機械学習
2-3

工学院情報通信系
中原 啓貴

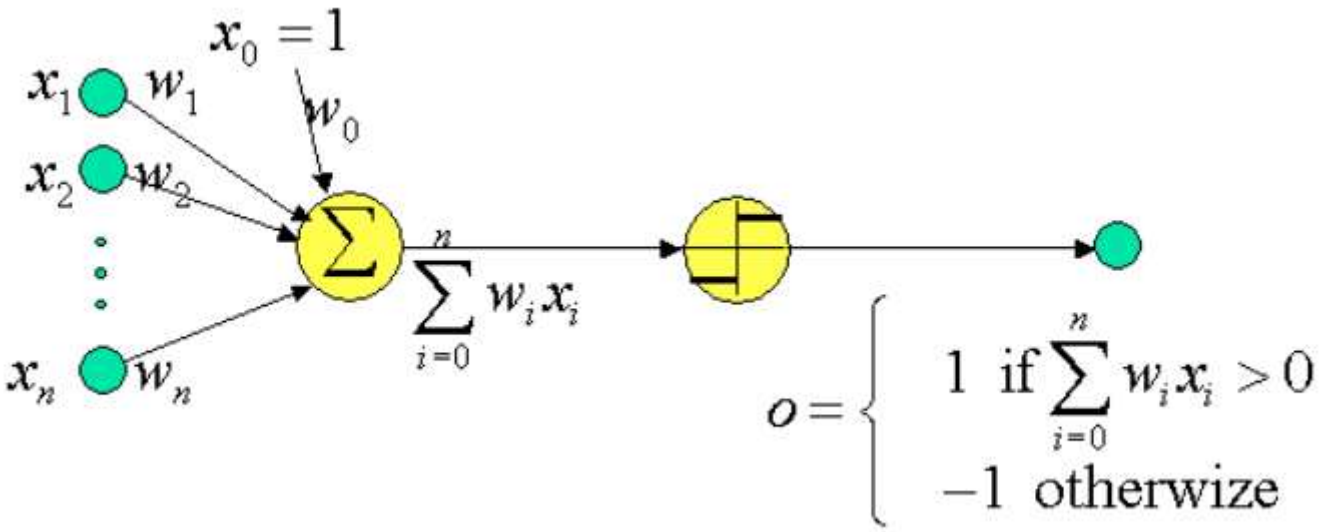
今日の内容

- scikit-learn機械学習ライブラリの使用法
- パーセプトロンの復習
- ロジスティック回帰
- サポートベクトルマシン
  - カーネルトリック
- 決定木学習
  - ランダムフォレスト

＋上記内容の演習

パーセプトロンによる2値分類

・ニューラルネットワークの原始的なモデル



ロジット関数

- パーセプトロンでの線形結合モデル＋シグモイド関数
- オッズ比: 事象の起こりやすさを定量化
- $\frac{p}{(1-p)}$,ここでpは正事象の確率変数
- ロジット関数: 入力x, $0 < x_i < 1$, $i \in N$を受け取り, 実数値を出力

$$logit\big(p(y=1|X)\big) = \sum_{i=0}^{m} w_i x_i = W^T X$$

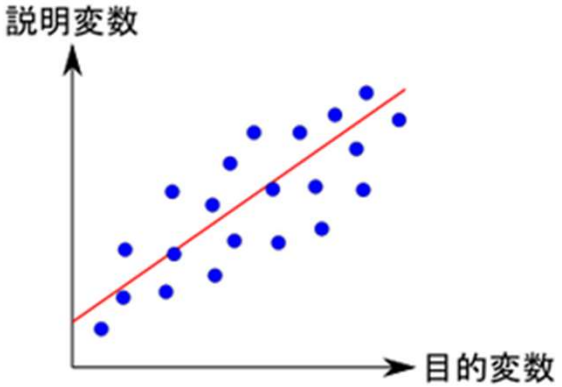
- ここで$p(y=1|X)$は特徴量 X が与えられた時のサンプルがクラス1に属するという条件付き確率

ロジスティック関数

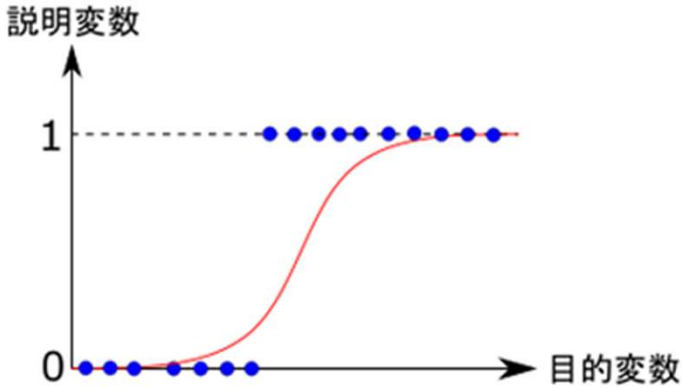
- サンプルが特定のクラスに属する確率を予測したい
  →ロジット関数の逆関数(**ロジスティック関数**)
  　シグモイド関数ともいう

$$sigmoid(Z) = \frac{1}{1+e^{-Z}},$$

ここで$Z = W^T X = w_0 + w_1 x_1 + \cdots + w_m x_m$



（線形）回帰分析
⇒「量的変数」の予測



ロジスティック回帰分析
⇒「発生確率」の予測

ロジステック回帰と分類

- 特徴量Xが重みWでパラメータ化されている
- シグモイド関数の出力: サンプルがクラス1に属している確率
- 例: 気象予報において,
- クラスラベル(晴れ($y = 0$), 雨($y = 1$))を学習した学習済みモデル$sigmoid(X) = 0.8$

