

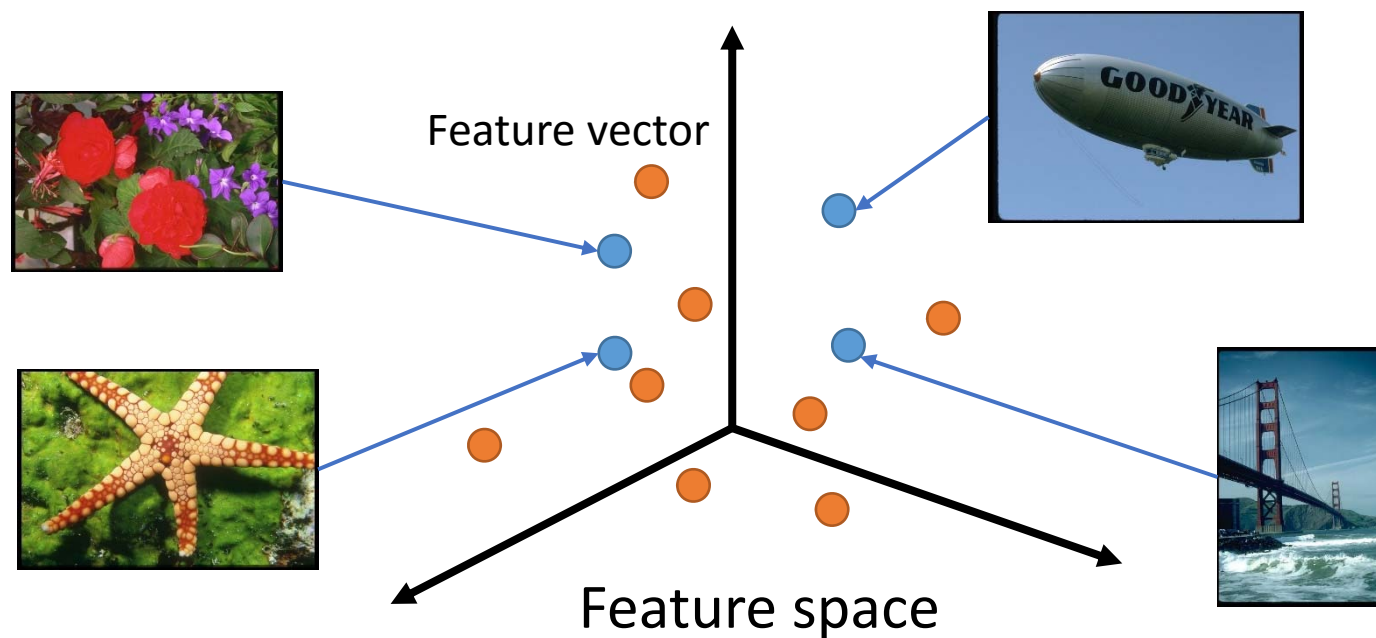
知能情報論 識別器

2016年6月8日

東京大学 大学院情報理工学系研究科

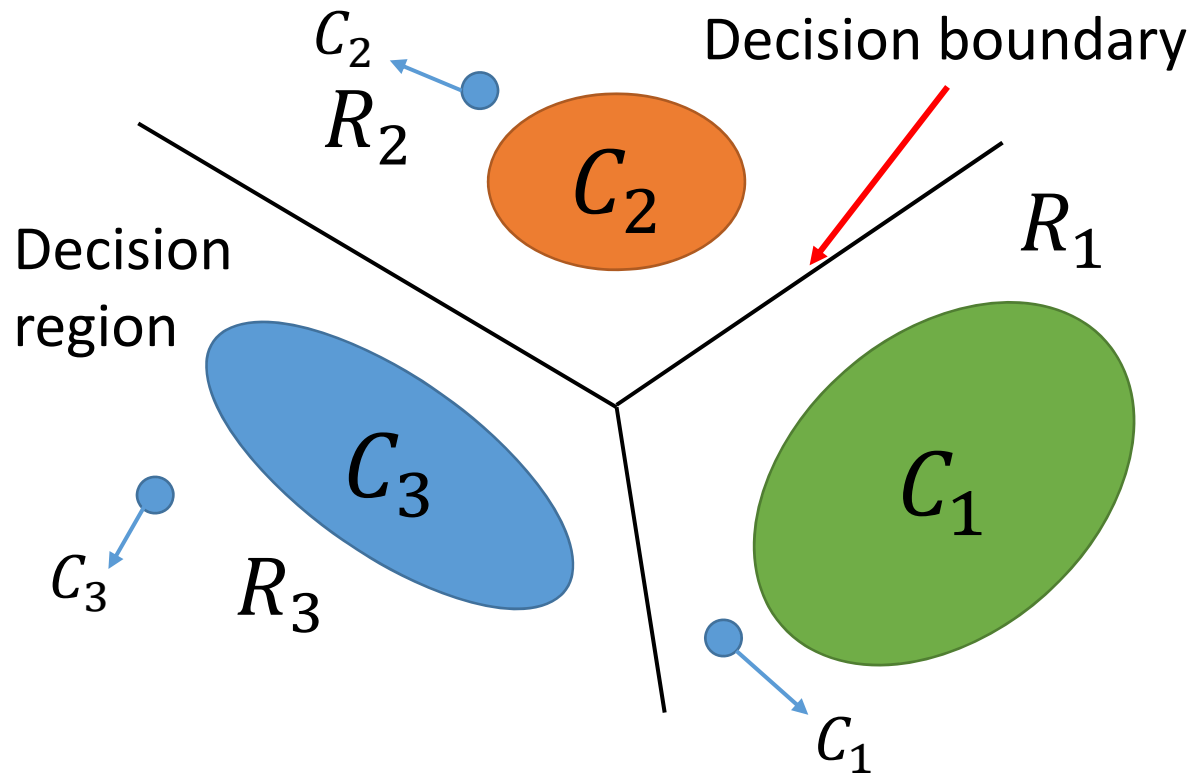
原田達也

特徴空間



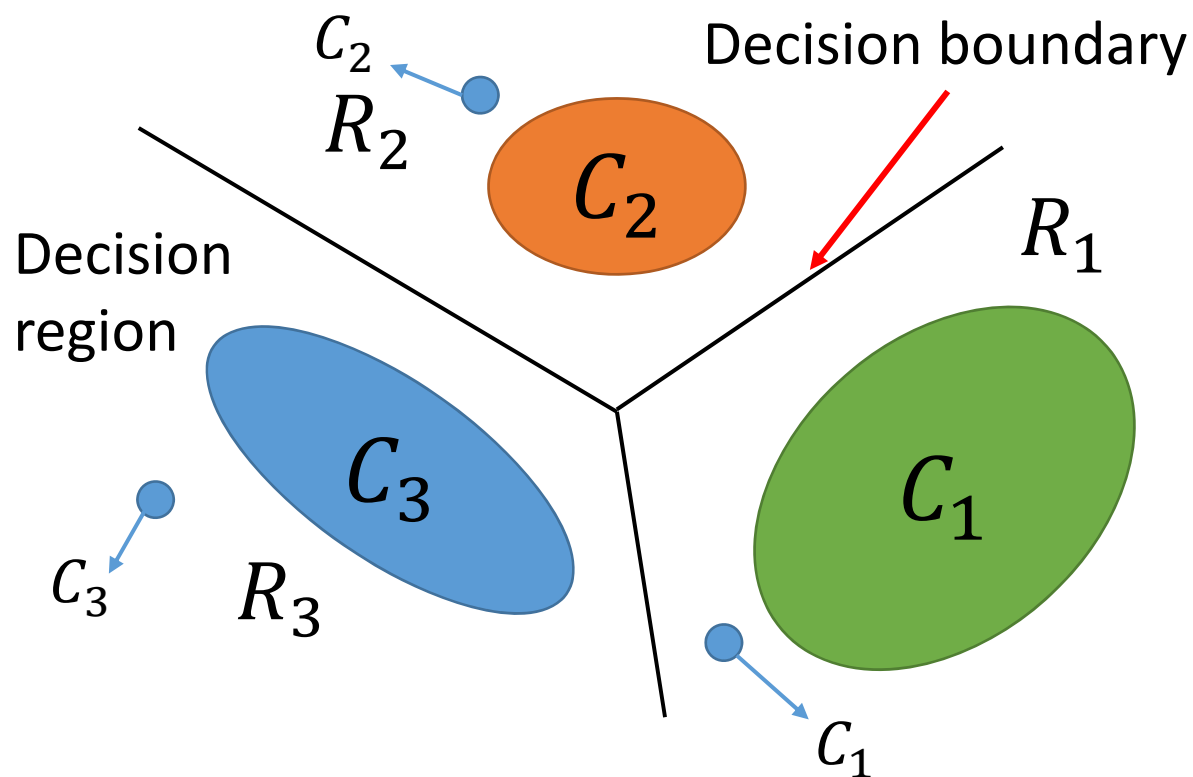
- 特徴ベクトルによって張られる空間を特徴空間 (feature space) という.
- 1つのパターンは d 次元の特徴空間の1点として表現される.

決定境界・決定領域



- 決定境界 (decision boundary)
 - クラス間を分離する境界
- 決定領域 (decision region)
 - クラスラベルの付与された分割された領域

識別器の設計

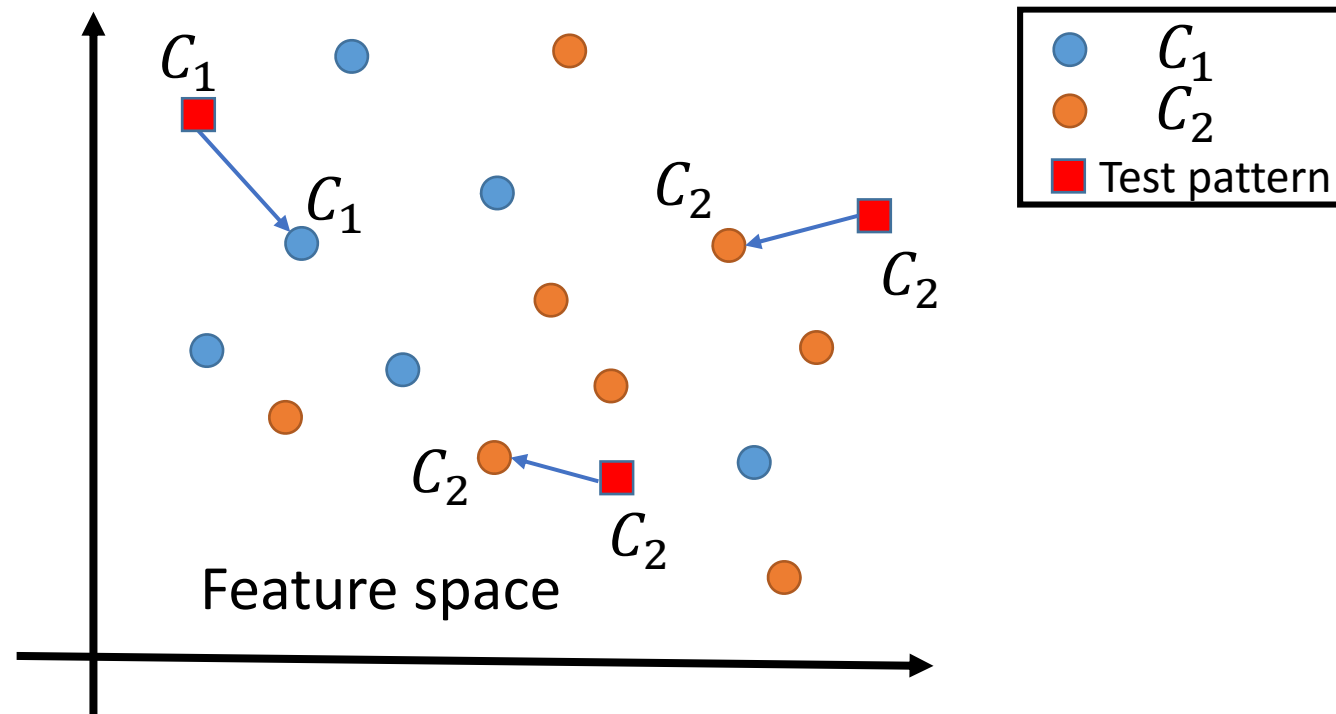


- 特徴空間の適切な領域分割の決定と領域に対するカテゴリ付与

学習データ，テストデータ，汎化

- 学習データ（training data）
 - 対応するクラスが与えられている少数のパターン集合から識別器を構成することを考える．このパターン集合を学習データ集合（training data）という．
 - 学習データから識別器を求める過程を学習（learning）という．
- テストデータ（test data）
 - 予測対象となるクラスが分かっていないパターン集合をテストデータ（test data）という．
- 汎化（generalization）
 - 学習データに含まれないパターンのクラスを予測する能力を汎化（generalization）という．

識別器の例：最近傍法



- 最近傍法 (nearest neighbor method)
 - 入力パターンと学習パターンとの距離を計算し、最も近いパターンが属するカテゴリを入力パターンのカテゴリと判断する手法.

最近傍法のアルゴリズム

$$\chi = \{(\mathbf{x}_i, t_i)\}_{i=1}^N$$

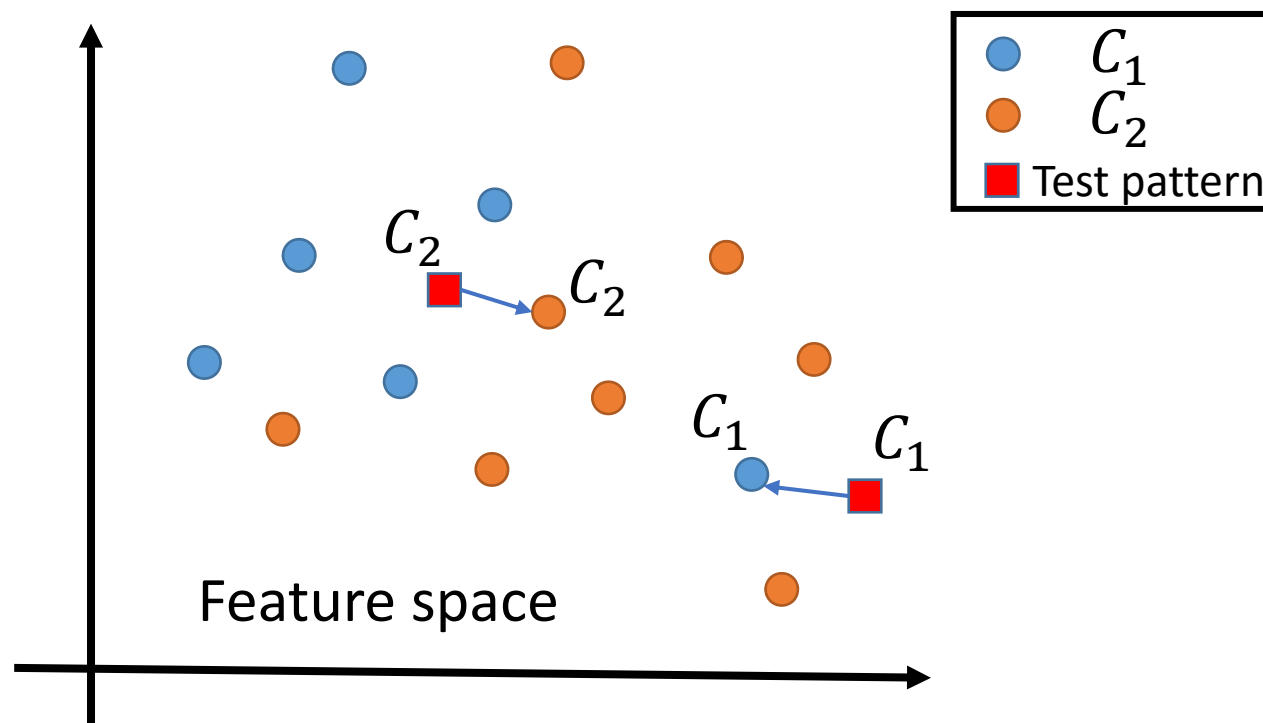
$$\mathbf{x}_j = \arg \min_{\mathbf{x}_i \in \chi} \underline{d(\mathbf{z}, \mathbf{x}_i)} \Rightarrow \mathbf{z} \in t_j$$

