BESOM Ver.3.0 β 版 のアルゴリズム

産業技術総合研究所 一杉裕志

2013 年 7 月 17 日

1 はじめに

本文書では、現在の BESOM の実装で用いているアルゴリズムについて述べる。（**間違いの指摘やコメントを歓迎いたします。**）

現在筆者は、 BESOM を機械学習アルゴリズムとして高速に安定動作させることを最優先で設計・実装を行っている。本文書ではアルゴリズムの生物学的妥当性については触れない。しかし、神経回路で実行可能なほど簡単かつ局所的な計算だけから構成されるアルゴリズムになっていると考えている。

各章の最後に現在の動作確認状況を簡単に書いたが、詳細な実験状況と結果については後日別途レポートする予定である。

以後の章では、筆者がこれまでに公開した文書を適宜参照する。その際には、以下の表記を用いる。

- IJCNN2007[1]
  近似確率伝搬アルゴリズムと大脳皮質の解剖学的構造との対応を述べたもの。
- IJCNN2011[5]
  近似 belief revision の提案と性能評価。
- TR2011[6]
  BESOM アルゴリズムの 2011 年時点での状況を報告するテクニカルレポート。

2 認識と学習の動作の定式化

認識と学習の原理は TR2011[6] 2 章で述べたものと変わっていない。以下に同じものを書く。

パラメタ θ のもとでの隠れ変数の値の組 $\mathbf{h}$ と観測変数の値の組 $\mathbf{i}$ との間の同時確率のモデルを $P(\mathbf{h},\mathbf{i}|\theta)$ とする。また、時刻 t における入力変数の値の組を $\mathbf{i}(t)$ とする。各時刻の入力は i.i.d. (独立同分布) に従うと仮定すると、θ のもとで入力データの列 $\mathbf{i}(1),\mathbf{i}(2),\cdots,\mathbf{i}(t)$ が生じる確率は以下のようになる。

$$\begin{aligned}&\prod_{i=1}^{t} P(\mathbf{i}(i)|\theta)\\ =\ &\prod_{i=1}^{t}\sum_{\mathbf{h}} P(\mathbf{h},\mathbf{i}(i)|\theta)\end{aligned} \tag{1}$$

学習の目的は、以下のようにパラメタを MAP 推定すること、すなわちパラメタ θ の事後確率を最大にすることである。

$$\theta^* \ =\ \operatorname*{argmax}_{\theta}\left[\prod_{i=1}^{t}\sum_{\mathbf{h}} P(\mathbf{h},\mathbf{i}(i)|\theta)\right]P(\theta) \tag{2}$$

本文書で述べる学習アルゴリズムは複雑だが、その本質は以下に述べるように、認識ステップと学習ステップの動作を表す 2 つの数式で書ける。

認識ステップでは、まず現在のパラメタ $\theta(t)$ に基づいて、与えられた入力 $\mathbf{i}(t)$ に対する隠れ変数の値の組の最大事後確率推定値 $\hat{\mathbf{h}}(t)$ （すなわち MPE, *most probable explanation*）を次のように求める。

$$\begin{aligned}\hat{\mathbf{h}}(t) &= \operatorname*{argmax}_{\mathbf{h}} P(\mathbf{h}|\mathbf{i}(t),\theta(t))\\ &= \operatorname*{argmax}_{\mathbf{h}} P(\mathbf{h},\mathbf{i}(t)|\theta(t))/P(\mathbf{i}(t))\\ &= \operatorname*{argmax}_{\mathbf{h}} P(\mathbf{h},\mathbf{i}(t)|\theta(t))\end{aligned} \tag{3}$$

次に学習ステップでは、式 (2) における隠れ変数の周辺化を推定値 $\hat{\mathbf{h}}(i)$ を用いることで近似してパラメタ推定し、結果を $\theta(t+1)$ とする [1]。

$$\theta(t+1) = \operatorname*{argmax}_{\theta}\left[\prod_{i=1}^{t} P(\hat{\mathbf{h}}(i),\mathbf{i}(i)|\theta)\right]P(\theta) \tag{4}$$

