知的情報処理
6. 簡単便利な決定木: 作るのは少々難しい

櫻井彰人
慶應義塾大学理工学部

決定木

- 決定木は、前スライド「その他」の典型例
  - 境界は、綺麗な関数ではかけない
  - 最適化すべきものが、式で書かれていない
  - アルゴリズムは、OR的なものとはかけ離れている
- けれども、ユーザに非常に分かりやすく、誤差もそこそこに小さいため、重要なツールである
- 単純かつ有効なだけに、様々な工夫がされてきている。
  - あなどってはいけない。
- 出来上がった決定木は理解しやすい（すぐ分かる。だから使われる）。しかし、作るのは、結構、難しい。

applet によるデモ

- applet によるデモがあります

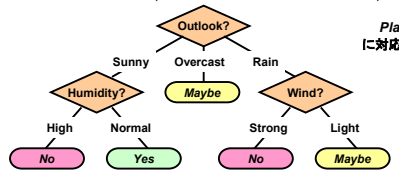
  http://www.cs.ualberta.ca/~aixplore/learning/DecisionTrees/Applet/DecisionTreeApplet.html

  http://www.cs.ubc.ca/labs/lci/CIspace/Version4/dTree/

- 後者の簡単な説明を最後に付加しておきます

決定木 Decision Trees

- 分類器 Classifiers である
  - 事例 (ラベルのついていないもの): 属性 attribute (または特徴 feature) のベクトル
- 内節 Internal Nodes: 属性値のテストを行う
  - 典型的: 等しいかどうかのテスト (e.g., "Wind = ?")
  - その他 不等式や様々なテストが可能
- 枝 Branches: 枝を選ぶ条件である属性値 (テストが等式以外のときはテストの結果)
  - 一対一対応 (e.g., "Wind = Strong", "Wind = Light")
- 葉 Leaves: 割当てた分類結果 (分類クラスのラベル Class Labels)



どんな時、決定木を用いるか

- 事例が属性 - 属性値ペアで表現される
- 目標関数が離散値をとる（分類問題）
- 選言を含む仮説が必要
  - 属性値に関する連言であれば、「概念学習」で可能
- ノイズが入っている可能性がある
- 例（実際に Mitchell が適用した）
  - 機器故障診断、病名診断
  - 与信リスクの分析
    - クレジットカード, ローン
    - 保険
    - 消費者による不正行為
    - 従業員の不正行為

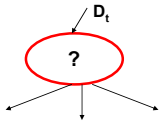
決定木を作る（帰納する）

- 多くのアルゴリズムがある:
  - Hunt のアルゴリズム（古いものの一つ）
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

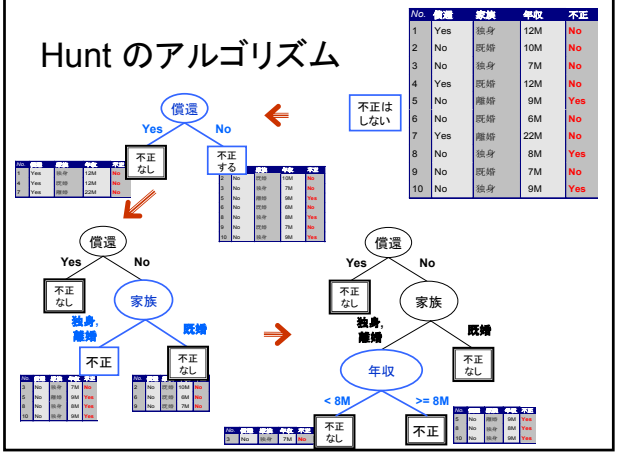
Hunt のアルゴリズムの構造

- D_t をノード t にたどりついた訓練データの集合とする
- 手続きの概要:
  - もし D_t に含まれるデータがすべて同じクラス y_t に属するなら, t は葉ノードであってそのラベルは y_t .
  - もし D_t が空集合なら、t は葉ノードであって、そのラベルは予め決めておくデフォールトラベル y_d となる
  - もし D_t に含まれるデータは複数個のクラスに属するとき, t には属性値のチェックを入れ, 訓練データ D_t をより小さな集合(部分集合)に分割する. この手続きを再帰的に、当該部分集合に適用する.

