知能と複雑系　121－11
（2000. 7. 19）

# ブースティングによる精度向上を実現する単一決定木の学習

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基礎研究所[†]　同志社大学大学院[‡]　サウジアラビア国立石油鉱物大学[§]

〒619–0237 京都府相楽郡[†]　〒602-8580 京都市上京区[‡]　P.O. Box 801, Dhahran[§]
精華町光台 2-4　今出川通り烏丸東入ル　31261, Saudi Arabia

あらまし　学習アルゴリズムのエラー率を削減する技巧として，Bagging 等の多数決によるクラス判別手法が近年注目されている．これらクラス判別では，クラス判別理由が論理表現で記述されない．また，エラー率を十分に削減するには，クラス判別関数１つだけでクラスを判別する場合に比べ，十数倍から数百倍も判別時間やメモリー量が必要となる．これらの問題を解決するために，本稿では多数決クラス判別を近似する単一の決定木を学習する手法を提案する．提案手法は，各判別関数と等価なメタ事例を訓練事例とし，オリジナルの事例の頻度分布を基に無用なノード分割を避けながら単一の決定木を学習する．提案手法を実験的に評価したところ，Bagging による多数決クラス判別の高精度を保ちつつ，判別関数２つ分程度の表現量の単一決定木が獲得できる事を確認した．

キーワード　機械学習，決定木，多数決，正解率向上，平均化

# Acquiring a Single Decision Tree Representation of Majority Voting Classifiers

Yasuhiro AKIBA[†]　Shigeo KANEDA[‡]　Hussein ALMUALLIM[§]

NTT Communication Science Labs.[†]　Doshisha Univ.[‡]　King Fahd Univ. of Petroleum & Minerals[§]

2–4 Hikaridai, Seika-cho,[†] Souraku-gun, Kyoto, 619–0237 Japan
Karasuma-Higashi-Iru,[‡] Imadegawa-Dohri, Kamigyo-ku, Kyoto, 602–8580 Japan
P.O. Box 801, Dhahran,[§] 31261, Saudi Arabia

**Abstract**　This paper addresses two problems in Majority voting classifiers (MVCs) like Bagging: (1) no logical reasoning behind the decision and (2) a large amount of classification time and space for significant accuracy boosting. To solve these problems, this paper proposes a method for learning a single decision tree that approximates MVCs. The method learns a DT from the original examples and the meta examples that are generated from each classifier joining MVCs. Experimental results show that the method has similar accuracy to Bagging and that the tree size by the method is as large as the size of two classifiers.

**Key words**　Machine Learning, Decision Tree, Majority Voting, Accuracy Boosting, Averaging

## 1 はじめに

学習アルゴリズムのクラス判別エラー率を削減する手法として，各種ブースティング法 [7, 11] が近年提案され，注目を集めている．ブースティング法 [4] では，まず，予め準備した事例（以下，実事例）の一部を訓練事例として学習アルゴリズムに入力し，クラス判別関数を学習する．入力される訓練事例の取り方を色々変えながら，クラス判別関数を繰り返し学習する．未知事例に対してそのクラスを判別するためには，学習した各クラス判別関数にクラスを判別させ，判別結果のクラスのうち，一番判定数の多かったクラスを未知事例に対するクラスとする．以下，この判別方法を多数決クラス判別と呼ぶ．ブースティング法は理論的にも実験的にも評価され，学習アルゴリズムのクラス判別エラー率が十分に改善される事が検証された [7, 11, 14, 6]．データマイニングや知識獲得を始め，色々な分野で学習アルゴリズムを適用しようとする気運が高まっており [3]，これらの評価結果は非常に意義深い．

しかし，多数決クラス判別により実現された，クラス判定エラー率の改善法には，2つの欠点がある．

(1) クラス判別の結果が，事例の属性値に関する論理式の論理積や論理和で明確に表現されていない．このため多数決クラス判別は，データマイニングや知識獲得と言った，知識を明確に表現する事が要求されるタスクへの適用には，不向きである．ベイズ統計やニューラルネットと言ったクラス判別を取り扱う関連手法に比べ，機械学習が優れている点は，機械学習がクラス判別に際し，その理由付けを与える点にある．多数決によるクラス判別では，この長所が残念ながら失われる．