[1] 似た近似は Olshausen らの sparse-coding [8] でも行っている（[9] の p.3315 参照）。また、隠れマルコフモデルにおけるビタビ学習のオンライン版とも本質的に同じであると思われる。ビタビ学習とはビタビアルゴリズムで求めた MPE を使う学習アルゴリズムで、収束が証明されているらしい。

3　パラメタ θ の事前分布

パラメタ θ の事前分布は、 TR2011[6] では、近傍学習などににによって近似的に表現されていると解釈した。現バージョンのアルゴリズムも近傍学習は行っているが、それに加えて、勝率ペナルティと側抑制ペナルティという２つの事前分布の因子を導入している。

$$P(\theta) = P^{WinRate}(\theta)P^{Lateral}(\theta)P^{Neighbor}(\theta) \quad (5)$$

勝率ペナルティ $P^{WinRate}(\theta)$ と側抑制ペナルティ $P^{Lateral}(\theta)$ については、8 章と 9 章で述べる。

4　条件付確率表 (CPT) のモデル

4.1　旧バージョンの問題点

TR2011[6] ５章で定義した旧 meanOR モデルは、正規化係数 Z が定数にならないという致命的欠陥があることが分かった。これは認識の精度を著しく落とす原因になる。そこで、この問題を解決した新 meanOR モデルを以下の節で説明する。

4.2　新 meanOR モデル

あらためて、ＣＰＴモデルに対する要求仕様を整理すると以下のようになる。

1. 認識・学習が高速に実行可能でなければならない。

   認識を高速に行えるようにするための近似確率伝搬アルゴリズム（IJCNN2007[1]）や後述の OOBP アルゴリズム（6.2 節）は、導出においてＣＰＴモデルが下記の形をしていることを仮定している。

   $$P(x|u_1,\cdots,u_m) = \frac{1}{m}\sum_{k=1}^{m} w(x,u_k) \quad (6)$$

   また、 BESOM の学習則は、ＣＰＴモデルが下記の形をしていることを仮定している。

   $$P(x|u_1,\cdots,u_m) = f(w_1,\cdots,w_m) \quad \text{ただし } w_k = P(x|u_k) \quad (7)$$

2. スパース符号化モデル [8] で使われている線形和モデルと近く、一次視覚野の特性が再現できる。

3. 子ノードの活性状態は、活性親ノードが決める値のＯＲで近似できる。不活性な親ノードは、子ノードの確率分布に大きな影響を与えない。(noisy-OR model [7] と同様に、外界を比較的自然に表現できる。)

4. ０に近い値の重みの学習を必要としない。（０に近い重みは高い精度で学習するのが難しいため。）

以上の要求仕様をほぼ満たすＣＰＴモデルとして、下記の新 meanOR モデルを、現在の実装では用いている。

$$\begin{aligned} P(x|u_1,\cdots,u_m) &= \frac{1}{m}\sum_{k} w(x,u_k) \\ w(x,u) &= \begin{cases} 1/(s+1) & (u = u_\phi) \\ P(x|u) & (u \neq u_\phi) \end{cases} \end{aligned} \quad (8)$$

ただし s はノード X の非 ϕ 値ユニットの数であり、比較的大きな値 (10〜100) を想定している。

4.3　meanOR の精度

meanOR モデルは次のような状況では「信号源のＯＲ」のモデルとしては近似精度が落ちるので注意を要する。

「子ノード側のユニット数が少なく、
かつ
親ノード側に不活性ノードが多いとき。」

実際の脳ではユニット数が 100 程度あるので１つ目の条件は満たしていない。上の層ほどスパース性が増し不活性ノードの数は多くなるが、エッジ選択をすることで２番目の条件も満たさないようにすることができる。

現在の実験状況では、実は 5 章で述べるように入力に２値ノードを用いているので、少し問題である。(脳の一次感覚野の子ノードも２値ノードと解釈すべきかもしれない。) 入力ノードの親ノードのスパース性を上げすぎないようにすれば問題ないかもしれない。

