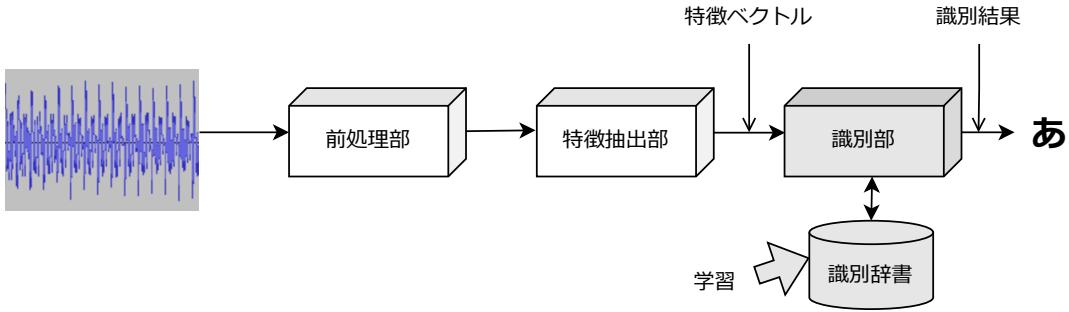
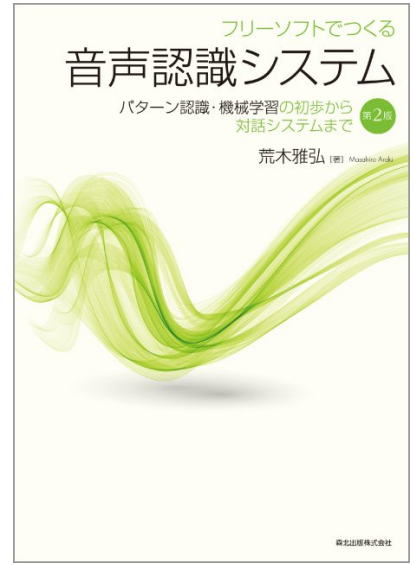
4. パターンを識別しよう



- 4.1 NN 法の定式化と問題設定
- 4.2 パーセプトロンの学習規則
- 4.3 区分的線形識別関数とk-NN法



- 荒木雅弘:『フリーソフトでつくる音声認識システム(第2版)』(森北出版, 2017年)
- スライドとJupyter notebook
- サポートページ

4.1 NN 法の定式化と問題設定

4.1.1 「もっとも近い」の定義 (1/2)

- 識別対象のクラス: $\omega_1, \ldots, \omega_c$
- プロトタイプ
  - 各クラスの代表となる点

$$\boldsymbol{p}_i = (p_{i1}, \ldots, p_{id})^T \quad (i = 1, \ldots, c)$$

- 識別したい入力データ

$$\boldsymbol{x} = (x_1, \ldots, x_d)^T$$

4.1.1「もっとも近い」の定義（2/2）

- 入力ベクトルとプロトタイプとの距離

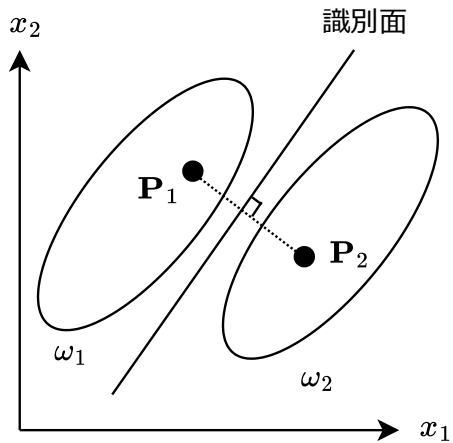
$$D(\boldsymbol{x}, \boldsymbol{p}_i) = \sqrt{(x_1 - p_{i1})^2 + \cdots + (x_d - p_{id})^2}$$

- NN法の判定式

$$\underset{i}{\operatorname{argmin}}\, D(\boldsymbol{x}, \boldsymbol{p}_i) = k \quad \Rightarrow \quad \boldsymbol{x} \in \omega_k$$

