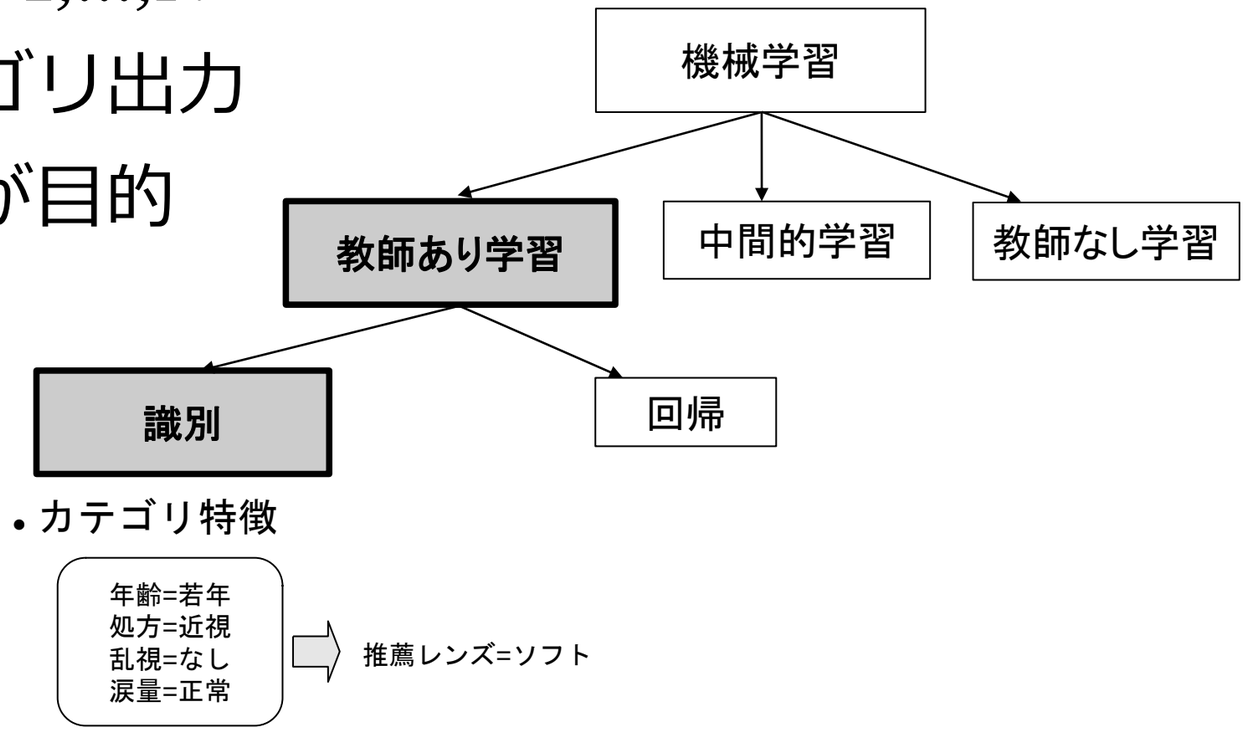
3. 識別 ―概念学習―

3.1 カテゴリ特徴に対する「教師あり・識別」問題の定義

- 問題設定
  - 教師あり学習　$\{(\boldsymbol{x}_i, y_i)\}$　i=1,...,N
  - カテゴリ入力　→　カテゴリ出力
  - クラスの概念を得ることが目的



contact-lensesデータ

表 3.1　コンタクトレンズデータ (contact-lens.arff)

