PNC TN9410 95-035

C2

分置

ニューラルネットワークの微係数出力手法の開発

1995年3月

動力炉・核燃料開発事業団

大洗工学センター

複製又はこの資料の入手については，下記にお問い合わせください。
〒311-13　茨城県東茨城郡大洗町成田町4002
動力炉・核燃料開発事業団
大洗工学センター　システム開発推進部・技術管理室

Enquires about copyright and reproduction should be addressed to: Technology Management Section O-arai Engineering Center, Power Reactor and Nuclear Fuel Development Corporation 4002 Narita-cho, O-arai-machi, Higashi-Ibaraki, Ibaraki-ken, 311-13, Japan

動力炉・核燃料開発事業団（Power Reactor and Nuclear Fuel Development Corporation）

PNC TN9410 95-035
１９９５年　３月

ニューラルネットワークの微係数出力手法の開発

吉川　信治、　大草　享一、　小澤　健二

要　旨

本報告書は、多層構造を有するニューラルネットワークに、出力変数の入力変数に対する微係数を同時に出力する機能を付加する手法を開発、提案するものである。

多層構造を有するニューラルネットワークは、非線形の多変数関数を比較的容易に学習させられること、構築が容易なこと等の理由から、近年ますます応範囲の目的に使用されている汎用の関数学習／発生器である。

しかしながら、学習の終了したニューラルネットワークが発生する関数の、特定の入力変数に関する感度解析等は、近接した２点での出力値を用いた計算によって行われるのが通常であり、非線形性の強い場合でも精度を確保できる微係数導出法が求められてきた。

筆者らは、ニューラルネットワークの各ノードの特性関数として用いられるシグモイド関数が、微分値が自らの多項式で表現されることに着目し、通常のニューラルネットワークの処理に、四則演算のみを組み合わせた処理を付加することで、出力変数の入力変数に対する微係数を同時に算出する機能を実現する手法を開発した。

大洗工学センター基盤技術開発部先進技術開発室

PNC TN9410 95-035

March 1995

Derivative Value Outputs for Neural Networks

Shinji Yoshikawa, Kyoichi Okusa, Kenji Ozawa

Abstract

This report discusses a method to equip a multi layer neural network(NN) with a calculational function to derive differential values of the output parameters against the input parameters.

Multi layer NNs have been applied in various domains of engineering, because of easy construction, flexible interpolation of nonlinear multi-input functions, and some other preferable features.

However, derivatives of those output parameters have been approximately calculated by interpolating between two different output values. And new methods to guarantee the accuracy of the derivatives have been desired.

We payed their attention at sigmoid functions, which are commonly used to realize the nonlinear characteristics of nodes in NNs, and at one of important features of this function type that the derivative is represented by a polinomial of itself. And, we developed a method to add a calculational function to derive differentiated values of the output parameters to multi layer NNs, whose CPU cost is smaller than the original NNs.

Frontier Technology Section, Advanced Technology Division, Oarai Engineering Center

目次

1．目的

多層構造を有するニューラルネットワークは、その構築が容易で、非線形な多変数関数に対しても学習能力を有することから、今日広範囲な目的に使用されている。このニューラルネットワークが他の関数近似手法と比較して優位性を発揮するのは、入力と出力の組み合わせの例が多数存在する一方で、任意の入力に対する出力を得ることが困難な場合である。例えば、ある系に種々の境界条件を与えた場合の応答解析結果が多数蓄積されており、しかも新たな解析を行うには多大な計算コストがかかる、というような場合である。

このような場合に、解析例全てをニューラルネットワークに学習させることにより、任意の境界条件を与えた場合の応答を容易に近似的に求めることができる。しかしながら、工学的な立場からは、境界条件から応答を単に求めることは最終目的ではなく、多くの場合、境界条件のどのパラメータが応答に大きく影響するか、更には、最も望ましい応答を実現するための境界条件は何か、という情報が求められている。

このような要求に応えるためには、ニューラルネットワークに、出力変数の入力変数に対する微分値を出力する機能が備わっていることが望ましい。従来、このような微分値は近接２点間の内挿によって近似的に計算されてきた[1]が、非線形性の強い場合にも精度が保証されるような手法が求められてきた。