距離関数

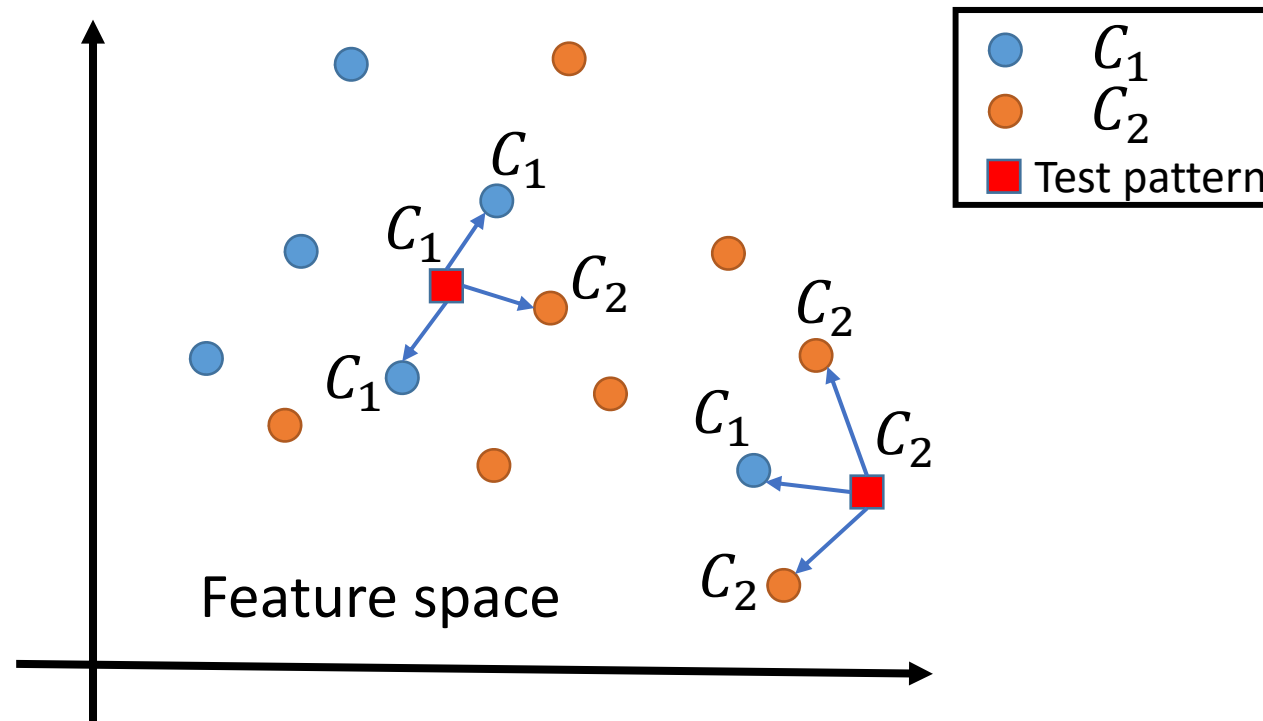
- 距離

- ユークリッド距離 : $d(\mathbf{p}, \mathbf{q}) = \sqrt{(\mathbf{q} - \mathbf{p})^T (\mathbf{q} - \mathbf{p})}$
- マハラノビス距離 : $d(\mathbf{p}, \mathbf{q}) = \sqrt{(\mathbf{q} - \mathbf{p})^T \Sigma^{-1} (\mathbf{q} - \mathbf{p})}$
- マンハッタン距離 : $d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^d |q_i - p_i|$
- など

最近傍法の問題点



K近傍法



- k近傍法 (k nearest neighbor method)
 - 入力パターンに近いk個の学習パターンを取り上げて、その中で最も多数を占めたカテゴリを入力パターンのカテゴリとする手法.

K近傍法のアルゴリズム

1. 入力パターンと全ての学習パターンとの距離を計算する.
2. 距離の昇順に学習パターンをソートする.
3. ソートした学習パターンの上位k個を取り上げ、最も出現回数の多いカテゴリを出力する.

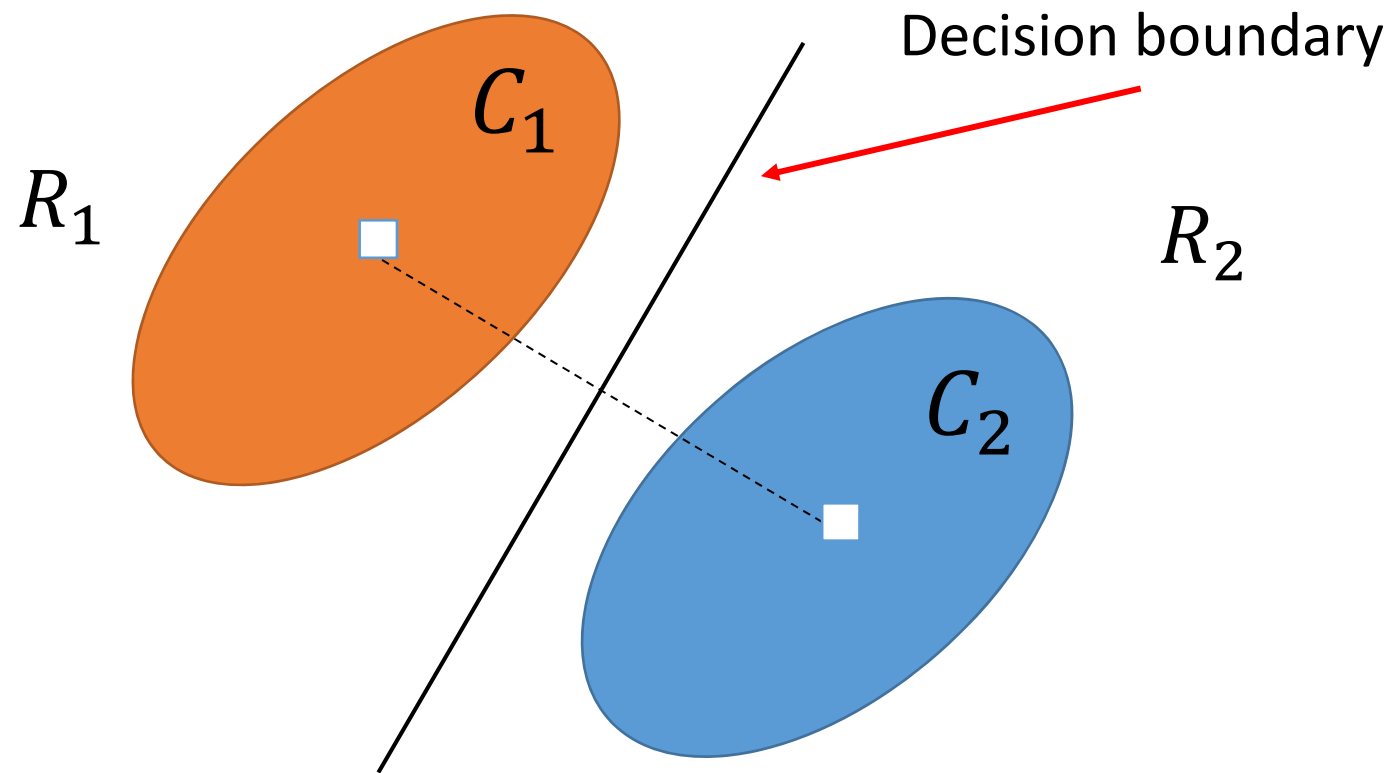
- 長所

- 単純であり実装が容易.
- 学習パターンが少なくても安定して動作する.

- 短所

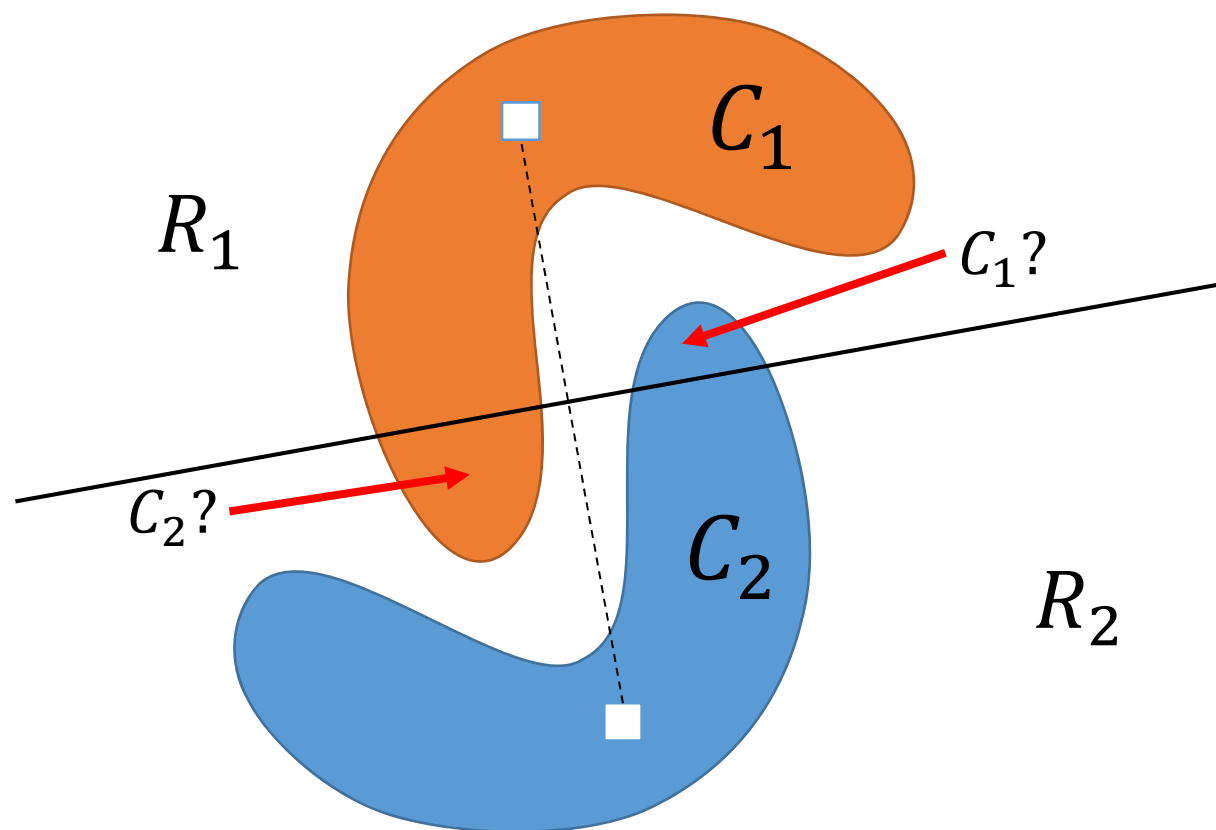
- 距離関数により結果が異なる.
- 学習データが増えると計算コストが増大する.

最近傍法と決定境界



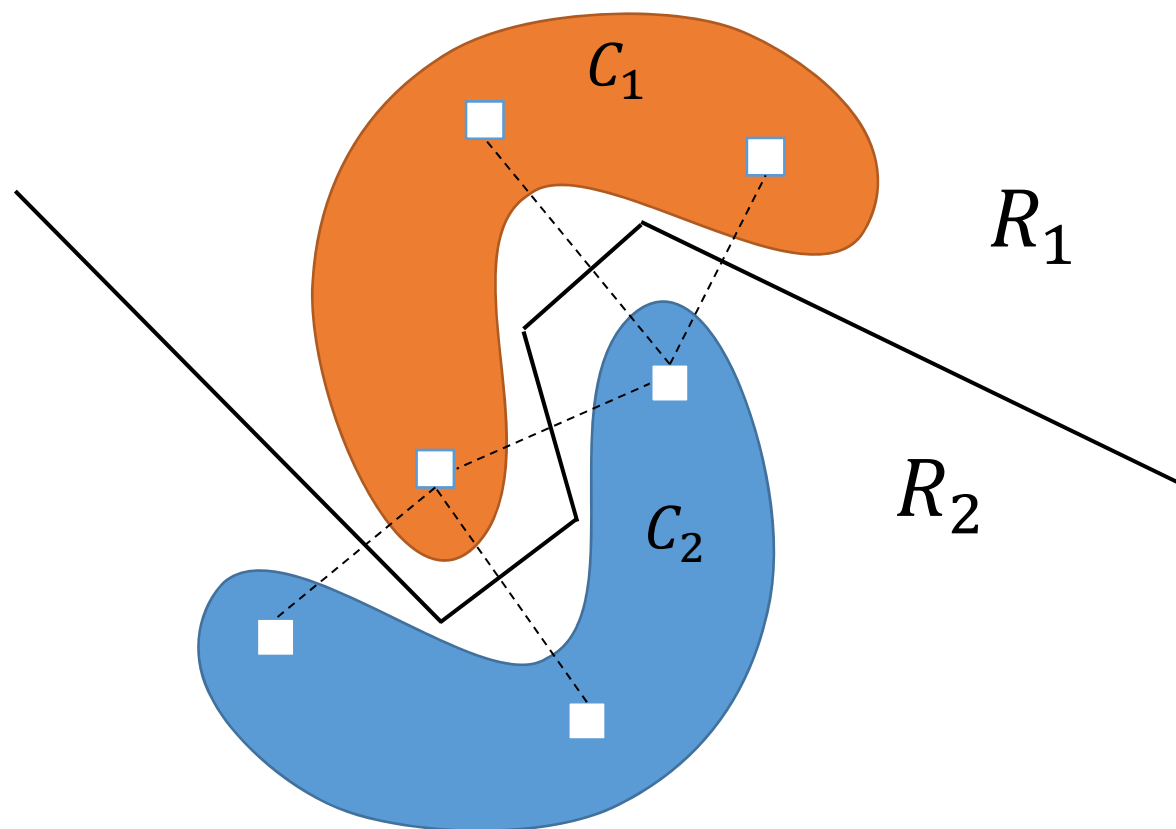
- 簡単のため，各クラス一つの代表点のみを考える．
- クラスあたり1つの代表点で最近傍法を適用すると，決定境界は代表点を結ぶ線の垂直二等分線となる

最近傍法と決定境界



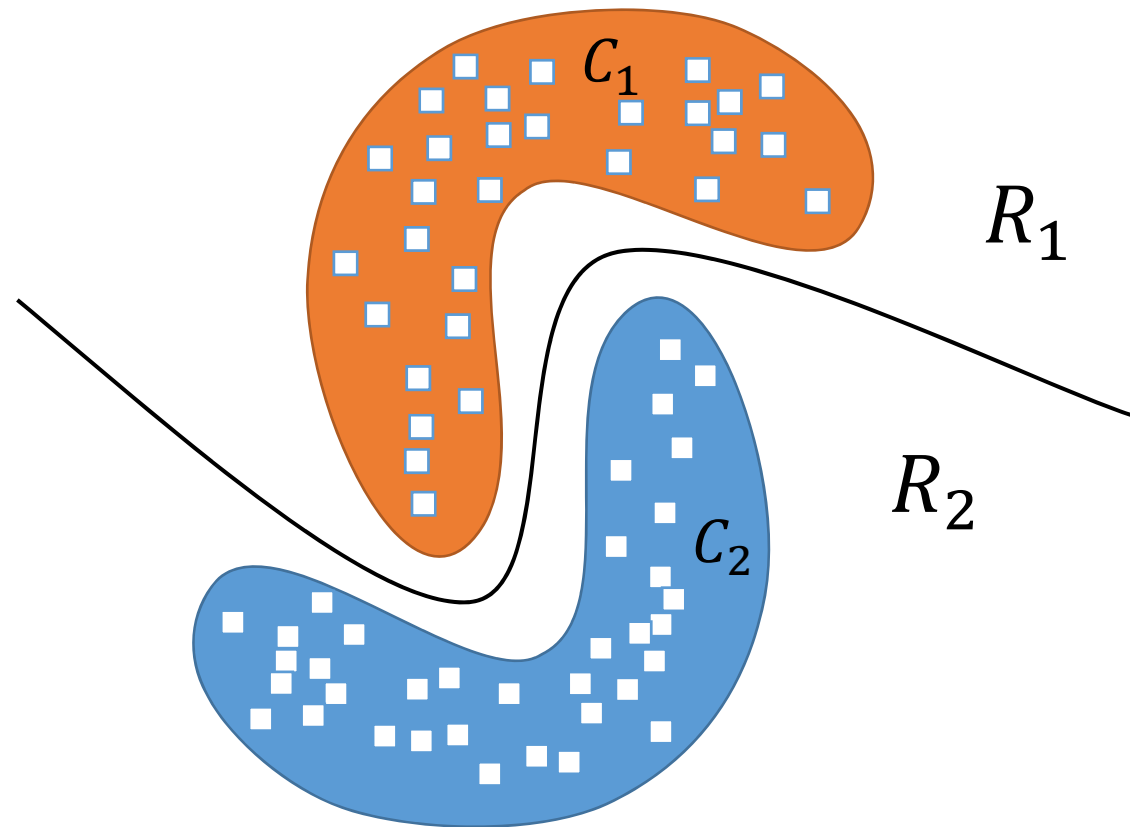
- 線形分離不可能な、複雑なパターン分布の場合、クラス当たり1つの代表点ではうまくいかない。

