

リカレントニューラルネットワークによる 単純なメロディーの記憶と想起に関する考察

荒木智行¹・堀 良幸²・山本富士男¹

1 情報工学科

2 (株) ニューメディア総研

Some Consideration on Memorizing Simple Melodies using Recurrent Neural Networks

Tomoyuki ARAKI¹⁾, Yoshiyuki HORI²⁾ and Fujio YAMAMOTO¹⁾

Abstract

This paper proposes a simple and quick method for searching melodies in music search systems using recurrent neural networks. The proposed method is based on the method proposed by M. Kinouchi and M. Hagiwara, which is based on memorization of melodies using recurrent neural networks. According to our examination of their method, memorization of melodies using their method was too unstable. Therefore, we tried to improve stability.

Improved points in our method are as follows: (1) By making the number for neural networks to learn variable, we derive a stable state of a neural network. (2) To confirm accuracy of memorization of a neural network, we introduce test-remembering-algorithm.

Keywords: music search system, recurrent neural network, test-remembering-algorithm

1. まえがき

コンピュータ技術の発展に伴い、従来から行われていた事務処理や計算などの実務的な処理だけではなく、趣味、ゲーム、情報家電などに対するコンピュータの需要が高まりつつあり、コンピュータを用いた音楽に対する需要も増大しつつある。こうした分野で使われる音楽は、その作成の過程からコンピュータが用いられることが多い。音楽に対するコンピュータの応用の一分野として、音楽の検索等があげられる。

従来、音楽データベースの検索は、曲名、歌詞、演奏者や作曲家などの文字情報からの検索が行われてきた。しかしながら音楽の検索においては、メロディーのような音情報としてのコンテンツによる検索が必要となる場合が多い。しかしながら検索キーとなる音楽情報は、曲の一部分の特徴的なメロディーなどである場合が多く、曲全体や小節全体である場合は希で、多くの場合、不完全な情報である場合がほとんどであると考えられる。

本研究ではこれらの点を補う手法として木内、萩原²⁾によって提案されているニューラルネットワークを用いたメロディーの記憶と想起の手法を基にして、メロディー、特にコンピュータゲームや携帯電話の着信メロディーなどのような単純なメロディーの記憶と想起を行う手法について考察する。ここで、本研究が単純なメロディーを対象とした理由は、上述したようにコンピュータを内蔵したゲームなどで使われる音楽では、音楽的に優れたものというより、画面上でのキャラクターの動きや、プレイヤーがゲームをするリズムを促すような単純なメロディーが用いられることが多く、今後、ゲームなどで使用される音楽の数は、爆発的に増加するものと考えられ、

「純粋な音楽としての音楽」の数を上回るものと考えられる。したがって、このような音楽をメロディーにより検索するようなシステムは、今後益々必要となるものと考えられる。

ニューラルネットワークによる想起は逐次的に行われるため、データベースとのパターンマッチングを用いる文字情報による検索方法と比べて、入力に対するインタラクティブな処理を行うことが可能であるという特性を持っている。しかし連想記憶モデルは、時間的なパターンを扱う際には一括入力によって静的パターンへの変換を行う必要がある。したがって、本質的な時系列処理にはならない。一方、ニューラルネットワーク自身に、時間的な処理能力を持たせる方法として、時間遅延を持つ結合を付加する方法がある。その一つにネットワーク内に遅延を伴うフィードバック結合を付加する方法がある。これをリカレントニューラルネットワーク (Recurrent Neural Network: RNN) と呼ぶ。そこで本論文ではRNNの一つである、中間層に自己フィードバックを持つネットワーク (Local Feedback Multilayered Network: LFMN) を用いてメロディーの記憶と想起を行うシステムについて考察した。LFMNを用いたメロディーの記憶は、階層型ニューラルネットワークの代表的な学習アルゴリズムであるバックプロパゲーション法を用いて、入力したメロディーの次の入力メロディーが出力される様に学習を行う。また、メロディーの想起は、ある一定の長さのメロディーを入力した後、その出力値をLFMNの外部でのフィードバックを行うことにより、次の入力値とする。その実験過程でメロディーの1回の記憶における誤差の推移に着目し、想起の正解率及び、記憶に掛かる時間に関して、木ノ内、

萩原により提案されている手法より優れた手法を提案し、実験的に検証を行った。実験により、提案したシステムでは、学習回数を可変長にし、“学習曲線の平行な場所を求めると”と“テスト想起”という二つのアルゴリズムをメロディの記憶に組み込むことにより、不安定なニューラルネットワークでも確実な想起が可能であることを示した。

2. 本研究で用いたニューラルネットワーク

2.1 自己フィードバック付きニューロン

図1は自己フィードバック付きニューロンである。Dとはディレイ素子である。vとは自己フィードバックにおける結合荷重である。これにより以前の出力を入力に加味することができる。このフィードバック結合を中間層に加えることにより、時系列の処理が可能になる。

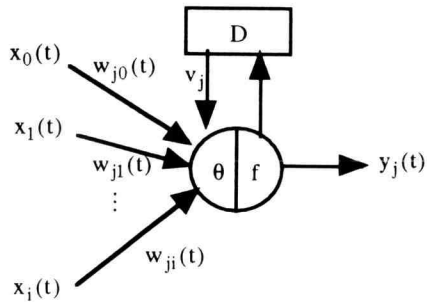


図1 自己フィードバックを持つニューロン

2.2 階層型ニューラルネットワーク

図2は階層型のニューラルネットワークである。本論文では、ニューラルネットワークとして中間層に自己フィードバックを持つものを使用する(図2)。以降、図2のネットワークをLFMN(Local Feedback Multilayered Network)と略記する。

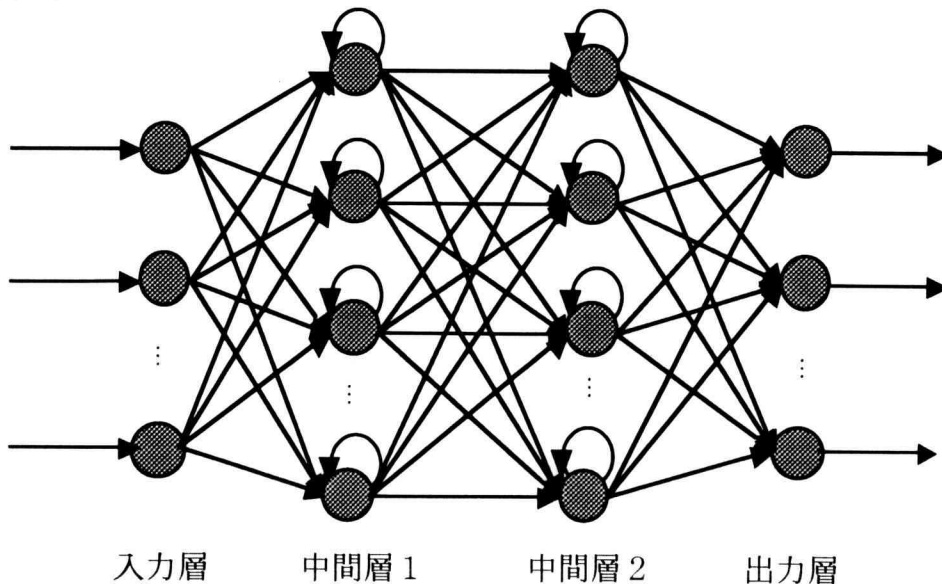


図2 自己フィードバック付き階層化ニューラルネットワーク (LFMN)