5　入力の与え方と学習結果の可視化

TR2011[6] 同様、グレイスケールの画像の１ピクセルを２値ノードで表現する。現在は、ピクセルの色の黒さに比例する確率で値１を入力している。

λメッセージを受け取れる認識アルゴリズムの場合は、ピクセル値をそのままλメッセージとして入力する入力方法も選択できる。（この方がＣＰＴが滑らかになりやすい。実際の脳のＶ１への入力にも近いと思われる。）

ユニットの可視化には、そのユニットが勝者になった時の入力画像の平均画像を用いている。

6　認識アルゴリズム

6.1　旧バージョンの問題点

IJCNN2011[5] では、大脳皮質のコラム構造・６層構造と類似した構造を持つ ABR（近似 belief revision）アルゴリズムを提案し、層構造をしたネットワークで性能評価を行った。結果はそれほど悪くない近似精度であった。

しかし、実際に４層程度の BESOM ネットワークで MNIST 手書き数字認識に適用してみたところ、 ABR では認識精度が５０％程度しか出なかった。考えられる１つの原因として、 ABR では近似する際に「十分に多くの数の親ノードがある」という仮定を置いているが、現段階でシミュレーションに用いるネットワークはあまり規模が大きくなく、その仮定が成り立っていないということが考えられる。

その後、実はＣＰＴが式 (6) の形の時、belief propagation アルゴリズムは近似しなくても高速化できることを見い出した。また、性能評価を行ったところ、belief propagation で計算される MPM [2] は、MPE の悪くない近似であることを確認した。

次の節ではそのアルゴリズム（ OOBP, optimized original belief propagation と名付けた）の導出を説明する。

[2]MPM (maiximum posterior marginal, 最大事後周辺確率) とは、確率変数ごとの周辺事後確率の最大値の組のこと。周辺事後確率が $BEL(x)$ のときその最大値 $argmax_x BEL(x)$ が MPM における変数 X の値となる。マルコフ確率場を用いた画像修復や、ターボ符号の復号などで MPM が使われるようである。

6.2　OOBP アルゴリズムの導出

まず、下記はオリジナルの Pearl の確率伝搬アルゴリズムである。

$$
\begin{aligned}
BEL(x) &= \alpha\lambda(x)\pi(x) \\
\pi(x) &= \sum_{u_1,\cdots,u_m} P(x|u_1,\cdots,u_m)\prod_k \pi_X(u_k) \\
\lambda(x) &= \prod_l \lambda_{Y_l}(x) \\
\pi_{Y_l}(x) &= \beta_1\pi(x)\prod_{j\neq l}\lambda_{Y_j}(x) \\
\lambda_X(u_k) &= \beta_2\sum_x \lambda(x) \\
&\qquad \sum_{u_1,\cdots,u_m/u_k} P(x|u_1,\cdots,u_m)\prod_{i\neq k}\pi_X(u_i)
\end{aligned}
$$

ただし α, β_1, β_2 は正規化定数である。

このアルゴリズムを、式 (6) の形のＣＰＴモデルを仮定したうえで変形する。（近似は一切行わない。）なお、IJCNN2007[1] と同様に、親ノードからのメッセージは下記のように正規化されることを前提とする。

$$\sum_{u_k}\pi_X(u_k) = 1 \tag{9}$$

なおこのとき、

$$\sum_{u_1,\cdots,u_m}\prod_{i=1}^{m}\pi_X(u_i) = 1 \tag{10}$$

が成り立つ。

例：

$$
\begin{aligned}
&\sum_{u_1}\sum_{u_2}\pi_X(u_1)\pi_X(u_2) \\
=\ &\sum_{u_1}\pi_X(u_1)(\sum_{u_2}\pi_X(u_2)) \\
=\ &\Big(\sum_{u_1}\pi_X(u_1)\Big)\Big(\sum_{u_2}\pi_X(u_2)\Big) \\
=\ &1
\end{aligned} \tag{11}
$$

