平成 15 年度

卒業論文

ニューラルネットワークによる演算

信州大学理学部物理科学科卒業論文

中垣友介

平成 16 年 3 月 1 日

目次

序章

本研究では、ニューラルネットワークによる処理の実用性について調べるため、C 言語を用いた階層型ネットワークのシミュレーションを行った。

ニューラルネットワークの一般的な特徴を簡単にまとめてみると以下のようになる。

- 処理が非常に速い。
- ダメージに対して耐性がある。
- ネットワーク全体で記憶をすることが出来る。
- プログラムではなく学習に基づいて処理をする。
- 学習された以上のことを推測できる。

本研究では特に、最後の“ 学習されていないことを推測する ”能力について注目した。答えが学習済みのものだけを判断しても実用性はならないからである。ニューラルネットワークの動作は入力の仕方に大きく関わるため、どのような入力にすれば上手く“ 推測する ” ことが出来るようになるのか考えていきたい。

また、詳しくは後述するが、各ユニットに 0,1 以外の数も対応させることが出来るかについても調べる。

本文に入る前に、各章の内容を以下にまとめておく。

第 1 章では、予備知識として、ヒトのニューロンの構造について簡単にまとめた。

第 2 章では、人工のニューラルネットのモデルについて構造、動作を説明することにする。

そして、第 3 章から実際にシミュレーションの結果について述べることにする。まず、第 3 章では AND, OR, XOR といった論理演算についてのシミュレーションを行った。

第 4 章では、多数入力についての研究のため、パターン認識についてのシミュレーションを行った。

第 5 章では、階層型ネットワークによる演算を行い、1 つのユニットに 0,1 以外の数も対応させられるかを調べた。また、“ 推測 ” についても調べることにする。

第１章　　ニューラルネットワークとは

人の脳は、質量約 1.3 キログラム、容積約 1.4 リットルほどで、その中に 1000 億ものニューロンが互いに複雑に結合しながら詰まっている。

この構造を元に人工のニューラルネットワークモデルは作られている。

第 1 章ではヒトのニューロンが、どの様な働きをしているかを説明し、以後の章の理解に役立てることにする。

１．１　ニューロンの構造

ニューロンの簡単な構造を図 1.1 に示す。

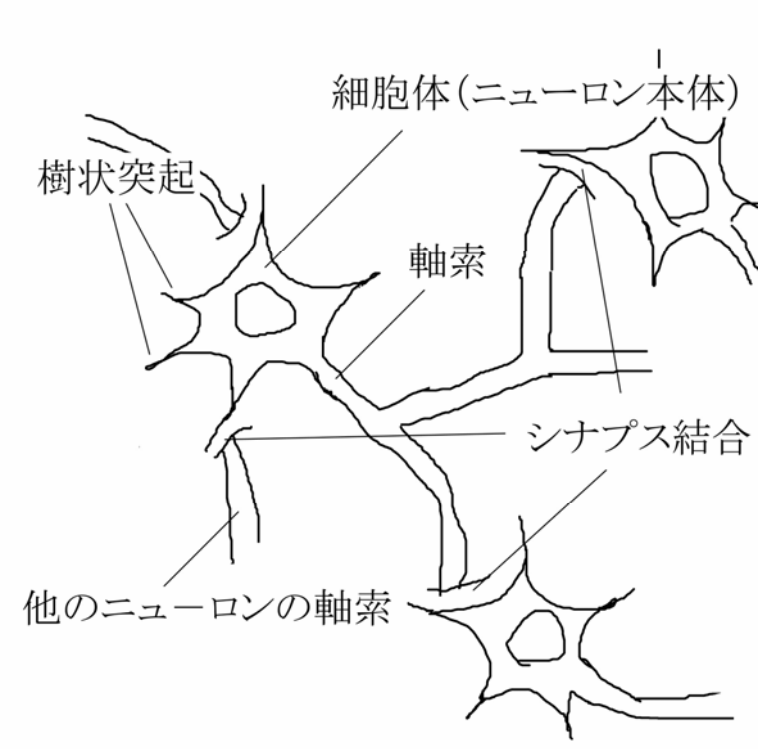


図 1.1　ニューロンの構造

ニューロンは細胞体、樹状突起、軸索からなっている。これらについて簡単に説明する。

・ 細胞体　　核などが含まれているニューロン本体といえる部分。

・ 樹状突起　細胞体から多数伸びている枝状の部分。ニューロンの出力端子にあた

る。

・ 軸索　　細胞体から一定の太さで長く伸びた 1 本の突起。途中で数多く枝分かれする。ニューロンの出力端子にあたる。

軸索と他のニューロンの樹状突起との結合はシナプス結合と呼ばれる。また、シナプス結合には非常に狭い隙間が開いている。

１．２ ニューロンの働き

軸索から電気信号が入るとシナプスで神経(化学)伝達物質が放出される。神経伝達物質がシナプスの隙間を超えて伝わると、再び電気信号に変換され次のニューロンへ信号が伝わっていく。シナプスの重要な働きは、電気信号を化学信号に変換する際に、入力信号に重み付けを与え、伝達効率を変えることである。

電気信号の伝達は、刺激がある一定値を越えた時に電位変化が起こり、これが次々に神経膜に沿って伝わっていくことでおきる。具体的には、神経伝達物質を受け取る方のシナプス(シナプス後細胞)の膜電位が閾値を超えると、神経幕の破壊によるナトリウムイオンの流入が次々に起こることで、信号が伝わっていくのである。

これを発火という。この発火が起きるか起きないかで情報が伝わるため、情報の違いは電位の大きさでなく、発火の発生回数で決まる。

シナプスの重み付けは発火を促す興奮性もの（興奮性シナプス）と、それを抑える抑制性のもの（抑制性シナプス）がある。

第２章　　人工ニューラルネットワークの構造

２．１　　モデルニューロンの構造

第１章にあったヒトのニューロンの機能を単純化して捉えると、以下のようになる。

・他のニューロンの発火が、シナプス結合を通じて１つのニューロンの膜電位に変化を及ぼす。
・多数のシナプスからの影響の和によってシナプス後細胞の膜電位が決まる。
・膜電位がある閾値を超えると、ニューロンは発火する。

これらの条件を満たすような単純なモデルを図 2.1 に示す。

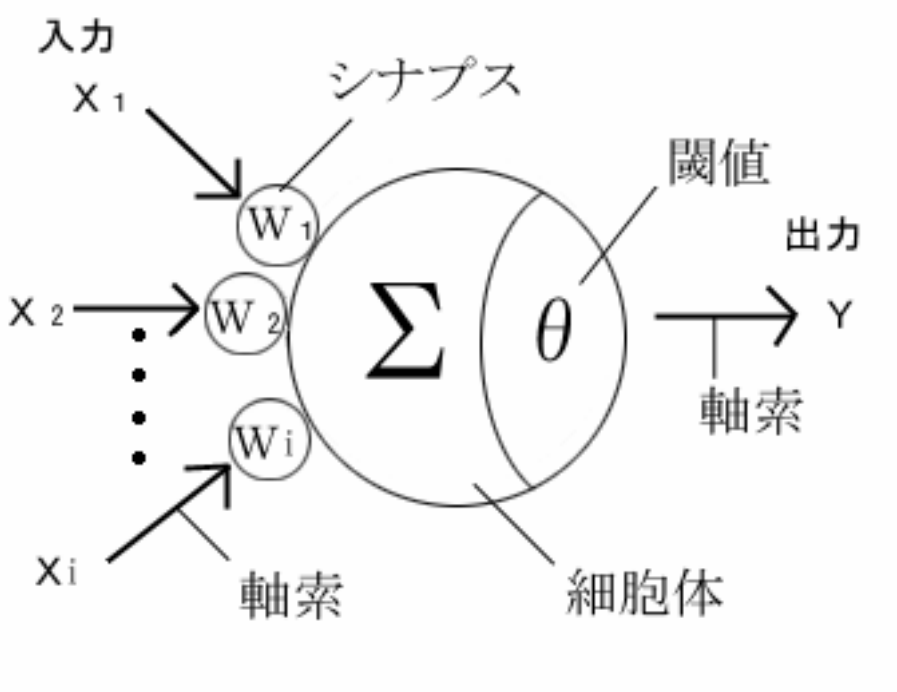


図 2.1　モデルニューロン

$$u = \sum W_i X_i - \theta \tag{2.1}$$

$$y = f(u) \tag{2.2}$$

Wi は i 番目のシナプス結合の強さ（結合加重）を示し、Wi が正であれば興奮性シナ

プス、負であれば抑制性シナプスを表し、結合が無いときには Wi＝0 となる。また、WiXi はシナプスからの影響を、θは閾値を表している。

式（2.2）の f (u)は伝達関数といい、0 と 1 を使った 2 値のみを出力するような 2 値モデルと、連続値モデルがある。

2 値モデルの場合、伝達関数は以下のようになる。(図 2.2)

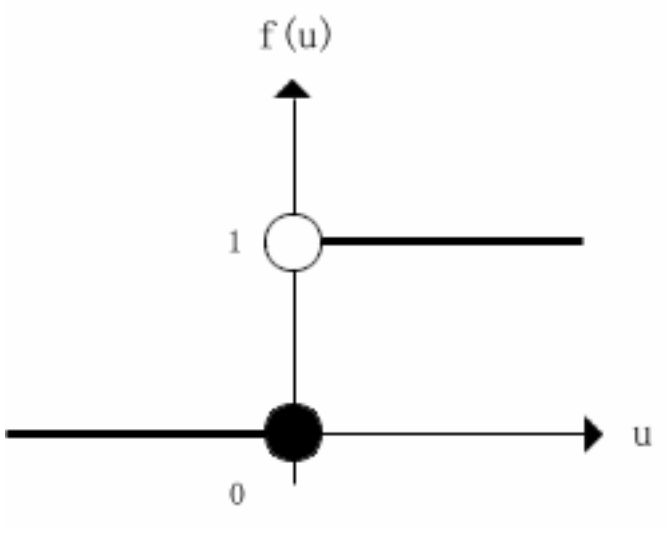


図 2.2　2 値モデルの伝達関数

$$f(u)=\begin{cases}1 & (u>0\ \text{のとき})\\ 0 & (u\leqq 0\ \text{のとき})\end{cases} \tag{2.3}$$

この様に出力値が 0,1 の 2 値となるようなモデルは、入力の線形和が閾値を超えたときのみ 1 を出力することから“線形閾値素子”モデルといわれる。

連続値モデルの場合には、伝達関数にシグモイド(Sigmoid)関数が使われることが多い。(図 2.3)

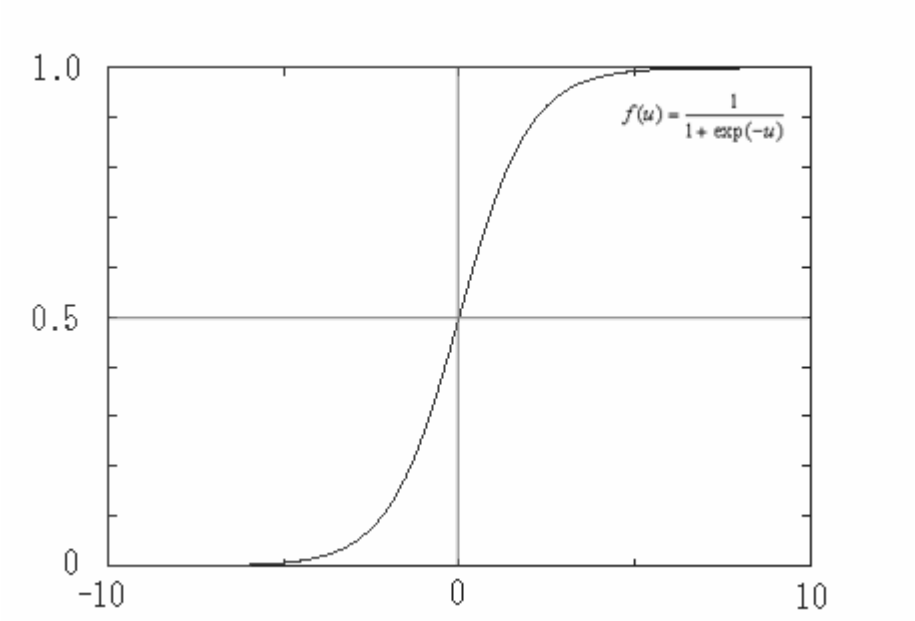


図 2.3　シグモイド関数

$$f(u)=\frac{1}{1+\exp(-u)} \tag{2.4}$$

２．２　　ニューラルネットワークの構造

多くのモデルニューロンをどのように結合させるかで、ネットワークの特性が決まってくる。その結合方法として代表的な、階層型ネットワークと相互結合型ネットワークの 2 種類を説明する。

２．２．１　　階層型ネットワーク

階層型ネットワークはいくつかの層から構成されている。図 2.4 は N 層からなるネットワークを示していて、1 番目の層のことを入力層、N 番目の層を出力層、間の層を中間層という。

図 2.4 のように同じ層内のニューロンは結合していなく、入力層から中間層、中間層から出力層へと結合していて、信号（パターン）もこの順に伝わる。

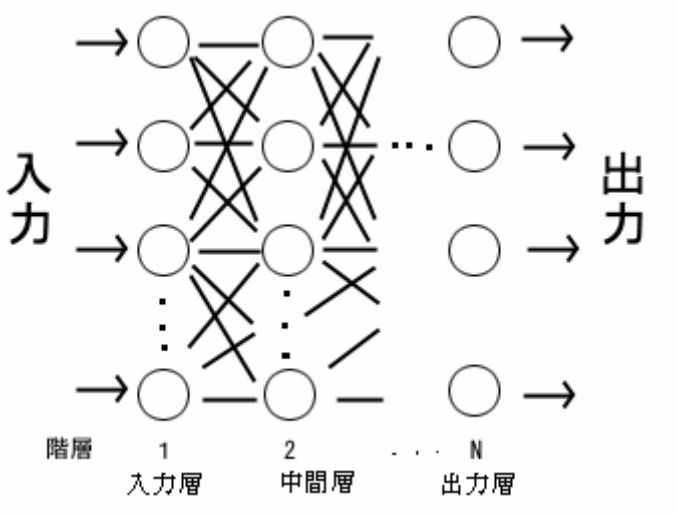


図 2.4　階層型ネットワーク
N 層からなる階層型ネットワークである。
中間層は第 2 層から N−1 層を指す。

学習の際、階層型ネットワークには繰り返し刺激が入り、与えられたある問題に対し望ましい入出力パターンを得るために変化していく。このとき、問題によくあった学習モデルが重要となる。

階層型ネットワークの代表的な学習モデルとして、誤差伝播法（バックプロパゲーション）がある。これを本研究の学習モデルとする。

２．２．２　　相互結合型ネットワーク

相互結合型ネットワークは階層型ネットワークのような層を持たず、各ニューロンが相互に他のニューロンと結合しているネットワークである。相互結合型ネットワークの簡単な構造を図 2.5 に示す。

このネットワークでは、全てのニューロンが平等になっていて、全体の発火ニューロンパターンがある意味を持つ。つまり、ニューロン全体である情報のパターンを記憶したり、思い出したりということが出来ることになる。

相互結合型ネットワークの代表的なものとして、ホップフィールド(Hopfield)ネットがある。その特徴をまとめると以下のようになる。

- ニューロン間の結合加重が互いに対称である。
- それぞれのニューロンは自分のペースで勝手に自分の出力を変化させる。
- 結合加重と閾値は、入力と出力の関係で定められる。これは相関学習と呼ばれる。

