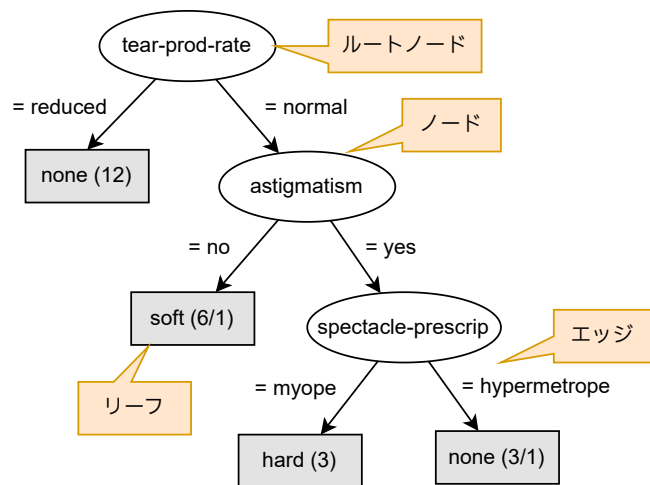


3. 識別 ー概念学習ー



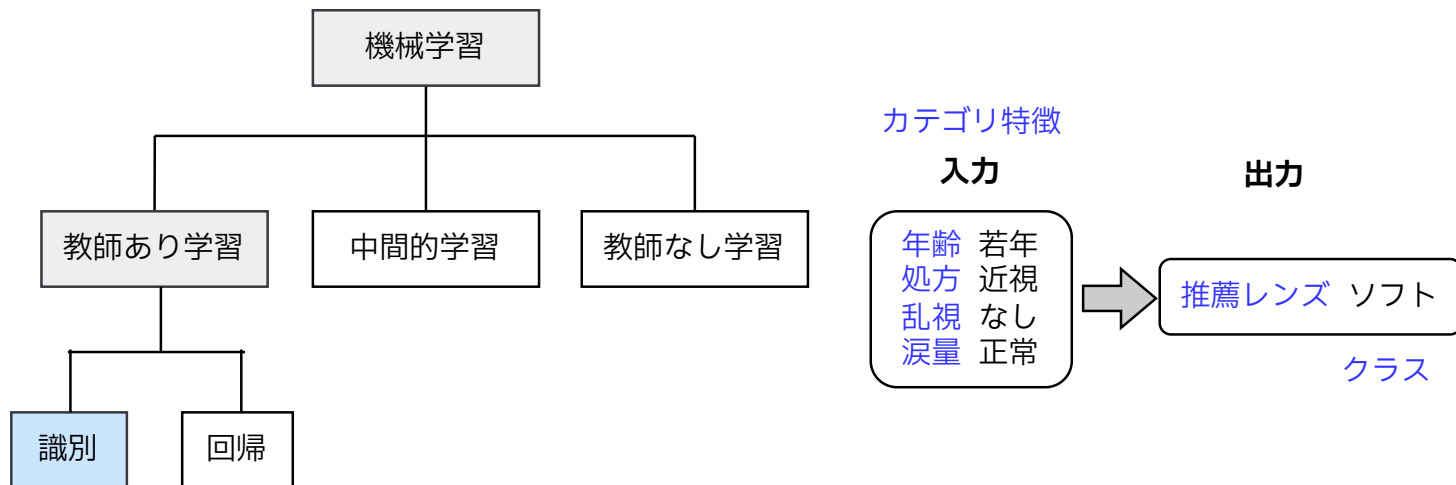
- 3.1 カテゴリ特徴に対する「教師あり・識別」問題の定義
- 3.2 概念学習とバイアス
- 3.3 決定木の学習
- 3.4 数値特徴のカテゴリ化



- 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

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: $\{(\mathbf{x}_i, y_i)\} \quad (i = 1, \dots, N)$
 - カテゴリ入力 \rightarrow 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

id	age	spectacle-prescrip	astigmatism	tear-prod-rate	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
	...				