階層型ニューラルネットワークにおいて、データの入力部分は入力層、中間部分は中間層(あるいは隠れ層)、出力部分は出力層と呼ばれる。一般的には層の数の制限はない。情報の流れには一種の制約があり、入力層→中間層1→…中間層n→出力層の一方方向のみで、同一層内にあるニューロン間での直線的な情報交換手段は存在しない。本研究では、入力層を第0層、中間層1を第1層、…、出力層を第N層と呼ぶ。また同一層のニューロンを0ニューロン、1ニューロン、…、jニューロンと呼ぶ。

次に、各ニューロンの入出力の関係について説明する。LFMNを構成するとき、時刻tにおける第n層のjニューロンの出力は次式によって計算する。

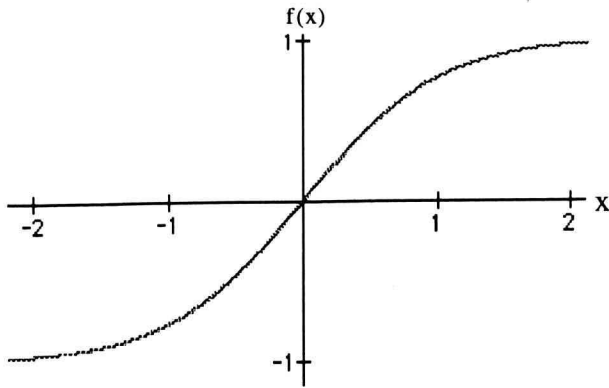
$$y_j^{(n)}(t) = f(u_j^{(n)}(t)) \quad (1)$$

$$u_j^{(n)}(t) = \sum_i w_{ji}^{(n)} y_i^{(n-1)}(t) + v_j^{(n)} y_j^{(n)}(t-1) + \theta_j^{(n)} \quad (2)$$

ここで、 $u_j^{(n)}(t)$ はニューロンへの入力、 $w_{ji}^{(n)}$ は第(n-1)層のjニューロンから第n層のjニューロンへの結合荷重、 $v_j^{(n)}$ は自己フィードバックの結合荷重、 $\theta_j^{(n)}$ は閾値である。また、 $n=N$ の場合、自己フィードバックがないので $v_j^{(n)}$ は0である。 $f(x)$ は次式で示されるシグモイド関数を使用した。

$$f(x) \equiv \tanh(x) \quad (3)$$

式(3)は図3にある様な曲線をとる。入力xの値によって出力 $f(x)$ が $-1 < f < 1$ の範囲の値をとることがわかる。この特性は、3.1項にて示すメロディの符号化において非常に都合が良い。

図3 $f(x) = \tanh(x)$ の曲線

2. 3 学習アルゴリズム¹⁾

連想記憶モデルにおける理想的なニューラルネットワークの条件とは、ある値 α を入力層に入力すると、得たい値 β が出力層に出力されることである。しかし、人間にも同じことがいえるが、十分に学習しないと望ましい答えが返ってこない。本項では、階層型のニューラルネットワークの学習法であるバックプロパゲーション法を示す。バックプロパゲーションアルゴリズムによるLFMNの学習は、以下のように行う。

(1) ネットワークの初期化

ニューロン間の結合荷重 w 、自己フィードバックの結合荷重 v 、各ニューロンの閾値 θ の初期値を、乱数を用いて小さな値に設定する。

(2) 入力パターンへの入力

まず、入力層に入力パターンを入力する。入力パターンは2. 2項に示したような入出力関係によって各ニューロンを通り、出力層へ向けて伝達する。これを前向きの処理とも呼ぶ。

(3) 出力層へ教師信号の入力

出力層へ、入力パターンに対応した教師パターンが与えられる。

(4) 誤差逆伝播による重みの学習

次式に基づいて w 、 v 、 θ を更新する。

時刻 t における誤差関数 $E(t)$ を次式によって定義する。

$$E(t) \equiv \frac{1}{2} \sum_i |d_i^{(N)}(t) - y_i^{(N)}|^2 \quad (4)$$

ここで、 $d_i^{(N)}(t)$ は第 N 層 (出力層) のニューロンへの教師信号である。この誤差関数 $E(t)$ を最小化することによって、ネットワークの学習を行なう。

そこで、 $w_{ji}^{(n)}$ を、 $E(t)$ の勾配と逆方向に変化させる。これにより、 $E(t)$ は順次小さくなり、最小化が行われる。したがって、時刻 t における $w_{ji}^{(n)}$ の更新量は次式によって表される。

$$\Delta w_{ji}^{(n)}(t) = -\eta \frac{\partial E(t)}{\partial w_{ji}^{(n)}} \quad (5)$$

ここで、 η は学習係数である。

次に、時刻 t における τ 時間前の入力に対する誤差項を次のように定義する。

(i) 中間層 ($n \neq N$)

$$\delta_j^{(n)}(t, \tau) \equiv \begin{cases} 0 & (\tau < 0) \\ f'(u_j^{(n)}(t - \tau)) \left(\sum_k w_{kj}^{(n+1)} \delta_k^{(n+1)} + v_j^{(n)} \delta_j^{(n)}(t, \tau - 1) \right) & (\tau \geq 0) \end{cases} \quad (6)$$

(ii) 出力層 ($n = N$)

$$\delta_j^{(N)}(t, \tau) \equiv \begin{cases} 0 & (\tau \neq 0) \\ -(d_j^{(N)}(t) - y_j^{(N)}(t)) & (\tau = 0) \end{cases} \quad (7)$$

ここで、 $f'(x)$ は、シグモイド関数 $f(x)$ の導関数に相当する関数である。

誤差項を用いることによって (5) 式は次のように表すことができる。

$$\Delta w_{ji}^{(n)}(t) = -\eta \sum_{\tau=0}^{T-1} y_i^{(n-1)} \delta_j^{(n)}(t - \tau) \quad (8)$$

ここで、 T は学習のために遡る時間 (学習考慮時間) である。 T の値は大きいほど勾配法に厳密に従うため、ある程度までは性能の向上が期待できる。しかし、1回の学習に要する計算時間は T にほぼ比例するため、学習が可能な範囲でできるだけ小さな T を設定することが好ましい。

同様に $v_j^{(n)}$ および $\theta_j^{(n)}$ の更新量は次のようになる。

$$v_j^{(n)} = -\eta \sum_{\tau=0}^{T-1} y_j^{(n-1)} (t - 1 - \tau) \delta_j^{(n)}(t - \tau) \quad (9)$$

$$\theta_j^{(n)} = -\eta \sum_{\tau=0}^{T-1} \delta_j^{(n)}(t - \tau) \quad (10)$$

以上はいずれも誤差関数 $E(t)$ の勾配に基づいている。

(8) 式、(9) 式、(10) 式により時刻 $t+1$ における各重みの値は、

$$w_{ji}^{(n)}(t+1) = w_{ji}^{(n)}(t) + \Delta w_{ji}^{(n)} \quad (11)$$

$$v_j^{(n)}(t+1) = v_j^{(n)}(t) + \Delta v_j^{(n)} \quad (12)$$

$$\theta_j^{(n)}(t+1) = \theta_j^{(n)}(t) + \Delta \theta_j^{(n)} \quad (13)$$

となる。

3. メロディーの記憶と想起

3. 1 前処理 — メロディーの符号化 —

ニューラルネットワークを用いてメロディーの記憶と想起を行うには、一つのニューロン自体は数値関数であるため、前処理をして音階をニューラルネットワークが理解できるような数字に変換しなければならない。

