社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－CVIM－159 (32)
2007／5／15

AdaBoostの基本原理と顔検出への応用

CVIM研究会 チュートリアルシリーズ

三田 雄志†

† 株式会社東芝 研究開発センター マルチメディアラボラトリー
〒221–8582 川崎市幸区小向東芝町1
E-mail: †takeshi.mita@toshiba.co.jp

あらまし AdaBoostはパターン識別のための学習アルゴリズムである．実装が容易で高い識別性能が得られる利点があるため，近年盛んに応用されている．本チュートリアルでは，AdaBoostの基本原理と実装法について分かりやすく紹介する．実装の際に知っておくと役立つ基本的な性質についても解説する．さらに，典型的な応用事例として顔検出をとりあげる．

キーワード AdaBoost，学習，パターン識別，顔検出

AdaBoost and its Application to Face Detection

CVIM Tutorial Series

Takeshi MITA†

† Multimedia Laboratory, Corporate R&D Center, Toshiba Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki, 221–8582 Japan
E-mail: †takeshi.mita@toshiba.co.jp

Abstract AdaBoost is a machine learning algorithm for pattern classification. It has been applied to many tasks because it gives good generalization performance with simple implementation. This tutorial paper introduces principles of the AdaBoost algorithm and how to implement it. Basic property, which makes implementation easy and accurate, and how it is applied to face detection are also explained.

Key words AdaBoost, Machine learning, Pattern classification, Face detection

1. はじめに

画像を対象とした研究分野では，入力データの中に含まれる特定のパターンを検出する，あるいは複数のパターンを区別するといった問題が頻繁に現れる．例えば，画像に含まれる人物の顔や車を検出するという課題である．音声を扱う分野でも，いま聞こえている音が人の話し声なのか音楽なのかを識別するといった類似した課題が存在する．

AdaBoostは，このようなパターン識別のための学習方式の1つである．高い識別精度が得られる，実装が容易であるなど利点があるため，近年盛んに応用されている．最も典型的な応用例は，画像から特定の対象を検出するオブジェクト検出という課題である．AdaBoostによる学習では，識別したい複数のクラスに属するパターンの例題（学習サンプル）を与えると，属するクラスが未知の入力パターン（未学習サンプル）を識別するための関数が得られる．AdaBoostは，逐次的に学習サンプルの重みを変化させながら異なる識別器を作り，これら複数の識別器の重み付き多数決によって最終的な識別関数を与える．個々の識別器は弱識別器（weak classifier）あるいは弱仮説（weak hypothesis）と呼ばれ，それらを組み合わせたものは強識別器（strong classifier）あるいは最終仮説（final hypothesis）などと呼ばれる．単純で弱い識別器を逐次的に学習し，識別器の精度を“boost（増強）”する方法を総称してBoostingと呼ぶ．AdaBoostは“adaptive（適応的）”にサンプルの重みを更新することにより，サンプルを使いまわすことに成功したBoostingアルゴリズムの一種である．図1に，AdaBoostによって得られる識別器を示す．入力パターンxに対し，そのパターンが属するクラスのラベルyを出力する．T個の弱識別器$h_t(x)$を信頼度α_tで重み付けて結合することによって強識別器が構成される．学習の際は，まず$h_1(x)$を選び，次の$h_2(x)$は$h_1(x)$が苦手なサンプルを識別できるように逐次的に選択される．

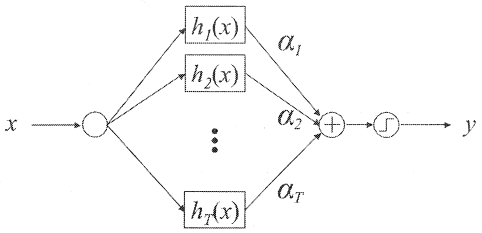

図 1　AdaBoost によって得られる識別器．弱識別器 $h_t(x)$ とその信頼度 α_t を学習する．

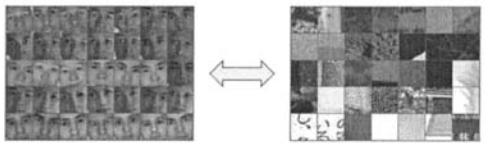

図 2　顔とそれ以外（非顔）の学習サンプルから 2 クラス識別器を構成．顔のサンプルは，目や鼻の位置を基準として，大きさを正規化して切り出す．

以下では，AdaBoost アルゴリズムについて実装の手助けとなる実例を交えて説明した後，訓練誤差（学習サンプルに対する識別誤り率）と汎化誤差（未学習サンプルに対する識別誤り率）の性質を検討してアルゴリズムの理解を深める．本稿では，基本的に 2 クラス識別問題（識別したいクラスの数が 2 つだけの場合）を扱う．多クラスを扱う方法については，最後に参考文献を示すにとどめる．

