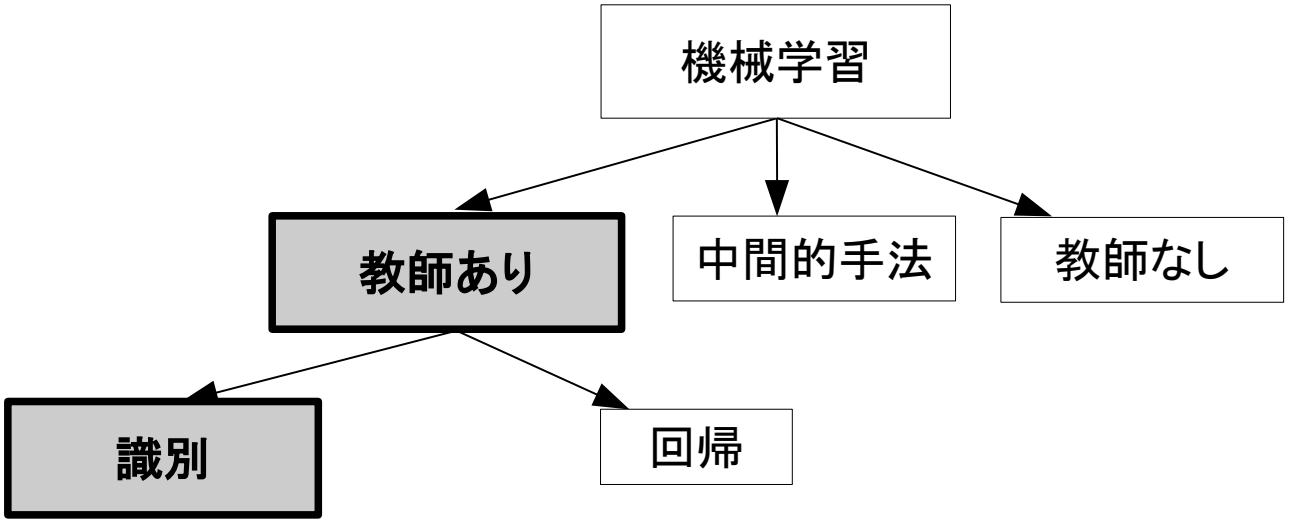
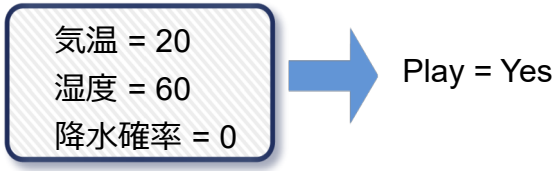
Section 3

- 識別2 (5～7章)