(2) エラー率を十分に削減するためには，多くのクラス判別関数で多数決を取る必要があり，クラス判別に必要な時間やメモリー量は判別関数の数だけ増加する．学習データにもよるが，1つの判別関数だけでクラス判別を行なう場合に比べ，十数倍から数百倍も判別時間やメモリーを浪費する．従って，機械学習が潜在的に適用が可能であると思われる応用分野でも，クラス判別が頻繁に必要となる応用分野や限られたメモリーにクラス判別関数を格納する必要がある応用分野では，その浪費が弊害になり適用が難しくなる．

本稿における目的はこれら2つの問題点を解決する事にある．そのために，本稿では，多数決クラス判別を単一の決定木として近似的に学習する手法を提案する．提案手法では，まず，各クラス判別関数から if-then ルールを生成し，これら if-then ルールと等価なメタ事例を訓練事例と見做して，実事例の頻度分布を基に無用なノード分割を避けながら単一の決定木を学習する．

カルフォルニア大学アーバイン校が管理するベンチマークデータベース，UC-Irvine Repository [12]，から10種のデータを選択し，それらで提案手法を実験的に評価した．提案手法に入力する複数のクラス判別関数には，Breiman による Bagging[7] 流の方法で獲得したクラス判別関数を使い，Bagging 及び C4.5 [13] と比較を行なった．実験の範囲では，提案手法により学習した決定木の大きさは，C4.5 により学習した決定木の大きさの平均2個分程度に納まった．また，提案手法によるエラー率は Bagging とほぼ同程度であり，提案手法で学習された決定木によるクラス判別時間は，Bagging におけるクラス判別時間より，約10倍～250倍（平均80倍）速い．

次節で以下本稿で取り上げる学習タスクを概説し，3節で，著者らのアプローチを提案する．4節で，提案手法に関する実験結果を示し，それらを考察する．最後5節でまとめる．

## 2 学習タスク

本稿で取り上げる学習タスクは，多数決クラス判別と同等のエラー率を保持したまま，多数決クラス判別を単一の決定木に近似表現する事である．

ここで強調したい点は，本稿の最終目標は多数決クラス判別を近似表現する事であり，正確に再現する事ではないと言う事である．多数決関数は対称関数であり，DNF や CNF で表現した場合と同様，決定木で表現した場合にも，その記述量が入力数の指数オーダーになる [15]．著者らは手頃な大きさの決定木にのみ興味があるので，多数決クラス判別の近似表現を求める事が必要となる．

無難な近似を獲得するために，各クラス判別関数を学習する際に選択する訓練事例の母集団分布（実事例の頻度分布）を参照する事を許す．この狙いは，各クラス判別関数と等価なメタ事例を訓練事例として，多数決判別を正確に表現した際に起こる，決定木の表現爆発を避け，また，多数決判別を決定木で近似表現した際，決定木のエラー率が激増する事を避ける事にある．

本稿で取り扱う学習タスクをまとめると，その入出力は以下の様になる．

入力： メタ事例 (複数のクラス判別関数)，及び実事例（クラス判別関数獲得時に利用した訓練事例の母集団）

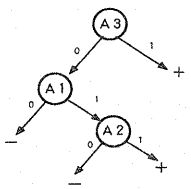


図 1: 決定木の例

出力： 多数決クラス判別を近似表現する決定木

## 3 提案手法

説明を簡略化するために，以下ではクラス判別関数が決定木で表現されている場合に限定して話しを進めるが，クラス判別関数が，以下に示すような論理和を含まない選言標準形（Pure CNF）を条件部に持つ if-then ルールへ変換可能な場合には，容易に拡張が可能である．

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

ここで，$A_{i_1}$ 等は属性を表し，$V_{i_1}$ 等は属性値を表す．

更に，実事例を表現する属性全ては記号属性（nominal attribute）であるとする．数値属性を取り扱う際には，何らかの離散化手法（例えば，[10]）で 実事例を前処理する事により，提案手法を適用出来る．実験報告 [9] によると，離散化手法 [10] で前もって訓練事例を離散化したとしても，学習アルゴリズムのエラー率が増す事は無かった．また，ベンチマークによっては，むしろエラー率が削減される事すらあった．この点に基づき，本稿でも，数値属性を取り扱う必要がある場合には，離散化手法を前処理として利用する．

