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

2008－DPS－135 (29)
2008／6／20

グレーゾーン付きニューラルネットツリーの生成

林 博友[†1] 趙 強福[†1]

ニューラルネットツリー（NNTree）は各中間ノードに小規模なニューラルネット（NN）を埋め込んだ決定木（DT）である．NNTreeの利点としては，フル結合型NNと比べ構造学習及びハードウエア実現に適していることと，通常の単一変量DTと比べ汎化能力が高いことなどが挙げられる．しかし，その汎化能力は通常の階層型NNと同程度であり，サポートベクターマシンには及ばない．より汎化能力の高いNNTreeを生成するために，本論文で我々は新しいアルゴリズムを提案する．このアルゴリズムでは，各中間ノードに割り当てたNNにグレーゾーンを設定することにより，NNTreeを複数のパスで探索することで，高い認識率を目指す試みである．

Induction of Neural Network Trees with soft decision making

HIROTOMO HAYASHI[†1] and QIANGFU ZHAO[†1]

Neural network tree (NNTree) is a decision tree (DT) with each internal node containing a small neural network (NN). NNTree is a model good for structural learning and for hardware implementation. A generalization ability of NNTrees is comparable to that of a multi-layer perceptron, but it is worse than that of support vector machine. In this paper, we propose a new algorithm for inducing NNTrees which have more generalization ability. The basic idea is to make soft decision at each NN like a fuzzy rule. The efficiency of the proposed algorithm is evaluated through experiments on several public databases.

1. はじめに

ニューラルネットワーク（NN: Neural Network）とは脳神経系をモデルとした情報処理システムのことであり，文字認識，音声認識といったパターン認識やデータマイニング等さまざまな分野において応用されている．というのも，NNは高次元で，かつ線形分離不可能な問題に対してしばしば良好な解を得られるからである．しかしながら，NNの構造をどのように決定したらよいか事前に決めることは困難で現実的には試行錯誤になることが多く，扱いづらい面がある．

そこで，我々はニューラルネットツリー（NNTree: Neural Network Tree）について考察する（図1を参照）．NNTreeは，特殊な決定木（DT:Decision Tree）であり，各中間ノードに対応するテスト関数は小規模なNNによって実現される．NNTreeの利点としては，システムの構造は木の生成と共に自動的に決定されること等がある．

NNTreeを生成する際に，訓練データを十分に分割できるようになるまで木に小規模なNNを追加し枝を伸ばしていく．つまり，与えられた問題に対して良い解が得られるようになるまで木を成長させていくということである．このようにNNTreeの構造は生成する時に動的に決められるので試行錯誤する必要がなくなる．各小規模なNNについては，その構造を予め決めておいて，同じ構造を持つものを複数コピーし，結合係数だけを変更すれば利用可能となる．従って，本研究ではNNTreeの中のモジュールとして使われている小規模なNNの構造を決定することを問題視しない．

NNTreeの計算速度を考えると，任意のパターンを認識するために必要な計算量は木の平均レベル数に比例する．クラス数をN_cとすれば，木の平均レベル数は$log_2 N_c$と表せる．一方，階層型NNのニューロン数は通常N_cに比例するので，NNTreeと階層型NNの任意のパターンを識別するのに必要な計算量を比べると，NNTreeの方がオーダー的に速いと考えられる．

しかし，それらの汎化能力を比較した場合，MLPと同程度であり，一般的に汎化能力に優れていると知られているサポートベクターマシン（SVM: Support Vector Machine）には及ばない．本論文では，NNTreeの中間ノードに割り当てられたNNにファジィルールのような曖昧な判断を許すことでNNTree全体としての認識率を向上させることを目指す．曖昧な判断を行うためにグレーゾーンを設定する．NNの出力がグレーゾーンの範囲内であったら，そのNNでの判断は保留し子ノードを探索する．その有効性を複数のデー

†1 会津大学大学院コンピュータ理工学研究科
Graduate School of Computer Science and Engineering, The University of Aizu

