TR-IT-0041

ファジィパーティションモデルに対する バックプロパゲーションアルゴリズムの高速化

Training speed improvement of the back-propagation algorithm for Fuzzy Partition Models

加藤 喜永 Y. Kato　　†伴 敏雄 T. Ban　　松永 昭一 S. Matsunaga　　片桐 滋 S. Katagiri

1994.2

大規模なニューラルネットワークをより短時間で設計するためにFPMの学習アルゴリズムを高速化した。実験結果から、結合パラメータの修正回数を少なくし、さらにDCP法を組合せることが有効であることがわかった。多数話者の音素識別に用いるニューラルネットワークを設計し、従来法に比べて約2.2倍高速になった。また、本手法によりアルゴリズムが高速になるだけでなく、識別率が高くなることも確認した。付録として本高速アルゴリズムを用いたソフトウェアの使用法を添付した。

†エスイーティ（株）
†SET

目次

1 はじめに

計算機の演算速度が向上したことにより、大規模なニューラルネットワークを用いたシミュレーションが可能になりつつある。またATRでは、ニューラルネットワークの学習アルゴリズムの一つである誤差逆伝播アルゴリズムの高速化を検討し、短時間でネットワークを設計できるプログラムを開発した[1][2]。一方、確率的なモデリングを意識したFuzzy Partition Model（以下、FPM）と呼ばれるニューラルネットワークは、Kullbackダイバージェンスを誤差評価関数として誤差逆伝播アルゴリズムを適用することにより、従来のパーセプトロン型ニューラルネットワークに比べて高速学習を可能としている。多数話者に対する音声認識実験において、通常の誤差逆伝播学習によるFPMの学習は、上記の高速学習アルゴリズムを適用したTDNNよりも約半分の時間で完了することを確かめている[3]。

しかしながら、音声認識の研究において不特定話者認識を対象にした場合、特定話者に比べて学習標本数は増大し、ネットワークの規模も大きくなる傾向にある。従ってシミュレーションを行う上で、学習時間をさらに短縮することが必要とされる。ところが、FPM素子の構造は従来のニューラルネットワークのものと異なるため、ATRで開発された高速化プログラムをそのままでは使用できない。そこで本稿では、FPMの学習アルゴリズムに対しDCP(Dynamic Control training Parameters)法[2]を中心に移植したので報告する。

2 Fuzzy Partition Model (FPM)

2.1 素子の構造

FPMは多入出力素子で構成されるニューラルネットワークであり、素子の構造は従来のパーセプトロンタイプの素子と異なる。図1に素子の構造を示す。以下N個の出力をもつFPM素子をN次元の素子と呼ぶことにする。第m層第s素子の中のk番目の入力、出力をそれぞれ$u_{k^m}^{(s)}$, $a_{k^m}^{(s)}$とし、$a_{j^{m-1}}^{(g)}$と$u_{k^m}^{(s)}$とを結合する自由パラメータを$w_{k^m j^{m-1}}^{(sg)}$とおけば素子の入出力関係は次式で表すことができる。但し、第m層の素子数をM_mと表す。

$$a_{k^m}^{(s)} = \frac{\exp(u_{k^m}^{(s)})}{\sum_{j=1}^{N_{m-1}} \exp(1 + u_{j^m}^{(s)})} (k = 1, \ldots, N-1) \tag{1}$$

$$a_{N^m}^{(s)} = \frac{1}{\sum_{j=1}^{N_{m-1}} \exp(1 + u_{j^m}^{(s)})} \tag{2}$$

$$u_{k^m}^{(s)} = \sum_{g=1}^{M_{m-1}} \sum_{j=1}^{N_{m-1}} w_{k^m j^{m-1}}^{(sg)} a_{j^{m-1}}^{(g)} \qquad (k = 1, \ldots, N-1) \tag{3}$$

素子の出力は非線形性をもつ指数関数で正規化されている。次式に示すように出力値は常に正でかつ素子内の出力値の総和は定数である。

$$\sum_{j=1}^{N_m} a_{j^m}^{(s)} = 1 \tag{4}$$

$$0 \leq a_{j^m}^{(s)} < 1 \qquad (\forall j) \tag{5}$$

2.2 FPMの誤差逆伝播アルゴリズム

FPMの学習には最急降下法あるいは確率的降下法を用いる。教師信号とネットワークの出力との誤差が最小になるように素子間の重みパラメータを逐次的に変化させて望ましいネットワークを形成していく。出力層の誤差評価関数にはKullbackダイバージェンスを用いる。最急降下法による重みパラメータの修正量$\Delta w_{k^m l^{m-1}(GD)}^{(su)}$を式(6), 確率的降下法による重みパラメータの修正量$\Delta w_{k^m l^{m-1}(PD)}^{(su)}$を式(7)に示す。

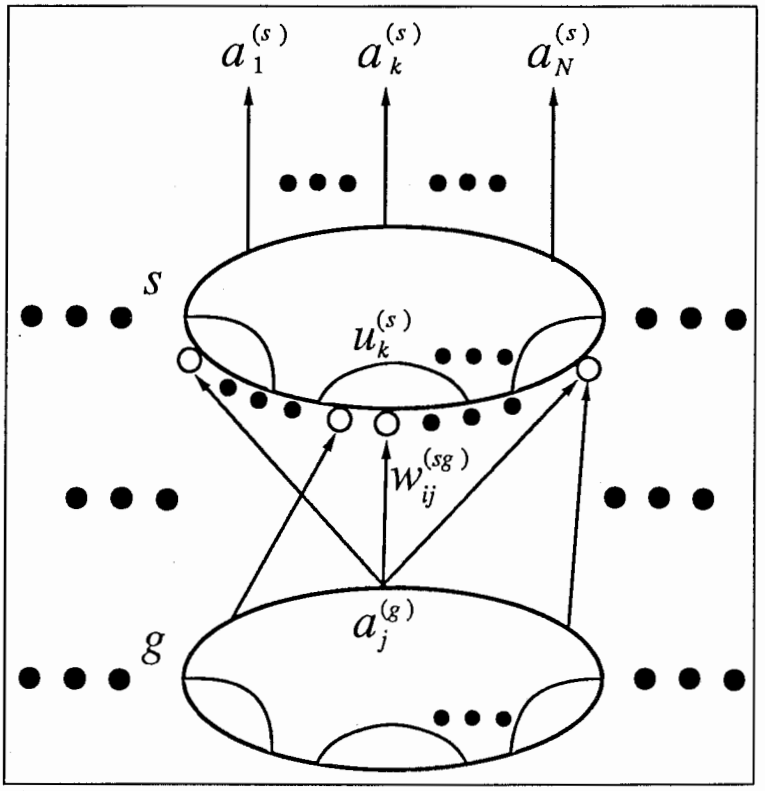


図 1: N 次元構造をもつ FPM 素子

$$\Delta w^{(sv)}_{k^m l^{m-1}(GD)} = -\sum_{p=1}^{P} \eta \frac{\partial D_p}{\partial w^{(sv)}_{k^m l^{m-1}}} \tag{6}$$

$$\Delta w^{(sv)}_{k^m l^{m-1}(PD)} = -\eta \frac{\partial D_p}{\partial w^{(sv)}_{k^m l^{m-1}}} \tag{7}$$

ここで D_p は、学習標本 p に対する誤差を示し、P は学習標本の総数を表す。また η は学習率である。式 (6) は、全ての学習標本の誤差を計算した後にパラメータを修正するのに対し、式 (7) は、一つの学習標本を提示するたびにパラメータを修正する。両式を展開した結果はほぼ同じ形となるので、以下では確率的降下法を例にして式 (7) を展開する。

$$\Delta w^{(sv)}_{k^m l^{m-1}} = -\eta \frac{\partial D_p}{\partial w^{(sv)}_{k^m l^{m-1}}} \tag{8}$$

$$D_p = \sum_{g=1}^{M_m} \sum_{j=1}^{N_m} t_j^{(g)} \log \frac{t_j^{(g)}}{a_{j^m}^{(g)}} \tag{9}$$

式 (9) を式 (8) に代入し、過去の修正の影響量 α も考慮すれば次式になる。

$$\Delta w^{(sv)}_{k^m l^{m-1}} = \eta \delta^{(s)}_{k^m} a^{(v)}_{l^{m-1}} + \alpha \Delta w^{(sv)}_{k^m l^{m-1}} \tag{10}$$

$\delta^{(s)}_{k^m}$ は出力層とその他の層によって異なり、m が出力層の場合には式 (11)、それ以外の場合には式 (12) となる。

$$\delta^{(s)}_{k^m} = t^{(s)}_{k^m} - a^{(s)}_{k^m} \tag{11}$$

$$\delta^{(s)}_{k^m} = a^{(s)}_{k^m} (\sigma^{(s)}_{k^m} - \sum_{j=1}^{N_m} a^{(s)}_{j^m} \sigma^{(s)}_{j^m}) \tag{12}$$

但し、

$$\sigma_{km}^{(s)} = \sum_{g=1}^{M_{m+1}} \sum_{i=1}^{N-1_{m+1}} w_{im+1\,km}^{(gs)} \delta_{im+1}^{(g)} \tag{13}$$

式(11), (12)が示すように、$\delta_{km}^{(s)}$ を再帰的に計算することによって、ネットワークの全結合パラメータを修正することができる。これは、出力層で計算された誤差を下層に向かって逆伝播するアルゴリズム（誤差逆伝播法）であることを示している。

