平成２１年度

修士論文

自己連想記憶理論を用いた
情報自動構造化アルゴリズムの構築

Building of an Automatic Information Structuring Algorithm
with Memory Prediction Framework

平成２２年３月

指導教員　　松尾 豊

東京大学大学院工学系研究科技術経営戦略学専攻

松尾研究室　０８６９１９　牧野 晃典

目次

図表目次

概要

大量のデータを構造化する必要が生じている．その手段のひとつとして，俯瞰マップが用いられている．俯瞰マップはノードをクラスタリングすることでデータを抽象化し，人に予測性を与える．これを本質的に改良するためには，予測性の向上を目的関数としたデータの抽象化を行うアルゴリズムが必要である．

本研究ではニューラルネットワークに着目し，最新の Hawkins や Edelman のアイデアに基づき補助問題を生成することによって，データを構造化する手法を提案する．データの抽象化を行う仕組みを選択淘汰，適応，多層化，時間方向の圧縮という四つの要素に分解し，それぞれの効果を確認する．さらに，生体的な制約に基づく統合的アルゴリズムの構築を試みる．本アルゴリズムが，論文，特許，企業情報，気象情報などのデータの構造化に将来的にいかに寄与するかを議論する．

本研究は，直接的に大量データの構造化には至っていないものの，大量データを自動的に構造化するための基盤となるアルゴリズムを提案しており，潜在的な有用性は極めて高いと考えている．

第１章 序論

この章では，本研究の背景と目的を明らかにし，立脚する前提と問題意識の所在を示す．また章末には，本論文の全体の構成を示す．

第１節 背景

有史以来，人類による知識の生産効率は，文明や技術の発展に支えられて，時の流れとともに向上しつづけてきた．しかし，近年の急速なウェブの発展に伴う増加ほど爆発的な傾きを示したことはない．あらゆる情報の絶対量は今や線形ではなく，幾何級数的に増えつづけている．

このことは情報収集の便宜と同時に，情報把握の困難を生み出した．我々は必要とする情報を，関連した情報も含め好きなだけ集めることはできるが，その意味の離散した莫大な情報群を理解可能な形に編集する術を欠いており，それゆえに集めた情報を充分有効には利用できていない．そんな中，ウェブ上では二次情報，三次情報が再生産され，情報量の増加は加速度的に進行していく．なんらかの手段を講じない限り，情報は我々にとって日を追うごとに把握困難なものになっていく．

一方で，情報へのアクセスが容易になればなるほど，知識を持っていることそれ自体は差別化された力たり得なくなり，それをどのように用いるかという知識，すなわち「知を使う知」が重要視されるようになった．ことに技術や経営の領域ではその傾向が顕著である．

この両傾向を併せ考えて導かれる結論は，知を使う知が必要であり，その使うべき知が人間の手に余るほど巨大になった今，人が直接膨大な知の海に直面する以前に，それをパースする機械に知能が求められているということである．その機械こそ，あらゆる情報を自動的に構造化できる人工知能に他ならない．

上述の問題は学問分野においても早くから顕在化していた．1963 年には既に Price が，学問分野の細分化に伴って論文数が指数関数的に増加していくことを指摘し，これに警鐘を鳴らしている[1]．そうして 2009 年現在，Price の予言通り，多くの学問領域で論文数は指数関数的に増加している．ひとつの学問領域の論文をすべて読むことはもはや一生かけても不可能な仕事となり，どの論文を選んで読むべきかという選択が問題となっている．

こうした事情を踏まえて，東京大学俯瞰工学部門では，ひとつの広大な学問領域における重要な論点や全体の方向性を容易に把握するための手段として，学術俯瞰マップの研究を行ってきた．

学術俯瞰マップは，関連論文を引用分析の手法により自動的にクラスター化することで生成されるネットワーク図である．マップ生成に用いた各論文は，引用関係のリンクとともにマップ上に図示される．この作業は計算機によって自動的に行われるものであり，いかなる学問領域のマップの作成にも人力や専門家を要しない．これによって，急速に膨張している学術研究をいつでも客観的に俯瞰することができるようになった．

学術俯瞰マップの一例を示す．図 1 はイノベーションに関する学術俯瞰マップである．

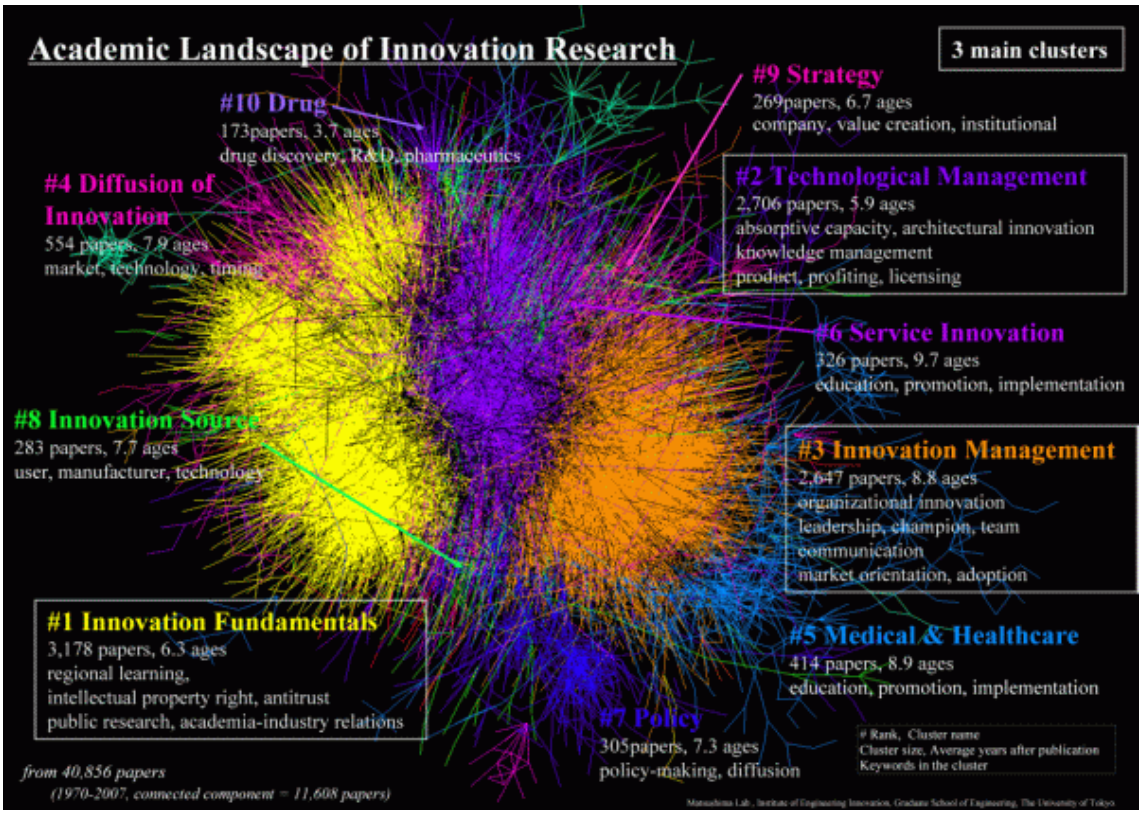


図 1　　イノベーションに関する学術俯瞰マップ

これを見れば，人はイノベーションに関連する研究領域について，「＃1 イノベーション創成の基盤」「＃2 技術革新」「＃3 イノベーションマネジメント」の三分野が代表的な研究領域であることを即座に読み取ることができる．また，自分が情報を得たいと思うクラスター周辺や，サブクラスターに位置する論文を集中的に読むことで，効率的かつ体系的に情報収集を行うことができる．

学術俯瞰マップは，これまでに様々な産業研究や学術研究で利用されており，いずれの領域においても重要な研究ツールとして機能している．梶川ら（2007）の「Structure of research on biomass and bio-fuels: A citation-based approach」や，大野（2006）の「Sustainability に関する学問俯瞰マップの作成[2]」などは，近年の代表的な研究例である．

大量の情報を構造化したいという需要は日増しに大きい．このような事情を背景として，本研究は次節に示す目的の達成を目指すものである．

第２節 研究目的

本研究の最終的な目的は，上述の学術俯瞰マップをより汎用的で有用な形に進化させることである．そのため，学術俯瞰マップが人にとって有用なツールである理由を考察し，その有用性の原理を一般的なアルゴリズムとして抽出することが必要である．

我々が俯瞰マップによって得ることのできる一般的な知見として，次のようなものが挙げられる．

領域間の関係性

色分けされた大小様々なクラスターのネットワーク上における位置関係から，領域同士の近接性や関係性を知ることが出来る．

下位領域の存在

各クラスターには内部にサブクラスターが形成されており，それによって下位領域のの存在と，領域の上位－下位の関係を知ることができる．

ハブの存在

クラスターとクラスターの中間部に位置し，どちらに対してもリンクを持っているような論文は，両クラスターのハブとして機能していることがわかる．そのような，異なる領域間を橋渡ししている点で重要な存在を発見することができる．

学術俯瞰マップがこれらの特性を持つのは，論文をノード，その引用・被引用関係をエッジとして，ネットワークという形式に情報を変換することで，論文というデータの抽象度を上げ，クラスターとして概念化しているからである．ネットワークには情報を

概念化して示すという特性がある．情報の構造化とは，扱う情報を正しくネットワーク化することに他ならない．

しかし，学術俯瞰マップで扱う論文というデータは，引用・被引用関係も含め，形式的に整理されたものであるが，我々が通常必要としている情報の多くは，論文のように形式的に整理されたものではない．ウェブ上に存在する情報などはその典型である．第１節で述べた現状を考慮すれば，論文の構造化のみならずあらゆる領域について，学術俯瞰マップに比する情報構造化ツールが必要である．すなわち，いかなる形式であれ何らかのデータが与えられれば，それを自動的に抽象化して概念をつくりだし，その概念間に関係を見出すようなシステムである．これは対象データの形式に依らない原理的なアルゴリズムを要するものであり，そのアルゴリズム発見のためには，情報の構造化という手法に関する基礎的な研究が必須であろう．

本研究は，知の構造化及びネットワークに関する従来研究を踏まえた上で，情報構造化のためのより深いアルゴリズムの方式を探る．その目的は俯瞰マップの汎化及びその手法の昇華である．

直接的に俯瞰マップの性能を向上するなどの貢献まで辿り着かなくとも，現在俯瞰マップが果たしている機能を分解し，その原理を解明して深めていくことは，求められている大量の情報の構造化と，それに基づく分析及び意思決定のための重要な基盤となる．

第３節 論文の構成

本論文は全８章から成る．本章では，情報構造化の必要性と本研究の目的を述べた．第２章では，本研究の中心的手法であるニューラルネットワークについて解説し，本研究に関連の深い既存研究について言及する．第３章では，本研究の提案アルゴリズムのアイデアとなっている既存研究について詳しく述べ，それら従来研究と本研究との関係を示す．第４章では，本研究の動機と目的についてより詳細に述べ，提案アルゴリズムの意義を示した上で，本研究の目的の定式化を行う．第５章では，本研究の提案アルゴリズムを実現するために重要であるデータの抽象化を行う仕組みを選択淘汰，適応，多層化，時間方向の圧縮という四つの要素に分解し，それらの要素を取り込んだアルゴリズムを設計する．また，いくつかの実験とその結果分析により，各要素の効果を確認する．第６章では，第５章の分析で得られた知見を，生物的制約条件の下で実現するアルゴリズムを考え，プログラムとして実装し，その動作確認を行う．第７章では，本アルゴリズムが論文，特許，企業情報，気象情報などのデータの構造化に将来的にいかに寄与するかを議論する．第８章では，本研究の結論を示す．

第２章 ニューラルネットワークの概説

この章では,本研究の中心的手法であるニューラルネットワークについて基礎を説明し，関連研究について紹介する．その中でも特に，本研究に関係の深い既存モデルについて，その詳細なアルゴリズムと意義を解説する．

第１節 概要

本章では，ニューラルネットワークについて基本的な情報を概観する．本研究の提案アルゴリズムはニューラルネットワークの形で実装されるものであり，本章で説明する知識がその基本となる．

第２節及び第３節ではニューラルネットワークの定義と，その手法の特徴を述べる．第４節では，ニューラルネットワークの基本要素である素子（ユニット）と識別関数について説明し，その基本的な動作を示す．第５節では，多層型ニューラルネットワークのもっとも一般的なものであるパーセプトロンについて説明する．パーセプトロンは，本研究の第５章で設計するアルゴリズムの前半部で主に用いるネットワークである．第６節では，多層パーセプトロンの学習方法であるバックプロパゲーション法について説明する．バックプロパゲーション法は，本研究の第５章第２節で行うアルゴリズムの設計・分析の一節で，数字認識の学習手法として用いる．第７節では，相互結合型ネットワークのもっとも一般的なものであるホップフィールドネットワークについて説明する．相互結合型ネットワークは，本研究の第５章で設計するアルゴリズムの後半部で主に用いる．第８節では，ループ構造を持つネットワークの挙動にダイナミクスを持たせたモデルである，リカレントニューラルネットワークについて説明する．ループ構造を持つネットワークは，本研究の第５章第５節で扱う．第９節では，ニューラルネットワークの枠組みからは外れるが，ネットワーク型の知識表現として代表的であるベイジアンネットワークについて，その概要を述べる．本研究の提案アルゴリズムも，ネットワークという形態を取ることで人に予測を可能にさせる，知識表現のひとつである．

本研究により深く関連する，もっとも新しい既存研究については，本章では扱わず，次章の第３章で関連研究として述べる．

第２節 ニューラルネットワークとは

ニューラルネットワークとは，生物の脳の神経回路網を模倣した計算メカニズムの総称である．シナプスの結合によりネットワークを形成したニューロンが，学習によってシナプスの結合強度を変化させ，問題解決能力を持つようなモデル全般を指す．比較的小さな計算量で良好な解を得られるため，データマイニングなどの領域では一般的な手法となっている．

ニューラルネットワークの研究は，1943 年に McCulloch-Pitts が考案したニューロンモデルにはじまり，1957 年の Rosenblatt によるパーセプトロンの研究により活性化したが，1969 年の MinskyPapert によるパーセプトロンの限界論により一時沈滞した．その後，1986 年に Rumelhart が従来の問題点を解決するバックプロパゲーションアルゴリズムを考案したことにより，ニューラルネットワークの研究は再び活性化した．1982 年には Hopfield により相互結合型のニューラルネットワークも考案され，以来ニューラルネットワークの様々な問題への適用に関心が集まっている[3]．

ニューラルネットワークのモデルにはさまざまな構造を持つものがあるが，大きくわけて，前の層から次の層への一方向のみ信号の伝播が行われる階層型ネットワークと，層を持たず各ニューロンが相互に結合している相互結合型ネットワークのふたつがある．それぞれについて代表的なモデルの詳細を，次節以降で後述する．

第３節 ニューラルネットワークの特徴

ニューラルネットワークの主な特徴には，以下の 3 つがある［4］.

非線形性

従来の信号処理では，理論的に体系化されている線形手法が広くもちいられてきたが，一方で非線形信号処理は理論的に体系化されていなかった．そのために処理するデータの特性に合わせて，理論やパラメータに場当たり的な工夫を施す必要があった．ニューラルネットワークは，初めて理論的に体系化され，利用されるようになった非線形近似手法である．

学習能力

ニューラルネットワークは，提示される例（訓練データ）に基づき自身の構造を逐次更新していくことができる．学習能力を有するシステムにおいては，人為的に機構を設計する必要がないため，データに対する汎用性が高くなる．

並列性

生体の神経細胞は，トランジスタに比べて数桁計算速度が遅いにも関わらず，脳は計算機が数年かけても解くことのできない問題を一瞬にして解くことができる．その理由は，脳の並列計算機能にあると言われている．ニューラルネットはその並列計算機の実現形態の１つである．他の超並列計算の手法に比べて構造が一様かつ単純であり，用途が限定されているが，並列動作を体系的に把握する理論体系が確立されている点で優れている．

第４節 閾素子と識別関数

第１項 閾素子

ニューラルネットワークでは，生体のニューロンを図 2 のようにモデル化したものを，ネットワークの素子として扱う．

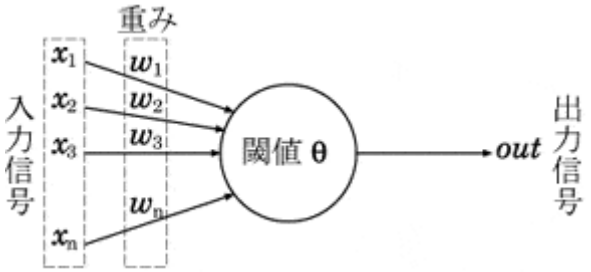


図 2　　ニューロンのモデル

出力信号を階段関数としたもっとも単純なニューロンモデルを，閾素子という．細胞(素子)への入力ベクトルを $x = (x_1, \ldots, x_N)$ とする時，１つの閾素子の動作は，以下の式で表される．

$$y = H\left[\sum_{i=1}^{N} w_i x_i - \theta\right]$$

w_i は入力要素 x_i にかかる結合の重み，θ は素子の閾値，$H[]$ は閾関数（ステップ関数）である．ステップ関数は識別関数のひとつで，括弧内の値が正の時には 1，負の時には 0 の値をとる．

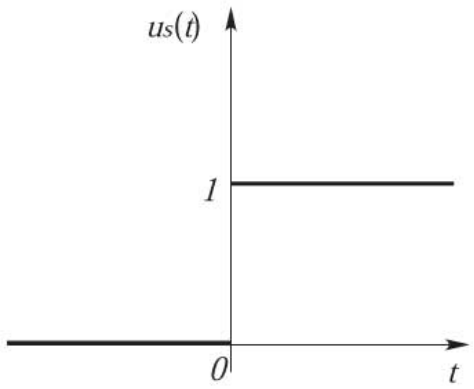


図 3　　ステップ関数

従って，ステップ関数を識別関数に持つ閾素子は，入力 x_i を w_i で重みをつけて足し合わせた和の値が，閾値 θ より小さい場合は 0 を出力し， θ を超えると 1 を出力する素子となる．このようなニューロンモデルはまた，その計算の可能性を理論化した McCulloch と Pitts の名をもとに，McCulloch-Pitts 型のニューロンモデルとも呼ばれる．

第２項 シグモイド関数

シグモイド関数は，以下の式で表される単調増加の関数であり，ロジスティック関数とも呼ばれる．

$$sigmoid(s) = \frac{1}{1 + e^{-\alpha s}}$$

閾値が θ である場合には，この関数を右に平行移動して，$y = sigmoid(s - \theta)$ として素子の動作を記述すればよい．

また関数中のパラメータ α をゲインと呼ぶ．ゲインの大小により，シグモイド関数は図のように形を変える．

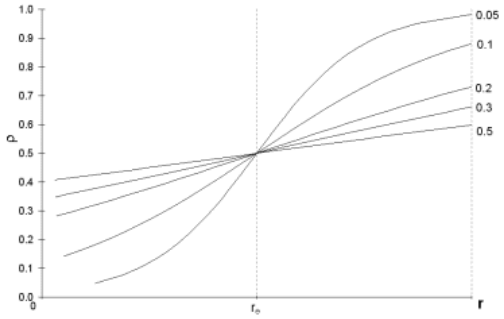


図 4　　シグモイド関数とゲイン α

第５節 パーセプトロン

パーセプトロンは，代表的な階層型ニューラルネットワークのひとつである．1958年に Rosenblatt によって提案された．ニューロンの発火に関連したシナプスの伝達効率は増加するというヘブの学習則をニューラルネットワークモデルに組み込んだもので，モデル内ではそれを結合荷重の値を大きくすることで表現している．パーセプトロンには，入力されたデータを識別した後、その出力が正解か誤りかによって，シナプスの伝達効率を変化させる学習機能もある．

もっとも基本的なパーセプトロンは前節の閾素子を階層構造状に並べたもので，図 5 のようなネットワーク構造を持つ．

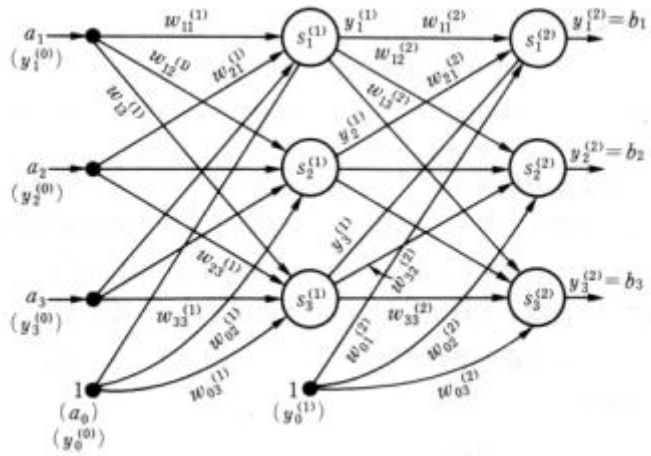


図 5　　パーセプトロン

第１層は入力特徴量を受け取ってそのまま出力するだけの層であり，入力層と呼ばれる．第２層から最終層の１つ前の層までは中間層または隠れ層と呼ばれ，最終層は出力層と呼ばれる．

もっとも基本的なパーセプトロンでは，中間層の結合荷重は更新出来ないため，XOR などの非線形分離問題を解くことはできない．しかし閾関数の代わりに，シグモイド関数を用いて出力を計算する連続値のモデルを用いることで，ネットワーク全体の入出力関係を連続で微分可能なものとし，基本的なパーセプトロンの学習アルゴリズムでは不可能だった中間層の素子の結合の重みの修正が行えるようになり，最適な結合の重みの探索に，最急降下法などの連続関数の最適化技法を用いることが可能になる．また，閾関数では切捨てであった，閾値以下の数値も他の素子に影響を与えられるようになる［5］［4］.

次節では，結合荷重探索の代表的な手法であるバックプロパゲーション法について解説する．

第６節 バックプロパゲーション

第１項 概要

バックプロパゲーション法（誤差逆伝播法）は，ニューラルネットワークを訓練するために用いられる代表的な教師あり学習手法のひとつである．結合荷重更新のために，出力ニューロンの誤差を後方ニューロンに伝播していくことから誤差逆伝播と呼ばれる．

バックプロパゲーションはネットワーク上の変更可能な重みについて，誤差の傾斜を計算するものである．通常すばやく収束して、対象ネットワークの誤差のローカルミニマムを探し出す．

あらかじめ決まった入出力関係を持つデータを用いて，それらを正しく判別するように誤差を逆伝播させて，ニューラルネット内部の結線重みと閾値のパラメータを変化させていくため，訓練されていない，すなわち入出力関係に用いられず学習させられなかったデータについては，必ずしも正しく判別されるとはいえない．また，バックプロパゲーションを行うためには，ネットワークは最低三層以上の構造を持つ必要がある．

以下にバックプロパゲーション法（図 6）と誤差逆伝播法（図 7）の概略図を示す．

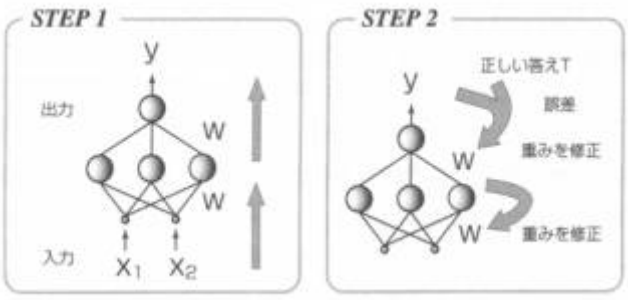


図 6　バックプロパゲーション概略図

バックプロパゲーション法は，層構造を下から上へ計算していくフォワード演算と，その逆向きに計算するバックワード演算の２ステップから成る．フォワード演算が通常のパーセプトロンにおける出力層の値計算に対応し，バックワード演算が誤差の逆伝播である．計算の詳細は，後述のアルゴリズムの項で述べる．

また，図 7 では，ニューラルネットの左側が入力層であり，右側が出力層である．

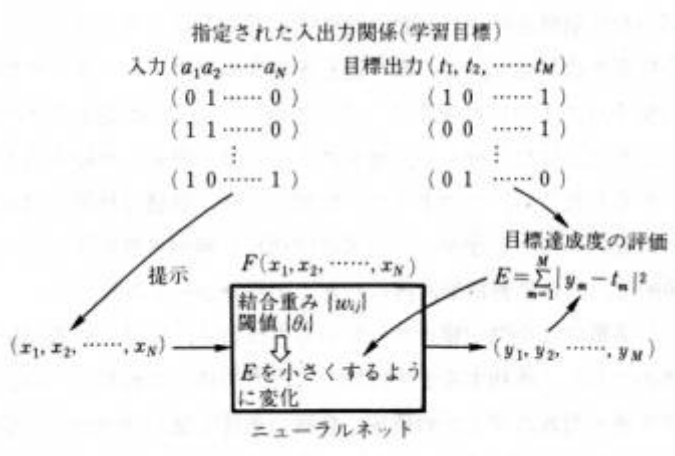


図 7　　誤差逆伝播法概略図

第２項 最急降下法

バックプロパゲーションにおける結合荷重の更新には，最急降下法を用いる．

関数 $f(x_1, \ldots, x_N)$ に対し，$f(x)$ を最小化または最大化するような $x = [x_1, \ldots, x_N]$ を求める際に最急方向へ下っていく方法を，最急降下法という．ただし，このとき $f(x)$ は必要な回数だけ微分出来る関数である必要がある．そのためステップ関数の代替に利用される関数として，シグモイド関数(第４節第２項)が代表的である．

たとえば最小化が目的の場合，パラメータ x_i は以下の数式にしたがって更新される．

$$x_i(t+1) = x_i(t) - \eta \frac{\partial f(t)}{\partial x_i}$$

η は正の値であり，パラメータ更新幅の係数である．η が小さくなるほど，修正幅は小刻みになり，学習には計算時間がかかるようになる．

最急降下法はローカルミニマムに陥りやすく，パラメータの初期条件やモーメンタム係数の導入など，最適化には工夫を用するが，ニューラルネットワーク関連の学習則のように，誤差関数やエネルギー関数を最小化する問題では，簡易なパラメータ逐次更新の手法として常套である．

第３項 アルゴリズム

バックプロパゲーション法では，「フォワード演算（前向き演算)」「バックワード演算（後ろ向き演算)」「結合荷重の更新」の３フェーズからなる演算を行う．そのアルゴリズムの概略図と手順を以下に示す．

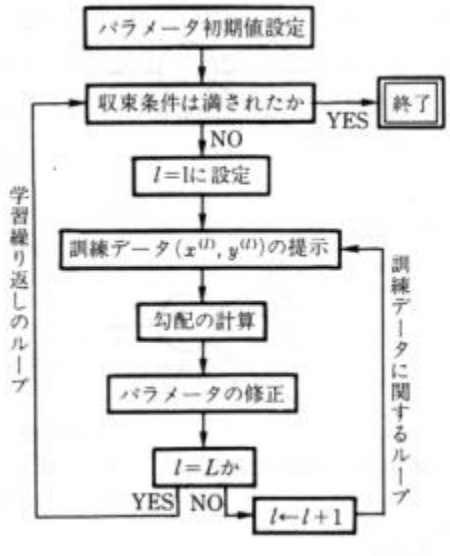


図 8　アルゴリズム概略図

フォワード演算

(1)入力ベクトル $x = (x_1, \ldots, x_N)$ をネットワークの入力層にセットし，各層のユニットの出力 y を計算してこれを保存する．

(2)出力層の各ユニットo_jにおいて，出力 y_j，教師信号 t_j から，誤差 $t_j - y_j$ を算出し，これにシグモイド関数の微分を乗じて，$\delta_i = sigmoid'(x)(y_j - t_j) = \alpha(1-y)(y-t)$ を算出する．

バックワード演算

(3) δ_i をひとつ前の層の各ユニットに重み付きで伝播させる．伝播されるのは，個々の結合荷重に δ_i をかけた値である．

(4)ユニットごとに逆伝播された δ の総和を取り，これを出力層に対する誤差と同じ扱いとして，再びそのユニットの δ を算出する．この操作を繰り返し，ネットワーク内のすべてのユニットに対して δ を求める．

結合荷重の更新

(5)各ユニットの重みベクトル w を，ユニットの δ と，そのユニットへの入力ベクトル x を用いて，$x_i(t+1) = x_i(t) - \eta\delta x$ のように更新する．

なお，(2)においてシグモイド関数の微分が $\alpha y(1-y)$ で表せることは，次のように導出できる．

$$y = sigmoid(s)$$

$$sigmoid(s) = \frac{1}{1+e^{-\alpha s}}$$

$$sigmoid'(s) = \frac{\alpha e^{-\alpha s}}{(1+e^{-\alpha s})^2} = \alpha \frac{1}{1+e^{-\alpha s}}\left(1 - \frac{1}{1+e^{-\alpha s}}\right)$$

$$= \alpha sigmoid(s)\big(1 - sigmoid(s)\big) = \boldsymbol{\alpha y(1-y)}$$

第４項 バックプロパゲーション法の特徴

誤差逆伝播を用いたニューラルネットワークの学習は，近傍点探索法などに比べて収束時間が長く，ローカルミニマムに陥りやすいという短所がある．一方で雑音耐性は高く，ノイズを含むデータに対しても精度の高い予測を得ることができる，かつ高次元のデータに対してもその一般化能力が失われないなどの長所がある．

バックプロパゲーション法を用いないパーセプトロンとの重要な差は，中間層の結合荷重を更新できることによって，XOR のような非線形分離問題が解けるようになっていることである．構造をより多層化し，中間層を適宜組み合わせることによって，いくらでも複雑な特徴空間を分類することができるようにになる．

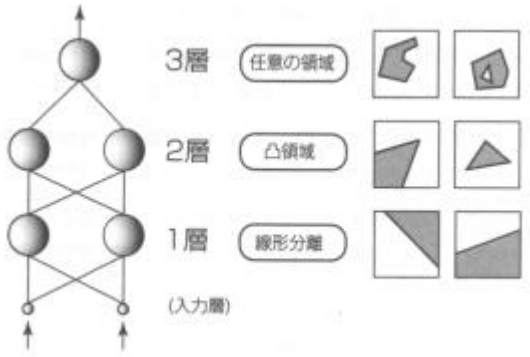


図 9　　多層パーセプトロンによる複雑な特徴空間の分類

第７節 ホップフィールドネットワーク

相互結合型ニューラルネットワークの代表的なモデルが，ホップフィールドネットワークである．1982 年に Hopfield によって提唱された．

各ユニットは McCulloch-Pitts 型の入出力特性を持ち，自分自身を除くネットワーク内の他のすべてのユニットと結合している．結合荷重の値は対象であり，すべてのユニット i, j の組について，$w_{ij}(t) = w_{ji}(t)$ が成立する．$i=j$ のとき，$w_{ij}(t) = 0$ となる．

ネットワーク全体のエネルギーを，次のように定義する．

$$E(t) = -\frac{1}{2}\sum_{i \neq j} w_{ji} x_i(t) - \sum_i \theta_i(t) x_i(t)$$

このエネルギー関数は，ネットワークの状態が変化するたびに必ず減少する．したがって，すべてのユニットに初期値を与えた後，この関数が変化しなくなるまで入出力を繰り返すことで，ネットワークの状態は必ず安定する．

ホップフィールドネットワークにおける結合荷重は，外部入力のセット S から一意に定めることができる．結合荷重の値は次の式にしたがって決定する．

$$W_{ij} = \begin{cases} \sum_{S=0}^{M-1} X_i^S X_j^S \ (i \neq j) \\ 0 \ (i = j) \end{cases}$$

こうして結合荷重を設定することで，入力パターンはネットワークに記憶される．これを連想記憶という．記憶したパターンの想起には，次の式を用いる．

$$\mu_i(t+1) = f\left[\sum_{i=0}^{N-1} w_{ji}\mu_i(t)\right] \ (0 \leq j \leq M-1)$$

ホップフィールドネットは，あらかじめ学習させたいパターンをネットワークに与えてニューロン間の結合荷重を変更し，想起時にはその結合荷重を用いて想起を行なう，学習過程と想起過程が分離されたモデルである．

また上述の二値を扱うホップフィールドネットワークの他に，0 から 1 の実数値を出力とする連続値ホップフィールドネットワークも，正しくエネルギー関数を定めることで，二値と同様，正しく動作することが，1984 年に Hopfield によって発表されている．

第８節 リカレントニューラルネットワーク

リカレントネットワークは，入力から出力へという階層型の一方向的な演算に加え，一部の出力から入力への再入力を行い，各ニューロンに時間遅れや積分などを持たせることで，ループ構造を持つネットワークの挙動にダイナミクスを持たせたモデルである．

離散時間では以下のように定式化される．

$$y_i(t+1) = g\left[\sum_{j=1}^{n} w_{ji} y_i(t) + I_i(t)\right]$$

ネットワークは初期値と結合荷重の設定に応じて，平衡状態や振動状態，カオス振動など様々な状態を取りうる．それらのダイナミクスは，連想記憶，意思決定，運動パタ

ーン生成などのモデルとして使われている[6].