本開発の目的は、このような要求に応えるために、ニューラルネットワークに、出力変数の入力変数に対する微係数を同時に出力する機能を実現することである。

2．多層構造を有するニューラルネットワークの概要

多層構造を有するニューラルネットワークの概要を、図１に示すような、入力層、中間層、出力層の３層構造を有するものを例にとって説明する。入力層は入力変数と同数のノードを有し、各々入力された値をそのまま全ての中間層のノードへ伝達

する。中間層のノード数は、入力変数と出力変数の数や、学習させる関数の非線形性等によって適切な値が変化する。中間層の各ノードでは、入力層から伝達された各入力値にそれぞれ重み係数を乗じたものの合計に対するシグモイド関数（単調増加関数で、下限値と上限値を有する、ある点に対して点対象な形を有する関数である。例えば、S(x)=1/(1+exp(-x))が用いられる。）を出力層の各ノードへ伝達する。出力層には出力変数と同数のノードがあり、中間層のノードと同じ処理をした結果が各出力値となる。

次にニューラルネットワークに関数を学習させるための逆誤差伝播法について、文献[2]から引用して述べる。

誤差逆伝播学習アルゴリズムは、出力ｙｊとNNから式２－１で求められる推定値o_j^mの誤差の２乗和Ｅ、

$$E = \sum_j (o_j^m - y_j)^2 \qquad \text{式２－１}$$

を最小にする結合係数ｗを求めるアルゴリズムであり、ｗの修正を以下の式２－２に従って行うと、誤差の２乗和Ｅの値は減少していく。

$$\Delta w_{i,j}^{k-1,k} = -\varepsilon d_j^k o_i^{k-1}$$
$$d_j^m = (o_j^m - y_j) f'(i_j^m)$$
$$d_j^k = (\sum_i w_{j,i}^{k,k+1} d_i^{k+1}) f'(i_j^k) \qquad \text{式２－２}$$

ｍは出力層を表す

例えば、ある出力層のユニットの推定値が目標とすべき値より小さく、中間層のユニットの出力の値が正のときは、これらのユニット間の結合係数は大きくなり、出力層のユニットの推定値が大きくなる、すなわち目標とする値へ近づく。

3．微係数出力機能を付加する手法

3－1　ニューラルネットワークの各ノードの特性関数

通常、シグモイド関数が用いられる。例えば、

$s(x)=1/(1+\exp(-x))$　　式3－1

この関数の特徴として、微分が自分自身の多項式で表わされることが挙げられる。式1の場合は、

$s'(x)=-s(x)+s^2(x)$　　式3－2

従って、あるニューラルネットワークがある関数を良く近似している時、その同じ結合係数を使って、出力変数を各入力変数で微分した値を同時に出力するようなネットワークを構築することができる。

$w1_{ij}$を、入力層ノードiから中間層ノードjへの信号の重み係数、$w2_j$を中間層ノードjから出力層の当該ノードへの信号の重み係数とすると、

この出力値は$u=S(\Sigma w2_j S(\Sigma w1_{ij}x_i))$　　式3－3

となり、中間層ノードjの出力をh_jとすると、

$h_j=S(\Sigma w1_{ij}x_i)$　　式3－4

なので、

$u=S(\Sigma w2_j hj)$　　式3－5

と表わされる。

3－2　∂出力変数／∂入力変数の導出

$$\frac{\partial S(\Sigma w2_j h_j)}{\partial x_k}=S'(\Sigma w2_j h_j)\cdot \Sigma w2_j \frac{\partial h_j}{\partial x_k}$$　　式3－6

また、

$$\frac{\partial h_j}{\partial x_k}=\frac{\partial S(\Sigma w1_{ij}x_i)}{\partial x_k}=S'(\Sigma w1_{ij}x_i)\cdot w1_{kj} \qquad \text{式３－７}$$

