岡山理科大学紀要 第36号 A pp. 137-146(2000)

進化的アルゴリズムによるニューラルネットワークの最適化

西山　美智子・井上　浩孝*・成久　洋之**

岡山理科大学大学院工学研究科修士課程情報工学専攻
*岡山理科大学大学院工学研究科博士課程システム科学専攻
**岡山理科大学工学部情報工学科

(2000年11月1日　受理)

1　まえがき

ニューラルネットワーク（Neural Network; 以下NN）は生物の神経細胞系をモデルにしたものであり，分類や認識などの知的処理に有効な方法と考えられている．NNの中で最も利用されているモデルは，階層型前向きネットワーク（feedforward NN）であり，その学習則は一般に誤差逆伝播法（Back Propagation;BP）が使用されている．ところがこのBP則に基づいた学習の収束性はNNの構造に依存し，そのNN構造を利用者の勘と経験によって決定している状況である．本研究では，進化的アルゴリズム（Evolutionary Algorithm;EA）に代表される遺伝的アルゴリズム（Genetic Algorithm;GA）と進化的プログラミング（Evolutionary Programming;EP）を用いたNN構造の最適化特性について検討するものである．

2　BPアルゴリズムの概要

階層型NNの学習アルゴリズムであるBPについて説明する．図1にニューラルネットワークの構成を示す．一般的に n 層のFF-NNにおいて，入力パターン p とすると，k 層 j 番目のユニットにおける入力データ i_{pj}^k と出力データ o_{pj}^k は

$$o_{pj}^k = f(i_{pj}^k) \tag{2.1}$$

$$i_{pj}^k = \sum_i w_{i,j}^{k-1,k} o_{pi}^{k-1} \tag{2.2}$$

と定義される．$w_{i,j}^{k-1,k}$ は結合重みである．$w_{i,j}^{k-1,k}$ は，$(k-1)$ 層目の i 番目のユニットから k 層目の j 番目のユニット間の結合荷重を意味している．$f()$ は活性化関数（activation function）であり，本研究では各ユニットの出力関数として以下に示すシグモイド関数を使用する．シグモイド関数とは，$\pm\infty$ で飽和値を取る単調増加で微分可能な関数の総称である．

$$f(x) = \frac{1}{1+\exp(-\epsilon x)} \tag{2.3}$$

ϵ は結合荷重の収束性に影響を与えるゲインと呼ばれる係数である．
ここでいう学習とは，入力データ i_{pj}^k をネットワークに加え，活性度伝播により計算した目標データ t_{pi} が与えられたとき，出力データ o_{pi}^n と目標データ t_{pi} ができるだけ一致するよう，即ち，次式で与えられる誤差関数 E が最小になるように結合荷重を調整することである．

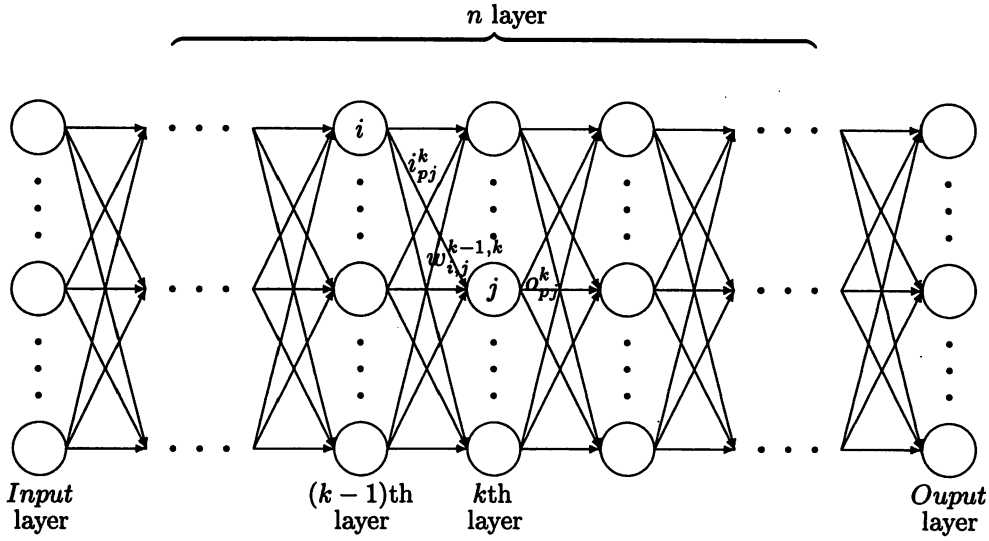

図1　ニューラルネットワークの構成

$$E = \frac{1}{2N_P}\sum_P\sum_i (t_{pi} - o_{pi}^n)^2 \tag{2.4}$$

N_P は入力パターン数である．BP の特徴は，最急降下法において，(2.4) 式によって定義され，これにより各結合荷重を微少変化させた場合の評価値の変化量，即ち最小二乗誤差和の各結合荷重に関する感度を求めることが可能になる．大抵，最小二乗誤差（Mean Square Error;M.S.E.）は評価関数として利用される．誤差 E に対する結合荷重による偏微分が常に負となるように更新すれば誤差 E を漸近的に小さくすることができる．故に適応度が小さいほど誤差が小さくなり最良の解を求めることができるといえる．結合荷重を更新する為のアルゴリズムは以下のとおりである．

