National Institute of
Advanced Industrial Science
and Technology
AIST

大脳皮質のアルゴリズム BESOM Ver.1.0

産業技術総合研究所テクニカルレポート

AIST09-J00006

一杉裕志

産業技術総合研究所 脳神経情報研究部門

y-ichisugi@aist.go.jp

http://staff.aist.go.jp/y-ichisugi/j-index.html

2009 年 9 月 30 日

独立行政法人 産業技術総合研究所

概 要

BESOM モデルと呼ぶ、大脳皮質の計算論的モデルの詳細アルゴリズムおよび計算機シミュレーションの結果について述べる。このモデルはまだ論文誌で発表する段階にはないが、様々な証拠から、大脳皮質の認識と学習に関する主要な機能を計算機上で効率的に再現する有望なモデルであると考えている。

BESOM は表現力の高い機械学習モデルであるが、学習パラメタの数が多いため、安定動作のためには正則化の機構が不可欠である。今回 BESOM に、近傍学習、条件付確率のノイズ除去、非線形ＩＣＡ、ノード間競合、特徴間競合という５種類の正則化の機構を導入し、計算機シミュレーションにより一部不安定ながら動作を確認した。このモデルは、先行研究におけるいくつかの大脳皮質モデルの本質的特徴を１つに統合したものになっている。

本稿で述べたアルゴリズムを拡張・修正して様々な実験を容易に行えるようにする BESOM の研究支援ツールを、共同研究先に提供可能である。

大脳皮質の機構の解明は、人間のような高い知能を持ったロボットの実現に向けた最も重要なステップである。解明はかなり進んでいるが、現在のところ、細かな未解決の問題がたくさんある。機械学習理論の基礎をある程度理解した工学的センスのある大勢の研究者が、これらの問題に取り組むようになることを期待する。

目 次

第１章　はじめに

神経科学と機械学習に関する膨大な知見を踏まえた上で脳の解明に取り組む研究者は、どういうわけか非常に少ない。筆者はその取り組みを行っており、遠くない将来に、人間のような知能の高いロボットを実現可能にするためのブレークスルーを目指している。

筆者は BESOM モデルと呼ぶ、大脳皮質の神経回路モデルを提案している [2]。BESOM モデルは４つの機械学習技術（自己組織化マップ、ベイジアンネット、独立成分分析、強化学習）をエレガントに組み合わせたもので、脳の機能を再現させるモデルとして計算論的に妥当な特徴を持っている。そして、計算論的に導かれた近似確率伝播アルゴリズムを実行する神経回路は、大脳皮質の主要な解剖学的特徴と非常によく一致しており [1]、大脳皮質の情報処理原理を説明する正しいモデルであることはほぼ間違いないと考えている。また、将来的には計算機上で効率的に実行可能であると考えており、新しい機械学習技術としての工学応用の面でも有望である。モデルの全体像およびその神経科学的・計算機科学的妥当性の詳細については [2] を参照されたい。

これまでの取り組み [2] では、大脳皮質が採用しているデータ構造、アルゴリズム、神経回路での実装に関するモデルを提案してきたが、計算論的モデル、特に大脳皮質全体の目的関数については明らかではなかった。また、アルゴリズムが不完全であるため計算機シミュレーションもできなかった。そこで本稿では、計算論的妥当性を特に重視し、トップダウンに詳細な学習・認識アルゴリズムを導くことを試みる。そして、導いたアルゴリズムによって、大脳皮質が持つと思われる非線形ＩＣＡ、混合分布の学習といった機能が計算機シミュレーションによって定性的に再現されることを示す。

アルゴリズム設計の際には以下の２つの仮説をもっとも重要な指導原理として位置づけている。

1. 大脳皮質はベイジアンネットである。
2. 大脳皮質が採用するアルゴリズムが必要とする計算量とメモリ量は大脳皮質の面積に対してほぼ線形である。

筆者の現在までの経験では、この２つの指導原理は大変有用である。これらは大変強い制約条件なので、これを満たすようにアルゴリズムを設計することは容易でない。しかし、設計したアルゴリズムが、当初予想していなかった機械学習理論的妥当性や神経科学的妥当性を持つことを、あとから発見するという経験を筆者は何度もしている。

本稿の読者は、ベイジアンネット [6]、ＳOM [3][4]、機械学習理論 [21][22] の基礎知識を持っていると想定している。特に、尤度、同時確率、事前分布、正則化、混合分布、交差確認法といった概念を理解していることを前提としている。BESOM についての予備知識や神経科学に関する基礎知識はなくてもよいが、[2] に目を通しておいた方が読みやすいであろう。

筆者が [2] の第１３章で書いたように、ヒトの脳全体に匹敵する計算速度を持つスーパーコンピュータはすでに実現されている。１０～２０年後であれば、より安価な計算機で実現されるようになるだろう。一方で、計算機上でどのようなアルゴリズムを用いれば人間のような知能が実現されるのかについては、現時点ではまだ完全には分かっていない。しかし脳のアルゴリズムを１０～２０年以内に解明することは、多くの研究者が高い目的意識を持って取り組さえすれば、決して不可能なことではないと筆者は考えている。多くの研究者が本稿を読んで、そのような取り組みに興味を持っていただけることを期待している。

本稿の草稿に対して多くの改善のコメントをしていただいた慶應義塾大学の萩原将文先生、長谷川宏聡さん、東京大学の細谷晴夫先生に感謝いたします。

また、研究をエンカレッジしていただいている産総研の内外の多くの研究者の方々にも感謝いたします。

数多くの機械学習理論の研究者の方たちは、健全な探究心のもと、非常に適用範囲が広くかつ精緻な理論体系を構築してきました。その枠組みがあって初めて脳の動作原理が理解可能になってきたと思います。彼らに心からの敬意を表したいと思います。

第2章　BESOM モデルの概要

2.1　BESOM モデルの構成要素

BESOM モデルは脳全体のマクロなスケールの構造から、個々のニューロンの機能というミクロなスケールの構造にいたるまで、幅広く関係している。

BESOM モデルは、現在のところ**ＢＥＳＯＭネット**と強化学習機構の２つの機構からなる。BESOM ネットは図 2.1 のような構造をしている。

BESOM ネットは、**基底**と呼ぶ単位の階層構造で構成される。

基底は、多数の**ノード**から構成される。ノードは確率変数を表す。１つの基底内のノードが表す情報は独立成分分析 (ICA) により互いに独立になる。

異なる階層の基底に含まれるノードどうしはエッジで結ばれる。従って、ノードは非循環有向グラフを構成する。このノードのネットワークはベイジアンネットとして動作する。外界の観測データは最下端のノードの値として与えられる。

ノードは複数の**ユニット**から構成される。ノードは確率変数だが、ユニットはその確率変数が取りうる値に対応する。各ノードは、自己組織化マップ（ＳOM）の競合層でもあり、自分の子ノードからの入力を圧縮する。個々の確率変数の値が持つ意味は、ＳOMによって獲得される。ＳOMの学習結果は、条件付確率表になる。

基底の階層構造、基底内のノードの数、ノード内のユニットの数はすべて最初に与えられ、学習により変化しない。学習により変化するのは、ユニット間の**結合の重み**のみである。

これら BESOM の構成要素は、脳の構成要素と構造的・機能的にうまく対応がつく。基底、ノード、ユニット、結合の重みは、それぞれ大脳皮質の領野、ハイパーコラム、コラム、シナプスに、ほぼ対応する。ヒトの大脳皮質は、２０万個程度のノードから成るベイジアンネットであると考えている。また、各ノードは、10 × 10 個程度のユニットから成る２次元ＳOMであると考えている。

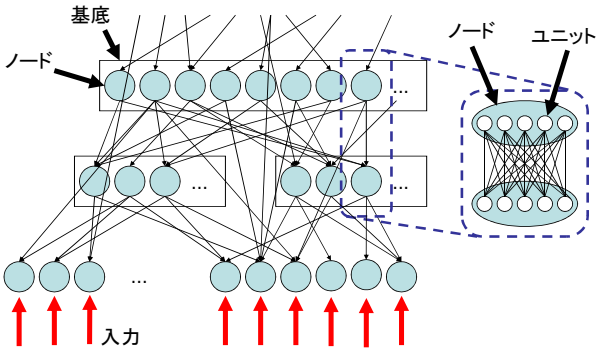


図 2.1: BESOM ネットの構成要素。四角は基底、基底の中の丸はノード、ノードの中の白い丸はユニットを表す。

大脳皮質、SOM、ベイジアンネットの構成要素の対応を簡単に表にまとめると表 2.1 のようになる。

2.2　従来の BESOM の問題点

これまでの BESOM には以下の問題があった。

1. 混合分布を表現できない、別の言い方をすれば、モジュール構造を持っていないという問題があった。脳は、大きく異なるカテゴリの情報は違うモジュールで表現しているのではないかと想像される。新しいカテゴリの知識（例えば新しい外国語など）を学習しても、既存の知識の大半はほとんど影響を受けずに保持されるからである。また、もしモジュール構造がなければ、すべてのシナプスの重みがあらゆる入力刺激に対して一斉に変化するはずである。シナプスの重みの変化はなんらかの生物学的コストを伴うであろうから、すべてのシナプスが常に変化するとしたら、大変効率が悪いことになってしまう。

2. ニューロン間の結合のスパース性による正則化の具体的方法が分からなかった。脳のニューロン間

BESOM	SOM	ベイジアンネット	大脳皮質
ノード	競合層	ノード（確率変数）	ハイパーコラム
ユニット	入力ベクトルの要素、競合層のユニット	確率変数が取りうる値	コラム
親ノード	入力層から見た競合層	親ノード（原因）	上位領野
子ノード	競合層から見た入力層	子ノード（結果）	下位領野
ユニットの出力	入力との類似度	事後確率/MPE	５層錐体細胞の発火率
結合の重み	参照ベクトルの要素	条件付確率	シナプスの重み

表 2.1: BESOM、SOM、ベイジアンネット、大脳皮質の構成要素の対応

の結合はスパースであり、それが正則化の１つ手段であると思われる。３層パーセプトロンでは、例えば weight decay という方法が使われ、結合の重みのほとんどが０になるように学習を進めることで、過適合を避け、汎化能力を向上させることができる。しかし、 BESOM では結合の重みは条件付確率を表しているため、単純に多くの重みを０にすると認識性能がかえって悪くなるという問題がある。

3. スパース符号化モデル [8] との統合がなされていなかった。Ｖ１の単純型細胞の応答は、自然画像をスパース符号化した際の受容野とよく一致していることが知られている。スパース符号化は入力データの統計的性質を利用して効率的に情報を圧縮するため、生物にとって合理的であると考えらる。しかし、BESOM モデルにスパース符号化の機能を持たせる方法はこれまで分からなかった。

4. 確率の計算における計算精度とダイナミックレンジの問題があった。ベイジアンネットを用いた種々の計算は、確率の掛け算を必要とする。もしヒトの大脳皮質の１つの領野が 10000 個のノードから構成されるとすると、通常のベイジアンネットでは 10000 個のオーダーの数の掛け算を行う必要がある。そのような掛け算をオーバーフロー・アンダーフローを起こさずに計算することは、計算機上で浮動小数点を用いる場合でも難しい。対数尤度を用いればダイナミックレンジの問題は解決するが、精度に関しては本質的な解決にはならない。

以上の問題を解決するため、以下の章で BESOM モデルを拡張する。また、基本機能が動作することを計算機シミュレーションで示した上で、大脳皮質のモデルとしての機械学習理論的妥当性と神経科学的妥当性について議論する。

2.3 先行モデルとの関係

BESOM モデルは過去の有名な大脳皮質の神経回路モデルの本質的な特徴を統合したものになっている (図 2.2)。

Malsburg のモデルに始まる一連の一次視覚野のモデルは、Kohonen によって工学的に扱いやすい形に整理され、SOM （Self-Organizing Map, 自己組織化マップ）[3] という教師なし学習アルゴリズムとして広く使われるようになった。

一方、 Fukushima は視覚野の階層構造や単純型細胞・複雑型細胞に関する知見から、ネオコグニトロンおよび Selective attention model[5] というモデルを提案し、実際に頑健なパターン認識を行う能力を持つことをシミュレーションで示した。

ベイジアンネット [6] と呼ばれる工学的な知識表現技術を用いた確率的推論は、Pearl によって効率的なアルゴリズムが開発されたおかげで、今日では様々な分野で応用されている。ベイジアンネットは脳のモデルとして考えられたわけではないが、George と Hawkins は視覚野の階層構造をベイジアンネットとみなす神経回路モデル [7] を提案している。

筆者が提案した最初の BESOM モデル [1] は、ネオコグニトロンおよび Selective attention model の基本構造を踏襲しつつ、ＳＯＭとベイジアンネットを用いてその理論的基礎付けを行ったものである。大脳皮質の学習と認識という基本機能をうまく説明できる上、

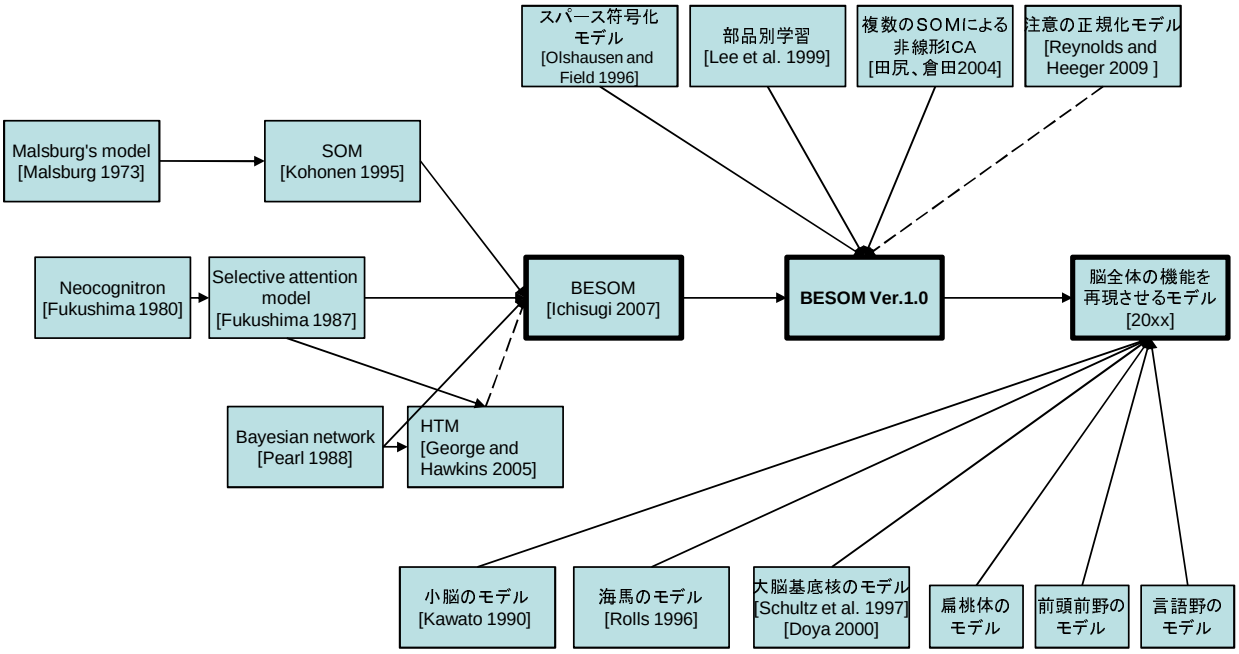


図 2.2: 主な先行モデルと BESOM モデルとの関係

アルゴリズムを実行する神経回路は大脳皮質の６層構造・コラム構造に関する解剖学的知見と非常によい一致を見せた。ただし、この時点では計算機シミュレーションが動いておらず、模式的モデルに過ぎなかった。

筆者が本稿で提案する BESOM Ver.1.0 は、最初の BESOM モデルにさらに以下に述べる４つの大脳皮質モデルの機能を統合したものになっている。

スパース符号化モデル [8] は一次視覚野の単純型細胞の応答特性を再現させるモデルである。

NMF (non-negative matrix factorization) [9] は部品別学習 (parts-based learning) と呼ばれる機能を持つ教師なしアルゴリズムである。大脳皮質もまた、部品別学習を行っていると思われる証拠がある。

田尻らは、複数のＳＯＭを用いて非線形ＩＣＡを行うアルゴリズム [11] を提案している。Oshiro らは、このアルゴリズムと似たアルゴリズムを用いて、ラットの嗅内皮質で発見されているグリッド細胞の性質を再現させることにも成功している [12]。

神経回路モデルではないが、Ｖ４，ＭＴ野などの視覚野のニューロン応答の注意による多種多様な変化を統一的に説明する「注意の正規化モデル」[13] という計算論的モデルが提案されている。

本稿で提案するモデルは、以上のモデルの機能をすべて統合したものになっており、これにより大脳皮質の主要な機能をほぼ再現できると考えている。ただし、現段階で確認されているのは、小規模シミュレーションによるいくつかの基本機能の動作のみである。

