重み更新規則の修正による Boosting アルゴリズムの改善について

中村雅之† 野宮浩揮† 上原邦昭†

AdaBoost は，学習アルゴリズムを複数回実行し，各回で得られた仮説を統合して分類精度を改善する学習手法である．AdaBoost の欠点として，訓練事例にノイズ事例や例外的な事例（hard example）が含まれるとき，性能が極端に悪化することがあげられる．この原因は，AdaBoost が hard example に対して極端に高い重みを設定してしまうことにある．本研究では，上記の欠点を解決するために，AdaBoost の重み更新規則にしきい値を導入し，極端に重みが高くならないようにしている．具体的には，AdaBoost と提案手法の理論的な誤差上限を比較しながら，つねに提案手法の誤差上限が AdaBoost の誤差上限よりも小さくなるようにしきい値を設定している．このため，提案手法は AdaBoost を上回る分類精度を実現している．

Improvement of Boosting Algorithm by Modifying Weighting Rule

Masayuki Nakamura,† Hiroki Nomiya† and Kuniaki Uehara†

AdaBoost is a method for improving the classification accuracy of a given learning algorithm by combining hypotheses created by the learning alogorithm. One of the drawbacks of AdaBoost is that AdaBoost worsens its performance when training examples include noisy examples or exceptional examples, which are called hard examples. The phenomenon causes that AdaBoost assigns too high weights to hard examples. In this research, we will introduce the thresholds into the weighting rule of AdaBoost in order to prevent weights from increasing too much. During learning process, we compare the upper bound of the classification error of our method with that of AdaBoost, and we set the thresholds so that the upper bound of our method can be superior to that of AdaBoost. Our method shows better performance than AdaBoost.

1. はじめに

機械学習の分野では，大規模な学習モデルを構成する際の計算量の増大と汎化能力の低下が問題となっている．この問題を回避する手法の 1 つとして，複数の仮説を統合し，単一の最終仮説として出力する Boosting アルゴリズムが提案されている．Boosting は，学習アルゴリズムを複数回実行し，各回で得られた仮説を統合して分類精度を改善する学習手法である．Boosting の代表的なアルゴリズムとして，AdaBoost [1)~3)] があげられる．AdaBoost は，与えられた学習アルゴリズムを用いて，1 回のラウンドで 1 個の仮説を生成する．生成された仮説は訓練事例によってテストされ，誤分類された訓練事例には高い重みが与えられる．次回のラウンドでは，重みを確率分布として訓練事例のサンプリングを行い，得られた部分集合を新たな訓練事例として用いる．この操作によって，各ラウンドで，それぞれ性質の異なる仮説が得られる．最終的に，これらの仮説を 1 つの仮説（最終仮説）に統合して，高い分類精度を実現している．

一方，訓練事例には，ノイズ事例や例外的な事例が含まれていることがある．このような事例を hard example と呼ぶ．AdaBoost では，重みの分布によって hard example ばかりがサンプリングされ，集中して学習されるため，hard example に特化した仮説，すなわち hard example は正しく分類できるが，未知の事例を誤分類してしまうような，特殊化された仮説が生成されることがある．この現象は「過学習」と呼ばれ，AdaBoost の欠点の 1 つとして報告されている [4)]．

過学習を防ぐためには，何らかの方法で，訓練事例を hard example とそうでない事例に区別し，生成される仮説から hard example の影響を排除しなければならない．このような目的を達成する方法として，し

† 神戸大学大学院自然科学研究科
Graduate School of Science and Technology, Kobe University

きい値の導入があげられる．AdaBoostでは，誤分類された事例の重みが高くなるように設定されるため，hard exampleに対する重みは必然的に高くなる傾向がある．このため，しきい値を超えた重みが与えられる事例はhard exampleと見なして，他の事例と区別すれば，仮説に与える影響を小さくできると考えられる．

しきい値を設定する際には，すべての可能な値を考慮しなければならない．しかし，各ラウンドごとに，とりうるしきい値をすべて考慮すると，ラウンド数の増加にともなって，組合せは指数関数的に増加し，計算時間が大きくなる．このため，すべての組合せを調べることはせずに，探索範囲を効果的に絞り込みつつ，しかも従来のAdaBoostを上回る性能が得られるような手法を開発しなければならない．

上記の考えに従って，新たなBoostingアルゴリズムNadaBoostを提案する．NadaBoostでは，Schapire and Singer [2]によって証明された，AdaBoostの誤差上限よりも低い誤差上限を実現するためのしきい値を設定する手法を導入している．具体的には，AdaBoostの性能を評価するためにコスト関数を導入し，各ラウンドごとに，しきい値候補の中からコスト関数の値が最小となるものを最適なしきい値として設定している．また，計算時間の増加を防ぐために，ビームサーチ[5]という探索手法を採用している．

2. AdaBoostの問題点

AdaBoostは，Freundら[1]によって提案されたBoostingアルゴリズムである．また，AdaBoostの一般化されたバージョンがSchapireら[2]によって提案されている．AdaBoostは，入力として m 個の訓練事例 $(x_1, y_1), \ldots, (x_m, y_m)$ を受け取る．ここで，x_i は任意の事例空間 X の要素である．また，y_i は任意のラベル空間 Y の要素である．AdaBoostは，与えられた学習アルゴリズムを実行して，1回のラウンドで1つの仮説を生成する．この学習アルゴリズムをWeakLearnと呼ぶ．ラウンドの繰返し回数 T は前もって決定される．また，それぞれの訓練事例に対して，重要度を反映した重みが設定されており，t 番目のラウンドにおける事例 i の重みは $D_t(i)$ と表記される．重みの集合 D_t は正規化定数 Z_t によって正規化され，$D_t(i) \in [0, 1]$ となる．

t 番目のラウンドにおいて，AdaBoostは正規化された重みを確率分布として用いて，訓練事例集合からサンプリングし，訓練事例の部分集合を生成する．WeakLearnはこの部分集合を用いて学習を行い，仮説 $h_t : X \to \Re$ を生成する．この仮説を「弱仮説」と呼ぶ．初期値として，重みはすべて等しく設定される．すなわち，すべての事例の重みは $1/m$ となる．また，t 番目のラウンドにおけるAdaBoostの重み更新規則は以下の式で与えられる．

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}.$$

各ラウンドで，正しく分類された事例の重みは減らされ，誤分類された事例の重みは増やされる．このため，WeakLearnはより難しい事例に集中して学習するようになる．また，弱仮説 h_t を得た後，AdaBoostはパラメータ α_t を設定する．Schapireらによる AdaBoostでは，α_t は以下の式に従って得られる．

