多数決関数の決定木表現*

3T－1

秋葉 泰弘† 金田 重郎‡ フセイン アルモアリム§
NTTコミュニケーション科学研究所†
同志社大学大学院‡ サウジアラビア国立石油鉱物大学§

1 はじめに

近年，Bagging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[2, 3]が注目されている．しかし，これら手法では，クラス判別が属性値のANDやORで理由付けされておらず，学習結果を人間が読み取れないために知識獲得には利用できない．また，大量の判別関数を必要とするため，判別に必要な時間・メモリ量が大きく実用的とは言えない．そこで，本稿では，これら問題点を解決した，単一のコンパクトな決定木で多数決判別関数と同等の精度を得る新しい学習手法を提案する．提案手法を，アーバインデータベースの事例で評価したところ，良好な学習結果を確認した．

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．pure CNFとは，

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

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．
ただし，R_{ij}に現れない属性については，その属性値は取り得る値ならなんでもいいので(以下，*Don't care*属性と言う．)，その値を$*$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．

(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 $T_i, i = 1, \cdots, N$の内，半数未満のT_iにしか，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大きさを数えることにより計算する．大きさは，

$$\prod_{\text{各Don'tCare属性}} \text{Don'tCare属性の属性値数}$$

により計算される．

特徴3：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4：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．

このようにして得られた最終的な決定木は，判別関数の多数決を旨く近似する．手続き(1)で，Bagging流[2]

---

*Learning a Single Decision Tree that Approximates Majority Voting Classifiers.
Yasuhiro Akiba†, Shigeo Kaneda‡, Hussein Almuallim§
† NTT Communication Science Laboratories, 2-4 Hikaridai Seika-cho Souraku-gun Kyoto 619-0237, Japan
‡ Graduate School of Policy and Management, Doshisha University, Karasuma-Higashi-Iru, Imadegawa-Toori, Kamigyo-ku, Kyoto-shi, 602-8580, Japan
§ Dept. of Information and Computer Science, King Fahd University of Petroleum & Minerals, Dhahran 31261,Saudi Arabia

[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．

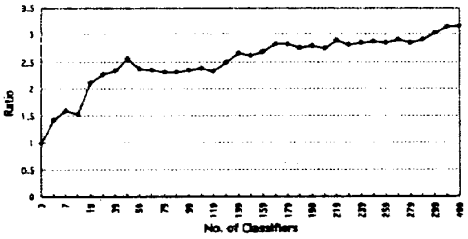

図 1: サイズ率 (Monk1)

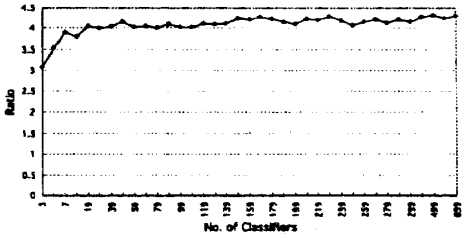

図 2: サイズ率 (Soybean)

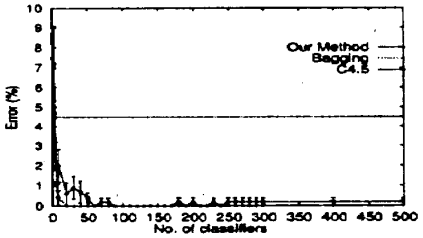

図 3: エラー率 (Monk1)

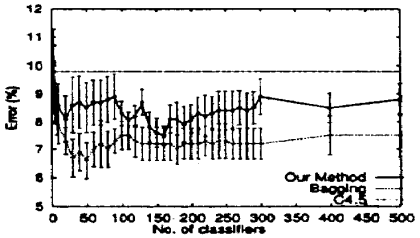

図 4: エラー率 (Soybean)

に判別関数を複数準備すれば，Bagging 流のクラス分けルールが学習され，Adaboost 流 [3] に判別関数を複数準備すれば，Adaboost 流のクラス分けルールが学習される．

3 実験的比較

本手法で得られる決定木のエラー率と大きさが，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．実験で利用した本手法に入力する判別関数は，Bagging 流に C4.5[4] 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データである．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大きさとエラー率は，10-fold cross validation により算出した[2]．

図 1〜図 2は，C4.5 による決定木の大きさに対する，提案手法による決定木の大きさの比に関するグラフである．提案手法による決定木の大きさは，C4.5 による決定木の大きさの約 4 倍であり，多数決関数が非常にコンパクト[3]に表現され得た．

図 3〜図 4は，Bagging や C4.5 との，エラー率[4]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．グラフ中の縦線はエラーバー (エラー率の標準偏差の幅) である．なお，C4.5 の標準偏差は，Monk1 で 1.627，Soybean で 0.855 である．最終的な決定木が非常にコンパクトであるにも関わらず，提案手法によるエラー率は，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．

本提案手法により，[1] で提案した手法の決定木サイズに関する問題点を改善できた．

4 おわりに

本稿では，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，C4.5 による決定木の約 4 倍の大きさとコンパクトに表現でき，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，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] 秋葉 他，“ブートストラップに基づく決定木学習”，第 55 回情報処理学会全国大会，Vol. 2, pp.521–522 (1997).

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

[3] Freund, Y. and Schapire R.,E., “A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting”,*Porc. EuroCOLT'95*, pp. 23–37 (1995).

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

[2]本稿では 2 つのデータセットに対する結果を示すが，他のデータについても同様な結果を得ている．

[3]Bagging に十分な性能を発揮させるには，100 以上の判別関数としての決定木をメモリーに蓄える必要があるが，それが 4 つ分程度で良くなったと言う意味で大きくコンパクト化されたと言える．

[4]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．