- 数値特徴



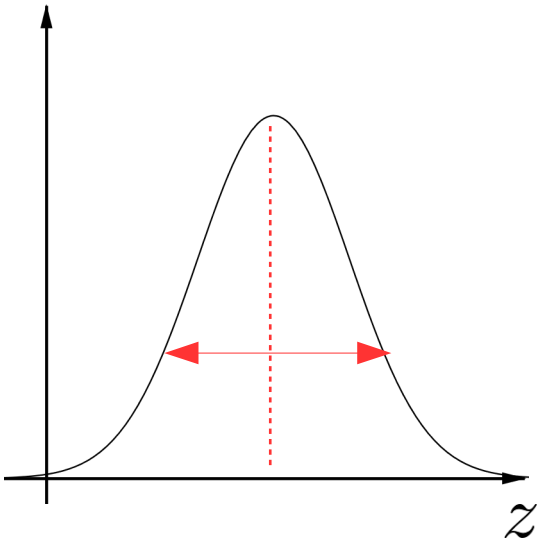
5.2 数値特徴に対するベイズ識別

5.2.1 数値特徴に対するナイーブベイズ識別

$$C_{NB} = \arg\max_i P(\omega_i) \prod_{j=1}^{d} p(x_j|\omega_i)$$

- 確率密度関数 $p(x_j|\omega_i)$ の推定
  - 正規分布を仮定

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$$



  - 平均 μ と分散 σ を最尤推定
    - それぞれ、学習データの平均と分散になる

5.3.1 識別モデルの考え方

- 事後確率を直接求める

正例で
大きな値

負例で
大きな値

あまり
関係ない

$$\boldsymbol{x} = (x_1, \quad x_2, \quad x_3, \quad \ldots, \quad x_d)$$

w_1
正の
大きな値

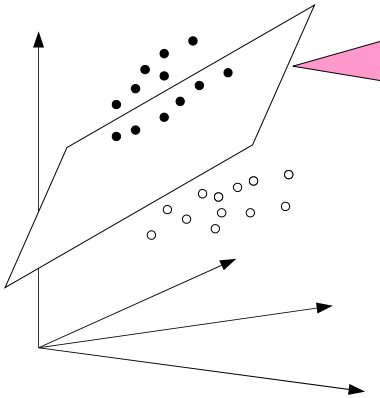
w_2
負の
大きな値

w_3
小さな値

正例／負例
の閾値

$$w_1 x_1 + w_2 x_2 + w_3 x_3 + \cdots + w_d x_d + w_0$$

この値が正なら正例、
負なら負例となるように
重み w を学習する



この平面を
求めている
ことになる

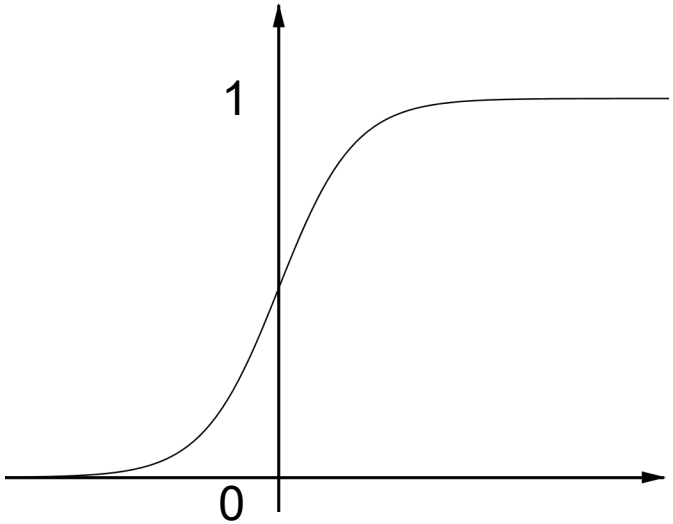
確率と対応づけるには？

5.3.1 識別モデルの考え方

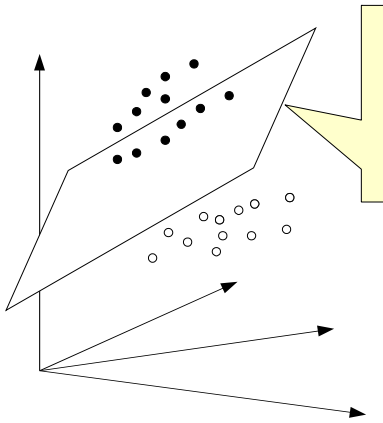
- ロジスティック識別
  - 入力が正例である確率

$$P(\oplus|\boldsymbol{x}) = \frac{1}{1+\exp(-(\boldsymbol{w}\cdot\boldsymbol{x}+w_0))}$$