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+r_t}{1-r_t}\right).$$

ただし，

$$r_t = \sum_i D_t(i) y_i h_t(x_i)$$

である．α_t は，直感的に，弱仮説 h_t の重要度を表している．すべてのラウンドが終了した後，得られたすべての仮説を結合して最終仮説 H が得られる．H は，α_t を h_t の重みとして用いた重み付き多数決となる．

hard exampleは誤分類されやすいため，AdaBoostでは，hard exampleに比較的高い重みが与えられる傾向がある．これにより，hard exampleばかりがサンプリングされ，集中して学習されることによって，hard exampleに特化した仮説が生成されることがある．この現象は「過学習」と呼ばれ，AdaBoostの欠点の1つとして報告されている．

3. 提案手法

3.1 重み更新規則の改良

本論文では，hard exampleとそうでない事例を区別するための判断基準として，事例に与えられる重みを用いている．すなわち，「すでに十分高い重みが与えられている事例はhard exampleである可能性が高い」と仮定して，「さらに重みが増やされてもよいかどうか」を判定するために，しきい値 $HighWeight$ を導入している．具体的には，(a) 訓練事例が正しく分類されたときは，従来の重み更新規則と同様の方法で重みが減らされる，(b) 事例が誤分類されても，重みがしきい値以上であれば，重みは減らされる，(c) 事例が誤分類され，かつ重みがしきい値を下回るときのみ，重みは増やされる．この結果，重み更新規則は以下のように修正される．ただし，$HighWeight_t$ は t 番目のラウンドでのしきい値を表している．

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t}E_t(i)$$
$$= \frac{D_t(i)}{Z_t}\exp\{-\alpha_t y_i h_t(x_i) \cdot I(y_i h_t(x_i), D_t(i))\}. \quad (1)$$

ここで，

(a) $h_t(x_i)y_i \geq 0$ のとき

$$E_t(i) = e^{-\alpha_t y_i h_t(x_i)}$$

(b) $h_t(x_i)y_i < 0 \ \wedge \ D_t(i) \leq HighWeight_t$ のとき

$$E_t(i) = e^{-\alpha_t y_i h_t(x_i)}$$

(c) $h_t(x_i)y_i < 0 \ \wedge \ D_t(i) > HighWeight_t$ のとき

$$E_t(i) = e^{\alpha_t y_i h_t(x_i)}.$$

また，

$$I(y_i h_t(x_i), D_t(i)) = \begin{cases} -1 : h_t(x_i)y_i < 0 \wedge D_t(i) > HighWeight_t \\ +1 : \text{otherwise.} \end{cases}$$

3.2 分類エラーの評価

一般的に，学習アルゴリズムの性能を測る指標として，数学的に証明された分類エラー率の上限値が用いられる．学習における分類エラーには，訓練事例に対する分類エラー（training error）と，テスト事例に対する分類エラー（generalization error）がある．このうち，修正された重み更新規則による training error の上限値の定式化を以下に示す．なお，generalization error については 6 章で再検討する．

定理 1 ラウンド数を T とすると，修正された重み更新規則による training error の上限値は以下の式で与えられる．

$$\prod_{t=1}^{T} Z_t \cdot \left\{ \sum_{i=1}^{m} D_{T+1}(i) \cdot \exp\left(-2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i) \right) \right\}.$$

証明：簡単のために，$I(y_i h_t(x_i), D_t(i))$ を $I_{t,i}$ と略記する．式 (1) より，

$$D_{T+1}(i) = \frac{\exp(-\sum_t \alpha_t y_i h_t(x_i) I_{t,i})}{m \prod_t Z_t}. \quad (2)$$

また，$I_{t,i} = -1$ ならば $y_i h_t(x_i) < 0$ である．したがって，

$$D_{T+1}(i) = \frac{1}{m \prod_t Z_t} \left\{ \exp\left(-\sum_t \alpha_t y_i h_t(x_i) + 2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i) \right) \right\}$$

となる．さらに，$f(x) = \sum_t \alpha_t h_t(x)$ として，

$$D_{T+1}(i) = \frac{1}{m \prod_t Z_t} \{\exp(-y_i f(x_i)) \cdot \exp\left(2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i) \right) \} \quad (3)$$

を得る．

ここで，$H(x_i)$ を事例 x_i に対する最終仮説 H の出力とし，π が真のとき $[\pi] = 1$，π が偽のとき $[\pi] = 0$ とすると，以下の式が得られる．

$$H(x_i) \neq y_i \Rightarrow y_i f(x_i) \leq 0$$
$$\Rightarrow \exp(-y_i f(x_i)) \geq 1$$
$$\Rightarrow [H(x_i) \neq y_i] \leq \exp(-y_i f(x_i))$$
$$H(x_i) = y_i \Rightarrow y_i f(x_i) \geq 0$$
$$\Rightarrow \exp(-y_i f(x_i)) \geq 0$$
$$\Rightarrow [H(x_i) \neq y_i] \leq \exp(-y_i f(x_i)).$$

したがって，つねに

$$[H(x_i) \neq y_i] \leq \exp(-y_i f(x_i)) \quad (4)$$

である．最後に，式 (3)，(4) より，training error $error(H(x))$ について，以下の関係が成り立つ．

$$error(H(x)) = \frac{1}{m}\sum_{i=1}^{m}[H(x_i) \neq y_i]$$
$$\leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i))$$
$$= \frac{1}{m}\sum_{i=1}^{m} D_{T+1}(i) \cdot m \prod_t Z_t \cdot \exp\left(-2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i) \right)$$
$$= \prod_t Z_t \cdot \left\{ \sum_{i=1}^{m} D_{T+1}(i) \cdot \exp\left(-2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i) \right) \right\}. \quad (5)$$

□

式 (5) において，パラメータ α_t の設定方法は確立されていない．すなわち，α_t は各ラウンドごとに式 (5) の右辺（training error の上限値）を最小にするように設定されるべきである．しかし，α_t が各ラウンドごとに設定されなければならないのに対して，式 (5) の右辺第

2 項 $\sum_{i=1}^{m} D_{T+1}(i) \cdot \exp(-2\sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i))$ は，すべてのラウンドが終了するまで定まらない．このため，途中のラウンドで α_t を決定することはできない．このようなジレンマを解消するために，式 (5) の右辺すべてを最小化することはせず，各ラウンドで最小化が可能な第 1 項，すなわち $\prod_t Z_t$ を最小化するように α_t を設定している．第 1 項のみを最小化することでも，α_t としきい値を設定すれば，最終的に AdaBoost より良い学習結果を得ることができる．

3.3 α_t の設定

$\prod_t Z_t$ を最小化するために，各ラウンドで Z_t を最小化するように α_t を設定する．t 番目のラウンドでは，Z_t は以下のように与えられる．

