

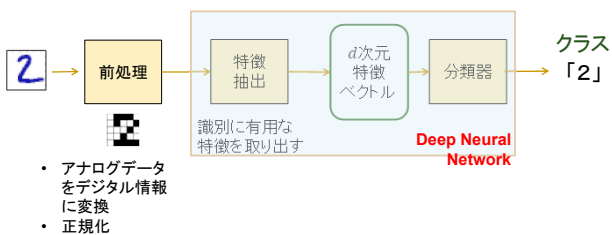
## 知的情報処理論 (第4回)

2023年5月9日(火)  
産業科学研究所  
駒谷 和範

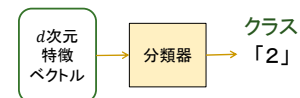
## 第4回:種々の機械学習アルゴリズム

1. 機械学習用パッケージソフトウェア Weka
2. 決定木
  - 学習アルゴリズム
  - ランダムフォレスト(Random Forest)への展開
3. 汎化性能
  - 過学習(overfitting)
  - 先週の「得られる重みの値に関する制約」

## 復習:パターン認識器の構成



## 復習:パターン認識の定式化 (教師あり学習による分類)



- 入力:  $d$ 次元ベクトル  $x = (x_1, \dots, x_d)$
- 出力:  $c$ 個のクラスのいずれか  $C_i$   $i \in \{1, \dots, c\}$
- 学習データ  $\chi = \{(x_1, \widehat{C}_1), \dots, (x_n, \widehat{C}_n), \dots\}$ 
  - 入力データと正解の組の集合
  - $n$ は学習データの一要素( $n$ 番目)を表す添え字
  - $\widehat{C}_1, \dots, \widehat{C}_n, \dots \in \{C_i\}$

## 機械学習用パッケージソフトウェア

## 機械学習パッケージ

- scikit-learn, liblinear, libsvm, ...
- Weka
- DNNも可能
  - Tensorflow, Keras
  - Chainer
  - PyTorch

## Weka

Project Software Book Courses Publications People Related

**Weka 3: Machine Learning Software in Java**

Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization.

Found only on the islands of New Zealand, the Weka is a flightless bird with an inquisitive nature. The name is pronounced like this, and the bird sounds like this.

Weka is open source software released under the GNU General Public License.

We have put together several free online courses that teach machine learning and data mining using Weka. The videos for the courses are available on YouTube.

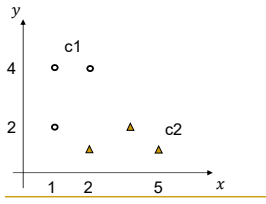
Weka supports deep learning!

<p>Getting started</p> <ul style="list-style-type: none"> <li>• Requirements</li> <li>• Download</li> <li>• Documentation</li> <li>• FAQ</li> <li>• Getting Help</li> </ul>	<p>Further information</p> <ul style="list-style-type: none"> <li>• Citing Weka</li> <li>• Datasets</li> <li>• Related Projects</li> <li>• Miscellaneous Code</li> <li>• Other Literature</li> </ul>	<p>Developers</p> <ul style="list-style-type: none"> <li>• Development</li> <li>• History</li> <li>• Subversion</li> <li>• Contributors</li> <li>• Commercial Licenses</li> </ul>
---	--	---

- ニュージーランドWaikato大学で開発  
<https://www.cs.waikato.ac.nz/ml/weka/>
- GUI付き
- 多くの機械学習アルゴリズムが利用可能
  - 決定木学習, SVM, Neural network, etc...
- Java実装
- パッケージ内にサンプルデータも含まれている

## ARFF file

- Wekaのファイル入力形式
  - 単純にCSVを読み込むことも可能



```

@relation pattern
@attribute x real
@attribute y real
@attribute class {c1,c2}

@data
1,2,c1
1,4,c1
2,4,c1
2,1,c2
5,1,c2
4,2,c2
          
```

名前(何でもいいい)

