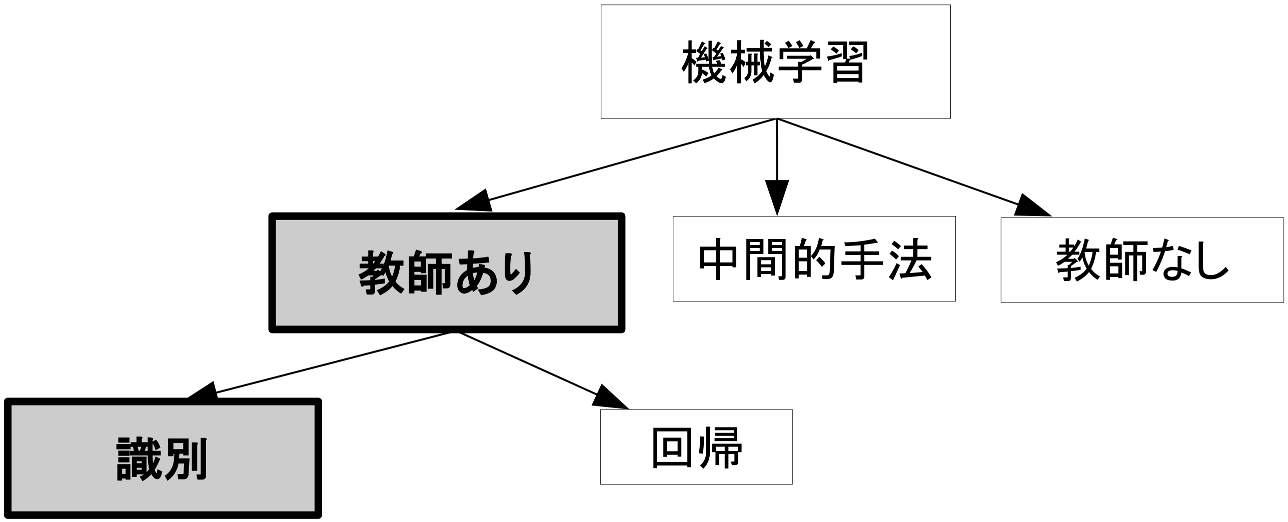
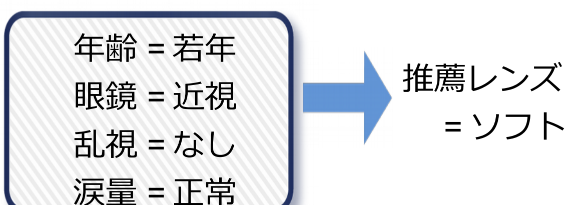
4. 識別　ー統計的手法ー