知能の高いロボットを実現するには、脳を構成する他の組織の機能も実現しなければならない。また、大脳皮質の基本機能が解明されたとしても、前頭前野や言語野などの個別の領野の機能の再現方法は自明ではない。しかし、これらについては、実はすでにかなり解明が進んでいる。

例えば、大脳基底核は強化学習に関与していることが今日ではほぼ確実である [14][15]。小脳 [16] や海馬 [17] についても、理解がかなり進んでおり、詳細な神経回路モデルが作られている。前頭前野による記号処理とパターン処理を統合した情報処理は、PATON という神経回路モデル [18] により再現されている。言語野による文法獲得は Elman によるもの [19] に代表される多くの研究がある。

これら、すでに解明されている多くのモデルを大脳皮質のモデルと統合することで、脳全体の機能を再現させるモデルが近い将来実現するだろう。

第3章　認識ステップと学習ステップ

3.1　背景

BESOM モデルによれば、大脳皮質の最も基本的な機能は、外界の状態の認識と学習である。大脳皮質は、外界の状態を感覚器官から送られてくる観測データに基づいて推定し、推定結果を学習するという動作を繰り返すことで、外界をよりよく近似する確率的モデルを教師なし学習すると考える。

筆者の以前の取り組み [1][2] では、認識ステップで近似確率伝播アルゴリズムを用いてノードごとの事後確率を計算し、学習ステップで、各ノードが子ノードにおける事後確率最大の値を入力として受け取って学習するという仮説を説明した。しかしネットワーク全体の目的関数が不明であり、意味のある動作をする保証がなかった。そのため、計算機シミュレーションも行っていなかった。

3.2　オンライン学習アルゴリズム

本稿では、大脳皮質の学習の目的はベイジアンネットの構造学習である、すなわち、観測データの尤度を最大とするベイジアンネットを獲得することであるという仮説を提案する。そして、具体的にその目的を達成するオンライン学習アルゴリズムの候補の１つを提案する。このアルゴリズムは、3.3 節で説明する MPE という概念を用いている。提案するアルゴリズムを図 3.1 に示す[1]。

現在動いている小規模シミュレーションは、このアルゴリズムに基づいており、確率伝播アルゴリズムを用いていない。MPE 計算には、9.2 節で述べる山登り法によるアルゴリズムを用いている。

[1] これはおそらく、確率的ＥＭアルゴリズム（参考：[22] p.251）をオンラインアルゴリズムにしたものに相当する。ただし、隠れ変数の値の組み合わせを事後分布からサンプリングする代わりにＭＰＥで近似している。

> 下記の認識ステップと学習ステップを、観測データが与えられるたびに繰り返す。
>
> 1. 認識ステップ：
>   BESOM ネットは、ベイジアンネットとして動作し、入力に対する MPE を求める。（3.3 節。）
> 2. 学習ステップ：
>   BESOM ネットを構成する各ノードが、SOM として動作し、自分の子ノードからの MPE に基づいた入力を学習する。この時、条件付確率表は、入力データの尤度が上がる方向に更新される。（3.4 節。）

図 3.1: BESOM のオンライン学習アルゴリズム

3.3　認識ステップ：MPE の計算

認識ステップでは現在の条件付確率表の値と観測データに基づき、MPE を計算する。

MPE (most probable explanation) とは、ベイジアンネットにおいて、与えられた観測データを最もよく説明する変数の値の組のことである[2]。与えられた観測データを表す確率変数とその値の組の集合を $\boldsymbol{i}$ 、隠れ変数（観測データ以外の確率変数）とその値の組の集合を $\boldsymbol{h}$ とすると、MPE となる値の組 $\boldsymbol{m}$ は次の式で与えられる。

$$\boldsymbol{m} = \underset{\boldsymbol{h}}{\operatorname{argmax}}\, P(\boldsymbol{h}, \boldsymbol{i}) \tag{3.1}$$

ただし $P(\boldsymbol{h}, \boldsymbol{i})$ は $\boldsymbol{h}$ と $\boldsymbol{i}$ との同時確率で、以下の式で表せる。

$$P(\boldsymbol{h}, \boldsymbol{i}) = \prod_{x \in \boldsymbol{h} \cup \boldsymbol{i}} P(x|parents(x)) \tag{3.2}$$

例えば図 3.2 のベイジアンネットにおいて、観測値 $D = d, E = e$ が与えられたとする。求めるＭＰＥは入力との同時確率がもっとも高い隠れ変数 A, B, C の値の組 $\{a, b, c\}$ で、以下の式で表される。

$$\boldsymbol{m} = \underset{\{a,b,c\}}{\operatorname{argmax}}\, P(a, b, c, d, e) \tag{3.3}$$

[2] [6] の p.250 または [22] の p.126 も参照。

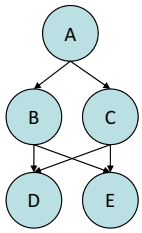


図 3.2: ５つのノードからなるベイジアンネット

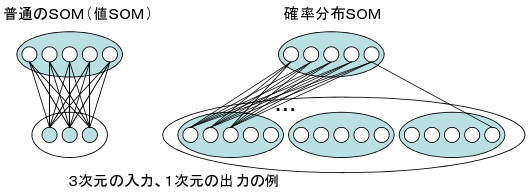


図 3.3: 普通のＳOM（値ＳOM）と確率分布ＳOM。

$$P(a,b,c,d,e) = P(d|b,c)P(e|b,c)P(b|a)P(c|a)P(a) \quad (3.4)$$

3.4 学習ステップ：確率分布ＳＯＭによる条件付確率表の学習

3.4.1 確率分布ＳＯＭ

ＳOMは、競合学習と近傍学習を特徴とする教師なし機械学習アルゴリズムである。ＳOMは、高次元の数値ベクトルの入力を、距離関係を保ったまま次元削減し、低次元のベクトルに変換する方法を学習する。学習した結果は、マップとも呼ばれる。

ＳOMはもともとは、大脳皮質の一次視覚野のコラム構造を再現させる神経回路モデルを、工学的に扱いやすいよう単純化したものである。

普通のＳOMは参照ベクトルが特徴量の値の代表値を学習するが、BESOM で使われるＳOMは、特徴量の確率分布（条件付確率）を学習する。このようなＳOMを**確率分布ＳＯＭ**と呼ぶことにする（図 3.3)。確率分布ＳOMは、入力と出力が同じ形式をしているため、ＳOMの出力を上の階層のＳOMに入力するという階層構造を持たせることができる。

確率分布ＳOMでは、本来連続値である特徴量を離散値で表現する。１つの特徴量は、ただ１つの要素が１の値、他の要素は０の値を持つ s 個の要素からなるベクトルで表現する。例えば、特徴量が [0,1) の範囲の値であれば、i 番目 (i=0,1,...,s-1) の要素が１である場合に、特徴量が区間 [i/s, (i+1)/s) に入る値であることを意味するものとする。

特徴量が n 個ならば、 s 次元のベクトルを n 個連結した ns 次元の入力ベクトルを確率分布ＳOMに与えることになる。参照ベクトルも入力ベクトルと同じ、ns 次元である。たとえば、 n=2, s=10 の場合、２つの特徴量の組 (0.3, 0.7) は、 20 次元の入力ベクトル (0,0,0,1,0, 0,0,0,0,0, 0,0,0,0,0, 0,0,1,0,0) として入力される。

なお、従来の普通のＳOM、すなわち１つの特徴量を１つの値で表現し、 n 個の特徴量を n 次元の入力ベクトルで与え、参照ベクトルも n 次元であるようなＳOMを、ここでは**値ＳＯＭ**と呼ぶことにする。

3.4.2 近傍学習を行わない学習則

学習ステップでは、 BESOM ネットの各ノードは SOM の競合層として働き、子ノードからの入力を圧縮して学習する。ＳOMの学習は競合学習と近傍学習を特徴とするが、この節では確率分布ＳOMの競合学習の部分について説明する。基本的には [1][2] で述べたものと同じ学習則だが、3.2 節で提案したアルゴリズムに合わせて表現を修正した。近傍学習については、4.2 節で述べる。

ノード X が子ノード Y_l $(l = 1, \cdots, n)$ を持つとする。学習ステップでは、SOM は MPE における各子ノードの値を、上で述べた入力ベクトルの形で受け取る。すなわち、ノード Y_l が値 y_i^l $(i = 0, \cdots, s-1)$ を取り得るとすると、Y_l からの入力ベクトル v^l の要素は以下のようになる。

$$v_j^l = \begin{cases} 1 & (\text{MPE における } Y_l \text{ の値が } y_j^l \text{ の場合}) \\ 0 & (\text{その他の場合}) \end{cases} \quad (3.5)$$

ノード X においては、 MPE の値を表すユニットが、競合学習における勝者になる。勝者ユニットでは、通常のＳＯＭと同様、参照ベクトルを入力ベクトルに近づける。ノード X の勝者ユニット x_i とノード Y_l のユニット y_j^l の間の結合の重みを w_{ij}^l とすると、更新式は次のようになる。

$$w_{ij}^l \leftarrow w_{ij}^l + \alpha_i(v_j^l - w_{ij}^l) \tag{3.6}$$

学習率 α_i の値は、 x_i が n 回目の勝者になったときに $1/n$ となるようにする。そのためには、 α_i の初期値を 1 として、x_i が勝者になるたびに以下の式で値を更新すればよい。

$$\alpha_i \leftarrow \alpha_i/(1+\alpha_i) \tag{3.7}$$

3.4.3 結合の重みと条件付確率

学習の後期において近傍学習の効果が十分小さく無視できると仮定すれば、結合の重み w_{ij}^l は条件付確率 $P(Y_l = y_j^l | X = x_i)$ となることが以下のように示される [1][2]。

ノード X のユニット x_i が勝者になった回数を n、X の子ノード Y からの n 回目の入力ベクトルの要素を $v_j(n) \in \{0,1\}$、その学習結果を $w_{ij}(n)$ とする。また、n 回のうち Y のユニット y_j も勝者であった回数を $m(n) = \sum_{i=1}^{n} v_j(i)$ とする。学習率を $a_i = 1/n$ 、また $w_{ij}(1) = v_j(1) = m(1)$ とする。

すると、数学的帰納法により $w_{ij}(n) = m(n)/n$ が示せる。$n = 1$ のときは自明、$n > 1$ のとき、帰納法の仮定により $w_{ij}(n-1) = m(n-1)/(n-1)$ なので、以下が成り立つ。

$$\begin{aligned}
w_{ij}(n) &= w_{ij}(n-1) + \alpha_i(v_j(n) - w_{ij}(n-1)) \\
&= (1-\alpha_i)w_{ij}(n-1) + \alpha_i v_j(n) \\
&= ((n-1)w_{ij}(n-1) + v_j(n))/n \\
&= ((n-1)(m(n-1)/(n-1)) + v_j(n))/n \\
&= (m(n-1) + v_j(n))/n \\
&= m(n)/n
\end{aligned} \tag{3.8}$$

この値は x_i が勝者の時に y_j も勝者であった比率、すなわち条件付確率 $P(Y = y_j | X = x_i)$ である。

なお、 α_i を一定値にしても学習アルゴリズムは動作する。その場合学習されるのは、過去の経験を忘却し最近の経験に比重を置いた条件付確率と解釈できる。

3.5 正則化の必要性

BESOM は学習すべきパラメタ（結合の重み）の数がきわめて多く、認識においても学習においても強い正則化が必要である。

神経科学的知見および BESOM モデルを前提にすると、大脳皮質が実現するベイジアンネットは以下のような特徴を持つと考えられる。

1. （一種の）noisy-OR モデル。
2. 10 段程度の階層構造を持つ。階層内のノード数は固定。同一階層内のノード間にはエッジがない。
3. ネットワークのマクロな構造は事前知識として与えられる。
4. 最下端のノード以外はすべて隠れ変数である。
5. 各隠れ変数は、100 個程度の固定した数の値を持つ。

これらの性質のうち、１、２、３はパラメタの数を減らし、モデルの自由度を下げるが、４、５は自由度を大幅に上げる。全体的には、非常に自由度が高く、かなり強い制約条件を入れないと、学習時に意味のない局所解に陥ることが想像される。

脳は極めて汎用的な学習器であるために、対象となる事象固有の事前知識の作り込みはあまりできない。作り込めるのは「汎用的な事前知識」しかない。そこで以下の章では、「滑らかさ」と「スパース性」を用いた正則化の機構を述べる。滑らかさは、「似た事象は似た信号を発生させる」という自然界に対する事前知識を表している。また、スパース性は、「事象の間の因果関係はスパースである」という自然界に対する事前知識を表している。滑らかさは条件付確率表の近傍学習とノイズ除去の機構によって実現され、スパース性は非線形ＩＣＡ、ノード間競合、特徴間競合の機構によって実現される。

これらの機構の一部は、汎化能力の向上だけでなく、認識ステップの計算量を劇的に減らす役割も果たす。

第4章　近傍学習による条件付確率表の円滑化

4.1　背景

確率分布ＳＯＭを使って、確率分布の連続なマップを獲得させようとすると、近傍学習をどのように行うべきかが問題となる。単純に従来の値ＳＯＭと同様な近傍学習をしても、連続なマップにはならないことが多い。これは、参照ベクトルの次元が n 次元から ns 次元と大幅に増え、機械学習モデルとしてのパラメタの数が増えたために、モデルの自由度が増し、望まれる解以外の膨大な数の局所解が存在するようになったことが理由の１つである。

もう少し詳細に原因を検討すると、参照ベクトルと入力ベクトルの類似度を計算する際、値ＳＯＭの場合と違って、内積計算では必ずしも適切な値とならないことも理由の１つである。例えば、0.3 と 0.4 は近い値であるが、それぞれの値を表す参照ベクトル (0,0,0,1,0,0,0,0,0,0) と入力ベクトル (0,0,0,0,1,0,0,0,0,0) との内積値は 0 であり、まったく類似していないことになってしまう。これでは近傍学習がマップを連続にする効果が出にくい。

例えば、区間の [0,1) の一様分布に従う値 x および $f(x) = (2x-1)^2$ という２つの値を、それぞれ３０次元の２つの特徴量として、１次元の競合層に３０個のユニットを配置した確率分布ＳＯＭ（図 4.1）に入力し

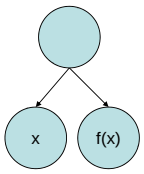


図 4.1: x と f(x) の値の組を学習する BESOM ネット。

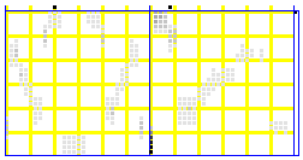


図 4.2: 通常の近傍学習を用いた学習例。

学習させてみると、図 4.2 のように、分断されたグラフが学習される。ここで、図 4.2 の縦軸は競合層（ユニットの並び）を表し、各ユニットの参照ベクトルの要素の値を濃淡で横方向に並べて表現している。（条件付確率表の可視化方法の詳細は 11.2 節参照。）

本来ならば、x と $f(x)$ という２つの値を表すグラフがそれぞれ１本の連続した線として学習されなければならない。

この例のように入力が正しく次元削減されていないと、確率分布ＳＯＭの出力を別の確率分布ＳＯＭに入力し、多層化することができなくなってしまうという問題がある。

この問題への対処として、参照ベクトルと入力ベクトルの類似度の計算式を工夫する方法、近傍学習の仕方を工夫する方法、勝者選択の仕方を工夫する方法などが考えられる。現在のシミュレーションでは、近傍学習の仕方を工夫することで、この問題を解決している。

例えば、特徴量 0.3 を学習後の参照ベクトルが (0.1, 0.2, 0.8, 1, 0.8, 0.2, 0.1, 0, 0, 0, 0) というふうにぼかした表現になるようにしておく。そうすれば、特徴量 0.4 を表す入力ベクトル (0,0,0,0,1,0,0,0,0,0) との内積値は 0.8 であり、２つが比較的近い値であること示すようになる。

そこで、このようなぼかした参照ベクトルを獲得するように、近傍学習の学習則を工夫すればよい。例えば、学習時に、参照ベクトルの値を近づける目標値を、入力ベクトルをぼかした値に設定すればよい。そうすれば、それを学習した参照ベクトルもぼかした表現になる。

4.2 アルゴリズム

前節で述べたアイデアに基づく具体的な学習則を提案する。

ユニット x_i と y_j^l の間の結合の重み w_{ij}^l の、近傍学習を含む学習則は以下のようになる。

$$v_j^l = \frac{1}{Z_b} b(\alpha, d_{x_i}, d_{y_j^l}) \tag{4.1}$$

$$w_{ij}^l \leftarrow w_{ij}^l + \alpha \frac{1}{Z_n} n(\alpha, d_{x_i})(v_j^l - w_{ij}^l) \tag{4.2}$$