3　高速学習アルゴリズム DCP の説明

3.1　DCP 法の概要

DCP(Dynamic Control training Parameters) 法はより誤差が小さくなるような式(10)で与えられる η, α の組合せをある学習回数ごとに設定し直すアルゴリズムである。 DCP 法を適用しない場合、η, α は、通常定数でありタスクに応じて経験的に定めている。しかし、学習の始まりの段階では、修正量を大きくして粗くネットワークを形成し、学習の最終段階では、修正量を小さくして精密な形成を行ったほうが、目的とする結合パラメータを得るまでの時間を短縮できると考える。このように両者の組合せをネットワークの学習段階に応じて動的に変化させることによって、高速な学習を実現している。ここで、B 個の候補からなる学習率と $(\eta_1, \ldots, \eta_b, \ldots, \eta_B)$ と C 個の候補からなる影響量 $(\alpha_1, \ldots, \alpha_c, \ldots, \alpha_C)$ の組合せにより、結合パラメータを修正したと仮定する。結合パラメータを修正した後の入力パタンに対するそれぞれの組合せに対する誤差を計算し、その中で最小値をもつものを以下の式により決定する。

$$\min_{1 \le b \le B, 1 \le c \le C} \sum_{p=1}^{P} D_{pbc} \tag{14}$$

D_{pbc} は p 番目のパタンに対する η_b, α_c の組合せに対する誤差であり、式(9)を用いて計算できる。P は誤差計算に用いた総パタン数 P である。 DCP 法では、式(14)の中で最小値をもつ η_b, α_c の組合せを次の学習から用いる。

3.2　アルゴリズムの説明

図2に， DCP 移植後の学習アルゴリズムのフローチャートを示す。また、通常学習時のフローチャートを図3に示す。通常学習時のフローチャートは DCP 法移植前のアルゴリズムでもあり、今回の作業内容は、図3のアルゴリズムを図 2へのアルゴリズムへと拡張したことにあたる。以下主要な処理の説明をする。

FPM-DCP ループ DCP 処理のメインループであり、主に以下の処理を繰返す。

α, η の候補算出 ある回数の学習を終了するごとに、次回からの学習に用いる α, η の候補を算出する。

結合パラメータ仮修正 候補となった α, η のそれぞれの組合せで結合パラメータの修正を行う。

DCP 学習ループ 結合パラメータの仮修正を所定の回数だけ行う。

DCP 評価ループ DCP 学習ループが終了した後、各 α, η の組合せに対する出力誤差を計算する。

α, η の最適値を決定 上記出力誤差が一番小さくなる α, η の組合せを式(14)による基準で選択し、次回からの学習に用いる。

4　音素認識実験によるアルゴリズムの評価

本章では音素認識実験により DCP 法を評価した結果について示す。

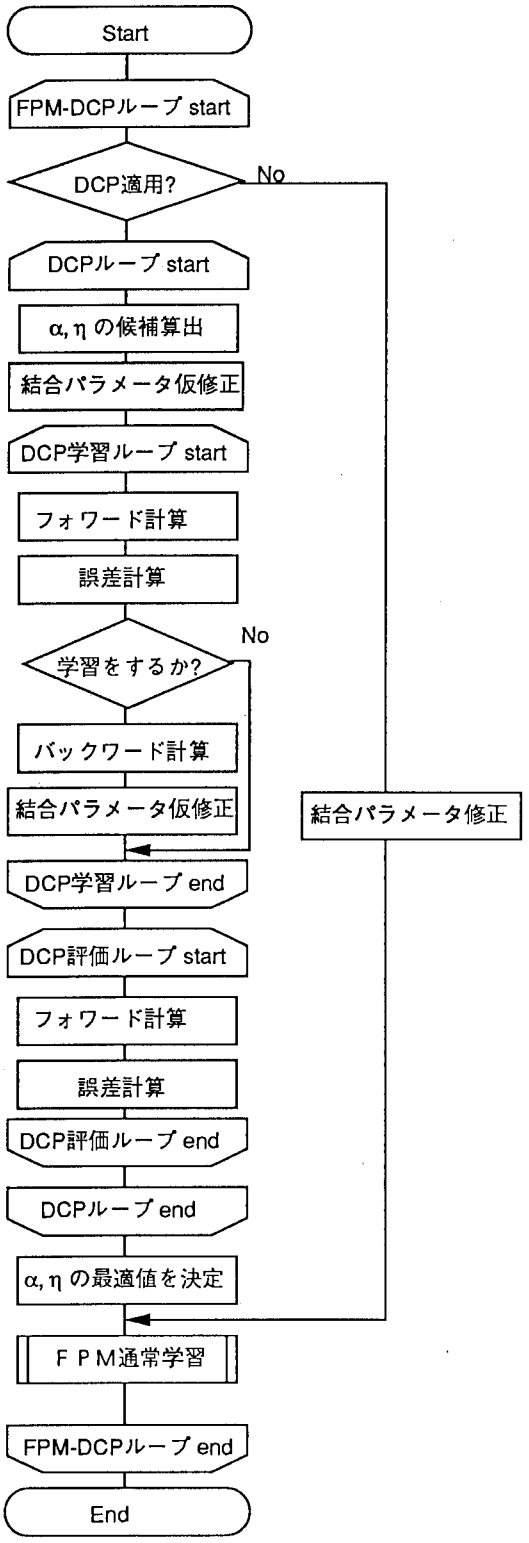


図 2: DCP 法による高速アルゴリズムのフローチャート

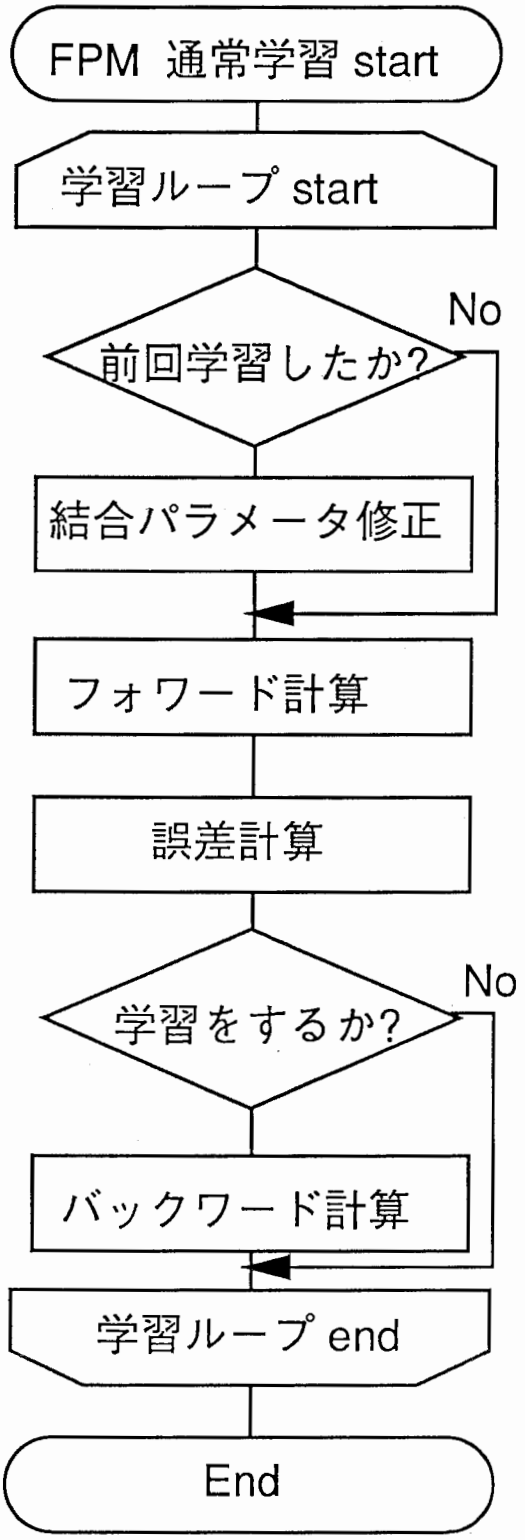


図 3: 通常学習時のフローチャート

表 1: 音響分析条件

Sampling Frequency	12 kHz
Windowing	256-point Hamming window
Frame Rate	10ms
Input Pattern	16 channel FFT mel-scaled spectrum 7 frames
Power Normalization	Normalized to lie between 0.0 and 1.0 with the average at 0.5

4.1 実験条件

本実験に用いた計算機は DEC7000/610[1]である。音響分析条件を表 1に示す。FPM を日本語の 25 音素を認識するように設計する。図 4に FPM の構造を示す。FPM は入力層を含む 4 層構造とし、音声を周波数分析して得られた 16 チャネルのパワー 7 フレーム分を入力する [3]。したがって、$16 \times 7 = 112$ 個の値を 1 標本として入力する。また、出力層の素子の個数は、25 次元素子 1 つであり、それぞれの出力からは音素らしさを確率的に得られるように設計される。以下では、特定話者による音素識別実験と多数話者による音素識別実験とのの両方について評価した。隠れ層の素子の個数は両実験によって異なるが、どちらの実験においても隠れ層 1、隠れ層 2 の個数はともに等しい。また、隠れ層の素子は常に 2 次元で一定である。

学習率の候補は η_1, η_2, η_3 の 3 種類とし、ぞれぞれ次の式により決定する。

$$\begin{aligned} \eta_1 &= x\eta_0 \\ \eta_2 &= \eta_0 \\ \eta_3 &= \eta_0/x, \ (\eta_1 > \eta_2 > \eta_3) \end{aligned} \tag{15}$$

ここで η_0 は、それまで用いていた学習率の値であり、x はそれぞれ学習率の増加、減少を決定するための正の実数である。影響量の候補数は最大 2 とし、その値は学習率の場合と違って定数である。式 (15) からわかるように、DCP で候補となる学習率の値は現在の値を含む近傍である。もし、現在の値と望ましい値とが離れていると DCP 法を一度適用しただけでは、現在の値からでは望ましい値を決定できない可能性がある。そこで本実験においては初回適用時に限り η, α の決定処理を 5 回連続する。また、n 回目の DCP 法適用を前回適用時から数えて 2^{n-1} 回目の学習後に行う。ただし、本実験においては 2^{n-1} が 32 を越えた場合でも、32 回学習した後に必ず DCP 法を適用する。また、ここでいう一回の学習とは全学習標本を一通り学習することである。これらの条件下で DCP 法の適用を含む誤差逆伝播法による学習を 300 回まで行う。