2.　2 クラスを識別できるだけで役に立つのか？

本稿では 2 クラス識別問題を扱うと述べたが，「本当に 2 クラスを識別できるだけで役に立つのか」という意見もあるかもしれない．もちろん，そのままでは多クラスを識別できないが，いくつもの 2 クラス識別器を組み合わせて識別を繰り返せば最終的には分類できる．また，2 クラス識別器をそのまま使って実現できる例もある．現在主流となっている顔検出の方法は，2 クラス識別器を用いたものである．まず，図 2 に示すように，顔と顔以外（非顔）の学習サンプルを用いて識別器を学習する．次に，図 3 のように識別器を入力画像の端から端まで走査して，各位置において顔か否かを判定することで顔を検出できる．入力画像中の顔の大きさは未知であるので，走査する識別器を拡大あるいは入力画像を縮小することで，大きさの変化にも対応できる．図 1 に対応させるなら，入力パターン x は各位置におけるウィンドウ内の画像パターンあるいはそこから抽出した特徴量，クラスラベル y は顔なら 1，非顔（背景）なら -1 とする．目や鼻といった顔部品，手や車など特定のオブジェクトを検出する問題は，同様の枠組みで対処できる．

3.　AdaBoost

ここでは，まず AdaBoost の学習手続きを概説する．次に，顔検出を例として具体的な実装方法を紹介する．最後に，なぜ

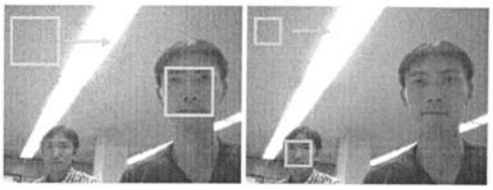

図 3　識別器の走査による顔検出（走査ウィンドウの位置と大きさを変えて，何度もスキャンする）

そのようなアルゴリズムが導出されるのかを示す．

3.1　アルゴリズム

T 個の弱識別器 $h_t(x)$ の重み付き多数決として，未知サンプル x のクラスラベル $y \in \{+1, -1\}$ を求める強識別器 $H(x)$ の学習手続きについて説明する．詳細は図 4 に示す．

まず，N 個の学習サンプル $(x_1, y_1), \cdots, (x_N, y_N)$ を用意する．x_i はサンプルであり，検出対象とそれと区別したい別の対象の画像パターンを意味する．例えば，顔を検出したいのであれば，大勢の人の顔を撮影して集めた画像と顔以外（非顔）の画像とする（図 2）．学習サンプルには，あらかじめクラスラベルをつけておく．例えば，顔には $+1$，非顔には -1 のようにつける．さらに，各サンプルの重みを評価するための変数 $D_t(i)$ も用意しておく．AdaBoost は，各サンプルの重みを更新することによりサンプル分布を変更し，それぞれの分布の下で弱識別器を学習していく．具体的には，初期の重みは均等に割り振っておき，その後は弱識別器が正しく識別できたサンプルについては重みを小さくし，間違えたサンプルについては重みを大きくする．これにより，直前の弱識別器が苦手とするサンプル分布の下で，次の弱識別器が学習される．これは，すなわち得意分野の異なる弱識別器を多数生成していることになる．

最終的に得られる強識別器 $H(x)$ は次式で与えられる．

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

ここで，α_t は t 番目の弱識別器 $h_t(x)$ の信頼度である．上式のように，強識別器 $H(x)$ は弱識別器の線形結合で表され，右辺の括弧内が正の値であれば $+1$，負であれば -1 を返す．これは，すなわち弱識別器の重みつき多数決である．想像しやすいように例えると，h_1〜h_T の T 名の判定者が入力パターン x を見て，それぞれクラスラベルを答える．通常は，$+1$ を答えた人の数を単純にカウントする多数決を行うが，AdaBoost による識別器では，各人の発言の信頼性 α_t を加算することによって最終的な判定を下す．

AdaBoost の学習の目的は，α_t と $h_t(x)$ を決定することである．実際には，$h_t(x)$ が決まると α_t も決まるので，$h_t(x)$ をいかに決定するかに帰着される．基本的に $h_t(x)$ は自由に設計してよく，満たすべき条件は以下の 3 点である．

(i) $h_t(x)$ は，$H(x)$ と同様に $+1$ もしくは -1 の 2 値を出力する関数とする．

(ii) $h_t(x)$ は，誤り率が 0.5 より小さくなければならない．

(iii) $h_t(x)$ は，AdaBoost によって更新されたサンプルの重

み分布 D_t のもとで求める．

条件 (i) は条件というほどのものではなく，満たすことは全く簡単である．条件 (ii) については 2 クラス識別の場合は容易にクリアできる．2 クラスの識別を行う場合，当てずっぽうでも識別結果が間違いとなる確率（誤り率）は 0.5 である．誤り率が 0.5 より大きくなった場合には判定を逆にすればよいので，誤り率がちょうど 0.5 になるような場合だけ避ければよいということになる．このような最悪の場合は，めったに起こらないのでほとんど問題にならない(注1)．条件 (iii) は，他の条件に比べれば厳しいかもしれない．学習サンプル 1 個 1 個に付与された重みを考慮しなければならないので，一工夫必要である．フィッシャーの線形判別法を用いる場合，一般の教科書には重みを考慮するような解説は記載されていないことが多いため，自分で式を変形しなければならない．よく用いられるのは，何らかの特徴量の生起確率に基づいてベイズ識別を行う方法である．生起確率は各サンプルの重みから計算すればよい．