最近傍法と決定境界



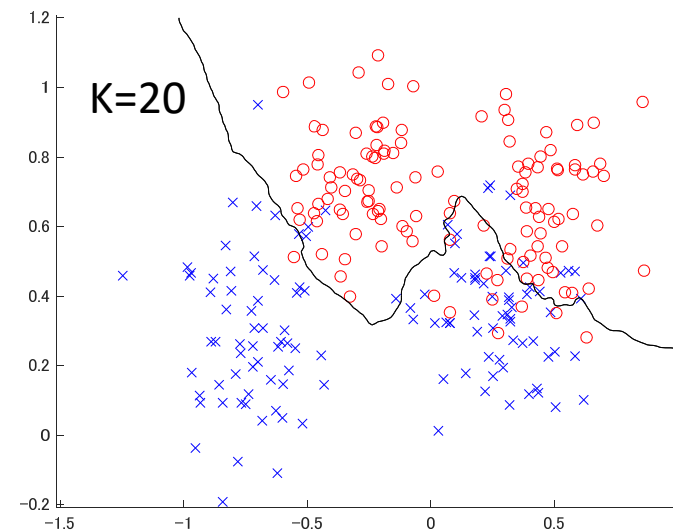
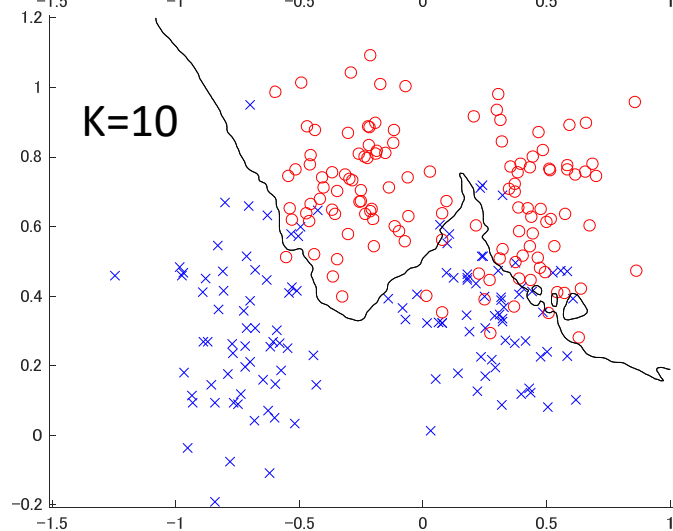
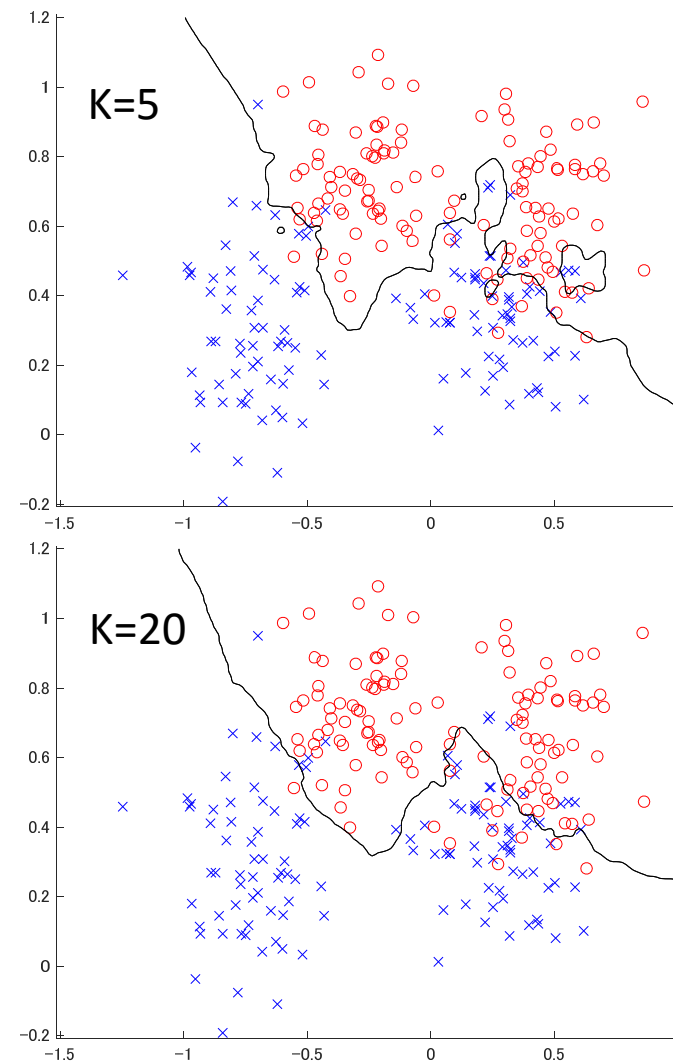
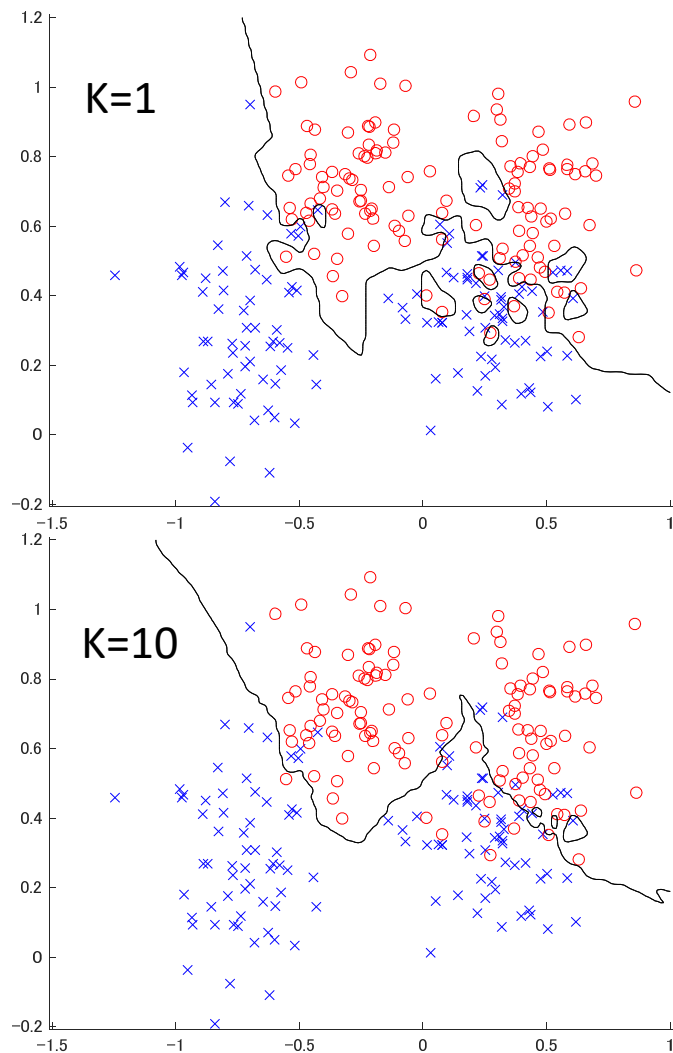
- クラス当たり3つの代表点とすると，決定境界は折れ線となり，適切に決定領域を設定可能.

最近傍法と決定境界



- クラス当たりの代表点が増えると，決定境界はなめらかになる．
- すべてのパターンを代表点と考えると最近傍法となる．

K近傍法と決定境界



- 近傍数 k が増えるにつれて決定境界が滑らかになる。

ベイズ識別

- 各クラスの事前確率, 条件付き確率密度分布が, 事前確率統計的知識として得られる場合には, 識別問題は統計的決定理論の枠組みで, 理論的には完全に定式化される.
- ベイズの定理 (Bayes' theorem)

事前確率 (prior probability)

尤度 (likelihood)

$$P(C_j | x) = \frac{P(C_j)p(x | C_j)}{p(x)}$$

事後確率 (posterior probability)

$$p(x) = \sum_{j=1}^K P(C_j)p(x | C_j) \quad \int p(x)dx = 1 \quad \sum_{j=1}^K P(C_j | x) = 1$$

ベイズ決定則

- 損失の期待値としてのリスク

$d(x)$: 特徴ベクトル空間からクラス集合への決定関数 (decision function)

$$R[d] = \sum_{j=1}^K \int \underline{r(d(x) | C_j)} P(C_j | x) p(x) dx$$

全リスク (overall risk)

$r(C_i | C_j)$: クラス C_j を C_i に決定した時の損失 (loss)

- これを最小とする決定関数 $d(x)$ を求めるのが統計的ベイズ決定則 (Bayes decision rule)

ベイズ識別方式

- **0-1損失**の場合, リスクを最小とする理論的に最適な識別方式はベイズ識別方式である.

$$R[d] = \sum_{j=1}^K \int r(d(x) | C_j) P(C_j | x) p(x) dx$$



$$r(C_i | C_j) = 1 - \delta_{ij}$$

$$x \in C_i, \quad \text{if} \quad P(C_i | x) \geq P(C_j | x) \quad \text{for} \quad j = 1, \dots, K$$

- すなわち, 事後確率最大のクラスに決定を下す識別方式として与えられる.

識別関数

- ベイズ識別方式は、各クラスの確率密度分布が事前に完全に分かっているといけない。そこで識別のため、何らかの関数を各クラスごとに構成して識別を行うことが多い。

$$x \in C_i, \quad \text{if} \quad P(C_i | x) \geq P(C_j | x) \quad \text{for} \quad j = 1, \dots, K$$



$$x \in C_i, \quad \text{if} \quad g_i(x) \geq g_j(x) \quad \text{for} \quad j = 1, \dots, K$$

識別関数 (discriminant function)

- ベイズ識別の意味で最適な識別関数は明らかに事後確率とするもの。

$$g_i(x) = P(C_i | x)$$

パラメトリックな識別方式

- 各クラスのサンプルが正規分布に従う場合

$$p(x|C_j) = \frac{1}{(\sqrt{2\pi})^N \sqrt{|\Sigma_j|}} \exp\left\{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)\right\}$$

- 事後確率の対数（識別に不要な項は無視）

$$g_j(x) = \log P(C_j) - \frac{1}{2} \left\{ \underbrace{(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)}_{\text{マハラノビス距離 (Mahalanobis distance) の二乗}} + \log |\Sigma_j| \right\} \rightarrow \text{2次の識別関数}$$

マハラノビス距離 (Mahalanobis distance) の二乗

- 各クラスの共分散行列が等しい場合 $\Sigma_j = \Sigma_0$

$$g_j(x) = x^T \Sigma_0^{-1} \mu_j - \frac{1}{2} \mu_j^T \Sigma_0^{-1} \mu_j + \log P(C_j) \rightarrow \text{線形識別関数 (linear discriminant function)}$$

- 確率密度分布のパラメータ推定を行い識別器を構成する手法をパラメトリック (parametric) な識別方式という。

一般的な教師付き学習

- 教師付き学習

- 訓練データの対 (\mathbf{x}, \mathbf{y}) から、入力のベクトル \mathbf{x} と教師ベクトル \mathbf{y} の対応関係を $\mathbf{y} = f(\mathbf{x})$ として学び取る枠組み.

$$R(f) = \iint L(f(\mathbf{x}), \mathbf{y}) p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y}$$

写像関数 $f(\mathbf{x})$ は何らかのパラメータ \mathbf{w} を用いて表されることが一般的

$$R(\mathbf{w}) = \iint L(f(\mathbf{x}; \mathbf{w}), \mathbf{y}) p(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y}$$

実際には、同時確率分布 $p(\mathbf{x}, \mathbf{y})$ は分からない場合が多いので、有限のデータを利用して経験リスク (empirical risk) を求める.

$$R(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i)$$

さらに、パラメータが過学習してしまうことを避けるためにペナルティ項を付加した損失関数を用いる

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) + \gamma \psi(\mathbf{w})$$

経験リスク最小化

- 識別器を経験リスクを最小化することで求める

$$\begin{aligned} \mathbf{w}^* &= \operatorname{argmin}_{\mathbf{w}} J(\mathbf{w}) \\ &= \operatorname{argmin}_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) + \gamma \psi(\mathbf{w}) \end{aligned}$$

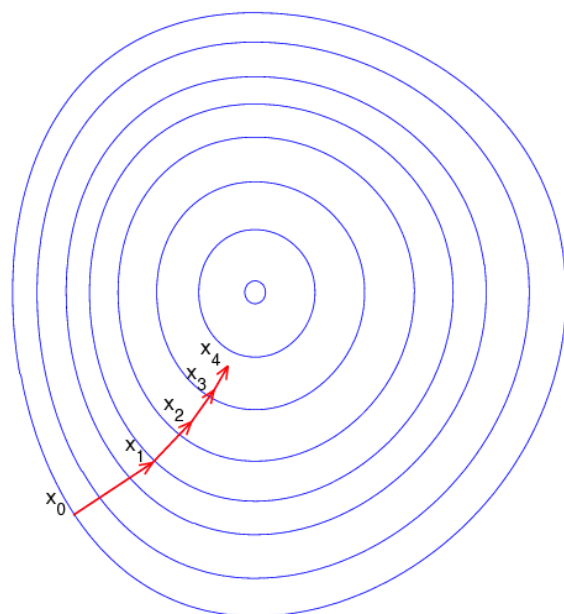
経験リスクを最小化するパラメータを求める
最適化手法についていくつか紹介

勾配降下法

- 勾配降下法 (gradient descent method)
 - 目的関数の微分を求めて、勾配の逆方向に各ステップごとにパラメータを更新していくアルゴリズム

$$\mathbf{w} \leftarrow \mathbf{w} - \varepsilon \nabla_{\mathbf{w}} J(\mathbf{w})$$

学習係数 (learning rate)



http://en.wikipedia.org/wiki/File:Gradient_descent.png

劣勾配降下法

- 劣勾配降下法 (subgradient descent)

- 目的関数が必ずしも微分できないことがある.