提案手法の概要は以下の通りである．

手続き（0） 多数決に利用する複数のクラス判別関数を準備する．以下，準備したクラス判別関数の数を $N$ と表記する．この数は予め決めておく．

手続き（1） 準備した各クラス判別関数 $T_i$ $(i = 1, \cdots, N)$ について，$T_i$ を条件部が Pure CNF である if-then ルールに変換する．以下，$T_i$ から生成される if-then ルールを $R_{i,j}$ $(j = 1, \cdots, N_i)$ と表記し，生成される if-then ルール数を $N_i$ と表記する．例えば，$T_1$ が図 1に示す決定木である場合， if-then ルールはルートと葉を結ぶ各パスに対応する．ルートと左端の葉を結ぶパスは， if-then ルール， If $(A_3 = 0) \wedge (A_1 = 0)$ then class $-$ に変換される．従って，$T_1$ は最終的に 4 つの if-then ルール，$R_{1,1}$: If $(A_3 = 0) \wedge (A_1 = 0)$ then class $-$, $R_{1,2}$: If $(A_3 = 0) \wedge (A_1 = 1) \wedge (A_2 = 0)$ then class $-$, $R_{1,3}$: If $(A_3 = 0) \wedge (A_1 = 1) \wedge (A_2 = 1)$ then class $+$, 及び $R_{1,4}$: If $(A_3 = 1)$ then class $+$, と変換される．

手続き（2） 各 if-then ルール $R_{i,j}$ $(i = 1, \cdots, \mathrm{N}, j = 1, \cdots, N_i)$ を属性ベクトル（メタ事例）に変換する．各属性ベクトルにおける各属性の属性値は，その属性に関する制約がルール中に存在するか否かにより決める．存在する場合には，その属性の属性値としてその制約中の属性値を指定する．存在しない場合には，$*$ を指定する．これは属性値としては取り得る値なら何でもよい事を示す．例えば，実事例が 4 つの属性，$A_1$，$A_2$，$A_3$及び$A_4$ で表現されている時，前述の if-then ルール $R_{1,1}$ には，$A_1$ と $A_3$ に関する制約は順に $(A_3 = 0)$，$(A_1 = 0)$ と存在するが，$A_2$ と $A_4$ に関する制約はない．従って，$R_{1,1}$ は，属性ベクトル $(0, *, 0, *, -)$ へと変換される．以下，$R_{i,j}$ から変換された属性ベクトル（メタ事例）を $V_{i,j}$ と表記する．

手続き（3） $V_{i,j}$ を訓練事例と見做して，図 2に示す決定木学習[1]を行ない，最終的な単一の決定木を学習する．本稿で提案する決定木学習は以下の特徴を持つ．

【特徴 1：属性選択】 各 $T_i, i = 1, \cdots, N$, において，どの $T_i$ にも現れなかった如何なる属性も，非関連属性（irrelevant attribute）[4] と見做し，削除する．

【特徴 2：各中間ノードにおけるテスト選択】 各中間ノードにおけるクラス頻度は， C4.5[13] の場合，そのノードに到達した訓練事例の個数をクラス別に数え上げる事により得るが，提案手法では，各ノードに到達した訓練事例が被覆する領域の大きさ[2]を合計する事により得る．例えば，前述の属性ベクトル $V_{1,1}$, $(0, *, 0, *, -)$, の場合，$V_{1,1}$ は事例空間の 4 点，$(0, 0, 0, 0, -)$, $(0, 0, 0, 1, -)$, $(0, 1, 0, 0, -)$, $(0, 1, 0, 1, -)$ を被覆する．従って，$V_{1,1}$ が被覆する領域の大きさ

[1]提案手法では，各中間ノードにおけるテスト候補を評価するために実事例の頻度を数えない．一方，事例を子ノード別に分割する際には，実事例を，訓練事例と共に分割する．実事例の数は，分割の終了条件においてのみ使われる．

[2]提案手法では，クラス頻度の計算に 実事例の個数を使わない．

```
S := a set of training examples
R := a set of real examples

(Step 1) Do feature selection using the first property.
(Step 2) If the stop condition of the third property is true,
      make a node labeled the most frequent class and return the node.
(Step 3) Assuming that each training example in S has a weight value
      calculated as per the second property,
      evaluate each candidate test in the same way as ordinary DT leaning, and
      select the best test with the best score.
(Step 4) Accoding to the answer to the test,
      make a test node, and divide S and R into subsets Si and Ri respectively,
      where each element of Si and Ri gives the same answer to the test.
(Step 5) For each pair of Si and Ri,
      let S=Si and R=Ri, and repeat the procedures in (Step 2) to (Setp 7).
(Step 6) Make a tree using all subtrees.
(Step 7) Prune the tree from (Step 6) using the fourth property, and
      return the pruned tree.
```

