修士論文

SpikePropにおける
複数パターンの連続入力に対する
応答の改善

平成30年度修了
三重大学大学院工学研究科
博士前期課程 電気電子工学専攻

小野田 憲悟

目 次

図 目 次

第1章　序論

本研究では，今日，注目を集めている情報処理技術の一つである機械学習のうち，人工ニューラルネットワーク（NN: Artificial Neural Network）に着目する．NN の中でも，ユニット間の信号のやり取りをスパイク（パルス）で行うスパイキングニューラルネットワーク（SNN: Spiking Neural Network）に着目し，SNN の一種である SpikeProp を研究対象とする．

1.1 節では今日の情報処理技術の一つである機械学習について，1.2 節では NN の歴史について，1.3 節では本研究の対象である SNN と，本研究の目的や本論文の構成について述べる．

1.1　はじめに

今日，情報処理が担う役割は大きなものとなっており，さらなる情報処理技術の発展が期待されている．その中でも，多数のデータをコンピュータに与え，学習させることによって複雑な動作を実現できるようにする技術である機械学習が注目を集めている．機械学習により，これまで計算機には困難とされてきたことが実現できるようになった．たとえば，機械学習の一種である NN を用いて囲碁のプロ棋士に勝利できるプログラムが開発された [1]．NN は，脳の仕組みをモデル化することにより，脳の優れた能力をコンピュータ上で実現しようとするものである．

1.2　ニューラルネットワーク

NN は, 1943 年に McCulloch と Pitts が形式ニューロンという最初の人工ニューロンを発表したことを起源とする [2]．1958 年，形式ニューロンに基づき，Rosenblatt

によって単純パーセプトロンと呼ばれる最初の NN が提案された [3]．単純パーセプトロンは，入力層と出力層の 2 層からなるシンプルな構造であるにも関わらず，簡単なパターン認識などを解くことができ，注目を集めた．しかし，Minsky らによって，単純パーセプトロンは線形分離不可能な問題を識別できないことが指摘された [4]．これにより，NN は一時世間の関心を失ったが，研究は続けられた [5–7]．その後，1986 年に Rumelhart らが NN に誤差逆伝搬法（BP 法）を提案した [8]．これは，入力層と出力層の間に，中間層（隠れ層）を導入し，学習を行うことで，非線形分離問題を扱うことができる．これにより，再び NN が注目され，研究が広く行われるようになった [9,10]．しかし，誤差逆伝搬法を 4 層以上の深層 NN に適用することが試みられたが，局所解への収束や勾配消失の問題が明らかとなり，90 年代後半には下火になった．その後，2006 年の Hinton らの研究 [11] がきっかけとなり，深層 NN の研究が再び盛んになった．2012 年には，Krizhevsky らの深層 NN が画像認識で従来の手法を大きく上回る性能を示した [12]．深層 NN の有効性が広く認知されるようになり，画像認識や音声認識，自然言語処理などあらゆる分野に応用されている．

一般的な NN では生体の神経細胞における平均発火率の概念に基づいた人工ニューロンモデルが用いられる．それとは別に，神経細胞のダイナミクスに注目したモデルであるスパイキングニューロンと呼ばれる人工ニューロンモデルが研究されてきた．代表的なものとして 1952 年に Hodgkin と Huxley が提案した Hodgkin-Huxley モデルがある [13]．その他にも神経細胞をモデル化する際の抽象化の程度が異なるさまざまなモデルが提案されている [14,15]．

1.3　スパイキングニューラルネットワーク

NN の中でも，ユニット間の信号のやり取りをスパイク（パルス）で行う SNN が注目を集めている [16,17]．生体の脳は，ニューロン間の情報伝達がスパイク（パルス）状の信号により行われ，スパイクを受け取ったニューロンは内部状態が変化する．SNN は，このような動作をモデル化した NN であり，実際の脳のニューロンのふるまいを重視した生体に近い動作をするモデルである．これは，NN の中では生体の模倣度が高いことから，高度な情報処理能力を持つことが期待されてい

る．スパイキングニューロンモデルは，シグモイド型のニューロンモデルよりも計算能力があるという報告がされている [18]．シグモイド型のニューロンモデルを用いた NN は入出力として実数値を用いるが，SNN は一連のスパイク（スパイク列）を入出力として用いる．ネットワーク内の信号伝達がスパイクによって行われるため，SNN はノイズに強く，ハードウェアでの実装に有利である．また，スパイク列により情報を表現するため，時系列情報の処理に活用することができる．

SNN の情報表現手法は，大きく 2 種類に分けることができる [19]．一方はスパイクの密度により情報を表現する手法 (Rate coding)，もう一方はスパイクの時刻により情報を表現する手法 (Temporal coding) である．応答速度の観点においては，密度を観測する時間がかからない後者の方が好ましいとされる [20]．Temporal coding を用いた SNN として，さまざまなモデル・学習法が提案されてきた [21–24]．その中の一つに Bohte らが提案した SpikeProp がある [25]．SpikeProp は，フィードフォワード型の階層型のネットワークを持ち，誤差逆伝搬法に基づき学習を行う．このモデルは，ユニット間の結合を多重にすることにより，時間遅れの学習を行うことなく出力スパイクの時刻を学習できる．しかし，Bohte らの SpikeProp はユニットの発火は 1 回に限られ，1 つのスパイクしか出力できない．つまり，複数のスパイクを出力することができないため，複数回の出力を必要とする時系列情報の処理は行えない．

そこで，Booij らは高々 1 回しか発火しないユニットを発火が複数回できるよう，また，複数の出力スパイクの時刻を学習できるように拡張した [26]．これにより，スパイクを次々と入力した場合においても，複数のスパイクを出力することが可能となり，時系列情報処理システムへの応用が期待されている．しかし，学習する出力スパイクの数よりも実際に出力されるスパイクの数が多い場合には，出力スパイクの数を陽に調整できない．つまり，所望の出力が得られた後にも，余分なスパイクを出力することがある．これは，SpikeProp の処理をスパイク列からスパイク列へのパターン変換とみなしたとき，望ましくない．そのため，所望の出力スパイクが得られたとき，それ以上の出力がされないように動作をリセットするような機能が必要となる．しかし，パターンにより，所望の出力スパイクの数が異なる場合，スパイクが余分かどうかの判断ができないため，そのような機能は実現が困難である．また，そのような機能がある場合，内部状態が消えてしま

うため，過去の入力により出力が変化するような処理への対応が難しい．

そこで，松本らによって，出力スパイクの時刻と数の両方を学習できるような調整荷重減衰（AWD）法と呼ばれる学習法が提案された [27]．これにより，所望の出力スパイクの数が異なる場合においても，リセットすることなく連続で使用することができる．これにより，スパイク列からスパイク列への変換と逐次入力が原理的に可能となった．しかし，実際には大量のスパイク列の逐次入力，つまり複数スパイク列（複数パターン）の連続入力には課題が残る．

本研究では，スパイクの時刻と数を学習する SpikeProp において複数パターンが連続入力された際に正常な出力ができるようにすることを目標とする．SpikeProp は，AWD 法を用いることで一組のスパイク列（パターン）がネットワークの初期状態から入力された際に，望みの一組のスパイク列を出力することができる．しかし，複数パターンが連続で入力された際，つまり初期状態でないネットワークに対してスパイク列が入力された際には望みの出力が得られない場合がある．これは，SpikeProp の動作原理，および学習法に由来するものである．そこで，各ユニットの内部状態を決定している応答関数を変更する．また，複数パターンを一組のパターンとして組み合わせて学習させる，組み合わせパターン学習法を提案する．

以下に本論文の構成を示す．第 2 章では SpikeProp のモデルとその学習法，またその問題点について述べる．第 3 章では複数パターンの連続入力に対する応答の改善手法について提案・検討を行う．第 4 章で本研究についてまとめる．

第2章　SpikeProp

本研究は，SNN の一種である SpikeProp を対象とする．出力スパイクの数を学習しない SpikeProp においては，出力スパイク時刻の学習性能を向上させるため，学習率を調整する方法がいくつか提案されてきた [28–30]．本研究では，スパイクの数も学習するために，荷重減衰法とパラメータの調整法を導入している松本らの学習法に着目する．ネットワークモデルには Booij らによって提案されたモデル [26]，学習法には松本らによって提案された学習法 [27] を用いる．

2.1 節では，Booij らのネットワークモデルについて，2.2 節では，松本らの学習法について述べる．2.3 節では，SpikeProp の問題点について述べる．

2.1　モデル

SpikeProp のネットワークは，スパイキングニューロンにより構成されたフィードフォワード型のニューラルネットワークである．これは，図 2.1 に示すような入力層，中間層，出力層からなる階層型のネットワークを構成する．入力層はネットワークに入力されたスパイクを分配する役割を持ち，中間層・出力層はスパイキングニューロン（ユニット）からなる．ユニットは内部状態変数がしきい値を超えるとスパイクを出力する．入力スパイクと内部状態変数の関係はスパイク応答モデル（SRM）[15] によって記述される．

入力層およびそれぞれのユニット間の結合は，副結合と呼ばれる複数の結合からなる．各副結合は，時間遅れ d と結合荷重 w という 2 種のパラメータを持つ．k 番目の副結合の時間遅れを d^k，接続元ユニット i から接続先ユニット j への k 番目の副結合の結合荷重を w_{ij}^k のように表す．

次に，ネットワークにおけるスパイクの伝達とユニットの動作について説明する．ユニット i が時刻 t_i に発火し，スパイクが出力されるとする．このとき，ユ

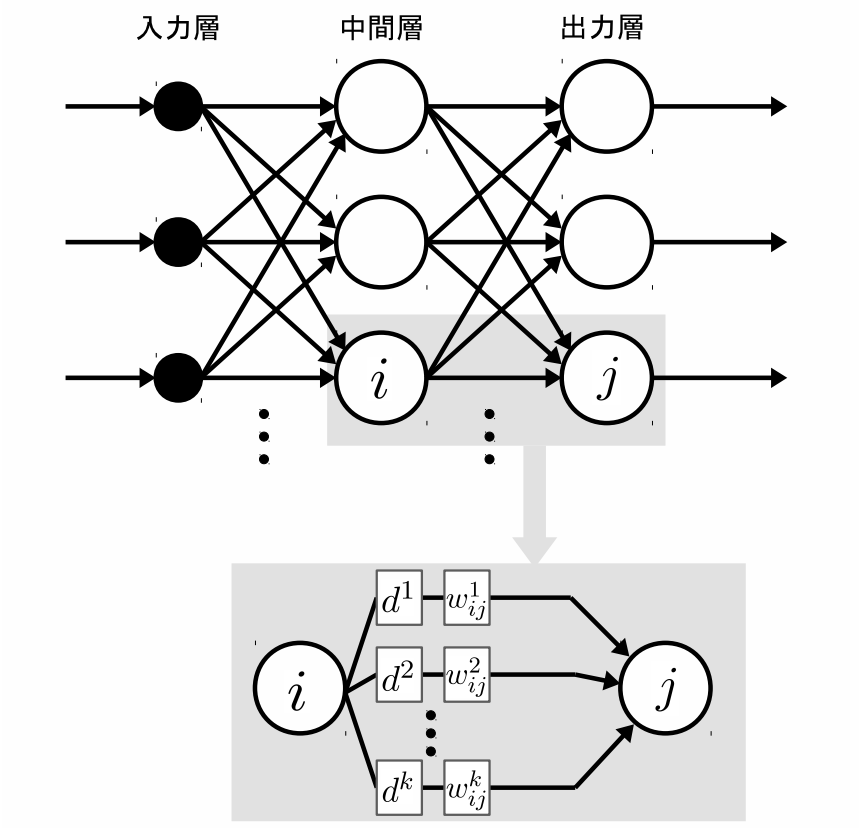

図 2.1: SpikeProp ネットワーク

ニット i からユニット j が接続先の結合にスパイクが入力される．入力されたスパイクは，時間遅れ d^k を持つ k 番目の副結合を通り，t_i から d^k だけ遅れてユニット j に伝達される．各結合の終端で，スパイクは式（2.1）で表されるスパイク応答関数に従って，時間変化する刺激になり，ユニット j の内部状態 x_j に応答を発生させる．ここで，τ_m，τ_s は時定数であり，A は任意の定数である．

$$\epsilon(t) = \begin{cases} 0 & (t < 0) \\ A\exp\left(-\frac{t}{\tau_m}\right) - \exp\left(-\frac{t}{\tau_s}\right) & (t \geq 0). \end{cases} \tag{2.1}$$