$$\Delta_p w_{i,j}^{k-1,k}(m) = \eta\delta_{pj}^{k-1}o_{pj}^{k-1} + \alpha\Delta_p w_{i,j}^{k-1.k}(m-1) \qquad (k = 1,2,\cdots,n) \tag{2.5}$$

$$\delta_{pj}^n = (t_{pj} - o_{pj}^n)\frac{\partial}{\partial i_{pj}^n} f(i_{pj}^n) \tag{2.6}$$

$$\delta_{pj}^k = \frac{\partial f(i_{pj}^k)}{\partial i_{pj}^k}\sum_s \delta_{ps}^{k+1} w_{j,s}^{k,k+1}(m-1) \qquad (k = 1,2,\cdots,n-1) \tag{2.7}$$

η は学習係数であり，η を小さくすることで収束の改善が行われる．だからといって η は小さすぎると学習に長時間を要し非効率であり，逆に大きすぎると振動し上手く学習できない．一般には振動しない範囲でできるだけ大きな η を用いるとよい．α はモーメンタムと呼ばれる 1 回前の変更量に対する定数であり，$0 < \alpha < 1$ で振動を抑えることができる．α が 1 に近い程，振動はより抑制される．次に，m は現在の位置を示し，$\Delta_p w_{i,j}^{k-1.k}(m-1)$ は一回前の重み修正量である．(2.5) 式において $\alpha\Delta_p w_{i,j}^{k-1.k}(m-1)$ は慣性項と呼ばれ，収束時間を減少させるために導入されたものである．多峰性関数の凹凸を無視する効果を発揮する為，学習速度を大きくできる．

BP の基本である最急降下法は，局所解に陥るため最適な探索ではない．そもそもその問題の影響を緩和するために BP 法は開発されたのである．BP 法では学習回数を低減するために学習係数を設定するので，η と α の組合せを調整することにより，最急降下法の問題点であった学習回数の増加と局所解へのトラップを若干ではあるが低減できることになる．

3 BP への進化的アルゴリズム (EA) の適用

EA と NN の融合を NN の観点から見ると，

(1) NN の学習に EA を適用するもの
(2) NN の構造の決定に EA を適用するもの

との二種類がある．(1) は NN の学習が数学的には主に最適化であるため，例えば BP 法の代わりに EA を適用するものである．この場合は EA を単なる最適化の一手法として用い，局所解（ローカルミニマム）にトラップされにくいという特長を利用している．EA は最適解の近傍にまでは非常に早く収束することができるが，そこから最適解への探索が苦手である為，EA を利用して最適な結合荷重（以下，単に重みと呼ぶ）の近くまで学習し，そこから BP に切り替えて最適解まで持って行くことを意図している．

BP 則では NN における初期重みはランダムにセットされる．しかしながら EA で学習させれば個体レベルで学習して実行されていると考えられているので，ランダム性による弊害は除去され，より本質的な融合が行われるといえる．(2) は今，実現しようとしている機能に最適な NN の構造を EA で決定した後，その構造のもとで通常の NN の学習を行うものである．これは従来，NN の設計者により経験的にしか決めることのできなかった NN の構造設計を自動化しようとするものである．

いずれにしても重みや前章で述べたような学習パラメータを決定するには試行錯誤が必要であるから，これらもまた NN 構造を決定する際に同時に更新させていく．

3.1 遺伝的アルゴリズム (GA) の概要

GA とは自然界における生物の進化過程を数学的にモデル化することにより，生物の持つ環境への適応能力を取り扱おうとするものである．その数学モデルは遺伝子における交叉，突然変異，個体における適者生存を基本原理としている．

NN を GA で探索するアプローチは，遺伝子型へのコーディング法により直接コーディング法を使用する．これは NN のユニットや結合に関する情報を表現型とし，それらの情報が直接的に遺伝子列にコーディングされる方法であるが，遺伝子型から表現型への変換が容易である反面，ユニット数や結合数が多いと遺伝子列が長くなり探索空間が大きくなり易い．またネットワークサイズが予め決まっているものにしか適用できない．そこで本研究では，同じ遺伝子の長さを持つ個体の集団を一つの種族とみなし，同じ種族間同士だけで交叉・突然変異を行うようにする．各々の種族の評価は，種族毎に適応度の平均を算出し，より良い適応度の平均を持つ種族を次世代へ残すようにする．(1) については重みを 2 進数表現したものと実数表現したものとを考える．

