

情報意味論(4) 決定木と過学習

櫻井彰人
慶應義塾大学理工学部

本日の話題

- 過剰適合 (overfitting), 過学習 (overlearning)
- 連続値属性、多値属性
- その他

オッカムの剃刀: ある選好バイアス

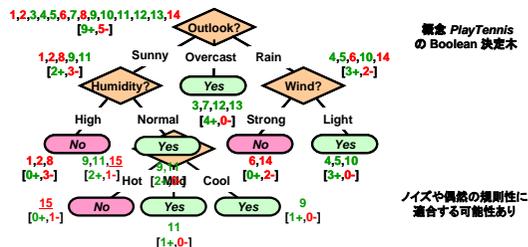
- 帰納バイアス2つ: 選好バイアス preference biases と言語バイアス language biases
 - 選好バイアス
 - ・ 学習アルゴリズムに(普通は暗黙的に)組み込まれている
 - ・ 言い換えれば: 探索順序の規定
 - 言語バイアス
 - ・ 知識 (仮説) の表現に(普通は暗黙的に)組み込まれている
 - ・ 言い換えれば: 探索空間の制限
 - ・ 別名 制限バイアス
- オッカムの剃刀 Occam's Razor: 賛成討論
 - 短い仮説の方が、長い仮説に比べ、個数が少ない
 - ・ 例えば、ビット列で考えれば、長さ n のものは $n + 1$ のものに比べ半数, $n \geq 0$.
 - ・ 短い仮説が、もしデータにぴったり合ったとしたら、偶然とは考え難い
 - ・ 短い仮説は、個数が少ないので、説明できる現象の数が少ない
 - ・ 長い仮説 (例: 200 個の節を持つ木, かつ $|D| = 100$) の場合には、偶然である可能性が高い
 - ・ いずれかの木がデータにぴったり合う。どれに合うかは偶然であるが、どれかに合うこと自体は当然。
 - 得るものと捨てたもの
 - ・ 他の条件が同一であれば、複雑なモデルの汎化能力は単純なモデルほどではない
 - ・ あとになってもっと柔軟な(微調整可能な)モデルが必要になることはない仮定

オッカムの剃刀と決定木: 二つの問題

- オッカムの剃刀 Occam's Razor: 反対討論
 - 仮説空間 H に依存して $size(h)$ が決まる。同じ h でも H が異なると $size(h)$ が異なる。
 - 「小ささ」を選好することへの疑問: 「少ない」ことは正当化にならない
- オッカムの剃刀 Occam's Razor は Well Defined か?
 - 内部の知識表現 knowledge representation によってどの h が「短い」かがきまる --- 恣意的?
 - 例えば、テスト "(Sunny \wedge Normal-Humidity) \vee Overcast \vee (Rain \wedge Light-Wind)" は一個?
 - 答: 表現言語を固定; 十分長いところでは、長い仮説は、内部表現によらず、やっぱり長い
 - 反論: 答えになっていない。実際には「短い仮説」に関する議論が重要
- 「短い仮説」であって、どうして他の「小さい仮説空間」ではないのか?
 - 小さい仮説集合を定義する方法はいろいろとある。
 - 選好バイアスで用いる $size$ が何であって、適当に基準 S を選べば $size(h)$ をその限界内に制限することができる (i.e., " S に合致する木のみ受理する")
 - ・ e.g., 節の個数が素数であって、文字 "Z" で始まる属性を用いている木
 - ・ なぜ「小さな木であって、(例えば) A_1, A_2, \dots, A_{17} を順番にテストするもの」ではないのか?
 - ・ $size(h)$ に基づいて小さな仮説集合を定義することに、特別の意味があるのか?
 - 参考: Chapter 6, Mitchell's Machine Learning

決定木における過学習: 例

- 既出例: 帰納した木



- 訓練事例にノイズがあると

- 事例 15: <Sunny, Hot, Normal, Strong, ->
 - ・ この例は noisy である。正しいラベルは +
 - ・ それ以前に作成した木は、これを、誤分類する
- 決定木はどのように更新すべきか (incremental learning)?
- 新しい仮説 $h' = T'$ の性能は $h = T$ より悪く なる と期待される

帰納学習における過学習

- 定義

- 仮説 h が訓練データ集合 D を過学習する (〜に overfits する) というのは、もし他の仮説 h' で $error_D(h) < error_D(h')$ であるが $error_{test}(h) > error_{test}(h')$ となるものがあること
- 原因: 訓練事例が少なすぎる (あまりにも少ないデータに基づく決断); ノイズ; 偶然の一致

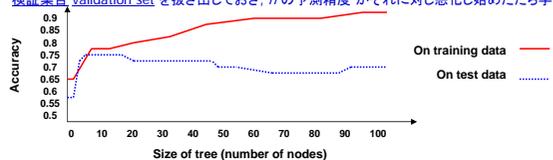
- 過学習に対応するには?

