1R-01

AdaBoostのサンプリング数に関する研究*

菅原啓介†
東北大学大学院情報科学研究科

徳山豪‡
東北大学大学院情報科学研究科

1 はじめに

機械学習アルゴリズムAdaBoostは、複数の分類器を組み合わせてより高性能な分類器を構成する手法である。原型となる手法は1996年に提案され[2]、今日では主に画像中の物体認識などに用いられている[3]。

弱分類器と呼ばれる複数の分類器から強分類器と呼ばれる高性能な分類器を構成する枠組みはアンサンブル学習と呼ばれるが、その中でAdaBoostの特徴は、訓練データに重み付けを行いながら学習していくということである。特徴量 $\mathbf{x} \in R^d$ をクラス $y_n \in \{0,1\}$ に分類する分類器を作成するAdaBoostのアルゴリズムを**Algorithm 1**に示す。**Algorithm 1**の w_n が各訓練データの重みである。目的関数 J_t に示される通り、AdaBoostはすべてのデータ点の訓練誤差を均等に見るのではなく、w_n により重み付けを行って弱分類器を作成する。重みの更新が行われるのは8〜11行目であり、この更新は、ステップ t において作成した弱分類器 $f_t(\mathbf{x}_n)$ が正しく分類できたデータ点の重みは減少させ、誤分類したデータの重みを増加させることを意味する。更新された重みは次の弱分類器を作るときに使用されるため、t 個目の弱分類器が正確に分類できないデータを重点的に $t+1$ 個目の弱分類器が作成されることになる。このようにしてAdaBoostアルゴリズムは互いの弱点を補い合うような弱分類器を作成していくことにより、高性能な強分類器 $\mathrm{h}(\mathbf{x})$ を構成することができる。

Algorithm 1 AdaBoost (by Reweighting)

Input: 訓練データ集合 $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
Output: 分類器 $h(\mathbf{x})$

1: $w_n^{(1)} = 1/N$ に初期化
2: **for** $t = 1$ to T **do**
3: 次の目的関数 J_t を最小化するように $f_t(x)$ を訓練データにフィットさせる
4: $J_t = \sum_{n=1}^{N} w_n^{(t)} |f_t(\mathbf{x}_n) - y_n|$
5: $\epsilon_t = \sum_{n=1}^{N} w_n^{(t)} |f_t(x_n) - y_n|$
6: $\alpha_t = \dfrac{1-\epsilon_t}{\epsilon_t}$
7: **for** $n = 1$ to N **do**
8: $v_n^{(t)} = w_n^{(t)} \alpha_t^{|f_t(x_n)-y_n|-1}$
9: **end for**
10: **for** $n = 1$ to N **do**
11: $w_n^{(t+1)} = \dfrac{v_n^{(t)}}{\sum_{k=1}^{N} v_k^{(t)}}$
12: **end for**
13: **end for**
14: $F(\mathbf{x}) = \dfrac{\sum_{t=1}^{T} (\log \alpha_t) f_t(\mathbf{x})}{\sum_{t=1}^{T} \log \alpha_t}$
15: $h(\mathbf{x}) = \begin{cases} 1 & (F(\mathbf{x}) \geqq 1/2) \\ 0 & (F(\mathbf{x}) < 1/2) \end{cases}$

2 AdaBoostのサンプリング数

各弱学習器の作成のために様々な機械学習アルゴリズムが用いられるが、学習時に訓練データの重みを適用できないアルゴリズムに対しては、**Algorithm 1**の一部を改変した手法[2]が使用できる。重み付き誤差として考慮するのではなく、その重みを使って訓練データ全体から重み付き復元サンプリングを行うことで部分サンプルデータを得て、そのデータのみを弱分類器の作成に用いるのである。弱分類器の作成に重みを直接用いる方法と重みによる部分サンプリングを行う方法の学習性能を比較すると、一般に後者の方が良い性能を示すことが実験により示されている[4]。

重みによる部分サンプリングを行う方法について、Freundら[2]はサンプリング数は訓練データ数と等しくなるように設定しているが、サンプリング数を小さくすると学習にかかる時間を短縮できることが期待される。そこで本研究では、重みによる部分サンプリングを行うAdaBoostのサンプリング数を小さくした場合の学習精度の変化について実験と考察を行う。

3 実験

本実験では、サンプリング数を変えながら分類器のテスト誤差を計測し、両者の関係について考

*A Research Related to Sampling Size of AdaBoost
†Keisuke Sugawara. Graduate School of Information Science, Tohoku University
‡Takeshi Tokuyama. Graduate School of Information Science, Tohoku University