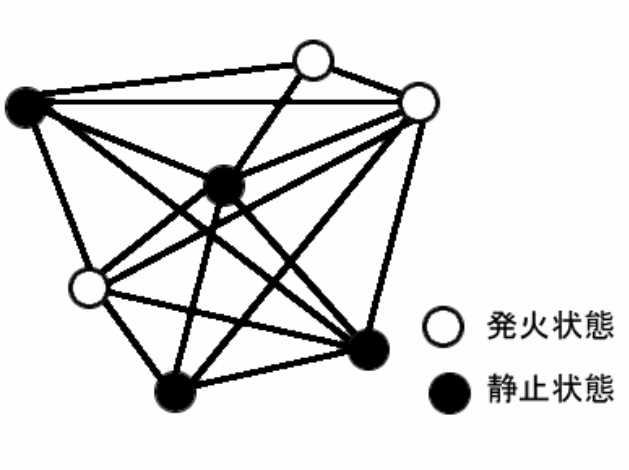


図 2.5　相互結合型ネットワーク

２．３　　階層型ネットワークの学習（誤差伝播法）

階層型ネットワークでは、前の層のユニット（モデルニューロン）の出力を次のユニットの入力としている。つまり、入力層に与えられたパターンが結合の重みによって変換されながら出力層のユニットの値を出力しているといえる。このときの具体的な個々の動作は“２．１　モデルニューロンの構造”を参考にしていただきたい。

出力パターンが、望ましいパターンと成る様に重みを調整することを「学習」という。また、望ましい出力パターンのことを教師信号といい、教師信号が存在する学習方法を教師付き学習と呼ぶ。先に述べたとおり、本研究では誤差伝播法というモデルを用いる。

誤差伝播法による学習方法を説明する。はじめに、ある入力パターンに対する出力を計算する。次にその出力パターンと教師信号との差を計算し、それを学習信号として出力層から入力層へ伝播させながら結合の重みを改善していく。

図 2.6 に N 層からなるネットワークの学習を示す。

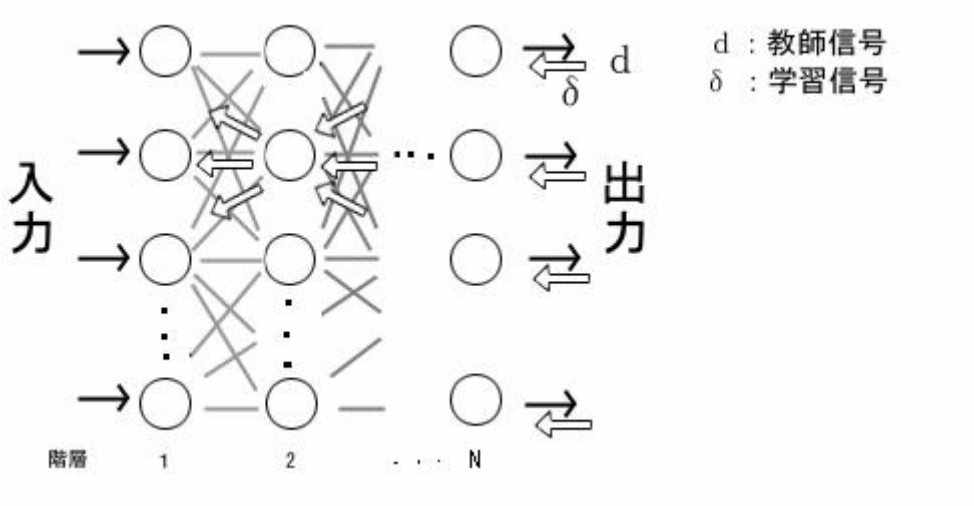


図 2.6　誤差逆伝播法

ネットワークを動かした時と逆の向き（左⇒右）に信号が流れている。

“２．２．１　階層型ネットワーク”で説明したように第ｎ層からは第ｎ＋１層へ結合しているだけであり、飛び越した結合はない。また層間の結合は各層の全てのユニットが互いに結合している完全結合型とする。

第ｎ層ｉ番目のユニットからの出力値を X_i^n とする。また、第ｎ－１層ｊ番目のユニットから第ｎ層ｉ番目のユニットへの結合の重みを $W_{i,j}^{n,n-1}$ とし、閾値を θ_i^n とする。

出力層からの出力値 X_i^N は式(2.1)(2.2)より、

$$X_i^N = f(u_i^N) \tag{2.5}$$

$$u_i^N = \sum_j W_{i,j}^{n,n-1} X_i^{N-1} \tag{2.6}$$

と表せる。ここで、$-\theta_i^n$ を $-\theta_i^n \times 1.0$ と考えれば、常に 1.0 を出力するユニットが重み $-\theta_i^n$ で結合しているとみなせる。

式(2.5)の伝達関数 $f(u_i^N)$ にはシグモイドを使うので式(2.4)より

$$f(u_i^N) = X_i^N = \frac{1}{1+\exp(-u_i^N)} \tag{2.7}$$

となる。

学習信号の与え方は、出力層(第 N 層)が戻す学習信号と、N－1 層よりも前の層が戻す学習信号とでは異なってくる。まず、第 N 層 i 番目のユニットから戻される学習信号 δ_i^N は、教師信号 d_i を使って、

$$\delta_i^N = (d_i - X_i^N) f'(u_i^N) \tag{2.8}$$

と求められる。

$f'(u_i^N)$ は伝達関数の微分であり、微分を行うのに都合の良い連続値を取りうるシグモイド関数を用いるのである。式(2.7)から計算して、

$$\begin{aligned} f'\left(u_i^N\right) &= \left(\frac{1}{1+\exp\left(-u_i^N\right)}\right)' \\ &= \frac{\exp\left(-u_i^N\right)}{(1+\exp(-u_i^N))^2} \\ &= \left(1-\frac{1}{1+\exp\left(-u_i^N\right)}\right)\cdot\frac{1}{1+\exp\left(-u_i^N\right)} \\ &= \left(1-f\left(u_i^N\right)\right)\cdot f\left(u_i^N\right) \\ &= \left(1-X_i^N\right)\cdot X_i^N \end{aligned} \tag{2.9}$$

となるので、δ_i^N は

$$\delta_i^N = \left(d_i - X_i^N\right)\cdot\left(1-X_i^N\right)\cdot X_i^N \tag{2.10}$$

となる。

次に、第 N－1 層よりも前の層が戻す学習信号を求める。

第 n 層 i 番目のユニットから n－1 層のユニットに戻される学習信号 δ_i^N は、第 n＋1 層 k 番目のユニットが戻す学習信号 δ_i^N とユニット i, k 間の重み $W_{k,i}^{n+1,n}$ を用いて次のように求められる。

$$\delta_i^n = f'\left(u_i^n\right)\sum_k \delta_k^{n+1}\cdot W_{k,i}^{n+1,n} \tag{2.11}$$

つまり、第 n 層 i 番目のユニットと n＋1 層への結合を逆に伝わる学習信号に、結合

が固有に持っている重みを掛けて合計したものと、伝達関数の微分との積が学習信号となる。

結合の重みの修正量は次の式で求められる。

$$\Delta W_{i,j}^{n,n-1}(t) = \eta \delta_i^n X_j^{n-1} + \alpha \Delta W_{i,j}^{n,n-1}(t-1) \qquad (2,12)$$

$\Delta W_{i,j}^{n,n-1}(t)$はｎ－１層ｊ番目のユニットとｎ層ｉ番目のユニットとの結合の重みに対する修正量を示し、$\Delta W_{i,j}^{n,n-1}(t-1)$は前回の修正量を示す。つまりｔは学習回数を表している。ηは学習定数といい、収束の早さに関係するが、大きな値を与えたからといって速く収束するとは限らない。αは安定化定数といって、前回の修正量を利用して、収束時の振動を抑える効果がある。この意味で式(2.12)の第 2 項は慣性項と呼ばれる。η，αは共に 1～0 の実数で、前もって適当な値を与えておく。

よって次の式によって結合の重みを修正する。

$$W_{i,j}^{n,n-1}(t+1) = W_{i,j}^{n,n-1}(t) + \Delta W_{i,j}^{n,n-1}(t) \qquad (2,13)$$

次に閾値の修正量を決める。閾値の修正については、前途したように常に 1.0 を出力するユニットから閾値に相当する結合の重みを持って信号がきていると考えられる。したがって、閾値の学習信号は式(2.10)(2.11)で求めたものを使い、修正量は式(2.12)に$X_i^{n-1} \equiv 1$とみなして

$$\Delta \theta_i^n(t) = \eta \delta_i^n + \alpha \Delta \theta_i^n(t-1) \qquad (2.14)$$

と求められる。

よって次の式で閾値を修正する。

$$\theta_i^n(t+1) = \theta_i^n(t) + \Delta \theta_i^n(t) \qquad (2.13)$$

以上の手順を繰り返すことで、出力値と教師信号との差は小さくなっていく。

この作業は、出力値と教師信号との差の自乗和の極小値を与える最急降下法の手順を示している。注意すべき事は、一般的に出力値と教師信号との差の自乗和の極小値は複数存在するため、誤差は必ずしも最小になるとは限らない事である。

第３章　　モデルニューロンによる論理演算

実際に C 言語を使って、ニューラルネットワークのシミュレーションを行う事にする。

まず簡単のためモデルニューロン１つだけを用いたプログラムを書いた。このモデルは、単純パーセプトロンと呼ばれている。このプログラムで、デジタル回路や論理演算の AND,OR,XOR をそれぞれ学習させ、学習回数による出力値の変化を測定した。

次に、入力層、中間層、出力層の 3 層からなる階層型ネットワークを使ったプログラムを書いた。モデルニューロンの数は、入力層２個、中間層 2 個出力層 1 個とした。このプログラムでも同様に、デジタル回路や論理演算の AND,OR,XOR をそれぞれ学習させ、学習回数による出力値の変化を測定した。

３．１　　単純パーセプトロンによる論理演算

２入力１出力のモデルニューロン(図3.1)を使った単純パーセプトロンのプログラムによって論理演算をしてみる。

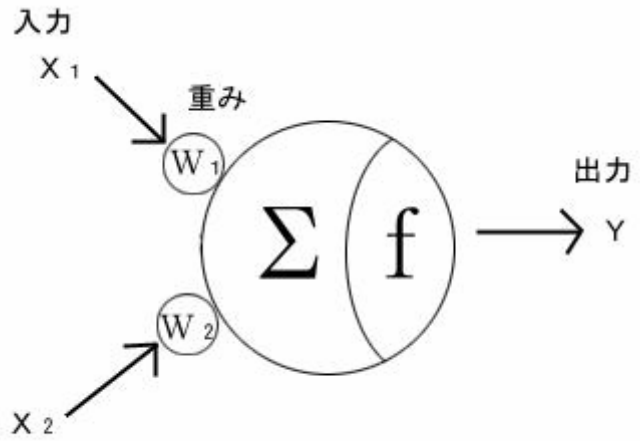


図 3.1　２入力１出力のモデルニューロン

今回用いたプログラムでは、学習は以下のようにした。

学習信号δは、教師信号ｄと実際の出力値ｙの差で次のように表した。

$$\delta = d - y \tag{3.1}$$

よって重み W1,W2 の修正量は、次のように求められる。

$$\Delta W1 = \alpha \cdot \delta \cdot X1$$
$$= \alpha \cdot (d - y) \cdot X1 \quad (3.2)$$
$$\Delta W2 = \alpha \cdot (d - y) \cdot X2 \quad (3.3)$$

ここで、α は重みに対する学習定数を示し、このプログラムでは 0.05 とした。

重みの修正は、

$$W1 = W1 + \Delta W1 \quad (3.4)$$
$$W2 = W2 + \Delta W2 \quad (3.5)$$

となる。

閾値の修正量は次の式で表せる。

$$\Delta \theta = \eta \cdot (d - y) \quad (3.6)$$

ここで、η は閾値に対する学習定数を示し、このプログラムでは 0.5 とした。

閾値の修正は、

$$\theta = \theta - \Delta \theta \quad (3.7)$$

となる。

また重みの初期値は、−0.3〜0.3 の間の疑似乱数で与え、閾値の初期値は 0 とした。

次に入力と学習の与え方を決める。

入力 X1,X2 と教師信号 d を 1 つのパターンとして、4 パターンを 1 つのデータとして学習させる。このデータを入出力パターンと呼ぶ。

学習は以下の手順で行った。

1. 1 番目の入力パターン X1,X2 とそれに対する教師信号 d を与える。
2. 式(2.5)(2.6)より出力 y を得る。
3. その出力 y と教師信号 d を使い、式(3.1)によって学習信号 δ を求める
4. 式(3.2)から(3.7)により、重み W1,W2 と閾値 θ の修正をする。

この作業を 4 パターンに対して行うことを、1 回の学習とした。

なお、初期の重みが疑似乱数で与えられるので、プログラム実行毎に出力はわずかに変化する。この変化は通常は非常に小さいため、以下のシミュレーション結果には、テスト結果としてある 1 回の値を載せることにする。

３．１．１　AND の論理演算

n	X1	X2	d
0	1	1	1
1	1	0	0
2	0	1	0
3	0	0	0

表 3.1　AND の学習パターン
X1,X2：入力
d：学習信号

表 3.1 の入出力データを用いて学習を行った。
ｎ＝０～３を１回の学習として、1000 回学習し、学習回数３回ごとに出力ｙを記録しグラフを作成した。
また、重みの学習定数を 0.05、閾値の学習定数を 0.5 とした。

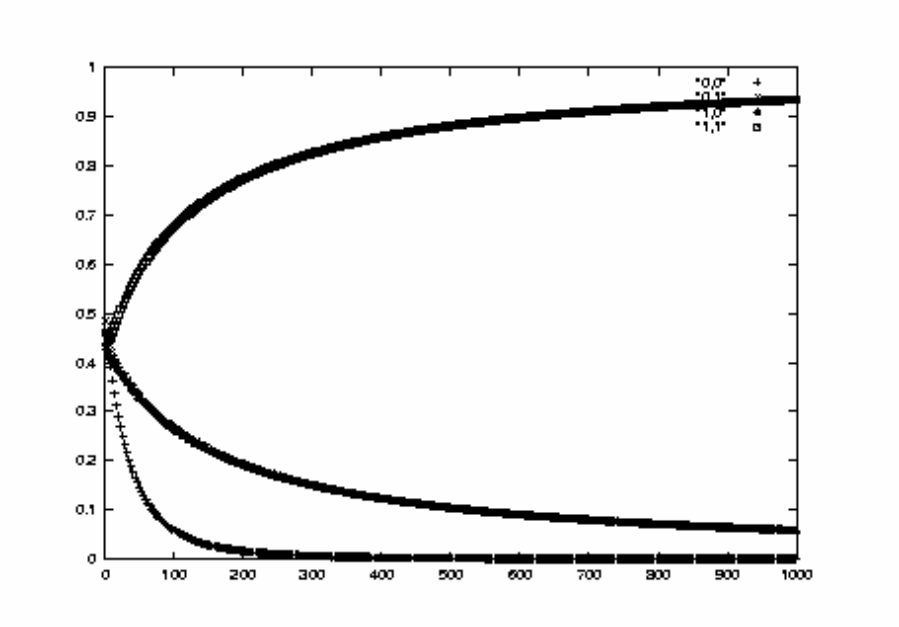


図 3.2　AND の学習による出力値の変化
X 軸は学習回数、Y 軸は出力値を示す。
(1,1)の時の出力が１に、その他の時の出力が
0 に収束している様子がわかる。

学習後のテスト結果

学習前の重みと閾値　(w1, w2, θ)=(-0.132 -0.238 0.0)
学習後の重みと閾値　(w1, w2, θ)=(5.441,5.444,8.2)