  → 降水確率80% と回帰

  → 雨であると分類

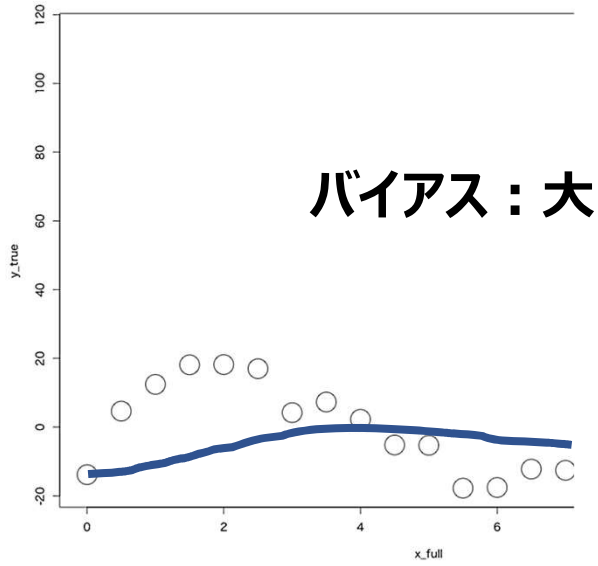
回帰を利用して分類問題に帰着

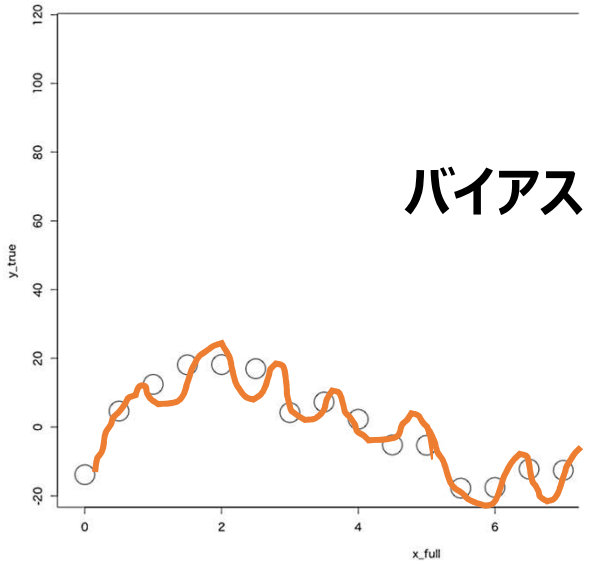
ロジスティック回帰の学習

- （興味があれば板書をメモ）
- S. Raschka, "Python機械学習プログラミング", pp.57, 3.3.2 ロジスティック関数の重みの学習
- 実際にはscikit-learnのAPIで容易にプログラミング可能