No.	償還	家族	年収	不正
1	Yes	独身	12M	No
2	No	既婚	10M	No
3	No	独身	7M	No
4	Yes	既婚	12M	No
5	No	離婚	9M	Yes
6	No	既婚	6M	No
7	Yes	離婚	22M	No
8	No	独身	8M	Yes
9	No	既婚	7M	No
10	No	独身	9M	Yes



Hunt のアルゴリズム



決定木の作成(帰納)

- グリーディな方略.
  - いったん決めたら、心変わりしない
    - 迷路を進むときに、後戻りしない。一度掴んだら離さない。
    - 最適ではないが、後戻りしない分、速い。
  - 訓練データを、ある評価基準を最適化するように、ある属性で分割する.
    - 一度分割してある枝を作ったら、それを取りやめることはない
- 課題
  - 訓練データの分割方法を決定する
    - 属性テスト方法をどう定めるか?
    - 最良の分割をどう決めるか?
  - (分割の)止め時の決め方

決定木の作成(帰納)

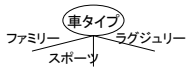
- グリーディな方略.
  - いったん決めたら、心変わりしない
    - 迷路を進むときに、後戻りしない。一度掴んだら離さない。
    - 最適ではないが、後戻りしない分、速い。
  - 訓練データを、ある評価基準を最適化するように、ある属性で分割する.
    - 一度分割してある枝を作ったら、それを取りやめることはない
- 課題
  - 訓練データの分割方法を決定する
    - 属性テスト方法をどう定めるか?
    - 最良の分割をどう決めるか?
  - (分割の)止め時の決め方

属性テスト条件の決め方

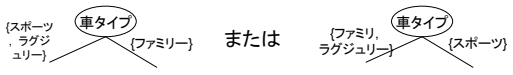
- 属性タイプによって異なる
  - 名義変数
  - 順序変数
  - 数値変数
- いくつに分割するかによって異なる
  - 2分割
  - 多分割

名義変数による分割

- 多分割: 当該変数の変数値の「異なり数分」、分割する.



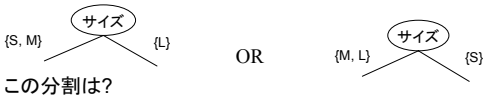
- 2分割: 変数値を2個に分割する.
  最適な分割を求める必要あり.



順序変数に基づく分割

- 多分割: 異なる値の個数分、分割.

- 2分割: 2つの部分集合に分割.
  最適分割を見つける必要あり.



- この分割は?

属性値の固有な順序を無視している.

数値変数に基づく分割

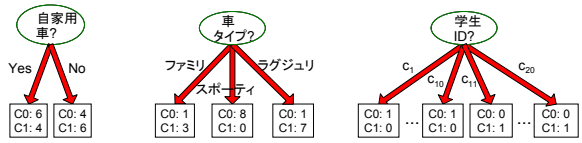
- 身長、体重、血圧、コレステロール値、、、、
  - 1単位ずつ分けると分けすぎ。
  - 離散化: いくつかの境(閾値ともいう)を設けて、いくつかに分ける。
- いくつかの方法がある
  - 離散化 して順序属性として扱う
    - 静的 – 最初に一回だけ離散化
    - 動的 – 等幅区間、等頻度区間(パーセンタイル)、クラスタリング
  - 2値判別: (A < v) または (A ≥ v)
    - すべての可能な分割を考え、ベストなものを見出す
    - 計算が一層必要となることも

決定木の作成(帰納)

- グリーディな方略.
  - いったん決めたら、心変わりしない
    - 迷路を進むときに、後戻りしない。一度掴んだら離さない。
    - 最適ではないが、後戻りしない分、速い。
  - 訓練データを、ある評価基準を最適化するように、ある属性で分割する.
    - 一度分割してある枝を作ったら、それを取りやめることはない
- 課題
  - 訓練データの分割方法を決定する
    - 属性テスト方法をどう定めるか?
    - 最良の分割をどう決めるか?
  - (分割の)止め時の決め方

最良な分割はどうやって見つける?

分割前: C0 (クラス0)に 10 データ,
C1 (クラス1)に 10 データ



どの条件が最適か?

最良な分割はどうやって見つける?

