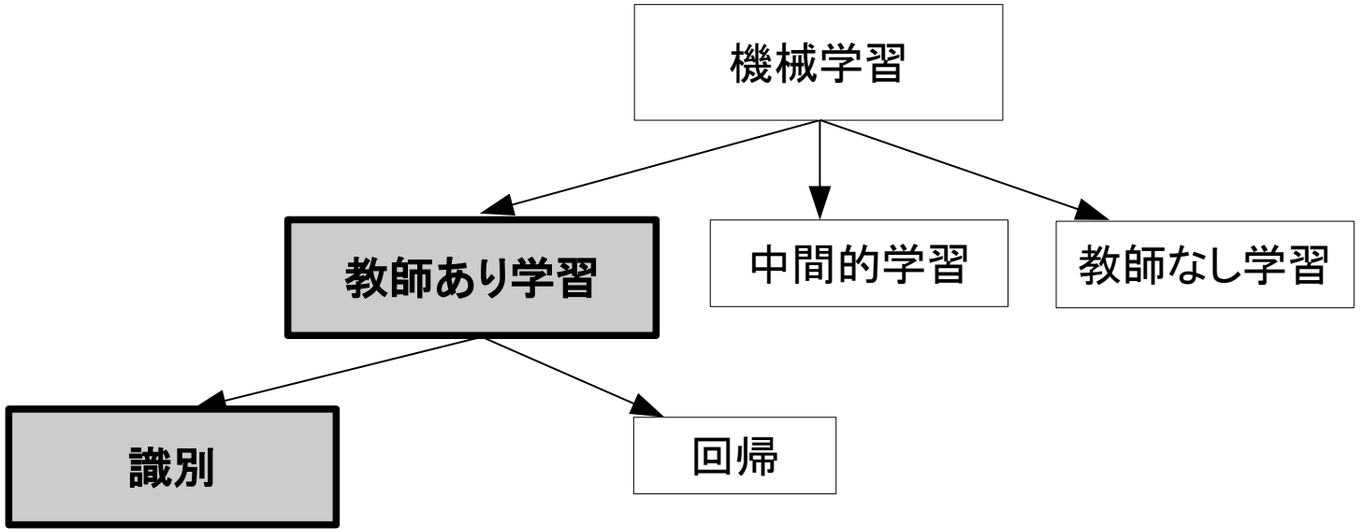
# 3. 識別 ー概念学習ー

- 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

年齢 = 若年
処方 = 近視
乱視 = なし
涙量 = 正常

⇒ 推薦レンズ = ソフト

- 数値特徴

# 3.1 カテゴリ特徴に対する「教師あり・識別」問題の定義

- 「教師あり・識別」問題のデータ
  - 特徴ベクトル $\boldsymbol{x}$ と正解情報 $y$ のペア

    $$\{(\boldsymbol{x}_i, y_i)\}, \quad i = 1 \ldots N$$

  - 特徴ベクトルは次元数 $d$ の固定長ベクトル

    $$\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})^T$$

  - カテゴリ形式の正解情報を**クラス**とよぶ
- 学習の目的
  - クラスを説明する概念モデルを得る（→ 3 章）
  - 与えられた特徴が、あるクラスに属する確率を計算するモデルを得る（→ 4 章）

# contact-lenses データ

特徴
年齢・眼鏡・乱視・涙量

Viewer

elation: contact-lenses

| No. | 1: age Nominal | 2: spectacle-prescrip Nominal | 3: astigmatism Nominal | 4: tear-prod-rate Nominal | 5: contact-lenses Nominal |
|---|---|---|---|---|---|
| 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 |