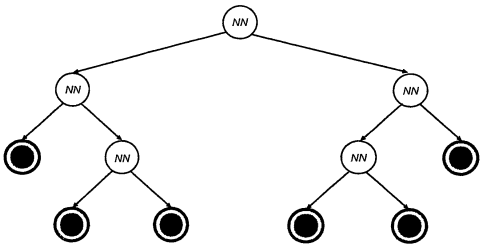

図 1 ニューラルネットツリーの例
Fig. 1 An example of neural network trees

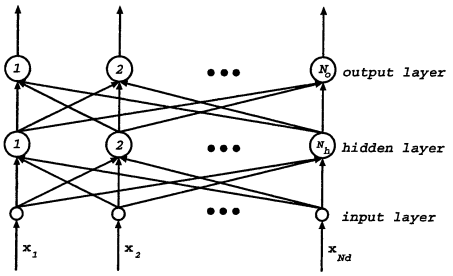

図 2 本研究に使われる NN の構造
Fig. 2 Structure of the NN used in this study

タベースを利用した実験で確認する．

本論文の構成は次の通りである．第 2 章では決定木の基礎知識と，我々が以前に提案した BP に基づく NNTree の生成方法について説明する．第 3 章はグレーゾーン付き NN を持つ NNTree の生成方法を紹介する．第 4 章では実験結果を示し，その性能評価する．第 5 章はまとめである．

2. 基礎知識

2.1 決定木の定義

決定木はサイクルのない有向グラフである．木の最初のノードはルートである．通常，決定木はルートが一番上にあるように描かれる（図 1 を参照）．各ノード（ルートを除いて）の上には一つのノードがあり，それは親ノートである．ノードの下にあるノードは子ノードである．子ノードがないノードは終端ノード（葉）であり，それ以外のものは中間（非終端）ノードである．本論文では，ノードを以下のように定義する:

$$node = \{I, F, Y, N, L\} \tag{1}$$

ただし，I はノードの識別番号，F はテスト関数，Y は子ノードへのポインターの集合，$N = |Y|$ は子ノードの数，L はノードのクラスラベルである．なお，ノートが中間ノードである場合，L は未定義である．また，ノードが終端ノードである場合，F は未定義で，かつ Y は空集合である．

決定木を使って未知のパターン $\boldsymbol{x}$ を認識する過程は以下のようになる:

- Step 1: ルートを初期の「現在ノード」とする．
- Step 2: 現在ノードが終端ノードである場合，現在ノードのラベルを $\boldsymbol{x}$ に与え，終了する．そうでなければ，テスト関数 $F(\boldsymbol{x})$ を使って，次に探索する子ノードを決める．
- Step 3: i (= $F(\boldsymbol{x})$) 番目の子ノードを現在ノードとして，Step 2 に戻る．

2.2 決定木の生成

決定木を生成するためには，まず訓練集合を用意する．決定木は訓練集合を再帰的に分割することによって生成される．生成過程において主にノードの分割，終端ノードの判断，終端ノードのクラスラベルの決定の 3 つの操作がある．その中で最も重要であり，かつ時間が掛かるものはノードの分割である．ノードを分割するために良いテスト関数を見つける必要がある．テスト関数の「良さ」を評価する基準がいくつか提案されている[1)2)] が，生成された決定木の性能はテスト関数の評価基準にあまり影響されないことが知られている[2)]．本研究において我々は情報利得率（IGR: Information Gain Ratio）を採用する．IGR はもともと Quinlan により提案され，良く知られている決定木生成プログラムである C4.5 で使われている[1)]．

IGR を最大にするようなテスト関数 $F(\boldsymbol{x})$ は，現在ノードに割り当てたデータを $F(\boldsymbol{x})$ を基にして分割すると，未知のデータを認識するための平均情報量は最小となる．いま，S は現在ノードに割り当てたデータの集合（$|S|$ はそのサイズである）であり，n_i は i 番目のクラスに属するデータの数（$i = 1, 2, \cdots, N_c$, N_C はクラス数である）であるとする．未知のデータを認識するための平均情報量（エントロピー）は以下のようになる：

$$info(S) = -\sum_{i=1}^{N_c} \frac{n_i}{|S|} \times \log_2(\frac{n_i}{|S|}). \tag{2}$$

