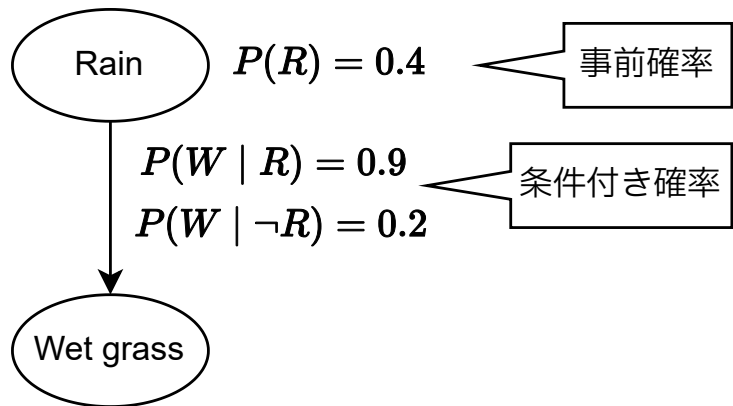


4. 識別 —統計的手法—



- 4.1 統計的識別とは
- 4.2 カテゴリ特徴に対するベイズ識別
- 4.3 ベイジアンネットワーク



- 荒木雅弘：『Pythonではじめる機械学習』（森北出版，2025年）
- スライドとコード

第3章（決定木）と第4章（統計的識別）の違い

- 決定木
 - 正解を表現する概念を得る（説明性が高い）
- 統計的識別
 - 識別結果の確率を得る（意思決定に役立つ）

weather.nominalデータ

No	outlook	temperature	humidity	windy	play
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	vercast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

- outlook (天候)
 - sunny, overcast, rainy
- temperature (気温)
 - hot, mild, cool
- humidity (湿度)
 - high, normal
- windy (風)
 - TRUE, FALSE
- play (=クラス)
 - yes, no

4.1 統計的識別とは (1/4)

- weather.nominalデータに対する統計的識別
 - 2クラス分類問題 (play: yes/no)
 - 特徴ベクトル \mathbf{x} が観測されていないとき
 - 事前確率 $P(\text{yes}), P(\text{no})$ だけから判断するしかない
 - 特徴ベクトル \mathbf{x} 観測後
 - 事後確率 $P(\text{yes} \mid \mathbf{x}), P(\text{no} \mid \mathbf{x})$ の大きい方に判定
- 多クラス ($\omega_i : i = 1, \dots, c$) に一般化
 - 最大事後確率則による識別 (ベイズ識別)

$$C_{\text{MAP}} = \arg \max_i P(\omega_i \mid \mathbf{x})$$

4.1 統計的識別とは (2/4)

- 事後確率の求め方
 - データの発生頻度に基づく方法
 - 特徴ベクトルが完全に一致する事例を大量に集めて、その正解ラベルの割合を求める
 - 例) $\boldsymbol{x} = (\text{晴}, \text{高}, \text{中}, \text{TRUE})$ を100事例集めて、yes:70事例, no:30事例ならば
$$P(\text{yes} \mid (\text{晴}, \text{高}, \text{中}, \text{TRUE})) = 0.7 \text{ とする}$$
 - しかし、上記の推定が行えるようなデータセットが得られることはほとんどない
 - そのため、事後確率に対して式変形・近似を行って、現実の規模のデータセットから確率を推定できるようにする

4.1 統計的識別とは (3/4)

- ベイズの定理

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

- 事後確率を尤度と事前確率の積に変形する手順
 - 事後確率にベイズの定理を適用
 - 最大値を求めるときに無関係な分母を払う

$$C_{\text{MAP}} = \arg \max_i P(\omega_i \mid \boldsymbol{x}) = \arg \max_i \frac{P(\boldsymbol{x} \mid \omega_i)P(\omega_i)}{P(\boldsymbol{x})} = \arg \max_i P(\boldsymbol{x} \mid \omega_i)P(\omega_i)$$

- 尤度：特定のクラスから，ある特徴ベクトルが出現する尤もらしさ

4.1 統計的識別とは (4/4)

- ベイズ統計とは
 - 結果から原因を求める
 - 通常の統計学は原因から結果を予測する
- ベイズ識別
 - 通常, 確率が与えられるのは 原因→結果 (尤度)
 - ベイズ識別では, 事前分布 $P(\omega_i)$ が, 観測結果 \boldsymbol{x} によって事後分布 $P(\omega_i \mid \boldsymbol{x})$ に変化したと考え、その結果をもたらした原因 ω_i を求める