本研究で提案するアルゴリズムはこれらの特性を持つもので, 後述するスパイクニューロンモデルをベースにしつつも, その原型はリカレントニューラルネットワークに分類することができる.

第９節 ベイジアンネットワーク

ニューラルネットワークではないが, ネットワーク形式での知識表現の代表的なものとして，ベイジアンネットワークがある.

ベイジアンネットワークとは, 複数の変数間の依存関係の有無を線の有無に対応させグラフ構造として表現したグラフィカルモデルのうち，特に線を矢印で書き，接続元変数で条件付けた接続先変数の条件付き確率という意味を持たせたものを言う. 条件付独立な命題の集合を表現するグラフであり, ネットワーク内にサイクル構造を持たない有向グラフである（図 10）.

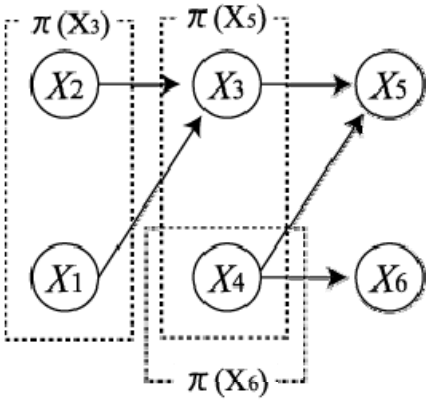


図 10　ベイジアンネットワーク

図 10 において，各ノードはそれぞれ確率変数で表されている．またノード間の矢印は因果関係を表し，その因果関係は条件付き確率によって定量化されている．X_1, X_2は親となるノードを持たないので，それらの確率変数によって定まる確率$P(X_1), P(X_2)$が付随する．これを事前確率という．対してX_3はX_1, X_2を親ノードに持つため，条件付き確率$P(X_3|X_1, X_2)$が付随する．例えば$P(X_3 = 0|X_1 = 0, X_2 = 0)$の条件付き確率の値は，$X_1$, X_2がともに起こらない(＝0)とき，X_3が起こらない確率を表している.

ベイジアンネットワークを用いると, 対象となる事象の確率的な構造をネットワーク

形式で表現することが可能になる．例えば音楽の生成系を表す確率構造を，データから以下のように構築することができる．

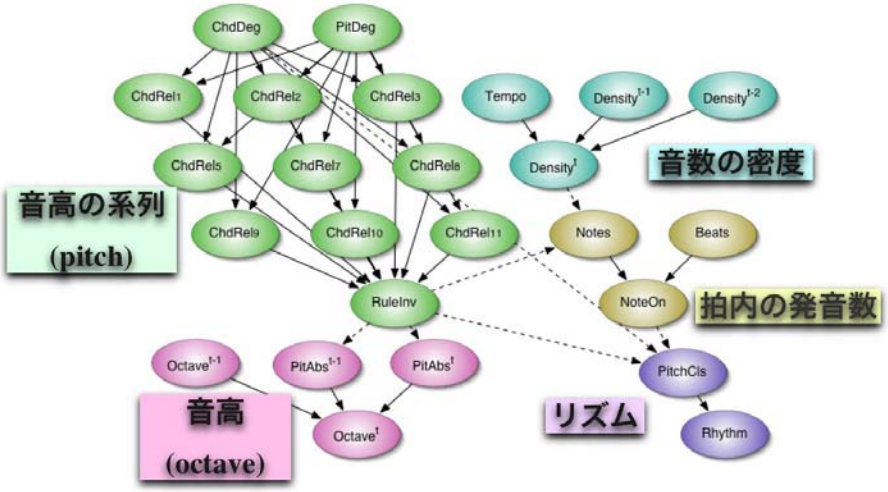


図 11　ベイジアンネットワークによる音楽生成系の確率構造

出典：http://www.eb.waseda.ac.jp/murata/yu.fujimoto/openhouse/20071204a.php

示されているのはあくまで確率構造であり，ここから直感的に何かを読み取ることは難しいが，対象知識を因果的に分析するための表現として有用である．

第二章　ニューラルネットワーク関連図の出典：
http://ryomiyo-web.hp.infoseek.co.jp/

第３章 関連研究

この章では，本研究の提案するアルゴリズムの基礎となっているいくつかのアイデアや関連研究について解説し，それぞれの意義と課題を明らかにした上で，本研究との関係を示す．

第１節 概要

本研究の提案アルゴリズムは，第二章で説明したニューラルネットワークの手法に，本章で示す既存研究のアイデアを適用したものである．

第２節では，転移学習について述べる．転移学習は．未知のタスクに対する仮説を効率的に発見するために，他の領域で得た知識を移転・適用する手法に関する研究である．本研究の提案アルゴリズムは，データに対する学習を終えたニューラルネットワークの構造が，未知のデータに対する予測性を持つことを意図して設計するものであり，したがって一種の転移学習と見なすことができる．領域間で転移・共有されるものは，ネットワークの中間構造である．

第３節では，補助問題と構造探索について述べる．構造探索は，データに内在する構造を学習する手法であるが，そのためにはデータから自動的に補助問題を生成して，これを解く必要がある．この補助問題をどのように生成するかが，構造探索の鍵となる．本研究の提案アルゴリズムは，後述する自己連想記憶理論に基づいてデータから補助問題の生成を行い，それを解くことで，データの予測性に基づくパターンを抽出し，構造化するものである．

第４節では，本研究のベースアイデアとなった Hawkins の自己連想記憶理論について説明する．脳の情報処理システムをアルゴリズムの形に書き下すために用いる理論であり，上述の補助問題の生成も，この理論に依って行われる．本研究の骨子となるアイデアである．

第５節では，Edelman の神経ダーウィニズムについて説明する．情報の構造化にあたって，データに人為的な措置を施したり，対象データの種類や性質に応じてその処理に特化したアルゴリズムを用いなければならないアルゴリズムでは汎用性がない．本研究の提案アルゴリズムが想定しているのは，種類や形式を問わずあらゆるデータを入力することで，自動的に構造化が行われるシステムである．そのため，人間の脳がそうした情報処理の柔軟性を持ち得た理由を説明する，神経ダーウィニズムの理論に着目する．

第２節 転移学習

ある未知のタスクの効果的な仮説を効率的に見つけ出すために，一つ以上の別のタスクで学習された知識を得て，それを適用する手法を，転移学習という．

転移学習（Transfer Learning）のほか，帰納転移（Inductive Transfer），ドメイン適応（Domain Adaptation），マルチタスク学習（Multitask Learning）などの呼び名があるが，本論文では以下，転移学習で統一する．

従来の機械学習と，転移学習の概略を比較すると，図 12 のようになる．

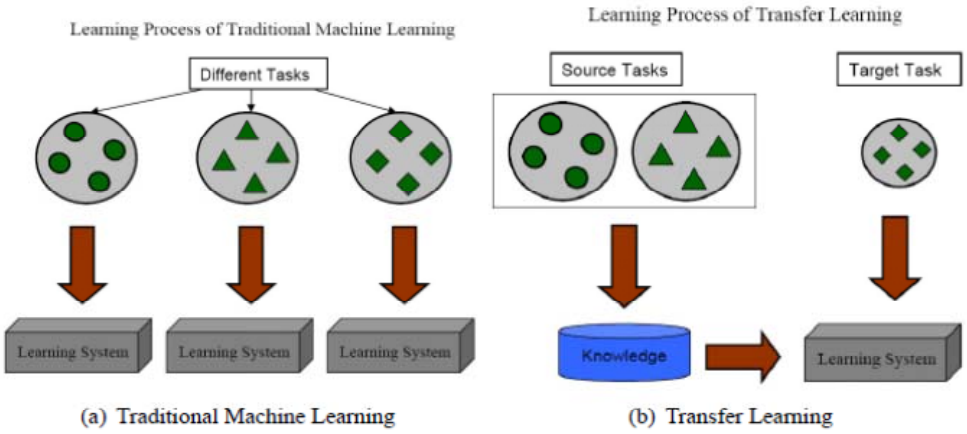


図 12　従来の機械学習と転移学習の比較概略図[7]

転移学習では，知識を転移する元のドメインを元領域（Source Domain），転移する先のドメインを目標領域（Target Domain）と呼ぶ．またそれぞれの領域におけるタスクを元タスク（Source Task），目標タスク（Target Task）呼ぶ．

元領域と目標領域のデータそれぞれに，ラベル（教師情報）のあるなしが考えられるため，４通りの問題に分けられる．このうち目標ドメインのラベル付きデータが使えるものは，帰納的転移学習（Inductive Transfer Learning）と呼ばれ，ひとまとめにされている．したがって，転移学習は大きく以下の三種類に分類できる．

帰納的転移学習（Inductive Transfer Learning）

ラベルの多寡に関わらず，目標領域のラベル付きデータが使える場合の転移学習を帰納的転移学習という．次の二通りが考えられる．

(a) 元領域のラベル付きデータが豊富に使える場合

(b) 元領域のラベル付きデータが使えない場合

前者はマルチタスクと似たところがあるが，転移学習では元領域で学習した知識を用いて，目標領域のパフォーマンスを向上させることが目的であるのに対し，マルチタスクは元領域，目標領域におけるパフォーマンスを同時に向上させることを目的としている点で異なっている．また，後者は Self-Taught Learning と呼ばれることがある．

帰納的転移学習の研究としては，AdaBoost の拡張アルゴリズムを利用した Dai(2007)[8]や，複数タスクにおける共通特徴（Low-Dimensional Representation）を探索する Argynou(2007)[9] などがある．

転動的転移学習（Transductive Transfer Learnng）

元領域のラベル付きデータは豊富に使えるが、目標領域にはまったくラベル付きデータがない場合の転移学習を，転動的転移学習という．

この転移学習で問題となるのは，元領域と目標領域の間で状態構造 $P(y|x)$ がどれくらい異なっているかという点である．状態構造が違うほど，元領域で学習した知識を目標領域に応用することは困難になるからである．したがって転動転移学習では，まず領域間の $P(y|x)$ の比を算出することが肝要であり，その効率的な手法研究として，領域別に分類問題を用意して解くことで計算する Bianca(2004)[10]や，Kernel Mean Matching(KMM)を用いたアプローチの Huang(2007) [11] などがある．

無教師転移学習（Unsupervised Transfer Learning）

元領域でも目標領域でもラベル付きデータをまったく利用できない場合の転移学習を，無教師転移学習という．この条件下では，学習機はラベル無しデータのみをもとに分類の構造を組み立てるしかない．特に自然言語を対象とする場合，通常存在するほとんどの情報はラベル無しデータであり，特定のコーパスなどを必要としない無教師転移学習に期待がかかっている．

無教師転移学習の既存研究としては，元領域に存在する大量のラベル無しデータを用いて，目標領域のラベル無しデータを小さなクラスターに分割する Dai(2008)[12] がある．

本研究は，与えられたデータ自身を予測するという問題を解くことで自己構造化を行うアルゴリズムの提案であり，この構造が未知の問題に対しても有効に使えることを示すものである．データには主に言語コーパスを用い，教師データの存在は意図しない．したがって転移学習としては，教師データを用いない無教師転移学習に分類することができる．

第３節 構造学習と補助問題

第１項 構造学習

構造学習（Structural Learning）は，ある入力データを持つ問題に対して，データ中の潜在的な構造を探索するための学習手法である．

既存研究として，ラベル無しデータから補助問題を生成し，それを解くことで入力データに内在している「予測に役立つ構造」を探る Ando (2005) らの研究[13] がある．以下，同研究のうち，本研究に関連の深い補助問題とその周辺について，基本的なことを確認する．

第２項 仮説空間の探索

ある未知の確率分布 D によって生成された訓練データ$\{(X_i, Y_i)\}$に基づいて，入力ベクトル x を出力ベクトル y に対応させ，かつもっとも誤差が小さいような Predictor **f** を探索する問題を考える．

m 個の学習問題があり($l \in \{1, \dots, m\}$)，それぞれの問題には n_l 個のサンプル (X_i^l, Y_i^l) があるとする $(i \in \{1, \dots, n\})$．また，これらは互いに独立して，分布D_lから得られたものとする．

これら l個の問題について，いま仮説空間の候補 $H_{i,\theta}$ があるとする．$\theta(\in T)$ はすべての問題で共有される構造決定パラメータである．l番目の問題について，$H_{i,\theta}$ の中から誤差を最少にするような Predictor F_l ：X->Y を見つけたい．

θ が与えられば，F_l は ERM（Empirical Risk Minimization：経験誤差最小原理）によって次のように求まる．ERM はトレーニングデータが与えられたとき，Predictor **f** を探し出すために用いられる一般的な手法である．

$$f_{l,\theta} = arg\,min \sum_{i=1}^{n_l} L\left(f\left(X_i^l\right), Y_i^l\right) (l = 1, \dots, m)$$

構造学習の目的は，学習問題に対して最適な θ，すなわちすべての l における **f** の平均値を最小化するような構造を見つけることにある．

補助問題は，この構造パラメータを見つけるために，ラベル無しデータから自動的に生成するものである．

第３項 補助問題とは

何らかの方法で自動的にラベル付けを行ったラベル無しデータを用いて，後述する手法のいずれかを用い，自動的に複数の予測問題を生成する．この生成された予測問題を

補助問題という．

補助問題を用いた最適構造パラメータ θ の探索は，次の二つの手順からなる．

(1)自動的にラベル付けされたラベル無しデータを用いて，補助問題の構造パラメータを学習する．
(2)ここで得られた θ を用いて，目的領域の問題の予測子を ERM により学習する．

ここでは，補助問題の予測子が共有している「予測に役立つ構造」の学習を通じて，仮説空間 H を探索している．もし補助問題が目的領域の問題に似通っていれば，これまでの議論から，この仮説空間 H は目的領域の問題に対しても有効であることが主張できる．

第４項 補助問題の生成手法

補助問題は以下の性格を備えている必要がある．

自動ラベリング：補助問題のため，ラベル無しデータを自動的にラベル付けできること
関連性：補助問題がターゲット問題にある程度関連していること

これらをクリアする補助問題の自動生成手法として，Ando[13]は次の二通りのやり方を提案している．

手法１：Unsupervised-Strategy

Unsupervised-Strategy は，入力データの一部を補助ラベルにしてしまう手法である．例えば，「対象文書中のある単語について，その単語が{名詞，動詞，その他}のうちどの品詞に属するかを分類する」というタスクにおいて，各単語についてそれぞれ現行の単語を，周りの単語を用いて予測したい補助ラベルと見なすことで，補助問題を生成することができる．

あるいはより一般的には，入力データの一部をマスクしてしまい（観測対象外のデータとして扱う），マスクしなかったデータを用いてこれらのマスクされたデータを予測するような Classifier を学習する方法がある．

手法２：Partially Supervised-Strategy

Partially Supervised-Strategy は，$\Phi 1 : X -> F$，$\Phi 2 : X -> F$ のふたつの

Feature Map を用いる手法である．具体的には，次の手順で行う．

1. ラベル付きデータ Zにより，$\Phi 1$を用いて Classifier T1 を訓練する．
2. T1 をラベル無しデータに適用することで，補助問題のためのラベル付きデータを生成する．
3. $\Phi 2$のみを用いて，補助問題の ERM により構造パラメータ θ を学習する．
4. ラベル付きデータ Zにより，θ と適当な Feature Map Ψ を用いて目的の Classifier を訓練する．

例えばターゲット問題が c 通りの分類であるとすると，この問題には c 個の二者択一（0 or 1）問題が含まれていることになる．このとき，ある classifier T1 が，全情報の半分だけを用いて分類問題を解くときに，先ほどの classifier が用いなかったもう半分の情報だけを用いて，「T1 があるカテゴリーを推薦するかしないか」を予測するような補助問題を考えることができる．

Partially Supervised-Strategy は入力データの種類や特性に依らず，大量の補助問題を生成することができる手法である．

一方，Unsupervised Strategy は，入力データの中に本質的な構造が眠っていることを前提として，その構造を，補助問題を解くことで明らかにしようとする手法である．たとえば入力データが自然言語であれば，ある単語群の共起関係や文法的な仕様が，「本質的な構造」に当たる．

第５項 本研究における補助問題

本研究の提案アルゴリズムでは，補助問題は入力データ自身を予測する問題とする．即ち，第４項で分類した手法のうち，「手法１：Unsupervised Strategy」にあたる手法の変形を用いる．

後述する実験では，提案アルゴリズムへのコーディングに先立ち，まず多層パーセプトロンとバックプロパゲーション法を用いた数字認識実験により，この補助問題の生成手法が妥当であることを確認する．

第４節 自己連想記憶理論

自己連想記憶理論（Memory-prediction Framework）は，2004 年に Jeff Hawkins

によって提唱された，脳についての統一理論である．

彼はその著書，”On Intelligence”(2004)[14]のなかで，知能とは何かという問いかけに答えて，認識したパターンから周辺世界について予測を行う能力こそが知能の本質であるとした．自己連想記憶理論はこの主張を神経科学的に裏付けようとするものである．

彼が着目したのは，哺乳類に共通している皮質組織の均一な物理的な構造である．彼はこの物理的な構造が，大脳新皮質の情報処理の基礎となるような，何らかの原則的で単一なアルゴリズムを反映しているはずだと仮定した．そうしてそのアルゴリズムの基本的な機能は，皮質及びその周辺部位（視床や海馬など）を巻き込んだ脳構造内における，情報のフィードバックと再帰ループにあるとした．この仮説に従えば，哺乳類をはじめとする知的な生物の，一見複雑に見える行動は，すべてこの単一の機能から生じたものに他ならない．これは，我々哺乳類が行っている種々の複雑な行動について，その神経科学的な制御の方法に，統一的な説明を与えるものである．

自己連想記憶理論における脳の情報処理は，以下のようなプロセスを辿る．まず外部入力は，大脳新皮質のネットワーク構造の最下層にセットされ，ニューロンの発火とシナプス伝達により階層構造を上位レベルへと上っていく．そうしてボトムアップ的に処理される過程で，構造内における上位ニューロンを活性化する．これが認識プロセスである．次に，活性化された上位のニューロンから，階層を下向きに下るシナプスを通じて，トップダウンの発火プロセスが生じる．これが予測プロセスである．階層状に積み重なったネットワークは，外部入力に対して常に認識と予測という双方向のプロセスを発生する．情報は階層内を上下に流れ，ネットワークは予測を立てながら複雑な外部入力を統合していく（図 13）．

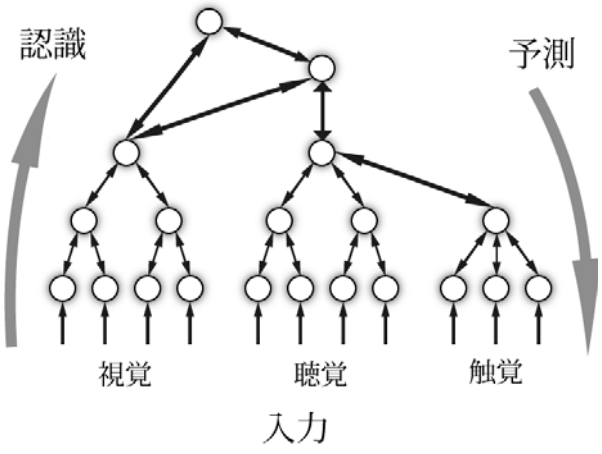


図 13　新皮質プロセスの自己連想記憶理論的解釈・概略図

新皮質を自己連想記憶理論的に解釈すれば，視覚や聴覚などの異なる感覚からの入力も，より上位の階層では統合され，視覚情報から聴覚情報を予測することも可能になる．

また，こうした階層構造によって，脳は抽象概念や名前付けといった機能を持つことができるようになる．そうした機能は，上位階層の構造としてネットワークに保存されている．一方で，下位階層は変遷の激しい外部入力情報の詳細を正確に反映している．これが大脳新皮質の情報処理の基礎となる「単一なアルゴリズム」である．

人工知能の実現にあたって，Hawkins は，脳が行っていることをそのままコンピュータにプログラムしても，知性を持つコンピュータを作ることは不可能であると主張する．そうして，我々の脳の大脳皮質で実現され，人間の知性の基底となっているものはただ自己連想記憶理論のシステムであり，それに基づいて，単純にパターンの発見方法と利用方法をコンピュータに教えることが重要であると説く．

本研究の提案アルゴリズムは，ニューラルネットワークモデルにこの自己連想記憶理論を適用し，与えられた情報から入力自身を予測するように自己構造化を行うことで，未知のパターンに対して迅速かつ有効な予測が行えるという仮説を踏まえ，設計したものである．

第５節 神経ダーウィニズム

神経ダーウィニズム（TNGS＝Theory of Neuronal Group Selection：神経細胞群選択説）は，1987 年に Gerald M.Edelman が提唱した，脳機能に関する仮説である．

Edelman は，「脳は進化によって生じたものであり，計画的に設計されたものではない」として，それゆえに脳の機能は，「機能しうる構造や生物個体は，同じ集団に属する多様な変異を抱えた個体が，互いの生存をかけた競争において淘汰選択される結果として出現する」というダーウィンの集団的思考原理に基づいて説明されるべきだと主張する．

すなわち脳のネットワーク構造形成において，特定の状態が選択され，その他の状態が淘汰されていくことによって，脳機能の多様性を説明する．

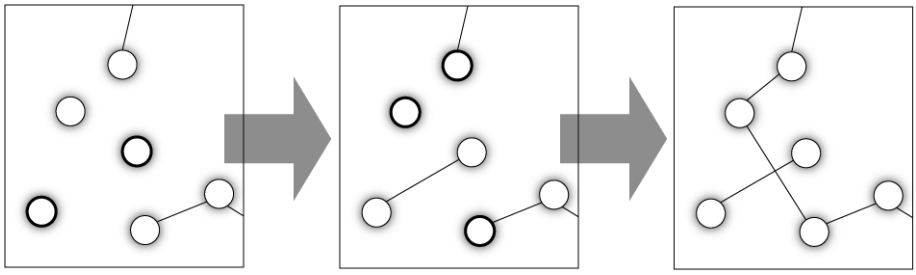
Edelman の神経細胞群選択説は，以下の３つの原理から成る．

(1)発生選択

脳が一応の解剖学的構造を整えるまでのあいだの第一段階．発生したニューロンは他のニューロンと配線し合い，ネットワークは様々なパターンを形成する．この結合パタ

ーンの多様性によって，脳領域には，何百万というシナプス回路からなるレパートリーが構成される．

この段階におけるシナプス形成のルールは，「同期して発火したニューロン同士は互いに配線され，ネットワークを形成する」というものである（図 14）．



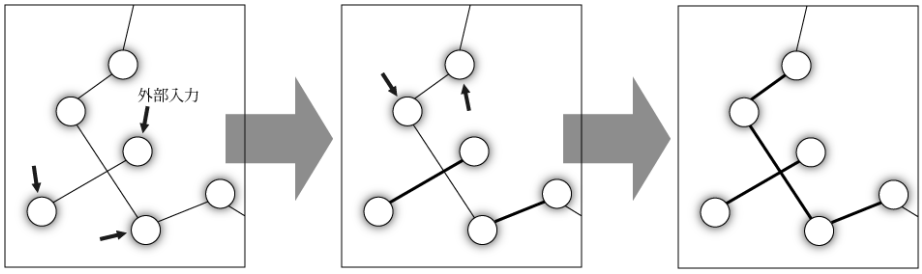
太線のニューロンは同時に発火したニューロン

図 14　TNGS における発生選択

(2)経験選択

発生選択とオーバーラップしつつ生じる第二段階．主要な解剖学的構造が完成した後，環境からの様々な入力によって，シナプスの結合荷重が強められあるいは弱められ，多様な結合強度を持つネットワークを形成する（図 15）．

発生選択と経験選択によって形成されたネットワークは，選択淘汰に耐えうる非常な多様性を持つ．



太線のシナプスは外部入力によって強化されたシナプス

図 15　TNGS における経験選択

(3)再入力

発生・発達の過程で出来上がった双方向性の連絡が，局所的にも広域的にも数多く形成される．再入力回路は，複数の脳領域間を結びつける同時進行的な信号伝達であり，これを用いて別々の機能を持つ脳領域間が時間的・空間的に統合されることが可能とな

る．すなわち，脳の様々な領域で起こっているニューロン活動が広範囲にわたって同期する．

TNGS では，神経細胞群選択説における選択淘汰の単位は，この再入力による相互作用で結びついたニューロン群であるとしている．

神経細胞群選択説を Edelman 自身が後に発展させた「拡張 TNGS」では，ダイナミック・コア仮説として，「機能クラスターとしてふるまう相互作用系内での再入力性の信号のやりとりが，コアの状態推移が意識状態を生み出す」と主張している．

本研究は，この拡張理論も視野に入れて，提案アルゴリズムの延長線上に位置づけつつ，ニューラルネットワークの発生・発達過程に神経ダーウィニズムの理論を用いる．また経験選択について，Edelman は特定のニューロン群が選択されると示唆するに留めたが，提案アルゴリズムでは選択淘汰の基準となるものは予測の精度である．すなわち予測の精度が高いニューロンは生き残り，その他のニューロンは淘汰される．これは第 4 節で説明した自己連想記憶理論に則ったものである．

第４章 ベースアイデア

この章では，本研究の動機となる問題意識を明らかにし，それを解決するベースアイデアを提示した上で，研究の目的を述べ，その定式化を行う．

第１節 知の構造化

知識量の爆発的増加や専門分野の微細化から生じる問題へのアプローチとして，小宮山（2004）は「知識の構造化」を提唱した[15]．そこでは知識の構造化とは「構造化知識，人，IT，およびこれらの相乗効果によって，知識の膨大化に適応可能な優れた知識環境を構築するすること」と定義されている．具体的には，個々の研究者の頭脳に散在している知識を単一のネットワークとして構築し，それを必要に応じて様々な形で取り出せるようなシステムを作ることによって問題を解決しようというアプローチである．そのシステムの構築に目覚しい進化を遂げる IT を利用し，知識の整理・抽出・俯瞰を可能にしようと主張している［16］．

本研究の提案アルゴリズムも，知識を構造化するものである．従来のアプローチと異なる点は，予測性という原則のみに基づいて構造化を行うことで，対象知識の種類や形式を問わないことと，構造化は一切の人為的な操作を加えることなく，機械によって自動的に行われることである．

次に，このアルゴリズムを提案するに至った動機と，提案する目的を示す．

第２節 動機と目的

学術俯瞰マップは，人間に概念及び概念間の構造をわかりやすく伝えることに貢献している．任意の大量情報を自動的に編集し，より人にわかりやすく伝えるためには，まず学術俯瞰マップの貢献の原理が解明される必要がある．

学術俯瞰マップは，論文とその引用・被引用関係いう整理された情報が与えられたとき，それを構造化して示すツールである．これに類するクラスタリングや抽象化の手法の研究は数多くなされているが，いずれの場合も，ネットワークとして可視化されたものは，整理されていない雑然とされた情報よりも見ていて面白く，有用である．では，なぜ面白く，有用だと感じるのだろうか．その答えとして，ネットワークという可視化の方法は，人の予測に貢献するものだからという説明が考えられる．

