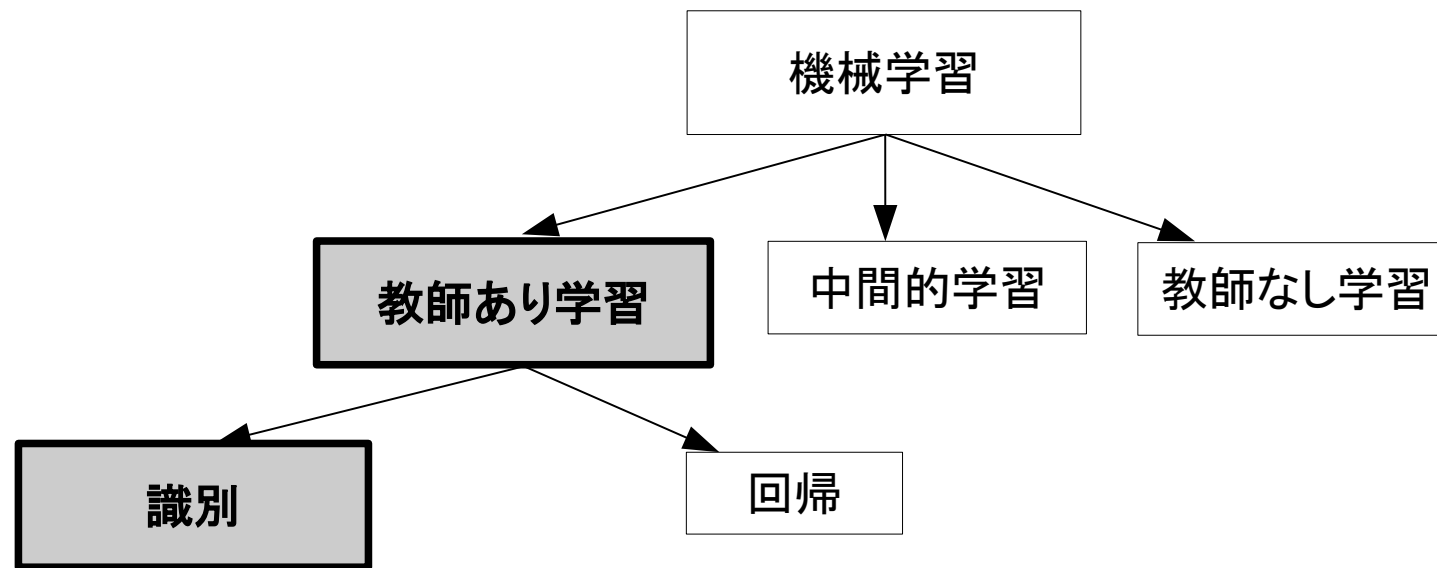
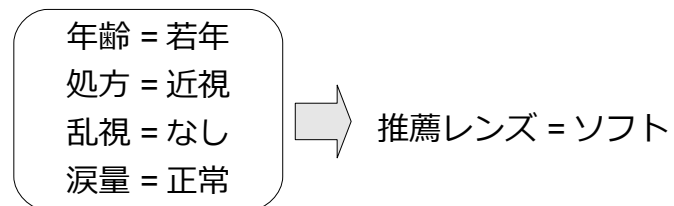


# 3. 識別 —概念学習—

- 問題設定
  - 教師あり学習
  - カテゴリ入力 → カテゴリ出力



- カテゴリ特徴



- 数値特徴

# contact-lenses データ

年齢・処方・  
乱視・涙量

No	age	spectacle-prescrip	astigmatism	tear-prod	contact-lenses
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none
6	young	hypermetrope	no	normal	soft
7	young	hypermetrope	yes	reduced	none
8	young	hypermetrope	yes	normal	hard
9	pre-presbyopic	myope	no	reduced	none
10	pre-presbyopic	myope	no	normal	soft
11	pre-presbyopic	myope	yes	reduced	none
12	pre-presbyopic	myope	yes	normal	hard
13	pre-presbyopic	hypermetrope	no	reduced	none
14	pre-presbyopic	hypermetrope	no	normal	soft
15	pre-presbyopic	hypermetrope	yes	reduced	none
16	pre-presbyopic	hypermetrope	yes	normal	none
17	presbyopic	myope	no	reduced	none
18	presbyopic	myope	no	normal	none
19	presbyopic	myope	yes	reduced	none
20	presbyopic	myope	yes	normal	hard
21	presbyopic	hypermetrope	no	reduced	none
22	presbyopic	hypermetrope	no	normal	soft
23	presbyopic	hypermetrope	yes	reduced	none
24	presbyopic	hypermetrope	yes	normal	none