S をテスト関数 F によって N 個の部分集合 $S_1, S_2, \cdots, S_N$ に分割したとする．このときの情報利得は

$$gain(F) = info(S) - info_F(S) \tag{3}$$

で与える．但し，

$$info_F(S) = \sum_{i=1}^{N} \frac{|S_i|}{|S|} \times info(S_i). \tag{4}$$

情報利得率（IGR）を以下のように定義する:

$$gain\ ratio(F) = gain(F)/split\ info(F) \tag{5}$$

但し，

$$split\ info(F) = -\sum_{i=1}^{N} \frac{|S_i|}{|S|} \times \log_2(\frac{|S_i|}{|S|}). \tag{6}$$

2.3 NNTrees の定義とその生成方法

前述のように，NNTree は多変量決定木の一種であり，各中間ノードに使われているテスト関数は小規模

なNNによって実現される．この小規模なNNの構造については特に制限はないが，本研究では，図2に示すような階層型ニューラルネットを使う．通常，NNの入力数と出力数はそれぞれ特徴の数 N_d と子ノードの数 N に対応するが，本研究の主旨は複数の小規模なNNを決定木に組み込むことにより複雑な問題を解決することであるため，本論文ではNNの隠れニューロン数 $N_h = 4$ とする．

NNTreeを利用する場合，未知のパターン $\boldsymbol{x}$ の認識過程は以下のようになる．

- Step 1: ルートを初期の「現在ノード」とする．
- Step 2: 現在ノードが終端ノードである場合，現在ノードのラベルを $\boldsymbol{x}$ に与え，終了する．そうでなければ，
$$i = F(\boldsymbol{x}) = \arg \max_{1 \le k \le N} o_k \qquad (7)$$
を用いて次に探索する子ノードを見つける．ここで o_k はNNの k 番目の出力である．
- Step 3: i 番目の子ノードを現在ノードとして，Step 2に戻る．

NNTreeを生成するために，C4.5を生成するプロセスとほぼ同じプロセスが利用できる．唯一の相違点は，テスト関数 $F(\boldsymbol{x})$ の求め方である．NNTreeの場合，テスト関数はNNであるため，通常の「生成と評価」に基づく方法はあまり効率的ではない．そこで，我々は，まず各中間ノードに割り当てるデータのグループラベル（教師信号）を発見的手法で定義し，教師付き学習アルゴリズムを使って多変数テスト関数を求める方法を提案した[4]．この手法により高速にNNTreeを生成することが可能となった．遺伝的アルゴリズム（GA）に基づく方法でも，個体を各ノードのNNの重み係数とすることでNNTreeの生成は可能である[3]が膨大な時間が必要とされるので本論文では割愛する．

NNTreeの生成方法は図3のフローチャートのようになる．各処理をアルゴリズムの形で説明すると以下のようになる．

- Step 1: 現在ノードが終端ノードであるか判定する．その判定基準は現在ノードに割り当てられたデータが複数のクラスを含むかどうかである．単一のクラスであった場合，終端ノードと判定し，Step 2へ行き，複数クラスを含むのであれば，中間ノードと判定し，Step3へ行く．
- Step 2: 現在ノードに割り当てられたデータで最も数の多いクラスをこの終端ノードに割り当て，このパスの処理を終了する
- Step 3: 現在ノードに割り当てられたデータをどちらの子ノードに割り当てたらよいかを決める．その結果はBPの教師信号として使用する．

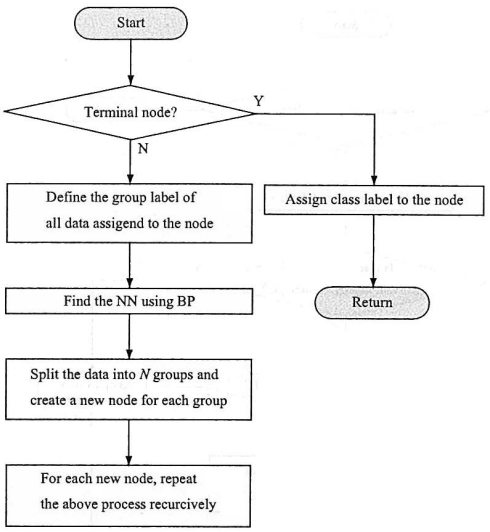

図3 提案方法に基づくNNTreeを生成するフローチャート
Fig. 3 Flowchart for inducing the NNTrees using the proposed method