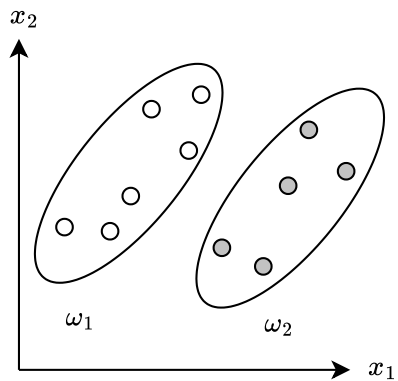
4.1.2 プロトタイプと識別面の関係 (1/2)

- 特徴空間の分割
  - 2次元特徴の2クラス問題($d = 2, c = 2$)を考える
  - クラスを分離する境界 = プロトタイプから等距離にある領域
    - 決定境界あるいは識別面とよぶ
    - 2次元のNN法では2つのプロトタイプの垂直2等分線(多次元では超平面)

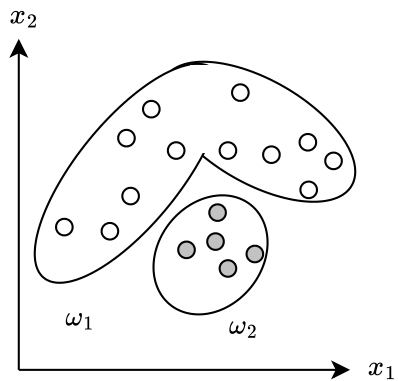


4.1.2 プロトタイプと識別面の関係（2/2）

- 線形分離可能性
  - 直線(超平面)で2つのクラスが誤りなく分割できる場合を線形分離可能とよぶ



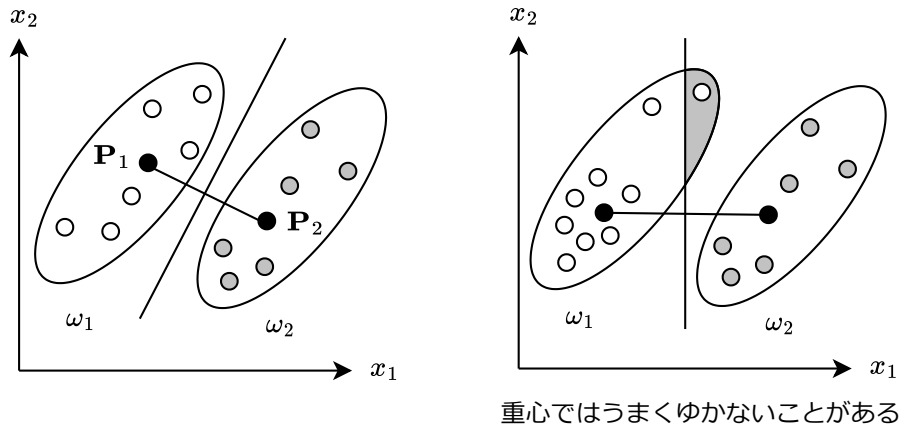
線形分離可能なデータ



線形分離不可能なデータ

4.1.3 プロトタイプの位置の決め方

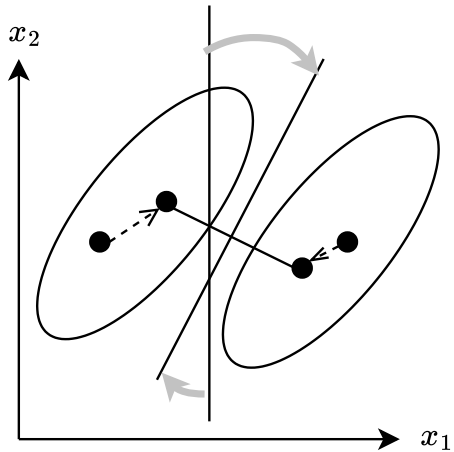
- クラスを代表するプロトタイプの設定法
  - 例:プロトタイプをクラスの重心にしたとき