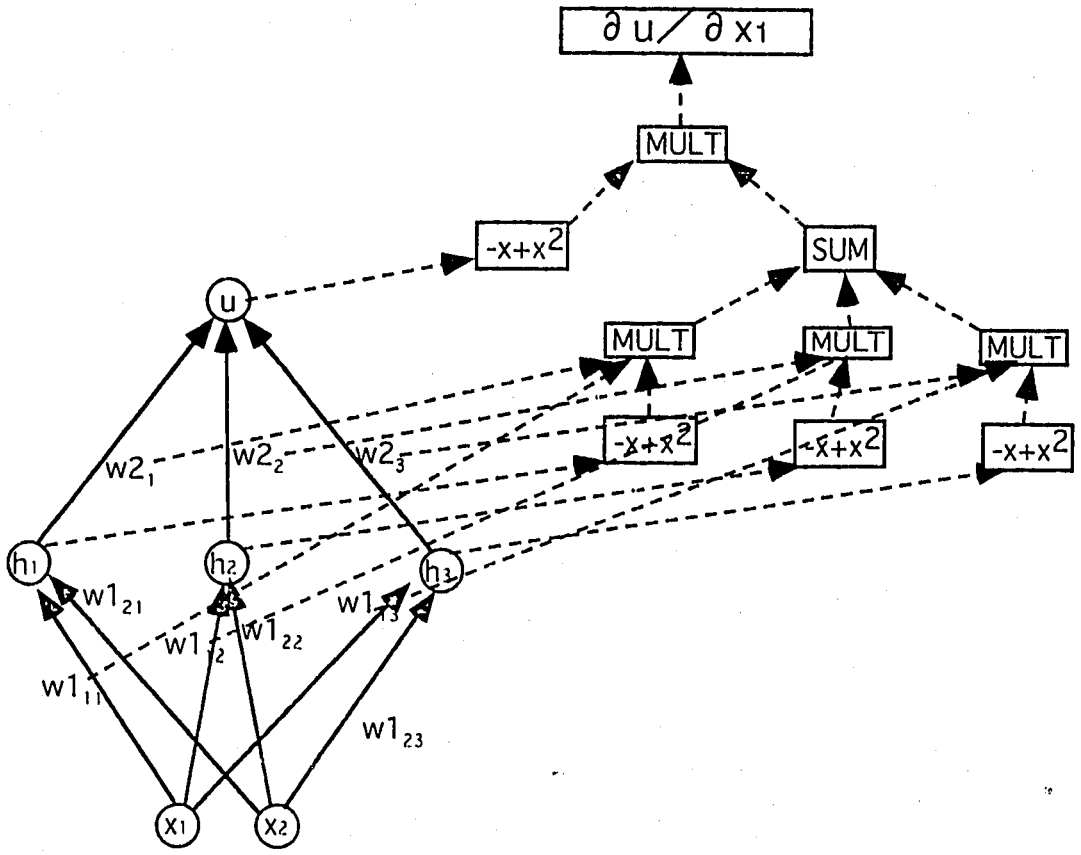
従って、

$$\frac{\partial S(\Sigma w2_j h_j)}{\partial x_k}=S'(\Sigma w2_j h_j)\cdot \Sigma[w2_j\cdot S'(\Sigma w1_{ij}x_i)\cdot w1_{kj}] \qquad \text{式３－８}$$

ここで、$f(x)=-x+x^2$とすると、

$$\frac{\partial S(\Sigma w2_j h_j)}{\partial x_k}=f(u)\cdot \Sigma[w2_j\cdot f(h_j)\cdot w1_{kj}] \qquad \text{式３－９}$$

従って、図２に示すように、２変数入力、中間層ノード２個のニューラルネットワーク（実線部）があった場合、上記の処理を行なう部分（点線部）を付加すれば、出力値の第１入力値に関する微分値を同時に出力できる。



y

図２：既存のニューラルネットワークに微分値出力機能を付加する手法

３－３　本手法の一般性

１）本報で取り上げた以外のシグモイド関数について

ニューラルネットワークには、本報で取り上げた以外のシグモイド関数、例えば

$S(x)=\tanh(x)$

$=(e^{x}-e^{-x})/(e^{x}+e^{-x})$が用いられる場合もある。この場合、

$S'(x)=4/(e^{x}+e^{-x})$

$=1-S(x)\cdot S(x)$　となるので、式３－９の$f(x)=-x+x^{2}$を$f(h)=1-x^{2}$に置き換えれば良い。この場合も微係数算出のための計算は四則演算に限られる。