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

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

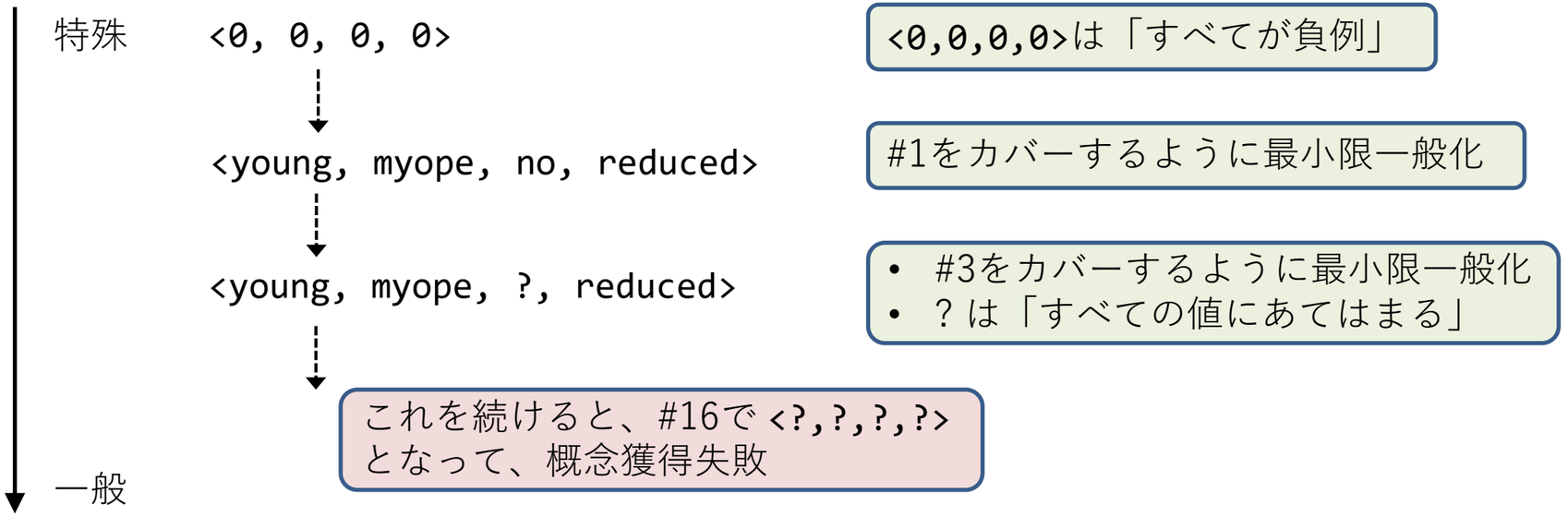
- 概念学習とは
  - 正解の概念を説明する特徴ベクトルの性質（論理式）を求めること
  - 「softレンズを勧める」という概念の例

（乱視=あり） ∧ （ドライアイ=なし） ⇒ soft

- 学習の方法
  - 初期の概念学習：学習対象の概念にバイアス（偏見）をかけて絞り込み
    - FIND-Sアルゴリズム
    - 候補削除アルゴリズム
  - **決定木学習**：探索方法にバイアスをかけて準最適解を探す