(0,0)　y =0.000271
(0,1)　y =0.058991
(1,0)　y =0.058817
(1,1)　y =0.935340

結果から、AND の学習に成功したといえる。

３．１．２　OR の論理演算

n	X1	X2	d
0	1	1	1
1	1	0	1
2	0	1	1
3	0	0	0

表 3.2　OR の学習パターン

X1,X2：入力

d：学習信号

ｎ＝０～３を１回の学習として、300 回学習し、学習回数３回ごとに出力ｙを記録しグラフを作成した。

また、重みの学習定数を 0.05、閾値の学習定数を 0.5 とした。

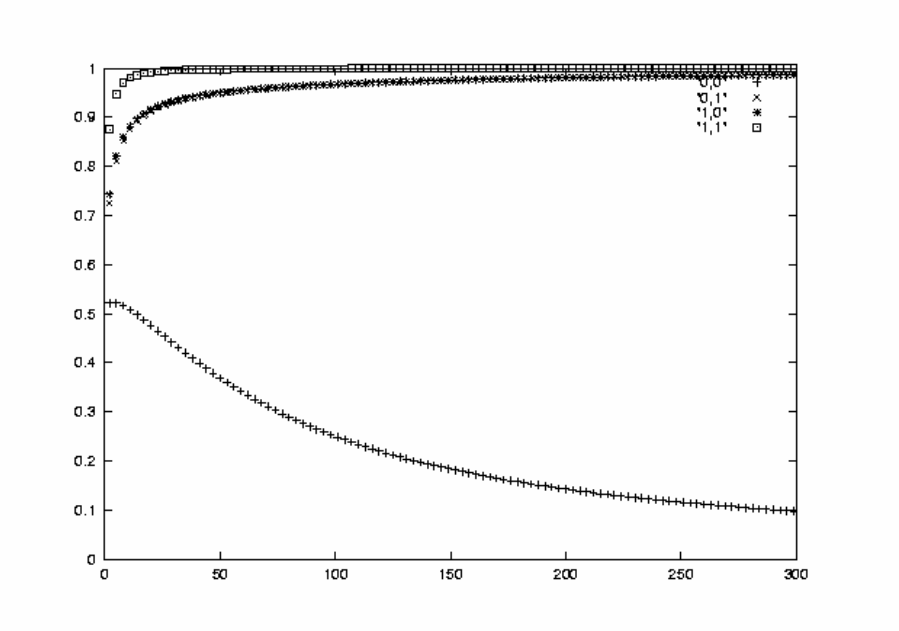


図 3.3　OR の学習による出力値の変化

学習後のテスト結果

学習前の重みと閾値(w1, w2, θ)=(0.012 0.230 0.0)

学習前の重みと閾値(w1, w2, θ)=(6.540,6.540,2.2)

(0,0) y =0.097200
(0,1) y =0.986758
(1,0) y =0.986759
(1,1) y =0.999981

この結果から、OR も学習できたと言える。

３．１．３　XOR の論理演算

n	X1	X2	d
0	1	1	0
1	1	0	1
2	0	1	1
3	0	0	0

表 3.3　XOR の学習パターン

X1,X2：入力

d：学習信号

表 3.3 の入出力データを用いて学習を行った。
ｎ＝０～３を１回の学習として、2000 回学習し、学習回数３回ごとに出力ｙをプロットした。これを図 3.4 に示す。
また、重みの学習定数を 0.05、閾値の学習定数を 0.5 とした。

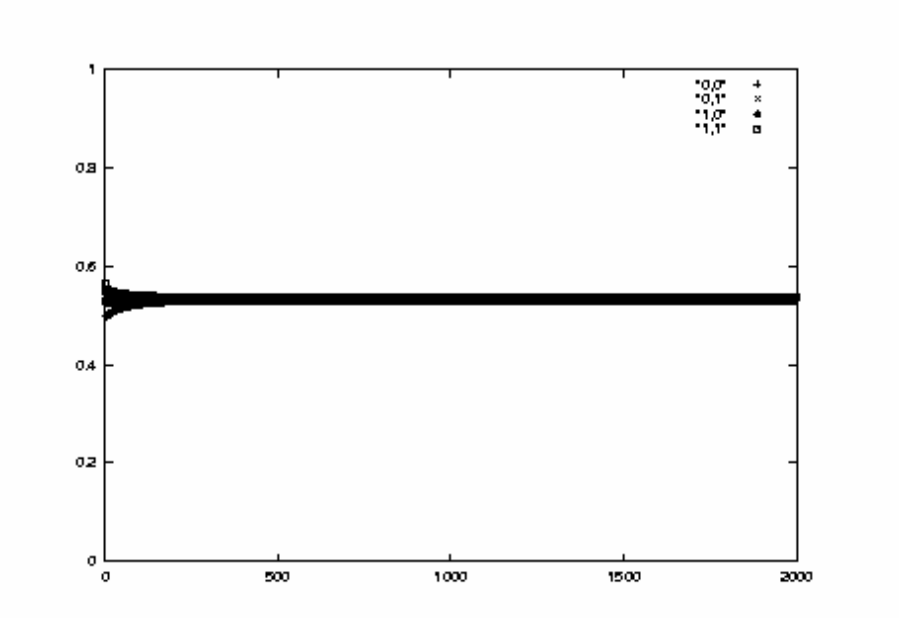


図 3.4　XOR の学習による出力値の変化

収束はしているが、出力の値が 0.5 に収束してしまっている。

学習後のテスト結果
学習前の重みと閾値(w1, w2, θ)=(0.176 0.247 0.0)
学習後の重みと閾値(w1, w2, θ)=(0.027,0.000,-0.1)

(0,0) y =0.530160
(0,1) y =0.530161
(1,0) y =0.536841
(1,1) y =0.536841

XOR は正しく学習できなかった。その理由については次節“ 3.1.4　考察 ”で述べる。

３．１．４　考察

単純パーセプトロンでは、OR と AND の学習については成功したが、XOR の学習については失敗した。これは一般に以下のように説明されている。

この単純パーセプトロンのプログラムでは対象を１本の直線で２つのカテゴリーに分類するという作業を行っている。

この作業について説明するため、X 軸を入力 X1、Y 軸を入力 X2 とした AND,OR,XOR の平面を図 3.5～3.7 に表した。

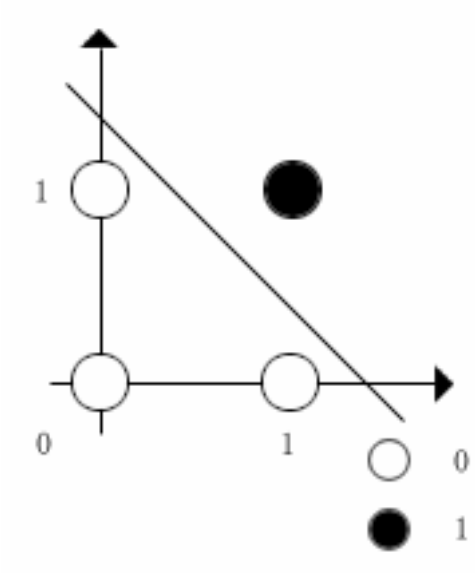


図 3.5　AND 問題の平面

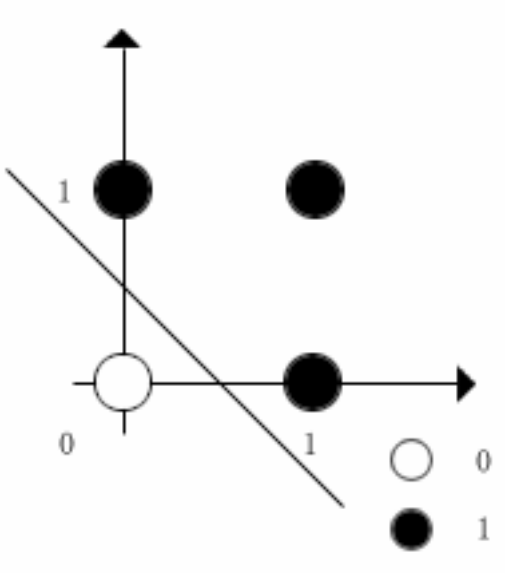


図 3.6　OR 問題の平面

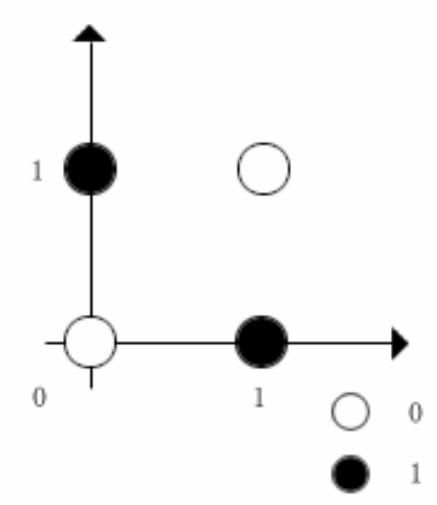


図 3.7　XOR 問題の平面

X 軸に入力 X1、Y 軸に入力 X２を用いた。また、黒丸,白丸はそれぞれ教師信号（または出力）が、1,0 であることを示している。

AND,OR どちらの図も斜めの直線で白丸と黒丸に分類できていることがわかる。

一方、XOR では１本の直線で分類できない事がわかる。

図 3.5,3.6 より、AND,OR の平面は 1 本の直線で 0 と 1 に分類でき、単純パーセプトロンで判別できることがわかる。このような問題を線形分離可能という。

図 3.7 の XOR のように 1 本の直線で分離できない問題は単純パーセプトロンで判別することは出来ない。このような問題を線形分離不可能という。

線形不可能な問題を判別するには、階層型ネットワークが必要になるといわれている。次項では階層型ネットワークによる論理演算を試してみる。

３．２　階層型ネットワークによる論理演算

XOR のような線形分離不可能な問題を解くために図 3.8 のような 3 層型ネットワークを用いて論理演算を行った。

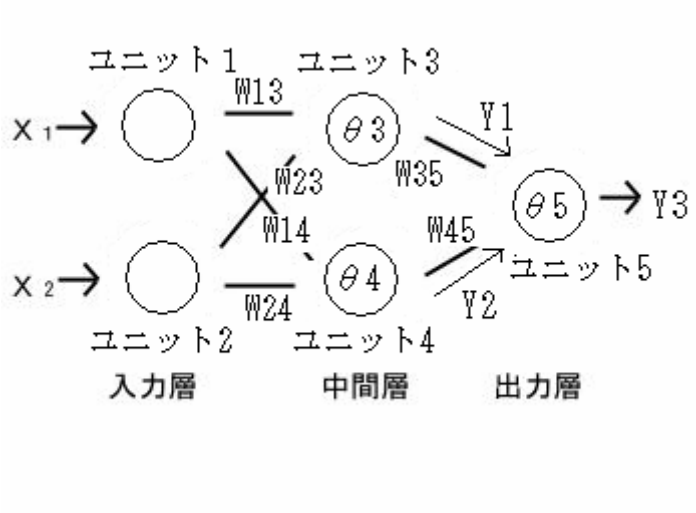


図 3.8　3 層型ネットワーク
入力層,中間層は 2、出力層は 1 ユニットで構成されている

図 3.8 のようにユニットや重みに名前を付けた。また、ユニット 3 からの出力を Y1、ユニット 4 からの出力を Y2、ユニット 5 からの出力を Y3 とした。

このプログラムでは学習方法は次のようにした。

まず、出力層のユニット(ユニット 5)からの学習信号δ5 は、式(2.10)より次のように表した。

$$\delta 5 \quad = \quad (d - Y3)\cdot(1 - Y5)\cdot Y3 \tag{3.8}$$

よって重み W35,W45 の修正量は、次のように求められる。

$$\begin{aligned} \Delta W35(t) &= \eta\cdot\delta 5\cdot Y3 + \alpha\cdot\Delta W35(t-1) \\ &= \eta\cdot(d-y)\cdot Y3 + \alpha\cdot\Delta W35(t-1) \qquad (3.9) \\ \Delta W45(t) &= \eta\cdot(d-y)\cdot Y3 + \alpha\cdot\Delta W45(t-1) \qquad (3.10) \end{aligned}$$

ここで、η は学習定数を示し、このプログラムでは 0.75 とおいた。α は安定化定数を示し、

このプログラムでは 0.８とした。

重みの修正は

$$W35 = W35 + \Delta W35 \quad (3,11)$$

$$W45 = W45 + \Delta W45 \quad (3,12)$$

となる。

閾値θ3 の修正量は

$$\Delta\theta 3(t) = \eta \cdot \delta 5 + \alpha \cdot \Delta\theta 3(t-1) \quad (3.13)$$

とした。

閾値の修正は

$$\theta 3 = \theta 3 - \Delta\theta 3 \quad (3.14)$$

となる。

ユニット 3,4 からの学習信号δ3,δ4 は、式(2.11)より次のように表した。

$$\delta 3 = (1-Y1)\cdot Y1 \cdot \delta 5 \cdot W35 \quad (3.15)$$

$$\delta 4 = (1-Y2)\cdot Y2 \cdot \delta 5 \cdot W45 \quad (3.16)$$

式(3.14)(3.15)を用いて、重み W13,W23 の修正量は、次のように求められる。

$$\Delta W13(t) = \eta \cdot \delta 3 \cdot X1 + \alpha \cdot \Delta W13(t-1) \quad (3.17)$$

$$\Delta W23(t) = \eta \cdot \delta 3 \cdot X2 + \alpha \cdot \Delta W23(t-1) \quad (3.18)$$

$$\Delta W14(t) = \eta \cdot \delta 4 \cdot X1 + \alpha \cdot \Delta W14(t-1) \quad (3.19)$$

$$\Delta W24(t) = \eta \cdot \delta 4 \cdot X2 + \alpha \cdot \Delta W24(t-1) \quad (3.20)$$

重みの修正は

$$W13 = W13 + \Delta W13 \quad (3.21)$$

$$W23 = W23 + \Delta W23 \quad (3.22)$$

$$W14 = W14 + \Delta W14 \quad (3.23)$$

$$W24 = W24 + \Delta W24 \quad (3.24)$$

となる。

閾値の修正量は次の式で表せる。

$$\Delta\theta 1(t) = \eta \cdot \delta 3 + \alpha \cdot \Delta\theta 1(t-1) \quad (3.25)$$

$$\Delta\theta 2(t) = \eta \cdot \delta 4 + \alpha \cdot \Delta\theta 2(t-1) \quad (3.26)$$

閾値の修正は

$$\theta 1 = \theta 1 - \Delta\theta 1 \quad (3.14)$$

$$\theta 2 = \theta 2 - \Delta\theta 2 \quad (3.14)$$

となる。

また重みの初期値は、－0.3～0.3 の間の議事乱数で与え、閾値の初期値は 0 とした。

入力と学習の与え方は“3.1　単純パーセプトロンによる論理演算”と同様にした。

３．２．１　階層型ネットワークによる AND の論理演算

“３．１．１　AND の論理演算”と同じく、表 3.1 の入出力データを用いてシミュレーションを行った。このとき、学習回数は 4000 回 、安定化定数は 0.5、学習定数は 0.7 とした。学習の際の、出力の変化を図 3.9 に示した。