- ラベル特徴



- 数値特徴

4.1 統計的識別とは

- 最大事後確率則による識別

$$C_{MAP} = \arg\max_i P(\omega_i|\boldsymbol{x})$$

$\boldsymbol{x}$ ：特徴ベクトル

$\omega_i \quad (1 \leq i \leq c)$ ：クラス

  - データから直接的にこの確率を求めるのは難しい

- ベイズの定理 $P(A|B) = \dfrac{P(B|A)P(A)}{P(B)}$

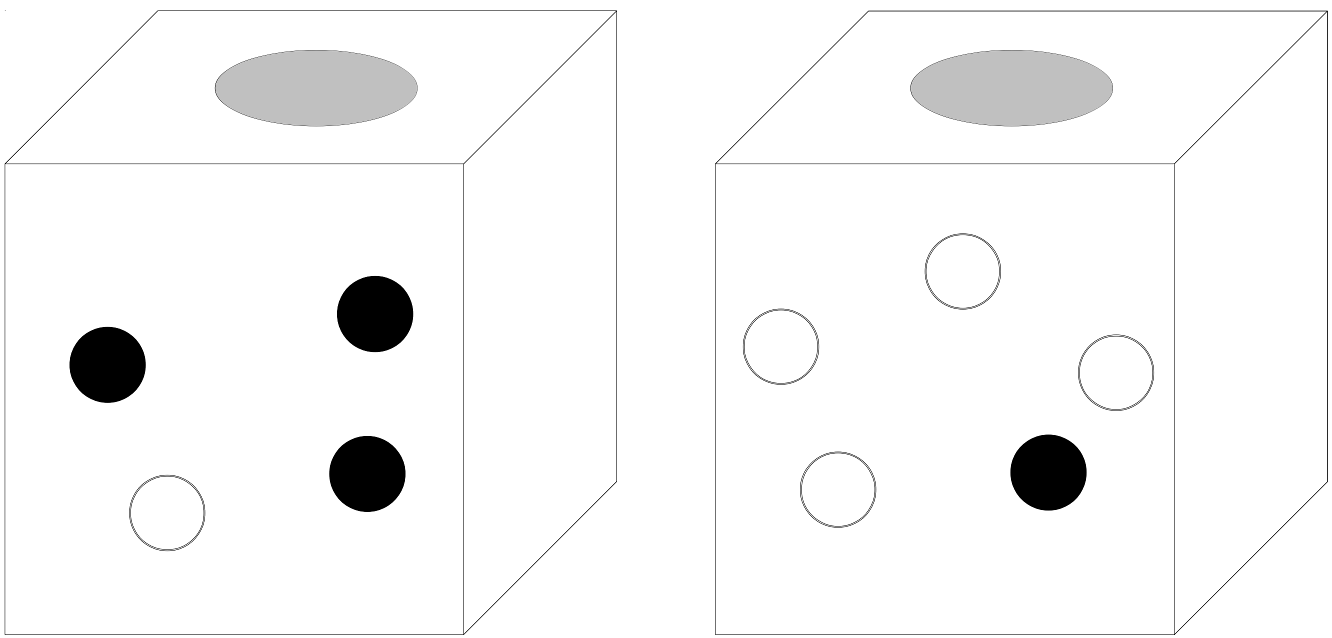
$$\begin{aligned} C_{MAP} &= \arg\max_i P(\omega_i|\boldsymbol{x}) \\ &= \arg\max_i \frac{P(\boldsymbol{x}|\omega_i)P(\omega_i)}{P(\boldsymbol{x})} \\ &= \arg\max_i P(\boldsymbol{x}|\omega_i)P(\omega_i) \end{aligned}$$

4.1 統計的識別とは

- ベイズ統計とは
  - 結果から原因を求める
  - ベイズ識別
    - 観測結果 $\boldsymbol{x}$ から、それが生じた原因 ω_i を求める
    - 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、観測によって事後分布 $P(\omega_i \mid \boldsymbol{x})$ に変化したと考えることができる
    -

4.1 統計的識別とは

- 例題 4.1　観測からの原因の計算
  1. 白玉が出る確率は
  2. 白玉が出たときの箱 A の確率は
  3. 事前確率が 9:1 のときの 2 の確率は



