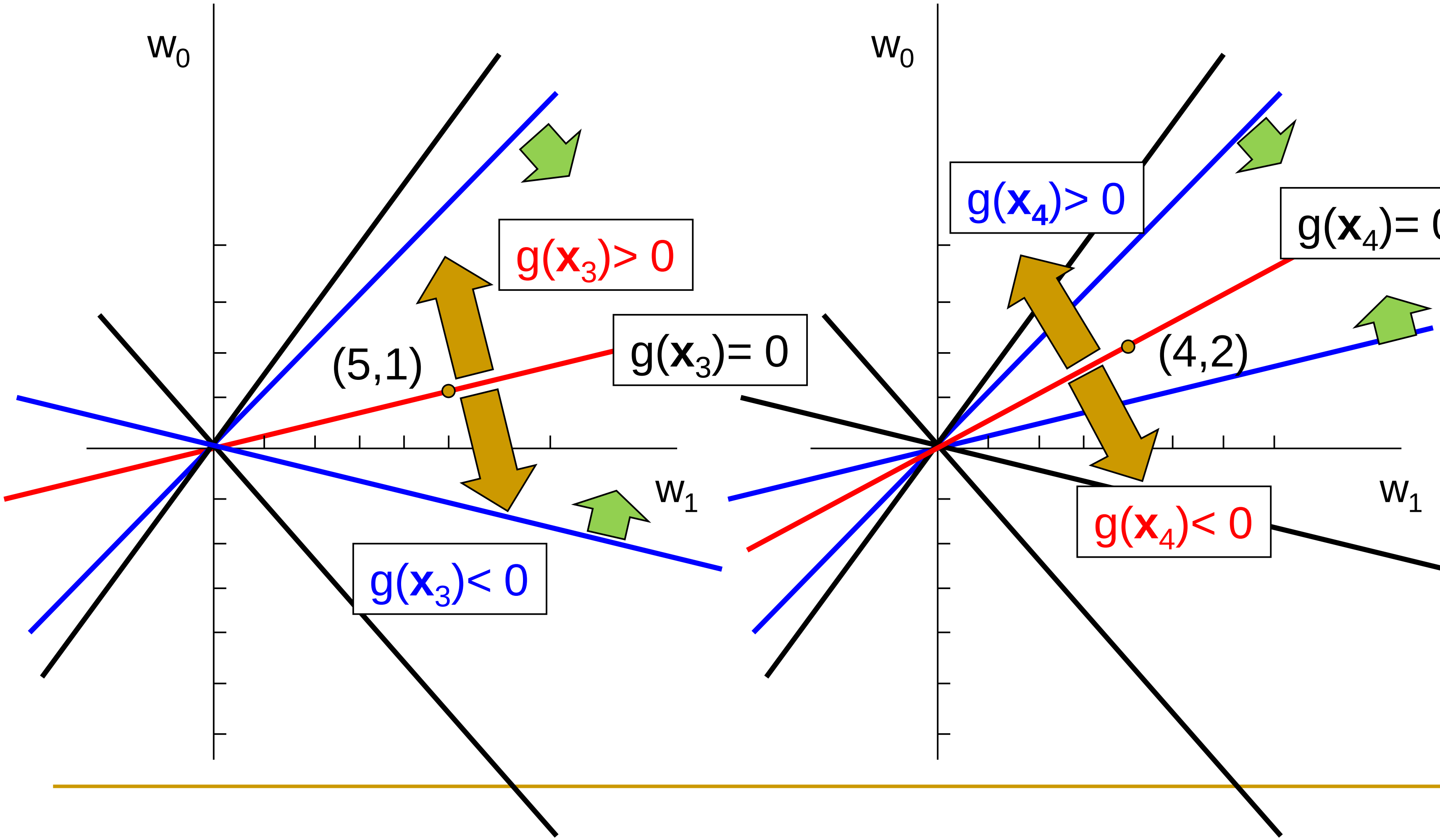
$\mathbf{x}_2 = (1,0.2)^t$ において$g(\mathbf{x}_2) > 0$

$\mathbf{x}_5 = (1,-1)^t$ において$g(\mathbf{x}_5) < 0$

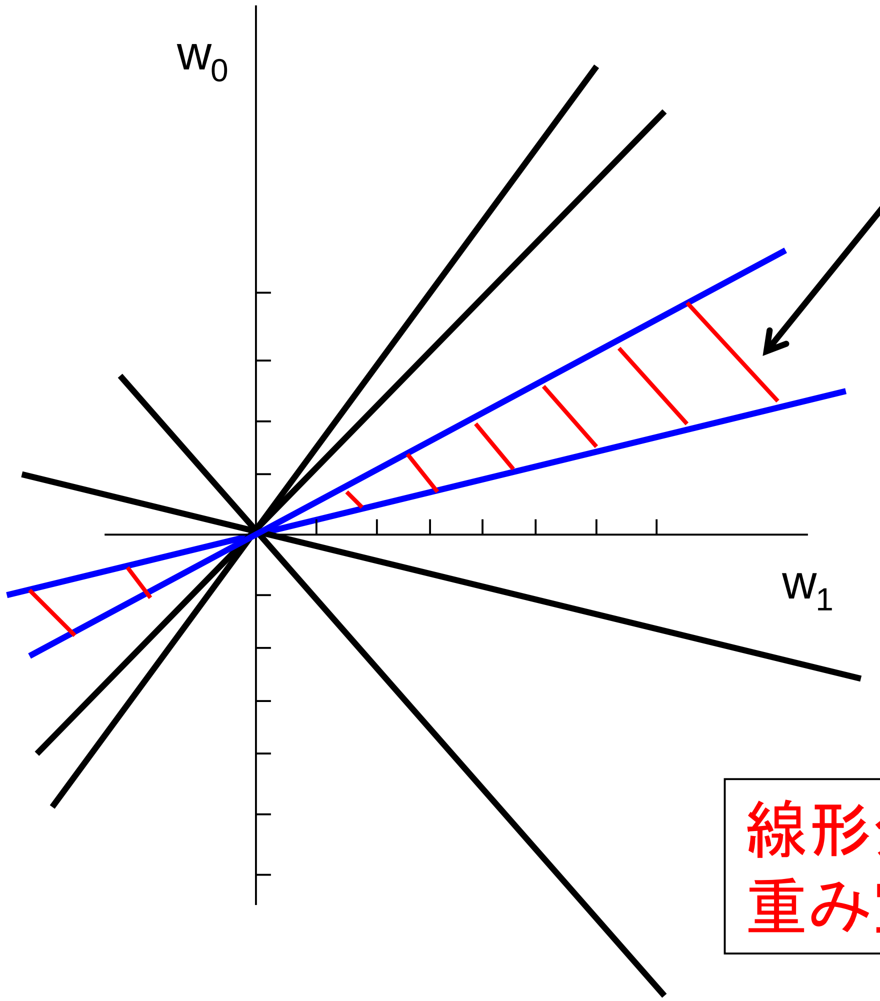


重み空間⑤

$\mathbf{x}_3 = (1,-0.2)^t$ において$g(\mathbf{x}_3) > 0$	$\mathbf{x}_4 = (1,-0.5)^t$ において$g(\mathbf{x}_4) < 0$



解領域



例）$\mathbf{w}=(2,5)^t$

$\mathbf{w}=(1,4)^t$

線形分離可能

重み空間上に解領域が存在する

線形分離とは

- クラス1
  - $g(\mathbf{x}_1)>0$, $g(\mathbf{x}_2)>0$, $g(\mathbf{x}_3)>0$
- クラス2
  - $g(\mathbf{x}_4)<0$, $g(\mathbf{x}_5)<0$, $g(\mathbf{x}_6)<0$

- 重み空間上にて，上記の条件の直線（超平面）によって，重みベクトルの存在する領域を指定できる場合，線形分離が可能

パーセプトロン

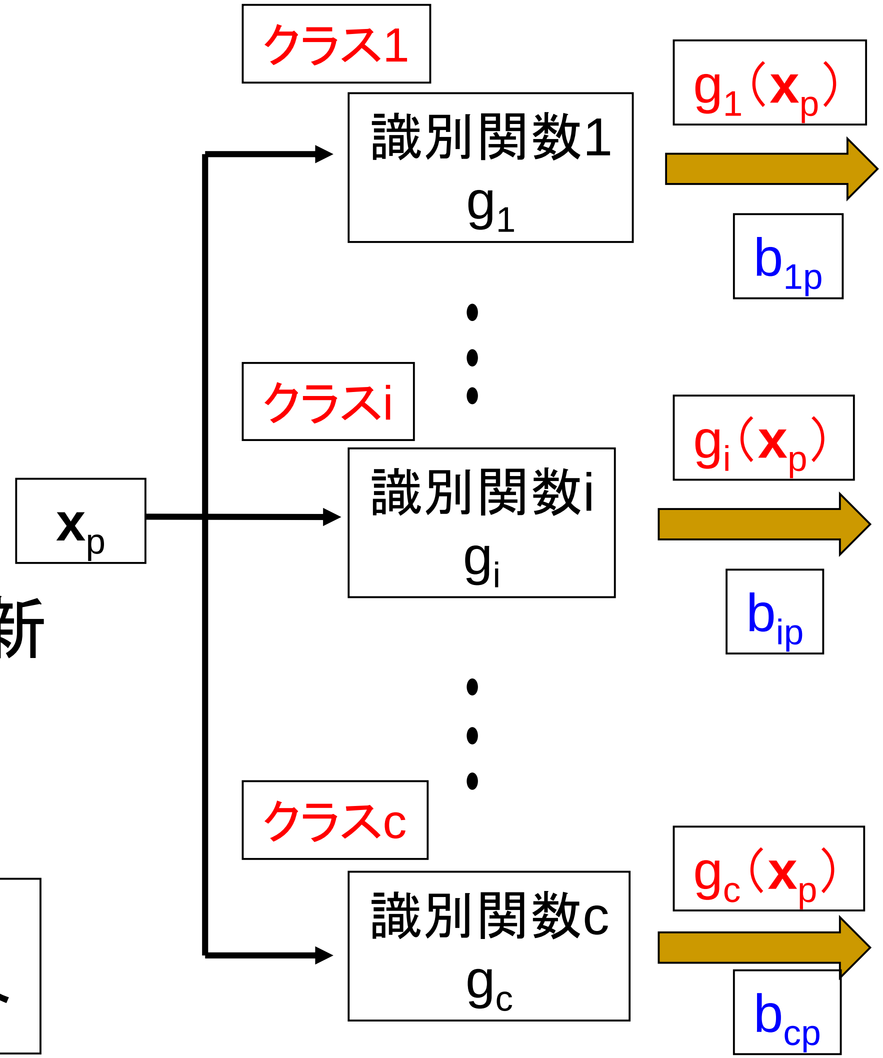
- Frank Rosenblatt, 1957
- 線形判別関数における重み係数の学習アルゴリズム
- 線形分離可能な問題であれば，有限回の繰り返しで，解領域の重みを求めることが可能（パーセプトロンの収束定理）