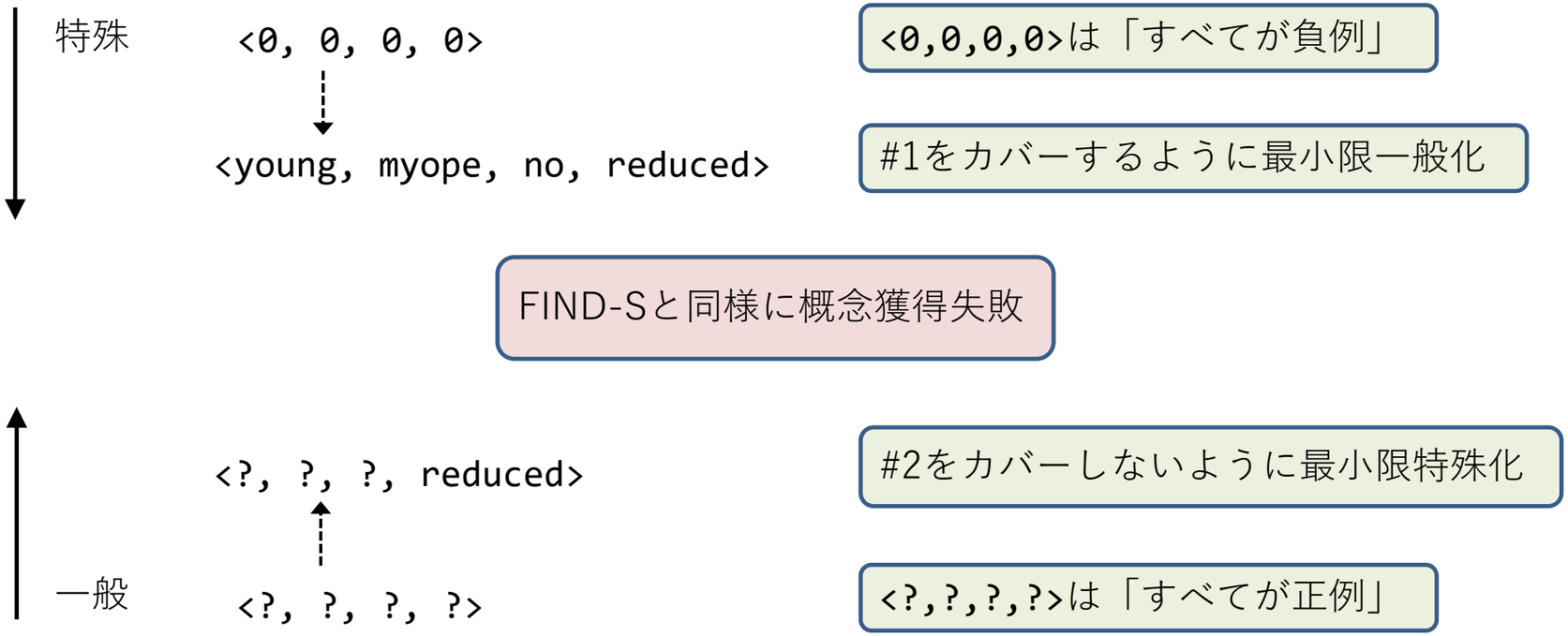
3.2.1 初期の概念学習

- FIND-Sアルゴリズム
  - 最も特殊な概念から始めて、正例を使って順次一般化する
    - 求める論理式を「リテラル（特徴名＝値）のAND結合（∧）」に限定
    - すべての正例をカバーする論理式が得られれば終了
    - 正解概念の候補（仮説空間）：4×3×3×3+1=109
  - 例：「コンタクトレンズの使用を勧めない」(none) の概念を獲得したい



3.2.1 初期の概念学習

- 候補削除アルゴリズム
  - FIND-Sに加えて、もっとも一般的な概念 <?,?,?,?>を負例をカバーしないように順次特殊化
  - すべてのデータの処理が終わって、残った論理式集合が正解候補