重心ではうまくゆかないことがある

4.2 パーセプトロンの学習規則 (1/2)

- 識別面の学習
  - 線形分離可能なデータに対して、識別誤りが生じない位置にプロトタイプを設定する



4.2 パーセプトロンの学習規則（2/2）

- NN法における学習とは
  - プロトタイプの正しい位置を自動的に求めること
- 学習パターン
  - 識別部設計（特徴空間の分割）用に収集されたパターン
- 学習の目標
  - 学習パターンをすべて正しく識別できるような識別面を見つけること

4.2.1 識別関数の設定（1/2）

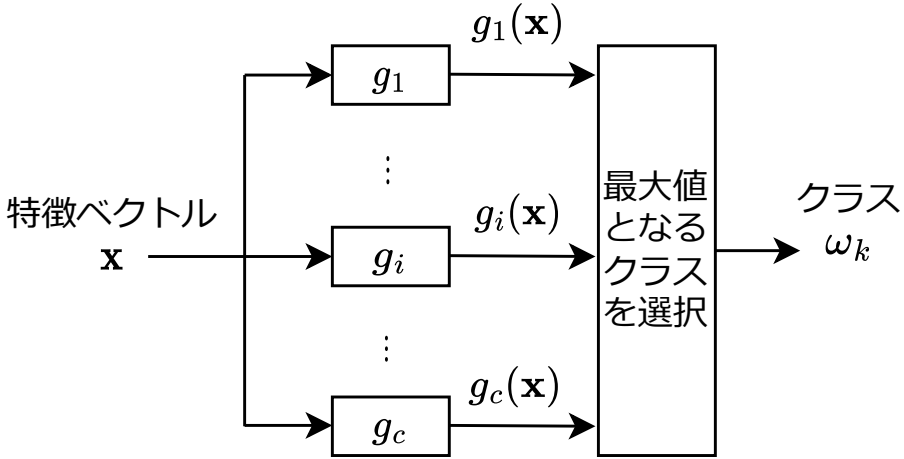
- 1クラス1プロトタイプのNN法の定式化
  - クラス：$\omega_1, \ldots, \omega_c$
  - プロトタイプ：$\boldsymbol{P}_1, \ldots, \boldsymbol{P}_c$
  - 入力パターン：$\boldsymbol{x}$（特徴ベクトル）
  - NN法：$D(\boldsymbol{x}, \boldsymbol{P}_i) = \|\boldsymbol{x} - \boldsymbol{P}_i\|$ を最小にする i を探す

$$\begin{aligned}\|\boldsymbol{x} - \boldsymbol{P}_i\|^2 &= \|\boldsymbol{x}\|^2 - 2\boldsymbol{P}_i^T\boldsymbol{x} + \|\boldsymbol{P}_i\|^2 \\ &= \|\boldsymbol{x}\|^2 - 2(\boldsymbol{P}_i^T\boldsymbol{x} - \frac{1}{2}\|\boldsymbol{P}_i\|^2)\end{aligned}$$

$\rightarrow$ 識別関数 $g_i(\boldsymbol{x}) = \boldsymbol{P}_i^T\boldsymbol{x} - \frac{1}{2}\|\boldsymbol{P}_i\|^2$ を最大にする i を探す

4.2.1 識別関数の設定（2/2）

- NN法による識別部の実現



4.2.2 識別関数とパーセプトロン (1/2)