- 例) ペナルティ項にL1正則化を用いた場合

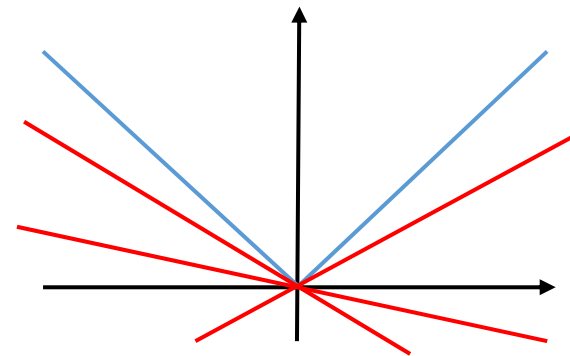
$$\psi(\mathbf{w}) = \sum_{j=1}^d |w_j|$$

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N L(f(\mathbf{x}_i; \mathbf{w}), \mathbf{y}_i) + \gamma \psi(\mathbf{w})$$

- 勾配降下法における目的関数の勾配を劣勾配で置き換えたアルゴリズム
- 点 \mathbf{w} における凸関数 $J(\mathbf{w})$ の劣勾配
 - 任意の \mathbf{v} に対して $J(\mathbf{v}) - J(\mathbf{w}) \geq \mathbf{z}^T (\mathbf{v} - \mathbf{w})$ を満たす \mathbf{z} のこと
 - \mathbf{z} の集合を劣微分 $\partial J(\mathbf{w})$ という.

例: $f(x) = |x|$

$$\partial f(x) = \begin{cases} -1 & \text{if } x < 0 \\ [-1, +1] & \text{if } x = 0 \\ +1 & \text{if } x > 0 \end{cases}$$



$$\mathbf{w} \leftarrow \mathbf{w} - \varepsilon \mathbf{z}, \quad \mathbf{z} \in \partial J(\mathbf{w})$$

ニュートン法


- ニュートン法 (Newton's method)
 - 目的関数の2次のテーラ展開を最小化するように、パラメータの修正量を各ステップごとに決定するアルゴリズム

パラメータ \mathbf{w} の修正量を $d\mathbf{w}$ とする.

修正後のパラメータは $\mathbf{w} \leftarrow \mathbf{w} + d\mathbf{w}$ となる.

$$J(\mathbf{w} + d\mathbf{w}) = J(\mathbf{w}) + \nabla_{\mathbf{w}}J(\mathbf{w})^T d\mathbf{w} + \frac{1}{2} d\mathbf{w}^T \underline{H} d\mathbf{w}$$

ヘッセ行列 (Hessian matrix) $H = \nabla_{\mathbf{w}}^2 J(\mathbf{w})$



$d\mathbf{w}$ で微分して停留点を求める

$$d\mathbf{w} = -H^{-1} \nabla_{\mathbf{w}} J(\mathbf{w})$$

ニュートン法によるパラメータの更新則

$$\mathbf{w} \leftarrow \mathbf{w} - H^{-1} \nabla_{\mathbf{w}} J(\mathbf{w})$$

確率的勾配降下法

- バッチ学習 (batch learning)
 - 各ステップ毎に全ての訓練パターンを用いてパラメータを更新する学習方法
- オンライン学習 (online learning)
 - 一つの訓練パターンが与えられるたびにパラメータを更新する逐次的な学習方法
- 確率的勾配降下法 (stochastic gradient descent, SGD)
 - 代表的なオンライン学習手法

$$\mathbf{w} = \mathbf{w} - \varepsilon_i \Gamma_i \nabla_{\mathbf{w}} L(f(\mathbf{x}_i; \mathbf{w}), \mathbf{t}_i)$$

正定値行列

↓ $\Gamma_i = I$ とする.

$$\mathbf{w} = \mathbf{w} - \varepsilon_i \nabla_{\mathbf{w}} L(f(\mathbf{x}_i; \mathbf{w}), \mathbf{t}_i)$$

確率的勾配降下法

- 確率的勾配降下法は、リスクを直接小さくするようにパラメータを学習するのではなく、リスクの期待値を小さくするようにパラメータを更新
- パラメータ \mathbf{w} が $\mathbf{w} + d\mathbf{w}$ と修正されたとして、リスクの差の期待値が負となっていればよい

$$\mathbb{E}[dR(\mathbf{w})] = \mathbb{E}[R(\mathbf{w} + d\mathbf{w}) - R(\mathbf{w})] \approx \mathbb{E}[d\mathbf{w}]^\top \nabla R(\mathbf{w}) < 0$$

これが負となるためには、任意の正定値行列 C を用いて以下のようにすればよい.

$$\mathbb{E}[d\mathbf{w}] = -C\nabla R(\mathbf{w})$$

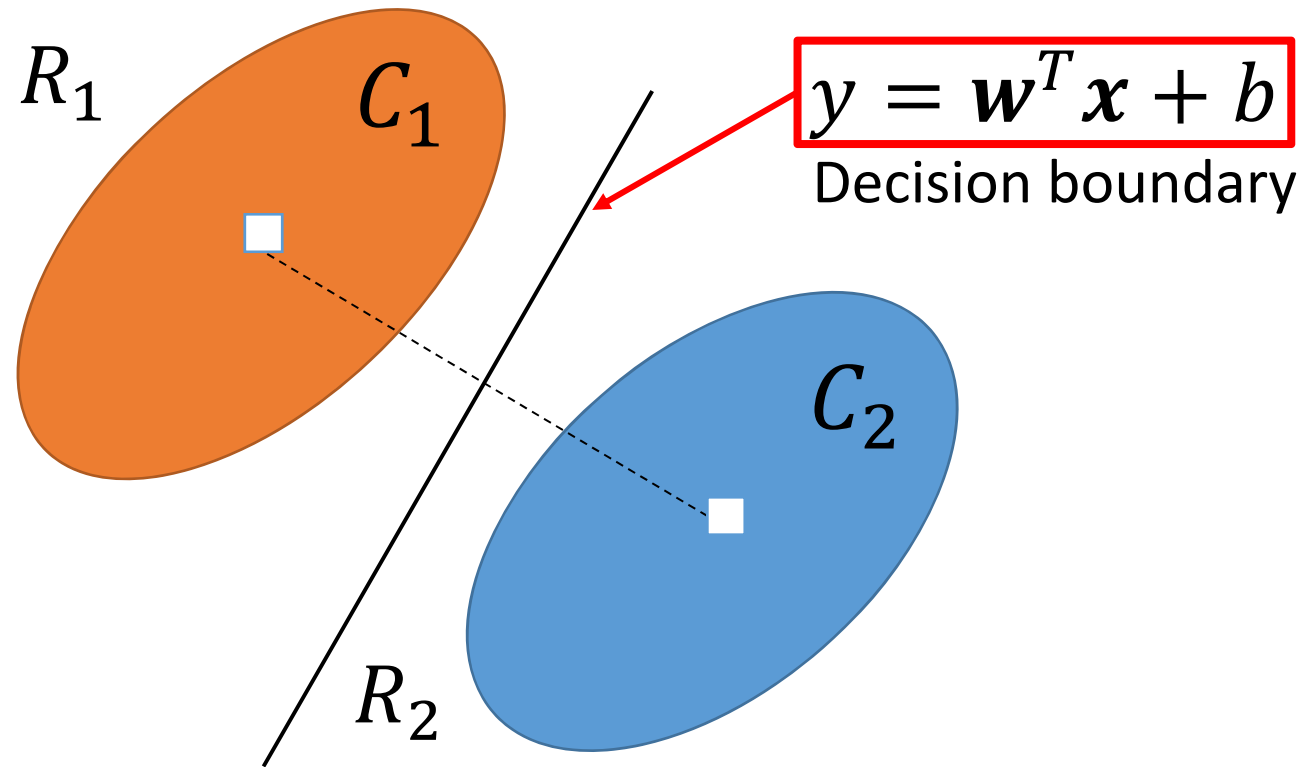
実際に代入すると $-\nabla R(\mathbf{w})^\top C\nabla R(\mathbf{w})$ これは正定値行列の定義より負となる.

リスクは損失の期待値として計算できる $R(\mathbf{w}) = \mathbb{E}[L(f(\mathbf{x}; \mathbf{w}), \mathbf{y})]$

$$\mathbb{E}[d\mathbf{w}] = -C\mathbb{E}[\nabla L(f(\mathbf{x}; \mathbf{w}), \mathbf{y})]$$

つまり $d\mathbf{w} = -C\nabla L(f(\mathbf{x}; \mathbf{w}), \mathbf{y})$ の方向でパラメータを修正すればよい

線形識別器



- 入力パターンに対して線形な識別器を線形識別器 (linear classifier) と呼ぶ。

線形回帰

- 線形回帰 (linear regression)
 - 入出力の係に線形関数を用い, 平均2乗誤差の意味で最適なパラメータを求める手法

線形モデル (linear model)

$$y = f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}, \quad \mathbf{w} \in \mathbb{R}^d, \mathbf{x} \in \mathbb{R}^d$$

訓練データ (training data)

$$\mathcal{D} = \{(\mathbf{x}_i, t_i)\}_{i=1}^N, \quad \mathbf{x} \in \mathbb{R}^d, t \in \mathbb{R}$$

コスト関数 (cost function), 損失関数 (loss function), 誤差関数 (error function)

$$J(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \mathbf{x}_i - t_i)^2 = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2$$

データ行列 (data matrix) $\mathbf{X} = (\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_N)^T \in \mathbb{R}^{N \times d}$

教師ベクトル $\mathbf{t} = (t_1 \ t_2 \ \cdots \ t_N)^T \in \mathbb{R}^N$

線形回帰のパラメータ

- ・ 損失関数を最小化するパラメータを求める

損失関数をパラメータで偏微分して0とする.

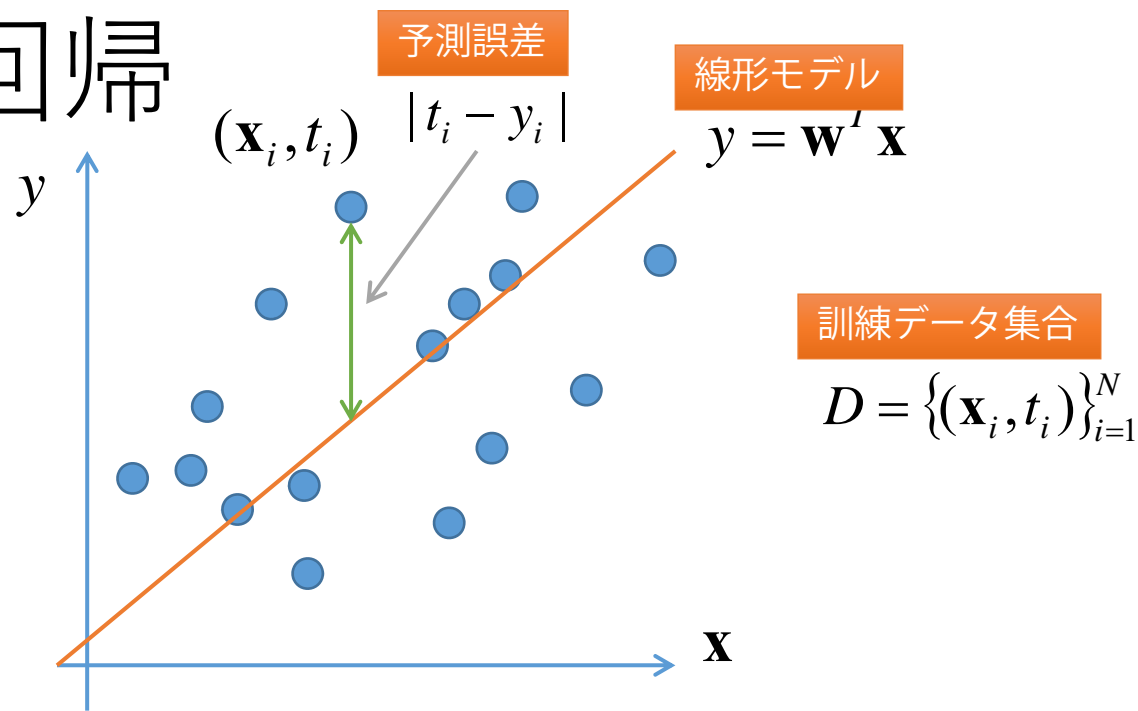
$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \nabla_{\mathbf{w}} J(\mathbf{w}) = X^T (X\mathbf{w} - \mathbf{t}) = 0$$

$X^T X$ が正則であるなら最適なパラメータは次のように求められる.

$$\mathbf{w} = (X^T X)^{-1} X^T \mathbf{t} = X^+ \mathbf{t}$$

疑似逆行列 (pseudo inverse matrix) $X^+ = (X^T X)^{-1} X^T$

リッジ回帰



- 誤差関数に係数が大きな値をとることを防ぐ罰則項を付加することで過学習を避ける。正則化.

- 誤差関数

$$L = \frac{1}{2} \sum_{i=1}^N (t_i - y_i)^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \|\mathbf{X}\mathbf{w} - \mathbf{t}\|^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2$$

Diagram annotations: An orange box labeled '罰則項' (Penalty Term) points to the second term $\frac{\lambda}{2} \|\mathbf{w}\|^2$. A blue box labeled '正則化係数' (Regularization Coefficient) points to the coefficient λ in the same term.

- 二乗誤差のパラメータによる偏微分

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{X}^T (\mathbf{X}\mathbf{w} - \mathbf{t}) + \lambda \mathbf{w} = 0 \longrightarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{t}$$

Diagram annotation: An orange box labeled '逆行列の安定化' (Stabilization of Inverse Matrix) points to the $\lambda \mathbf{I}$ term in the matrix inverse.

Adaline

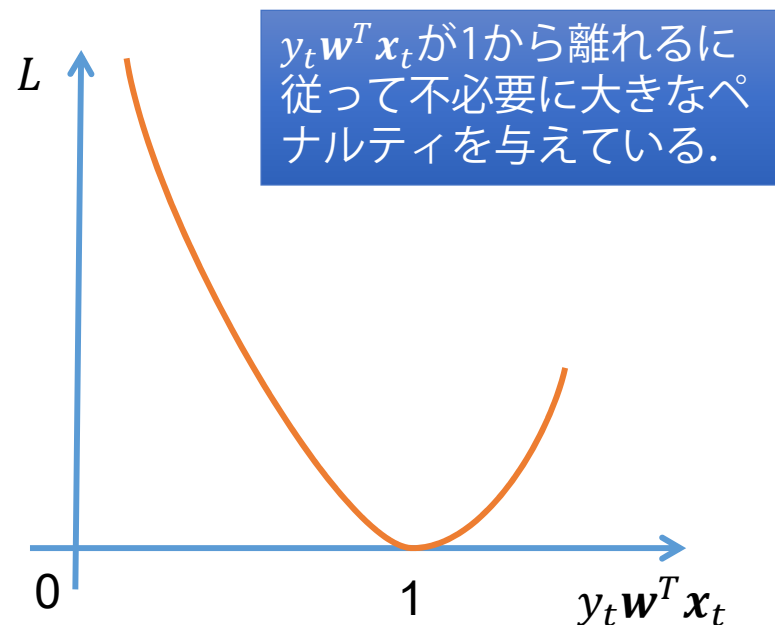
データ集合

$$\{\mathbf{x}_i, y_i\}_{i=1}^N$$

教師ラベル

$$y = \{-1, +1\}$$

損失関数



$$L(f(\mathbf{x}; \mathbf{w}), y) = \frac{1}{2} (y - \mathbf{w}^\top \mathbf{x})^2 \Rightarrow L = \frac{1}{2} (1 - y \mathbf{w}^T \mathbf{x})^2$$

SGDによるパラメータ更新式

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t (y_t - \mathbf{w}_t^\top \mathbf{x}_t) \mathbf{x}_t$$

