大規模属性空間における多数決関数の決定木学習*

5J－3

秋葉 泰弘†
NTTコミュニケーション科学基礎研究所†

1 はじめに

近年，Bagging[2] 等，複数の判別関数を事例から作成し，これら判別関数の多数決により，未知事例のクラスを高精度で判別する手法が注目されている．この手法における多数決判別結果を決定木で近似表現する技術[1] も研究され，知識獲得への応用及び判別時間やメモリ量の大幅な短縮が可能となった．しかし，[1] は数値属性を直接取り扱えず，数値属性を含む訓練事例への適用が難しかった．本稿では，[1] のアルゴリズムを数値属性をも取り扱えるようにした，拡張化アルゴリズムを提案する．提案手法を，Irvine データベースの数値属性を含むデータで評価したところ，良好な学習結果を得た．

2 提案手法

提案手法の概要は以下の通り：

(1) 多数決に利用する判別関数を N 個学習する．
ここで，判別関数は，条件部が pure CNF の if-then ルールに変換出来れば，どのような判別関数でもよい．ここで取り扱う pure CNF とは，

$$(A_{i_1} = V_{j_{i_1}}) \wedge (A_{i_2} > V_{j_{i_2}}) \wedge (A_{i_3} \leq V_{j_{i_3}})$$

のように，() の中身が，or 条件で結合されていない CNF のことである．

(2) 各判別関数 $(T_i, i = 1, \cdots, N)$ を条件部が pure CNF である if-then ルールに表現する．ここで，判別関数 T_i から生成された if-then ルールを R_{ij} $(j = 1, \cdots, N_i.\ N_i$ は，生成された if-then ルールの数) と表記する．

(3) R_{ij} $(i = 1, \cdots, N, j = 1, \cdots, N_i)$ を属性ベクトルで表現する．以下，R_{ij} に対応する属性ベクトルを V_{ij} と表記する．ただし，R_{ij} に現れない属性については，属性値はその値域の値全てであり得るので (以下，*Don't care* 属性と言う．)，その値を * で表す．また，数値属性の属性値は区間 (例えば，$(-\infty, 1], (1, 5.5], (5, +\infty)$ 等) とする．

(4) V_{ij} を訓練事例[1] と見なし，以下の特徴を持つ決定木学習手続きを実行する．

特徴 1：Feature Selection $T_i, i = 1, \cdots, N$ のいずれにも現れない属性は，irrelevant 属性とする．

特徴 2：各ノードにおけるテスト選択 各ノードに対するクラス頻度は通常，そのノードに包含される訓練事例数を数えることにより計算するが，ここでは各訓練事例 即ち 条件部が pure CNF で表現された if-then ルールが覆う領域の直積測度（各属性値の測度の積）により計算する．属性値の測度は，(1)Don't care な記名属性の場合は該属性の取り得る属性値数，(2)Don't care でない記名属性の場合は 1，(3)Don't care な数値属性の場合は，実事例中の該属性に対する最大値と最小値の差，(4)Don't care でない数値属性の場合は，区間の上端と下端の差 (ただし，上端が $+\infty$ 時には最大値で，下端が $-\infty$ 時には最小値で置き換える事とする．) と定義する．

特徴 3：各事例の流し方 各ノードにおけるテストに合致するように，必要に応じて事例を分割し，下位ノードに流す．例えば，最適なテストとして数値属性 A_{i_k} に対し閾値 0.5 が採用された場合，$(\cdots, (-1, 2], \cdots)$ なる事例を下位ノードに流す場合，$(\cdots, (-1, 0.5], \cdots), (\cdots, (0.5, 2], \cdots)$ と分割後，前者を $A_{i_k} \leq 0.5$ が真のエッジに流し，

*On Handling Numerical Attributes in Approximating Majority Voting Classifiers
Yasuhiro Akiba†
† NTT Communication Science Laboratories, 2-4 Hikaridai Seika-cho Souraku-gun Kyoto 619-0237, Japan

[1] これに対し，各判別関数を学習する際に利用した事例をここでは実事例と呼ぶ．

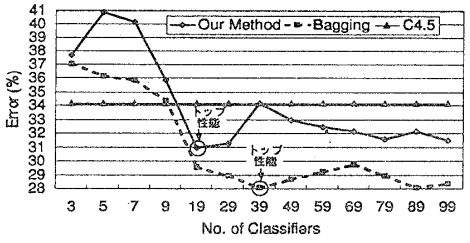

図 1: エラー率 (Bupa)