なぜ，誤り率が 0.5 より小さくなければならないかを図 4 の数式を参照しながら説明する．信頼度 α_t は，弱識別器の誤り率 ϵ_t から，

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{2}$$

によって計算する．誤り率が大きいほど，信頼度の値は小さくなる．最悪の場合すなわち $\epsilon_t = 0.5$ の場合は，信頼度は $\alpha_t = 0$ となる．この場合，$\exp(-\alpha_t) = \exp(\alpha_t) = 1$ となるので，Step 2(C) において $D_{t+1}(i) = D_t$ となり，サンプルの重みを更新できなくなる．重みが更新できないと，何度やっても同じ弱識別器ばかりが得られるので学習を継続する意味はない．逆に，誤り率が 0 の場合は，α_t を計算できないため，やはり学習を継続できない．誤り率が 0 の弱識別器（間違いを犯さない識別器はすでに「弱」識別器ではないが）は，そもそも “boost（増強）” する必要がないから当たり前である．現実問題として，誤り率が 0 の識別器が得られることがあるのだろうか？もし，そんな識別器を見つけられたとすると素晴らしいことであるが，これを達成するのは一般には大変なことである．例えば，顔検出では，図 5 のような顔にとてもよく似た非顔パターンが存在し，これらと顔を区別しなければならない．このようなサンプルを全く間違えずに識別することは難しいので，誤り率が 0 になったのは，おそらく与えた学習サンプルに難しいサンプルが含まれていなかった，すなわち学習サンプルが簡単すぎたためだろう．このような問題を避けるには，十分な数の学習サンプルを用意することが重要である．顔検出では，10,000 枚程度の顔サンプルが用いられる場合が多い．

3.2 顔検出：簡単な例

画像から対象を探す処理で最も代表的なものはテンプレートマッチングである．テンプレートには多数の顔画像から作成した平均顔が用いられることが多い．また，テンプレートと注目領域の類似性を判定する尺度として，正規化相関がよく用いら

(注1)：ただし，これは学習を進められるか否かという条件であり，できあがる識別器の精度の良し悪しは別である．誤り率が 0.49999$\cdots$ のような弱識別器ばかりを組み合わせても，高い汎化性能は得られない．

N 個の学習サンプル $(x_1, y_1), \cdots, (x_N, y_N)$ が与えられているとする．ただし，x_i はサンプルで，$y_i \in \{+1, -1\}$ は x_i が属するクラスのラベルである．
→ 検出対象は $y_i = +1$，それ以外は $y_i = -1$ のようにラベルを付与するのが一般的である．

Step 1. 各サンプルの重みを $D_1(i) = \frac{1}{N}$ で初期化する．

Step 2. For $t = 1, ..., T$:

(A) サンプル分布 D_t において，弱識別器 $h_t(x)$ を学習する．学習サンプルに対する誤り率
$\epsilon_t = \sum_{i:y_i \neq h_t(x_i)} D_t(i)$
が最小となる $h_t(x)$ を選ぶ．
→ 誤り率はサンプルの個数ではなく重みに基づいて算出する．

(B) 誤り率から信頼度 α_t を計算する．
$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
→ 誤り率が小さいほど信頼度は大きくなる．

(C) サンプルの重みを更新する．
$D_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i))$
→ 弱識別器で正しく識別できたサンプルについては $y_i h_t(x_i) = 1$ であるから，
$D_{t+1}(i) = D_t(i)\exp(-\alpha_t)$
となる．すなわち，重みは小さくなる．
→ 間違えたサンプルについては $y_i h_t(x_i) = -1$ であるから，
$D_{t+1}(i) = D_t(i)\exp(\alpha_t)$
となる．すなわち，重みは大きくなる．
→ $\epsilon_t = 0.5$ の場合は $\alpha_t = 0$ となり，重みが更新されないので学習を継続できない．

(D) サンプルの重みの和が 1 になるように正規化する．
$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_{t+1}}$
→ ここで，$Z_{t+1} = \sum_{i=1}^{N} D_{t+1}(i)$
$= \sum_{i=1}^{N} D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

Step 3. 最終的な識別器は，すべての弱識別器を信頼度で重み付けて多数決をとった
$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$
となる．

図 4　AdaBoost アルゴリズムによる識別器の学習手順

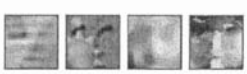

図 5　顔によく似た非顔の例

れる．ここでは，平均顔テンプレートと正規化相関に基づく弱識別器 $h_t(x)$ を用いて学習を行う例を紹介する．

ここでは，少し拡張して非顔の平均テンプレートも用い，顔・非顔 2 つのテンプレートと注目領域との正規化相関の差に基づく弱識別器を以下のように設計する．

$$h_t(x) = \begin{cases} +1 & \text{if } CC(A_f, x) - CC(A_n, x) > \theta \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

これにより，入力画像 x が顔テンプレートに類似している場合は顔，非顔テンプレートに近い場合は非顔と判定する弱識別器となる．ここで，$CC(A, x)$ はテンプレート A と入力画像 x の正規化相関を表し，次式によって計算する．

$$CC(A, x) = \frac{\sum (A - \bar{A})(x - \bar{x})}{\sqrt{\sum (A - \bar{A})^2} \sqrt{\sum (x - \bar{x})^2}} \tag{4}$$