- 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 (涙液分泌量) ('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 (1/6)

- 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） and （ドライアイ=なし） \Rightarrow 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）をかけて候補を絞り込む
 - 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

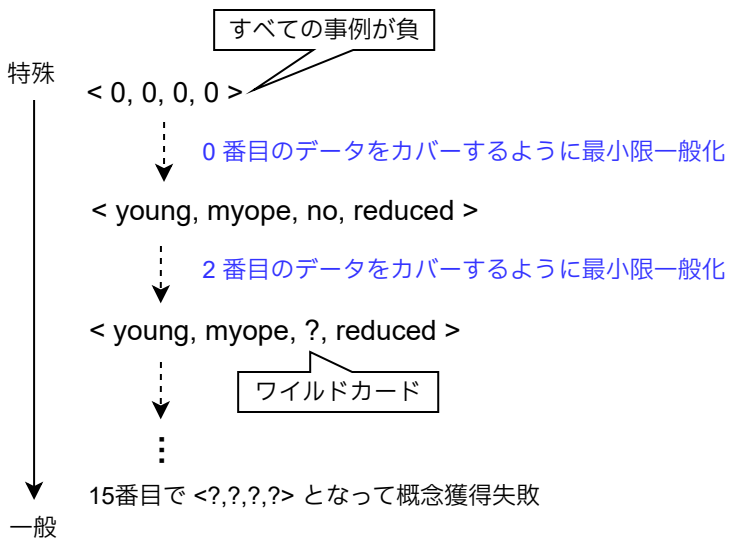
3.2 概念学習とバイアス (2/6)

- 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（ \wedge ）」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）： $4 \times 3 \times 3 \times 3 + 1 = 109$

3.2 概念学習とバイアス (3/6)

- FIND-Sアルゴリズム

- 例：「コンタクトレンズの使用を勧めない」(none) の概念獲得



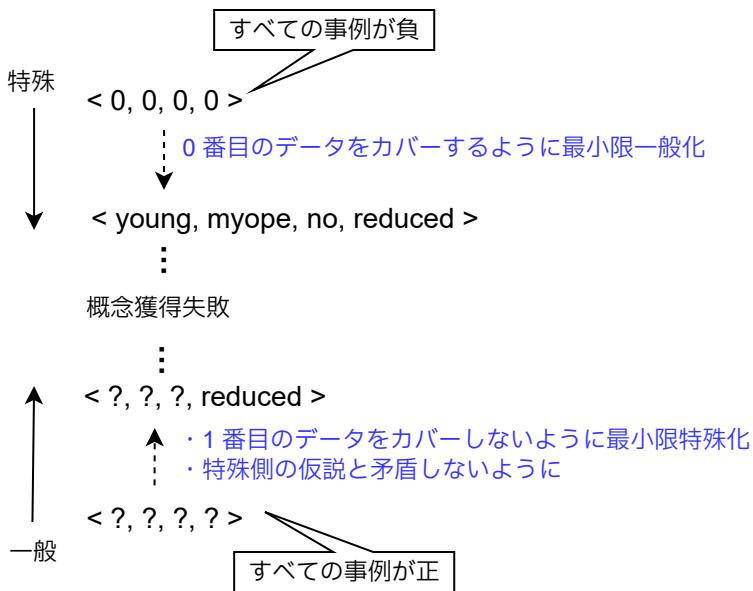
3.2 概念学習とバイアス (4/6)

- 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 $\langle ?, ?, ?, ? \rangle$ 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 概念学習とバイアス (5/6)

- 候補削除アルゴリズム

- 例：「コンタクトレンズの使用を勧めない」(none) の概念獲得



3.2 概念学習とバイアス (6/6)