Perceptron

データ集合

$$\{\mathbf{x}_i, y_i\}_{i=1}^N$$

教師ラベル

$$y = \{-1, +1\}$$

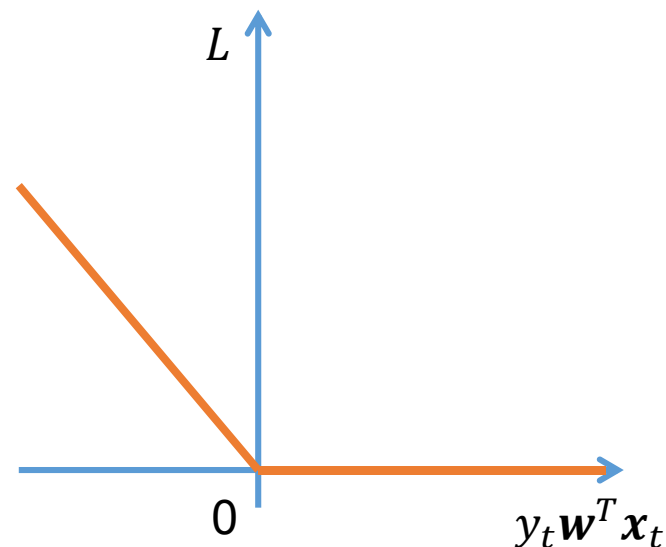
損失関数

識別に成功した場合 : $y_t \mathbf{w}^T \mathbf{x}_t \geq 0$

識別に失敗した場合 : $y_t \mathbf{w}^T \mathbf{x}_t < 0$

識別に成功した場合の損失 : 0

識別に失敗した場合の損失 : $-y_t \mathbf{w}^T \mathbf{x}_t$



$$L(f(\mathbf{x}; \mathbf{w}), y) = \max [0, -y \mathbf{w}^\top \mathbf{x}]$$

SGDによるパラメータ更新式

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta_t \begin{cases} y_t \mathbf{x}_t & \text{if } y_t \mathbf{w}_t^\top \mathbf{x}_t \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

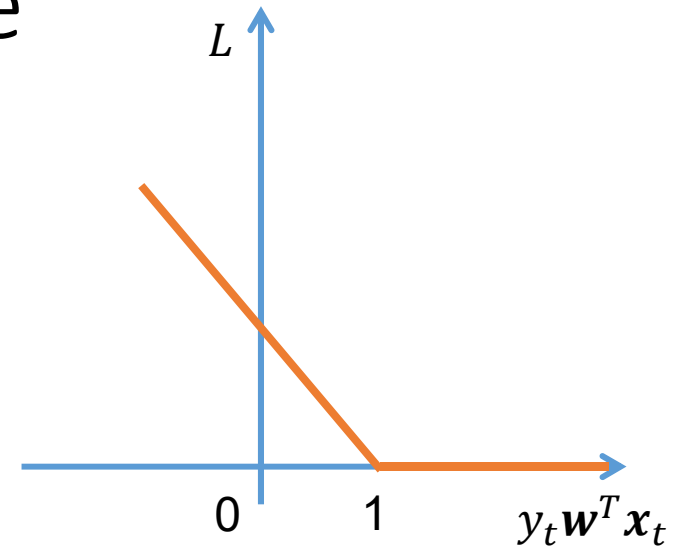
Support Vector Machine

データ集合

$$\{\mathbf{x}_i, y_i\}_{i=1}^N \quad y = \{-1, +1\}$$

ヒンジ損失関数 (hinge loss function)

$$L(f(\mathbf{x}; \mathbf{w}), y) = \max [0, 1 - y\mathbf{w}^\top \mathbf{x}]$$



SVMの損失関数

$$l(f(\mathbf{x}; \mathbf{w}), y) = \lambda \|\mathbf{w}\|^2 + \max [0, 1 - y\mathbf{w}^\top \mathbf{x}]$$

ヒンジ損失に正則化項を追加

SGDによるパラメータ更新式

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta_t \begin{cases} \lambda \mathbf{w}_t & \text{if } y_t \mathbf{w}_t^\top \mathbf{x}_t > 1 \\ \lambda \mathbf{w}_t - y_t \mathbf{x}_t & \text{otherwise} \end{cases}$$

ロジスティック回帰

- 線形回帰 (linear regression)
 - 線形モデルによって連続値を予測する.
- ロジスティック回帰 (logistic regression)
 - 非連続なクラスラベルを予測する.
 - 例) 2クラス識別, 正例: 1, 負例: 0

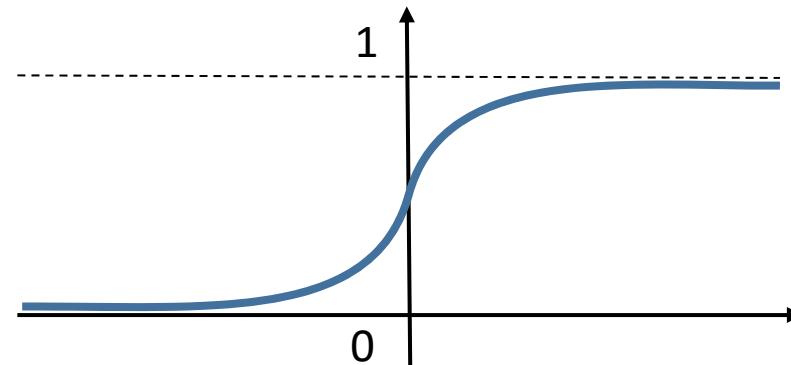
$$\mathcal{D} = \{(\mathbf{x}_i, t_i)\}_{i=1}^N, \quad \mathbf{x} \in \mathbb{R}^d, t \in \{0, 1\}$$

$$P(y = 1|\mathbf{x}) = f(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})} = \sigma(\mathbf{w}^T \mathbf{x})$$

$$P(y = 0|\mathbf{x}) = 1 - P(y = 1|\mathbf{x}) = 1 - f(\mathbf{x})$$

シグモイド関数 (sigmoid function)

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



スカッシング関数 (squashing function) : $-\infty$ から $+\infty$ の出力を 0 から 1 の値に変換

交差エントロピー誤差

- 尤度（likelihood）を最大化するパラメータを求める。

尤度関数（likelihood function）

$$L = \prod_{i=1}^N P(y = 1|\mathbf{x}_i)^{t_i} (1 - P(y = 1|\mathbf{x}_i))^{1-t_i}$$

負の対数尤度（negative log likelihood）

$$\begin{aligned} J(\mathbf{w}) &= -\ln L \\ &= -\sum_{i=1}^N \{t_i \ln P(y = 1|\mathbf{x}_i) + (1 - t_i) \ln(1 - P(y = 1|\mathbf{x}_i))\} \end{aligned}$$

交差エントロピー誤差関数（cross-entropy error function）

偏微分 $\nabla_{\mathbf{w}} J(\mathbf{w}) = 0$ として計算しても一発で最適なパラメータが求まらない。

勾配降下法

- 勾配降下法 (gradient descent method)
 - 目的関数の微分を求めて、勾配の逆方向に各ステップごとにパラメータを更新していくアルゴリズム

$$\mathbf{w} \leftarrow \mathbf{w} - \varepsilon \nabla_{\mathbf{w}} J(\mathbf{w})$$

学習係数 (learning rate)

ロジスティック回帰の勾配


$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \sum_{i=1}^N (P(y = 1 | \mathbf{x}_i) - t_i) \mathbf{x}_i = X^T (\mathbf{p} - \mathbf{t})$$

$$\mathbf{p} = (P(y = 1 | \mathbf{x}_1) \ P(y = 1 | \mathbf{x}_2) \ \cdots \ P(y = 1 | \mathbf{x}_N))^T$$

$$\mathbf{t} = (t_1 \ t_2 \ \cdots \ t_N)^T$$

ロジスティック回帰のニュートン法による解

- ヘッセ行列を求める.

$$H = \nabla_{\mathbf{w}}^2 J(\mathbf{w}) = \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T P(y = 1 | \mathbf{x}_i) (1 - P(y = 1 | \mathbf{x}_i))$$
$$= \mathbf{X}^T \mathbf{W} \mathbf{X}$$


$$\mathbf{W} = \text{diag} \left(P(y = 1 | \mathbf{x}_1) (1 - P(y = 1 | \mathbf{x}_1)) \cdots P(y = 1 | \mathbf{x}_N) (1 - P(y = 1 | \mathbf{x}_N)) \right)$$

ロジスティック回帰のニュートン法によるパラメータの更新則

$$\mathbf{w} \leftarrow \mathbf{w} - (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{p} - \mathbf{t})$$

反復重み付き最小二乗法 (iterative reweighted least squares method, IRLS)

ソフトマックス回帰

- ソフトマックス回帰 (softmax regression, multinomial logistic regression)
 - ロジスティック回帰の一般化で, 多クラスに対応した手法
 - K 個のクラス識別問題を考える.

$$\mathcal{D} = \{(\mathbf{x}_i, t_i)\}_{i=1}^N, \quad \mathbf{x} \in \mathbb{R}^d, t \in \{1, \dots, K\}$$

各クラスの事後確率を求める. 各クラスごとに重みベクトル $\mathbf{w}^{(j)}$ を持つ.

$$P(y = 1|\mathbf{x}) = \frac{\exp(\mathbf{w}^{(1)T} \mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}^{(j)T} \mathbf{x})}$$

$$P(y = 2|\mathbf{x}) = \frac{\exp(\mathbf{w}^{(2)T} \mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}^{(j)T} \mathbf{x})}$$

$$\vdots$$

$$P(y = K|\mathbf{x}) = \frac{\exp(\mathbf{w}^{(K)T} \mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}^{(j)T} \mathbf{x})}$$

ソフトマックス関数
(softmax function)

$$g(z^{(j)}) = \frac{\exp(z^{(j)})}{\sum_{i=1}^K \exp(z^{(i)})}$$
$$\sum_j g(z^{(j)}) = 1$$

多クラスの交差エントロピー誤差

- 尤度を最大化するパラメータを求める。

尤度関数 (likelihood function) $L = \prod_{i=1}^N \prod_{j=1}^K P(t_i = j | \mathbf{x}_i)^{I(t_i=j)}$

指示関数 (indicator function)
 $I(\text{true}) = 1, I(\text{false}) = 0$

$$W = (\mathbf{w}^{(1)} \mathbf{w}^{(2)} \dots \mathbf{w}^{(K)}) \in \mathbb{R}^{d \times K}$$

負の対数尤度

$$\begin{aligned} J(W) &= -\ln L = -\sum_{i=1}^N \sum_{j=1}^K I(t_i = j) \ln P(t_i = j | \mathbf{x}_i) \\ &= -\sum_{i=1}^N \sum_{j=1}^K I(t_i = j) \ln \frac{\exp(\mathbf{w}^{(j)T} \mathbf{x}_i)}{\sum_{j=1}^K \exp(\mathbf{w}^{(j)T} \mathbf{x}_i)} \end{aligned}$$

多クラスの交差エントロピー誤差関数 (cross-entropy error function)

損失関数の微分

$$\nabla_{\mathbf{w}^{(j)}} J(W) = -\sum_{i=1}^N [(I(t_i = j) - P(t_i = j | \mathbf{x}_i)) \mathbf{x}_i]$$

ソフトマックス回帰モデルの学習

- 勾配降下法やニュートン法でパラメータを求める.

データ行列 $X = (\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_N)^T \in \mathbb{R}^{N \times d}$

事後確率の行列 $P = (\mathbf{p}_1 \ \mathbf{p}_2 \ \cdots \ \mathbf{p}_N)^T \in \mathbb{R}^{N \times K}$

$$\mathbf{p}_i = (P(y = 1|\mathbf{x}_i) \ P(y = 2|\mathbf{x}_i) \ \cdots \ P(y = K|\mathbf{x}_i))^T \in \mathbb{R}^K$$

教師情報の行列 $T = (\mathbf{t}_1 \ \mathbf{t}_2 \ \cdots \ \mathbf{t}_N)^T \in \mathbb{R}^{N \times K}$

\mathbf{t}_j : one of K符号化, \mathbf{x}_j の正解クラスが k の時,
 k 番目の要素が1, それ以外の要素が0の K 次元のベクトル

損失関数の微分

$$\nabla_{\mathbf{w}^{(j)}} J(W) = - \sum_{i=1}^N [(I(t_i = j) - P(t_i = j|\mathbf{x}_i)) \mathbf{x}_i]$$



行列を用いると簡便に表記できる！

$$\nabla_W J(W) = X^T (P - T)$$

ソフトマックス回帰の勾配降下法

$$W \leftarrow W - \varepsilon X^T (P - T)$$

ソフトマックス回帰とロジスティック回帰

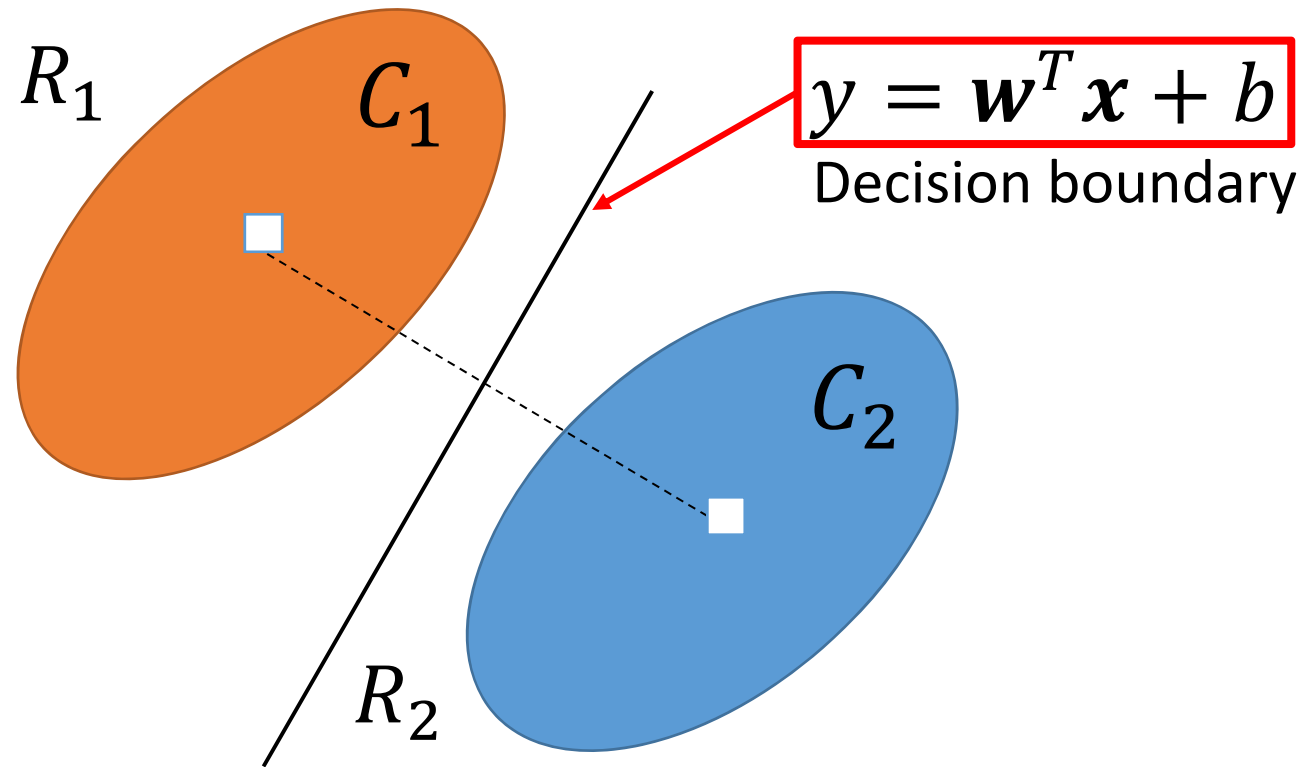
- 2クラスのソフトマックス回帰がロジスティック回帰と等価であることを示す.