3.2 進化的プログラミング (EP) の概要

GA が交叉・突然変異によって子孫生成して解の更新をさせていくのに対して，EP とは突然変異のみで進化させて行く方法である．そして全ての個体集合に対して突然変異を行い，親と子の双方から次世代の集合を選択する．EP と GA が決定的に異なる点はそこである．EP において突然変異を行う場合，解の更新において乱数分布を使用し，その標準偏差によって解の更新ステップ幅を規制する一種のランダムサーチ法を用いる．使用する乱数分布によって解の収束度・精度がかなり変化するため，EP の精度は使用する乱数分布に依存することが分かる．分布によっては局所解に陥ることもある．乱数分布は正規 (Gaussian) 分布，対数正規 (LogNomal) 分布，正規分布とコーシ (Cauthy) 分布を併用したものとがあるが，今回は中でも最も一般的に利用されている正規分布を使用することにした．

3.2.1 正規分布による突然変異

各個体 (x_i, η_i) について正規分布に従った突然変異による子孫 (x_i', η_i') 生成を以下のように行う.

$$x_i'(j) = x_i(j) + \eta_i(j)N(0,1) \tag{3.1}$$
$$\eta_i'(j) = \eta_i(j) + \alpha\eta_i(j)N(0,1) \tag{3.2}$$

ここで $x_i(j), x_i'(j), \eta_i(j), \eta_i'(j)$ は n 次元実数ベクトル対 $(x_i, \eta_i), (x_i', \eta_i')$ の j 番目のベクトル値とする. $N(0,1)$ は平均 0 , 標準偏差 1 の正規乱数とする. また α は正規分布による解の更新時のスケール要素であり, 0.3 に固定している.

上式において $\sigma = \eta_i(j)$ とおくと, 正規分布 $N(x_i(j), \sigma^2)$ つまり平均 $x_i(j)$ 標準偏差 $\eta_i(j)^2$ の正規分布に従う正規乱数を発生する. よって子孫 $x_i'(j)$ は, 平均 (正規分布の面積) の約 97.8% を占める為, 生成された子孫 $x_i'(j)$ は 99.7% の確率で $-3\eta_i(j) + x_i(j) \leq x_i'(j) \leq 3\eta_i(j) + x_i(j)$ を満たす. このことより, 子孫 $x_i'(j)$ は勝ち残ってきた親の値を損ねることなく受け継ぐことが可能となる. ここで解の更新ステップ幅である $\eta_i(j)$ の値が大きければ $x_i'(j)$ も大きく更新されるし, $\eta_i(j)$ が小さければ $x_i'(j)$ の更新幅は小さくなる. 前者の場合, 局所解が多い時はそこから抜け出すのには優位であるが的を絞り込むのには不利であり, 後者には逆のことがいえる. よって $\eta_i(j)$ の値をどのように更新するかが EP で問題を解く上で重要となる.

4 構造最適化手法における個体の定義

一つの NN は一つの個体で定義される. 異なった NN 構造を持つ NN は異なる個体を持つだけでなく同じ NN 構造を持つ個体でさえ異なる個体を持つ. なぜなら各 NN に対して重みは可変であるからである.

これらの NN 構造は染色体で符号化される. 以下に表現した NN の個体を示す.

gene1	gene2	gene3	gene4	gene5	gene6
ϵ	η	H	α	w_1	w_2

図 2 NN の個体表現

ϵ : coefficient of activation function
η : learning rate
H : unit numbers of Hidden layer
α : coefficient of momentum term
w_1 : connection weights from Input to Hidden layer
w_2 : connection weights from Hidden to Output leyer

中身の情報は構造・学習パラメータを表現した. gene3 は NN 構造自身を表現している為, gene3 で決定された NN 構造によって可変である gene5, gene6 は長さが決定される. つまり gene5, gene6 は gene3 で与えられた NN 構造によって制限される. 一方 gene1, gene2, gene4 は NN の性能に影響する非常に重要な値である. それ故にこれらの値も組み込むことにより個体を表現した.