ただし、b はぼかし関数、n は近傍関数、d_{x_i} はノード X における勝者ユニットとユニット x_i の間の距離、$d_{y_j^l}$ はノード Y_l における勝者ユニットとユニット y_j^l の間の距離である。ノード X と Y_l の勝者ユニットをそれぞれ x_{w_x}, $y_{w_y}^l$ とすると、d_{x_i}, $d_{y_j^l}$ は下記のように定義される。

$$d_{x_i} = |i - w_x| \tag{4.3}$$

$$d_{y_j^l} = |j - w_y| \tag{4.4}$$

α は t 回目の学習ステップにおける学習率であるが、同時に、ぼかし半径と近傍半径を決めるパラメタであり、t が大きくなるにつれ 0 に近づくようにする。なお、3.4 節で述べたようにユニット x_i ごとに局所的に学習率 α_i を持つことが望ましいが、ここでは簡略化し全体で 1 つとした[1]。

正規化定数 Z_b と Z_n は、それぞれ下記のように定める[2]。

$$Z_b = \sum_{y \in \{y_1, \cdots, y_s\}} b(\alpha, d_x, d_y) \tag{4.5}$$

$$Z_n = \sum_{x \in \{x_1, \cdots, x_s\}} n(\alpha, d_x) \tag{4.6}$$

近傍関数もぼかし関数もともにガウス関数を用いる場合は、以下のようになる。

$$b^{Gauss}(\alpha, d_x, d_y) = g(d_y, C_1\alpha + C_2) \tag{4.7}$$

$$n^{Gauss}(\alpha, d_x) = g(d_x, C_1\alpha + C_2) \tag{4.8}$$

[1]近傍学習と局所学習率 α_i の両方をどう組み合わせれば最適なのかは自明ではない。これは今後の課題とする。少なくとも、n の半径は学習するノードにおける局所学習率、b の半径は入力を送るノードにおける局所学習率に依存すべきだろう。

[2]正規化の方法は [20] に従った。ぼかし関数をこの方法で正規化しておけば、それを学習した結果の条件付確率が $\sum_j P(y_j|x_i) = 1$ という条件を常に満たすようになるという利点がある。近傍関数に関しては、正規化しなければならない理論的必然性はないかもしれない。

$$g(x, \sigma) = exp(-\frac{x^2}{2\sigma^2}) \tag{4.9}$$

また、階段関数を使ってぼかす場合は以下のようになる。

$$b^{Step}(\alpha, d_x, d_y) = step(d_y, C_1\alpha + C_2) \tag{4.10}$$

$$n^{Step}(\alpha, d_x) = step(d_x, C_1\alpha + C_2) \tag{4.11}$$

$$step(d, r) = \begin{cases} 1 & (d < 1 \text{ or } d < r) \\ 0 & (d \geq r) \end{cases} \tag{4.12}$$

階段関数の境界付近で値を連続にする場合は、以下のようにする。

$$b^{SmoothStep}(\alpha, d_x, d_y) = smoothStep(d_y, C_1\alpha + C_2) \tag{4.13}$$

$$n^{SmoothStep}(\alpha, d_x) = smoothStep(d_x, C_1\alpha + C_2) \tag{4.14}$$

$$smoothStep(d, r) = \begin{cases} 1 & (d < 1 \text{ or } d < r - 1) \\ r - d & (r - 1 \leq d < r) \\ 0 & (r \leq d) \end{cases} \tag{4.15}$$

入力をぼかさない普通の近傍学習アルゴリズム場合、ぼかし関数は以下のようになる。

$$b^{NoBlur}(\alpha, d_x, d_y) = \begin{cases} 1 & (d_y < 1) \\ 0 & (d_y \geq 1) \end{cases} \tag{4.16}$$

$$n^{NoBlur}(\alpha, d_x) = g(d_x, C_1\alpha + C_2) \tag{4.17}$$

図 4.2 の学習で用いたのはこのぼかし関数である。

4.3 実験

図 4.3 は、上から順に Gauss($C_1 = s, C_2 = 1$) Step($C_1 = s, C_2 = 2$)、SmoothStep($C_1 = s, C_2 = 2$)（ただし s はノード内のユニット数であり、この実験では $s = 30$）の 3 つの近傍学習アルゴリズムを用いて得られた条件付確率表である。入力は x と $f(x) = (2x - 1)^2$ の値の組であり、x には区間 [0,1) の一様分布から生成される値を与えた。いずれの実験結果でも、連続した確率分布のマップが正しく獲得されている。

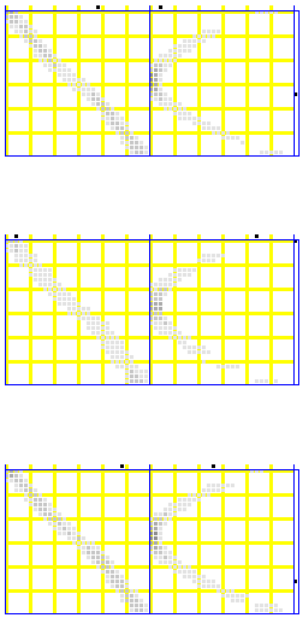


図 4.3: 上から順に、Gauss、Step、SmoothStep を用いた近傍学習の結果。

ここでは、s 個の値を取り得る 2 つの入力ノード Y_l $(l = 1, 2)$ が、区間 [0,1) のアナログ値 a_l を表現するようにする。このとき、確率変数 Y_l の値 y_i^l の i は、以下のように計算する。

$$i = \lfloor (a_l)s \rfloor \tag{4.18}$$

学習率 α は、t 回目の入力に対し以下の値になるようにスケジューリングし、およそ $t = 100,000$ で学習を止めた。

$$\alpha = 1/(0.001t + 1) \tag{4.19}$$

Gauss はもっとも滑らかなマップが獲得されやすいが、計算量が多い。Step は計算量が少ないが、マップが不連続になりやすい。SmoothStep は計算量の割には滑らかなマップが獲得される。Step と SmoothStep では、多くのユニットで学習率 $n(\alpha, d_x)$ の値が 0 になるため、そのユニットの学習をスキップすれば、さらに高速になる。

この後の章のシミュレーションでは、すべて SmoothStep を用いている。

4.4 機械学習理論的妥当性

本章で提案した近傍学習則は、機械学習理論的に正当かどうかを検討する必要がある。

学習の初期には子ノードからの入力も、親ノードへの出力も多くのノイズを含んでいる。確率分布ＳOM の近傍学習則は、条件付確率表をガウス関数や階段関数を使って滑らかにすることで入出力のノイズを除去していると解釈することができる。ぼかし関数は入力のノイズを除去し、近傍関数は次回以降の出力のノイズを除去している。この理解は、 BESOM を多層化する際に重要になる。詳しくは 13.1 節で述べる。

獲得される条件付確率表は、パラメタの事後平均の近似とみなせるであろうと考えている。近似の精度が十分かどうかなどは、今後実験により検証する必要がある。この理解が正しければ、確率分布ＳOMが非常に少ないサンプルに対しても高い汎化性能を出す理論的根拠になる。また、個々の学習対象の性質に応じて近傍学習則を調整し汎化性能を上げる際の指針を与えてくれる。

4.5 神経科学的妥当性

提案する学習則は、神経回路で十分に実現可能である。

シナプス前細胞の出力の強さが d_y 、シナプス後細胞の出力の強さが d_x を表すとしよう。また、局所学習率 α は、そのシナプス周辺のなんらかの化学物質の濃度か、あるいは学習すべきシナプスの近傍にある、学習率を伝えるシナプスの重みだと仮定しよう。すると、シナプスの近傍に、式 (4.1), 式 (4.2) の値の計算に必要なすべての値がそろっている。したがって学習則はこの神経回路モデルで十分に実現可能である。

また、子ノード Y が関数 n の計算に用いる d_y の値は、そっくりそのまま親ノードが関数 b の計算に用いる d_y の値に使えると言う点は注目に値する。このことは、本章で提案する近傍学習のための神経回路が、認識のための神経回路とうまく共有できる可能性を示唆している。

4.6　今後の課題

どのようなぼかし方が最適かは学習対象となるデータの性質に依存する一種の事前知識であるため、理論だけからは決定することはできない。また、このレベルの詳細な神経科学的知見はおそらく得られておらず、電気生理的な実験等で決定することも、あまり容易ではないだろう。最適なぼかし方は、理論的考察および計算機実験により求める必要がある。

本稿の実験ではすべて１次元ＳＯＭを用いているが、実際の大脳皮質はおそらく２次元ＳＯＭを用いている。本章で述べた学習アルゴリズムは全く変更しなくても２次元ＳＯＭに適用できると考えているが、パラメタの再調整が必要になるかもしれない。

第5章　小さい条件付確率のノイズ除去

5.1　背景

一般にある入力ノード Y の観測データ $Y = y$ に対し、$P(y|x)$ が0であると、他の証拠がいかに強く $X = x$ を支持していても事後確率 $BEL(X = x)$ の値が0になってしまうため、実用上問題である。

条件付確率が0になるのは、少ないサンプルから条件付確率表を最尤推定で求める場合に、一般に起こり得る問題であり、一種の過学習である。そこで工学的には、何らかの方法で条件付確率の最低値が0になるのを防ぐことが行われる。これはスムージングと呼ばれる。

例え条件付確率が0でなくても、小さな値であれば、やはり問題となる。ある条件付確率の値が ϵ である場合と 2ϵ である場合では計算される同時確率の値には2倍の違いが出るが、ϵ が小さいことはそれを経験したサンプル数が小さいことを意味し、その情報はノイズの影響を強く受けているはずである。

前章で述べた近傍学習アルゴリズムは条件付確率を滑らかにするもので、一種のスムージングと解釈できるが、あくまで近似的なものなので、小さい条件付確率の値はやはりノイズの影響を受けていると思われる。

また、8章で述べる機構を導入すると、学習により重みの多くが小さな値を取るようになるため、スムージングの機構は必要不可欠になる。

5.2　具体的方法

スムージングの方法はいろいろ考えられるが、1つの方法を以下に示す。認識時に、結合の重みに対して以下の関数 ψ を適用し、条件付確率の最低値が θ となるようにする（図5.1）。

$$P(y_j|x_i) = \psi(w_{ij}) = (1-\theta)w_{ij} + \theta \qquad (5.1)$$

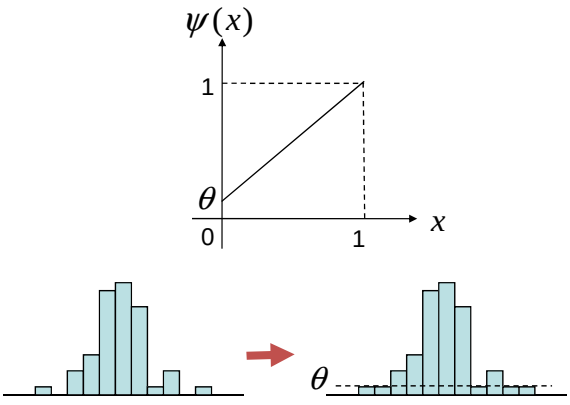


図 5.1: 結合の重みと条件付確率。

本稿のシミュレーションでは $\theta = 0.1$ という値を用いている[1]。

スムージングを学習時に行い、スムージング後の値を記憶する方法も考えられるが、この章で提案する方法では、認識時にスムージングを行っている。この方法は、結合行列がスパース、すなわちほとんどの結合の重みが0である時に、データ構造を工夫することで、メモリの量を減らすことができるという利点がある。生物にとっては、重みが0に近いシナプスを切ってシナプス維持コストを節約することができるという利点がある。

条件付確率の値に最低値を設けることは、同時確率計算にけるダイナミックレンジの問題も緩和する。θ がある程度大きな値であれば、同時確率計算がアンダーフローを起こしにくくなる。

5.3　今後の課題

この方法では ψ を適用することで条件付確率の総和が1を超えてしまうため、何らかの対策が必要かもしれない。

ここではグラフが直線になる関数を用いたが、小さな重みのノイズの影響をより抑えるためには、非線形の関数を用いた方がよいだろう。

[1]学習に使うパラメタ α と同じオーダーで小さくしていくとよいかもしれない。その場合、α と同様に、θ もユニットごとに持たせるべきだろう。現在のシミュレーションでは簡単化のため全体で1つの固定値とした。

第6章　ユニット間の側抑制による非線形ＩＣＡ

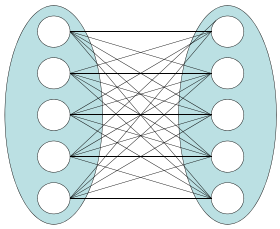


図 6.1: 同一層内にある２つのＳＯＭの各ユニットどうしを、アンチヘブ則で学習する抑制性シナプスを通して結合する回路。

6.1　背景

[2] の６章で述べたように、同一階層内にある兄弟ノードどうしは何らかの非線形独立成分分析 (非線形 ICA) の機構によって独立になっていると考えている。

実際に、大脳皮質が非線形ＩＣＡを行っていると思われる証拠はいくつもある。例えば、サルの側頭葉には、視覚刺激として見せた顔の向きに応答するコラムが見つかっている [10]。顔の向きという情報は、顔の種類や顔の位置、大きさといった情報とは独立な情報であり、大脳皮質にそのようなコラムが存在すると言うことは、大脳皮質がＩＣＡを行っている証拠である。さらに、顔の向きと言う情報から線形の演算だけで顔の視覚刺激を構成することは不可能であるから、これは非線形なＩＣＡである。

筆者は、田尻と倉田によって提案されていた複数のＳＯＭを用いた非線形ＩＣＡの手法 [11] を、確率分布ＳＯＭに適用し、動作を確認したので、以下の節で説明する。

なお、ここでは入力層と隠れ層のみからなる２層ＢＥＳＯＭを扱う。

6.2　アルゴリズム

２つの SOM の間のユニットを、アンチヘブ則で学習するシナプスを通して結合するネットワークを考える（図 6.1)。

アンチヘブ則とは、抑制性シナプスにおけるヘブ則のことで、シナプスの前のニューロンの活動と後のニューロンの活動の相関が強ければ強く抑制し、無相関であれば抑制しないような方向に向かう学習則である。

図 6.1 の回路において、あるユニットの出力が大きくても相関の大きい他のユニットの出力が大きければ強く抑制されるので、出力が弱まり勝者になりにくい。このような条件下でそれぞれのＳＯＭの学習を続けていけば、やがてすべてのＳＯＭのユニットの出力どうしが無相関になる。すなわち、ノードが表現する値どうしが独立になる。

今回実装したアルゴリズムを具体的に説明する。

同じ階層に属する２つの兄弟ノードを X, Y とする。X の値 x_i と Y の値 y_j の、ＭＰＥ（3.3 節参照）になる頻度の相関の度合いを S_{ij} とする。ある入力に対するＭＰＥにおいて $X = x_i, Y = y_j$ （すなわち２つのＳＯＭの勝者ユニットがそれぞれ x_i, y_j ）だとすると S_{ik} $(k = 1, \cdots, s)$ は下記の式に従い更新する。

$$S_{ik} \leftarrow S_{ik} + \alpha^S(\delta_{kj} - S_{ik}) \tag{6.1}$$

ただし δ_{kj} はクロネッカーのデルタ、α^S は学習率である。現在のシミュレーションでは、α^S はすべての S_{ij} で共通の値にしており、下記の式に従って、ＳＯＭの学習率 α と同様に０に近づけている。

$$\alpha^S = 0.03\alpha \tag{6.2}$$

同時確率の計算式 (3.2) は下記のように修正する。

$$P(\boldsymbol{h}, \boldsymbol{i}) = \frac{1}{Z} e^{-\lambda S(\boldsymbol{h})} \prod_{x \in \boldsymbol{h} \cup \boldsymbol{i}} P(x|parents(x)) \tag{6.3}$$

ここで Z は正規化定数、$S(\boldsymbol{h})$ はユニット間の側抑制を表す式、λ は側抑制の強さを表す定数で、本稿では $\lambda = 100$ でシミュレーションを行っている。

２層ＢＥＳＯＭの場合は、$S(\boldsymbol{h})$ は以下のように定

義される。

$$S(\boldsymbol{h}) = \sum_{x,y \in \boldsymbol{h}, x \neq y} S(x,y) \tag{6.4}$$

ただし $S(x,y)$ はノード Y のユニット y からノード X のユニット x への側抑制で、以下の式で定義される。

$$S(x_i, y_j) = S_{ij} \tag{6.5}$$

なお、7 章で述べる φ 値を導入する場合は以下のようにする。(S_{ij} は $i = \phi$ または $j = \phi$ の場合には未定義とする。)

$$S(x_i, y_j) = \begin{cases} S_{ij} & (i \neq \phi, j \neq \phi) \\ 0 & (i = \phi \ or \ j = \phi) \end{cases} \tag{6.6}$$

すなわち、式 (6.3) は、同じ層にある他のノードのＭＰＥ候補の値から、相関の度合いが高いほど強い抑制を受けることを意味している。

したがって、過去において相関の高い値の組み合わせは、徐々にＭＰＥとして選択されにくくなる。そして最後には、どの値がＭＰＥの値になるかはノードごとに独立になる。

6.3 実験

6.3.1 ２次元平面上の１点の入力の学習

この節では、ある確率分布に従って生成される２次元平面上の１点をＢＥＳＯＭに入力し、非線形ＩＣＡを行う実験例について述べる。

学習に用いたのは２層ＢＥＳＯＭであり、隠れ層のノード数は２個、入力層のノード数は $7 \times 7 = 49$ 個である。すべてのノードのユニット数は $s = 9$ である。