- グリーディ方略:
  - 新ノード内のクラス分布が 同質となる 分割がベター
  - どこかのクラスが圧倒的な多数となる(これが同質)ということは、それだ!といっても間違いが少ない
- (ノードの)同質さの物差しが必要:

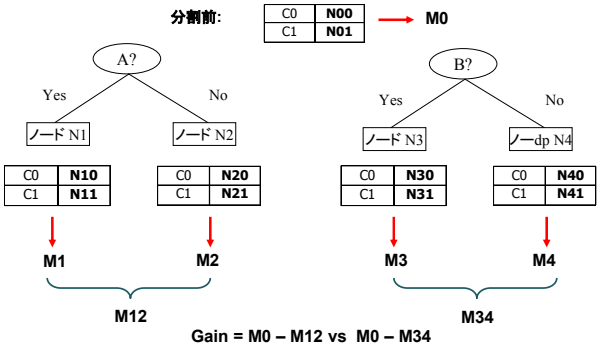
非-同質, 純度が低い
不純度が高い

同質, 純度が高い
不純度が低い

不純度のものさし

- エントロピー
- ジニ・インデックス Gini Index
- 誤分類率

最良な分割はどうやって見つける？



復習: エントロピー

ビット

- 確率変数 X の独立サンプルを観測しているとする
- X は4個の値をとる

P(X=A) = 1/4	P(X=B) = 1/4	P(X=C) = 1/4	P(X=D) = 1/4

- 従って、例えば、: BAACBADCDADDDA…
- シリアルリンク（1本の信号線）で2進符号を送る場合、個々の観測を2ビットに符号化できる (e.g. A=00, B=01, C=10, D = 11)

0100001001001110110011111100…

より少ないビット数で

- もし、誰かが、実は、等確率ではないのだヨと教えてくれたら

P(X=A) = 1/2	P(X=B) = 1/4	P(X=C) = 1/8	P(X=D) = 1/8

- 可能なのは…

… 平均では、1観測あたり 1.75ビットとなるような符号の作り方.

A	0
B	10
C	110
D	111

一般論

- 確率変数 X は *m* 個の値をとるとする…

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	….	$P(X=V_m) = p_m$

- X の独立試行から得られる値（の観測）の列を送信する場合、1観測（記号）あたりのビット数（の期待値）を最小化する場合、その値は何か？それは、実は

$$entropy(p_1,\ldots,p_m) = -p_1 \log_2 p_1 - \ldots p_m \log_2 p_m$$

- H(X) : X のエントロピー
- これを見出したのは Shannon であるが、彼は、いくつかの仮定をおいて、この性質を導いている.

決定木へ戻る

決定木の構築

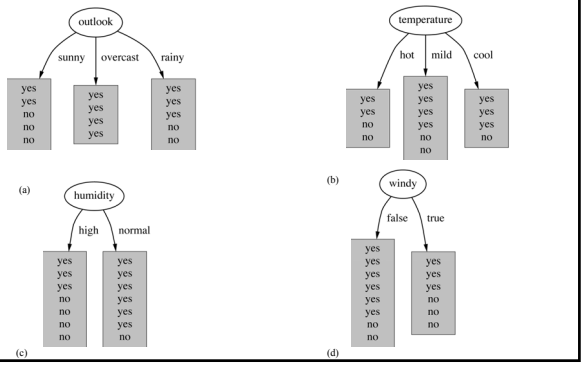
- 通常の手順: 上から下に(根から葉へ)、再帰的かつ分割統治(divide-and-conquer)
  - まずは: 一つの属性を選び根とする。属性値ごとに枝を作る
  - 次は: 訓練データを部分集合に分割 (枝一本につき一個)
  - 最後に: 同じ手順を、個々の枝について行う。その場合、個々の枝に割り当てられた訓練データのみを用いる(全体は用いない)
- ノードに(それへの枝に)割り当てられた訓練データがすべて同じクラスになったら、終了

テニスをするや否や

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Tom Mitchell "Machine Learning" の例題. よく使われる

どの属性がいいのか?



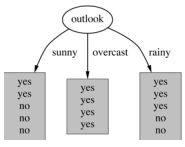
属性選択の基準

- どの属性が最適化?
- できあがる決定木が最小のものがよい
  - ヒューリスティック: "純度" 最高の属性を選ぶ
  - 「最小」のものを選ぶことに関し、深遠な議論がある
- 良く使われる "不純度" の基準: (ノードの)エントロピー
  - エントロピーが低いほど、ノードの "純度" は高い.
- 方略: 子供のノードのエントロピーが最小となる属性を選べ.