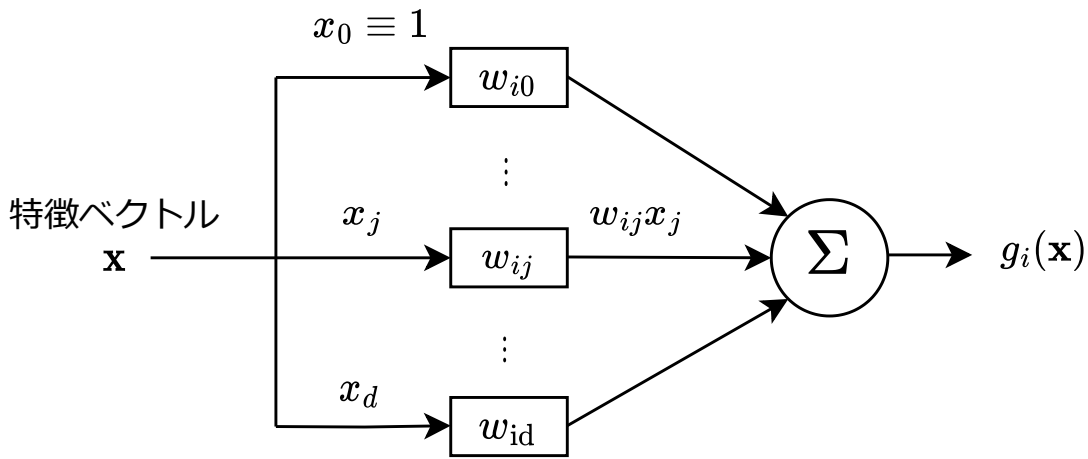
- 線形識別関数
  - 識別関数の係数を $p_{ij} = w_{ij} \quad (j = 1, \ldots, d), \quad -\frac{1}{2}\|\boldsymbol{p}_i\|^2 = w_{i0}$ と置き換える

$$
\begin{aligned}
g_i(\boldsymbol{x}) &= \sum_{j=1}^{d} w_{ij} x_j + w_{i0} \\
&= \sum_{j=0}^{d} w_{ij} x_j \quad (x_0 \equiv 1) \\
&= \boldsymbol{w}_i^T \boldsymbol{x}
\end{aligned}
$$

  - $\boldsymbol{w}_i, \boldsymbol{x}$ は $d+1$ 次元

4.2.2 識別関数とパーセプトロン（2/2）

- 線形識別関数の計算法
  - この計算機構は神経細胞の振舞いを単純化したモデルであり、パーセプトロンとよばれる



4.2.3 2クラスの識別関数の学習（1/5）

- 多クラスの線形識別関数の学習
  - 学習パターン全体：χ
  - クラス ω_i に属する学習パターンの集合 χ_i の全ての要素 $\boldsymbol{x}$ に対して、以下の条件が成り立つように重み w_i を決定する

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad (j = 1, \ldots, c,\ j \neq i)$$

4.2.3 2クラスの識別関数の学習（2/5）

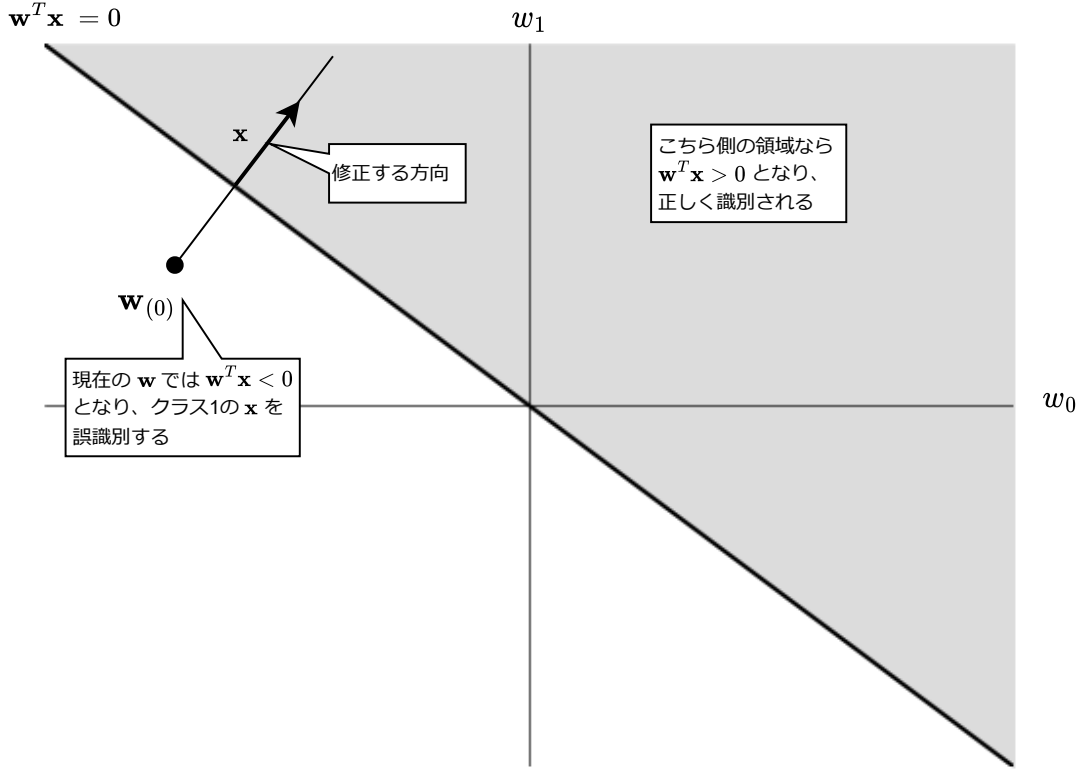
- 2クラスの場合
  - 識別関数を $g(\boldsymbol{x}) = g_1(\boldsymbol{x}) - g_2(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ とし、出力の正負がクラスを表すような $\boldsymbol{w}$ を求める