このように、シグモイド関数が、自らの入力値に対する導関数が自分自身の多項式で表わされる場合には、本手法が適用可能である。

２）４層以上の構成を有するニューラルネットワークについて

シグモイド関数を S(x)、

S(x)の導関数として表わされる四則計算式を$f(S)=\partial S/\partial x$、

入力層を第0層としたとき第k層の第iノードの出力をh^{k}_{i}、

入力層の第ｋ－１層の第ｉノードから第ｋ層の第ｊノードへの信号の結合係数をw^{k}_{ij}、とそれぞれ表記したとき、

微係数算出のために付加するネットワークにおいて、

$$\partial h^{k}_{j}/\partial x_{m}=f(h^{k}_{j})\Sigma_{i}(w^{k}_{ij}\cdot\partial h^{k-1}_{i}/\partial x_{m})$$

ただし、

$$\partial h^{0}_{i}/\partial x_{m}=1\ (i=m)$$

$$\partial h^{0}_{i}/\partial x_{m}=0\ (i\neq m)$$

の各式によって帰納的に表現される演算処理を実行することにより、４層以上の構成を有するニューラルネットワークに対しても同様に微係数を出力することが可能である。

4．近接２点間の内挿による微係数計算手法との比較

４－１　計算量の比較

ここでは、本研究で開発した手法と、従来の近接２点間の内挿による微係数算出法との、計算量の比較を行う。

加減算、乗算、シグモイド関数の１回当たりの計算負荷をそれぞれa,m,sとする。

また、対象とするニューラルネットワークの入力層、中間層、出力層のそれぞれのノード数をNi,Nh,Noとする。

ニューラルネットワークの出力値計算１回に要する負荷：

$Ni \cdot Nh \cdot m+(Ni-1)Nh \cdot a+Nh \cdot s+Nh \cdot No \cdot m+(Nh-1)No \cdot a+No \cdot s$

$=\{Nh(No+Ni)\}m+\{Nh(Ni+No-1)-No\}a+\{Nh+No\}s$

本研究で開発した微係数算出のために新たに必要となる負荷

出力変数の数に無関係な処理に要する負荷：

$(m+a)+Nh(m+a)$

各出力変数につき必要な負荷：

$Nh \cdot m+(Nh-1)a+m$

合計：

$(m+a)+Nh(m+a)+No[Nh \cdot m+(Nh-1)a+m]$

$=\{(Nh+1)(No+1)\}m+\{No(Nh-1)+Nh+1\}a$

従って、シグモイド関数１回当たりの計算負荷sが加減算や乗算の１回当たりの計算負荷a,mに比して大きいことを考慮しなくとも、中間層と入力層のノード数Nh,Niが２以上であれば、本報で述べた手法が従来の近接２点間の内挿による手法よりも計算負荷が小さいことが明らかである。

４－２　精度の比較

従来の近接２点間の内挿による微分係数の計算では、図３に示すように、２階の微係数が大きな、即ち非線形性の強い箇所で、大きな誤差を生じる。

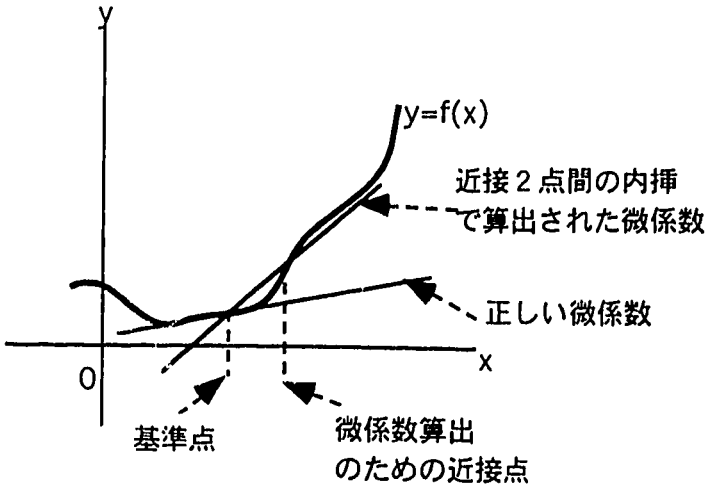


図３：近接２点間の内挿によって生じる微係数の誤差

また、２点間の距離を小さく設定しすぎると、ｅｘの計算における丸め誤差によって計算上の微係数が０となり、精度が不十分となる。これに対し、本報告書で述べた手法は、常に安定した精度で微係数を出力できる（添付２参照）。