計算例: 属性 "Outlook"

"Outlook" = "Sunny":
  info([2,3]) = entropy(2/5,3/5) = – (2/5)log(2/5) – (3/5)log(3/5) = 0.971
"Outlook" = "Overcast":
  info([4,0]) = entropy(1,0) = – 1 log(1) – 0 log(0) = 0
"Outlook" = "Rainy":
  info([3,2]) = entropy(3/5/,2/5) = – (3/5)log(3/5) – (2/5)log(2/5) = 0.971

この属性を用いたときの情報量は
  info([3,2],[4,0],[3,2]) = (5/14)×0.971 + (4/14) ×0 + '5/14) ×0.971
  = 0.693 bits



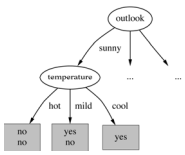
情報量増分 Information gain

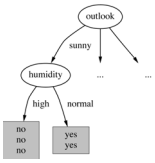
- ただし、通常は、ノードのエントロピーを直接用いることはない。情報量増分を用いる.

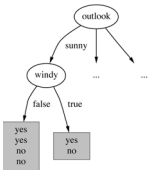
  情報量増分: 分割前の情報量 – 分割後の情報量
  gain("Outlook") = info([9,5]) – info([2,3],[4,0],[3,2]) = 0.940 – 0.693
  = 0.247 bits
  同様に計算すると
  gain("Outlook") = 0.247
  gain("Temperature") = 0.029
  gain("Humidity") = 0.152
  gain("Windy") = 0.048

- 情報量増分が多いほど、純度が高い。従って、"Outlook" を選ぶことにする。

分割を続ける

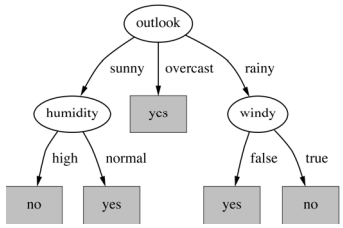






$$gain(\text{"Temperature"}) = 0.571\,bits$$
$$gain(\text{"Humidity"}) = 0.971\,bits$$
$$gain(\text{"Windy"}) = 0.020\,bits$$

最終的に得られる決定木



- 注: すべての葉が “純” である必要はない; というのも、同じデータなのにクラスが違うことがあるから（ノイズのせい）

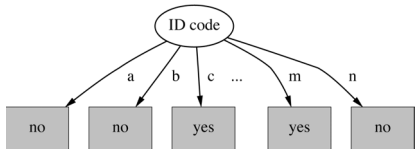
⇒ データがそれ以上分割しない方がよくなったら、やめ

枝数が非常に多くなる属性があると、、、

- IDコードをつけてみよう

ID code	Outlook	Temp.	Humidity	Windy	Play
A	Sunny	Hot	High	False	No
B	Sunny	Hot	High	True	No
C	Overcast	Hot	High	False	Yes
D	Rainy	Mild	High	False	Yes
E	Rainy	Cool	Normal	False	Yes
F	Rainy	Cool	Normal	True	No
G	Overcast	Cool	Normal	True	Yes
H	Sunny	Mild	High	False	No
I	Sunny	Cool	Normal	False	Yes
J	Rainy	Mild	Normal	False	Yes
K	Sunny	Mild	Normal	True	Yes
L	Overcast	Mild	High	True	Yes
M	Overcast	Hot	Normal	False	Yes
N	Rainy	Mild	High	True	No

IDコードを根にもってくると、“切株”



この分割のエントロピー
info(“IDcode”) = info([0,1]) + info([0,1]) + … + info([0,1]) = 0 bits
⇒ 情報量増分は最大となる（すなわち、0.940 bits ）

枝分かれの多い属性

従って,

- 属性値が多いと、訓練データの部分集合は “純” になりやすい
  - 情報量増分は、属性値の多い属性の方にバイアスしている
  - この結果、過学習 overfitting (過去のデータの学習という意味では素晴らしいが、予測のためには最適でない属性を選んでしまう)になってしまう。

増分比

- 増分比 Gain ratio: 情報量増分のもつバイアスを減少させる
- 増分比は、枝の本数とそれに割り当てられる訓練データの大きさの両方を勘定に入れる
  - 情報量増分の修正は、訓練データの集合をどのような（大きさと要素数の）部分集合に分割するかという分割の情報量を用いて、行われる

増分比の計算例

計算例: IDコードの分割情報量（split information）
info([1,1,…,1]) = 14 × (- (1/14) log(1/14)) = 3.807 bits