入力データは２次元だが、以下に述べる方法で４９次元の特徴ベクトルに変換する。入力データは、区間 [0,1) の２つの値 x, y とする。まず、x 座標と y 座標の区間 [0,1) を７等分する 7 x 7 個の格子点と、座標 (x,y) とのユークリッド距離を計算することで、４９個の数値 d_i $(i = 1, \cdots, 49)$ を得る（図 6.2)。次に d_i から a_i を以下のように計算する。

$$a_i = 0.8 - 3d_i; \tag{6.7}$$

こうして得られた値 a_i から、4.3 節と同じ方法で入力ノード I_i に入力する。(ただし、a_i が 0 より小さい場合は入力ノードの値は 7 章で述べる φ 値とした。)

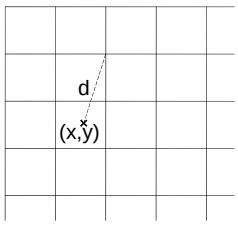


図 6.2: 格子点と入力点 (x,y) との間のユークリッド距離 d 。

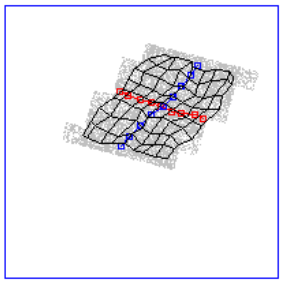


図 6.3: 平行四辺形の形をした確率分布からの入力の学習結果。

なお、現在の実験では隠れ層と入力層の間の条件付確率 $P(Y_j|X_i)$ のみを学習しており、隠れノードの事前確率 $P(X_i)$ は定数だと仮定している。(この後の章の実験も同様である。)

図 6.3、図 6.4 は、それぞれ平行四辺形、扇型をした確率分布から生成される入力を学習した結果である。灰色の背景が入力する点を生成する分布を表している。赤と青の点はそれぞれＳＯＭの９つのユニットの受容野の重心、格子点は 9 x 9 個あるＭＰＥの受容野の重心を表す。(可視化の方法についての詳細は 11.3 節参照。)

図 6.3 では斜交座標、図 6.4 では極座標が獲得されていることが分かる。極座標が獲得できていることは、非線形ＩＣＡができている証拠である。

図 6.5 は $\lambda = 0$ として学習した結果である。すなわちこの例ではユニット間の側抑制を行わない。学習した結果は、意味のないものになっている。２つのノードが表す情報も、独立にはなっていない。ベイジアンネットの学習アルゴリズムとしては $\lambda = 0$ でも問題な

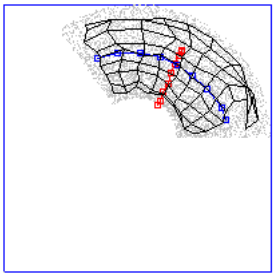


図 6.4: 扇型の確率分布からの入力の学習結果。

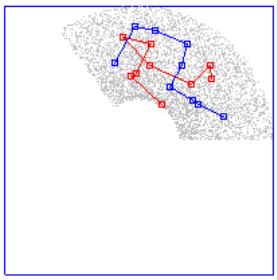


図 6.5: 側抑制を行わない例。 $\lambda = 0$ として図 6.4 と同じ入力を学習させたもの。なお MPE の受容野重心は描画していない。

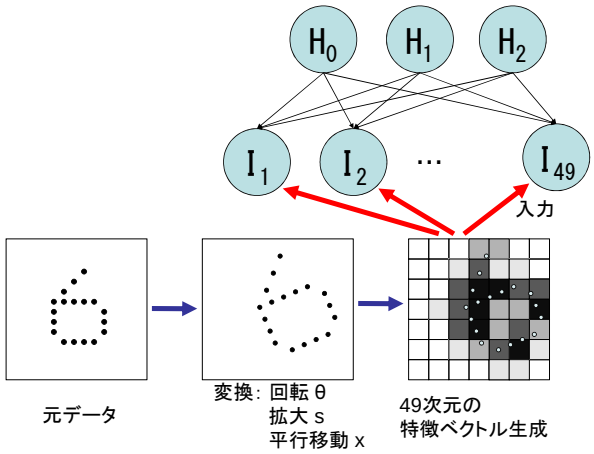


図 6.6: 点の集合からなる入力データを４９次元の特徴に変換する過程。

いはずなので、獲得されたベイジアンネットはおそらく局所解の１つである。このことから、意味のある座標軸を獲得するためには側抑制もしくはそれに代わるＩＣＡの機構が不可欠であることが分かる。

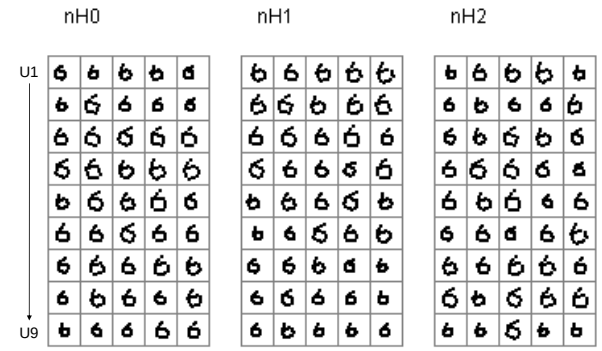
6.3.2 複数の点からなる入力の学習

この節では、人工的に生成した複数の入力点からなる画像をＢＥＳＯＭに入力し学習させた例について説明する。

図 6.7 は、元となる点の集合のデータに対し、回転、拡大、水平方向の移動の３つの変換を施した文字の画像入力（図 6.6）を３つのノードで学習したときの、各ユニットの受容サンプルと受容野である。（受容野の可視化の方法についての詳細は 11.4 節参照。）変換の３つパラメタは、それぞれ一様分布に従って生成される。この図から、３つノード H0, H1, H2 がそれぞれ、平行移動、拡大、回転という３つの独立成分を獲得しているのが分かる。

図 6.8 は、独立に動く３つの点から構成される画像入力を３つのノードで学習した例である。３つのノードが、３つの独立成分を学習している。

図 6.9 は人工的に生成した、４つの点から構成される顔の画像を学習させた例である。顔の画像を生成するパラメタは２つあり、顔の向きと、顔の位置の平行移動量である。ノード H0 が顔の向き、ノード H1 が



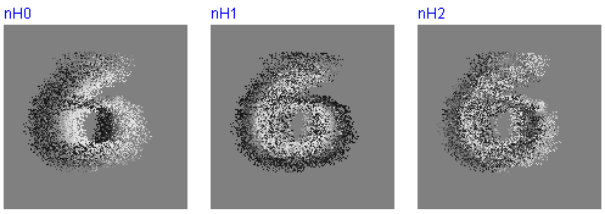
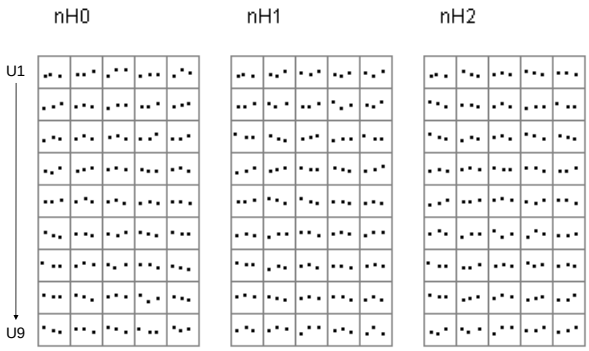


図 6.7: 回転、拡大、水平方向の移動の 3 つの変換を施した文字の画像入力を 3 つのノードで学習したときの、各ユニットの受容サンプルと受容野。上の図は、各ユニットごとに、そのユニットが勝者ユニットとなったときの入力サンプルを 5 個ずつ示したもの。下の図は、ユニット U1 を黒、ユニット U9 を白、その間は灰色のグラデーションを用い、各ユニットが勝者ユニットとなったときの入力サンプルの点を大量にプロットしたもの。



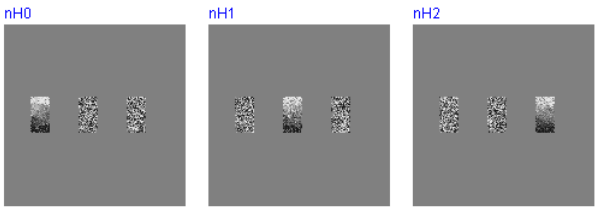


図 6.8: 独立に動く 3 つの点から構成される画像入力を 3 つのノードで学習したときの、各ユニットの受容サンプルと受容野。

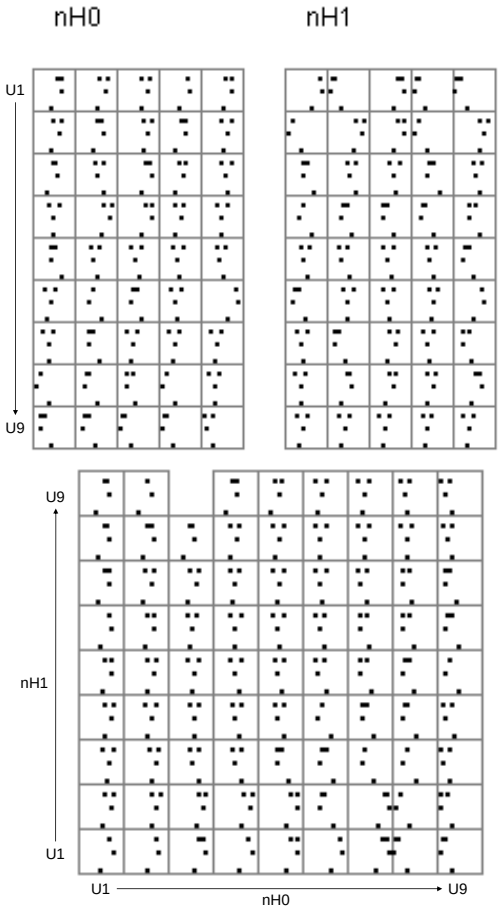


図 6.9: 顔の向きと水平方向の移動で生成される画像を２つのノードで学習したときの、各ユニットの受容サンプルおよび各ＭＰＥの受容サンプル。（注：nH0=U3,nH1=U9 の受容サンプルが欠けているのは対応する入力サンプルが見つからなかったため。）

顔の平行移動量という独立成分をほぼ表現するように学習されている。

6.4 冗長な特徴ベクトルに変換する必要性

図 6.4 は２次元の入力データを４９次元の特徴ベクトルに変換して学習させたが、２次元のまま学習させた結果が図 6.10 である。この例では意味のある軸の獲得に失敗している。この図の受容野重心を見ただけでは分かりにくいが、図 6.11 の条件付確率表を見てみる

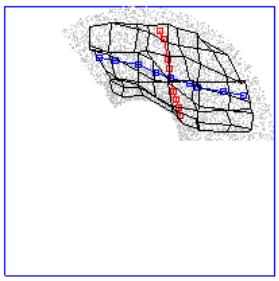


図 6.10: 入力ノードを２つにした学習結果。信号源の獲得に失敗しているが、２つのノードは独立になっているので、非線形ＩＣＡにはなっている。

と分かるように、２つのノードは単に x 軸と y 軸をほぼそのまま学習したにすぎない。なお、この時でも、２つのノードが表す情報は独立になっているので、非線形ＩＣＡとしては成功している。

入力が２次元のままでは信号源の獲得に失敗し、４９次元の冗長な特徴ベクトルにすると成功する理由を説明しよう。極座標を表現するためには、２つのうち１つのノードの各ユニットは、図 6.12 の白い点線で示したような形の受容野を持つ必要がある。しかし、実は２つの子ノードしか持たない１次元確率分布ＳＯＭは、複雑な形の受容野を表現する能力を持っていないのである。一方、４９次元の特徴ベクトルであれば、仮に特徴量と条件付確率が０か１の値しか取らないとしても、2^{49} 通りの受容野を表現することができる。例えば図 6.12 の黒いマスで示した受容野が表現可能であり、これは白い点線の受容野を近似するのに十分である。

この例から分かるように、一般に n 次元の入力ベクトルに対し、任意の受容野を表現可能にするためには $O(2^n)$ 次元の特徴ベクトルを使って冗長に表現しなければならない。この時、条件付確率表を記憶するメモリの量も $O(2^n)$ となる。つまり、受容野の表現力とメモリの量の間にトレードオフがある。

おそらく実用上は $O(2^n)$ 次元にまで冗長にしなくても、十分に意味のある受容野が表現できる場合が多いのではないかと予想する。また、仮に真の信号源の獲得に失敗したとしても、学習された条件付確率表はベイジアンネットとしては正しく意味を持つので、それなりの認識能力を発揮するはずである。この予想の検証は今後の課題である。

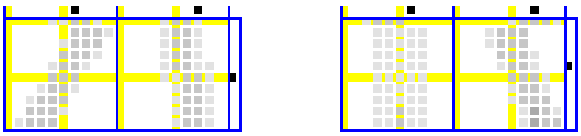


図 6.11: 入力ノードを２つにした時の学習後の条件付確率表。

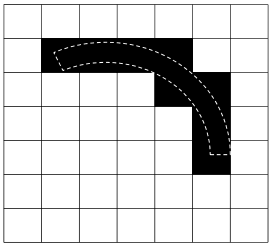


図 6.12: 複雑な形の受容野の例。

6.5　神経科学的妥当性

ここで述べた学習則は、ユニット同士がアンチヘブ則で学習する抑制性シナプスを通して結合することで、神経回路で実現可能である。

抑制性細胞の一種であるシャンデリア細胞が、アンチヘブ則を実現する機構の有望な候補であると考えている。

6.6　今後の課題

ユニット数 s が小さい場合、学習が収束しノードどうしが完全に独立になっても、S の要素が完全に０にならない。これが原因でノード間競合の機構と組み合わせた時に学習が振動してしまう。この問題への対処方法はいろいろ考えられるが、その動作検証は今後の課題である。

この章で述べた学習則では、 α^S の値によっては、学習が振動する場合がある。本来、最急降下法で最適化問題を解く場合は振動は起こり得ないはずである。この現象の原因は、相関の度合いをオンライン学習しているために、その時点の条件付確率表が持つ真の相関の度合いとは時間的に遅れた情報を S が表現しているせいだと思われる。現在は α^S の値を小さくすることで振動を回避しているが、今後、根本的な対処が必要になるかもしれない。例えば、偏微分を使って導いた正しい最急降下アルゴリズムを使う必要があるかもしれない。

現在のアルゴリズムでは２つのノード間に $O(s^2)$ の結合が必要になってしまうが、おそらく $O(s)$ 程度の結合でも十分にＩＣＡは行えるのではないかと考えている。これについては、今後検討する。

相関度の学習でも近傍学習を行えば、汎化能力が向上するだろう。これも今後検証する。

一般に非線形ＩＣＡは解に一意性がない。現在は、ＳＯＭの近傍学習によって「軸が滑らかである」という制約条件を与えることで、それなりに意味のある解が得られている。大脳皮質は、領野ごとに扱う情報が決まっているので、それに合わせてさらに別の制約条件を追加しているだろう。これを明らかにしていくことも今後の重要な課題である。

第7章　ノード間競合による混合分布の学習

7.1　背景

生物が認識すべき外界は、異なる信号を生成させる信号源を混合したものである。例えば、人の顔、木の実の形、捕食者の形などが、それぞれが異なる生成モデルで視覚刺激を生成する。実際の生物の目の前に提示される視覚刺激は、目の前にあるどれか１つの物体が生成したものであるはずである。個々の生成モデルが作る分布は連続なのでＳＯＭで近似すれば補完され汎化能力が上がるが、木の実の形と捕食者の形のようにかけ離れた分布の間は補完するとかえって汎化能力が落ちることが想像される。

7.2　解決のアイデア

混合分布を複数のＳＯＭで近似する場合、図 7.1 のように、個々の連続した領域を別々のＳＯＭで近似するようにすればよい。学習則としては、まずＳＯＭ間の競合により入力にもっとも近いＳＯＭを選んだあと、その入力をＳＯＭ内の競合学習（と近傍学習）で学習すればよいと考えられる。BESOM ではＳＯＭはノードでもあるので、この機構を以下 **ノード間競合**と呼ぶ。

ノード間競合の勝者は必ずしも１つのノードである必要はない。詳細に表現する必要のある対象ほど、多くのノードを使って近似すればよい。（その場合、6 章でのべたＩＣＡの機構により、個々のノードが表す情報は独立になる。）

BESOM モデルとこのノード間競合のアイデアを統合する方法は自明ではない。入力の種類ごとに、ベイジアンネットの異なるサブグラフのみが「活性化」する（図 7.2）ことになるが、通常のベイジアンネットには、このような機能はない。（すべてのノードが常に何らかの値を持ち、エッジを持つ他のノードの値に影響を与えてしまう。）また、神経科学的にも、無理なく実現可能な機構でなければならない。