ユニット j の内部状態 x_j は，結合荷重 w_{ij}^k により重みづけされた応答がすべて足し合わされることで定まる（式（2.2））．

$$x_j(t) = \sum_{t_j^f \in F_j} \kappa(t - t_j^f) + \sum_{i \in \Gamma_j} \sum_{t_i^f \in F_i} \sum_{k=1}^{m} w_{ij}^k \epsilon(t - t_i^f - d^k), \tag{2.2}$$

$$\kappa(t) = \begin{cases} 0 & (t < 0) \\ -\theta \exp\left(-\frac{t}{\tau_r}\right) & (t \geq 0). \end{cases} \tag{2.3}$$

$\kappa(t)$ は不応期関数であり，式（2.3）で表される．θ は発火しきい値，Γ_j はユニット j に繋がる結合の接続元ユニットの集合，F_i はユニット i のスパイクの発火時刻の集合，F_j はユニット j のスパイクの発火時刻の集合，m は副結合数，τ_r は時定数である．

ユニットは，内部状態があらかじめ決められたしきい値 θ を超えたとき発火し，スパイクを出力する．ユニット j が時刻 t_j にスパイクを出力したとすると，不応期関数 $\kappa(t)$ により，内部状態は $x_j(t_j) = 0$ となる．その後，内部状態が再びしきい値を超えることで複数回の発火ができる．また，不応期関数は発火直後に再発火することを防止している．最終的に出力層から送り出されるスパイクがネットワークの出力となる．

2.2 学習法

ここで，複数組の入力スパイク列から出力スパイク列への変換を学習することを考える．この際，ネットワークの最終的な出力は，各スパイクの時刻を合わせるだけでなく，スパイクの数も合わせるものとする．

SpikeProp は誤差逆伝搬法に基づき，評価関数である誤差 E（式（2.4））を最小化するように結合荷重を繰り返し調整することでスパイクの時刻を学習する．

$$E = \frac{1}{2} \sum_{p \in P} \sum_{j \in J} \sum_{f=1}^{\hat{f}_j} \left(t_j^{p,f} - \hat{t}_j^{p,f} \right)^2. \tag{2.4}$$

ここで，P は 1 回の更新に用いる教師データの集合，J は出力ユニットの集合，$\hat{f}_j$ はユニット j の期待されるスパイク数である．また，$t_j^{p,f}$，$\hat{t}_j^{p,f}$ は教師パターン p におけるユニット j の f 番目のスパイク出力時刻および期待されるスパイク時刻である．結合荷重の更新量は式（2.5）に従い，繰り返し更新される．

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k}. \tag{2.5}$$

ここで，$\eta(> 0)$ は学習率である．

スパイク列からスパイク列の変換を考えたとき，スパイクの時刻だけでなくその数を調整する必要がある．そこで，余分な出力スパイクを抑制するために，荷重減衰（WD:Weight Decay）法 [31] を用いる．荷重減衰では，評価関数に L2-正則化項を付加することで，不要な結合荷重を減衰させる．荷重減衰の導入により，評価関数は式（2.6）のようになる．

$$E' = E + \frac{\rho}{\eta} \sum_{w \in W} w^2. \tag{2.6}$$

ここで，W はすべての結合荷重の集合，ρ は荷重減衰の係数である．結合荷重の更新量 Δw は，式（2.7）で求められる．更新式は式（2.8）となる．

$$\Delta w_{ij}^k = -\eta \frac{\partial E}{\partial w_{ij}^k} - 2\rho w_{ij}^k, \tag{2.7}$$

$$w_{ij}^k \leftarrow w_{ij}^k + \Delta w_{ij}^k. \tag{2.8}$$

もし，学習により過度に結合荷重が抑制され，ユニットの発火が不足した場合，必要な数の発火をするように結合荷重の補正が行われる．この補正は，誤差逆伝搬法と荷重減衰法による学習に関係なく実行されるため，学習過程に悪影響を及ぼすと考えられる．

そこで，過度に荷重減衰を適用することを避けるために，調整 WD（AWD: Adaptive WD）法が提案された [27]．この手法は学習率 η と荷重減衰の係数 ρ を学習の進行に応じて調整する．具体的には，学習率 η は誤差が十分に減少したときに小さくし，荷重減衰の係数 ρ は余分なスパイクが出力されないときに小さくする．このようにすることで，学習終盤でスパイクの発火時刻かスパイクの数が調整しきれていない場合は，それに集中して調整するようになり，学習が効果的に進むと考えられる．なお簡単のため，いずれのパラメータも 2 段階で値を切り替える．

学習率 η は，η_H，η_L（$\eta_H > \eta_L$）を次の基準で切り替える．学習開始から誤差がある値（$E_{th_\downarrow}$）まで減少するまでは，η_H を用いる．これにより，誤差を減少させる効果を強くする．誤差が $E_{th_\downarrow}$ を下回った後は，η_L を用いて誤差を減少させる効果を弱くし，誤差を減少させる学習から余分なスパイクを押さえる学習に切り替える．その後，誤差が $E_{th_\uparrow}(> E_{th_\downarrow})$ 以上になれば，再び学習率に η_H を用い，以下同様に学習率を調整する．図 2.2 に切り替えの様子を示す．このようにヒステリシスな特性を持たせることで，η が頻繁に切り替わることを防いでいる．

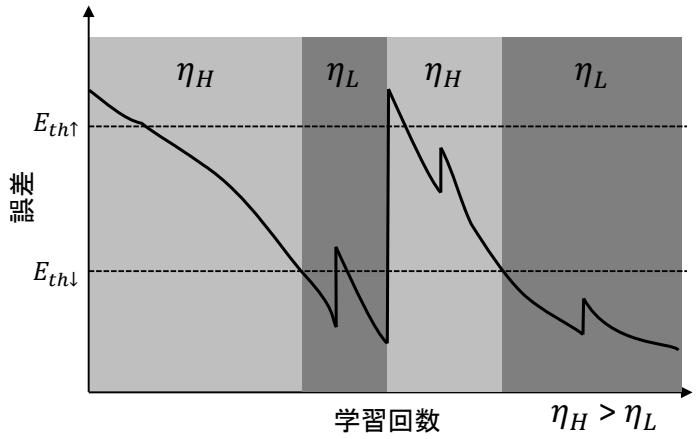

図 2.2: 学習率の切り替え

荷重減衰の係数 ρ は，ρ_H，ρ_L（$\rho_H > \rho_L$）を次の基準でユニットごとに切り替える．余分な発火をしているユニットへとつながる結合の荷重の更新については ρ_H を用いることで，発火を抑制する効果を強くする．それ以外の余分な発火をしていないユニットについては ρ_L を用いることで，学習の妨げとならないようにする．余分な発火をしているかどうかは，出力層ユニットについては教師データにより判定できる．中間層ユニットについては，必要な発火数が不明であるため，あらかじめ決めておいたしきい値（θ_{num}）を超えた場合に余分な発火と判定する．図 2.3 に ρ の選択例を示す．また，η_H，η_L，ρ_H，ρ_L，θ_{num} については，学習前に適切な値を定める必要がある．

結合荷重の更新過程において，内部電位がしきい値に達しないためにユニットが発火しない場合や発火回数が不足する場合がある．このようなユニットが存在する場合には，誤差の逆伝搬量を計算することができない．そのため，再びユニットが発火するように，発火しないユニットにつながる結合の結合荷重を大きくする．この動作を発火補正と呼びユニットが発火するまで繰り返す．本稿では該当の結合の結合荷重に一定値を加算することで行う．

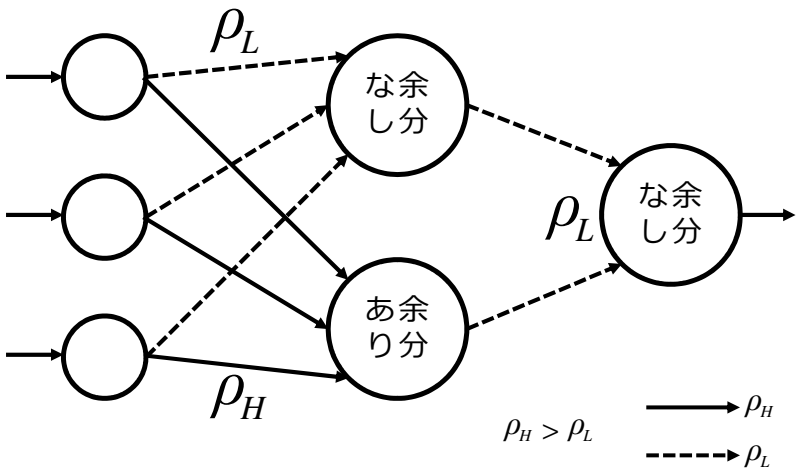

図 2.3: 係数の選択

2.3 問題点

SpikeProp は複数の入力パターンを連続して与えたときに，後続の入力パターンに対して好ましくない動作をする．この原因は，主に SpikeProp の動作原理に由来するものである．SpikeProp は，ユニット間の副結合における時間遅れ d および式 (2.1) で表されるスパイク応答関数によって，入力された情報をしばらくの間保持する．また，応答関数によって保持される情報は時間減衰する．このようにして複数保持した情報をもとにして，時系列情報の処理を実現している．