$$g(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} \begin{cases} > 0 & (\boldsymbol{x} \in \chi_1) \\ < 0 & (\boldsymbol{x} \in \chi_2) \end{cases}$$

  - クラス1を正例、クラス2を負例とよぶ

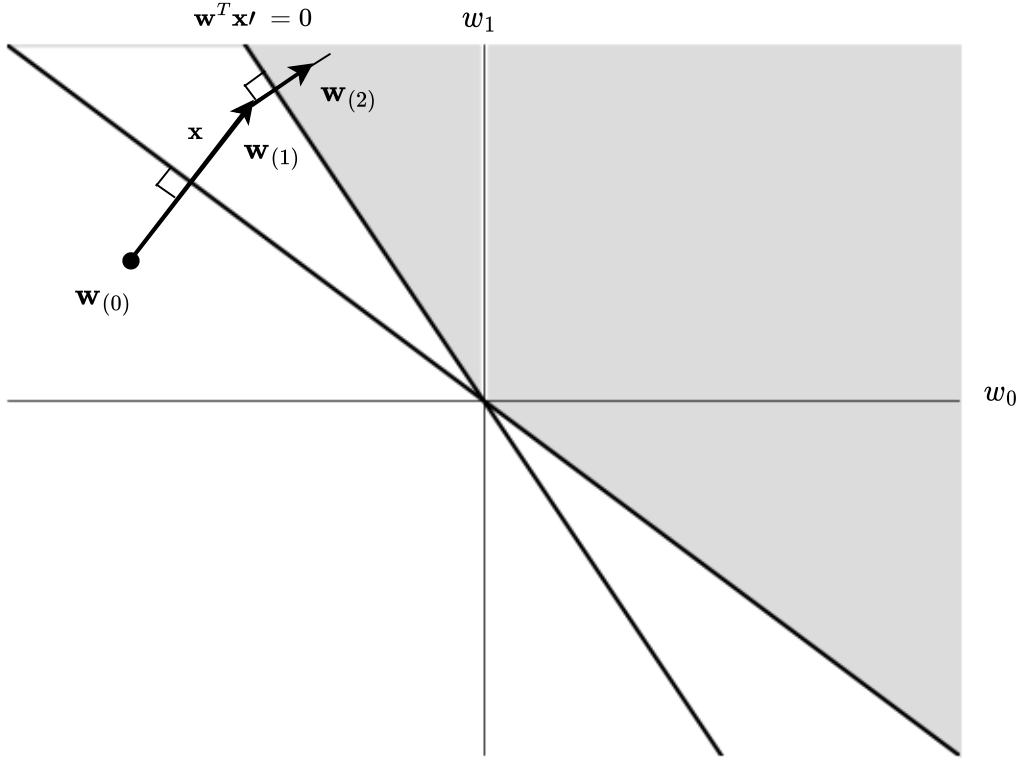
4.2.3 2クラスの識別関数の学習（3/5）

- 重み空間での重みの修正



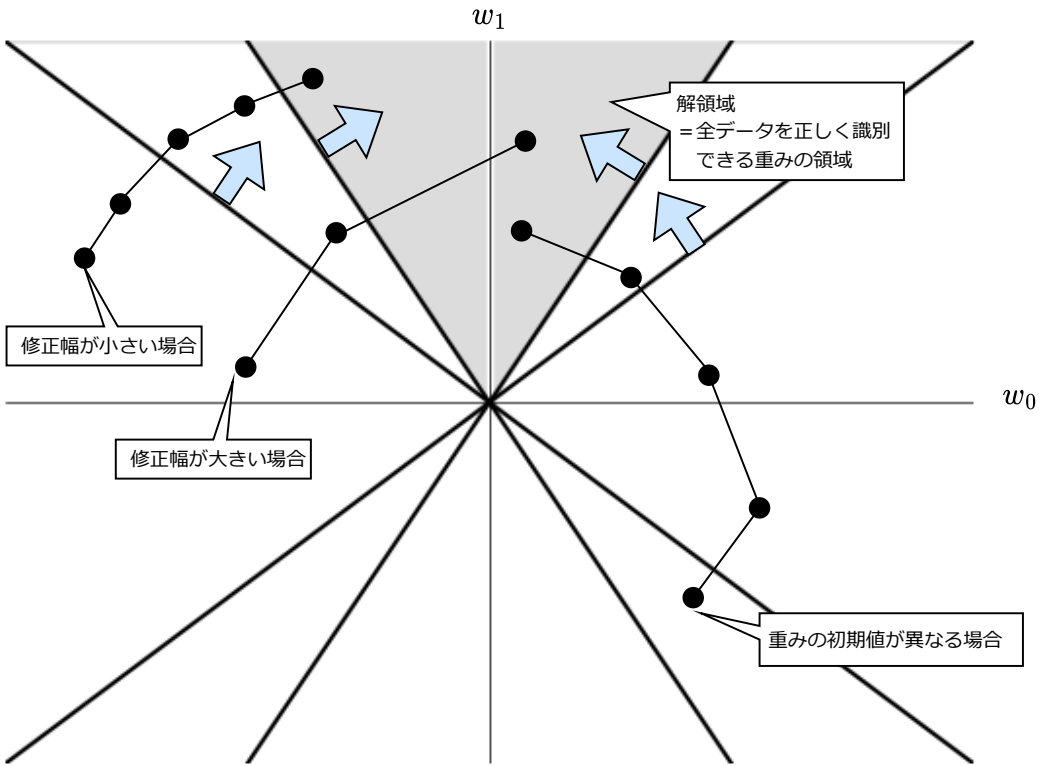
4.2.3 2クラスの識別関数の学習（4/5）

- 別の学習データに対する重みの修正



4.2.3 2クラスの識別関数の学習（5/5）

- 解領域への重みの修正プロセス



4.2.4 パーセプトロンの学習アルゴリズム（1/2）

- パーセプトロンの学習規則

1. $\boldsymbol{w}$ の初期値を適当に決める
2. 学習データからひとつ $\boldsymbol{x}$ を選び、$g(\boldsymbol{x})$ を計算
3. 誤識別が起きたときのみ、以下の式を用いて $\boldsymbol{w}$ を修正する。ρ は学習係数

   $\boldsymbol{w}' = \boldsymbol{w} + \rho\boldsymbol{x}$（正例を負例と誤ったとき）
   $\boldsymbol{w}' = \boldsymbol{w} - \rho\boldsymbol{x}$（負例を正例と誤ったとき）
4. 2,3を全ての学習データについて繰り返す
5. すべて識別できたら終了。そうでなければ2へ

