|||||
連載
|||||

ニューラルネット Part II:基礎入門編[†]

高木 英行^{*}

1. まえがき

前回はニューラルネットの全体像を手っとり早く掴むために、既存のニューラルネット解説や書籍を紹介した。今回は自らプログラミングして実験をしたい入門者を念頭において解説をしたい。ノウハウ的な面も後半に含める。定性的にならざるを得ない箇所や、問題に依存する箇所など、個別のノウハウについては、多々ご意見がお有りの方がおられるかもしれない。しかし、前半の解説だけでは、従来のニューラルネットの解説の範疇を越えることができず、あえて個性を出すためにプログラミングやノウハウにも及んだ次第である。

2. モデルの種類と特徴

2.1 ニューロンのモデル

実際のニューロン(図1)では、複数の他のニューロンから送られてきた電気パルスがシナプスで一旦化学物質の伝達に変換される。この変換された形の複数の情報はニューロンの本体に取り込まれ、ここで総合的に判断してニューロンの出力を決定する。具体的には、抑制性・興奮性の情報のすべてを加算し、ある閾値を越えた場合に神経パルスを発して、軸索を通じて他のニューロンに情報を伝達する、と考えられている。

ニューロンモデル(図2)はこの多入力一出力を基本としている。図3や図4では1つのニューロン出力が複数の他のニューロンへの入力になっているが、すべて同じ値を出力するものとする。更にニューロンの情報処理を、入力の総和が閾値を越えたら ON/OFF 信号を出力するようにモデル化している。この閾値関数は図2のようなS字状の sigmoid 関数がよく使われる。

図1 ニューロン

図2 ニューロンモデル

[†] Neural Networks, Part II: Introduction for beginners

^{*} Hideyuki TAKAGI

(Computer Science Division, University of California at Berkeley)

松下電器産業株式会社 中央研究所

Central Research Laboratories, Matsushita Electric Industrial Co., Ltd.

ニューロンモデルで最も重要な点は、可変的なシナプスの実現で、入力信号への重み付けとして表現される。この可変重みがニューラルネットの学習機能を実現する。

2.2 代表的なネットワークモデル

ニューロンモデルを複数結合し、ネットワーク構成をしたものがニューラルネットである。代表的なネットワークに、相互結合型ネットワーク(図3)と階層型ネットワーク(図4)がある。

相互結合型ネットワークは連想記憶や組合せ最適化問題によく応用される。ニューロンの出力が他のニューロンを回りめぐり形を変えて自分自身へ影響を及ぼす構造で、システム全体が平衡状態になって安定すると求める解が得られるよう設計する。具体的には第3章で説明する。

階層型ネットワークは入力パターンを別の出力パターンに変換する機能がある。backpropagationに代表される学習アルゴリズムが存在するので、応用に際してこのモデルがよく用いられる。代表的な応用分野に、パターン認識・フィルタリング・特徴抽出・非線形判別などがある。制御やファジィ論理の分野でもっぱら使われているニューラルネットはこのタイプである。

図3 相互結合型ネットワーク

図4 階層型ネットワーク

3. Hopfield 型モデルと巡回セールスマン問題

3.1 Hopfield 型ネットワーク

相互結合型のネットワークはフィードバック系なので、一旦動き始めると安定状態に落ちつくのか、リミットサイクルのように特定の状態が周期的に移動するのか、はたまたカオス状態になるのか分からない。ここで安定とは、各ニューロンがいつも同じ値を出力し続ける状態をいう。

ここでネットワークにある拘束条件を付けると、ネットワークの状態を、非線形力学や制御の分野でよく使われている Lyapunov 関数で表わすことができる。Lyapunov 関数は、状態進行にしたがって出力値が小さくなる関数なので、最後には極小値に到達する(ただし、局所的な最小値かもしれ

ない)。この拘束条件とは、各ニューロンは対称に結合されている(ニューロン i からニューロン j への結合重み係数 w_{ij} と、ニューロン j からニューロン i への結合重み係数 w_{ji} とが等しい)ことである。

もし組合せ最適化の問題において、最小値を求める式が Lyapunov 関数になるように問題をセットしておいてからニューラルネットを動かせば、安定状態になった時に組合せ最適化の解が得られることになる。組合せ最適化問題は、半導体回路や IC 基板の配線問題、マルチタスク OS の最適負荷決定、ポートフォリオなどいろいろ身近にある問題である。次節では具体的にどのように応用するかを、巡回セールスマン問題に適用してみよう。