以上の前提のもとに、 OOBP アルゴリズムを導出する。

まず、ノード U_k から X へのメッセージ $\kappa_{U_k}(x)$ を下記のように定義する。

$$\kappa_{U_k}(x) = \sum_{u_k} w(x,u_k)\pi_X(u_k)$$

$\pi(x)$ の計算式は下記のように変形できる。（途中から定数 $1/m$ を省略している。）

$$\pi(x) = \sum_{u_1,\cdots,u_m} P(x|u_1,\cdots,u_m)\prod_i \pi_X(u_i)$$

$$= \sum_{u_1,\cdots,u_m} (\frac{1}{m}\sum_k w(x,u_k))\prod_i \pi_X(u_i)$$

$$\propto \sum_{u_1,\cdots,u_m}\sum_k w(x,u_k)\prod_i \pi_X(u_i)$$

$$= \sum_{u_1,\cdots,u_m}\sum_k w(x,u_k)\pi_X(u_k)\prod_{i\neq k} \pi_X(u_i)$$

$$= \sum_{u_1,\cdots,u_m} w(x,u_1)\pi_X(u_1)\prod_{i\neq 1} \pi_X(u_i)$$

$$+\cdots+\sum_{u_1,\cdots,u_m} w(x,u_m)\pi_X(u_m)\prod_{i\neq m} \pi_X(u_i)$$

$$= \sum_{u_1} w(x,u_1)\pi_X(u_1)\sum_{u_1,\cdots,u_m/u_1}\prod_{i\neq 1} \pi_X(u_i)$$

$$+\cdots+\sum_{u_m} w(x,u_m)\pi_X(u_m)\sum_{u_1,\cdots,u_m/u_m}\prod_{i\neq m} \pi_X(u_i)$$

$$= \sum_k\sum_{u_k} w(x,u_k)\pi_X(u_k)\sum_{u_1,\cdots,u_m/u_k}\prod_{i\neq k} \pi_X(u_i)$$

$$= \sum_k\sum_{u_k} w(x,u_k)\pi_X(u_k)$$

$$= \sum_k \kappa_{U_k}(x)$$

$\pi_{Y_l}(x)$ の計算式は下記のように変形できる。（ただし $\rho(x) = \pi(x)\lambda(x)$ と定義する。）

$$\pi_{Y_l}(x) = \beta_1\pi(x)\prod_{j\neq l}\lambda_{Y_j}(x)$$

$$= \beta_1(\pi(x)\prod_j \lambda_{Y_j}(x))/\lambda_{Y_l}(x)$$

$$= \beta_1\rho(x)/\lambda_{Y_l}(x)$$

$\lambda_X(u_k)$ の計算式の中に現れる式 $\sum_{u_1,\cdots,u_m/u_k} P(x|u_1,\cdots,u_m)\prod_{i\neq k}\pi_X(u_i)$ は下記のように変形できる。（途中、$\pi(x)$ の計算式の変形と同じ考え方を使う。）

$$\sum_{u_1,\cdots,u_m/u_k} P(x|u_1,\cdots,u_m)\prod_{i\neq k}\pi_X(u_i)$$

$$= \sum_{u_1,\cdots,u_m/u_k}\frac{1}{m}(\sum_{j\neq k} w(x,u_j) + w(x,u_k))\prod_{i\neq k}\pi_X(u_i)$$

$$= \frac{1}{m}\Big[\sum_{u_1,\cdots,u_m/u_k}\sum_{j\neq k} w(x,u_j)\prod_{i\neq k}\pi_X(u_i)$$

$$+w(x,u_k)\sum_{u_1,\cdots,u_m/u_k}\prod_{i\neq k}\pi_X(u_i)\Big]$$

$$= \frac{1}{m}(\pi(x) - \sum_{u_k} w(x,u_k)\pi_X(u_k) + w(x,u_k))$$

$$= \frac{1}{m}(\pi(x) - \kappa_{U_k}(x) + w(x,u_k))$$

これを使うと $\lambda_X(u_k)$ の計算式は下記のようになる。

$$\lambda_X(u_k)$$

$$= \beta_2\sum_x \lambda(x)\sum_{u_1,\cdots,u_m/u_k} P(x|u_1,\cdots,u_m)\prod_{i\neq k}\pi_X(u_i)$$

$$= \frac{\beta_2}{m}\sum_x \lambda(x)(\pi(x) - \kappa_{U_k}(x) + w(x,u_k))$$

以上の結果を整理し、正規化定数を付け直し、さらに loop のあるネットワークに適用可能なように適宜添え字 $t, t+1$ を付けた結果の OOBP アルゴリズムは以下のようになる。