4.2 特定話者音素識別実験の結果

男性話者 1 名の単語発声による音声データから作成した合計 250 標本の学習セットにより、DCP の性能を評価した。図 4に示す隠れ層の素子数をともに 32 に設定する。候補となる η, α の組合せに対する誤差計算には学習セットの中からランダムに選んだ 25 標本を用いる。結合パラメータは式 (7) に示すように標本を一つ FPM に提示するたびに修正される。

4.2.1 学習率と影響量との組合せ数について

図 5に DCP を適用する対象として学習率だけにした場合と影響量も考慮した場合との比較を示す。上図には CPU 時間に対する誤差の変化を示し、同下図には CPU 時間に対する音素識別率の変化を示す。図中の凡例に η とあるのは式 (15) の $\eta_0 = 0.02, x == 1.2$ を初期値とし、$\alpha = 0.9$(固定) とした場合であり、$\eta + \alpha$ とあるのは前記初期設定に加え、α も 0.9 と 0.95 とのどちらかを選択するようにした場合である。DCP 法は音素識別率が 40% から 95% の間で適用され、η, α の決定処理の連続回数を DCP 法適用初回時には 5 回、それ以外は 2 回に設定した。η は、$\eta + \alpha$ よりも速く学習していることがわかる。$\eta + \alpha$ の曲線は η の曲線を 150 秒付近まで追随している。従って両者とも同じ学習率、影響量を選択して学習を進めたことになる。しかし、学習率、影響量の選択をするのに η は 3 通り、$\eta + \alpha$ は 6 通りの学習率、影響量の組合せに対して DCP 法を適用しなければならないの

[1]速さは、203.3[MIPS], 94.8[SPECint'92], 182[SPECfp'92] である。

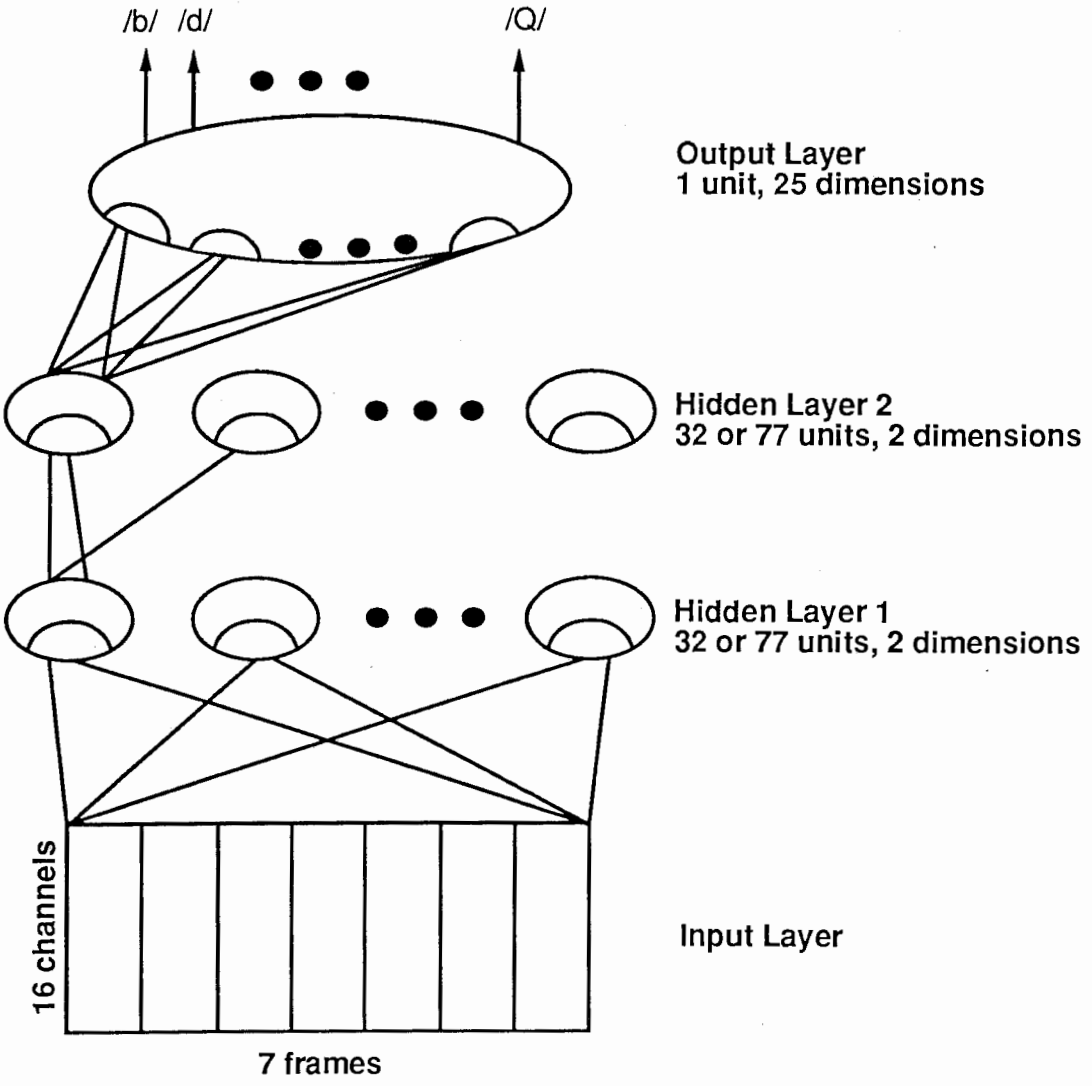
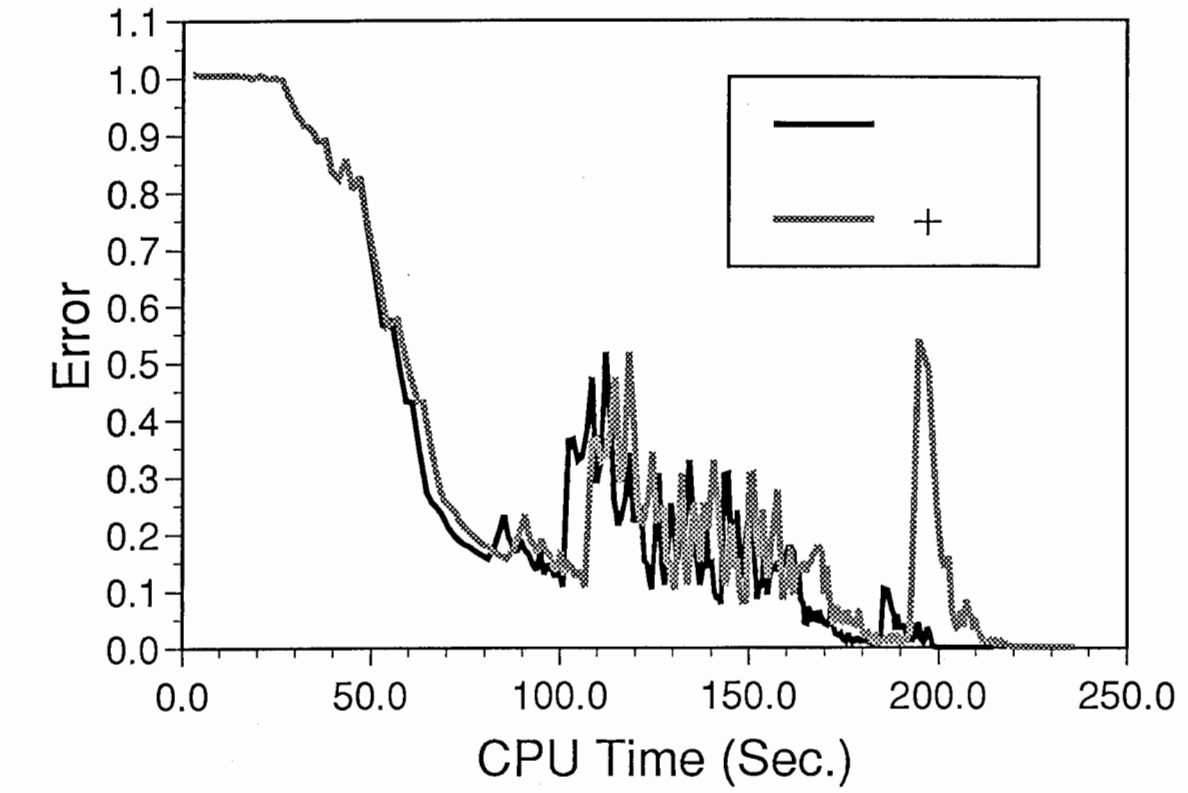