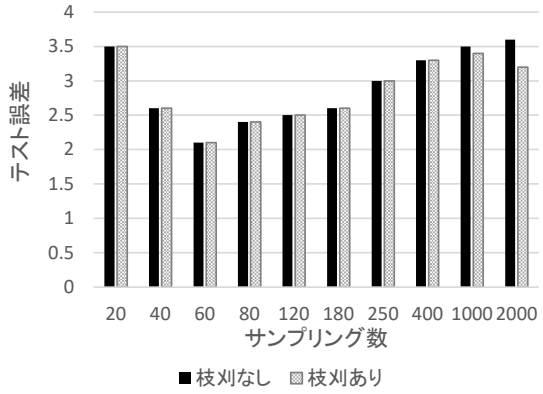

図 1: MNIST データセット使用時のサンプリング数とテスト誤差の関係

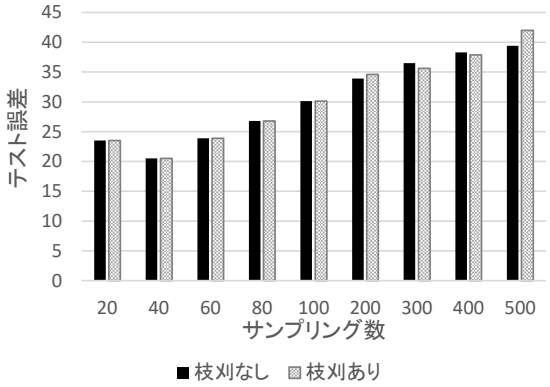

図 2: Banana データセット使用時のサンプリング数とテスト誤差の関係

察する。弱分類器の生成には CART[1] を用いて枝刈ありとなしの両方を試し、枝刈ありの場合の複雑度パラメータは 0.02 に設定した。また弱分類器数は 80 とした。データセットとして手書き文字 MNIST の 6 と 8 と、人工データセット Banana を用いて 2 値分類を行った。それぞれの情報は表 1 のとおりである。

表 1: データセットの情報

データセット	次元数	訓練データ数
MNIST 6&8	784	2000
Banana	2	4300

MNIST を用いた実験結果を図 1 に示す。サンプル数 60 の時にもっとも汎化誤差が小さくなり、この時の学習時間は 0.608 秒であった。一方でサンプル数 2000 では学習に 148 秒を費やした。サンプル数が 20〜400 では枝刈の有無で汎化誤差が変わっていない。このときに生成された決定木を確認したところ、枝刈が実際には行われておらず、両者で全く同じ決定木を作成していることが分かった。枝刈は決定木が複雑になりすぎたときに行われるため、サンプル数 20〜400 のときは単純な木が作成されたと言える。このことによる 2 つのメリットが挙げられる。1 つは、枝刈の必要がないほど単純な決定木を作成し分類に使用してもテスト誤差を低く抑えられるため、CART の複雑度パラメータのチューニングを行う必要がないということである。もう 1 つは、単純な決定木は深さが小さいため、データの分類を高速に行えるということである。

Banana による実験結果は図 2 の通りである。テスト誤差が最小になるサンプリング数は 40 となり MNIST の場合と異なるため、データセットによってこの値は変わってくると考えられる。サンプリング数が少ない場合は単純な決定木が作成されること、サンプリング数が多いとテスト誤差が大きくなっていることは MNIST を用いた場合と共通している。

4 おわりに

本研究では、AdaBoost について、訓練データ数よりも少ないサンプリング数の方がテスト誤差を小さくできることを実験により示した。

今後の課題として、テスト誤差を最も小さくできるようなサンプリング数を自動的に求める方法を考案したい。現状ではサンプリング数は学習パラメータであり、チューニングが必要であるという問題がある。サンプリング数の最適値は訓練データのサイズや次元数、分類器逐次生成時の訓練誤差などの情報に依存すると考えられる。様々なデータセットや弱学習アルゴリズムに対し、それらの依存性について解析を行っていきたい。

謝辞

本研究は革新的研究開発推進プログラム（ImPACT）の助成を受けたものである。

参考文献

[1] L Breiman, J Friedman, CJ Stone, RA Olshen. Classification and regression trees. 1984

[2] Y.Freund and R.E.Schapire. Experiments with a new boosting algorithm. 1996.

[3] Paul Viola and Michael J. Jones. Robust Real-Time Face Detection. 2004

[4] Jason Van Hulse Amri Napolitano Chris Seiffert, Taghi M. Khoshgoftaar. Resampling or reweighting: A comparison of boosting implementations. 2008.