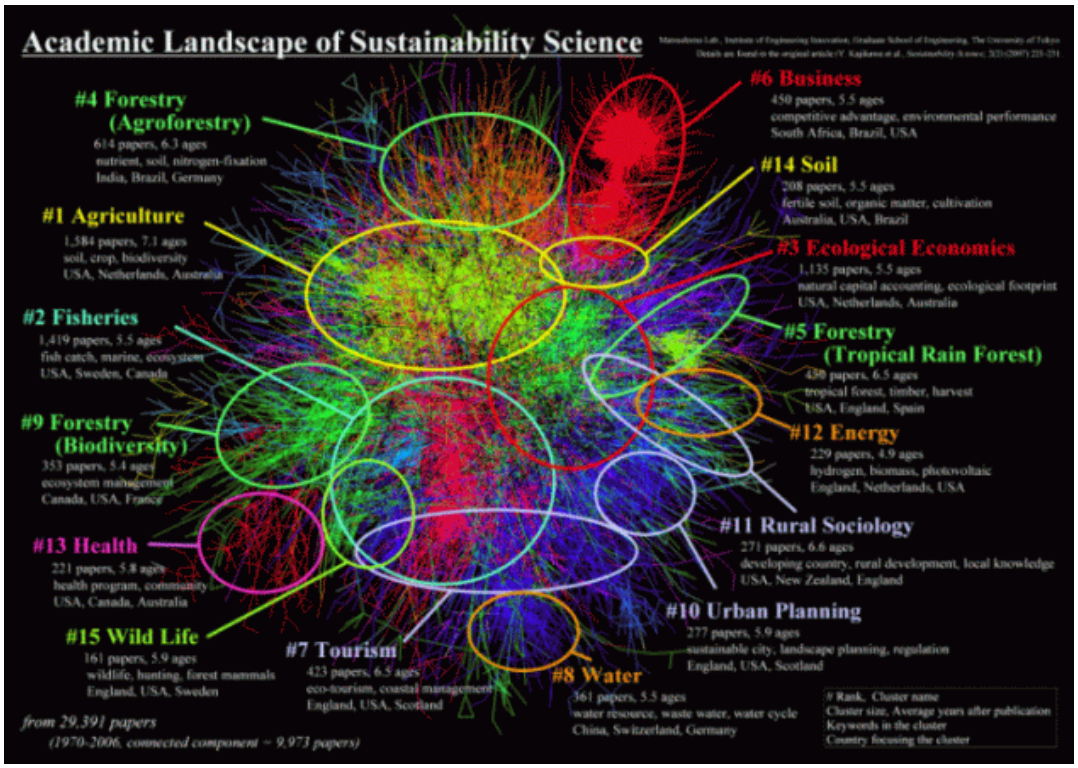


図 16　サステナビリティ学の俯瞰マップ

図 16 に示したのは，サステナビリティ学の俯瞰マップである．サステナビリティ学を 15 の領域に分類し，クラスター化している．ノードは論文，エッジは引用関係である．クラスターの番号はクラスター内に含まれる論文数の多い順に振られている．

この俯瞰マップと，あるサステナビリティに関する論文を示し合わせたとき，人は様々な予測を立てることができる．例えば手にした論文が「#1 Agriculture」に属するものであるとすると，この俯瞰マップから，それが農業というクラスに属する論文であるということと同時に，クラスターの隣接関係から，漁業（#2 Fisheries）とも関連の深い研究であるということを予測することができる．農業に所属する研究者は，漁業に関する知識も持っているのではないかという予測も立てられる．あるいは健康（#13 Health）の論文について，ビジネス（#6 Business）やエネルギー（#12 Energy）の研究とはほとんど関係がないと予測することもできるだろう．ほかにも，クラスターとクラスターの間隙区域にプロットされる論文は，それらのクラスターをつなぐハブの役目を果たしており，両領域の研究に関する共通の知見が得られる可能性が高いと予測できるし，クラスターの中央部にある論文はそのクラスター内での重要度が高く，末端にある論文はそれほど重要視されていないという予測を立てることもできる．いずれにしてもこれらの予測は，すべて大量の論文データをネットワークという形に可視化して，はじめて可能となるものである．こうして予測を立てられるということによって，人はネ

ットワーク図から何らかの知見を得ていると考えてよいだろう．

ネットワークは人に予測を可能にさせるものである．しかし，学術俯瞰マップをはじめとする従来のネットワーク図は，予測性を高めることを目的に作られたものではない．学術俯瞰マップは，単に論文の引用・被引用関係を，数学的な手法によりネットワーク化しただけである．

ここに情報構造化手法のさらなる改良の余地がある．すなわち，ネットワークの本質である予測性に着目し，予測性を目的関数として構造化を行えば，その構造化された情報は，従来の手法で構造化されたものよりもより人に予測を可能にさせるものとなるはずである．これは前述の議論から，より面白く，有用な情報の構造化が可能になるということであり，その意義は大きい．

さらに，予測はいかなるデータに対しても，そのデータが何らかのパターンを内包している限り，成立する．したがって予測性を構造化の目的関数とすることで，引用・被引用関係といった明示的な関係情報に頼ることなく， いかなるデータに対しても情報を自動的に構造化できるはずである．本研究は，そのようなアルゴリズムを設計・構築することを目的とした研究である．

第３節 アイデア：情報の構造化と「脳」

整理されていない情報を上手く構造化する既存のツールとして，われわれにもっとも身近なものは，人間の脳である．脳が情報の構造化を行うものであることは，前述したHawkins や Edelman の説からも明らかである．特に，本研究が立脚するベースアイデアである自己連想記憶理論において，Hawkins は，脳は予測性を本質として情報の構造化を行っていると主張した．これは前節の議論に対して，ひとつの解の方向性を示している．脳の情報処理に着目することは，予測性を本質とする情報構造化のアルゴリズムの構築に役立つはずである．

脳の情報処理は早くから注目されてきたが，これまでに目論まれたコンピュータによる脳の再現，あるいはそれに類する人工知能の研究はことごとく行き詰ってきた．しかしここ数年のウェブの飛躍的な発展を受けて，脳をめぐる研究の事情は好転してきている．ウェブに由来するネットワーク理論は近年ますます盛んになり，同時に大量の情報処理技術の研究もふたたび活発になってきた．ウェブに端を発する様々の新たな知見と活動をもとに，改めて脳の情報処理に関する新たな展開が期待できる．

脳の進化の目的は，限られた情報で出来るだけ効率的かつ迅速に将来を予測するよう，自己の構造を変更していくことである．そのアルゴリズムを解明し，脳の情報処理アルゴリズムをシステムとして実現すれば，現在のコンピュータでは達成し得ていないような、あらゆる種類の有益な仕事ができる知能機械を作ることができるようになる．

第４節 情報自動構造化アルゴリズムの意義

ここまで議論してきた構造自動化アルゴリズムが実現されたとき，それが現実的・社会的にどのような意義を持つかを考える．

もっとも直接的な貢献は，現在の様々な情報認識システムの精度を高め，燻っている多くのシステムに対して実用化への道を拓くことである．

たとえば現在の音声認識は，人の話す言葉を単語または文レベルで「認識」することはできても，「理解」することはできない．それゆえに，背後で偶然話されてマイクに捉えられた文脈的にまったく脈絡のない言葉や，意図せずに発生してしまった声までが認識されてしまったり，認識の僅かな間違いの結果，まったく意味を成さない文章が入力されてしまたったりといった問題を，未だに解決できていない．ここに本アルゴリズムを適用すれば，次のようになる．アルゴリズムは人間の話し言葉の音声パターンを学習して構造化を行い，記憶の階層システムを作り上げる．記憶の階層システムが充分に学習されれば，アルゴリズムは入力に対して，それがどのような文字列かではなく，どのような抽象概念を喚起するかを示すようになる．これは音声入力を人が「理解」するプロセスと同じである．理解をベースにした音声認識が可能になり，上述したようなイージーミスは起こりえなくなる．

視覚情報処理システムも，現在のシステムではかろうじて人の顔を認識することができる程度である．画像の中にどのような物が配置されているかや，映し出されている映像の意味を把握することはできない．例えば現在のセキュリティシステムでは，監視カメラに映る人物について，それを不審者とそうでない人物に見分けることはできない．したがって，現在ではまだ人間による監視体制を取らざるを得ないが，これも音声認識と同様，映像の意味を理解できるアルゴリズムであれば，明らかな意図を持って侵入行動を起こしている不審者に対してのみ警報を鳴らすことも可能になる．

同様の議論は，輸送交通システムにも応用可能である．自動車の周囲にセンサーを取り付けることで，自動車の周辺状況についてのパターンを学習し，やがてアルゴリズムが走行における様々な現象の意味を理解するようになれば，用意されたプログラムの必要ない，真の自律走行が可能になるだろう．

ウェブに活躍の場所を限定しても，用途の想像には事欠かない．現在の検索システムはクエリによる文書とのマッチングのみであるが，画像に移りこんでいるものを指定しての画像検索や，音楽検索なども可能になるだろう．のみならず，パターンを持つ情報であれば，人間の言語で「説明」することによって，あらゆるものが検索可能になるはずである．

さらに，知能を持つ記憶システムが脳のアルゴリズムで実現されれば，その物理的な恩恵のために，様々な点で人の脳を凌駕する性能を発揮すると考えられる．記憶システムが人間の脳に対して持つ長所は以下の点である．

スピードが速い

ニューロンの反応時間は数ミリ秒単位であるが，半導体の動作時間はナノ秒単位である．また，記憶システムは人間のように「現実が変化していく時間」に対応する必要がないため，いくらでも高速で情報を取り込み，記憶の階層を訓練していくことができる．従来の人間の思考速度では処理できなかった問題に，解法を与えることができるようになる．

容量が無制限

人間の脳は，母体の骨盤の大きさや，脳の活動を維持するための栄養素など，様々な生物的制約を受けており，必然的にその容量に制限がある．しかし記憶システムは，この構造をこそ模倣するが，物理的な制約までは模倣しない．記憶容量はハードディスクと同じように，記憶媒体を連結することによって好きなだけ増やすことができる．これによって，人間では不可能なレベルまで記憶の階層構造を高めることもできるし，単純にひとつの領域にあてる記憶容量を増やせば，より詳細な情報をストックすることができるようになる．

初期化・複製の容易さ

人間の脳と異なり，記憶システムは階層構造の訓練に失敗した場合，いくらでも初期化して最初からよりよい条件で学習しなおすことができる．また，偶然にしろ必然にしろ，一度目的のタスクをこなすために準最適な構造になり，これ以上余計な情報を学習する必要がない状態になれば，その時点で学習を止め，誰もがその状態のシステムを使えるように構造を複製することができる．

感覚の多様性

知能を持つ記憶システムは，自然界に存在する感覚に加えて，人工的な感覚さえセンサーとして利用することができる．ソナー，レーダー，赤外線カメラなどは，人間が利用することのできない感覚器であるが，記憶システムはこれらの感覚もセンサーとして

扱い，そこからの入力をパターンとして認識し，記憶の階層構造を作り上げることができる．中枢となるアルゴリズムさえあれば，センサーはどこにどのように配置されていてもよいのである．

たとえば地球全体に点在するセンサー網を利用すれば，時間変化とともに移り変わっていく各地の観測装置から送られてくる情報のパターンを学習し，大域的な気象現象を予測することも可能になる．のみならず，観測可能な情報は，何もかもが可能な予測の対象になる．動物の移動，人口の変化，病気の流行など，社会的に有意義な予測はいくらでもできるようになる．

情報自動構造化アルゴリズムの機能は入力データのパターンを認識して構造化することであり，それ以上でもそれ以下でもない．したがってどのようなセンサーからのどのように入力に対しても，アルゴリズムの挙動と性能は変わらない．これが，予測性を本質とした情報構造化アルゴリズムの極めて高い柔軟性である．

また，記憶システムは階層が高くなるほど，パターンのより深い学習が可能になり．より複雑な類推ができるようになるが，これも人間の脳は何らかの生物的制約を受けていると考えられる．この制約を取り払えば，記憶システムはさらに高次の抽象化，類推を行えるようになるかもしれない．

膨大な量の情報を人間とは比較にならないほど迅速に記憶し，抽象的なパターンを予測する．感覚器に生物的制約はなく，センサーは自由自在に配置・実装することができる．そのような知能を持つ記憶システムの実現に向け，すべては「知能」とは何か，という問いの答えからはじまる．その答えを探る試みが，本研究の自己連想記憶理論による情報自動構造化アルゴリズムの構築である．

第５節 解くべき問題の定式化

次に，本研究が解くべき問題を定式化する．

第３章第３節で，データから補助問題を生成して，それを解くことでデータに内在する構造探索を行う手法があることを述べた．しかし，データにラベル付けを行い，その一部をマスクして他のデータからマスクしたデータを予測するという Ando らの補助問題の生成手法は，理論的には優れていても現実的には妥当ではない．現実世界では，そのような状況でデータを取り入れることはほとんどないからである．

そこで本研究では，現実世界においても妥当な方法で，大量のデータの入力から 自

動的かつ連続的に補助問題を生成し，それを解きながら自己構造化を行う方法として，「現在までに観測されたデータを用いて将来観測されるであろうデータを予測する」という補助問題の生成方法を提案する．これは Hawkins の自己連想記憶理論によれば，人間の脳が自己を構造化するために日々行っているものであり，したがってそれを補助問題とすることは，脳の情報処理システムを模倣しようとする本研究のアルゴリズムにとって適切であると考えられる．

アルゴリズムは，階層記憶システムを内部に作り出すため，現在までに観測された過去の情報に基づき，次に起こる観測について予測を立てる．このことを考えるために，いま観測と時間の関係を模式的に図示すると次のようになる．

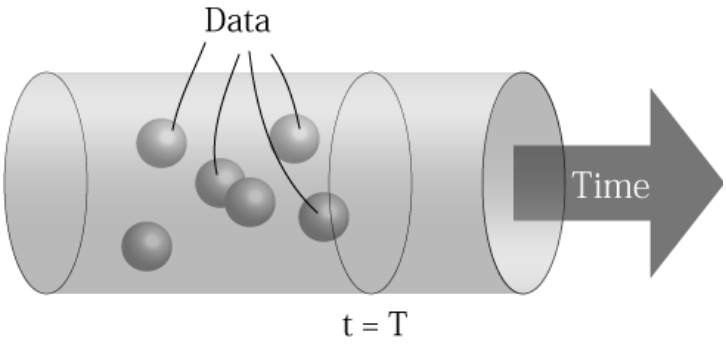


図 17　データを内包する時空間チューブ

図 17 は，データの散在している空間を時間軸方向に引き伸ばしたものである．これを以下，時空間チューブと呼ぶ．我々が外部に観測する情報は，この時空間チューブ内の任意の点として表すことができる．

時空間チューブ上においては，予測という機能は図 18 のように表わせる．

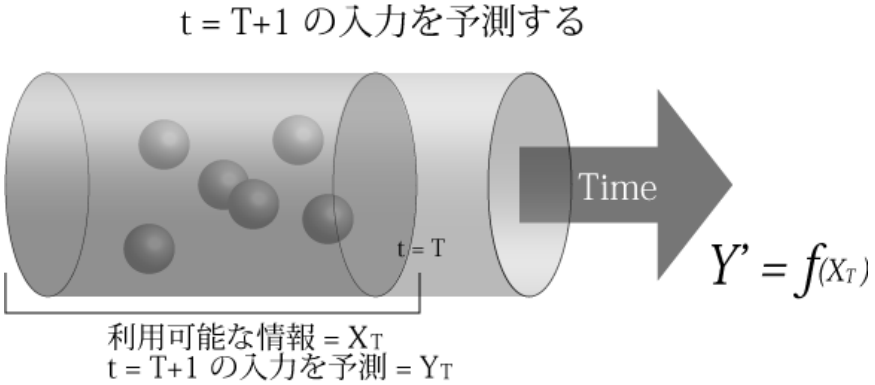


図 18　時空間チューブ上における予測の表現

予測のために使える情報は，現在（$t = T$）以前の時空間内に存在するすべてのデータである．これを入力 X とし，次の時刻におけるデータの観測予測を出力 Y' とすると，予測という機能は，$Y' = f(X)$ における f であると言い換えることができる。そうして，予測性を最大化することは，実際の観測を Y としたとき，次の式で表される誤差 E を最小にすることである．ただし，$g(a,b)$ は，何らかの形で a と b の乖離を表す関数である．

$$minimize\ E = \sum_{y} g(f(x), y) \quad (x \in X, y \in Y)$$

多くの分類問題は，f をデータの分離平面として，その平面の探索にアルゴリズムの粋を尽している．しかし階層記憶システムでは，過去の観測データを自在に用いて未来を予測するということを端的に表すのであれば，f は図 19 のように，観測データ同士を自在に繋いで作る構造として表現することができる．

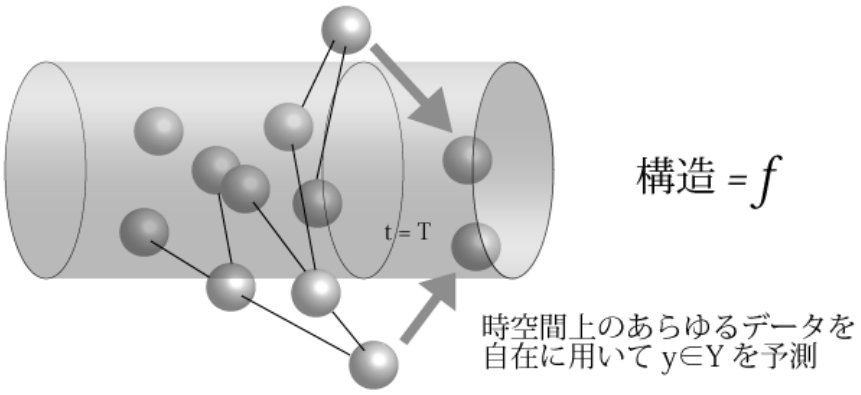


図 19　構造自体を f とする解釈・概略図

過去の観測データに基づいて未来を予測する，その予測精度を最適化する問題は，この構造の最適な形を探索するタスクであると言い換えることができる．

しかし，データの量と存在範囲が大きくなればなるほど f の解空間は広大になり，探索には時間がかかるようになる．本研究の目的は，この広大になった解空間から，いかにして迅速に，かつ良好な f を探索するか，その方法論に原理を見出し，統一的な説明を与えることである．

ネットワーク構造を変形・更新する方法はいくらでも考えられる（図 20）．未知の予測を行うためにもっとも効率的で効果的な更新方法は何か．本研究ではそれを，第３章第４節で説明する自己連想記憶理論に基づき，過去の観測データ自身を予測するという補助問題を解くことによって構造化を行うことであると仮定する．そうしてこの手法が，未知の予測に対して優れた f を迅速に探索する手段となっていることを証明する．

以上が本研究の目的の定式化である．

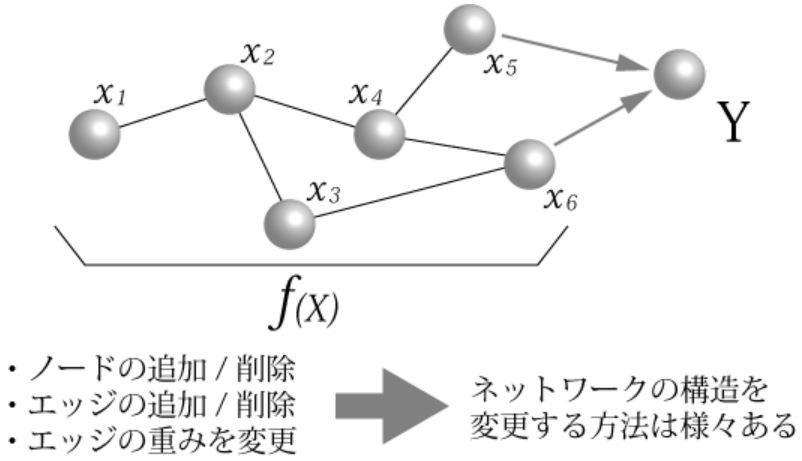


図 20　ネットワークの構造更新

第５章 アルゴリズムの設計と分析

この章では，本研究の提案アルゴリズムを設計するための理論的な段階を示すために，提案アルゴリズムの機能を要素に分解し，それぞれの効果を検証する．また，それらをモデル化してプログラムを実装し，人工データによる実験を行い，考察を行う．

第１節 概要

これまでに述べてきた本研究の提案アルゴリズムの全貌を定義すると，「なんらかの入力データを大量に与えることで，そのデータ群に内在するパターンを発見し，ネットワークの内部に構造化された階層記憶を作り出すようなアルゴリズム」と記述することができる．

いま，これをデータの抽象化という言葉に集約する．具体的にどのような仕組みがデータの抽象化を実現しうるだろうか．

本章では，その仕組みを選択淘汰，適応，多層化，時間方向の圧縮という四つの要素に分解して考え，個々の要素について，効果を確認するためのアルゴリズムを設計し，人工データを用いた実験と考察を行った．

本章で行う設計及び実験の順序は次のようになっている．

第２節では，個々の要素を検討する前に，まずアルゴリズム全体の基板となる補助問題についての検証を行う．すなわち，データ自身を予測するという補助問題を解くことによるネットワークの自己構造化が，データの抽象化に寄与することを確認する．そのために，多層パーセプトロンとバックプロパゲーションを用いて実験を行った．実験データには，結果を視覚的な形で容易に確認することのできるデジタル数字認識を用いた．こちらで用意した実験用のデータと，一般に機械学習のため公開されている手書き文字の認識実験用データを用意して，二通りの実験を行った．

実験の結果，ここでは，多層パーセプトロンに対して数字をあらわすベクトルデータを与え，データ自身を予測するという補助問題を解きながらバックプロパゲーションによって構造を更新することで，データの抽象化を行う中間ニューロンが自動的に生成されることが確認された．

第３節以降は，上述した各要素の効果を確認するための実験群である．

第３節では，選択淘汰の検討を行う．神経ダーウィニズムに基づき，データの予測に役立つニューロンのみを生き残らせ，予測に役立たないニューロンを淘汰するというプ

ロセスを，三層のネットワークに対して繰り返し行う．

実験の結果，ここでは，上述のプロセスを繰り返すことで，計算時間はかかるものの，最終的にはデータの予測に最適なネットワーク構造に辿りつけることが確認された．

第４節では，適応の検討を行う．前節で設計した神経ダーウィニズムによる選択淘汰のモデルに，ニューロンの適応力を評価するリワードの概念を導入することで，ネットワーク内の軸索の重み更新を可能にしたモデルを設計する．

実験の結果，ここでは，適切な更新ロジックを与えることで，ネットワークの構造最適化が，前節の神経ダーウィニズムのみに基づくモデルよりも早くなることが確認された．

第５節では，多層化の検討を行う．上述のモデルはいずれも三層のネットワークであるが，構造が三層である限り原理的に解くことのできない問題が存在することを示し，その問題を多層化されたネットワークに解かせる実験を行う．多層化のプロセスは，いくつかの手法を比較検討する．

実験の結果，ここでは，多層化されたネットワークは，前述の三層構造では解けない問題を解くことができることが確認された．また，多層化のプロセスの比較により，優れた多層化の方法も確認された．

第６節では，時間方向の圧縮の検討を行う．多層化モデルは計算に膨大な時間がかかることを確認し，その解決方法として，入力データを全て等価なネットワークへの入力情報として扱うのではなく，観測の時間方向に圧縮し，実数値として入力層に与えるモデルを設計する．ネットワーク構造の経済化を試みる．

実験の結果，ここでは，入力データの時間方向の圧縮により，入力層及び中間層の構造が経済化されたネットワークは，もっとも優れた多層モデルにも劣らない精度と，遥かに速い構造最適化の速度を示すことが確認された．

最後に，設計したすべてのモデルについてその比較検討を行い，はじめに分解した各々の要素がデータの抽象化にどのように寄与していたかを考察する．

第２節 BackPropagation Model

第１項 概要

本節では, データ自身を予測するという補助問題を解くことによるネットワークの自己構造化が，データの抽象化に寄与することを確認する．

本節で行う実験では，学習のためのネットワークに図 21 のような三層パーセプトロンを用いる．またすべての結合荷重は，後述する改変されたバックプロパゲーションにより行うものとする．

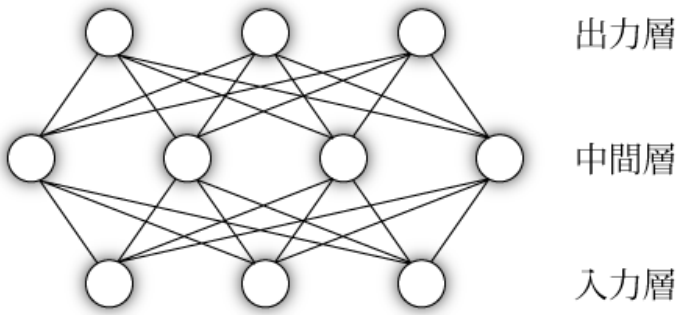


図 21　三層パーセプトロン

また，実験は同じアルゴリズムを用いて，以下のふたつのデータに対して行う．

実験 A：25 マスの黒または白のセルから成る，デジタル数字認識問題
実験 B：64 マス，16 段階のセル濃度から成る，手書き文字の認識問題

実験 A では, 実験のために人工的に用意したデジタル数字の訓練データを用いて，バックプロパゲーションを用いてデータ自身を予測する補助問題を解くことで, ネットワークがどのように訓練データを学習しているか, またそれが適切な自己構造化であるかどうかを検討し，同時に中間層でどのような現象が起きているかを確認する．

実験 B では，ウェブ上で機械学習のためのデータとして公開されている，手書き文字の訓練データを用いて，文字実験 A で得られた補助問題に関する知見が，既存の訓練データセットに対しても適用できるかどうかを確認する．同時に，中間層で起こっている現象を確認する．

第２項 実験Ａ：デジタル文字の認識

25 マスの黒または白のセルから成る，デジタル数字データに対して，次の問題を設定する．

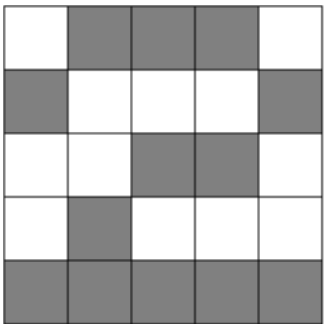


図 22　デジタル数字の認識問題

分類対象は縦横五マスの黒または白のセルである．これを一次元のベクトルに変換して入力データとし，同じ数字に対して何通りかのセルのパターンを用意して訓練データとする．今回は，各数字の理想的な状態（もっともデジタルな数字に見える状態）のベクトルを訓練データとした．以下にベクトルを示す．

【実験 A:各数字のベクトル表現】

0：{1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1}
1：{0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0}
2：{1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1}
3：{1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1}
4：{1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0}
5：{1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1}
6：{1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1}
7：{1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1}
8：{1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1}
9：{1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1}

このベクトルが入力層に入力データとして与えられる．したがってネットワークの入力層と出力層のニューロンの個数は，それぞれ 25 個である．

ここで上記のベクトルが表す正解の数字が何であるかを教師データとして与え，学習

を行う方法が従来の機械学習である．対して，本研究の手法では，まず訓練データに対して次のような補助問題を想定する．

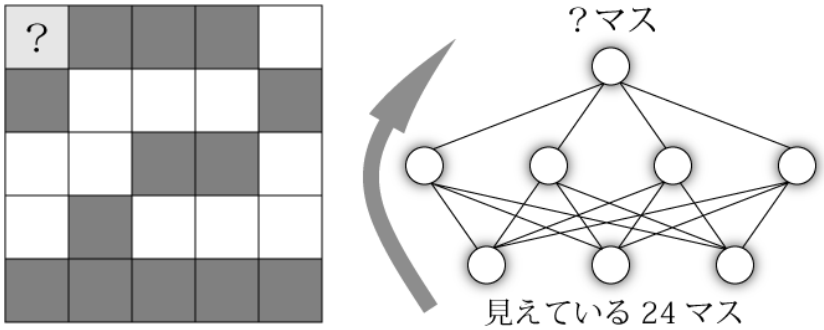


図 23　数字認識における補助問題（1）

すべてのマスについてこの問題を解くとすれば，ひとつの訓練データに対して全部で 25 問の補助問題を生成できる．

そこで，この 25 の問題を単一のネットワークで同時に解くことを考える．即ち図 24 のように，すべてのマスが，自分自身を除く他のすべてのマスから自分自身を予測するようなネットワーク構造である．

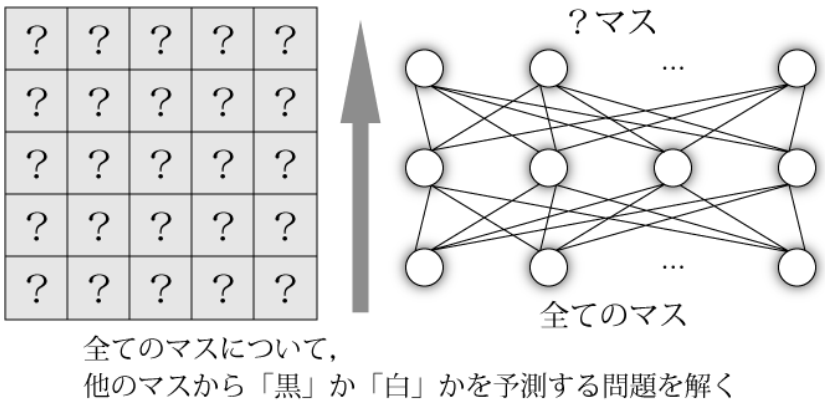


図 24　数字認識における補助問題（2）

このネットワークをバックプロパゲーションによって訓練する．訓練終了後，今度は訓練データとは異なるベクトルを試験データとして与え，訓練されたネットワークがどのような出力層の値を取るかを確認する．

今回は，全部で 10 の訓練データ，40 の試験データに対して実験を行った．以下に実験のアルゴリズムと，パラメータ別の実験結果を示す．

アルゴリズム

反復試行回数を定め，反復試行を開始する．

まず訓練フェーズを行う．

(1)訓練データベクトルを読み込み，入力層のニューロンに値をセットする．

(2)中間層のニューロンの値 V_j を，次の式に基づき算出する．ただし，$Sigmoid()$ はシグモイド関数，W_{ij} はニューロン i からニューロン j への結合荷重，θ_j はニューロンの閾値である．

$$V_j = sigmoid\left(\sum_i W_{ij}V_i + \theta_j\right)$$

(3)出力層の値 V_k を，次の式に基づき算出する．このとき，通常のバックプロパゲーション演算と異なり，自分自身に対応する入力層からの値は，出力層の値の計算に用いない．（改良されたバックプロパゲーション．）

$$V_k = sigmoid\left(\sum_j W_{jk}\left(sigmoid\left(\sum_{i\neq k} W_{ij}V_i + \theta_i\right)\right) + \theta_j\right)$$

(4)出力層の各値と，正解（訓練データの値）を比較し，次の式に基づいて訓練データ m に対する平均二乗誤差 E を算出する．

$$E_m = \left(\sum_{i=k} V_i - V_k\right)^2$$

(5)次に，結合荷重修正のためのバックワード演算を始める．まず，中間層のニューロンの誤差 D_j を次の式に基づいて算出する．ただし， α は学習係数である．

$$D_j = \alpha \times V_j(1 - V_j) \times \sum_k \left\{\left(\sum_{i=k} V_i - V_k\right) \times W_{jk}\right\}$$