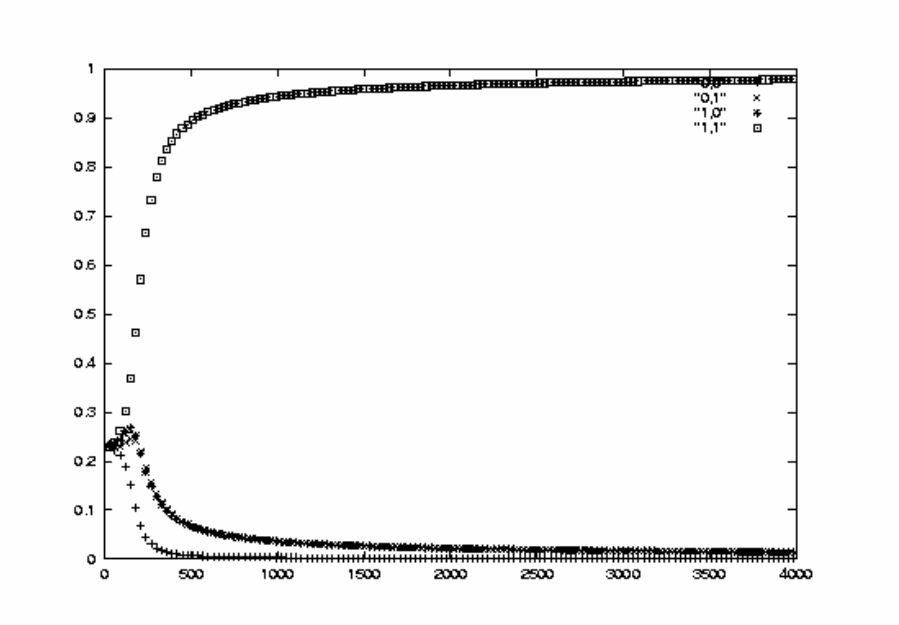


図 3.9　AND の学習による出力値の変化(3 層型ネットワーク)
入力が 1,1 のときの出力が１に、その他の出力は０に近づいていることがわかる。

テスト結果
　学習前の重みと閾値
(w13, w23, θ1)=(-0.058 0.284 0.0)
(w14, w24, θ2)=(0.289 0.236 0.0)
(w35, w45, θ3)=(-0.183 -0.101 0.0)

学習後の重みと閾値
(w13, w23, θ1)=(-3.952,-3.933,-5.6)
(w14, w24, θ2)=(1.488,1.521,1.4)
(w35, w45, θ3)=(-9.147,3.887,-1.4)

(0,0) y =0.000939
(0,1) y =0.014378

(1,0) y =0.014269
(1,1) y =0.977966

構造が複雑になった分、学習回数を多く必要としたが、学習に成功したといえる。

３．２．２　OR の論理演算

“３．１．２　OR の論理演算”と同じく、表 3.2 の入出力データを用いてシミュレーションを行った。このとき、学習回数は 4000 回 、安定化定数は 0.5、学習定数は 0.7 とした。学習の際の、出力の変化を図 3.10 に示した。

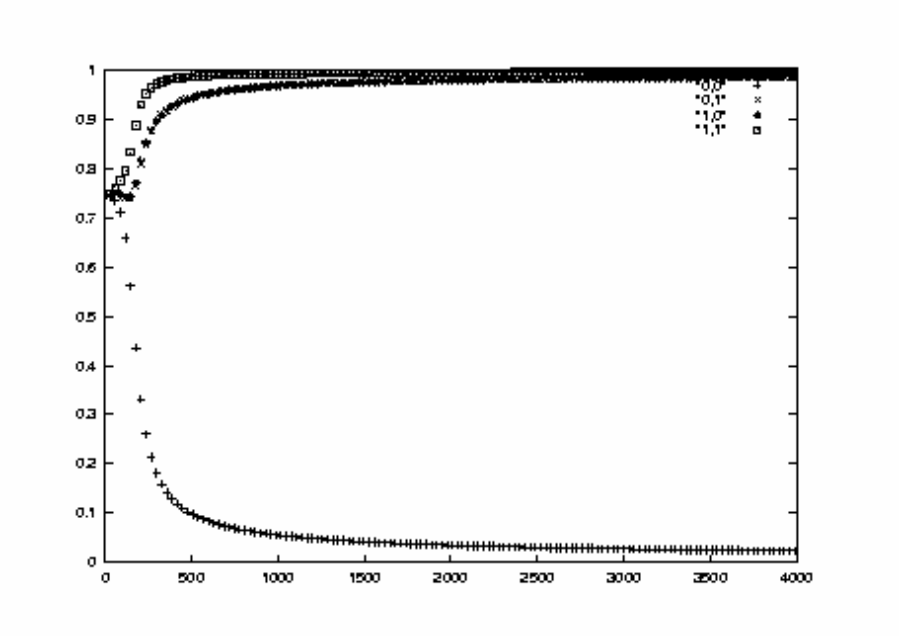


図 3.10　OR の学習による出力値の変化(3 層型ネットワーク)
入力が 0,0 のときの出力が 0 に、その他の出力は 1 に近づいていることがわかる。

テスト結果
学習前の重みと閾値
(w13, w23, θ1)=(-0.033 -0.182 0.0)
(w14, w24, θ2)=(-0.158 0.081 0.0)
(w35, w45, θ3)=(0.106 -0.053 0.0)

学習後の重みと閾値
(w13, w23, θ1)=(4.265,4.216,2.3)
(w14, w24, θ2)=(3.267,3.321,1.9)
(w35, w45, θ3)=(6.542,4.493,5.0)

(0,0) y =0.022664

(0,1) y =0.987697
(1,0) y =0.987663
(1,1) y =0.997561

OR の学習についても、成功したといえる。

３．２．３　XOR の論理演算

“３．１．３　XOR の論理演算”と同じく、表 3.3 の入出力データを用いてシミュレーションを行った。このとき、学習回数は 20000 回 、安定化定数は 0.5、学習定数は 0.7 とした。学習の際の、出力の変化を図 3.11 に示した。

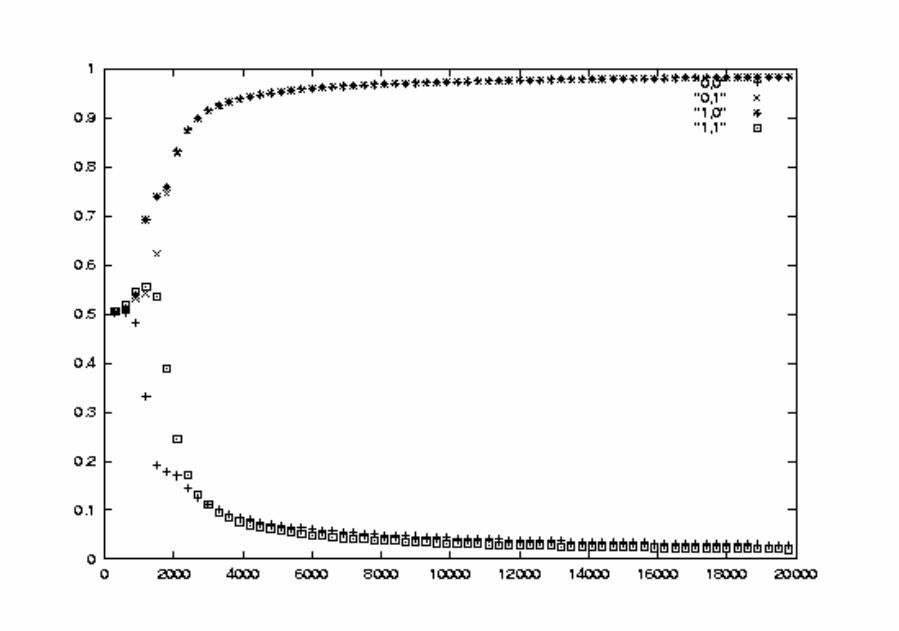


図 3.11　XOR の学習による出力値の変化(3 層型ネットワーク)その１

入力が 0,0，1,1 のときの出力が 1 に、1,0，0,1 の出力が 1 に近づいていることがわかる。

テスト結果
学習前の重みと閾値
(w13, w23, θ1)=(-0.081 0.002 0.0)
(w14, w24, θ2)=(0.052 0.049 0.0)
(w35, w45, θ3)=(0.112 0.020 0.0)

学習後の重みと閾値
(w13, w23, θ1)=(4.928,4.935,7.6)
(w14, w24, θ2)=(6.900,6.930,3.1)
(w35, w45, θ3)=(-11.325,10.608,4.9)

(0,0) y =0.010941

(0,1) y =0.990673
(1,0) y =0.990657
(1,1) y =0.009602

この結果からは XOR の学習にも成功したといえる。

しかし、 “３．１．１ AND の論理演算”で指摘したとおり、プログラムを実行する度に出力値が変わってきてしまう。普通、ほとんど変わることは無いのだが、このプログラムでは２回に１回位の確率で次のような結果が出てしまった。

条件は同じ（学習回数 20000 回 安定化定数 0.5 学習定数 0.7）として、学習を行い出力の変化を図 3.12 に示した。

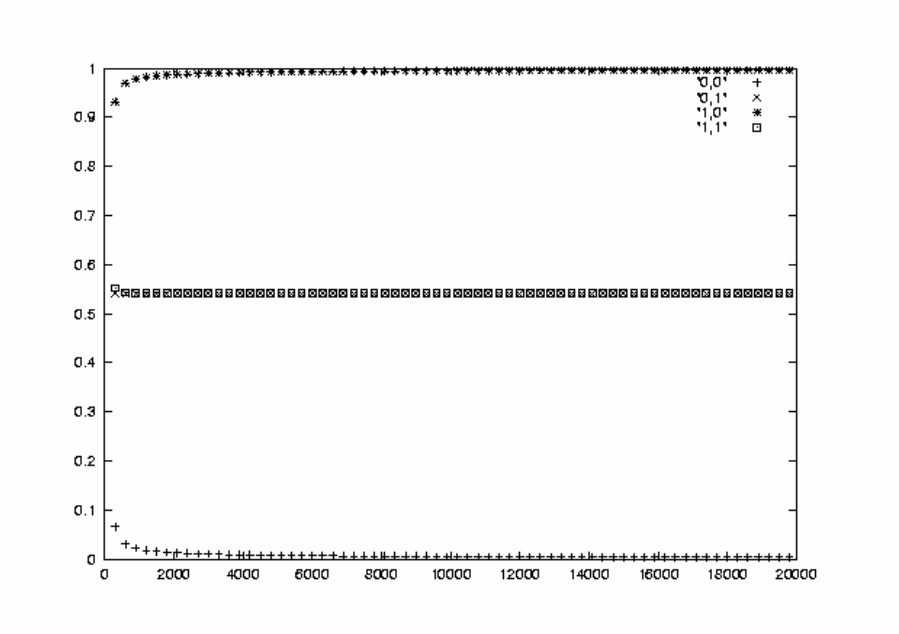


図 3.12 XOR の学習による出力値の変化(3 層型ネットワーク)その 2
入力が 0,1, 1,1 のときの出力が 0.5 位になってしまっている。

テスト結果
学習前の重みと閾値
(w13, w23, θ1)=(0.269 0.187 0.0)
(w14, w24, θ2)=(-0.124 0.013 0.0)
(w35, w45, θ3)=(0.212 -0.099 0.0)

学習後の重みと閾値

(w13, w23, θ1)=(5.353,10.505,1.1)
(w14, w24, θ2)=(-2.986,10.177,-1.9)
(w35, w45, θ3)=(7.792,-6.454,1.2)

(0,0) y =0.007963
(0,1) y =0.534771
(1,0) y =0.992042
(1,1) y =0.535090

このように、上手く学習できないこともあった。また、失敗するときはいつも(0,0)　0, (0,1)　0.5,(1,0)　1, (1,1)　0.5 となった。学習定数や安定化定数を変えても改善しなかった。

３．２．４　考察

複雑になった分学習回数は多くなったが、AND,OR の学習には成功した。XOR については 2 回に 1 回位の割合で失敗してしまった。この原因としては、“ 2.3　階層型ネットワークの学習（誤差伝播法）” で述べたように、最小二乗誤差を求める際、極小値に落ち込んでしまったということが考えられる。

しかし、*他の文献ではこの現象は確認できなかった。このことからなんらかのプログラム上のミスが考えられる。しかし、以下の点において単純なミスによるものではないと主張する。

・図 3.12 より出力値は充分に収束している。

・学習による重みの変化を図 3.13 に示してみると、重みも充分収束している。

・新たに別に作ったプログラムでも、同様の現象がみられた。(“ 付録 3　単純パーセプトロンによる XOR のプログラム 2 ” 参照)

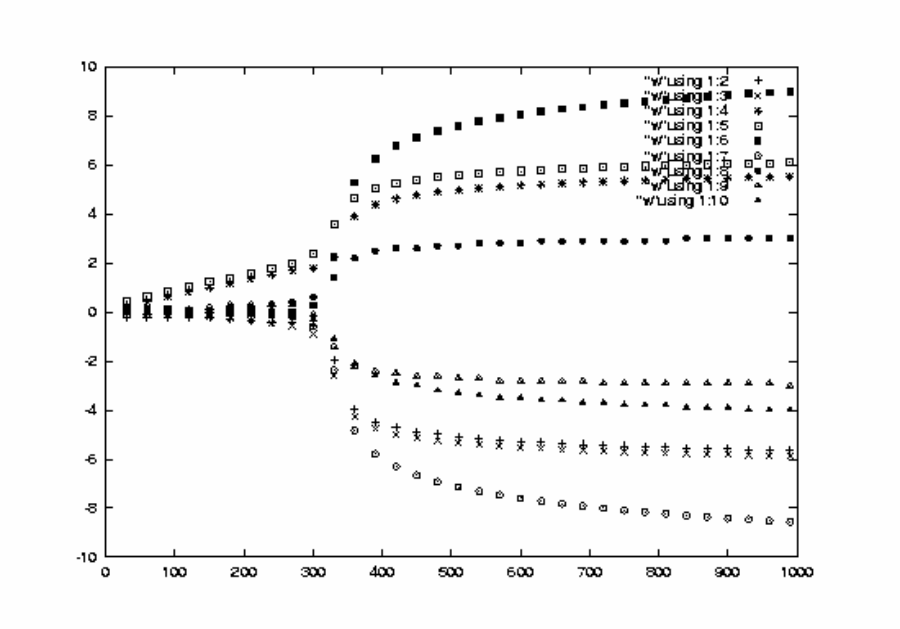


図 3.13　XOR の学習に失敗したときの学習による重みの変化
各重みが収束していることがわかる。

（＊）他の文献
平成 10 年度信州大学卒業論文　「ニューラルネットワークによるパターン認識」倉地広志
平成 13 年度信州大学学士学位論文　「ニューラルネットワークによる判別回路」高田徳之
を指す。

第４章　　四角と三角のパターン認識

ニューラルネットワークを使えば、丸や四角といったパターン認識の問題も解くことが出来る。ここでは 10×10 のマスに描いた四角と三角のパターンの認識を行った。これは、100 入力１出力の判断をさせていることになる。ここでは入力層 100 ユニット、中間層 16 ユニット、出力層１ユニットからなる 3 層型ネットワークを用いた。パターン認識のシミュレーションを行う上で、まず学習パターンを学習させた後、学習パターンとテストパターンの両方を入力し動作を確認した。

４．１　パターン認識のための入力

パターン認識の上で、学習パターンの入力の仕方は重要な問題である。図 4.1 のパターンを入力させる場合を例に、本研究での入力方法を説明する。

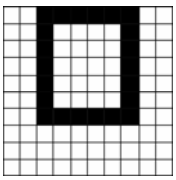


図 4.1　入力パターン例

まず、図 4.1 にあるマス目のうち黒なら１、白なら 0 として考える。これを左上から右に順番に入力層のユニットに入力した。つまり、ヒトの目には四角に見える図 4.1 の入力も機械にとっての入力パターンは００１１１１１１０００１０００１０００１・・・・・・となる事になる。この入出力データをまずテキストファイルに作っておいて、プログラムで読み込んで処理をするようにした。

パターン 1～6 までを学習パターン、7～12 までをテストパターンとする入出力データは次のように作られる。(“ 付録５　パターン認識のための入出力データ１ ” 参照)

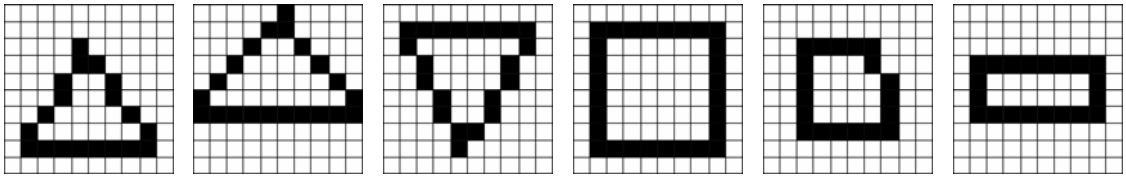
```
6                          ←　学習パターン数
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0        ←　パターン１の入力
        ・
        ・
```