図 2: 提案手法の手続き (3)

は，4となる．一般に各訓練事例が被覆する領域の大きさは，

$$\prod_{\text{各 Don't Care 属性}} \begin{pmatrix}\text{Don't Care 属性の取り得る}\\ \text{値の数}\end{pmatrix}$$

と計算される．

【特徴 3：終了条件】 以下に示す条件 (1) ～ (3) のいずれかを満たす時，分割を中止する．即ち (1) そのノードに到達した訓練事例が全て同じクラスに属する場合，(2) そのノードに到達した訓練事例の数が $N$ 個になった場合，(3) そのノードに到達した 実事例の数が1個以下になった場合[3]．いずれかの条件で分割を中止した場合，現在のノードを葉（終端ノード）とし，C4.5 流の方法で葉にクラスレベルを付ける．

【特徴 4：枝刈り】 ある中間ノードの子ノードが全て同じクラスにラベル付けられた葉であれば，これらの子ノードを枝刈りする．即ち，その中間ノードを葉に置き換え，刈られる葉のクラスでラベル付ける．

以上が提案手法の概要である．各訓練事例は，クラスを異にする可能性のある訓練事例を必ず $N$ 個持つ．従って，意味の無い分岐が生成される可能性があるが，上述の枝刈りの手続きにより，そのような意味のない分割は最終的に取り除かれる．

得られた単一の決定木は，対応する多数決クラス判別を近似する．というのは，第3の特徴の終了条件 (3) により分岐を止めるため，決定木学習における分割は 実事例の頻度分布が正である限り実行され，その結果，無意味な分割が回避される．

[3]実事例は各クラス判別関数を学習するために抽出される訓練事例の母集団であり，if-then ルールから生成された訓練事例とは異なる．

## 4 実験比較

本節では，提案手法よるエラー率と決定木の大きさを実験的に評価した．入力するクラス判別関数の数を順に増加させ，その数と共に，最終的な決定木の大きさとエラー率がどの様に変化するかを調べた．エラー率と決定木の大きさは，10-fold Cross Validation により計算した．提案手法に入力する複数のクラス判別関数は，Bagging 流の方法で訓練事例を生成し [7, 4]，C4.5[13] により生成した．実事例の頻度分布を計算するためには，クラス判別関数を準備するために利用した訓練事例の母集団を利用した．提案手法の数値計算を実現するために，GNU MP 2.0.2を利用した．比較のために，Bagging と C4.5 も又評価した．Bagging に利用するクラス判別関数もまた，C4.5 で学習した．

評価データはカルフォルニア大学アーバイン校が管理するベンチマークデータベース，UC-Irvine repository [4] [12] より選んだ，表1に示す，数値属性を含まない 10 種類のデータである．

### 4.1 未知事例に対するエラー率

評価データに対するエラー率は図3～図12に示す通りである．ここで，学習アルゴリズムは，提案手法，Bagging 及び C4.5 である．横軸がクラス判別関数の数を示し，縦軸は未知事例に対するエラー率を示す．実線，破線，及び一点鎖線は，各々，提案手法，Bagging，C4.5 に対する

[4]The breast-cancer, primary-tumor domains were obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Thanks go to M. Zwitter and M. Soklic for providing these data.