デルタルールからのパーセプトロンの導出

- 重みの更新方法

$$\mathbf{w}_i' = \mathbf{w}_i - \alpha(g_i(\mathbf{x}_p) - b_{ip})\mathbf{x}_p$$
$$= \mathbf{w}_i - \alpha(\mathbf{w}_i^t \mathbf{x}_p - b_{ip})\mathbf{x}_p$$

  - 全ての$\mathbf{w}_i$において更新
  - (i=1,2,・・・,c)



パーセプトロン
デルタルールの特殊な場合

閾値関数の導入①

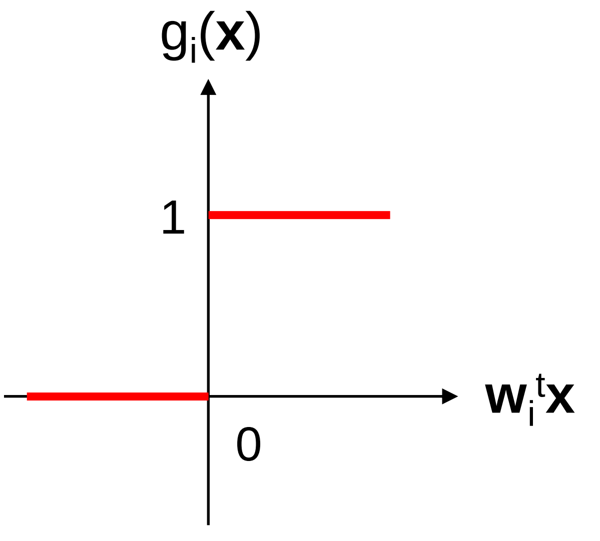
- 識別関数g_iの重みベクトル$\mathbf{w}_i$
  - 以下の式を満たすように学習する

$$\begin{cases} \mathbf{w}_i^t \mathbf{x} > 0 & (x \in \omega_i) \\ \mathbf{w}_i^t \mathbf{x} < 0 & (x \notin \omega_i) \end{cases}$$

$$i = 1,2,\cdots,c$$

- 閾値関数

$$\boxed{g_i(\mathbf{x}) = T_i(\mathbf{w}_i^t \mathbf{x}) = \begin{cases} 1 & \mathbf{w}_i^t \mathbf{x} > 0 \\ 0 & \mathbf{w}_i^t \mathbf{x} < 0 \end{cases}}$$



閾値関数の導入②

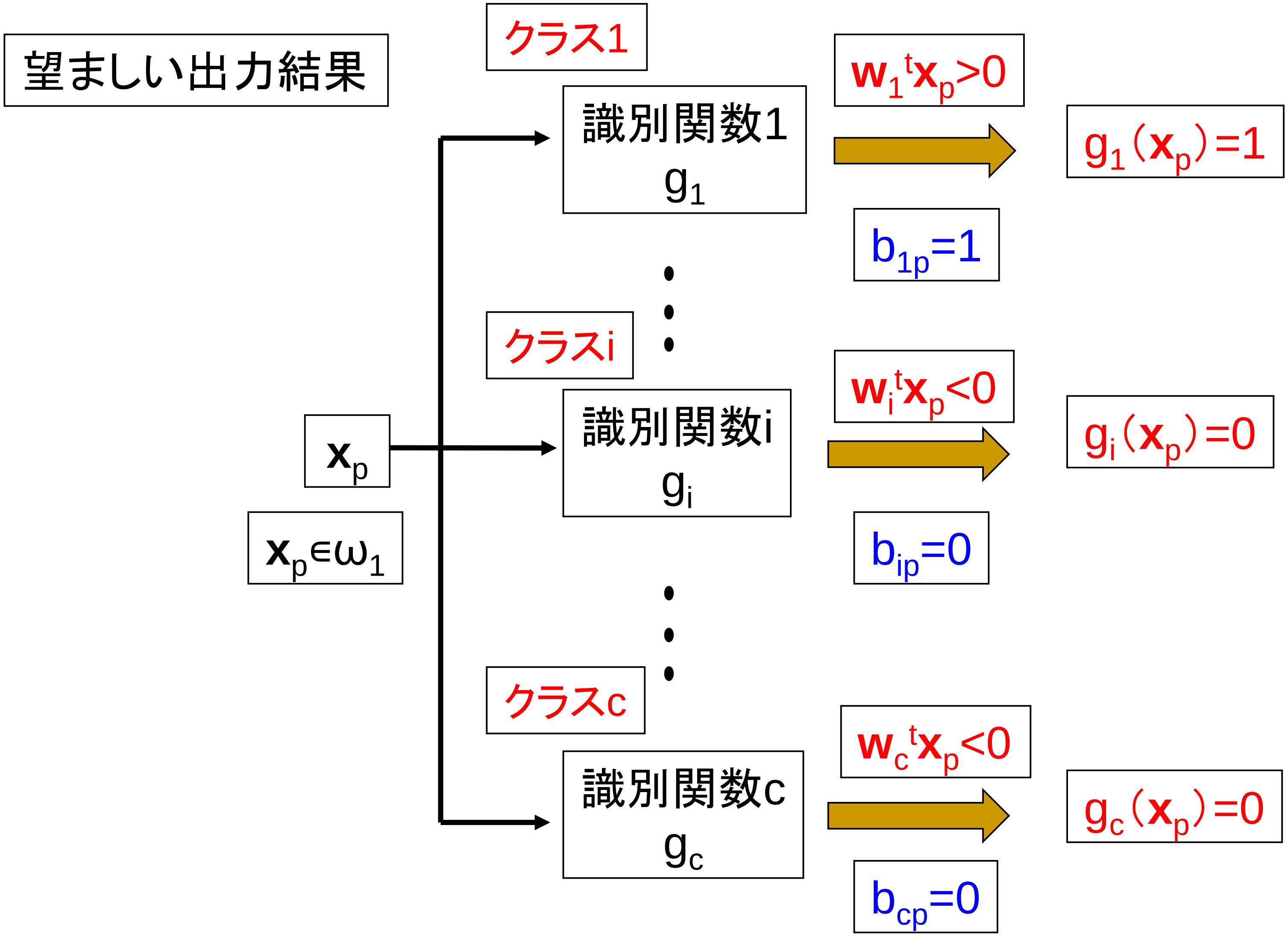
- 識別関数の出力値

$$\begin{cases} g_i(\mathbf{x}) = 1 & (x \in \omega_i) \\ g_i(\mathbf{x}) = 0 & (x \notin \omega_i) \end{cases}$$