- コンピュータウイルスの感染やプロセスがデッドロックになることへの対策と類似
- 予防策
 - ・ 過学習が発生する前に対応する
 - ・ 関連する relevant 属性 (i.e., モデルにとって有用そうなもの) のみを用いる
 - ・ 注意: 鶏と卵の問題: 関連性 relevance を予測する尺度が必要
- 回避策
 - ・ 問題が起こりそうときに、脳をすりぬける
 - ・ テスト集合を確保しておく、仮説 h がその上で悪くなりそうときに、学習を停止する
- 泳がせ策
 - ・ 問題は発生するにまかせ、発生を検出し、その後回復する
 - ・ モデルを作ってみて、過学習に寄与する要素を発見・除去する (刈る prune)

決定木学習: 過学習の予防と回避

- 過学習にどう立ち向かうか?

- 予防策
 - ・ 関連する relevant 属性を選択 (i.e., 決定木では有用)
 - ・ 関連性 relevance の予測方法: 属性 filter または 部分集合選択 wrapper
- 回避策
 - ・ 検証集合 validation set を抜き出しておき、 h の予測精度 がそれに対し悪化し始めたから学習を停止



- “最良の”モデル (決定木) の選び方

- 上述: 性能を測定するにあたって、訓練データとそれとは別の検証データを用いる
- 別法: 最小記述長 Minimum Description Length (MDL):
 - 最小化せよ: $size(h = T) + size(\text{誤分類 misclassifications } (h = T))$

決定木学習: 過学習の予防と回避

- 基本的なアプローチが2つある

- Pre-pruning (回避): 木を作成する途中で木の生長を止める。信頼性ある選択をするに十分なデータはないと判断されたとき
- Post-pruning (回復): 木を一杯まで構築し節を削除する。削除するのは、十分な証拠がないとみなされるもの

- 枝刈りすべき部分木を評価する方法

- Cross-validation: T の有用性を評価するために、予めデータをとりおく (Mitchell 第4章)
- 統計的検定: 観測された規則性が偶然起こったものとして捨ててよいかどうかをテストする (Mitchell 第5章)
- 最小記述長 Minimum Description Length (MDL)
 - ・ 仮説 T の増加する複雑度は、単に(説明しようとしているデータの)例外を記憶するのに必要な複雑度の増加分より大きいかわ/小さいか?
 - ・ Tradeoff: モデルを記述する versus 残余誤差を記述する

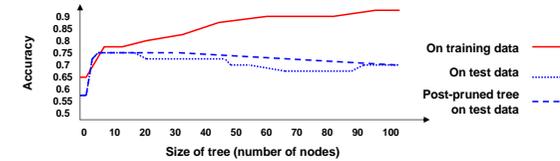
Reduced-Error Pruning

- Post-Pruning, Cross-Validation Approach
- 所与のデータを 訓練データ training set と 検証データ Validation set に分割する
- 関数 $Prune(T, node)$
 - 引数 $node$ を根節とする部分木を除去
 - 引数 $node$ を葉節とする (そこにある事例には多数派のラベルを付与)
- アルゴリズム *Reduced-Error-Pruning* (D)
 - D を分割する: D_{train} (訓練 training / "growing"), $D_{validation}$ (検証 validation / "pruning")
 - D_{train} に $ID3$ を適用して, 完全な木 T を作る
 - UNTIL $D_{validation}$ で計測した精度が減少する DO
 - FOR T 中のそれぞれの内節 $candidate$
 - $Temp[candidate] \leftarrow Prune(T, candidate)$
 - $Accuracy[candidate] \leftarrow Test(Temp[candidate], D_{validation})$
 - $T \leftarrow T \in Temp$ 中で $Accuracy$ が最良のもの
 - RETURN (pruneしおえた) T