4.2 カテゴリ特徴に対するベイズ識別

- 事前確率： $P(\omega_i)$
 - 特徴ベクトルを観測する前の各クラスの起きやすさ
- 事前確率の最尤推定
 - 学習データ中のクラスの割合から推定する
 - N : 全データ数, n_i : クラス ω_i のデータ数

$$P(\omega_i) = \frac{n_i}{N}$$

学習データの対数尤度 (1/3)

- 尤度の導出
 - 特徴ベクトル \mathbf{x} を生成するモデルを考え、そのモデルが（クラスごとの）パラメータ $\boldsymbol{\theta}_j$ に従ってデータを生成していると仮定

$$P(\mathbf{x} \mid \omega_j; \boldsymbol{\theta}_j)$$

- これらはクラス毎のデータから推定することになるので、以後、1クラス分のデータを全データ D とみなして ω_j を省略し、 $\boldsymbol{\theta}_j$ を $\boldsymbol{\theta}$ と表記
- i.i.d. (independent and identically distributed, 独立同分布) を仮定
 - 全データ D は、各データが同じ分布から独立に生成されていると仮定して尤度を計算

$$P(D \mid \boldsymbol{\theta}) = \prod_{i=1}^N P(\mathbf{x}_i \mid \boldsymbol{\theta})$$

学習データの対数尤度 (2/3)

- 対数尤度: $\mathcal{L}(D)$
 - 確率の積のアンダーフローを避けるため、尤度を対数で計算

$$\mathcal{L}(D) = \log P(D \mid \boldsymbol{\theta}) = \sum_{i=1}^N \log P(\boldsymbol{x}_i \mid \boldsymbol{\theta})$$

- 尤度関数の仮定の例
 - 特徴ベクトルが1次元、値0 or 1で、ベルヌーイ分布に従うと仮定
 - ベルヌーイ分布：確率 θ で値1、確率 $1 - \theta$ で値0をとる分布

$$\mathcal{L}(D) = \sum_{i=1}^N \log \theta^{x_i} (1 - \theta)^{(1-x_i)} = \sum_{i=1}^N x_i \log \theta + (N - \sum_{i=1}^N x_i) \log(1 - \theta)$$

学習データの対数尤度 (3/3)

- 対数尤度を最大にするパラメータ： $\hat{\theta}$
 - $\frac{\partial \mathcal{L}(D)}{\partial \theta} = 0$ の解である $\hat{\theta}$ を求める

$$\begin{aligned}\frac{\partial \mathcal{L}(D)}{\partial \theta} &= \sum_{i=1}^N x_i \frac{1}{\theta} + (N - \sum_{i=1}^N x_i) \frac{1}{1 - \theta} \\ &= \frac{1}{\theta(1 - \theta)} \left\{ (1 - \theta) \sum_{i=1}^N x_i - \theta (N - \sum_{i=1}^N x_i) \right\} = 0\end{aligned}$$

$$\hat{\theta} = \frac{1}{N} \sum_{i=1}^N x_i$$

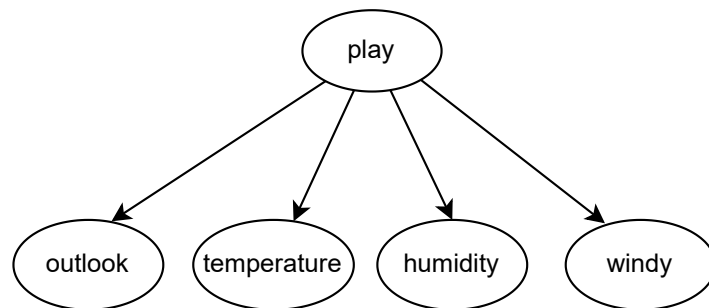
- 値1がでる確率の最尤推定値として、値1がでた回数を全データ数 N で割ったものが得られた

ナイーブベイス識別 (1/5)

- 多次元ベクトルの尤度関数を求める
 - 特徴値のすべての組合せがデータセット中に何度も出てくる必要があるが、これも非現実的
- ナイーブベイズの近似
 - すべての特徴が独立であると仮定すると、同時確率は各次元の確率の積で求められる
 - 各次元で、クラスごとの特徴値の確率分布（尤度）を推定すればよい

$$P(\mathbf{x} \mid \omega_i) = P(x_1, \dots, x_d \mid \omega_i) \approx \prod_{k=1}^d P(x_k \mid \omega_i)$$

$$C_{NB} = \arg \max_i P(\omega_i) \prod_{k=1}^d P(x_k \mid \omega_i)$$



ナイーブベイス識別 (2/5)