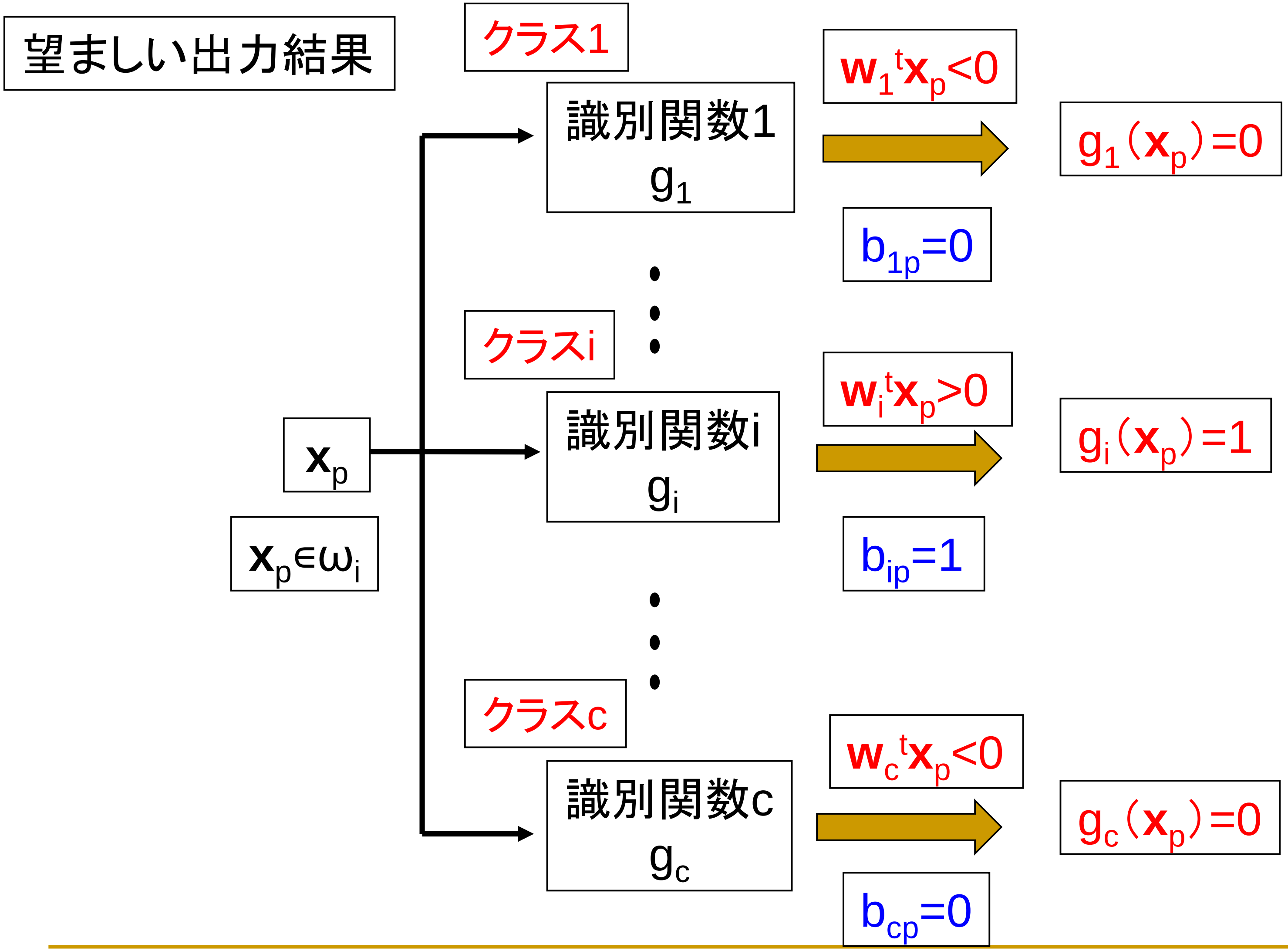
- 教師信号

$$\begin{cases} b_{ip} = 1 & (x \in \omega_i) \\ b_{ip} = 0 & (x \notin \omega_i) \end{cases}$$

閾値関数を導入した場合①



閾値関数を導入した場合②



デルタルールの変更①

- デルタルール

$$\mathbf{w}_i^{'} = \mathbf{w}_i - \alpha(g_i(\mathbf{x}_p) - b_{ip})\mathbf{x}_p$$

- 識別関数g_i
  - $\mathbf{x}_p \in \omega_i$ をω_iと正しく予測した場合

$$g_i(\mathbf{x}_p) = 1 \quad b_{ip} = 1 \quad \Rightarrow \quad \boxed{\mathbf{w}_i^{'} = \mathbf{w}_i}$$

  - $\mathbf{x}_p \in \omega_i$ をω_iではないと誤って予測した場合

$$g_i(\mathbf{x}_p) = 0 \quad b_{ip} = 1 \quad \Rightarrow \quad \boxed{\mathbf{w}_i^{'} = \mathbf{w}_i + \alpha\mathbf{x}_p}$$

デルタルールの変更②

- デルタールール

$$\boxed{\mathbf{w}_i^{'} = \mathbf{w}_i - \alpha(g_i(\mathbf{x}_p) - b_{ip})\mathbf{x}_p}$$

- 識別関数g_j(i≠j)
  - $\mathbf{x}_p \in \omega_i$ をω_jと誤って予測した場合

$$\boxed{g_j(\mathbf{x}_p) = 1 \quad b_{jp} = 0} \Rightarrow \boxed{\mathbf{w}_j^{'} = \mathbf{w}_j - \alpha \mathbf{x}_p}$$

  - $\mathbf{x}_p \in \omega_i$ をω_jではないと正しく予測した場合

$$\boxed{g_j(\mathbf{x}_p) = 0 \quad b_{jp} = 0} \Rightarrow \boxed{\mathbf{w}_j^{'} = \mathbf{w}_j}$$

重みベクトルの更新方法

- 識別関数g_iにおいて，$\mathbf{x}_p$をω_iと予測しなければならないのに，ω_iではないと予測してしまった場合

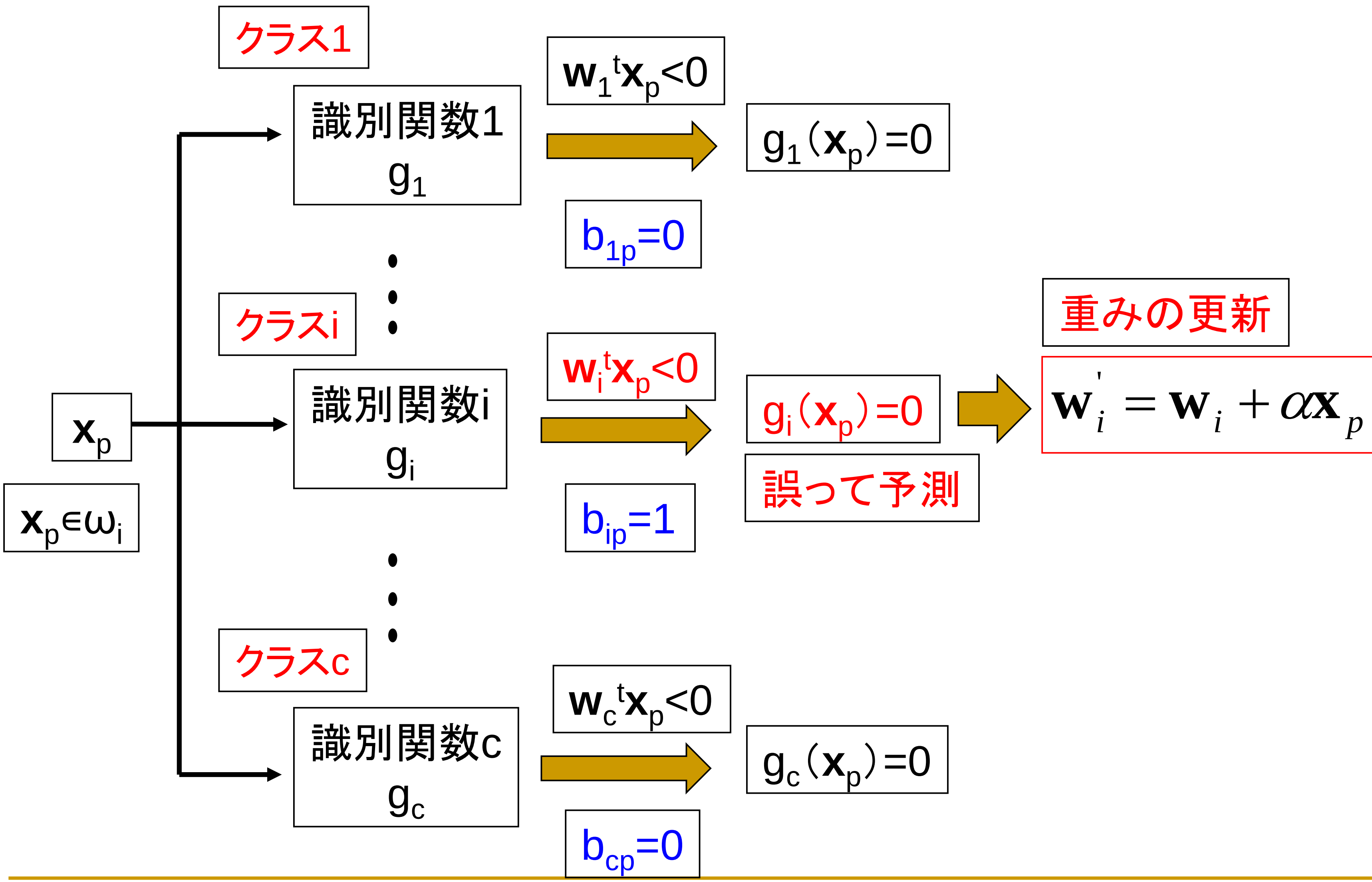
$$\boxed{g_i(\mathbf{x}_p) = 0 \quad b_{ip} = 1} \Rightarrow \boxed{\mathbf{w}_i' = \mathbf{w}_i + \alpha \mathbf{x}_p}$$