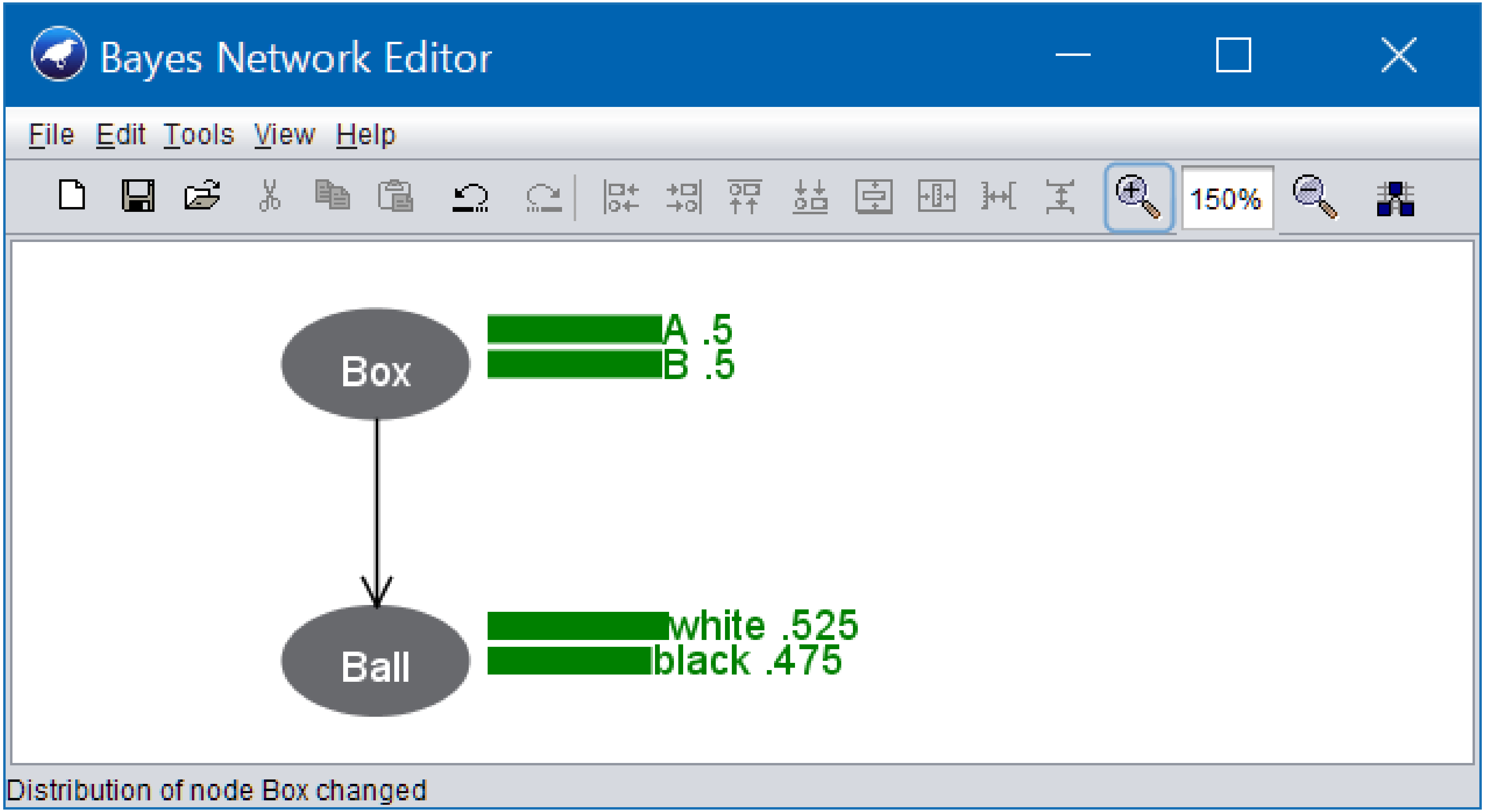
箱 A　　箱 B

4.1 統計的識別とは

- Weka Bayes net editor を用いた解法
  - 起動画面から　Tools → Bayes net editor
  - Tools → Set Margins　で確率値が見えるように
  - Add node　で確率変数にあたるノードを配置
  - Add parent　で原因を特定
  - Edit CPT で条件付き確率を設定
  - 観測結果を設定するときは　Set evidence

4.1 統計的識別とは

- Weka Bayes net editor を用いた解法



4.1 統計的識別とは

- 事前確率 $P(\omega_i)$
  - 特徴ベクトルを観測する前の、各クラスの起こりやすさ
- 事前確率の最尤推定

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

N: 全データ数、n_i: クラス ω_i のデータ数

4.1 統計的識別とは

- 尤度 $P(\boldsymbol{x}|\omega_i)$
  - 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
- *d* 次元ベクトルの場合の最尤推定
  - 値の組合せがデータ中に出現しないもの多数

Viewer

Relation: weather.symbolic

No.	1: outlook Nominal	2: temperature Nominal	3: humidity Nominal	4: windy Nominal	5: play Nominal
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	overcast	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

Undo OK Cancel

Weka の weather.nominal データ

$3 \times 3 \times 2 \times 2 = 36$ 種類の組合せ

4.2 ベイズ識別

4.2.1 学習データの対数尤度

- データの尤度
  - データを生成するモデルを考え、そのモデルがパラメータ $\boldsymbol{\theta}$ に従ってデータを生成していると仮定

$$P(\boldsymbol{x}|\omega_i, \boldsymbol{\theta})$$

以後、1 クラス分のデータを全データとみなす

  - 全データは、それぞれ独立に生成されていると仮定
    - i.i.d (independent and identically distributed)

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

4.2.1 学習データの対数尤度

- 対数尤度
  - 確率の積のアンダーフローを避けるため、対数尤度で計算

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

4.2.1 学習データの対数尤度

- 最尤推定法
  - 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-θ で値 0 をとる分布

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

4.2.1 学習データの対数尤度

- 対数尤度を最大にするパラメータ
  - $\dfrac{d\mathcal{L}(D)}{d\theta} = 0$ の解を求める

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

値 x_i をとる回数を全データ数で割ったもの

4.2.2 ナイーブベイス識別

- ナイーブベイズの近似
  - 全ての特徴が独立であると仮定

$$
\begin{aligned}
P(\boldsymbol{x}|\omega_i) &= P(x_1, \ldots, x_d|\omega_i) \\
&\approx \prod_{j=1}^{d} P(x_j|\omega_i)
\end{aligned}
$$