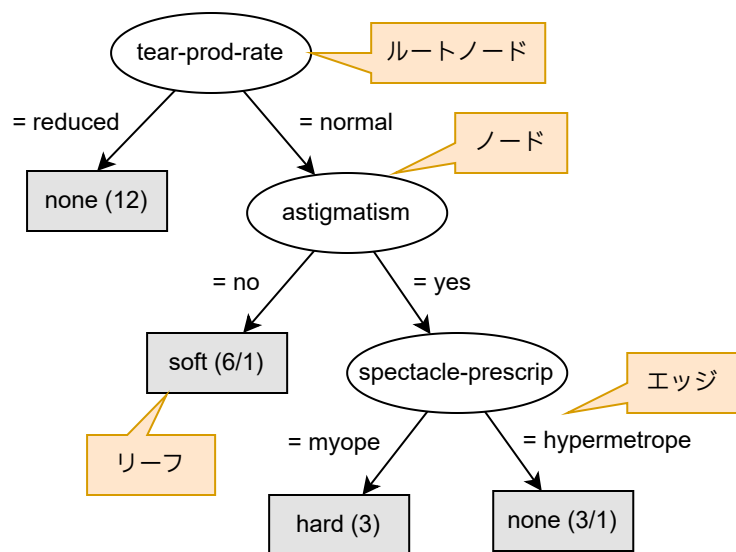
- 初期の概念学習の問題点
 - リテラルのOR結合が表現できないので，正解が仮説空間内に存在しないことが多い
 - 例) (年齢=若年 V 年齢=老眼前期) が表現できない
- 単純にOR結合を許す場合
 - 正解概念の候補数が増大する
 - 候補数: 2の「可能なすべての事例数」乗 $2^{24} = 16777216$
 - 正例のOR結合が自明な解となり，未知事例に対して判定する根拠を持たない
- 解決策
 - 仮説空間はOR結合を許し，探索方法にバイアスをかけて候補を限定
 - 新たな問題設定
 - 上記の方法で見つかった候補が未知事例に対しても有効な概念であるようなバイアスとは何か

3.3 決定木の学習 (1/8)

- 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
- 決定木の論理表現
 - エッジをたどることがAND結合
 - 同じ値のリーフが複数あることでORを表現

3.3 決定木の学習 (2/8)

- 決定木の例



3.3 決定木の学習 (3/8)

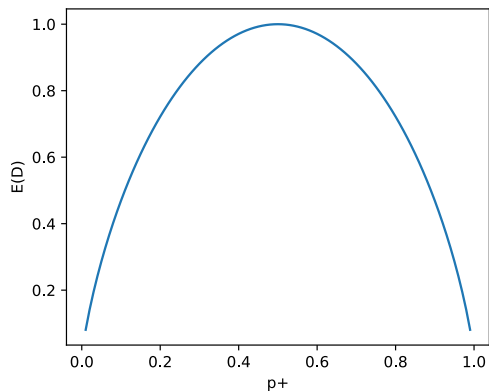
- 決定木の作り方
 1. ルートノードとする特徴を決める
 - 分割後のデータができるだけ同一クラスが偏るような特徴を選択する
 - 特徴の値に基づいてデータを分割する
 2. 分割後のデータが単一のクラスになればリーフとする
 3. そうでなければ，分割後のデータ集合に対して同様の操作を行う
 - ただし，これまでに使用した特徴は選択しない

3.3 決定木の学習 (4/8)

- 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

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

- 正例の割合: P_+ , 負例の割合: P_-



3.3 決定木の学習 (5/8)

