アンサンブル自己生成ニューラルネットワークのための高速枝刈り法

井上　浩孝†　　成久　洋之‡
†岡山理科大学大学院工学研究科　‡岡山理科大学工学部情報工学科

既発表論文において，我々は高速学習を行う自己生成ニューラルネットワーク (self-generating neural networks:SGNN) の分類に対する汎化誤差を減少するため，アンサンブル自己生成ニューラルネットワーク (ensemble self-generating neural networks:ESGNN) を提案した．ESGNN は単一の SGNN に比べ高い精度を得られるが，計算コスト (記憶容量，計算時間) はアンサンブル学習に使用する SGNN の数が増えるにつれて増大する．本研究では，分類に対する記憶容量を削減するための新しい ESGNN の枝刈り法を提案する．UCI 機械学習データベースの 10 のデータセットに対し，枝刈り後の SGNN を基本予測器とする ESGNN と最近傍法によるモデルの解の精度，メモリ使用量，計算時間を比較する．実験結果より，本提案手法はメモリ使用量を枝刈り前の 30% から 90% 削減し，最近傍法を用いたモデルに比べ高い精度が得られることを示す．更に，二つのサンプリング法による枝刈り後の ESGNN の性能評価を行う．

A Fast Pruning Method for Ensemble Self-Generating Neural Networks

Hirotaka Inoue† and Hiroyuki Narihisa‡
†Graduate School of Engineering, Okayama University of Science
‡Department of Information & Computer Engineering, Okayama University of Science

In an earlier paper, we introduced an ensemble model called ESGNN (ensemble self-generating neural networks) which can be used to reduce the generalization error of a single self-generating neural network (SGNN) for classification. Although this model can obtain the high accuracy than a single SGNN, the computational cost increase in proportion to the number of SGNN in the ensemble learning. In this paper, we propose a new pruning method for ESGNN to reduce the memory requirement for classification. We compare ESGNN with nearest neighbor classifier using a collection of ten benchmark problems in the UCI machine learning repository. Experimental results show that our method could reduce the memory requirement from 30% to 90% of unpruned ESGNN and improve the accuracy over the accuracy of the nearest neighbor classifier. A performance analysis of pruned ESGNN on two sampling methods is also discussed.

1. はじめに

大量のデータから特徴を抽出し，未学習データに対する入出力関係を予測する数理モデルには，(i) 高い解の精度と (ii) 取り扱いが容易であるという性質が望まれている [1]．ニューラルネットワークは分類，クラスタリング，予測，パターン認識等の知的処理の分野で幅広く用いられている数理モデルである [2]．ニューラルネットワークは生物学，物理学，統計学，計算科学の分野からさまざまな手法が提案されているが，現在最も広く利用されているのはバックプロパゲーション (backpropagation: BP) 学習を行う階層型ニューラルネットワークである．階層型ニューラルネットワークは順応性や柔軟性に優れ，非線型入出力写像能力を持つ．しかしながら，中間層数や中間層内の各ユニット数，学習係数等を各設計者が問題の規模や複雑度に応じて決定しなければならない．

一方，自己生成ニューラルネットワーク（self-generating neural networks: SGNN） [3] がネットワーク設計の容易さのために注目を集めている．SGNN は代表的な教師なし学習モデルである自己組織化地図（self-organizing maps: SOM） [4] を拡張したものであり，訓練データを一度提示することで競合学習により自動的に自己生成ニューラル木（self-generating neural tree: SGNT）を生成する．このモデルは数理モデルの性質 (ii) を満たしているものの，(i) の解の精度は階層形ニューラルネットワークに劣る [5]．

SGNN の解の精度を改善するため，我々は同一の

訓練データから生成した複数の SGNN の出力の平均を全体の出力とするアンサンブル自己生成ニューラルネットワーク（ensemble self-generating neural networks: ESGNN）を提案した [6]．ESGNN は階層型ニューラルネットワークと同程度の解の質をより高速に算出することが可能である．しかし，計算時間および記憶容量は使用する SGNN の数が増加するにつれて増大する．