バイアスとバリアンス

- バイアス→モデルの単純さによる誤差
- バリアンス→学習データの違いによる誤差

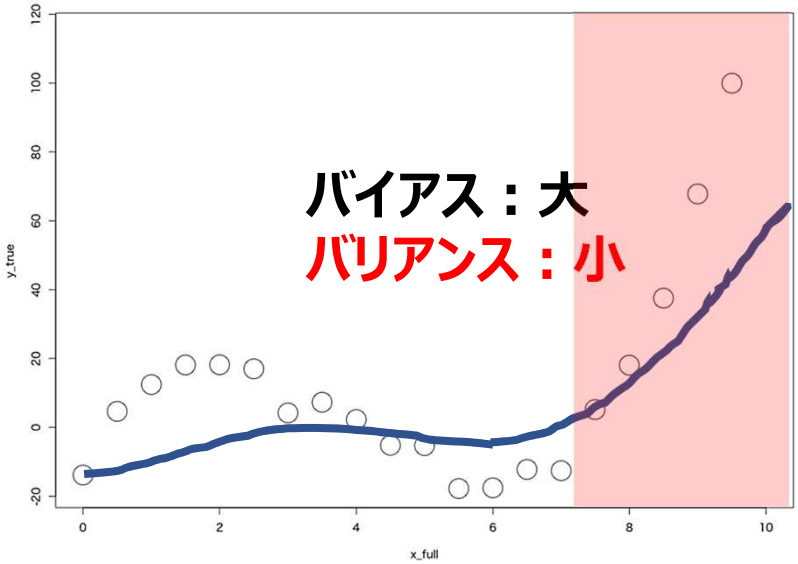


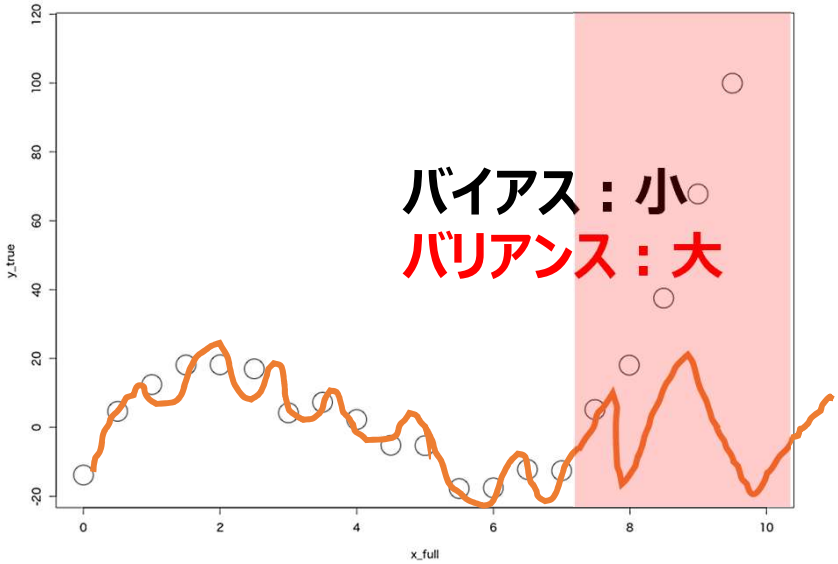


バイアスとバリアンス

- バイアス→モデルの単純さによる誤差
  - ノイズに強いが, 複雑な表現はできない: SVM, 最小2乗法
- バリアンス→学習データの違いによる誤差
  - 過学習しがち, 複雑な表現可能: Neural Network, 決定木

  →集団(アンサンブル)学習によりバリアンスを下げる





正則化によるモデルの調整

- 複雑さ(バリアス・バリアンス)を調整
  →汎化性能と可適合の調整
- L2正則化→荷重減衰(Weight decay)ともいう
- S. Raschka, "Python機械学習プログラミング", pp.64-65
- Scikit-learnのハイパーパラメータ"C"を調整

サポートベクトルマシン (Support Vector Machine: SVM)

- 広く使われている分類器(SVR回帰もあり)
- パーセプトロンの拡張
  - 「ノイズによる汎化能力の低下, 線形分離不可能な問題」の解決
  - マージン(超平面な決定境界)を最大化する
  - 複雑な超平面をカーネルトリックで実現
    →ということは、超平面のハイパーパラメータ探索…

SVMの学習:マージン最大化

$$w_0 + W^T X_{pos} = 1$$

$$w_0 + W^T X_{neg} = -1$$

上記より

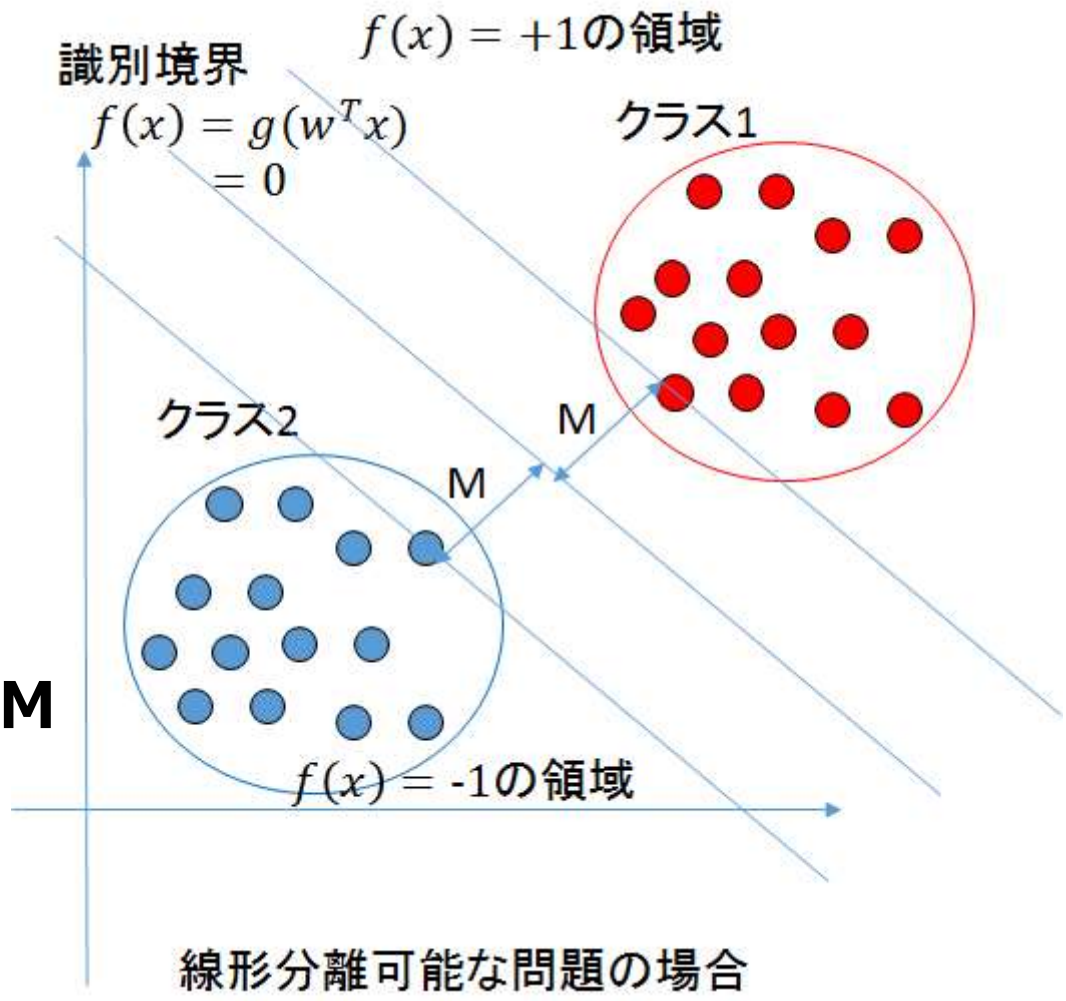
$$W^T\left(X_{pos} - X_{neg}\right) = 2$$