その導入として、変数の説明を初めにする。

i 番目のニューロン出力 : v_i
 i, j 番目のニューロン間の重み : $w_{ij} = w_{ji}$
 j 番目のニューロンへの入力 (w_{0j} は閾値) :

$$U_j = \sum_{i=1}^N w_{ij} v_i + w_{0j}$$

j 番目のニューロン出力 :

$$v_j = \frac{1}{2} \left\{ 1 + \tanh \left(\frac{U_j}{\mu_0} \right) \right\}$$

ここで、 μ_0 は傾きを決める定数

以上の変数を持つ相互結合型のニューラルネットが非同期に状態を変えれば、次式で表わされるエネルギー E は Lyapunov 関数であるので、自然減少する。($w_{ij} = w_{ji}$ の拘束条件が大切)

$$E = -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N w_{ij} v_i v_j - \sum_{i=1}^N w_{0i} v_i$$

3.2 組み合わせ最小化問題への応用

(巡回セールスマン問題の場合)

巡回セールスマン問題とは、 n 都市を重複なく最短に巡回する訪問順位を決める古典的な組合せ最適化問題である。考えられる道順をしらみつぶしに探せば最適解が得られるのであるが、比較すべき道順は、 $(n-1)!/2$ 通りもあるため、都市数が多くなると現実的には計算できなくなってしまう NP 完全問題である。(ちなみに、20 都市の場合、約 6×10^{16} 通り。1 つの道順の距離計算を 200 ns で実行する高速マシンでも 380 年以上かかる)

(a) モデルの定式化

ニューラルネットでは次のように解く。まず、 $N \times N$ のニューロンを用意し、全ニューロンを相互に結合する。重み数は $N^2(N^2-1)$ 個になる。ここで、ニューロン v_{xi} が発火するということは「都市 x を i 番目に訪れる」ことを意味する。

次に、前節で示したニューロンの出力記号と、エネルギー E の式をこの問題のように、2 次元に拡張しよう。

$$\begin{aligned} U_{xi} &= \sum_{y=1}^N \sum_{j=1}^N w_{xi,yj} v_{yj} + w_{0,xi} \\ v_{xj} &= \frac{1}{2} \left\{ 1 + \tanh \left(\frac{U_{xj}}{\mu_0} \right) \right\} \\ E &= -\frac{1}{2} \sum_{x=1}^N \sum_{i=1}^N \sum_{y=1}^N \sum_{j=1}^N w_{xi,yj} v_{xi} v_{yj} \\ &\quad - \sum_{x=1}^N \sum_{i=1}^N w_{0,xi} v_{xi} \end{aligned} \quad (1)$$

(b) 評価関数の定式化

以上のモデルとは別に、最適解を求めるための条件式を作る。巡回セールスマン問題では、4 つの拘束式がある。すなわち、「同じ都市を 2 回以上訪れない」「同時に違う都市を訪れない」「さばら

	訪問順位					
	1	2	3	...	N	
都市 1	○	○	○	○	○	○
都市 2	○	○	○	○	○	○
都市 3	○	○	○	○	○	○
都市 4	○	○	○	○	○	○
⋮	○	○	○	○	○	○
都市 N	○	○	○	○	○	○

図 5 Hopfield 型モデルの記号の説明

図 6 巡回セールスマン問題のニューロン配置

ない」「巡回距離が最小」である。この拘束条件を見つけたことが問題を解く最大のポイントである。なお、A, B, C, D は任意の定数値である。

E_1 = 同じ都市を 2 度と訪れない条件用関数

$$= \frac{A}{2} \sum_{x=1}^N \sum_{i=1}^N \sum_{j \neq i}^N v_{xi} v_{xj}$$

(都市 x を i 番目と j 番目に訪れなければ $E_1=0$)

E_2 = 同時に 2 都市を訪れない条件用関数

$$= \frac{B}{2} \sum_{i=1}^N \sum_{x=1}^N \sum_{y \neq x}^N v_{xi} v_{yi}$$

(i 番目に都市 x と都市 y を訪れなければ $E_2=0$)

E_3 = 巡回をさぼらない条件用関数

$$= \frac{C}{2} \left(\sum_{x=1}^N \sum_{i=1}^N v_{xi} - N \right)^2$$

(訪問都市数が N の時 $E_3=0$)

E_4 = 訪問都市と前後の都市の距離の評価用関数

$$= \frac{D}{2} \sum_{x=1}^N \sum_{i=1}^N \sum_{y=1}^N d_{xy} v_{xi} (v_{y(i+1)} + v_{y(i-1)})$$