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i) I_{t,i}).$$

以降では，各変数の記述から添え字 t を省略する．また，$u_i = y_i h_t(x_i)$，$I_i = I(y_i h(x_i), D(i))$ とする．ここで，$h \in [-1, +1]$ であるので，$u_i \in [-1, +1]$ である．したがって，

$$\begin{aligned} Z &= \sum_i D(i) \exp(-\alpha u_i I_i) \\ &= \sum_{i:I_i=1} D(i) e^{-\alpha u_i} + \sum_{i:I_i=-1} D(i) e^{\alpha u_i} \\ &\le \sum_{i:I_i=1} D(i) \left(\frac{1+u_i}{2} e^{-\alpha} + \frac{1-u_i}{2} e^{\alpha} \right) \\ &\quad + \sum_{i:I_i=-1} D(i) \left(\frac{1-u_i}{2} e^{-\alpha} + \frac{1+u_i}{2} e^{\alpha} \right) \end{aligned} \tag{6}$$

となる．方程式 $\partial Z / \partial \alpha = 0$ を解くことにより，以下のようにパラメータ α を設定すれば，Z は最小となる．

$$\alpha = \frac{1}{2} \ln\left(\frac{1+r'}{1-r'} \right). \tag{7}$$

ただし，

$$r' = \sum_{i=1}^{m} D(i) u_i I_i$$

である．式 (6) に式 (7) を代入することにより，以下を得る．

$$Z \le \sqrt{1 - r'^2}.$$

以上のことから得られる training error の上限値を定理 2 に示す．

定理 2 $h_t \in [-1, +1]$ とする．また，α_t を

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+r'_t}{1-r'_t} \right)$$

のように設定する．ただし r'_t は

$$r'_t = \sum_{i=1}^{m} D_t(i) y_i h_t(x_i) I_{t,i}$$

である．このとき，最終仮説 H の training error の上限値は以下のようになる．

$$\begin{aligned} error(H(x)) \le & \left(\prod_{t=1}^{T} \sqrt{1 - r'^2_t} \right) \\ & \cdot \left\{ \sum_{i=1}^{m} D_{T+1}(i) \right. \\ & \left. \cdot \exp\left(-2 \sum_{t:I_{t,i}=-1} \alpha_t y_i h_t(x_i) \right) \right\}. \end{aligned} \tag{8}$$

以降では，この上限値を *errorbound* と表記する．

しきい値 $HighWeight_t$ が低くなるにつれて，式 (8) の右辺第 1 項は小さくなる．逆に，式 (8) の右辺第 2 項は指数関数的に増加する．このため，低すぎるしきい値が選ばれると，式 (8) の右辺第 2 項の値の増加によって，右辺第 1 項の減少が打ち消されてしまうことになる．したがって，しきい値を決定する際には，*errorbound* の値に注意をはらう必要がある．具体的には，各ラウンドで AdaBoost の *errorbound* を下回る *errorbound* を実現するようにしきい値を選択すれば，最悪の場合でも AdaBoost と等しい学習結果を得ることができる．

3.4 しきい値の設定

AdaBoost の *errorbound* は，Schapire ら[2]によって以下のように与えられている．

定理 3 （Schapire ら）$h_t \in [-1, +1]$ とする．α_t を

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+r_t}{1-r_t} \right)$$

のように設定する．ただし r_t は

$$r_t = \sum_i D_t(i) y_i h_t(x_i)$$

である．このとき，最終仮説 H の *errorbound* は

$$\prod_{t=1}^{T} \sqrt{1 - r_t^2}$$

となる．以降，この上限値を *errorbound_ada* と呼ぶ．

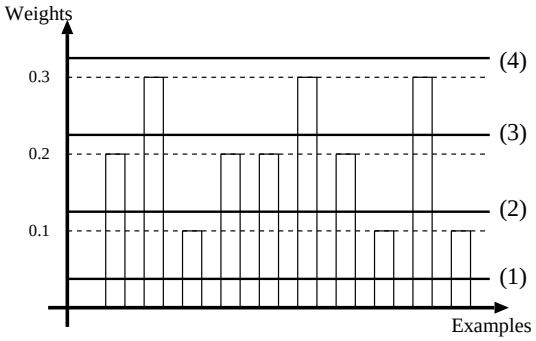


図 1 しきい値候補の例
Fig. 1 Example of the candidates of thresholds.

定理 2 と 3 から，各ラウンドで，$errorbound\_ada$ より優れた $errorbound$ を実現できるように，すなわち，

$$errorbound \leq errorbound\_ada \quad (9)$$

を満たすようにしきい値を設定すれば，提案手法による学習結果は AdaBoost よりも優れているという，理論的な根拠が得られることになる．

一方，重みは連続値であるため，しきい値となりうる値も連続値である．しかし，しきい値の性質上，とりうる範囲はある程度制限することができる．たとえば，t 番目のラウンドで $D_t(i) \in \{0.1, 0.2, 0.3\}$ であるとする．この場合，とりうるしきい値は「(1) すべての事例が hard example である」,「(2) 重みが 0.2, 0.3 である事例が hard example である」,「(3) 重みが 0.3 である事例が hard example である」,「(4) すべての事例が hard example でない」の 4 通りが考えられる．これを図示すると**図 1** のようになる．

図 1 のうち，すべてが hard example のみで構成される訓練事例はありえないため,「(1) すべての事例が hard example である」というしきい値は除外できる．したがって，上記の例では，3 種類のしきい値候補が生成される．具体的な数値としては，各ラウンドでとりうる重みをしきい値候補として採用している．つまり，$D_t(i) \in \{0.1, 0.2, 0.3\}$ のとき，0.1，0.2，0.3 がしきい値候補となる．この結果，しきい値として 0.1 が選ばれると，0.1 を超えるすべての重みは，重み更新の際に減らされる．

しきい値は重みに影響を与え，重みは弱仮説，α_t，第 t ラウンドでの $errorbound$ の値を表す見積りエラー（$estimated\_error_t$）に影響を与える．ここで，各ラウンドごとの各事例の重み $D_t[i]$，弱仮説，α_t，$estimated\_error_t$ の値の 4 つ組を「状態」と定義し，t 番目のラウンドの j 番目のしきい値候補による状態を $S_t[j]$ と表記する．各ラウンドにおける状態はしきい値に依存するので，状態を節点，とりうるすべてのしきい値候補を枝と見なせば，提案手法の状態遷移を木構造として表すことができる．

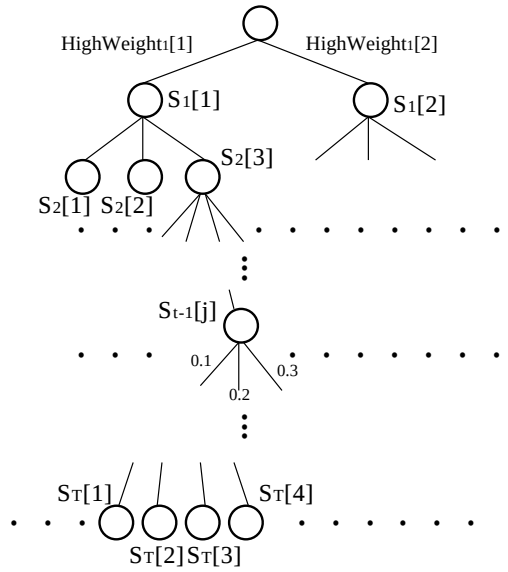


図 2 状態の木構造
Fig. 2 Tree structure of states.