音符と休符を 9 ビットの $\{-1, +1\}$ パターンに符

号化する。具体的には、7つの音階（‘ド’から‘シ’）に7ビット（‘a’から‘g’）を割り当てる。同様に休符にも1ビット（‘r’）を割り当てる。これらの符号には音符の長さは含まれないため、「前の音が続いている」ことにもう一つのビット（‘-’）を割り当てる。表1にメロディの符号化の仕組みを示す。

表1 メロディの符号化

音階	符号化	パターン化
ド	a	{+1,-1,-1,-1,-1,-1,-1,-1,-1}
レ	b	{-1,+1,-1,-1,-1,-1,-1,-1,-1}
ミ	c	{-1,-1,+1,-1,-1,-1,-1,-1,-1}
ファ	d	{-1,-1,-1,+1,-1,-1,-1,-1,-1}
ソ	e	{-1,-1,-1,-1,+1,-1,-1,-1,-1}
ラ	f	{-1,-1,-1,-1,-1,+1,-1,-1,-1}
シ	g	{-1,-1,-1,-1,-1,-1,+1,-1,-1}
休符	r	{-1,-1,-1,-1,-1,-1,-1,+1,-1}
前音の続き	-	{-1,-1,-1,-1,-1,-1,-1,-1,+1}

には、図4に示すように、逐次的に $x(t)$ を入力し、次の入力である $x(t+1)$ を出力の教師信号として、LFMNを学習させる。即ち $d(t) = x(t+1)$ である。これにより(4)式は、次のように書ける。

$$E(T) = \frac{1}{2} \sum_i |x_i^{(N)}(t+1) - y_i^{(N)}(t)|^2 \quad (14)$$

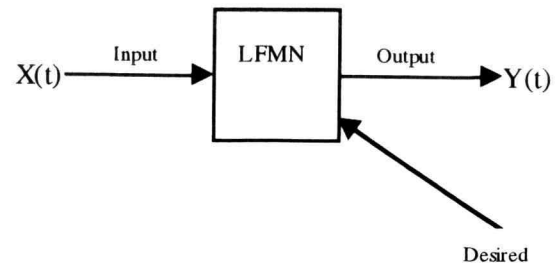


図4 記憶の仕組み

$x(t)$ が逐次的に与えられるために、入力 $x(t)$ から $x(t+1)$ への写像ではなく、以前の数時刻の入力 $x(t-\tau)$, $\tau = 0, 1, 2, \dots$ から $x(t+1)$ への写像が学習される。

(例1) “ドミソ” というメロディを記憶する。

まず始めに‘ド’という音階に相当するメロディコードを $x(t)$ に入力する。 $x(t)$ はLFMNを経て、 $y(t)$ という出力を返す。出力 $y(t)$ が‘ミ’という音階に相当するメロディコードを返せば良いのだが、学習が十分されてなければ理想どおりにはいかない。

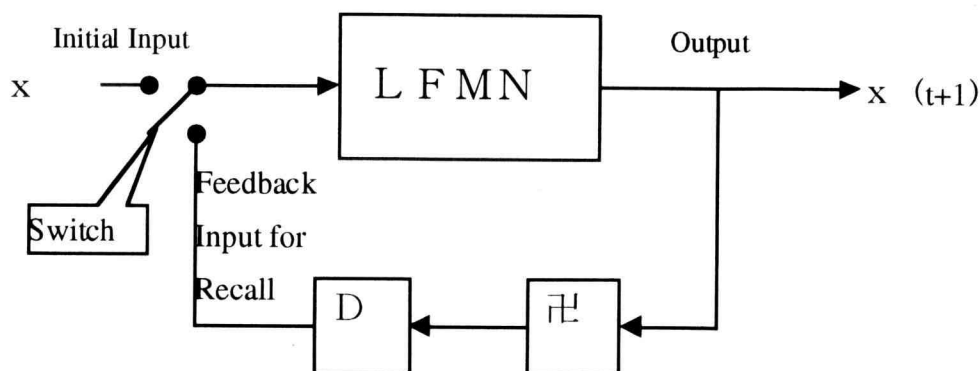


図5 想起の仕組み

3.2 記憶

符号化されたメロディを時系列 $x(t)$ とする。記憶時

そこで、望ましい出力（Desired Output）‘ミ’のメロディコードを教師信号としてLFMNに入力する。その後は

2. 3項に示した学習アルゴリズムによってLFMNの各重みを変えていく。次の‘ミ’という音階に対しても同じように行われる。最後の‘ソ’という音階の場合、出力がメロディーの終了を知らせるメロディコード

$\{-1, +1, -1, +1, -1, +1, -1, +1, -1\}$ を出力するようにする。この終了のメロディコードは、メロディーを想起する場合に特定の値にしないと、いつ終了するのかわからなくなる。ここまでの処理で1回の記憶を終了する。これを繰り返すことにより望ましい出力が得られるようになる。

3. 3 想起

3. 2項で示した学習を繰り返すと、 $x(t)$ を入力すると出力 $y(t)$ が次の入力 $x(t+1)$ と同じものになっていく。このような状態が安定して続く場合、想起が可能になる。図5に想起の仕組みを示す。ただし Γ は閾値処理装置である。

想起では、ある一定の長さのメロディーを入力した後その出力値を次の入力値にする。図5で言えば切替装置Switchが始めは横にあり、一定時間を過ぎると縦になる。Switchが縦になるとLFMNの出力値が閾値処理装置を通り、次の入力値となる。このようにLFMNの外部でのフィードバックを行うことにより、以降は記憶された時系列が再現されることになる。

このような動作は、メロディーの一部を手掛かりとして与えて、その続きを想起することに相当する。

4. 自己フィードバック付きニューラルネットワークの構成法

4. 1 構成法

ここからはメロディーの記憶と想起を実現するのに相応しいニューラルネットワークの形を模索していく。まず、図6は、2. 2項にて説明した自己フィードバックを持つニューロンを中間層に持つ場合と持たない場合を比較した学習曲線である。ここでいうERRORとは(4)式に示した誤差2乗和を1音符ごとの前向き処理ごとに足した総和である。環境としては両方とも、記憶するメロディーは付録の表Aのメロディー、ニューロン数は $9-60-60-9$ (入力層=9個—中間層1=60個—中間層2=60個—出力層=9個の意味)、学習定数 $\eta=0.01$ 、学習考慮時間 $T=2$ である。また、各重みの初期値は -0.5 から $+0.5$ の範囲で一様に分布する乱数を与えた。以上のような条件下で500回学習させた。図6から明らかなように、自己フィードバック有りの場合は学習するごとに収束していくのに対し、自己フィードバック無しの場合はあまり収束しない。以上のことから、自己フィードバックを中間層に持つことがメロディーの記憶を行うのに有効に働くといえる。

4. 2 中間層の数の検討

次に、中間層の層の数を検討する。図7は、中間層の層の数、及び中間層の層の中にあるニューロンの個数を変えて、誤差2乗和の推移を比較した学習曲線である。環境としては両方とも、記憶するメロディーは付録の表Aのメロディー、学習定数 $\eta=0.01$ 、学習考慮時間 $T=2$ である。また、各重みの初期値は -0.5 から $+0.5$ の範囲で一様に分布する乱数を与えた。また、