増分比の定義
gain_ratio(“Attribute”) = gain(“Attribute”) / split_info(“Attribute”)
計算例:
gain_ratio(“IDcode”) = 0.940 bits / 3.807 bits = 0.246

他の属性に関する増分比

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.362
Gain ratio: 0.247/1.577	0.156	Gain ratio: 0.029/1.362	0.021
Humidity		Windy	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

増分比について

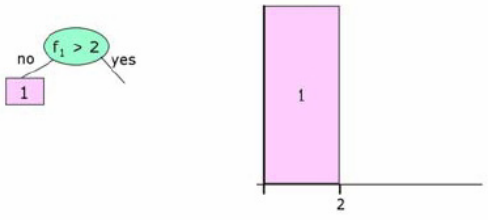
- “Outlook” がトップであるが、今度は “Humidity” が肉薄している。というのも、“Humidity” は2個に分割するため、増分比が相対的に良くなるためである.
- 見ればわかるように: “ID code” の増分比が最大！. もっともそのアドバンテージは大分と減少したが.
- 増分比の問題点: 過補償となるおそれがあること
  - 分割情報量が小さいために、不適当な属性が選ばれる可能性
  - よくある修理方法: 増分比が最大のものを選ぶのだが、当該属性の情報量増分は、少なくとも、情報量増分の平均値（全属性で考えて）はあるものという条件を課す.

補足

- 決定木のトップダウン（根から葉へ）アルゴリズム(“ID3”)は、Ross Quinlan (University of Sydney Australia) が開発
- 増分比は、このアルゴリズムの基本的な改良の一つ
  - これに引き続き開発されたのが C4.5。数値属性、欠測値、ノイズのあるデータが扱える
- 属性選択には他の方法がたくさんある! (といっても、結果の精度にはあまり違いがない)

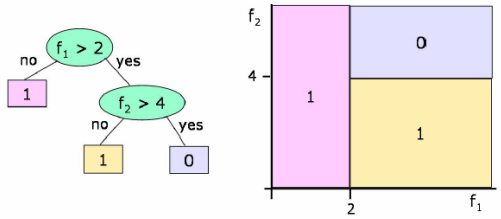
数値属性

- 属性テストは次の形をとる x_j > ある定数
- 属性値のなす空間を短冊に分割する



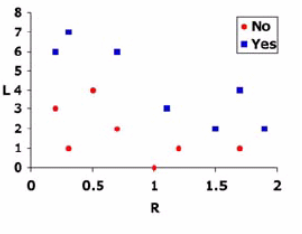
数値属性

- 勿論、これでもいい x_j > ある定数
- 短冊への分割は同じ



破産の予測

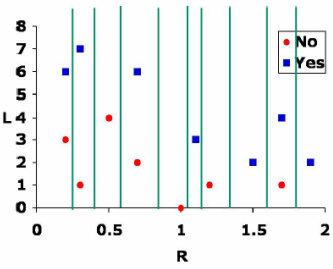
L	R	B
3	0.2	No
1	0.3	No
4	0.5	No
2	0.7	No
0	1.0	No
1	1.2	No
1	1.7	No
6	0.2	Yes
7	0.3	Yes
6	0.7	Yes
3	1.1	Yes
2	1.5	Yes
4	1.7	Yes
2	1.9	Yes



L: 一年あたりの支払い遅延回数
R: 支出/ 収入
B: 破産

分割を考えよう

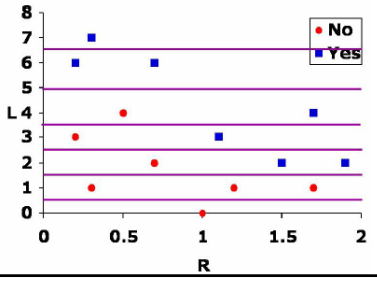
- 各属性ごとに、分割することを考えよう



- 今回の例では、R軸 に沿っての分割の仕方は、高々 9 方法ある
  - 一般に, 訓練データが m 個あれば、m – 1 方法ありそう
  - しかし今回の場合は、R 軸の値が同じデータがあるので、その分、減った.

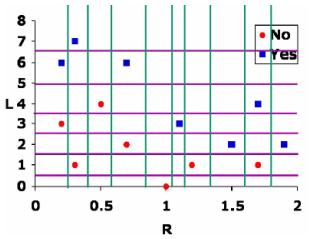
分割その II

- L 軸では高々 6 方法ある
  - L 軸は整数値をとるので、値が重複するデータは多い.