状態の木構造の例を**図 2** に示す．図 2 における状態 $S_{t-1}[j]$ 以下の枝は，上記の例で用いたしきい値候補の例 0.1，0.2，0.3 に対応している．この木構造から式 (9) を満たす最終状態を探索すればよいことになる．

しかし，木構造全体をくまなく探索することは，非常に大きな計算時間を要する．このため「ビームサーチ」[5]という探索方法を用いている．ビームサーチは最良優先探索の一種で，コスト関数の値が最も低いノードを優先して探索を進める．ビームサーチでは，展開できるノード数を制限しており，このノード数を $beam\_width$ と呼ぶ．$beam\_width$ により，コスト関数の値が低いノードが優先され，値の高いノードは探索対象から除外されるため，探索対象を絞り込むことができる．

t 番目のラウンドにおける見積りエラー $estimated\_error_t$ は，学習が t 番目のラウンドで終了したと仮定したときの $errorbound$ の値を表している．

入力：$(x_1, y_1), \ldots, (x_m, y_m); x_i \in X, y_i \in \{-1, +1\}$
初期値：$D_1(i) = 1/m, t = 0, end = 0$
ラウンド数：T

入力データに対して AdaBoost を実行し，AdaBoost の training error の上限値 $errorbound\_ada$ を得る．
While $end \neq 1$:

1. D_t に従って弱仮説 $h_t : X \to [-1, +1]$ を生成する．
2. しきい値の候補 $HighWeight_t[j]$ を生成する．
3. すべての $HighWeight_t[j]$ について
   (1). α_t，D_{t+1} を決定する．
   (2). 見積りエラー $estimated\_error_t$ を計算する．
   (3). このときの状態を $S_t[j]$ とする．
4. すべての $S_t[j]$ を S_list に加える．このとき，S_list 内の状態数が $beam\_width$ を超えている場合は，$estimated\_error$ の大きいものから順に，S_list 内の状態数が $beam\_width$ となるように状態を削除する．
5. If S_list が空：$end = 1$ とする．
   else:最も小さい $estimated\_error$ を持つ $S_k[j]$ を選び，
   (1). S_list からその $S_k[j]$ を取り出す．
   (2). If $k = T$:
   If $errorbound \leq errorbound\_ada$: $S_k[j]$ を $Goal\_list$ に加え Step 5 へ．
   else: $t = k + 1$ とする．

$Goal\_list$ が空でない場合は，$Goal\_list$ から最小の $errorbound$ を持つ $S_k[j]$ を選び，出力として最終仮説

$$H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

を得る．$Goal\_list$ が空である場合は，$error\_bound$ の値を得る際に実行した AdaBoost の結果を出力とする．

図 3 NadaBoost アルゴリズム
Fig. 3 NadaBoost algorithm.

$$estimated\_error_t = \left(\prod_{k=1}^{t} \sqrt{1 - r_k'^2}\right) \cdot \left\{\sum_{i=1}^{m} D_{t+1}(i) \cdot \exp\left(-2 \sum_{k:I_{k,i}=-1} \alpha_k y_i h_k(x_i)\right)\right\}. \quad (10)$$

ただし，

$$r_k' = \sum_{i=1}^{m} D_k(i) y_i h_k(x_i) I_{k,i}.$$

一般に，ラウンド数が増加するにつれて training error は単調に減少するため，$estimated\_error$ も同様に単調に減少する．したがって，あるノードが高い $estimated\_error$ を持つことは，そのノードに至る経路上のノードで，性能の悪い弱仮説が生成されていることを示している．このような状態から学習を進めたとしても，すでに生成されている性能の悪い弱仮説が最終仮説に統合されてしまうため，良い結果が得られる可能性は低くなる．したがって，見積りエラーをコスト関数として除外すれば，低い $estimated\_error$ を持つ最終状態を効率的に探索することができる．

3.5 NadaBoost

AdaBoost アルゴリズムの重み更新規則を修正し，各ラウンドでつねに AdaBoost より優れた $errorbound$ を実現できるようにしきい値を設定したアルゴリズム NadaBoost を提案する．NadaBoost は，ビームサーチを導入して探索空間を制限して，効率的な探索を行っている．NadaBoost アルゴリズムを**図 3** に示す．*Step 1* では，WeakLearn は弱仮説 h_t を生成する．*Step 2* では，とりうるしきい値の候補の集合を決定する．なお，t 番目のラウンドにおける j 番目のしきい値候補は $HighWeight_t[j]$ と表記している．*Step 3* では，すべてのしきい値候補について，D_{t+1} と α_t を計算する．さらに，見積りエラー $estimated\_error_t$ を得る．なお，$HighWeight_t[j]$ によって決まる状態を $S_t[j]$ と表記する．*Step 4* では，すべての状態を S_list に加える．このとき，S_list 内の状態数が $beam\_width$ を超えている場合は，$estimated\_error$ の大きいものから順に，S_list 内の状態数が $beam\_width$ と等しくなるように状態を削除する．*Step 5* では，S_list に含

まれているすべての状態の中から，最も低い見積りエラーを持つ状態を取り出す．取り出された状態 $S_k[j]$ が最終状態でなければ，すなわち $k \neq T$ であれば，引き続いて $(k+1)$ 番目のラウンドの操作を実行する．また，$S_k[j]$ が最終状態であれば，すなわち $k = T$ であれば，$S_k[j]$ によって $errorbound$ の値を計算する．得られた $errorbound$ の値が

$$errorbound \leq errorbound\_ada$$

を満たせば，状態 $S_k[j]$ を $Goal\_list$ に加える．この一連の操作を S_list が空になるまで繰り返す．最後に，$Goal\_list$ の中から最も低い $errorbound$ を持つ状態を選び出し，最終仮説 H を出力する．

NadaBoost の各ラウンドにおいて，つねにすべての事例の中で重み $D_t[i]$ が最大となるものを $HighWeight$ として選択すると，AdaBoost と等価なアルゴリズムとなる．したがって，全探索を行う場合は，NadaBoost は必ず $errorbound \leq errorbound\_ada$ を満たす節点を見つけることができるが，AdaBoost と等価となる経路が探索されなかった場合，条件を満たす節点は見つけられず，$Goal\_list$ は空となりうる．したがって，$Goal\_list$ が空となった場合には，$errorbound\_ada$ を求める際に得られた，AdaBoost による学習結果を出力している．このため，最悪の場合でも，AdaBoost と等しい結果が得られることが保証されている．