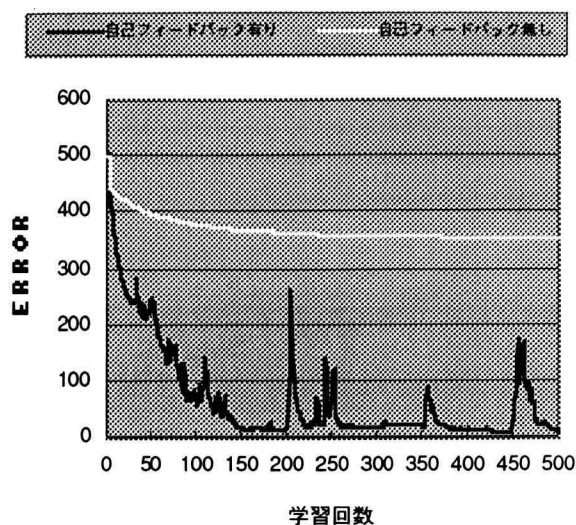


図6 自己フィードバック有りとなしとの比較

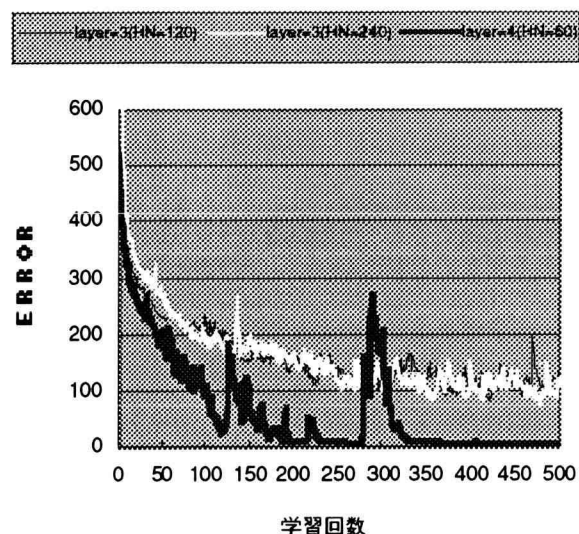


図7 中間層の数を比較した学習曲線

ニューロン数は最初の曲線 $layer=3$ ($HN=120$) は層の数が3層 (中間層は1層)、中間層の層内のニューロン数が120個、すなわち $9-120-9$ である。同じように、次の曲線が $9-240-9$ 、最後の曲線が $9-60-60-9$ である。以上のような条件下で500回学習させた。

図7から、中間層の層内のニューロン数を、中間層の層の数が一つの場合は二つの場合の2倍または4倍にしたにもかかわらず、二つにしたほうが性能はいいことがわかる。 $layer=3$ ($HN=120$) の500回までの誤差2乗和の最小値が80.19、 $layer=3$ ($HN=240$) が76.60に対し、 $layer=4$ ($HN=60$) は6.69であり、104回学習後の誤差2乗和は70以下になっていた。また、 $layer=3$ ($HN=120$) の2000回までの誤差2乗和を計測したところ、その最小値は39.67であった。そして、中間層の層の数が一つの場合、層内のニューロン数を2倍にしても性能は殆ど変わらない。このことにより、

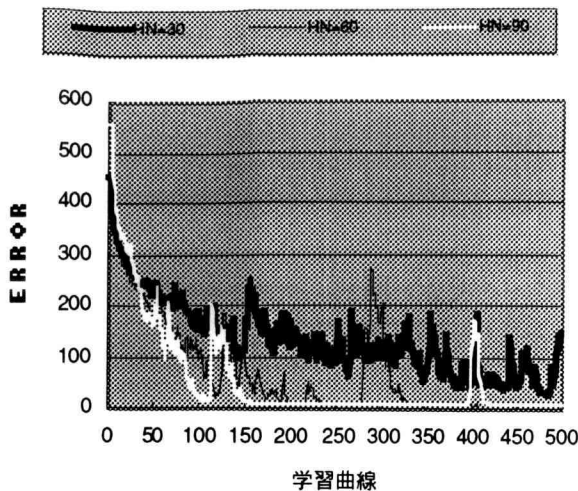


図8 中間層ニューロン数の比較

これ以上層内のニューロン数を増やしても、学習時間が増えるだけで性能は変わらないといえる。また、中間層の数が一つの場合にも見られるが、中間層の数を二つにした場合の学習曲線を見ると、時々、誤差2乗和がより跳ね上がり、同一学習回の一つの層の誤差2乗和よりも高くなっている。これは学習の振動と呼ばれる現象である。文献⁵⁾に示されている振動を低減させる方法を適用しても(詳細略)、 $layer=4$ ($HN=60$)の振動を抑えることはできなかった。これは、層の数を増やすことによって各重みの数が増え、さらに学習は出力層から後ろ向きに行われるため、その波及効果が大きいためだと思われる。しかし、 $layer=4$ ($HN=60$)の学習曲線はその後も十分に収束する。よって、これを中間層の層の数を二つにした場合の特徴と捉える。以上のことから、中間層の層の数が二つの場合の方が優れているといえる。しかしながら、学習時間が多い点が問題である。 $layer=3$ ($HN=120$)の500回までの学習時間が345.7秒であるのに対し、 $layer=4$ ($HN=60$)は631.0秒である。よって、階層型ニューラルネットワークを構成する場合、十分収束する最小の層数にすることが必要である。

4.3 中間層のニューロン数を検討

次に、中間層のニューロンの数を検討する。図8は、中間層の層の中にあるニューロンの個数を変えて、誤差2乗和の推移を比較した学習曲線である。環境としては両方とも、記憶するのは付録の表Aのメロディ、学習定数 $\eta=0.01$ 、学習考慮時間 $T=2$ である。また、各重みの初期値は -0.5 から $+0.5$ の範囲で一様に分布する乱数を与えた。ニューロンの数は、最初の曲線が $9-30-30-9$ 、次の曲線が $9-60-60-9$ 、最後の曲線が $9-90-90-9$ である。以上のような条件下で500回学習させた。

図8から、中間層のニューロンの数が30個の場合のみ、十分に収束してないのがわかる。ニューロン数が30個の場合の、500回までの誤差2乗和の最小値が27.89。それに対して60個の場合6.69、90個の場合7.83であった。これにより、中間層のニューロンの数が足りないと誤差2乗和は十分に収束されないといえる。また、中間層のニューロンの数が

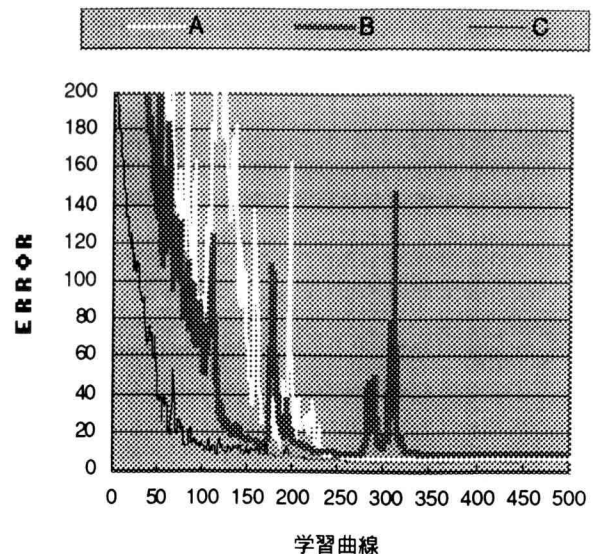


図9 メロディデータが異なる場合の比較

60個と90個の場合、跳ね上がりの場所には違いはあるものの、十分に収束されるのがわかる。しかし、ニューロンの数が1個増えるとそれだけ1回の学習時間が増える。以上のことから、中間層のニューロンの数は十分に収束する最小の個数にすることによって、より良いニューラルネットワークになる。