図 4: 認識実験に用いた 4 層 FPM の構造



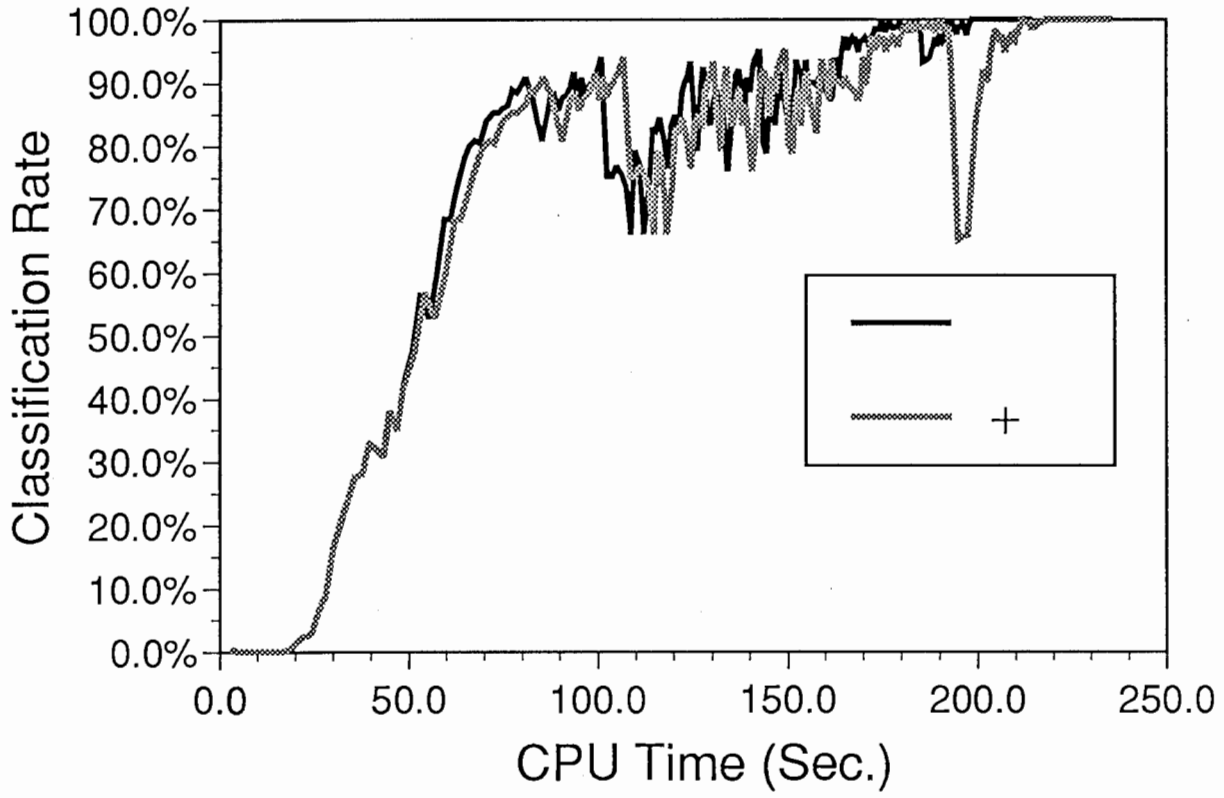


図 5: DCP 適用の対象となる候補数の違いによる学習状況の比較

表 2: 学習の速さの比較

CPU 時間 (sec.)		
従来法 (A)	DCP 法 (B)	比率 (A/B)
195	155	1.26

で $\eta+\alpha$ の方が時間がかかっている。150 秒を越えた付近で両者の曲線の動きが異なることから、$\eta+\alpha$ が影響量を 0.95 に切り替えたことがわかる。しかし、その後 $\eta+\alpha$ における 200 秒付近で急激に誤差と識別率が劣化している。これは過去の影響量も DCP 法適用の対象にすると不安定になることを示している。そこで以下の実験では学習率に対してだけ DCP 法を適用する。

4.2.2 学習率の変更幅について

図 6に式 (15) における x を 2.0 及び 1.2 にした場合の学習状況を示す。DCP 法により学習率が変化する範囲は $x = 1.2$ の方が小刻みになる。$\eta_0 = 0.01$ を初期値とし、$\alpha = 0.9$ に固定した。DCP 法は音素識別率が 60% から 95% の間で適用され、η の決定処理の連続回数を DCP 法適用初回時には 5 回、それ以外は 2 回に設定した。同図より 100 秒付近では $x = 2.0$ の方が学習は進んでいるが 150 秒付近から $x = 1.2$ に逆転する。一般に学習が進むにつれて結合パラメータの修正を小刻みにした方がよいとされる。$x = 2.0$ において学習の後半に収束が遅れたのは学習率の変更幅が多きすぎるために起きたと考えられる。

4.2.3 従来法との比較

図 7に従来法と DCP 法との比較を示す。従来法では学習率を 0.1 で固定し、標本を一つ提示するごとに結合パラメータを修正する。この値は予備実験により従来法では最も高速に学習できることがわかっている。DCP 法における条件として、式 (15) における $x = 1.2$ とし、音素識別率で 40% から 95% を得ている間に適用する。η の決定処理の連続回数を DCP 法適用初回時には 5 回、それ以外は 1 回に設定した。同図より DCP 法を用いると高速に学習できることがわかる。また、表 2に学習の速さの比較を示す。表中の値は音素識別率で 100% を達成した時のものである。DCP 法を適用することによって小規模なタスクにおいては 1.26 倍高速になった。

4.3 多数話者音素識別実験の結果

男性話者 8 名の単語発声による音声データから作成した 50,000 標本による学習を行った。図 4に示す隠れ層の素子数をともに 77 に設定する。

4.3.1 高速法の比較

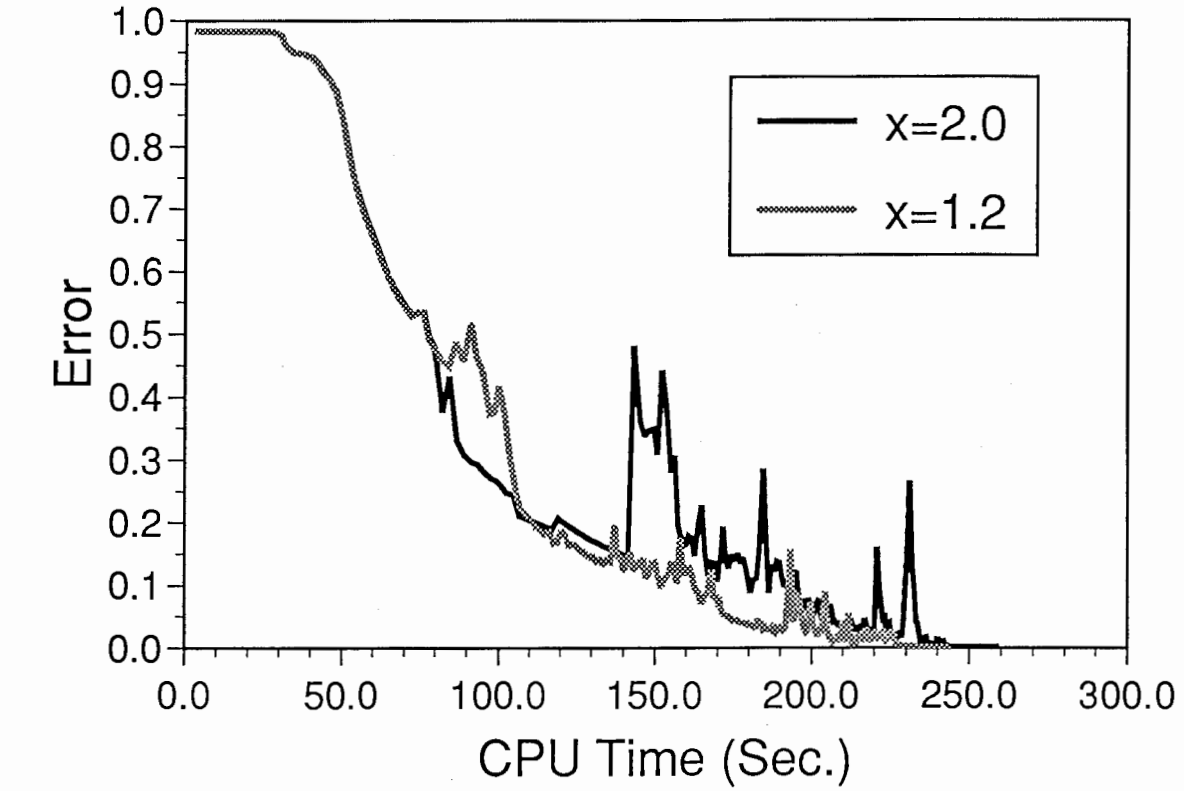
以下の 3 種類の高速法について比較した。

Case 1 DCP 法を用いる。$\eta_0 = 0.02$, $x = 1.2$ を初期値とし、$\alpha = 0.9$ で固定する。DCP 法は音素識別率が 40% から 95% の間で適用され、η の決定処理の連続回数を DCP 法適用初回時には 5 回、それ以外は 1 回に設定した。候補となる η に対する誤差計算には学習セットの中から無作為に選んだ 500 標本を用いる。結合パラメータは標本を一つ FPM に提示するたびに修正される。

Case 2 P 個の標本を FPM に提示した後に結合パラメータの修正を行なう。すなわち、式 (6) により結合パラメータを修正する。この方法によれば、修正回数が減るので、計算時間を短縮することができる。P が 1 であれば従来法となり、P が全標本数であれば最急降下法となるが、本実験においてはこれらの中間的な方式を採用し、$P = 25$ に設定した。また、$\eta_0 = 0.01, \alpha = 0.9$ とした。

Case 3 Case 1 と Case 2 とを組合せた方法。

図 8に学習状況を示す。case 2 は case 1 よりも速く学習が進む。このことから結合パラメータ数が大きくなればなるほどパラメータの修正回数を減らすことが重要であることがわかる。しかし、case 2 においても時間がたつにつれ誤差の減少率あるいは識別率の上昇率が悪くなり、case 3 を適用することでさらに高速に学習できることわかる。