ここで

$$\|W\| = \sqrt{\sum_{j=1}^{m} w_j^2}$$

とすると，最大化したいマージンM

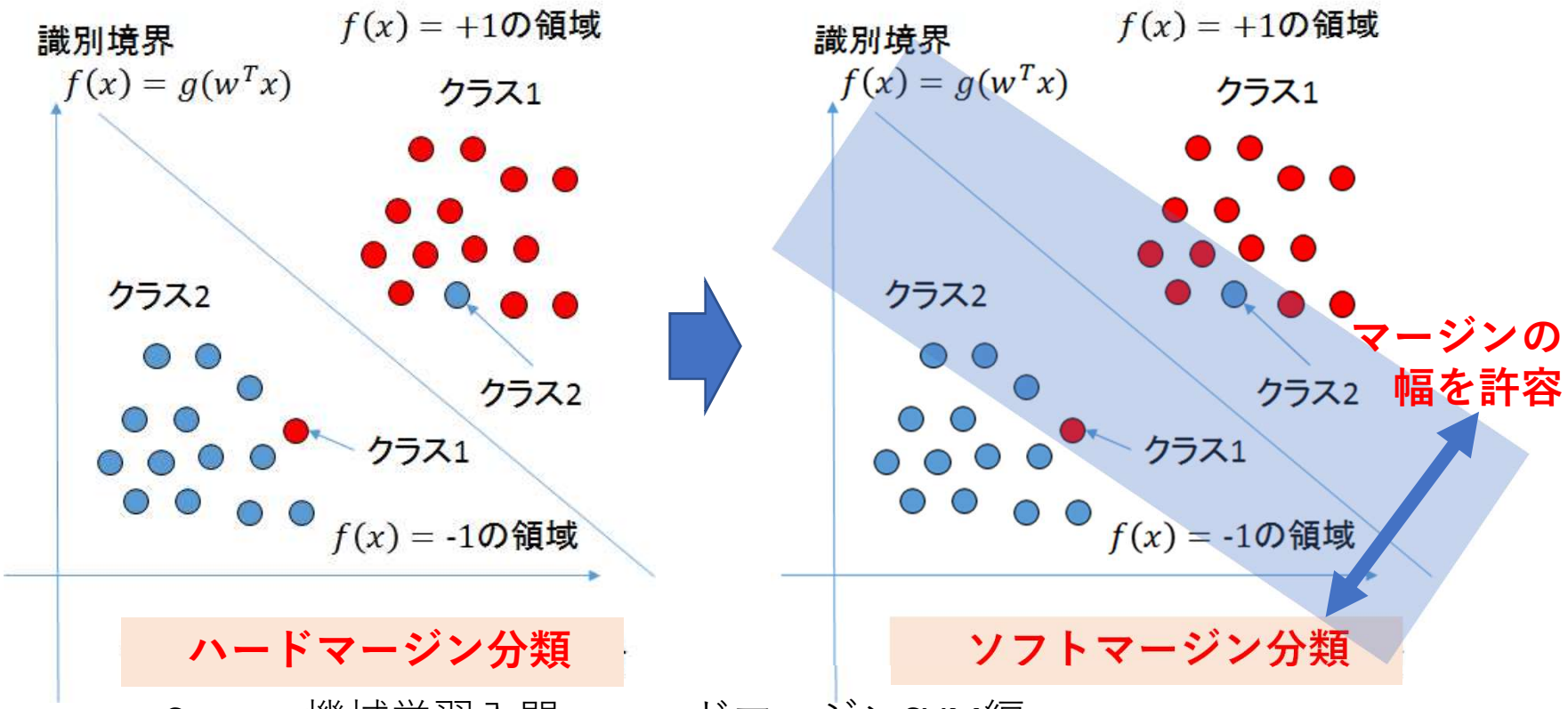
$$\frac{W^T(X_{pos} - X_{neg})}{\|W\|} = \frac{2}{\|W\|}$$



Source: 機械学習入門～ハードマージンSVM編～,
https://qiita.com/pesuchin/items/c55f40b69aa1aec2bd19

スラック変数の導入による非線形分離問題への対処

- 非線形分離となるデータ（ノイズ）を許容しつつパラメータを学習



Source: 機械学習入門～ハードマージンSVM編～,
https://qiita.com/pesuchin/items/c55f40b69aa1aec2bd19