特徴の名前と型の宣言  
• 数値 real  
• ラベル 要素を記述  
※ラベルは大小関係なし

1行1サンプル  
@attributeで宣言した順

## 例: 線形SVMによる分類

1. Wekaを起動し、[Explorer]を選択、その後[Open File]でARFFファイルを読み込む。(Windowsで、ARFFファイルにWekaが紐づけられている場合はダブルクリック。)
2. 次に[Classify]タブの[Choose]ボタンから[SMO]を選択。
3. 識別平面の式を確認するため(結果の解釈を容易にするため)に、データの正規化を行わない。オプション設定画面を開き(SMOの後にオプションが書かれたテキストフィールドをクリックすると開く)、[filter type]の値として[No normalization/standardization]を選ぶ。
4. さらに[Test Option]領域で、[Use training set]を選ぶ。これは、学習データそのものを使って識別率を計算することを意味する。
5. Startボタンを押すと、図2のような結果が得られる。

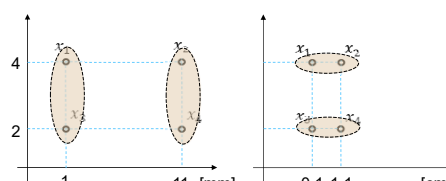
## SVMによる分類結果

```

SMO
Kernel used:
  Linear Kernel: K(x,y) = <x,y>
Classifier for classes: c1, c2
BinarySMO
Machine linear: showing attribute weights,
not support vectors.
      1      * x
+    -1      * y
-      0
《中略》
=== Confusion Matrix ===
 a b  <-- classified as
 3 0 | a = c1
 0 3 | b = c2
          
```

図 2: SVM の出力結果

## データ(入力特徴値)の正規化



- データのバラつき具合が各軸(特徴量ベクトルの各次元)で異なる
  - 特徴量の各次元の値のレンジをおおよそ揃える
    - 外れ値にも影響される
  - 各次元の値が分散が1になるように正規化
 
$$x'_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i} \quad (\text{平均を0にする場合}; \mu_i \text{は平均}, \sigma_i \text{は分散})$$

## 様々な機械学習アルゴリズム

### 決定木学習

## 決定木学習 (decision tree learning)

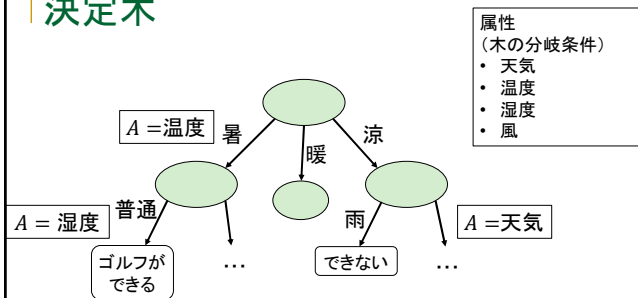
- 機械学習(データマイニング)の代表的アルゴリズムのひとつ
- 多値分類器
  - 入力  $x$  はラベルまたは連続値
  - 出力  $y$  はラベル(複数可;二値に限らない)
 入力  $x_n$  をいずれかのクラスに分類する木を学習する
- トップダウン方式の分割統治法に相当

表 1: 例: 気象条件とゴルフプレイに関するデータ

天気	温度	湿度	風	ゴルフプレイ
晴	暑	高	無	×
晴	暑	高	有	×
曇	暑	高	無	○
雨	暖	高	無	○
雨	涼	普通	無	○
雨	涼	普通	有	×
曇	涼	普通	有	○
晴	暖	高	無	×
晴	涼	普通	無	○
雨	暖	普通	無	○
晴	暖	普通	有	○
曇	暖	高	有	○
曇	暑	普通	無	○
雨	暖	高	有	×

目的変数:  
このラベルを  
予測する

## 決定木



- 与えられたデータに対して「最適な」決定木をどうやって作る?