$$\lambda_{Y_l}^{t+1}(x) = \beta_2\sum_{y_l}\lambda^t(y_l)(\pi^t(y_l) - \kappa_X^t(y_l) + w(y_l,x))$$
$$\lambda^{t+1}(x) = \prod_{l=1}^n \lambda_{Y_l}^{t+1}(x)$$
$$\pi_{Y_l}^{t+1}(x) = \beta_1\rho^{t+1}(x)/\lambda_{Y_l}^{t+1}(x)$$
$$\kappa_{U_k}^{t+1}(x) = \sum_{u_k} w(x,u_k)\pi_X^t(u_k)$$
$$\pi^{t+1}(x) = \sum_{k=1}^m \kappa_{U_k}^{t+1}(x)$$
$$\rho^{t+1}(x) = \lambda^{t+1}(x)\pi^{t+1}(x)$$
$$BEL^{t+1}(x) = \alpha\rho^{t+1}(x) \quad (12)$$

6.3 動作確認状況

このアルゴリズムの精度を IJCNN2011[5] で書いた方法で評価したところ、実際に loopy belief propagation の素朴な実装と同じ精度で MPM （各ノードの周

辺事後確率の最大値の組）を計算でき、計算速度は素朴な実装よりはるかに速いことを確認した。（各ノードのメッセージの計算に必要な計算量は親ノード・子ノードの数に対して線形時間ですむ。）

また、MPM は MPE をそこそこ近似することも確認している。

４層 BESOM （入力層＋中間層２層＋教師信号を与える最上位層）で MNIST 手書き数字認識を試してみたところ、（現在のところ pre-training などの工夫なしで）９０%前後の認識精度を達成している。

6.4 その他の認識アルゴリズム

現在のところ、 OOBP の他に厳密 MPE 計算、厳密 MPM 計算、山登り法（TR2011[6]3.2 節参照）、ABP（近似 belief propagation アルゴリズム、IJCNN2007[1] 参照）、ABR（近似 belief revision アルゴリズム、IJCNN2011[5] 参照）、loopy belief propagation アルゴリズム [7]、loopy belief revision アルゴリズム [7] が実装済みである。

7 学習アルゴリズム

TR2011[6] ４章に書いたものとほぼ同じであるが、「ぼかし関数」は今のところ実装していない。

また、TR2011[6]4.3 節で述べた、局所解を避けるための勝者ノイズの機構は、現在は採用していない。勝率ペナルティ、側抑制ペナルティによる正則化がその代わりになっている。

8 勝率ペナルティ

8.1 旧バージョンの問題点

TR2011[6] ７章に書いたスパース符号化の方法では、２層 BESOM では定性的には動いていたが、学習が進むにつれて活性ノード数が変化し、最適なスパース性パラメタの調整が難しいという問題があった。また、３層以上の BESOM ではうまく動かないという問題が明らかになった。

各ユニットの勝率（ MPE の値として選ばれる頻度）を一様にする勝率ペナルティについては、TR2011[6]4.2 節で述べた方法では、ノード数、ユニット数などの状況によっては効果が出ない場合があるという問題があった。

8.2 方針

勝率ペナルティを用いた学習則を、自然な原理から演繹的に導く。各ノードが目標とする勝率の分布と、実際の分布とのＫＬ情報量に基づいて、差が大きいときにペナルティを与えるものとする。こうすることで、学習が進むにつれ各ユニットの勝率が目標とする値に近づくと期待できる。

8.3 現バージョンのアルゴリズム

勝率ペナルティ $P^{WinRate}(\theta)$ を以下のように定義する。

$$P^{WinRate}(\theta) = \prod_{X \in \mathbf{X}} e^{-C^{WinRate} D_{KL}(Q(X)||P(X;\theta))} \quad (13)$$