- 識別関数g_jにおいて，$\mathbf{x}_p$をω_j と予測してはいけないのに，ω_j と予測してしまった場合

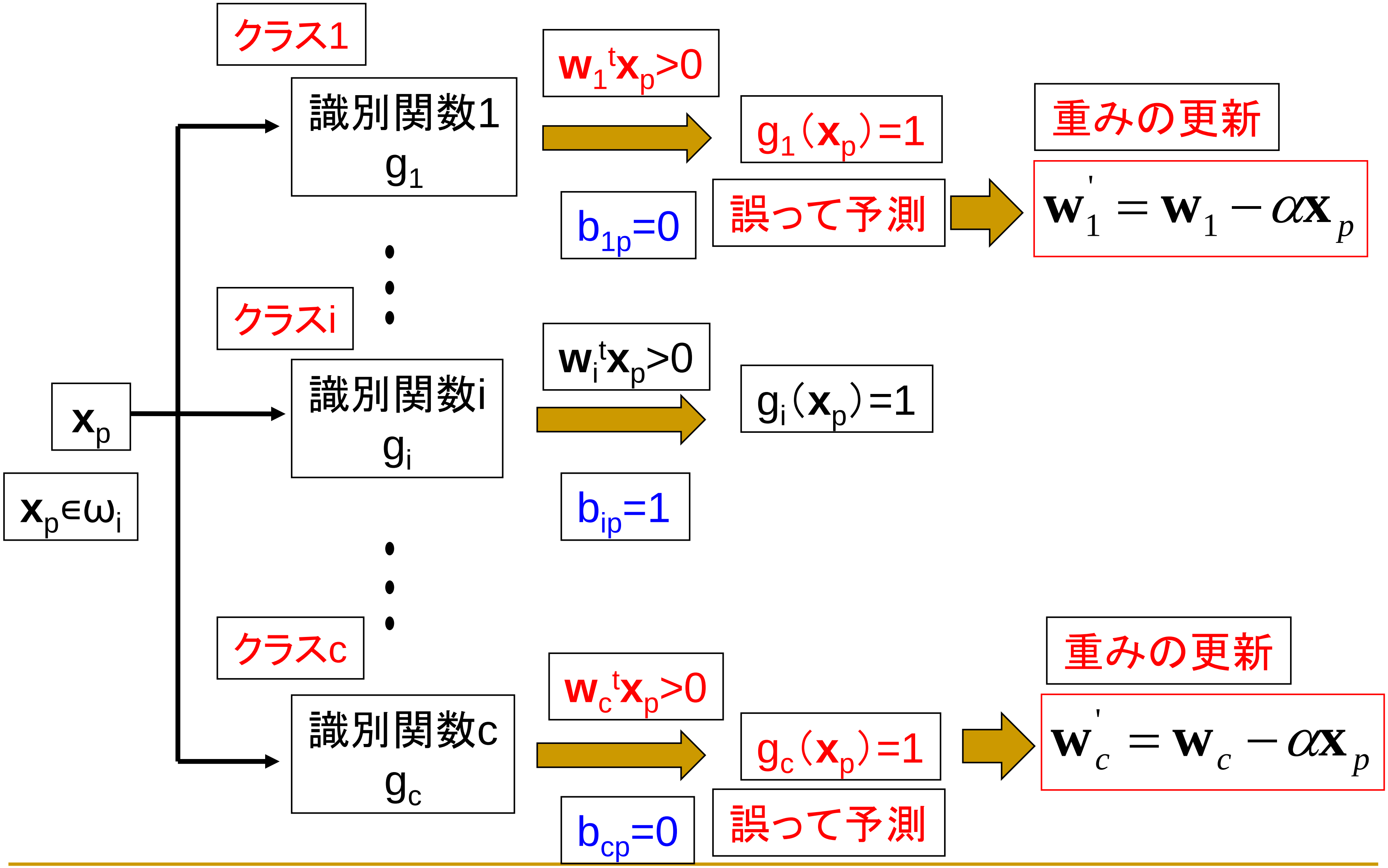
$$\boxed{g_j(\mathbf{x}_p) = 1 \quad b_{jp} = 0} \Rightarrow \boxed{\mathbf{w}_j' = \mathbf{w}_j - \alpha \mathbf{x}_p}$$

パーセプトロンの学習規則

パーセプトロンの学習規則①



パーセプトロンの学習規則②



パーセプトロンの学習規則

1. 重みベクトル$\mathbf{w}_i$を乱数にて初期化
   - クラス数はc個（i=1,2,･･･,c）
2. 学習データ$\mathbf{x}_p$ を選択
3. 全ての $g_i(\mathbf{x})$ を計算，正しく予測できなかった識別関数の重みベクトル$\mathbf{w}_i$を修正
   - $\mathbf{x}_p$をω_i と予測しなければならないのに，ω_iではないと予測してしまった場合→ $\mathbf{w}_i' = \mathbf{w}_i + \rho\mathbf{x}$
   - $\mathbf{x}_p$をω_i と予測してはいけないのに，ω_i と予測してしまった場合→ $\mathbf{w}_i' = \mathbf{w}_i - \rho\mathbf{x}$
4. 全ての学習データについて，正しく判別できるまで，2と3を繰り返す

パーセプトロンのまとめ①

- パーセプトロンの学習規則
  - 誤識別をした場合のみ，修正
  - 誤り訂正法と呼ばれる

- ニューラルネットワーク（人工的神経回路網）の代表的なモデルの一つ

- 問題点
  - 線形分離可能な問題のみ対応
  - 線形分離可能かどうかを調べることは困難
  - 学習した重みベクトルによって，未知データの識別が可能かどうか

パーセプトロンのまとめ②

- Marvin Minsky
  - 線形分離不可能な問題には対応できないことを指摘（1969）
- 誤差逆伝播則*
  - Error Back propagation Algorithm（1986）
  - David Rumelhart, Geoffrey Everest Hinton
  - パーセプトロンの学習アルゴリズムを改良
  - 線形分離不可能な問題に対応可能

*一般化デルタルールとも呼ばれる

パーセプトロンのプログラム

Breast cancer dataset

パーセプトロン（Cancer_Perceptron.py）

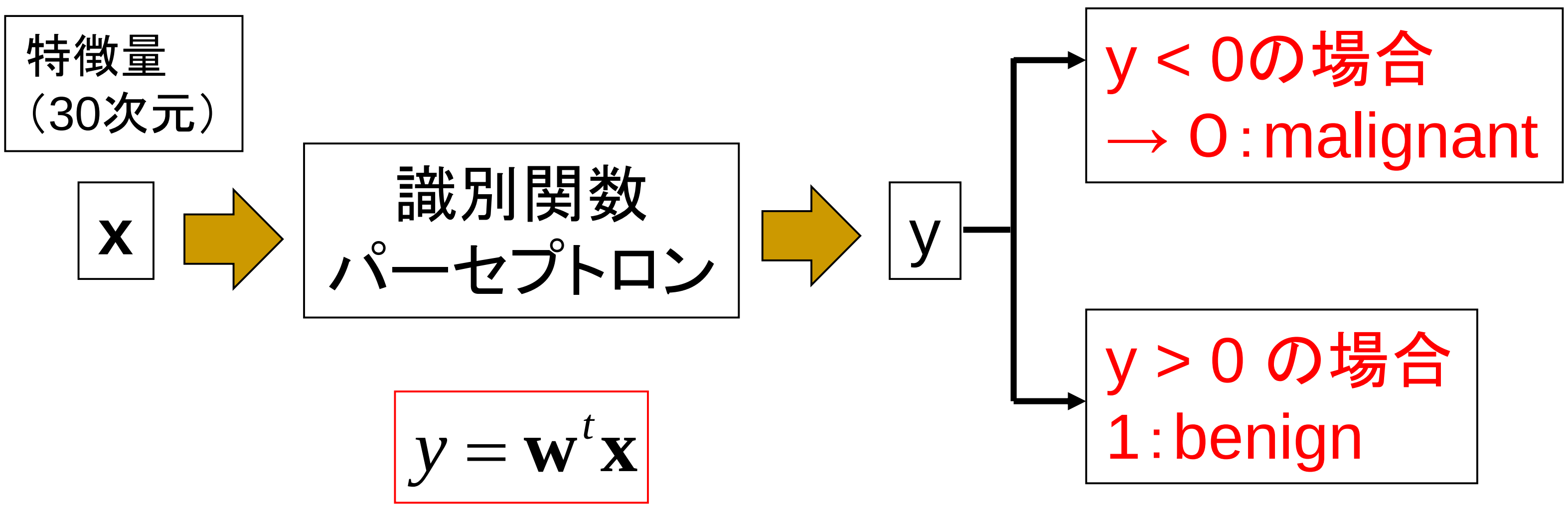
- 乳がんの分類問題
  - Breast cancer dataset

用途	クラス分類
データ数	569
特徴量	30
目的変数	2

クラス名	データ数
malignant	212
benign	357

*ITCのPCの場合，Cancer_Perceptron-ITC.pyを実行して下さい

乳がんの分類問題


特徴量
(30次元)
x
識別関数
パーセプトロン
$y = \mathbf{w}^t \mathbf{x}$
y
y < 0の場合
→ 0 : malignant
y > 0 の場合
1 : benign

Cancer_Perceptron.py

```
import numpy
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import classification_report, accuracy_score,
confusion_matrix

# データのロード
cancer = datasets.load_breast_cancer()

# 種類( malignant, benign )
name = cancer.target_names
label = cancer.target

# 特徴量
feature_names = cancer.feature_names
data = cancer.data
```

パーセプトロンのために必要

breast cancerデータセットの読み込み

label：正解ラベル
malignant → 0
benign → 1

個数
569

data
特徴量

大きさ
（569,30）

ホールドアウト法

```
# 学習データ, テストデータ
train_data, test_data, train_label, test_label = train_test_split(data, label,
test_size=0.5, random_state=None)

model = Perceptron(eta0=0.1, max_iter=1000, early_stopping=True,
validation_fraction=0.1, n_iter_no_change=5)

# 学習
model.fit(train_data, train_label)

# 予測
predict = model.predict(test_data)
df = model.decision_function(test_data)
```

eta0：学習係数
defaultは0.1

max_iter：繰り返し回数
defaultは1000

early_stopping
defaultはFalse

validation_fraction
検証用データの割合

n_iter_no_change
early_stoppingまで，改善がない回数

predict
ラベルの予測

decision_function
識別関数の値の計算

```
print( "¥n [ 重みベクトル ]" )
print( model.coef_ )

print( "¥n [ 切片 ]" )
print( model.intercept_ )

# 予測値，正解ラベルの表示
print( "¥n 予測値，正解ラベル" )
for i in range(len(test_data)):
    print( "{0:12.3f}({1}) :{2}".format( df[i] , predict[i], test_label[i] ) )

print( "¥n [ 予測結果 ]" )
print( classification_report(test_label, predict) )

print( "¥n [ 正解率 ]" )
print( accuracy_score(test_label, predict) )

print( "¥n [ 混同行列 ]" )
print( confusion_matrix(test_label, predict) )
```

coef_
重みベクトル

intercept_
切片

識別関数の値

予測ラベル

正解ラベル

accuracy
precision
recall
F値

accuracyの表示

混同行列の表示

```
from sklearn.linear_model import Perceptron
Perceptron(eta0=0.1, max_iter=1000, early_stopping=True,
validation_fraction=0.1, n_iter_no_change=5)
```