以上により、ニューラルネットワークによって単純なメロディの記憶と想起を行う場合、

- ・自己フィードバックを持つことは有効である。
- ・中間層の層の数は2層が良い。
- ・中間層のニューロンの数は今回用いたデータでは60個程。

と言える。

4.4 メロディデータが異なる場合を検討

最後に、メロディデータを変えた場合、どのような学習曲線をとるかを検討した。図9にその結果を示す。Aは付録の表Aのメロディ、Bは表Bのメロディを使用した。また、Cは表Aのメロディの#1と#2、表Bのメロディの#1と#2を使用した。メロディの数は、Aが7曲、Bが8曲、Cが4曲。音符の総数はAが248、Bが225、Cが103である。環境としては全て、ニューロン数は $9-60-60-9$ 、学習定数 $\eta=0.01$ 、学習考慮時間 $T=2$ である。また、各重みの初期値は -0.5 から $+0.5$ の範囲で一様に分布する乱数を与えた。以上のような条件下で500回学習させた。実験の結果、図9のような学習曲線が得られた。Aの500回学習後の誤差2乗和は6.54、Bが8.50、Cが3.72。Aの学習曲線の最小値は6.54、Bは8.50、Cは3.72であった。

5節で示すが、学習曲線は初期値によっても大きく変わってくる。図9の三つの曲線は両方とも同じ初期値をとるように、また表Aと表Bのメロディデータも音符の総数は近い値にしたのだが、学習曲線は大きく違うと言える。また、Cの誤差2乗和は他の二つよりも低い。これは音符の総数が低いために(4)式の Σ によって加算される総数が少ないためである。そして、AとBを

10000回学習させたところ、Aの最小値は6.12, Bは7.79であった。これにより、記憶させるメロディーによって学習曲線と最小値が違ってくるのがわかる。

5. 実験

5.1 木内、萩原による方法の評価

5.1.1 実験

ニューラルネットワークを用いたメロディーの記憶と想起は、3節で示したように(1)学習させたいメロディーをパターン化する→(2)メロディーを学習する→(3)メロディーを想起する、という手順で行う。図10(1), 図10(2), 図10(3)に、それらを行うアルゴリズムの概要を示した。

```
begin
  while (メロディーがなくなるまで繰り返す) |
    for (0～メロディーのサイズ) |
      各メロディーの解析
      for (0～入力層の層内にあるニューロンの数 (=9))
        解析したメロディーの入力パターン化
      |
    for (0～入力層の層内にあるニューロンの数)
      メロディーの終了コード (= |-1, +1, -1, +1, -1,
        +1, -1, +1, -1, | ) を入力パターンの
        最後に入力
    |
  for (中間層～出力層)
    for (0～層内のニューロンの数) |
      各層のニューロンにある重みの初期値を-0.5～+0.5
      の範囲でランダムで入力
    |
  end;
```

(1) メロディーのパターン化のアルゴリズム

```
begin
  while (無限ループ)
    |
    for (0～メロディーの数)
      for (0～メロディーの長さ)
        |
        前向きの処理
        誤差2乗和の計算
        各重みの更新
      |
    |
  if (学習回数が規定数 (=500) を超えたら)
    break
  |
end;
```

(2) メロディーの記憶アルゴリズム

```
begin
  while (想起を終了するまで) |
    for (0～想起の手掛かりメロディーの数) |
      手掛かりメロディーのコード化
      前向きの処理
    |
  前向きの処理による最後の出力パターンを符号化

  while (出力値が終了コードを返すか, メロディーサイズ
    が一定値 (=64) を超えるまで) |
    フィードバックした入力値を閾値処理
    前向きの処理
    前向きの処理による出力パターンを符号化
  |
end;
```

(3) メロディーの想起アルゴリズム

図10 木内、萩原のアルゴリズム

図11は付録の表Aのメロディーデータを入力データとした学習曲線である。各重みの初期値をそれぞれ変化させ、10個の学習曲線を得た。環境としては、ニューロン数 $9-60-60-9$, 学習定数 $\eta=0.01$, 学習考慮時間 $T=2$ である。また、各重みの初期値は -0.5 から $+0.5$ の範囲で一様に分布する乱数を与えた。また、想起時には表Aのメロディーデータにある最初の4音符を入力値とし、5音符目からはフィードバックによる入力とした。以上のような条件下で5000回学習させた。

初期値によって学習曲線は様々な変化をすることが分かる。さて、図11にある10個の学習曲線の中から3個の曲線に着目し、その想起の様子を見ることにする。

図12の5000回学習後の誤差2乗和は7.04, 最小値も同じく7.04であった。また、表2を見れば明らかのように、想起はすべて成功した。

図13の5000回学習後の誤差2乗和は10.38, 最小値は10.19であった。また、表3より、想起は“#3”のみ誤った出力を返した。さて、図12の学習曲線によると、誤差2乗和が単調に収束しないように見えるが、同じ初期値で2000回学習させると、その最小値は6.13であり、十分に収束するといえる。

図14の5000回学習後の誤差2乗和は47.09, 最小値は6.77であった。また、表4から明らかのように、想起はすべて失敗した。終了のメロディーコードを読み取っていないため、メロディーの想起が終わらずに同じメロディーを出力し続けているのが分かる。