Reduced-Error Pruning の効果

- Reduced-Error Pruning によるテスト誤差の減少



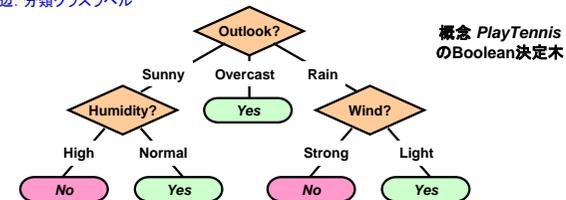
- 節を刈ることによってテスト誤差が減少する
- 注: $D_{validation}$ は D_{train} と D_{test} のどちらも異なる
- 賛成論 と 批判論
 - 賛成: 最も正確な T (T の部分木) のうちで最小のものが生成できる
 - 批判: T を作るのにわざわざデータ量を減らしている
 - ・ $D_{validation}$ をとっておくだけの余裕があるか?
 - ・ データ量が十分でなければ, 誤差をなおさら大きくする (D_{train} が不十分)

Rule Post-Pruning

- しばしば用いられる方法
 - これもよく知られたoverfitting 対応策
 - $C4.5$ でその亜種が用いられた. $C4.5$ は $ID3$ の派生・後継.
- アルゴリズム *Rule-Post-Pruning* (D)
 - D から T を生成 ($ID3$ を使用) – 可能な限り D に適合するまで成長させる (過学習も許す)
 - T を等価な規則集合に変換 (根節から葉節へ道一つにつき1規則)
 - それぞれの規則を, 独立に, 条件をどれでも, 推定精度が改善する限り, 除去することにより刈り込む (一般化する)
 - 刈り込んだ規則をソートする
 - ・ 推定精度に従ってソートする
 - ・ 列に並べて, D_{test} に適用する

決定木を規則に変換する

- 規則の構文
 - 左辺: 条件 (属性の等式テスト上の連言標準形 conjunctive formula)
 - 右辺: 分類クラスラベル



- 例
 - IF ($Outlook = Sunny$) \wedge ($Humidity = High$) THEN $PlayTennis = No$
 - IF ($Outlook = Sunny$) \wedge ($Humidity = Normal$) THEN $PlayTennis = Yes$
 - ...

連続値属性

連続値属性を扱う2つの方法

- 離散化
 - 実数値属性を、予め、いくつかの範囲に分ける
 - e.g., $\{high \equiv Temp > 35^\circ C, med \equiv 10^\circ C < Temp \leq 35^\circ C, low \equiv Temp \leq 10^\circ C\}$
- 内節を分けるのに、閾値を用いる
 - e.g., $A \leq a$ によって二つの部分集合 $A \leq a$ と $A > a$ ができる
 - この離散化に際して、情報増分が同様に計算される

情報増分を最大にする分割はどうやって得るか?

- FOR 連続値属性 A のそれぞれ
 - 事例 $\{x \in D\}$ を $x.A$ に従って、分割する
 - FOR 異なったラベルを持つ A の値の順序対 (l, u) それぞれ
 - 閾値の候補として、中点 mid-point の情報増分を評価, i.e., $D_{A \leq (l+u)/2}, D_{A > (l+u)/2}$
- 例
 - $A = Length$: 10 15 21 28 32 40 50
 - Class: - + + - + + -
 - 閾値のチェック: $Length \leq 12.5?$ $\leq 24.5?$ $\leq 30?$ $\leq 45?$

コスト付き属性

応用分野毎

- 医療: 体温検査のコストは 1000円; 血液検査 1500円; 生検 50000円
 - また検査の侵襲性・無侵襲性も考慮する必要あり
 - 患者へのリスクも (e.g., 羊水検査)
- 他のコスト
 - サンプリング時間: e.g., ロボットのソナー (レンジファインダー, etc.)
 - 人工物, 生体へのリスク (どんな情報を収集するか)
 - 関連する分野 (e.g., 断層装置): 非破壊検査

低い期待コストでいかに consistent な木を作るか?

- 一つのアプローチ: 情報増分 gain を コスト正規化増分 Cost-Normalized-Gain で置き換える
- 正規化関数の例
 - [Nunez, 1988]:

$$Cost - Normalized - Gain(D, A) = \frac{Gain^2(D, A)}{Cost(D, A)}$$
 - [Tan and Schlimmer, 1990]:

$$Cost - Normalized - Gain(D, A) = \frac{2^{Gain(D, A)} - 1}{(Cost(D, A) + 1)^w} \quad w \in [0, 1]$$
 但し w はコストの重要性を定める

多値属性に伴う問題

問題

- もしある属性が多値であると, $Gain(\bullet)$ はそれを選びやすい (なぜ?)
 - $Date$ (2004/11/01等) を属性として用いることを想像してみればわかる!
- 一つのアプローチ: $GainRatio$ を $Gain$ の代わりに使用

$$Gain(D, A) = -H(D) - \sum_{v \in values(A)} \frac{|D_v|}{|D|} \cdot H(D_v)$$

$$GainRatio(D, A) = \frac{Gain(D, A)}{SplitInformation(D, A)}$$

$$SplitInformation(D, A) = - \sum_{v \in values(A)} \frac{|D_v|}{|D|} \log \frac{|D_v|}{|D|}$$