推薦コンタクトレンズ  
none, soft, hard

# contact-lenses データ

表 3.2 コンタクトレンズデータの特徴値

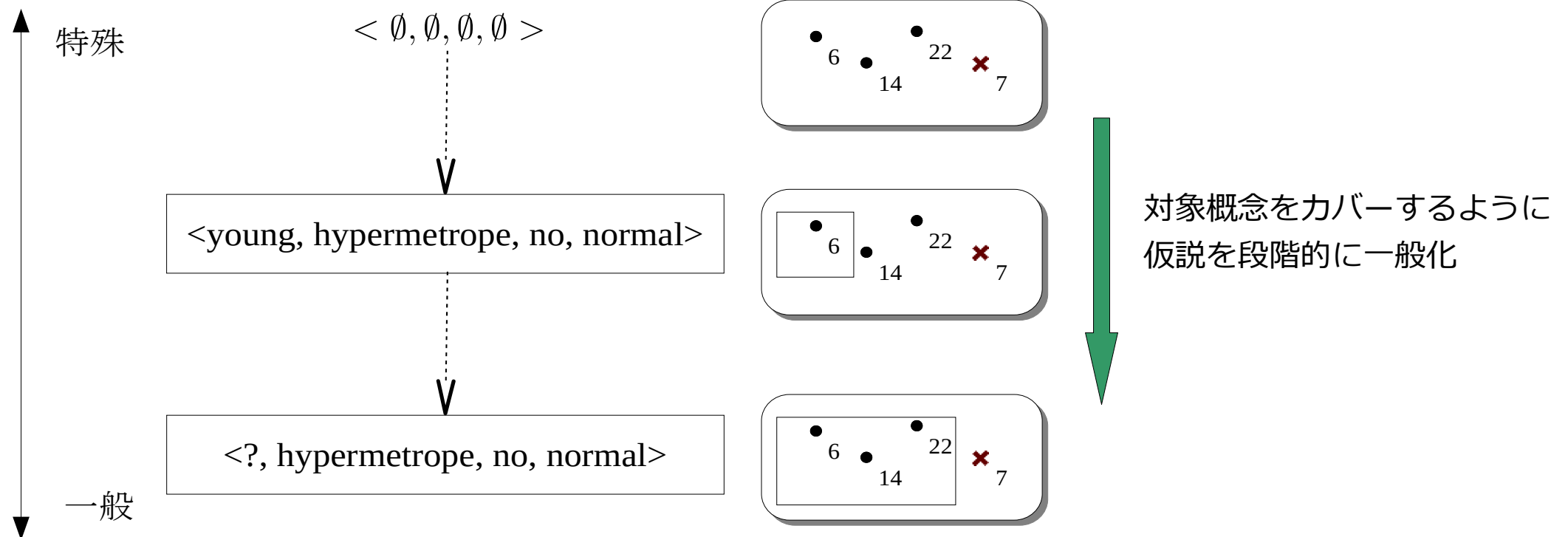
特徴	値
age(年齢)	{young, pre-presbyopic, presbyopic} (若年, 老眼前期, 老眼)
spectacle-prescrip(眼鏡)	{myope, hypermetrope} (近視, 遠視)
astigmatism(乱視)	{no, yes} (なし, あり)
tear-prod-rate(涙量)	{reduced, normal} (減少, 正常)
contact-lenses(クラス)	{soft, hard, none} (ソフト, ハード, なし)

## 3.2 概念学習とバイアス

- 概念学習とは
  - 正解の概念を説明する特徴ベクトルの性質（論理式）を求めること
  - 論理式の例  
 $(\text{乱視} = \text{あり}) \wedge (\text{ドライアイ} = \text{なし}) \Rightarrow \text{soft}$
- 学習の方法
  - 可能な論理式が少数
    - 正解概念の候補を絞り込んでゆく（候補削除アルゴリズム）
  - 可能な論理式が多数
    - バイアス（偏見）をかけて探索する  決定木