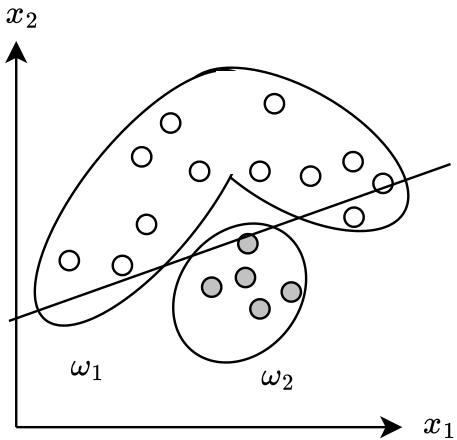
4.2.4 パーセプトロンの学習アルゴリズム（2/2）

- パーセプトロンの収束定理
  - 学習データが線形分離可能であれば、パーセプトロンの学習規則は有限回の繰り返しで線形識別面を得ることができる
- 学習係数 ρ の設定
  - 大きすぎると重みの値が振動する
  - 小さすぎると収束に時間がかかる

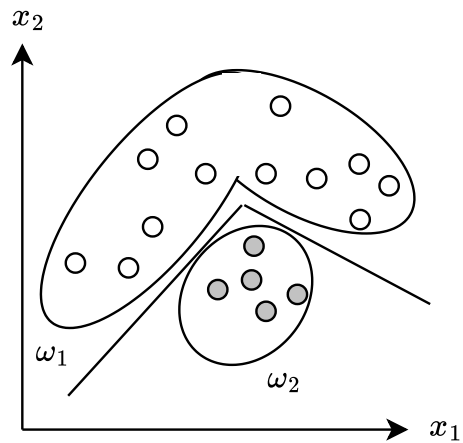
4.3 区分的線形識別関数とk-NN法

4.3.1 平面で区切れない場合

- 区分的線形識別関数を用いる



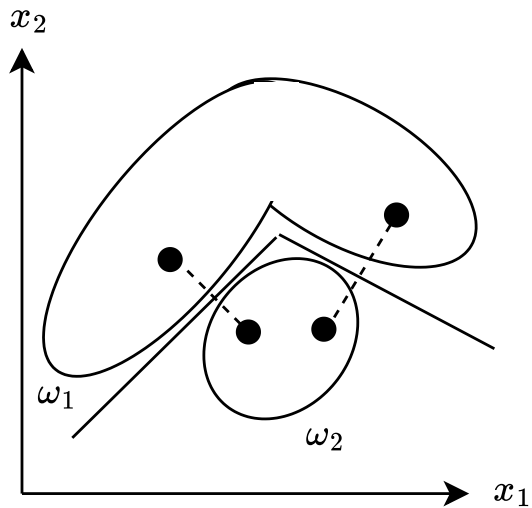
線形分離不可能なデータ



区分的線形識別関数を用いた場合

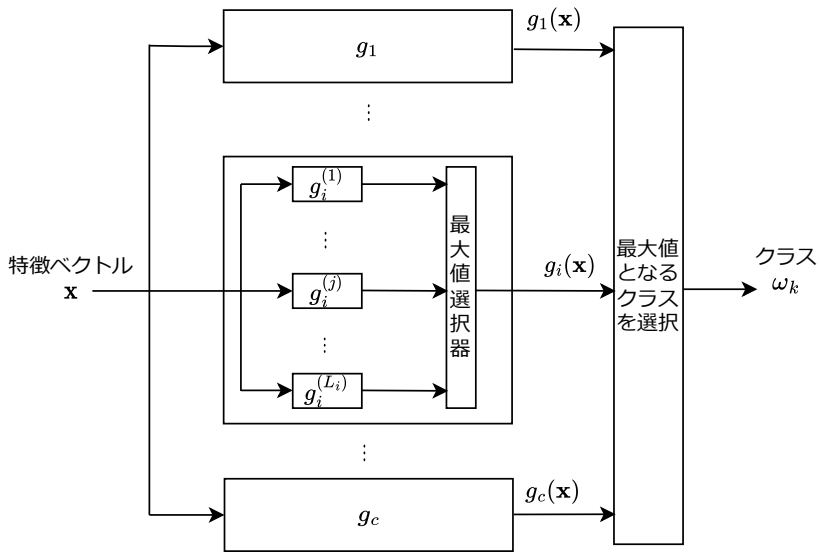
4.3.2 区分的線形識別関数の実現（1/2）

- 区分的線形識別関数の決め方
  - 各クラス複数のプロトタイプを設定



4.3.2 区分的線形識別関数の実現（2/2）

- 区分的線形識別関数の定義
  - クラス ω_i の識別関数 $g_i(\boldsymbol{x})$ を L_i 個の副次(線形)識別関数 $g_i^{(l)}(\boldsymbol{x}) \quad (l = 1, \ldots, L_i)$ の最大値としてあらわす

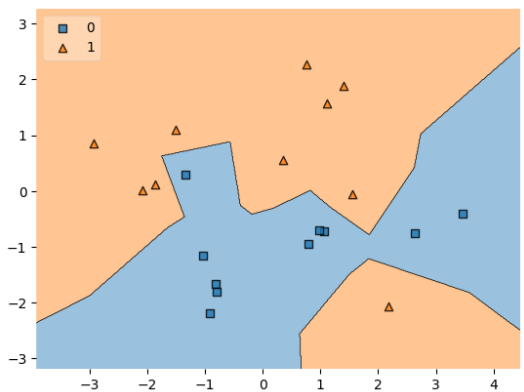


4.3.3 区分的線形識別関数の識別能力と学習

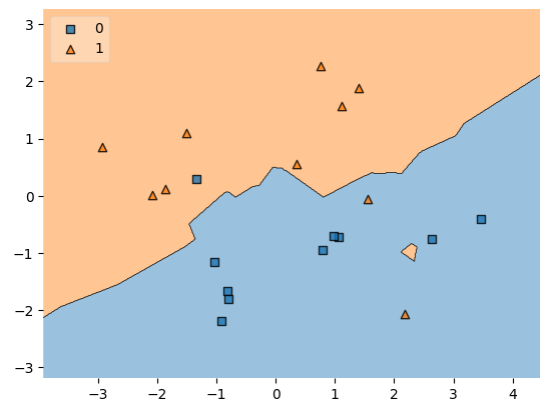