(d_{xy} は都市 xy 間の距離)

(c) 上記(a)と(b)の結合

巡回セールスマン問題として、評価関数 ($E_1 + E_2 + E_3 + E_4$) が最小になるようなニューロン出力 V を求めたい。評価関数が(1)式の E になるように重み w をセットしてニューラルネットを動かせば評価関数は自然減少する。評価関数が最小値になった時には N 個のニューロンのみが出力を出すようになっているはず(そのような評価関数を(b)で決めた)なので、そのニューロンが示す巡回する都市と訪問順位が得られ、組み合わせ最小化問題が解けることになる。

$$\begin{aligned} E &= \sum_{k=1}^N E_k \\ &= \frac{A}{2} \sum_{x=1}^N \sum_{i=1}^N \sum_{j \neq i}^N v_{xi} v_{xj} \\ &= + \frac{B}{2} \sum_{i=1}^N \sum_{x=1}^N \sum_{y \neq x}^N v_{xi} v_{yi} \\ &= + \frac{C}{2} \sum_{x=1}^N \sum_{i=1}^N \sum_{y=1}^N \sum_{j=1}^N v_{xi} v_{xj} \end{aligned}$$

$$\begin{aligned} &- CN \sum_{x=1}^N \sum_{i=1}^N v_{xi} + \frac{C}{2} N^2 \\ &+ \frac{D}{2} \sum_{x=1}^N \sum_{i=1}^N \sum_{y=1}^N d_{xy} v_{xi} (v_{y(i+1)} + v_{y(i-1)}) \\ &= \frac{1}{2} \sum_{x=1}^N \sum_{i=1}^N \sum_{y=1}^N \sum_{j=1}^N \{ A \delta_{xy} (1 - \delta_{ij}) \\ &+ B \delta_{ij} (1 - \delta_{xy}) + C \\ &+ D d_{xy} (\delta_{j(i+1)} + \delta_{j(i-1)}) \} v_{xi} v_{yj} \\ &+ \sum_{x=1}^N \sum_{i=1}^N (CN) v_{xi} + \frac{C}{2} N^2 \end{aligned} \quad (2)$$

(δ_{xy} はクロネッカ・デルタ)

(2)式の第3項は定数項で減少には関係ないので無視する。(2)式が(1)式の形式になるようにするには、次の重み係数を設定すればよい。

$$\begin{aligned} w_{xi,yj} &= -A \delta_{xy} (1 - \delta_{ij}) \\ &- B \delta_{ij} (1 - \delta_{xy}) \\ &- C \\ &- D d_{xy} (\delta_{j(i+1)} + \delta_{j(i-1)}) \\ w_{0,xi} &= -CN \end{aligned}$$

上式の重み係数値の時、(1)式エネルギー E の意味で巡回セールスマン問題の評価関数を自然減少させることができる。 E が減少すれば、評価関数が巡回セールスマン問題の解の時最小になるように選んでいるので、最終的に出力しているニューロンの番号が解答となる。

各ニューロンの出力値は逐時に求められニューラルネットの状態が時々刻々変化するが、あくまで重み係数は固定したままである。Hopfield モデルは入力データを用いて学習をしているわけではなく、この点が次節の backpropagation 学習アルゴリズムや自己組織化のような概念と異なる。

このモデルの応用上のポイントは、評価関数をいかに定義するかにかかっている。また、この解を求める解法上の難しさは重み係数を決定する定数(上例では $A \sim D$)を決定する方法がなく、かつ、一般にはこれらの値がニューラルネットの収束に大きな影響を与えがちである点である。

4. feed-forward 型モデル

4.1 モデルと変数の定義

feed-forward 型のネットワークは、入力層、中間層、出力層と層状構造をしている。各層内には複数のニューロンが配置され、隣接する層のニューロンと結合されている。この結合は入力層から出力層方向への一方通行の結合である。層内のニューロン間での結合や出力層が入力層にフィードバックするニューラルネットがあっても構わないわけであるが、本節では、そのような結合がないモデルを扱う。なお入力層は入力データの入力口であってニューロンは配置しない。また、中間層は一層である必要はない。なお中間層は hidden layer と呼ばれるため、隠れ層という用語が用いられる場合もある。

このネットワークの学習方法の 1 つに、back-propagation 法がある。出力値と目的値を最小にするよう、勾配法で重み係数を修正する方法である。目的値という教師データを与えるので、教師付き学習 (supervised learning) の 1 つである。