ただし $\mathbf{X}$ はすべてのノードの集合、$C^{WinRate}$ は勝率ペナルティの強さを決める定数である。

$Q(X)$ は、ノード X の勝率の目標となる分布であり、ネットワーク設計者が分布の形を決める。例えば、ある層内のノード数を n 、各ノードのユニット数を $s+1$ 、目標とする活性ノード数を a とすると、

$$Q(X = x_i) = \begin{cases} (n-a)/n & (i = \phi) \\ a/ns & (i \neq \phi) \end{cases} \quad (14)$$

となる。

$D_{KL}(Q(X)||P(X;\theta))$ は分布 P と Q の間のＫＬ情報量で、以下の式で定義される。

$$D_{KL}(Q(X)||P(X;\theta)) = \sum_x Q(x) \log \frac{Q(x)}{P(x;\theta)} \quad (15)$$

ここで

$$R(x;\theta) = (s+1) \frac{Q(x)}{P(x;\theta)} \log \frac{Q(x)}{P(x;\theta)} \quad (16)$$

と定義する。ただし $s+1$ はノード X のユニット数である。$x(i)$ を θ のもとでの時刻 i における変数 X

の MPE の値の十分によい近似値とすれば[3]、

$$D_{KL}(Q(X)||P(X;\theta)) \approx \sum_{i=1}^{t} \frac{1}{t} R(x(i);\theta) \tag{17}$$

である。

すると $P^{WinRate}(\theta)$ は次のように書き直せる。

$$\begin{aligned} & P^{WinRate}(\theta) \\ = & \prod_{X\in\mathbf{X}} e^{-C^{WinRate} D_{KL}(Q(X)||P(X;\theta))} \\ \approx & \prod_{X\in\mathbf{X}} e^{-C^{WinRate} \sum_{i=1}^{t} (1/t) R(x(i);\theta)} \\ = & \prod_{X\in\mathbf{X}} \prod_{i=1}^{t} e^{-(1/t) C^{WinRate} R(x(i);\theta)} \\ = & \prod_{i=1}^{t} \prod_{X\in\mathbf{X}} e^{-(1/t) C^{WinRate} R(x(i);\theta)} \end{aligned} \tag{18}$$

このとき、学習ステップにおけるパラメタ θ を MAP 推定する式 (4) は次のように書き直せる。(ここでは説明を簡単にするために $P(\theta) = P^{WinRate}(\theta)$ とする。)

$$\begin{aligned} & \theta(t+1) \\ = & \operatorname*{argmax}_{\theta} \left[\prod_{i=1}^{t} P(\hat{\mathbf{h}}(i), \mathbf{i}(i)|\theta) \right] P^{WinRate}(\theta) \\ = & \operatorname*{argmax}_{\theta} \left[\prod_{i=1}^{t} \prod_{X\in\mathbf{X}} P(x(i)|parents(x(i));\theta) \right] \\ & \left[\prod_{i=1}^{t} \prod_{X\in\mathbf{X}} e^{-(1/t) C^{WinRate} R(x(i);\theta)} \right] \\ = & \operatorname*{argmax}_{\theta} \prod_{i=1}^{t} \prod_{X\in\mathbf{X}} \Big[P(x(i)|parents(x(i));\theta) \\ & e^{-(1/t) C^{WinRate} R(x(i);\theta)} \Big] \end{aligned} \tag{19}$$

ただし、$parents(x(i))$ は時刻 i におけるノード X の親ノードの値の組である。

上記の式は、すべてのノード X の下に制約ノード R_X があり、条件付き確率が

$$P(R_X = 1|X = x;\theta) = e^{-(1/t) C^{WinRate} R(x;\theta)} \tag{20}$$

で定義されているようなベイジアンネットワークにおけるパラメタの最尤推定と同じ形をしている。(ノード R_X には常に観測値 1 が与えられるものとする。) したがって、MPE 計算とカウンティングを用いた BESOM の学習則が使えることになる。認識時には TR2011[6]10.3.1 節で書いた方法で、belief propagation や belief revision を用いることができる。

正則化パラメタ $C^{WinRate}$ に $1/t$ がかかっているので、時刻とともにペナルティの影響は小さくなっていく。実装上は $1/t$ の代わりにユニット x_i の局所学習率 α_i を用いるのがよいだろう。(ただし現在の実装では $1/t$ での減衰は用いておらず、正則化パラメタ $C^{WinRate}$ をスライダで調整して実験している。後述の側抑制ペナルティについても同様である。)