図 4 は，hard example を含む人工的に作成したデータセットの図である．生成される仮説に対する hard example の影響を AdaBoost と NadaBoost の間で比較するために，このデータセットを訓練集合として，AdaBoost，NadaBoost により 100 ラウンドの学習を行った．生成された最終仮説を，**図 5** に示す．使用したデータセットは，クラス数，属性数がそれぞれ 2 であり，データ数は 200 である．

図 5 において，斜線部の部分が hard example によって過学習を起こしていると考えられる領域である．正事例，負事例ともに，NadaBoost における過学習の領域は AdaBoost に比べてかなり小さくなっている．このことから，NadaBoost が hard example に対する過学習を軽減していることが分かる．

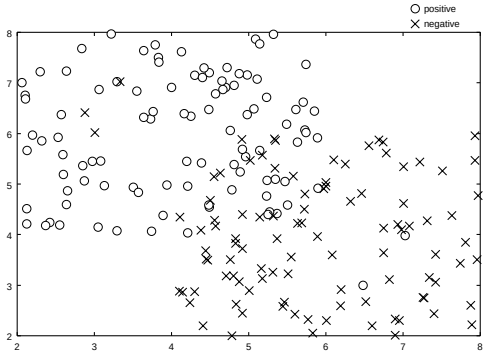
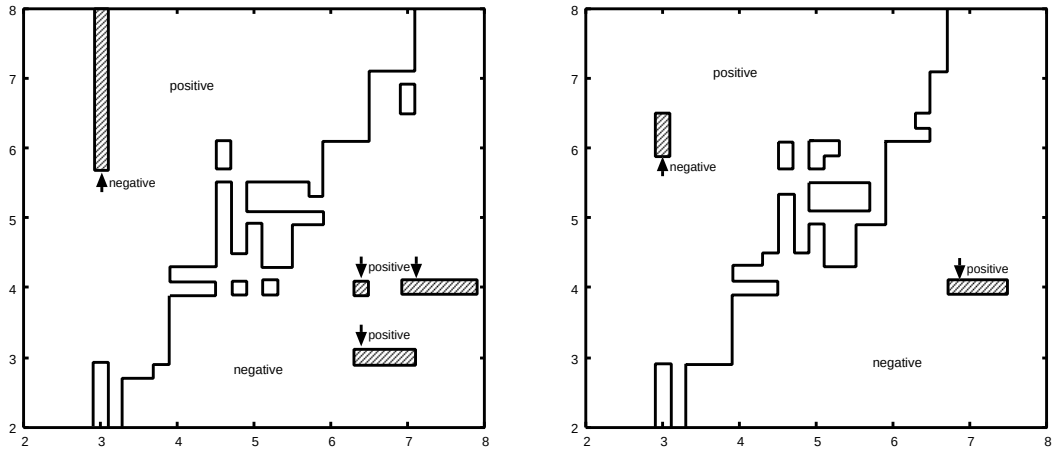


図 4 訓練事例
Fig. 4 The training examples.

4. 実　　験

本章では，AdaBoost と NadaBoost の比較実験の結果を示す．本実験では，UCI Machine Learning Repository [6] から取得した 10 種類のデータセットを用いている．WeakLearn として C4.5 [7] を用いている．



(a) AdaBoost　　(b) NadaBoost

図 5 100 ラウンド実行後の最終仮説
Fig. 5 The final hypotheses after 100 rounds.

表 1 実験結果
Table 1 Results of experiments.

Data Set Name	error(%)									
	NadaBoost								AdaBoost	C4.5
	w=10		w=30		w=50		w=100			
		t		t		t		t		
breast-w	3.58	-	3.29	*	3.29	*	3.15	*	3.86	5.29
crx	13.48	-	13.48	-	13.33	-	13.48	-	13.91	15.94
german	25.20	-	25.00	-	23.90	*	24.80	-	25.40	29.10
heart-c	21.73	*	21.46	*	20.84	*	20.77	*	22.77	27.31
hepatitis	15.16	*	15.05	*	13.13	*	12.31	*	17.20	20.05
ionosphere	5.67	*	6.27	*	5.40	*	5.97	*	6.56	9.69
pima-indians	26.68	-	26.44	-	25.64	*	25.66	*	27.34	26.91
sonar	19.19	-	19.29	*	19.21	*	18.17	*	21.21	31.81
tic-tac-toe	0.84	*	0.84	*	0.84	*	0.63	*	1.05	13.78
vote	5.04	-	5.05	-	4.82	-	4.37	*	5.51	5.07

ラウンド数については適切な決定法はないが，Dietterich [4]は AdaBoost と他の学習アルゴリズムの比較実験でラウンド数を 100 としており，関連研究でも多く用いられていることから，本実験でも同様にラウンド数を 100 としている．*beam_width* は，計算時間の妥当性および *beam_width* の違いによる提案手法の安定性を示すことを考慮して，10，30，50，100 としている．これらの条件のもとで，10-fold cross-validation paired t test [8]を行い，それぞれの分類精度を比較した．この検定は，通常の cross-validation の結果に対して t 検定を行い，結果に確率的な正当性を与えるものである．t 検定の有意水準は 5%としている．さらに，比較のために C4.5 を単体で実行し，分類精度を求めている．実験結果を**表 1** に示す．なお，NadaBoost 欄における w は *beam_width* の値を表しており，t 欄における $*$ は t 検定により実験結果に有意差があると判定されたことを表している．

各データセットにおいて，すべての *beam_width* の値について，NadaBoost の分類精度は AdaBoost よりも優れているか，または同程度の値となっている．したがって，NadaBoost は AdaBoost と比較して，分類精度の点から優れた性能を示しているといえる．

beam_width 10，30，50，100 で有意差があると判定されたデータセットは，それぞれ 4 種類，6 種類，8 種類，8 種類となっている．このように，多くの場合は *beam_width* の値が増加するにつれて分類精度も上昇しているが，**crx**，**tic-tac-toe** のように，*beam_width* の値が増加しても分類精度にあまり改善が見られない場合もある．これは，データ中に hard example がないか，もしくは少なくて，AdaBoost での過学習による影響が小さいためであると考えられる．これに対し，AdaBoost が，hard example に対して過学習を起こす傾向を顕著に示すデータセットは **pima-indians** および **vote** である．**pima-indians**，**vote** では，AdaBoost の分類精度が C4.5 の分類精度よりも劣っている．それにもかかわらず，NadaBoost はいずれの *beam_width* でも C4.5 の分類精度を上回るほど改善している．これは，NadaBoost が目的どおり hard example に対する過学習を軽減することを示している．