同じ NN 構造をした個体の集団を同種族として考えることにする. 図 3 に個体集団生成の手順を示す. 前述したように, NN 構造が異なると個体の染色体自体の長さが異なるため, 異なる種族間では進化させることは不可能である. 同種族間のみの土俵で EA を適用する以外に方法はない. 以下に子孫生成アルゴリズムを記述する.

step1 初期集団をランダムに生成する.

step2 同じ NN 構造を持つ個体を種族別にする.

step3 各個体の適応度を求め各種族の平均適応度を算出し, 種族間で評価する.

step4 最良の平均適応度を持つ種族を次世代に残す．**step2** へ戻る．

但し，**step2** に戻る際に選ばれた種族を含むそれ以下の小さな構造を持つ種族を生成する場合，選ばれた種族より長さが短いので生成可能であるが，選ばれた種族より大きい構造を持つ種族を生成する場合，長さが足りないので生成不可能である．よってその場合は前世代でその種族の平均適応度より良い個体をそのまま使用する．それでも個体数不足（前種族の個体を使い果たした，或いは前世代に同種族の個体は存在しなかった）の場合，**step1** と同様にランダムに生成する．

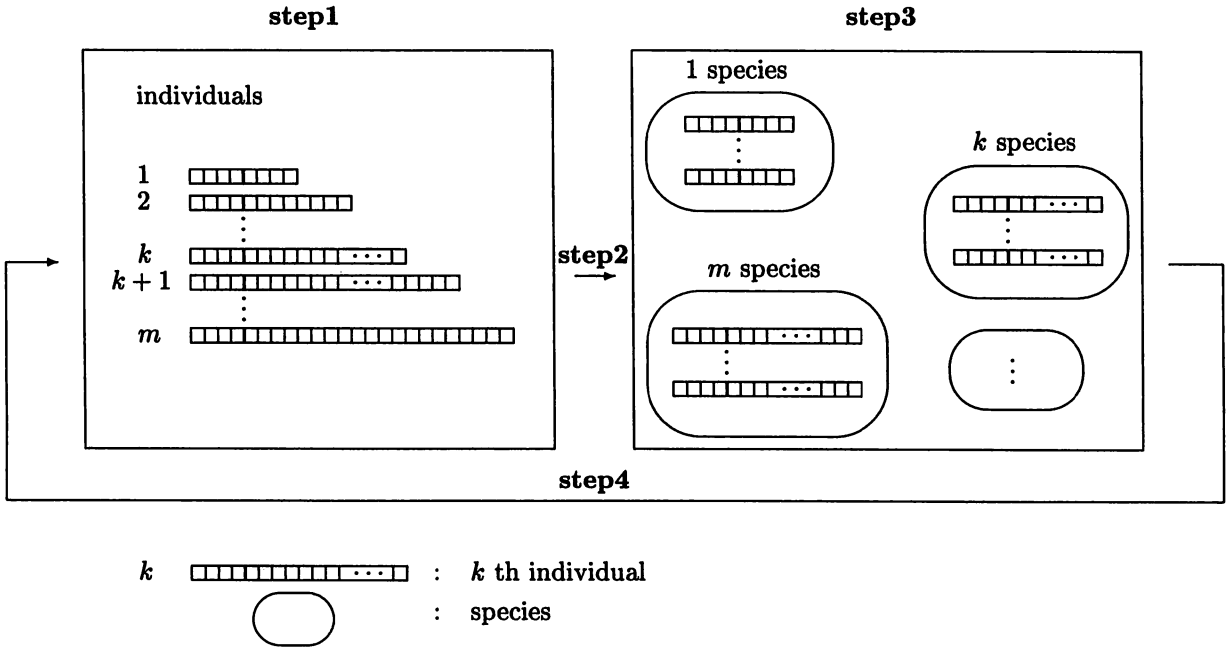

図 3　個体集団生成の手順

5　実験方法

研究対象とする問題として，問題規模の小さな排他的論理和 (EXOR) の問題，文字認識問題を取り扱い，各々の問題に対して，BP 単独の場合と (1),(2) のように EA を適用した後 BP の手法で処理する場合との比較により EA 適用の有効性について研究した．文字認識問題については，0〜9 の数字 (SUUJI) 認識，26 文字の A〜Z のアルファベット (ALPHABET) 認識を行った．GA についての重みの表現方法としては，2 進数表現したものと最初から実数表現したものとを比較し，また EA については構造自体を組み込み表現したものとを比較検討した．このときの GA においては個体を 2 進数で表現している．EA である程度まで精度を上げて，その後 NN に切り替えるタイミングなども調査した．

各々の問題に対して以下のようなパラメータを使用した．表 1 に各々の問題に対して使用したパターン数，入力層数，隠れ層数，出力層数を示す．表 2 に各々の問題に対して (2) のみに使用した個体で表現した学習・構造パラメータを示す．表 3 に全問題に共通して使用した個体数，交叉率，突然変異率，EA の繰り返し回数，NN の繰り返し回数，個体の良さの尺度を判定する為の NN の繰り返し回数である．

表 1　BP 単独，(1),(2) 共通の使用パラメータの個数

	Pattern	Input	Hidden	Output
EXOR	4	3	3	1
SUUJI	10	50	50	10
ALPHABET	26	101	101	26