(6)算出した誤差に基づいて，結合荷重と閾値を更新する．更新は以下の式に基づく．ただし，M はモーメンタム係数，$W_{jk}(t-1)$はニューロン j からニューロン k の結合荷重の前回の更新量，β はスレッショルド係数である．

$$\Delta W_{jk}(t) = M \times W_{jk}(t-1) + (1-M) \times V_j \left(\sum_{i=k} V_i - V_k \right)$$

$$\Delta \theta_j = \beta \times \left(\sum_{i=k} V_i - V_k \right)$$

(7)入力層から中間層への結合荷重と閾値の更新も，中間層の誤差を用いて同様に行う．

(8) E_m の平均 E を算出し，訓練フェーズの平均二乗誤差とする．

ここまでの手順を，用意した訓練データすべてに対して繰り返し行う．

次に試験フェーズを行う．

(9)試験データベクトルを読み込み，入力層のニューロンに値をセットする．

(10)中間層，出力層の値，出力層の誤差，平均二乗誤差を訓練フェーズと同様の手段で計算する．

(11)これをすべての試験データに対して繰り返し，平均二乗誤差の平均を得る．

以上の手順を反復試行回数だけ繰り返す．

結果

以ページ以降に実験結果を示す．３つの異なる設定による実験の結果と，入力層から中間層，中間層から出力層への結合荷重表である．

図中の数字左列は，試験データの一例である．訓練データに対してやや欠けたデータを意図的につくり，試験データとしている．中央列は試験データを訓練済ネットワークに与えたときの，出力層の値を再度 25 マスの形式に変換し直したものである．右列は，各数字の訓練データである．各セルに色が付いているのは，数値の高いところほど濃い橙で色を付け，結果を視認しやすくしたものである．

また，図の下の括弧内に示した各数値は，それぞれ以下のパラメータである．

HIDDEN：中間層のニューロン数
ALPHA：重みの学習係数
BETA：閾値の学習係数
MC：モーメンタム係数

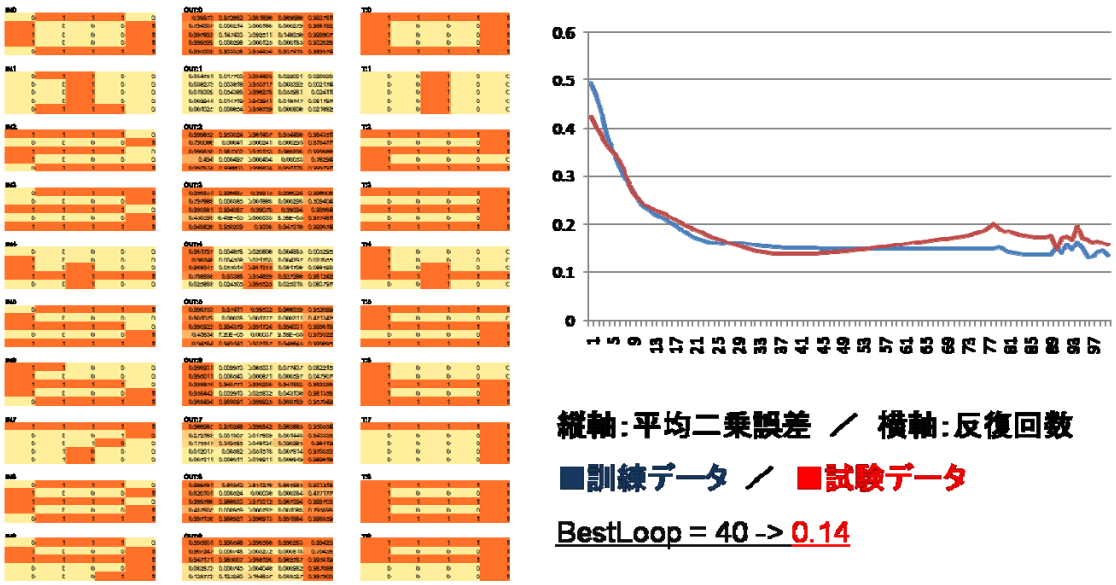


図 25　デジタル数字認識実験・結果１
(HIDDEN = 10 / ALPHA = 0.4 / BETA = 0.4 / MC = 0.8)

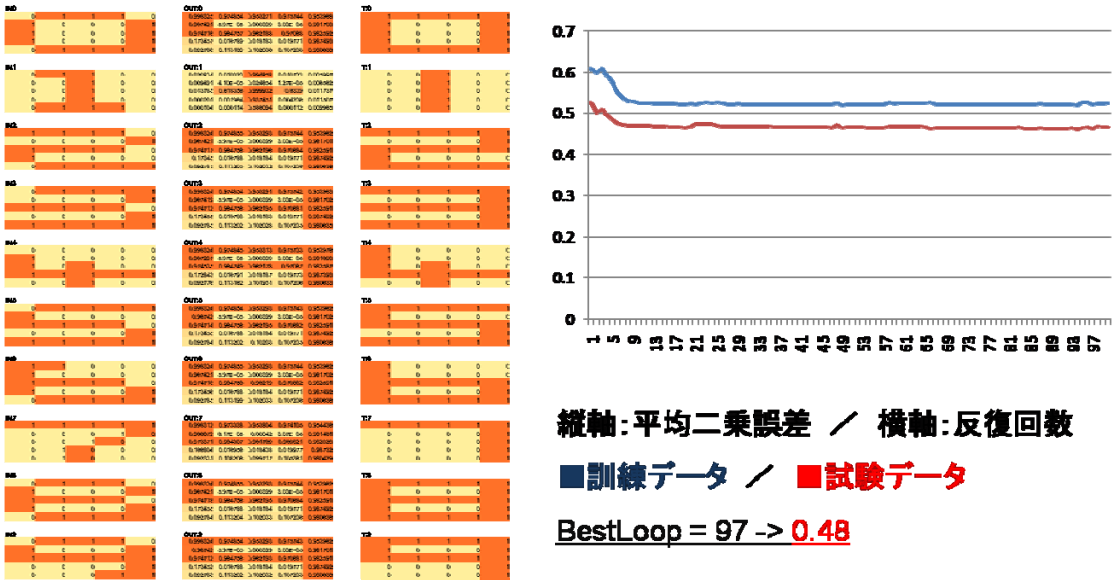


図 26　デジタル数字認識実験・結果２
(HIDDEN = 10 / ALPHA = 1.2 / BETA = 0.4 / MC = 0.8)

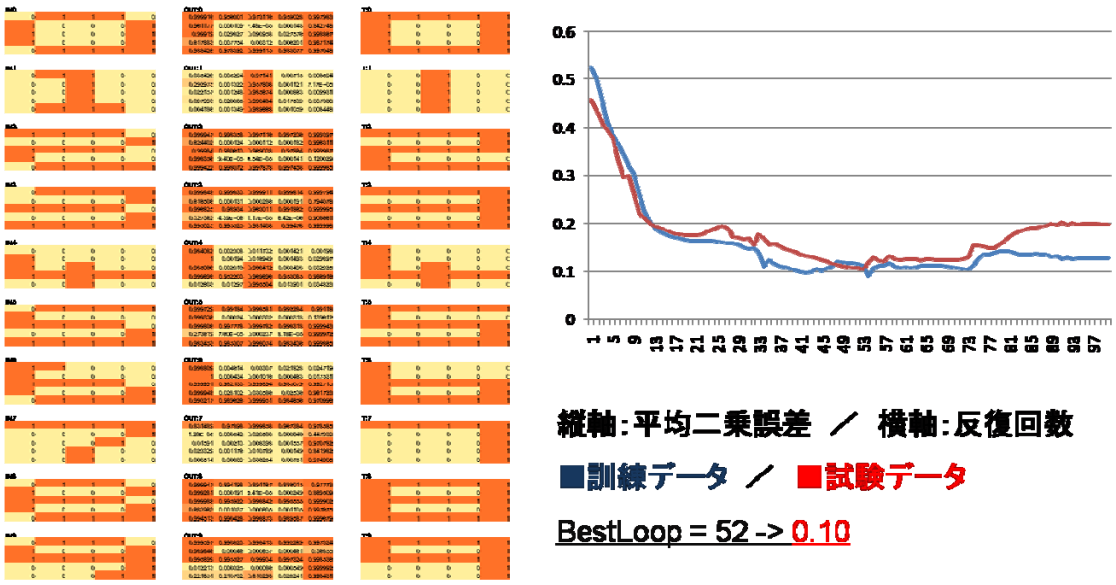


図 27　デジタル数字認識実験・結果３
(HIDDEN = 15 / ALPHA = 0.4 / BETA = 0.4 / MC = 0.8)

表 1　　実験 A：入力層から中間層への結合荷重表

IN−MIDDLE	26	27	28	29	30	31	32	33	34	35
1	0.278164	0.953837	0.784111	−1.01454	0.007429	−0.17219	0.911738	−1.41778	−0.98488	0.189272
2	−0.89088	0.174621	0.761123	0.211094	0.674637	0.661033	0.701575	−0.47377	−0.78569	−0.24024
3	−2.15809	0.976121	0.472104	−2.13229	−0.73655	0.436357	0.868904	−1.64655	−1.425	0.332808
4	−0.82912	0.206695	0.616317	−0.0194	−0.20753	0.733665	0.090033	−0.11262	−0.48579	−0.17835
5	−0.059	0.465484	0.617934	0.185704	−0.0847	0.046135	0.423072	0.136586	−0.15703	0.142116
6	5.247719	0.332881	0.310692	0.871102	−0.74098	−2.739	0.74814	−0.20089	−1.87801	0.153586
7	0.903046	0.522705	0.71817	0.841003	0.227142	0.657654	0.540131	0.047974	0.527069	0.177155
8	−1.3244	0.555528	0.378492	−1.36645	−0.28613	0.048547	1.253266	−0.9006	−0.48678	1.331225
9	0.834839	0.739532	0.837555	0.753021	0.940094	0.392426	0.052948	0.344482	0.173645	0.506042
10	0.372718	0.660604	0.372502	−1.00251	5.504935	8.910731	0.964379	−0.70785	6.139338	−5.92678
11	0.672397	0.414305	0.077063	1.376219	−0.58548	−0.52041	0.935291	0.767051	−0.30917	−0.11342
12	0.418166	0.15902	0.122009	3.335148	2.76651	−0.23976	0.166779	1.11635	0.246437	−0.00521
13	−1.06516	0.864552	0.769736	1.571341	0.399725	−1.114	0.93749	−1.71062	−1.77499	1.20215
14	−0.02834	0.651909	0.81784	2.819859	2.308716	0.006094	0.227936	0.288804	0.405434	0.256172
15	−0.63882	0.86734	0.237838	−0.59388	0.342819	0.505394	0.790747	−0.11479	−0.22151	−0.41758
16	3.454059	0.096029	0.288187	0.419763	−5.59934	1.04003	0.938464	1.504235	2.466025	0.316197
17	1.156901	0.701395	0.919809	−0.06859	0.10565	−0.44355	0.895161	−0.69813	−0.12919	0.465155
18	−0.96471	0.86874	0.898826	−1.63301	−1.43087	−0.80998	0.689626	−1.30305	−0.98371	1.710931
19	0.788483	0.929026	0.957742	0.391585	−0.20587	−0.08921	0.892036	−1.02571	−0.26328	1.001136
20	2.709026	0.878996	0.421773	−0.95465	0.380092	−0.64148	0.766168	−1.38072	−1.63931	−0.83787
21	−0.59161	0.606476	0.826463	0.402275	0.418975	−0.47848	0.82707	2.522269	2.053296	0.070383
22	−0.67861	0.364349	0.711564	0.322624	−0.22406	−0.4375	0.971144	2.481436	1.424664	0.941171
23	−1.61738	1.061672	1.210671	−1.58513	−2.06164	−1.17604	1.117801	0.317914	−0.38293	1.160281
24	−0.37246	0.885468	0.995469	0.431053	0.064299	−0.62598	0.95384	2.93981	1.85545	0.154825
25	−0.12065	0.670196	0.47972	−0.43283	0.686905	0.534141	0.898413	−0.15052	−0.58372	−0.10035

表 2　実験 A：中間層から出力層への結合荷重表

MIDDLE-OUT	1	2	3	4	5	6	7	8	9	10
26	3.411779	0.01943	-3.25319	-0.05012	0.475761	10.66944	-0.80337	-3.64522	-0.51762	1.628042
27	-0.01608	-0.46331	0.906556	-0.88072	-0.68076	-0.1445	-1.31264	0.038009	-1.69198	-0.21988
28	-0.54975	-0.05053	0.590713	-0.55186	-0.42568	0.238715	-0.95291	-0.18965	-0.89409	0.562779
29	2.04509	0.44885	-1.3776	0.291488	0.075875	3.258631	-1.35195	-2.18706	-0.9138	0.275077
30	1.539709	3.678014	3.893785	4.167484	4.236667	-1.9583	-0.60183	-1.81645	-0.63959	3.827376
31	1.349158	3.766516	4.062263	3.70364	3.400956	-1.2358	-0.687	-0.88377	-0.59188	18.00321
32	-0.58696	-0.67588	1.081556	-0.15147	-0.46367	0.099839	-1.29068	0.736407	-1.66386	0.272492
33	1.472185	1.13674	-0.05008	0.933798	0.78866	1.106973	-0.43686	-1.04801	-0.93949	0.758427
34	1.571569	1.958128	1.355247	1.608913	1.523353	-2.10031	-0.79458	-1.28213	-0.80762	11.45821
35	-0.97882	-2.57758	-0.60255	-2.19815	-2.35679	-1.2866	-1.44347	1.482562	-0.60582	-6.89292

MIDDLE-OUT	11	12	13	14	15	16	17	18	19	20
26	3.696104	0.142732	-0.54172	0.114201	0.770855	4.382825	2.632473	-0.67246	2.931899	5.890041
27	-0.80751	-1.6079	0.765261	-1.19028	-0.16776	-1.04834	-0.48712	0.245084	-0.80817	0.288127
28	-0.17458	-1.34312	0.609064	-0.89626	-0.69568	-0.52195	-0.2925	0.955449	-0.75493	0.182231
29	4.804632	7.359586	6.033861	6.746749	1.603636	1.414968	0.389205	-1.65889	0.114253	0.109223
30	-0.36402	3.360647	-0.23523	2.816217	3.102255	-6.16688	-2.76999	-3.3869	-2.41386	1.857251
31	-1.11119	-1.1369	-2.60477	-1.13955	1.948413	1.239471	-2.16155	-1.99127	-1.89141	-1.08305
32	-0.87786	-1.43852	0.267623	-1.88877	-0.68241	-1.49904	-1.29089	0.442921	-1.04823	0.63171
33	3.011038	2.550494	0.449035	2.206762	3.114355	3.510571	-2.95288	-3.051	-2.81647	-0.17571
34	0.20682	0.063207	-1.40699	-0.07356	1.779256	2.948466	-1.43429	-1.53681	-1.99362	-1.38996
35	-0.19216	-1.46873	1.977086	-1.62556	-2.17637	1.0624	0.58616	2.445342	1.002664	-2.00493

MIDDLE-OUT	21	22	23	24	25
26	-0.35514	0.085182	-0.60008	-0.10078	0.638411
27	-1.0076	-1.66929	0.631772	-0.97851	-0.42417
28	-1.18996	-0.73121	0.637943	-1.15857	-0.45527
29	1.960676	2.113206	1.184703	2.316364	1.626536
30	0.382953	0.109117	-2.29885	0.098967	2.984908
31	-1.23111	-0.95437	-2.57061	-1.47634	1.947388
32	-1.27303	-1.5052	0.506715	-1.04897	-0.56723
33	6.730364	5.278673	5.318617	6.661433	3.096813
34	2.501528	2.710259	0.704969	2.771642	2.03294
35	-0.56239	-0.3312	2.293407	-1.03075	-2.19331

考察

図 26 から，学習勾配が強いとネットワークは正しく訓練データから学習できないことがわかる． 図 26 において，1 以外のすべての数字が 9 と認識されているのは，デジタル数字における安定解が 9 だからであると考えられる．（9 の形状を含む数字は多く，安定解になりやすい.）

中間ニューロンの個数は，10 個でも十分訓練できていると見ることもできるが，15 個の方がより安定しており，これ以上増やしても精度に変わりはない．また後述する抽象化現象は，その他のパラメータが適切であれば，入力層ニューロン数の平方根程度の中間層ニューロンを用意することで可能になることが，経験的に確認されている．

また，それぞれの中間ニューロンがどのような働きをしているか，その結合荷重を確認すると， 図 28 のようなものがいくつも確認される．

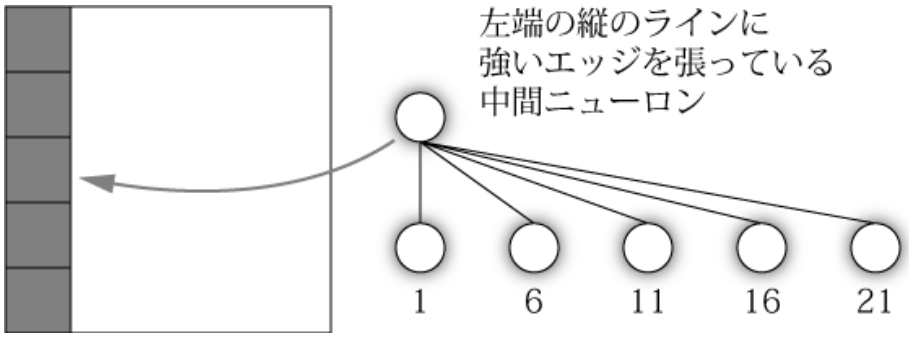


図 28　抽象化された中間ニューロン

表 3　抽象化されたニューロンの例（1）

MIDDLE-OUT	1	2	3	4	5	6	7	8	9	10
33	1.472185	1.13674	-0.05008	0.933788	0.78666	1.106973	-0.43686	-1.04601	-0.83949	0.758427
	11	12	13	14	15	16	17	18	19	20
	3.011038	2.550484	0.449035	2.206782	3.114355	3.510571	-2.95288	-3.061	-2.61647	-0.17571
	21	22	23	24	25					
	6.730364	6.278673	5.318617	6.661433	3.096813					

表 4　抽象化されたニューロンの例（2）

MIDDLE-OUT	1	2	3	4	5	6	7	8	9	10
34	1.571569	1.958128	1.355247	1.608913	1.523353	-2.10031	-0.79458	-1.28213	-0.80762	11.45821
	11	12	13	14	15	16	17	18	19	20
	0.20682	0.063207	-1.40699	-0.07356	1.779256	2.948466	-1.43429	-1.53681	-1.99362	-1.38996
	21	22	23	24	25					
	2.501528	2.710259	0.704969	2.771642	2.03294					

たとえば表 3 における中間ニューロン 33 番は，1,6,11,16,21,22,23,24,25 番の入力層ニューロンに対して強い値を張っている．これは左端の縦棒と，最下部の横棒を抽象化したニューロンであり，対象セルのうち数個の観測が確認されれば，図 29 のように残りの部分を予測する機能を果たしている．

図 29　抽象化された中間ニューロンによる観測予測

表 4 におけるニューロン 34 番も同様に，最上部と最下部の二本の棒線の抽象化を行っている．注目すべきは，このニューロンから 10 番ニューロンへの出力が有意に大きいことである．これは正しい予測であると言える．何故なら，訓練データを確認すればわかるように，最上部と最下部に棒線を持ち，10 番セルが白であるような数字データは，今回の場合は「5」を除いてひとつも存在しないからである．

このように，与えられたデータ自身を予測するという補助問題を解くことによって，ネットワークの中間層に位置するニューロンは，データに内在する構造を自動的に取り出して抽象化を行うことが確かめられた．（図 30）．

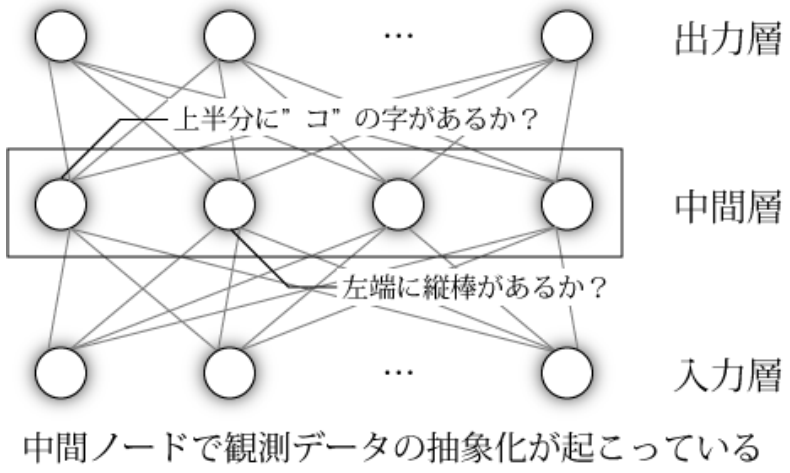


図 30　中間ノードによる観測データの抽象化

第 3 項 実験 B：手書き文字の認識

次に，既存の手書き文字データセットに対して実験を行った．実験の詳細は，入力層のニューロンの個数が 64 個であることを除いて，実験 A と同様である．アルゴリズム等は省略する．

以下に使用したデータセットの詳細を示す．

データセット名：Optical Recognition of Handwritten Digits
作者：E. Alpaydin, C. Kaynak
所属機関：Department of Computer Engineering Bogazici University, 80815 Istanbul Turkey

作成年：1998
データ数：3823（Training），1797（Test）

各数字データは 64 個の要素を持つベクトルで表され，各要素は 0～15 の「強度」値を持っている．手書き文字の濃淡が強度に反映している．実験ではこれを 0 から 1 に正規化して，実数値を入力層に与えた．

結果と考察

図 31，図 32 は，実験 A で得られた最適なパラメータを用いて行った実験の結果である．左列が試験データ，右列が訓練されたネットワークの出力である．ここで確認すべきことは，手書き文字に対してもデジタル文字同様，中間層による抽象化が行われていることであるため，結合荷重表など詳細なデータは省略する．

実験 A 同様，抽象化された中間層のニューロンはいくつも発見されたが，その多くは図 31 に示したような矩形を表現するように，入力層のニューロンへの軸索を張っていた．これは，図 32 から分かる通り，手書き文字の数字データはこのような矩形を組み合わせて表現できるものが多いからと考えられる．（図 31 の数値は，形を視認しやすくするために，0 に限りなく近い値は 0 に切り捨て，0.2～0.5 の範囲内に収まった数値はすべて 0.2 に切り捨てている．）

また図 32 の結果については，ネットワークの出力はそれほど視認性に優れているとは言えないが，試験データに対してその傾きを修正するなど，やはり予測と修正を行っていることが確認される．なお，この実験結果における中間ニューロンの個数は 20 であるが．

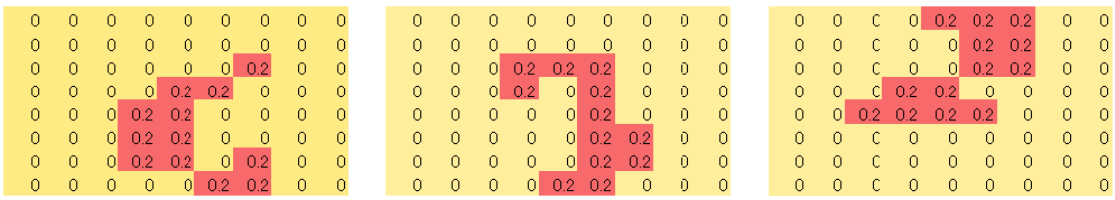


図 31　抽象化されたニューロンの例（3）

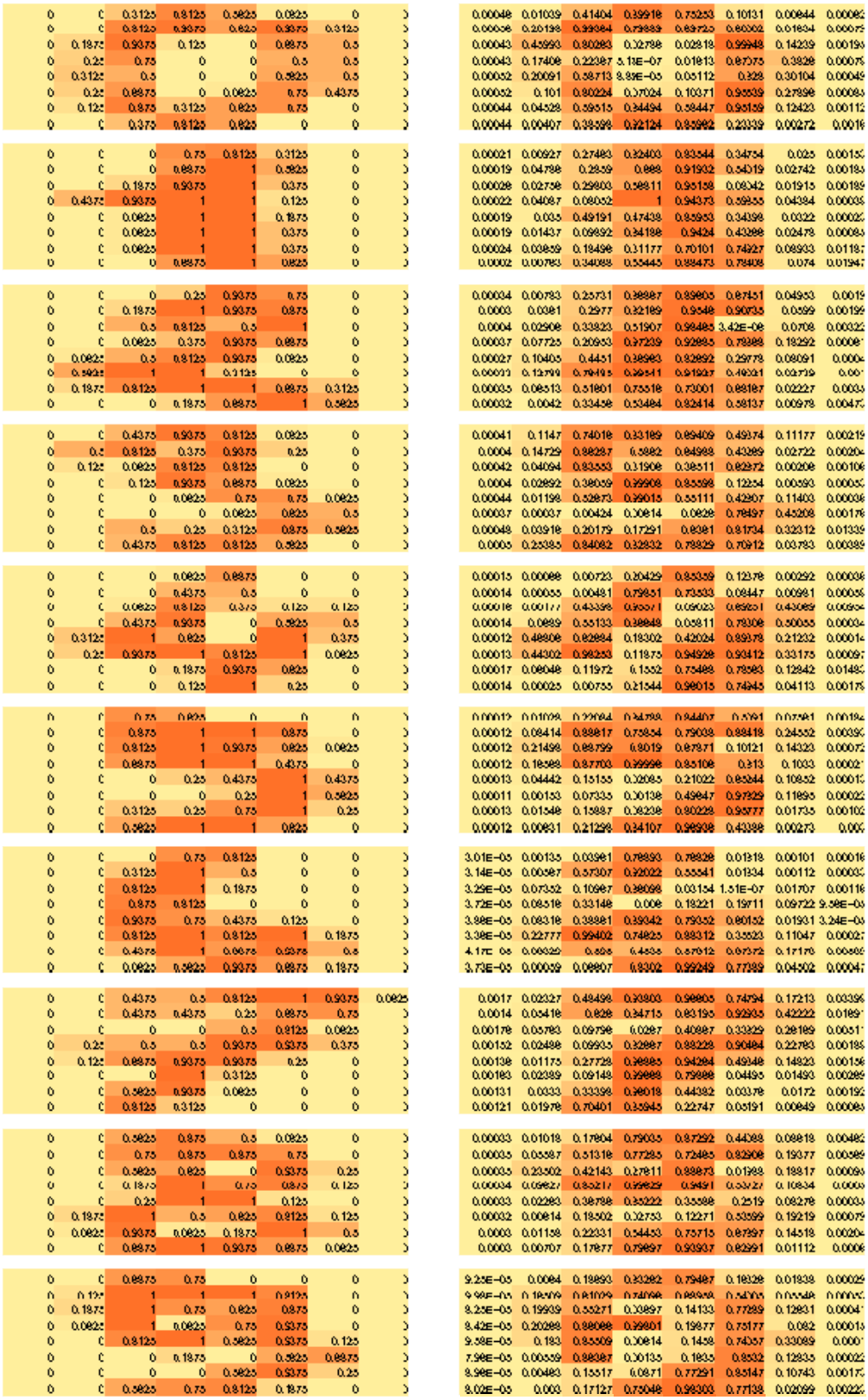


図 32　手書き文字の認識実験：結果

第４項 バックプロパゲーション実験に関する考察

バックプロパゲーションを用いたふたつの実験によって，「与えられたデータ自身を予測する」という補助問題を解くことにより，ネットワークの中間構造が自動的に抽象化・概念化されることを確認した．

このことから，ネットワークを用いた情報の自動構造化にあたって，上記の補助問題の生成方法は妥当であると言える．したがって以後構築していくすべてのアルゴリズムは，この補助問題を解くことによって自己構造化を行うものである．

第３節 Model 1. AI_Darwinism

第１項 概要

上述のバックプロパゲーションモデルでは，ネットワークは恣意的な三層のパーセプトロンが用いられ，重みの修正にも平均二乗誤差を用いるなど，ネットワークを外から眺めて構造を修正する，言わば「神」のような存在がいた．

しかし神経ダーウィニズムの考え方にしたがえば，適者生存，即ち最も環境に適合したものが生き残ることが進化であり，それゆえに脳の機能もまた，最適に動作するよう神によって設計されたわけではない．

ニューラルネットワークによる脳の持つ知能の再現は，もっとローカルな選択淘汰の繰り返しによって実現されうるはずである．したがってここからは，思考単位をニューロンに限定し，与えられた環境に対して適合するニューロンのみが生き残っていくことで，結果としてネットワーク構造が最適化されていくようなモデルを考える（図 33）．

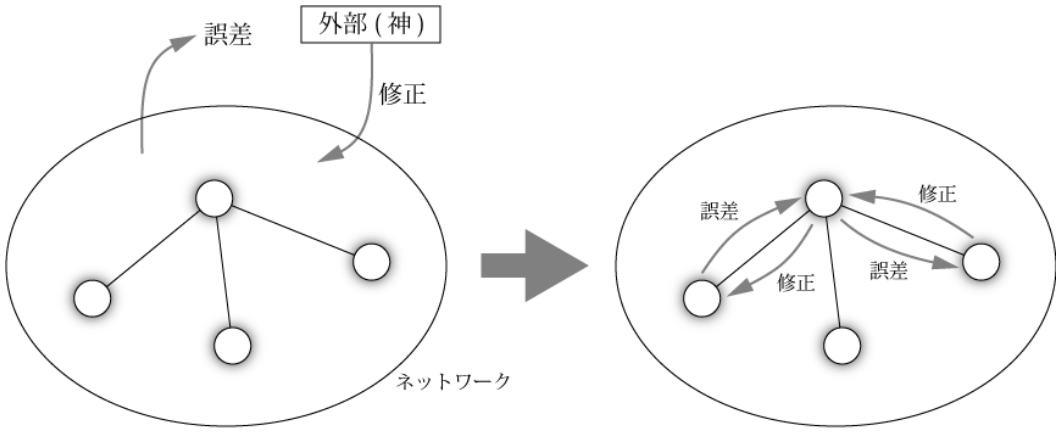


図 33　ローカルな選択淘汰による最適化・概略図

本節ではその最初の段階として，神経ダーウィニズムの原理のみで構造を更新していく，極めて単純な三層ネットワークを考える．

下の図 35 は，図 34 に再掲した時空間チューブを単位時間ごとにスライスし，並列に並べて入力層としたものである．以下，この層のニューロンをセンサーニューロンと呼ぶことにする．