Undo OK Cancel

推薦コンタクトレンズ
none, soft, hard

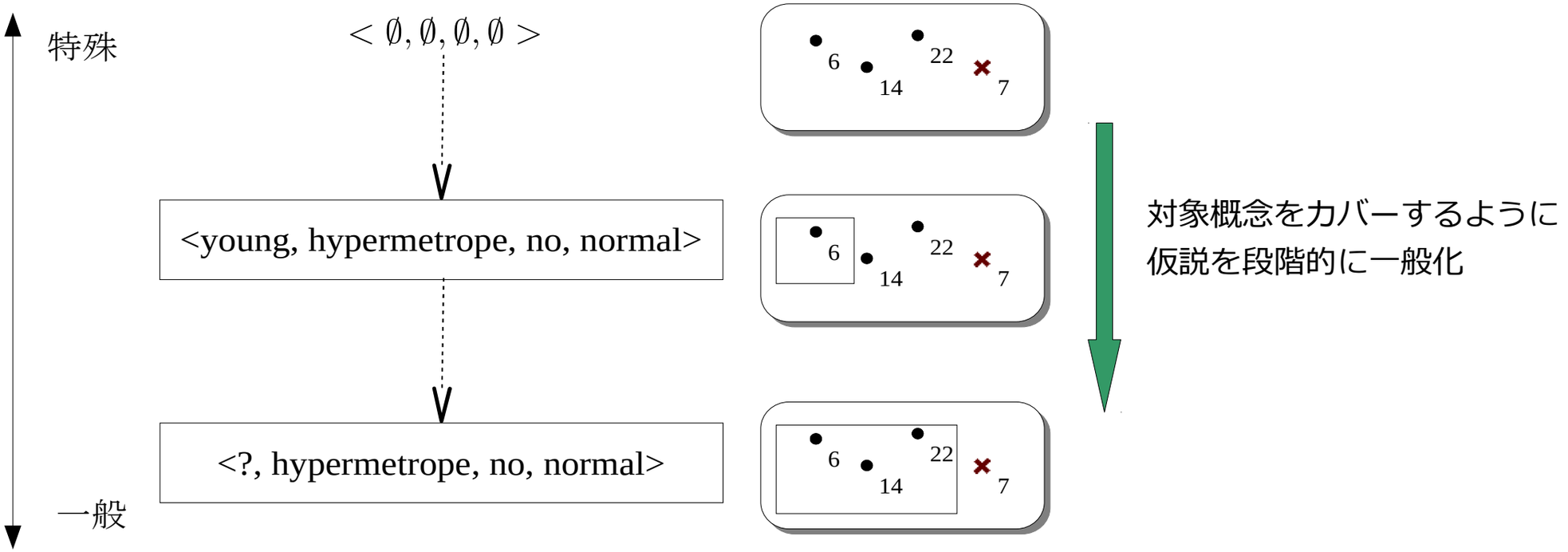
# 3.2 概念学習とバイアス

- 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

    （乱視 = なし） ∧ （涙量 = 減少）⇒ 使用を勧めない
- 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