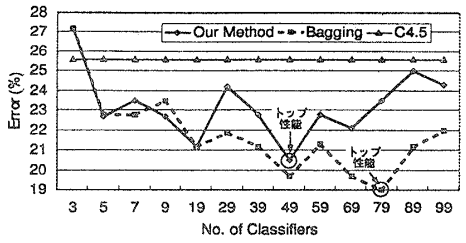

図 2: エラー率 (Hayes-Roth)

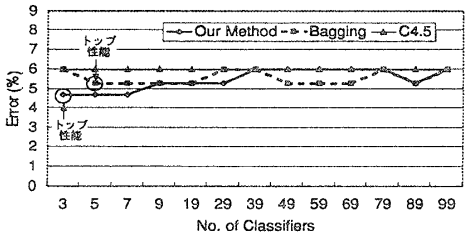

図 3: エラー率 (Iris)

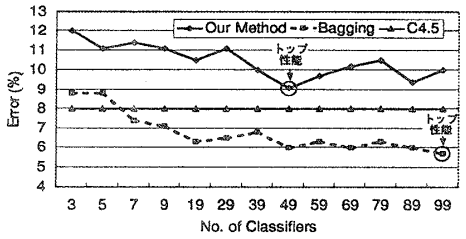

図 4: エラー率 (Ionosphere)

後者を偽のエッジに流す. $(\cdots,(0,0.1],\cdots)$ なる事例は，分割なしに真のエッジに流す.

特徴 4：終了条件 (1) ノードに包含される訓練事例が，全て同一クラスに属するか，(2) ノードに包含される訓練事例が N 個になるか，(3) そのノードに被覆される実事例が 1 つ以下になったら，それ以上分割を行なわない.

特徴 5：枝刈 中間ノードの子が全て同じクラスでラベル付けされていれば，それらの子を刈り，そのクラスでラベル付けをした葉に置き変える.

3 実験的比較

提案手法で得られる決定木のエラー率が，多数決を構成する判別関数の数に応じてどう推移するかを実験した．実験で利用した提案手法に入力する判別関数は，Bagging 流に C4.5[3] で決定木を学習する事により生成した．if-then ルール R_{ij} は生成された決定木のルートから葉に至るパスとした．実験に利用したデータは，UC Irvine のデーターベースからの数値属性を含むデータ，Bupa，Hayes-Roth，Iris，Ionosphere である．各データにつき，多数決を構成する木の数をいろいろ変えて得られる，各手法より生成される決定木のエラー率は，10-fold cross validation により算出した.

図 1〜図 4は，Bagging や C4.5 との，エラー率[2]の差異を示す．横軸は，多数決に加わった判別関数の数である．図 1〜図 3では，提案手法によるエラー率は，トップ性能 (エラー率が一番小さい点) を比較すれば，Bagging に比べてエラー率が大きく上がらず，また C4.5 に比べてエラー率は旨く削減されている．更に，図 3では，提案手法は Bagging よりエラー率が僅かに小さい．従って，提案手法は属性空間が数値属性を含む場合でも判別関数の多数決を旨く近似していると言える．しかし，図 4では，提案手法は C4.5 よりエラー率が大きいが，Bagging は小さい．この意味では，提案手法には改善する余地が残っているかもしれない.

[2]C4.5 のグラフは，エラー率を単に横に伸ばしたものである.

4 おわりに

本稿では，数値属性を含む属性空間上の判別関数の多数決を決定木で近似表現する手法を提案した．Irvine データでその効果を実験的に評価したところ，提案手法による決定木は，未知事例に対するエラー率が大きく増加する事なしに，Bagging による多数決関数を近似出来た．今後は，更なる改良の余地を調べると共に，種々の問題に適用し，その有効性の検証を行なう予定である.

参考文献

[1] Akiba, Y., Kaneda, S. and Allmualim, H., "Turning Majority Voting Classifiers into a Single Decision Tree", *Proc. ICTAI-98*, pp.224–230 (1998).

[2] Breiman, L. "Bagging Predictors", *Machine Learning*, Vol. 24, pp.123–140 (1996).

[3] Quinlan, J., R. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann (1993).