## 決定木学習アルゴリズム

- ① 根ノードに置く属性を決定し、その属性値に応じて分岐を作成する
- ② データ集合を各分岐に応じて部分集合に分割して子ノードを作成し、その子ノードを根ノードとする。
- ③ ①と②のプロセスを再帰的に繰り返す、決定木を成長させる
- ④ 子ノードの全ての事例が同一クラスに属していれば、その葉の展開を止める

## 分岐に用いる属性を決定する方法

エントロピー(情報利得)を用いる方法  
(ID3アルゴリズム)

- ある属性  $A$  による分割前後のエントロピーを計算し、情報利得が最大となる属性を選択する
- あるデータ集合(の分割方法)のエントロピー  $I$ 
  - 予測ラベル  $v = \{v_i\}$  が確率  $p(v_i)$  で含まれる場合
  - $I(p(v_1), \dots, p(v_N)) = \sum_{i=1}^N -p(v_i) \log_2 p(v_i)$
  - 例えばデータが7個で、Yesが4個、Noが3個のとき

$$I(p(\text{Yes}), p(\text{No})) = -\frac{4}{7} \log_2 \frac{4}{7} - \frac{3}{7} \log_2 \frac{3}{7}$$

## 分岐に用いる属性を決定する方法

属性  $A$  による情報利得  $\text{Gain}(A)$

$= (\text{分岐前の } I) - (A \text{ による分岐後の } I \text{ の期待値})$

- (右辺の第二項)  $= \sum_j \frac{N_j}{N} I_j(p(v_1), \dots, p(v_n))$ 
  - $N$  は分岐前のデータ数
  - $N_j$  は分岐後の  $j$  番目の集合のデータ数
  - $I_j()$  は分岐後の  $j$  番目の集合におけるエントロピー
- $\text{Gain}(A)$  が最大となる属性  $A$  で分割する

## 例題の解答

### 最初の分岐属性を求める

- 全データのエントロピー(分岐前の $I$ )

$$\square I\left(\frac{9}{14}, \frac{5}{14}\right) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940$$

$-0.637$        $-1.485$

- 次に「 $A$ による分岐後の $I$ の期待値」を求める

## 例題の解答

### 最初の分岐属性を求める

各属性で分岐した際のラベルの分布

- $A = \text{天気} \rightarrow$ 

晴	×	×	×	○	○
曇	○	○	○	○	
雨	○	○	×	○	×
- $A = \text{温度} \rightarrow$ 

暑	×	×	○	○	
暖	○	×	○	○	○
涼	○	×	○	○	
- $A = \text{湿度} \rightarrow$ 

高	×	×	○	○	×	○	×
普通	○	×	○	○	○	○	○
- $A = \text{風} \rightarrow$ 

無	×	○	○	○	×	○	○	○
有	×	×	○	○	○	×		

## Mini Quiz #1

天気を選択して分岐させた後のエントロピー $I$ の期待値を求めよ。

$$\square \log_2 5 = 2.32, \log_2 3 = 1.58$$

- 晴の場合:  $I\left(\frac{2}{5}, \frac{3}{5}\right) =$
- 曇の場合:
- 雨の場合:
- $A = \text{天気}$ でのエントロピー $I$ の期待値 =

## Mini Quiz #1の解答

$$\square \log_2 5 = 2.32, \log_2 3 = 1.58$$

- 晴の場合:  $I\left(\frac{2}{5}, \frac{3}{5}\right) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5}$   
 $= -0.4 \times (-1.32) - 0.6 \times (-0.74) = 0.972$
- 曇の場合:  $I\left(\frac{4}{4}, \frac{0}{4}\right) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$
- 雨の場合:  $I\left(\frac{2}{5}, \frac{3}{5}\right) = 0.972$
- $I$ の期待値  $= \frac{5}{14} \times 0.972 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.972$   
 $= 0.694$