- 尤度の最尤推定
 - n_j : クラス ω_j のデータ数
 - n_k : クラス ω_j のデータのうち, k 次元目の値が x_k であるデータ数

$$P(x_k \mid \omega_j) = \frac{n_k}{n_j}$$

- ゼロ頻度問題: n_k が 0 の場合, 確率の推定値も 0 となってしまう
 - 解決法 → スムージング
 - k 次元目の m 種類の値が, 事前に α 回ずつ生じていたと仮定する
 - $\alpha = 1$ のときをラプラス推定とよぶ

$$P(x_k \mid \omega_j) = \frac{n_k + \alpha}{n_j + \alpha m}$$

ナイーブベイズ識別 (3/5)

- scikit-learnのナイーブベイズ識別
 - カテゴリ特徴は `OrdinalEncoder` で整数値に置き換える

```
enc = OrdinalEncoder()  
X_en = enc.fit_transform(X)  
enc.categories_  
  
[array(['overcast', 'rainy', 'sunny'], dtype=object),  
 array(['cool', 'hot', 'mild'], dtype=object),  
 array(['high', 'normal'], dtype=object),  
 array([False, True], dtype=object)]
```

- 変換例
 - `['sunny', 'hot', 'high', False] → [2, 1, 0, 0]`
 - 異なる値に異なる整数値を割り当てているだけであって、数値の近さが概念の近さを表しているのではないことに注意

ナイーブベイズ識別 (4/5)

- scikit-learnのナイーブベイズ識別
 - 正解のラベルは `LabelEncoder` で整数値に置き換える

```
le = LabelEncoder()  
y_en = le.fit_transform(y)  
le.classes_  
  
array(['no', 'yes'], dtype=object)
```

- no → 0, yes → 1

ナイーブベイズ識別 (5/5)

- scikit-learnのナイーブベイズ識別
 - カテゴリ特徴に対するナイーブベイズ識別は `CategoricalNB` を用いる
 - 識別器のパラメータ
 - `alpha` : 事前に仮定するサンプル数. 教科書の m_p に対応
 - `fit_prior` : 事前確率を学習の対象とするかどうか
 - `class_prior` : 事前確率を別途与えるときに用いる
 - 典型的なコード

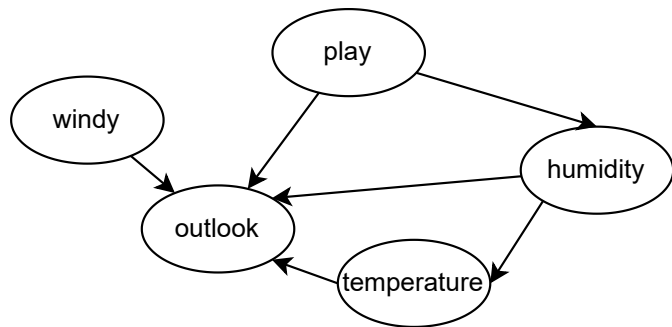
```
clf = CategoricalNB()  
clf.fit(X, y)  
clf.predict_proba(X_test[1])
```

4.3 ベイジアンネットワーク (1/7)

- ベイジアンネットワークの仮定: 変数の部分集合が, ある分類値のもとで独立
 - $Parents(X_k)$ は値 x_k をとるノードの親ノード集合の値

$$P(x_1, \dots, x_d) \approx \prod_{k=1}^d P(x_k \mid Parents(X_k))$$

- 条件付き確率のベイジアンネットワークによる表現

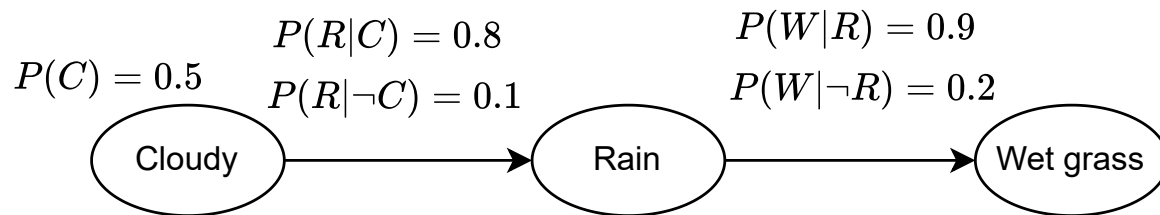


4.3 ベイジアンネットワーク (2/7)