-∞ ～ +∞ の値域を持つものを、順序を変えずに 0 ～ 1 にマッピング



シグモイド関数



この平面上は
$\boldsymbol{w}\boldsymbol{x}+w_0=0$
⇔ P(+|x)=0.5

5.3.2 ロジスティック識別器の学習

- 最適化対象＝モデルが学習データを生成する確率

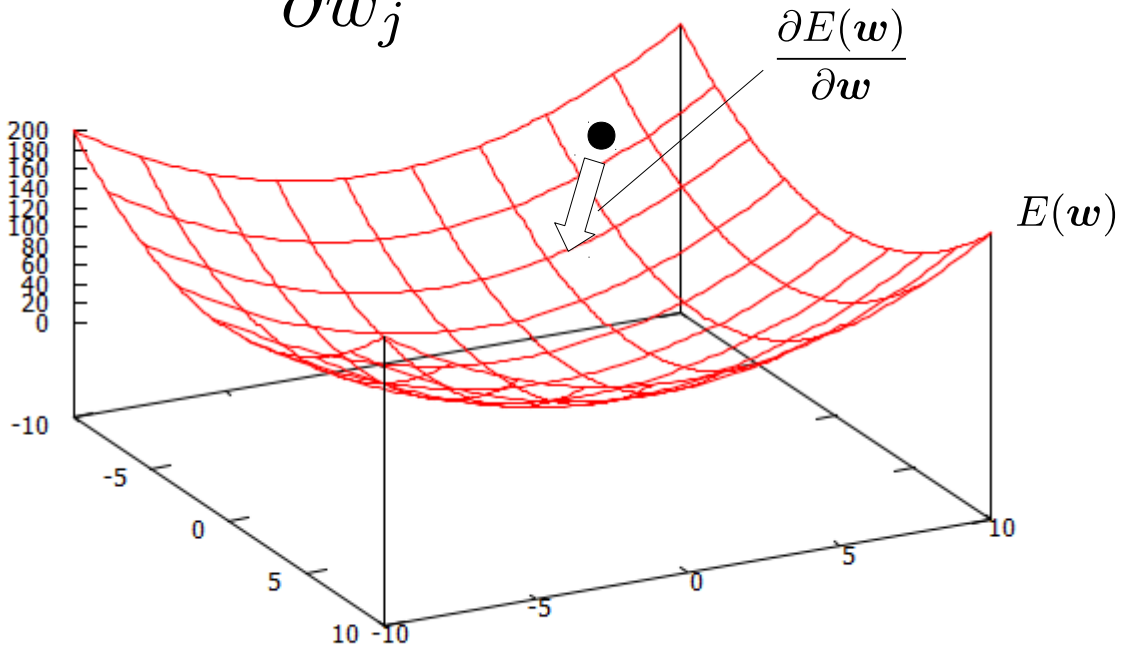
$$E(\boldsymbol{w}) = -\log P(D|\boldsymbol{w}) = -\log \prod_{\boldsymbol{x}_i \in D} o_i^{y_i} (1-o_i)^{(1-y_i)}$$

  - $E(\boldsymbol{w})$ を最急勾配法で最小化

$$w_j \leftarrow w_j - \eta \frac{\partial E(\boldsymbol{w})}{\partial w_j}$$

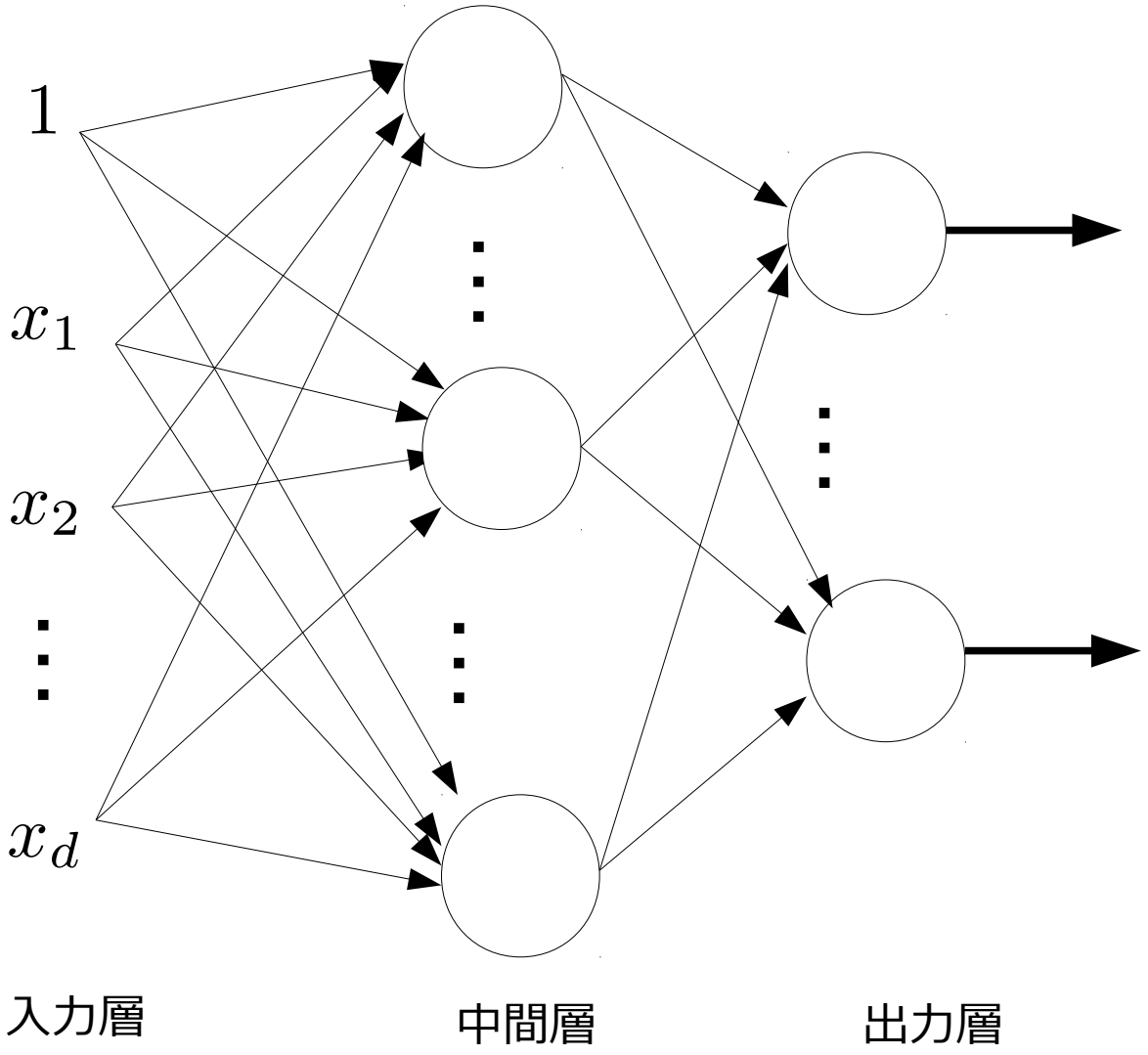
$o = P(\oplus|\boldsymbol{x})$
$y = 0 \ \ or \ \ 1$