- Step 4: Step3で得られた教師信号を元に，BPアルゴリズムでNNを設計する．
- Step 5: 現在ノードに割り当てられた全てのデータをStep 4で設計したNNに入力させ，NNの出力に従ってN個のサブセットを求める．各サブセットに対応して一つの子ノードを作る．
- Step 6: 現在ノードの子ノードを現在ノードとして同様の処理を再帰的に行う．

次にデータの集合を任意の複数の集合に分けるの方法[5]はを図3のフローチャートのようになり，各処理をアルゴリズムの形で説明すると以下のようになる．

S を現在ノードに割り当てられたデータの集合であるとする．S を N 個のサブセット $S_1, S_2, \cdots, S_N$ に分けたいと仮定する．各サブセットの初期状態は空集合である．S が空集合になるまで，以下のことを繰り返す:

- Step 1: 任意のデータ $\boldsymbol{x}$ を S から取りだし，
- Step 2: もし S_i に $label(\boldsymbol{y}) = label(\boldsymbol{x})$ を満たす $\boldsymbol{y}$ が存在するならば，$\boldsymbol{x}$ を S_i に割り当てる．そうでないならば，Step3に移動する．
- Step 3: もし空集合の S_i が存在するならば，$\boldsymbol{x}$ を S_i に割り当てる．そうでないならば，Step4に移動する．
- Step 4: $\cup S_i$ の中で $\boldsymbol{x}$ の最近傍である $\boldsymbol{y}$ を見つけ，$\boldsymbol{x}$ を $\boldsymbol{y}$ と同じサブセットに割り当てる．

任意のデータ $\boldsymbol{x}$ の教師信号は，それが割り当てられ

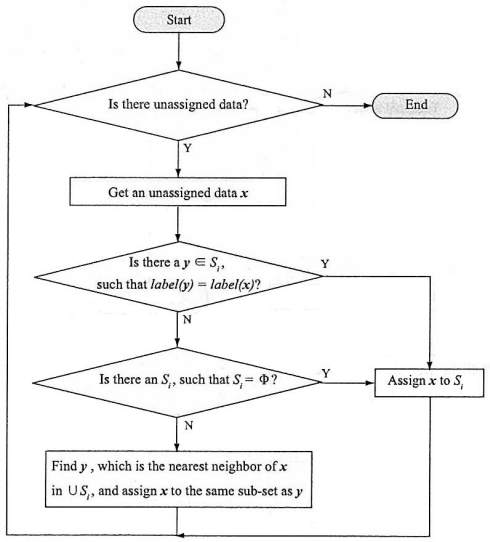

図 4 グループラベルを割り当てるために提案した方法のフローチャート

Fig. 4 Flowchart of the proposed method for assigning group labels

たサブセットの番号になる．即ち，$\boldsymbol{x}$ が S_i に割り当てれた場合，その教師信号は i となる．すべてのデータの教師信号がわかると，テスト関数を実現する NN を求める問題は教師付き学習となる．本研究では，学習のために良く知られている誤差逆伝搬法（BP）を使用する．

3. グレーゾーン付き NN を持つ NNTree の生成

NNTree は中間ノードに割り当てられた NN の出力に従い，子ノードを決めるわけであるが，これまでの判定方法は必ず 1 つの子ノードに決めるものであった．NN の出力値にはっきりした違いがない場合，わずかな差に従って子ノードを決定してきた．各出力の差が僅しかない場合，認識するデータにわずかにノイズが乗っているだけでもその出力を誤る可能性があると考えられる．そこで，我々は NN に曖昧な判断をすることを許す．曖昧な判断とは，あるノードの NN で，あるデータに対しての判断を一旦保留し，先に進む．子ノードを探索した後で，探索されたパスを総合的に評価する．NN で曖昧な判断をするために，NN の出力の特定の範囲をグレーゾーンと設定する．テストとトレーニングの場合に分けて説明する．

テストの場合，あるデータに対して NN の出力がグレーゾーンの範囲内であったらすべての子ノードを探索する．この場合，各中間ノードでグレーゾーン判定をするので複数のパスで終端ノードに達する．最終的判断を行うため各 NN の出力を信頼度として記憶しておいて，すべてのパスの探索が終了した時点で終端ノードに達したパスの平均の信頼度が最も高かった終端ノードのラベルを最終的な判断として使う．