スラック変数を導入した正則化

$$W^T x^{(i)} \geq 1 - \xi^{(i)} \quad for\ y^{(i)} = 1$$

$$W^T x^{(i)} < -1 + \xi^{(i)}\ for\ y^{(i)} = -1$$

従って, 最小化すべきコスト関数は

$$\frac{1}{2}\|W\|^2 + C\left(\sum_i \xi^{(i)}\right),$$

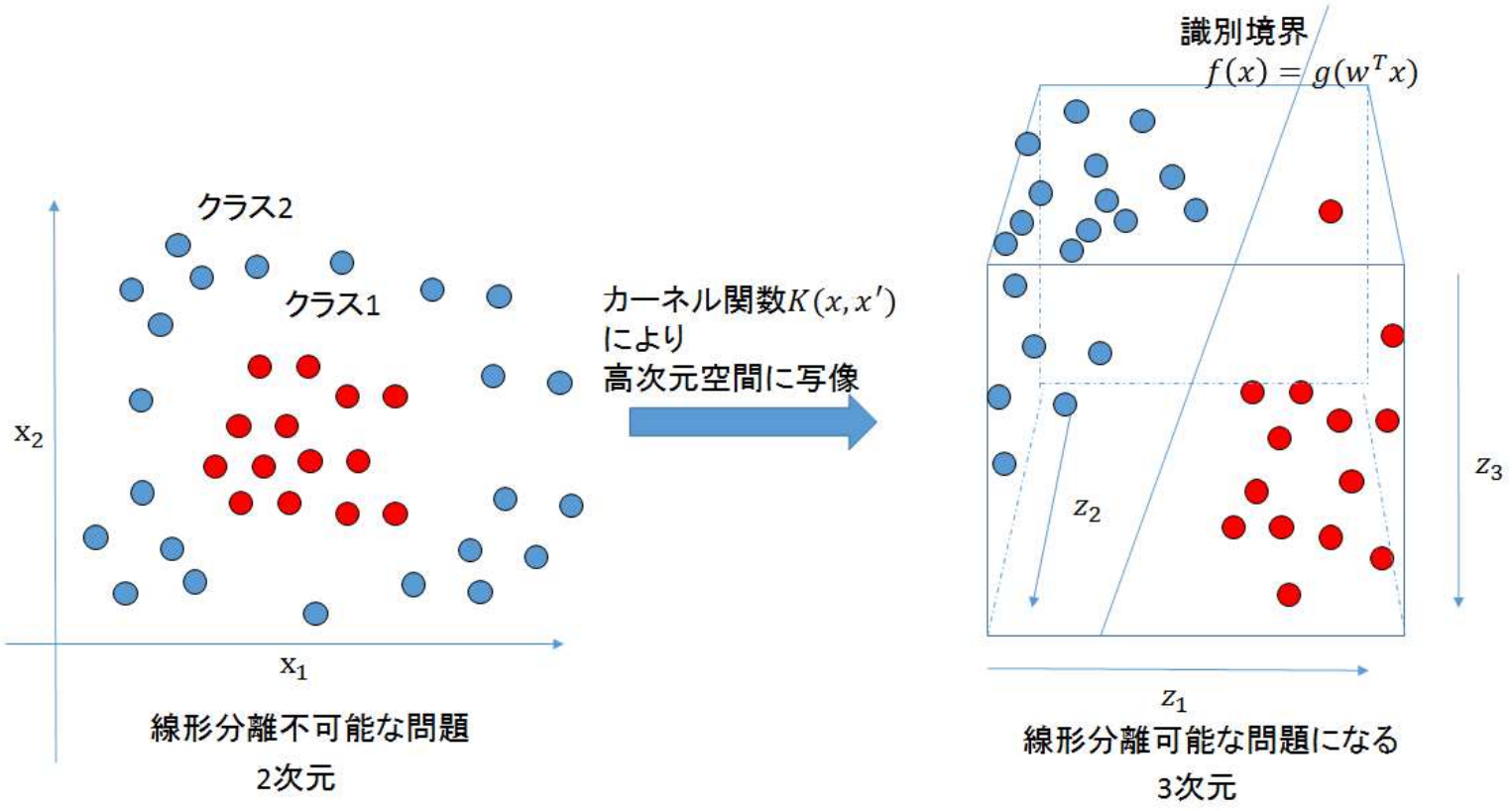
ここでCは正則化パラメータ

Cが大きいと幅が狭い(誤分類に非寛容, ただし探索が収束しない)

Cが小さいと幅が広がる（誤分類に寛容, 収束が速い）

カーネルトリック(カーネルSVM)

- 線形分離可能な高次元空間にデータを写像
- 詳細はカーネル法を参照
  - 再生核ヒルベルト空間, Representer定理, Mercerの定理



カーネル関数とハイパーパラメータ

- SVMの学習中に使われる指標を工夫

  (学習は2次計画法を参照)

$$\mathrm{K}\left(x^{(i)}, x^{(j)}\right) = \boldsymbol{\phi}\left(x^{(i)}\right)^{T} \boldsymbol{\phi}\left(x^{(j)}\right)$$

- Kは動径基底関数カーネル(Radial Basis Function kernel)が使わることが多い

$$\mathrm{K}\left(x^{(i)}, x^{(j)}\right) = exp\left(-\frac{\left\|x^{(i)} - x^{(j)}\right\|^{2}}{2\sigma^{2}}\right)$$