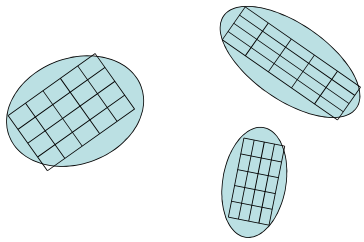


図 7.1: 混合分布を近似する複数のＳＯＭの概念図。

次の節で、このアイデアを実現する特殊な制約条件のついたベイジアンネットを提案する。この特殊なベイジアンネットは神経回路によって生物学的に無理なく実現可能である。神経科学的妥当性の詳細については、さらに後に続く節で考察する。

7.3　アルゴリズム

7.3.1　学習するベイジアンネットの特徴

本稿では入力ノードからなる層と隠れノードからなる層を持つ２層ＢＥＳＯＭのみを扱う。層の中のノード同士はエッジを持たない。

本章で提案するアルゴリズムが獲得するベイジアンネットは、さらに以下の制約を満たすものに限定する。

1. 全てのノード（確率変数） X は、次に示す $s+1$ 個の値のうちのどれかをとるものとする。

$$X \in \{x_\phi, x_1, x_2, \cdots, x_{s-1}, x_s\} \tag{7.1}$$

（ノードごとに s の値が異なっていてもよいのだが、簡単のため、本稿ではすべてのノードで s は同一とする。）以下、 値 x_ϕ のことを **ϕ 値**、x_ϕ 以外の値のことを**非 ϕ 値**と呼ぶ。

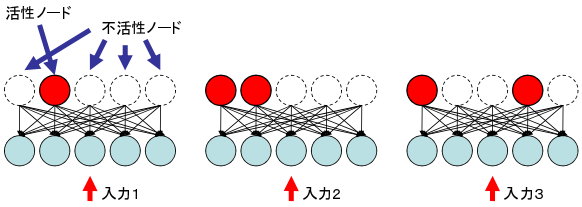


図 7.2: 混合分布を表現する BESOM ネットの概念図。入力ごとに、活性化する親ノードの集合が異なる。不活性なノードは、あたかも存在しないかのようにふるまう必要がある。

2. 条件付確率が下記のように近似できるものとする。（この近似式は将来変更する予定である。詳しくは 7.6.3 節参照。）

$$P(x_i|u_1,\cdots,u_m) \approx \frac{1}{Z}(1-\prod_{k=1}^{m}(1-P(x_i|u_k))) \tag{7.2}$$

ただし Z は $\sum_i P(x_i|u_1,\cdots,u_m)=1$ にするための正規化定数で、下記の式で表せる。

$$Z=\sum_{i=\phi,1,\cdots,s}(1-\prod_{k=1}^{m}(1-P(x_i|u_k))) \tag{7.3}$$

3. ノード X における非φ値 x_i $(i=1,\cdots,s)$ の事前確率 $P(x_i)$（MPE において $X=x_i$ となる確率）は、ノードごとに異なるが i にはよらない定数 δ_X である。

$$P(x_i)=\delta_X \ (i=1,\cdots,s) \tag{7.4}$$

この制約条件を**非 ϕ 値の等確率の制約**と呼ぶ[1]。

この制約条件はＳＯＭの性質によりある程度近似的に満たされるが、より厳密に満たす必要がある場合は、簡単な神経回路を付加すれば実現可能である。詳しくは 7.3.4 節で述べる。

[1]最下端にある入力ノードもこの条件を満たすべきだが、今回のシミュレーションでは満たしていない。

4. すべてのノード X において、φ値の事前確率 $P(x_\phi)$ は 1 に近い値である。

$$P(x_\phi)\approx 1 \tag{7.5}$$

すなわち、 MPE においてほとんどの隠れノードの値がφ値になる。

この制約条件を **ノード活性のスパース性の制約**と呼ぶ。この制約条件は、ネットワーク内の全ノードの活性度を調整する特殊なノードをベイジアンネットに追加することで実現される。詳しくは 7.3.3 節で述べる。

なお、等確率の制約（式 (7.4)）と合わせると、下記の式が成り立つため、δ_X は $1/s$ よりも十分に小さい値となる。

$$P(x_\phi)=1-\sum_{i=1}^{s}P(x_i)=1-s\delta_X\approx 1 \tag{7.6}$$

5. すべてのノード X において、φ値 x_ϕ は、X のすべての子ノード Y_l の値と因果関係を持たない。

$$P(y_i^l|x_\phi)=P(y_i^l)\ (i=\phi,1,\cdots,s) \tag{7.7}$$

この制約条件を **ϕ 値の中立性の制約**と呼ぶ。

これを満たしていれば、値がφ値であるノード（不活性ノード）は子ノードに対してあたかも存在していないかのようにふるまう。（ただし、一般にφ値は親ノードとは因果関係を持つので注意が必要である。つまり、不活性ノードは親ノードに対しては、「あたかも存在していないかのよう」にはふるまわない。）

この制約条件もある程度自動的に満たされると考えている（要シミュレーション）が、厳密に満たすよう強制するのも容易である。詳しくは 7.3.5 節で述べる。

なお、等確率の制約（式 (7.4)）と合わせると、下記の式が成り立つ。

$$P(y_i^l|x_\phi)=P(y_i^l)=\delta_{Y_l}\ (i=1,\cdots,s) \tag{7.8}$$

7.3.2 ϕ 値を含むノードの学習則

4.2 節で述べた学習則は、φ値を含むノードに対しては以下のように修正される。

子ノード Y_l の勝者ユニットが y_j^l であった場合の v_j^l を計算する式 (4.1) は次のように拡張される。(なお、v_j^l は $j \in \{1, \cdots s\}$ に対してのみ定義され、v_ϕ^l は定義されない。)

$$v_j^l = \begin{cases} 0 & (Y_l \text{ の勝者がφ値の時}) \\ \frac{1}{Z_b} b(\alpha, d_{x_i}, d_{y_j^l}) & (Y_l \text{ の勝者が非φ値の時}) \end{cases} \tag{7.9}$$

近傍関数は、従来の近傍関数 n を拡張した n' を用いる。勝者ユニットが非φ値 x_i であった場合の n' は次のように定義される。

$$n'(\alpha, i) = \begin{cases} 0 & (i = \phi) \\ \alpha \frac{1}{Z_n} n(\alpha, d_{x_i}) & (i \neq \phi) \end{cases} \tag{7.10}$$

勝者ユニットがφ値 x_ϕ であった場合は近傍学習はせず、n' は次のように定義される[2]。

$$n'(\alpha, i) = \begin{cases} \alpha & (i = \phi) \\ 0 & (i \neq \phi) \end{cases} \tag{7.11}$$

w_{ij}^l $(j \neq \phi)$ に対する重みの更新式は、式 (4.2) を、次のように修正する。

$$w_{ij}^l \leftarrow w_{ij}^l + n'(\alpha, i)(v_j^l - w_{ij}^l) \tag{7.12}$$

$w_{i\phi}^l$ の値は、w_{ij}^l $(j \neq \phi)$ から計算する。

$$w_{i\phi}^l = 1 - \sum_{j=1}^{s} w_{ij}^l \tag{7.13}$$

7.3.3 認識ステップにおけるノード間競合

この節では、ノード活性のスパース性の制約(式 (7.5))を満たすようにするための機構について述べる。

図 7.3 のように、すべての隠れ変数ノード H_i の子ノードとして、スパース性の制約を表すノード S を追加する。S は、H_i の値の集合 $\boldsymbol{h}$ がスパース性の制約を満たすとき $S = 1$、そうでなければ $S = 0$ になる確率変数である。ノード S に関する条件付確率表は、下記の式で表せるものとする。

$$P(S = 1 | \{H_1, \cdots, H_n\} = \boldsymbol{h}) = \frac{1}{Z} e^{-\beta A(\boldsymbol{h})} \tag{7.14}$$

[2]実は、7.3.5 節で述べるように、重み $w_{\phi j}^l$ を明示的に学習する必要はない。その場合は、この学習則は不要である。

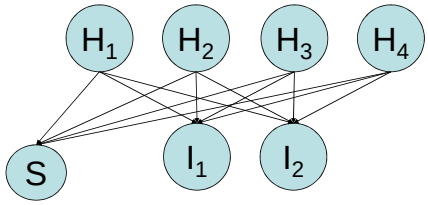


図 7.3: すべての隠れノードのスパース性の制約を表すノード S 。

ただし、Z は正規化定数、β はスパース性を制御するパラメタである。$A(\boldsymbol{h})$ は $\boldsymbol{h}$ がどの程度スパース性に関する制約を満たしているかを表す値であり、例えば以下のように定義する[3]。

$$A^{Soft}(\boldsymbol{h}; m) = ((\sum_{x \in \boldsymbol{h}} a(x)) - m)^2 \tag{7.15}$$

$$a(x_i) = \begin{cases} 1 & (i \neq \phi) \\ 0 & (i = \phi) \end{cases} \tag{7.16}$$

A^{Soft} は「非φ値を取る要素の数」が m 個から離れるほど大きな値を取る関数である。

あるいは、以下のようなものも考えられる。

$$A^{Hard}(\boldsymbol{h}; m) = \begin{cases} 1 & ((\sum_{x \in \boldsymbol{h}} a(x)) = m) \\ \infty & ((\sum_{x \in \boldsymbol{h}} a(x)) \neq m) \end{cases} \tag{7.17}$$

A^{Hard} は「非φ値を取る要素の数」が m 個ではないときに同時確率を 0 にするもので、用いるアルゴリズムによっては計算機上での MPE 探索の計算量を減らすことができる。

H_i と S の間の条件付確率表は固定であり、学習により変化しない。MPE を求める際には、S の観測値として $S = 1$ を与える。変数 S は noisy-OR モデルには従わず、従って条件付確率表 $P(S|H_1, \cdots, H_n)$ のサイズは s^n となるが、実際に MPE を求める際にそのような表を明示的に持つ必要はないので、問題ない。

[3]上の層ほどスパースにしたり、モダリティ間で優先度をつけたりと、$A(\boldsymbol{h})$ に対して様々な事前知識の作りこみが考えられるが、本稿では最も単純なものだけを示した。

3.2 節で述べたように、 BESOM モデルにおいて認識とは、MPE の選択、すなわち観測データとの同時確率を最大にする隠れ変数の値の組を求めることである。ノード S の追加により、同時確率の計算式 (3.2) は次のように変更される。(なお、ここでは 2 層 BESOM を想定している。)

$$P(S=1,\boldsymbol{h},\boldsymbol{i}) = \frac{1}{Z}e^{-\beta A(\boldsymbol{h})} \prod_{x\in\boldsymbol{h}\cup\boldsymbol{i}} P(x|parents(x)) \tag{7.18}$$

ノード間競合により、与えられた入力に対して活性化するノードの数が制限される。これは、パラメタ (結合の重み) の自由度を直接制限するものではないが、学習モデルの表現力を制限することになるため、代わりに汎化能力は向上すると期待できる。

ノード間競合には、他にも様々な効果がある。

この拡張により、観測データができるだけ少ない数の活性ノードを使って説明できるように、ベイジアンネットの学習が進むようになる。これは一種のスパース符号化であり、 BESOM の各層は、入力を効率的に圧縮した情報を上位層に送るようになる。詳しくは 7.6.2 節で述べる。

また、実質的に少数のノードだけを使って与えられた値の組の同時確率を計算することができるので、同時確率計算におけるダイナミックレンジや計算精度の問題が解決する。この性質は、計算機上でも生物にとっても有益である。

ノード間競合により、混合分布が表現できるようになる。詳しくは、7.4 節の実験結果で示す。

ノード間競合の機構の生物学的実現はおそらく容易である。大脳皮質と視床とを相互に接続する視床－皮質ループの解剖学的構造と電気生理学的振る舞いは、ノード間競合の機構とたいへんよく似ており、ノード間競合を実現する組織の有力な候補であると考えている。

7.3.4 非 ϕ 値の等確率の制約

非φ値の等確率の制約 (式 (7.4)) は、近似的には、競合学習の性質により自然に成り立つのではないかと筆者は予想している。

より厳密に成り立たせるためには、各ユニットの事前分布 $P(x_i)$ を学習し、その値が他のユニットよりも大きすぎる場合は勝者になりにくいようペナルティを与えるように認識アルゴリズムを修正すればよい。

7.3.5 ϕ 値の中立性の制約

φ値の中立性 (式 (7.7)) は、ある程度自然に成り立つと予想している。子ノードと因果関係の強いノードがノード間競合の勝者ノードになりやすいからである。

この制約を学習則で強制するのは簡単である。φ値ユニットが勝者になったときに $P(y_i^l|x_\phi)$ を明示的に学習せず、代わりにノード Y の事前確率 δ_Y を学習したものを用いればよい。

7.4 実験

7.4.1 混合分布から生成される入力の学習

本稿で提案したノード間競合の機構が、実際に混合分布の学習に有効であることを簡単な計算機シミュレーションで確認した。

学習に用いたのは２層ＢＥＳＯＭであり、隠れ層のノード数は２個、入力層のノード数は $7\times 7 = 49$ 個である。入力データは２次元であり、6.3.1 節と同じ方法で４９次元に変換したあと４９個の入力ノードに入力する。

図 7.4 はノード間競合に $A^{Hard}(\boldsymbol{h};m)$ $(m=1)$ を使用して２つのノードで混合分布を学習させた例である。灰色の領域は、入力の分布を表しており、この分布の中からランダムにサンプル点を生成し、 BESOM に入力した。２つのノードの学習結果は、それぞれ赤と青で示されている。２つのノードの各ユニットが、分離している２つの領域のそれぞれに張り付くことで、効率的に入力分布を表現していることが分かる。(可視化の方法についての詳細は 11.3 節参照。)

図 7.5 は、同じ混合分布の入力を、ノード間競合を行わないで学習した例である。２つのノードが同時に入力を学習するため、意味のない学習結果になっている。

7.4.2 混合分布に対する非線形ＩＣＡ

現在のところまだ正しく動作していないが、混合分布に対する非線形ＩＣＡの実験について説明する。

図 7.6 は、人工的に生成される４点から構成される、２種類の動物の顔の画像を４つのノードで学習した例である。一度に活性化するノードは２つである。

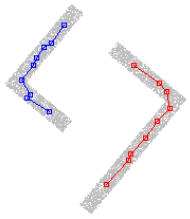


図 7.4: 混合分布を２つのノードで学習した例。ノード間競合には $A^{Hard}(\boldsymbol{h}; m)$ を使用。$m = 1$ 。

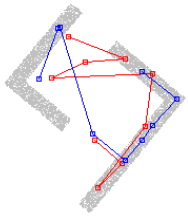


図 7.5: 混合分布を、ノード間競合を行わないで２つのノードで学習した例。

２種類の動物の顔が、それぞれ２つのノードを使って学習されている。ただし、２つのノード間のＩＣＡは安定して動いていない。この原因は把握している（6.6 節参照）が、解決は今後の課題である。

7.5 神経科学的妥当性

7.3.5 節で述べたように、φ値ユニットの条件付確率表 $P(y_j^l|x_\phi)$ を明示的に記憶する必要はない。また、φ値ユニットは近傍学習の対象とならない。したがって、大脳皮質上にφ値に対応するコラムが、通常のコラムと同じ形態で存在する必要はない。おそらくφ値ユニットは通常のコラムよりも少ないニューロンで実現可能である。一方 $P(y_\phi^l|x_i) = 1 - \sum_j P(y_j^l|x_i)$ の値は認識時に必要になるはずだが、その値を学習もしくは計算するニューロンの数も比較的少ないだろう。以上の予想どおり、φ値が比較的少ないニューロンで実現可能であるとすれば、φ値のように応答するニューロンがこれまで実験的に知られていない理由になる。

ノード間競合は、視覚野における exogenous attention （外因性注意）および視床－皮質ループと視床網様核の解剖学的構造と関係があると考えているが、詳細な検討は今後の課題である。

注意の正規化モデル [13] に、φ値ユニットの存在の１つの証拠がある。近似確率伝播アルゴリズム [1][2] はφ値を含む BESOM ネットにも適用可能だが、その場合ノードの各ユニットの出力 r_i は下記の値 Z を使って正規化されるはずである。

$$Z = r_\phi + \sum_{i=1}^{s} r_i \tag{7.19}$$

一方、注意の正規化モデルでは、空間方向の正規化を無視すると、下記のようになる。

$$Z = \sigma + \sum_{i=1}^{s} r_i \tag{7.20}$$

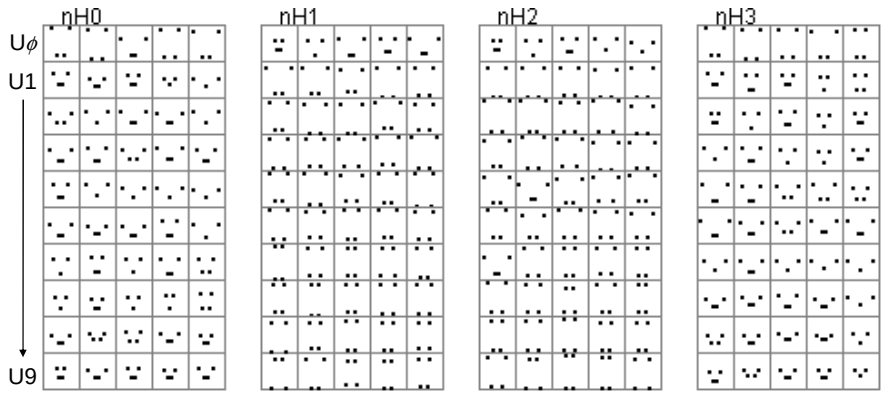
σ は、「入力のコントラストが小さく応答が小さいときには正規化の効果は表れない（正規化係数 Z は定数になる）」という実験事実を説明するために注意の正規化モデルに導入された定数である。ところで r_ϕ はφ値の中立性（式 (7.7)）により、入力刺激にはあまり依存しない値である。（全く依存しないわけではないが、入力刺激が弱い場合はおそらく影響が少ない。）従って、もしトップダウン信号が同一であれば r_ϕ はほぼ定数となり、注意の正規化モデルと整合性を持つことになりそうである（要シミュレーション）。ただし、 r_ϕ はトップダウン信号の影響は受けるため、この点で注意の正規化モデルとは異なっている。