図15は図11の10個の誤差2乗和を平均したものの学習曲線であり、図16はその想起結果である。

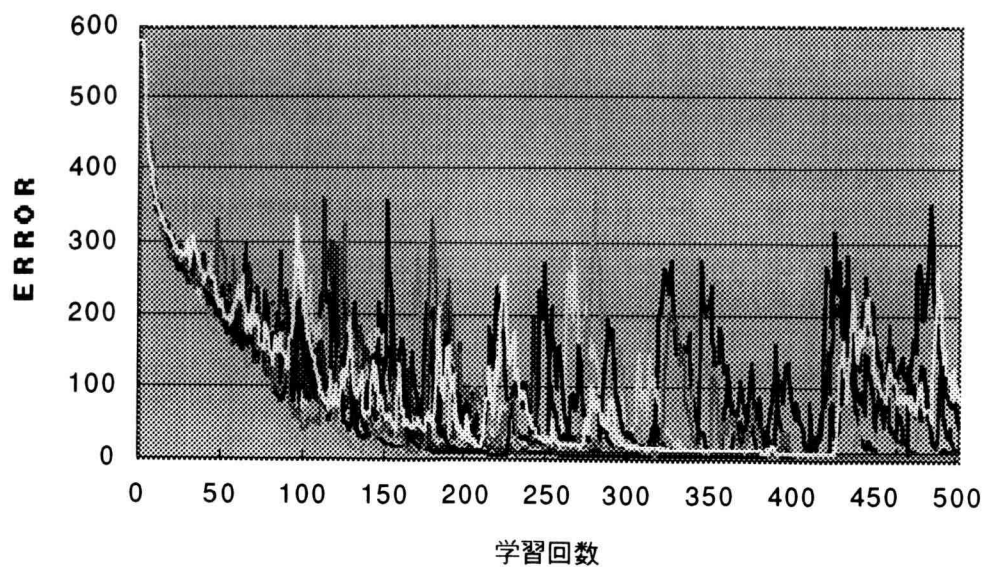


図 1.1 初期値を変えた場合の学習曲線

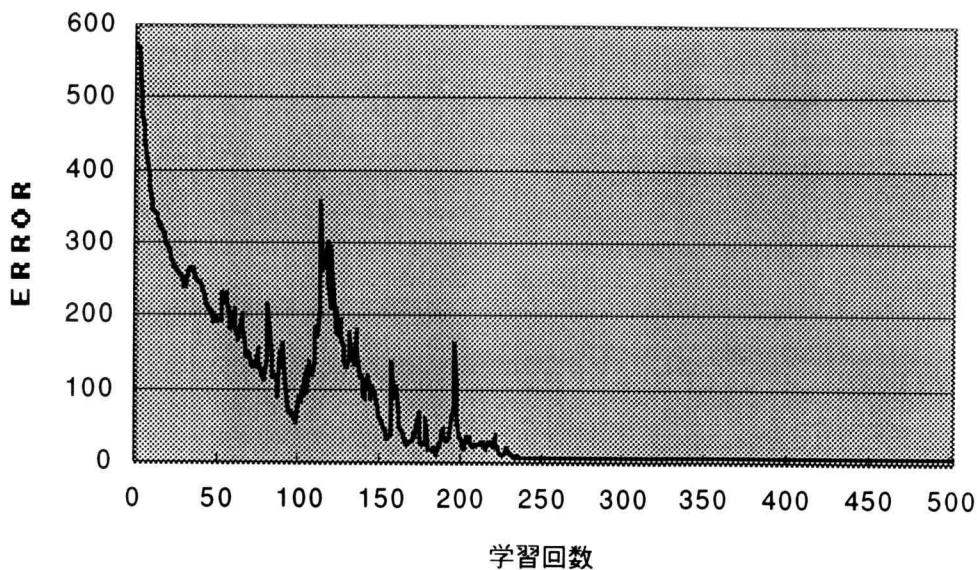


図 1.2 学習曲線の例 1

表 2 想起の例 1

#1	cde-fedrefgfedc-
#2	cdedefg-fedcd-e-fgfedcd-efedc----
#3	edcdcbabcdded-c-babcbded-cbaba---
#4	a-g-fefga---g-f-gfedc---f-g-g-e-fga-g---feded---
#5	bcd-cde-dcdef-e-fed-e-dcbcd-c---
#6	abc-d-cba---bcd-c-bab---cbaga-b-cdcba-g-abc-b-aga---
#7	gab-cba-babcb---cdc-bab-rabcb-a-g---

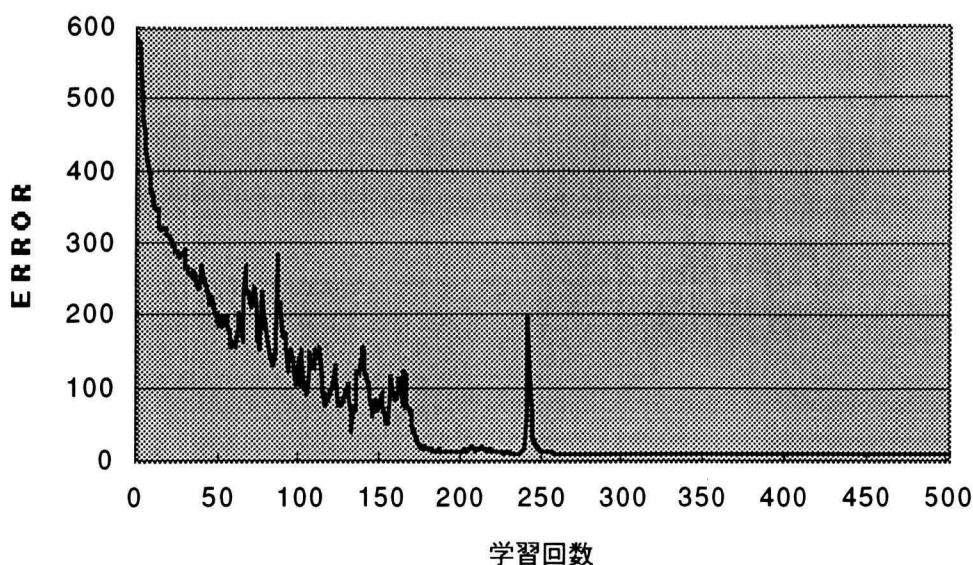


図 1.3 学習曲線の例 2

表3 想起の例2

```
#1 cde-fedrefgfdc-
#2 cdedefg-fedcd-e-fgfedcd-e-fedc---
#3 edcdcbabcded-cbaba---
#4 a-g-fefga---g-f-gfede---f-g-g-e-fga-g---feded---
#5 bcd-cde-dcdef-e-fed-e-dcbcd-c---
#6 abc-d-cba---bcd-c-bab---cbaga-b-cdcbag-abc-b-aga---
#7 gab-cba-babacd---cdc-bab-rabcb-a-g---
```