本研究では，記憶容量を削減し，汎化能力改善を行うための SGNN の枝刈り法を提案し，枝刈りを行った SGNN を基本予測器とするアンサンブルモデルの性能を検討する．UCI 機械学習レポジトリ [7] の 10 のデータセットを用いて，訓練データの提示順をランダムに入れ換えて生成した枝刈り SGNN，bagging [8] によって生成した枝刈り SGNN による ESGNN の記憶容量削減率，汎化誤差改善特性を比較検討する．また，比較対象とする既存の数理モデルとして，最近傍法によるモデルを用いて同一環境で実験を行う．

2. アンサンブル自己生成ニューラルネットワーク

SGNN は SOM [4] と同様に競合学習を用いて SGNT に与えられた訓練データセット $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}, |\mathcal{D}| = N$ 内のデータを動的に組み込む [9]．ここで，$\boldsymbol{x}_i \in \Re^m$ は入力ベクトルで y_i は期待出力を表す．図 1 に擬似 C 言語による SGNT 生成アルゴリズムを，表 1 に SGNT 生成法で使用される副手続きの説明を示す．

生成過程において，競合学習により訓練実例ベクトル $\boldsymbol{x}_i$ に対する SGNT 内の勝者ニューロン n_{win} を決定する．根から n_{win} に至るノードに存在するニューロン n_j の重み w_{jk} は次式を用いて更新する．

$$w_{jk} \leftarrow w_{jk} + \frac{1}{c_j + 1} \cdot (x_{ik} - w_{jk}) \ . \qquad (1)$$

ここで，c_j は n_j に含まれる葉の数を表す．SOM の場合，学習係数の初期値は学習前に任意に設定し，訓練データセット $\mathcal{D}$ を繰り返し提示する毎に単調減少させる．これに対して，SGNN では全訓練データを SGNT の葉として直接組み込むことで学習を行い，(1) 式によりそのノードに含まれる葉の持つ m 次元入力ベクトルの各要素の平均値となるように m 次元重みベクトルを修正する．故に，c_j が学習中に動的に変化することで学習係数が適応的に決定するため，ネットワークの設定およびパラメータの設定が不要である．更に，SGNN は $\mathcal{D}$ を 1 回提示するだけで学習を終了するため，繰り返し訓練データを提示して学習を行う既存のニューラルネットワーク学習法に比べ高速な学習特性を持つ．

SGNN は高速学習と大規模問題への適用可能性の優れた特性があるが，分類精度は BP 法のような教師あり学習が実装されるフィードフォワード型ネットワークに比べ劣る．そこで我々は SGNN の高速学習特性を利用し，与えられた訓練データからより高い分類精度を引き出すため，K 個の SGNT による ESGNN を考える．SGNT の構造は学習中に動的に変化する．SGNT アルゴリズムはすべての訓練データが加えられた後に木構造を決定する．異なる木構造をもつ SGNT は訓練データの入力順を入れ換えることで生成される．以後，訓練データを入れ換えて作成した入力データを用いる手法を “shuffling” とよぶ．本研究では，shuffling と bootstrap サンプリングに基づく bagging [8] を用いた ESGNN による実験を行う．

訓練過程において，N 個の要素からなる訓練データセット $\mathcal{D}$ のデータは shuffling と bagging を用いて K セットずつ用意され，それぞれ K 個の SGNT からなる ESGNN を生成する．テスト過程において，M 個の要素からなるテストデータセット $\mathcal{T}$ の実例ベクトルはこれらの ESGNN に入力される．各 SGNT のその実例ベクトルに対する葉のもつクラスの多数決を取り，最も多かったクラスをそのテストデータに対する出力とする．

Input :

- A set of training examples $\mathcal{D} = \{\boldsymbol{x}_i, y_i\}$, $i = 1, \ldots, N$.
- A threshold value $\xi \geq 0$.
- A distance measure $d(\boldsymbol{x}_i, \boldsymbol{w}_j)$.

Method :

```
1  copy(n1,x1);
2  for  (i = 2, j = 2; i <= N; i++) {
3    n_win = choose(x_i, n1);
4    minDistance = distance(x_i, w_win);
5    if  (minDistance > ξ) {
6      if (leaf(n_win))  {
7        copy(n_j, n_win);
8        connect(n_j, n_win);
9        j++;
10     }
11     copy(n_j, x_i);
12     connect(n_j, n_win);
13     j++;
14   }
15   update(x_i, w_win)
16 }
```

Output : Constructed SGNT by $\mathcal{D}$.