その反対に目的値を与えない学習方法を教師なし学習 (unsupervised learning) と呼び、各ニューロンの出力値をある規則に当てはめながら重み係数の修正を行う。Hebb の学習則や競合学習などが代表的な学習方法で、各種のバリエーションがある。

両者の中間が reinforcement learning で、ニューラルネットの出力が正しいかどうかだけを教師信号にする。教師付き学習のような具体的な教師信号を与えない方法である。人間がシステムの調整に介入する場合、うまくいっているかどうかは即座に答えられても、最適なシステムの振舞いを数値で答える事が出来ない場合がある。このような場合に適した学習方法である。

本解説は、数あるニューラルネットのモデルや学習方法から、わずか 2 つのモデルと back-propagation 学習法のみを取り上げる入門編であるので、他のモデルや学習法について更に詳しく

勉強されたい方には、教科書として評判の高い文献 [1] をお勧めする (書評は IEEE trans. NN, Vol.3, No.1, pp 158-159 (1992) にあり)。

さて話を本題に戻し、以下ではその back-propagation の説明をする。初めに、変数を定義する。

v_i^k : 第 k 層第 i 番目ニューロン出力

y_i : 出力層第 i 番目ニューロンへの教師データ

U_i^k : 第 k 層第 i 番目のニューロンへの入力
総和

$w_{i,j}^{k-1,k}$: 第 $(k-1)$ 層第 i ユニットから第 k 層第 j ユニットへの重み

$f()$: 閾値関数 (sigmoid 関数など)

N_k : 第 k 層のニューロン数

以上の記号を用いれば、ニューロンの出力と内部への入力値は次のように表わされる。

$$v_i^k = f(U_i^k)$$

$$U_j^k = \sum_{i=1}^{N_{k-1}} w_{i,j}^{k-1,k} v_i^{k-1}$$

第 0 層目が出力層であるニューラルネットを考える。このニューラルネットに学習データを入力した時の、ネットワークの出力 v^o と教師データ y との誤差 $E()$ は、

$$E(w) = \frac{1}{2} \sum_{j=1}^{N_k} (v_j^o - y_j)^2$$

と表わされる。この $E()$ を最も小さくするような重み係数 w を求める。この誤差は $w_{i,j}^{k-1,k}$ の関数なので、各重み係数で偏微分すれば、誤差が少なくなる方向がわかる (勾配法)。これが、back-propagation の基本である。

4.2 backpropagation 学習アルゴリズム

$E(w)$ の w についての最急降下方向を g とすると、重み修正量 Δw は

$$\Delta w = \epsilon g \quad (\epsilon \text{ は学習率})$$

$$\Delta w_{i,j}^{k-1,k} = -\epsilon \frac{\partial E}{\partial w_{i,j}^{k-1,k}}$$

$$= -\varepsilon \frac{\partial E}{\partial U_j^k} \cdot \frac{\partial U_j^k}{\partial w_{i,j}^{k-1}}$$

$$= -\varepsilon \frac{\partial E}{\partial U_j^k} v_{i,j}^{k-1}$$

ここで

$$d_j^k = \frac{\partial E}{\partial U_j^k}$$

とおけば、

(i) $k=0$ 、つまり出力層の場合

$$\begin{aligned} d_j^k &= \frac{\partial E}{\partial U_j^k} \\ &= \frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial U_j^k} \\ &= (v_{o,j} - y_j) f'(U_j^k) \end{aligned}$$

(ii) $k \neq 0$ 、つまり中間層の場合

$$\begin{aligned} d_j^k &= \frac{\partial E}{\partial U_j^k} \\ &= \sum_{h=1}^{N_{k+1}} \frac{\partial E}{\partial U_h^{k+1}} \cdot \frac{\partial U_h^{k+1}}{\partial v_{j,h}^k} \cdot \frac{\partial v_{j,h}^k}{\partial U_j^k} \\ &= \sum_{h=1}^{N_{k+1}} d_h^{k+1} \cdot w_{j,h}^{k,k+1} f'(U_j^k) \end{aligned}$$

以上をまとめると、

$$\Delta w_{i,j}^{k-1} = -\varepsilon d_j^k v_{i,j}^{k-1} \quad w_i$$

ただし

$$d_j^k = \begin{cases} (v_{o,j} - y_j) f'(U_j^k) & \text{出力層の場合} \\ \sum_{h=1}^{N_{k+1}} d_h^{k+1} \cdot w_{j,h}^{k,k+1} f'(U_j^k) & \text{中間層の場合} \end{cases}$$