しかしながら，hard example を含まない **tic-tac-toe** では分類精度の差が有意となっており，hard example の多い **pima-indians** および **vote** では，*beam_width* を小さく設定した場合でも有意となっていない．これは，**tic-tac-toe** ではデータセットの部分集合のとり方の違いが学習結果に影響しにくく，**pima-indians** と **vote** では学習結果に影響しやすいためであると考えられる．このため，**tic-tac-toe** では分類精度の差が小さい場合でも有意な差が生じやすく，**pima-indians** や **vote** では分類精度の差が比較的大きくなければ有意な差は生じにくい．したがって，単に hard example の量が多ければ有意な差が生じるとはいえず，データセットの性質も分類精度に影響していると考えられる．

このため，hard example を含んでいないデータセットに 10%のノイズを加え，NadaBoost が hard example の影響を軽減しているかを検証する実験を行った．実験に用いたデータセットは，**tic-tac-toe** と我々が独自に作成した **maze** の 2 つである．いずれのデータセットも hard example が学習結果に与える影響を確認するために使用するもので，hard example は含まれていない．なお，**tic-tac-toe** は，三目並べの可能な終盤のうち，×が勝っているものは positive，負けているものは negative をクラスとして持つデータ

表 2 ノイズを加えたデータセットに対する実験結果
Table 2 Results of experiments for noisy data sets.

Data Set Name	error(%)									
	NadaBoost								AdaBoost	C4.5
	w=10		w=30		w=50		w=100			
		t		t		t		t		
maze	24.85	*	24.43	*	24.02	*	23.98	*	25.48	26.43
tic-tac-toe	21.33	*	21.18	*	21.09	*	20.96	*	23.33	27.40

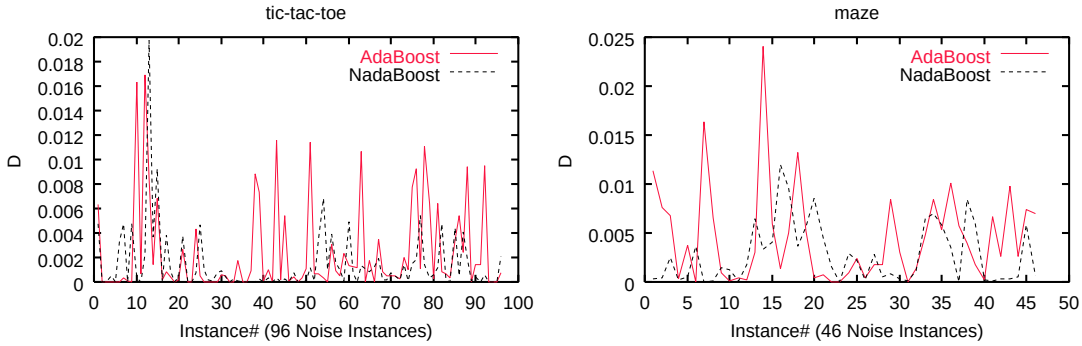


図 6 hard example の最終的な重みの分布
Fig. 6 Final weight distributions of hard examples.

セットである．**maze** は，迷路を進んだ結果，障害物のマスを通らずに出口に着ける場合は goal，そうでない場合は not_goal をクラスとして持つデータセットである．**maze** には属性の欠落している事例はない．また，事例数は 462，属性数は 11 である．なお，UCI Machine Learning Repository には hard example を含まないことが明らかなデータセットが **tic-tac-toe** 以外にほとんどないため，あえて **maze** を用意している．実験結果を**表 2** に示す．

データセットへのノイズの付加によって，AdaBoost の分類精度は大きく悪化している．逆に，NadaBoost の分類精度は AdaBoost と同様に悪化しているものの，各 *beam_width* の値については悪化の程度が小さくなっている．これより，表 1 に示した結果と同様に，NadaBoost は hard example に対する過学習を軽減していることが分かる．

図 6 に，各データセットに含まれる最終的な（100 ラウンド終了後の）重みの分布を示す．グラフの横軸は hard example を表しており，**maze** では 46 個，**tic-tac-toe** では 96 個の hard example について，最終的な重み D の値を縦軸に示している．また，**表 3** に最終的な hard example の重みの平均値を示す．

図 6 より，いずれのデータセットについても，多くの hard example の重みが AdaBoost よりも低い値となっている．また，重みの平均値についても，NadaBoost は AdaBoost よりも低くなっている．これより，NadaBoost は hard example の重みが高くなりすぎるのを抑えていることが分かる．

表 3 hard example の最終的な重みの平均値
Table 3 Mean value of final weights of hard examples.

	NadaBoost	AdaBoost
maze	0.00277611	0.00455820
tic-tac-toe	0.00144982	0.00238997

一方，AdaBoost では，WeakLearn を実行する回数はラウンド数と等しい．したがって，ラウンド数を 100 としたことから，AdaBoost が WeakLearn (C4.5) を実行する回数も 100 回となる．これに対し，NadaBoost では，状態の木構造を探索するため，WeakLearn を実行する回数は，*errorbound_ada* を得るために実行する AdaBoost のラウンド数と探索節点数の和に等しくなる．探索節点数は *beam_width* によって制限されるため，NadaBoost で WeakLearn を実行する回数は AdaBoost に比べて極端に多くなることはない．したがって，計算量にもそれほど大きな差は生じない．さらに，探索節点数が制限されても，NadaBoost では各ラウンドでつねに *estimated_error* が最小となるような状態，およびしきい値を選択している．また，最終ラウンドでは，$errorbound \leq errorbound\_ada$ を満たす状態を最終仮説として選択しているので，結果として，AdaBoost よりも精度の高い学習結果を得

表 4 実行時間
Table 4 Execution time.

Data Set Name	execution time (sec)				
	NadaBoost				AdaBoost
	w=10	w=30	w=50	w=100	
breast-w	63	119	176	338	17
crx	87	153	219	375	24
german	130	180	247	443	33
heart-c	49	106	161	346	15
hepatitis	40	96	155	336	12
ionosphere	58	110	175	342	21
pima-indians	100	156	200	400	29
sonar	53	111	172	357	20
tic-tac-toe	118	174	232	443	28
vote	52	110	165	337	14

ることができる．

表 4 に，ラウンド数 100 での NadaBoost および AdaBoost の実行時間を示す．NadaBoost において，*beam_width* の値を大きくすると，弱学習アルゴリズムを実行する回数が増加して，実行時間が長くなる．さらに，*S_list* に含まれる状態数が増えるため，*S_list* 更新の際の計算量が増大することも実行時間に影響している．しかし，*beam_width* の増加に対する実行時間の増加はほぼ線形であり，また *beam_width* は比較的小さい値であっても分類精度は改善されるので，実行時間の増加をある程度抑えることができる．