$R(x;\theta)$ を計算する際に必要となる勝率 $P(x;\theta)$ は学習によって推定したものを使う。

8.4 勝率ペナルティの強さの近似式

$f(x) = x\log(x)$ のとき $f'(x) = \log(x) + 1, f(1) = 0, f'(1) = 1$ 、また、P と Q がほぼ一致するならば $Q(x)/P(x;\theta) \approx 1$ なので、下記の近似式が成り立つ。

$$\begin{aligned} R(x;\theta) &= (s+1)(Q(x)/P(x;\theta)) \log Q(x)/P(x;\theta) \\ &\approx (s+1)(Q(x)/P(x;\theta) - 1) \end{aligned} \tag{21}$$

8.5 動作確認状況

すでに実装し、動作を確認している。ただし、OOBP はスパース性が高いときに振動しやすく、うまく学習が進まないことがある。認識に山登り法を使う場合は安定して動作している。

9 側抑制ペナルティ

9.1 旧バージョンの問題点

TR2011[6] では、側抑制ＩＣＡとスパース符号化を用いたクラスタリングとを両立させるために、あえてノードどうしが完全に独立にならない学習則を採用していた。しかし、ϕ 値の扱いがアドホックで妥当性に疑問があった。

[3]学習がある程度収束に近づいた時にはじめてこの仮定が成り立つ点には要注意。9 章の側抑制ペナルティについても同様である。

9.2 方針

側抑制を用いた学習則を、自然な原理から演繹的に導く。側抑制を行う変数のペアの相互情報量が大きくなるようなパラメタに対してはペナルティを与える。こうすることで、変数のペアの相互情報量が０、すなわち独立に近づくと期待できる。

9.3 現バージョンのアルゴリズム

側抑制に相当するペナルティ $P^{Lateral}(\theta)$ を以下のように定義する。

$$P^{Lateral}(\theta) = \prod_{(U,V)\in L} e^{-C^{Lateral} I(U,V;\theta)} \tag{22}$$

ただし $C^{Lateral}$ は側抑制ペナルティの強さを決める定数、L は側抑制を行うノードのペアの集合とする。通常、同じ層に属し、同じ子ノードを共有するノードどうしがお互いに側抑制するように、側抑制のネットワークを構築しておく。

$I(U,V;\theta)$ はノード U と V の間の相互情報量で、以下の式で定義される。(θ は省略した。)

$$I(U,V) = \sum_x \sum_y P(u,v) \log \frac{P(u,v)}{P(u)P(v)} \tag{23}$$

ここで

$$\begin{aligned} & R(u,v) \\ = \; & (s+1)^2 \frac{P(u,v)}{P(u)P(v)} \log \frac{P(u,v)}{P(u)P(v)} \\ = \; & (s+1)^2 (P(u|v)/P(u)) \log P(u|v)/P(u) \end{aligned} \tag{24}$$

と定義する。$u(i), v(i)$ を θ のもとでの時刻 i における変数 U, V の MPE の値の十分によい近似値とすれば、

$$I(U,V;\theta) \approx \sum_{i=1}^{t} \frac{1}{t} R(u(i), v(i); \theta) \tag{25}$$

である。

あとは 8 章と同様の議論により、側抑制するノードのペア $(U,V) \in L$ ごとに共通子ノード R_{UV} があってその条件付き確率が

$$P(R_{UV} = 1 | u, v; \theta) \quad = \quad e^{-(1/t) C^{Lateral} R(u,v;\theta)} \tag{26}$$

であるとみなすことができる。認識時には TR2011[6]10.4.2 節で述べたメッセージ計算式を使うことで、 belief propagation や belief revision を用いることができるようになる。(belief propagation を使う場合はメッセージ計算式の max の代わりに Σ を用いる。)

なお、 $R(u,v;\theta)$ の値の計算に必要なる $P(u|v;\theta)$ 、$P(u;\theta)$ の値は、8 章と同様、学習により推定したものを使う。