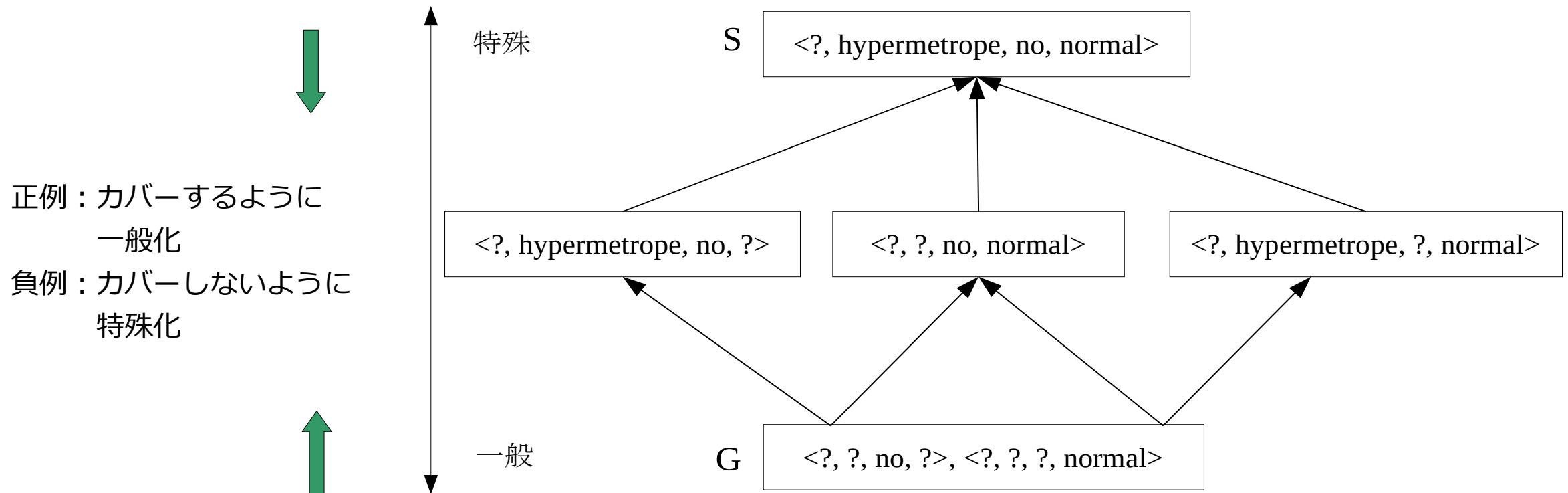
## 3.2.1 初期の概念学習

### FIND-S アルゴリズム



## 3.2.1 初期の概念学習