3.2.2 概念学習のバイアスを考える

- 初期の概念学習の問題点
  - リテラルのOR結合（∨）が表現できないので、正解概念が仮説空間内に存在しないことが多い

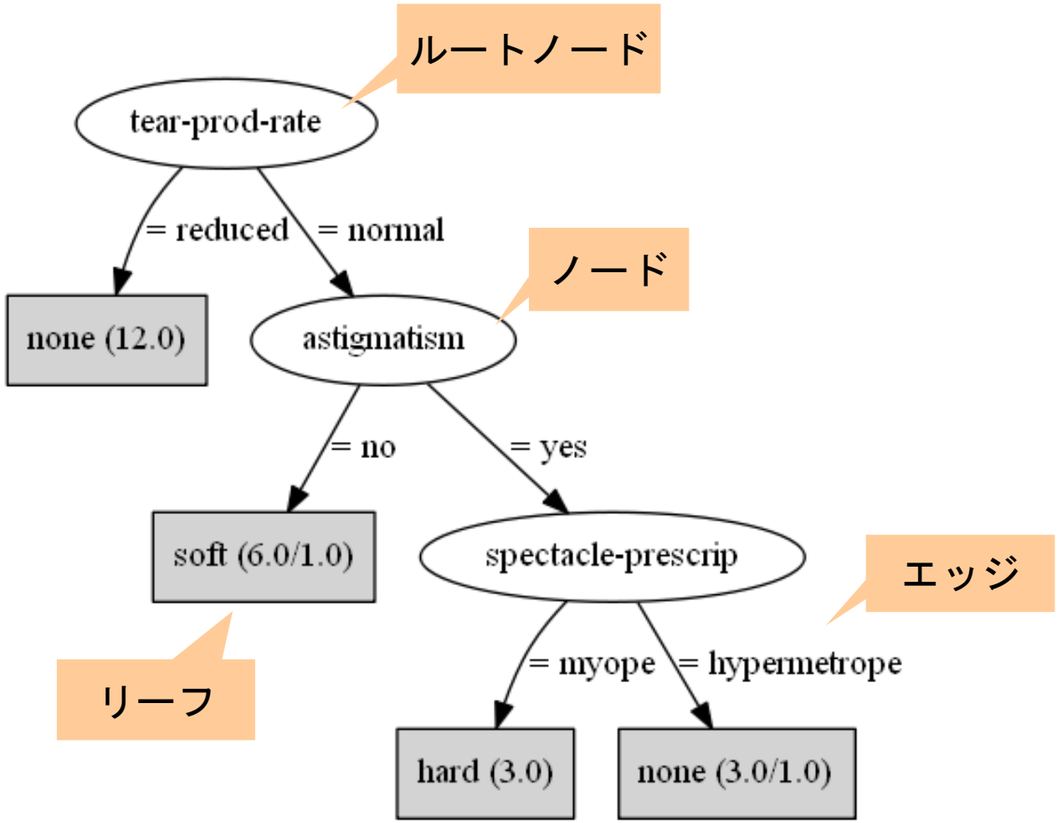
    例）（年齢=若年 ∨ 年齢=老眼前期） が表現できない

3.2.2 概念学習のバイアスを考える

- 単純にリテラルのOR結合を許す場合
  - 正解概念の候補数が増大する
    - 2の事例数乗 $2^{24}=16777216$
  - 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根拠を持たない
- 解決策
  - 仮説空間はリテラルのOR結合を許し、探索方法にバイアスをかけて候補を限定する
  - 見つかった候補が未知データに対しても適用可能な概念であるようなバイアスとは何か

3.3 決定木の学習

- 概念を決定木で表す
  - ルートノードから始めて、特徴の値によって事例を分類する
  - リーフに至れば分類は終了
- 決定木の要素と意味
  - ノード（節）：特徴
  - エッジ（枝）：値
  - リーフ（葉）：出力



3.3.1 決定木とは

- 決定木学習の作り方
  - ノードとする特徴を決める
    - 分割後のデータができるだけ同一クラスが偏るように特徴を選択する
  - 特徴の値に基づいてデータを分割する
  - すべてのデータが単一のクラスになればリーフとする
  - そうでなければ、分割後のデータ集合に対して同様の操作を行う
    - ただし、これまでに使用した特徴は選択しない