5. 関連研究との比較

AdaBoost のノイズに対する脆弱性については，さまざまな観点から研究が行われている．その中でも，特に訓練事例上のマージン[9]に基づいた研究が行われている．マージンとは，ある訓練事例に対する分類結果の信頼度を表す値である．事例 (x, y) のマージンは以下のように定義される．

$$\frac{y \sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} \alpha_t}.$$

明らかに，マージンは $[-1, +1]$ の範囲をとる．最終仮説 H が事例を正しく分類したとき，マージンの値は正となる．逆に H が誤分類したとき，マージンの値は負となる．また，マージンの値が $+1$ に近づくほど，多くの弱仮説が正しく事例を分類することを表し，-1 に近づくほど，誤分類することを表している．すなわち，マージンの絶対値は分類結果の信頼度の指標である．

AdaBoost は hard example，すなわちマージンの絶対値が小さい事例に集中して学習を行うため，すべての訓練事例のマージンを $+1$ に近づけることができる．Schapire ら[9]は，AdaBoost が訓練事例のマージンの最大化に対して効果的に働くことを示している．さらに，マージンの値の増加が generalization error の上限値の改善につながることを示している．

Schapire らの分析結果に示されているように，AdaBoost では学習が進むにつれて，すべてのマージンは徐々に増加していく．数ラウンドの実行が終了した時点で，すべてのマージンの値は正になっている．すなわち，AdaBoost は数ラウンドの繰返しを行うのみで，すべての訓練事例を正しく分類できるようになる．しかし，仮に hard example が訓練事例に含まれていても，すべてのマージンの値が正になることは，hard example も正しく分類していることを意味している．これは，AdaBoost が hard example に対して過学習を起こしていることを示している．近年，上記のような AdaBoost の特性が過学習の原因であるという考えに基づいて，いくつかの研究がなされている．

Mason ら[10]は，勾配降下法による Boosting アルゴリズムについて述べ，DOOM II というアルゴリズムを提案している．DOOM II では，マージンに関する関数がコスト関数として用いられている．Rätsch ら[11]は，AdaBoost が $g(b)$ と呼ばれるコスト関数を最小化する過程であると論じている．また，重み分布がマージンの値と $g(b)$ によっても得られることを証明している．さらに，すべての訓練事例のマージンを増やしすぎることが過学習の原因になっているとして，マージンの下限値を下げる Boosting アルゴリズムを提案している．Vapnik[12] は，マージンの最大化におけるノルムとしてユークリッド距離を用いた Support Vector Machine（SVM）アルゴリズムを提案している．SVM は，入力ベクトルの特徴空間への写像から識別関数（超平面）を生成するアルゴリズムである．

これらのマージンに基づくアルゴリズムと提案手法

表 5 実験結果
Table 5 Result of experiments.

Data Set Name	error(%)					
	NadaBoost				DOOM II	SVM
	w=10	w=30	w=50	w=100		
breast-w	3.58	3.29	3.29	3.15	3.29	3.29
crx	13.48	13.48	13.33	13.48	13.19	14.78
german	25.20	25.00	23.90	24.80	25.40	23.10
heart-c	21.73	21.46	20.84	20.77	20.77	16.50
hepatitis	15.16	15.05	13.13	12.31	15.82	14.19
ionosphere	5.67	6.27	5.40	5.97	5.98	11.97
pima-indians	26.68	26.44	25.64	25.66	27.08	22.92
sonar	19.19	20.19	19.21	18.17	20.67	19.23
tic-tac-toe	0.84	0.84	0.84	0.63	0.94	1.67
vote	5.04	5.05	4.82	4.37	5.29	5.06

を比較するために，DOOM II および SVM と比較実験を行った．用いたデータセットは，4 章の実験と同様に UCI Machine Learning Repository [6]から取得した 10 種類のデータセットおよび **maze** である．また，SVM では gaussian kernel を用いており，$C = 100$ としている．実験結果を**表 5** に示す．

DOOM II と NadaBoost を比較すると，全体的に DOOM II は NadaBoost より若干分類精度が低いか，または同程度となっている．しかし，**pima-indians** や **vote** などの比較的 hard example を多く含むデータセットについては，NadaBoost よりも分類精度が低い．これは，DOOM II では過学習を抑えるため hard example の影響を排除しているが，NadaBoost では hard example も考慮に入れた学習を行っているためだと考えられる．

SVM と NadaBoost を比較すると，データセットに依存する結果となっている．すなわち，**heart-c** および **pima-indians** では，SVM は NadaBoost よりも分類精度がかなり高くなっている．これは，各クラスの事例の分布範囲が比較的小さいため，SVM の生成する単一の仮説（判別関数）でも，データセット中の大部分の事例が分類可能なためと考えられる．hard example の点から考えると，**pima-indians** は hard example を多く含むが，**heart-c** は比較的少ない．また，SVM は hard example に影響を受けやすい学習アルゴリズムではないので，両者の精度の違いに与える影響はそれほど大きくはない．一方，**ionosphere** では，SVM は NadaBoost よりも分類精度がかなり低くなっている．これは，データセット中の各クラスの事例の分布範囲が広く，それぞれのクラスの事例の分布範囲が重なっている部分が大きいため，単一の仮説では事例を分類しきれないことが原因であると考えられる．

すでに述べたように，一般に AdaBoost はすべてのマージンを正にする．これに対し，上記の DOOM II および $g(b)$ によるアルゴリズムでは，いくつかのマージンは負の値のままであるという結果が示されている．このような現象は，各アルゴリズムがあまりにも分類が難しい事例，すなわち hard example を正しく分類することを「あきらめて」いることを表している．すなわち，hard example の分類をあきらめて，最終仮説から hard example の影響を排除し，性能の向上を実現している．

これらのアルゴリズムと比較するために，ノイズの含まれる環境における NadaBoost の振舞いを，マージンの観点から分析を行った．**tic-tac-toe**，**maze** の 2 つのデータセットについて 10%のノイズを加え，AdaBoost と NadaBoost を実行した．また，ラウンド数を 100，*beam_width* を 30 に設定した．実験によって得られたマージン分布を**図 7** に示す．また，hard example とそうでない事例それぞれについて，マージンの値の平均値を**表 6** に示す．

図 7 に関する限り，AdaBoost と NadaBoost の間に明らかな差異は見つけられない．DOOM II および $g(b)$ によるアルゴリズムでは，いくつかのマージンは負の値をとるが，NadaBoost ではすべてのマージンが正の値をとる．しかし，表 6 より，NadaBoost が hard example のマージンを効果的に減らしていることが確認できる．すでに述べたように，マージンの絶対値は分類結果の信頼度を表している．したがって，この実験結果より，NadaBoost は hard example の分類を「あきらめる」のではなく，hard example が仮説に与える影響を「弱める」ことにより，過学習の程度を軽減していることが分かる．hard example の分類を「あきらめる」ことは，hard example に対するマージンの値を負のままにすること，すなわち訓練