トレーニングの場合，通常子ノードにデータを割り当てる時，訓練データは NN の出力に従ってデータを子ノードに割り振るが，ある訓練データの NN の出力値がグレーゾーンの範囲内であったら，複数の子ノードにそのデータを割り当てるものとする．これまでは 1 つの訓練データが複数の子ノードに割り当てられることはなかったが複数の子ノードに割り当てることで有益な情報を子ノードに伝達できると考えられる．

次にグレーゾーンの判定方法について説明する．グレーゾーンの判定方法は 2 通り考えらる．1 つ目は，NN の出力ニューロン数がクラス数と同数ある場合で，出力は winner-take-all に基づき出力が最大のニューロンとなるがグレーゾーンを考える場合，最大出力ニューロンと他の出力ニューロンの出力値の差を考える．この差がある閾値よりも小さければグレーゾーンにあると判定する．2 つ目は NNTree が 2 分木でかつ NN の出力数が 1 の場合である．2 分木であるならば，理論上出力ニューロンは 1 つでも 2 分木を構築可能である．この場合，NN の出力範囲の中央値を中心としたある範囲内（閾値）に NN の出力値があるときにグレーゾーンにあると判定する．本論文では出力ニューロン数 $N_o = 1$ とし，後者の方法によって実装する．その認識過程は以下のようになる．

- Step 1: ルートを初期の「現在ノード」とする．
- Step 2: 現在ノードが終端ノードである場合，信頼度と現在ノードのラベルをメモリに保存して，このパスの探索を終了する．そうでなければ，

$$i = F(\boldsymbol{x}) = \begin{cases} 1 & (o \leq 0.5 - \theta) \\ 1, 2 & (0.5 - \theta < o < 0.5 + \theta) \\ 2 & (0.5 + \theta \leq o) \end{cases} \tag{8}$$

  を用いて次に探索する子ノードを見つける．ここで o は NN の出力値，θ は閾値である．
- Step 3: i 番目の子ノードを現在ノードとして，Step 2 に戻る．
- Step 4: すべてのパスの探索が終了したら，信頼度が最も高かった終端ノードのラベルを $\boldsymbol{x}$ に与え終了する．

---

4. 実験と考察

本論文で提案した方法の有効性を確認するために，カルフォルニア大学アーヴァイン校の公開データベースから得たデータベースを使い，実験を行った．使用したデータベースは car, crx, dermatology, ecoli, housevotes84, ionosphere, iris, optdigits, pen-based,

表 1　データベースのパラメーター
Table 1　Parameters of the Databases

	Number of examples (N_t)	Number of features (N_d)	Number of classes (N_c)
car	1728	6	4
crx	690	15	2
dermatology	366	34	6
ecoli	336	7	8
housevotes84	435	16	2
ionosphere	351	34	2
iris	150	4	3
optdigits	5620	64	10
pen-based	10992	16	10
tic-tac-toe	958	9	2

tic-tac-toe の 10 個である．表 1 にデータベースのパラメーターを示す．すべての実験において，10-fold クロスバリデーション（cross validation）を行い，それを 5 回を行った．実験に使用したコンピュータはサンワークステーションの SunJavaWorkstationW1100z モデル，1CPU，1.8GHz AMD Optron 144 である．ここでは，次の NNTree の各方法と BP で得られた階層型 NN（BP-MLP）について比較する．NNTree は次の 4 つの方法について行った．

方法（1） トレーニング時，テスト時両方共グレーゾーン判定を行わない

方法（2） トレーニング時にグレーゾーン判定を行わず，テスト時のみグレーゾーン判定を行う

方法（3） トレーニング時のみグレーゾーン判定を行い，テスト時はグレーゾーン判定を行わない

方法（4） トレーニングとテスト両方にグレーゾーン判定を行う

実験で使用したパラメータを示す．NNTree のパラメータは次の通りである．1) BP に使う学習率=0.5, 2) 学習回数（エポック数）=1,000, 3) NN の入力数=N_d（特徴数），4) 隠れニューロン数=4, 5) 出力ニューロン数=1, 6) グレーゾーン判定の閾値=0.05．次に BP に基づくフル結合型 NN を設計する際のパラメータは，1) 入力数=N_d, 2) 出力ニューロン数=N_c（クラス数），3) 学習回数=1,000, 4) 学習率=0.5．これらは実験を通して得たパラメータ値である．