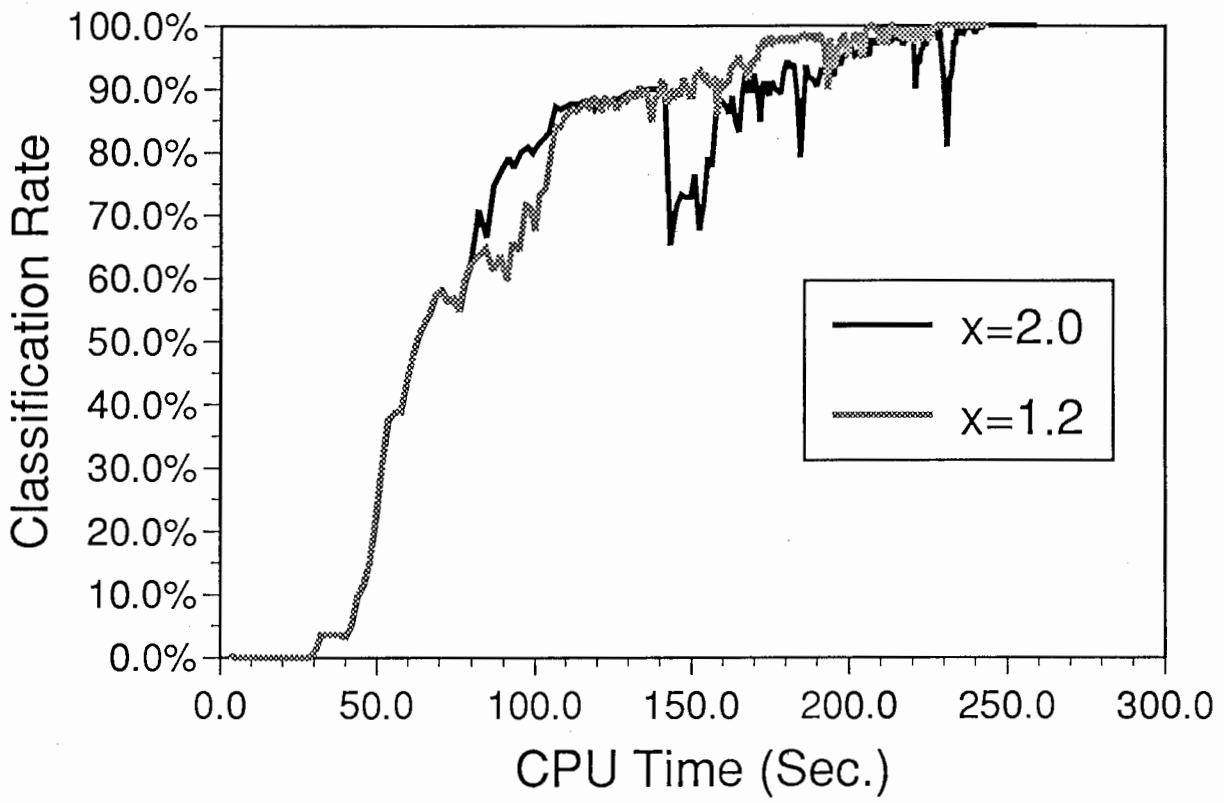
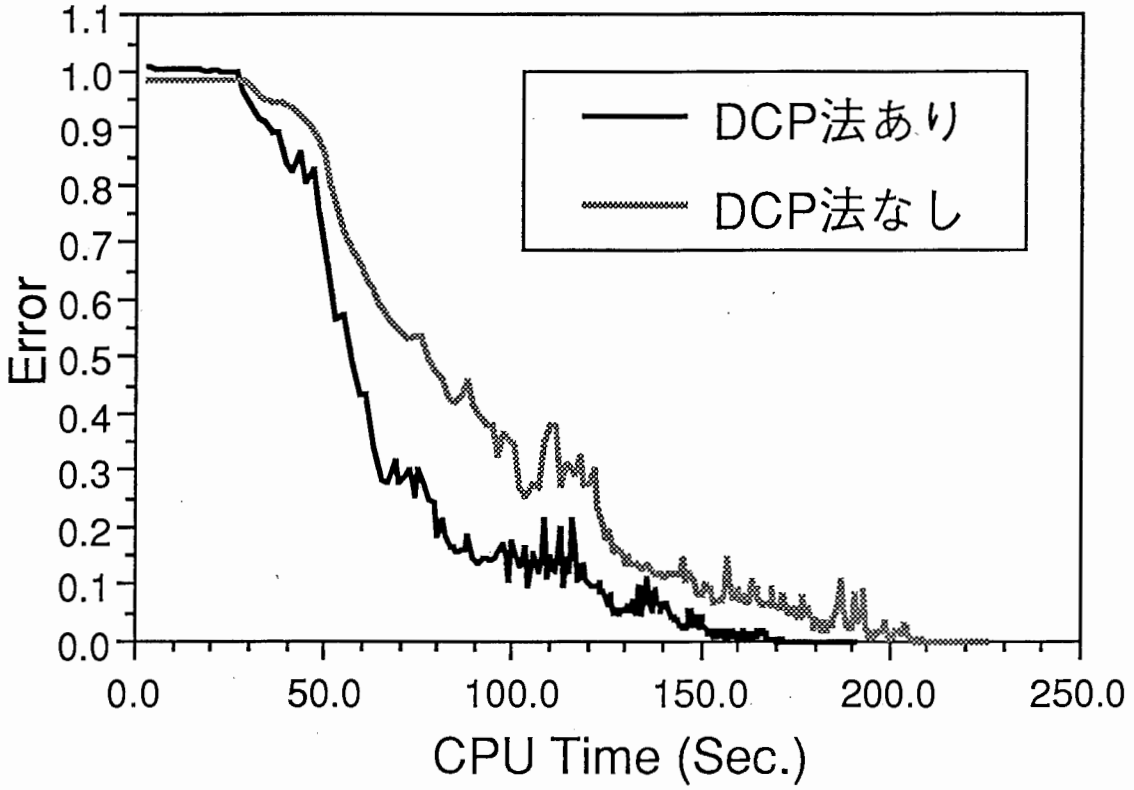


図 6: 学習率の変更幅の違いによる学習状況の比較



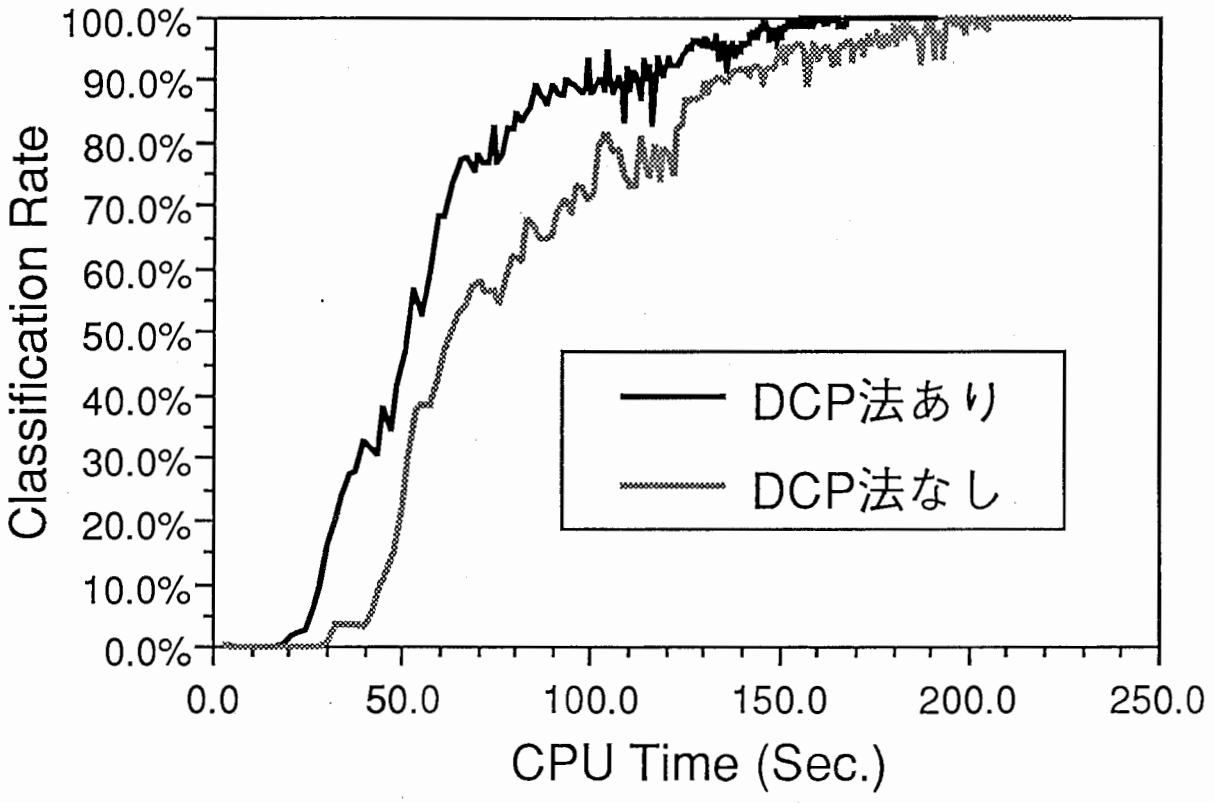
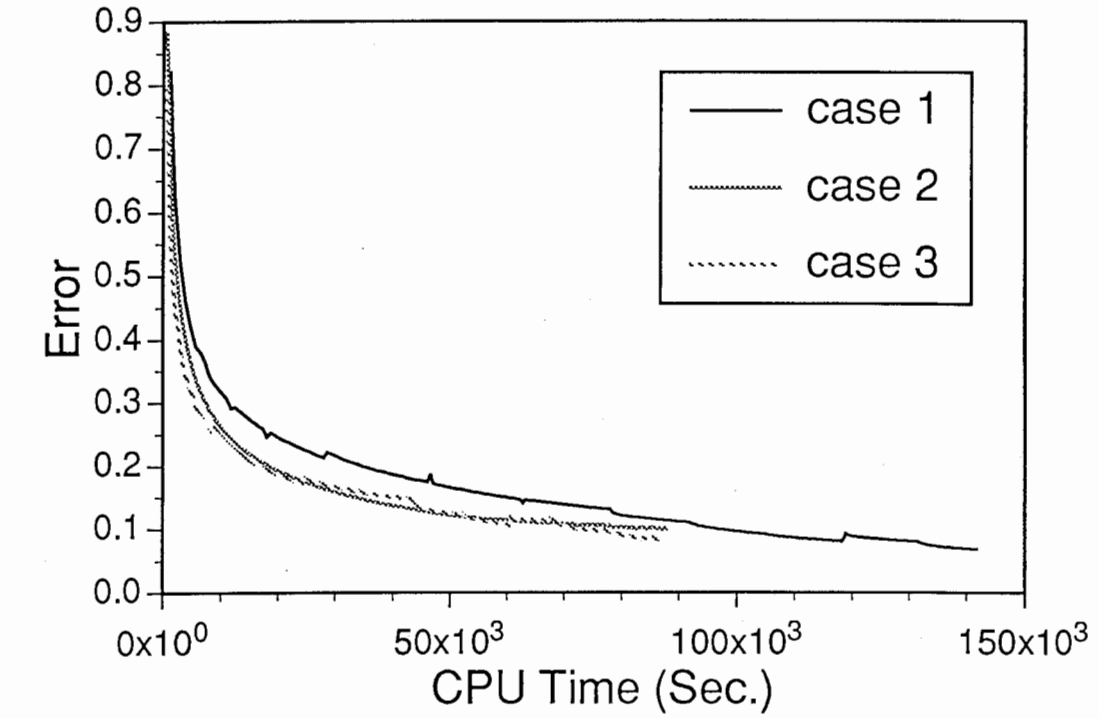