表 2 (2) のみの使用パラメータの範囲

	Hidden	ϵ	η	α	w_1	w_2
EXOR	[1, 16]	[0, 4]	[0, 4]	[0, 1]	[−1, 1]	[−0.1, 0.1]
SUUJI	[1, 32]	[0, 4]	[0, 4]	[0, 1]	[−1, 1]	[−0.1, 0.1]
ALPHABET	[1, 32]	[0, 4]	[0, 4]	[0, 1]	[−1, 1]	[−0.1, 0.1]

表 3 各々の問題に共通する BP 単独，(1),(2) 共通の使用パラメータ

population(num)	pc(%)	pm(%)	g_t(times)	NN_t(times)	nn_t(times)
30	0.5	0.8	10	3000	30

使用コンピュータは NEC Versa Pro NX VA 23D(Pentium II 233MHz)．交叉は一様交叉を使用した．

6 結果

各々の問題に対して BP 単独の場合，GA による (1)，(2) の場合についての収束特性を図 4〜9，表 4〜12 に示した．図の内容は，縦軸は M.S.E. による誤差，横軸は NN の繰り返し回数とする．GA における (1) については重みを 2 進数表現したもの (GA_weight) を採用した．GA における (2) については構造を最適化したもの (GA_structure) である．EP における (2) については構造を最適化したもの (EP_structure) である．表の内容は，行を上から順に説明すると，最良の収束回数／平均収束回数，EA の処理時間，収束終了までの処理時間，NN 繰り返し終了までの処理時間，収束率，EA の減少率，最終誤差である．GA における (1) については重みを 2 進数表現したもの (weight2) と実数表現したもの (weight10) とを採用した．GA における (2) については構造を最適化したもの (GA_structure) である．EP における (2) については構造を最適化したもの (EP_structure) である．各々の問題に対して実行回数はすべて 30 回の試行とし，収束の終了条件は 0.0001 とした．

7 考察

EXOR 問題において表 4,5,6 より，全ての方法において M.S.E. を見ると完全に収束したことが分かる．しかし図 4,5 を見ると分かる様に，BP 単独の場合は収束速度が遅いのは一目瞭然である．それは比較的規模の小さい EXOR のような問題に対してはネットワークが局所的最小に落ち込んでしまう為，収束するのは難しいからである．(2) の構造を最適化した場合，他の方法と比べて所要時間はかかるが，それは構造を決定する時間を圧倒的に要しているからであるのは明らかである．BP 単独の場合は収束に時間を要するが，EA で要した時間を BP 単独で収束するまで繰り返した場合，どちらが有効であるかは考えなければならない点である．しかし現時点では，時間は例えかかろうとも収束回数の早い方が有効であるとみなす．

SUUJI 認識問題において表 7,8,9 より，(1) の重みを最適化した場合のみ収束率が高い．それに比べて他の方法では収束率が悪いどころか最終平均誤差も悪く，特に BP 単独の場合では殆ど収束しなかった．GA による問題点として指摘されるのは，計算の初期段階で個体群中の個体の多様性が失われる初期収束 (premature convergence) 現象である．探索空間の広さはこの多様性によって決まるので初期収束を避ける為の工夫が必要となってくる．単純には pc と pm，特に pm を比較的大きな値に設定しておくことによってこの初期収束現象の可能性を低減させることができるが，同時に適応度の高い個体が消滅し易くなってしまうのが難点である．(2) の構造を最適化したものの収束率は極端に悪いので，その設定法は有効であるかもしれない．図 6,7 を見ると，どのグラフも似たり寄ったりであると思われがちだが，(1) の重み自体を最適化する場合 (特に 2 進数) は，収束率が高い上に EA 減少率も高く，(2) の構造を最適化する場合に比べて有効である．

ALPHABET 認識問題において表 10,11,12 より，(1) の重みを最適化した場合では収束率が圧倒的に高く，M.S.E. と収束率から言えるのは GA は有効であるということである．BP 単独の場合の欠点は，大きい問題に対しては収束しないことである．(2) の構造を最適化した場合の最良収束回数は，明らかに少ない NN の繰り返し回数で収束することができた．ここで ALPHABET 認識問題における (2) の場合，メモリ不足に因りパラメータの変化に限度があった為，個体数を増やせば EA 回数は減らさざるを得ないし，EA 回数を増やそうと思えば個体数を減少させるしかなく，ここで表記しているのは両方を平均した結果である．

換言すると，EAを1回しか適用していないというのに高い収束率を得ることができた.(図8,9参照) しかし良い結果ばかりが得られる訳ではなく悪影響はEAの誤差減少率に出た．小規模のEXOR問題に比べ大規模のALPHABET認識問題では構造を決定する為に試行錯誤している際に，逆に誤差が大きくなっていることが分かる．EAを繰り返す度に誤差は良くなるとは限らなかったのである．しかし逆に考えれば，大規模の問題では，あまりGAを適用しなくても収束が可能ではないかといえる．