分割によるエントロピーを計算

境界	下方にあるNoの個数	下方にあるYesの個数	上方にあるNoの個数	上方にあるYesの個数	エントロピー
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93

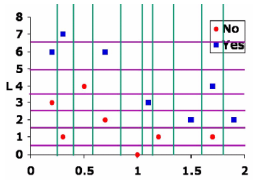


エントロピー	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
境界	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

承前

- それぞれの軸でのすべての可能性を考え、分割した場合のエントロピーを計算した

境界	下方にあるNoの個数	下方にあるYesの個数	上方にあるNoの個数	上方にあるYesの個数	エントロピー
6.5	7	6	0	1	0.93
5.0	7	4	0	3	0.74
3.5	6	3	1	4	0.85
2.5	5	2	2	5	0.86
1.5	4	0	3	7	0.63
0.5	1	0	6	7	0.93



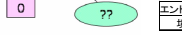
エントロピー	1.00	1.00	0.98	0.98	0.94	0.98	0.92	0.98	0.92
境界	0.25	0.40	0.60	0.85	1.05	1.15	1.35	1.60	1.80

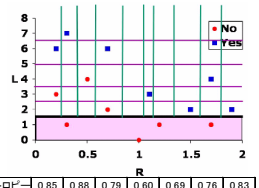
- たまたま、L 軸で、境界を 1.5 とした場合、片側が No だけになることがわかった(エントロピーも最小)

承前

- 残りの空間のすべての分割を考える.
- エントロピーは再計算が必要. すでに葉に割り当てられた訓練データは取り除いて考えなければならないから.

境界	下方にあるNoの個数	下方にあるYesの個数	上方にあるNoの個数	上方にあるYesの個数	エントロピー
6.5	3	6	0	1	0.93
5.0	3	4	0	3	0.74
3.5	2	3	1	4	0.85
2.5	1	2	2	5	0.86



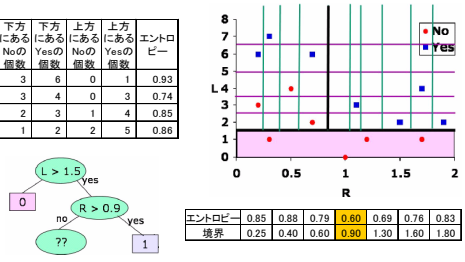


エントロピー	0.85	0.88	0.79	0.60	0.69	0.76	0.83
境界	0.25	0.40	0.60	0.90	1.30	1.60	1.80

承前

- 今度の最適な分割は R > 0.9 である. しかも、すべて No であるので、葉を作ることができる.

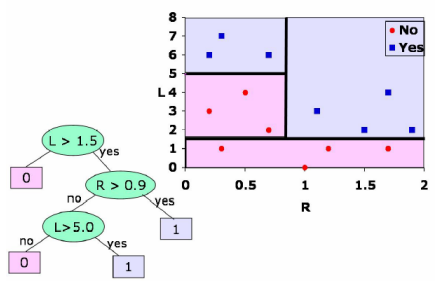
境界	下方にあるNoの個数	下方にあるYesの個数	上方にあるNoの個数	上方にあるYesの個数	エントロピー
6.5	3	6	0	1	0.93
5.0	3	4	0	3	0.74
3.5	2	3	1	4	0.85
2.5	1	2	2	5	0.86



エントロピー	0.85	0.88	0.79	0.60	0.69	0.76	0.83
境界	0.25	0.40	0.60	0.90	1.30	1.60	1.80

承前

- これを続ければ次のものが得られる:



補足 GINI と回帰木

GINI に基づく分割基準

- これまで説明してきた分割基準はエントロピーであった:

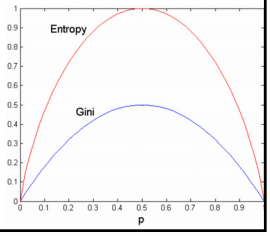
$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(注: $p(j \mid t)$ はノード t におけるクラス j データの相対頻度).