図 7: 従来法と DCP 法との学習状況の比較


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Error
case 1
case 2
case 3
0x10^0
50x10^3
100x10^3
150x10^3
CPU Time (Sec.)

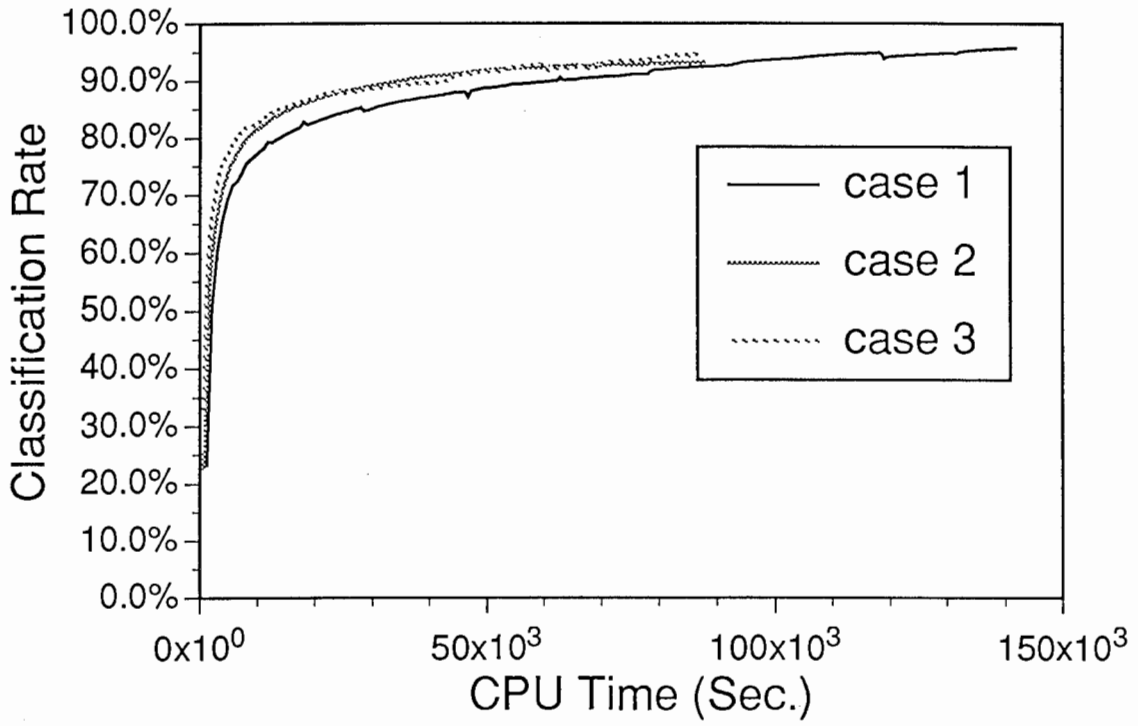

100.0%
90.0%
80.0%
70.0%
60.0%
50.0%
40.0%
30.0%
20.0%
10.0%
0.0%
Classification Rate
case 1
case 2
case 3
0x10^0
50x10^3
100x10^3
150x10^3
CPU Time (Sec.)

図 8: 高速学習法の比較

表 3: 学習の速さの比較

識別率 (%)	CPU 時間 (10^3 sec.) 従来法 (A)	高速法 (B)	比率 (A/B)
92.0	89.6	51.7	1.73
93.0	124.4	66.1	2.18
94.0	達成せず	93.3	–

4.3.2 従来法との比較

図 9に従来法と case 3 による高速法との比較を示す。従来法では学習率を 0.1 で固定し、標本を一つ提示するごとに結合パラメータを修正する。高速法における条件として、$\eta = 0.0016, x = 1.1, P = 25$ とした。また、表 3に学習の速さの比較を示す。表中の CPU 時間は各音素識別率を達成した時のものである。従来法においては 300 回の学習では 94.0% の識別率を達成することができなかった。従って、本手法は識別率を上げようとすればするほど高速法による学習の方が短時間で学習できるだけでなく、識別率も向上することができる。

5 むすび

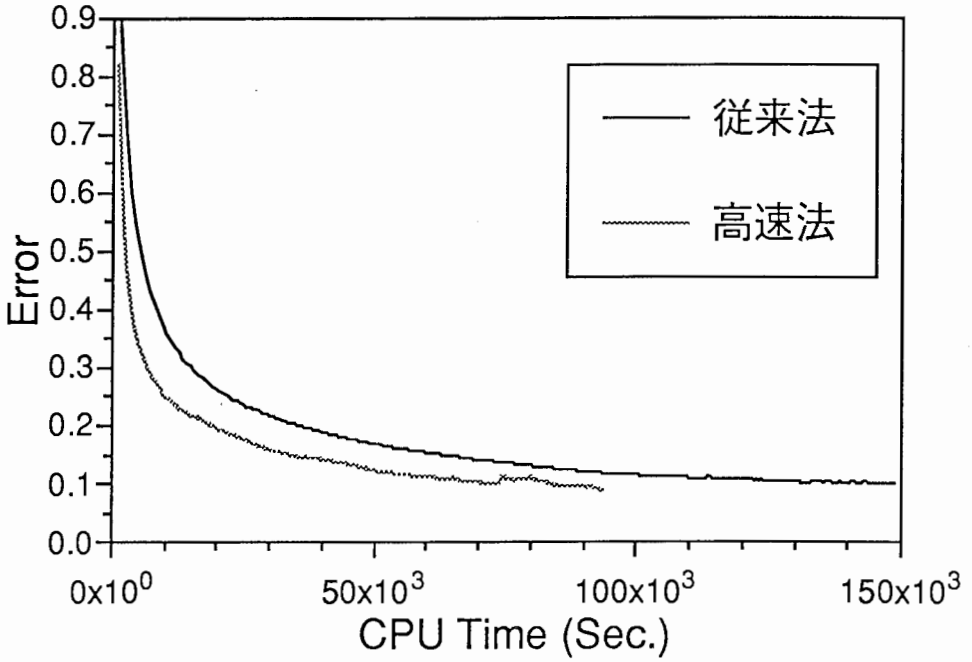
大規模なニューラルネットワークをより短時間で設計するために FPM の学習アルゴリズムを高速化した。実験結果から、結合パラメータの修正回数を少なくし、さらに DCP 法を組合せることが有効であることがわかった。また、高い識別率を得ようとするほど本報告による方法が有効であることがわかった。

参考文献

[1] P. Haffner, "*DyNet, a fast program for learning in nerural networks*," ATR Technical Report(非公開), **TR-I-0059** (1988).

[2] 中村, 鹿野, "ニューラルネットにおけるバックプロパゲーション学習の効率化方法," ATR Technical Report(非公開), **TR-I-0119** (1989).

[3] 福沢, 加藤, 杉山, FPM-LR を用いた不特定話者連続音声認識, 信学論 (D-II), **J76-D-II**, 11, pp.2253–2263 (1993-11).