### 候補削除アルゴリズム

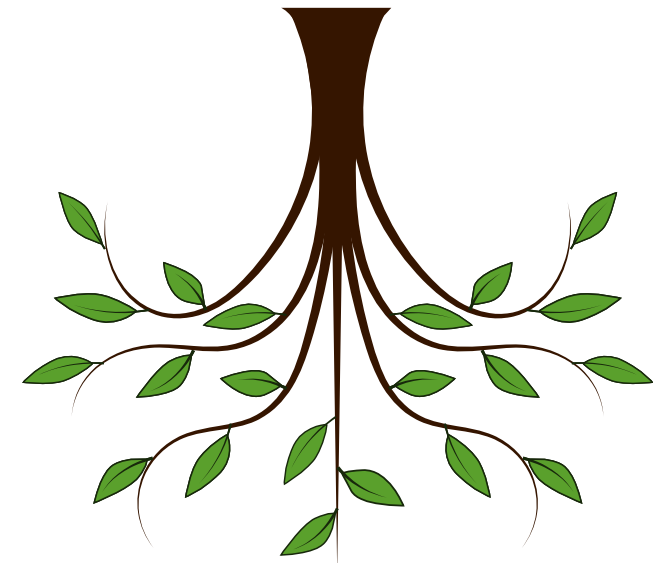
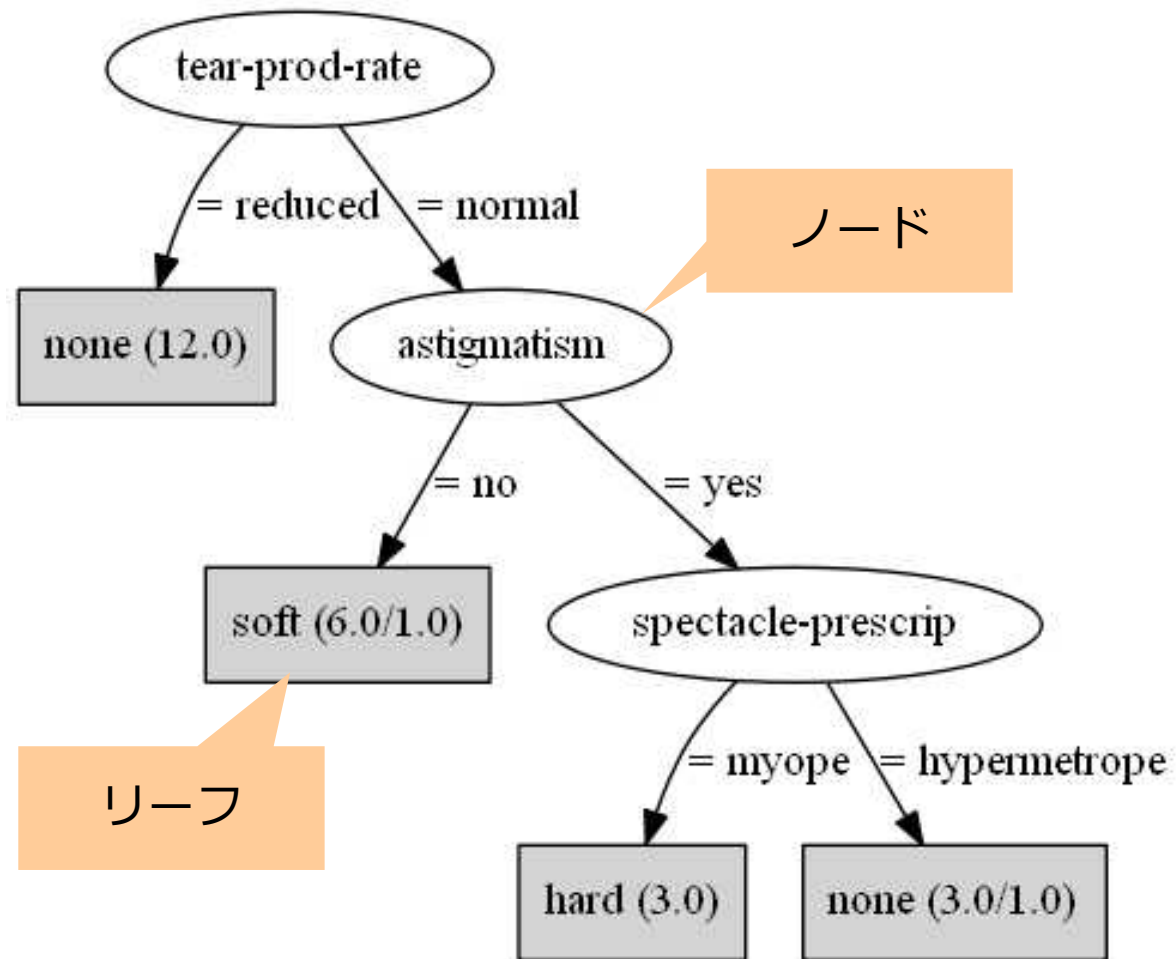