５．考察

ここでは、精度の高い微係数出力機能が、第１章に述べた応用例以外にも持つであろう有効性について、ニューラルネットワークが使用される、より具体的な状況を想定して、考察を加える。

５－１　物性値データの関数化

従来型のニューラルネットワークに、蒸気に関する温度、圧力、及びエンタルピーに関するデータから、入力に圧力と温度、出力に蒸気のエンタルピーを与えて学習を行う。これを基に、本報告書で述べた微係数演算機構を付加したニューラルネットワークを構築することにより、エンタルピーの温度に対する微係数、即ち定圧比熱を求めることが可能となる。

数値解析コード等において、物性値計算ルーチンのエンタルピーと定圧比熱の整合性は、解析結果の精度、更には計算そのものの安定性に大きく影響する。従来の、実測データから近似式を作成する手法では、圧力や温度の範囲を複数に分割し、そのそれぞれに対して異なる近似式を割り当てる必要があった。この手法では、分割された各範囲の間で微係数の不連続性を回避することが困難で、この場合、１時間ステップである範囲から隣接した範囲へ温度や圧力の条件が推移する際に、エンタルピーと定圧比熱等の、一方が他方の微分値になっている関係にある物性値間に不整合を生じる。

この従来手法に比較して、本報告書に述べた手法では、一方が他方の微分値になっている関係にある物性値間の整合が保証されており、精度の高い安定な数値解析が可能となる。

５－２　ニューラルネットワークを利用した制御システムへの応用

近年、ニューラルネットワークの有する柔軟な学習機能に着目した、モデル適応制御等への応用例が多数報告されている[3],[4]。

制御の対象となる機器の各プロセスパラメータには、通常安全性確保等のため、上限値或いは下限値等の制約が存在すると考えられる。

このような場合、最小安全裕度を最大にするためには、各制御変数の、制約された各パラメータに対する感度を精度よく求められることは極めて有効であると考えられる。

特に、複雑な系を制御するための制御系を構築するアプローチとして近年ますます盛んに研究されている協調型制御システムにおいては、個別の各制御モジュールが、自ら担当する系の裕度と、その裕度の制御変数に対する感度を把握し、同じ階層に位置する他のモジュールと交渉しつつ裕度を平坦化することが不可欠であるので、本報告書で述べた手法の意義は大きいと考えられる。

6．参照文献

[1]Nabeshima, K. et al.:"Nuclear Plant Monitoring Using Real-time Learning Neural Network", Proceedings of the 2nd Specialist' Meeting on Application of Artificial Intelligence and Robotics to Nuclear Plants,pp157-168, 1994.

[2]麻生英樹著、「ニューラルネットワーク情報処理」産業図書、１９８８年、他

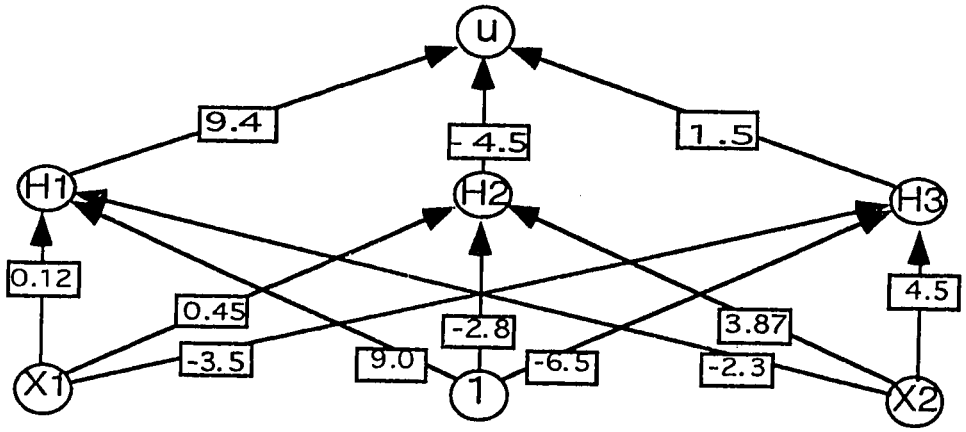
[3]大草享一他、「ニューラルネットワーク駆動型ファジイ推論を用いた空気冷却器温度制御の研究」、電気学会原子力研究会資料、１９９４年９月

[4]Ugolini, D. et al.:"Implementation of a Model Reference Adaptive Control System Using Neural Network to Control a Fast Breeder Reactor Evaporator",Proceedings of the 2nd Specialist' Meeting on Application of Artificial Intelligence and Robotics to Nuclear Plants, pp313-322, 1994.