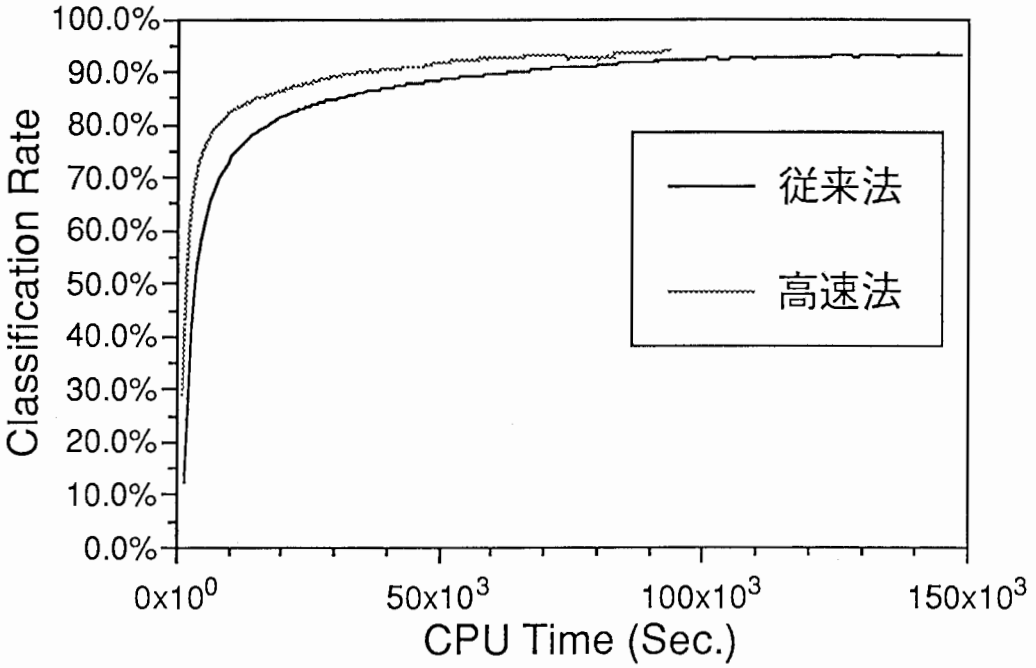


図 9: 従来法と高速学習法法との学習状況の比較

付録

付録 A　プログラムの説明

Name

fpml3_dcp, fpml4_dcp – DCP 法による FPM の高速学習プログラム[2]

Syntax

fpml3_dcp -f オプションファイル名 パラメータ…
fpml4_dcp -f オプションファイル名 パラメータ…

Example

fpml4_dcp -f fpml.opt -sf phon25.train -ai 0.9 0.95 -ei 0.005 0.01 0.02 ...

Description

fpml3_dcp、fpml4_dcp はそれぞれ 3 層、4 層構造の FPM を設計するプログラムタである。パラメータ指定は、ファイルでも、コマンドラインからでも良い。但しコマンドラインの指定を優先する。何も指定されていない場合は、プログラム中に定義してあるデフォルトの設定で動作する。以下では、その他に必要な入出力ファイルについて詳しく述べる。

A.1　学習標本データファイル

ネットワークへ入力する学習標本を記述したファイルである。ファイル名は、オプションファイルにて指定することができる。ファイルの先頭にはヘッダ情報として、見出し、入力ユニット数、出力ユニット数、標本数を記述し、次に入力ユニットへのデータと出力教師データの、標本数のセットを記述する。 ASCII テキストファイルである。標本ファイルの例を以下に示す。

標本ファイルの例
SAMPLE FILE:
113 inputs
25 outputs
250 patterns
1.000000
0.606130 0.951239 0.851405 0.392460 0.251233 0.189186 0.301906 0.338575 0.129235 0.006852 0.052548 0.229590 -0.129373 0.213300 -0.330492 -0.369755
0.579302 0.862206 0.482299 0.225234 0.035761 -0.007716 -0.119122 -0.214098 -0.257640 -0.395438 -0.287975 -0.295414 -0.370422 -0.345334 -0.409941 -0.409721
0.550500 0.841135 0.430977 0.184514 -0.037943 -0.138044 -0.050524 -0.341733 -0.358021 -0.397296 -0.387815 -0.375191 -0.413152 -0.426232 -0.434700 -0.422168
0.525605 0.809953 0.324679 0.132681 -0.104281 -0.211016 -0.097881 -0.269141 -0.500710 -0.386851 -0.425067 -0.402918 -0.454291 -0.414777 -0.459939 -0.459475
0.501378 0.781557 0.258388 0.098335 -0.109570 -0.222436 -0.125037 -0.251948 -0.504172 -0.375752 -0.426828 -0.408718 -0.437683 -0.424118 -0.451881 -0.437851
0.489518 0.769785 0.298737 0.154780 -0.032937 -0.159658 -0.106985 -0.174063 -0.299244 -0.188445 -0.260962 -0.271864 -0.277790 -0.233374 -0.290536 -0.325062
0.498366 0.880498 0.606284 0.451000 0.244899 0.081615 -0.019702 0.154612 0.179804 0.387623 0.399658 0.364183 0.196642 0.295675 0.135974 0.000408
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
.
.
.

[2]本プログラムは /usr/common/itl/src/kato/FPML_DCP.src に格納されている。(1994 年 2 月 22 日現在)

A.2 出力ファイル

weight?? (?? : 学習回数)

?? 回数の学習を終えた時点での、ネットワークの結合パラメータファイルである。書き出しディレクトリは、オプションファイルにて指定することができる。書式は、4バイト IEEE 浮動小数点バイナリ表記である。

例 od -f weight??

```
0000000  -2.3264533e-01  7.8169847e-01  1.2196698e+00  1.0838127e+00
0000020   2.8577563e-01 -3.2847930e-02 -1.9901727e-01 -4.1875237e-01
0000040  -1.5592242e+00 -8.8956821e-01 -6.2117529e-01  3.8615800e-02
0000060   1.3764627e-02 -1.0447618e+00 -4.8943946e-01 -1.4827628e+00
0000100   2.5916868e-01  1.2580215e+00  1.6856966e+00  1.1763791e+00
0000120   6.5560079e-01  3.7304094e-01 -2.3440969e-01 -3.9754134e-01
0000140  -1.4035692e+00 -6.4004225e-01  3.5846982e-02  6.5841889e-01
0000160   3.8271222e-01 -7.3546755e-01 -2.1501905e-01 -1.2185456e+00
```

./display/reco1

上記ファイル名はデフォルトであり、オプションファイルにて変更可能である。各カテゴリ毎の学習正解率を記述する。最終カラムの数値は全カテゴリの平均正解率であり、() 内は学習回数である。ASCII テキストファイルである。

reco1 の例

0:0.000000 1:0.000000 2:0.000000 3:0.000000 4:0.000000 5:0.000000 6:0.000000 7:0.000000 8:0.000000 9:0.000000 10:0.000000 11:0.000000 12:0.000000 13:0.000000 14:0.000000 15:0.000000 16:0.000000 17:0.000000 18:0.000000 19:0.000000 20:0.000000 21:0.000000 22:0.000000 23:0.000000 24:0.000000 0.000000(299)

A.3 標準出力

本プログラムでは、学習途中経過を示すログデータを出力している。その書式は、 dcp の on/off、学習繰り返し回数、実学習標本数、出力誤差、認識率、影響量 (α)、学習率 (η)、 CPU 時間 (user time) である。

例

```
Learning condition in display/reco1
read sample in sample/phon25.train
113 inputs      25 outputs      250 samples     10 samples/cat.
DCP? Epoch Samples M.S.E Rate Alpha Eta CPU Time (sec.)
0 0 250 1.015256 0.004 0.900000 0.010000 16.883333
0 1 250 1.012210 0 0.900000 0.010000 32.666667
0 2 250 1.012148 0 0.900000 0.010000 48.400000
0 3 250 1.012125 0 0.900000 0.010000 64.200000
0 4 250 1.012107 0 0.900000 0.010000 79.916667
0 5 250 1.012083 0 0.900000 0.010000 95.650000
```

A.4 その他

本プログラムを終了した後に、再び続けて学習をさせたい場合には、ディレクトリ./result0 の下に，続けたい結合パラメータファイルをオプションファイルで設定し (デフォルトファイル名 "weight")、再起動する。

例
cp ./weight300 ./result0/weight
fpml4_dcp .....

付録 B　オプションファイル

オプション指定の為のファイル。書式は、第一欄に二文字の記号、第二欄を数字で設定する。行の'#' 記号以降は、コメントと見なす。これらの値を指定しない場合には [] 内の値をデフォルトとする。

```
#
#          オプションファイル
#

#   α   モーメンタム
ai 0.9            # α選択値                                    [0.9 0.95]

#   η   ステップサイズ
em 1.0            # ηの上限                                    [1.0 (0:制限無し)]
en 0.0            ηの下限                                     [0.0 (0:制限無し)]
ei 0.015 0.02051316702 0.03 0.03948683298 0.06 # ηの選択値     [0.005 0.01 0.02]
#   ηの変更倍率は, 上記のように可変値をとれる。

#  学習について
ln 300            # 学習回数                                    [100]
on 0.0001         # 学習終了出力誤差                            [0.0]
om 0.005          # 学習スキップ出力誤差                        [0.005]

cw 0              # 結合パラメータの修正間隔                    [0]
#         出力ユニット数の   cw 倍で修正する

ef 0              # 教師信号出力誤差計算関数の選択              [0]
#         0: Mean Squared Error  1: Kullback Divergence

wi 0              # 結合パラメータファイルの書き出し間隔        [0]
rs 5              # 初期結合パラメータを設定するための乱数の種  [0]
rr 0.1            # 乱数の幅                                    [1.0]

#   DCP について
rm 0.9            # DCP 作動上限認識率                          [1.0]
rn 0.1            # DCP 作動下限認識率                          [0.0]
ci 15             # DCP の初期連続実行回数                      [10]
ce 2              # DCP 実行間隔 (ce, ce^2 ce^3 ...)            [1]
cm 16             # DCP 実行間隔上限                            [32]
cn 1              # DCP 連続実行回数                            [2]
ds 25             # DCP に用いる標本数                          [0]

#    その他
rc 0              # fpml_dcp 実行モード                         [0]
#         1: 認識テストモード  0: 学習モード

sf ./sample_1     # 学習標本ファイル                            [./sample/ln_sample]
rp ./d_reco       # 学習正解率記述ファイル                      [./display/reco1]
```

```
wp ./weight_result # 結合パラメータファイル書き出しパス        [./]
wr def_weght.200 # 初期結合パラメータファイル                  [./result0/weight]
```