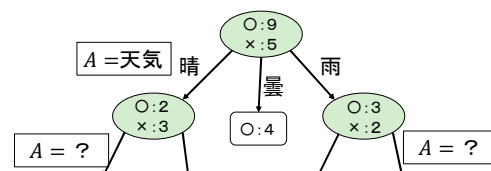
## 例題の解答

### 最初の分岐属性を求める

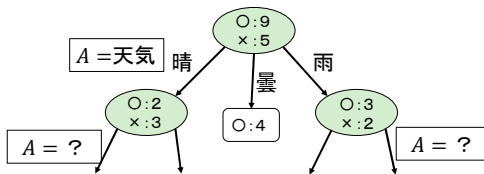
- この結果,  
 $\text{Gain}(\text{天気}) = 0.940 - 0.694 = 0.246$
- 同様に全ての属性について計算すると,  
 $\text{Gain}(\text{温度}) = 0.029$   
 $\text{Gain}(\text{湿度}) = 0.152$   
 $\text{Gain}(\text{風}) = 0.048$
- 以上より,  $A = \text{天気}$ を選択すると, 情報利得が最大

## 例題の解答

### 再帰的に繰り返す

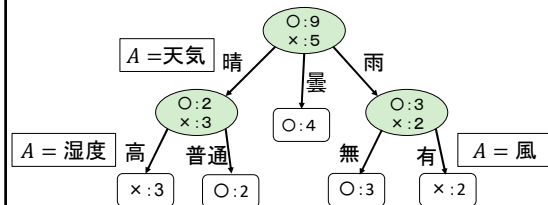


## Mini Quiz #2



- 決定木を完成させよ
  - 左側の枝と右側の枝のそれぞれについて、分岐させるのに最適な属性は何か？
  - 天気による分岐後の集合を具体的に見ればわかる

## Mini Quiz #2 の解答



## 分割基準の改良

- 情報利得を使う場合の問題点
  - もし例題の14個のデータにID(1.~14.)がついていた場合、IDで分割すると最適となる
  - 事実上役に立たない
- 情報利得比を利用 (C4.5やC5.0アルゴリズム)
  - 情報利得を分割情報量で正規化
  - 上記の例の場合  $\log_2 14 = 3.807$  で除することになる
- Wekaでの実装
  - 情報利得 [tree] -> [ID3]
  - 情報利得比 [tree] -> [J48]      C4.5のJava実装

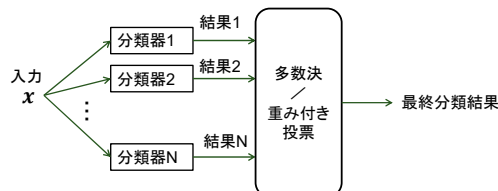
## 実用上必要な要素

- 過学習 (over-fitting) の防止
  - 枝刈り (pruning) を行う。細かい枝の生成をやめる
- 連続値 (数値属性) の扱い
  - その属性の値を2分割するしきい値を求めて、離散化して分割 (二分探索)
- 欠損値の扱い
  - 単純な方法では、カテゴリ型属性の場合は最頻値、数値型属性の場合は平均値を割り当てる、等

## ランダムフォレスト (Random Forest)

アンサンブル学習 (ensemble learning) の一種

- 複数の分類器の結果を持ち寄って統合
  - 「N人寄れば文殊の知恵」
  - ベース分類器の出力の組み合わせ



## アンサンブル学習 (Ensemble learning)

どうやって「独立な」分類器を用意するか

- バギング (bagging)
  - 学習データの部分集合を複数作り分類器を学習
  - ランダムフォレスト (Random Forest)
    - 元となる学習器: 決定木学習
    - (学習データだけでなく) 特徴も一定個数までランダムに選択して分類器を学習
- ブースティング (boosting)
  - 学習データに重みをつけて逐次的に分類器を学習
  - 各分類器の結果を、誤り率の重みを付けて統合
  - AdaBoost

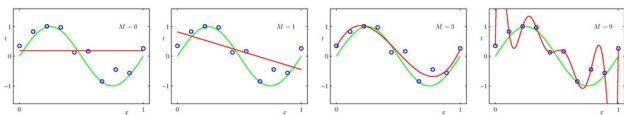
## 汎化性能

## 機械学習／パターン認識の目的

- 機械学習の本来の目的
    - × 学習データを正しく分類する
    - 学習データにない未知のデータを正しく分類する
- 汎化 (generalization) 性能**