付録　微係数算出法と近接２点補間法の比較実験

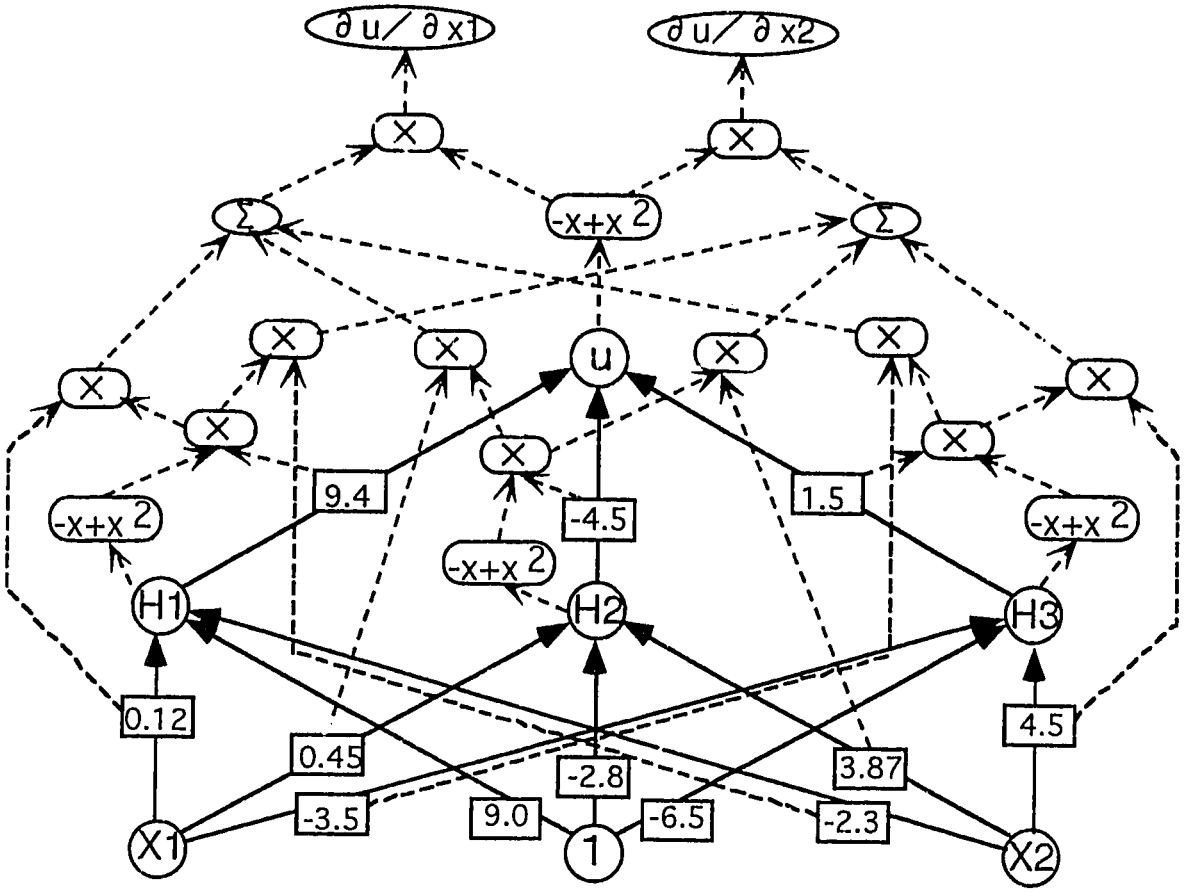
付録－Ａ．実験に用いたニューラルネットワーク

本報告書で述べた手法を検証するために、付録図－１に示すニューラルネットワークを、学習の終了したものと想定して計算機上に構築した。なお、入力層には、入力変数X1及びX2に対応するノードの他に、常に１を出力するノードを追加した。これは、中間層におけるシグモイド関数にバイアス（例：$f(x)=1/(1+\exp(-(x-\alpha)))$）を導入することと等価であり、この場合にも本報告書で述べた手法が有効であることを確認するためである。