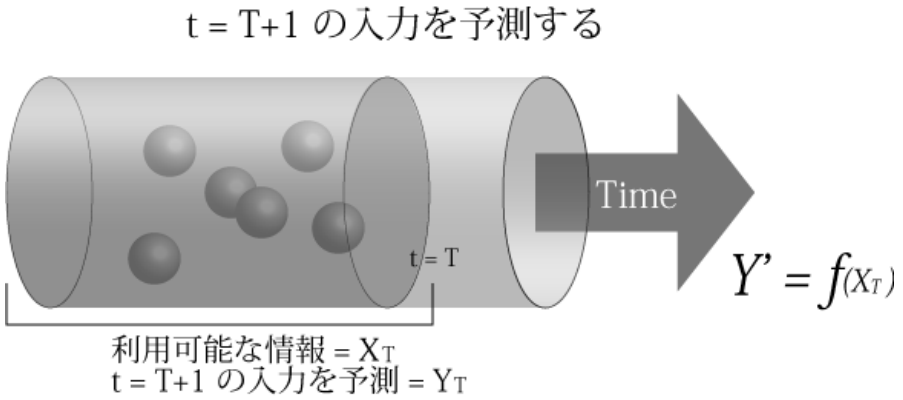


図 34　時空間チューブ上における予測の表現・再

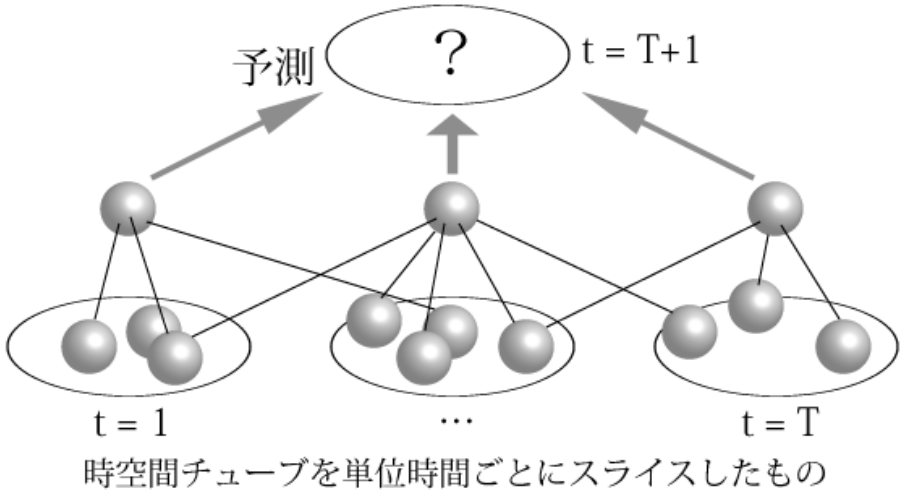


図 35　AI_Darwinism のネットワークモデル・イメージ

この構造は，過去に観測されたあらゆる情報を平等に入力として使うことができる，もっともシンプルなセンサーニューロンの構造である．この構造に対して，神経ダーウィニズムの原理に基づく最適化を試みる．

選択淘汰されるのは中間ニューロンである．したがって観測されたデータも予測するデータも，中間ニューロンにとっては外部の環境となる．中間ニューロンは，任意のセンサーニューロンからパルスを受け取って，自身が発火するかどうかを決定し，それによって出力ニューロンのひとつの発火を予測する．自身が発火すれば，次の単位時間でその出力ニューロンに対応するデータが観測されることを予測したことになる．

今回の実験は，データ自身を予測する補助問題のうち，特にデータの最終文字を予測する問題のみを切り出して補助問題としている．したがって，ひとつの訓練データに対してひとつの正解が与えられているのと，見かけ上は同じ恰好になっている．たとえば訓練データが{abcdef}という文字列であるとすると，ネットワークは{abcde}という観測がセンサーに与えられたとき，{f}を出力に予測しなければならない，ということである．

この予測には精度（Precision）と再現率（Recall）を定義することができる．また，精度と再現率からはF値が算出できる．それぞれの計算式を以下に示す．ただし，S_m は中間ニューロンが発火したときに 1，そうでないときに 0 を取り，T_m は出力ニューロンが発火したときに 1，そうでないときに 0 を取るものとする．また，m は訓練データの番号である．

$$Precision = \frac{\sum_m (S_m \times T_m)}{\sum_m S_m}, \quad Recall = \frac{\sum_m (S_m \times T_m)}{\sum_m T_m}$$

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

中間ニューロンの生き残りは，このF値の大小によって決定される．

F 値の大きい中間ニューロンは，それだけ出力の予測に貢献したということであり，ひいてはよいセンサーニューロンとよい出力ニューロンに軸索を伸ばしているということである．したがって，その中間ニューロンは与えられた環境に対して，よいアプローチをしていると見なすことができる．このニューロンが生き残ることによって，ネットワーク構造は最適構造に近づいていくはずである．

生き残ったニューロンに付随する軸索は，受け取っているものも伸ばしているものもすべて次世代に受け継がれる．そうして，死滅したニューロンはふたたびランダムに軸索を形成する．どのセンサーニューロンからパルスを受け取るか，どの出力ニューロンを予測するか，いずれもランダムである．構造最適化にあたっては，完全に神経ダーヴィニズムによる適者生存の原理のみが適用される（図 36）．

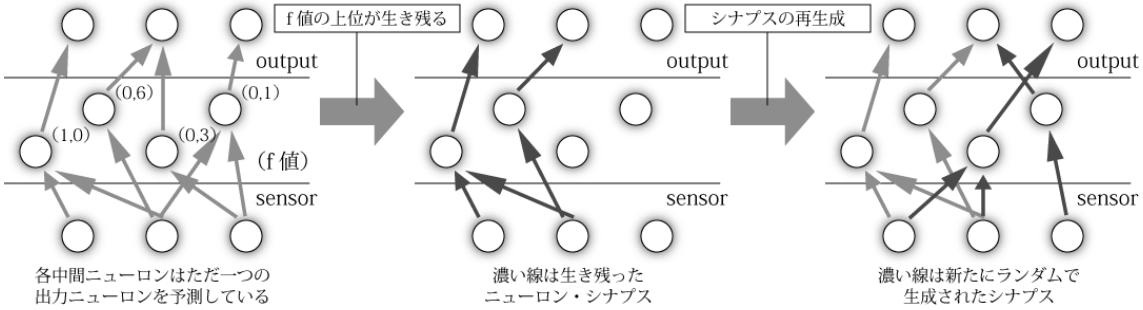


図 36　AI_Darwinism の世代プロセス概略図

第２項 アルゴリズム

AI_Darwinism の初期ネットワーク形成から構造最適化までのアルゴリズムを示す.

(1) 初期ネットワークをランダムな確率に基づき形成する. 中間ニューロンは軸索をセンサーニューロンに対してランダムに，出力ニューロンに対してはただ一本，軸索を形成する. このとき形成される軸索の重みは, -1 から 1 のあいだでランダムな値を取る. またアルゴリズムの表記において，以下 NoSynapse は，中間ニューロンが受け取る軸索を持たない場合，または出力する軸索を持たない場合に真であるブール変数とする.

```
For All Middle Neuron
 Do
  CreateSynapseProbability = Random(0-1)
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse from Sensor Neuron
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse To Output Neuron (Only Once)
 While(NoSynapse)
```

(2) 各中間ニューロンについて, 軸索を張っている先の出力ニューロンに対する予測の精度と再現率を計算し，F 値を算出する.

(3) F 値の高い中間ニューロンのみを生き残らせ，残りのすべての中間ニューロンはふたたびランダムに軸索を張り直す. また，自身の閾値も-1 から 1 のあいだでランダムに取る.

(4) (1)～(3)を一世代として，試行を繰り返す.

第３項 実験

上記アルゴリズムのコーディングを行い，プログラムに対して実際にデータを与えて実験を行った．そのパラメータ設定及びデータ，実験の意義，結果を以下に示す．

また，以下本章の実験にはすべて｛用いた実験データ，用いたネットワーク，設定｝のセットに対して通し番号を付け，結果を示す表のキャプションや本文中での言及の際には，実験番号を示す．

実験：1

表 5　AI 実験データ 1

[Feature]	[Label]
abecd	a
dcbea	b
deabc	c
dbcea	d
daded	e
bbcea	a
caceb	b
addbc	c
eeeac	d
eacbd	e

実験の意義

五キャラクター，五文字から成る，ほとんどランダムな文字列から，六文字目を予測する問題を解く．この問題はデータの構造上，確実に予測出来る問題である．

神経ダーウィニズムの原理のみによっても，十分な回数反復試行を繰り返した後には，すべての中間ニューロンの F 値が 1 に収束するかどうかを確認する．

パラメータ設定

ITERATION: 500

ALL NEURON / MIDDLE NEURON: 50 / 20

SURVIVAL RATE: 0.5

以下すべての実験において，ITERATION は反復試行回数，All NEURON/MIDDLE

NEURON は全体のニューロン数／中間ニューロン数，SURVIVAL RATE は全体の何%が次世代に生き残るかを表すパラメータである．また，F 値表における数値は，左列が反復試行回数，その他各セルには「ニューロン ID;F 値」が表示されている．

結果

表 6　実験 1：反復初期の F 値表

	TOP60% of F-Value (NEURON-NUM F-Value)									
1	0;0	0;0	0;0	0;0	0;0	0;0	0;0	0;0	0;0	0;0
2	42;0.6666667	34;0.4444444	36;0.4444444	35;0.4	43;0.4	46;0.4	40;0.3636364	45;0.3636364	32;0.3333333	37;0.3333333
3	30;0.8	33;0.8	42;0.6666667	34;0.4444444	36;0.4444444	45;0.4444444	35;0.4	41;0.4	43;0.4	46;0.4
4	30;0.8	33;0.8	42;0.6666667	34;0.4444444	36;0.4444444	46;0.4444444	35;0.4	41;0.4	43;0.4	46;0.4
5	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	34;0.4444444	30;0.4444444	44;0.4444444	45;0.4444444	35;0.4
6	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	49;0.5	34;0.4444444	35;0.4444444	44;0.4444444	45;0.4444444
7	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	49;0.5	32;0.4444444	34;0.4444444	38;0.4444444	44;0.4444444
8	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	37;0.5	45;0.5	45;0.5	32;0.4444444	34;0.4444444
9	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	37;0.5	45;0.5	45;0.5	32;0.4444444	34;0.4444444
10	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	37;0.5	45;0.5	45;0.5	32;0.4444444	34;0.4444444
11	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	37;0.5	41;0.5	45;0.5	48;0.5	49;0.5
12	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	35;0.5	37;0.5	41;0.5	45;0.5	48;0.5
13	30;0.8	33;0.8	39;0.6666667	42;0.6666667	31;0.5	35;0.5	37;0.5	41;0.5	44;0.5	45;0.5
14	30;0.8	33;0.8	39;0.6666667	42;0.6666667	43;0.5714286	47;0.5714286	31;0.5	35;0.5	37;0.5	41;0.5
15	30;0.8	33;0.8	39;0.6666667	42;0.6666667	43;0.5714286	47;0.5714286	31;0.5	35;0.5	37;0.5	41;0.5
16	30;0.8	33;0.8	39;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	45;0.5714286	47;0.5714286	31;0.5	36;0.5
17	30;0.8	33;0.8	39;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	45;0.5714286	47;0.5714286	31;0.5	36;0.5
18	30;0.8	33;0.8	32;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	46;0.5714286	47;0.5714286
19	30;0.8	33;0.8	32;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	46;0.5714286	47;0.5714286
20	30;0.8	33;0.8	32;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	46;0.5714286	47;0.5714286
21	30;0.8	33;0.8	32;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	46;0.5714286	47;0.5714286
22	30;0.8	33;0.8	32;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	46;0.5714286	47;0.5714286
23	30;0.8	33;0.8	32;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286	46;0.5714286	47;0.5714286
24	35;1	30;0.8	33;0.8	32;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	43;0.5714286	43;0.5714286	46;0.5714286
25	35;1	30;0.8	33;0.8	32;0.6666667	36;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286
26	35;1	30;0.8	33;0.8	32;0.6666667	36;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286
27	35;1	30;0.8	33;0.8	32;0.6666667	36;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286	43;0.5714286
28	35;1	30;0.8	33;0.8	34;0.8	32;0.6666667	36;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286
29	35;1	30;0.8	33;0.8	34;0.8	32;0.6666667	36;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286
30	35;1	30;0.8	33;0.8	34;0.8	32;0.6666667	36;0.6666667	39;0.6666667	41;0.6666667	42;0.6666667	40;0.5714286

表 7　実験 1：F 値の完全収束

371	34:1	35:1	43:1	44:1	46:1	47:1	48:'	49:1	30:0.8	31:0.8
372	34:1	35:1	43:1	44:1	46:1	47:1	48:'	49:1	30:0.8	31:0.8
373	34:1	35:1	37:1	43:1	44:1	46:1	47:'	48:1	49:1	30:0.8
374	34:1	35:1	37:1	43:1	44:1	46:1	47:'	48:1	49:1	30:0.8
375	34:1	35:1	37:1	43:1	44:1	46:1	47:'	48:1	49:1	30:0.8
376	34:1	35:1	37:1	43:1	44:1	46:1	47:'	48:1	49:1	30:0.8
377	33:1	34:1	35:1	37:1	43:1	44:1	46:'	47:1	48:1	49:1
378	33:1	34:1	35:1	37:1	43:1	44:1	46:'	47:1	48:1	49:1
379	33:1	34:1	35:1	37:1	43:1	44:1	46:'	47:1	48:1	49:1
380	33:1	34:1	35:1	37:1	43:1	44:1	46:'	47:1	48:1	49:1

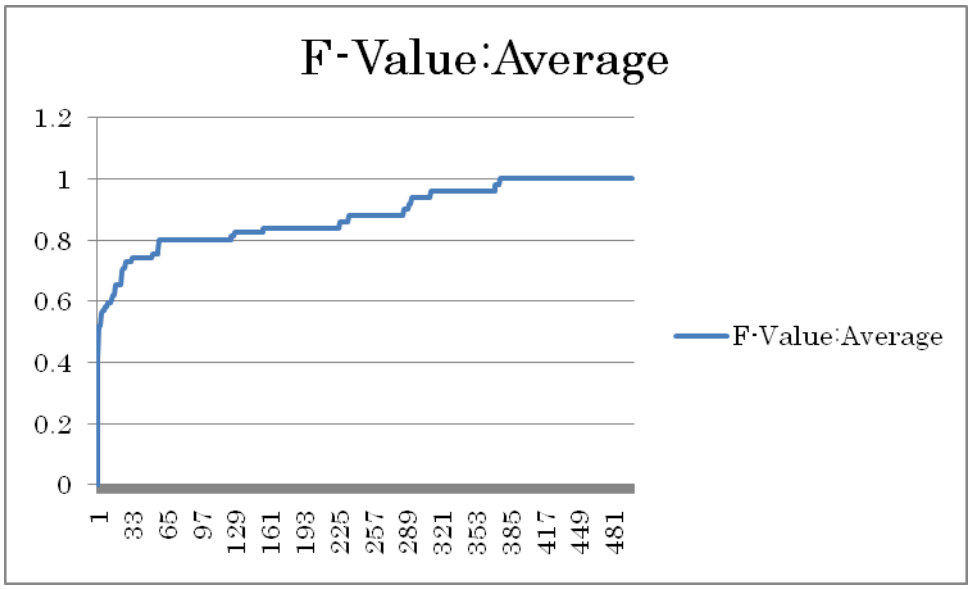


図 37　実験 1：ニューロン F 値平均の収束

考察

表 6 から，F 値の高いニューロンが保存され，F 値の低いニューロンが徐々に淘汰されていく様子がわかる．また 500 回程度の反復試行の後，すべての上位 50%の中間ニューロンの F 値が 1 に収束していることが確認出来る．

このことが示しているのは，各中間ニューロンが各々，自分が軸索を形成している先の出力ニューロンを，センサーニューロンの発火パターンから完全に予測できるようになったということである．したがって，ここに F 値によるニューロンの完全淘汰（F 値の低いニューロンは軸索のみならず，ニューロン自体を抹消する）というロジックを導入することによって，ネットワーク構造がこの問題に対しては完全に最適化されることが明らかである．

第４節 Model 2. AI_RewardBack

第１項 概要

前節の実験によって，神経ダーウィニズムによる適者生存の原理のみによってもネットワークは最適化されうることが確認された．次に，より賢い進化のためのアルゴリズムを考える．

AI_Darwinism モデルでは，F 値の高いニューロンは生き残り，低いニューロンは軸索をランダムに張りなおしていたが，これは突然変異のみで最適構造を探索する遺伝的アルゴリズムに近い発想であり，収束に極めて時間のかかることが容易に予想される．この適者生存の原理の上に，一世代間で何らかの構造の修正・更新ロジックを持ち込むことで，より早い進化ができるはずである．

AI_RewardBack はその可能性のひとつの実現形態である．その概要は，重み更新に基づく三層モデルと定義できる．

ネットワークの基本的な構成及び，選択淘汰のロジックは，AI_Darwinism と変わりない．しかし AI_RewardBack では，淘汰が行われる直前に，センサーニューロンから中間ニューロンへの軸索の重みを，中間ニューロンの予測精度に応じて更新するプロセスが追加されている．

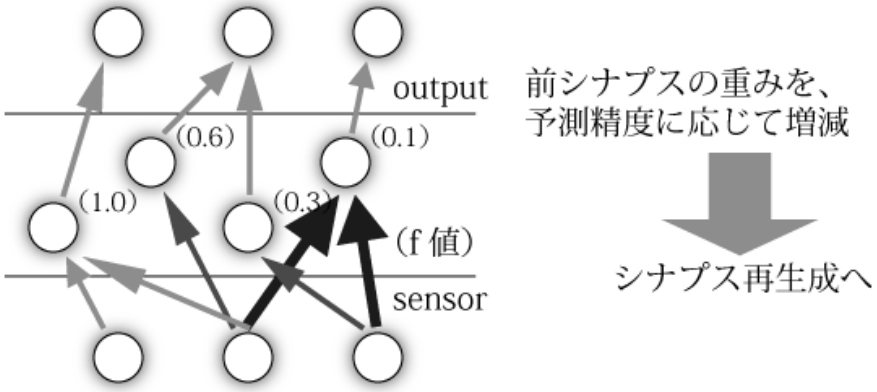


図 38　AI_RewardBack の重み更新

中間ニューロンは出力ニューロンに対する予測の誤差（1 or −1）を受け取り，発火したセンサーニューロンから受け取っている後方軸索の重みを増減させる．発火しなかったセンサーニューロンは，予測の結果に無関係であり，したがって軸索の重みは更新しない．

予測において中間ニューロンが勇み足をした場合（出力は 0 なのに自身は発火した）

には，自身に値を送ってきた軸索の重みを減じ，取りこぼした場合（出力は 1 なのに自身は発火しなかった）には，自身に値を送ってきた軸索の重みを増す．重みの増減幅は軸索の重み更新勾配パラメータによって決定する．

このように AI_RewardBack は，神経ダーウィニズムによる選択淘汰に加えて，生存する適者の環境に対する適性が徐々に補正されていくモデルであるといえる．

第２項 アルゴリズム

AI_RewardBack の初期ネットワーク形成から構造最適化までのアルゴリズムを示す．

(1) 初期ネットワークをランダムな確率に基づき形成する．中間ニューロンは軸索をセンサーニューロンに対してランダムに，出力ニューロンに対してはただ一本，軸索を形成する．このとき形成される軸索の重みは，-1 から 1 のあいだでランダムな値を取る．

```
For All Middle Neuron
 Do
  CreateSynapseProbability = Random(0-1)
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse from Sensor Neuron
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse To Output Neuron (Only Once)
 While(NoSynapse)
```

(2) 各中間ニューロンについて，軸索を張っている先の出力ニューロンに対する予測の精度と再現率を計算し，F 値を算出する．

(3) 各中間ニューロンについて，接続先出力ニューロンの値（発火したかどうか）を受け取り，自身の値（発火したかどうか）と比較して，以下のアルゴリズムに従い後方軸索の重みを更新する．

```
For All Middle Neuron
 IF Output has Fired
  IF This has NOT Fired
   Modify Weight (+Bias*AxonWeight) for All Received Axon
 IF Output has NOT Fired
```

```
IF This has Fired
  Modify Weight (-Bias*AxonWeight) for All Received Axon
```

(4) F 値の高い中間ニューロンのみを生き残らせ，残りのすべての中間ニューロンはふたたび(1)のようにランダムに軸索を張り直す．また，自身の閾値も-1 から 1 のあいだでランダムに取る．
(5) (1)～(4)を一世代として，試行を繰り返す．

第３項 実験

実験 2

表 5 に示した実験データ 1 の問題を，AI_RewardBack モデルで解く．基本的なパラメータの設定は，実験 1 と同様である．

実験の意義

実験 1 と同様の設定下で，AI_RewardBack モデルを用いることによって，ネットワークの最適化＝F 値の収束の速度が早くなることを確認する．

パラメータ設定

ITERATION: 500
ALL NEURON / MIDDLE NEURON: 50 / 20
SURVIVAL RATE: 0.5
AXON MODIFY RATE: 0.2
以下，AXON MODIFY RATE は軸索の重み更新勾配を表すパラメータである．

結果

次頁に示す．

考察

表 8 から，20 回程度の反復試行の後，すべての上位 50%の中間ニューロンの F 値が 1 に収束していることが確認出来る．また，図 37 と図 39 とを比較すれば，同じ実験設定下では AI_Darwinism モデルよりも AI-RewardBack モデルの方が，遥かに収束の早いことが確認できる．一世代のあいだに後方軸索の重みを更新できることが，ネットワーク全体の構造最適化に対して極めて有効に機能しているといえる．

表 8　実験 2：F 値の完全収束

TOP50% of F-Value (NEURON-NUM;F-Value)

1	32;0.3636364	35;0.3636364	46;0.3636364	48;0.3636364	30;0.3333333	31;0.3333333	33;0.3333333	34;0.3333333	36;0.3333333	38;0.3333333
2	32;0.5714286	30;0.4	31;0.4	38;0.4	46;0.4	33;0.3636364	36;0.3636364	48;0.3636364	34;0.3333333	37;0.3333333
3	32;0.8	33;0.6666667	38;0.6666667	48;0.5714286	31;0.5	46;0.5	34;0.4444444	35;0.3333333	36;0.3333333	37;0.3333333
4	32;1	46;1	31;0.8	38;0.8	48;0.6666667	33;0.5	34;0.4	30;0.3333333	36;0.3333333	37;0.3333333
5	32;1	33;1	38;1	46;1	31;0.8	48;0.8	34;0.5	36;0.4	30;0.3333333	35;0.3333333
6	31;1	32;1	33;1	38;1	46;1	48;1	34;0.8	36;0.6666667	30;0.3333333	35;0.3333333
7	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	30;0.3333333	35;0.3333333
8	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	35;0.3636364	30;0.3333333
9	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	35;0.3636364	30;0.3333333
10	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	30;0.3333333	37;0.3333333
11	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	37;0.3636364	30;0.3333333
12	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	37;0.4444444	30;0.3333333
13	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	37;0.5714286	30;0.3333333
14	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	37;0.6666667	30;0.3333333
15	31;1	32;1	33;1	34;1	36;1	38;1	46;1	48;1	37;0.8	39;0.3636364
16	31;1	32;1	33;1	34;1	36;1	37;1	38;1	46;1	48;1	39;0.4444444
17	31;1	32;1	33;1	34;1	36;1	37;1	38;1	46;1	48;1	39;0.6666667
18	31;1	32;1	33;1	34;1	36;1	37;1	38;1	46;1	48;1	39;0.8
19	31;1	32;1	33;1	34;1	36;1	37;1	38;1	39;1	46;1	48;1
20	31;1	32;1	33;1	34;1	36;1	37;1	38;1	39;1	46;1	48;1

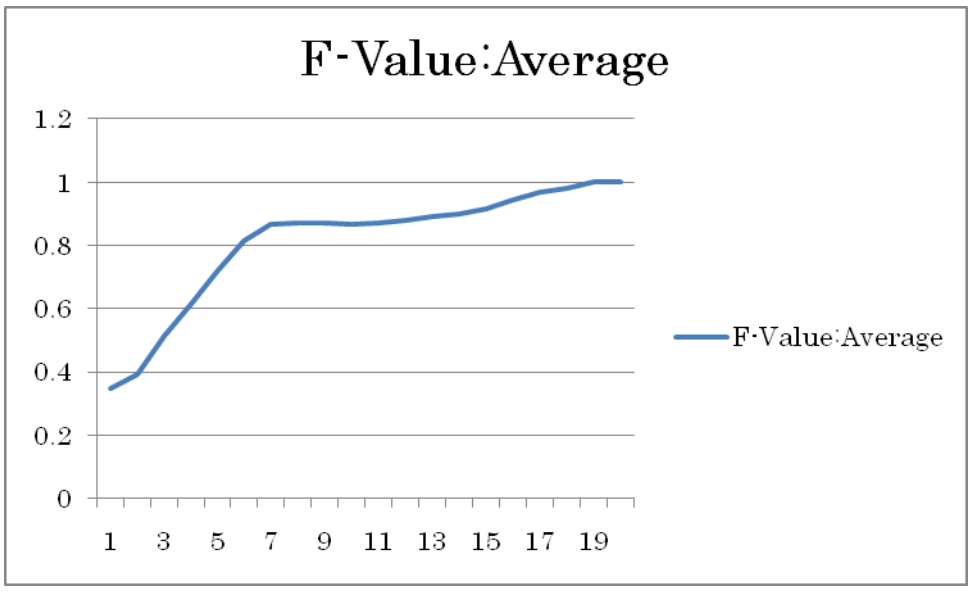


図 39　実験 2：ニューロン F 値平均の収束

実験 3

表 9　AI 実験データ 2

[Feature]	[Label]
Aa	a
ab	b
ba	b
bb	a

実験の意義

二キャラクター，二文字から成る文字列から，三文字目を予測する．この問題は三層構造では解くことのできない構造をしている．ダーウィニズム，リワードバック，どちらのモデルでも，ついに中間ニューロンの F 値が 1 に収束出来ないことを確認する．

パラメータ設定

ITERATION: 300
ALL NEURON / MIDDLE NEURON: 20 / 12
SURVIVAL RATE: 0.5
AXON MODIFY RATE: 0.2

結果

次頁に示す．

考察

表 10，表 11 から，学習は進行するが，反復試行が進んでも F 値は 1 に収束しないことが確認される．値が 0.8 に収束しているのは，上記の問題設定では F 値の最大は（Recall：1.0，Precision：0.66／（2*1.0*0.66/1.66）＝0.80）となるからである．

一方リワードバックでは 0.66 が大勢を占めます．これは重みの更新によりどちらのセンサーも予測しようとして（Recall:1.0，Precision：0.5）になっているため，（2*1.0*0.5/1.5）＝0.66 となっていると考えられます．

この実験結果から，分析性能を高めるためには三層以上の構造を持つマルチレイヤーモデルを必要とすることがわかる．

次節では，まず単純に多層構造を許したモデルである AI-SimpleMultiLayer モデル

により，上記の問題が解けることを確認する．

表 10　実験 3：反復初期の F 値表

1	0;0	0;0	0;0	0;0	0;0	0;0	0;0
2	17;0.8	6;0.6666667	11;0.6666667	15;0.6666667	16;0.6666667	18;0.6666667	19;0.6666667
3	17;0.8	6;0.6666667	10;0.6666667	11;0.6666667	14;0.6666667	15;0.6666667	16;0.6666667
4	7;0.8	17;0.8	6;0.6666667	10;0.6666667	11;0.6666667	12;0.6666667	14;0.6666667
5	7;0.8	17;0.8	6;0.6666667	10;0.6666667	11;0.6666667	12;0.6666667	14;0.6666667
6	7;0.8	17;0.8	6;0.6666667	8;0.6666667	10;0.6666667	11;0.6666667	12;0.6666667
7	7;0.8	13;0.8	17;0.8	6;0.6666667	8;0.6666667	10;0.6666667	11;0.6666667
8	7;0.8	13;0.8	17;0.8	6;0.6666667	8;0.6666667	10;0.6666667	11;0.6666667
9	7;0.8	13;0.8	17;0.8	6;0.6666667	8;0.6666667	9;0.6666667	10;0.6666667
10	7;0.8	13;0.8	17;0.8	6;0.6666667	8;0.6666667	9;0.6666667	10;0.6666667
11	7;0.8	11;0.8	13;0.8	17;0.8	6;0.6666667	8;0.6666667	9;0.6666667
12	7;0.8	11;0.8	13;0.8	17;0.8	18;0.8	6;0.6666667	8;0.6666667
13	7;0.8	11;0.8	13;0.8	17;0.8	18;0.8	6;0.6666667	8;0.6666667
14	7;0.8	11;0.8	13;0.8	17;0.8	18;0.8	6;0.6666667	8;0.6666667
15	7;0.8	11;0.8	13;0.8	17;0.8	18;0.8	6;0.6666667	8;0.6666667
16	7;0.8	11;0.8	13;0.8	17;0.8	18;0.8	6;0.6666667	8;0.6666667
17	7;0.8	11;0.8	13;0.8	17;0.8	18;0.8	6;0.6666667	8;0.6666667
18	7;0.8	11;0.8	13;0.8	15;0.8	17;0.8	18;0.8	6;0.6666667
19	7;0.8	11;0.8	13;0.8	15;0.8	17;0.8	18;0.8	6;0.6666667
20	7;0.8	8;0.8	11;0.8	13;0.8	15;0.8	17;0.8	18;0.8
21	7;0.8	8;0.8	11;0.8	13;0.8	15;0.8	17;0.8	18;0.8

表 11　実験 3：反復後期の F 値表

293	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**
294	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**
295	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**
296	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**
297	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**
298	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**
299	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**
300	**6;0.8**	**7;0.8**	**8;0.8**	**9;0.8**	**10;0.8**	**11;0.8**	**12;0.8**

第５節 AI_MultiLayer

第１項 概要

前節までの実験により，神経ダーウィニズムと重みの更新がネットワークの最適化に役立つことを示したが，三層構造では解けない問題のあることが明らかになった．実験データ 2 を完全に予測できるようなネットワーク構造は，以下のようなものである．