## 3.2.2 概念学習のバイアスを考える

- 初期の概念学習
  - 学習対象の概念にバイアスをかけて探索
    - 可能な論理式をリテラル（属性名＝値）の AND 結合に限定
  - 正解概念の候補数：  $4 \times 3 \times 3 \times 3 + 1 = 109$
- 問題点
  - リテラルの OR 結合が表現できないので、正解概念が探索空間内に存在しないことが多い
  - 解決策として OR 結合を許すと、正例の OR 結合が自明な解となり、未知事例に対して判定する材料を持たない
  - 正解概念の候補数： 2 の事例数乗  $2^{24} = 16777216$

## 3.3 決定木の学習

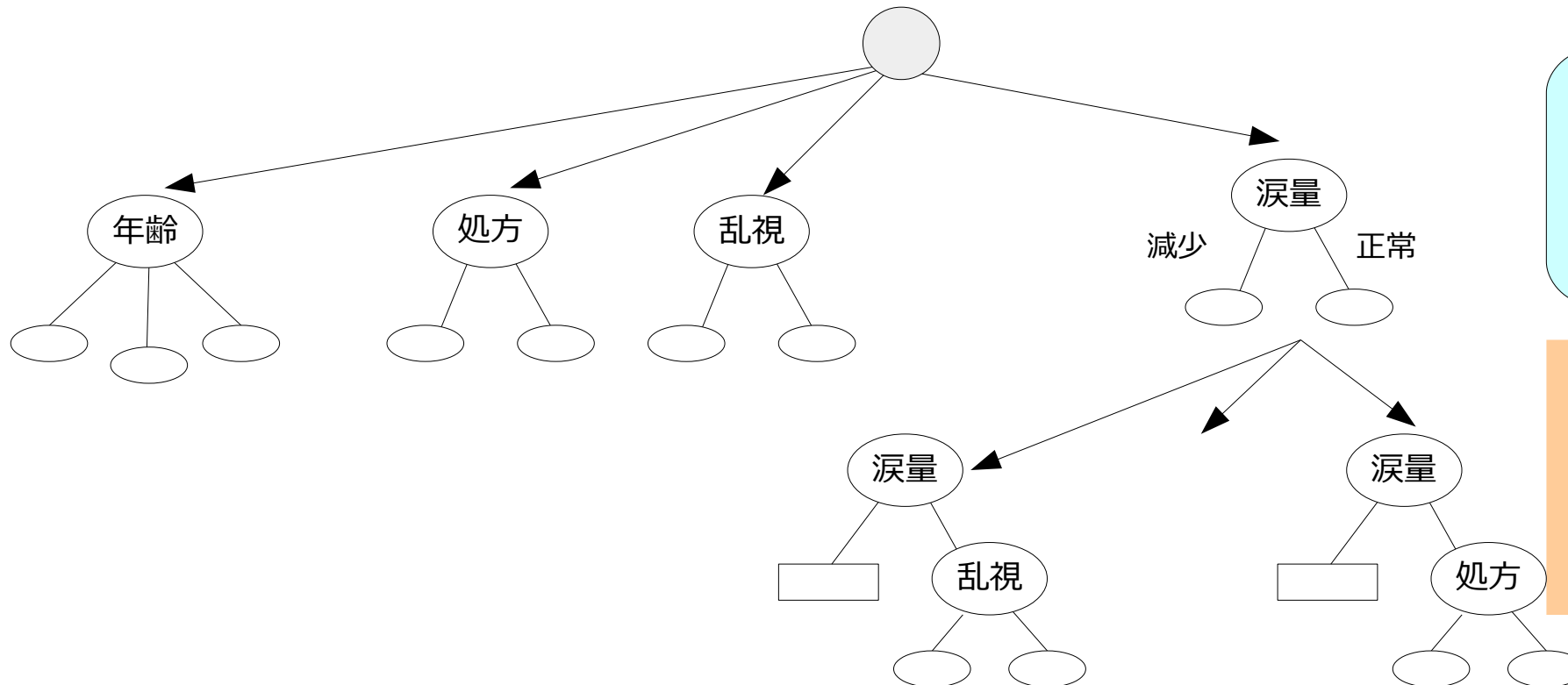
- 決定木の例





## 3.3.1 決定木とは

- 決定木学習の考え方
  - ノードは、データを分割する条件を持つ
    - できるだけ同一クラスのデータがリーフに偏るように
  - 分割後のデータ集合に対して、同様の操作を行う
  - 全てのリーフが単一クラスの集合になれば終了



この手順に従うと、  
一般には小さな木  
ができる

**バイアス**  
複雑な説明よりも  
単純な説明の方が  
汎用性が高い

# 決定木の構築 (1/2)

---

## Algorithm 3.1 ID3 アルゴリズム

---

入力: 正解付き学習データ  $D$ , クラス特徴  $y$ , 特徴集合  $A$

出力: 決定木  $T$

root ノードを作成

**if**  $D$  が全て正例 **then**

**return** ラベル Yes

**else if**  $D$  が全て負例 **then**

**return** ラベル No

**else if** 特徴集合  $A == \emptyset$  (空集合) **then**

**return**  $D$  中の最頻値のクラス

**else**

# 決定木の構築 (2/2)

$a \leftarrow A$  中で最も分類能力の高い特徴

root ノードの決定特徴  $\leftarrow a$

**for all**  $a$  の取りうる値  $v$  **do**

$a = v$  に対応する枝を作成

データの中から値  $v$  を取る部分集合  $D_v$  を作成

**if**  $D_v == \emptyset$  **then**

**return**  $D$  中の最頻値のクラス

**else**

ID3(部分集合  $D_v$ , クラス特徴  $y$ , 特徴集合  $A - a$ )

**end if**

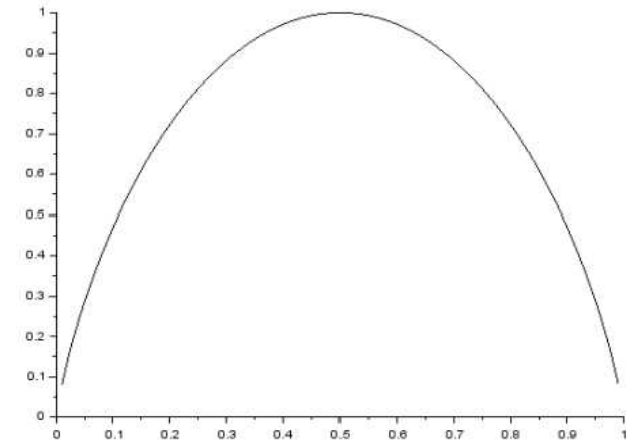
**end for**

**end if**

**return** root ノード

# 属性の分類能力 (1/2)

- 分類能力の高い属性を決定する方法
  - その属性を使った分類を行うことによって、なるべくきれいにクラスが分かれるように
- エントロピー
  - データ集合  $S$  の乱雑さを表現
  - 正例の割合 :  $p^+$  , 負例の割合 :  $p^-$
  - エントロピーの定義



$$Entropy(S) = -p^+ \log p^+ - p^- \log p^-$$

# 属性の分類能力 (2/2)

- 情報獲得量
  - 属性  $A$  を用いた分類後のエントロピーの減少量
  - 値  $v$  を取る訓練例の集合 :  $S_v$
  - $S_v$  の要素数 :  $|S_v|$
  - 情報獲得量の定義