7.6 今後の課題

7.6.1 スパース性の制御

スパース性の度合いを表すパラメタ β は、一種の正則化パラメタである。スパース性が下がれば多くのノードを使って入力をより正確に近似できるようになるが、それに見合うだけの入力サンプル数がない場合は過適合におちいり、汎化能力は落ちる。最適なスパース度がどのくらいなのかは、入力データの性質に依存するので、入力に関する事前知識なしでは理論的に決めることはできない。

工学的には、正則化パラメタの最適な値を決めるために、交差確認法がよく使われる。さまざまな値の正則化パラメタで汎化誤差を見積もり、もっともよい値を実際の応用に採用すればよい。



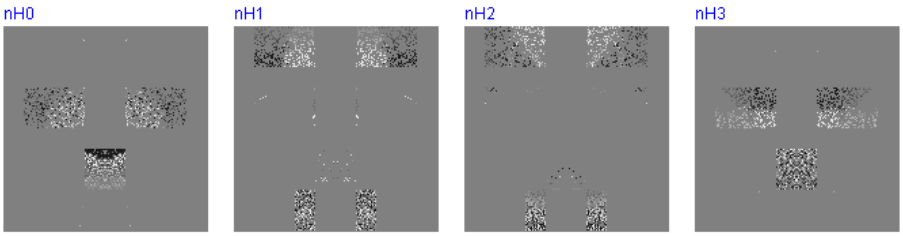


図 7.6: 混合分布の学習とＩＣＡを同時に行おうとした例。４点から構成される、２種類の動物の顔の画像を４つのノードで学習したときの、各ユニットの受容サンプルと受容野。一度に活性化するノードは２つ。

しかし、脳はオンラインで動作しなければならない。筆者は脳は交差確認法をオンラインで実行しているのではないかと考えているが、その検証は今後の課題である。

7.6.2 スパース符号化との関係

この章で提案したアルゴリズムは、非線形のスパース符号化を行っている。基底ベクトルの線形和ではなく、より複雑な演算によって符号から元の信号を復元できるため、より表現力が強い。逆にいえば、線形に限定されたスパース符号化よりも、本アルゴリズムは、よりスパースな符号ができる可能性がある。

このことは脳にさまざまな利点をもたらす。スパース符号化をすることで入力データが効率的に圧縮して記憶される。また、上位層に入力する特徴の数を情報を落とさずに減らせるので、上位層における認識の性能が上がる。また、情報を記憶するためのシナプス数、配線数、ニューロン発火率が減る。

BESOM による非線形スパース符号化の実験は、今後の課題である。

7.6.3 条件付確率表の近似モデル

本章で述べた条件付確率の近似式よりも好ましい近似式があり得る。

式 (7.2) の近似式は、ＭＰＥを求める際にアルゴリズムによっては正規化係数 Z を無視できるという利点がある。しかし活性値と不活性値を一様に扱っているため、noisy-OR モデルとみなすことができず、好ましくない。また、簡単な神経回路で確率伝播アルゴリズムやＭＰＥ計算アルゴリズムを実現する場合は、条件付確率の値が十分に小さいと仮定して、さらに近似する必要があるという欠点がある。

現在、よりよい近似式の候補として考えているのは下記の式である。

$$\begin{aligned} P(x_i|u_1,\cdots,u_m) &\approx \frac{1}{Z}\sum_k P(x_i|u_k) \\ P(x_\phi|u_1,\cdots,u_m) &\approx \prod_k P(x_\phi|u_k) \end{aligned} \tag{7.21}$$

ただし Z は正規化定数で、下記の式で表せる。（比較的複雑だが、さらに近似できるかもしれない。例えば分子はおそらくほぼ１である。）

$$\begin{aligned} 1/Z &= (1-P(x_\phi|u_1,\cdots,u_m))/\sum_i(\sum_k P(x_i|u_k)) \\ &= (1-\prod_k P(x_\phi|u_k))/\sum_k\sum_i P(x_i|u_k) \\ &= (1-\prod_k P(x_\phi|u_k))/\sum_k(1-P(x_\phi|u_k)) \end{aligned} \tag{7.22}$$

この式には下記のような利点がある。

1. 活性値 $x_1,\cdots,x_s$ をまとめて１つの値とみなすと、２値の noisy-OR モデルに厳密に一致する。（ただし $P(x_\phi|u_\phi)\approx 1$ という仮定が必要である。）
2. 条件付確率表をこれ以上近似しなくても、比較的簡単な神経回路で確率伝播アルゴリズムやＭＰＥ計算アルゴリズムが実現できる可能性が高い。

今後、この近似式に基づいてシミュレーションを行うことで、有効性を検証する必要がある。

7.6.4 スパース性と兄弟ノードの独立性

7.3.3 節で定義した A^{Soft} および A^{Hard} に従って学習した結果は、おそらく兄弟ノードのφ値どうしが独立にならない、すなわち $P(x_\phi,y_\phi)=P(x_\phi)P(y_\phi)$ とならないだろう。これでは「同一階層内にあるノード同士は独立でありエッジが不要」とするＢＥＳＯＭの前提に反してしまう。そこでφ値に対しても（近似的に）兄弟ノードが独立になるように、スパース性を制約する式を改良する必要がある。

階層内のノードどうしが独立ならば、階層内の重複しないノードの部分集合の値の組み合わせどうしも独立になる。例えば２つのノードの値の組み合わせ $A=a,B=b$ と別のノードの値の組み合わせ $X=x,Y=y$ は $P(a,b,x,y)=P(a)P(b)P(x)P(y)=P(a,b)P(x,y)$ であるから独立である。これは望ましい性質である。ＢＥＳＯＭでは、１つの信号源は１つノードではなく複数のノードのポピュレーションで表現される。ポピュレーションどうしが（ノードの重複が無視できれば近似的に）独立であるという性質により、膨大な数の独立な信号源の noisy-OR モデルが少ないノードで近似的に表現可能になる。

第8章　特徴間競合による部品別学習に向けて

この章では、部品別学習 (parts-based learning)[9] ができるように、7.3.2 節で述べた学習則をさらに拡張する。

ただし、アルゴリズムは不完全で、意図したとおりに動作していない。

8.1　部品別学習

自然画像は、複数の部品から構成されていることが多い。例えば顔画像は、目、鼻、口などの部品から構成される。それぞれの部品の形が異なる遺伝子から決定づけられているとすれば、個々の形状はほぼ独立な信号源から生成されることになる。

顔画像に限らず、一般に網膜座標において遠くに位置する視覚刺激は独立な信号源から生成される場合が多いだろう。この事前知識を学習則に作り込むことで汎化能力を上げることができるはずである。

また部品別学習は、信号源と特徴の間の因果関係の数を劇的に減らすので、記憶量と計算量の大幅な低減にもつながる。

実際に、脳の視覚野が部品別の情報表現をしていると思われる知見もある [9]。

8.2　重み減衰をする学習則

部品別学習の実現に向けて、1つの学習則を提案する。

w_{ij}^l $(j \neq \phi)$ に対する重みの更新式は、式 (7.12) を修正し、次のように重みに対する減衰項 D_i を追加する[1]。

$$w_{ij}^l \leftarrow max(w_{ij}^l + n'(\alpha, i)(v_j^l - w_{ij}^l - D_i), 0) \quad (8.1)$$

$max(a, b)$ は、a と b のうち大きいものを返す関数である。減衰項 D_i の値は以下のように定義される。

$$D_i = C((\sum_l \sum_{j=1}^{s} a(w_{ij}^l)) - \theta_a) \quad (8.2)$$

$$a(w) = \begin{cases} 1 & (w > \theta_w) \\ 0 & (w \leq \theta_w) \end{cases} \quad (8.3)$$

C は定数である。D_i はユニット x_i と下の階層のノード Y_l の非φ値ユニット y_j^l との間の重みのうち、θ_w より大きな値を持つものの数を与える[2] [3]。θ_a は定数 θ_w より大きな値を持つ重みの数の目標値である。

以上の学習則により、非φ値ユニットの重みの多くが小さな値をとるようになる。

この学習則により、ユニット x_i と因果関係を持つ特徴ノードの数は減っていく。この機構を **特徴間競合** と呼ぶことにする。

このアルゴリズムは**シナプス間競合**と呼ばれる神経科学的現象から着想を得ている。

なお、この機構により多くの条件付確率が0になるが、5 章で述べた機構により、認識に不都合が大きな生じることはない。

8.3　今後の課題

シミュレーションしてみたところ、部品別学習の効果は出たとしても非常に弱い。

また、条件付確率の値をゆがめるので、認識性能に影響が出る懸念がある。

現在別のアプローチのアルゴリズムを検討中である。

---

[1] この式では減衰項 D_i の値はユニット x_i ごとに決まる。計算機上で実現する場合は、勝者ユニットの D_i の値を近傍ユニットの学習の際にも使用しても問題ないかもしれない。その方が計算量は減る。現在の実装は、そのようになっている。

[2] ユニット x_i と下の階層のノード Y_l の非φ値ユニット y_j^l との間の重みの総和を与える、という方法も考えられる（$D_i = \sum_l \sum_{j=1}^{s} w_{ij}^l$）。しかしこの方法は、重みが薄く広がってしまい、特徴間競合にならず好ましくないかもしれない。

[3] 扱う情報に関する事前知識を用いて、ノードごとに異なる事前分布を作り込むことも可能だが、本稿では最も単純なものだけを示した。

第9章　ＭＰＥ計算の効率化

9.1　大脳皮質の計算量のオーダー

大脳皮質の面積は種によって大きく異なるが、大脳皮質の厚さ、ニューロンの密度、ニューロンあたりのシナプスの数、ニューロンの演算速度、物体を認識する速度は種によって大きくは違わない。(と筆者は思っている。)

それが正しいとすると、大脳皮質が物体を認識するアルゴリズムの計算量は、ニューロン数を n とすると、 $O(n)$ であることになる。

BESOM モデルの認識ステップに行われる MPE 計算は、一般にはノードの数を n とすると $O(2^n)$ の計算量を必要とする。しかし、特殊なベイジアンネットのもとでの近似解法でよければ、計算量を大きく減らすことができる。

以下の節では、実際に平均計算量 $O(n)$ で動作すると思われる MPE 計算アルゴリズムについて述べる。

9.2　$O(n^4)$ アルゴリズム

まず手始めとして、素朴な局所探索法によってＭＰＥの近似解を求めるアルゴリズムを考える。

図 9.1 はベイジアンネットのＭＰＥの近似解を山登り法を用いて求めるアルゴリズムである。

このアルゴリズムの平均の計算量を見積もってみる。(本当にこの見積もり通りの平均計算量になるかどうかは、実験によって確認する必要がある。)

ノードの数を n 、各ノードの取り得る値の数を定数 s とする。現在の隠れ変数の値の組を現在の状態とすると、次の状態の候補の数（$\boldsymbol{H}$ の要素の数）は ns 個である。

状態遷移の候補 $\boldsymbol{h}^*$ の入力 $\boldsymbol{i}$ との同時確率の計算式

1. すべての隠れノードの値を何らかの値で初期化する。その時の値の組を $\boldsymbol{h}$ とする。
2. $\boldsymbol{h}$ の中の高々１つの隠れノードの値を別の値に変更したものを次の状態の候補とする。候補の集合を $\boldsymbol{H}$ とすると、$\boldsymbol{H}$ の要素のうち、入力 $\boldsymbol{i}$ との同時確率が最大のものを $\boldsymbol{h}'$ とする。
$$\boldsymbol{h}' = \underset{\boldsymbol{h}^* \in \boldsymbol{H}}{\operatorname{argmax}} P(\boldsymbol{h}^*, \boldsymbol{i})$$
3. $P(\boldsymbol{h}', \boldsymbol{i}) > P(\boldsymbol{h}, \boldsymbol{i})$ 、すなわち同時確率が大きくなっていれば、 $\boldsymbol{h} := \boldsymbol{h}'$ として 2. に戻る。大きくなっていなければ終了。

図 9.1: 素朴な山登り法によってＭＰＥの近似解を求めるアルゴリズム。

は、式 (7.2) を用いると下記のように表される。

$$P(\boldsymbol{h}^*, \boldsymbol{i}) \propto \prod_{x \in \boldsymbol{h}^* \cup \boldsymbol{i}} (1 - \prod_k (1 - P(x|u_k))) \qquad (9.1)$$

ただし u_k は、値の組 $\boldsymbol{h}^*$ における、ノード $X = x$ の k 番目の親ノード U_k の値である。ベイジアンネットの各ノードの親ノードの数が平均 $O(n)$ であるとすれば、この同時確率計算の計算量は $O(n^2)$ である。

もし平均 $O(n)$ 回の状態変化で局所解にたどりつくとすれば、ＭＰＥの近似解を求めるための平均の計算量は $O(n^4)$ となる。

9.3　$O(n^3)$ アルゴリズム

前節で述べた素朴なアルゴリズムでは、１回の同時確率計算のコストは $O(n^2)$ とした。しかし、これを $O(n)$ に減らすことができる。

現在の状態の入力との同時確率を p, ノード X の値を x から x' に変更した後の同時確率を p' とすると、p' は次の式で表せることに着目する。

$$p' = \frac{P(x'|u_1, \cdots, u_m) \prod_l P(y_l|x_1, \cdots, x', \cdots, x_q)}{P(x|u_1, \cdots, u_m) \prod_l P(y_l|x_1, \cdots, x, \cdots, x_q)} p \qquad (9.2)$$

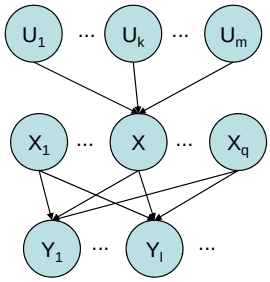


図 9.2: ノード X と親ノード U_k および子ノード Y_l 。

ただし、 y_l は、ノード $X = x$ の l 番目の子ノード Y_l の値、x_i $(i = 1, \cdots, q)$ はノード Y_l から見た i 番目の親ノード X_i の値である（図 9.2)。

まず、最初に隠れノードの値を初期化した後、各ノード X ごとに、下記の値 Λ_X を計算する。

$$\Lambda_X = \prod_k (1 - P(x|u_k)) \tag{9.3}$$

あるノードの親ノードの数は $O(n)$ であり、Λ_X を n 個すべてのノードに対して計算するから、この計算量は $O(n^2)$ である。

また、初期状態の同時確率 p を計算しておく。この計算量は $O(n^2)$ である。（ただし、ＭＰＥ計算で必要なのは同時確率の変化率 p'/p だけなので、これは必ずしも必要ではない。）

以上の初期化の計算量は $O(n^2)$ であるが、これは最初に一度実行されるだけなので、ＭＰＥ計算の計算量のオーダーを変えることはない。

次に、ノード X の値を x から x' に変更した場合の同時確率計算について説明する。

まず、下記式の値は定数時間で計算できる。

$$P(x|u_1, \cdots, u_m) = 1 - \Lambda_X \tag{9.4}$$

X の親ノードの数は $O(n)$ なので、下記の式の値は $O(n)$ で計算できる。

$$P(x'|u_1, \cdots, u_m) \propto 1 - \prod_k (1 - P(x'|u_k)) \tag{9.5}$$

また、下記の２つの値は $\Lambda_{Y_l} = \prod_i (1 - P(y_l|x_i))$ の値を使って定数時間で計算できる。

$$P(y_l|x_1, \cdots, x, \cdots, x_q) = 1 - \Lambda_{Y_l} \tag{9.6}$$

$$P(y_l|x_1, \cdots, x', \cdots, x_q) = 1 - \frac{1 - P(y_l|x')}{1 - P(y_l|x)} \Lambda_{Y_l} \tag{9.7}$$