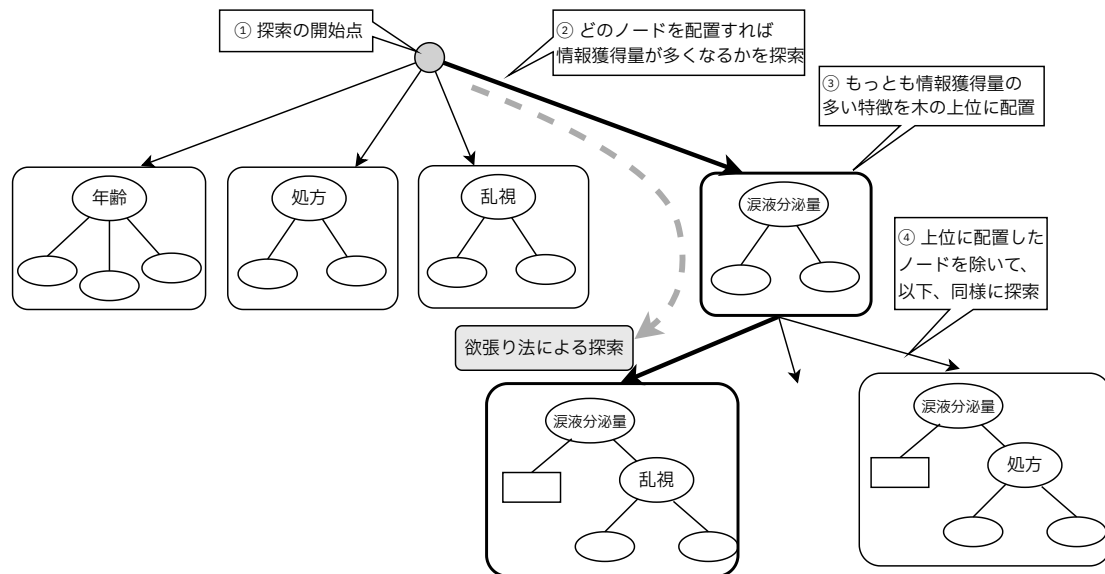
- 情報獲得量
 - 特徴 a を用いた分割後のエントロピーの減少量

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

- 特徴 a が値 v を取るデータの集合: D_v
- D_v の要素数: $|D_v|$

3.3 決定木の学習 (6/8)

- 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



3.3 決定木の学習 (7/8)

- 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
 - データを偶然に説明できる木が作れる確率も高い
 - 単純な仮説
 - 表現能力が低い
 - 偶然にデータを説明できる木が作れる確率は著しく低い
 - でも説明できた！
 - 必然!

3.3 決定木の学習 (8/8)

- 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
 - 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
 - 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
 - $R_\alpha(T)$ （誤分類率とリーフ数の重み付き和）が最小となる木を探索

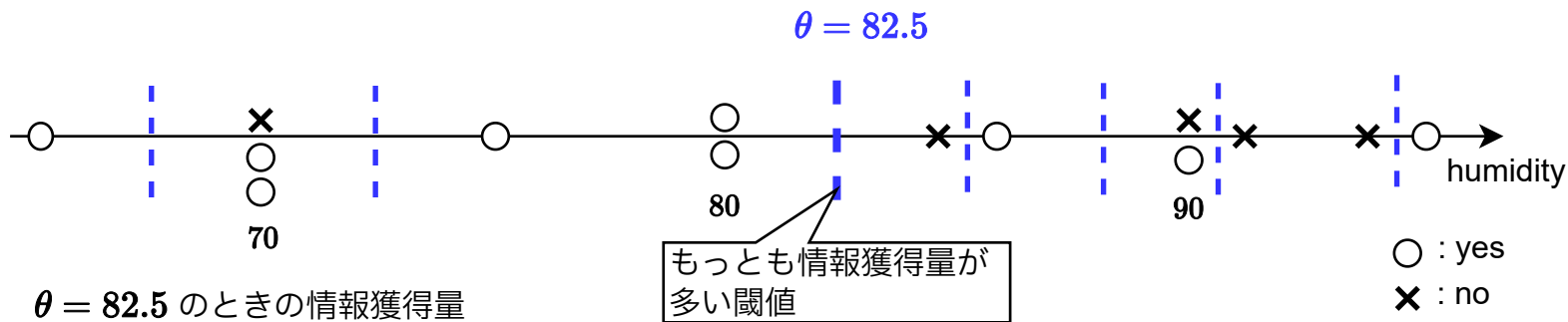
$$R_\alpha(T) = R(T) + \alpha|T|$$

3.4 数値特徴のカテゴリ化 (1/3)

- 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 数値特徴のカテゴリ化 (2/3)

- 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



3.4 数値特徴のカテゴリ化 (3/3)

- `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
 - エントロピー
 - 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