A_f，A_n は顔および非顔の平均テンプレートである．また，θ は正規化相関の値の差に対するしきい値であり，学習サンプルを識別したときの誤り率が最小となるようにを調節すればよい．なお，平均テンプレート A_f と A_n の j 番目の画素値は，以下のようにサンプルの重み $D_t(i)$ に基づいてそれぞれ算出される．これによって，上述の条件 (iii) を考慮したものとなる．

$$A_f(j) = \sum_{i:y_i=1} x_i(j) \cdot D_t(i) \tag{5}$$

$$A_n(j) = \sum_{i:y_i=-1} x_i(j) \cdot D_t(i) \tag{6}$$

学習結果を図 6 に示す．1 個目の弱識別器では，すべてのサンプルの重みが均等であるので，単純なサンプル平均のテンプレートが用いられる．その後は重みが更新され難しいサンプルが強調されるので，最後の弱識別器（100 個目）では平均テンプレートが変わってくる．図では，$h_{100}(x)$ の顔テンプレートは若干ぼけており，分かりづらいが非顔テンプレートには顔に類似した明暗が浮き出てきている．

識別結果を図 7 に示す．横軸は弱識別器の数 T で，縦軸は識別誤り率である．訓練誤差，汎化誤差とも減少している様子がわかる．グラフの左端のデータがテンプレート 1 枚だけ用いた結果であるので，100 枚のテンプレートを用いると誤差を 1/5 程度に削減できている．

以上の例は，AdaBoost に複数のテンプレートを生成させ，マルチテンプレートに基づく識別器を構成した例ととらえることもできる．

3.3　顔検出：Viola と Jones の方法

もう 1 つの例として，Viola と Jones によって提案された顔検出の定番とも言える方法 [2] を紹介する．この方法では，Haar-like 特徴（図 8）という方形を組み合わせた特徴に基づく弱識別器を用いる点が特色である．特徴量の値は，白い方形

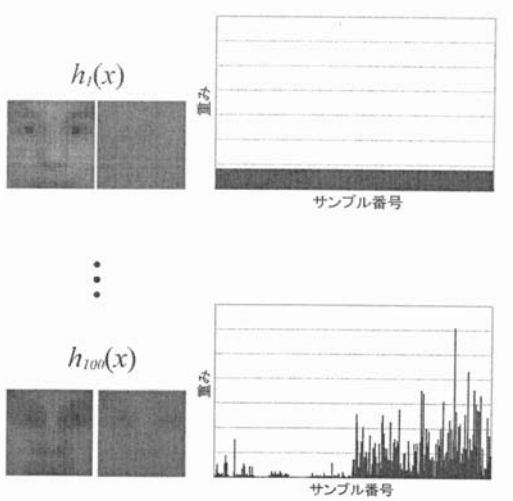

図 6　学習結果

図 7　識別結果

内の明度平均値と黒い方形内の明度平均値の差で表現する．図では，Haar-like 特徴を顔画像上に配置した例も示しているが，この特徴によって右目とその下の領域の明るさを比較することが可能となる．

Haar-like 特徴に基づく弱識別器を以下のように設計する．

$$h_t(x) = \begin{cases} +1 & \text{if } p \cdot z(x) > p \cdot \theta \\ -1 & \text{otherwise} \end{cases} \tag{7}$$

ここで，z は 1 つの Haar-like 特徴から算出された特徴量であり，θ はしきい値である．また，p は特徴量 z としきい値 θ を比較する不等号の向きを決定する変数で，$+1$ もしくは -1 の値をとる．θ と p は，学習サンプルに対する識別誤り率が最小となるように値を求める．

学習に先だって，図 9 に示すように，学習サンプルの画像内で方形の位置や大きさを網羅的に変化させることにより，様々な Haar-like 特徴を事前に生成しておく．例えば，20×20 画素のサンプルを与えると，数万個の特徴を生成できる．弱識別器の学習段階では，多数の特徴候補から最も識別誤り率の小さいものを選択する．これを繰り返して，識別に有効な特徴を逐次選択していく．

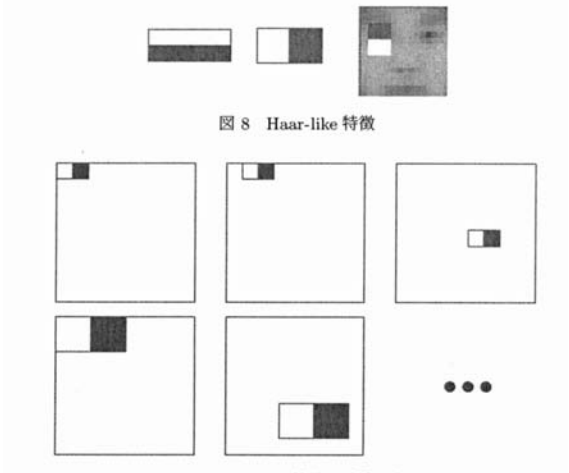

図 8　Haar-like 特徴

図 9　Haar-like 特徴の候補を生成

この方法の特長は，Integral Image [2] をあらかじめ作成することにより Haar-like 特徴が高速演算でき，しかも選ばれた比較的少数の特徴を使って識別を行う点にある．識別時の計算量を小さくできるため，顔を高速に検出できる．また，AdaBoost は特徴選択にも利用できるということを示した点も意義がある．

なお，Viola と Jones [2] は上記に加えてカスケード構造を取り入れることにより，さらに高速化・高精度化することを提案しており，実時間での顔検出を実現している．カスケード構造については，Boosting に基づく識別器に固有の技術ではないので，ここでは解説を割愛する．