9.4 側抑制の強さの近似式

U と V がほぼ独立ならば $P(u|v)/P(u) \approx 1$ なので、下記の近似式が成り立つ。

$$\begin{aligned} R(u,v) \quad &= \quad (s+1)^2 (P(u|v)/P(u)) \log P(u|v)/P(u) \\ &\approx \quad (s+1)^2 (P(u|v)/P(u) - 1) \end{aligned} \tag{27}$$

9.5 動作確認状況

すでに実装し、多くの場合、学習が進むと速やかにノード同士が独立になることを確認している。

10 エッジ選択アルゴリズム

TR2011[6] 9 章に書いたノード単位のエッジ選択のアルゴリズムを、層間の結合と、側抑制結合に対して実装して、動作を確認ずみである。

エッジ選択は単に認識に必要な計算量を減らすだけでなく、OOBP アルゴリズムが振動しにくくなる効果もあるようである。また、一種の正則化の機構として働き、汎化能力を向上させる効果も期待できる。

(なお、側抑制結合に関しては、エッジ選択に用いるスコアの計算の実装がまだ不完全であり、修正する予定である。)

11 おわりに

現在の実装は、 OOBP とエッジ選択の機構により、１入力の認識・学習にかかる時間がノード数に対してほぼ線形時間で抑えられている。すなわち、計算量の

観点からはスケーラブル（大規模化可能）なオンライン機械学習アルゴリズムになっている。

認識精度に関しては、次の２点が問題である。

1. OOBP アルゴリズムの振動の問題。認識精度が悪くなる時には、メッセージの振動が頻繁に起きていることが多い。この問題に対して何らかの工夫をすることが精度向上に向けた大きな課題である。これまでの実験では、エッジ数を減らすと振動が起きにくくなるようである。また、後述の短期記憶の機構も振動を減らす効果があるのではないかと期待している。

2. OOBP を認識アルゴリズムとして使う場合、MPM を MPE の近似として用いていることになるが、その妥当性が不明である。実際の脳は MPM と MPE のどちらを計算しているかについては、それぞれの証拠と思われる知見があるため、よくわからない。より詳細な神経科学的知見の文献調査が必要である。

現在、 BESOM は機械学習アルゴリズムとしてなんとか動作するようになったが、まだ大脳皮質の基本機能を完全に模倣するには至っていない。未実装として残っている重要機能に、短期記憶がある。これは視覚野において「目の前の物体は急には変化しない」という事前知識を作り込むことで自然に発生する現象だと筆者は考えており、ベイジアンネットの基本的枠組みの範囲内で実現可能である。短期記憶の機構を作り込めば、 slow feature analysis と同じ効果が出て、視覚刺激の様々な変化に対する不変性が特別な作り込みなしで獲得できるのではないかと期待している。

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 (IJCNN 2007), pp.1065–1070, Aug 2007.
http://staff.aist.go.jp/y-ichisugi/besom/20070509ijcnn-paper.pdf

[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] 一杉裕志、「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, Sep 2009.
http://staff.aist.go.jp/y-ichisugi/besom/AIST09-J00006.pdf

[4] Yuuji Ichisugi, Haruo Hosoya: Computational Model of the Cerebral Cortex that Performs Sparse Coding Using a Bayesian Network and Self-Organizing Maps, In Proc. of 17th International Conference on Neural Information Processing (ICONIP 2010), Part I, LNCS 6443, pp.33–40, Nov 2010.
http://staff.aist.go.jp/y-ichisugi/besom/2010iconip.pdf

[5] Yuuji Ichisugi: "Recognition Model of Cerebral Cortex based on Approximate Belief Revision Algorithm", To appear in Proc. of IJCNN 2011.
http://staff.aist.go.jp/y-ichisugi/besom/2011ijcnn.pdf

[6] 一杉裕志、「大脳皮質のアルゴリズム BESOM Ver.2.0」産業技術総合研究所テクニカルレポート AIST11-J00009, Sep 2011.
http://staff.aist.go.jp/y-ichisugi/besom/AIST11-J00009.pdf

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

[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] Olshausen BA, Field DJ, Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1 ?, Vision Res., Vol. 37, No. 23, pp. 3311-3325, 1997.