表 2，3 に 10 個のデータベースに対する実験結果を示す．これらの表では，学習後のテストセットに対するグレーゾーンなしのエラー率（%）(: Error Rate1)，グレーゾーン有りのエラー率（%)(: Error Rate2)，(すべてのノードを含む）木のサイズ，学習に要した計算時間（秒）を示す．各項目 2 段で表している数値は，上段の値が平均値であり，下段が 95%の信頼区間である．また，テスト時のグレーゾーン判定の有り無しに関してはそれぞれ同じ NNTree で実験を行っているため 1 つの表にまとめた．エラー率の対応関係について，表 2 の Error Rate1 は方法（1）と対応し，Error Rate2 は方法（2）と対応する．また，表 3 の Error Rate1 は方法（3）と，Error Rate2 は方法（4）と対応する．

（1） まず，テスト時のみにグレーゾーン判定を行った場合（方法（2））と行わなかった場合（方法（1））を比較する．表 2 の見ると，7 勝 3 敗で方法（1）の NNTree の方が平均エラー率が小さい結果となったが，エラー率について T 検定を行った結果，有意差は認められなかった（$p > 0.05$）．

（2） 次に，トレーニング時にグレーゾーン判定を行った場合（方法（4））と行わなかった場合（方法（3））の結果について見る．表 3 より，8 勝 1 敗 1 分けで方法（3）の NNTree の方が平均エラー率が小さい結果となった．これについても T 検定を行ったが，有意差は認められなかった（$p > 0.05$）．

（3） トレーニング時にグレーゾーン判定を行った場合（方法（3））と行わなかった場合（方法（1））を比較する．表 2，3 より，方法（1）と方法（3）のエラー率を見ると，6 勝 3 敗 1 分けで方法（1）の平均エラー率が小さい結果となった．これらのエラー率について T 検定を行った結果，有意差は認められなかった（$p > 0.05$）．一方，計算時間はにレーゾーン判定を行わない方が 8 勝 2 敗であり，T 検定を行った結果，4 つのデータベースで有意差が認められた（$p < 0.05$）．これはトレーニング時にグレーゾーン判定を行うと訓練データが増えるためであると考えられる．

（4） 以上，NNTree の 4 つの方法の結果を見てきた．その結果から，有意差は認められなかったがグレーゾーン判定は行わない方が平均エラー率は小さくなる傾向にあるといえる．また，同じような結果であったのでここには掲載しないが閾値を 0.15 まで 0.05 刻みで実験を行っている．グレーゾーンの範囲内であると判定される回数は増えるものの閾値を 0.05 の場合と同様，有効に作用していなかったように思われる．NN の出力値をそのまま使用しているが，それを信頼度として使用することが誤っていることが考えられる．

（5） 最後に階層型 NN（BP-MLP）とトレーニング時にのみグレーゾーン判定を行う方法（3）で生成された NNTree を比較する．表 3，4 のエラー率を見ると，7 勝 3 敗で階層型 NN の方がエラー率で小さい結果となった．T 検定を行った結果，階層型 NN の optdigits と方法（3）の pen-based に有意差が認められた（$p < 0.05$）が，残りの 8 つについては有意差は認められなかった．

表 2　トレーニング時にグレーゾーン判定を行わない場合
Table 2　Results of NNTree with soft decision at test only

	Error Rate1	Error Rate2	Tree Size	Time
car	2.81 ± 0.76	2.86 ± 0.77	16.04 ± 1.34	2.06 ± 0.18
crx	16.99 ± 2.12	16.7 ± 2.07	14.08 ± 1.07	3.73 ± 0.2
dermatology	3.83 ± 1.20	4.39 ± 1.23	15.56 ± 0.6	0.1 ± 0.02
ecoli	15.58 ± 2.56	15.88 ± 2.47	16.68 ± 0.92	1.01 ± 0.06
housevotes84	5.35 ± 1.28	5.44 ± 1.28	5.12 ± 0.4	0.38 ± 0.06
ionosphere	8.97 ± 1.87	8.91 ± 1.92	4.96 ± 0.36	0.59 ± 0.08
iris	3.60 ± 1.46	3.87 ± 1.47	5.48 ± 0.28	0.15 ± 0.02
optdigits	4.13 ± 0.84	4.15 ± 0.84	28.32 ± 1.71	1.68 ± 0.15
pen-based	2.63 ± 0.66	2.67 ± 0.67	33.68 ± 2.52	4.45 ± 0.72
tic-tac-toe	0.86 ± 0.43	0.82 ± 0.42	5 ± 0.18	0.13 ± 0.02