・
0 0 0 0 0 0 0 0 0 0
1　　← パターン１の教師信号

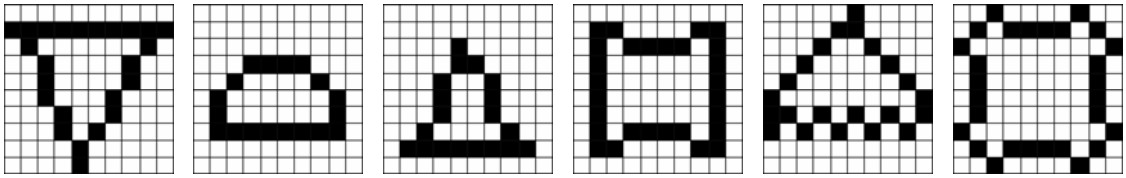
0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0　　← パターン２の入力
・
・
・
0 0 0 0 0 0 0 0 0 0
1　　← パターン２の教師信号
・
（パターン 6 まで繰り返す）
・
・
6　　← テストパターン数
0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
・　　← パターン７の入力
・
0 0 0 0 1 0 0 0 0 0
・
・
・
（パターン 12 まで繰り返す）

４．２　四角と三角のパターン認識

シミュレーションに使う入力パターンを図 4.2 に示した(パターンデータ 1)。パターンデータ１の内、パターン１から６までを学習パターンとし、パターン７から１２をテストパターンとした。



パターン 1　パターン 2　パターン 3　パターン 4　パターン 5　パターン 6



パターン 7　パターン 8　パターン 9　パターン 10　パターン 11　パターン 12

図 4.2　パターンデータ 1

パターン 1,2,3 が三角、パターン 4,5,6 が四角を表している。教師信号として三角を 1、四角を 0 とした。

学習を 10 回行い、そのときの出力 y をプロットした。このときの出力値の変化を図 4.3 に示した。

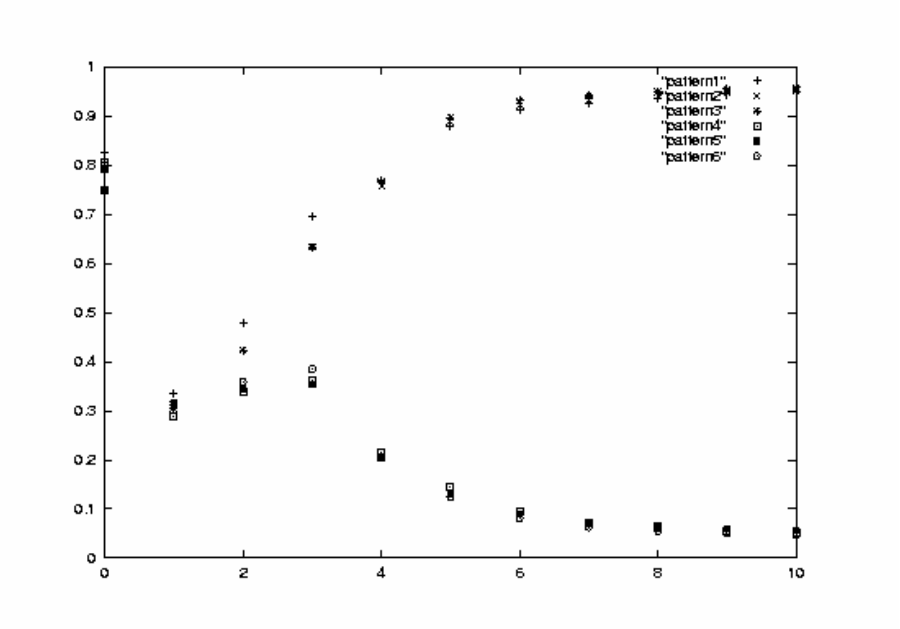


図 4.3　学習時の出力の変化
パターン 1 から 3 の出力が 1、パターン 4 から 6 の出力が 0 に
近づいている事が確認できる。

学習が終了した後、学習パターンとテストパターンをどのように判断するかテストさせた。このとき出力 y≧0.5 のとき三角、y＜0.5 のとき四角として判断した。

学習後のテスト結果

pattern 1 = 0.952　三角　　pattern 2 = 0.955　三角
pattern 3 = 0.965　三角　　pattern 4 = 0.047　四角
pattern 5 = 0.040　四角　　pattern 6 = 0.047　四角

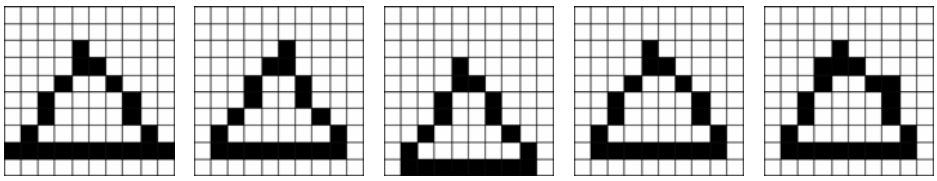
pattern 7 = 0.907　三角　　pattern 8 = 0.031　四角
pattern 9 = 0.956　三角　　pattern 10 = 0.022　四角
pattern 11 = 0.958　三角　　pattern 12 = 0.052　四角

この結果からは、四角と三角の判断が出来たと考えられる。

４．３　少しずつ変化を与えた場合のパターン認識

4.2 のシミュレーション結果が、偶然の産物でないかを確かめるため、学習パターン、テストパターンを変えて実験することにした。

ここでは、学習パターンは、パターンデータ 1（図 4.2　参照）を同じとした。テストパターンは、パターン 1 の三角形を少し変化させて作った。これをパターンデータ 2 とした。パターンデータ 2 のテストパターンを図 4.4 に示す。



パターン 7　　パターン 8　　パターン 9　　パターン 10　　パターン 11

図 4.4　パターンデータ 2 のテストパターン

学習後のテスト結果

pattern 1 = 0.946　三角　　pattern 2 = 0.952　三角
pattern 3 = 0.955　三角　　pattern 4 = 0.049　四角
pattern 5 = 0.044　四角　　pattern 6 = 0.045　四角

pattern 7 = 0.634　三角　　pattern 8 = 0.916　三角
pattern 9 = 0.798　三角　　pattern 10 = 0.641　三角
pattern 11 = 0.529　三角

学習パターンの三角形を少し変化させたものをテストパターンに使用しているので、テストパターン全てを三角形と判断した事は充分に予想の範囲内であるといえる。

パターン 7 はパターン 1 の底辺を左右 1 マスずつ伸ばした物で、ヒトにはより三角形と判断しやすい形だと考えることが出来る。しかし、その分学習パターンから、離れてしまっているため、出力値が小さくなってしまったと考えられる。

パターン 9 は全体を一段下げたのものである。これは、入力があるユニットの位置（番号）がまったく変わってしまうことを意味する。しかしこの出力値は 0.798 であり、一応三角形と判断できたことになる。このことから、単に入力のあったニューロンの位置で判断をしているのではなく、入力のあるユニットの位置パターンからも判断していると推測される。

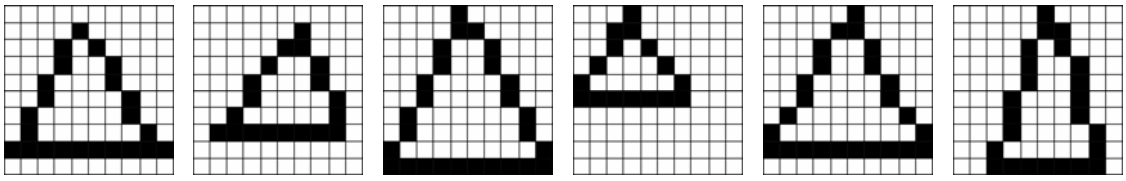
パターン 10、11 は四角に近づけているので出力値が 0.5 に近くなった。このことは、

ニューラルネットが曖昧なものを曖昧に判断することが出来るという事を示す一例となるのではないかと考える。

４．４　いろいろな三角に対するパターン認識

4.3 の結果から、ニューラルネットは学習パターンと似たパターンを認識出来るという結果が出た。そこで、この節では、学習パターンと大きく離れたパターンの三角形をテストパターンとして、どこまできちんと判断が出来るかを調べてみた。

このシミュレーションに用いたテストパターンを図 4.5 に示す。



パターン 7　パターン 8　パターン 9　パターン 10　パターン 11　パターン 12

図 4.5　パターンデータ３のテストパターン

学習パターンはパターンデータ１と同じものとした。

学習後のテスト結果

pattern 1 = 0.950　三角　　pattern 2 = 0.951　三角
pattern 3 = 0.949　三角　　pattern 4 = 0.055　四角
pattern 5 = 0.051　四角　　pattern 6 = 0.043　四角

pattern 7 = 0.238　四角　　pattern 8 = 0.196　四角
pattern 9 = 0.134　四角　　pattern 10 = 0.653　三角
pattern 11 = 0.405　四角　　pattern 12 = 0.937　三角

テストパターンは全て三角形を用いたが、結果はほとんど四角と判断してしまった。パターン 10 の出力値は 0.65 とあまり良い判断をしているとはいえないことから、きちんと判断できたのはパターン 12 のみということも出来る。

これらのパターンは学習のパターンと大きく違うものを意識して作ったものではあるが、予想していたよりも正解率が低いと感じた。逆に、大きく位置をずらしたパターン 10 や、学習パターンと大きく形の異なるパターン 12 を三角と判断できたことも意外だった。

このことから、学習パターンと大きく離れたパターンの判断はほとんど出来無いといえ、パターン認識としての道具としては実用的ではないといえる。

しかし、学習パターンと離れたパターンが判断できないということは、逆にいえば非常

に多くのパターンを学習させればいろいろなパターンに対応できる可能性があるということである。実際、ヒトが形を判断できるのも、莫大な数の（パターンの）形を見てきたからだと考えられる。

４．５　学習パターンを増やした場合のパターン認識

4.4 の結果より学習パターンを増やす事で、いろいろなパターンに対応できるという予想が立てられる。この節ではさまざまなタイプの四角や三角の学習パターンを 15 個用意したパターンデータ４を用いて実験を行った。テストパターンはパターンデータ１のものを使った。パターンデータ 4 を図 4.5 に示す。

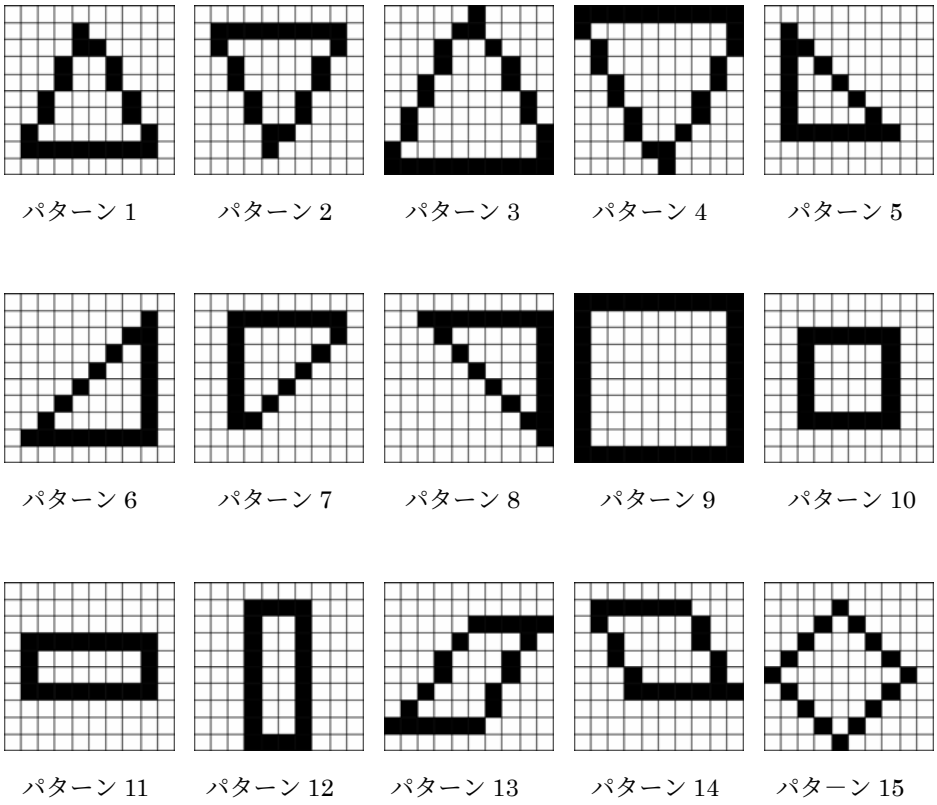


図 4.6　パターンデータ 4

直角三角形や平行四辺形も学習パターンとした。

pattern 1 = 0.977　三角　　pattern 2 = 0.987　　三角
pattern 3 = 0.915　三角　　pattern 4 = 0.980　　三角
pattern 5 = 0.986　三角　　pattern 6 = 0.977　三角
pattern 7 = 0.971　三角　　pattern 8 = 0.949　三角
pattern 9 = 0.087　四角　　pattern 10 = 0.063　四角
pattern 11 = 0.028　四角　　pattern 12 = 0.048　四角
pattern 13 = 0.039　四角　　pattern 14 = 0.053　四角
pattern 15 = 0.036　四角

pattern 16 = 0.993　三角　　pattern 17 = 0.871　三角
pattern 18 = 0.507　三角　　pattern 19 = 0.994　三角
pattern 20 = 0.070　四角　　pattern 21 = 0.982　三角

学習パターンについては正しく判断できたが、テストパターンはほとんど判断できなかった。多くのパターンに対応するためにいろいろな学習パターンを作ったが、基本的なテストパターンにも対応できない結果となった。これを克服するためには、さらに多くのパターンを学習する必要があると考えられる。しかし、このプログラムでは手作業でパターンを作る必要があり、これ以上パターンを増やすことは難しいと思われる。

第５章　　演算

第 4 章で行ったパターン認識のシミュレーションでは学習パターンでないパターンを認識できるかが重要なポイントであった。この‘学習していないことに答える能力’のことを“推測”と呼ぶことにする。

この“推測”について調べるため、0～5 までの整数 2 つを足すシミュレーションを行った。

また、第 3 章、4 章では、ユニットに 1,0 の値を入力し、出力を 1,0 として読んでいたが、このシミュレーションでは 1 つのユニットに 0,1 以外の数も対応させたモデルを用いた。この 0,1 以外の数を対応させたモデルが、きちんと動作するかについても確認することにする。

５．１　演算のためのモデル

0～5 までの整数同士の足し算をさせるにあたって、どのようなモデルを用いるかについて説明する。

構造

１.入力層 2、中間層 30、出力層 1 ユニット

２.入力層 2、中間層 100、出力層 1 ユニット

この 2 つのモデルを用いてシミュレーションを行い、動作を比較した。

入力

入力層の 2 つのユニットにそれぞれ 0～5 の整数を入力した。

これまで(第 3,4 章)1 つのユニットには 1 つの数しか対応させなかったが、0,1 以外の数を入力させてもいいと考える根拠を以下に述べる。

- シグモイド関数は全ての実数の入力に対応している点（図 2.3,式(2.4)参照）
- 各ユニットの出力は 0～1 の範囲であるが、次の層のユニットには重み W をかけた値を入力しているため 0,1 以外の数を入力していることになる。例えば“3.1.1 AND の論理演算”の学習後の重みは(w1, w2, θ)=(5.441,5.444,8.2)となっていて、5 以上の入力にも対応出来ることが予想できる。