正解ラベル



6.4 ニューラルネットワーク

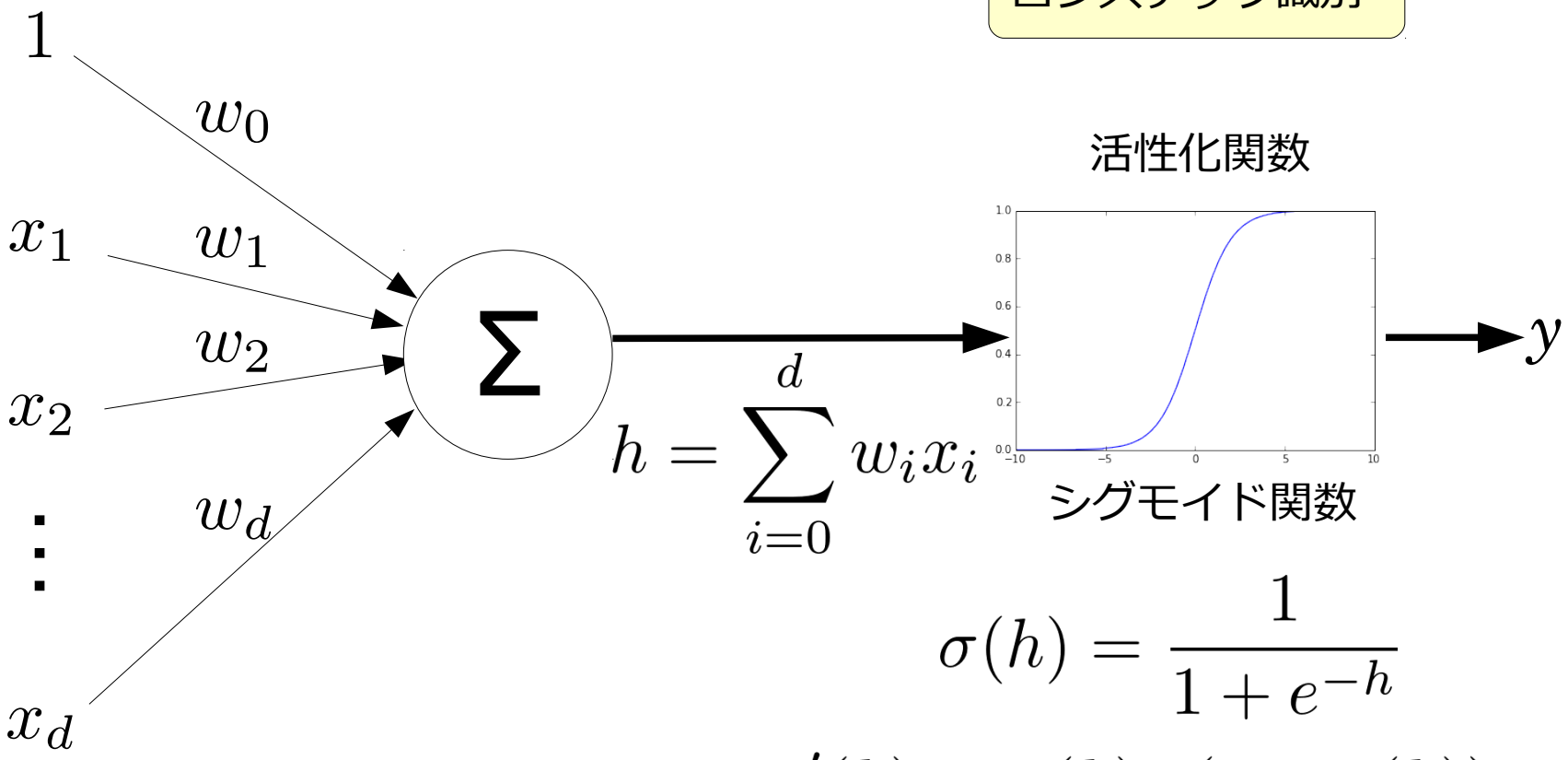
- 3層のフィードフォワードネットワーク



6.3 最小二乗法による学習

- シグモイド関数の適用
  - 多層の誤差修正に対応するために、勾配計算の際に微分可能な活性化関数を用いる

ロジステック識別



$$\sigma(h) = \frac{1}{1 + e^{-h}}$$

$$\sigma'(h) = \sigma(h) \cdot (1 - \sigma(h))$$

6.4 ニューラルネットワーク

- フィードフォワードネットワークのユニット
  - 中間層の活性化関数：シグモイド関数
  - 出力層の活性化関数：シグモイド関数または softmax 関数

$$f(h_i) = \frac{\exp(h_i)}{\sum_{j=1}^{c} \exp(h_j)}$$

6.4 ニューラルネットワーク

- 誤差逆伝播法

1.リンクの重みを小さな初期値に設定

2.個々の学習データ $(\boldsymbol{x}_i, y_i)$に対して以下繰り返し

- 入力 $\boldsymbol{x}_i$ に対するネットワークの出力 $\boldsymbol{o}_i$ を計算