しかし，複数パターンが連続入力される際に，先行入力パターンによって生じた情報の減衰が不十分なまま次のパターンが入力されると，その出力は先行入力パターンによって生じた情報の影響を受けて変化する．図 2.4 に，あるユニットに 2 パターンが連続入力された際の，第 1 パターンの影響によって第 2 パターンの出力が変化する様子を示す．図の紺色の実線が 2 パターンが連続入力された際にユニット内部に実際に発生する内部電位，灰色の実線が第 2 パターンのみが単独で入力された際に発生する内部電位，破線が単一スパイクが入力された際に発生する内部電位を表したものである．先行入力パターンである第 1 パターンの出力後も，第 2 パターンの最初のスパイク (P2-1) が入力される時点では，内部電位がネット

ワークに残っている．また，この残っている内部電位によって，第 2 パターンの入力に対する出力時刻が望みの出力時刻から変化している．このように，小さい間隔で複数パターンが連続入力された際には，望みの出力が得られない．

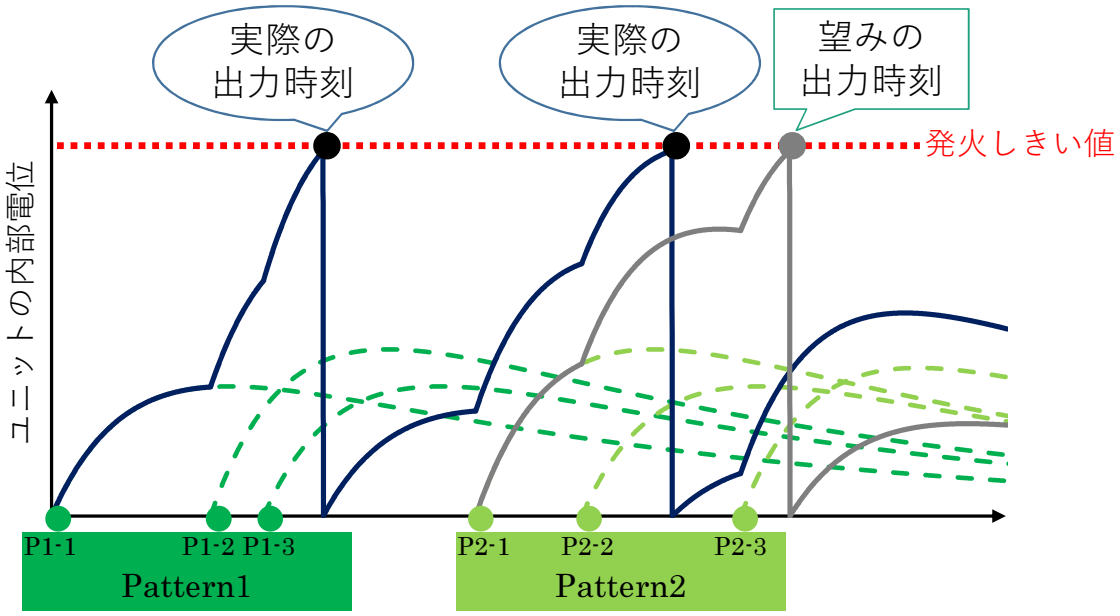

図 2.4: あるユニットに複数パターンが連続入力された際に出力が変化する様子

第3章　複数パターンの連続入力に対する応答改善手法

本章では，複数パターンが小さい間隔で連続して入力される際の応答改善のために，スパイク応答関数の変更および学習法の変更を提案する．前節で述べた通り，複数パターンが連続入力される際に，先行入力パターンによって生じた情報の影響で望みの出力が得られないことがある．この問題を解決するために，スパイク応答関数を変更する手法，学習法を変更する手法，これら両手法を併用する手法の 3 手法を提案し，複数パターンの連続入力に対する応答の改善をめざす．

3.1　スパイク応答関数の変更

本節では，先行入力パターンによって生じた情報の影響を軽減するために，スパイク応答関数を素早く時間減衰するものへと変更する手法を提案する．

3.1.1　時定数の変更による時間減衰速度の向上

式 (2.1) で表されるスパイク応答関数は，2 つの時定数 τ_m，τ_s および，定数 A によって応答の立ち上がり速度，減衰速度，ピーク値が決定される．これらのパラメータについて，従来研究では学習の速度や安定性，計算速度の観点からの議論がされている [32]．しかし，スパイク出力後のネットワークの内部状態，つまり複数パターンが連続入力される際に，先行入力パターンによって生じる情報の影響については議論されていない．

そこで，これらのパラメータを変更し，スパイク応答関数を素早く時間減衰するものへと変更する．具体的には，τ_s および A を調整する．これにより，先行入

力パターンによる影響を軽減することができ，複数パターンの連続入力に対する応答が改善することが期待できる．

3.1.2 実験条件

応答関数を素早く時間減衰するものへと変更する手法の複数パターンの連続入力に対する効果を確認するために実験を実施した．

実験には先行研究で用いられている時間版 Iris 問題 [27] を用いた．これは，パターン認識の分野で古くからベンチマーク問題として用いられてきた Iris 問題 [33] の，4 つの特徴量をスパイク発火時刻によりエンコードしたものである．本研究では先行研究との比較と，非線形なパターン分類器としての性能を確認するためにこれを使用する．学習には各クラスからランダムに 25 パターンずつ，合計 75 パターンを抽出したものを使用し，残りの 75 パターンをテストに使用した．入力は各特徴量を 0ms から 6ms までに正規化したものを用い，先行研究同様に 0ms のバイアス入力を付加した．出力は各分類のユニットを設け，該当のユニットからの出力を 10ms，それ以外のユニットからの出力を 16ms にエンコードした．

ネットワーク構造は，入力層，中間層，出力層のユニット数はそれぞれ 5，10，3 個とし，副結合数は 8 とした．また，τ_m，τ_r はそれぞれ 10ms とした．τ_s は文献 [27] で用いられていた 5.0ms と，より素早く時間減衰する 2.0ms，0.5ms の 3 通りについて実験し，その際に応答関数のピーク値が 0.25 となるように A の値を調整した．本実験で使用した応答関数の概形を図 3.1 に示す．初期荷重の異なる 50 個のネットワークについて，それぞれの応答関数ごとに 30,000 回オンライン学習した．

学習後のネットワークは，テストパターンに対する応答により評価した．テストパターンは，テスト用に用意した 75 パターンから 3 個のパターンを選び，第 1 パターンとして 1 つのパターンをそのまま，第 2 パターンとして 1 つのパターンを一定の時間遅れ（組み合わせ間隔）をほどこし，第 3 パターンとして 1 つのパターンを一定の時間遅れ（組み合わせ間隔の倍）をほどこして準備し，これらをまとめて 1 つのテストパターンとして作成した．組み合わせ間隔を 20ms から 100ms まで 5ms 刻みで変更し，それぞれの間隔ごとに異なる 50 のテストパターンに対する出力の成功率を評価した．このとき，テストパターンに対する出力の成功条件を，

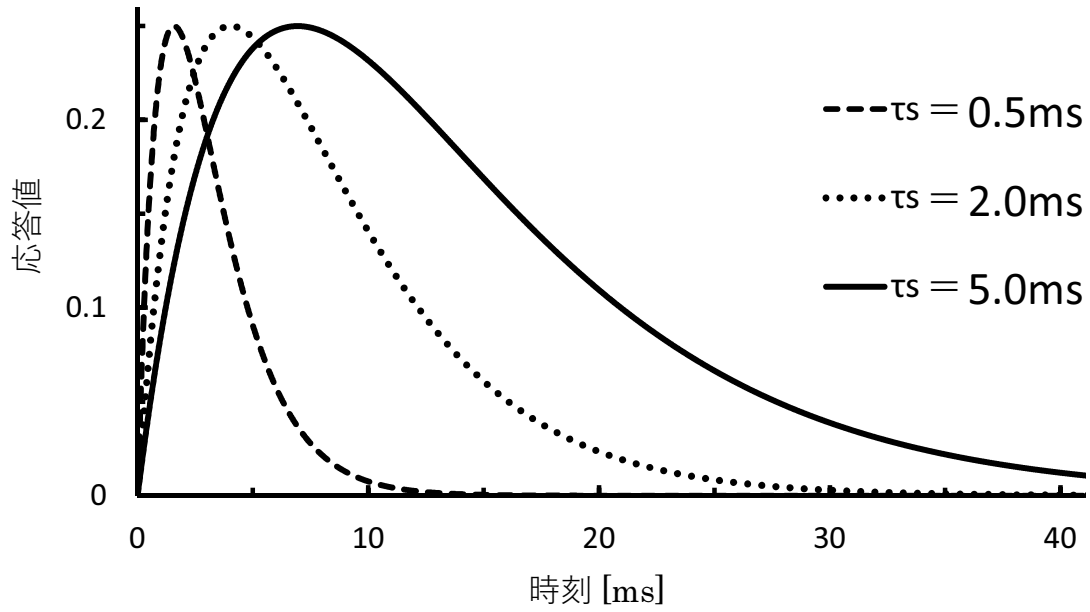

図 3.1: 時定数 τ_s を変更した際のスパイク応答関数の概形

余分なスパイクを出力せず，それぞれの出力スパイク時刻の誤差が 2ms 以下になることとした．すなわち，3 個の連続入力パターンすべてに対して所望の応答をした場合を成功とみなした．

3.1.3 結果と考察

応答関数を変更した際の，テストパターンに対する出力の成功率を図 3.2 に示す．図の縦軸が出力の平均成功率，横軸がテストパターンの組み合わせ間隔を表す．

まず，テストパターンの組み合わせ間隔が単一パターンの入出力間隔 (16ms) の約 4 倍以上の場合（65ms 以上）について着目する．この場合の成功率は，テストパターンの組み合わせ間隔が 65ms の場合を除き，文献 [27] で用いられていた $\tau_s = 5.0$ms の場合が最も高く，最も素早く時間減衰する $\tau_s = 0.5$ms の場合が最も低い．例えば，テストパターンの組み合わせ間隔が 100ms の場合，$\tau_s = 5.0$ms の場合の成功率約 82.5%に対し，$\tau_s = 2.0$ms の場合の成功率が約 80.1%，$\tau_s = 0.5$ms の場合の成功率が約 78.1%であり，応答関数の変更によって成功率が約 4 ポイント低下した．また，テストパターンが 65ms の場合には τ_s の値による成功率の大きな差異は見られなかった．

次に，テストパターンの組み合わせ間隔が単一パターンの入出力間隔 (16ms) の約 2 倍から 4 倍の場合（35ms∼60ms）について着目する．この場合の成功率は，最も素早く時間減衰する $\tau_s = 0.5$ms の場合が最も高く，$\tau_s = 5.0$ms の場合が最も低い．例えば，テストパターンの組み合わせ間隔が 40ms の場合，$\tau_s = 5.0$ms の場合の成功率約 1.3%に対し，$\tau_s = 2.0$ms の場合の成功率が約 32.9%，$\tau_s = 0.5$ms の場合の成功率が約 70.9%であり，応答関数の変更によって成功率が約 70 ポイント上昇した．

また，テストパターン作成時の組み合わせ間隔が単一パターンの入出力間隔 (16ms) の約 1 倍から 2 倍の場合（20ms∼30ms）について着目すると，組み合わせ間隔が単一パターンの入出力間隔 (16ms) の約 2 倍から 4 倍の場合と同様に，素早く時間減衰する応答関数を使用した場合の成功率が最も高い．しかし，$\tau_s = 0.5$ms の場合においても，例えば組み合わせ間隔 25ms での成功率が約 6.3%であるなど，非常に低い成功率となっている．