答え

出力は 0～1 の範囲の実数なので、*10 倍してから小数点以下を四捨五入した値を答えと読んだ。

(＊) 10 倍してから小数点以下を四捨五入

この方法だと、出力と答えの対応は以下のようになる。

出力	答え
0.000‥～0.049‥	0
0.050‥～0.149‥	1
0.150‥～0.249‥	2
・	
・	
0.950‥～1.000‥	10

この時、0 と 10 と答えを読む範囲が他の数字よりも狭いという問題が起きる。しかし、このことは目的である“推測”について調べる上では問題とならないためこのモデルでシミュレーションを行った。

5．2　ニューラルネットワークによる演算

0～5 までの整数 2 つの足し算を 36 通り全てについて学習させた。このシミュレーションに用いた入出力パターンを表 5.1 に示した。

問題	答え	教師信号	問題	答え	教師信号	問題	答え	教師信号
0　0	0	0.0	2　0	2	0.2	4　0	4	0.4
0　1	1	0.1	2　1	3	0.3	4　1	5	0.5
0　2	2	0.2	2　2	4	0.4	4　2	6	0.6
0　3	3	0.3	2　3	5	0.5	4　3	7	0.7
0　4	4	0.4	2　4	6	0.6	4　4	8	0.8
0　5	5	0.5	2　5	7	0.7	4　5	9	0.9
1　0	1	0.1	3　0	3	0.3	5　0	5	0.5
1　1	2	0.2	3　1	4	0.4	5　1	6	0.6
1　2	3	0.3	3　2	5	0.5	5　2	7	0.7
1　3	4	0.4	3　3	6	0.6	5　3	8	0.8
1　4	5	0.5	3　4	7	0.7	5　4	9	0.9
1　5	6	0.6	3　5	8	0.8	5　5	10	1.0

表 5.1　演算のための入出力パターン 1

構造１.入力層 2、中間層 30、出力層 1 ユニット
学習回数 1200 回　　学習定数 0.75　　安定化定数 0.8
としてシミュレーションを行った。結果を表 5.2 に示す。

問題	出力	答え		問題	出力	答え		問題	出力	答え	
0　0	0.032	0	正解	2　0	0.199	2	正解	4　0	0.395	4	正解
0　1	0.088	1	正解	2　1	0.300	3	正解	4　1	0.487	5	正解
0　2	0.189	2	正解	2　2	0.393	4	正解	4　2	0.583	6	正解
0　3	0.303	3	正解	2　3	0.482	5	正解	4　3	0.690	7	正解
0　4	0.404	4	正解	2　4	0.571	6	正解	4　4	0.797	8	正解
0　5	0.494	5	正解	2　5	0.668	7	正解	4　5	0.884	9	正解
1　0	0.097	1	正解	3　0	0.301	3	正解	5　0	0.496	5	正解
1　1	0.191	2	正解	3　1	0.396	4	正解	5　1	0.593	6	正解

1 2	0.302	3	正解	3 2	0.485	5	正解	5 2	0.698	7	正解
1 3	0.403	4	正解	3 3	0.576	6	正解	5 3	0.807	8	正解
1 4	0.490	5	正解	3 4	0.674	7	正解	5 4	0.898	9	正解
1 5	0.578	6	正解	3 5	0.777	8	正解	5 5	0.952	10	正解
								正解数		36	
								正解率		100.00%	

表 5.2　演算のシミュレーション結果(中間層 30 ユニット)

結果は全て正解となった。

次に中間層を増やしたモデルと比較してみる。
構造２.入力層２、中間層 100、出力層１ユニット
　学習回数 2000 回　　学習定数 0.75　　安定化定数 0.8
としてシミュレーションを行った。結果を表 5.3 に示す。

問題	出力	答え		問題	出力	答え		問題	出力	答え	
0 0	0.032	0	正解	2 0	0.190	2	正解	4 0	0.401	4	正解
0 1	0.098	1	正解	2 1	0.301	3	正解	4 1	0.496	5	正解
0 2	0.197	2	正解	2 2	0.401	4	正解	4 2	0.588	6	正解
0 3	0.297	3	正解	2 3	0.493	5	正解	4 3	0.689	7	正解
0 4	0.395	4	正解	2 4	0.584	6	正解	4 4	0.798	8	正解
0 5	0.496	5	正解	2 5	0.679	7	正解	4 5	0.892	9	正解
1 0	0.096	1	正解	3 0	0.295	3	正解	5 0	0.499	5	正解
1 1	0.197	2	正解	3 1	0.401	4	正解	5 1	0.594	6	正解
1 2	0.302	3	正解	3 2	0.495	5	正解	5 2	0.697	7	正解
1 3	0.401	4	正解	3 3	0.586	6	正解	5 3	0.806	8	正解
1 4	0.497	5	正解	3 4	0.685	7	正解	5 4	0.897	9	正解
1 5	0.594	6	正解	3 5	0.791	8	正解	5 5	0.952	10	正解
								正解数		36	
								正解率		100.00%	

表 5.3　演算のシミュレーション結果(中間層 30 ユニット)

　これも全て正解となった。中間層が増えたことにより学習回数が増えたこと以外は、特に動作の違いは見られなかった。中間層を増やしても動作に影響が無いということから、中間層は 30 ユニットあれば十分であると考えることが出来る。

５．３　演算と推測

学習されていないものをどれだけ推測できるかを確かめるため 6 パターンの足し算のみを学習させる。

この入出力パターンを表 5.4 にまとめる。

問題	答え	教師信号
0　0	0	0
1　2	3	0.3
2　4	6	0.6
3　1	4	0.4
4　3	7	0.7
5　5	10	1

表 5.4　演算のための入出力パターン 2

この入出力パターンを用いて、シミュレーションを行い、学習回数と正解率の関係について調べることにした。5.2 と同じく、
学習定数 0.75　　安定化定数 0.8
構造１　入力層２、中間層 30、出力層 1 ユニット
構造２　入力層２、中間層 100、出力層 1 ユニット
の 2 種類とした。

まず学習回数と学習パターンの正解率の関係を図 5.1，5.2 に示す。これよりどちらの構造も 1200 回頃には学習パターンの学習が出来ている事がわかる。

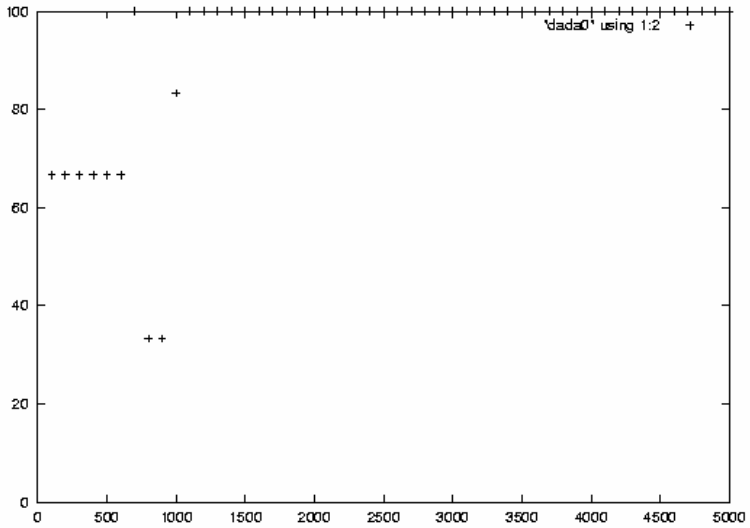


図 5.1　学習回数と学習パターンの正解率その 1

縦軸：正解率　　横軸：学習回数

構造 1(入力層 2，中間層 30，出力層 2 ユニット)を用いた。

1100 回の学習で学習パターンの学習が出来ていることが解る。

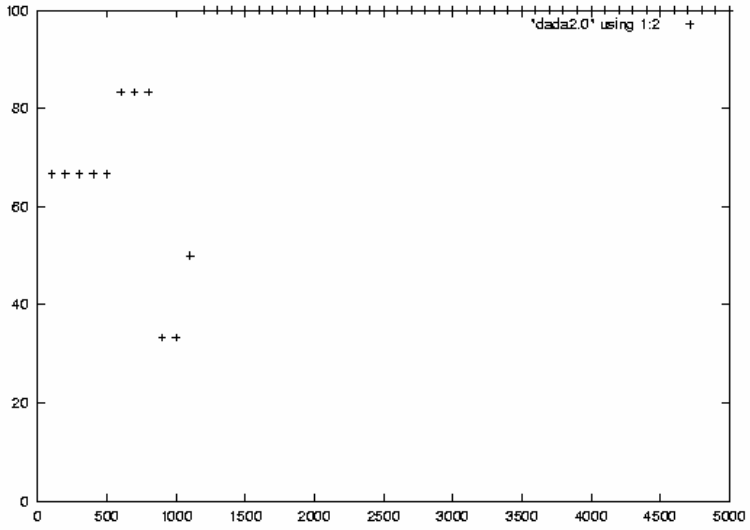


図 5.2　学習回数と学習パターンの正解率その 2

縦軸：正解率　　横軸：学習回数

構造 2(入力層 2，中間層 100，出力層 2 ユニット)を用いた。

1100 回の学習で学習パターンの学習が出来ていることが解る。

次に 36 パターン全てをテストさせた時の、学習回数と正解率の関係を図 5.3,5.4 に示す。どちらの構造も学習回数 900 回で正解率は低くなり、その後少し正解率が上がっている。全体的に見ると学習回数が増えると正解率が下がるといえる。

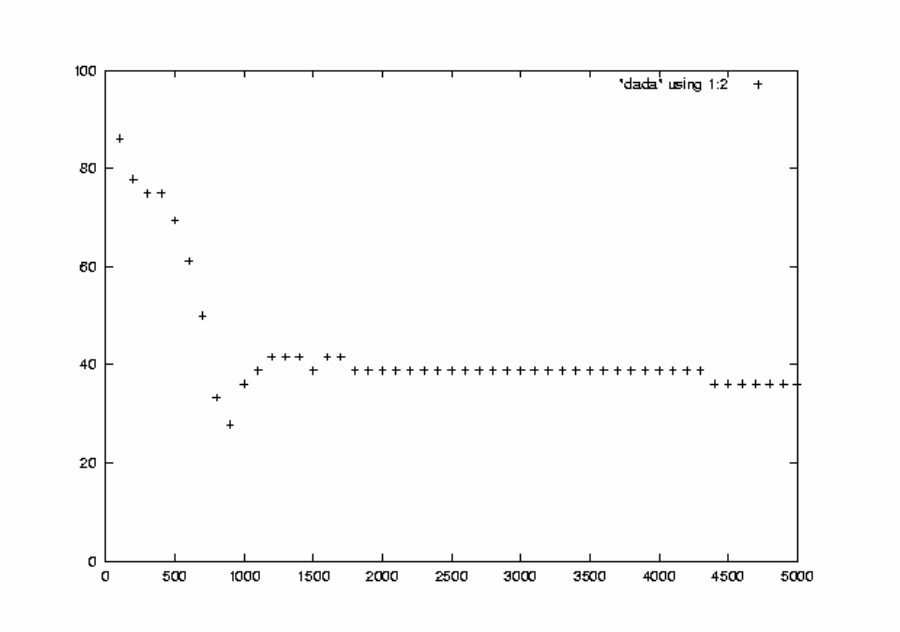


図 5.3　学習回数と全パターンの正解率その 1
縦軸：正解率　　横軸：学習回数
構造 1(入力層 2，中間層 30，出力層 2 ユニット)を用いた。

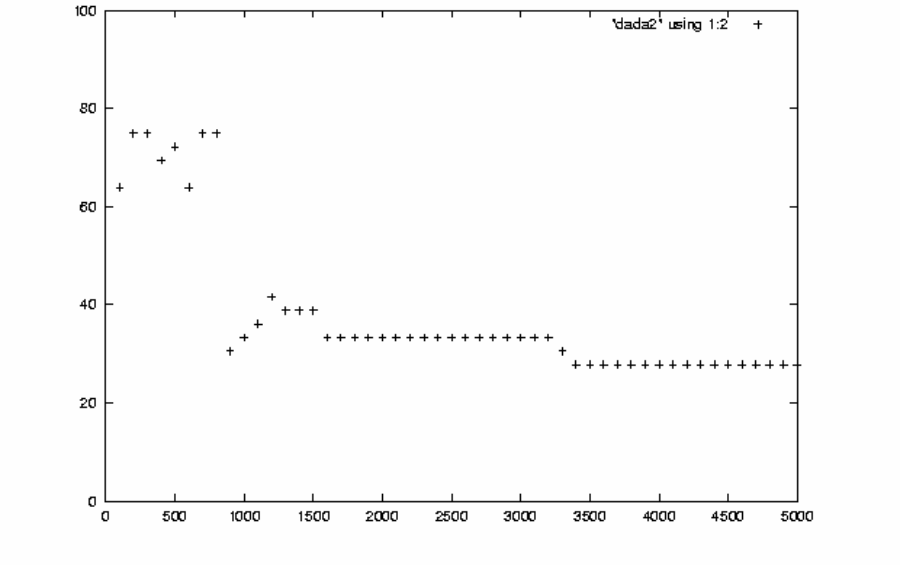


図 5.4　学習回数と全パターンの正解率その 1
縦軸：正解率　　横軸：学習回数

構造 1(入力層 2，中間層 100，出力層 2 ユニット)を用いた。

第６章　　まとめ

・ニューラルネットワークを用いてシミュレーションを行い、動作を確認することが出来た。
・パターン認識についても動作を確認することが出来た。
・パターン認識では学習させたパターンと近いパターンの認識は出来た。
・数多くのパターンに対応させるためには、それだけ多くのパターンを学習することが必要となる事が予想される。
・ユニットに 0,1 以外の数を対応させた“ 加算 ”もニューラルネットワークで行えることがわかった。
・学習回数が増えると“ 推測 ”の値が低くなることがある。

付録

この研究で用いたプログラムを紹介する。

付録１　単純パーセプトロンによる XOR のプログラム

```
/* 単純パーセプトロンによる XOR のプログラム */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define   MAX_L   2000                  /*       学習上限       */
#define   sig(u)   1.0/(1.0+exp(-u));       /*     シグモイド関数     */
#define   Wmin    -0.3                  /*  重みの初期化の最小値  */
#define   Wmax    0.3                   /*  重みの初期化の最小値  */


main()
{
  const float eta    = 0.05;            /*     重みの学習定数     */
  const float alpha = 0.5;              /*     閾値の学習定数     */
  enum{x1,x2,out,Te=out};
  const float T_Tbl[4][3]=
   {
     1.00, 1.00, 0.0,
     1.00, 0.00, 1.0,
     0.00, 1.00, 1.0,
     0.00, 0.00, 0.0                  /*    XOR の学習パターン    */
   };

  float w1,w2,threth,u,y,a,b;

  int    i,j,n,l;
  char   buf[80];
```

```
   FILE   *fopen(),*fp1, *fp2, *fp3,*fp4;
   fp1 = fopen("1,1","w");
   fp2 = fopen("1,0","w");
   fp3 = fopen("0,1","w");
   fp4 = fopen("0,0","w");

   /*   重みと閾値の初期化   */
   srand(time(NULL));
   w1 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
   w2 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
   threth = 0.0;

   /* 学習前の重みと閾値の表示 */
   printf("(w1, w2, θ)=(%2.3f %2.3f %2.1f)¥n",w1,w2,threth);

   l = 0;
   for(i=0;i<=MAX_L;++i)
    {  l = l + 1;
    for(n=0;n<=3;n++)
     {
   /* ネットワークを動かし、出力値を求める */
u = (w1*T_Tbl[n][x1] + w2*T_Tbl[n][x2] - threth);
y = sig(u);

    /*    重みと閾値の修正   */
w1 += alpha*(T_Tbl[n][Te] - y)*T_Tbl[n][x1];
        w2 += alpha*(T_Tbl[n][Te] - y)*T_Tbl[n][x2];
threth -= eta*(T_Tbl[n][Te] - y);
      }
    /* 学習回数 3 回ごとに出力値をファイルに落す */
    if(l==3)
     for(j=0;j<=3;++j)
{
   u = (w1*T_Tbl[j][x1] + w2*T_Tbl[j][x2] - threth);
   y = sig(u);
```

```
    if(j==0)
      fprintf(fp1,"%d %f¥n",i,y);
    else if(j==1)
      fprintf(fp2,"%d %f¥n",i,y);
    else if(j==2)
      fprintf(fp3,"%d %f¥n",i,y);
    else if(j==3)
      fprintf(fp4,"%d %f¥n",i,y);
    l = 0;
}
     }
    fclose(fp1);
    fclose(fp2);
    fclose(fp3);
    fclose(fp4);

    /* 学習後の重み、閾値の表示 */
    printf("(w1, w2, θ)=(%4.3f,%4.3f,%2.1f)¥n",w1,w2,threth);

    /* テスト入力に対しての実行結果 */
    while( 1 )
     {
       printf("¥n 入力 1 は?  "); scanf("%f",&a);
       printf("入力 2 は?  "); scanf("%f",&b);
       u = (w1*a + w2*b - threth);
       y = sig(u);
       printf("y =%7.6f ¥n",y);

       printf("Stop? ( Y or other )¥n");
       gets(buf);
       if( strchr( buf,'y' ) || strchr( buf,'Y' )){break;}
     }
 }
```