3.4　T の決定方法

残された問題として，弱識別器の数 T をどのように決めるかがある．T は学習を始める段階で決めておかなければならないハイパーパラメータである．後述するように AdaBoost では過学習が起こりづらいので，基本的には T は大きいほどよい．しかし，T を大きくするほど弱識別器の数が増えるので，識別処理に要する計算コストも増えることになる．計算コストを削減したい場合には，誤差曲線において汎化誤差が下げ止まったあたり，例えば図 6 では $T = 70$ あたりの値を用いればよい．それでもまだ計算コストと精度の折り合いがつかない場合は問題である．この場合は，根本的に弱識別器の設計を考え直し，高速かつ高精度な方法を見出す必要がある．

3.5　アルゴリズムの詳細

3.5.1　弱識別器の更新後の分布における誤り率は 0.5

ここで示す性質は，AdaBoost を実装する上で大変役に立つ．この性質が成り立っていることを確認すれば，誤り率や信頼度の計算および重みの更新部分の実装はほぼ正しいと判断できる．また，この性質は AdaBoost によって互いに異なる得意分野を持つ弱識別器が選ばれることを示しており，非常に興味深い．「三人寄れば文殊の知恵」と例えられる所以が理解できる．

図 4 の Step 2. (D) に示したように，弱識別器 $h_t(x)$ の更新後の分布 Z_{t+1} は，

$$\begin{aligned} Z_{t+1} &= \sum_{i=1}^{N} D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right) \\ &= \sum_{i:y_i=h_t(x_i)} D_t(i) \exp\left(-\alpha_t\right) \\ &\quad + \sum_{i:y_i \neq h_t(x_i)} D_t(i) \exp\left(\alpha_t\right) \end{aligned} \tag{8}$$

であるが，

$$\sum_{i:y_i=h_t(x_i)} D_t(i) = 1 - \epsilon_t \tag{9}$$

$$\sum_{i:y_i \neq h_t(x_i)} D_t(i) = \epsilon_t \tag{10}$$

$$\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{11}$$

であるので，

$$\begin{aligned} Z_{t+1} &= \epsilon_t \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} + (1-\epsilon_t)\sqrt{\frac{\epsilon_t}{1-\epsilon_t}} \\ &= 2\sqrt{(1-\epsilon_t)\,\epsilon_t} \end{aligned} \tag{12}$$

のように式変形できる．この分布のもとでの $h_t(x)$ の誤り率は，

$$\begin{aligned} \epsilon_t' &= \sum_{i:y_i \neq h_t(x_i)} \frac{D_t(i) \exp\left(\alpha_t\right)}{Z_{t+1}} \\ &= \frac{\sqrt{(1-\epsilon_t)\,\epsilon_t}}{2\sqrt{(1-\epsilon_t)\,\epsilon_t}} \\ &= 0.5 \end{aligned} \tag{13}$$

となる．すなわち，更新された分布 D_{t+1} は弱識別器 $h_t(x)$ が最も苦手とするものであり，次に選択される弱識別器 $h_{t+1}(x)$ は直前の弱識別器では識別できないものをうまく扱えるようなものとなる．

3.5.2　アルゴリズムの導出

ここでは，図 4 に示すアルゴリズムがどのように導出されたかを説明する．鍵となるのは，指数損失という損失関数を導入する点である．以下では，なぜアルゴリズムが指数損失を小さくすることができるのかを説明する．その後，指数損失が用いられる理由について述べる．式展開などは [1] をベースとしている．

$T-1$ 個の弱識別器の線形結合

$$F(x) = \sum_{t=1}^{T-1} \alpha_t h_t(x) \tag{14}$$

は，識別関数 $H(x)$ の $sign(\cdot)$ の中身をとったものであるが，この関数の指数損失を

$$L(F) = \frac{1}{N} \sum_{i=1}^{N} \exp\left(-y_i F(x_i)\right) \tag{15}$$

と定義する．$F(x)$ に t 個目の弱識別器 $h_t(x)$ を重み $\alpha_t (> 0)$ で 1 つ加えて，

$$F(x) + \alpha_t h_t(x) \tag{16}$$

とし，指数損失 $L(F+\alpha_t h_t)$ を小さくすることを考える．

$$
\begin{aligned}
&L(F+\alpha_t h_t)\\
&=\frac{1}{N}\sum_{i=1}^{N}\exp\left(-y_i\left(F(x_i)+\alpha_t h_t(x_i)\right)\right)\\
&=\frac{1}{N}\sum_{i=1}^{N}\exp\left(-y_i F(x_i)\right)\exp\left(-\alpha_t y_i h_t(x_i)\right)\\
&=\frac{1}{N}\sum_{i:y_i=h_t(x_i)}\exp\left(-y_i F(x_i)\right)\exp(-\alpha_t)\\
&+\frac{1}{N}\sum_{i:y_i\neq h_t(x_i)}\exp\left(-y_i F(x_i)\right)\exp(\alpha_t)\\
&=\frac{\exp(-\alpha_t)}{N}\sum_{i=1}^{N}\exp\left(-y_i F(x_i)\right)\\
&-\frac{\exp(-\alpha_t)}{N}\sum_{i:y_i\neq h_t(x_i)}\exp\left(-y_i F(x_i)\right)\\
&+\frac{\exp(\alpha_t)}{N}\sum_{i:y_i\neq h_t(x_i)}\exp\left(-y_i F(x_i)\right)\\
&=\frac{\exp(-\alpha_t)}{N}\sum_{i=1}^{N}\exp\left(-y_i F(x_i)\right)\\
&+\frac{\exp(\alpha_t)-\exp(-\alpha_t)}{N}\sum_{i:y_i\neq h_t(x_i)}\exp\left(-y_i F(x_i)\right) \qquad (17)
\end{aligned}
$$