これらの結果から，応答関数を素早く時間減衰するものへと変更することで，組み合わせ間隔が大きな場合での成功率がわずかに低下するが，組み合わせ間隔が比較的小さな場合での成功率が大きく向上すると言える．つまり，本手法は単一の時系列情報パターンの処理能力がわずかに低下するものの，複数パターンの連続入力に対する応答の改善に有効であると言える．しかし，組み合わせ間隔がとても小さい場合には，成功率は非常に低い．これは，応答関数によって前の入力パターンの影響が十分減衰される前に次のパターンが入力されてしまうためであると考えられる．また，応答関数をより素早く時間減衰するものへと変更することは，時系列情報を処理するための，ネットワーク内部に情報を保持する能力をさらに削ることとなる．つまり，単一の時系列情報パターンの処理能力のさらなる低下を招く．そのため，応答関数の変更は複数パターンの連続入力に対する応答の改善にある程度有効であるものの，本手法のみでより小さい入力間隔での複数パターンの連続入力に対して十分に応答を改善することは困難である．

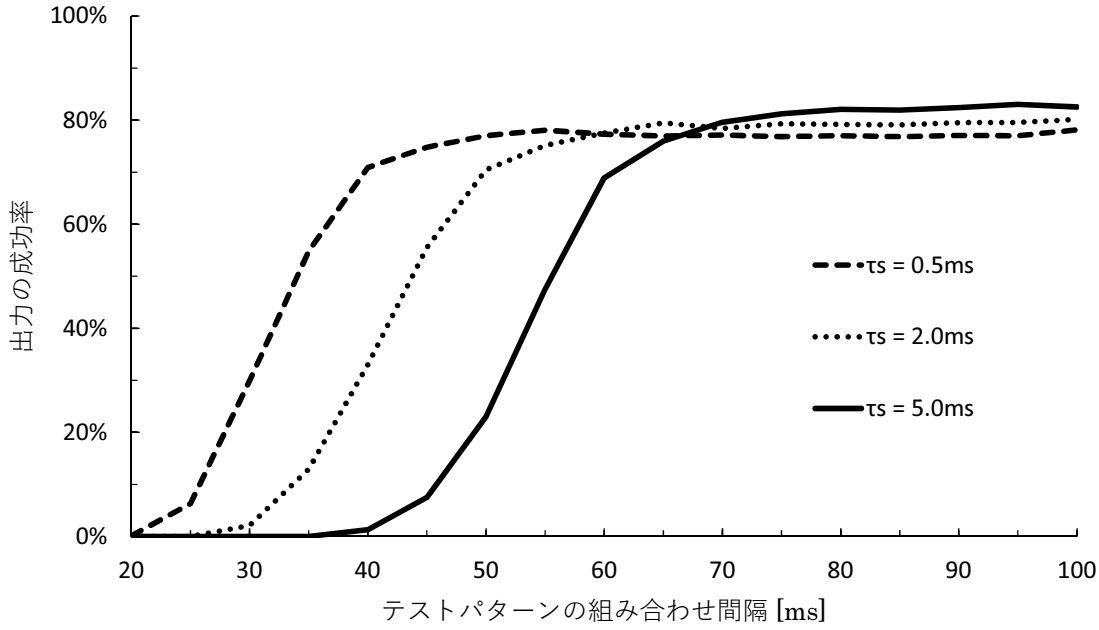

図 3.2: 応答関数変更時のテストパターンに対する出力の成功率

3.2　学習法の変更

本節では，先行入力パターンの影響がある状態からの動作を学習するために，複数パターンを組み合わせたものを学習する組み合わせパターン学習法を提案する．

3.2.1　組み合わせパターン学習法

従来の学習法では，初期状態からの入力に対する出力のみを学習している．つまり，複数パターンが連続入力される際の，先行入力パターンによって生じる情報の影響を考慮せずに学習している．そのため，小さい間隔で複数パターンが連続入力された際には，先行入力パターンによって生じる情報の影響によって学習時とは異なる条件でパターンが入力されるため，望みの出力が得られない．

そこで，n 個の複数パターンを組み合わせたものを学習する組み合わせパターン学習法を提案する．これは，第 1 パターンとして 1 つの教師パターンをそのまま，第 k $(k \leq n)$ パターンとして 1 つのパターンを一定の時間遅れ（組み合わせ間隔の $k-1$ 倍）を施して準備し，これらの n 個のパターンをまとめて 1 つのパターンとして学習する手法である．第 1 パターンにより先行入力パターンの影響がない状態からの動作を，第 2 パターン以降のパターンにより先行入力パターンの影響がある状態からの動作を学習する．このとき，教師パターン数によっては組み合わせ総数が膨大になってしまうため，バッチ学習ではなくオンライン学習を用いる．これにより，第 2 パターン以降の後続パターンよりも先行パターンの影響が小さい場合については，正常な出力が得られるよう学習できる．つまり，学習時の組み合わせ間隔よりも大きい間隔での連続入力に対して応答を改善することができる．

また，組み合わせパターン学習法において，どの程度先行入力パターンの影響を考慮するかによって，複数パターンの連続入力に対する性能が変化すると考えられる．つまり，組み合わせるパターンの個数によって，連続入力に対する応答の性能が変化すると考えられる．先行入力パターンの影響が小さすぎる場合，つまり組み合わせるパターン数が少なすぎる場合には，当然連続入力パターンに対する応答が十分改善しない．しかし，先行入力パターンの影響が大きすぎる場合，つまり組み合わせるパターン数が多すぎる場合には，学習が困難になることや，学

習時の組み合わせ間隔での連続入力以外の間隔での連続入力に対する応答が悪化することが考えられる．

3.2.2　実験条件

組み合わせパターン学習法を実施する際の組み合わせるパターン数による連続入力に対する応答の性能への影響を確認するため，および組み合わせパターン学習法の連続入力に対する応答改善への有効性を確認するために実験を実施した．

実験には 3.1 節同様に時間版 Iris 問題を用いた．また，ネットワーク構造は τ_s を 5.0ms とし，その他のパラメータは 3.1 節同様に設定した．

学習時のパターン組み合わせ間隔を 25ms から 50ms まで 5ms 刻みに変更し，また学習時の組み合わせパターン数を 2，3，4 個に変更し，それぞれの間隔，個数について初期荷重の異なる 50 のネットワークに対して 30,000 回組み合わせパターン学習を実施した．学習後のネットワークは，3.1 節同様にテストパターンに対する応答で評価した．

3.2.3　結果と考察

それぞれの組み合わせ間隔で組み合わせパターン学習を実施した際の，各組み合わせパターン数ごとのテストパターンに対する出力の成功率を図 3.3～ 図 3.8 に示す．図の縦軸が出力の平均成功率，横軸がテストパターンの組み合わせ間隔を表す．

組み合わせパターン数による連続入力に対する応答の性能への影響について

第 1 に，組み合わせパターン数による連続入力に対する応答の性能への影響について考える．

まず，学習時の組み合わせ間隔が比較的大きい場合（40ms~50ms）について着目する．それぞれの結果から，学習時の組み合わせ間隔が 50ms かつ組み合わせパターン数が 3 個の場合には，ほかの組み合わせパターン数と比べ大きい組み合わ

せ間隔のテストパターンに対する出力の成功率の低下がみられるが，その他の場合には成功率に大きな差異は見られない．

次に，学習時の組み合わせ間隔が比較的小さい場合（25ms~35ms）について着目する．学習時の組み合わせ間隔が 30ms，35ms の場合，学習時の組み合わせパターン数が 2 個の場合が，ほかの組み合わせパターン数とくらべ，全ての組み合わせ間隔のテストパターンに対する出力の成功率が最も高かった．また，学習時の組み合わせ間隔が 25ms の場合，テストパターンの組み合わせ間隔が 35ms 以上のテストパターンに対する出力の成功率は，学習時の組み合わせパターン数が 2 個の場合が最も高かった．

これらの結果から，大きい組み合わせ間隔での学習時には組み合わせパターン数による成功率への影響はあまりなく，小さい組み合わせ間隔での学習時には組み合わせパターン数が 2 個の場合が最も成功率が高かった．そのため，組み合わせパターン学習法においては，2 パターンを組み合わせて学習するのが連続入力に対する応答の改善に最も有効であるといえる．

組み合わせパターン学習法の連続入力に対する応答改善への有効性について

第 2 に，組み合わせパターン学習法の連続入力に対する応答改善への有効性について考える．ここでは，2 パターンを組み合わせて学習した場合（図 3.3~ 図 3.8 における実線の結果）について議論する．

まず，学習時の組み合わせ間隔が単一パターンの入出力幅の 2 倍よりも大きい場合（35ms~50ms）について着目する．それぞれの結果から，学習時の組み合わせ間隔よりも大きな間隔であれば，組み合わせパターン学習を実施していない際には成功率が低かった組み合わせ間隔のテストパターンに対しても 70%以上望みの出力が得られた．例えば学習時の組み合わせ間隔が 35ms の場合，図 3.2 の $\tau_s = 5.0$ms の場合と比較すると，組み合わせ間隔 35ms のテストパターンに対して，組み合わせパターン学習を用いなかった場合には成功率が 0%であったが，組み合わせパターン学習法を使用することで約 70.1%まで成功率が上昇した．

次に，学習時の組み合わせ間隔が単一パターンの入出力幅（16ms）の 2 倍よりも小さい場合（25ms，30ms）について着目する．学習時の組み合わせ間隔が 30ms の場合，図 3.2 の $\tau_s = 5.0$ms の場合と比較すると，テストパターンの組み合わせ

間隔が30ms~50msの場合には出力の成功率が上昇しているものの，さらに大きい組み合わせ間隔のテストパターンに対しては成功率が低下している．また，すべての組み合わせ間隔のテストパターンに対する成功率が60%未満であった．また，学習時の組み合わせ間隔が25msの場合には，すべての組み合わせ間隔のテストパターンに対する成功率が15%未満と低かった．

学習時の組み合わせ間隔が小さい場合に成功率が低かった原因を確かめるため，学習中の出力誤差の推移を確認した．図3.9に，成功率が低かったある初期荷重のネットワークに対して従来手法を用いた場合と，組み合わせ間隔25msで組み合わせパターン学習法を用いた場合の学習中の出力誤差の推移を示す．縦軸は出力誤差を対数軸で示したものであり，教師パターンの出力時刻の2乗誤差を，学習回数100回ごとに移動平均をとった値を表す．横軸が学習回数を表す．従来手法では，学習回数5,000回付近で誤差が急落し，その後も誤差が学習回数に応じて徐々に小さくなっている．しかし，組み合わせ間隔25msの組み合わせパターン学習法では，学習回数が増加しても，学習回数5,000回付近の二乗誤差10からほとんど減少していない．この傾向は，成功率が低かった他の初期荷重のネットワークに対しても見られた．これらの結果から，組み合わせ間隔が小さい場合の組み合わせパターン学習では，学習途中で誤差が停滞していることがわかる．これは，学習が停滞していることを意味する．そのため，連続入力パターンに対する成功率が低い．