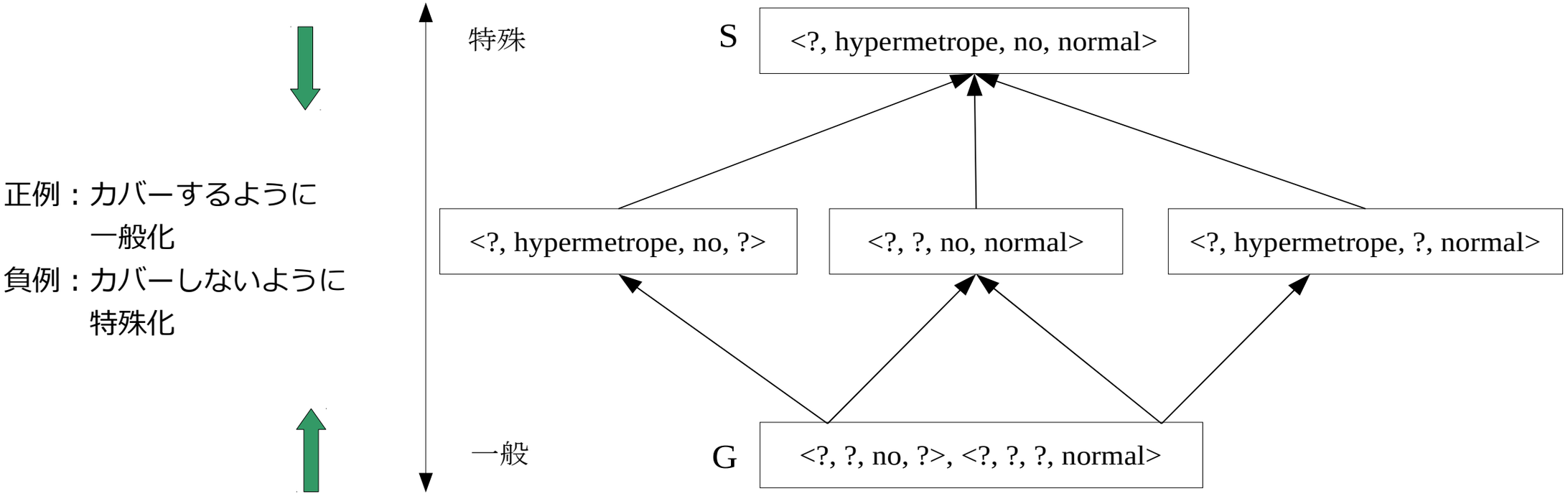
# 3.2.1 初期の概念学習

FIND-S アルゴリズム



# 3.2.1 初期の概念学習

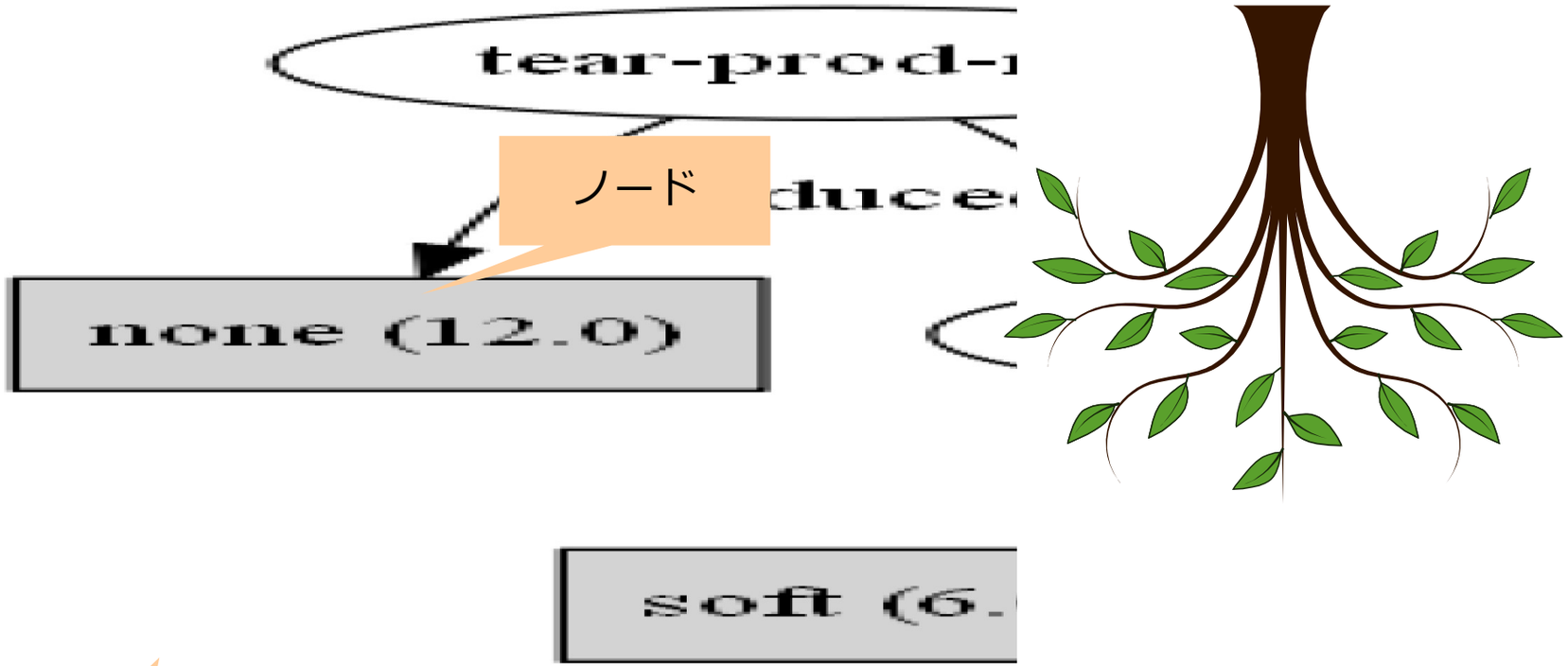
候補削除アルゴリズム




- 正解となり得る概念に強いバイアス（条件の AND 結合）をかけているため、正解が得られないことが多い

# 3.3 決定木の学習

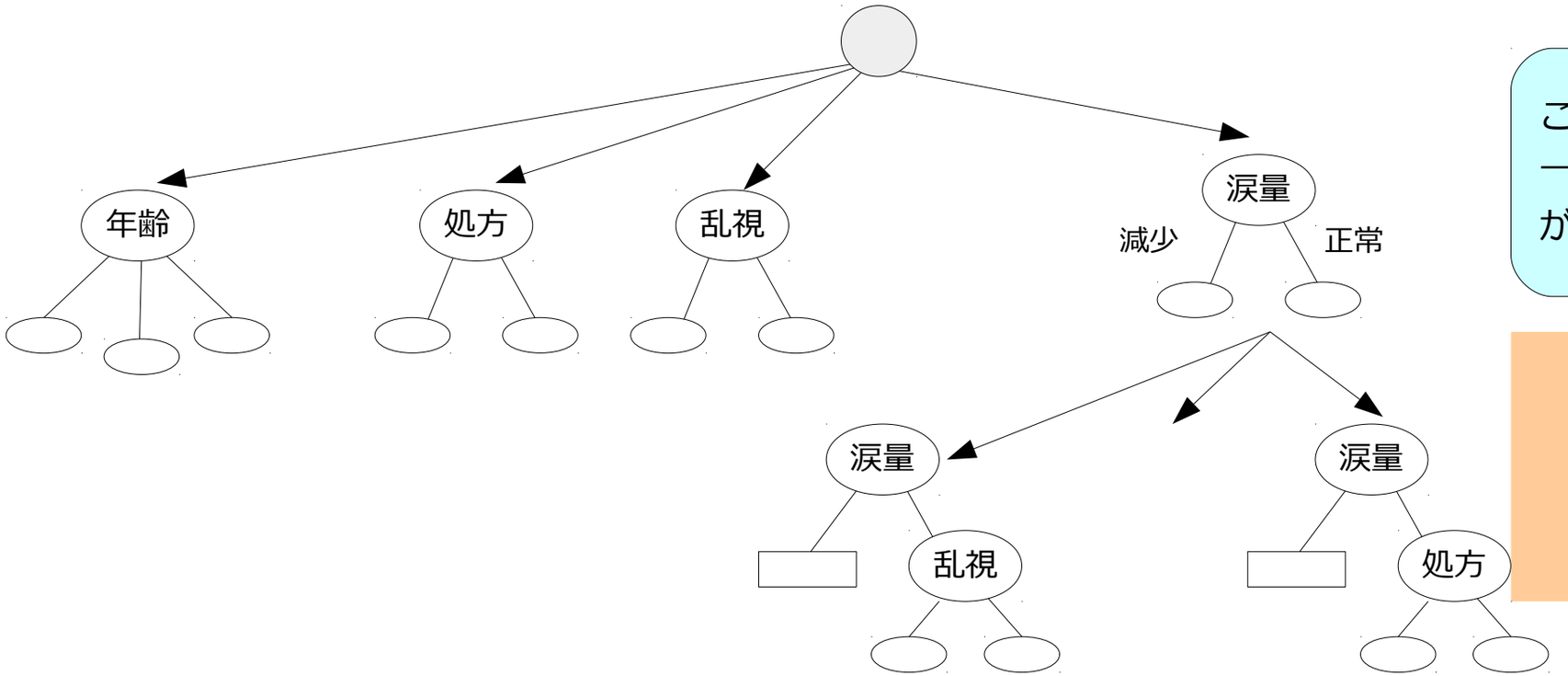
- 学習した決定木の例




概念の OR が表現
できている

# 3.3 決定木の学習

- 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**

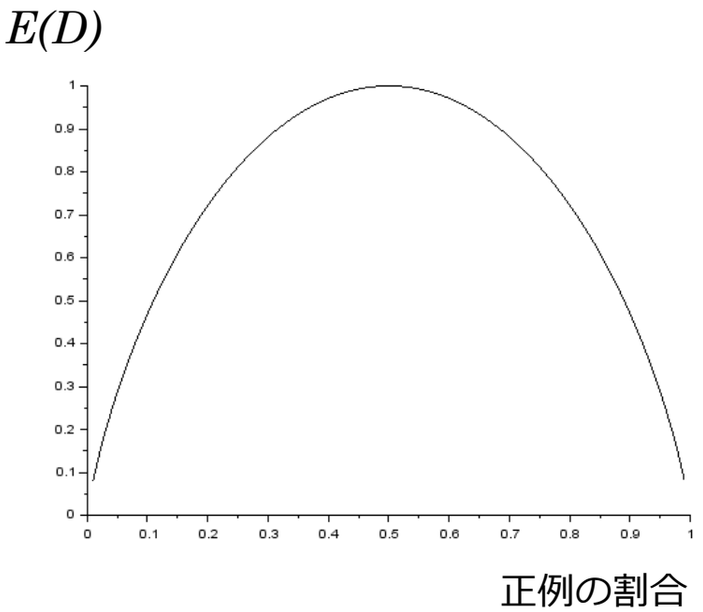
# 決定木の構築 (2/2)

$a \leftarrow A$ 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 $D$ 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