$$\begin{aligned} P(y = 1|\mathbf{x}) &= \frac{\exp(\mathbf{w}^{(1)T} \mathbf{x})}{\sum_{j=1}^K \exp(\mathbf{w}^{(j)T} \mathbf{x})} \\ &= \frac{\exp(\mathbf{w}^{(1)T} \mathbf{x})}{\exp(\mathbf{w}^{(1)T} \mathbf{x}) + \exp(\mathbf{w}^{(2)T} \mathbf{x})} \\ &= \frac{\exp(0)}{\exp(0) + \exp(\mathbf{w}^{(2)T} \mathbf{x} - \mathbf{w}^{(1)T} \mathbf{x})} \\ &= \frac{1}{1 + \exp(-(\mathbf{w}^{(1)} - \mathbf{w}^{(2)})^T \mathbf{x})} \\ &= \frac{1}{1 + \exp(-\mathbf{w}'^T \mathbf{x})} = \sigma(\mathbf{w}'^T \mathbf{x}) \end{aligned} \quad \mathbf{w}' = \mathbf{w}^{(1)} - \mathbf{w}^{(2)}$$

$$P(y = 2|\mathbf{x}) = 1 - \sigma(\mathbf{w}'^T \mathbf{x})$$

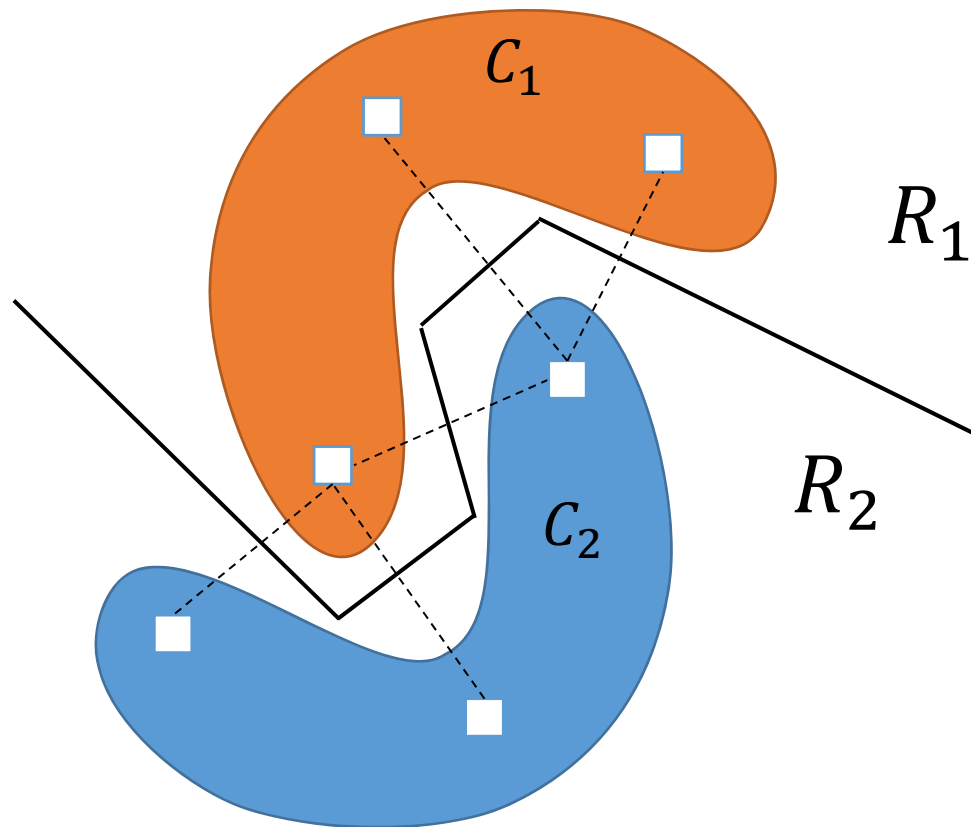
非線形識別関数

線形識別器



- 入力パターンに対して線形な識別器を線形識別器 (linear classifier) と呼ぶ。

区分的線形識別器



クラス C_k の識別関数

$$f_k(\mathbf{x}) = \max_{l=1, \dots, L_k} \{f_k^l(\mathbf{x})\}$$
$$f_k^l(\mathbf{x}) = b_k^l + \langle \mathbf{w}_k^l, \mathbf{x} \rangle$$

- 複数の線形識別器で構成される識別器を区分的線形識別器 (piecewise linear classifier) という。
- 最近傍法は区分的線形識別器となる。
- フィードフォワード型多層ニューラルネットワークは区分的線形識別器と関連が深い。中間層のユニット数 \Leftrightarrow 副次識別関数の数。

一般識別関数

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$$

- 線形決定境界を識別する線形決定境界は、超平面で扱える。複雑な非線形決定境界を識別する線形決定境界は、超平面で扱える。

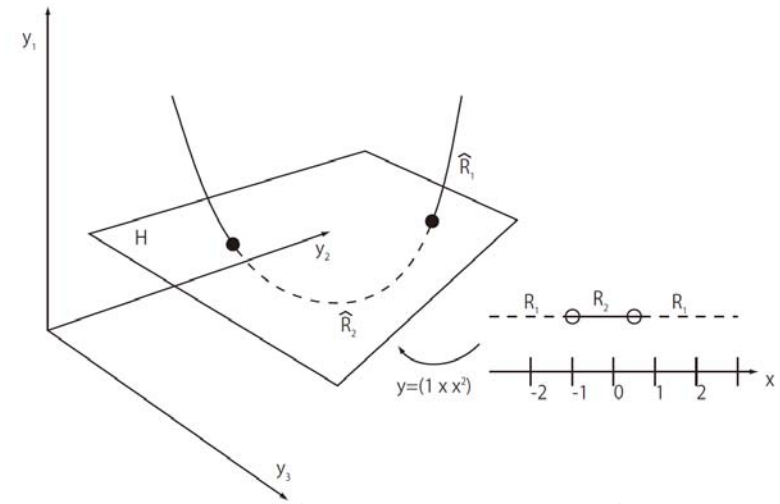
例：2次の識別関数 $g(x) = w_0 + w_1x + w_2x^2$

$$\mathbf{y} = (1, x, x^2)^T$$

と置く

$$\mathbf{w} = (w_0, w_1, w_2)^T$$

$g(x) = \mathbf{w}^T \mathbf{y}$  線形識別関数



\mathbf{x} に関する任意の k 個の関数 $\phi_i(\mathbf{x}), i = 1, \dots, k$ とおく. 基底関数 (basis function).

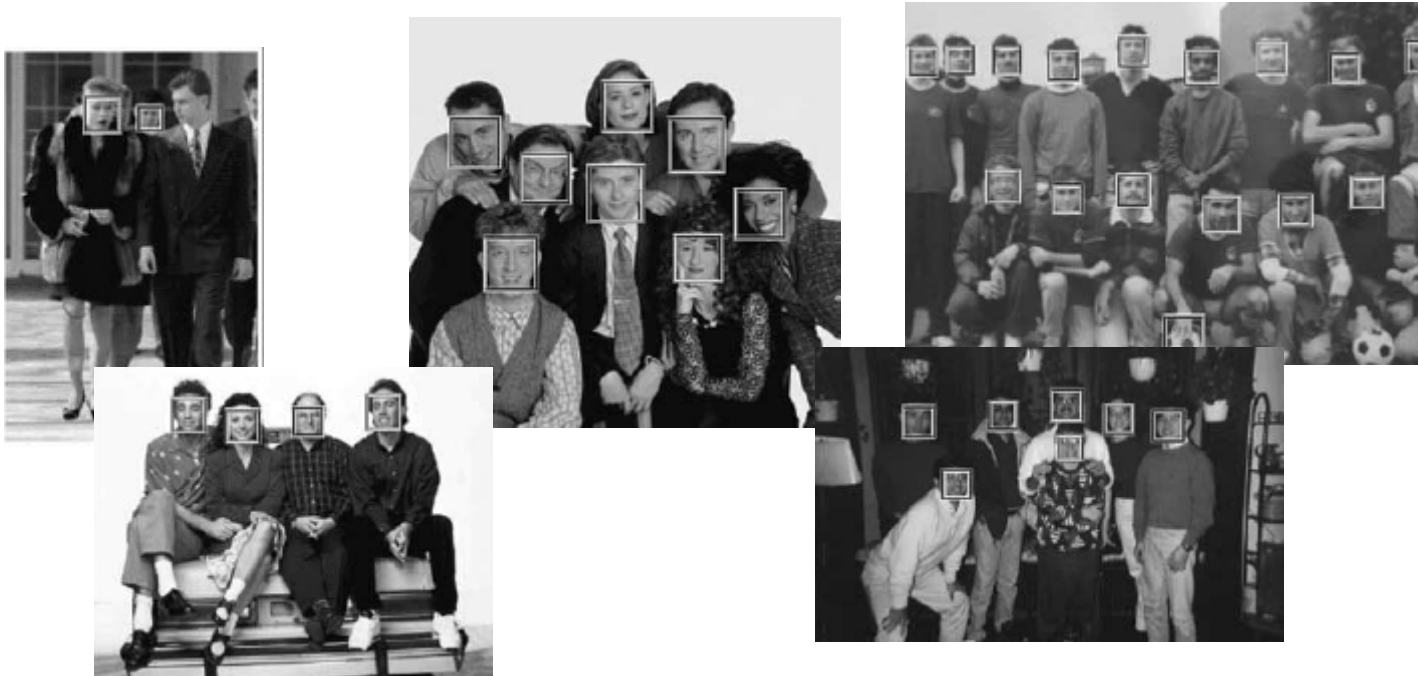
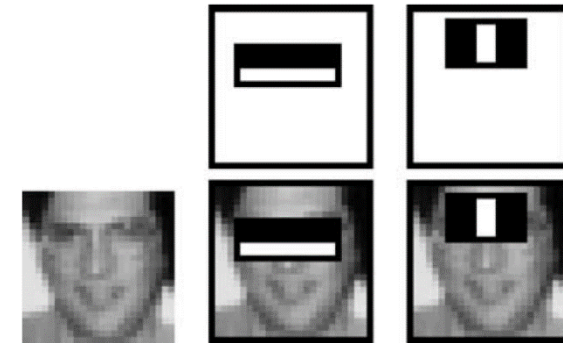
$$g(x) = \sum_{i=1}^k w_i \phi_i(\mathbf{x}) + w_0 = \mathbf{w}^T \mathbf{y} \quad \mathbf{y} = (1, \phi_1(\mathbf{x}), \dots, \phi_k(\mathbf{x}))^T = \boldsymbol{\phi}(\mathbf{x})$$

一般識別関数 (generalized discriminant function)

一般識別関数を用いれば、非線形を含む任意の識別関数を実現できる。
またこれは線形識別関数の学習法が適用できる。

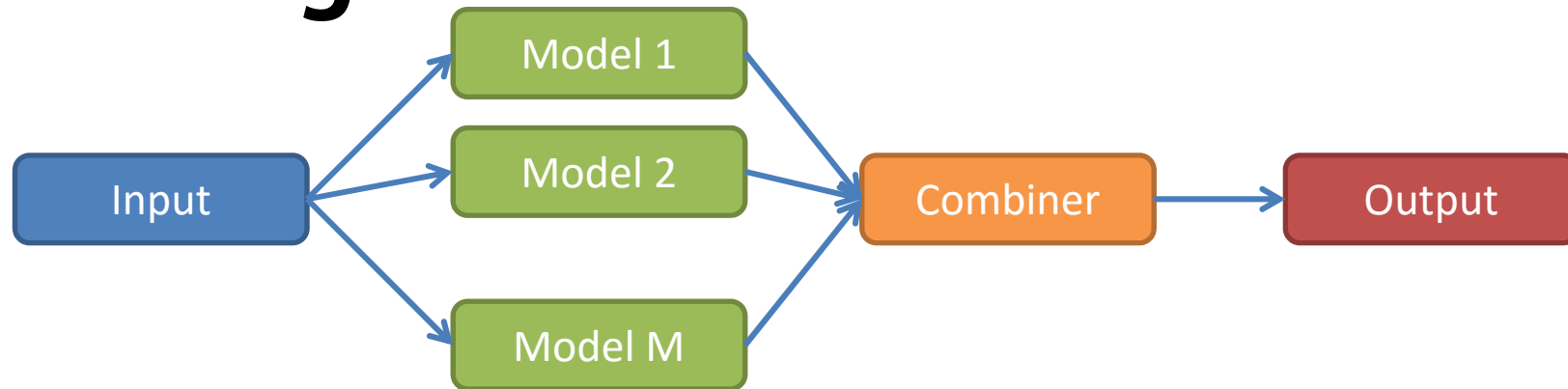
Viola Jones Face Detector

- リアルタイム物体検出手法
- 訓練は遅いが検出は非常に高速
- 主要なアイデア
 - 高速な特徴評価のための積分画像
 - 特徴抽出のためのBoosting
 - 非顔領域を高速に排除するためのAttentional cascade



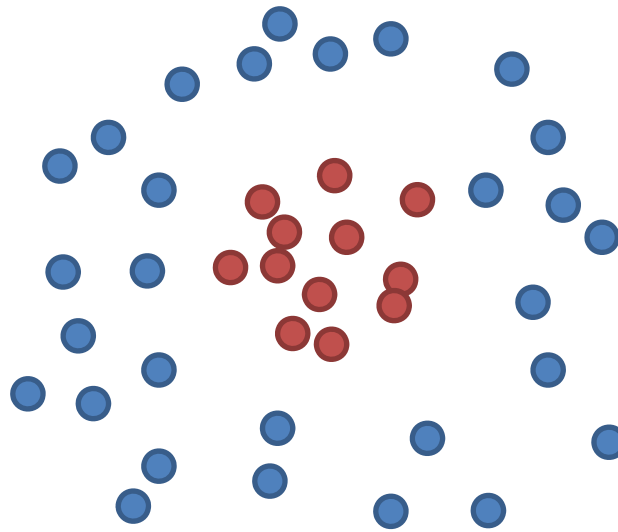
Viola and Jones. Robust Real-Time Face Detection. International Journal of Computer Vision 57(2), 137–154, 2004.

Boosting

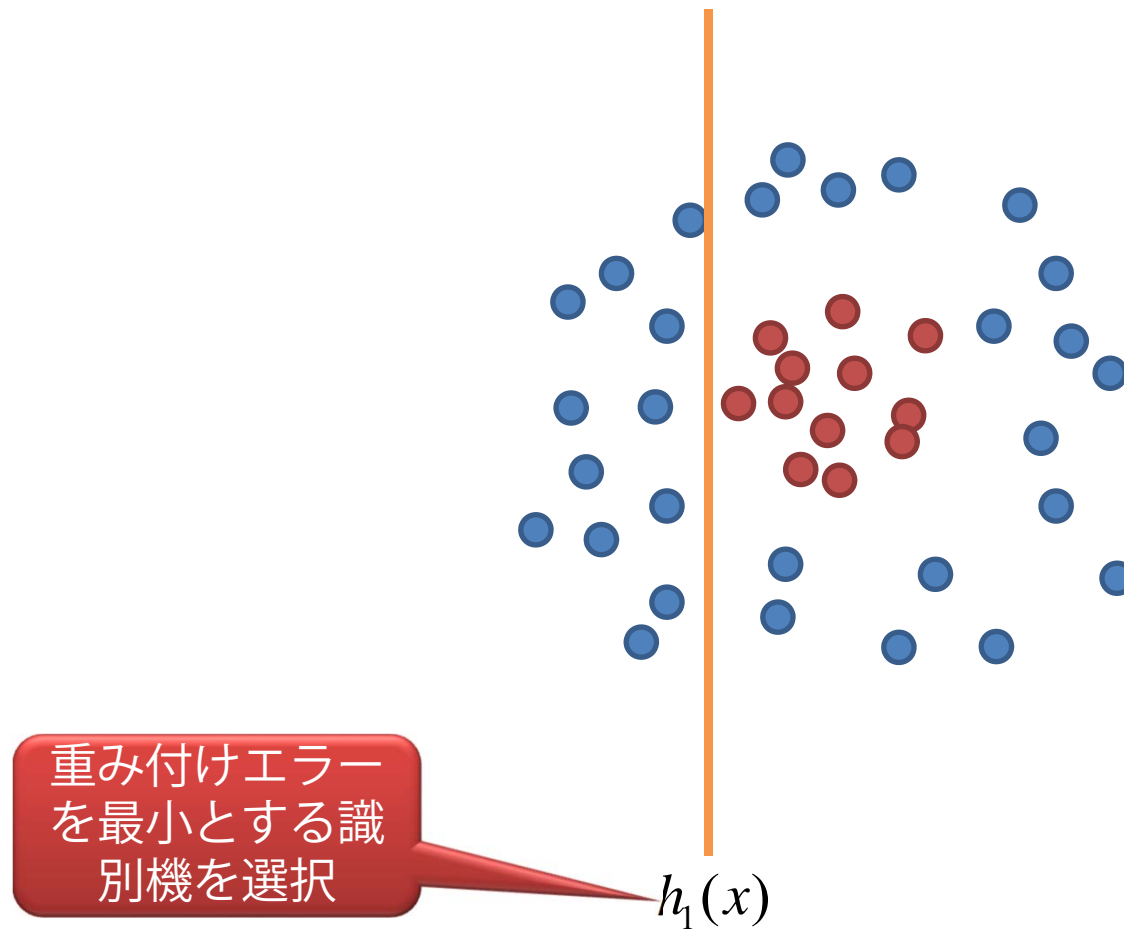


- Boostingとは
 - 複数の基本となる学習機械を組み合わせて、精度の高い学習機械を構成する手法.
 - 以前に学習した学習機械の結果を用いて間違いの多い訓練サンプルに重み付けを行い、この重み付けされたサンプルで新たに学習機械を学習する.
 - 逐次的に生成される学習機械を統合して一つの学習機械とする.
- AdaBoostにおける訓練
 1. 各訓練サンプルに等価な重みを与える
 2. For $m=1:M$
 - A) 訓練エラーを最小とする弱学習器を選択する
 - B) 現在の弱学習器によって誤識別された訓練サンプルの重みを増加
 3. 全ての弱学習器を重み付け線形結合し、最終的な識別器を構成する
 - 各弱学習器の重み付けは、識別精度に比例