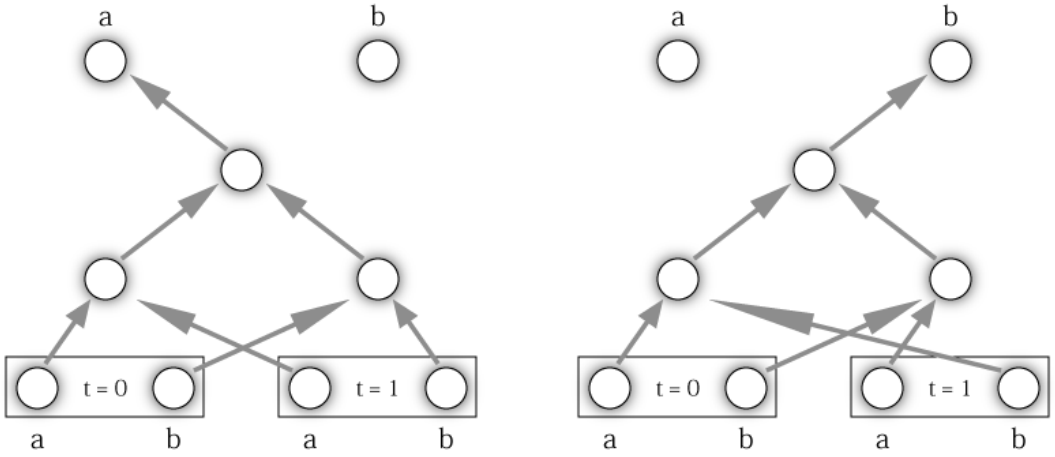


図 40　三層では解けない予測問題の解法

このような種類の問題を解くために，本節ではこれまでの議論に加えて，多層構造を形成するようなネットワークモデルを考える．多層構造の作り方にはいくつかの方策が考えられるが，ここでは３つに大別し，実装して同一の実験データにあてることで，それぞれ特徴を確認する．

Model 3-a. AI_SimpleMultiLayer

AI_SimpleMultiLayer は，重み更新に基づく多層モデルである．軸索をセンサーに対してランダムに，中間ニューロンからランダムに，出力に対してただ一本張り，予測精度にしたがってリワードを与えることで，センサーから中間ニューロンへの重みを更新する．その後，予測精度の高かった中間ニューロンのみを生き残らせ，残りのすべての中間ニューロンはふたたびランダムに軸索を張る．これを一世代として，試行を繰り返す．

AI_SimpleMultiLayer では，生き残った中間ニューロンがランダムに軸索を張り直す際，軸索を中間ニューロンから受け取ることを許可する．これによって，一世代目では三層構造であったネットワークが，世代を下るごとに多層化されていく．かつ，中間構造にループ構造を生じない．（図 41）

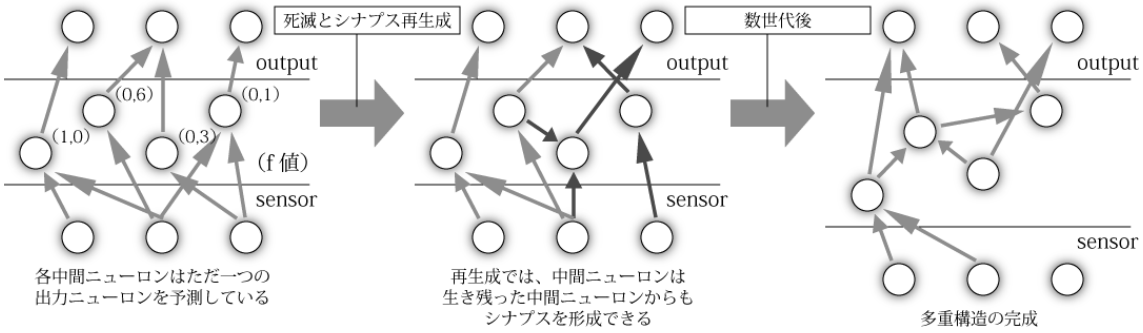


図 41　AI_SimpleMultiLayer の多層化ロジック

Model 3-b. AI_CleverMultiLayer

AI_CleverMultiLayer は，重み更新に基づく多層モデルである．軸索をセンサーに対してランダムに，中間ニューロンからランダムに，出力に対してただ一本張り，予測精度にしたがってリワードを与えることで，センサーから中間ニューロンへの重みを更新する．このとき，AI_CleverMultiLayer モデルでは，中間ニューロンは受け取ったリワードを後方のニューロンに伝達する．つまり，他の中間ニューロンと出力ニューロンにそれぞれ軸索を形成している中間ニューロンは，出力ニューロンから直接もらうリワードに加えて，他の中間ニューロンを経由してもリワードをもらうことになる．これは，自分自身が出力ニューロンを正しく予測できていなくても，出力ニューロンを正しく予測している中間ニューロンに対して，その予測の役に立っていれば，生き残ることができることを示している．

その後，スコアの高かった中間ニューロンのみを生き残らせ，残りのすべての中間ニューロンはふたたびランダムに軸索を張る．これを一世代として，試行を繰り返す．

Model 3-c. AI_PerfectMultiLayer

AI_PerfectMultiLayer は，重み更新に基づく多層モデルである．軸索をセンサーに対してランダムに，中間ニューロンからランダムに，出力に対して一本まで張ることを許し（すなわち張らなくても良い），予測精度にしたがってリワードを与えることで，センサーから中間ニューロンへの重みを更新する．リワードは AI_CleverMultiLayer と同様，後方の中間ニューロンにも伝達する．

その後，スコアの高かった中間ニューロンのみを生き残らせ，残りのすべての中間ニューロンはふたたびランダムに軸索を張る．これを一世代として，試行を繰り返す．

AI_PerfectMultiLayer の多層化ロジックを図示すると，図 42 のようになる．この

多層化においては，中間構造にループ構造が生成しうるため，計算時にはループを適切に処理しなければならない．ループにおける計算処理については，アルゴリズムの項目で後述する．

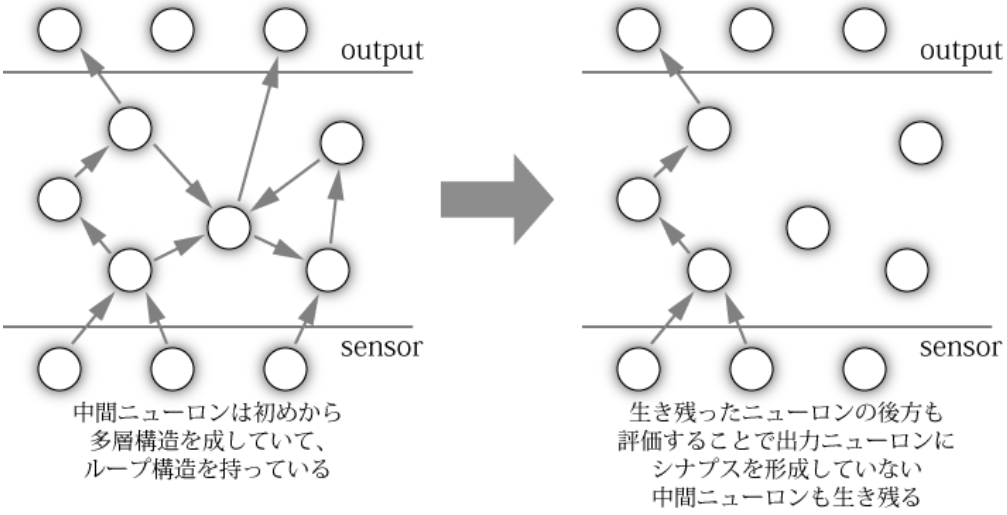


図 42　AI_PerfectMultiLayer の多層化ロジック

第２項 アルゴリズム

Model 3-a. AI_SimpleMultiLayer

(1) 初期ネットワークをランダムな確率に基づき形成する．中間ニューロンはセンサーニューロンまたは中間ニューロンからランダムに，出力ニューロンに対してはただ一本，軸索を形成する．このとき形成される軸索の重みは，−1 から 1 のあいだでランダムな値を取る．

```
For All Middle Neuron
 Do
  CreateSynapseProbability = Random(0-1)
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse from All Neuron Except Output Neuron
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse To Output Neuron (Only Once)
 While(NoSynapse)
```

(2) 各中間ニューロンについて，軸索を張っている先の出力ニューロンに対する予測の精度と再現率を計算し，F 値を算出する．
(3) 各中間ニューロンについて，接続先出力ニューロンの値（発火したかどうか）を受け取り，自身の値（発火したかどうか）と比較して，以下のアルゴリズムに従い後方軸索の重みを更新する．

```
For All Middle Neuron
 IF Output has Fired
  IF This has NOT Fired
   Modify Weight (+Bias*AxonWeight) for All Received Axon
 IF Output has NOT Fired
  IF This has Fired
   Modify Weight (-Bias*AxonWeight) for All Received Axon
```

(4) F 値の高い中間ニューロンのみを生き残らせ，残りのすべての中間ニューロンはふたたび(1)のようにランダムに軸索を張り直す．また，自身の閾値も-1 から 1 のあいだでランダムに取る．
(5) (1)～(4)を一世代として，試行を繰り返す．

Model 3-b. AI_CleverMultiLayer

(1)～(3)は AI_SimpleMultiLayer と同様の手順を行う．
(4) 各中間ニューロンの F 値に基づいて，以下のアルゴリズムに従い，全ての中間ニューロンのスコアを計算する．

```
For All Middle Neuron
 IF This has Axon To Output Neuron
  Score += F-Value
 For All Neuron (Which Receives Axon From This Neuron)
  Score += Neuron's F-Value* ScoreAttenuationRate
```

(5) スコアの高い中間ニューロンのみを生き残らせ，残りのすべての中間ニューロンはふたたび(1)のようにランダムに軸索を張り直す．また，自身の閾値も-1 から 1 のあいだでランダムに取る．

(6) (1)～(5)を一世代として，試行を繰り返す.

Model 3-c. AI_PerfectMultiLayer

(1) 初期ネットワークをランダムな確率に基づき形成する. 中間ニューロンはセンサーニューロンまたは中間ニューロンからランダムに, すべてのニューロンに対してランダムに，軸索を形成する．このとき形成される軸索の重みは，-1 から 1 のあいだでランダムな値を取る.

```
For All Middle Neuron
 Do
  CreateSynapseProbability = Random(0-1)
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse from All Neuron Except Output Neuron
  If CreateSynapseProbability > Random(0-1)
   CreateSynapse To All Neuron
 While(NoSynapse)
```

(2) ネットワークに内部時間を定義し, センサーニューロンの値を各ニューロンに伝達して，単位時間ごとの値を算出する．(擬似的なスパイクニューロンモデル（第 6 章第 3 節）となる.）理由は，このモデルに限り中間構造にループ構造があり，出力層から後方ニューロンに対して再帰的に値を要求していく方法ではネットワークの出力が計算できないからである.

(3) ニューロンの総数 N に対し，充分大きな回数だけ内部時間を進め，値の安定したニューロンについてはその値を, 振動するニューロンについては確率的に 0 か 1 を値と定め，ネットワークの出力とする.

(4)～(5)は CleverMultiLayer と同様の手順を行う．ただし，ループ構造に関して，一度 F 値を受け取ったニューロンからは，二度目は F 値を受け取れないものとする.

(6) (1)～(5)を一世代として，試行を繰り返す.

第 3 項 実験

実験 4

表 9 に示した実験データ 2 の問題を，AI_SimpleMultiLayer モデル，AI_CleverMultiLayer モデル，AI_PerfectMultiLayer モデルでそれぞれ解く.

実験の意義

二キャラクター，二文字から成る，ほとんどランダムな文字列から，三文字目を予測する．この問題は三層構造では絶対に解くことのできない設定である．実験 3 において三層の AI_RewardBack モデルが解けなかったこの問題を，マルチレイヤーモデルであれば解けることを確認する．

また，後方にリワードを流す AI_CleverMultiLayer モデルでは，F 値が 1 になる中間ニューロンがすぐには淘汰されず，よい構造が長期にわたって保持されることを確認する．

同時に，AI_PerfectMultiLayer では，出力ニューロンに軸索を形成していない中間ニューロンが，他の中間ニューロンから受け取るリワードのみで生き残っている場合のあることを確認する．

パラメータ設定

ITERATION:300
ALL NEURON / MIDDLE NEURON:20 / 12
SURVIVAL RATE:0.5
SCORE_ATTENUATION_RATE:0.3

以下，SCORE_ATTENUATION_RATE は後方に流すスコアの，出力が中間ニューロンに与えるスコアに対する比率である．

結果

次頁に示す．

また以下に示す F 値表について，後方にリワードを伝達する AI_CleverMultiLayer モデル，AI_PerfectMultiLayer モデルでは，左から右へのソートは F 値ではなくスコア順で行っており，値は「ニューロンの ID;F 値;スコア」の順で各セルに表示されている．

表 12　実験 4：反復後期の F 値（AI_SimpleMultiLayer）

231	15;1	17;1	9;0.8	11;0.8	18;0.8	6;0.6666667	7;0.6666667
232	15;1	16;1	17;1	9;0.8	11;0.8	18;0.8	6;0.6666667
233	8;1	15;1	17;1	9;0.8	11;0.8	12;0.8	18;0.8
234	8;1	12;1	15;1	17;1	9;0.8	11;0.8	18;0.8
235	8;1	12;1	15;1	17;1	9;0.8	10;0.8	11;0.8
236	8;1	12;1	15;1	17;1	9;0.8	10;0.8	11;0.8
237	8;1	12;1	15;1	17;1	7;0.8	9;0.8	10;0.8
238	8;1	12;1	15;1	17;1	7;0.8	9;0.8	10;0.8
239	8;1	12;1	13;1	15;1	17;1	7;0.8	9;0.8
240	8;1	12;1	13;1	15;1	16;1	17;1	7;0.8
241	7;0.6666667	9;0.6666667	10;0.6666667	13;0.6666667	17;0.6666667	11;0.5	11;0
242	6;0.6666667	7;0.6666667	9;0.6666667	10;0.6666667	12;0.6666667	13;0.6666667	15;0.6666667
243	6;0.6666667	9;0.6666667	10;0.6666667	11;0.6666667	12;0.6666667	13;0.6666667	15;0.6666667
244	6;0.6666667	7;0.6666667	9;0.6666667	10;0.6666667	11;0.6666667	12;0.6666667	13;0.6666667
245	6;0.8	7;0.6666667	8;0.6666667	9;0.6666667	10;0.6666667	11;0.6666667	12;0.6666667
246	15;0.8	6;0.6666667	7;0.6666667	8;0.6666667	9;0.6666667	10;0.6666667	11;0.6666667
247	6;0.8	7;0.6666667	8;0.6666667	9;0.6666667	10;0.6666667	11;0.6666667	13;0.6666667
248	6;0.8	15;0.8	8;0.6666667	9;0.6666667	10;0.6666667	11;0.6666667	12;0.6666667

表 13　実験 4：反復後期の F 値（AI_CleverMultiLayer）

201	16;1;1.182588	11;1;1.174505	18;0.8;1.028868	12;0.8;1.024481	7;0.8;1.020598	14;0.8;1.012155	8;0.8;0.86
202	16;1;1.187022	11;1;1.176838	8;1;1.148072	12;0.8;1.023974	14;0.8;1.00156	18;0.8;1.000925	7;0.8;0.9967812
203	16;1;1.175861	11;1;1.154242	8;1;1.145846	14;0.8;1.054557	12;0.8;1.022493	7;0.8;1.011164	18;0.8;0.9945501
204	16,1,1.198505	11,1,1.177139	8,1,1.122044	12,0.8,1.02299	14,0.8,1.01964	7,0.8,1.019292	18,0.8,1.015252
205	16;1;1.161471	11;1;1.135859	8;1;1.073273	7;0.8;0.9915614	12;0.8;0.9842479	14;0.8;0.9772135	18;0.8;0.9716535
206	16;1;1.221862	11;1;1.192693	8;1;1.155086	7;0.8;1.034412	18;0.8;1.026068	12;0.8;1.014737	14;0.8;1.00204
207	16;1;1.158044	11;1;1.151245	8;1;1.117968	18;0.8;1.021231	12;0.8;0.985213	7;0.8;0.9745303	14;0.8;0.9726224
208	16;1;1.184739	11;1;1.15724	12;0.8;1.017161	7;0.8;1.0127	18;0.8;1.000354	8;1;1	14;0.8;0.9994323
209	16;1;1.220679	11;1;1.176441	8;1;1.142771	18;0.8;1.054135	14;0.8;1.020121	12;0.8;1.018037	7;0.8;1.008908
210	16;1;1.238443	11;1;1.195529	8;1;1.173718	14;0.8;1.05548	12;0.8;1.053087	7;0.8;1.046594	18;0.8;1.021243
211	16;1;1.204185	11;1;1.190089	8;1;1.181309	7;0.8;1.046558	18;0.8;1.034806	12;0.8;1.024353	14;0.8;1.023719
212	16;1;1.19858	11;1;1.182735	8;1;1.128725	18;0.8;1.043396	14;0.8;1.043067	12;0.8;1.028128	7;0.8;1.012224
213	16;1;1.179056	11;1;1.172341	8;1;1.138499	6;1;1.06	18;0.8;1.023939	12;0.8;1.003802	14;0.8;0.9984627

表 14　リワードのみで生き残る中間ニューロン（AI_PerfectMultiLayer）

16	7;0.68418;0.666	9;0.668;0.666	11;0.667;0.666	13;0.667;0.666	8;0.507;0.5	18;0.106;0	14;0.076;0	19;0.068;0

考察

表 12, 表 13 から， AI_SimpleMultiLayer モデル，AI_CleverMultiLayer モデル，どちらのモデルでも問題に対して F 値を 1 にした中間ニューロンがいることが確認できる．三層モデルでは原理的に解くことのできなかった問題が，多層モデルでは容易に解かれることが確認される．

また表 14 から，AI_PerfectMultiLayer において，出力ニューロンに軸索を形成していないにも関わらず，他の予測に役立つ中間ニューロンへ軸索を張っていることで，伝達されたリワードを受け取り，それによって生き残っているニューロンが存在することを確認出来る．（表中右端，ニューロン ID18,14,19.）

表 12 において，F 値が 1 になった中間ニューロンが，反復試行 241 回目に F 値を落としているのは，当該ニューロンの後方に位置する中間ニューロンが消滅してしまったために，正しい予測を行えなくなったものと思われる．

AI_SimpleMultiLayer モデルは，AI_CleverMultiLayer モデルと異なり出力ニューロンからのリワードが，中間ニューロンの後方まで伝達されないため，ある中間ニューロンの予測精度が高くても，その後方の中間ニューロンの予測精度が低いと，結果として予測精度の高い中間ニューロンが巻き添えとなって精度を落としてしまう事態が生じうることを示している．対して表 13 における F 値上位のニューロンは，その後方ニューロンのスコアも高くなるために，精度が比較的安定していることが確認出来る．

AI_PerfectMultiLayer では，ネットワーク形成のアルゴリズム上，構造が複雑かつ再帰的に入り組むため，世代を通じて最適な構造を保つことが難しく，F 値の高い中間ニューロンも次の世代では消滅してしまい，よい結果が得られなかった．自由な多層化を許すことでデータの分解能は向上するが，選択淘汰における構造の更新で，ひとつのニューロンの消滅がネットワーク全体に影響を与えてしまうことは致命的である．計算時間も，ループ構造のために，他のモデルに比べて極端に長い．

したがって，ここでは AI_RewardBack などの三層モデルよりもデータの分解能に優れ，かつ構造更新のロジックが単純であるために，構造の安定性が高く，高速な計算と F 値の収束に信頼のおける AI_CleverMultiLayer モデルを，もっともよい多層モデルと見なすことにする．

第 6 節 AI_DelaySensor

第 1 項 概要

前節までの実験で，多層化モデルの特徴が確認され，中でも AI_CleverMultiLayer モデルがもっとも優れた性能を持つことが確認された．

しかし，これまでに設計したネットワークは，いずれも現在までに観測されたすべてのデータをセンサーとして置いている．ゆえにデータ長が長くなればなるほど，センサーニューロンの個数は線形に増えていき，その分中間ニューロンの数も増やさざるを得ず，計算時間は莫大になっていく．すべての観測情報が等価にセンサーとして使えることは理想的ではあるが，センサーニューロンが膨大な数になった場合，軸索の形成パターンが爆発するために，いくら反復試行を繰り返しても最適構造に辿り着けない可能性もある．

そこで，中間部の多層構造は維持したまま，過去の観測データを数値化して減衰しながら加算していくことで，時間方向に観測データを圧縮するモデルを考える．ネットワークモデルのイメージを図示すると，図 43 のようになる．

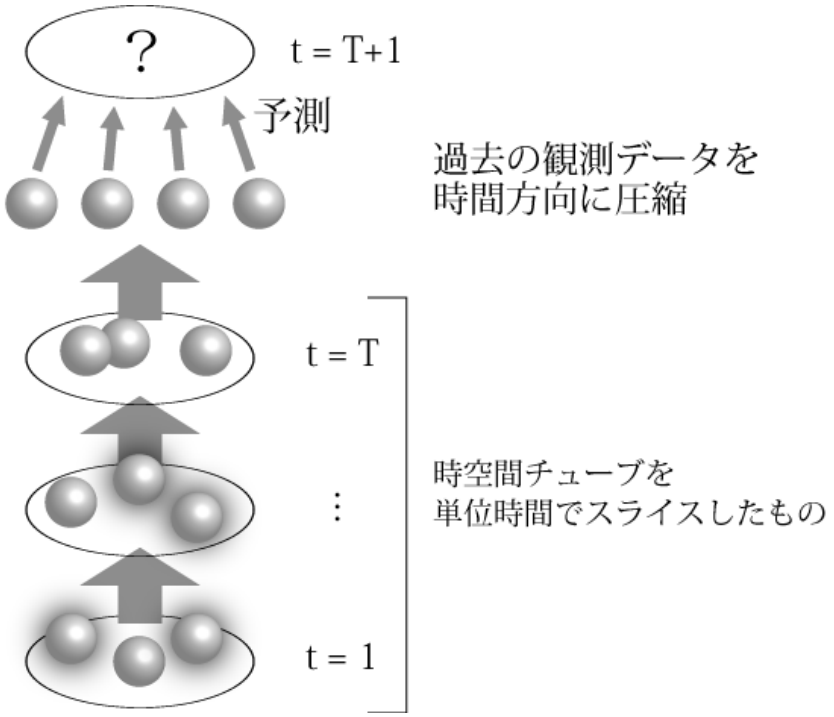


図 43　AI_DelaySensor のネットワークモデル・イメージ

AI_DelaySensor では，センサーニューロンはセンサーの観測文字数分しか確保されない．例えばデータが{a, b, c}の三文字からのみ構成されている場合，予測のために与

えられる文字列の全長に関わらず, センサーニューロンは全部で 3 つである. 代わりに, 各センサーニューロンには 0 か 1 の二値ではなく, 時間遅れの加味された実数値が初期値として与えられる（図 44）. 詳細はアルゴリズムの項に示す.

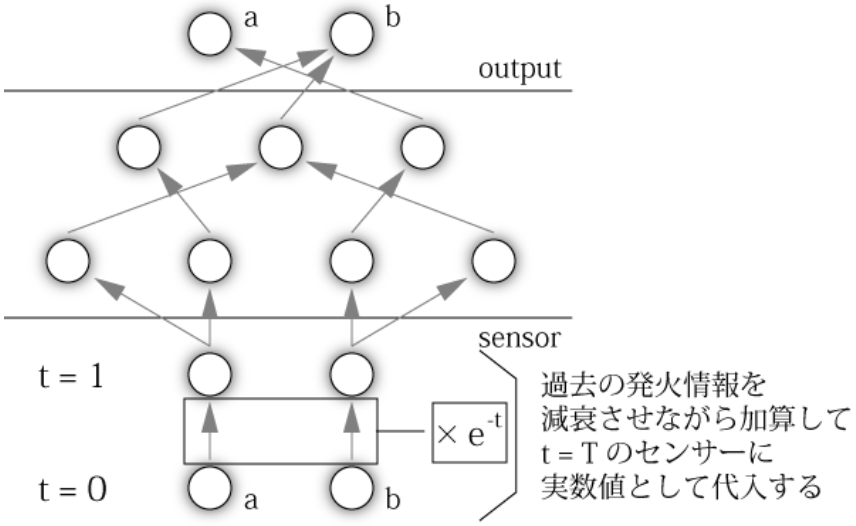


図 44　AI_DelaySensor の構造図解

センサーニューロン部の構造はこれまでのモデルと異なるが, それ以外は重み更新に基づく多層モデルである. 軸索をセンサーに対してランダムに，中間ニューロンからランダムに，出力に対して一本まで張ることを許し，予測精度にしたがってリワードを与えることで，センサーから中間ニューロンへの重みを更新する. その後，スコアの高かった中間ニューロンのみを生き残らせ, 残りのすべての中間ニューロンはふたたびランダムに軸索を張る. これを一世代として，試行を繰り返す.

多層化のプロセスは AI_CleverMultiLayer と同様である. したがって内部にループ構造は生じない.

第２項 アルゴリズム

(1) センサーニューロン i に与える初期値 V_i を, 以下の式に基づき計算する. ただし, $S_i(t)$ はセンサーニューロン i に対応するデータが，時刻 t で観測されたときに 1 を取り，それ以外にときに 0 を取る関数である.

$$V_i = \sum_{t=0}^{T} \left(S_i(t) \times e^{-\frac{t}{\tau}} \right)$$

(2) これをセンサーニューロンに初期値として与え，全ニューロンの出力を計算する.

センサーニューロンに与えられる値は実数値であるが，以降の中間ニューロン及び出力ニューロンの値は，0 または 1 の二値である．
(3) 以降の手順は，AI_CleverMultiLayer と同様に行う．

第３項 実験

実験５

表 9 に示した実験データ 2 の問題を，AI_DelaySensor モデルで解く．

実験の意義

二キャラクター，二文字から成る（ほとんどランダムな）文字列から，三文字目を予測する．この問題を，センサーニューロンの個数が 2 つだけになった AI_DelaySensor モデルでも解くことができることを確認する．

パラメータ設定

ITERATION: 300
ALL NEURON / MIDDLE NEURON: 20 / 16
SURVIVAL RATE: 50%
SCORE_ATTENUATION_RATE:0.6
SENSOR_ATTENUATION_RATE:0.8

以下，SENSOR_ATTENUATION_RATE は，センサーニューロンに与えられる初期値に加算される入力情報の減衰係数である．

結果

次頁に示す．

考察

表 15 から，僅か 2 つのセンサーニューロンで，AI_CleverMultiLayer モデルと大差のない予測精度を示していることが確認できる．実数値の僅かな差を識別することで，複数のセンサーニューロンではなく単一のセンサーニューロンから，訓練データの提示された順番を学習していると考えられる．

また，F 値が 1 のニューロンも存在しており，多層モデルとして問題なく機能していることも確認できる．

表 15　実験 5：F 値 1 の中間ニューロンの確認（AI_DelaySensor）

1	16;0.7555556;0.667	7;0.667;0.667	12;0.08888889;0	17;0.03333334;0
2	18;0.667;0.667	7;0.667;0.667	12;0.1066667;0	16;0.08888889;0
3	18;0.8444445;0.667	12;0.7306668;0.667	11;0.6666669;0.667	7;0.4000045;0.4
4	10;0.8444445;0.667	11;0.8027087;0.667	18;0.7950877;0.667	15;0.7950877;0.667
5	18;0.9196373;0.8	11;0.7477334;0.667	13;0.6666668;0.667	10;0.5066667;0.4
6	18;0.7442604;0.667	11;0.6789538;0.667	7;0.6711869;0.667	13;0.667;0.667
7	13;0.8270606;0.667	5;0.8270569;0.667	6;0.8019698;0.667	4;0.8019698;0.667
8	19;0.8482814;0.8	13;0.8021995;0.667	5;0.8021995;0.667	18;0.7454146;0.667
9	19;0.8172455;0.8	4;0.8011175;0.8	13;0.7582114;0.667	18;0.7340927;0.667
10	12;0.881632;0.8	18;0.7562039;0.667	13;0.7438479;0.667	19;0.6773361;0.667
11	13;0.7436011;0.667	4;0.6740544;0.667	6;0.5812478;0.5	18;0.5579302;0.5
12	12;0.8119592;0.8	18;0.733807;0.667	13;0.7264237;0.667	4;0.6727056;0.667
13	4;0.8055702;0.8	13;0.7311124;0.667	11;0.6710359;0.667	6;0.6710054;0.667
14	17;0.8825349;0.8	13;0.7203792;0.667	7;0.7129219;0.667	4;0.6713996;0.667
15	13;0.7048716;0.667	15;0.7024229;0.667	8;0.6692462;0.667	4;0.668668;0.667
16	17;1.075651;1	15;0.8006033;0.667	12;0.7733612;0.667	19;0.7574691;0.667
17	13;0.7263542;0.667	17;0.7102239;0.667	8;0.6751823;0.667	4;0.6735595;0.667
18	14;0.7876649;0.667	12;0.7658455;0.667	17;0.7516583;0.667	19;0.7435399;0.667
19	14;0.9068321;0.8	16;0.8;0.8	15;0.7561299;0.667	17;0.7338929;0.667
20	14;1.076724;1	16;1.000045;1	15;0.7432724;0.667	17;0.7327539;0.667

実験 6

表 16　実験データ 3

[Feature]	[Label]
bcdefghij	a
acdefghij	b
abdefghij	c
abcefghij	d
abcdfghij	e
abcdeghij	f
abcdefhij	g
abcdefgij	h
abcdefghj	i
abcdefghi	j

実験の意義

訓練データ中に存在しない文字列を予測する問題を解く．AI_CleverMultiLayer モデルでも，AI_DelaySensor モデルでも，等しくこの問題が解けることを確認し，また F 値の収束時間にどれほどの差が生じるかを確認する．

パラメータ設定

ITERATION:1000

ALL NEURON / MIDDLE NEURON:

120/20(CleverMultiLayer)　40/20 (CleverDelaySensor)