一方，膨大な計算時間が記載以上にかかるのはいうまでもない．それはNN構造の規模の増大と共に学習所要時間の問題は深刻化するからである．つまり評価の微分情報により重みを修正する最急降下法を用いているため，学習の際の評価関数が最小値に達せず極小値に留まるという極小値問題は，問題・ネットワーク規模の増大と共に深刻になるのである．また解が最適点に近付くと傾斜が小さくなり，毎回の重みの修正量が小さくなるため学習速度が遅くなるのである．だからといって，ネットワークの大きさが過小であると十分学習できず，逆に過大であると過剰学習のため汎化能力が減退する．それに加えて，より大きくより複雑なネットワークでは，ネットワークが学習中に最小化を行う表面がより大きくなってしまう．これにより今度はネットワーク自体が局所的最小点に落ち込んでしまう．従ってタスクに対応した適切な大きさのネットワークで学習することが重要である．

全体の最良の収束回数に目を向けると，(2)について全ての問題に対して小さな収束回数を得ることができた．しかし図4,5で表れたGA_structureの収束の悪さと表4,5の平均収束回数から分かるように，おそらく最適化して選ばれた構造が，うまく適用した場合もあればそうでなかった場合もあり，平均を取るとその差が激しい為に良いグラフが得られなかったのだと思われる．それはEXOR問題についてはあくまでも小さな問題だからであろう．またグラフに示された不安定な動きは，構造を決定した後にその構造が上手く適用しなかった為であり，個体を次世代に残す際に不足分をランダムに選んだからではないかと思われる．

GAはEPに比べて2進数で個体を表現している為，問題規模が大きくなるとGAオペレータに時間を要する．一方EPはGAに比べて，個体を進化させる際に親と子全てから評価する為，GAの場合と同じ個体数を設定したとしても，倍の個体数間で評価しなければならないので別の時間を要する．よって今後は(2)において，GAについては最初から実数表現した個体を使い，EPについては範囲内におさまらなかった子は除外して考えるなどして工夫したい．

8 むすび

本研究では，EAを用いたNN構造の最適化特性について検討した．実験結果より，BP単独の場合に比べEAを適用した場合の方が学習収束特性に関して有効であることが分かった．しかし本研究は現在発展途上の段階であって，BPだけではなくEAにも考察で述べたような不具合があり，まだ改善すべき点は沢山ある．構造を決定する際に今回は最良の適応度を持つ種族を選び出したが，それだけでなく例えば全種族の適応度の平均より良い種族を考え，その各種族からも子孫を継承していき，全種族で枝分かれ的に子孫を残すよう工夫したい．しかしそれでは今回悩まされたメモリがまた不足してしまうので，より要領良いメモリ領域の確保を考慮して進化のさせ方，個体表現の仕方などを改善したい．個体数，EAの繰り返し回数，隠れ層の数などを限定し，最適化した構造パラメータを導入してBP単独で処理させたら面白いかもしれない．他には，最初から大きめのネットワーク構造から出発し，不要なユニットや結合を逐次削除して適切な大きさのネットワーク構造を求める削除的学習法を考えている．以上を課題として今後研究していきたい．

参考文献

[1] E Vonk, L C Jain, R P johnson: Automatic Generation of Neural Network Architecture Using Evolutionary Computation, World Scientific, 1997

[2] A J F Rooji, L C Jain, R P johnson:Neural Network Training Using Genetic Algorithms, World Scientific, 1996

[3] 松岡清利:ニューロコンピューティングー基礎と応用ー, 朝倉書店, 1992

[4] Judith E. Dayhoff 桂井浩 訳:ニューラルネットワークアーキテクチャ入門, 森北出版 (株), 1992

[5] 臼井支朗 他:基礎と実践ニューラルネットワーク, コロナ社, 1998

[6] H. Takahashi and M. Nakajima: Evolutional Design and Training Algorithm for Feedforward Neural Networks, IEICE TRANS. INF. & SYST., VOL.E82-D, NO.10, pp.1384-1392, 1999

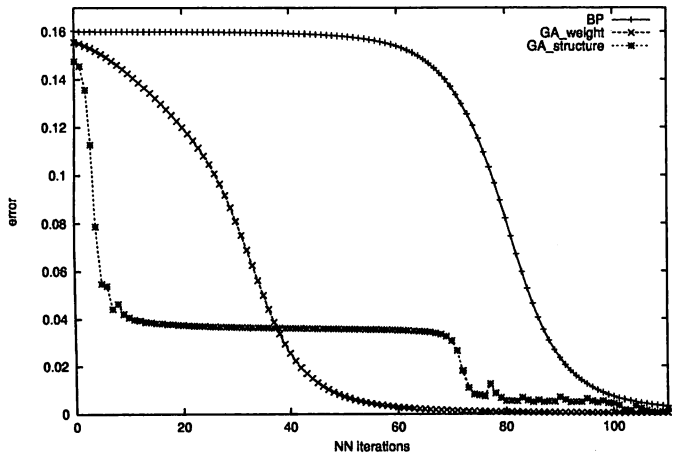

図4 EXOR (GA)