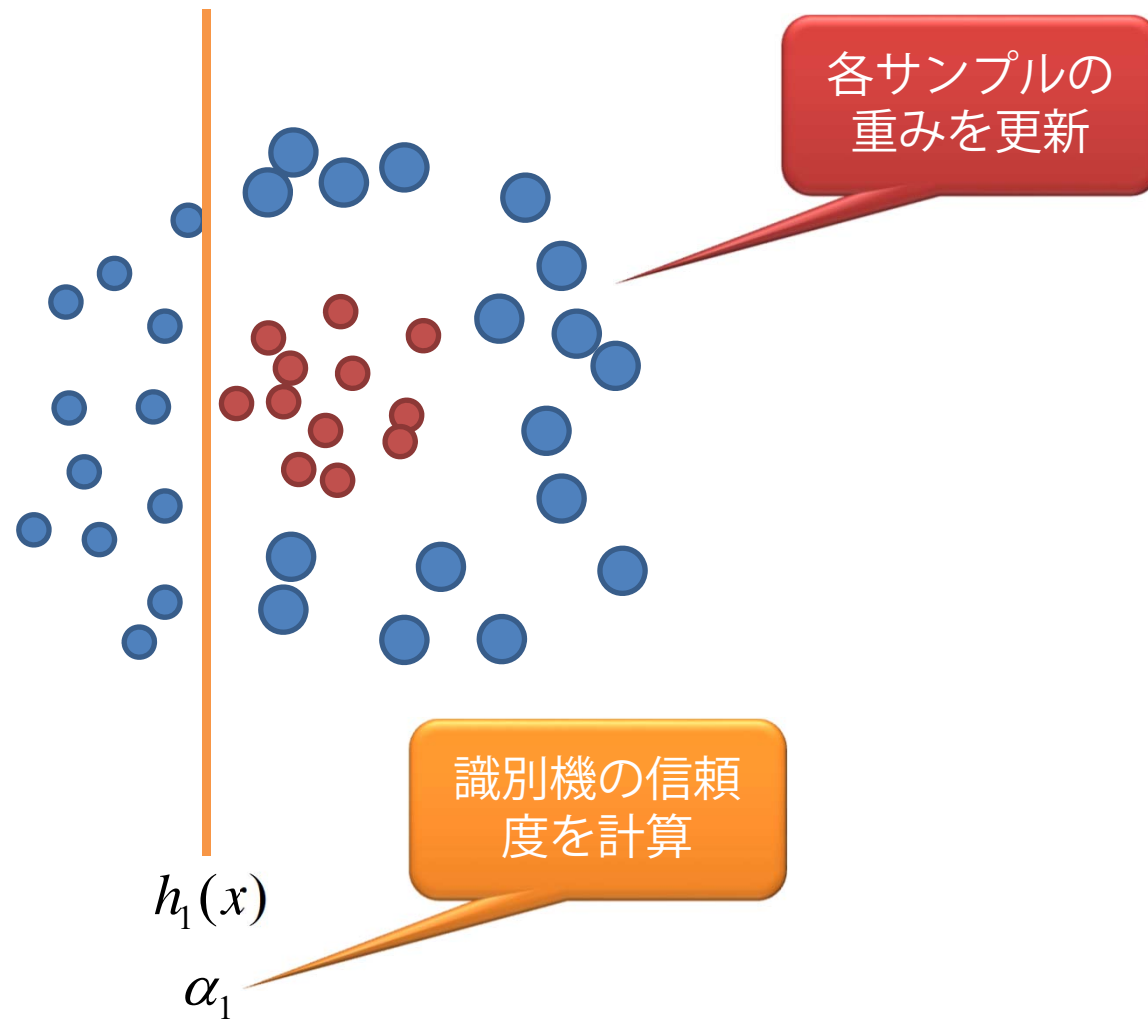
AdaBoostの学習過程



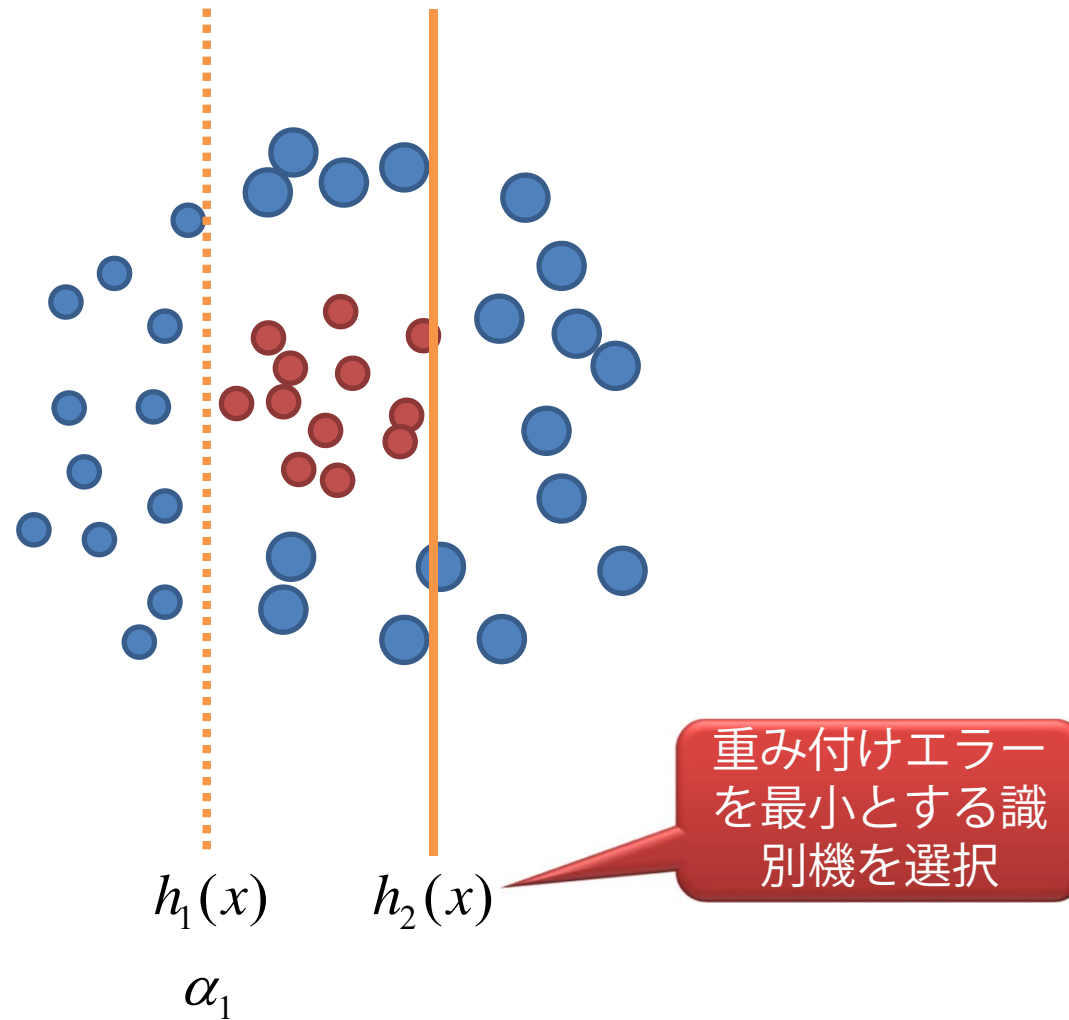
AdaBoostの学習過程



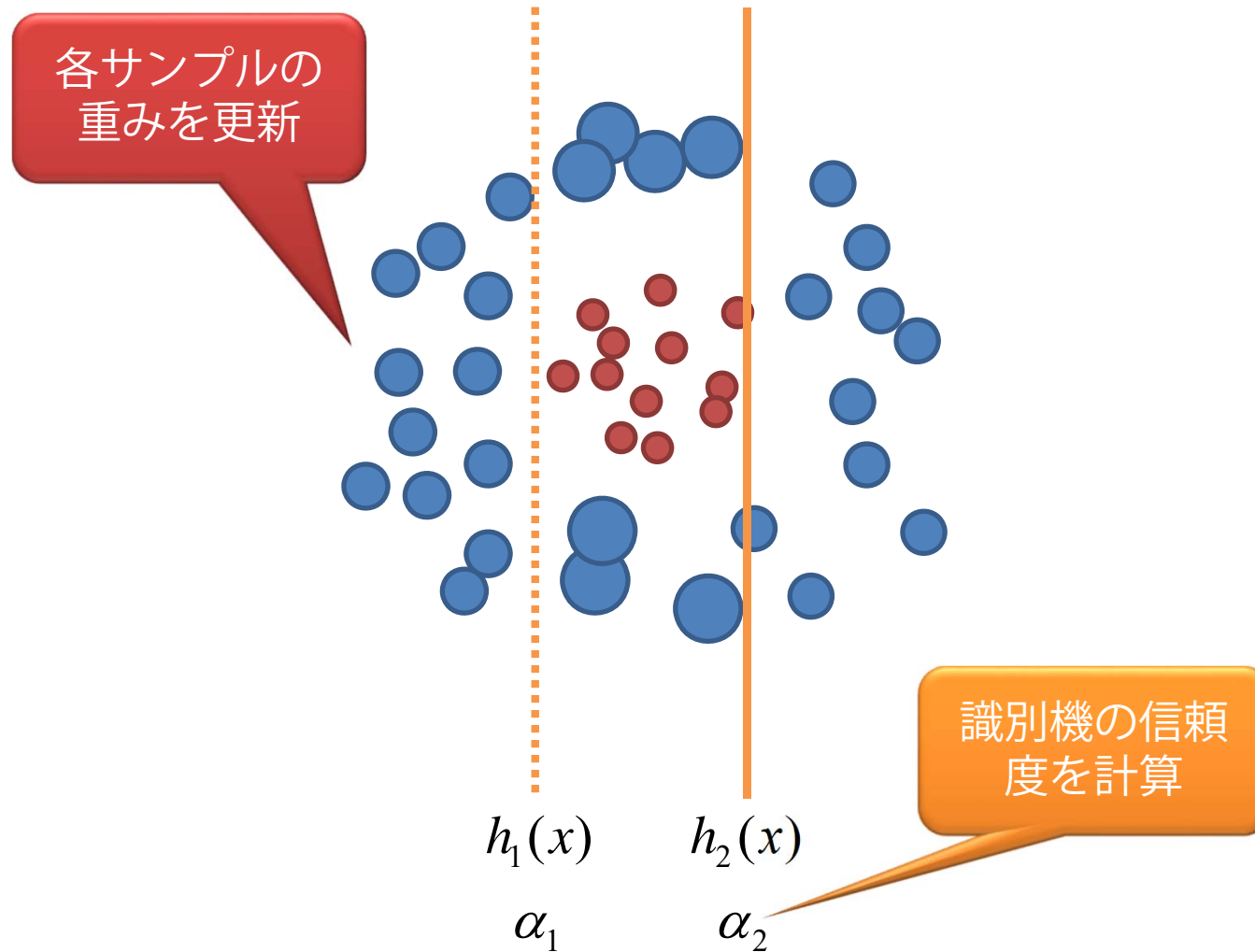
AdaBoostの学習過程



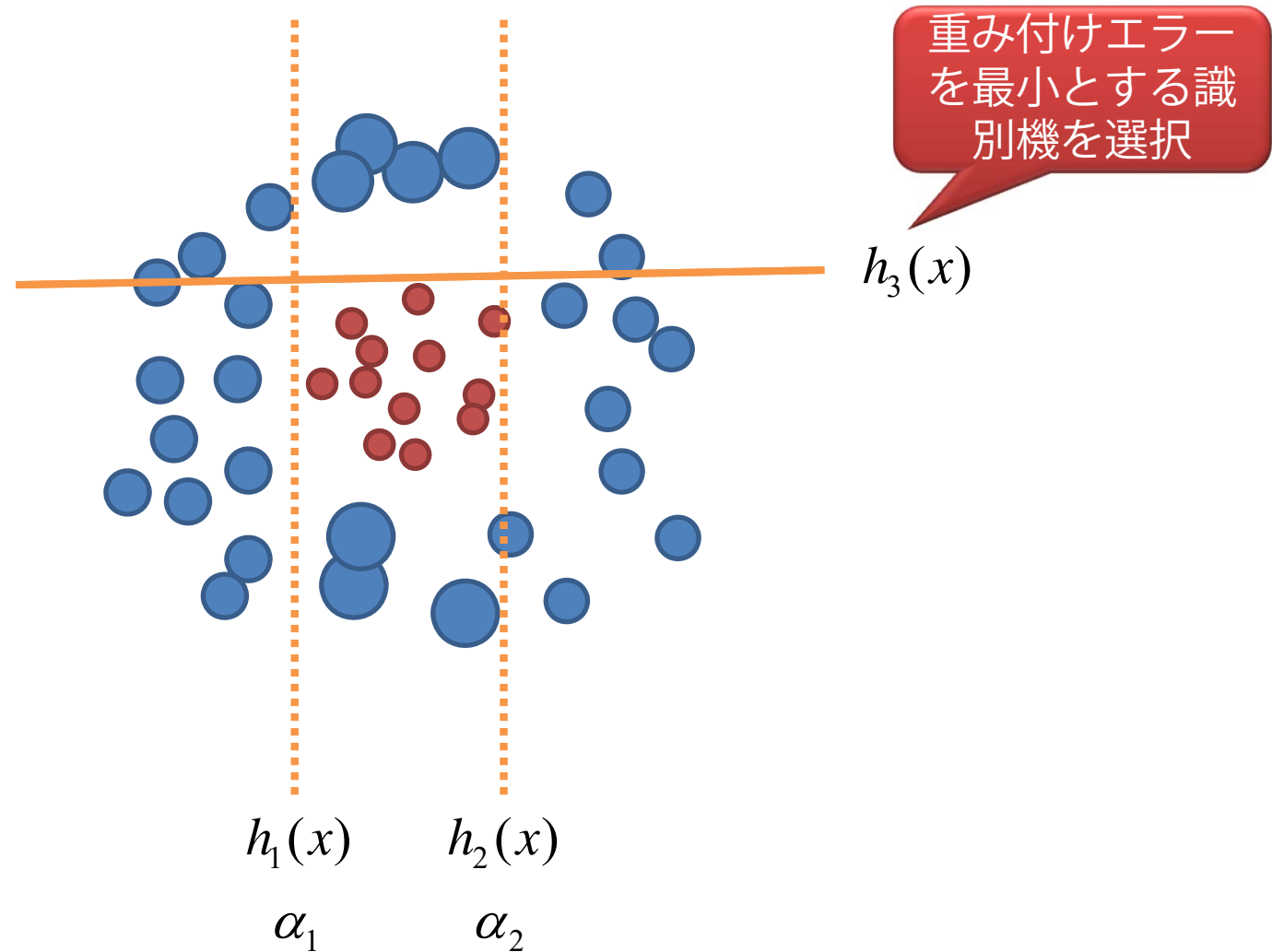
AdaBoostの学習過程



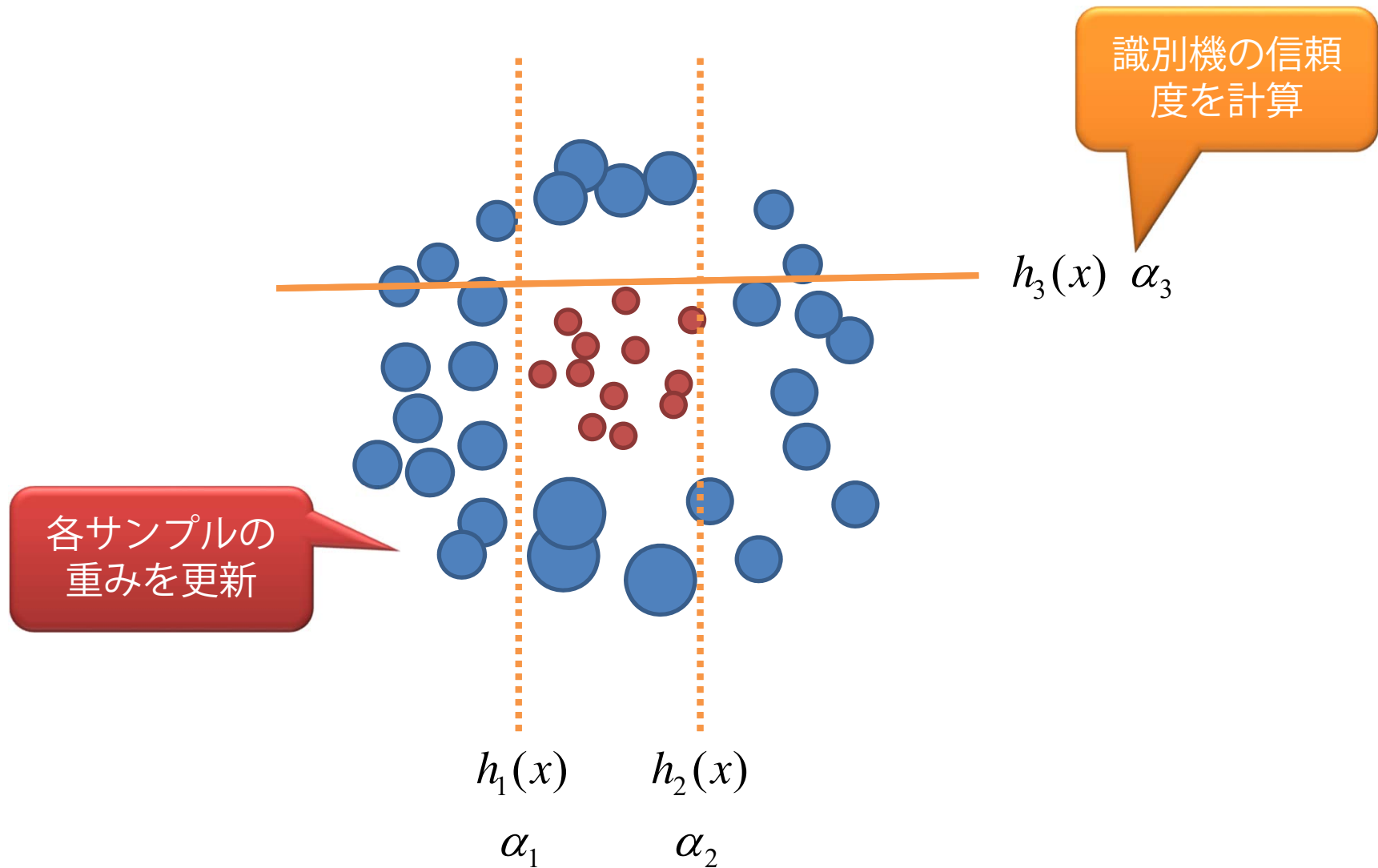
AdaBoostの学習過程



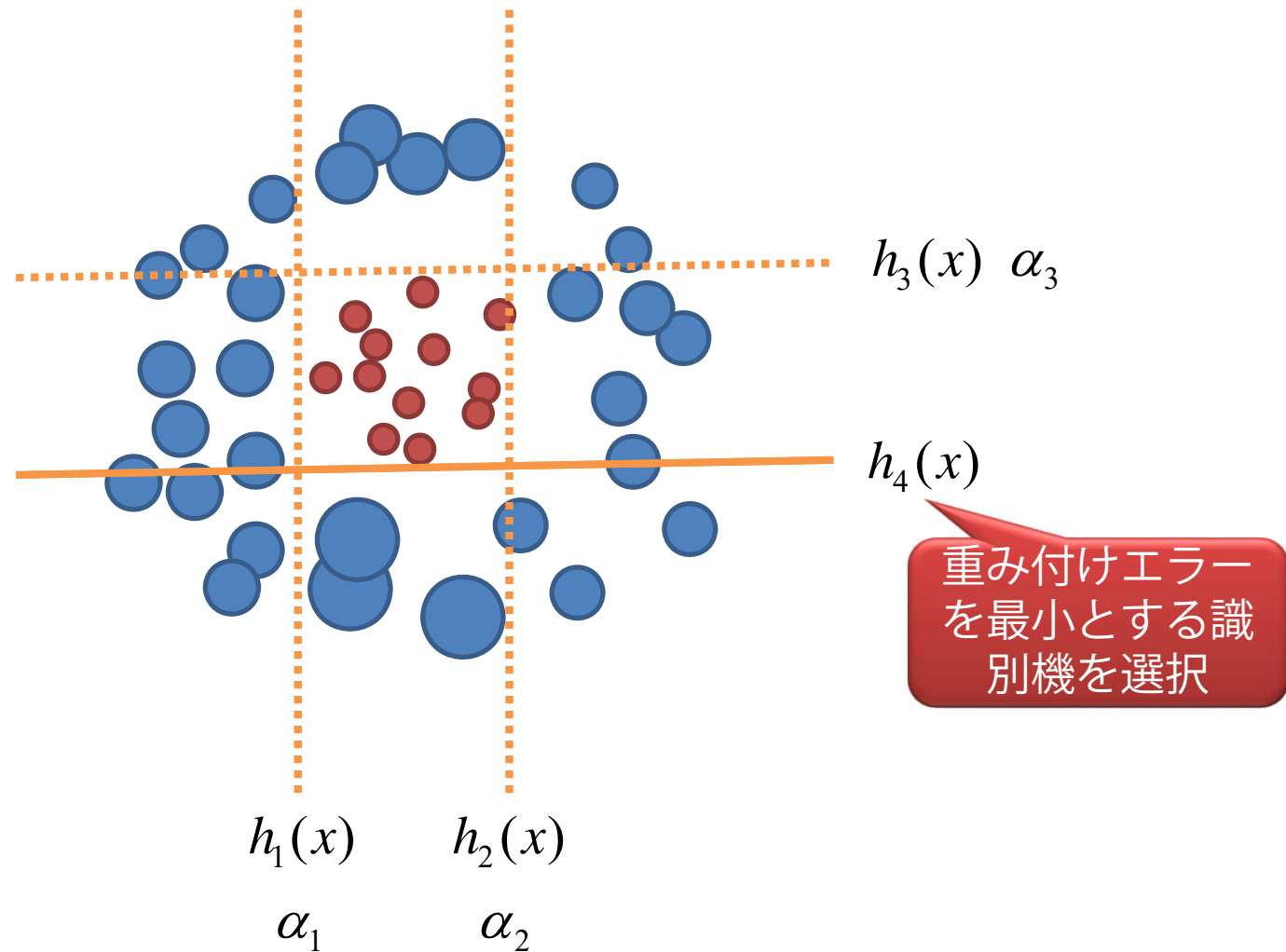
AdaBoostの学習過程



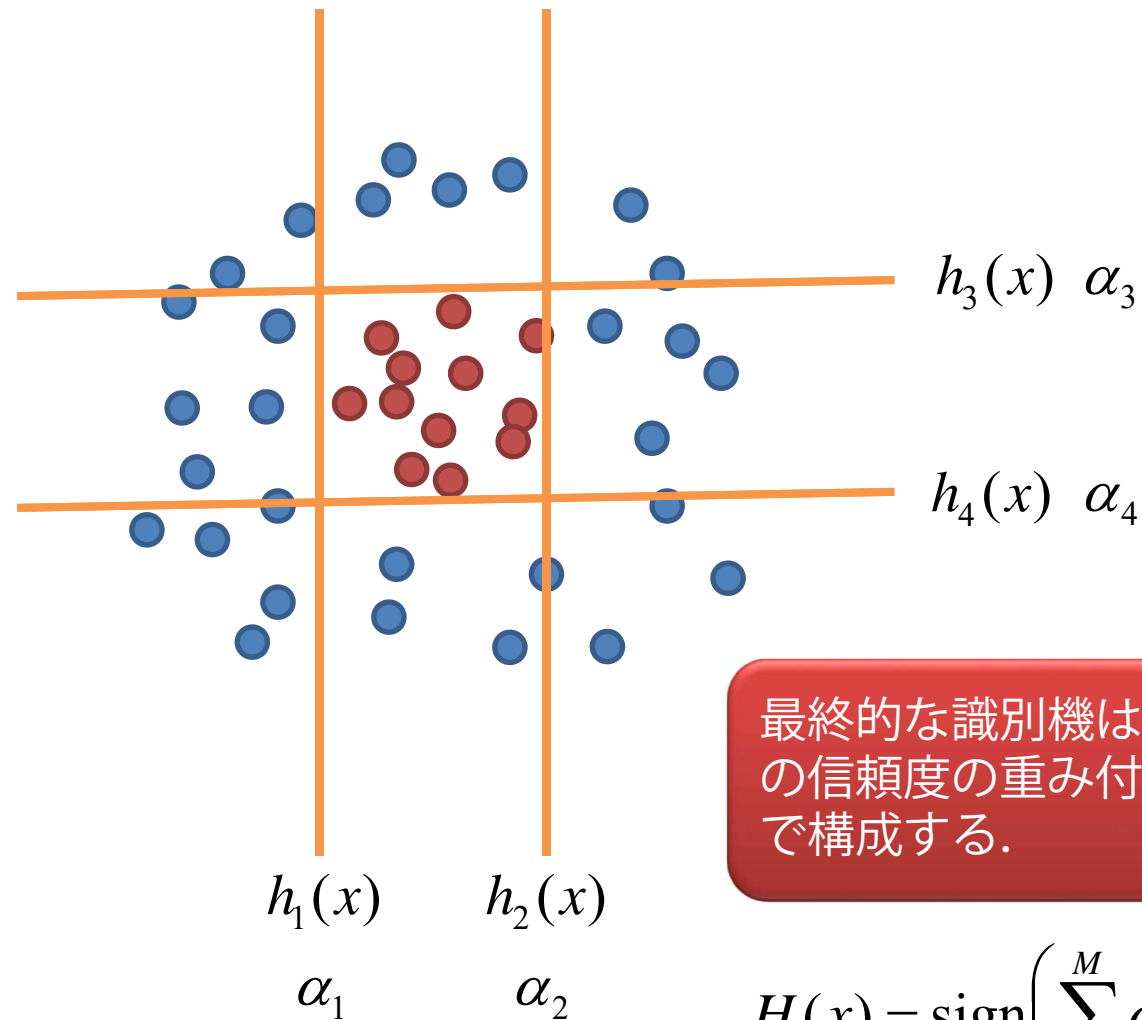
AdaBoostの学習過程



AdaBoostの学習過程



AdaBoostの学習過程



最終的な識別機は各識別機の信頼度の重み付け多数決で構成する。

$$H(x) = \text{sign} \left(\sum_{m=1}^M \alpha_m h_m(x) \right)$$

AdaBoostのアルゴリズム

1. 各訓練サンプルに等価な重みを与える

$$\left\{ w_n^{(1)} = \frac{1}{N} \right\}_{n=1}^N$$

2. For $m=1:M$

1. 訓練エラーを最小とする弱学習器を選択する

$$J_m = \sum_{n=1}^N w_n^{(m)} I(h_m(x_n) \neq y_n)$$

$$I(h_m(x_n) \neq y_n) = \begin{cases} 1 & \text{if } h_m(x_n) \neq y_n \\ 0 & \text{otherwise} \end{cases}$$

2. 現在の弱学習器によって誤識別された訓練サンプルの重みを増加