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

# 決定木構築手順の例

- 学習データ ( ゴルフをする日 ; weather.nominal.arff)

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	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

# 決定木構築手順の例

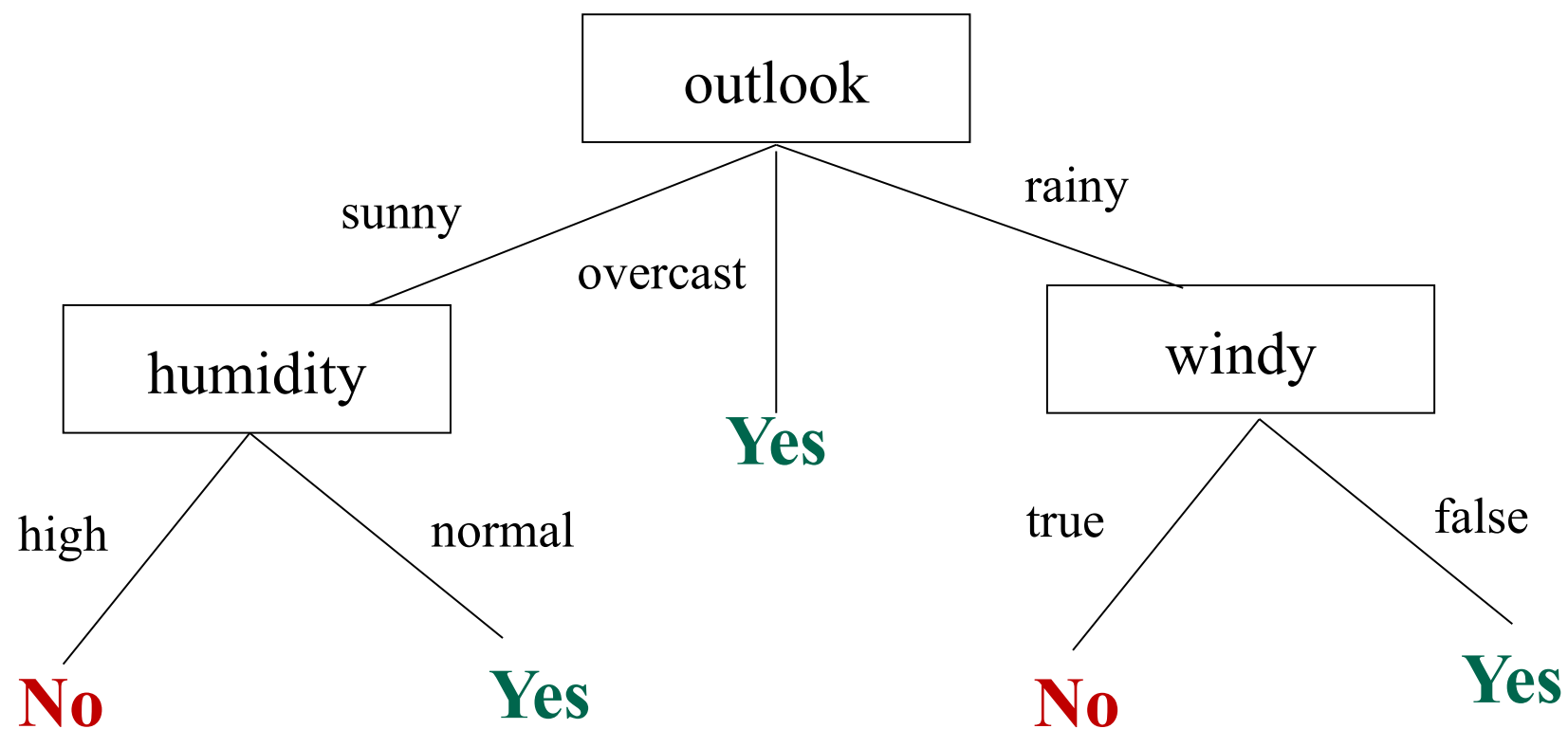
- 特徴

表 3.4 weather.nominal データの特徴値

特徴	値
outlook(天候)	{sunny, overcast, rainy} (晴, 曇, 雨)
temperature(気温)	{hot, mild, cool} (高温, 適温, 低温)
humidity(湿度)	{high, normal} (多湿, 標準)
windy(風)	{TRUE, FALSE} (あり, なし)
play(クラス)	{yes, no} (正例, 負例)