## 過学習 (overfitting)

- 学習データは常に有限
  - 統計的機械学習では、有限のデータから真の分布を推定
- 過学習 (overfitting)
  - 学習データに過剰に適応し汎化性能が低くなること
  - 学習データの量に対して、自由度の大きい(次元数, パラメータ数が大きい)モデルで起こりやすい



上記グラフはPRMLで示されていたもの

- モデル選択 (model selection)
  - 適切な学習モデル(ハイパーパラメータを含む)を選ぶ
- 正則化 (regularization)
  - 目的関数に正則化項(制約)をつける. 重みが複雑な値になるのを避ける

## 得られる重みの値に関する制約

- 得られた重みの値は「最適」なのか  
何をもって「最適」とするか？
  - 最適になるように損失関数を工夫する
- ① 重みの値が極端に大きい値を取らない方がよい
  - 正則化項 (regularization term)の導入
    - 損失関数に重みの値を含める.
      - L1正則化(Lasso), L2正則化(Ridge)
      - L1正則化(Lasso)
 
$$\min\{\text{誤差の総和} + \lambda \sum_{i=1}^n |w_i|\}$$

## 得られる重みの値に関する制約

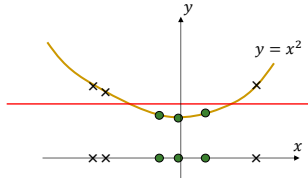
何をもって「最適」とするか？

- ② 識別面が各クラスの点のちょうど間くらいにある方がよい
  - マージン(margin)最大化
- SVM (support vector machine) の特徴
  1. マージン最大化
  2. カーネル(kernel)関数の導入

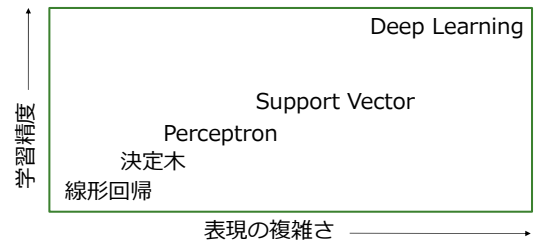
## SVMの特徴

### 2. カーネル(kernel)関数の導入

- 非線形な高次元空間に写像することで分類可能とする



## 学習に必要なデータ量



- 表現(モデル)が複雑 ← パラメータ数が多い
  - きちんと学習させるには学習データ量が必要
  - 学習データを少し変えた場合にもsensitive
- 「何でも最新のDeep Learningを使えばいい」はウソ

## 分類問題への3つのアプローチ

入力  $x$  に対してクラスラベル  $C_i$  を決める

$$\hat{C} = \operatorname{argmax}_i p(C_i|x) \text{ を求めたい}$$

### ① 識別関数

- 入力からクラスラベル  $C_i$  への写像関数  $f(x)$  を直接求める
- 確率や統計は不使用

### ② 識別モデル

### ③ 生成モデル

## 使われる確率モデルによる大別

データからの統計的な予測(確率分布の推定)に基づく

### ② 識別モデル (discriminative model)

- 事後確率  $p(C_i|x)$  を直接モデル化

### ③ 生成モデル (generative model)

- $p(C_i|x) \propto p(x|C_i)p(C_i)$  (ベイズの定理より)
- クラスの条件付き密度  $p(x|C_i)$  と 事前クラス確率  $p(C_i)$  をモデル化

「損失関数が最小となるようパラメータ更新」は共通

## 評価方法

- 汎化性能を測るには評価データと学習データを分ける

- 分割学習法 (open test)
- closed test: 学習データそのまま評価データにすること

- ただし、それだと学習データが不十分な場合

⇒ 交差検証法 (cross validation)

- 10分割交差検証

1. 学習データを10分割する
2. そのうち9個を使って学習、残り1個で評価
3. 2を10個分繰り返す、その結果の平均を用いる

- 同じデータを分割することから、(おそらく)均質なデータの中での評価

ハイパーパラメータをデータで最適化する  
なら、検証用セット (validation set) も必要