a)出力層の k 番目のユニットに対してエラー量 δ 計算

$$\delta_k \leftarrow o_k(1-o_k)(y_k - o_k)$$

b)中間層の h 番目のユニットに対してエラー量 δ 計算

$$\delta_k \leftarrow o_k(1-o_k) \sum_{k \in outputs} w_{kh}\delta_k$$

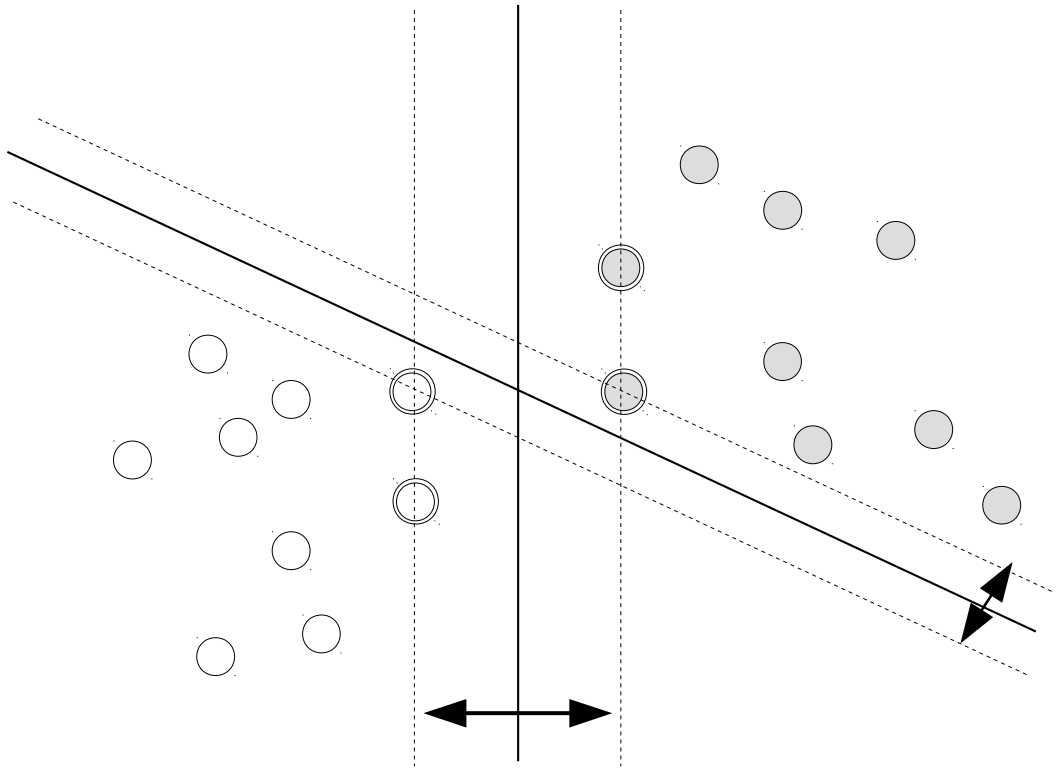
c)重みの更新

$$w'_{ji} \leftarrow w_{ji} + \eta\delta_j x_{ji}$$

7. 識別 - サポートベクトルマシン -

- マージンを最大化する識別面を求める

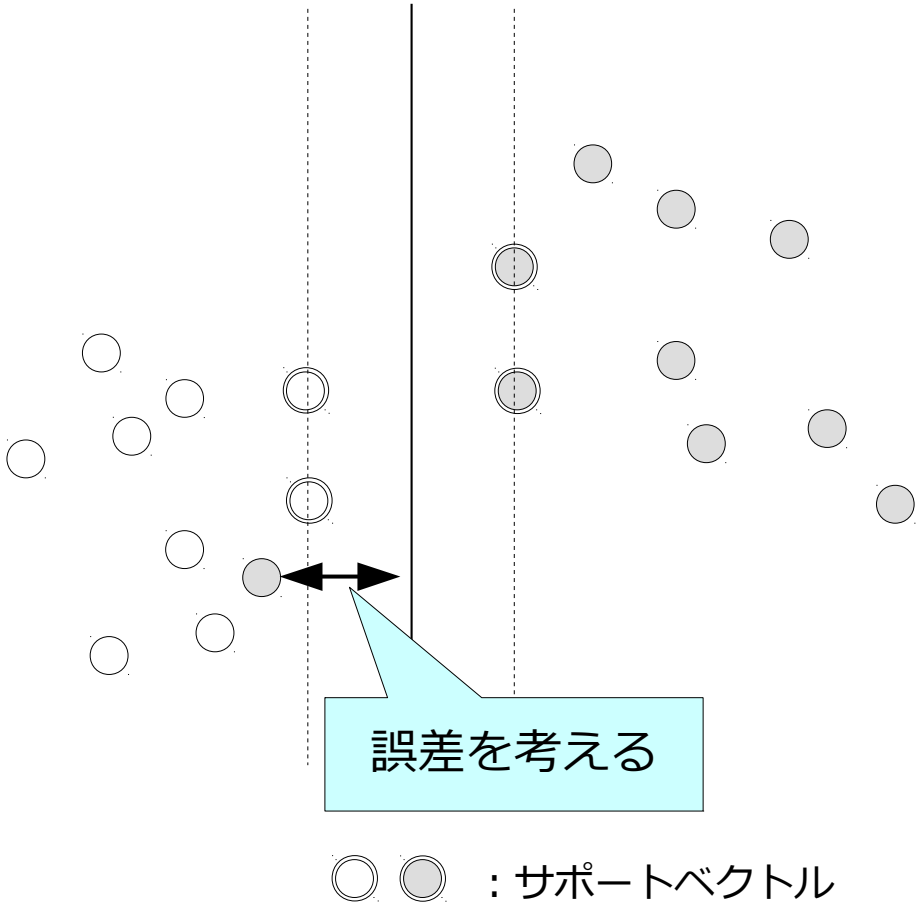
識別面と、最も近いデータとの距離



：サポートベクトル

7.2 ソフトマージンによる誤識別データの吸収

- 少量のデータが線形分離性を妨げている場合



7.2 ソフトマージンによる誤識別データの吸収

- スラック変数 ξ_i の導入

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + w_0) \geq 1 - \xi_i \quad i = 1, \ldots, N$$