- 能力
  - 複雑な非線形決定境界でも、任意の精度で近似可能
- 学習
  - 副次識別関数の個数と、各関数の重みの両方を学習しなければならない
  - パーセプトロンの学習規則が適用できず、一般に学習は難しい

4.3.4 学習をあきらめるのも一手 k-NN法 (1/2)

- k-NN法とは
  - 全ての学習データをプロトタイプとする
  - 入力に近い順から k 個のプロトタイプのクラスを調べ、多数決を取る
    - k が大きいと識別面がなめらかになる
  - 実験の際のベースラインとして用いられる



k=1



k=3

4.3.4 学習をあきらめるのも一手 k-NN法 (2/2)

- k-NN法の特徴
  - 非線形性を示すデータにも対処できる可能性がある
- k を大きくすると識別境界が滑らかになる
- k-NN法の(かつての)問題点
  - 記憶容量
  - 計算時間

    → 現在ではあまり問題にならない

まとめ

- NN 法の定式化と問題設定
  - NN法は各クラスに1つプロトタイプを設定し、入力との距離が最小となるクラスに識別する
  - プロトタイプの位置を決めることは線形識別面を決めることと同じ
- パーセプトロンの学習規則
  - 線形識別面を決めることは、線形識別関数の重みを調整することと同じ
  - パーセプトロンの学習規則は線形分離可能なデータに対して、有限時間で正しい重みを求めることができる
- 区分的線形識別関数とk-NN法
  - 線形識別不可能な学習データに対して、区分的線形識別関数が適用可能だが学習が難しい
  - 学習を行わずに全学習データをプロトタイプとするk-NN法がある
- Jupyter notebook