eta0：学習係数
defaultは0.1

max_iter：繰り返し回数
defaultは1000

early_stopping
defaultはFalse

validation_fraction
検証用データの割合

n_iter_no_change
early_stoppingまで，改善がない回数

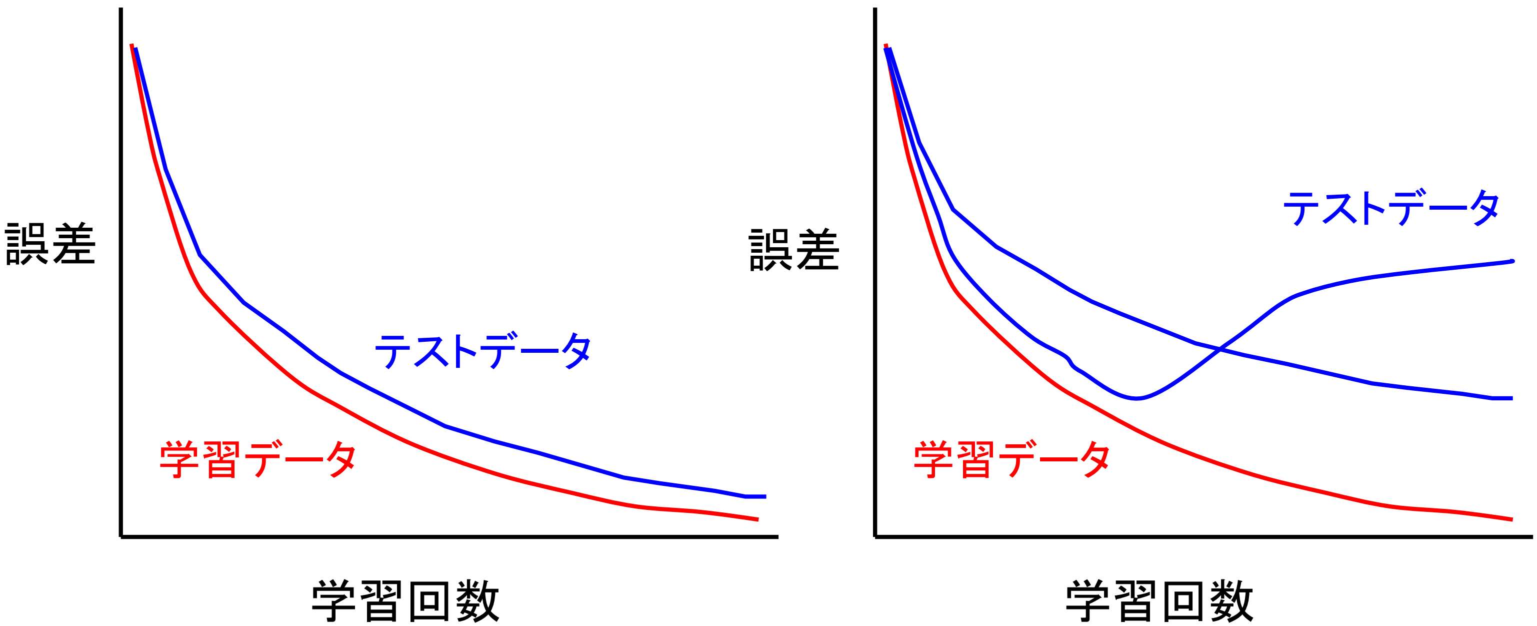
```
model = Perceptron()
```

パラメータをdefaultのままで学習する場合

```
model = Perceptron(eta0=0.1, max_iter=1000, early_stopping=True,
validation_fraction=0.1, n_iter_no_change=5)
```

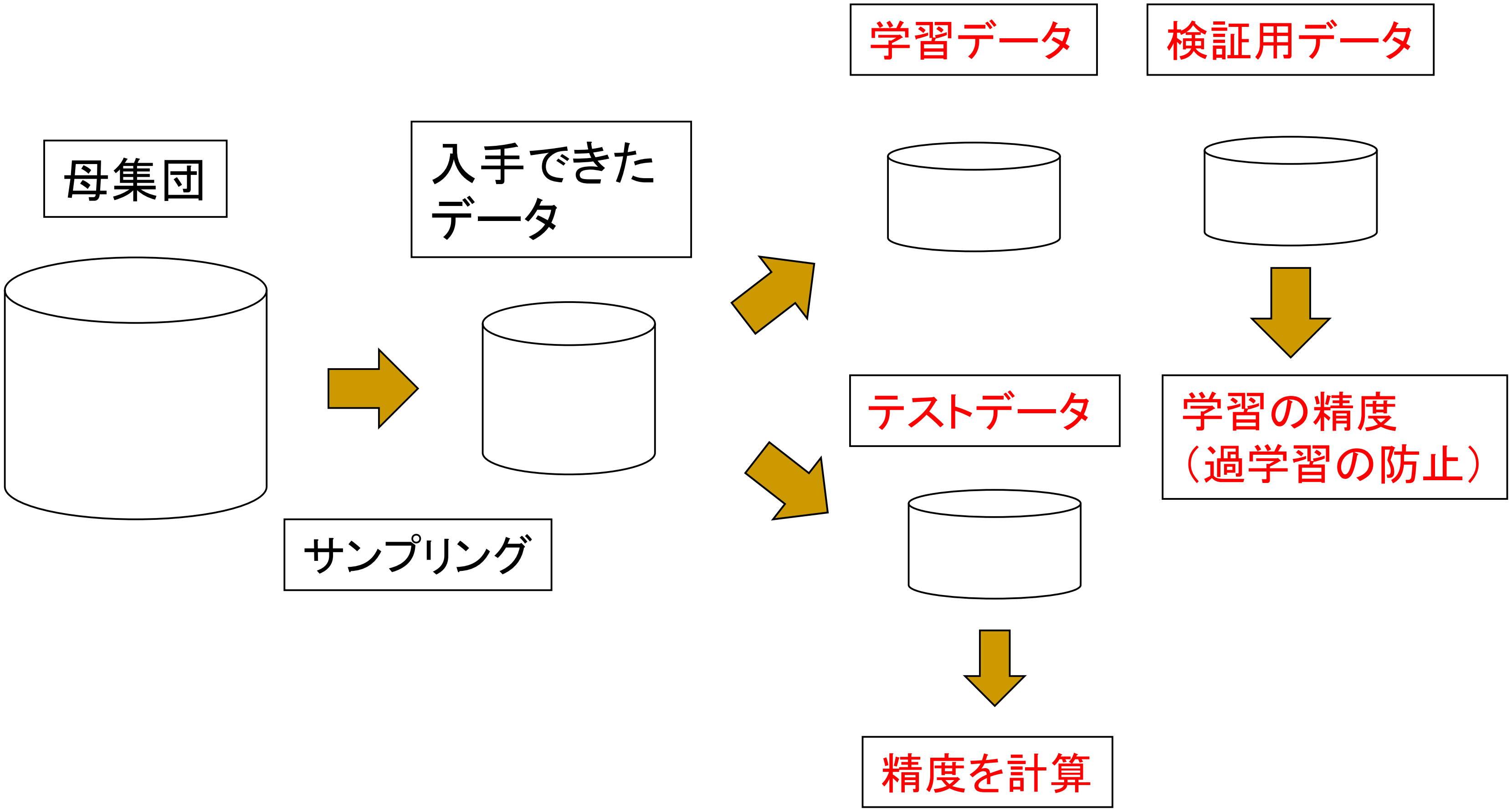
過学習(Overtraining)

- 機械学習の最大の問題
- 学習データにのみ適用したモデルが学習され，テストデータに適用できない(汎化性の欠如)