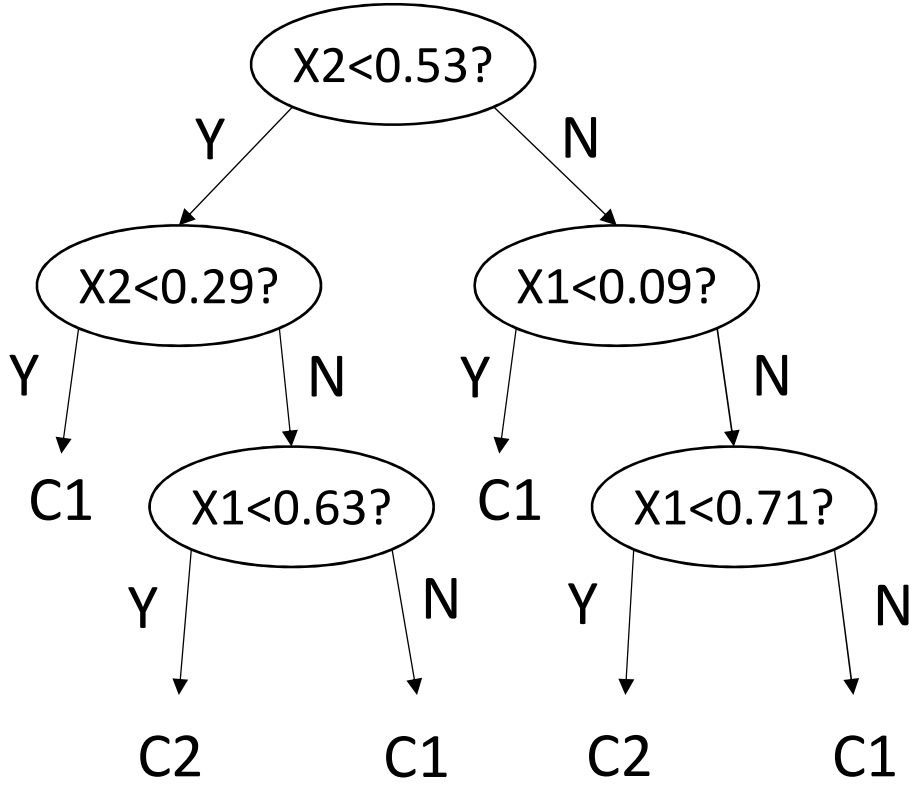
- 多くの場合, 以下のように簡略化

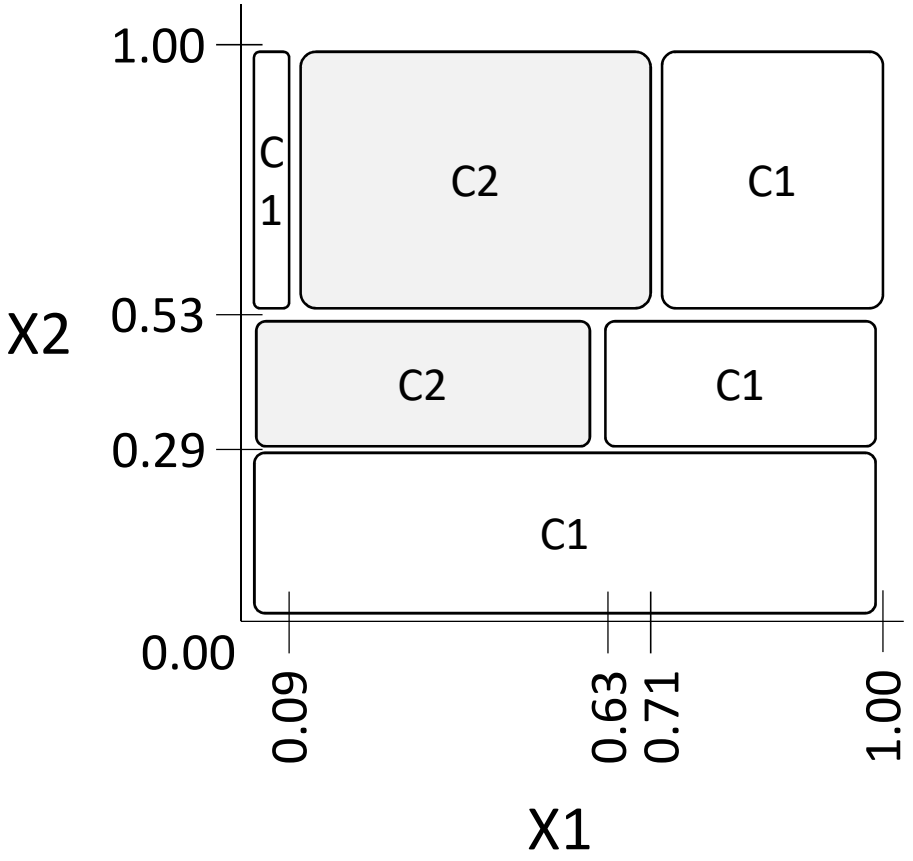
$$\mathrm{K}\left(x^{(i)}, x^{(j)}\right) = exp\left(-\gamma\left\|x^{(i)} - x^{(j)}\right\|^{2}\right),$$

ここで$\gamma = \frac{1}{2\sigma^{2}}$はハイパーパラメータ

分類木 (Decision Tree)