- 最小化問題の修正

$$\min(\frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{N} \xi_i)$$

スラック変数も
小さい方がよい

- 計算結果
  - α_i の 2 次計画問題に $0 \leq \alpha_i \leq C$ が加わるだけ

7.2 ソフトマージンによる誤識別データの吸収

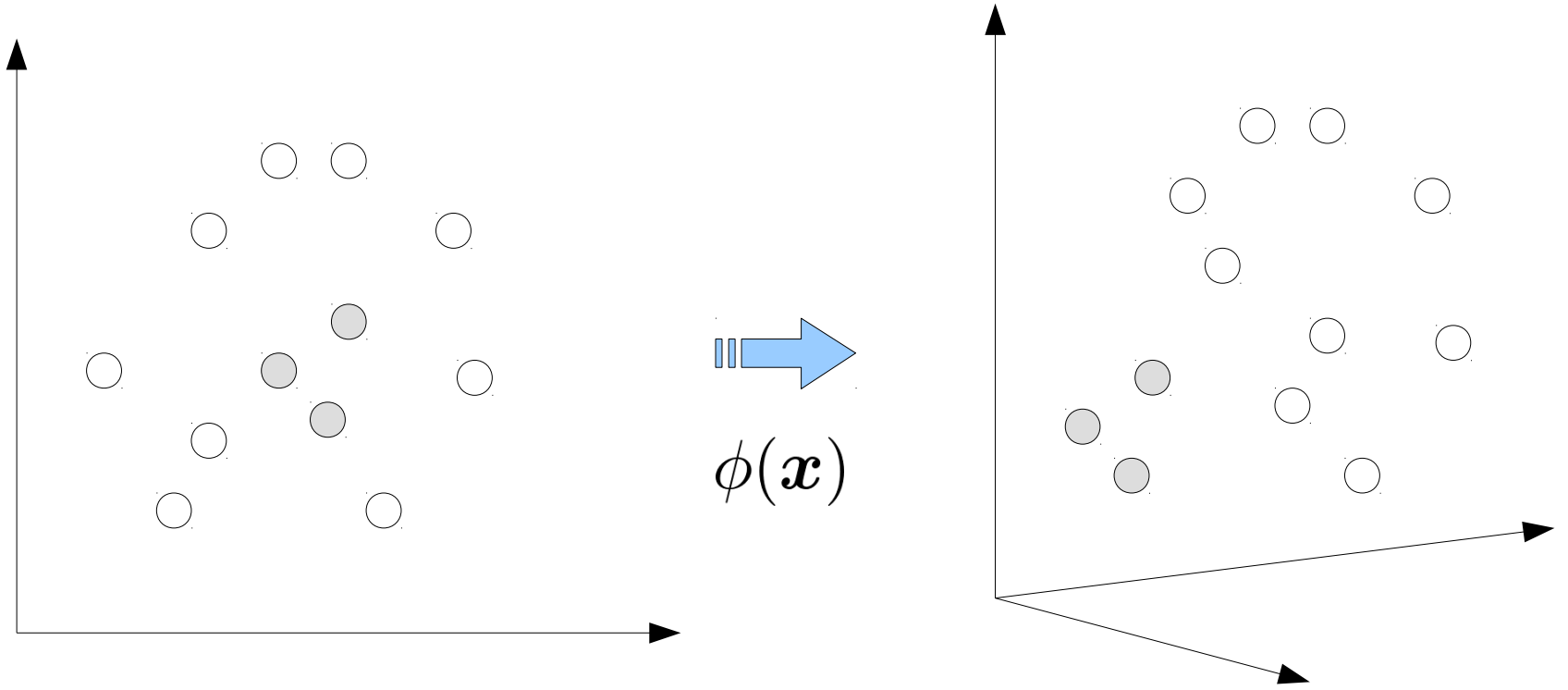
- C: エラー事例に対するペナルティ
  - 大きな値：誤識別データの影響が大きい

    → 複雑な識別面
  - 小さな値：誤識別データの影響が小さい

    → 単純な識別面

7.3 カーネル関数を用いた SVM

- 特徴ベクトルの次元を増やす



2 次元で線形分離不可能なデータ

線形分離可能な 3 次元へ

ただし、元の空間でのデータ間の
距離関係は保持するように

7.3 カーネル関数を用いた SVM

- 非線形変換関数： $\phi(\boldsymbol{x})$
- カーネル関数

$$K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$$

2 つの引数値の
近さを表す

  - 元の空間での距離が変換後の空間の内積に対応
  - $\boldsymbol{x}$ と $\boldsymbol{x}'$ が近ければ $K(\boldsymbol{x}, \boldsymbol{x}')$ は大きい値

