

## ブートストラップに基づく決定木学習\*

3AH-2

秋葉 泰弘† フセイン アルモアリム‡ 横尾 昭男† 金田 重郎§

NTTコミュニケーション科学研究所†

サウジアラビア国立石油鉱物大学‡ 同志社大学大学院§

## 1 はじめに

本稿では、決定木学習 ([3] を参照) の安定化問題を取り上げる。最近、決定木学習を始めとする、学習アルゴリズムの不安定性に関する議論がなされ、その対処方法としてブートストラップに基づく解決法がいくつか提案されている ([1, 2] を参照)。しかし、これらの手法では、学習結果が記号レベルで記述されていないため学習結果が可視化されず、応用上問題がある。本稿では、学習結果が可視化される、決定木学習の安定化法を提案する。また、人工データを利用した学習実験で、その効果を示す。

## 2 タスク

まず、ここで問題としている、学習アルゴリズムの安定性について簡単に説明し、その後、本稿で取り上げる学習タスクを明確にする。

学習アルゴリズムが安定であるとは、入力される訓練事例が少し変わっても、そのアルゴリズムによる学習結果も少ししか変わらないことを言う。逆に、学習アルゴリズムが不安定であるとは、入力される訓練事例が少し変わると、そのアルゴリズムによる学習結果が大きく変わることを言う。実問題解決によく利用される決定木学習も、不安定な学習アルゴリズムである。

この学習アルゴリズムの不安定性は、実問題にアルゴリズムを適用する際弊害となる。というのは、準備できる訓練事例により学習結果が大きく変わるのであれば、学習結果の信頼性に大きな問題が生ずるからである。

不安定な学習アルゴリズムを安定化する方法として、ブートストラップに基づく方法が提案されている ([1, 2] を参照)。ブートストラップは、

- (i) 何らかの方法で訓練事例をリサンプリングし、
- (ii) そのリサンプリングされた訓練事例で学習する。

という操作を複数回行ない、複数の学習結果の多数決により結論の判定を行なう。推定するものは、クラス判定問題の場合はクラスとなる。[1]では、Baggingと言う、ブートストラップ法が提案され、復元ランダム抽出によりサンプリングを行なっている。[2]の提案手法は、Adaboostと言う、ブートストラップ法に似た手法が提案され、前回未学習な事例を重み付けして学習を行なう。

ブートストラップによりクラス分けを判定すると、複数の学習結果の多数決により結論の判定を行なうため、どのような基準で、その結論が導かれたかが分らない。機械学習における学習アルゴリズムの長所は、クラス分けの判定が事例を表現する属性値で説明可能な点にあるのに、ブートストラップによりクラス分けを判定するとその長所がなくなり、実問題への応用上問題がある。

本稿では、以下、ブートストラップによるクラス分け関数を決定木で学習するアルゴリズムを提案する。

## 3 提案手法

ここでは、簡単のために、多数決に利用されるクラス分けルールを決定木に限定して話しを進めるが、条件部が pure CNF に変換できる if-then ルールであれば、アルゴリズムを容易に拡張することができる。pure CNF とは、

$$(A_{i_1} = V_{j_{1,1}}) \wedge (A_{i_2} = V_{j_{2,2}}) \wedge (A_{i_3} = V_{j_{3,3}})$$

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。

提案手法の概要は以下の通り：

- (1) 多数決に利用する決定木を学習する。  
ここでは、N 個準備したとする。
- (2) 各決定木 ( $T_i, i = 1, \dots, N$ ) について、ルートから葉に至る各パスを条件部が pure CNF である if-then ルールに表現する。  
ここで、決定木  $T_i$  から生成された if-then ルール

\*Learning Decision Trees using the Bootstrap

Yasuhiro Akiba †, Hussein Almuallim ‡, Akio Yokoo †, Shigeo Kaneda §

†)NTT Communication Science Laboratories, 1-1, Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239, Japan

‡) the Dept of Information and Computer Science, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia

§) Graduate School of Policy and Management, Doshisha University, Karasuma-Higashi-Iru, Imadegawa-Toori, Kamigyo-ku, Kyoto-shi, 602-80, Japan

を  $R_{ij}$  ( $j = 1, \dots, N_i$ ,  $N_i$  は、生成された if-then ルールの数。) と表記する。

- (3)  $R_{ij}$  ( $i = 1, \dots, N, j = 1, \dots, N_i$ ) 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**  $T_i, i = 1, \dots, N$  の内、半数未満の  $T_i$  にしか、現れない属性は、irrelevant 属性として取り扱う。

**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されるが、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される。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** 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、または、 $2N$  個未満になったら、それ以上分割を行わない。

このようにして得られた決定木は、ブートストラップによるクラス分け関数を近似する。手続き(1)で、Bagging 流に決定木を複数準備すれば、Bagging 流のクラス分けルールが学習され、Adaboost 流に決定木を複数準備すれば、Adaboost 流のクラス分けルールが学習される。

#### 4 実験的比較

本手法で得られる決定木のエラー率と木の大きさが、多数決を構成する木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 inputs する決定木は、Bagging 流の方法により生成した。

実験に利用した事例は、カルフォルニア大学アーヴァイン校で提供されている、データベースからのものであり、各データの事例数、属性数、クラス数、平均属性値数は、表1が示す通りである。

表1の各データにつき、多数決を構成する木の数をいろいろ変えて得られる、本手法より生成される決定木の

表1: 提案手法を評価に利用したデータの情報

データ	事例数	属性数	クラス数	平均属性値数
monk1	556	6	2	2.8
soybean	683	35	19	2.9

表2: 提案手法で獲得された決定木の未知事例に対するエラー率(%)と木の大きさ

多数決数	枝刈なし		枝刈あり	
	エラー率	ノード数	エラー率	ノード数
monk1				
3	11.5	137.8	9.5	41.1
5	7.0	252.0	5.9	49.5
7	2.3	258.9	2.0	65.3
9	2.9	254.5	2.0	60.8
soybean				
3	14.9	70572.4	12.9	20259.9
5	13.9	526019.5	12.1	130182.7
7	11.5	1159342.6	9.8	273871.9
9	11.7	1465055.0	8.8	235199.0

エラー率と木の大きさは、表2に示す通りで、10-fold cross validation による結果である。

学習の結果得られる決定木は、かなり大きい。しかし、木の深さは、大きく見積もっても、データの順に2.5, 5.6と、事例を表現する属性数に比べ、かなり小さいので、「テスト事例のクラス分けの判定を事例を表現する属性値で説明を可能にする」という当初の目標は十分果たすことができた。

#### 5 おわりに

本稿では、決定木学習の安定化問題を取り上げた。学習結果が可視化される、決定木学習アルゴリズムの安定化法を提案した。具体的には、ブートストラップによるクラス分け関数を決定木表現するというアプローチを取っている。また、人工データでその効果を実験的評価したところ、学習の結果得られる決定木は大きい、その深さは比較的浅いので、従来法の欠点のある程度解決できた。今後は、正解率を保ったままより小さい木が生成されるようアルゴリズムを改良する予定である。

#### 参考文献

- [1] Breiman, L. "Bagging Predictors", *Machine Learning*, Vol. 24, pp.123-140 (1996).
- [2] Freund, Y. and Schapire R.,E., "A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting", *Proc. EuroCOLT'95*, pp. 23-37 (1995).
- [3] Quinlan, J., R. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann (1993).