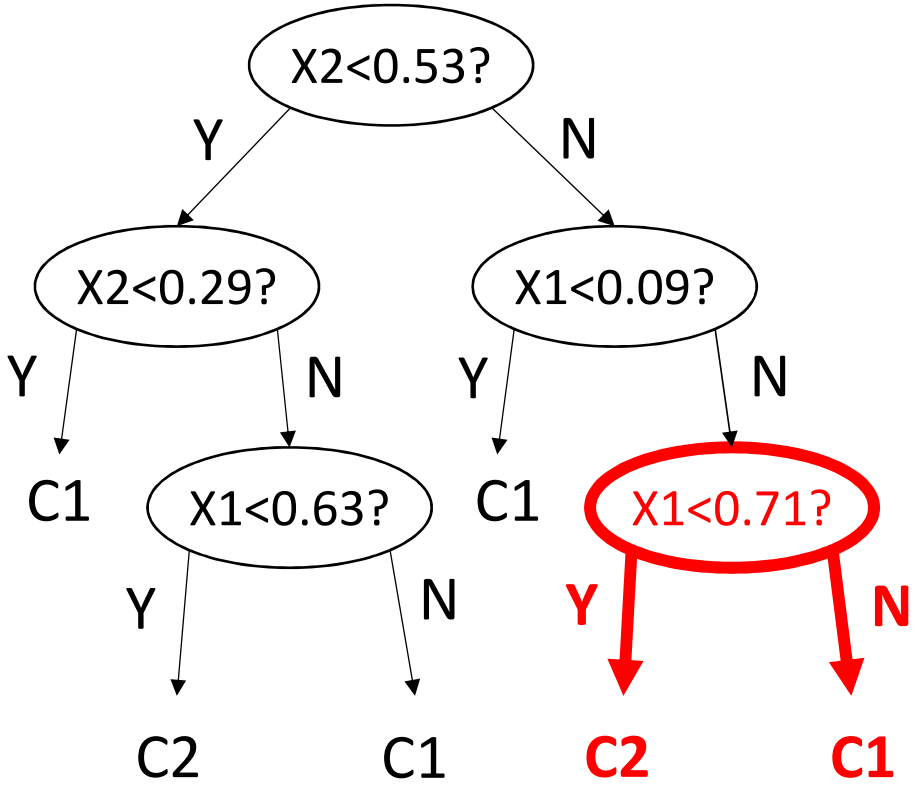
- 特徴マップを分類する決定木学習，弱学習器

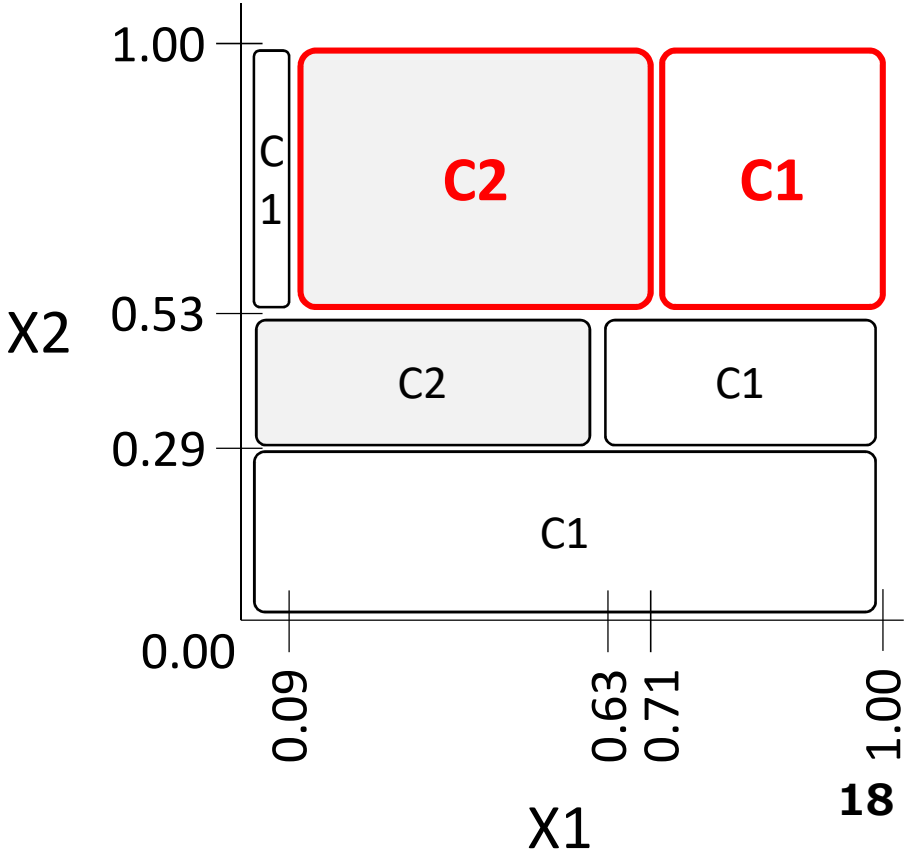




分類木の学習

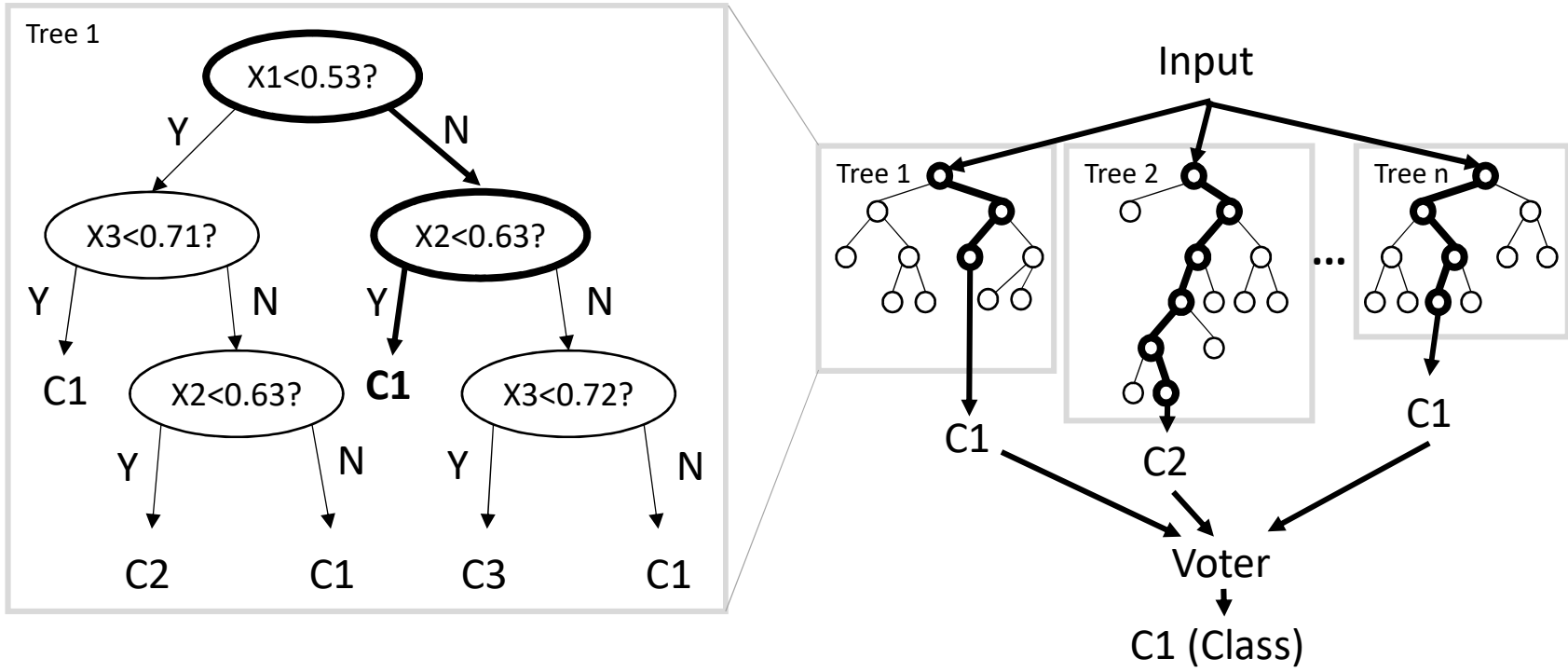
- ランダムにサンプリング
- エントロピーが最大になるように再帰的に分割
- 深さに対するハイパーパラメータが存在→深すぎると可適合





Random Forest (RF)

- アンサンブル学習の一種
- **複数の分類木（弱学習器）で構成**
- **クラス分類と回帰が可能**

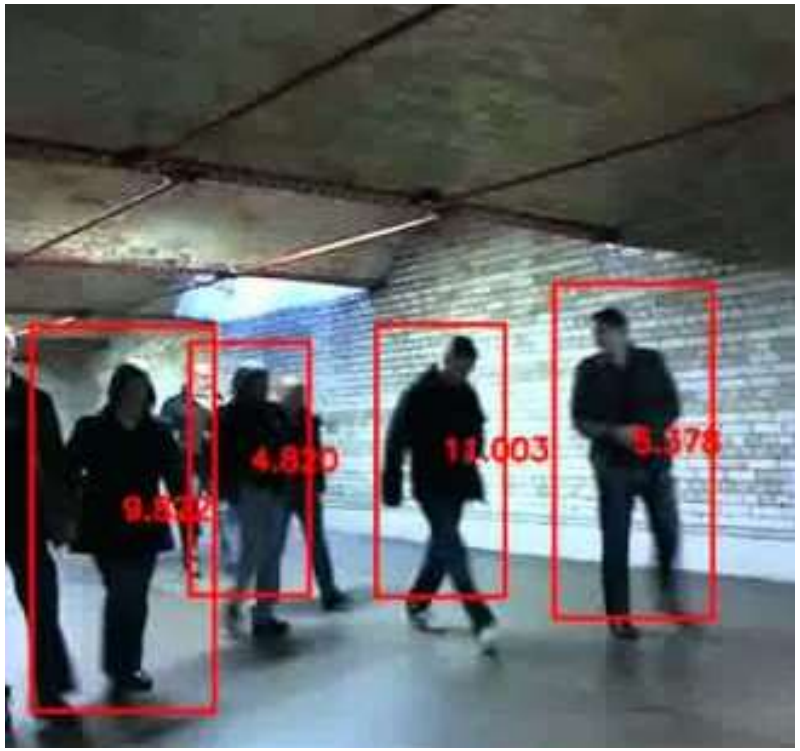


Binary Decision Tree (BDT)

Random Forest

RFのアプリケーション

- Key point matching [Lepetit et al., 2006]
- Object detector [Shotton et al., 2008][Gall et al., 2011]
- Hand written character recognition [Amit&Geman, 1997]
- Visual word clustering [Moosmann et al.,2006]
- Pose recognition [Yamashita et al., 2010]
- Human detector [Mitsui et al., 2011] [Dahang et al., 2012]
- Human pose estimation [Shotton 2011]



PythonでRF

- Scikit-learnを使うだけ
- ハイパーパラメータ：各決定木で学習させるサンプル数（ブートストラップ標本、後日）→$\sqrt{n}$を奨励
- 決定木の深さ→深いと可適合
- 決定木の個数→多いと計算時間が長い

```python
from sklearn.ensemble import RandomForestClassifier

train_data = [[1, 1], [2, 2], [-1, -1], [-2, -2]]
train_label = [   1,      1,       -1,       -1]
test_data  = [[3, 3], [-3, -3]]
# estimated = [   1,       -1]

model = RandomForestClassifier()
model.fit(train_data, train_label)
output = model.predict(test_data)

for label in output:
    print(label)
```

オブジェクトを宣言
学習
推論

実習：様々な分類器

https://github.com/HirokiNakahara/MachineLearning_Lecture/tree/master/lec2_3

- ロジスティック回帰
- サポートベクトルマシン (SVM)
- 決定木
- ランダムフォレスト