- $SplitInformation$: $c = |values(A)|$ に直接に比例
- i.e., 多くの値をもつ属性にハンディを負わせる
 - e.g., 仮定: $c_1 = c_{Date} = n$ として $c_2 = 2$
 - $SplitInformation(A_1) = \log(n)$, $SplitInformation(A_2) = 1$
 - もし $Gain(D, A_1) = Gain(D, A_2)$ とすると, $GainRatio(D, A_1) << GainRatio(D, A_2)$
- こうして, (分岐数が少ない方への) 選択バイアスが $GainRatio(\bullet)$ を用いて表現される

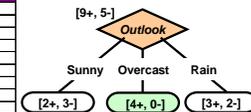
欠測値: 属性値が不明

問題: 属性 A の値がない事例があるとどうなるか?

- しばしば, 訓練時やテスト時に, 必ずしも全ての属性値が入手できるとは限らない
- 例: 医療診断
 - < Fever = true, Blood-Pressure = normal, ..., Blood-Test = ?, ... >
 - 時には値は全く未知であったり, また時には優先度が低かったりする (非常にコスト高のことも)
- 欠測値: 学習時 versus 分類時
 - 訓練時: ある $x \in D$ について A の値が与えられていないとき $Gain(D, A)$ を評価する
 - テスト時: A の値を知らずに, 新しい事例 x を分類する

解: $Gain(D, A)$ の計算の中に推測を入れる

Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	???	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No



欠測値: 対応策

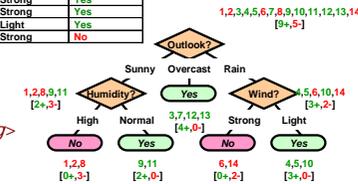
- 訓練事例はとにかく使用する, 木を(根節から)辿る・作ってくるとき
 - 考慮すべき属性のどれについても, 事例中でもし値が知られていないなら, それを推測する
 - その推測は, 今いる節に割当てられた事例の知られている値に基づく
- $x.A$ の最もありそうな値を推測する
 - 第一案: 節 n で属性 A をテストするなら, n を通る事例の A の値でもっとも多いものを用いる
 - 第二案 [Mingers, 1989]: 節 n で属性 A をテストするなら, n を通る事例で x と同じクラスラベルをもつものの A の値でもっとも多いものを用いる
- 推測値を分散させる
 - 両賭け: 値の分布に従い, 推測値を分散させる
 - $x.A$ の可能な値 v_i の分布に比例して確率 p_i を割当てる [Quinlan, 1993]
 - 木の子孫に, x の内の p_i 分を割当てる
 - これを用いて $Gain(D, A)$ or $Cost-Normalized-Gain(D, A)$ を計算する
- どのアプローチにおいても, 新事例も同様に分類する

欠測値: 例

- $x.A$ の最もありそうな値を予測する
 - 第一案: $Humidity = High$ or $Normal$ ($High: Gain = 0.97, Normal: < 0.97$)
 - 第二案: $Humidity = High$ (No 事例はすべて $High$)

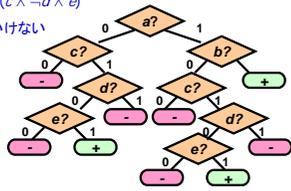
Day	Outlook	Temperature	Humidity	Wind	PlayTennis?
1	Sunny	Hot	High	Light	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Light	Yes
4	Rain	Mild	High	Light	Yes
5	Rain	Cool	Normal	Light	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	???	Light	No
9	Sunny	Cool	Normal	Light	Yes
10	Rain	Mild	Normal	Light	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Light	Yes
14	Rain	Mild	High	Strong	No

- 確率で重み付ける
 - Guess $0.5 High, 0.5 Normal$
 - $Gain < 0.97$
- テスト事例: $<?, Hot, Normal, Strong>$
 - $1/3 Yes + 1/3 Yes + 1/3 No = Yes$