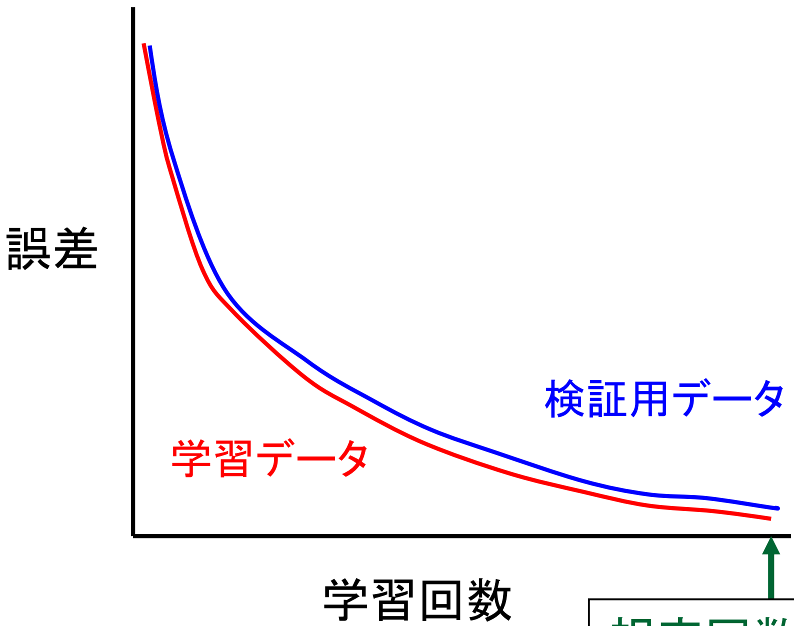


過学習の防止

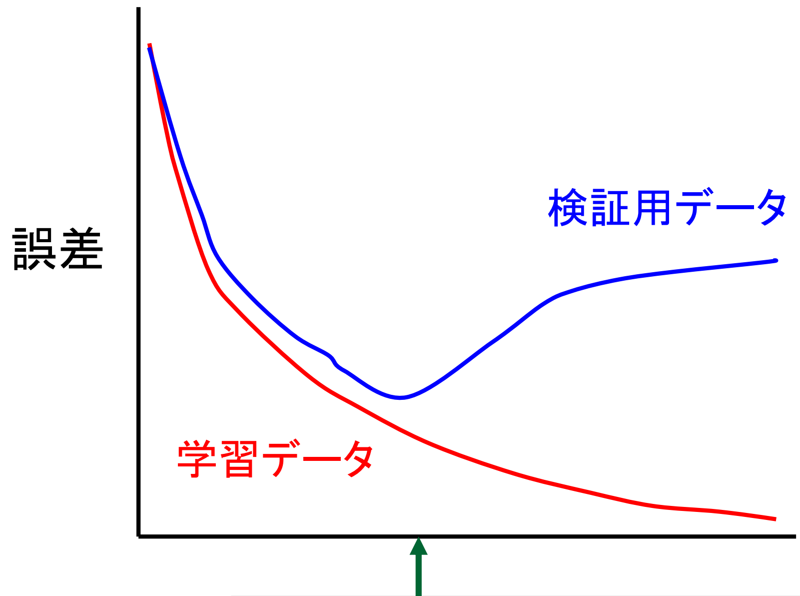
- 検証用データ(Validation)の利用



早期終了（Early Stopping）

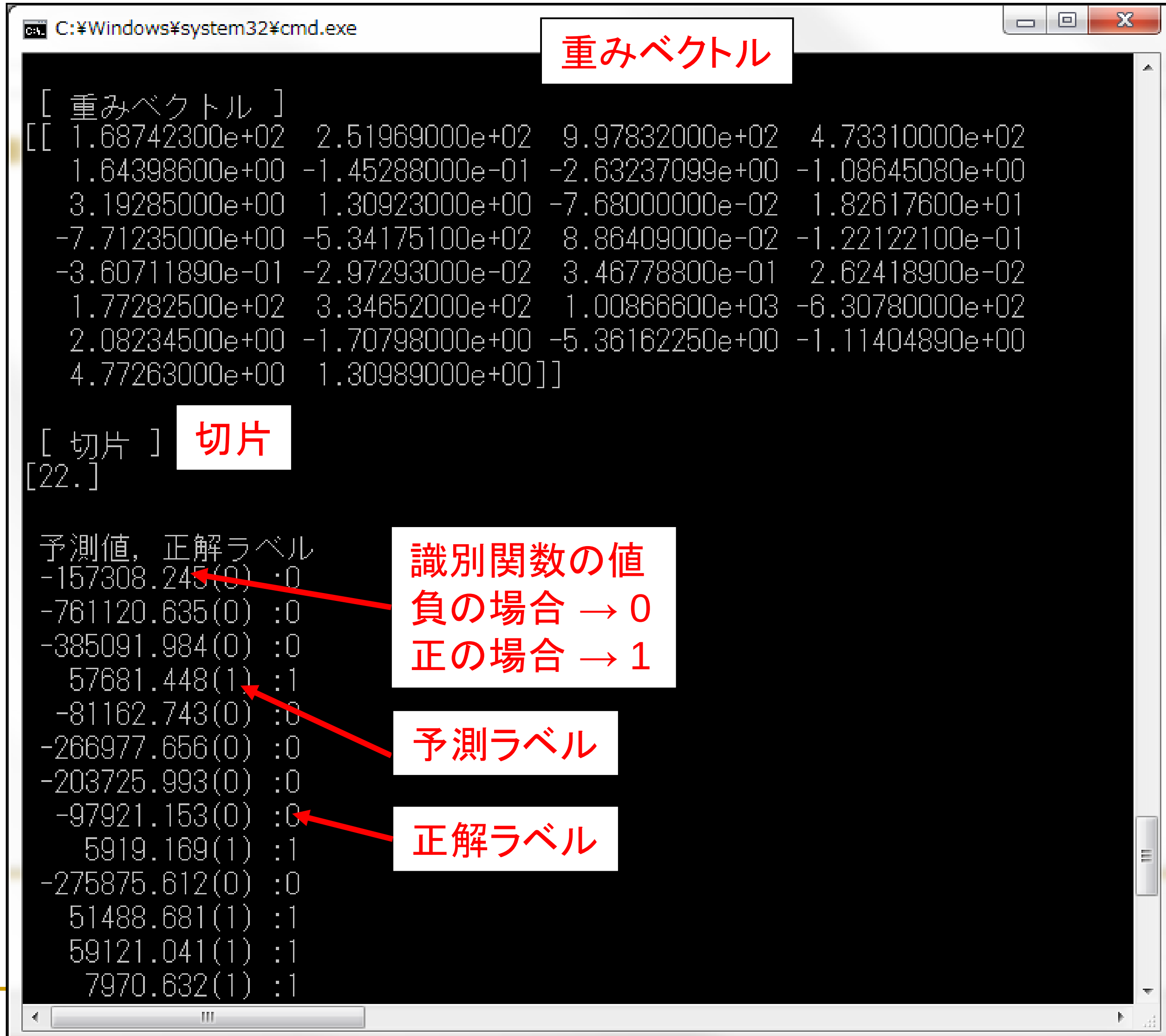


規定回数で停止



規定回数繰り返していないが，検証用データの誤差が大きくなって来たので，早期に終了

実行結果①


C:¥Windows¥system32¥cmd.exe
重みベクトル
[重みベクトル]
[[1.68742300e+02 2.51969000e+02 9.97832000e+02 4.73310000e+02
1.64398600e+00 -1.45288000e-01 -2.63237099e+00 -1.08645080e+00
3.19285000e+00 1.30923000e+00 -7.68000000e-02 1.82617600e+01
-7.71235000e+00 -5.34175100e+02 8.86409000e-02 -1.22122100e-01
-3.60711890e-01 -2.97293000e-02 3.46778800e-01 2.62418900e-02
1.77282500e+02 3.34652000e+02 1.00866600e+03 -6.30780000e+02
2.08234500e+00 -1.70798000e+00 -5.36162250e+00 -1.11404890e+00
4.77263000e+00 1.30989000e+00]]
[切片]
切片
[22.]
予測値, 正解ラベル
-157308.245(0) :0
-761120.635(0) :0
-385091.984(0) :0
57681.448(1) :1
-81162.743(0) :0
-266977.656(0) :0
-203725.993(0) :0
-97921.153(0) :0
5919.169(1) :1
-275875.612(0) :0
51488.681(1) :1
59121.041(1) :1
7970.632(1) :1
識別関数の値
負の場合 → 0
正の場合 → 1
予測ラベル
正解ラベル

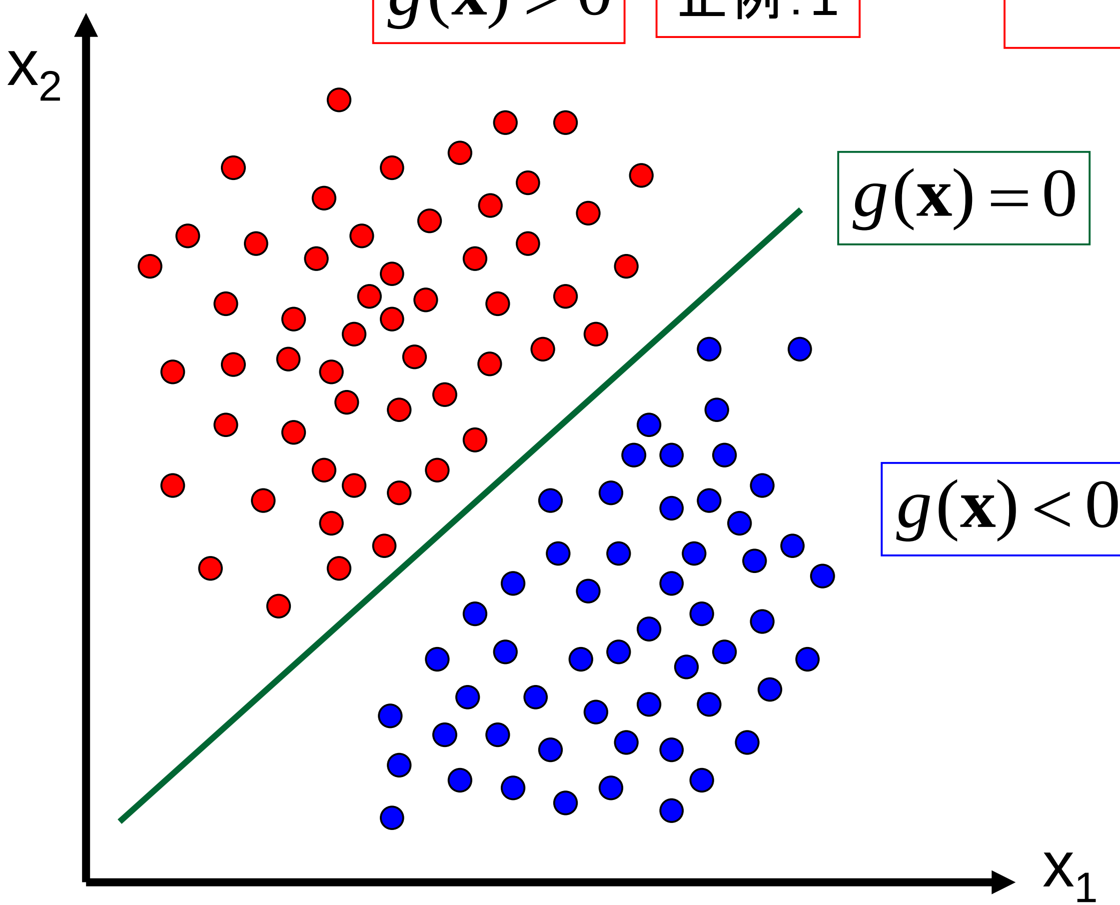
パーセプトロンによる二値分類

$$f(\mathbf{x}) = \begin{cases} 1 & g(\mathbf{x}) > 0 \\ 0 & g(\mathbf{x}) < 0 \end{cases}$$