# 結果として得られる決定木





# 計算例

- 情報獲得量

Gain(S, outlook)=0.246

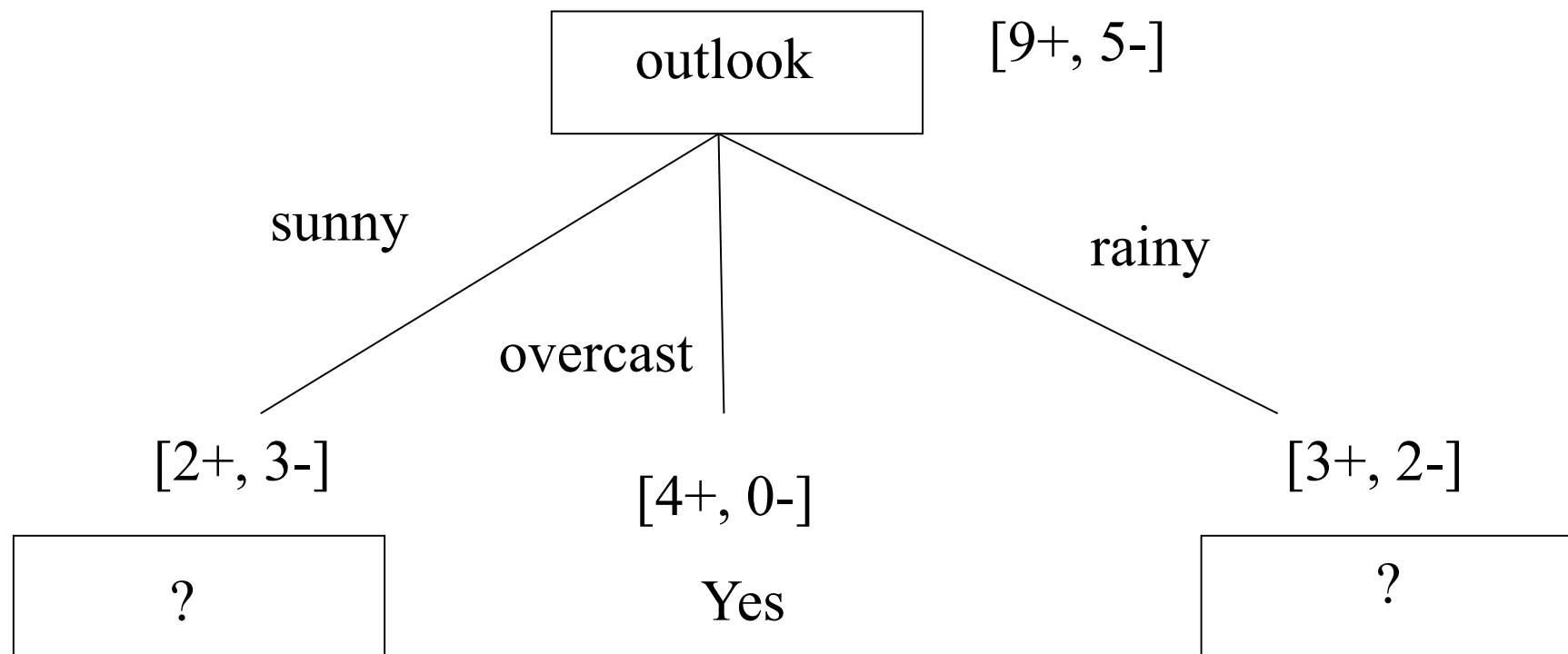
Gain(S, humidity)=0.151

Gain(S, windy)=0.048

Gain(S, temperature)=0.029

$$E(D) = -P_+ \log_2 P_+ - P_- \log_2 P_-$$

$$Gain(D, a) \equiv E(D) - \sum_{v \in Values(a)} \frac{|D_v|}{|D|} E(D_v)$$



### 3.3.3 過学習を避ける

#### なぜ単純な木の方がよいか

- オッカムの剃刀

「データに適合する最も単純な仮説を選べ」

- 複雑な仮説

- 表現能力が高い

- 偶然にデータを説明できるかもしれない

- 単純な仮説

- 表現能力が低い

- 偶然にデータを説明できる確率は低い

- でも説明できた！

- **必然**

### 3.3.3 過学習を避ける

- 学習過程での制御
  - 木の成長を止める
    - 葉に所属する最小データ数を多くする
    - 木の段階の最大数を決めておく
- 枝刈り
  1. 学習用データを用いてできるだけ成長した木を作成する
  2. 検証用データを用いて、あるノード以降の分類性能が、そのノードに属するデータの多数決を用いた分類よりも劣れば、その多数クラスをラベルとする葉に置き換える