- 別法に GINI インデックスを用いるものがある:

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

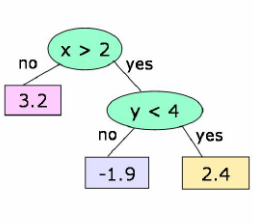
- 両者とも:
  - 最大値 ($\log n_c$ または $1 - 1/n_c$) が得られるのは, 当該データがどのクラスにも等分に分配されているときである. 「等分である」ということは何の面白さもない. しかし、この状態で、「実はこれ！」と教わることは非常に価値のあることである。
  - 最小値 (0.0) が得られるのは, すべてのデータが同一のクラスに属するとき. 非常に面白い。けれども、(もう一つのクラスに属することを知っているので)「実はこれ！」という情報には何の価値もない。

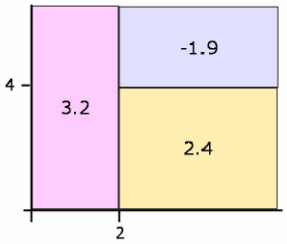


2個のクラスに分ける場合:

回帰木 Regression Trees

- 決定木と同じ, 但し 葉において、実数値定数を出力する.





葉における値

- 今いる葉ノードには複数個のデータがあると仮定しよう。なおかつ、何らかの理由により、このノードはこれ以上分割しないものとする.
- 離散値の場合（これまでの場合）, 葉における値（出力値）は、その葉における多数派の値としていた.
- 数値属性の場合, 妥当な値は平均値であろう.
- 従って, もし葉ノードにおける出力値として平均値を用いるならば, (これからノードを分割して子供が葉ノードになろうというときには）枝分かれして作られる新たな葉ノードにおいて、データのもつ値が、当該葉ノード内の値の平均値よりあんまり離れていない方がよかろう.
- 統計学には、数値の集合がどのくらい分散しているかを表す尺度がある
  - (言い換えれば, 個々の数値が平均値からどれだけ離れているか);
  - ご存じの分散である.

平均: 念のため（分散の説明のため）

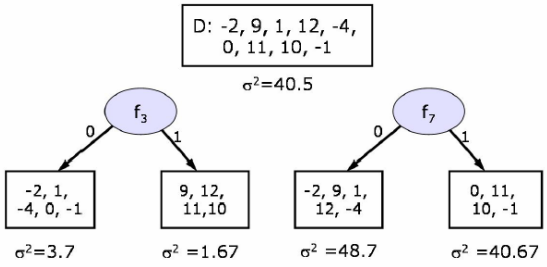
- 観測した数値データのばらつきを均した値。たくさんの意味がある
- 今回、興味があるのは、母平均と標本平均.
  - 母平均は、確率変数の1次モーメント。すなわち、
$$\mu = \int_{-\infty}^{\infty} z\, f(z) dz$$
  - z_1 から z_m の標本平均値:
$$\hat{\mu} = \frac{1}{m}\sum_{k=1}^{m} z_k$$
  - 標本平均は母平均の不偏推定量である。すなわち、観測を無限に繰り返せば、標本平均は母平均に収束する。
- 母平均は存在するとは限らない

分散: 念のため

- 観測数値データがどれだけ散らばっているかの指標・尺度.
- 母分散、標本分散、不偏分散が代表的。データは z_1 から z_m とする:
  - 母分散: 確率変数の2次の中心化モーメント
$$\sigma^2 = \int_{-\infty}^{\infty} (z-\mu)^2 f(z) dz \qquad \mu = \int_{-\infty}^{\infty} z\, f(z) dz$$
  - 標本分散
$$s^2 = \frac{1}{m}\sum_{k=1}^{m} (z_k - \hat{\mu})^2 \qquad \hat{\mu} = \frac{1}{m}\sum_{k=1}^{m} z_k$$
  - 不偏分散
$$\hat{\sigma}^2 = \frac{1}{m-1}\sum_{k=1}^{m} (z_k - \hat{\mu})^2 \qquad \hat{\mu} = \frac{1}{m}\sum_{k=1}^{m} z_k$$
- 注意
  - 母分散は存在するとは限らない
  - 不偏分散は、母分散の不偏推定量である。すなわち、何度も測定を（今の場合は、測定して計算を）繰り返したときに、当該値が真の分散値に収束するようなものである。
  - 標本分散は、不偏推定量ではない。
  - 標準偏差は、分散の平方根。標準偏差は、偏差を計測・比較するときの標準値（多分）

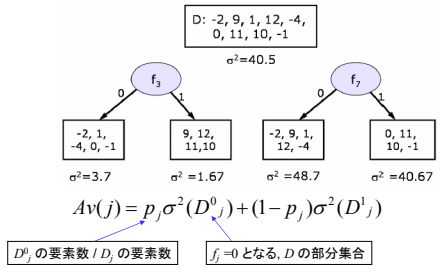
ノード分割（データの分割）

- 分割のよさを計る尺度として分散の平均値を用いることにする.
- （出力たるべき）y 値のみを記す.