X の子ノードの数も $O(n)$ なので、式 (9.2) の同時確率 p' の値は $O(n)$ で計算できることになる。

探索により値を変更するノード X が確定したら Λ_X の値も再計算する必要がある。

$$\Lambda_X = \prod_k (1 - P(x'|u_k)) \tag{9.8}$$

再計算の計算量は $O(n)$ である。再計算は１回の状態変化につき１回であり、状態変化１回に必要な計算量 $O(n)$ を増やすことはない。

以上の工夫により、ＭＰＥ計算の平均計算量は $O(n^4)$ から $O(n^3)$ に下がる。

なお、 Λ_X を記憶するためのメモリ量は $O(n)$ である。

9.4 $O(n)$ アルゴリズム

ノード間競合の機構を導入し、活性ノードの平均数が全ノード数 n によらず、一定値 m だとする。すると、平均 $O(m)$ 回の状態変化で局所解に到達すると予想されるので、ＭＰＥ計算の平均計算量は $O(n^2)$ にまで落ちることになる。

さらに、各ノードの親ノードと子ノードの数の平均値が全ノード数 n によらず、一定値だとする。すると、前節で述べた同時確率計算アルゴリズムの計算量は $O(1)$ になるので、ＭＰＥ計算の計算量は $O(n)$ になる。

なお、 Λ_X を記憶するためのメモリ量は $O(n)$ のまま変わらない。

9.5 局所解を避ける方法

局所探索法のうち、局所解を避ける工夫があるものには、多スタート局所探索、タブーサーチ、焼きなまし法、粒子群最適化などがある。

当然ながら、ほぼ確実に厳密解に到達できるようにするには指数関数的な計算時間が必要になる。

実用上はそこまでする必要はなく、いかに計算量のオーダーを上げないで、無意味な局所解を避けるかが課題になる。具体的に、どの方法をどのようなパラメ

タで実行すると最適なのかは、扱うデータの性質にも依存する。今後実験により試行錯誤で決める必要があるだろう。図 9.1 の素朴な山登り法でも実用上あまり問題がないかもしれない。

なお筆者は、近似確率伝播アルゴリズムと局所探索法をうまく組み合わせることで、無意味な局所解をある程度避けることができるのではないかと考えている。このアイデアの妥当性の検証も今後の課題である。

9.6 $O(1)$ アルゴリズムの可能性

これまでに述べたアルゴリズムでは、毎回の状態変化ごとに、値を変更するノードを n 個のノードから探すため、状態変化に $O(n)$ の計算量がかかる。

これは、データベースのリニアサーチに似ている。大脳皮質は n 個の並列演算装置でこれを実行するので、同じことをほぼ $O(1)$ の時間で実行できる。

しかし、汎用計算機ならば並列計算しなくても、リニアサーチよりも優れたアルゴリズムを使って、計算量そのものを $O(1)$ に減らせる可能性がある。

もしそれができれば、実用的価値は非常に大きい。

それが可能かどうかは、実データを学習した結果得られる条件付確率表がどのような性質を持つかを観察しながら検討すべきだろう。

9.7 2層 BESOM の計算量

2層 BESOM の場合、すべての隠れノードは親ノードを持たないため、計算量はかなり少なくなる。

隠れノード数 n 、入力ノード数が m とすれば、同時確率の計算量は、素朴な方法でも計算量は $O(mn)$ となる。ＭＰＥ計算の平均計算量は $O(mn^3)$ である。

同時計算で 9.3 節の工夫をした場合、平均計算量は $O(mn^2)$ に減る。

9.4 節と同様にノード間競合を導入すれば、平均計算量は $O(mn)$ となる。さらに入力ノードの親ノードの数が n によらず定数であれば、平均計算量は $O(m)$ となる。

9.8 今後の課題

スパース性制約 $A(\boldsymbol{h})$ や側抑制 $S(\boldsymbol{h})$ を含めた高速アルゴリズムについてはまだ検討していない。

9.3 節の $O(n^4)$ アルゴリズムはシミュレーションにより動作を確認したが、9.4 節の $O(n^3)$ アルゴリズムは未確認である。

第10章　近似ＭＰＥ計算アルゴリズムと神経回路

この章では、まだ不完全ではあるが、ＭＰＥを近似計算できる可能性のある簡単な神経回路モデルのアイデアについて述べる。

10.1　近似ＭＰＥ計算アルゴリズム

9.3 節で述べた $O(n^3)$ アルゴリズムは、神経回路での実現に適した形に変形することができる。あるノードの値をどの値に更新するかは、そのノードと接続を持つノードとの間の局所的な情報だけで決定することができるからである。しかし、そのアルゴリズムのままでは神経回路はかなり複雑になってしまう。以下に、比較的簡単な神経回路で実現できるように近似したＭＰＥ計算アルゴリズムを述べる。

まず各ノード X はメッセージ $b(x_i)$ を親ノードと子ノードに送るものとする。

$$b(x_i) = \begin{cases} 1 & (\text{現在の状態が } X = x_i) \\ 0 & (\text{それ以外}) \end{cases} \tag{10.1}$$

次に各ノードは、自分の周辺のノードからメッセージを受け取り、$s(x_i)$ という値を計算する。この値は、ノード X の状態が x_i に変化したときに、ネットワーク全体の同時確率がどのくらい変化するかを表す値であるとする。その具体的定義はすぐ後に示す。

$s(x_i)$ が計算されれば、ノード X の次の状態 x' は次の式で決まる。

$$x' = \operatorname*{argmax}_x s(x) \tag{10.2}$$

次の状態 x' が決まれば、メッセージ $b(x_i)$ を計算しなおし、周辺ノードに送る。

以上の計算をすべてのノードが繰り返すことで、ネットワーク全体の同時確率は単調に増加し、局所解に収束する。

さて、 $s(x)$ の具体的定義を示す前に、同時確率の式 (9.2) を思いだそう。

$$p' = \frac{P(x'|u_1,\cdots,u_m)\prod_l P(y_l|x_1,\cdots,x',\cdots,x_q)}{P(x|u_1,\cdots,u_m)\prod_l P(y_l|x_1,\cdots,x,\cdots,x_q)}p \tag{10.3}$$

今求めたいのは他の変数の値との同時確率を最大にする X の値であり、新しい値の候補 x' に依存しない分母の値や p の値は不要である。従って、同時確率の代わりに下記の式 $s_1(x')$ を最大化すればよい。

$$s_1(x') = P(x'|u_1,\cdots,u_m)\prod_l P(y_l|x_1,\cdots,x',\cdots,x_q) \tag{10.4}$$

ここで、[1] と同様に下記の近似が成り立つと仮定する。（φ値に対しては特別な扱いが必要だが、その検討は今後の課題とする。）

$$P(x'|u_1,\cdots,u_m) \approx \sum_k P(x'|u_k) \tag{10.5}$$

すると下記の近似が成り立つ。

$$\begin{aligned} P(x'|u_1,\cdots,u_m) &= 1 - \prod_k (1 - P(x'|u_k)) \\ &\approx \sum_k P(x'|u_k) \end{aligned} \tag{10.6}$$

さらに [1] と同様に十分に多くの親ノードが同じ特徴を生成すると仮定すると、下記の近似が成り立つ。

$$\begin{aligned} & P(y_l|x_1,\cdots,x',\cdots,x_q) \\ \approx\ & P(y_l|x_1)+\cdots+P(y_l|x')+\cdots+P(y_l|x_q) \\ \approx\ & (\sum_i P(y_l|x_i)) + P(y_l|x') \end{aligned} \tag{10.7}$$

従って、$s_1(x')$ は下記に定義される $s_2(x')$ で近似できる。

$$\begin{aligned} s_1(x') &\approx s_2(x') \\ &= \sum_k P(x'|u_k)\prod_l ((\sum_i P(y_l|x_i)) + P(y_l|x')) \end{aligned} \tag{10.8}$$

$s_2(x)$ を、周辺ノードからのメッセージを用いるように書き換えることで $s(x)$ が得られる。

$$s(x) = \lambda(x)\pi(x) \tag{10.9}$$

$$\lambda(x) = \prod_l \lambda_{Y_l}(x) \tag{10.10}$$

$$\lambda_{Y_l}(x) = Z_{Y_l} + \sum_{y_l} b(y_l)P(y_l|x) \tag{10.11}$$

$$\pi(x) = \sum_k \kappa_{U_k}(x) \tag{10.12}$$

$$\kappa_{U_k}(x) = \sum_{u_k} P(x|u_k)b(u_k) \tag{10.13}$$

$$\begin{aligned} Z_X &= \sum_k \sum_x \sum_{u_k} P(x|u_k)b(x)b(u_k) \\ &= \sum_x b(x) \sum_k \sum_{u_k} P(x|u_k)b(u_k) \\ &= \sum_x b(x)\pi(x) \end{aligned} \tag{10.14}$$

ただし、 Z_X, Z_{Y_l} はノード X および Y_l から親ノードに向かって送られる、もう一種類のメッセージである。

10.2 近似確率伝播アルゴリズムとの比較

前節のアルゴリズムを実行する神経回路は、[1] で述べた近似確率伝播アルゴリズムとかなり似たものになっている。

ボトムアップ信号は子ノードごとに内積計算されたあと掛け算され、トップダウン信号は親ノードごとに内積計算されたあと足し算され、その２つが掛けあわされる。

近似確率伝播アルゴリズムでは最終的な出力は正規化されることから、多くの電気生理実験を説明する「正規化モデル」[1]とも整合性がある。一方、この近似ＭＰＥ選択アルゴリズムでは argmax が選択されるため、正規化モデルなどの電気生理実験の結果を説明できない。

トップダウンのメッセージ Z_X の計算式も２つのアルゴリズムで異なる。

近似確率伝播アルゴリズムと近似ＭＰＥ選択アルゴリズムの大きな違いは、ボトムアップの情報の流れである。近似確率伝播アルゴリズムでは、コラム内の情報処理の最終的な出力 $BEL(X = x_i)$ ではなく、情報処理の途中の値 $\lambda(X = x_i)$ が親ノードに送られる。そして、実際の大脳皮質の解剖学的構造に近いのは近似確率伝播アルゴリズムの方である。大脳皮質の回路のこの特徴は、ボトムアップの１パスの情報の流れだけで、高速にある程度の精度の認識ができるという利点があるのではないかと筆者は想像している。近似ＭＰＥ選択アルゴリズムにもこの機能を持たせるよう回路を修正することは可能だろう。

大脳皮質の学習の目的を説明するためには、大脳皮質は認識ステップでＭＰＥ計算を行っていると考えざるを得ない。認識がＭＰＥ計算ならば、認識と学習の繰り返しは一種のＥＭアルゴリズムをオンラインで行っていると解釈可能である。一方、近似確率伝播アルゴリズムだけでは、その意味に理論的な解釈をつけることはできない。

以上のようにこれら２つのモデルは、一長一短があり、将来は統合されなければならない。

10.3 ホップフィールドネットワーク・ボルツマンマシンとの比較

前節で述べた神経回路の動作は、ホップフィールドネットワークの動作と定性的にとてもよく似ている。どちらも、局所的な値の更新のみによって、全体の目的関数を最適化している。

また、ボルツマンマシンの認識動作とも似ている。ボルツマンマシンにおける認識動作は、全ノードの値の同時確率を最大化するＭＰＥをＭＣＭＣで求めるものである[2]。

おそらく、過去にこれらの学習モデルに関する研究で得られた多くの知見が、 BESOM にも応用可能で

[1]参考：「総説論文「注意の正規化モデル」の紹介」 http://staff.aist.go.jp/y-ichisugi/besom/20090408norm.pdf

[2]参考：「Boltzmann マシン - 情報論的学習理論と機械学習の「朱鷺の杜 Wiki」」 http://ibisforest.org/index.php?Boltzmann マシン

あろう。

なお、 9.3 節で述べたアルゴリズムはこれら先行研究と比較すると、一般性の点で大きく異なっている。ホップフィールドネットワークとボルツマンマシンは、いずれも学習モデルとしては制限が強い。一方 9.3 節で述べたアルゴリズムは noisy-OR モデルのみを仮定した、より一般性の高いベイジアンネットを対象としている。

10.4 今後の課題

工学的な応用が目的であれば、9.3 節、9.4 節の考え方を基本としたアルゴリズムで十分かもしれない。しかし、大脳皮質にある実際の神経回路に対する理解を深めるためには、本節で述べた神経回路モデルを発展させる必要がある。また、実際の脳が採用している詳細なアルゴリズムが分かれば、工学的アルゴリズムの性能向上に役立つかもしれないという期待もある。

近似確率伝播アルゴリズム同様、近似ＭＰＥ計算アルゴリズムは、大胆な近似を行っているため、何らかの補正をしないと実用的な動作をしないかもしれない。補正の１つの可能性は、総和が１を超えないように、シグモイド関数 $\phi(x)$ を使うことである。例えば、$\lambda_{Y_l}(x)$ と $\pi(x)$ の計算式は次のように補正するとよいかもしれない。

$$\lambda_{Y_l}(x) = \phi(Z_{Y_l} + \sum_{y_l} b(y_l)P(y_l|x)) \qquad (10.15)$$

$$\pi(x) = \phi(\sum_{k} \kappa_{U_k}(x)) \qquad (10.16)$$

ただしこの補正は、 7.6.3 節で述べた近似式を用いる場合は不要である。

第11章　可視化の詳細

11.1　可視化の意義

アルゴリズムのデバッグおよびパラメタの調整の際には、学習の状況を様々な角度から可視化することが不可欠である。

11.2　条件付確率表

4 章の図で使われている条件付確率表の可視化は、以下のように行う。

通常のベイジアンネットでは親ノード U_k の値の組み合わせに対するノード X の値の条件付確率 $P(X|U_1,\cdots,U_m)$ をノード X ごとに保持するが、本稿で用いているシミュレーターは、ノード X と子ノード Y_l との間の条件付確率 $P(Y_1|X),\cdots,P(Y_n|X)$ をノード X ごとに保持するデータ構造を採用している。可視化も、このデータ構造を反映したものになっている。

ノード X が保持する条件付確率表は、ノード X および Y_l $(l=1,\cdots,n)$ のユニット数を s とすると、$s\times s\times n$ 次元の配列になる。

条件付確率表 $P(Y_l|X)$ を $W_l=(w_{ij}^l)$, $w_{ij}^l=P(y_j^l|x_i)$ という $s\times s$ の行列で表し、$W_1,\cdots,W_n$ というふうに n 個行列を横に並べ、行列の要素を白黒の濃淡で表現したものを画面に表示することで、可視化している。

この方法で可視化すると、BESOM の条件付確率表を関連データベースとみなす場合に理解しやすい。縦方向にデータベースの各行が並んでおり、横方向には各行の属性値（特徴量）が並んでいるというふうに見ることができる。

なお、φ値がある場合は0番目のユニットがφ値を表すものとする。

条件付確率表の上と右に出ている黒い点は、入力（特徴量）と出力（認識結果）を表している。

11.3　１点入力時のユニットおよびＭＰＥの受容野重心

7.4 節や 6.3.1 節の図で使われている受容野重心の可視化は、以下のように行う。

まず、シミュレーション中、過去 T 回分の入力データ I_t と、その時のＭＰＥ $\boldsymbol{m}_t$ $(t=1,\cdots,T)$ をリングバッファに保持する。

背景となる確率分布は I_t を灰色の点でプロットしている。ユニットの受容野重心は、リングバッファに保持されている入力データからそのユニットが勝者になった時の入力点の集合のみを選び出し、その重心を計算することで得られる。ＭＰＥの受容野も同様である。

なお、この可視化方法では、学習が収束していない時、画面に表示されている受容野と真の受容野の間にタイムラグがある点に注意する必要がある。

本稿のシミュレーションでは $T=10000$ で可視化している。

11.4　多点入力時のユニット受容野

6.3.2 節の図で使われているユニットの受容野の可視化は、以下のように行う。まず灰色の背景を描画した後、リングバッファに保持されているデータの中から、そのユニットが勝者になった時の入力点の集合を選びだし、そのユニットに対応する色でプロットする。

本稿のシミュレーションでは $T=1000$ で可視化している。

11.5　今後の課題

シミュレーションするタスクごとにこまめに最適な可視化ルーチンを書くことが、シミュレーションの状況を正しく把握しパラメタ調整していく上で重要である。

どのようなタスクにも有効な万能な可視化の方法はあり得ない。むしろ、タスクごとに、できるだけ少ない手間で新しい可視化ルーチンを書けるように支援する、ライブラリの整備が必要である。

さらに、可視化したものをクリックして詳細な情報を表示できるようにしたり、大きさの調整やウインドウの移動、過去の可視化の状態の記憶など、様々な研究支援機構が考えられる。

第12章　研究支援ツール BESOM-lab

12.1　背景

筆者は、機械学習アルゴリズムの設計とシミュレーションを支援する研究支援ツールを、 Java 言語を用いて自作している。このツールにより、以下のような、機械学習アルゴリズム開発に独特な、様々な困難を克服しようとしている。