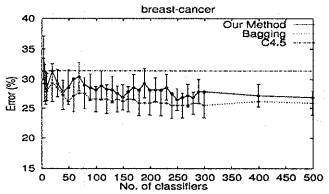


図 3: エラー率（breast-cancer）
C4.5's $\sigma = 2.616$

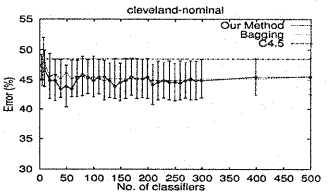


図 4: エラー率（cleveland-nominal）
C4.5's $\sigma = 2.721$

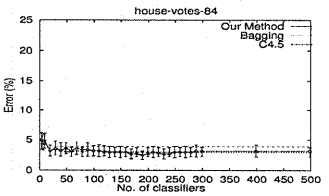


図 5: エラー率（house-votes-84）
C4.5's $\sigma = 1.130$

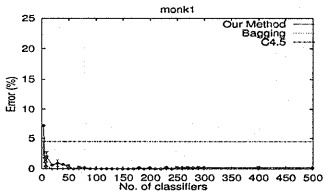


図 6: エラー率（monk1）
C4.5's $\sigma = 1.627$

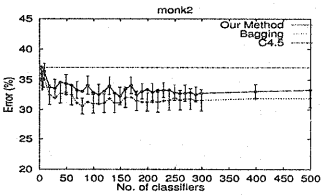


図 7: エラー率（monk2）
C4.5's $\sigma = 2.078$

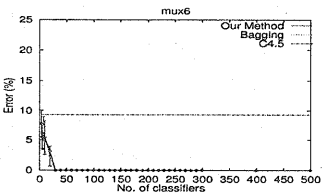


図 8: エラー率（mux6）
C4.5's $\sigma = 3.203$

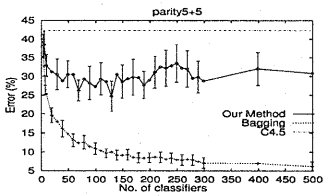


図 9: エラー率（parity5+5）
C4.5's $\sigma = 4.357$

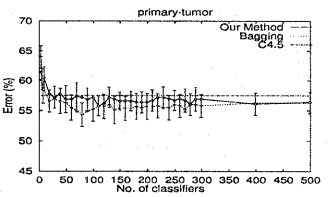


図 10: エラー率（primary-tumor）
C4.5's $\sigma = 2.502$

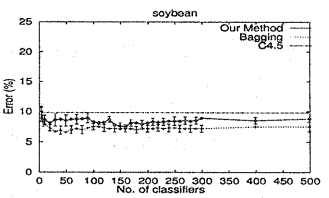


図 11: エラー率（soybean）
C4.5's $\sigma = 0.855$

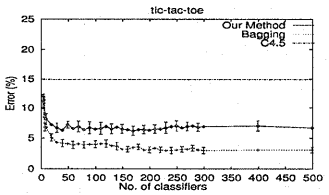


図 12: エラー率（tic-tac-toe）
C4.5's $\sigma = 3.950$

エラー率を示す．各グラフのエラーバーは，各エラー率の標準偏差を示す．ただし，見易さのために，エラーバーは，提案手法と Bagging 交互に記してある．C4.5 のエラー率は，1 つの決定木に対するエラー率であり，従ってクラス判別関数の数とは関係ない．各図の注釈中の $\sigma$ は，C4.5 に対するエラー率の標準偏差を示す．実験結果は，以下通りである．

- 多くのデータ（図 6，図 7，図 8，図 9，図 11，図 12）上で，入力されるクラス判別関数の数が十分に大きい場合にはエラー率の振れを示すエラーバーが離れており，クラス判別関数の数が十分に大きいければ，提案手法は，C4.5 よりエラー率が低い．
- 殆んどのデータ（図 9，図 12を除く）で，そのエラーバーが重なるか接しており，提案手法は，Bagging と同程度のエラー率であると言える．
- 提案手法によるエラー率削減は，入力される判別関数が 50 前後で，ある程度起きている．より多くのエラー率削減を得るためには，より多くの判別関数を入力する必要がある．