- 変数間の独立性を表す基本パターン
 - Head-to-tail
 - Tail-to-tail
 - Head-to-head

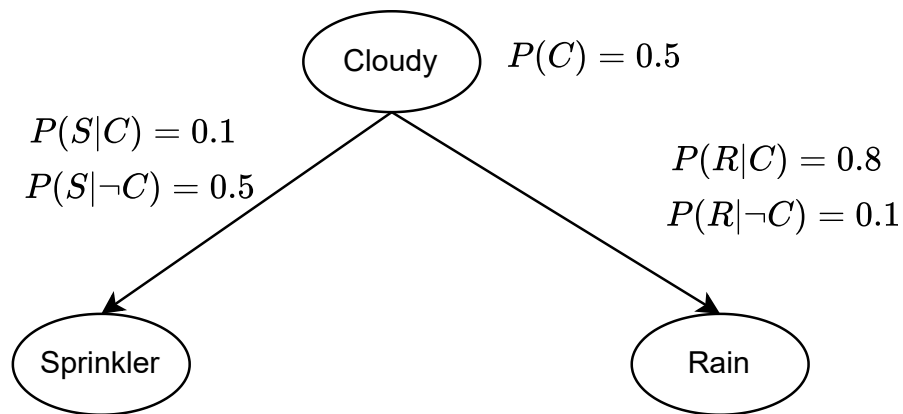
4.3 ベイジアンネットワーク (3/7)

- Head-to-tail
 - 真ん中のノードの値が与えられると、左右のノードは独立



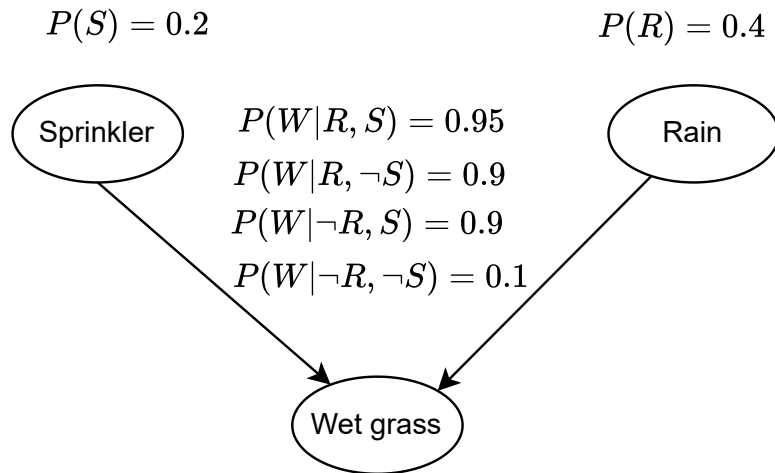
4.3 ベイジアンネットワーク (4/7)

- Tail-to-tail
 - 親ノードの値が与えられると、子ノードどうしは独立



4.3 ベイジアンネットワーク (5/7)

- Head-to-head
 - 子ノードの値が与えられると、親ノードどうしが独立でなくなる



4.3 ベイジアンネットワーク (6/7)

- 確率計算
 - 正確な計算：周辺化によって、すべての値に対する確率を合計する
 - 近似計算：乱数を用いてベイジアンネットワークの確率分布に従った事例を生成し、確率を推定する

$$P(y \mid x_1, \dots, x_d) \approx \frac{C(y, x_1, \dots, x_d)}{C(x_1, \dots, x_d)}$$

4.3 ベイジアンネットワーク (7/7)

- ベイジアンネットワークの学習
 - 入力: ノードの集合 Node, 各変数の観測データ
 - 出力: 学習されたベイジアンネットワーク
 1. ノードの順番を決める (通常はクラスを表す特徴を最初に)
 2. 親ノードの追加を貪欲法で繰り返す
 - repeat**
 - for** n in Node:
 - for** n' in n以降のノード:
 - if** n から n' へのエッジを追加することにより対数尤度が増加
 - n から n' へエッジを追加
 - until** 対数尤度が変化しない
 3. **return** 学習されたベイジアンネットワーク

まとめ

- カテゴリ特徴の識別問題に対する統計的識別
 - ベイズ識別
 - 事後確率 $P(\omega_i | \mathbf{x})$ を最大とするクラス ω_i を求める
 - 事後確率をデータから推定するのは難しいので、ベイズの定理を用いて尤度 $P(\mathbf{x} | \omega_i)$ と事前確率 $P(\omega_i)$ の積に分解
 - ナイーブベイズ法
 - 特徴のすべての次元が独立であると仮定して、尤度をそれぞれの次元の確率の積に分解
 - 確率が0となることを避けるためにスムージングを行う
- ベイジアンネットワーク
 - 変数の部分集合が、ある分類値のもとで独立であるとして構造を推定