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

4.2.2 ナイーブベイス識別

- 尤度の最尤推定

$$P(x_j|\omega_i) = \frac{n_{ij}}{n_i}$$

n_{ij}: クラス ω_i のデータのうち、
j 次元目の値が x_j の個数

ゼロ頻度問題

- 確率の m 推定

$$P(x_j|\omega_i) = \frac{n_{ij} + mp}{n_i + m}$$

p: 事前に見積もった各特徴値の割合
m: 事前に用意する標本数

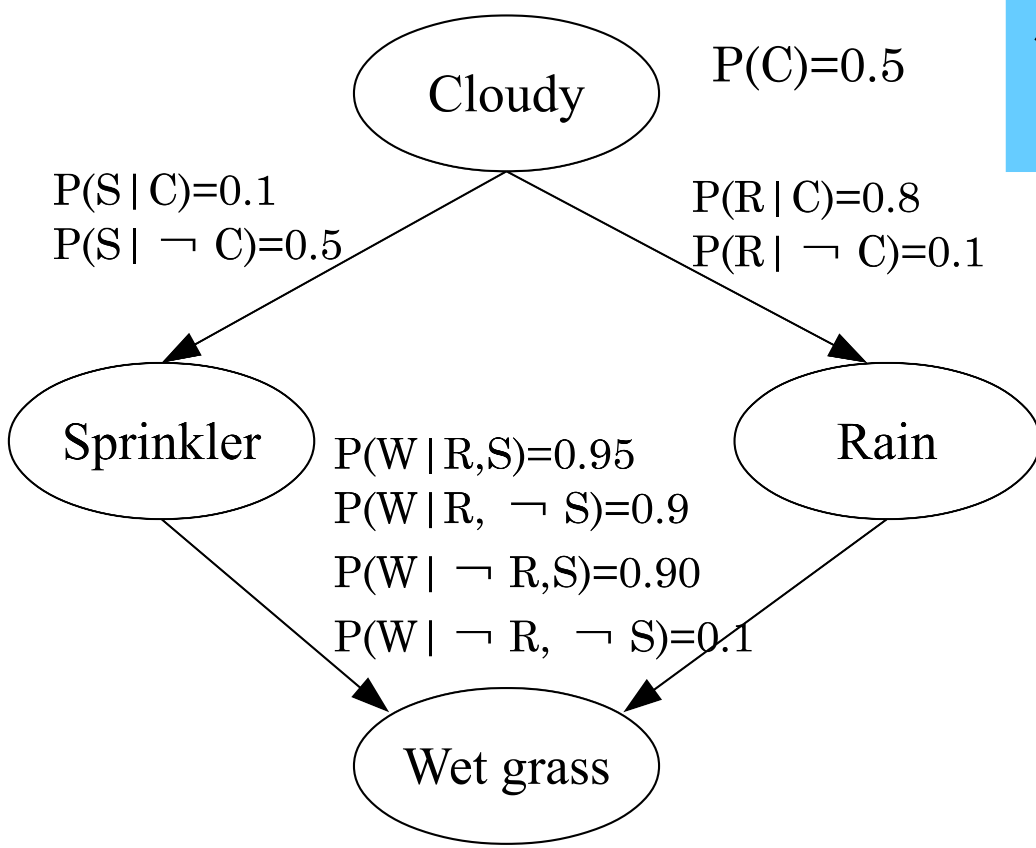
- ラプラス推定
  - m: 特徴値の種類数、 p: 等確率　とすると、 $mp=1$

4.3 ベイジアンネットワーク

- 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である

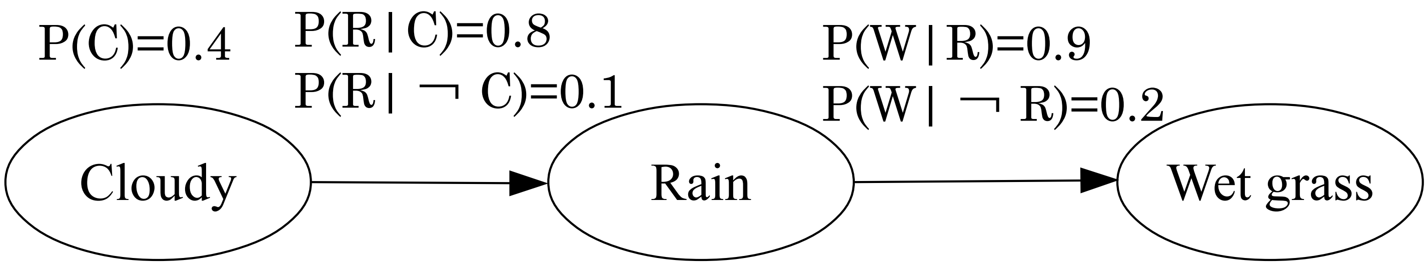
$$P(x_1, \ldots, x_d) \approx \prod_{i=1}^{d} P(x_i | Parents(X_i))$$