エラー率の差に関する厳密な議論は，最小エラー率に絞って，4.3節で paird t-test[8] を用いて行なう．

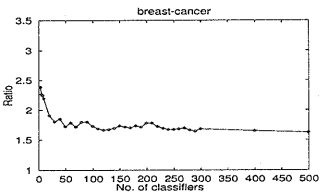


図 13: サイズ比（breast-cancer）

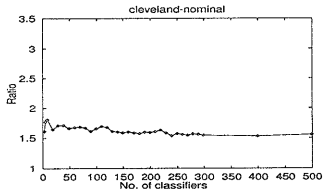


図 14: サイズ比（cleveland-nominal）

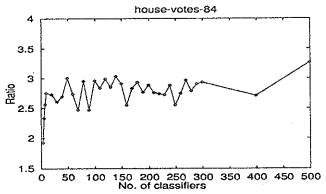


図 15: サイズ比（house-votes-84）

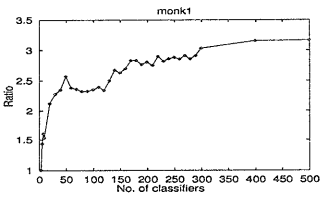


図 16: サイズ比（monk1）

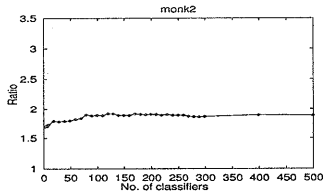


図 17: サイズ比（monk2）

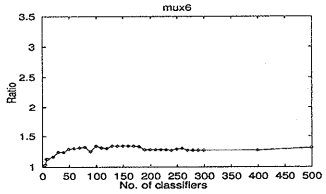


図 18: サイズ比（mux6）

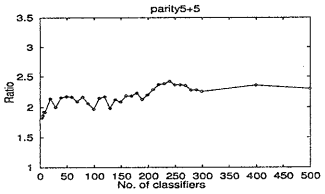


図 19: サイズ比（parity5+5）

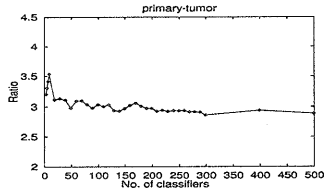


図 20: サイズ比（primary-tumor）

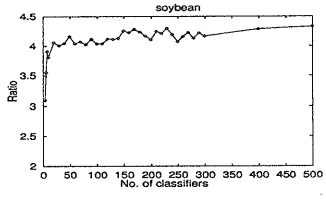


図 21: サイズ比（soybean）

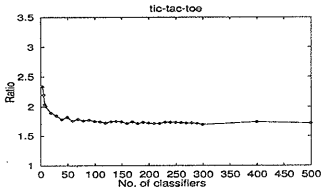


図 22: サイズ比（tic-tac-toe）

### 4.2 最終的に学習された決定木の大きさ

図 13〜図 22は提案手法で生成された決定木の大きさを示す．横軸は入力されたクラス判別関数の数を示し，縦軸は 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相対比である．ここで言う，C4.5 による決定木は 実事例を訓練事例として学習された決定木である．従って，相対比が 1 であると仮定すると，入力された複数のクラス判別関数による多数決クラス判別がたった 1 つの決定木に劇的に圧縮された事を意味する．主な所見をまとめると以下の様になる．

- 図 13〜図 22が示す様に，高い正解率を得るために入力するクラス判別関数を大きく増加させた場合でも，その増加率と同程度には増加せず，殆んど一定である．実際，図 13〜図 22においては，入力される評価関数の数が，3 〜 499 と増加させても，相対比は 4.5 を越えない．

この効果的な圧縮は，実事例の頻度分布を使って無用な分割を避けた効果であり，最終的に得られる決定木の大きさが爆発する事を防止出来たと言える．

### 4.3 トップ性能における比較

提案手法と Bagging の最小エラー率及び C4.5 のエラー率を表 1で比較する．第 4 欄は，図 3〜図 12におけるエラー率の最小値を示す．第 3 欄は，最小エラー率を得た際のクラス判別関数の数を示す．第 5 欄は，第 2 欄で示す手法のエラー率が，Bagging のエラー率に対して統計的に有意な（以下単に‘有意な’と略す）差があるか否かを 10-fold Cross Validated Paird t-test（$p = 0.05$）[8] により検定した結果ある．‘≈’は有意な差が検出されなかった事を示し，

表 1: トップ性能比較：第 4 欄は，図 3〜図 12の各グラフにおけるエラー率の最小値を示す．第 3 欄は，最小エラー率を得た際のクラス判別関数の数を示す．第 5 欄は，第 2 欄の手法のエラー率が，Bagging のエラー率に対して統計的に有意な差があるか否かを示す．'≈' は差が検出されなかった事を示し，'≈' 以外は差が検出された事を示す．第 6 欄は，第 5 欄と同様で Bagging の代わりに C4.5 で検定を行なった結果である．第 7 欄は，各手法に対するクラス判別時間で，C4.5 によるクラス判別時間に対する相対比を示す．