これらの結果をまとめると，組み合わせパターン学習法を用いることで，従来手法では正常な出力が得られなかった組み合わせ間隔のテストパターンに対する出力の成功率が大幅に上昇した．つまり，本手法は複数パターンの連続入力に対する応答の改善に有効であるといえる．しかし，組み合わせパターン学習時の組み合わせ間隔よりも小さな間隔で組み合わされたテストパターンに対しては応答の改善がみられなかった．また，学習時の組み合わせ間隔が小さい場合には途中で学習が停滞してしまい，テストパターンに対する成功率も低かった．これは，先行入力パターンによる影響が大きすぎることで学習が困難になったためと考えられる．そのため，組み合わせパターン学習法は複数パターンの連続入力に対する応答の改善にある程度有効であるものの，本手法のみでより小さい入力間隔での複数パターンの連続入力に対して十分に応答を改善することは困難である．

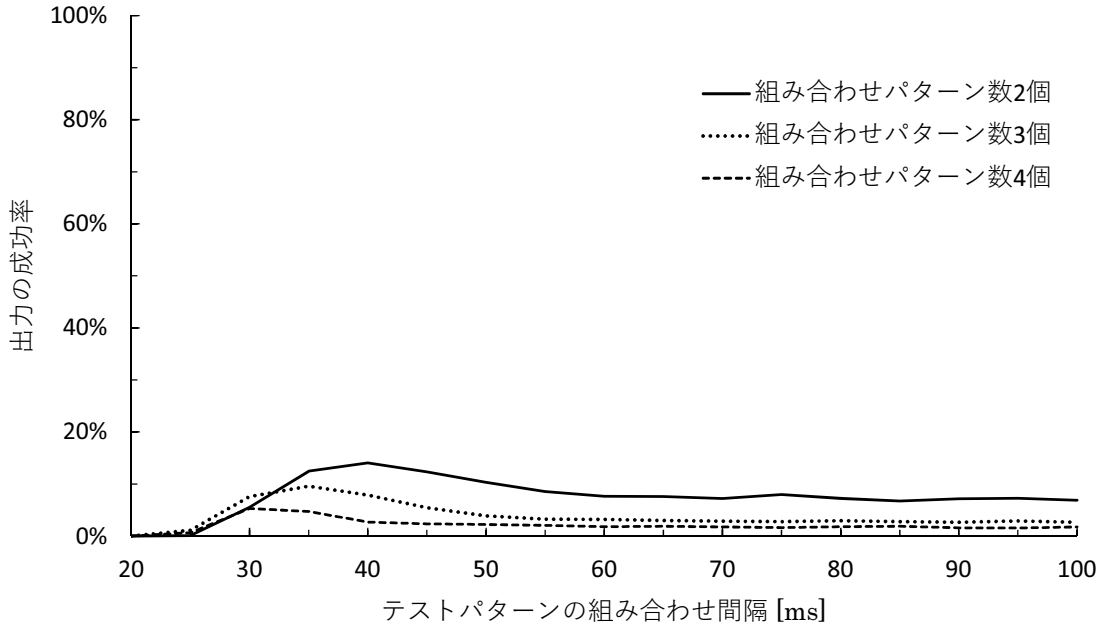

図 3.3: 学習時の各組み合わせパターン数ごとの組み合わせ間隔 25ms で学習したネットワークのテストパターンに対する出力の成功率

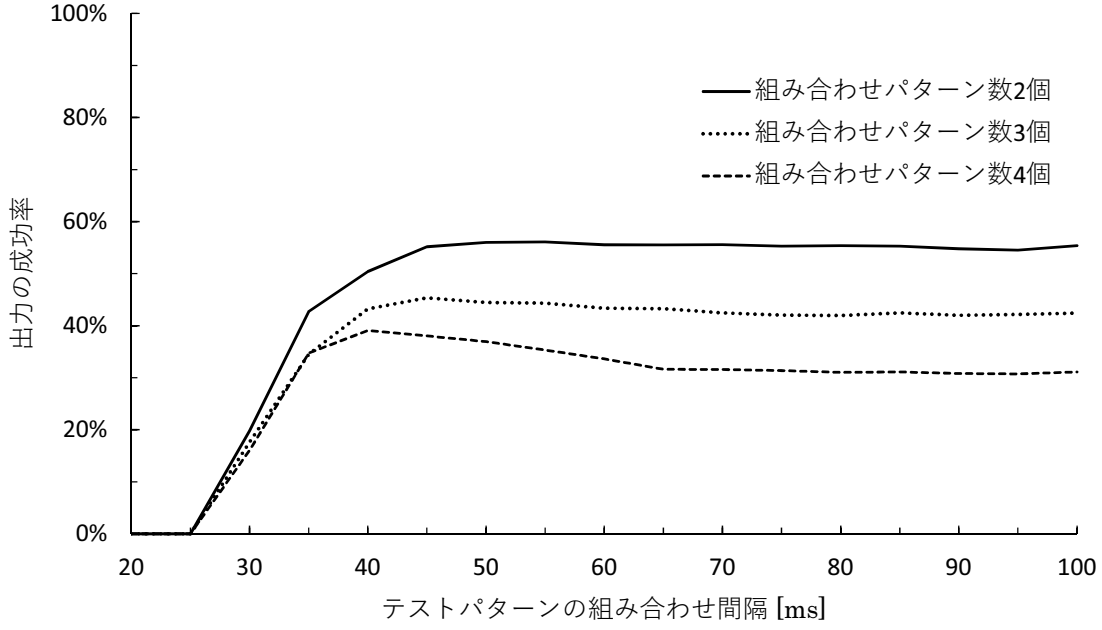

図 3.4: 学習時の各組み合わせパターン数ごとの組み合わせ間隔 30ms で学習したネットワークのテストパターンに対する出力の成功率

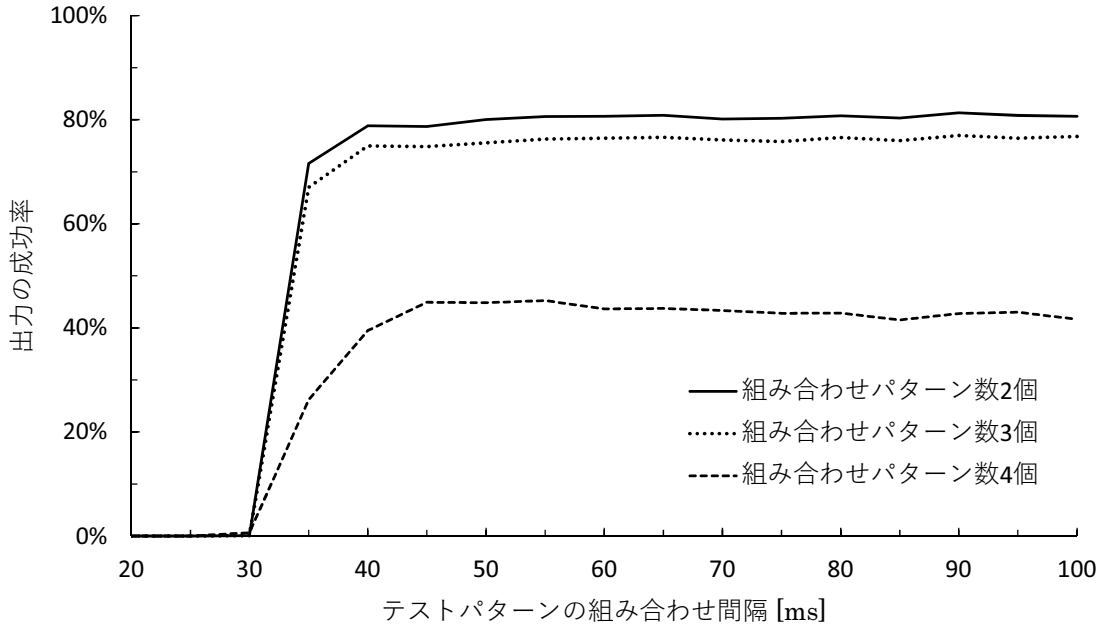

図 3.5: 学習時の各組み合わせパターン数ごとの組み合わせ間隔 35ms で学習したネットワークのテストパターンに対する出力の成功率

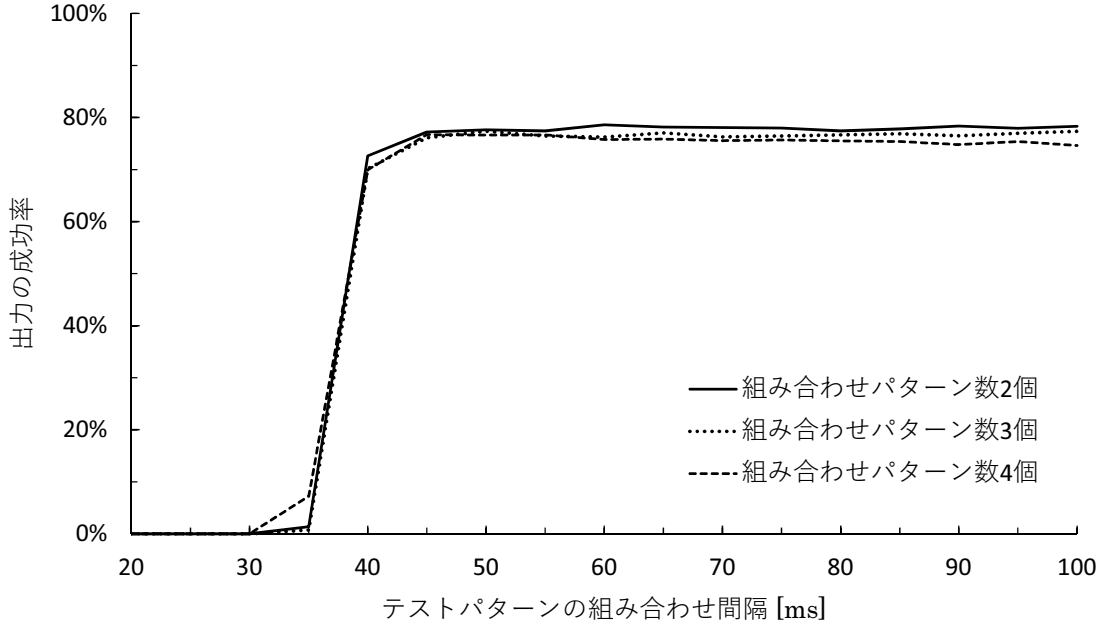

図 3.6: 学習時の各組み合わせパターン数ごとの組み合わせ間隔 40ms で学習したネットワークのテストパターンに対する出力の成功率

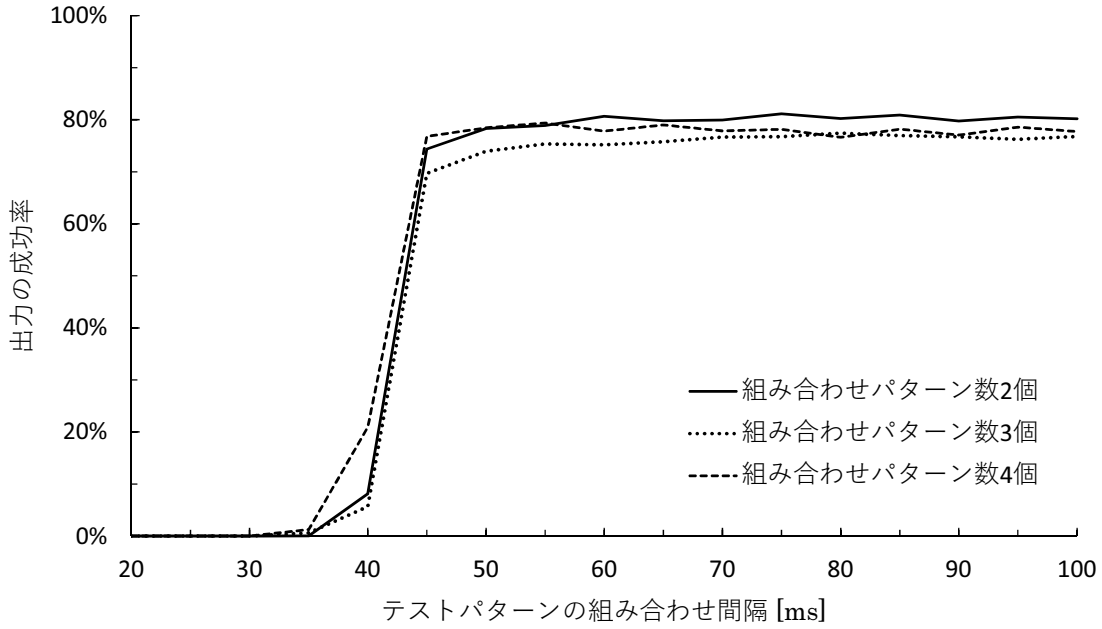

図 3.7: 学習時の各組み合わせパターン数ごとの組み合わせ間隔 45ms で学習したネットワークのテストパターンに対する出力の成功率

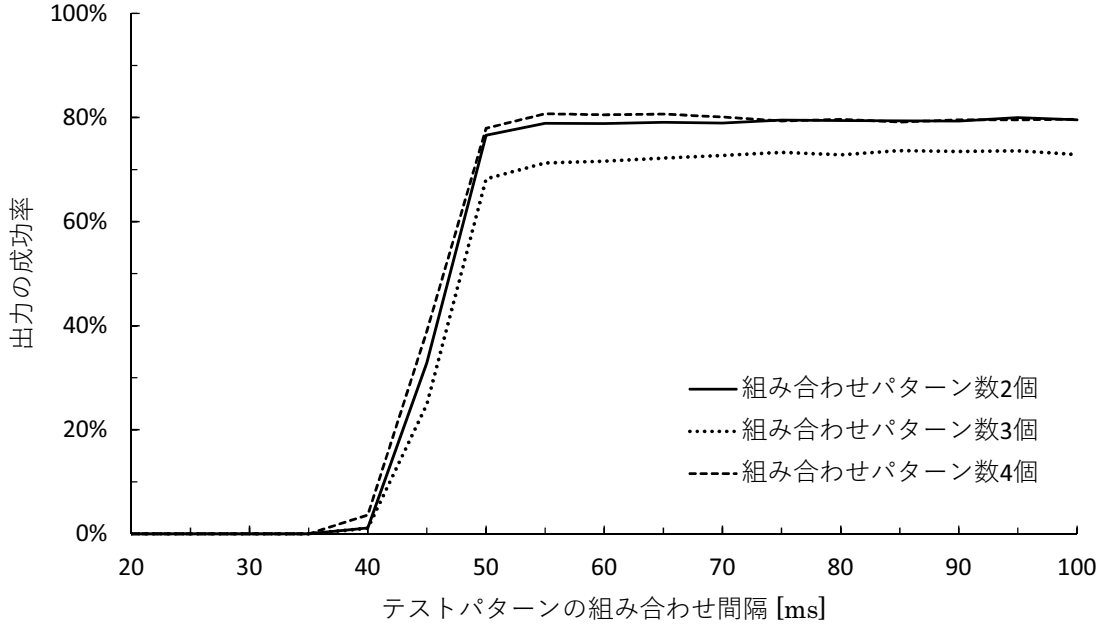

図 3.8: 学習時の各組み合わせパターン数ごとの組み合わせ間隔 50ms で学習したネットワークのテストパターンに対する出力の成功率

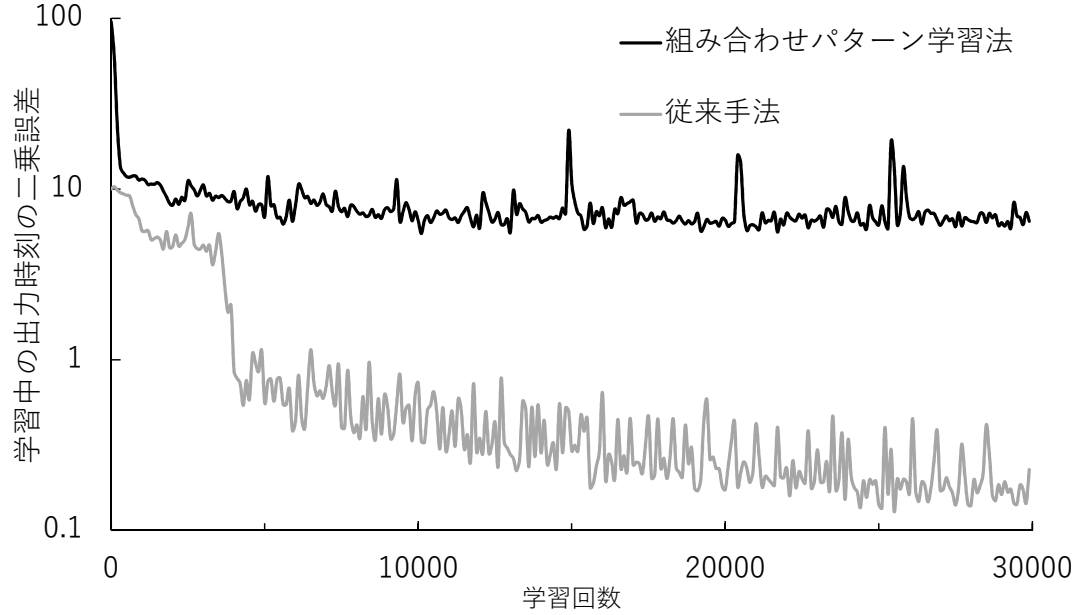

図 3.9: 組み合わせ間隔 25ms で 2 パターンを組み合わせパターン学習した際の学習中の誤差の推移の一例

3.3　2手法を併用する手法

本節では，3.1 節，3.2 節で述べた手法を併用する手法を提案する．

3.3.1　2手法の併用による応答の改善

これまで本稿で提案してきた応答関数を変更する手法および組み合わせパターン学習法の 2 手法は，それぞれ連続入力パターンに対する応答の改善にある程度有効であるものの，単一で適用した場合には別個の原因によりそれぞれ十分に応答を改善することはできなかった．

3.1 節で提案した応答関数を変更する手法では，応答関数によって先行入力パターンの影響が減衰しきる前にパターンが入力されると正常な出力が得られず，また，より応答関数の時間減衰を素早くすることは時系列情報パターンの処理能力の低下を招くため，小さい入力間隔での複数パターンの連続入力に対する応答を十分改善することはできなかった．3.2 節で提案した組み合わせパターン学習法は，学習時の組み合わせ間隔が小さい場合には学習が停滞してしまうため，小さい入力間隔での複数パターンの連続入力に対する応答を十分改善することはできなかった．