$g(\mathbf{x}) > 0$　正例:1

線形識別関数

$$g(\mathbf{x}) = \mathbf{w}^t \mathbf{x} + b$$



負例:0

実行結果②

C:¥Windows¥system32¥cmd.exe

```
[ 予測結果 ]
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       109
           1       0.93      0.92      0.93       176

    accuracy                           0.91       285
   macro avg       0.90      0.91      0.90       285
weighted avg       0.91      0.91      0.91       285


[ 正解率 ]
0.9087719298245615

[ 混同行列 ]
[[ 97  12]
 [ 14 162]]
```

accuracy
precision
recall
F値

accuracyの表示

混同行列の表示

実行結果について

- 何度か実行して下さい
- 評価指標にばらつきがあります
  - なぜでしょうか

デルタルールのプログラム

Iris dataset

デルタルールのプログラム（Iris_SGD.py）

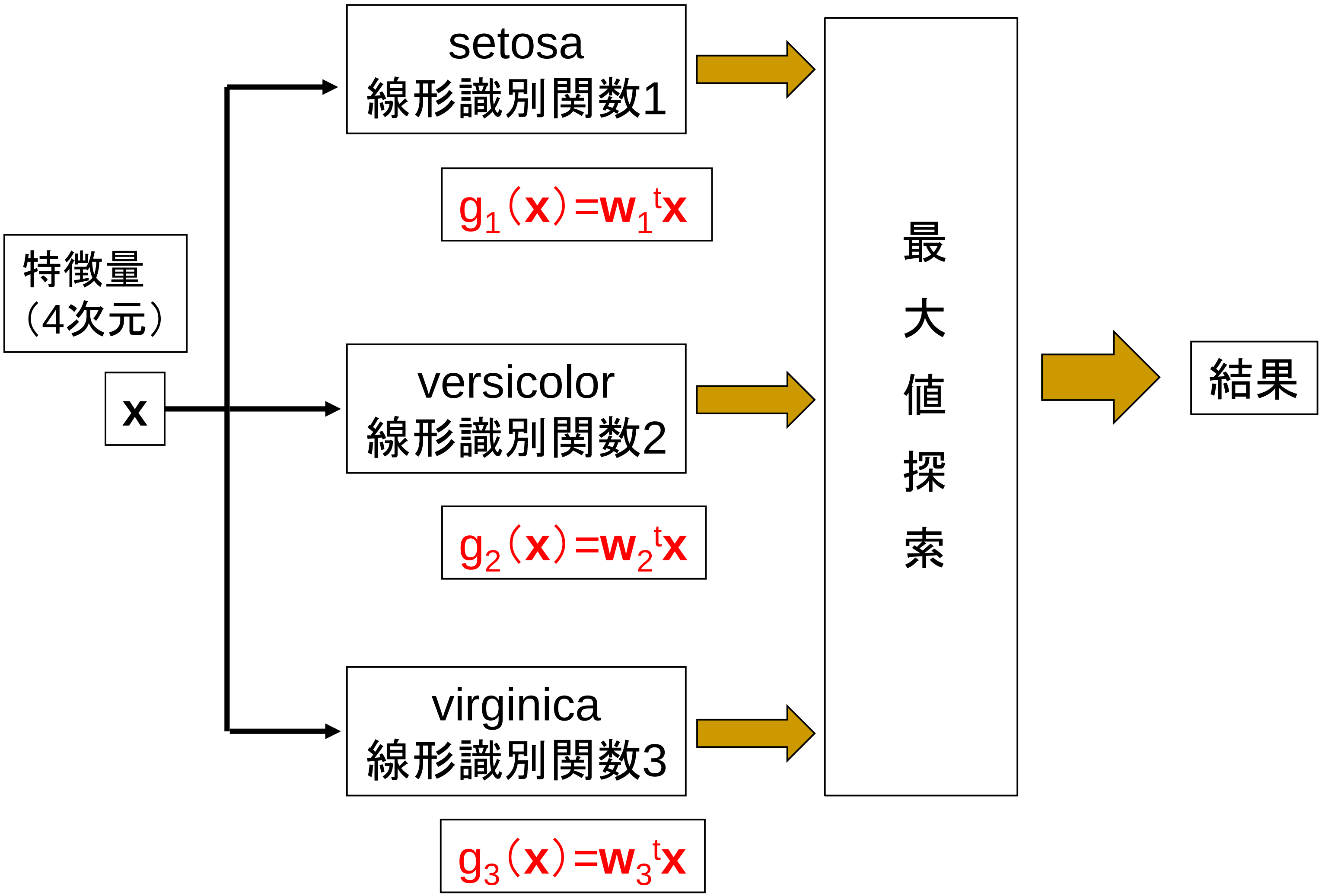
- Irisデータセット
  - アヤメの分類問題

用途	クラス分類
データ数	150
特徴量	4
目的変数	3

クラス名	データ数
setosa	50
versicolor	50
virginica	50

*ITCのPCの場合，Iris_SGD-ITC.pyを実行して下さい

irisデータセットの分類問題



Iris_SGD.py

```
import numpy
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix

# データのロード
iris = datasets.load_iris()

# 種類
name = iris.target_names
label = iris.target

# 特徴量
feature_names = iris.feature_names
data = iris.data
```

SGDのために必要

標準化のために必要

irisデータセットの読み込み

label
目的変数の値（0,1,2）

個数
150

data
特徴量

大きさ
（150,4）

```
# 学習データ，テストデータ
train_data, test_data, train_label, test_label = train_test_split(data, label,
test_size=0.5, random_state=None)

# 学習データの平均値，標準偏差の計算
sc = StandardScaler()
sc.fit( train_data )

print( "¥n [ 平均値 ]" )
print( sc.mean_ )
print( "¥n [ 分散 ]" )
print( sc.var_ )
print( "¥n [ 標準偏差 ]" )
print( sc.scale_ )

# 標準化
train_data_std = sc.transform(train_data)
test_data_std = sc.transform(test_data)
```

平均値，分散，標準偏差の計算

mean_
平均値

var_
分散

scale_
標準偏差

transform
標準化

$$x' = \frac{(x-a)}{s}$$

x : データ
a : 平均値
s : 標準偏差

loss：損失関数
hinge：ヒンジ関数

eta0：学習係数
defaultは0.1

max_iter：繰り返し回数
defaultは1000

learning_rate='optimal'
学習係数を小さくしていく

```
model = SGDClassifier(eta0=0.1, max_iter=1000, learning_rate='optimal' ,
loss='hinge' , early_stopping=True, validation_fraction=0.1,
n_iter_no_change=5, penalty='l2')
```

validation_fraction
検証用データの割合

n_iter_no_change
early_stoppingまで，改善がない回数

early_stopping
defaultはFalse

penalty
L2正則化

```
# 学習
model.fit(train_data_std, train_label)

# 予測
predict = model.predict(test_data_std)
df = model.decision_function(test_data_std)

print( "¥n [ 重みベクトル ]" )
print( model.coef_ )
print( "¥n [ 切片 ]" )
print( model.intercept_ )
```

predict
ラベルの予測

decision_function
識別関数の値の計算

coef_
重みベクトル

intercept_
切片

```
# 予測値，正解ラベル
print( "¥n 予測値，正解ラベル" )
for i in range(len(test_data_std)):
    for j in range(len(name)):
        print( "{0:12.3f}".format( df[i][j] ) , end=" " )
    print( "({0}) : {1}".format( predict[i], test_label[i] ) )

print( "¥n [ 予測結果 ]" )
print( classification_report(test_label, predict) )

print( "¥n [ 正解率 ]" )
print( accuracy_score(test_label, predict) )

print( "¥n [ 混同行列 ]" )
print( confusion_matrix(test_label, predict) )
```

識別関数の値

予測ラベル

正解ラベル

accuracy
precision
recall
F値

accuracyの表示

混同行列の表示

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit( data )
```

平均値

```
sc.mean_
```

分散

```
sc.var_
```

標準偏差

```
sc.scale_
```

標準化

```
sc.transform( data )
```

$$x' = \frac{(x-a)}{s}$$

x：データ
a：平均値
s：標準偏差

eta0：学習係数
defaultは0.1

max_iter：繰り返し回数
defaultは1000

learning_rate='optimal'
学習係数を一定ではなく
小さくしていく

```
from sklearn.linear_model import SGDClassifier
SGDClassifier(eta0=0.1, max_iter=1000, learning_rate='optimal' ,
loss='hinge' , early_stopping=True, validation_fraction=0.1,
n_iter_no_change=5, penalty='l2')
```

loss：損失関数
hinge：ヒンジ関数

early_stopping
defaultはFalse

penalty
L2正則化

validation_fraction
検証用データの割合

n_iter_no_change
early_stoppingまで，改善がない回数