| Data set | | Methods | #Classifiers | Error Rate(%) | B | C | Classification times |
|---|---|---|---|---|---|---|---|
| 〈 1 〉 | breast-cancer | 提案手法 | 249 | 26.5±1.9 | ≈ | ≈ | 1.7 (1) |
| | | Bagging | 239 | 25.5±2.2 | — | ≈ | 239 (140.6) |
| 〈 2 〉 | cleveland-nominal | 提案手法 | 39 | 43.5±3.4 | ≈ | ≈ | 1.7 (1) |
| | | Bagging | 119 | 44.9±3.4 | — | ≈ | 119 (70.0) |
| 〈 3 〉 | house-votes-84 | 提案手法 | 189 | 2.5±0.7 | ≈ | ≈ | 2.8 (1) |
| | | Bagging | 49 | 3±0.8 | — | ≈ | 49 (17.5) |
| 〈 4 〉 | monk1 | 提案手法 | 59 | 0±0.0 | ≈ | 提 | 2.4 (1) |
| | | Bagging | 19 | 0±0.0 | — | **B** | 19 (7.9) |
| 〈 5 〉 | monk2 | 提案手法 | 149 | 32.3±1.5 | ≈ | ≈ | 1.9 (1) |
| | | Bagging | 79 | 30.6±1.3 | — | **B** | 79 (41.6) |
| 〈 6 〉 | mux6 | 提案手法 | 29 | 0±0.0 | ≈ | 提 | 1.2 (1) |
| | | Bagging | 29 | 0±0.0 | — | **B** | 29 (24.2) |
| 〈 7 〉 | parity5+5 | 提案手法 | 129 | 24.7±4.0 | **B** | 提 | 2.0 (1) |
| | | Bagging | 499 | 6.2±1.2 | — | **B** | 499 (249.5) |
| 〈 8 〉 | primary-tumor | 提案手法 | 109 | 55.8±1.6 | ≈ | ≈ | 3.0 (1) |
| | | Bagging | 79 | 54.3±1.9 | — | ≈ | 79 (26.3) |
| 〈 9 〉 | soybean | 提案手法 | 159 | 7.5±0.3 | ≈ | 提 | 4.2 (1) |
| | | Bagging | 59 | 7±0.6 | — | **B** | 59 (14) |
| 〈 10 〉 | tic-tac-toe | 提案手法 | 169 | 6.2±0.7 | **B** | 提 | 1.7 (1) |
| | | Bagging | 189 | 3.1±0.5 | — | **B** | 189 (111) |
| 〈 1 〉〜 | Data sets without | 提案手法 | Ave:127.9 | Ave:20.1±1.6 | **B**×2 | 提 ×5 | 2.0 (1) |
| 〈 10 〉 | numerical attributes | Bagging | Ave:130.0 | Ave:19.3±1.8 | — | **B**×6 | 130.2 (77.1) |
| Remarks | | C4.5 | — | Ave:26.1±2.6 | — | — | — |

'**B**' は，有意な差が検出された事，及び Bagging のエラー率の方が小さい事を示す．第 6 欄は第 5 欄と同様で，第 2 欄で示す手法のエラー率が，C4.5 のエラー率に対して有意な差があるか否かを 10-fold Cross Validated Paird t-test ($p = 0.05$) により検定した結果ある．'≈' は有意な差が検出されなかった事を示し，'提' は有意な差が検出された事，及び提案手法のエラー率の方が C4.5 のエラー率に比べ小さい事を示す．'**B**' は有意な差が検出された事，及び Bagging のエラー率の方が C4.5 のエラー率に比べ小さい事を示す．第 7 欄は，各手法に対するクラス判別時間で，C4.5 によるクラス判別時間に対する相対比を示す．なお，決定木と Bagging のクラス判定は各々，決定木の深さ，及び多数決の数に比例するので，これらから各相対比を概算した．この欄のカッコ中の値は，提案手法に対するクラス判定時間に対する Bagging のクラス判定時間の相対比である．主な結果は以下の通りである．

- 提案手法は，10 個中 5 個のデータ上で C4.5 に比べ最小エラー率が有意に小さい．それに対し，Bagging では，10 個中 6 個のデータ上で C4.5 に比べ最小エラー率が有意に小さい．エラー率の削減の効果は，Bagging には若干劣るものの大意にあると言える．
- 提案手法の最小エラー率は，10 個中 8 個のデータ上で，Bagging の最小エラー率と有意な差は無かった．
- データ 〈7〉 の結果からすると，パリティ問題の様に多くの学習アルゴリズムが不得意とする複雑なクラス判別問題に対しては，エラー率の削減幅は低くなると思われる．
- 提案手法は Bagging と比べ，クラス判別速度が，約 10 〜 250 倍（平均約 80 倍）速い．