なお閾値関数に、

$$f(x) = \frac{1}{1 + e^{-x/T}} \quad (T \text{ は閾値})$$

と sigmoid 関数を用いた場合は、

$$f'(x) = (1 - f(x))f(x)$$

となり、演算が簡略化できる。

また、慣性項を考え、

$$\Delta w(n) = -\varepsilon g + \alpha \Delta w(n-1)$$

とする場合も多い。 $\Delta w(n)$ と $\Delta w(n-1)$ が千鳥足で学習を進める場合(図12(a))、慣性項はそれを打ち消す働きをするので高速化に役立つ。

4.3 学習のイメージ

学習はニューラルネットの出力値と教師データとの誤差を尺度にして収束状態を観察し、学習終了の判定を行なっている。この誤差は重み係数を変更する度に变化するわけであるから、誤差は重み係数の関数である。重み係数がわずか2個から構成されるニューラルネットがあるとする、図7のような3次元空間の誤差曲面がイメージでき

図7 誤差曲面の例

る。実際には重み係数は数百数千になるので、高次の誤差超曲面になる。

ニューラルネットの学習とは、この誤差曲面上の最小点の重み係数座標を探し出すことである。ニューラルネットは sigmoid 関数のような閾値関数をユニットに内蔵するので、非線形関数である。すると誤差曲面には山谷が複数生じ、大局的な最小値を見つけることが困難になる。このいくつもの局所的な最小点(local minimum)に陥る事なく大局的な最小点(global minimum)を探し出す方法が非線形最適化と呼ばれる分野の手法である。本解説でも、第6章でその手法の一部を簡単に紹介する。

図を眺めていると、大局的な最小点を見つけだすための取り組み方法がいろいろ浮かぶであろう；例えば、素性のいい地点から学習を始められるように、初期値のよい決定方法はないか、逆に素性の悪い地点であることを判別し、初期化をし直す方法、山谷のデコボコを少なくする方法、局所最

小値付近の窪みに入ってしまった場合の脱出方法、学習方向の決定方法、学習によって進む長さの決定方法、等々。

5. 関数近似

ニューラルネットはどうして非線形な連続関数を近似できるのだろうか。単純なニューラルネットを用いて内部を解析してみよう。

入力と出力が1個しかない図8のニューラルネットを用い、図9の4つの▲を学習させる。横軸の x を入力に、縦軸の y を教師データとして学習するものである。図9は、その学習過程を示したものであり、480回の繰り返し学習で、全ての点を通る非線形な連続関数ができあがった。

次に、学習が完了したニューラルネットの内部を解析してみよう。4つの中間層ユニット出力は中間層と入力層との間の重み係数で決められる。

図8 関数近似解析実験のニューラルネット

各ユニット出力値とその関数値を図10に示す。

例えば $f(-22.3x-16.1)$ を出力する中間層第1ユニットは、入力信号 x に重み係数 -22.3 を乗じた値に定数項 -16.1 が加算され、sigmoid関数 $f()$ を通した値を出力する。

4つの中間層ユニット出力を重ねて表示すると、

図9 関数の近似過程

図10 学習後の中間層各ユニットと出力層ユニットの出力

ネットワークの最終出力が見えて来る(図 11)。重み係数の大きさに sigmoid 形状の伸縮を行い、負号で形状を反転するなどして最終出力値を合成しているのである。これで、ニューラルネットとは、sigmoid 関数を伸縮移動して重ね合わせることで、非線形な連続関数を近似する関数であることが視覚的に分かりいただけたことと思う。この特性をパターン識別や制御などの各応用分野で利用しているわけである。

この sigmoid 関数のように基になる関数を基底関数という。フーリエ級数展開では任意の関数を正弦波の組合せとして表わすように、ニューラルネットも閾値関数の組合せとして任意の関数を近似できる。そしてその個数を増やせば層をふやすことなく限りなく近似誤差を小さくすることができる^[2]。ニューラルネットにもいろいろなモデルがあって、sigmoid 関数以外の関数をユニットの閾値関数として用いてもこの近似が可能である。例えば、日本ではほとんど議論されていないが国際会議では盛んに議論される Radial Based Function(よく使われている日本語訳があるかどうか筆者は知らない)ではガウス分布関数を基底関数としている。