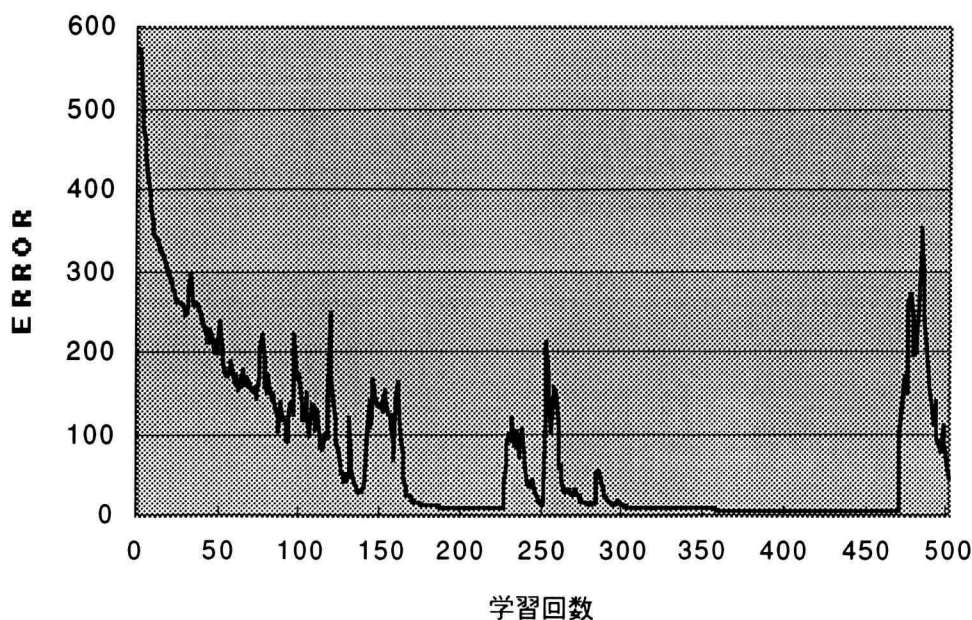


図 1 4 学習曲線の例 3

表 4 想起の例 3

```
#1 cde-fedrefgabaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#2 cdefgf-fedcd-e-fgaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#3 edcdba-bcaadadaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
#4 a-g-fedaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

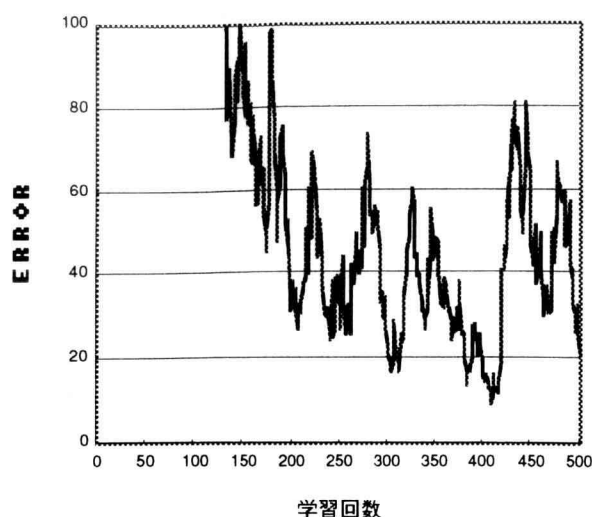


図15 図11の平均学習曲線

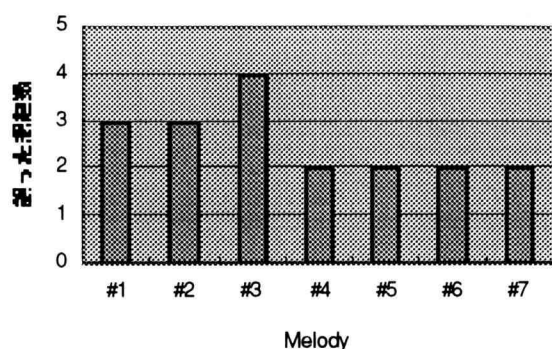
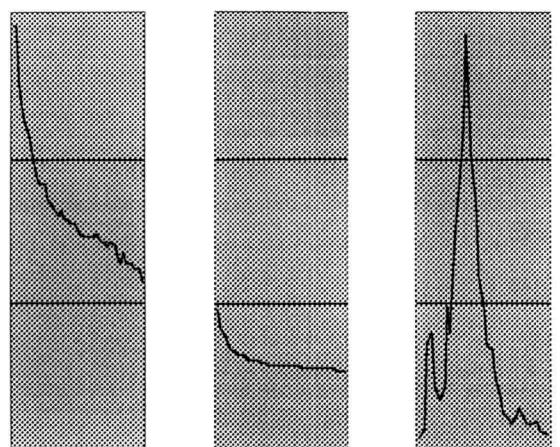


図16 図11の想起結果

5. 1. 2 評価と考察

図16から、多くの音を誤って出力している。70回の想起に対して18回誤った出力を返しており、これは約26%にあたる。また、表2、表3、表4により、今回のシステムは望ましい出力が得られるかどうか分からない不安定なシステムだといえる。ただし、500回



(1)下り坂 (2)平衡状態 (3)跳ね上がり

図17 学習曲線の三つの特徴

学習後の誤差2乗和が一桁の場合、その割合は0%であった。このことに着目し、次節ではより正確な想起をするための手法を提案する。

5. 2 木内、萩原の手法の改良

5. 2. 1 改良点

今まで見てきた学習曲線は、大別すると図17(1)、(2)、(3)にあるような3つの特徴的な部分に分けられる。図17(1)は初期の段階で、1回の学習によって誤差が少なくなる。

図17(2)はある程度望ましい出力が得られるようになり、式(4)の $E(t)$ の値も小さくなる。よって、各重みの変化量も小さくなり、出力が安定してくる。その結果、学習曲線の傾斜がなだらかになってくる。図17(3)は学習の振動の場合である。この状態で想起をすると表4のようになる。

5. 1. 1項の実験結果より、図17(2)の平衡状態の時に想起すれば正解率が高くなる。しかし、平衡状態の時に想起をしても望ましい出力が得られない場合がある。また、想起時の誤差2乗和が一桁の場合は完全な想起ができたが、誤差2乗和が2以上ならば想起が失敗する可能性もある。例えば、一つの出力層ニューロンの望ましい出力が-1の場合の出力が+0.99であり、別の出力層ニューロンの望ましい出力が+1の場合の出力が+0.98であったとする。これにより、式(4)は

$$E(t) = \frac{1}{2} (1 - (-1 - 0.99))^2 + 1 - (0.98)^2 \approx 2$$

となる。想起のとき、3. 3項にもあるように閾値処理をするのだが、最も1に近い値を+1とし、その他を-1とする。それから表1にあるメロディパターンと照らし合わせて符号のメロディ化を行い、次の入力とする。これにより+0.99が+1に、+0.98が-1となり、間違ったメロディが入力され、想起が失敗する。よって誤差2乗和が一桁の場合でも完全な想起が可能とは言えない。また図9のCのメロディを、初期値を変えて記憶・想起を行った。その結果、学習後の誤差2乗和が8.58であったが四つのメロディコードの内二つを間違えた。記憶させるメロディによって最小値が違ってくるためである。よって、誤差2乗和が一桁になったら想起をするという方法は現実的ではない。

ここで、1回の記憶に掛かる時間と想起に必要な時間を計測した。その結果、1回の記憶に必要な時間は約1.21秒であり、想起に必要な時間は約0.22秒である。学習時間の増加はニューラルネットワークにおける大きな問題点であるが、記憶に必要な時間に比べて想起に必要な時間はそれほど大きくないと言える。

そこで今回の改良法の提案として、“平衡状態を求め”と“テスト想起”という二つの評価法をメロディの記憶アルゴリズムに組み込む。具体的には、

1. 前の誤差2乗和と今の誤差2乗和を比較して、その差が0.1以下の状態が5回続けば平衡状態とする。
2. 想起を行う。もし全て望ましい出力が得られれば学習を終了する。もし得られなければ今の誤差2乗和が大きすぎる可能性が高いので、現在の誤差2乗和-0.5という次の目標値を保持し、学習を進める。これはむやみにテスト想起を行わない為の処置である。その後誤差2乗和が目標値よりも低ければもう

一度想起を行う。また、誤差2乗和が十分低いと想起に失敗してしまう場合や目標値が最小値よりも低い場合があるので、目標値が更新されてから10回ごとに保持している値に0.1を加算し、数値値の調整を行う。

という方法をとる。これにより、図10(2)にあるメロディーの記憶のアルゴリズムは図18のように改良される。

```
begin
while (無限ループ)
  for (0～メロディーの数)
    for (0～メロディーの長さ)
      前向きの処理
      誤差2乗和の計算
      各重みの更新
    }

    if (学習曲線が平行でない場合)
      if (前回と今回の誤差2乗和が0.1未満)
        1カウント
      else カウントした値をクリア

      if (5カウント) 平行状態
    }

    if (平行状態)
      if (今の誤差2乗和が目標値より低いなら)
        テスト想起
        if (テスト想起に失敗したら)
          今の誤差2乗和-0.5を保持
        }