3.3.2 ID3アルゴリズム

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

$a \leftarrow A$ 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

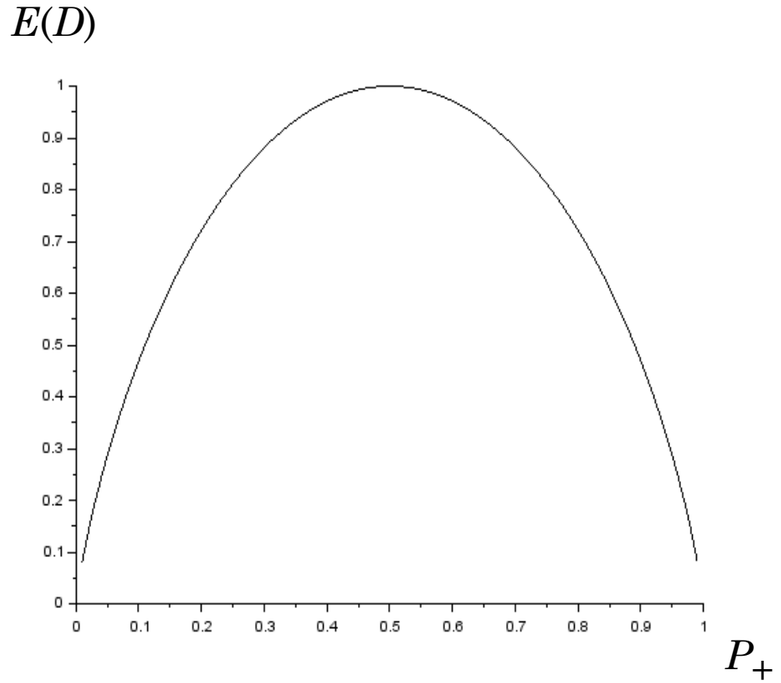
end if

return root ノード

3.3.2 ID3アルゴリズム

- 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

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



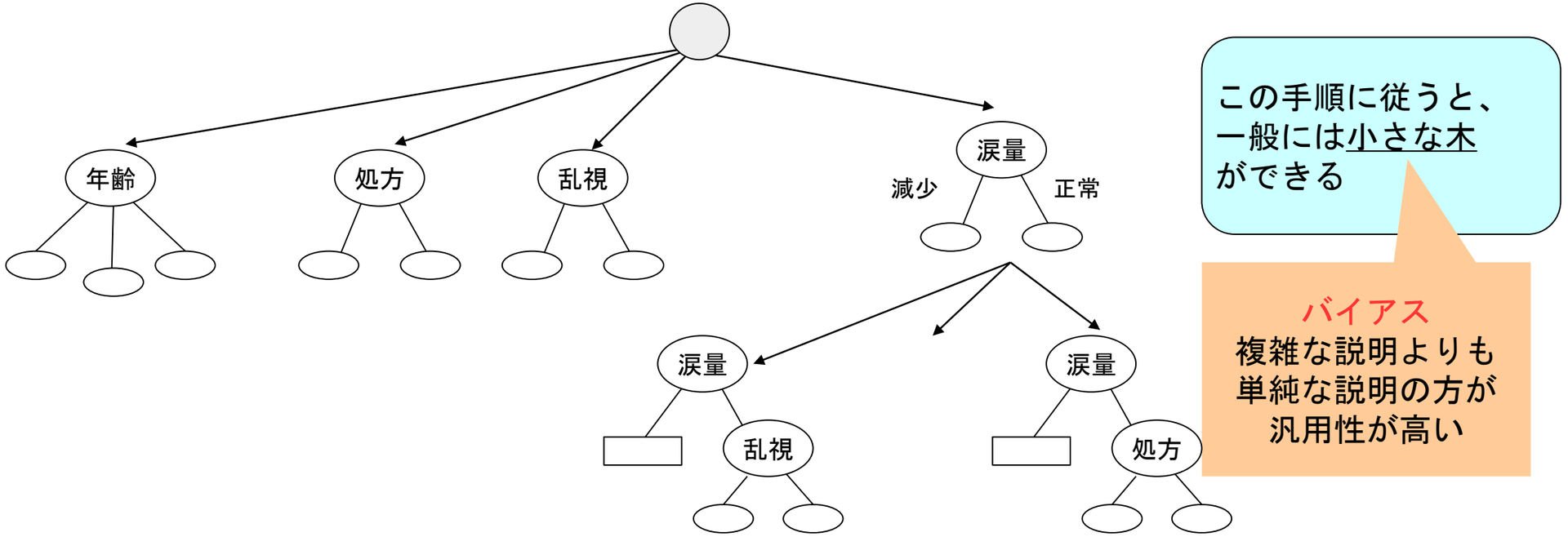
3.3.2 ID3アルゴリズム

- 情報獲得量
  - 特徴aを用いた分割後のエントロピーの減少量
  - 特徴aで値vを取るデータの集合: Dv
  - Dvの要素数: |Dv|
  - 情報獲得量の定義

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

3.3.2 ID3アルゴリズム

- ID3アルゴリズムのバイアス
  - 分類能力の高いノードをなるべく根の近くにもつ
  - 欲張り法で探索を行い，すべ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

3.3.2 ID3アルゴリズム

なぜ小さな木の方がよいか

- 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著しく低い

    → でも説明できた！

    → **必然!**

3.3.3 過学習を避ける

- 決定木学習における過学習の避け方
  - 学習過程で木の成長を止める
    - リーフに所属することができる最小データ数（または割合）を多くする
    - 木の段階の最大数を小さくする
  - 十分に成長させた後に枝刈り
    - Reduced error pruning
      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. 各枝について検証用データを用いて分類能力を測定し、多数決より性能が劣る枝を刈り取る
    - Minimal cost-complexity pruning
      - $R_\alpha(T)$ が最小となる木を探索