1. 理論だけからは決めにくいメタパラメタの値があり、正しく動かすためには試行錯誤が必要となる。(メタパラメタの数が多いと、網羅的に最適な値を探索するのも難しい。)
2. 学習で獲得されたパラメタは、うまく可視化しないとその意味を解釈できない。
3. ある程度複雑なアルゴリズムだと、学習が失敗した場合、アルゴリズムに問題があるのか実装のバグなのかが判断しにくい。
4. 浮動小数点を使うので、 NaN(not a number) 、アンダーフロー、オーバーフローなどに関わる見つけにくいバグが発生しやすい。
5. BESOM の場合、ＳOM, ＩＣＡ、ＭＰＥ計算など複数の部分アルゴリズムの集合体であり、それぞれのアルゴリズムの間の干渉も問題になる。

12.2　BESOM-lab の概要

研究支援ツール BESOM-lab (図 12.1) はオブジェクト指向フレームワークであり、前節で述べた困難さを少しでも軽減するために、以下の工夫を行っている。

1. テンプレートクラスの継承により、比較的少ないコードの記述で新たなアルゴリズムの実験に取り組めるようになっている。新たなタスクの追加も容易である。
2. シミュレーション中に、学習率などいくつかのメタパラメタの値をスライダで調整し、その影響をリアルタイムで確認することが可能である。新しくスライダを追加する場合も３行程度（将来的には１行程度）のコードの追加で実現できる。
3. 部分アルゴリズムのいくつかのバージョンが実装済みであり、シミュレーション中にセレクタを使って変更できる。例えば、 SOM の近傍学習のアルゴリズムは、4.2 節で述べたものの中から選択できるようになっている。他にも、 MPE 計算や ICA についても、新たなアルゴリズムの追加・選択が容易になっている。
4. シミュレータ本体と可視化のスレッドを分けて非同期に実行することで、シミュレーションの負荷が変わっても滑らかに可視化されるように工夫している。
5. NaN のチェックなど防御的コードが随所に埋め込まれている。

この研究支援ツールは、筆者以外の研究者も利用することを想定して開発を行っている。BESOM のアルゴリズムは、今後も大勢の研究者によって改良・拡張されるべきである。全体のアルゴリズムが比較的複雑になってきているため、その全体を個々の研究者が０から実装しなおすことは効率的とは言えない。しかしこの研究支援ツールを使えば、改良したい部分を実装するだけで済む。論文を書く場合も、使用するアルゴリズムとパラメタ値のすべてを書く必要がなく、ページ数に制約がある場合はデフォルト値から変更した部分だけを書けばよい。将来的には、性能評価のベンチマークも提供することで、様々なバージョンの部分アルゴリズムの性能を比較する共通の土台となるだろう。

12.3　今後の課題

今後はユーザからのフィードバックを得ることで、より使いやすいツールに改良していきたい。

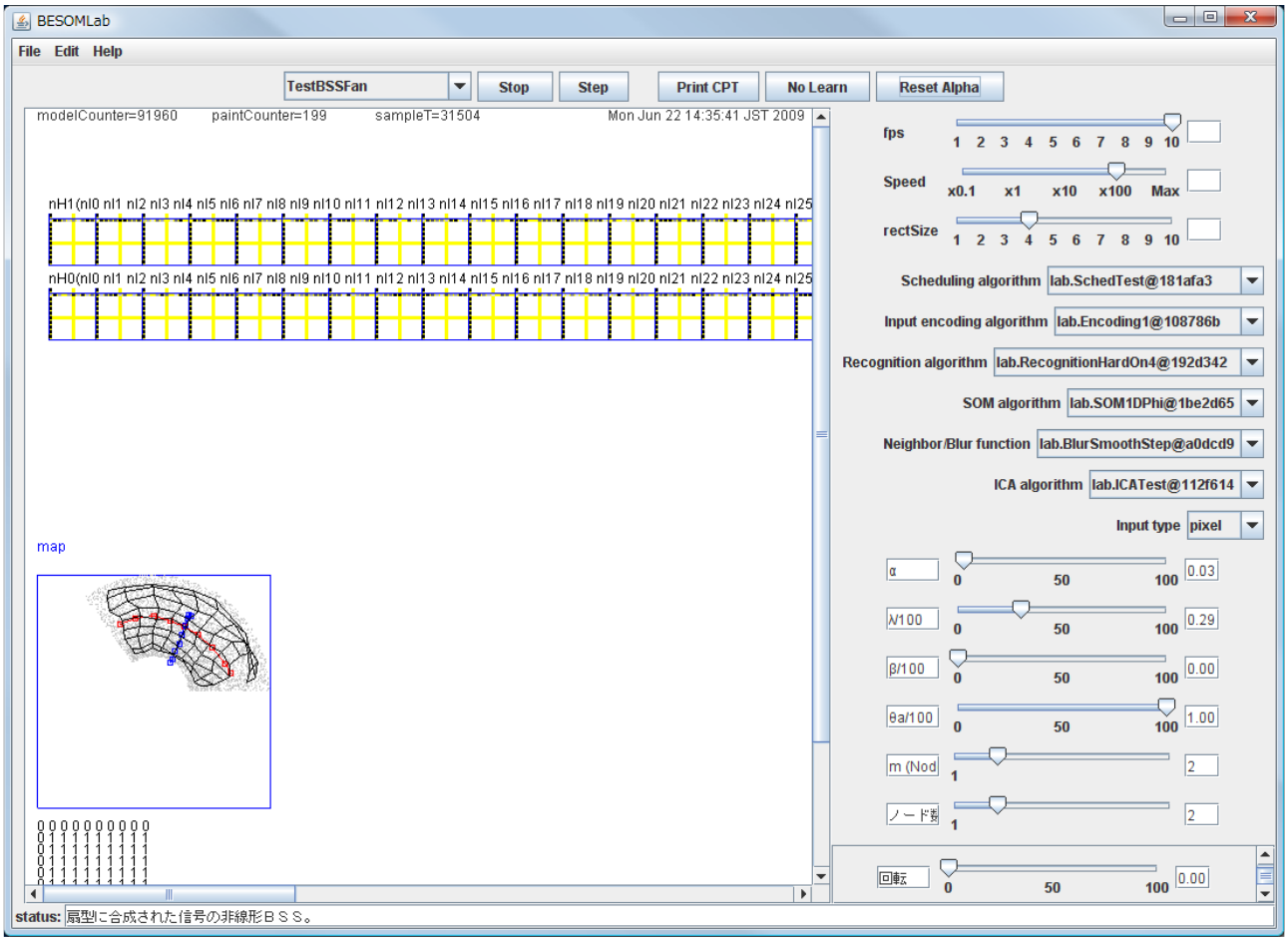


図 12.1: BESOM-lab のスクリーンショット。

第13章　その他の今後の課題

BESOM モデルの機械学習アルゴリズムとしての基本技術の確立、神経回路モデルとしての神経科学的妥当性の確立に向けて、今後明らかにしなければならない点はまだ多く残っている。

本章では、これまでの章で触れなかった今後の課題について述べる。

13.1　多層化

これまでの章では2層 BESOM を扱ってきたが、3層以上に拡張することができる。ただし、学習則に少し工夫が必要で、下の層から順に学習が収束するように工夫しなければならない。

工学的に用いられている多くの階層的な機械学習アルゴリズムでは、下の層から学習をする場合が多い。これは、下の層の学習が収束していない状況での入力を上の層に送って学習しても無意味だからである。

BESOM のように、トップダウンの信号が下位の層の学習に影響を与える場合はとくに深刻で、学習の初期のおいて、無意味な情報がトップダウンに送られると、下位の層の意味のある学習を阻害していまう。

BESOM に下の層から学習する機構を追加することは技術的には難しくないが、その理論的根拠は自明ではない。理論的根拠が不明であれば、BESOM モデルの機械学習理論的合理性が損なわれるだけでなく、学習のメタパラメタの定量的な調整の指針も立たない、という問題がある。

筆者は、「近傍半径は出力のノイズの大きさに対する事前知識を反映している」という原理が、下の層からの学習する機構に対する理論的根拠を与えるのではないか考えている。我々は、学習の初期においては、上位層の条件付確率表はほとんど意味を持たないノイズであるという事前知識を持っている。従って、学習初期の上位層の適切な近傍半径は無限大である。すると上位層の学習結果はすぐに一様分布になるので、トップダウンの信号は下位層の認識結果に影響を与えなくなる。このような考えに基づいて学習アルゴリズムを設計すれば、ベイズ理論と矛盾せずに下の層から学習が行われ、全体として常に意味のある学習結果が得られるのではないかと考えている。

13.2　層間競合

解剖学的事実との対応を考えると、ある層のノードは、自分よりも一段下の層からだけでなく、下位のすべての層のノードから入力を受け取るべきである。

したがって、特徴間競合は、層内だけでなく異なる層との間でも起きる必要があるだろう。

13.3　構造学習

本稿では具体的なアルゴリズムは説明していないが、初期状態でネットワーク全体のエッジの数が $O(n^2)$ あり、学習が進むにつれ条件付独立なノード間のエッジが切れて $O(n)$ になると考えている。

しかし生物学的には、個体発生の初期状態でシナプスの数が $O(n^2)$ ほどに増えることはないかもしれない。(赤ん坊時にのシナプスの数が最大になりその後徐々に減っていくが、最大の時でも $O(n^2)$ ほどではないだろう。)

従って実際の脳では、$O(n^2)$ のエッジから独立なものを減らすのではなく、少ないエッジ数の初期状態からはじめて、エッジの追加と不要なエッジの削除を常に行っている、という可能性も考えられる。工学的にも、そのようなアルゴリズムを用いた方が学習初期の計算量が少なく、有用である。

また、環境の変化への適応の必要性という意味でも、エッジの追加の機構は不可欠である。

そのようなアルゴリズムの検討は今後の課題である。エッジの追加は、ランダムに行うのも1つの方法だが、活性状態にある2つのノードの間を結ぶという方法も考えられる。無意味な局所解に陥らないような構造学習アルゴリズムがどのようなものかは、やはり扱うデータの性質に依存するので、実験により決める必要がある。

13.4 サイクルのある BESOM ネット

ベイジアンネットにサイクルがないことを仮定するアルゴリズムもあるが（例えば loopy でない確率伝播アルゴリズム）、9 章、10 章で述べた認識アルゴリズムはいずれもサイクルがあっても動作するはずである。4 章で述べた学習則もまたサイクルのある BESOM ネットで動作するものである。したがって BESOM は「ネットワークにサイクルがない」という制約を一切必要としない。[2] の 6.5.2 節で述べたように、サイクルのある BESOM ネットは再帰的な文法獲得の実現に必要である可能性がある。また、前頭前野による再帰的な行動プログラムの記憶や、側頭葉による相互依存的な知識構造の記憶にも必要かもしれない。サイクルのある BESOM ネットの動作確認およびその性質の詳細な調査、さらにそれを用いた再帰的な認知機能の再現は、今後の課題である。

13.5 強化学習との統合

本稿が主張する「認識とは MPE 計算である」という仮説により、「ポピュレーションによる強化学習の実現」の問題を解決できると考えている。

[2] の 7.6.3 節で述べたように、脳はポピュレーションとして状態認識し、ポピュレーションのまま行動選択し、ポピュレーションのまま運動を命令し、ポピュレーションのまま行動価値を更新するはずであると筆者は考えている。

このようなポピュレーションによる強化学習を、状態認識と行動選択を１つの MPE 計算で行うことで実現するアルゴリズムを筆者は現在設計中である。状態行動対を学習する隠れノードが１ノードの場合は簡単だが、複数の隠れノード、時系列学習、多層化などの拡張を行うにつれ、アルゴリズムは徐々に複雑になりそうである。

なお、強化学習との統合の第一歩として、BESOM の１個のノードに強化学習の機能を持たせる試みが Hosoya によってなされている [20]。

13.6 モデルからの予言の検証

本稿ではアルゴリズムが一部不確定なのでこれまで全く書いてこなかったが、各部分アルゴリズムを実現する神経回路から、自明でない様々な解剖学的特徴、電気生理学的現象を予言することができる。

例えばノード間競合の機構はφ値ニューロンや $A(\boldsymbol{h})$ の値を計算する機構の存在を予言し、非線形ＩＣＡの機構はアンチヘブ則で学習する側抑制シナプスの存在を予言する。

今後アルゴリズムを確定的なものにしつつ、実験で検証しやすい自明でない予言にどのようなものがあるかを、整理していく必要がある。

第14章　まとめと今後

大脳皮質の計算論的モデルである BESOM モデルの詳細アルゴリズムを提案し、小規模な計算機シミュレーションの結果を示した。

今後の中規模・大規模シミュレーションに向けては、特徴間競合アルゴリズムの再設計、 $O(n)$ アルゴリズムの実装、非線形ＩＣＡとノード間競合の機構の干渉問題の解決が最優先課題である。

本稿で提案したアルゴリズムの改良と詳細な動作確認が終われば、知能の高いロボットに向けた最大の難関が突破できたことになると思う。時系列学習との統合や、大脳基底核、海馬、小脳、扁桃体等の機構との統合などやるべき問題は残っているが、大脳皮質以外のこれらの組織の計算論的モデルの研究はすでにかなり進んでいる。多くの優秀な研究者がこの問題に取り組みさえすれば、脳全体の機能を工学的に再現できる日は遠くないだろう。

各章で書いたように、細かな未解決の問題がたくさんある。機械学習理論の基礎をある程度理解した工学的センスのある大勢の研究者が、これらの問題に取り組むようになることを期待する。

参考文献

[1] Yuuji ICHISUGI, The cerebral cortex model that self-organizes conditional probability tables and executes belief propagation, In Proc. of International Joint Conference on Neural Networks (IJCNN2007), pp.1065–1070, Aug 2007.

[2] 一杉裕志、「脳の情報処理原理の解明状況」産業技術総合研究所テクニカルレポート AIST07-J00012, Mar 2008.
`http://staff.aist.go.jp/y-ichisugi/besom/AIST07-J00012.pdf`

[3] T. Kohonen, Self-Organizing Maps. Springer-Verlag, 1995.

[4] T. コホネン, 自己組織化マップ（改訂版）, シュプリンガー・フェアラーク東京, 2005. （[3] の邦訳。）

[5] K. Fukushima, Neural network model for selective attention in visual-pattern recognition and associative recall, APPLIED OPTICS 26 (23): 4985-4992 Dec 1 1987.

[6] J. Pearl , Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.

[7] George, D. Hawkins, J., A hierarchical Bayesian model of invariant pattern recognition in the visual cortex, In proc. of IJCNN 2005, vol. 3, pp.1812-1817, 2005.

[8] Olshausen BA, Field DJ, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, NATURE 381 (6583): 607-609 JUN 13 1996.

[9] Daniel D. Lee and H. Sebastian Seung, Learning the parts of objects by non-negative matrix factorization Nature 401, 788-791 (21 October 1999).

[10] Wang G, Tanaka K and Tanifuji M, Optical imaging of functional organization in the monkey inferotemporal cortex, SCIENCE 272 (5268): 1665-1668 JUN 14 1996.

[11] 田尻 隆, 倉田 耕治: 二つの 1 次元 SOM の結合による独立成分分析と主成分分析, 電子情報通信学会技術研究報告 ニューロコンピューティング研究会, Vol.104, No.139(20040617) pp. 61-66, 2004.

[12] Naoki Oshiro, Koji Kurata, Tetsuhiko Yamamoto A self-organizing model of place cells with grid-structured receptive fields, Artificial Life and Robotics, Vol.11, No.1, pp.48–51, 2007.

[13] Reynolds JH, Heeger DJ: The normalization model of attention, Neuron. 2009 Jan 29;61(2):168-85.

[14] Schultz W, Dayan P, Montague PR, A neural substrate of prediction and reward, Science 275(5306):1593-1599, Mar 1997.

[15] K. Doya, Complementary roles of basal ganglia and cerebellum in learning and motor control, Current Opinion in Neurobiology 10 (6): 732-739 Dec 2000.

[16] 川人光男: 脳の計算理論, 産業図書, 1996.

[17] Rolls, ET: A theory of hippocampal function in memory, HIPPOCAMPUS, Volume: 6 Issue: 6 Pages: 601-620, 1996.

[18] T. Omori et al., Emergence of symbolic behavior from brain like memory with dynamic attention, Neural Networks 12 (7-8): 1157-1172 Oct-Nov 1999.

[19] Elman, J. L., Distributed representations, simple recurrent networks, and grammatical structure. Machine Learning, 7:195–224, 1991.

[20] Haruo Hosoya: A motor learning neural model based on Bayesian network and reinforcement learning, In Proceedings of International Joint Conference on Neural Networks, 2009.

[21] C. M. ビショップ: パターン認識と機械学習 上 - ベイズ理論による統計的予測, シュプリンガー・ジャパン株式会社, 2007.

[22] C. M. ビショップ: パターン認識と機械学習 下 - ベイズ理論による統計的予測, シュプリンガー・ジャパン株式会社, 2008.

大脳皮質のアルゴリズム BESOM Ver.1.0
産業技術総合研究所テクニカルレポート AIST09-J00006
2009 年 9 月 30 日
独立行政法人 産業技術総合研究所
〒305-8568　茨城県つくば市梅園 1-1-1　中央第 2
TEL：029-861-2000