図 1 SGNT 生成アルゴリズム

表 1 SGNT 生成で用いられる副手続き

Sub procedure	Specification
copy(n_j,$\boldsymbol{x}_i$)	Create n_j, copy $\boldsymbol{x}_i$ as $\boldsymbol{w}_j$ in n_j.
distance($\boldsymbol{x}_i$,$\boldsymbol{w}_j$)	Compute $d(\boldsymbol{x}_i, \boldsymbol{w}_j)$.
choose($\boldsymbol{x}_i$,n_1)	Decide n_{win} for $\boldsymbol{x}_i$.
leaf(n_{win})	Check n_{win} whether it is a leaf.
connect(n_j,n_{win})	Connect n_j as a child of n_{win}.
update($\boldsymbol{x}_i$, $\boldsymbol{w}_j$)	Update $\boldsymbol{w}_j$ of n_j.

3. 自己生成ニューラルネットワークの枝刈り

ESGNN は単一の予測器に比べ，複数の予測器を用いることで未学習データに対する解の精度を改善することが可能であるが，アンサンブルに使用する SGNN の数が増加するにつれて計算時間，記憶容量を消費する．そこで，分類問題に対する ESGNN の記憶容量を削減するため，SGNN の枝刈りを行う．

全データが提示された後，木の枝刈りを行う．木の生成は訓練データを根から葉の方向へトップダウンに学習を行うが，枝刈りは葉から根に向かってボトムアップに行う．本手法はマージフェーズと枯葉削除フェーズの 2 つのステップを交互に行う．以下各段階について説明する．

マージフェーズでは，木の深さ優先探索により段階的にマージする葉を決定する．もし各階層中で同一の親をもつ部分木で全ての葉のクラスが一致すれば，その親ノードにそれらのクラスを与え，新たな葉としてマージし，その階層内の葉を全て刈り取る．この作業を根へ向かって再帰的に行うことで入力空間内で同一クラスの密度の高い部分の冗長なデータを削減する（図 2）．

枯葉削除フェーズでは，マージフェーズで単純化された SGNT の全ての葉のもつ実例データを木の根から再度入力する．もしそのデータが元の葉と同じ場所に到達しないのであれば，元の葉を枯葉とみなし削除する（図 3）．枯葉が無くなるまでマージフェーズと枯葉削除フェーズを繰り返すことで枝刈り前の SGNT の分類能力を維持しながら記憶容量を削減する．

```
1 begin initialize j = the height of the SGNT
2   do for each subtree's leaves in the height j
3     if all leaves belong to the same class,
4     then merge all leaves to parent node
5     if all subtrees are traversed in the height j,
6     then j ← j − 1
7   until j = 0
8 end.
```

図 2 マージアルゴリズム

```
1 begin initialize j = 0, N =# of leaves on the
merged tree
2   do j ← j + 1; for each attribute x_j
3     search the nearest leaf of x_j
4     if the nearest leaf differ from the original leaf,
5     then prune the original leaf as the dead leaf
6   until j = N
7 end.
```

図 3 枯葉削除アルゴリズム

4. 実験結果

SGNN，K 個の SGNN を用いた ESGNN，最近傍法，そして最近傍法の拡張で，与えられた訓練データの中から最も近いもの上位 k 個のクラスの多数決を出力とする k-最近傍法の計算コストと解の精度を算出する．各手法に対して，UCI 機械学習レポジトリ [7] の 10 のデータセットを shuffling と bagging を用いて実験を行った．ここで，shuffling は訓練データの順序をランダムに入れ換え必ず一度学習に用いる．これに対して，bagging [8] は重複を許す復元抽出により訓練データ数と同数のデータを訓練データセットからサンプリングする．各モデルの分類精度の評価法として，10-fold cross-validation を用いた．各モデルで使用する距離測度として，次に示す修正ユークリッド距離測度を使用した：

$$d(\boldsymbol{x}_i, \boldsymbol{w}_j) = \sqrt{\sum_{k=1}^{m} a_i \cdot (x_{ik} - w_{jk})^2}\ , \qquad (2)$$