6. 学習の高速化

backpropagation 学習の高速化手法は色々提案されている^[3]。ここでは代表的な以下の方法について説明する。

図11 出力層ユニット出力と中間層各ユニットの出力

a. 慣性項の導入

ニューラルネットの学習とは重み空間の中の誤差最小の座標点を捜すことである。この手がかりは誤差曲面であるが、急峻な谷間で最急降下法を用いると図 12(a)のように谷間の側壁を上下してしまい、なかなか目指すべき誤差最小点に行き着かない。そこで、このような状況下の対策法として一回前の学習方向との合成ベクトルを考え学習方向の極端な変化を抑え、大局的な方向を向かせることが考えられる。これが慣性項である。

$$\Delta w(n) = -\epsilon g + \alpha \Delta w(n-1)$$

合成する割合 α の決定は経験による。更に2つ前の重み修正量を使おうという提案もある。さらに3つ4つと過去に遡って……、という考えも当然考えられるが、これはチェビシェフ加速法と呼ばれる非線形最適化手法の一つである。

b. 可変学習率 ϵ の導入

学習は学習ベクトルの方向と長さによって決まる。学習方向は通常の場合、最急降下法が用いられる。しかし、backpropagation 学習法は学習ベ

図12 慣性項の効果

クトルの方向を決定するアルゴリズムであり、学習率(学習ベクトルの長さ)を決定する方法ではない。そのため従来は、学習率に経験値を用いる場合が多かった。この値が最適であれば、図 12 の谷間の極小値を探索する方法に最急降下法を用いても容易に収束する。しかし一般的には最適値が得られないし、その最適値は学習の進行に伴って変化すると考えられるので、学習の進行状況に応じてこの学習率 ε を変えた方がよい。

よく用いられる方法が直線探索である。直線探索にも色々あるが、探索区間を 2 倍 2 倍と広げて行き、極小点が存在する区間に 2 次関数を当てはめて極小点の位置を求める DSC サーチ(図 13)については筆者らの評価がある^[4]。その他に、誤差減少量から求める方法も提案されている。簡単なレビューは文献 [4] を参照されたい。

図13 DSC サーチの探索方法

c. 学習方向の決定

一般の backpropagation は最急降下法を用いている。最急降下法はその計算時点においては最適であるが、大局的に最適であるという保証がない。そこで、他の決定方法として、共役勾配法やニュートン法が試みられており、それなりの効果が報告されている。筆者の非常に主観的な印象を述べさせていただければ、最急降下法(鈍牛)、共役勾配法(暴れ馬)、ニュートン法(重戦車)であろうか。ニュートン法は重み係数の 2 乗のメモリが必要になるのでモデルが少し大きくなるとメモリ負荷が耐え難いものになってくる。

d. 重み係数の初期値の決定

一般的には運を天に任せて乱数で初期化をする。統計的に決める方法、例えばネットワークが線形識別関数を構成するように固有値ベクトルを割り振る方法^[5]や重回帰係数を初期値にする方法^[6]が提案されている。これは、最初に統計的な素性のよい分離をしておいてから backpropagation で微調整をするようなものであるから、大局的な最小値に近いところからスタートすることになり、一般的には学習時間が短縮するだけでなく大局的な最小値に到達しやすいと考えられる。さらに、事前知識として探索すべき大体の位置が分かっているのであれば、事前知識を組み込み、大体の重み係数をセットすることが可能にある。

e. 学習データの提示方法

(1) データを入力する毎に学習する方法(最もオーソドックス)、(2) 学習データから代表データを作り、このデータで素性よく学習しておいてから、全体の学習データで学習を行なう、(3) 複数の学習データを提示後、平均誤差ベクトルに対して学習を行なう、などの方法が考えられる。(2) の前学習用の代表データとして学習データクラスターの重心点^[7]や、クラスターの外周に位置するデータの利用^[8]が試みられている。

f. 結合数の削減

一般に問題の難易度とモデルの情報処理能力が定量的に表わせないで、モデルサイズの決定は経験に頼っている。しかし、事前知識がモデルに組み込めたり、重み係数が 0 になる(役立っていない)ニューロンを切り捨てて行けば結合数が減るので、学習の演算が減る。ただし、プログラミング的に見ると、削減数に比例して高速になるわけではない。というのは特別な状況でない限り、結合がなくても演算の DO ループは減らないし、積和演算をしなくてもよいかわりにそのための判定が必要になるからである。