## 5 おわりに

本稿では，多数決によるクラス判別における，判別理由の表現化，判別速度，及びメモリー量に関する問題を取り上げた．これらの問題を解決するために，複数のクラス判別関数による多数決を単一のコンパクトな決定木として学習する手法を提案した．具体的には，まずオリジナル事例から何らかの方法で複数の判別関数を学習し，各判別関数をそれらと等価なメタ事例に変換する．これらメタ事例を訓練事例とし，オリジナル事例の分布情報を基に無用なノード分割を避けながらクラス判別関数の多数決を単一の決定木として近似表現する．提案手法は，多数決に使われるクラス判別関数が if-then ルールに変換出来さえすれば如何なる判別関数に適用可能である．

提案手法を評価するために数値属性を含まない 10 種類の Irvine データで実験比較したところ，提案手法による決定木は Bagging による決定木と同程度のエラー率であった．提案手法により学習した決定木の大きさは，C4.5 により学習した決定木の大きさの平均 2 個分に納まり，提案手法による判別速度は，Bagging による判別速度の平均 80 倍速かった．これは，提案手法が多数決によるクラス判別をコンパクトな単一の決定木としてうまく近似出来たためである．また，多数決によるクラス判別はコンパクトな決定木により表現され，かつ Bagging と同程度の誤り率である事から，そのクラス判定理由が事例の属性値に関する論理式でうまく論理表現出来たと言える．

## 参考文献


[1] 秋葉泰弘，アルモアリムフセイン，横尾昭男，金田重郎，“ブートストラップに基づく決定木学習，” 第 55 回情処全大，vol.2，pp.521–522，Sep. 1997.

[2] 秋葉泰弘，金田重郎，アルモアリムフセイン，“多数決関数の決定木表現，” 第 57 回情処全大，vol.3，pp.323–324，Oct. 1998.

[3] 秋葉泰弘，アルモアリムフセイン，金田重郎，“例からの学習技術の応用に向けて - 1. 基本技術とその応用上の課題，” 情報処理，vol.39，no.2，pp.145–151，Feb. 1998.

[4] 秋葉泰弘，アルモアリムフセイン，金田重郎，“例からの学習技術の応用に向けて - 2. 応用上の課題に対する解決法，” 情報処理，vol.39，no.3，pp.245–251，March 1998.

[5] Y. Akiba, S. Kaneda and H. Almuallim, “Turning majority voting classifiers into a single decision tree,” Proc. IEEE ICTAI98, pp.224–230, Taipei, Taiwan, Nov. 1998.

[6] E. Bauer and R. Kohavi, “An empirical comparison of voting classification algorithms: Bagging, Boosting, and variants,” Machine Learning, vol.36, no.1–2, pp.105–139, July 1999.

[7] L. Breiman, “Bagging predictors,” Machine Learning, vol.24, no.1, pp.123–140, Aug. 1996.

[8] T.G. Dietterich, “Approximate statistical tests for comparing supervised classification learning algorithms,” Neural Computation, vol.10, no.7, pp.1895–1923, Oct. 1998.

[9] J. Dougherty, R. Kohavi and M. Sahami, “Supervised and unsupervised discretization of continuous features,” Proc. ICML95, pp.194–202, California, USA, July 1995.

[10] U.M. Fayyad, and K.B. Irani, “Multi-interval discretization of continuous-valued attributes for classification learning,” Proc. IJCAI93, pp.1022–1027, Chambéry, France, Aug. 1993.

[11] Y. Freund and R.E. Schapire, “A decision-theoretic generalization of on-line learning and an application to Boosting,” Porc. EuroCOLT95, pp.23–37, Barcelona, Spain, March 1995.

[12] C.J. Merz and P.M. Murphy, UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html], Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[13] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, 1993.

[14] J.R. Quinlan, “Bagging, Boosting and C4.5,” Proc AAAI96, pp.725–730, Oregon, USA, Aug. 1996.

[15] I. Wegener, “Optimal decision tree and one-time-only branching programs for symmetric boolean function,” Information and Control, vol.62, no.2–3, pp.129–143, Aug.–Sept. 1984.