ここで，m は属性数を，x_{ik} と w_{jk} は m 次元ベクトル $\boldsymbol{x}_i$ と $\boldsymbol{w}_j$ の k 番目の属性である．また，a_i は i 番目の属性に対する重み係数である：

$$a_i = \frac{1}{\max_i - \min_i}\ , \qquad (3)$$

ここで，$\max_i$ と $\min_i$ はそれぞれ i 番目の属性における最大値と最小値である．アンサンブル学習に使用する SGNN の数 K を 25 とし，k-NN で用いる近傍点の数 k を 3 とした．なお，全てのアルゴリズムは C で実装し，数値実験に PC-AT 互換機のパーソナルコンピュータ (CPU: Intel Pentium II 450MHz, Memory: 322MB, OS: Linux 2.2.18) を使用した．

表 2 に SGNN と最近傍法に shuffling と bagging を用いた場合の 10 回の平均計算コストを示す．計算コストとして，平均記憶容量と計算時間を算出した．ここで，平均記憶容量は枝刈り前の SGNN の平均ノード数を 1 とした場合の割合として表しており，0 に近い程高い削減がなされたことを意味する．また，計算時間は 10 回の試行の訓練とテストに必要な総処理時間を表している（単位は秒）．表 2 より，接点ノードが増加するために枝刈りを行う前の SGNN は最近傍法よりも大きな記憶容量を必要としているが，枝刈りを行うことで枝刈り前の 30% から 90% の記憶容量を削減し，接点ノードと葉を合計したものでも全数記憶方式の記憶容量を下回ることがわかる．shuffling と bagging で比較すると，bagging の方が高いノード削減がなされている．これは，重複するデータが枯葉となり枝刈りがなされるためである．また，計算時間に関しては木を生成，枝刈りし，その木を用いて分類を行う必要があるものの，学習を必要としない最近傍法よりも短時間で処理ができている．これは，最近傍法では学習サンプルが N 個存在する場合，各テストデータに対して N 回比較を行う必要があるのに対して SGNN は $\sum_i P_i$ 回でよいためである．ここで，P_i は第 i 層におけるノード数である ($P_i \ll N$)．実際，データ数 20000，各属性

表 2 shuffled SGNN，bagged SGNN，最近傍法に対する 10 回の試行における平均記憶容量と計算時間 (s)

	shuffled SGNN		bagged SGNN		nearest neighbor	
Dataset	memory	time(s)	memory	time(s)	memory	time(s)
balance-scale	0.384	0.08	0.304	0.07	0.664	0.4
breast-cancer-w	0.125	0.11	0.093	0.10	0.703	0.63
glass	0.582	0.03	0.459	0.02	0.654	0.05
ionosphere	0.423	0.10	0.299	0.09	0.697	0.26
iris	0.196	0.02	0.153	0.01	0.657	0.03
letter	0.290	15.91	0.239	14.14	0.677	979.23
liver-disorders	0.690	0.06	0.483	0.05	0.670	0.15
new-thyroid	0.270	0.02	0.235	0.02	0.654	0.05
pima-diabetes	0.570	0.15	0.428	0.14	0.672	0.72
wine	0.238	0.02	0.179	0.01	0.703	0.05
Average	0.376	1.65	0.282	1.47	0.675	98.16

表 3 SGNN, shuffled ESGNN (SH)，bagged ESGNN (BG), 最近傍法（1-NN），3-最近傍法（3-NN）に対する 10 回の試行における平均分類精度（括弧内は標準偏差を示す）

Dataset	SGNN	ESGNN(SH)	ESGNN(BG)	1-NN	3-NN
balance-scale	0.781(0.053)	0.843(0.059)	**0.850**(0.047)	0.771(0.057)	0.816(0.049)
breast-cancer-w	0.954(0.020)	0.967(0.023)	**0.973**(0.017)	0.954(0.025)	0.963(0.024)
glass	0.632(0.102)	0.692(0.075)	**0.702**(0.060)	0.669(0.086)	0.697(0.078)
ionosphere	0.858(0.042)	**0.883**(0.043)	0.869(0.041)	0.872(0.034)	0.858(0.044)
iris	0.953(0.055)	**0.967**(0.047)	0.960(0.047)	0.960(0.047)	0.960(0.047)
letter	0.893(0.007)	0.958(0.003)	0.955(0.003)	**0.960**(0.004)	0.956(0.005)
liver-disorders	0.574(0.086)	**0.635**(0.055)	0.614(0.051)	0.626(0.066)	0.623(0.064)
new-thyroid	0.944(0.070)	0.953(0.049)	**0.967**(0.050)	0.958(0.040)	0.940(0.063)
pima-diabetes	0.687(0.078)	0.711(0.070)	0.719(0.050)	0.692(0.084)	**0.737**(0.058)
wine	0.938(0.042)	**0.960**(0.039)	**0.960**(0.039)	0.949(0.032)	**0.960**(0.027)
Average	0.821	**0.856**	**0.856**	0.841	0.851