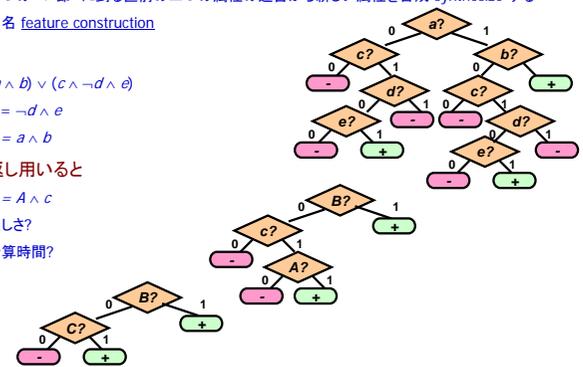
決定木における重複

- 決定木: 表現上の短所
 - 決定木は, 一番簡単な表現というわけではない
 - ポイント: 属性を重複 replication させる必要がある場合がある
- 属性重複の例
 - e.g., Disjunctive Normal Form (DNF): $(a \wedge b) \vee (c \wedge \neg d \wedge e)$
 - (どちらかの) 連言は部分木として重複させないといけない
- 部分解
 - 新しい属性を作る
 - 別名 constructive induction (CI)
 - Mitchell の第10章参照



少しだけ: 決定木の構成的帰納法

- 新しい属性の合成
 - 一つの "+ 節" に到る直前の二つの属性の連言から新しい属性を合成 synthesize する
 - 別名 feature construction
- 例
 - $(a \wedge b) \vee (c \wedge \neg d \wedge e)$
 - $A = \neg d \wedge e$
 - $B = a \wedge b$
- 繰り返し用いると
 - $C = A \wedge c$
 - 正しさ?
 - 計算時間?



他の問題、未解決問題

- 含めなかったこと
 - 目的は何であったか? 予測器の作成? 評価基準は?
 - 何時停止する? 良い汎化能力を保証するにはどうするか?
 - うまくできたのか?
 - ・ 正しさ
 - ・ 複雑さ
- 偏った決定木 Oblique Decision Tree
 - “軸平行” に限定しない決定木
- 漸進的な Incremental 決定木の帰納
 - 新しい事例が説明できるように、既存の決定木を、漸進的に incrementally 更新する
 - consistency の問題あり
 - 最小性の問題

決定木研究の歴史

- 1960 年代
 - 1966: Hunt (心理学) が人間の概念学習をモデル化するため決定木を使用
- 1970 年代
 - 1977: Breiman, Friedman, (統計学) Classification And Regression Trees (CART) を開発
 - 1979: Quinlan の最初の仕事 proto-ID3
- 1980 年代
 - 1984: CARTソフトの一般配布 (今では様々に)
 - 1986: Quinlan の ID3 に関する代表的論文
 - 様々な改良: ノイズ対応, 連続値属性, 欠測値, 偏った決定木, etc.
- 1990 年以降
 - 1993: Quinlan の新しいアルゴリズム C4.5
 - 枝刈り, 過学習を制御するヒューリスティック (C5.0, etc.); 複数決定木の使用, ...

術語

- オッカムの剃刀 Occam's Razor と決定木
 - 選択バイアス preference biases: 仮説空間の探索アルゴリズムに込められている
 - 言語バイアス language biases: 仮説記述言語 (仮説空間の定義) に込められている
- 過学習
 - 過学習 overfitting: k は h に比べ、訓練データでは良好であるが、テストデータでは不良
 - 予防 prevention, 回避 avoidance, 回復 recovery 技術
 - ・ 予防: 属性部分集合の選択 attribute subset selection
 - ・ 回避: 停止判定条件, cross-validation, pre-pruning
 - ・ 発見と回復: post-pruning (reduced-error, rule)
- 決定木をよりロバストにする他の方法
 - 不等式でテスト: 連続値属性を取扱う方法
 - 情報増分比: 多値属性への偏向を防ぐための正規化の一つ
 - Cost-normalized gain: コストや有用性を加味する方法
 - 欠測値 missing data: 未知の属性値
 - 属性構築 feature construction: 構成的帰納 constructive induction の一つ; 新属性の生成
 - 重複 replication: 部分木を繰返す

まとめ

- オッカムの剃刀 Occam's Razor と決定木
 - 選択バイアス preference biases versus 言語バイアス language biases
 - オッカム アルゴリズムに関する2つの問題
 - ・ 何故小さい木を選ぶのか? (偶然の一致が少ない)
 - ・ オッカムの剃刀は well-defined か? (yes, 適当な条件のもと)
 - MDL 原理とオッカムの剃刀: 後ほど
- 過学習
 - 問題: 訓練データに近づきすぎ
 - ・ 過学習の形式的定義
 - ・ なぜ起こるのか
 - 過学習の 予防 prevention, 回避 avoidance, 回復 recovery 技術
- 決定木帰納をロバストに行う方法