7. プログラミング

他の各学会誌のニューラルネット特集・解説記事で取り上げられることのなかったプログラミングについて章を割きたい。ここでは feed-forward 型のニューラルネットを backpropagation アルゴリズムで学習するプログラムを対象にする。

7.1 処理フローと配列

(A) メイン処理

一般に解説される backpropagation のアルゴリズムは、第3章と同じく backward の重み修正量の決定方法についての説明である。処理プログラムとして全体を完成させるためには図14の処理を作成する必要がある。

(B) 必要な配列

最低限必要な配列は、図15に示す各層の出力配列、教師データ配列、重み係数の配列である。その他にあった方がよい配列は一回前の重み係数をセーブする配列で、慣性項を使って高速化を図る場合に必要になる。あっても良いのが重み修正量を保持する配列で、学習の様子を観察するのに用いる。従ってメモリを減らす必要がある場合は本質的ではないので削っても構わない。

7.2 実行上のコメント

(A) 学習データ値の正規化

1つのニューロンへの入力と重み係数の積和の絶対値が大きくなりすぎて sigmoid 関数の飽和領域に入ると、学習が遅くなる(微分値が0に近い)。学習開始時はこのような状態をできるだけ防ぐ意味で、入力が大きすぎないようにした方がよい。入力の絶対値が大きくても重み係数との積和が0に近ければよいが、少し値が偏ると sigmoid 関数の飽和領域に達しやすい。

そこで、入力値を例えば $[-1, +1]$ の範囲に正規化しておいて学習を行なうことが多い。中間層ユニットへは入力値と重み係数との積が入力されるので、この(A)は次の(B)と表裏一体の関係がある。

(B) 重み係数の初期値幅

(A)と同様の理由で、1つのニューロンへの入力数が多い場合、正または負の方法の飽和領域に偏り易いので、数が多くなるほど、絶対値が小さい重み係数に初期化して学習を始めるのがよい。その指標としては文献[9]が参考になろう。

(C) ON/OFF の教師データ

学習では ON/OFF の教師データを用いる場合が多々あるが、この ON/OFF 値を (0,1) にする必要はない。2分判別をするのであれば (0.4,0.6) でもよい。こうすると収束が速くなる。

これを利用して初め (0.4,0.6) で大まかに学習させ、続いて (0.3,0.7)、(0.2,0.8) と順に学習条件を厳しくして全体では学習を速くするという方法も考えられる。

(D) 汎化性(過学習の問題とモデルの大きさ)

モデルサイズと学習能力の問題は、ニューラルネットの理論面における最もホットな話題である。ニューラルネットのユニット数が多すぎると、与えた問題を表現するのに必要な情報処理能力以上の表現能力を持ってしまうので、本質的な情報だけでなく、データの偏り(ノイズ)まで学習してしまう。未学習データに対しても十分な性能を保証できるようなネットワークサイズをどのように決定すべきか、現時点ではまだ満足できる実用的

図14 メイン処理フロー

規準は得られていない。現在の理論的アプローチには VC 次元を考える方法^[10]、統計量に基づく方法^[11,12]、学習曲線に注目する方法^[13]などがある。この分野の概略は文献^[14,15,16]をご参考いただきたい。

しかしながら、実際にニューラルネットを使う立場からは、理論面の進展を単に待つだけでなく、もう少し実用的な決定手段が欲しい。基本は、誤差曲線が収束する範囲内でできるだけユニット数を減らすことである。一般的には試行錯誤的に中間層のユニット数を決めて学習を行うことが多いが、もう少しアルゴリズム的にユニット数を変化させる方法が提案されている。1つは、少ない

ユニット数から学習を始めて順次ユニット数を増加させる方法と、多めのユニット数から学習を始めて順次削減して行く方法がある。これらの解説は文献 [17] に詳しい。なお一時期、ニューラルネット研究者間の電子メールでは、前者のモデルの1つである cascade-correlation^[18]について議論が活発であった。

(E) 追記学習

n 個のパターン識別をするニューラルネットの学習が終了したとする。その後、 $(n+1)$ 番目のパターンを追加して識別する必要がある場合、1 から学習を行なうのではなく、学習が終了した時の重み係数を初期値として学習を続ける方がよい。大局的な学習は終わっているので、乱数値で1から学習をやり直すより収束が速い。

図15(a) forward 計算に必要な配列

- 1) 直接重み係数配列を修正するならば必要ない
- 2) 慣性項をつけて加速させる場合に必要 ($\Delta w(n) = -\epsilon d^k_j v^{k-1}_i + \alpha \Delta w(n-1)$)