      if (誤差2乗和が10回繰り返しても小さくならない)
        目標値+0.1
      }

    if (テスト想起に成功) break
  }
end.
```

図18 改良型メロディー記憶アルゴリズム

提案したアルゴリズムの学習の例を図19に示す。黒い線が学習曲線、白い線が目標値の曲線である。160回学習後に十分平行であると認識した。黒い線が白い線より下にくればテスト想起を行う。この例の場合、6回のテスト想起、256回の学習で完全な想起が可能になった。

次に、図11と同じように各重みの初期値をそれぞれ変化させ、メロディーの記憶と想起を行った。環境としては、入力データは付録の表Aのメロディーデータ、ニューロン数9-60-60-9、学習定数 $\eta=0.01$ 、学習考慮時間 $T=2$ である。また、各重みの初期値は-0.5から+0.5の範囲で一様に分布する乱数を与えた。また、想起時には表Aのメロディーデータにある最初の4音符を入力値とし、5音符目からはフィードバックによる入力とした。

以上の条件下で30回の試行を行った。まずは従来の方法で300回、500回、700回の学習を行い、想起した。次に改良型の記憶方法で学習を行い想起した。最後に、改良型の記憶方法では初期値によっては学習に時間が増大してしまう事があったので、1000回学習後も学習が終わらない場合は、次のテスト想起の時に完全な想起ができなくても学習を終了し、想起に移る方法を試した。その結果を表5および図20に示す。また、本研究で使用したマシンのスペックを表6に示す。

5.2.3 評価と考察

表5より、メロディーの学習回数を可変的にすることによりどのような初期値であっても完全な想起が可能であること、平均の学習時間においても500回固定で学習するよりも速いことが示された。また、学習回数を限定する方法は完全な想起はできなくなったが、改良型の場合に見られた極端に学習時間が長くなる場合の学習を途中で止めることにより、より短い時間でのほぼ確実な想起(99.5%の正解率)が可能である。両方の方法とも、従来型よりも正確な想起が可能になったと考えられる。

6. むすび

リカレントニューラルネットワークの一つであるLFMNを用いてメロディーの記憶と想起を行うシステムについて考案した。実験により、提案したシステムでは、

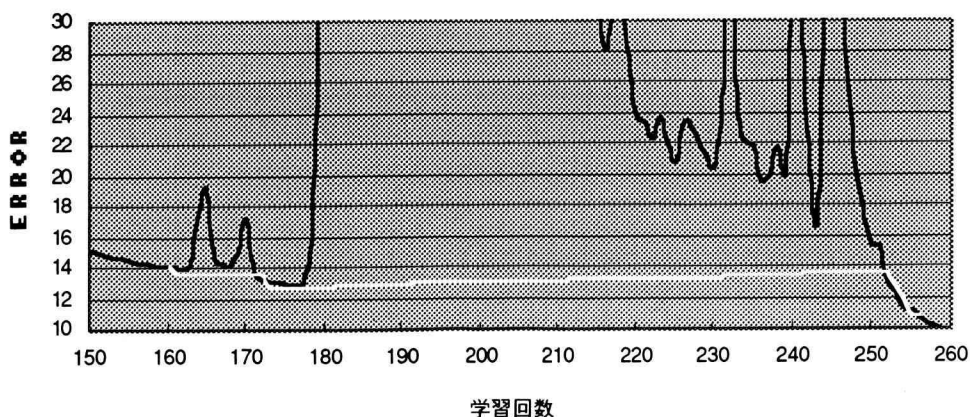


図19 改良型メロディー記憶アルゴリズムによる例

表5 学習時間とい想起正解率の比較

学習方法	300回	500回	700回	改良型	改良型 (学習回数限定)
平均学習時間 (秒)	346.1	631	804.5	551.9433	435.0633
最大学習時間 (秒)	—	—	—	4689.1	1252.3
最小学習時間 (秒)	—	—	—	204.9	204.9
平均学習回数	300	500	700	473.4	374.0677
最大学習回数	—	—	—	3994	1083
最小学習回数	—	—	—	177	177
想起正解数 [回]	7 6 5 4 3 2 1 0	13 5 7 0 1 2 4 3	16 4 3 1 1 1 0 4	30 0 0 0 0 0 0 0	29 1 0 0 0 0 0 0
想起正解率 (%)	69	65.7	76.2	100	99.5

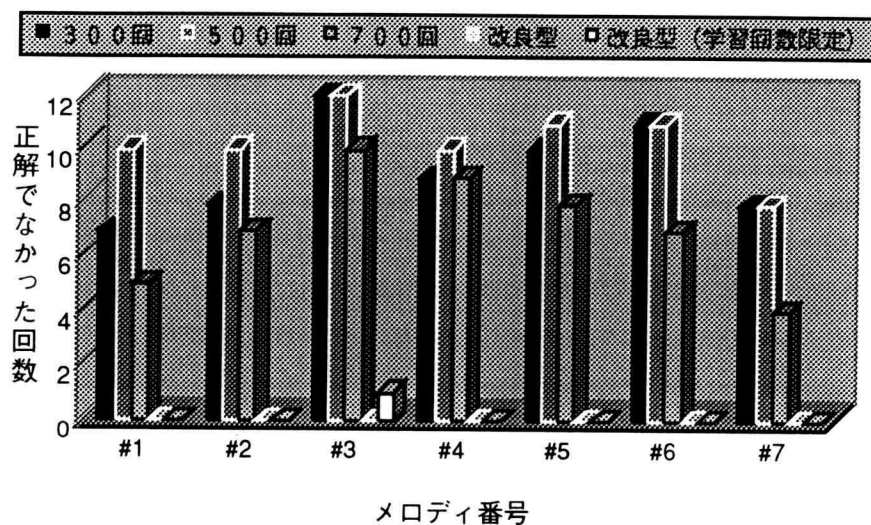


図20 誤った音の想起数

表6 使用したマシンのスペック

CPUの種類	PentiumII
クロック数	266MHz
メモリ	64MB
コンパイラの種類	Visual C++ 6.0

学習回数を可変長にし，“学習曲線の平行な場所を求める”と“テスト想起”という二つのアルゴリズムをメロディーの記憶に組み込むことにより，不安定なニューラルネットワークでも確実な想起が可能であることを示した．提案したシステムでは入力を与えるにつれて，次第に次の入力と一致した音を出力し始める．

今後の課題としては，実際のメロディー検索システムに応用するためには曲数を増やさなければならないが，学習時間が膨大になってしまう．メロディーデータの大規模化に対する対策が必要である．

最後に，音楽情報の時系列処理に関して多くの参考意見を頂きました情報工学科徳弘一路先生に感謝致します．

参考文献

- 1) 木ノ内誠，萩原将文．複素ニューロンによる時系列の学習．電気学会論文誌，Vol.116-C，No.7，pp.748-754 (1996)．
- 2) 木ノ内誠，萩原将文．複素リカレントニューラルネットワークを用いたメロディーの記憶と想起．情報処理学会論文誌，Vol.39，No.5，pp.1232-1239 (1998)．
- 3) 萩原将文．ニューロ・ファジィ・遺伝的アルゴリズム．産業図書，1994．
- 4) 上坂吉則．ニューロコンピューティングの数学的基礎．近代科学社，1993．
- 5) 臼井支朗，岩田彰，久間和生，浅川和雄 編著．基礎と実践ニューラルネットワーク．コロナ社，1995．
- 6) 株式会社ほにーてーる．ケータイ着メロ♪ドレミBOOK 2．双葉社，1998．

付 録

表A メロディーデータ 1²⁾

#1	cde-fedrefgfedc-
#2	cdedefg-fedcd-e-fgfedcd-efedc----
#3	edcdcba-bcded-c-babcbded-cbaba---
#4	a-g-fefga---g-f-gfede---f-g-g-e-fga-g---feded---
#5	bcd-cde-dcdef-e-fed-e-dcbcd-c---
#6	abc-d-cba---bcd-c-bab---cbaga-b-cdcba-g-abc-b-aga---
#7	gab-cba-babcb---cdc-bab-rabcb-a-g---

表B メロディーデータ 2⁶⁾

#1	abad-c-fa-b-rbbababae-f-
#2	f-edcrdcbagargaf-d-c-rcddeffe
#3	cdecfedcbdbfe-rfrgfcbbc-e-d-
#4	ararafgrarabaccr-cfagfgfgaa---
#5	e-g-erdeddbcrabcccrccaed-
#6	bbcdf-raabcb-rbbcdf-rffede-
#7	fedfearabdbcfrrfabfadrdbdbfe
#8	cde-rfecbadadfafabceecbeeba-