誤分類率　　リーフの数

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|$$

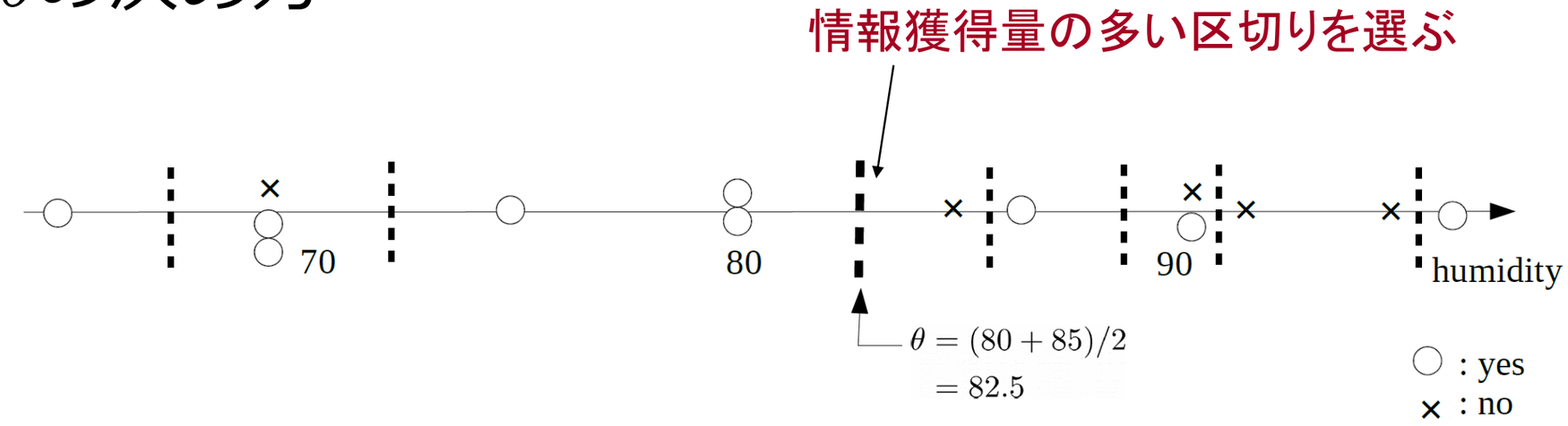
大きいほど小さい木になるパラメータ

3.4 数値特徴に対する決定木

- CART (Classification and Regression Trees)
  - 必ず2つの子ノードを持つ2分木構造
  - 数値特徴はノードとして何度でも出現可能
  - 識別と回帰のいずれにも使える
  - scikit-learnでは、DecisionTreeClassifier, DecisionTreeRegressor として実装されている

3.4 数値特徴に対する決定木

- ノードにおけるデータ分割
  - カテゴリ特徴：特徴の値で分割
  - 数値特徴：閾値との比較で分割
- 閾値θの決め方



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

3.4 数値特徴に対する決定木

- sklearn.tree.DecisionTreeClassifier の主なパラメータ
  - criterion：情報獲得量の計算法
    - ジニ不純度　$\mathrm{GiniImpurity}(D) \equiv 2 \cdot P_+ \cdot P_-$
    - エントロピー　$E(D) = -P_+ \log_2 P_+ - P_- \log_2 P_-$
    - ジニ不純度のほうが最頻クラスの分離に少し偏る
  - max_depth：木の最大の深さ
  - min_samples_split：ノードが分割可能な事例数の最小値
  - ccp_alpha: 枝刈り後の木の複雑さを表すパラメータ

まとめ

- 初期の概念学習
  - 論理式の形式を制限することでバイアスを実現
  - 制限が強すぎてしばしば概念獲得に失敗する
- 決定木
  - 論理式の形式は自由にして、探索でバイアスを実現
  - 学習データの少しの変動で、得られる木がまったく異なることがある
  - 原理的には学習データに対して誤りのない識別器を実現できるので、過学習への対処が必要になる