付録２　３層型ネットワークによる XOR のプログラム１

```
/* 3 層型ネットワークによる OR のプログラム  */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define   MAX_L    4000                    /*        学習上限        */
#define   sig(u)   1.0/(1.0+exp(-u));      /*     シグモイド関数     */
#define   Wmin     -0.3                   /*   重みの初期化の最小値   */
#define   Wmax     0.3                    /*   重みの初期化の最小値   */


main()
{
   const float walpha = 0.5;              /*    重みの安定化定数     */
   const float halpha = 0.5;              /*    閾値の安定化定数     */
   const float weta    = 0.7;             /*     重みの学習定数      */
   const float heta    = 0.7;             /*     閾値の学習定数      */
   enum{x1,x2,out,Te=out};
   const float T_Tbl[4][3]=
    {
      1.00, 1.00, 1.0,
      1.00, 0.00, 1.0,
      0.00, 1.00, 1.0,
      0.00, 0.00, 0.0                    /*    OR の学習パターン     */
    };

   float w13,w14,w23,w24,w35,w45;                /*  重み  */
   float dw13,dw14,dw23,dw24,dw35,dw45;          /*  重みの修正値  */
   float dwp13,dwp14,dwp23,dwp24,dwp35,dwp45;    /*  前回の重みの修正値  */
   float h1,h2,h3;                            /*  閾値  */
   float dh1,dh2,dh3;                          /*  閾値の修正値  */
   float dhp1,dhp2,dhp3;                       /*  前回の閾値の修正値  */
   float u1,u2,u3;
```

```
float y1,y2,y3;
float e3,e4,e5;                          /* 学習信号 */
float a,b;
int    i,j,n,l;
char   buf[80];

FILE   *fopen(),*fp1, *fp2, *fp3,*fp4;
fp1 = fopen("1,1","w");
fp2 = fopen("1,0","w");
fp3 = fopen("0,1","w");
fp4 = fopen("0,0","w");

/*   重みと閾値の初期化   */
srand(time(NULL));
w13 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
w14 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
w23 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
w24 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
w35 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
w45 = ((float)rand()/(float)RAND_MAX *(Wmax - Wmin) + Wmin);
dwp13 = 0.0; dwp14 = 0.0; dwp23 = 0.0; dwp24 = 0.0; dwp35 = 0.0; dwp45 = 0.0;
h1 = 0.0; h2 = 0.0; h3 = 0.0; dhp1 = 0.0; dhp2 = 0.0; dhp3 = 0.0;


/* 学習前の重みと閾値の表示 */
printf("(w13, w23, θ1)=(%4.3f %4.3f %2.1f)¥n",w13,w23,h1);
printf("(w14, w24, θ2)=(%4.3f %4.3f %2.1f)¥n",w14,w24,h2);
printf("(w35, w45, θ3)=(%4.3f %4.3f %2.1f)¥n",w35,w45,h3);

l = 0;
for(i=0;i<=MAX_L;++i)
 {
 l = l + 1;
 for(n=0;n<=3;n++)
  {
/* ネットワークを動かし、出力値を求める */
```

```
u1 = (w13*T_Tbl[n][x1] + w23*T_Tbl[n][x2] - h1);
y1 = sig(u1);
        u2 = (w14*T_Tbl[n][x1] + w24*T_Tbl[n][x2] - h2);
y2 = sig(u2);
        u3 = (w35*y1 + w45*y2 - h3);
y3 = sig(u3);

    /*   出力層の学習信号を求める   */
e5 = (T_Tbl[n][Te]-y3)*y3*(1-y3);

    /*  出力層の重みと閾値の修正    */
        dw35 = weta*e5*y1+walpha*dwp35;
        w35 += dw35;
        dwp35=dw35;
        dw45 = weta*e5*y2+walpha*dwp45;
        w45 += dw45;
        dwp45=dw45;
dh3 = heta*e5+halpha*dhp3;
        h3 -= dh3;
        dhp3=dh3;

    /*   中間層の学習信号を求める   */
        e3 = y1*(1-y1)*e5*w35;
        e4 = y2*(1-y2)*e5*w45;

    /*   中間層の重みと閾値の修正   */
        dw13 = weta*e3*T_Tbl[n][x1]+walpha*dwp13;
        w13 += dw13;
        dwp13=dw13;
        dw23 = weta*e3*T_Tbl[n][x2]+walpha*dwp23;
        w23 += dw23;
        dwp23=dw23;
dh1 = heta*e3+halpha*dhp1;
        h1 -= dh1;
        dhp1=dh1;
```

```
        dw14 = weta*e4*T_Tbl[n][x1]+walpha*dwp14;
        w14 += dw14;
        dwp14=dw14;
        dw24 = weta*e4*T_Tbl[n][x2]+walpha*dwp24;
        w24 += dw24;
        dwp24=dw24;
dh2 = heta*e4+halpha*dhp2;
        h2 -= dh2;
        dhp2=dh2;
      }
     {
    /*  学習回数 30 回ごとに出力値をファイルに落す  */
     while( l == 30 )
      {
       for(j=0;j<=3;++j)
   {
   u1 = (w13*T_Tbl[j][x1] + w23*T_Tbl[j][x2] - h1);
   y1 = sig(u1);
         u2 = (w14*T_Tbl[j][x1] + w24*T_Tbl[j][x2] - h2);
   y2 = sig(u2);
         u3 = (w35*y1 + w45*y2 - h3);
   y3 = sig(u3);

   if(j==0)
    fprintf(fp1,"%d %f¥n",i,y3);
   else if(j==1)
    fprintf(fp2,"%d %f¥n",i,y3);
   else if(j==2)
    fprintf(fp3,"%d %f¥n",i,y3);
   else if(j==3)
    fprintf(fp4,"%d %f¥n",i,y3);
   }
       l = 0;
      }
     }
     }
```

```
  fclose(fp1);
  fclose(fp2);
  fclose(fp3);
  fclose(fp4);

  /* 学習後の重み、閾値の表示 */
  printf("¥n(w13, w23, θ1)=(%4.3f,%4.3f,%2.1f)¥n",w13,w23,h1);
  printf("(w14, w24, θ2)=(%4.3f,%4.3f,%2.1f)¥n",w14,w24,h2);
  printf("(w35, w45, θ3)=(%4.3f,%4.3f,%2.1f)¥n",w35,w45,h3);

  /* テスト入力に対しての実行結果 */
  while( 1 )
   {
     printf("¥n 入力 1 は?  "); scanf("%f",&a);
     printf("入力 2 は?  "); scanf("%f",&b);
     u1 = (w13*a + w23*b - h1);
     y1 = sig(u1);
     u2 = (w14*a + w24*b - h2);
     y2 = sig(u2);
     u3 = (w35*y1 + w45*y2 - h3);
     y3 = sig(u3);
     printf("y =%7.6f ¥n",y3);

     printf("Stop? ( Y or other )¥n");
     gets(buf);
     if( strchr( buf,'y' ) || strchr( buf,'Y' )){break;}
   }
}
```

付録３　３層型ネットワークによる XOR のプログラム２

さまざまなモデルに作り変えやすいように XOR のプログラムを作り直した。
主な変数の意味は次のようになる。

tsignal[学習パターン番号][出力層の番号]　教師信号
out_in[学習パターン番号][入力層の番号]　入力層からの出力
out_hid[中間層の番号]　中間層からの出力
out_out[出力層の番号]　出力層からの出力
witoh[中間層の番号][入力層の番号]　入力層と中間層のユニットの、結合の重み
dwitoh[中間層の番号][入力層の番号]　〃　の修正値
whtoo[出力層の番号][中間層の番号]　中間層と出力層のユニットの、結合の重み
dwhtoo[出力層の番号][中間層の番号]　〃　の修正値
hbias[中間層の番号]　中間層のユニットの重み
dhbias[中間層の番号]　〃　の修正値
obias[出力層の番号]　出力層のユニットの重み
dobias[出力層の番号]　〃　の修正値

```
/* パターン認識のプログラムを利用した XOR */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define MAX_L   1000                    /*      学習上限        */
#define sig(u)  1.0/(1.0+exp(-u))       /*     シグモイド関数     */
#define Wmin    -0.30                   /*  重みの初期化の最小値  */
#define Wmax    0.30                    /*  重みの初期化の最小値  */
#define urand() ((float)rand()/(float)RAND_MAX * (Wmax - Wmin) + Wmin)

#define iunits   2                      /*  入力層のユニット数   */
#define hunits   2                      /*  中間層のユニット数   */
#define ounits   1                      /*  出力層のユニット数   */
#define patterns 4
```

```
#define eta      0.75                  /*      学習定数       */
#define alpha    0.8                   /*      安定化定数     */


const float out_in[patterns][iunits]=       /*  XOR の学習パターン  */
    {
      1.00, 1.00,
      1.00, 0.00,
      0.00, 1.00,
      0.00, 0.00,
    };
 const float tsignal[patterns][ounits]=
   {
     0.0,
     1.0,
     1.0,
     0.0,
   };
float out_hid[hunits],out_out[ounits];
float witoh[hunits][iunits],dwitoh[hunits][iunits];
float whtoo[ounits][hunits],dwhtoo[ounits][hunits];
float hbias[hunits],dhbias[hunits];
float obias[ounits],dobias[ounits];

main()
    {
      int   i,j,k,l;

      FILE   *fopen(),*fp1, *fp2, *fp3, *fp4;

      void   forward_propagation(),back_propagation();
      void   read_data(),initialize(),test_mode();

  srand(time(NULL)); initialize();
```

```
        fp1 = fopen("1,1","w");
        fp2 = fopen("1,0","w");
        fp3 = fopen("0,1","w");
        fp4 = fopen("0,0","w");


        l = 0;

        for( i = 0; i <= MAX_L; i++ )
{
   l = l + 1;
   for( j = 0; j < patterns; j++ )
     {
       forward_propagation( j );
       back_propagation( j );
     }
   while( l == 30 )
     {
       for( j = 0 ; j < patterns ; j++ )
{
   forward_propagation( j );
   if( j == 0 )
     {
       for( k = 0; k < ounits; k++ )
fprintf(fp1,"%4.3f ¥n",out_out[k]);
     }
   else if( j == 1 )
     {
       for( k = 0; k < ounits; k++ )
fprintf(fp2,"%4.3f ¥n",out_out[k]);
       }
   else if( j == 2 )
     {
       for( k = 0; k < ounits; k++ )
fprintf(fp3,"%4.3f ¥n",out_out[k]);
     }
```

```
    else if( j == 3 )
     {
       for( k = 0; k < ounits; k++ )
fprintf(fp4,"%4.3f ¥n",out_out[k]);
     }
   l = 0;
}
     }
}
       fclose(fp1);
       fclose(fp2);
       fclose(fp3);
       fclose(fp4);



       test_mode();
     }

    /*  データを入力層から出力層に流す  */
void forward_propagation(int p)
{
  int i,j;
  float sum,u;

  for( i = 0; i < hunits; i++ )
{
   sum = 0.0;
   u    = 0.0;
   for( j = 0; j < iunits; j++ )
     sum += witoh[i][j]*out_in[p][j];
   u = sum - hbias[i];
   out_hid[i] = sig( u );
}
  for( i = 0; i < ounits; i++ )
{
```

```
    sum = 0.0;
    u    = 0.0;
    for( j = 0; j < hunits; j++ )
      sum += whtoo[i][j]*out_hid[j];
    u = sum - obias[i];
    out_out[i] = sig( u );
}
}

    /*      重みと閾値の修正      */
void back_propagation(int p)
{
  int i,j;
  float dwih[hunits],dwho[ounits];

  for( i = 0; i < ounits; i++ )
    dwho[i] = ( tsignal[p][i] - out_out[i])*out_out[i]*(1.0 - out_out[i]);

  for( i = 0; i < hunits; i++ )
    {
      for( j = 0; j < ounits; j++ )
{
  dwhtoo[j][i] = eta*dwho[j]*out_hid[i]+alpha*dwhtoo[j][i];
  whtoo[j][i] += dwhtoo[j][i];
  dwih[i] = out_hid[i]*( 1.0 - out_hid[i] )*dwho[j]*whtoo[j][i];
}
    }
 for( i = 0; i < ounits; i++ )
{
dobias[i] = eta * dwho[i] + alpha * dobias[i];
obias[i] -= dobias[i];
}

  for( i = 0; i < iunits; i++ )
{
   for( j = 0; j < hunits; j++ )
```

```
    {
     dwitoh[j][i] = eta * dwih[j] * out_in[p][i] + alpha * dwitoh[j][i];
     witoh[j][i] += dwitoh[j][i];
    }
}

   for( i = 0; i < hunits; i++ )
     {
       dhbias[i] = eta * dwih[i] + alpha * dhbias[i];
       hbias[i] -= dhbias[i];
     }
}

    /*   重みの初期化及び表示   */
 void initialize()
{
   int i,j;

   for( i = 0; i < hunits; i++ )
{
    for( j = 0; j < iunits; j++ )
      {
        witoh[i][j] = urand();

      }
    hbias[i] = 0.0;
      }
   for( i = 0; i < ounits; i++ )
     {
       for( j = 0; j < hunits; j++ )
{
   whtoo[i][j] = urand();
}

       obias[i] = 0.0;
     }
```

```
}

  /*  テストパターンの実行及び結果 */
  void test_mode()
{
  int i,j;


  for( i = 0; i < patterns; i++ )
   {
   forward_propagation( i );

    if(i == 0){
     for( j = 0; j < ounits; j++ )
       printf("(1,1) %4.3f ¥n" ,out_out[j] );
   }
   else if(i == 1){
     for( j = 0; j < ounits; j++ )
       printf("(1,0) %4.3f ¥n" ,out_out[j] );
   }
   else if(i == 2){
     for( j = 0; j < ounits; j++ )
       printf("(0,1) %4.3f ¥n" ,out_out[j] );
   }
   else if(i == 3){
     for( j = 0; j < ounits; j++ )
       printf("(0,0) %4.3f ¥n" ,out_out[j] );

       }
    }
}
```