そこで，これら 2 手法を併用する手法を提案する．それぞれの手法は，独立した原因によって複数パターンの連続入力に対する応答を十分改善することはできない．応答関数を変更する手法で十分に改善することが出来ない要因は，先行入力パターンの影響が減衰しきれない場合に，わずかな影響であっても正常な出力が得られないためと考えられる．そのため，組み合わせパターン学習法によってそのわずかな先行入力パターンの影響を学習することで，さらなる応答の改善が期待できる．また，組み合わせパターン学習法で十分に改善することが出来ない要因は，先行入力パターンの影響が大きすぎる場合に，正常に学習ができないためと考えられる．そのため，応答関数の変更によって先行入力パターンの影響を軽減することで，正常に学習ができるようになり，さらなる応答の改善が期待できる．

このように，それぞれの手法の欠点を，別の手法によって補うことができるため，これらの手法を併用することで，連続入力に対する応答をより改善することが期待される．

3.3.2 実験条件

2 手法を併用する手法の複数パターンの連続入力に対する効果を確認するために実験を実施した．

実験には 3.1 節同様に時間版 Iris 問題を用いた．また，ネットワーク構造は τ_s を 0.5ms とし，その他のパラメータは 3.1 節同様に設定した．

学習時のパターン組み合わせ間隔を 25ms から 40ms まで 5ms 刻みに変更し，3.2 節同様に組み合わせパターン学習を実施した．学習後のネットワークは，3.1 節，3.2 節同様にテストパターンに対する応答で評価した．

3.3.3 結果と考察

それぞれの組み合わせ間隔で組み合わせパターン学習を実施した際の，テストパターンに対する出力の成功率を図 3.10〜図 3.13 に示す．図の縦軸が出力の平均成功率，横軸がテストパターンの組み合わせ間隔を表す．

まず，3.1 節で実施した応答関数の変更のみを適用した場合と比較する．図 3.2 の $\tau_s = 0.5$ms の場合の成功率と比較すると，2 手法の併用によって，学習時の組み合わせ間隔よりも大きい間隔で組み合わされたテストパターンに対する応答が改善している．例えば，学習時の組み合わせ間隔が 25ms の場合，組み合わせ間隔 25ms のテストパターンに対する出力の成功率が，応答関数の変更のみでは約 6.3%であったが，2 手法の併用では約 72.7%まで上昇している．また，大きい間隔で組み合わされたテストパターンに対する応答は，$\tau_s = 5.0$ms の場合と比べるとわずかに低下しているものの，$\tau_s = 0.5$ms の場合と比べるとほとんど差がない．

次に，3.2 節で実施した組み合わせパターン学習法のみを適用した場合と比較する．学習時の組み合わせ間隔 25ms（図 3.3）や学習時の組み合わせ間隔 30ms（図 3.4）といった，組み合わせパターン学習法のみを適用した際には成功率が低かった場合において，2 手法の併用によって成功率の上昇がみられた．例えば，学習時の組み合わせ間隔が 25ms の場合，組み合わせパターン学習法のみを適用した場合にはテストパターンに対する出力の成功率がテストパターンの組み合わせ間隔によらず 15%未満であった．しかし，2 手法を併用した場合には学習時の組み合わせ