## 3.4 数値特徴に対する決定木

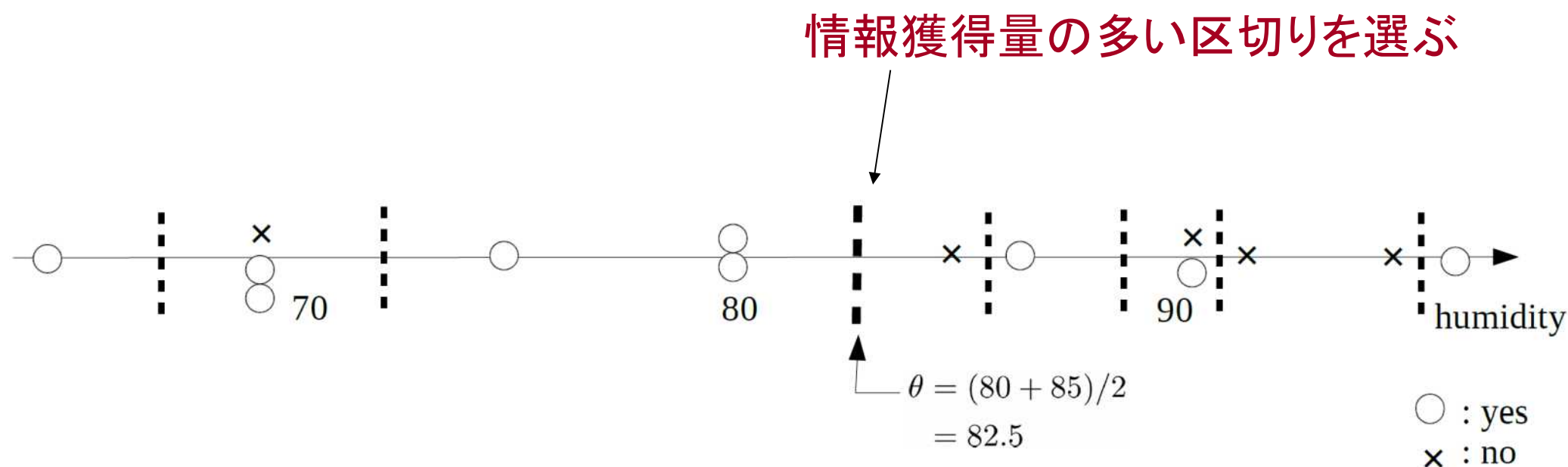
- weather.numeric.arff

表 3.5 weather データ（数値特徴・カテゴリ特徴が混在）

No.	outlook	temperature	humidity	windy	play
1	sunny	85	85	FALSE	no
2	sunny	80	90	TRUE	no
3	overcast	83	86	FALSE	yes
4	rainy	70	96	FALSE	yes
5	rainy	68	80	FALSE	yes
6	rainy	65	70	TRUE	no
7	overcast	64	65	TRUE	yes
8	sunny	72	95	FALSE	no
9	sunny	69	70	FALSE	yes
10	rainy	75	80	FALSE	yes
11	sunny	75	70	TRUE	yes
12	overcast	72	90	TRUE	yes
13	overcast	81	75	FALSE	yes
14	rainy	71	91	TRUE	no

## 3.4 数値特徴に対する決定木

- 連続値  $A$  を持つ属性から真偽値 ( $A < c?$ ) を値とするノードを作成  
→  $c$  をどうやって決めるか



$$\begin{aligned}\text{Gain}(D, \text{humidity}_{82.5}) &= 0.94 - \frac{7}{14} \text{Entropy}(D, < 82.5) - \frac{7}{14} \text{Entropy}(D, \geq 82.5) \\ &= 0.152\end{aligned}$$

# まとめ

- Weka デモ
  - contact-lenses （カテゴリ特徴）
  - weather.nominal （カテゴリ特徴）
  - weather.numeric （ 数値・カテゴリの混合特徴 ）
- 決定木の学習
  - 分割後のデータ集合ができるだけ均一となるような特徴を選んで、分割規則を学習
  - 一般に小さめの木の方が汎化能力が高い