表 3　トレーニング時にグレーゾーン判定を行う場合
Table 3　Results of NNTree with soft decision at training

	Error Rate1	Error Rate2	Tree Size	Time
car	2.49 ± 0.71	2.53 ± 0.7	15.92 ± 1.34	2.12 ± 0.26
crx	17.04 ± 2.04	17.07 ± 2.04	14.28 ± 0.95	3.63 ± 0.16
dermatology	3.72 ± 1.13	4.17 ± 1.14	15.12 ± 0.48	0.11 ± 0.02
ecoli	15.39 ± 2.40	15.15 ± 2.42	14.12 ± 0.81	1 ± 0.07
housevotes84	5.12 ± 1.31	5.16 ± 1.33	5.64 ± 0.57	0.43 ± 0.06
ionosphere	9.37 ± 1.88	9.49 ± 1.9	5.28 ± 0.4	0.62 ± 0.09
iris	4.27 ± 1.61	4.27 ± 1.61	5.56 ± 0.29	0.17 ± 0.02
optdigits	4.26 ±0.88	4.30 ± 0.88	29.36 ± 1.65	2.34 ± 0.34
pen-based	2.80 ± 0.69	2.80 ± 0.69	30.64 ± 1.59	9.12 ± 1.82
tic-tac-toe	0.86 ± 0.46	0.91 ± 0.49	5.04 ± 0.36	0.19 ± 0.05

5. おわりに

本論文では，ニューラルネットツリー（NNTree）の汎化能力を向上させてるため，NNTree の中間ノードの NN にグレーゾーンを設定して学習，判別を行う方法を提案した．しかしながら，実験の結果を見ると性能が向上したとは言えず，むしろ悪くなっている傾向にある．今後さらなる改善で性能を向上させることができないか検討したい．

表 4　BP 学習により得られたフル結合型 NN の結果
Table 4　Results of fully connected NNs obtained through BP learning

	Error Rate (%)	Number of Hidden Neuron	Time (second)
car	2.21 ± 0.37	28	1.47 ± 0.10
crx	15.74 ± 1.43	32	6.53 ± 0.03
dermatology	3.00 ± 0.64	24	0.17 ± 0.02
ecoli	14.91 ± 1.58	28	3.83 ± 0
housevotes84	5.21 ± 0.98	8	0.28 ± 0.09
ionosphere	8.4 ± 1.27	8	0.53 ± 0.11
iris	3.47 ± 1.31	8	0.28 ± 0.03
optdigits	2.15 ± 0.20	60	5.32 ± 0.14
pen-based	4.57 ± 0.19	60	292.33 ± 0.97
tic-tac-toe	1.05 ± 0.38	8	0.04 ± 0.01

謝辞　本研究の一部は日本学術振興会科学研究費補助金（No. 19500128）の補助を得て行われたものである．

参考文献

1) Quinlan, J.: *C4. 5: programs for machine learning*, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA (1993).
2) Breiman, L., Friedman, J. H., Olshen, R. A. and Stong, C.J.: *Classification and Regression Trees*, Wadsworth Pub. Co. (1984).
3) Zhao, Q.: Evolutionary design of neural network tree - integration of decisiontree, neural network and GA, *Proc. IEEE Congress on Evolutionary Computation*, pp.240–244 (2001).
4) Hayashi, H. and Zhao, Q.: A Comparative Study on GA Based and BP Based Induction of Neural Network Trees, *IEEE International Conference on Systems, Man and Cybernetics*, pp.822–826 (2005).
5) Zhao, Q.: A New Method for Efficient Design of Neural Network Trees, *Technical Report of IEICE*, Vol.PRMU2004-115, pp.59–64 (2004).