```
model = SGDClassifier()
```

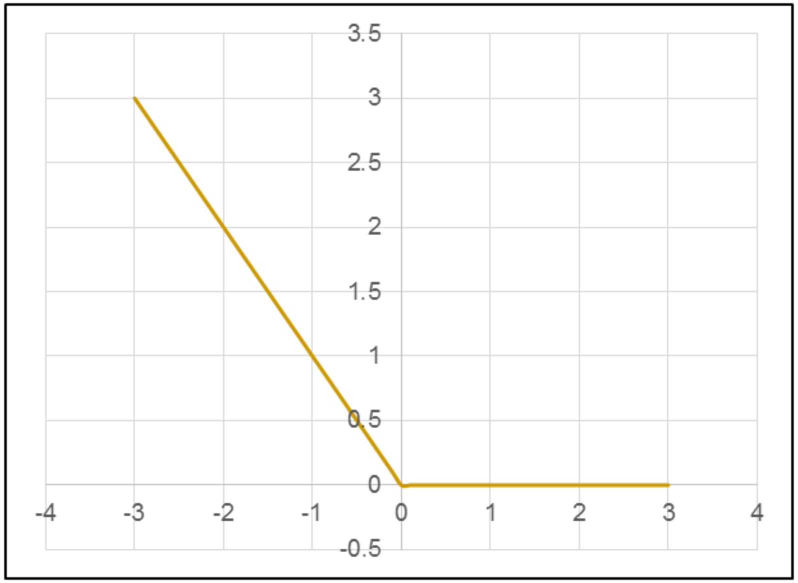
パラメータをdefaultのままで学習する場合

ヒンジ損失

$$L(f(w,x),t) = \max(a - tf(w,x), 0) = \max(a - t(\mathbf{w}^t\mathbf{x}), 0)$$

a=0の場合

$$L(f(w,x),t) = \max(-t(\mathbf{w}^t\mathbf{x}), 0)$$



教師信号

識別関数

正解の場合

t	$\mathbf{w}^t\mathbf{x}$	$t(\mathbf{w}^t\mathbf{x})$	ヒンジ損失
正	正	正	0
負	負	正	0
正	負	負	正
負	正	負	正

不正解の場合

正則化

誤差関数（ヒンジ損失）

$$L(f(w,x),t) = \max(-t(\mathbf{w}^t\mathbf{x}),0)$$

$$L(f(w,x),t) = \max(-t(\mathbf{w}^t\mathbf{x}),0) + \boxed{C\sum_{j=1}^{N} w_j^2}$$

正則化項

C : 正則化パラメータ

L1ノルム $\|\mathbf{w}\|_1 = \sum_{i=1}^{N} |w_i|$

L2ノルム $\|\mathbf{w}\|_2 = \sqrt{\sum_{i=1}^{N} |w_i|^2}$

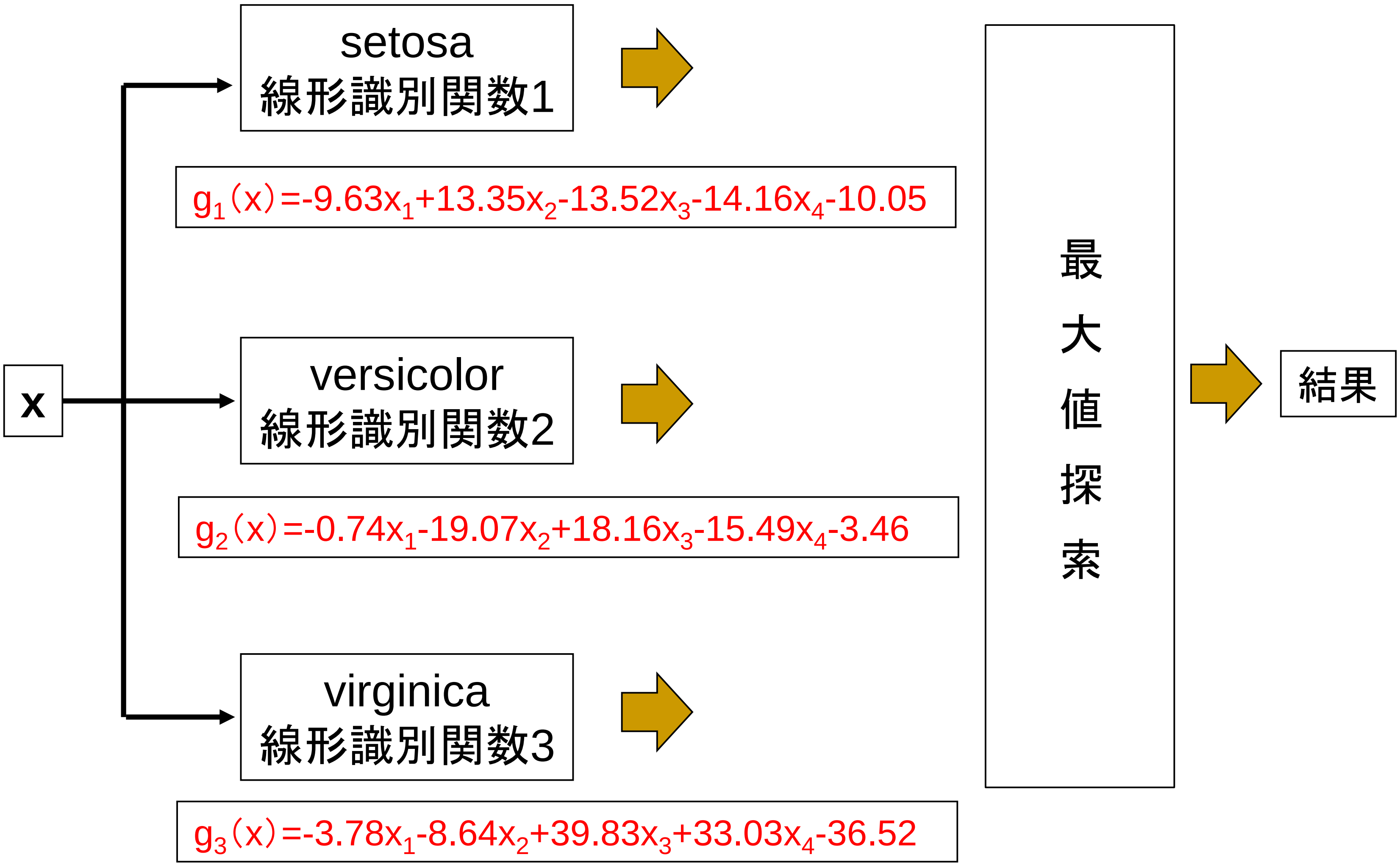
SGDClassifierで，正則化パラメータCはalpha

*L2ノルム正則化はweight decay（重み減衰）とも呼ばれます

実行結果


C:¥Windows¥system32¥cmd.exe
[平均値]
[5.91333333 3.00533333 4.03733333 1.31866667]
[分散]
[0.64782222 0.14210489 2.61113956 0.48418489]
[標準偏差]
[0.80487404 0.37696802 1.61590209 0.69583395]
[重みベクトル]
[[-9.63674327 13.35524205 -13.5284037 -14.16950703]
 [-0.74441607 -19.07306108 18.16873545 -15.49924534]
 [-3.78032873 -8.64973137 39.83456435 33.03689405]]
[切片]
[-10.05858603 -3.46654817 -36.5230574]
データ(4次元)の平均値
分散
標準偏差
識別関数1の重み
識別関数2の重み
識別関数3の重み
識別関数1の切片
識別関数2の切片
識別関数3の切片

学習後の線形識別関数



x
setosa
線形識別関数1
$g_1(x)=-9.63x_1+13.35x_2-13.52x_3-14.16x_4-10.05$
versicolor
線形識別関数2
$g_2(x)=-0.74x_1-19.07x_2+18.16x_3-15.49x_4-3.46$
virginica
線形識別関数3
$g_3(x)=-3.78x_1-8.64x_2+39.83x_3+33.03x_4-36.52$
最大値探索
結果

実行結果②

識別関数1の値
識別関数2の値
識別関数3の値
予測ラベル
正解ラベル

```
予測値，正解ラベル
   -18.607      25.718     -37.224 (1) : 1
   -27.472      -5.423       4.846 (2) : 2
    49.450     -10.931    -149.597 (0) : 0
   -49.356     -15.051      47.355 (2) : 2
   -51.226     -15.500      47.472 (2) : 2
   -36.952       4.511       8.496 (2) : 2
   -74.891      20.032      57.784 (2) : 2
   -20.333      19.370     -33.245 (1) : 1
   -35.491      -1.594       1.741 (2) : 1
   -15.142      -5.394     -23.719 (1) : 1
    70.038     -35.125    -168.283 (0) : 0
   -45.721     -37.866      50.724 (2) : 2
   -41.304       8.354      25.453 (2) : 2
    66.804     -37.445    -164.005 (0) : 0
   -40.254      11.086      24.812 (2) : 2
    62.062     -35.738    -159.428 (0) : 0
```

識別関数の3個の値の中，最大値が予測結果

実行結果③

C:¥Windows¥system32¥cmd.exe

```
 [ 予測結果 ]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        32
           1       1.00      0.90      0.95        21
           2       0.92      1.00      0.96        22

    accuracy                           0.97        75
   macro avg       0.97      0.97      0.97        75
weighted avg       0.98      0.97      0.97        75


 [ 正解率 ]
0.9733333333333334

 [ 混同行列 ]
[[32  0  0]
 [ 0 19  2]
 [ 0  0 22]]
```

accuracy
precision
recall
F値

accuracyの表示

混同行列の表示

練習問題

- 10/7の練習問題①の学習データ（train-1.csv）を用いて，線形識別関数を学習して下さい．
  - デルタルール（SGD）でかまいません
- 学習後の線形識別関数を用いて，テストデータ（test-1.csv）を予測して下さい．

*宿題とします．12/16（月）13時までにkeio.jpにプログラム（python）とワープロに実行画面を貼り付けて提出して下さい．

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)
- 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)
- Richard O. Duda他:パターン識別, アドコム・メディア(2009)

参考文献

- Perceptron
- https://scikit-learn.org/stable/modules/generated/
sklearn.linear_model.Perceptron.html#sklearn.linear_model.Perceptron

- SGDClassifier
- https://scikit-learn.org/stable/modules/generated/
sklearn.linear_model.SGDClassifier.html