と変形できる．第 1 項は h_t によらないので，どのような h_t を加えるかを決めるには第 2 項のみを考えればよい．ここで，

$$D_t(i)=\frac{\frac{1}{N}\exp\left(-y_i F(x_i)\right)}{Z_{t-1}} \qquad (18)$$

$$Z_t=\frac{1}{N}\sum_{i=1}^{N}\exp\left(-y_i F(x_i)\right) \qquad (19)$$

であるので，第 2 項は分布 D_t における h_t の誤り率

$$\epsilon_t=\sum_{i:y_i\neq h_t(x_i)}D_t(i) \qquad (20)$$

を用いて，

$$
\begin{aligned}
&\frac{\exp(\alpha_t)-\exp(-\alpha_t)}{N}\sum_{i:y_i\neq h_t(x_i)}\exp\left(-y_i F(x_i)\right) \qquad (21)\\
&=\left(\exp(\alpha_t)-\exp(-\alpha_t)\right)Z_t\epsilon_t \qquad (22)
\end{aligned}
$$

と書ける．すなわち，損失を小さくするには分布 D_t において誤り率が最小となる h_t を探せばよいということになる．

次に，h_t が決まった後で損失関数

$$
\begin{aligned}
&L(F+\alpha_t h_t)\\
&=Z_t\left\{\exp(-\alpha_t)+\left(\exp(\alpha_t)-\exp(-\alpha_t)\right)\epsilon_t\right\} \qquad (23)
\end{aligned}
$$

を最小にする α_t を考える．右辺は下に凸な関数である．α_t で微分して 0 とおくことにより，

$$\alpha_t=\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t} \qquad (24)$$

で最小値をとることが分かる．$\epsilon_t<0.5$ であれば，

$$L(F+\alpha_t h_t)=L(F)\cdot 2\sqrt{(1-\epsilon_t)\epsilon_t}<L(F) \qquad (25)$$

となるので，

$$F(x)+\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}h_t(x) \qquad (26)$$

とすることで損失関数をさらに小さくできる．

このようにして，AdaBoost は指数損失のサンプル平均を最小化する．誤り率が最小となる弱識別器を学習することは，損失関数が最も減少する方向を探索していると解釈できる．また，信頼度 α の決定は，損失関数を弱識別器の方向へ移動させる操作に当たる．したがって，AdaBoost は greedy（貪欲）な探索により損失関数を最小化する最急降下法の 1 つであるとも言える．

では，このようなアルゴリズムを導出するそもそもの出発点となった指数損失は，どこから出てきたものだろうか？図 10 に示すように指数損失は 0-1 損失(注2)の上界になっている．横軸の $yF(x)$ は識別境界とのマージンを表すので，このグラフは左に行くほど（すなわち，識別が大間違いであるほど）大きな損失が与えられるということを示している．AdaBoost は 0-1 損失を最小化するために，その上界である指数損失を最小化する．これは，$yF(x)$ を最大化することにあたるから，0-1 損失を小さくするだけでなく識別境界とのマージンを大きくしようとしていることにもなる．

次に，指数損失に基づく損失関数の意味を考えてみる．実際の学習では指数損失のサンプル平均（経験損失）を最小化するが，ここではサンプルを生成している真の確率分布 $P(x,y)$ を用い，P による期待損失

$$
\begin{aligned}
E[\exp\left(-yF(x)\right)] &=\int\int P(x,y)\exp\left(-yF(x)\right)dxdy\\
&=\int P(x)[\exp\left(-F(x)\right)\cdot P(y=+1|x)\\
&+\exp\left(F(x)\right)\cdot P(y=-1|x)]dx \qquad (27)
\end{aligned}
$$

を最小化する $F(x)$ について考える．これを x で微分して 0 とおく．

$$
\begin{aligned}
\frac{d}{dx}E[\exp\left(-yF(x)\right)] &=P(x)[\exp\left(-F(x)\right)\cdot P(y=+1|x)\\
&+\exp\left(F(x)\right)\cdot P(y=-1|x)]\\
&=0 \qquad (28)
\end{aligned}
$$

これを $F(x)$ について解くと，

$$F(x)=\frac{1}{2}\log\frac{P(y=+1|x)}{P(y=-1|x)} \qquad (29)$$

となる．$F(x)$ の符号だけを考えた $H(x)=sign\left(F(x)\right)$ はベイズ則である．このように，指数損失を導入したことにより，ベイズ識別との関係も明らかとなった．