付録４　パターン認識のためのプログラム

```
   /* パターン認識のプログラム */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define MAX_L    10                    /*        学習上限        */
#define sig(u)   1.0/(1.0+exp(-u))       /*     シグモイド関数     */
#define Wmin     -0.30                 /*   重みの初期化の最小値   */
#define Wmax     0.30                  /*   重みの初期化の最小値   */
#define urand() ((float)rand()/(float)RAND_MAX * (Wmax - Wmin) + Wmin)

#define iunits    100                  /*   入力層のユニット数    */
#define hunits    16                   /*   中間層のユニット数    */
#define ounits    1                    /*   出力層のユニット数    */
#define patterns 12

#define eta       0.75                 /*       学習定数        */
#define alpha     0.8                  /*      安定化定数       */

int    out_in[patterns][iunits],tsignal[patterns][ounits];
float out_hid[hunits],out_out[ounits];
float witoh[hunits][iunits],dwitoh[hunits][iunits];
float whtoo[ounits][hunits],dwhtoo[ounits][hunits];
float hbias[hunits],dhbias[hunits];
float obias[ounits],dobias[ounits];

int nlpattern,ptest,testpattern;

main()
    {
      int   i,j,k,l;

      FILE   *fp;
```

```
        FILE   *fopen(),*fp1, *fp2, *fp3, *fp4, *fp5, *fp6;

        void   forward_propagation(),back_propagation();
        void   read_file(),initialize(),test_mode();

   srand(time(NULL)); initialize();

        read_file( fp );


        fp1 = fopen("pattern1","w");
        fp2 = fopen("pattern2","w");
        fp3 = fopen("pattern3","w");
        fp4 = fopen("pattern4","w");
        fp5 = fopen("pattern5","w");
        fp6 = fopen("pattern6","w");



        l = 0;

        for( i = 0; i <= MAX_L; i++ )
{
    l = l + 1;
    for( j = 0; j < nlpattern; j++ )
      {
        forward_propagation( j );
        back_propagation( j );
      }
    while( l == 1 )
      {
        for( j = 0 ; j < nlpattern ; j++ )
{
    forward_propagation( j );
    if( j == 0 )
      {
```

```
        for( k = 0; k < ounits; k++ )
        fprintf(fp1,"%d %4.3f ¥n",i,out_out[k]);
      }
    else if( j == 1 )
      {
        for( k = 0; k < ounits; k++ )
        fprintf(fp2,"%d %4.3f ¥n",i,out_out[k]);
        }
    else if( j == 2 )
      {
        for( k = 0; k < ounits; k++ )
        fprintf(fp3,"%d %4.3f ¥n",i,out_out[k]);
      }
    else if( j == 3 )
      {
        for( k = 0; k < ounits; k++ )
        fprintf(fp4,"%d %4.3f ¥n",i,out_out[k]);
      }
    else if( j == 4 )
      {
        for( k = 0; k < ounits; k++ )
        fprintf(fp5,"%d %4.3f ¥n",i,out_out[k]);
      }
    else if( j == 5 )
      {
        for( k = 0; k < ounits; k++ )
        fprintf(fp6,"%d %4.3f ¥n",i,out_out[k]);
      }
    l = 0;
}
      }
}
        fclose(fp1);
        fclose(fp2);
        fclose(fp3);
        fclose(fp4);
```

```
        fclose(fp5);
        fclose(fp6);


        test_mode();
      }

     /*  データを入力層から出力層に流す  */
void forward_propagation(int p)
{
   int i,j;
   float sum,u;

   for( i = 0; i < hunits; i++ )
{
    sum = 0.0;
    u    = 0.0;
    for( j = 0; j < iunits; j++ )
      sum += witoh[i][j]*out_in[p][j];
    u = sum - hbias[i];
    out_hid[i] = sig( u );
}
   for( i = 0; i < ounits; i++ )
{
    sum = 0.0;
    u    = 0.0;
    for( j = 0; j < hunits; j++ )
      sum += whtoo[i][j]*out_hid[j];
    u = sum - obias[i];
    out_out[i] = sig( u );
}
}

   /*      重みと閾値の修正      */
void back_propagation(int p)
{
```

```
  int i,j;
  float dwih[hunits],dwho[ounits];

  for( i = 0; i < ounits; i++ )
    dwho[i] = ( tsignal[p][i] - out_out[i])*out_out[i]*(1.0 - out_out[i]);

  for( i = 0; i < hunits; i++ )
    {
      for( j = 0; j < ounits; j++ )
{
  dwhtoo[j][i] = eta*dwho[j]*out_hid[i]+alpha*dwhtoo[j][i];
  whtoo[j][i] += dwhtoo[j][i];
  dwih[i] = out_hid[i]*( 1.0 - out_hid[i] )*dwho[j]*whtoo[j][i];
}
    }
 for( i = 0; i < ounits; i++ )
{
dobias[i] = eta * dwho[i] + alpha * dobias[i];
obias[i] -= dobias[i];
}

  for( i = 0; i < iunits; i++ )
{
   for( j = 0; j < hunits; j++ )
   {
    dwitoh[j][i] = eta * dwih[j] * out_in[p][i] + alpha * dwitoh[j][i];
    witoh[j][i] += dwitoh[j][i];
   }
}

  for( i = 0; i < hunits; i++ )
    {
      dhbias[i] = eta * dwih[i] + alpha * dhbias[i];
      hbias[i] -= dhbias[i];
    }
}
```

```
    /*  データファイルを読みこむ   */
 void read_file( fp )
      FILE *fp;
{
   int i,j;

   fp=fopen("learn","r");
   fscanf( fp, "%d", &nlpattern );
   for( i = 0; i < nlpattern; i++ )
{
    for( j = 0; j < iunits; j++ )
      fscanf( fp, "%d", &out_in[i][j] );
    for( j = 0; j < ounits; j++ )
      fscanf( fp, "%d", &tsignal[i][j] );
}
   ptest = nlpattern;

     fscanf( fp, "%d", &testpattern );
     for( i = nlpattern; i < nlpattern + testpattern; i++ ){
       for( j = 0; j < iunits; j++ )
fscanf( fp, "%d", &out_in[i][j]);
     }
 fclose(fp);
}

    /*   重みの初期化及び表示   */
 void initialize()
{
   int i,j;

   for( i = 0; i < hunits; i++ )
{
    for( j = 0; j < iunits; j++ )
      {
        witoh[i][j] = urand();
```

```
    }
   hbias[i] = 0.0;
    }
  for( i = 0; i < ounits; i++ )
   {
     for( j = 0; j < hunits; j++ )
{
  whtoo[i][j] = urand();
}

     obias[i] = 0.0;
   }
}

  /*  テストパターンの実行及び結果  */
  void test_mode()
{
  int i,j;
  char buf[80];

  printf("¥n");
  for( i = 0; i < nlpattern + testpattern; i++ )
   {
   forward_propagation( i );
   printf(" pattern %d = ",i+1 );

   for( j = 0; j < ounits; j++ )
    {
      printf("%4.3f¥n ",out_out[j] );
      if ( out_out[j] < 0.5 )
printf("  四角  ¥n" );
      else
printf("  三角  ¥n" );
    }
   }
}
```

付録５　パターン認識のための入出力データ１

6

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
1

0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1

0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0
0 1 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
1

0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0
0 0 1 0 0 0 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 1 1 1 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0

6

0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1
0 1 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 0 1 0 0
0 0 1 0 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 1 1 1 1 1 1 1 1 0
0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 1 1 0
0 1 0 1 1 1 1 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 1 1 1 1 0 1 0
0 1 1 0 0 0 0 1 1 0
0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 1 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 1
1 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

0 0 1 0 0 0 0 1 0 0
0 1 0 1 1 1 1 0 1 0
1 0 0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
0 1 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0 1
0 1 0 1 1 1 1 0 1 0
0 0 1 0 0 0 0 1 0 0

付録６　演算のためのプログラム

```
/* 0～5 の足し算 */
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define MAX_L   2000                    /*       学習上限       */
#define sig(u)  1.0/(1.0+exp(-u))       /*     シグモイド関数     */
#define Wmin    -0.30                   /*  重みの初期化の最小値  */
#define Wmax    0.30                    /*  重みの初期化の最小値  */
#define urand() ((float)rand()/(float)RAND_MAX * (Wmax - Wmin) + Wmin)
/*#define urand()  0.0   */
#define iunits   2                      /*  入力層のユニット数   */
#define hunits   100                    /*  中間層のユニット数   */
#define ounits   1                      /*  出力層のユニット数   */
#define patterns 72

#define eta      0.75                   /*       学習定数       */
#define alpha    0.8                    /*      安定化定数      */

float out_in[patterns][iunits],tsignal[patterns][ounits];
float out_hid[hunits],out_out[ounits];
float witoh[hunits][iunits],dwitoh[hunits][iunits];
float whtoo[ounits][hunits],dwhtoo[ounits][hunits];
float hbias[hunits],dhbias[hunits];
float obias[ounits],dobias[ounits];

int nlpattern,ptest,testpattern;

main()
    {
     int  i,j,k,l;

     FILE  *fp;
```

```
    /*    FILE   *fopen(),*fp1, *fp2, *fp3, *fp4, *fp5, *fp6;*/

    void   forward_propagation(),back_propagation();
    void   read_file(),initialize(),test_mode();

    /* srand(time(NULL))*/; initialize();

    read_file( fp );


    /*        fp1 = fopen("(0,0)","w");
    fp2 = fopen("(1,2)","w");
    fp3 = fopen("(2,4)","w");
    fp4 = fopen("(3,1)","w");
    fp5 = fopen("(4,3)","w");
    fp6 = fopen("(5,5)","w");*/



    l = 0;

    for( i = 0; i <= MAX_L; i++ )
{
  l = l + 1;
  for( j = 0; j < nlpattern; j++ )
   {
    forward_propagation( j );
    back_propagation( j );
   }
  /*   while( l == 10 )
   {
    for( j = 0 ; j < nlpattern ; j++ )
{
  forward_propagation( j );
  if( j == 0 )
   {
```

```
      for( k = 0; k < ounits; k++ )
      fprintf(fp1,"%d %4.3f ¥n",i,out_out[k]);
    }
  else if( j == 1 )
    {
      for( k = 0; k < ounits; k++ )
      fprintf(fp2,"%d %4.3f ¥n",i,out_out[k]);
      }
  else if( j == 2 )
    {
      for( k = 0; k < ounits; k++ )
      fprintf(fp3,"%d %4.3f ¥n",i,out_out[k]);
    }
  else if( j == 3 )
    {
      for( k = 0; k < ounits; k++ )
      fprintf(fp4,"%d %4.3f ¥n",i,out_out[k]);
    }
  else if( j == 4 )
    {
      for( k = 0; k < ounits; k++ )
      fprintf(fp5,"%d %4.3f ¥n",i,out_out[k]);
    }
  else if( j == 5 )
    {
      for( k = 0; k < ounits; k++ )
      fprintf(fp6,"%d %4.3f ¥n",i,out_out[0]);
    }
  l = 0;
  }
    }*/
}
  /*    fclose(fp1);
      fclose(fp2);
      fclose(fp3);
      fclose(fp4);
```

```
        fclose(fp5);
        fclose(fp6);*/


        test_mode();
}

    /*  データを入力層から出力層に流す  */
void forward_propagation(int p)
{
  int i,j;
  float sum,u;

  for( i = 0; i < hunits; i++ )
{
   sum = 0.0;
   u    = 0.0;
   for( j = 0; j < iunits; j++ )
     sum += witoh[i][j]*out_in[p][j];
   u = sum - hbias[i];
   out_hid[i] = sig( u );
}
  for( i = 0; i < ounits; i++ )
{
   sum = 0.0;
   u    = 0.0;
   for( j = 0; j < hunits; j++ )
     sum += whtoo[i][j]*out_hid[j];
   u = sum - obias[i];
   out_out[i] = sig( u );
}
}

   /*      重みと閾値の修正      */
void back_propagation(int p)
{
```

```
  int i,j;
  float dwih[hunits],dwho[ounits];

  for( i = 0; i < ounits; i++ )
    dwho[i] = ( tsignal[p][i] - out_out[i])*out_out[i]*(1.0 - out_out[i]);

  for( i = 0; i < hunits; i++ )
    {
      for( j = 0; j < ounits; j++ )
{
  dwhtoo[j][i] = eta*dwho[j]*out_hid[i]+alpha*dwhtoo[j][i];
  whtoo[j][i] += dwhtoo[j][i];
  dwih[i] = out_hid[i]*( 1.0 - out_hid[i] )*dwho[j]*whtoo[j][i];
}
    }
 for( i = 0; i < ounits; i++ )
{
dobias[i] = eta * dwho[i] + alpha * dobias[i];
obias[i] -= dobias[i];
}

  for( i = 0; i < iunits; i++ )
{
   for( j = 0; j < hunits; j++ )
   {
    dwitoh[j][i] = eta * dwih[j] * out_in[p][i] + alpha * dwitoh[j][i];
    witoh[j][i] += dwitoh[j][i];
   }
}

  for( i = 0; i < hunits; i++ )
    {
      dhbias[i] = eta * dwih[i] + alpha * dhbias[i];
      hbias[i] -= dhbias[i];
    }
}
```

```
    /*  データファイルを読みこむ   */
 void read_file( fp )
      FILE *fp;
{
  int i,j;

  fp=fopen("plearn30","r");
  fscanf( fp, "%d", &nlpattern );
  for( i = 0; i < nlpattern; i++ )
{
   for( j = 0; j < iunits; j++ )
     fscanf( fp, "%f", &out_in[i][j] );
   for( j = 0; j < ounits; j++ )
     fscanf( fp, "%f", &tsignal[i][j] );
}
  ptest = nlpattern;

    fscanf( fp, "%d", &testpattern );
    for( i = nlpattern; i < nlpattern + testpattern; i++ ){
      for( j = 0; j < iunits; j++ )
fscanf( fp, "%f", &out_in[i][j]);
    }
 fclose(fp);
}

    /*   重みの初期化及び表示   */
 void initialize()
{
  int i,j;

  for( i = 0; i < hunits; i++ )
{
   for( j = 0; j < iunits; j++ )
     {
       witoh[i][j] = urand();
```

```
    }
   hbias[i] = 0.0;
    }
  for( i = 0; i < ounits; i++ )
   {
     for( j = 0; j < hunits; j++ )
{
  whtoo[i][j] = urand();
}

     obias[i] = 0.0;
   }
}

  /*  テストパターンの実行及び結果  */
  void test_mode()
{
  int i,j,k,x,y;
  float z;
  char buf[80];
  y=0;
  printf("¥n");
  for( i =nlpattern ; i < nlpattern + testpattern; i++ )
   {
   forward_propagation( i );

   printf("%2.1f %2.1f   ",out_in[i][0],out_in[i][1]);

   for( j = 0; j < ounits; j++ ){
   k = out_out[j]*10+0.5;
   x=out_in[i][0]+out_in[i][1];
       printf("%4.3f   %d ",out_out[j],k);
       if(k == x){
printf("正解＼n");
y++;
       }
```

```
        else printf("不正解＼n");
    }
     }
   printf("  正解数  %d¥n",y);
   z=y;
   printf("  正解率  %5.3f%¥n",z*100/36);
}
```

参考文献

「C でつくるニューラルネットワーク」平野広美　パーソナルメディア
（第 2 章　階層型のニューラルネットワーク）
「ロボットの心　7 つの哲学物語」柴田正良　　講談社現代新書
（第 5 章　コネクショニズムって何？）
平成 10 年度信州大学卒業論文
「ニューラルネットワークによるパターン認識」倉地広志
平成 13 年度信州大学学士学位論文
「ニューラルネットワークによる判別回路」高田徳之

謝辞

今回の研究は、さまざまな人の助けがあって初めて完成を見ることが出来ました。

指導教官である竹下、長谷川先生には、知識もなく、不真面目ですらあった私に、一年に渡ってご指導を頂き、また、つまらない質問にも熱心に答えて下さいました。この場を借りて御礼を申し上げたいと思います。

また、質問、相談に答えで下さったり、励まして頂いたり、いろいろな点でご協力いただいた HE 研の院生の皆様、4 年生の皆様、そして同じ部屋で研究をしてきたテラヘルツ分光研の方々にも御礼を申し上げたいと思います。