間隔よりも大きい間隔で組み合わされたテストパターンに対して，70%以上の成功率が得られた．

これらの結果から，提案した応答関数の変更と組み合わせパターン学習法を併用する手法によって，大きい入力間隔での連続入力パターンに対する出力の成功率を大きく低下させることなく，小さい入力間隔での連続入力パターンに対する出力の成功率を大きく向上することができた．このことから，提案手法は複数パターンの連続入力に対する応答の改善に有効であるといえる．

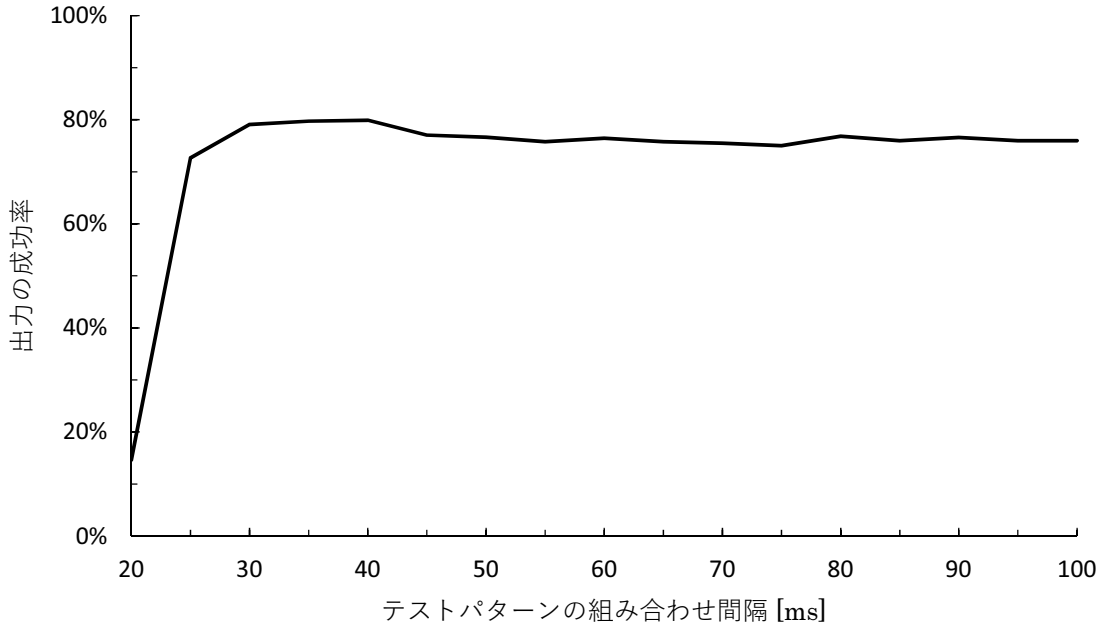

図 3.10: 2 手法を併用し，組み合わせ間隔 25ms で学習したネットワークのテストパターンに対する出力の成功率

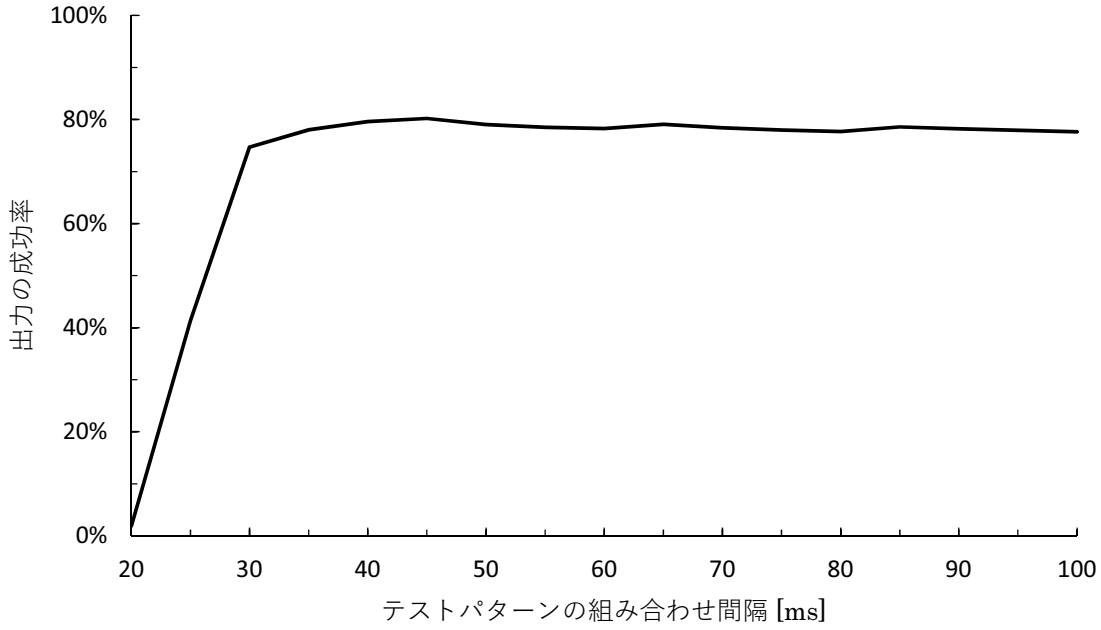

図 3.11: 2 手法を併用し，組み合わせ間隔 30ms で学習したネットワークのテストパターンに対する出力の成功率

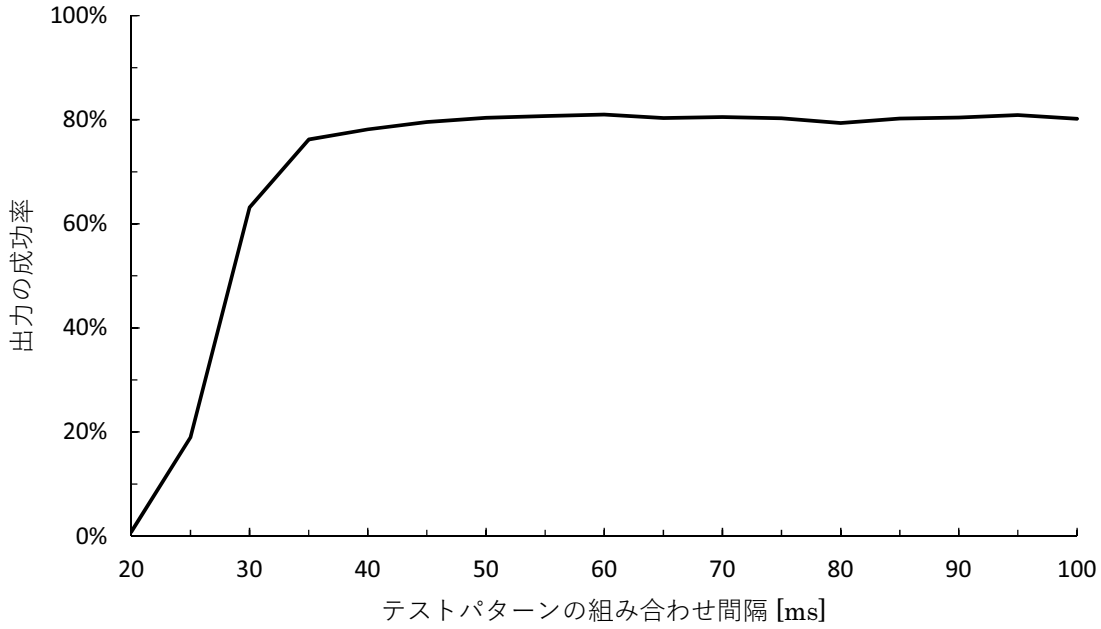

図 3.12: 2 手法を併用し，組み合わせ間隔 35ms で学習したネットワークのテストパターンに対する出力の成功率

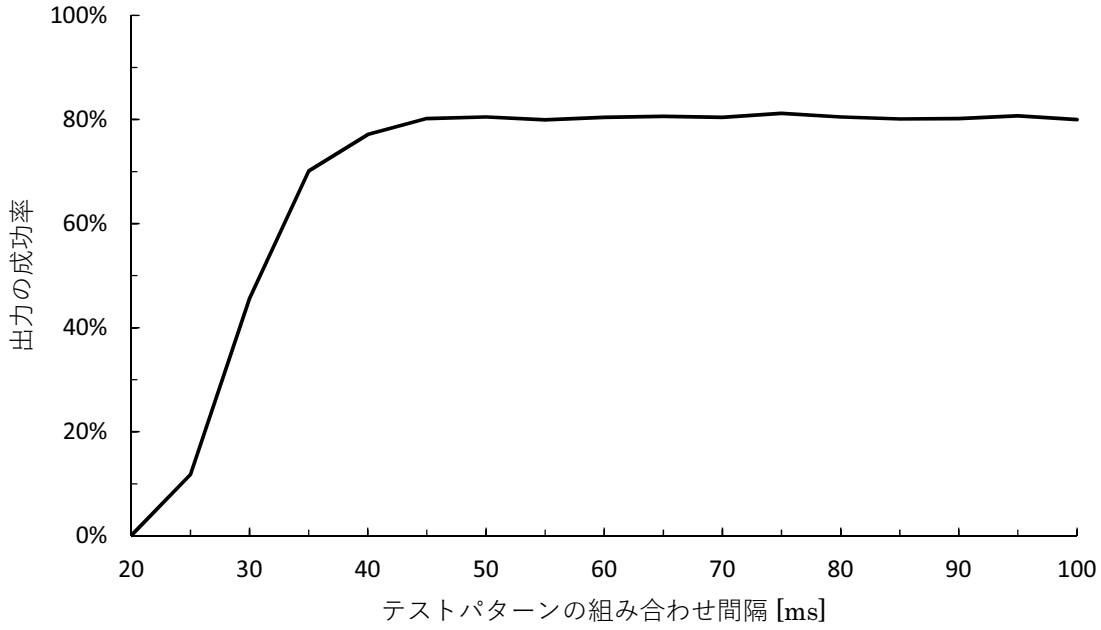

図 3.13: 2 手法を併用し，組み合わせ間隔 40ms で学習したネットワークのテストパターンに対する出力の成功率

第4章　結論

本研究では，スパイキングニューラルネットワークの一種である SpikeProp において，複数パターンの連続入力に対する応答を改善することを目的とし，ネットワークの構造と学習法について検討した．

SpikeProp は，松本らが提案した AWD 法を用いることで一組のスパイク列（パターン）がネットワークの初期状態から入力された際に，望みの一組のスパイク列を出力することができるが，初期状態からの単一パターンの入力を念頭に置いている．そのため，複数パターンが連続で入力された際には先行入力パターンによって生じた情報の影響で出力が変化し，望みの出力が得られないことがある．

そこで，ネットワークの各ユニットにスパイクが入力された際の応答を決定するスパイク応答関数を変更する手法を検討した．不要な先行入力パターンの影響を軽減するために，スパイク応答関数をより素早く時間減衰するものに変更する手法を提案した．ベンチマーク問題を用いた実験により，複数パターンの連続入力に対する応答の改善に有効であることを確認した．しかし，この方法を適用しても小さい間隔での連続入力に対する応答を十分改善することはできなかった．