指数損失以外の損失関数を使って Boosting は一般化される．例えば，LogitBoost [8] では，次式の 2 項対数尤度が用いられる．

(注2)：正しく識別できたら損失 0，間違えたら損失 1 とカウントする損失

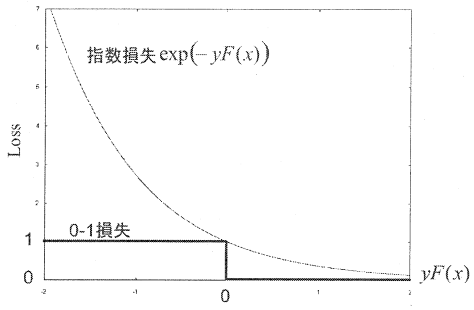

図 10　指数損失は 0-1 損失の上界

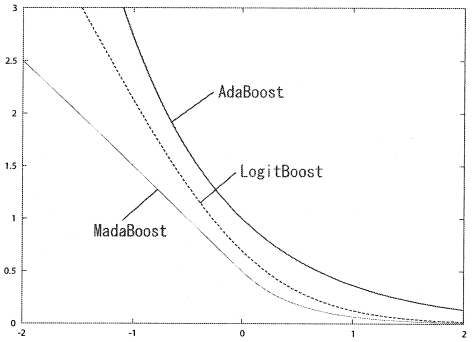

図 11　様々な損失関数

$$L(F) = \log\left(1 + \exp\left(-2yF(x)\right)\right) \tag{30}$$

また，MadaBoost [9] では，次式が用いられる．

$$L(F) = \begin{cases} -yF(x) + \frac{1}{2} & \text{if } yF(x) < 0 \\ \frac{1}{2}\exp\left(-2yF(x)\right) & \text{otherwise} \end{cases} \tag{31}$$

比較のため，これらと AdaBoost の指数損失をプロットしたグラフを図 11 に示す．

4.　訓練誤差の性質

上述したように指数損失は 0-1 損失（訓練誤差(注3)）の上界である．AdaBoost はその指数損失を最小化するアルゴリズムであり，図 4 の手順にしたがって弱識別器を追加することによって，指数損失を単調に減少させることができる(注4)．ここでは，[1] でなされている考察を抜粋しながら，これを確認する．

識別関数 $H(x) = sign\left(F(x)\right)$ の訓練誤差は以下のように指数損失でおさえられる．

$$\frac{1}{N}\sum_{i=1}^{N} I\left(y_i \neq H(x_i)\right) \leqq \frac{1}{N}\sum_{i=1}^{N} \exp\left(-y_i F(x_i)\right)$$

(注3)：訓練誤差は，学習サンプルに対して識別を行ったときに間違えた回数がサンプル総数に占める割合．

(注4)：指数損失が単調減少しているかどうかを確認すれば，実装が正しいかどうか検証する手助けとなる．なお，0-1 損失（訓練誤差）が単調に減少すると誤解されることが多いので注意．訓練誤差は単調に減少するとは限らない．

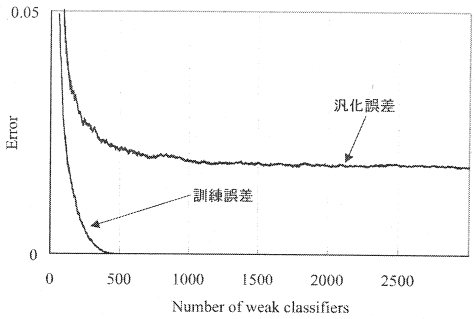

図 12　Viola-Jones 識別器の訓練誤差と汎化誤差

$$= \prod_t Z_t$$
$$= \prod_t \left(2\sqrt{(1-\epsilon_t)\epsilon_t}\right) \tag{32}$$

右辺はさらに次のように展開できる．

$$\prod_t \left(2\sqrt{(1-\epsilon_t)\epsilon_t}\right) = \prod_t \sqrt{1-4\gamma_t^2}$$
$$\leqq \exp\left(-2\sum_t \gamma_t^2\right) \tag{33}$$

ここで，$\epsilon_t = 1/2 - \gamma_t$ とする．γ_t は誤り率が 1/2 よりどれだけ小さいかを表す．すべての t に対して $\epsilon_t \leqq 1/2 - \gamma$ となるような共通の $\gamma(>0)$ が存在するとき，

$$\frac{1}{N}\sum_{i=1}^{N} I\left(y_i \neq H(x_i)\right) \leqq \exp\left(-2\gamma^2 T\right) \tag{34}$$

と書け，訓練誤差が指数的に減少することが分かる．

したがって，常に 0.5 より小さい誤り率を持つ弱識別器の追加を繰り返すことによって，訓練誤差をいくらでも小さくすることができる．すなわち，当てずっぽうより少しでもマシな識別器があれば，それらを多数組み合わせ強い識別器を作ることが可能となる．実際に，Viola ら [2] の顔検出器を学習してみると，訓練誤差は比較的早い段階で 0 に収束することを確認できる（図 12）．ただし，これはあくまでも学習サンプルに対しての識別性能の話である．誤り率が 0.4999 $\cdots$ のようなとても弱い弱識別器を多数学習しても，未学習サンプルに対する識別性能（汎化性能）は向上しない．実際，図 12 では，弱識別器の数が 1,000 を超えて 3,000 に至っても，汎化誤差はほとんど減少していない．このときの弱識別器の誤り率は 0.45〜0.49 であった．すなわち，当てずっぽうに近い精度の弱識別器は，訓練誤差を減少させることはできても，汎化誤差を減少するのには貢献しない．現実的には，弱すぎず強すぎずちょうどよい弱識別器を設計することが重要である．