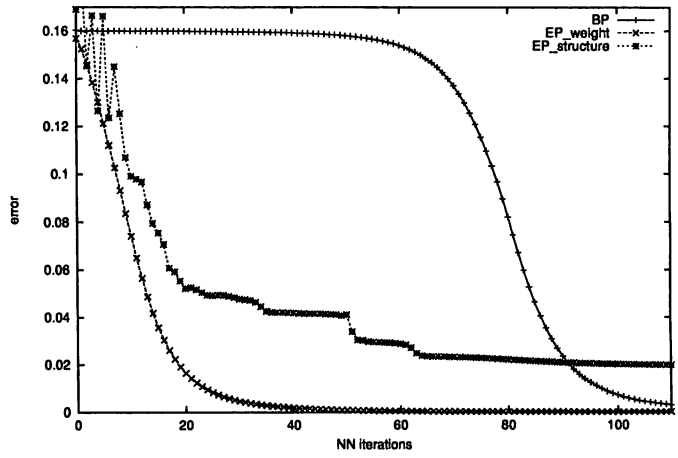

図5 EXOR (EP)

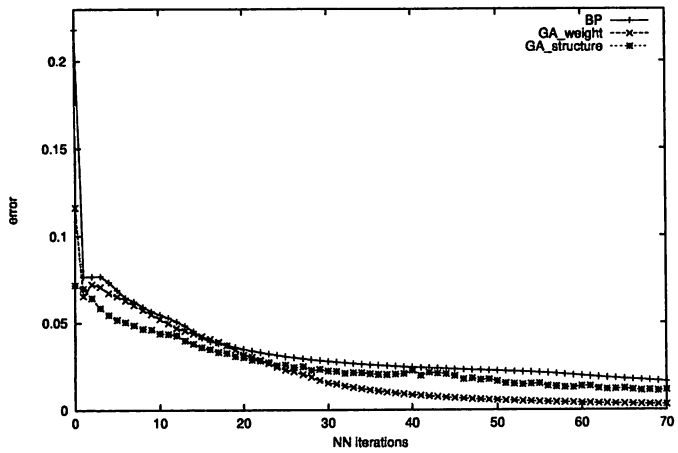

図6 SUUJI (GA)

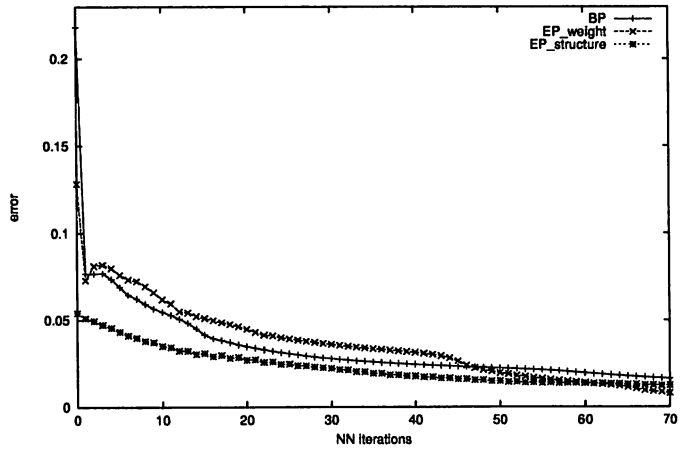

図7 SUUJI (EP)

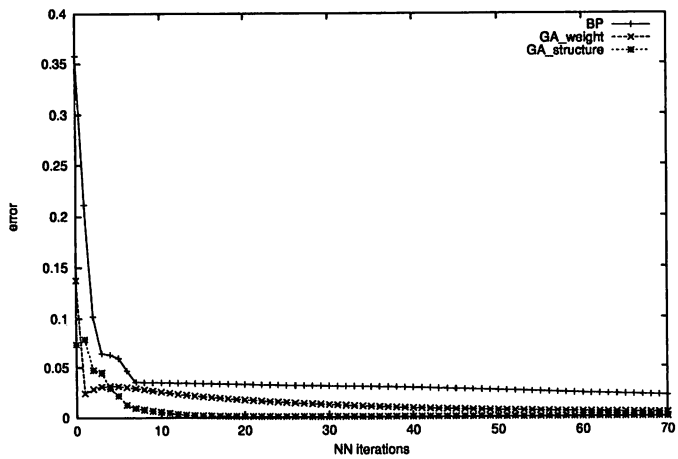

図8 ALPHABET (GA)