値 x_i をとるノードの
親ノードの値



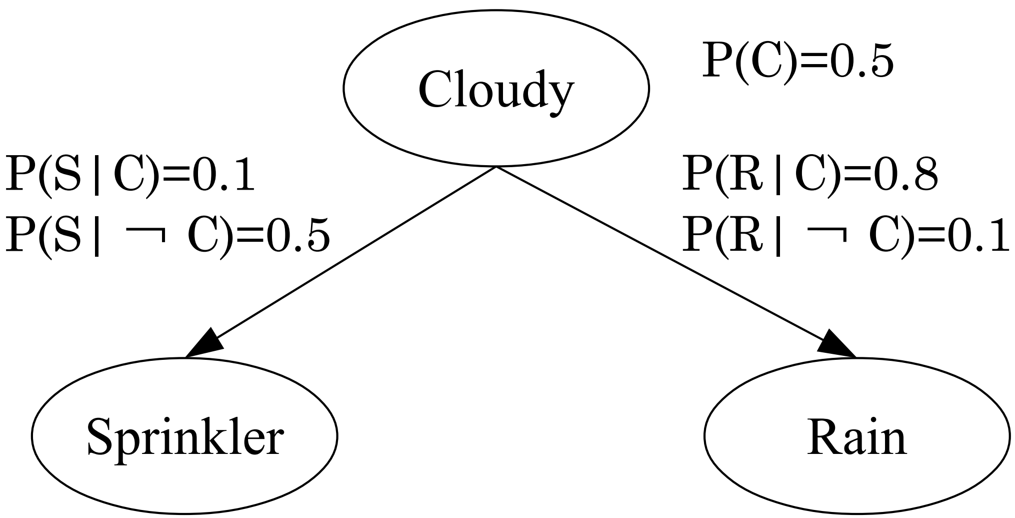
4.3 ベイジアンネットワーク

- ベイジアンネットワークの構成
  - Head-to-tail



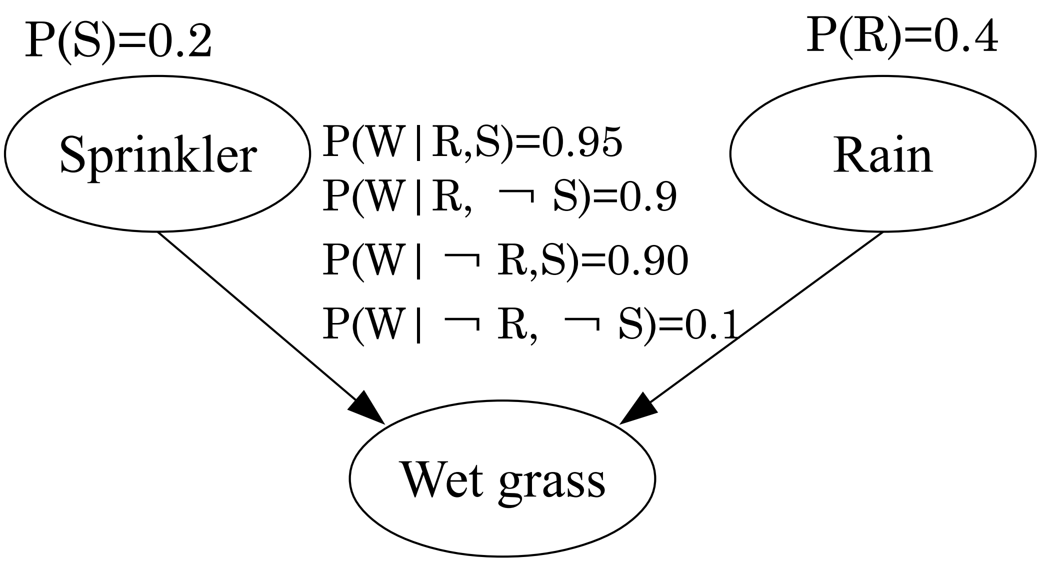
4.3 ベイジアンネットワーク

- ベイジアンネットワークの構成
  - Tail-to-tail



4.3 ベイジアンネットワーク

- ベイジアンネットワークの構成
  - Head-to-head



4.3 ベイジアンネットワーク

- ベイジアンネットワークの学習

Algorithm 4.2 K2 アルゴリズム

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）
repeat
  for all $n \in Node$ **do**
    for all $n+1$ 以降のノード **do**
      if 対数尤度が増加 **then**
        n から、現在のノードへエッジを作成
      end if
    end for
  end for
  ノードの順番を変える
until 対数尤度が変化しない
return 学習されたベイジアンネット