付録 C　主要関数及びヘッダファイルの説明

1. fpml_dcp.h

   InUnit 1 入力素子数

   InState 112 入力素子一つあたりの次元数

   MidUnit1 77 隠れ層第一層の素子数

   MidState1 2 隠れ層第一層の素子一つあたりの次元数

   MidUnit2 77 隠れ層第二層の素子数

   MidState2 2 隠れ層第二層の素子一つあたりの次元数

   OutUnit 1 出力層素子数

   OutState 25 出力層素子一つあたりの次元数

2. init1()

   学習を行なう前に FPM の結合パラメータを初期化する。オプション [rr rs] で指定される範囲の乱数で値を与える。

3. init2()

   オプション [wr] に指定された、結合パラメータファイルが存在する場合には、そのファイルを、読み込んで FPM の結合パラメータの初期値とする。

4. is_dcp()

   DCP 実行させるか否かを判断する。その条件は、オプション [rm rn ci ce cm cn] にて指定する。

5. judge0()

   2 乗誤差を基準とする誤差計算を行う。教師信号とネットワーク出力信号の誤差の 2 乗和を求めている。オプション [ef] にて 0 指定の時、以下に示す計算を行う。

$$D_{MSE} = \sum_{g=1}^{M_m} \sum_{j=1}^{N_m} (t_j^{(g)} - a_{jm}^{(g)})^2 \tag{16}$$

   また、この関数で学習標本に対するバックワード (誤差逆伝播) 計算のスキップを判断している。

6. judge1()

   Kullback ダイバージェンスを基準とする誤差計算を行う。 オプション [ef] にて 1 指定の時、以下に示す計算を行う。

$$D_{KD} = \sum_{g=1}^{M_m} \sum_{j=1}^{N_m} t_j^{(g)} \log \frac{t_j^{(g)}}{a_{jm}^{(g)}} \tag{17}$$

   また、この関数で学習標本に対するバックワード計算のスキップを判断している。

7. learningphase0()

8. learningphase1()

   式 (11)、(12)、(13) に示すバックワード計算と式 (10) に示すパラメータ修正計算の一部を行なっている。

9. maintenance()

   標準出力へ現在の学習状況に関するログを書き出す。ログの内容については A.3を、参照されたい。

10. new_alpha_eta

DCP で使用する過去の影響量 α と学習率 η を求め、組合せを返す。α は、初期設定された値で定数である。η は、オプション [ei] での値の変化率を元に、以下のように動的に変化する。ただし、η_0 を DCP 法を適用する前の値とする。また、N は [ei] で設定した学習率の候補数、$x_1, \cdots, x_N$ は [ei] で設定された学習率の初期値である。

$$\eta_i = \eta_0 \ (i = \frac{N}{2} + 1)$$

$$\eta_i = \frac{x_i}{x_{i+1}} \eta_{i+1} \ (i = \frac{N}{2}, \cdots, 1)$$

$$\eta_i = \frac{x_i}{x_{i-1}} \eta_{i-1} \ (i = \frac{N}{2} + 2, \cdots, N)$$

DCP は、N 個の学習率の中から誤差を最小にするものを新しい学習率 η_0' を選択する。

11. new_weight()

learningphase0() または learningphase1() で求めた値を用いて以下の式のように結合パラメータ値の修正を行っている。ただし、修正後の値を $w'^{(sv)}_{k^m l^{m-1}}$ とする。

$$w'^{(sv)}_{k^m l^{m-1}} = w^{(sv)}_{k^m l^{m-1}} + \Delta w^{(sv)}_{k^m l^{m-1}} \tag{18}$$

12. searchphase()

フォワード計算を行う関数である。式 (1)、(2)、(3) に示す計算を行なっている。

13. set_in()

学習データを FPM に入力する。

14. set_teach()

教師信号データをセットする。

付録 D　関数構成図

```
main(){
        パラメータのセット
        set_option(){
                init_param()
                find_opt_file()
                オプションファイルより読み込む
                if read_optfile() {
                        set_argp()
                        read_param()
                }
                a r g v より読み込む
                set_argp()
                read_param()

        }
        学習データの読み込みと、結合パラメータの初期設定
        sample_read()
        if {
                init1()
        } else {
```

```
        init2()
}
エラー計算関数の選択
if {
        judge = judge0()
} else {
        judge = judge1()
}
ネットワーク使用モードの選択
if {  認識モード
        recog_only(){  認識ルーチン
                for {
                        set_teach()
                        set_in()
                        searchphase()
                        (*judge)()
        }
} else {  学習モード
        learning_main(){  学習メインルーチン
                各データ設定の為の初期学習
                set_in()
                searchphase()
                learningphase0()
                for {  ネットワーク学習繰り返しループ
                        if is_dcp() {  ＤＣＰ学習ルーチン
                                α、ηの候補の算出と、結合パラメータの仮修正
                                new_alpha_eta()
                                new_weight()
                                for {  DCP 学習
                                        set_teach()
                                        set_in()
                                        searchphase()
                                        if (*judge)() {
                                                if {
                                                        learningphase0()
                                                } else {
                                                        learningphase1()
                                                }
                                        }
                                        new_weight()
                                }
                                for {  α、ηを決定する認識テスト
                                        set_teach()
                                        set_in()
                                        searchphase()
                                        (*judge)()
                                }
                        } else {  前学習回における結合パラメータの最終修正
                                new_weight()
                        }
                        結合パラメータのファイルへの出力
```

```
                                write_weight()
                                for {  通常学習ルーチン
                                        new_weight()
                                        set_teach()
                                        set_in()
                                        searchphase()
                                        if (*judge)() {
                                                if {
                                                        learningphase0()
                                                } else {
                                                        learningphase1()
                                                }
                                        }
                                }
                                ネットワーク学習情報の出力
                                maintenance()
                        }
                        学習終了時における結合パラメータのファイルへの出力
                        new_weight()
                        write_weight()
                }
        }
}
```

付録 E　移植時の変更点

E.1　移植前後のＦＰＭ学習アルゴリズムの違い

移植後の変更点は、(1)DCP 法の導入と、(2) 結合パラメータ修正間隔の設定である。移植前後のアルゴリズムの違いについては、図 3と図 2とを参照していただきたい。図 2の、” FPM 通常学習 ”が移植前の FPM アルゴリズムの処理 (図 3) にあたることに注意していただきたい。移植前のプログラムでは、関数 ” learningphase() ” の中で、バックワード計算と結合パラメータ修正を行っていたが、移植後ではバックワード計算と結合パラメータ修正とを ” learningphase() ” と ” new_weight() ” との二つに分割し、DCP 法における複数の α, η の組合せを使ったパラメータ修正と仮学習を可能としている。またこの変更に伴い、ネットワークのデータエリアも複数個用意することとなった。また、結合パラメータの修正間隔を設定する（確率的降下法だけでなく、最急降下法による誤差逆伝播法も選択できるようにする）ため、移植前では、一つの標本の学習毎に結合パラメータの修正を行なっていたのに対し、移植後では指定した回数の間、修正量を累積し、その後まとめて修正している。この操作するためにも、前述の関数の分割する必要があった。

E.2　TDNN の DCP 法と FPM の DCP 法との違い

本報告で使ったプログラムでは、FPM の DCP 法は TDNN の DCP 法を参考にコーディングを行なっている。しかし、DCP 法における仮学習の方式を多少変更している。TDNN 版 [2] においては、その時点の学習で対象とする標本について誤差を累積し、その後一度にパラメータを修正している。この標本数は、初めから全標本を使うのではなく、学習の進み具合に合わせて増加されたいく。それに対して FPM の DCP 法では、一つの学習標本ごとに、あるいはまとまった標本ごとに誤差を累積し、パラメータ修正を行なっている。またその時の DCP 学習は、学習標本全てに対して行うのではなく、ある任意の数の標本（例えば、学習カテゴリの数）を全標本の中から任意に選んで行なっている。本報告での高速化は大量の標本を用いて FPM を設計することを想定しており、DCP 学習を TDNN 版と同じ要領で進めると、DCP 仮学習に要する時間がネットワーク全体の学習時間を増加させる大きな原因となる。そのため、少量の標本による仮学習方式を選択した。また、TDNN 版にはなかった DCP 評価ループが FPM 版には存在する (図 2)。これは少量の標本により η, α の正確な組合せを得ようとするために、TDNN 版のように学習しながら誤差を計算するのではなく、対象標本に対するパラメータの修正が全て終了した

後で、組合せを判断することが必要になった。

付録 F　使用上のこつ

オプションファイルを使った、様々な指定による種々の実験により、今回の高速化の手法が有効であることがわかった。そこで、得た経験による使い方のこつは以下のとおりであった。

- η を可変とし、α を固定する。
- η の変化率については、適度に小さい方が良い（1/1.2・1.2 倍ぐらい）。
- 結合パラメータの修正をある回数ごとにする。本報告の場合、カテゴリ数と同じにした。

本報告の DCP 法の設定値においては、η, α の組合せを減らし、他のオプション（[rm rt ce cw cm] など）も利用して高速化を行なっている。結合パラメータ修正間隔については、その値を大きくすることで、学習時間の短縮に寄与するのだが、間隔が長すぎると実験結果からもわかるように毎回更新時に比べて識別率が高くならなかったので、先のような値が求まった。これらを参考に、利用されたい。

付録 G　開発に関する今後の課題

- 仮学習に要する時間を短くするアルゴリズムを考える必要がある。
- DCP 法の仮学習用のネットワークデータエリア、学習サンプル読み込みエリア等で占有するメモリの節約化。これに対しては学習サンプル読み込み方法の改善などが、有効であるかと思われる。今回の実験での、多数話者音素識別などでは、テストプログラムで実メモリー 45MB 余りを必要とした。これ以上大規模なネットワークの学習をさせるとなると、学習計算以外の要素で（例えばスワップ）、実行時間を長引かせることとなる。
- 汎用化のための修正をする。現在 3 層及び 4 層固定となっているが、オプションで層数、ユニット数、素子の次元数などを指定できるようにする。