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(h_m(x) \neq y_n)}{\sum_{n=1}^N w_n^{(m)}} \quad w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(h_m(x_n) \neq y_n)\}$$
$$\alpha_m = \ln \left(\frac{1 - \epsilon_m}{\epsilon_m} \right)$$

3. 全ての弱学習器を重み付け線形結合し、最終的な識別器を構成する
 - 各弱学習器の重み付けは、識別精度に比例

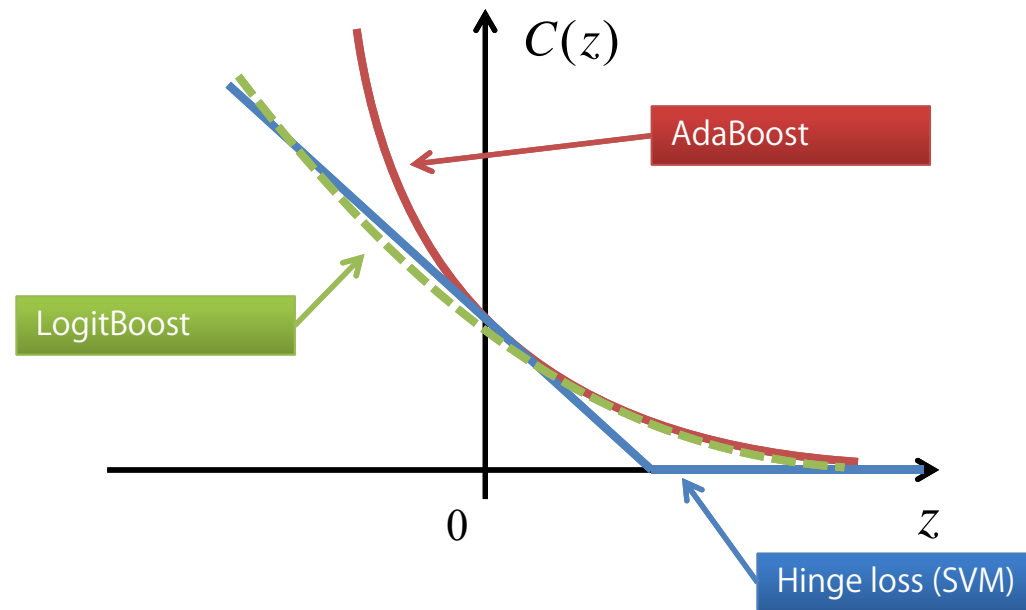
$$H(x) = \text{sign} \left(\sum_{m=1}^M \alpha_m h_m(x) \right)$$

Boostingの統一的解釈

- AnyBoost
 - L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting Algorithms as Gradient Descent. NIPS, 2000.
 - 期待値損失の座標降下法による最小化
 - 下記の日本語の参考書ではRiskBoostとして紹介されている.
 - 金森敬文. ブースティング - 学習アルゴリズムの設計技法 (知能情報科学シリーズ). 森北出版, 2006.

様々なBoosting手法

- AnyBoostの損失関数を変更することで様々なBoosting手法を設計できる.



Algorithm	Cost function	Step size
AdaBoost [9]	$e^{-yF(x)}$	Line search
ARC-X4 [2]	$(1 - yF(x))^5$	$1/t$
ConfidenceBoost [19]	$e^{-yF(x)}$	Line search
LogitBoost [12]	$\ln(1 + e^{-yF(x)})$	Newton-Raphson