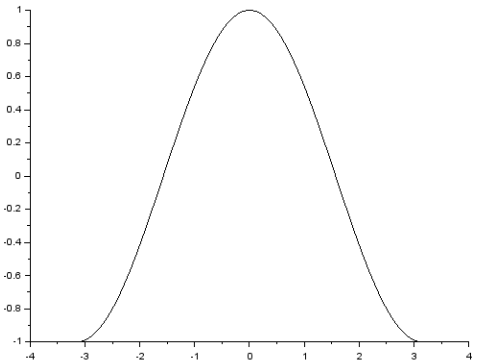
7.3 カーネル関数を用いた SVM

- カーネル関数の例（ scikit-learn の定義）
  - 線形　　$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$
    - 元の特徴空間でマージン最大の平面
  - 多項式　　$K(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + r)^d$
    - d 項の相関を加える
  - RBF　　$K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma ||\boldsymbol{x} - \boldsymbol{x}'||^2)$
    - γ の値：大→複雑　小→単純な識別面
  - シグモイド　$K(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\boldsymbol{x}^T \boldsymbol{x}' + r)$
    - ニューラルネットワークと似た振る舞いを実現

7.3 カーネル関数を用いた SVM

- 多項式カーネルの解釈

$$\boldsymbol{x}^T\boldsymbol{x}' = ||\boldsymbol{x}|| \cdot ||\boldsymbol{x}'|| \cdot \cos\theta$$



$y = \cos(x)$

- 多項式カーネルの展開例

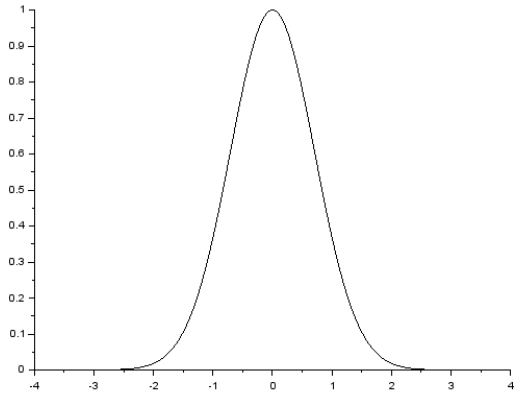
$$K(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = (\mathbf{x}^{(i)} \cdot \mathbf{x}^{(j)} + 1)^2$$
$$= (x_1^{(i)}x_1^{(j)} + x_2^{(i)}x_2^{(j)} + 1)^2$$
$$= (x_1^{(i)}x_1^{(j)})^2 + (x_2^{(i)}x_2^{(j)})^2 + 2x_1^{(i)}x_1^{(j)}x_2^{(i)}x_2^{(j)} + 2x_1^{(i)}x_1^{(j)} + 2x_2^{(i)}x_2^{(j)} + 1$$
$$= ((x_1^{(i)})^2, (x_2^{(i)})^2, \sqrt{2}x_1^{(i)}x_2^{(i)}, \sqrt{2}x_1^{(i)}, \sqrt{2}x_2^{(i)}, 1)$$
$$\cdot ((x_1^{(j)})^2, (x_2^{(j)})^2, \sqrt{2}x_1^{(j)}x_2^{(j)}, \sqrt{2}x_1^{(j)}, \sqrt{2}x_2^{(j)}, 1)$$

2 変数の相関を表す項

7.3 カーネル関数を用いた SVM

- RBF カーネルの解釈

$$e^{-||\boldsymbol{x}-\boldsymbol{x}'||^2}$$



$y = \exp(-x^2)$

- RBF カーネルの展開
  - e^x のマクローリン展開

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots$$

  - ガウシアンカーネルは無限級数の積で表されるので、無限次元ベクトルの内積と解釈できる

7.3 カーネル関数を用いた SVM

- 変換後の識別関数： $g(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x}) + w_0$
- SVM で求めた $\boldsymbol{w}$ の値を代入

$$g(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}_i) + w_0$$

$$= \sum_{i=1}^{N} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + w_0$$

非線形変換の
式は不要！！！

カーネルトリック

Section3 のまとめ

- 生成モデル：データの分布を示す関数を推定
- 識別モデル：データの境界を推定
  - 最急勾配法を用いて誤差最小のパラメータを求める
- ニューラルネットワーク
  - 複雑な非線形識別面を求める
  - 現在は、ディープニューラルネットワークの発展により識別問題解決の中心技術になった
- サポートベクトルマシン
  - 低次元で識別しにくいデータを、高次元の疎らなデータに変換し、マージン最大の線形識別面を求める手法