5.　汎化誤差の性質

学習に用いたサンプルとは別のサンプルに対して識別を行ったときの誤り率を汎化誤差と呼ぶ．マージンという概念と VC

次元を用いた理論検証により，AdaBoost では汎化誤差も小さくしようとしていることが示されている．

1 つのサンプル (x, y) のマージンは，

$$M_F(x,y) = \frac{yF(x)}{\sum_t \alpha_t} \tag{35}$$

で定義される．マージンは $[-1,+1]$ の間で，正しく識別されるとき正の値をとる．すべての弱識別器が正しく識別したときに $+1$，間違ったときに -1 となることから，その大きさは識別の信頼度を測る量と解釈できる．AdaBoost は損失関数 $E[\exp(-yF(x))]$ を最小化しようとするが，これは $yF(x)$ を大きくすることによってマージンを大きくしようとしていると理解できる．図 12 では，弱識別器の数が 500 個程度になった段階で訓練誤差が 0 となっているが，その後も汎化誤差が減少している．こうした現象は，AdaBoost がマージンを大きくしているためであると解釈できる．

なお，マージンに基づく汎化誤差の上界に関して以下の定理を参考に記述しておく．

定理　十分小さな δ に対して，確率 $1-\delta$ で

$$Pr_D\{M_F(x,y) \leqq 0\} \leqq Pr_S\{M_F(x,y) \leqq \theta\} + O\left(\frac{d}{N\theta^2}\right) \tag{36}$$

が成立する [1]．

左辺は入力 x が従う真の分布のもとでマージンが 0 より小さくなる確率（汎化誤差を意味する）である．右辺第 1 項は，例題に基づく経験分布のもとでマージンが θ より小さくなる確率（訓練誤差を意味する）である．右辺第 2 項の d は弱識別器の空間の VC 次元，N は例題の数である．θ が大きくなると，右辺第 1 項は大きくなり第 2 項は小さくなるので，適当な θ で右辺は最小となる．これが，汎化誤差の上界である．右辺は弱識別器の数 T によらないので，汎化誤差の上界は T に依存しないことが分かる．また，所詮は弱識別器を結合したものであるので，T を大きくしすぎても過学習が起きにくいという説もある [3]．

5.1 SVM との関連性

2 クラス識別器でマージンを大きくする学習機械という点では，Support Vector Machine(SVM) と共通している．特に，非線形カーネル SVM の識別関数は，

$$H(x) = sign\left(\sum_{i:SV} \alpha_i y_i K(x, x_i) + b\right) \tag{37}$$

と，AdaBoost の識別関数の形に非常によく似ている．カーネル関数 $K(x, x_i)$ がサポートベクター x_i と入力 x との類似度を意味するのに対し，AdaBoost ではその部分が弱識別器になっている．

6. ま と め

本稿では，AdaBoost アルゴリズムについて実装例を紹介しながら説明した．AdaBoost を支える理論の中心的な部分については，ほぼ紹介できていると思われるが，より詳細に勉強したい方のために参考文献を紹介しておく．まず，Freund と Shapire の論文 [4] がある．日本語での解説は，[1] [3] [5] [6] がある．特に，[5] [6] では，AdaBoost に発展する前の Boosting 手法として「フィルタに基づく Boosting」についても紹介されているので，Boosting の発展の歴史に興味をお持ちの方にお奨めする．AdaBoost の改良である Real AdaBoost [7] や LogitBoost [8] も参考となる．多クラス化については，[4] [7] でも考察されているほか，Sharing features [10] や Vector Boosting [11] などがある．また，興味深い論文としては，Friedman らによる AdaBoost とロジスティック回帰の関係を考察したもの [8] がある．

文　献

[1] 麻生英樹, 津田宏治, 村田昇, “パターン認識と学習の統計学,” 岩波書店.

[2] P. Viola and M. Jones. ”Rapid object detection using a boosted cascade of simple features,” Proc. of CVPR, pp. 511–518, 2001.

[3] 上田修功, “アンサンブル学習,” CVIM 論文誌, 46(SIG15), pp.11–20, 2005.

[4] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” Computational Learning Theory: Eurocolt, pp. 23–37, 1995.

[5] R. O. Duda, P. E. Hart and D. G. Stork, “パターン識別,” 新技術コミュニケーションズ.

[6] 村田昇, 金森敬文, 竹之内高志, “ブースティングと学習アルゴリズム　－三人寄れば文殊の知恵は本当か？－,” 信学誌, Vol.88, No.9, pp.511–518, 2001.

[7] R. E. Schapire and Y. Singer, “Improved boosting algorithms using confidence-rated predictions,” Machine Learning, Vol.37, No.3, pp.297–336, 1999.

[8] J. Friedman, T. Hastie and R. J. Tibshirani, “Additive logistic regression: a statistical view of boosting,” Technical report, Department of Statistics, Sequoia Hall, Stanford Univerity, July 1998.

[9] C. Domingo and O. Watanabe, “MadaBoost: a modification of AdaBoost,” Proc. of the 13th Conf. on Computational Learning Theory, COLT’00, 2000.

[10] A. Torralba, K. P. Murphy and W. T. Freeman, “Sharing features: efficient boosting procedures for multiclass object detection,” Proc. of CVPR, 2004.

[11] C. Huang, H. Ai, Y. Li and S. Lao, “Vector boosting for rotation invariant multi-view face detection,” Proc. of ICCV, 2005.