SURVIVAL RATE:50%

SCORE_ATTENUATION_RATE:0.6

SENSOR_ATTENUATION_RATE:0.8

結果

表 17　実験 6：F 値の完全収束（AI_CleverMultiLayer）

368	119;1;1.122011	117;1;1.11826	103;1;1.113541	110;1;1.023036	118;0.666;0.8147
369	119;1;1.131127	117;1;1.106302	103;1;1.098941	118;0.666;0.8160	112;0.666;0.7585
370	103;1;1.07721	117;1;1.063593	119;1;1.06285	118;0.666;0.7814	115;0.666;0.7262
371	117;1;1.10766	119;1;1.10766	103;1;1.104879	118;0.666;0.8259	115;0.666;0.7743
372	117;1;1.067169	103;1;1.062493	119;1;1.062214	118;0.666;0.7751	115;0.666;0.7403
373	103;1;1.107998	117;1;1.100478	119;1;1.088076	118;0.666;0.8096	115;0.666;0.7769
374	103;1;1.083472	117;1;1.055245	119;1;1.053552	101;1;1.0199	118;0.666;0.7909
375	103;1;1.057133	119;1;1.0549	117;1;1.053181	118;0.666;0.7810	115;0.666;0.7285
376	103;1;1.088013	119;1;1.0833	117;1;1.076802	118;0.666;0.7894	115;0.666;0.7648
377	117;1;1.09318	103;1;1.082209	119;1;1.071692	102;1;1.014089	118;0.666;0.7759
378	103;1;1.079881	117;1;1.055725	119;1;1.051419	101;1;1.020492	118;0.666;0.8088
379	103;1;1.091407	117;1;1.081191	119;1;1.069714	101;1;1.030988	118;0.666;0.7856
380	103;1;1.113992	119;1;1.085108	117;1;1.080625	106;1;1.054177	118;0.666;0.8069
381	103;1;1.075927	117;1;1.065441	119;1;1.053387	106;1;1.009254	118;0.666;0.7902
382	103;1;1.101759	117;1;1.066215	119;1;1.058446	101;1;1.028967	106;1;1.022992
383	119;1;1.171333	103;1;1.169152	117;1;1.118106	102;1;1.102797	106;1;1.1

表 18　実験 6：F 値の完全収束（AI_DelaySensor）

183	30;1;1.424011	35;1;1.410185	31;1;1.347646	37;1;1.283182	39;0.5;0.8607985
184	35;1;1.390773	30;1;1.332772	31;1;1.269358	37;1;1.175614	39;0.5;0.8272576
185	31;1;1.344955	30;1;1.341023	35;1;1.303821	37;1;1.096784	39;0.5;0.754151
186	35;1;1.355398	30;1;1.331361	31;1;1.293705	37;1;1.215506	39;0.5;0.8162119
187	30;1;1.303617	35;1;1.220103	31;1;1.184363	37;1;1.13766	39;0.5;0.7105027
188	35;1;1.629489	30;1;1.605656	31;1;1.595719	37;1;1.489288	39;0.5;1.087098
189	35;1;1.374775	30;1;1.332318	31;1;1.268805	37;1;1.215311	39;0.5;0.8421592
190	35;1;1.297783	30;1;1.251817	31;1;1.202563	37;1;1.106727	39;0.5;0.7572736
191	35;1;1.289707	30;1;1.270989	31;1;1.19961	37;1;1.142562	39;0.5;0.7282565
192	35;1;1.315531	30;1;1.294886	31;1;1.183015	37;1;1.12815	39;0.5;0.7348485
193	35;1;1.523749	30;1;1.510954	31;1;1.473333	37;1;1.321969	21;1;1
194	30;1;1.459274	35;1;1.430529	31;1;1.374421	37;1;1.324437	21;1;1.304573

考察

表 17，表 18 から，どちらのモデルにおいても F 値は完全に 1 に収束し，構造が最適化されていることが確認できる．また，この問題においては F 値の収束時間に差があり，AI_DelaySensor モデルの方が若干早く収束に辿りついている．計算時間は同程度であるが，ニューロンの総数が少ないのは AI_DelaySensor モデルである．観測データの時間方向への圧縮が有効に機能していることが確認される．

実験 7

[Feature][Label]

文字列長 70 の一文が訓練データとなっているものを，40 文用意する．正解は abcd の四通りである．また，正解を予測するための決め手となる文脈は，訓練データの後半寄りである．データの詳細は省略する．一文の文字列長の長い，予測可能な訓練データであれば，この実験の意図から何でもよい．

実験の意義

文字列長が長くなったときの，AI_CleverMultiLayer モデルと AI_DelaySensor モデ

ルのパフォーマンスの違いを確認する．両モデルに同一数の中間ニューロンを持たせた場合，ニューロン数の差は 2500 対 80 となり，このことから精度のみならず計算時間にも圧倒的な差が出てくることを確認する．

パラメータ設定

ITERATION: 3000

ALL NEURON / MIDDLE NEURON:

2500 / 20 (CleverMultiLayer)　80/20 (CleverDelaySensor)

SURVIVAL RATE: 50%

SCORE_ATTENUATION_RATE:0.6

SENSOR_ATTENUATION_RATE:0.9

結果

表 19　実験 7：F 値の収束（AI_CleverMultiLayer）

2351	1783;1;1.14535	1776;1;1.12681	1769;1;1.02795	1788;0.8;0.903	1787;0.666;0.814
2352	1776;1;1.110527	1783;1;1.102893	1769;1;1	1788;0.8;0.908	1787;0.666;0.791
2353	1776;1;1.156648	1783;1;1.092391	1769;1;1.004039	1788;0.8;0.920	1787;0.666;0.811
2354	1776;1;1.12355	1783;1;1.081528	1769;1;1	1788;0.8;0.931	1773;0.666;0.793
2355	1776;1;1.123801	1783;1;1.091719	1769;1;1	1788;0.8;0.932	1787;0.666;0.795
2356	1776;1;1.119285	1783;1;1.103573	1769;1;1.026696	1788;0.8;0.918	1787;0.666;0.787
2357	1776;1;1.111162	1783;1;1.080873	1769;1;1	1788;0.8;0.911	1787;0.666;0.778
2358	1776;1;1.116137	1783;1;1.084975	1769;1;1.004503	1788;0.8;0.920	1787;0.666;0.770
2359	1776;1;1.099381	1783;1;1.088811	1769;1;1.00844	1788;0.8;0.891	1773;0.666;0.764
2360	1776;1;1.140366	1783;1;1.123987	1769;1;1.024	1788;0.8;0.971	1773;0.666;0.795
2361	1776;1;1.17946	1783;1;1.114437	1769;1;1.007205	1777;1;1.004	1788;0.8;0.955
2362	1776;1;1.133018	1783;1;1.07645	1769;1;1.021348	1777;1;1.018	1788;0.8;0.909
2363	1776;1;1.117524	1783;1;1.046364	1769;1;1	1777;1;1	1788;0.8;0.891

表 20　実験 7：F 値の収束（AI_DelaySensor）

489	75;0.8;1.090477	58;1;1	71;0.666;0.994	73;0.666;0.915
490	66;1;1.128986	58;1;1.014358	71;0.666;0.955	65;0.666;0.952
491	66;1;1.157318	71;0.666;1.056	58;1;1.055	70;0.666;0.950
492	75;0.8;1.068547	58;1;1	71;0.666;0.888	65;0.666;0.875
493	75;0.8;1.281942	66;1;1.257772	71;0.666;1.1307	58;1;1.082286
494	75;0.8;1.300628	66;1;1.194162	58;1;1.1733	71;0.666;1.078
495	75;0.8;1.148833	66;1;1.128569	71;0.666;1.022	58;1;1
496	75;0.8;1.139412	66;1;1.097033	58;1;1	71;0.666;0.907
497	75;0.8;1.156495	66;1;1.083001	71;0.666;1.046	58;1;1
498	75;0.8;1.136109	66;1;1.057256	58;1;1	71;0.666;0.982
499	66;1;1.238891	75;0.8;1.161694	71;0.666;1.092	58;1;1.041111
500	66;1;1.09516	75;0.8;1.084242	58;1;1	71;0.666;0.927
501	75;0.8;1.145575	66;1;1.113222	58;1;1	71;0.666;0.994
502	75;0.8;1.183411	66;1;1.149557	58;1;1.0698	71;0.666;1.040
503	66;1;1.239344	75;0.8;1.19465	71;0.666;1.040	58;1;1.040206
504	75;0.8;1.194018	66;1;1.088876	71;0.666;1.043	58;1;1.004571

考察

表 19，表 20 から，どちらのモデルにおいても，かなりの反復試行後に F 値が収束していることが確認できる．しかし，実験 6 と比べて，両モデルの収束時間には圧倒的な差がある．AI_CleverMultiLayer モデルは，同程度の F 値の収束までに，AI_DelaySensor モデルに比べて反復試行回数上で 5 倍強，実際の計算時間では約 50 倍の時間がかかっている．このようにデータが大規模であり，かつ予測の決め手となる文字列が予測対象の比較的近くに存在する問題に対しては，AI_DelaySensor モデルの方が圧倒的によいパフォーマンスを示すことがわかる．

第 7 節 全体の考察

以上の実験群から，設計した各理論モデルについて，収束速度と精度の観点からその性能を図示すると，図 45 のようになる．ただし，AI_SimpleMultiLayer モデルは，AI_CleverMultiLayer モデルの下位互換として，ここでは省略する．

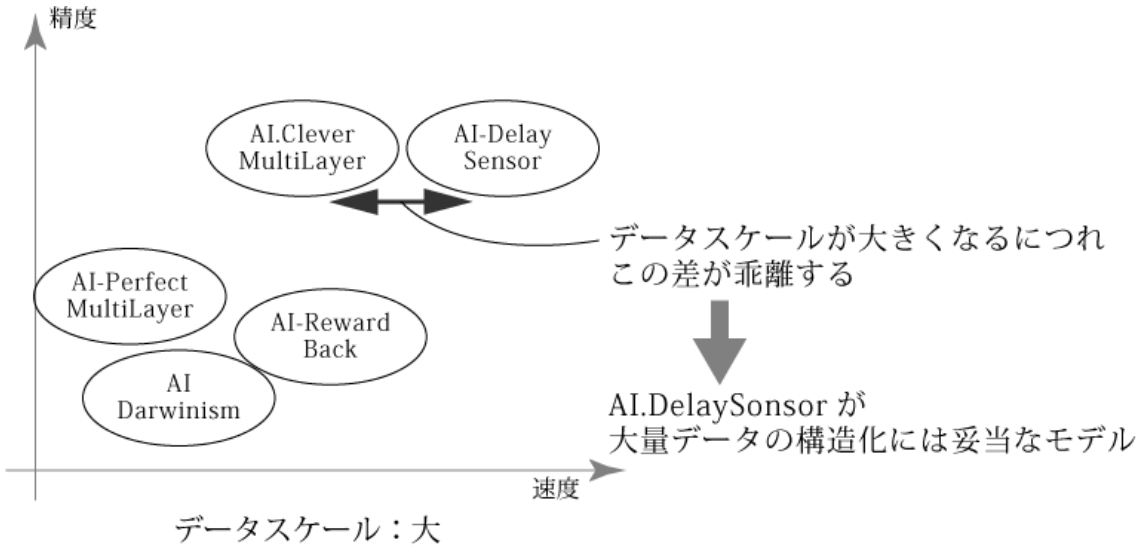


図 45　各理論モデルの精度と収束時間の比較

AI_Darwinism モデルでは，選択淘汰の原理を三層ネットワークに適用することで，データに対して予測性を持つように構造が更新されていき，最終的には最適な構造にたどり着くことが確認された．このモデルでネットワークが予測性を持ったのは，予測をするために適切なセンサーニューロンに後方軸索を形成している中間ニューロンが，選択淘汰の原理によって生き残り続けた結果である．このような中間ニューロンは，ネットワークの最終形において，複数のセンサーニューロンの発火パターンを抽象化しているものと見做すことができるだろう．データの抽象化を行う中間ニューロンが，選択淘汰の原理によって形作られることが示されたと考えてよい．

AI_RewardBack モデルは，上述のモデルに適応の原理を加えたものである．中間ニューロンが，センサーニューロンの発火パターンと出力ニューロンの発火という環境に対して，よりよい予測ができるように自身の後方軸索の重みを更新できるようになったことで，AI_Darwinism モデルよりも圧倒的に早く最適構造に辿りついた．適応の原理は，データの抽象化を行うために，与えるデータの数をより少なくできるという貢献をしていることが示されたと考えてよい．

AI_SimpleMultiLayer モデルは，AI_RewardBack モデルを多層化したものである．多層化することによって，三層構造では原理的に解けない問題が解けることが確認された．これは，データに対する分解能が向上したと解釈できる．多層化は，より複雑で予測しにくいパターンを持つデータを抽象化することに貢献すると考えてよい．

また，AI_PerfectMultiLayer モデルは，AI_CleverMultiLayer モデルと同じく多層化されたネットワークモデルであるが，その多層化のプロセスが，ループを許可するなどネットワークに極めて複雑な構造を作らせてしまうものであるために，性能と収束速

度で AI_CleverMultiLayer モデルに劣った．このことから，ネットワーク多層化のプロセスは AI_CleverMultiLayer モデルの，「淘汰されたニューロンは，生き残った中間ニューロンをセンサーニューロンと見做して後方軸索を形成できる」という手法の優れていることが確認された．

AI_DelaySensor モデルは，その手法を用いて多層化しつつ，入力データを時間方向に圧縮することでセンサーニューロンの数を大幅に減らし，構造を経済化したことで精度と収束時間を共に向上した．また，AI_CleverMultiLayer モデルと AI_DelaySensor モデルの収束時間差は，図 45 に示したように，データのスケールが大きくなればなるほど乖離していく．通常，抽象化を行いたいデータは複雑かつ大量であることを考えれば，データの複雑さや量に対して収束時間の変化量が少ないことは重要な機能である．時間方向への圧縮は，モデルの複雑さと計算時間を減少させ，データの抽象化をより現実的な条件下で行うことに貢献していると考えてよい．

以上の議論で，本章の最初に示した選択淘汰，適応，多層化，時間方向の圧縮という四つの要素が，いずれもそれぞれ違った形でデータの抽象化に貢献していることが確認された．

第６章 生体的制約に基づくアルゴリズムの実装

この章では，前章の設計と分析を踏まえて，それらが生体的な制約下でどのように実装されているかを考え，可能なアルゴリズムを実装する．また，実装したプログラムについて動作確認を行い，考察を行う．

第１節 概要

第５章で設計，分析したアルゴリズムはいずれも理論上のものであり，生体的な制約を考慮していない．しかし我々の脳は，様々な生体的制約下で，予測に基づく自己構造化を実現しているはずである．前述のアルゴリズムの特性をすべて持ちながら，かつ神経科学的に可能な形でのインプリメンテーションを試みる．

本章で実装するアルゴリズムにおいては，ネットワークの構造更新はすべてニューロンのローカルな挙動に集約されていなければならない．すなわち，全体の精度や振る舞いを確認しながらネットワークを恣意的に更新していく神のような存在は仮定しない．現実に脳の内部で化学的反応として起こりうる事象のみがモデル化の対象となる．

第２節 生体ニューロンの構造

本節では脳のアルゴリズムをモデル化するため，生体ニューロンの構造を確認する．生体ニューロンの構造は，概略的に図 46 のように示される．特に，ニューラルネットワークとの関係が深い部分のみを図示したものである．

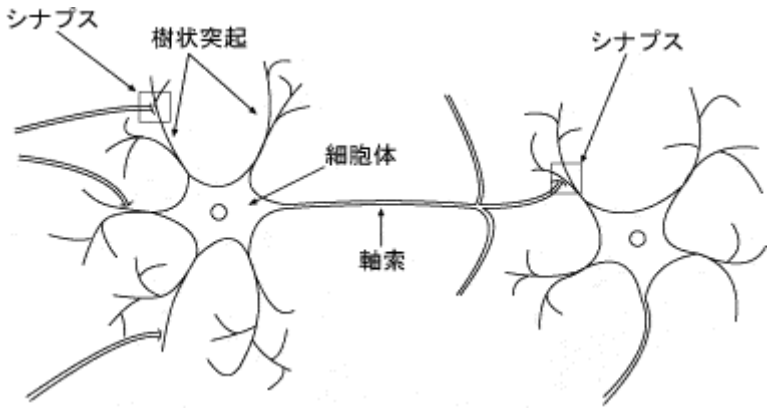


図 46　生体ニューロンの構造

ニューロン（細胞体）は他のニューロンへ向けて軸索を伸ばしており，ニューロンの樹状突起にある受容体とシナプスを形成している．シナプスは，伝達される化学物質によって興奮性シナプスと抑制性シナプスに分けられる．興奮性シナプスはシナプス後電位を増加させ，抑制性シナプスはシナプス後電位を減少させる．

ニューロン同士の相互作用は次のように行われる．ニューロンは内部膜電位を持っており，あるニューロンの内部膜電位の値が閾値を越えたとき，ニューロンはパルスを発生させる．そのパルスが軸索を通過し，シナプス伝達により接続先の受容体に伝達され，それによって生じたシナプス後の電位が，接続先ニューロンの内部膜電位に影響を及ぼしている．

これらの構造と相互作用をニューラルネットワークに取り入れることを考える．

第３節 Spike Neuron Model

第１項 概要

前述の生体的ニューロンの構造を，神経科学的に可能な形で実現したモデルのひとつに，スパイクニューロンモデルがある．

McCulloch-Pitts モデルのパーセプトロンや，リカレントニューラルネットワークのように，スパイク発火頻度を変数としたモデルは，数学的解析や結合加重の学習による設定が行いやすいという利点を持つ．しかし発火頻度というものを実際のニューロンのデータから定義するには，スパイクをある時間の間数えて平均するという操作が必要である．そのため，スパイクのタイミングを考慮にいれたモデルが必要になる．

最も単純なものとして，スパイクは入力に対してただ確率的に生成されるという，ポアソンスパイクモデルがある．これはスパイク生成が過去の発火履歴によらないモデルである．しかし実際のニューロンでは，一度スパイクを生成すると，その後数ミリ秒間は次のスパイクを生成しないという不応期特性がある．そこで，スパイクを生成するとニューロンの電位がリセットされ，その後シナプス入力の積分によって電位が上昇し，ある閾値を超えるとふたたびスパイクが生成されるという現象を近似したものが，積分発火(Integrate-and-Fire)モデルである[6]．本研究の提案アルゴリズムにおけるニューロンモデルは，この積分発火モデルに基づいている．

次項ではこの積分発火モデルをもとにしたスパイクニューロンモデルについて，その詳細なアルゴリズムを示す．

第２項 アルゴリズム[17]

ある時間 t において，ニューロン i が発火したかどうかを，関数 $S_i(t)$ で表す．$S_i(t)$ はニューロンが発火したとき 1，発火しなかったときに 0 となる関数である．

ニューロン i が入力パルスを受け取ると，$\Delta_{\mathrm{i}}^{\mathrm{ax}}$ だけ遅れてニューロンの内部膜電位に変化が生じる．入力側のシナプス結合が興奮性結合であれば膜電位は上昇し，抑制性結合であれば膜電位は降下する．

内部膜電位の変化は以下の式に従う．

$$e(t) = \left(\frac{t}{\tau_e^2}\right) e^{-\frac{t}{\tau_e}}$$

τ_{e} はニューロンの内部膜電位の変化を決定する時定数である．

膜電位の大きさによってニューロン i は確率的に発火し，パルスを発生させる．パルスが発生すると $\Delta_{\mathrm{i}}^{\mathrm{inh}}$ だけ遅れてニューロン i の内部電位は大きく下がり始める．内部電位の降下は次の式に従う．

$$\eta(t) = e^{-\frac{t}{\tau_n}}$$

τ_{n} は内部膜電位の降下特性を決定する時定数である．ニューロン i の膜電位は，前結合しているすべてのニューロンから与えられるパルスによって生じる膜電位の変化と，ニューロン自身の膜電位の抑制を線形結合したものとして表わされる．

$$h_i^{syn}(t) = \sum_{j=1}^{N} J_{ij} \sum_{\tau=0}^{t} e(\tau)\, S_j(t - \tau - \Delta_{\mathrm{i}}^{\mathrm{ax}}) + h_i^{inf}(t)$$

J_{ij} はニューロン i からニューロン j へのシナプス結合の重みである． $J_{ij} > 0$ のときシナプスは興奮性結合であり，$J_{ij} < 0$ のとき，抑制性結合である．

また，$h_i^{inf}(t)$ は，抑制信号の振幅 J^{inf} を用いて，次のように表わされる．

$$h_i^{inf}(t) = -\sum_{j=1}^{N} J^{inf} \eta(\tau) S_i\left(t - \tau - \Delta_{\mathrm{i}}^{\mathrm{inf}}\right)$$

ニューロンには発火後に不応期の電圧降下があり，これは，

$$h_i^{ref}(t) = \begin{cases} -R \ \left(if\ t_F \le t \le t_F + \tau_{ref}\right) \\ 0 \end{cases}$$

と表わされる．ただしニューロン i は $t = t_F$ に前回発火したものとし，R は不応期の電圧降下量を決定する定数，τ_{ref} は不応期の期間である．

以上から，いまニューロンに外部入力がないと仮定すると，時刻 t におけるニューロン i の内部膜電位は，

$$h_i(t) = h_i^{syn}(t) + h_i^{ref}(t)$$

と表すことができる．この値により，ニューロン i は次に示す確率で発火する．

$$P[S_i(t+\Delta t)=1|h_i(t)]=\frac{1}{1+e^{-\beta\{h_i(t)-\theta\}}}$$

単位時間ごとにすべてのニューロンについて発火判定を行うことで，ネットワークは動的に変化していく．その取りうる状態はリカレントネットワーク同様，平衡状態や振動状態，カオス振動など様々である．

本研究の提案アルゴリズムは，原理的には上述のスパイクニューロンモデルに基づきながら，予測の補助問題を解いて自己構造化を行う動的なニューラルネットワークとしてモデル化し，不要なパラメータを省略するとともに，必要なパラメータを追加したものである．次章では，そのアルゴリズムを実装する．

第４節 環境

一連のプログラム作成にあたっては，言語は C#を用い，開発環境には Microsoft Visual C# 2008 Express Edition を利用した．

第５節 モデル

第１項 クラス構造

プログラムのうち，上述した脳内の構造をモデル化した部分は，次のクラス群から形成されている．

Field クラス

Field クラスは脳自体を表すクラスである．脳全体の物理時間や，ニューロンを呼び出す役割を果たしている．

Neuron クラス

Neuron クラスはニューロン単体を表すクラスである．ニューロンは自身の内部膜電位に基づいて確率的に発火し，パルスを軸索に伝達する．また，他のニューロンとの発火の相関関係により，血流を受け取って自身の軸索を強化したり，軸索のないニューロンへ軸索を形成したりする．上位クラスに Sensor Neuron クラスがあり，センサーからの入力以外には反応しない Sensory Receptor を持っている．これが，ニューラルネットワークにおける入力層に対応している．

Axon クラス

Axon クラスは，軸索単体を表すクラスである．軸索は，ニューロンから受け取ったパルスを受容体に伝達し，ニューロンから送られてくる重み更新の命令にしたがって自身の重みを変更する．シナプスの種類には，生体的には興奮性（Excitatory）と抑制性（Inhibitory）があるが，これは軸索の重みの正負として表現し，プログラム上は区別していない．

Receptor クラス

Receptor クラスは，受容体単体を表すクラスである．受容体は重みを持たず，シナプス後電位を持っている．軸索から送られてきた重み付きのパルスを受け取り，自身のシナプス後電位を更新する．更新されたシナプス後電位は，ニューロン固有の時間遅れを通じてニューロンの内部膜電位に伝達される．

上位クラスに Sensory Receptor クラスがある．これは，軸索とはシナプスを形成せず，プログラムが与えるセンサー情報を入力するための受容体であり，ニューラルネットワークにおける入力層に対応している．

第２項 モデルの全体像

以下にプログラムモデルの全体像を示す．各クラスのプログラムにおける位置付けと相互関係は次のようになっている．

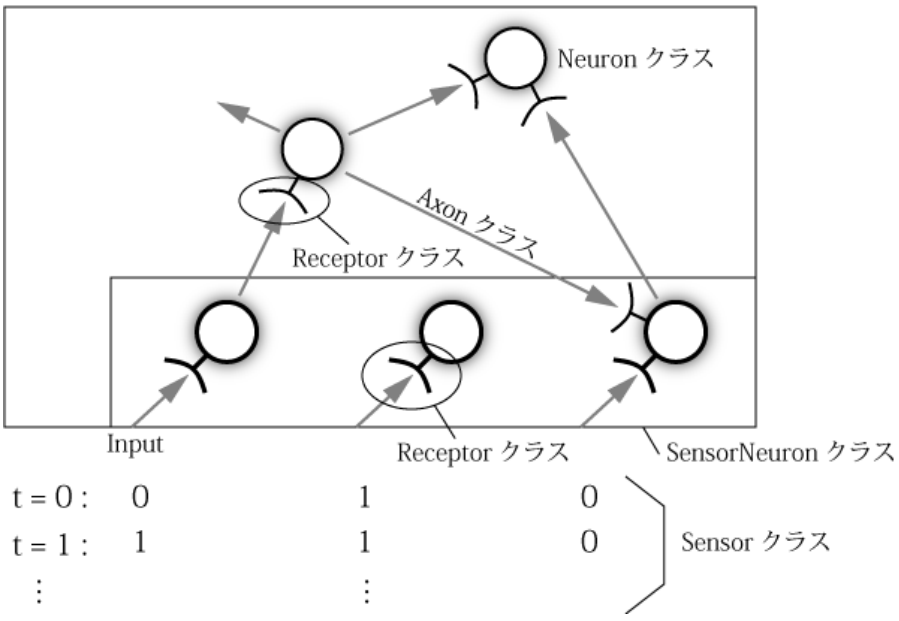


図 47　プログラムモデルの全体像

第６節 アルゴリズム

以下に，本プログラムのアルゴリズムの詳細を示す．また，第５章で設計・分析した各理論モデルに対応する箇所を示し，本アルゴリズムが各モデルの特性を実装していることを確認する．ただし，あまりにプログラムの細部に関わる，本研究の本質的な論理と関係のない手順については，便宜のためしばしば省略した．

(1) 入力データを読み込み，フィーチャーマップを生成する．フィーチャーマップは，データのトークンをセンサーベクトルに対応させるマップである．センサーベクトルはひとつのデータの観測に対応するベクトルである．

[例]

訓練データ：{a, b, c}{a,b,d}

フィーチャーマップ：a -> {1,0,0,0}, b-> {0,1,0,0}, c -> {0,0,1,0} d-> {0,0,0,1}

(2) フィーチャーマップを用いて，入力データをセンサーベクトルパッケージリストに変換する． センサーベクトルパッケージは，ひとつの訓練データに対応するベクトルのベクトルであり，センサーベクトルパッケージリストはそれをベクトル化したものである．また，フィーチャーマップの要素数のセンサーニューロンを生成する．

対応するモデル：Model 4. AI_DelaySensor

センサーベクトルパッケージリストをフィールドに与え，物理時間 t を定義して，反復試行を開始する．

(3) フィールドは，初期のニューラルネットワークとして，各ニューロン間にシナプス網を形成する．形成するシナプスの形状は，完全グラフ，センサー間にシナプスを生じない完全グラフ，二部グラフ，確率的二部グラフなど，コンスタントクラスであらかじめ指定する．ここでは確率に基づくランダムなグラフとする．これはループ構造を許可するもので，理論モデルとしては AI_PerfectMultiLayer に対応するものであるが，物理時間が定義されており，ニューロンは順次発火していくため，計算時間上の問題は生じない．

対応するモデル：Model 3-c. AI_PerfectMultiLayer

(4) 最初のセンサーベクトルを読み込み，センサーニューロンにセットする．

(5) すべてのニューロンについて，以下の処理をタイムテーブルにセットする．

NEURON_GET_PSP：シナプス後電位を，時間遅れ後に取得する．

NEURON_RENEW_MEMBRANE_POTENTIAL：取得したシナプス後電位に基づき，自身の内部膜電位を更新する．

NEURON_FIRE：ニューロンの発火判定を行い，発火した場合には軸索へ値を受け渡す．

スパイクニューロンモデル同様，ニューロンは内部膜電位を持ち，この値によって確率的に発火する．ニューロンｉの時刻ｔにおける内部膜電位を$h_i(t)$とすると，時刻ｔにニューロンｉが発火する確率は，次のように書ける．

$$P[S_i(t) = 1|h_i(t)] = \frac{1}{1 + e^{-\beta\{h_i(t)-\theta\}}}$$

NEURON_VOLTAGE_DROP：発火した場合には，自身に電圧降下を発生させる．

NEURON_VOLTAGE_DROP_ATTENUATION：電圧降下値の自然減衰を行う．

NEURON_SET_REWARD：自身の発火リワードを，各受容体にセットする．

これは，自身の発火を予測した（自身より早く発火した）ニューロンに対して与えられるものであり，したがって各受容体のシナプス後電位に基づいて配分される．シナプス後電位の大きい受容体に接続している軸索の接続元ニューロンは，自身より早く発火している可能性が高く，予測に寄与したニューロンと考えてよい．

対応するモデル：Model 2. AI_RewardBack

NEURON_GET_REWARD：軸索を通じて接続先の受容体にセットされたリワードを受け取る．

(6) すべての軸索について，以下の処理をタイムテーブルにセットする．

AXON_SEND_VALUE_TO_RECEPTOR：ニューロンから受け取った値を，自身の重みを乗じて，時間遅れ後に接続先受容体へ送る．

AXON_MODIFY_WEIGHT：自身の重みを，受容体から受け取ったリワードに基づいて更新する．

(7) すべての受容体について，以下の処理をタイムテーブルにセットする．

RECEOTOR_PSP_ATTENUATION：自身のシナプス後電位を，減衰時定数に基づいて自然減衰させる．

(8) 一定時間経過した後，検査を行う．

REWARD_VALIDATION：ニューロンは，自分に軸索を伸ばしているニューロンのうち，自身より先に発火したニューロンに対して，リワードを与えている．このリワードは検査までニューロンに蓄積され，検査時にニューロンはそのリワードを用いて後方軸索の重みを更新する．あるニューロンが，接続先ニューロンの予測に成功した場合，予測に貢献したのは，接続前ニューロンを発火せしめた軸索である．したがって，軸索の更新モデルは AI_RewardBack と同様である（図 48）．

またこのとき，リワードは神経科学的には，脳内の血量に相当するものとして扱われると考えられる．

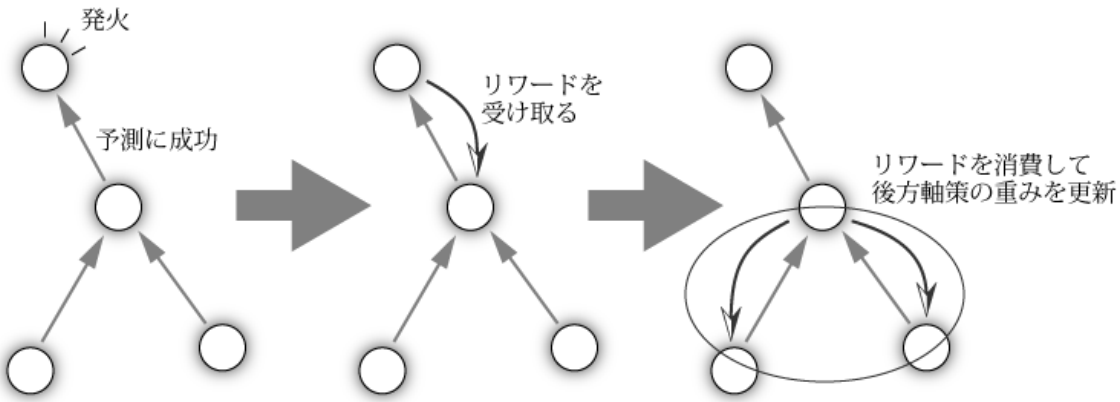


図 48　軸索の重み更新ロジック

FIELD_VALIDATION：次に，構造の更新を行う．まず，時間遅れで発火したニューロン間に軸索を形成する（図 49）．

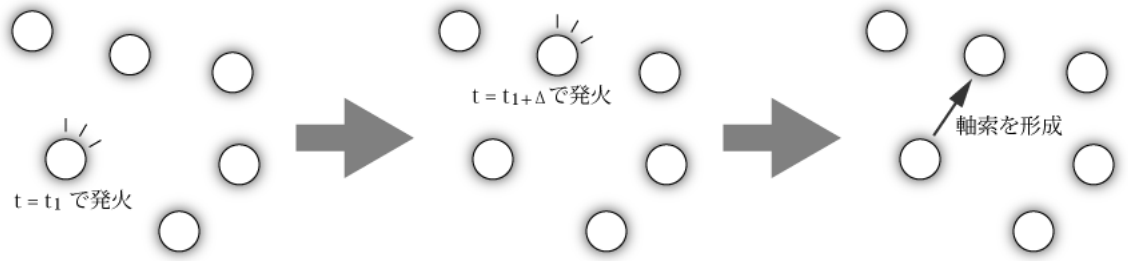


図 49　軸索の生成ロジック

このとき，ニューロン間の発火の時間遅れが，軸索のパルス伝達時間遅れに反映される．

次に，軸索の重みを減衰時定数にしたがって自然減衰させ，その重みが一定の閾値以下になった軸索を消滅させる．これは，予測に役立たない構造を淘汰するモデルであり，理論モデルとしては，AI_Darwinism に対応している．

神経科学的にも，血流が少なくなり，その結果細くなった軸索が消滅するというプロセスは，妥当なものと考えられる．

対応するモデル：Model 1. AI_Darwinism

以上の手順を，センサーベクトルパッケージリストの最後まで繰り返す．

常に次の観測データを予測しながら，自己の構造化を行っていくニューラルネットワークのアルゴリズムが実現されていることがわかる．

第７節 パラメータ

本節では，アルゴリズムの主要なパラメータを示し，そのアルゴリズムにおける役割を記述する．

第１項 パラメータ

Const for Trial Parameter

反復試行回数：TRIAL_ITERATION

初期ニューロンの総数：NUM_OF_TOTAL_NEURON

Const for Neuron

伝播リワードの基準値に対する比重：

NEURON_BACKPROPAGATE_REWARD_WEIGHT

▲大きくすると，より後方のニューロンまでリワードが伝達されるようになります．

ニューロン発火閾値の抑制更新傾き：

NEURON_THRESHOLD_MODIFY_BIAS_INHIBITION

▲大きくすると発火過多のニューロンの閾値上昇率が大きくなります。

ニューロン発火閾値の励起更新傾き：

NEURON_THRESHOLD_MODIFY_BIAS_EXCITATION

▲大きくすると発火不足のニューロンの閾値減少率が大きくなります。

Const for Axon

軸索の重みの自然減衰時定数：AXON_NATURAL_ATTENUATION

▲小さくするとシナプス前ニューロンの過去の発火に対する評価が高くなります。

軸索のパルス伝達の時間遅れ：AXON_PALSE_DELAY_DEFAULT

▲軸索のパルス伝達の時間遅れのデフォルト.