また，先行入力パターンの影響がある状態でパターンが入力された際の出力を学習するために，複数パターンを一組のパターンとして組み合わせて学習させる，組み合わせパターン学習法を検討した．ベンチマーク問題を用いた実験により，複数パターンの連続入力に対する応答の改善に有効であることを確認した．しかし，この方法を適用しても小さい間隔での連続入力に対する応答を十分改善することはできなかった．

そこで，これら 2 手法の問題点を補うために，2 手法を併用する手法を提案した．ベンチマーク問題を用いた実験により，それぞれの手法を単独で用いた場合と比べ，併用することで複数パターンの連続入力に対する応答がより改善されることを確認した．

この2手法を併用する手法を適用することで，従来手法では望みの出力が得られなかった小さな間隔での連続入力に対し，応答を改善することができたといえる．

今後の課題としては，提案法を実際の時系列情報へと適用した際の影響についての議論と，パラメータのさらなる議論および自動調整法の検討がある．本研究では先行研究との比較，および非線形なパターン分類器としての性能を検証するために時間版 Iris 問題を用いたが，順序情報などの，実際の時系列情報に含まれる情報が重要な問題に対しての応答については十分に議論されたとはいえない．そのため，これについての詳細な議論が必要である．また，今回使用した応答関数について，よりデータ数が多い複雑な問題など，学習対象により適切な値が異なると考えられる．学習に用いるデータセットや，学習中のふるまいに合わせて自動で調整できれば，手間を省くことができる．

謝辞

本研究の遂行および本論文の作成にあたり，懇切丁寧なご指導と御督励を賜った本学工学研究科電気電子工学専攻の高瀬治彦教授，川中普晴准教授，北英彦准教授，本学理事・副学長の鶴岡信治教授に感謝いたします．貴重な時間を割いて本論文を査読していただいた教養教育機構の野呂雄一教授に深く感謝いたします．また，日頃熱心に討論していただいた計算機工学研究室，情報処理研究室の皆様方に厚く御礼申し上げます．最後に，本論文をまとめるにあたり，助言，討論，その他お世話になったすべての方々に感謝いたします．

参考文献

[1] David Silver and Demis Hassabis, Google DeepMind (2016), "Research Blog: AlphaGo: Mastering the ancient game of Go with Machine Learning," https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html, accessed: Dec./24/2017.

[2] W. S. McCulloch, and W. Pitts. "A logical calculus of the ideas immanent in nervous activity," The bulletin of mathematical biophysics, Vol. 5, No. 4, pp. 115–133, 1943.

[3] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," Psychological Review, Vol. 65, No. 6, pp. 386–408, 1958.

[4] M. Minsky, and S. Papert, "Perceptrons," MIT Press, 1969.

[5] K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," Biological Cybernetics, Vol. 193, pp. 193–202, 1980.

[6] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proceedings of the national academy of sciences, Vol. 79, No. 8, pp. 2554–2558, 1982.

[7] D. H. Ackley, G. E. Hinton, T. J. Sejnowski, "A learning algorithm for Boltzmann machines," Cognitive science, Elsevier, Vol. 9, No. 1, pp. 147–169, 1985.

[8] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors," Nature, Vol. 323, pp. 533–536, 1986.

[9] 山下裕, 島公脩, 石動善久, “ニューラルネットワークによる学習・適応制御,” 計測と制御, Vol. 30, No. 4, pp. 302–308, 1991.

[10] 堀川慎一, 古橋武, 内川嘉樹, “ファジィニューラルネットワークの構成法と学習法,” 日本ファジィ学会誌, Vol. 4, No. 5, pp. 906–928, 1992.

[11] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” Neural computation, Vol. 18, No. 7, pp. 1527–1554, 2006.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks”, In Advances in Neural Information Processing Systems 25, pp. 1097–1105, 2012.

[13] A. L. Hodgkin, and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” The Journal of physiology, Vol. 117, No. 4, pp. 500–544, 1952.

[14] E. M. Izhikevich, “Simple model of spiking neurons,” IEEE Transactions on Neural Networks, Vol. 14, No. 6, pp. 1569–1572, 2003.

[15] W. Gerstner, “Time structure of the activity in neural network models,” Physical Review E, Vol. 51, No. 1, pp. 738–758, 1995.

[16] W. Maass, and C. M. Bishop, “Pulsed Neural Network,” The MIT Press, 1998.

[17] F. Ponulak, and A. Kasiński, “Introduction to spiking neural networks: Information processing, learning and applications,” Acta neurobiologiae experimentalis, Vol. 71, No. 4, pp. 409–433, 2010.

[18] W. Maass, “Noisy Spiking Neurons with Temporal Coding have more Computational Power than Sigmoidal Neurons,” M. Mozer, M.I. Jordan, T. Petsche (Eds.), Advances in Neural Information Processing Systems, MIT Press, Vol. 9, pp. 211–217, 1997.

[19] 銅谷 賢治・ほか編, “脳の情報表現—ニューロン・ネットワーク・数理モデル—,” 朝倉書店, 2002.

[20] S. M. Bohte, “The evidence for neural information processing with precise spike-times: A survey,” Neural Computing, Vol. 3, No. 2, pp. 195–206, 2004.

[21] R. V. Florian, “The Chronotron: A Neuron That Learns to Fire Temporally Precise Spike Patterns,” PLoS ONE, Vol. 7, No. 8, e40233, 2012.

[22] F. Ponulak, and A. Kasiński, “Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting,” Neural Computation, Vol. 22, No. 2, pp. 467–510, 2010.

[23] A. Mohemmed, S. Schliebs, S. Matsuda, N. Kasabov, “SPAN: Spike Pattern Association Neuron for Learning Spatio-Temporal Spike Patterns,” International Journal of Neural Systems, Vol. 22, No. 4, 1250012, 2012.

[24] B. Gardner, and A. Grüning, “Supervised Learning in Spiking Neural Networks for Precise Temporal Encoding,” PLoS ONE, Vol. 11, No. 8, e0161335, 2016.

[25] S. M. Bohte, J. N. Kok, H. La Poutré, “Error-backpropagation in temporally encoded networks of spiking neurons,” Neurocomputing, Vol. 48, pp. 17–37, 2002.

[26] O. Booij, H. tat Nguyen, “A gradient descent rule for spiking neurons emitting multiple spikes,” Information Processing Letters, Vol. 95, pp. 552–558, 2005.

[27] 松本崇, 高瀬治彦, 川中普晴, 鶴岡信治, “スパイキングニューラルネットワークにおけるスパイク時刻と数の学習法,” 電子情報通信学会論文誌, Vol. J99-D, No. 7, pp. 662–668, 2016.

[28] S. B. Shrestha, and Q. Song, “Adaptive learning rate of SpikeProp based on weight convergence analysis,” Neural Networks, Vol. 63, pp. 185–198, 2015.

[29] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the SpikeProp algorithm,"Proceedings of the 2006 IEEE International Joint Conference on Neural Network, pp. 3970–3977, 2006.

[30] J. Xin, and M. J. Embrechts, "Supervised learning with spiking neural networks," Proceedings of the 2001 IEEE International Joint Conference on Neural Network, pp. 1772–1777, 2001.

[31] S. J. Hanson, and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," Advances in neural information processing systems 1, pp.177–185, 1989.

[32] K. Kawanishi, H. Takase, H. Kawanaka, S. Tsuruoka, "Reduce the Computing Time for SpikeProp by Approximation of Spike Response Function," Procedia Computer Science, vol.96, pp. 1186–1192, 2016.

[33] M. Lichman, "UCI Machine Learning Repository," Irvine, CA: University of California, School of Information and Computer Science, http://archive.ics.uci.edu/ml, accessed: Aug./1/2017.

研究業績

[A01] 小野田憲悟, 高瀬治彦, 川中普晴, 鶴岡信治, “複数パターンの逐次入力に対応できる SpikeProp の学習法に関する一検討,” 平成 29 年度　電気・電子・情報関係学会東海支部連合大会予稿集, pp. H4–H5, 2017.

[A02] Kengo Onoda, Haruhiko Takase, Hiroharu Kawanaka, Shinji Tsuruoka, “Training Algorithm of SpikeProp Accepting Sequential Patterns: A Discussion on Effective Combined Teacher Patterns,” Proceedings of the 9th International Workshop on Regional Innovation Studies (IWRIS2017), pp. 38–41, 2017.

[A03] 小野田憲悟, 澤村将周, 高瀬治彦, 川中普晴, 鶴岡信治, “時系列情報を処理するためのニューラルネットワーク —SpikeProp におけるパラメータが学習に及ぼす影響の調査—,” 地域イノベーション学会誌, Volume 6, 2017, p. 34, 2017.

[A04] 小野田憲悟, 高瀬治彦, 川中普晴, “SpikeProp における連続入力パターンの処理に適した応答関数に関する一検討,” 第 44 回東海ファジィ研究会予稿集, pp. P3-10 1–P3-10 4, 2018.

[A05] 小野田憲悟, 高瀬治彦, 北英彦, 川中普晴, “SpikeProp における連続入力パターンに対する出力の改善 —学習データ量と学習性能との関係性についての一検討—,” 第 45 回東海ファジィ研究会予稿集, pp. 9-1–9-4, 2018.

[A06] 小野田憲悟, 高瀬治彦, 北英彦, 川中普晴, “SpikeProp における連続入力パターンに対する出力の改善 —学習法および応答関数の見直し—,” 第 34 回ファジィシステムシンポジウム講演論文集, pp. 700–705, 2018.

[A07] Kengo Onoda, Haruhiko Takase, Hidehiko Kita, Hiroharu Kawanaka, “Improve Response to Successive Input Patterns in SpikeProp,” Proceedings of the 8th International Symposium for Sustainability by Engineering at Mie University (Research Area C) (IS^{2}EMU2018-C), pp. 21–22, 2018.

[A08] Kengo Onoda, Haruhiko Takase, Hidehiko Kita, Hiroharu Kawanaka, “Improve Response to Successive Input Patterns in SpikeProp: A Discussion on Training Performance by the Amount of Training Data,” Proceedings of the 10th International Workshop on Regional Innovation Studies (IWRIS2018), pp. 83–86, 2018. (Excellent Paper Award 受賞)

[A09] Kengo Onoda, Haruhiko Takase, Hidehiko Kita, Hiroharu Kawanaka, “A discussion on Improve Response for Successive Input Patterns in SpikeProp,” Proceedings of Intelligent Systems Workshop 2018 (ISWS2018), pp. 621–625, 2018.

[A10] 小野田憲悟, 高瀬治彦, 北英彦, 川中普晴, “SpikeProp における応答関数と学習法の見直しによる連続入力パターンに対する応答の改善,” 知能と情報, Vol.31, No.1, pp. 613–616, 2019.

[A11] 小野田憲悟, 滝川裕磨, 高瀬治彦, “SpikeProp における実時系列データの学習の試み,” 第 46 回東海ファジィ研究会予稿集 , pp. P3-09 1–P3-09 4, 2019.