付録図－１：学習が終了したニューラルネットワークの例

これに、本報告書で述べた手法により微係数を出力する演算機構を付加したものを付録図－２に示す。



付録図－２：微係数出力機能を付加したニューラルネットワーク

付録－B．計算例

説明のため、以下の２つの関数を定義する。

関数P(x1,x2,i)は、本報告書で述べた手法により求めた、「入力値がx1,x2の時の出力値ｕの、変数ｉに対する微係数。

関数Q(x1,x2,i,d)は、近接２点間の補間により求めた、入力値がx1,x2の時の出力値の、変数ｉに対する微係数の近似値で、次式で計算されている。

（ｉがx1の場合）

$$Q(x1,x2,i,d) = \frac{u(x1+d,x2) - u(x1-d,x2)}{2d}$$

以下に実験の結果を示す。

	x1に対する微係数 （ｉ："x1")	x2に対する微係数 （ｉ："x2")
本手法による値 P(1.5, -.0, ｉ)	-8.607964665117367e-9	-7.670916403817239e-8
Q(1.5, -2.0, ｉ, 0.5)	-8.681814711586355e-9	-1.334704408106901e-7
Q(1.5, -2.0, ｉ, 1.0e-9)	0.0	-1.110223024625156e-7

付録－C．プログラムリスト

本プログラムは、UNIXで動作するワークステーション（東芝製SparcLT AS1000）上のCommon Lispで作成した。

ソースリスト

```
(defun ldn()
 (load "nnwithd.lsp")
)

(defun sigmoid (x)
 (/ 1.0 (+ 1.0 (exp (- 0 x))))
)

(defun deri (x)
 (* x (- x 1))
)

(setq w1_11 0.12 w1_12 0.45 w1_13 -3.5
      w1_21 -2.3 w1_22 3.87 w1_23 4.5
      w1_01 9.0 w1_02 -2.8 w1_03 -6.5

      w2_1 9.4 w2_2 -4.5 w2_3 1.5
)

(defun h1 (x1 x2)
 (sigmoid (+ (* w1_11 x1) (* w1_21 x2) w1_01))
)
(defun h2 (x1 x2)
 (sigmoid (+ (* w1_12 x1) (* w1_22 x2) w1_02))
)
(defun h3 (x1 x2)
 (sigmoid (+ (* w1_13 x1) (* w1_23 x2) w1_03))
)

(defun u (x1 x2)
 (sigmoid (+ (* w2_1 (h1 x1 x2))
             (* w2_2 (h2 x1 x2))
             (* w2_3 (h3 x1 x2))
 )          )
)

(defun d1 (x1 x2)
 (* (deri (u x1 x2))
```

```
      (+
         (* w1_11 w2_1 (deri (h1 x1 x2)))
         (* w1_12 w2_2 (deri (h2 x1 x2)))
         (* w1_13 w2_3 (deri (h3 x1 x2)))
      )
   )
  )

  (defun d2 (x1 x2)
   (* (deri (u x1 x2))
      (+
         (* w1_21 w2_1 (deri (h1 x1 x2)))
         (* w1_22 w2_2 (deri (h2 x1 x2)))
         (* w1_23 w2_3 (deri (h3 x1 x2)))
      )
   )
  )

  (defun test (x1 x2 d)
   (print 'du/dx1)
   (print (list 'approximation
                    (/ (- (u (+ x1 d) x2) (u (- x1 d) x2)) (* d 2))
   )        )
   (print (list 'derivative--- (d1 x1 x2)))
   (print 'du/dx2)
   (print (list 'approximation
                    (/ (- (u x1  (+x2 d)) (u x1 (- x2 d))) (* d 2))
   )        )
   (print (list 'derivative--- (d2 x1 x2)))
  )
```