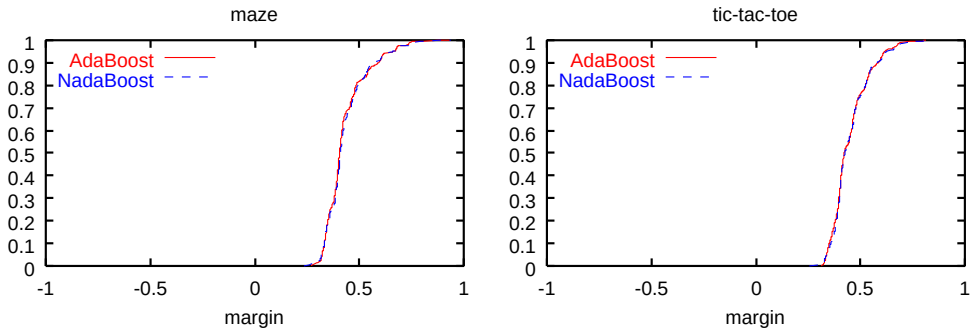


図 7 マージン分布図
Fig. 7 Margin distributions.

表 6 マージンの平均値
Table 6 Mean values of margins.

Data Set Name		Mean value Normal	Hard
maze	AdaBoost	0.450038	0.371218
	NadaBoost	0.455097	0.357785
tic-tac-toe	AdaBoost	0.464483	0.390105
	NadaBoost	0.470284	0.376698

集合中の hard example を正しく分類しようとしないことを表している．これにより，最終仮説から hard example の影響を排除し，過学習を回避することができるが，仮説生成において hard example を考慮に入れることはなくなる．

これに対し，NadaBoost では，AdaBoost と同様に，すべての事例のマージンの値は正となるが，しきい値を設けて hard example の重みが大きくなりすぎることを避けている．したがって，AdaBoost では，サンプリングの際に hard example が必然的に選択されやすくなるが，NadaBoost では hard example が選択される確率を低くしている．つまり，hard example に対して集中した学習が行われるのを避けて，仮説に与える影響を「弱めた」ことになっている．

6. おわりに

本論文では，hard example に対して過学習を起こすという AdaBoost の問題を克服するために，重み更新規則の改良を行った．また，改良された重み更新規則による training error の上限値を定式化した．さらに，新たな Boosting アルゴリズム NadaBoost を提案した．計算時間を軽減するために，NadaBoost にビームサーチという探索法を適用した．また，AdaBoost と NadaBoost を比較するための実験を行い，NadaBoost が AdaBoost よりも優れた性能を実現していることを示した．

しかしながら，NadaBoost の generalization error については実験的な結果は示したものの，理論的な保証は与えていない．Freund and Schapire [1)] は，AdaBoost の generalization error について，training error，訓練事例の数，弱仮説空間の VC 次元，ラウンド数によって解析している．Schapire ら [9)] は，マージンの増加が generalization error の減少につながることを示している．今後，NadaBoost についても，これらの観点から generalization error の解析を行う必要がある．また，3 章で述べたように，各ラウンドで AdaBoost の errorbound を下回るように，すなわち式 (9) を満たすようにしきい値を決定している．この決定方法をマージンに基づく観点によって決定し，5 章で紹介したアルゴリズムとの詳しい比較を行うことも考えている．最後に，NadaBoost は二値問題を対象としており，多クラスの分類問題に対応していない．NadaBoost を多クラスの分類問題に拡張することも今後の課題としてあげられる．

参考文献

1) Freund, Y. and Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences*, Vol.55, No.1, pp.119–139 (1997).
2) Schapire, R.E. and Singer, Y.: Improved boosting algorithms using confidence-rated predictions, *Proc. 11th Annual Conference on Computational Learning Theory*, pp.80–91 (1998).
3) Schapire, R.E.: Theoretical views of boosting, *Proc. 4th EuroCOLT'99*, pp.1–10 (1999).
4) Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization, *Machine Learning*, Vol.32, No.1, pp.1–

22 (1999).
5) Lowerre, B.T. and Reddy, R.D.: The Harpy Speech Understanding System, *Trends in Speech Recognition*, Lea, W.A. (Ed.), pp.340–360, Prentice Hall (1980).
6) Blake, C. and Merz, C.J.: UCI repository of machine learning databases, Department of Information and Computer Science, University of California at Irvine, Irvine, CA. http://www.ics.uci.edu/˜mlearn/MLRepository.html
7) Quinlan, J.R.: Bagging, boosting and C4.5, *Proc. 13th AAAI*, pp.725–730 (1996).
8) Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms, *Neural Computation*, Vol.10, No.7, pp.1895–1923 (1998).
9) Schapire, R.E., Freund, Y., Bartlett, P. and Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods, *Proc. 14th International Conference on Machine Learning*, pp.322–330 (1997).
10) Mason, L., Baxter, J., Bartlett, P. and Frean, M.: Boosting algorithms as gradient descent, *Advances in Neural Information Processing Systems 12*, pp.512–518, MIT Press (2000).
11) Rätsch, G., Onoda, T. and Müller, K.R.: Soft margins for AdaBoost, *Machine Learning*, Vol.42, No.3, pp.287–320 (2000).
12) Vapnik, V.: Three Remarks on the Support Vector Method of Function Estimation, *Advances in Kernel Methods: Support Vector Learning*, pp.25–41, The MIT Press (1998).

(平成 15 年 5 月 26 日受付)
(平成 16 年 1 月 6 日採録)

中村 雅之

1977 年生．2000 年神戸大学工学部情報知能工学科卒業．2002 年同大学大学院自然科学研究科修士課程修了．機械学習の研究に従事．同年，（株）毎日放送入社．

野宮 浩揮

1979 年生．2002 年神戸大学工学部情報知能工学科卒業．現在，同大学大学院自然科学研究科修士課程在学．機械学習の研究に従事．

上原 邦昭（正会員）

1954 年生．1978 年大阪大学基礎工学部情報工学科卒業．1983 年同大学大学院博士後期課程単位取得退学．同産業科学研究所助手，講師，神戸大学工学部情報知能工学科助教授，同都市安全研究センター教授を経て，同大学院自然科学研究科教授．1989 年より 1990 年まで Oregon State University，Visiting Assistant Professor．工学博士．人工知能，特に機械学習，マルチメディア処理の研究に従事．1990 年度人工知能学会研究奨励賞，2001 年度電子情報通信学会オフィスシステム研究賞．人工知能学会，電子情報通信学会，計量国語学会，日本ソフトウェア科学会，AAAI 会各会員．