Const for Receptor

シナプス後電位の減衰時定数：RECEPTOR_PSP_ATTENUATION

▲大きくするとシナプス前ニューロンの過去の発火に対する依存度が小さくなります。

Const for Validation

軸索生成の閾値：CREATE_SYNAPSE_THRESHOLD

▲大きくするとニューロン間に軸索が生成されにくくなります。

ニューロン間スコアの減衰時定数：INTERNEURON_SCORE_TIMECONST

▲大きくすると発火遅れの時間を厳しく見積もります。

その他，二次的なパラメータ

反復処理の時間間隔：INTERVAL_OF_ITERATION

検証処理の時間間隔：INTERVAL_OF_VALIDATION

センサーベクトルの読込間隔：INTERVAL_OF_SENSOR_VECTOR

センサーベクトルパッケージの読込間隔：

INTERVAL_OF_SENSOR_VECTOR_PACKAGE

ニューロンの内部ノイズ：NEURON_INTERNAL_NOIZE

▲大きくすると発火確率の関数勾配が急になり発火の確率依存度が減少します。

ニューロン内部膜電位のデフォルト：

NEURON_MEMBRANE_POTENTIAL_DEFAULT

▲大きくすると初期のニューロンが発火しやすくなります。

内部膜電位更新の遅延時間：

NEURON_MAMBRANE_POTENTIAL_DELAY_DEFAULT

▲大きくするとニューロンの反応が鈍くなります。

ニューロン発火閾値のデフォルト：NEURON_THRESHOLD_DEFAULT_VALUE

▲大きくすると初期のニューロンが発火にしくくなります。

パルス発生後電圧降下の振幅：NEURON_VOLTAGE_DROP_AMPLITUDE
パルス発生後電圧降下の時定数：NEURON_VOLTAGE_DROP_ATTENUATION
パルス発生後電圧降下の遅延時間：NEURON_VOLTAGE_DROP_DELAY_DEFAULT
軸索消滅の重み閾値：AXON_WEIGHT_ELIMINATE_THRESHOLD

第８節 動作確認と考察

構築したプログラムについて，簡単な動作実験を行った．第５章で用いたような人工の訓練データを用いてアルゴリズムの挙動を確認し，正常に動作していることを確認したが，今回の研究では正式な言語コーパスを使っての実験までには至らなかった．

しかし，直前の観測データ以外の過去の観測データから，予測に役立つものへ軸索を形成して次の観測を予測しようとしている中間ニューロンが生成され，「is」と「are」を特に多く含む恣意的な訓練データに対して，{i,s}{a,r,e}をそれぞれ抽象化する中間ニューロンの生成が確認されるなど，プログラムは大枠で当初意図した通りに機能しており，パラメータの調整を要するものの，パターンを持ったデータの構造化能力を持っていると見てよい．

また本アルゴリズムにおいては，すべてのニューロンが等価に扱われていることを加味すれば，これら一段階の抽象化が行われたのちに，軸索固定のロジックが整えられるか，あるいはそのようなロジックの必要もなく，中間ニューロンが第二のセンサーとして振舞うことで，抽象概念の抽象化という，脳が行っている高次の抽象化を行うことができるようになることが期待される．

第７章 適用可能性と考察

この章では，本研究について前章までの実験結果から主張できることを確認し，その社会的な適用可能性を示した上で，課題及び展望を含めた全体的な考察を行う．

第１節 システムの有用性について

本研究の提案アルゴリズムは，与えられたデータに内在するパターンを認識し，中間ニューロンによってその抽象化を行うものである．これは本アルゴリズムがデータの種類に関わらず，そのパターンを発見できるほど充分な観測があれば，論文における引用・被引用関係といったような関係が明示的に示されていなくても，自動的に構造化を行うことが可能であることを示している．

本アルゴリズムによる抽象化された中間ニューロンの生成は，こちらが恣意的に定めたルールに則ったものではなく，与えた大量のデータについて，各データの予測に重要な中間ニューロンを勝手に提出してくるものである．これは元のデータドメインにおいても非常に重要なデータの集まりとして，何らかの抽象概念を示しているはずである．従来の俯瞰マップでは，抽象化は人がそこに直感的に見出し理解していたに過ぎなかったが，予測の仕組みと直接リンクさせたアルゴリズムによって，自動的に出現した抽象概念は，予測性というネットワークの本来の意義に近く，人にとって利用の可能性が高いと考えられる．

第２節 提案アルゴリズムの特徴

本研究の提案アルゴリズムは，入力データのパターンを認識し，自己構造化を行うことでネットワークに階層記憶システムを構成するものである．したがって，入力データは何らかのパターンを内包するものでありさえすれば，どのようなものでもよい．

本研究の実験では，文章データを入力とし，アルファベット一文字をひとつのセンサーニューロンに対応させていたが，データは必ずしも文章である必要はなく，入力も文字単位である必要はない．

本アルゴリズムのこのセンサーの多様性は，実際のデータへの適用に際して非常に重要である．というのは，同じデータ群に適用しても，センサーの定め方によって，アルゴリズムの階層記憶が内部に構築する予測性の構造は異なったものになるからである．言い換えれば，同じデータ群に対しても，入力単位を変えることによって，何を抽象化

するかを決めることができるということである．

文字の抽象化を行って単語を得ることは，文字という入力単位から容易に想像されることであるが，ほかにも様々なデータを入力単位とすることができるだろう．

従来の単語分析やクラスタリングと決定的に違う点は，様々な種類や形式の異なるデータに連続的に適用するときでも，一切の人為的な処置が不要なことである．従来手法で言えば，ストップワードも人為的な処置にあたる．本アルゴリズムであれば，従来手法ではストップワードに入っていても重要だと思われる語は，そうでないものは予測に寄与しないため，結果的に何も影響を及ぼさない．また，特定の語の組み合わせが，組み合わされたときのみある意味を持つような場合，従来手法ではそれをひとつひとつ特徴（素性）として処理する必要があったが，本アルゴリズムではその必要はない．ひとつの単語も，コンテキストに応じて複数のトピックに分解されうるからである．ある単語とくっついたときのみ，特別の意味を生じるような現象は，階層記憶システムのアルゴリズムにおいては，むしろ自然なことである．

このように，与えるデータの形式を問わない汎用性と，組み合わせによる特徴を自動的に構造の中から類推する拡張性も，本アルゴリズムの大きな特徴のひとつである．

第３節 適用可能性

本アルゴリズムを適用すべき，構造化したいデータをいくつか例に上げ，それらに対してどのような抽象化が行われ，その結果どのような知見が得られうるかを考察する．

特許

特許のメタデータには，発明者，出願人・機関，国際特許分類（IPC：International Patent Classification）などがある．これらのメタデータを提案アルゴリズムの入力として与えることを考える（図 50）．

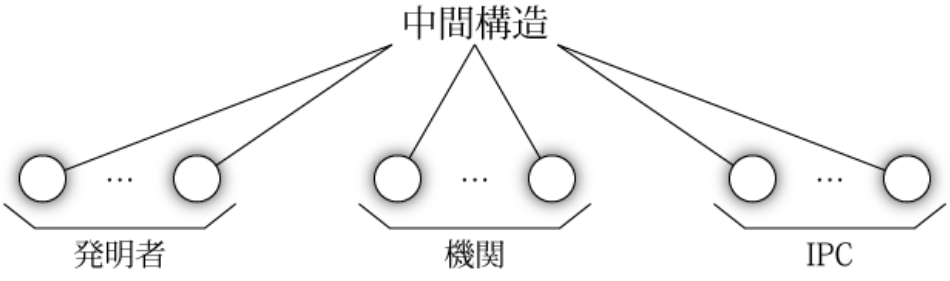


図 50　特許データへの構造化アルゴリズムの適用例

アルゴリズムは，与えられたデータの一部から，与えられない部分を予測するように構造化されている．したがって構造内部に，たとえば「国際特許分類 G（物理学)」を予測するような中間ニューロンを持っている．この中間ニューロンが表しているのは，物理学という技術分類特許を予測するような発明者，機関のデータのあつまりである．これは，物理学周辺で活躍する研究者のグループや機関を表していると考えられる（図 51）.

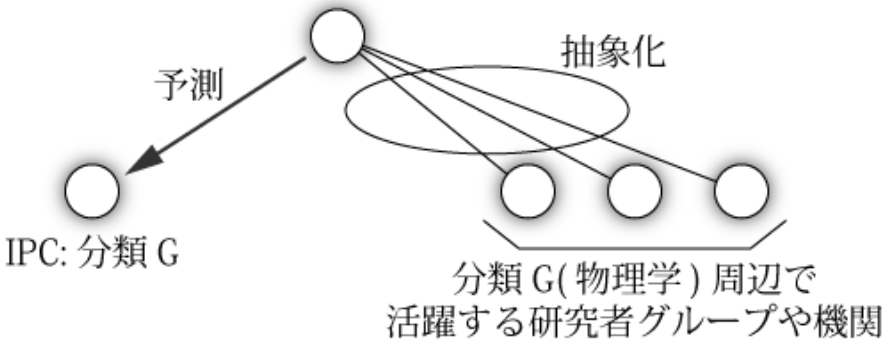


図 51　特許におけるデータ抽象化の例（1）

同様に「発明者 X」を予測している中間ニューロンは，発明者 X をキーパーソンとするような技術分野や，機関のあつまりを表していると考えられる（）.

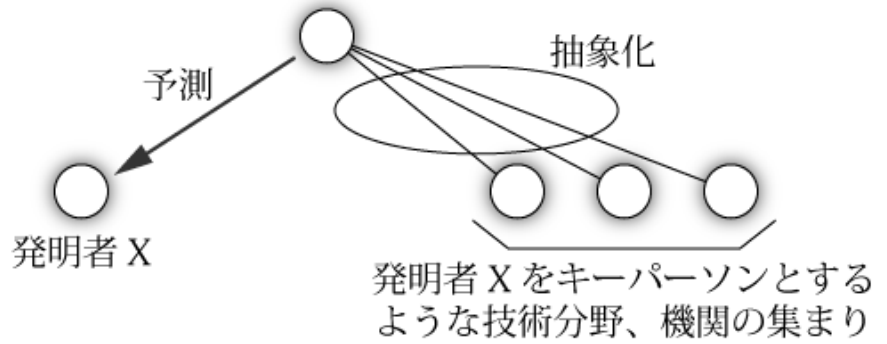


図 52　特許におけるデータ抽象化の例（2）

「機関 A」を予測している中間ニューロンは，ある一機関が手がけている技術分野の集合と考えれば，産業的に近い領域にある技術分野のあつまりであると見ることができ，ある機関の周辺に所属する研究者のグループと考えれば，その機関に関連して何らかの共同研究を行っているグループを表していると見ることができる．いずれも，特許分析に際して我々が知りたいと思うような事柄ばかりである．

論文

学術俯瞰マップが引用関係によってクラスタリングを行っている論文データはどうだろうか.

俯瞰マップ同様，論文そのものを入力とする場合を考えてみよう．訓練データは，論文中の引用論文リストである．このとき「論文 A」を予測している中間ニューロンが表しているのは，「論文 A と一緒によく引用される論文」の集合であると考えられる．したがって，それらの論文は（論文 A も含めて）近い研究領域の論文である可能性が高い．これは，学術俯瞰マップが可視化している情報と酷似している.

また，このアルゴリズムに，ある論文 X の引用論文をすべて入力として与えると，ひとつ以上の予測＝論文が返ってくるはずである．この論文は，論文 X が「引用していてもいいはずの」論文であると考えられる.

次に，アブストラクトと著者，所属機関を入力データとすると，どうなるだろうか. アブストラクトは単語単位か文字単位で与え, 単語や文字の抽象化は出来ているものとする. このとき, あるアブストラクトを予測している中間ニューロンが表しているのは, そのアブストラクトの内容に似ている研究をしている研究者のグループや, その手の研究が活発な機関の集合であると考えられる．これは関連研究を探すときなどに，かなり有用な情報である.

企業情報

企業情報の入力として考えられるものはいくらでもあるが, 一例として取引先を考えてみれば, 即座に論文における引用と同様の発想で, 潜在的な取引先を発見することができることがわかる.

種々の財務指標や業績評価指標などのデータを大量に与えてパターンを学習させた上で，企業の特徴や制度，固有の情報を与えれば，どのような特徴がどの指標に影響を与えているか，その相関を知るためにも使うことができるだろう．また，階層記憶システムの内部状態を覗き込み, ひとつの指標を予測している中間ニューロンに着目すれば, 特定の指標の周辺で他の指標がどのように変動するかのパターンを発見することもできるはずである.

企業に関連する情報を片端から入力していくだけで, 自由な視点から予測を可能にする情報の構造化ができるとすれば，経営ツールとしてもこれほど有用なツールはない.

ニュース

単語を入力とすれば，単純なニュースのカテゴリー分類問題と解くこともできるし，ニュース自体を入力とすれば，社会現象の俯瞰図を作ることも可能になる．抽象化の仕組みに関しては既に述べたので，ここでは繰り返さないが，たとえば世界各地のニュース群をその地図情報と同時に入力とすることで，ニュースやそれに関連する社会現象の地域性を知ることができ，問題のグローバル性やローカル性が明らかになる．

気象情報

世界各地にセンサーを点在させることで，人間の網膜のようなセンサー群として，そこからの入力パターンを学習すれば，局地的なパターン，広域的なパターン，何時間，何年，何十年にわたる緩急様々な変化のパターン，どのようなパターンでもアルゴリズムは発見することが可能である．それによって，特定地域の天気の予測から，広い帯域の気象現象予測まで，様々な気象情報の予測を立てることができるようになる．

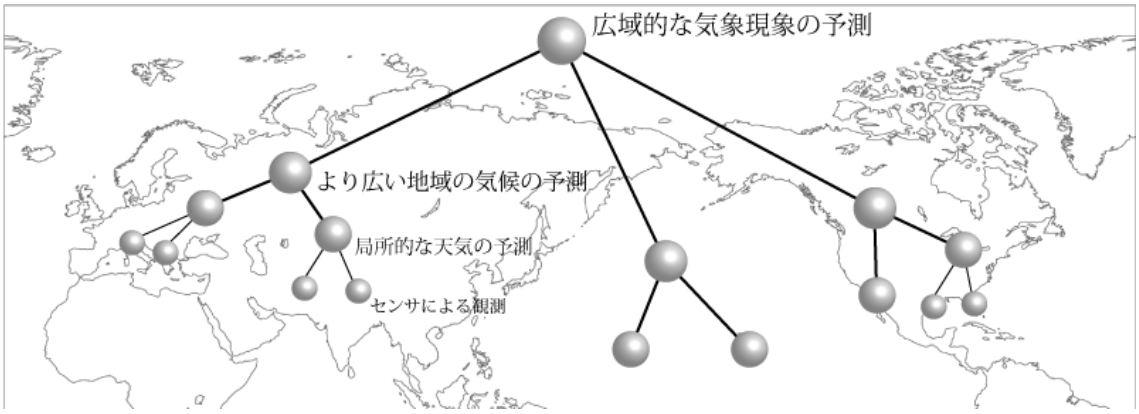


図 53　気象情報におけるアルゴリズム適用例

高分子化学

タンパク質を構成するアミノ酸の配列を入力として学習すれば，タンパク質がどのように折り重なって分子の形状を形成し，相互作用するかを予測出来るようになる．このことは，多くの病気の治療法確立に貢献するはずである．

以上に挙げた例も，情報自動構造化アルゴリズムの持つポテンシャルのごく一部を例示したに過ぎない．このように非常に多岐にわたる領域で有用に利用できながら，それがすべて単一なアルゴリズムによって実現可能である，すなわちどの領域においても専門家や既存の知識をただのひとつも要しないところが，本アルゴリズムの最大の有用性

であり，従来手法にはなかった画期的な点である．社会的なインパクトは極めて大きいと言える．

第４節 課題と展望

本研究では，自己連想記憶理論に基づく補助問題の生成による自己構造化や，神経ダーウィニズムの原理がネットワークによる情報構造化の基礎原理として存在していることを明らかにしたが，そのアルゴリズムを用いて直接俯瞰マップに貢献するまでは至っていない．今後，本研究で得られた知見をどのようにして俯瞰マップ，あるいはそれに類似する情報の構造化手法に応用し，新たな情報構造化ツールとして確立していくかという課題が残されている．

その第一歩として構築した，提案アルゴリズムを生体的制約下で実装したプログラムについても，軸索の固定ロジックや生成ロジック，高次の抽象化をより少ないデータセットから確実に行うためにはどのようなパラメータ設定がふさわしいかなど，残された課題を検討しなければならない．

本研究が，本章で考察したような社会的インパクトの極めて大きい数々の貢献を可能とする，あらゆる種類のデータを自動的に構造化するシステムを実現するための橋頭堡となり，今後より実用的な形の階層記憶システムが構築されることを期待する．

第８章 結論

本研究は，大量の情報を構造化する必要が生じていることを背景に，現行の情報構造化ツールである学術俯瞰マップを考察した．その結果，情報構造化の有用性はノードをクラスタリングすることでデータを抽象化し，人に予測性を与えるという機能にあることを明らかにした．一方，学術俯瞰マップやそれに類するネットワーク図の従来手法は，この予測性を目的に据えたものではないことを踏まえ，これらを本質的に改良する手法として，予測性を目的関数としてデータの抽象化を行う情報の構造化アルゴリズムを提案した．

そのアルゴリズムを実現するための足がかりとして，ニューラルネットワークと脳の情報処理に着目した．特に，脳は観測されたデータから周辺世界に対する予測を行うことで自己構造化を行っているとする Hawkins の自己連想記憶理論や，脳は進化によって生じたものであり，計画的に設計されたものではないため，脳の機能はダーウィンの集団的思考原理に基づいて説明されるべきであるという Edelman の神経ダーウィニズムに立脚し，これを本研究のベースアイデアとした．

第５章では提案アルゴリズムの妥当性を検証するための実験を行った．バックプロパゲーション実験では，自己連想記憶理論による補助問題の生成と，それを解くことによるネットワークの自己構造化が中間ニューロンに観測の抽象化を行わせていることを明らかにした．また，データの抽象化を行う仕組みを選択淘汰，適応，多層化，時間方向の圧縮という四つの要素に分解し，それぞれの効果を確認するための実験を行った．選択淘汰の実験では，ニューロンの選択淘汰のアルゴリズムのみからネットワークの構造は最適化可能であることを確認した．適応の実験では，前述の選択淘汰モデルに適応のアルゴリズムを追加し，軸索の重み更新ロジックを導入することで，ネットワークはより早く進化できることを確認した．多層化の実験では，三層モデルでは原理的に解き得ない問題を多層モデルでは解けることを確認し，様々な多層化のプロセスを比較検討した．その結果，前世代で生き残った中間ニューロンから軸索を張って多層化させるモデルが，もっとも優れた性能を示すことがわかった．時間方向の圧縮の実験では，観測データをすべて別々のセンサーに入力せず，ひとつのデータにつきひとつのセンサーニューロンを定め，データの観測を時間で減衰させながら加算した実数値を初期値として与えるモデルを考え，それが小規模なデータに対しては多層モデルと変わらない性能を示し，大規模なデータに対しては多層モデルよりも圧倒的に早く最適化されることを確認した．

さらに，これらの理論モデルを包含する，生体的な制約に基づいた統合的アルゴリズムの構築を試みた．構築したプログラムの動作確認を通じて，第５章で理論的に設計・構築したモデルが，神経科学的に可能な形で実装されうることを確認した．

最後に，本アルゴリズムを論文，特許，企業情報，産業情報などのデータに対して適用することで，どのようなデータの抽象化が可能になり，従来得られなかった，または得難かったどのような知見がそこから得られるかを議論した．

以上の実験及び考察から，本研究は，直接的に大量データの構造化には至らないものの，大量データを自動的に構造化するための基盤となるアルゴリズムを提案しており，潜在的な有用性は極めて高いと結論する．本研究の提案アルゴリズムを深めることは，第３節に示したような数々の有益な仕事を実現する，知能を持つシステムの実現に近づいていくことにつながると確信している．

参考文献

1. **Derek J. de Solla Price.** Little Science, Big Science. NewYork：Columbia University Press, 1963. 119.

2. **大野順子.** Sustainability に関する学問俯瞰マップの作成, 東京大学工学部, 2006.

3. **山崎浩.** ニューラルネットワークの近似理論と最適制御への適用. http://markun.cs.shinshu-u.ac.jp/kiso/projects/paper/Doctor/1999/yamazaki/data/doctor.pdf.

4. **岩脇正浩.** 基礎的機械学習手法の比較, 島根大学 総合理工学部 数理･情報システム学科.

5. **麻生秀樹，津田直治，村田昇.** 統計科学のフロンティア 6 パターン認識と学習の統計. 2003.

6. **銅谷賢治.** 計算神経科学への招待 脳の学習機構の理解を目指して 第 2 回. http://www.cns.atr.jp/~doya/naist/mathsci/icns02.pdf.

7. **Sinno Jialin Pan, Qiang Yang.** A Survey on Transfer Learning. 2008.

8. **Wenyuan Dai, Qiang Yang, Guirong Xue, Yong Yu.** Boosting for transfer learning. Corvalis, Oregon, USA：Proceedings of the 24th International Conference on Machine Learning, 2007. 193–200.

9. **Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil.** Multi-task feature learning. Vancouver, British Columbia, Canada：Proceedings of the 19th Annual Conference on Neural Information Processing Systems, 2007. 41–48.

10. **BiancaZadrozny.** Learning and evaluating classifiers under sample selection bias. Banff, Alberta, Canada, July：Proceedings of the 21st International Conference on Machine Learning, 2004.

11. **Jiayuan Huang, Alex Smola, Arthur Gretton, Karsten M. Borgwardt, Bernhard Sch ¨ olkopf.** Correcting sample selection bias by unlabeled data.：Proceedings of the 19th Annual Conference on Neural Information Processing Systems, 2007.

12. **Wenyuan Dai, Qiang Yang, Guirong Xue, and Yong Yu.** Self-taught clustering.：Proceedings of the 25th International Conference of Machine Learning, 2008, 200–207.

13. **Rie Kubota Ando, Tong Zhang.** A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data, Journal of Machine Learning Research 6, 2005. 1817-1853.

14. **Jeff Hawkins, Sandra Blakeslee.** On Intelligence: How a New Understanding of the Brain will Lead to the Creation of Truly Intelligent Machines, Times Books. 2004.

15. **小宮山宏.** 知識の構造化, 株式会社オープンナレッジ, 2004.

16. **永阪崇行.** 自然言語処理を用いた学問分野俯瞰マップに関する研究, 東京大学大学院工学系研究科. 2006.

17. スパイクニューロンモデル シミュレータ（Spike Neuron Model Simulator）. 2000. http://www.neuro.sfc.keio.ac.jp/~masato/jv/spikeDemo/index.html.
18. **Palmes PP, Hayasaka T, Usui S.** Mutation-based genetic neural network, IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council, 2005. vol.16 587-600.
19. **MauriceGeraldEdelman.** Wider than the Sky: The Phenomenal Gift of Consciousness, Yale University Press. 2004.
20. Memory-prediction framework, From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Memory-prediction_framework.

謝辞

本研究を進めるにあたり，東京大学大学院工学系研究科 松尾豊 准教授には，研究の着手から完成まで，終始手厚いご指導と激励を賜りました．毎週の研究会では，曖昧な内容への鋭い指摘と，悩ましい問題を解決に導くアドバイスをくださいました．時に進捗の芳しくないときにも，私が自分から答えを出すのを辛抱強く待ってくださいました．研究者としての心得や，効率的に作業を進める方法など，数多くの助言も今後に役立つものとして心に残っています．松尾先生の下で研究できたことを誇りに思います．心から深謝いたします．

東京大学大学院工学系研究科 総合研究機構俯瞰工学部門 松島克守名誉教授，坂田一郎教授，梶川裕矢講師には，研究室合宿や合同ゼミで研究に関する重要なご指摘を頂きました．ここに感謝いたします．また，松島先生には学部時代からお世話になり，機会のあるごとに物事への取り組み方や心の持ち方，仕事の進め方などについて，生涯忘れられない言葉の数々をいただきました．深く感謝いたします．

助教の森純一郎さん，友部博教さんには，研究のことのみならず私生活のことでも数多くアドバイスを頂き，大変お世話になりました．助教室ではいつも楽しく議論させていただきました．心より感謝いたします．

石原絢 学術研究支援員には，研究室生活を多岐にわたって支えて頂き，大変お世話になりました．事務的な作業を一手に引き受けて頂き，私の不注意による手落ちも度々救っていただきました．終始恵まれた環境で快適な研究生活が送れたのは，石原さんのおかげです．

D1 の山本覚さん，榊剛史さんは，研究内容に関していつも親身に議論してくださいました．山本さんには，毎週の研究会で数多くの有益なリマークを頂きました．榊さんには，研究室内外で親切に接して頂きました．深く感謝いたします．

B4 の丸井淳己君，岡崎真君，末並晃君，アグチバヤル アマルサナー君，潘睿さんとは，研究会で互いの進捗を報告し合い，よい切磋琢磨ができました．皆，快活で前向きな態度で研究に臨んでおり，和気藹々とした研究生活を送れました．また丸井君には度々，研究に必要な数理的知識を補填してもらいました．ここに感謝します．

M1 の早川裕太君，松村憲君，杉本浩司君は，しばしば研究室を訪ねて来てくれました．そのたびに楽しく話をさせて頂きました．また，杉本君の度々の差し入れは，私を含め研究室皆の力になっていました．ここに感謝します．

同級生の小野塚健太君，松本悠揮君は素晴らしい研究仲間でした．小野塚君との実り多い議論のおかげで，研究の方向はしばしば修正されました．また研究室で夜を徹して

の作業のときには，いつでも松本君が居てくれました．広い研究室で寂しい思いをすることもなく，お互いの構想や進捗を語り合いながら，楽しく研究が進められました．ここに感謝します．

弟の牧野啓太には，論文執筆にあたって細々とした作業を手伝ってもらい，大変助けになりました．東京大学医学部医学科 山本晃大君には，本研究のプログラム構築にあたって大変お世話になりました．不慣れな C#のいろはを短期間で教え込んでくれ，実装上の困難には度々アドバイスをくれました．終日のディスカッションにも倦む事なく付き合ってくれました．ここに感謝します．

最後に，長い学生生活を支えてくれた家族と，理解ある友人達に，深く感謝します．皆様のおかげで，一片の悔いもない，充実した楽しい研究生活が送れたことを，心から嬉しく思います．

平成 22 年 02 月 08 日

東京大学大学院工学系研究科技術経営戦略学専攻

松尾研究室　修士二年　牧野 晃典