の次元数 16 の letter の問題に対しては計算時間において差が顕著に現れている．故に，本手法は大規模な問題に対して特に有効な手法であるといえる．

表 3 に SGNN，shuffling と bagging を用いた ESGNN，最近傍法，3-最近傍法の各問題に対する 10 回の試行における平均分類精度を示す．10 の問題に対して ESGNN は解の質を改善し，かつ最近傍法の結果より高い分類精度が得られている．shuffling の場合，ESGNN は最近傍法，3-NN 法と比べて，10 のデータセット中 7 つで上回っている．残りのデータセットに関しても同程度もしくは多少劣るという結果が得られている．また，bagging の場合は shuffling に比べ大幅に解の精度を改善し，10 のデータセット中 5 つで shuffling による ESGNN の解の精度を上回っている．平均では shuffling と bagging の ESGNN は同程度の解が算出されている．以上の結果より，shuffling と bagging の手法を比較すると，記憶容量を多く削減し，解の精度は同程度の bagging の方が ESGNN には適しているといえる．

5. まとめ

本論文では，ESGNN のための新しい枝刈り法を提案し，計算コストと解の精度を算出した．更に，二つのサンプリング法において既存の最近傍法，k-最近傍法と比較検討した．実験結果より，記憶容量が大幅に削減され，解の精度は最近傍法よりも高い精度が得られることを示した．そして，shuffling と bagging を用いた ESGNN の比較では，記憶容量の削減率と解の精度改善特性より，bagging を用いた ESGNN の方が優れていることがわかった．ESGNN は各 SGNN を独立に構築可能であるため，並列分散処理との親和性が高い．従って，本手法はデータマイニングに関して簡便かつ汎用性の高いモデルであるといえる．今後，ESGNN の更なる効率化のため，記憶容量を更に削減し，汎化能力を枝刈り前より改善する新たな枝刈り法を検討する予定である．

参考文献

[1] 松嶋敏泰, “情報論的学習理論での数理モデル,” 人工知能学会誌, vol. 16, no. 2, pp. 252–255, March 2001.

[2] S. Haykin, Neural Networks: A comprehensive foundation, Prentice-Hall, Upper Saddle River, NJ, 2nd ed., 1999.

[3] W. X. Wen, A. Jennings and H. Liu, “Learning a neural tree,” Proc. International Joint Conf. on Neural Networks, Beijing, China, 1992.

[4] T. Kohonen, Self-Organizing Maps, Springer-Verlag, Berlin, 1995.

[5] H. Inoue and H. Narihisa, “Performance of self-generating neural network applied to pattern recognition,” Proc. 5th International Conf. on Information Systems Analysis and Synthesis, vol. 5, Orlando, FL, pp. 608–614, Aug. 1999.

[6] 井上浩孝, 成久洋之, “アンサンブル自己生成ニューラルネットワークの性能分析,” 電子情報通信学会論文誌（A）, vol. J83-A, no. 10, pp. 1227–1230, Oct. 2000.

[7] C. Blake and C. Merz, “UCI repository of machine learning databases,” 1998.

[8] L. Breiman, “Bagging predictors,” Machine Learning, vol. 24, pp. 123–140, 1996.

[9] W. X. Wen, V. Pang and A. Jennings, “Self-generating vs. self-organizing, what’s different?,” Neural Networks Theory, Technology, and Applications ed. P. K. Simpson, IEEE Technology Update Series, IEEE Technical Activities Board, Piscataway, NJ, pp. 210–214, 1996.