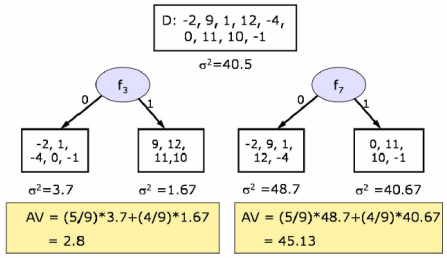
ノード分割

- 離散値の場合と同様に、分散の平均値を計算するとき、各新ノードに渡されるデータ数で重み付けた、重み付き平均値を用いることにする（そうでなければならない）.



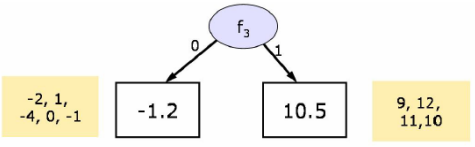
ノード分割

- 簡単に計算できることだが、属性3を用いた分割の場合の分散平均値は、属性7を用いた場合のそれより、非常に小さい。従って、属性3を選ぶことになる.



停止条件の例

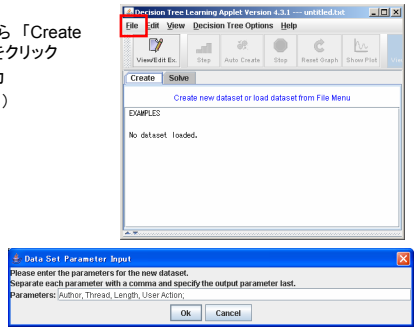
- 分散が十分小さくなったらば、停止する（それ以上、ノード分割はしない）
- この場合、予め考えておいたように、葉の出力値はその平均値とする.



dTree の使い方（一部）

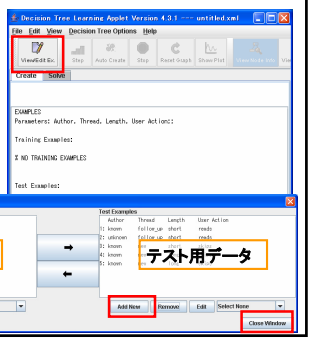
データセットの作り方①

- 左上の「File」から「Create New Dataset」をクリック
- パラメータを入力（カンマで区切る）
- 「OK」をクリック



データセットの作り方②

- 左上の「ViewEdit Examples」をクリック
- 「Add New 」より、各パラメータの値を入力
- 左側に学習用データ、右側にテスト用データを入力し、終わったら「Close Window」を選択



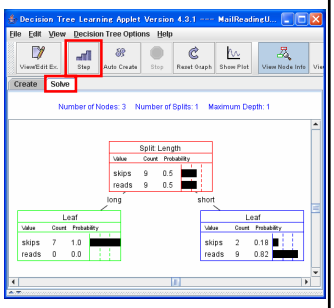
決定木の作成①

- 左上の「Solve」を選択
- 「Step」を選択すると、一段階ずつ木が作成される
- 各ノードの上で左クリックをすると、ノードの情報などを見ることができる

赤：根ノード

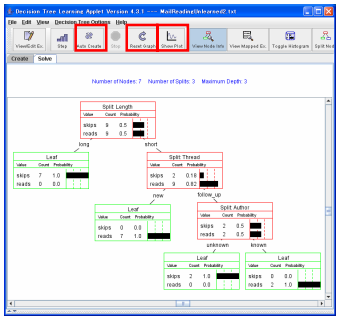
青：葉ノード（さらに分割可能）

緑：葉ノード（これ以上分割不可）



決定木の作成②

- 「Auto Create」を選択すると、最後まで木を生成する
- 「Reset Graph」を選択すると、木を作る前の状態に戻る
- 「Show Plot」を選択すると、error rate をグラフで見ることができる



テストデータへの適用

- 「Test」を選択すると、生成した決定木をテストデータへ適用した結果が表示される
- 「Test New Example」を選択すると、各要素を変更した際、結果がどう変わるかがわかる