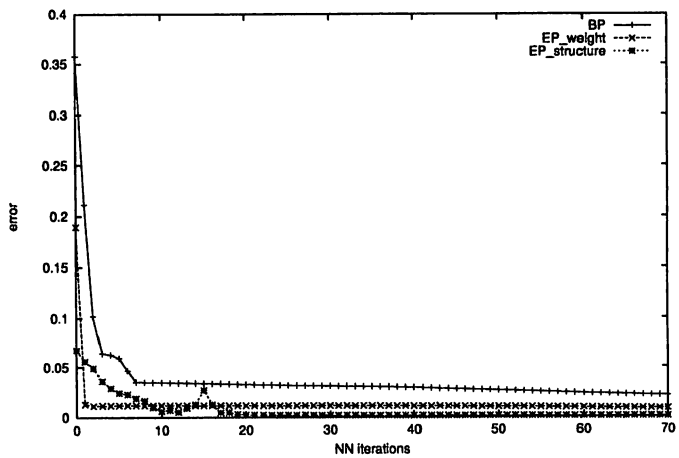

図9 ALPHABET (EP)

表 4 EXOR(GA)

	GA_structure	weight2	weight10
convergence number(times)*1	62/298	96/278	83/554
GA or EP_over(s)	1.98	0.0511	0.1275
convergence time(s)	2.50	0.16	0.1775
process time(s)	3.19	0.26	0.35
convergence_ratio(%)	88.2	80.0	44.4
switching_ratio(%)	6.66	1.01	4.3
M.S.E.*3	0	0	0

表 5 EXOR(EP)

EP_structure	weight
58/407	61/152
2.72	0.115
3.03	0.298
3.86	0.540
10.0	38.0
56.4	34.3
0	0

表 6 EXOR(BP)

BP
190/465
*
0.32
0.45
63.3
*
0

表 7 SUUJI(GA)

	GA_structure	weight2	weight10
convergence number(times)*1	58/407	583/1899	318/833
GA or EP_over(s)	58.19	7.78	8.494
convergence time(s)	112.8	31.19	17.38
process time(s)	132.66	42.62	39.90
cinvergene_ratio(%)	10	66.6	100
switching_ratio(%)	56.4	79.94	73.62
M.S.E.*3	0.000195	0	0

表 8 SUUJI(EP)

EP_structure	weight
58/407	333/757
98.5	10.618
122.13	16.948
126.69	33.553
11.3	13.5
1.2	73.62
0.000365	0

表 9 SUUJI(BP)

BP
1847/2768
*
no convergence
36.7
0
*
0.0011039

表 10 ALP(GA)*2

	GA_structure	weight2	weight10
convergence number(times)*1	50/123	195/1978	356/880
GA or EP_over(s)	92.75	53.6	21.23
convergennce time(s)	159.25	229.23	118.47
process time(s)	190.29	334.7	245.3
convergence_ratio(%)	88.8	100.0	67.45
switching_ratio(%)	-95.0	78.0	78.0
M.S.E.*3	0	0	0

表 11 ALP(EP)*2

EP_structure	weight
49/109	1605/2040
55.50	50.849
63.64	165.227
165.43	257.316
33.3	56.6
34.8	6.82
0	0.0000608

表 12 ALP(BP)*2

BP
1972/2374
*
86.4
107.43
26.9
*
0.000104

*1 convergence number においては，最良の収束回数／平均収束回数
*2 但し，(2) における ALPHABET 問題ではプログラムによるメモリ不足の為，パラメータの変化に限度があったので時間は記載以上に要するものと思われる．
*3 M.S.E について，収束の終了条件は 0.0001 とした．

Optimization of Neural Network Architecture Using Evolutionary Algorithms

Michiko NISHIYAMA, Hirotaka INOUE* and Hiroyuki NARIHISA**

Graduate School of Engineering
**Doctor Program of System Science,*
Graduate School of Engineering
***Department of Information & Computer Engineering,*
Faculty of Engineering,
Okayama University of Science,
Ridaicho 1-1, Okayama 700-0005, Japan

(Received November 1, 2000)

Neural networks are constructed by using many artificial neurons as an imitation of biological nervous system and have been known as an effective method for intelligent process such as classification and pattern recognition. The most famous model in neural networks is feedforward type neural networks, whose training rules generally use backpropagation method. However, convergence of training based on backpropagation rules is greatly influenced by the neural network architecture and training parameters setting. Usually, neural network architecture and training parameters are decided by trial and error based on its user's knowledge, experience and empirical observations. In this paper, we investigate optimization of neural network architecture using evolutionary algorithms.