図15(b) backward 計算に必要な配列

8. まとめ

巡回セールスマン問題を例に Hopfield 型のモデルがいかに組合せ最適化問題に適用するか、と、backpropagation アルゴリズムの説明を通じてニューラルネットのプログラミングをいかにすべきか、を中心に実際のノウハウを織りまぜて解説した。

まだニューラルネットを使われた事がない方への最大のアドバイスは、まず排他論理和などのいくつかの小さな問題に適用してみることである。実際に自分で使ってみるとそれなりにニューラルネット適用の感覚が養われて来るので、道具として研究にとりいれたい方は、まず使ってみることをお勧めする。商業ベースのツールやパブリックドメインのツールが多く流れているが、高速化のための改良を自分なりにするには、自分自身でプログラムを作るのが一番よい。ニューラルネットプログラミング自体はそれほど大きな負担ではない(計算時間は適用問題とマシンの性能に大きく依存しますが)。本文がニューラルネットをまだ手がけていない人へのきっかけになれば幸いです。

参 考 文 献

- [1] J. Hertz, A. Krogh, and R. G. Palmer: Introduction to the theory of neural computation, Addison-Wesley Publishing Company(1991)
(日本語版は(株)トッパンから出版予定)
- [2] K. Funahashi: On the approximate Realization of continuous mappings by neural networks, Neural Networks, Vol.2, pp.183-192(1989)
- [3] 麻生英樹: 誤差逆伝搬学習の数理的性質, 電子情報通信学会技術研究報告 PRU89-14, pp.1-8(1989)
- [4] 高木, 阪上, 戸川: ニューラルネット学習における非線形最適化手法の効果, 電子情報通信学会論文誌 D-II, Vol.J74-D-II, No.4, pp.528-535(1991)
- [5] 仁木徹: 識別問題におけるバックプロパゲーションアルゴリズムの初期値決定法, 電気学会論文誌 C, Vol.110-C, No.3, pp.119-125(1990)
- [6] 李, 板倉: 重回帰係数の適応化による中国語音声認識, 音響学会春季講演論文集, 1-2-10(1988)

- [7] 城, 森: ニューラルネットの学習における新しいタイプの教師, 電子情報通信学会技術研究報告 PRU88-94(1988)
- [8] 高木英行: n 次元空間上の点分布からの外周データ簡易抽出法とその応用, 電子情報通信学会論文誌 D-II, Vol.J74-D-II, No.11, pp.1617-1620(1991)
- [9] 賈, 戸田, 臼井: ニューラルネットワークにおける逆伝搬学習アルゴリズムの初期値決定に関する一考察, 電子情報通信学会論文誌, Vol.J73-D-II, No.8, pp.1179-1185(1990)
- [10] E.B.Baum and D.Haussler: What size net gives valid generalization?, Neural Computation, No.1, pp.151-160(1989)
- [11] E. Levin, N. Tishby and S.A. Solla: A statistical approach to learning and generalization in layered neural networks, Proc. IEEE, Vo.78, No.10, pp.1568-1574(1990)
- [12] Y. Wada and M. Kawato: Estimation of Generalization Capability by Combination of New Information Criterion and Cross Validation, IEEE IJCNN'91, Vol.II, pp.1-6(1991)
- [13] S. Amari, N. Fujita and S. Shinomoto: Four types of learning curves, Neural Computation, Vol.4, pp.605-618 (1992)
- [14] 倉田耕治: 脳解明の数理的アプローチ, 数理科学, No.338, pp.5-9(1991)
- [15] 篠本滋: 学習の数理, 数理科学, No.338, pp.20-25(1991)
- [16] 倉田, 麻生: 神経回路網の理論的研究における最近の動向, 電子情報通信学会論文誌, Vol.J73-D-II, No.8, pp.1103-1110(1990)
- [17] 石川眞澄: ネットワークアルゴリズムの最近の話題, 計測と制御, Vol.30, No.4, pp.285-290(1991)
- [18] S. E. Fahlman and C. Lebiere: The cascade correlation learning architecture, in D. S. Touretzky(ed.) Advanced in Neural Information Processing Systems, Vol.2, pp.642-649, Morgan Kaufman(1990)

(1992年 6月27日 受 付)

[問い合わせ先]

Dr. Hideyuki TAKAGI
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720 USA
☎<+1>510-642-8015 ☎<+1>510-642-5775
e-mail: takagi@cs.berkeley.edu