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



# 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：$Dv$
  - $Dv$ の要素数：$|Dv|$
  - 情報獲得量の定義

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

# 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

- 枝刈りによる過学習の回避
  - 学習データを正しく識別できるまで木を成長させる
  - 下記手順で枝刈りを行う

**Algorithm 3.2** 決定木の枝刈りアルゴリズム

入力: 学習済みの決定木 $T$, 検証用データ $D'$

出力: 枝刈り後の決定木 $T'$

**for all** $T$ のノード $N$, ルートから遠いものから順に **do**

$T_N \leftarrow N$ をルートとする木

$D_N \leftarrow D'$の中で, $T_N$によってカバーされるデータ

**if** $\mathrm{accuracy}(T_N, D_N) < \mathrm{majority}(D_N)$ **then**

$T_N$を，リーフ $\mathrm{majority\_class}(D_N)$ に置き換え

**end if**

**end for**

**return** 枝刈り後の決定木 $T'$

# 3.3.4 分類基準の再検討

- 獲得率

$$\mathrm{SplitInformation}(D, a) \equiv -\sum_{v \in \mathrm{Values}(a)} \frac{|D_v|}{|D|} \log_2 \frac{|D_v|}{|D|}$$

$$\mathrm{GainRatio}(D, a) \equiv \frac{\mathrm{Gain}(D, a)}{\mathrm{SplitInformation}(D, a)}$$

- ジニ不純度

$$\mathrm{GiniImpurity}(D) \equiv 2 \cdot P_+ \cdot P_-$$

- Root Gini Impurity

$$\mathrm{RootGiniImpurity}(D) \equiv \sqrt{P_+ \cdot P_-}$$

# 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

  → 最もエントロピーが低くなる分割を行う