BDD 上の命題化確率計算に基づく EM アルゴリズム

Propositionalizing the EM algorithm by BDDs

石畠正和 [1*]　亀谷由隆 [1]　佐藤泰介 [1]　湊真一 [2]
Ishihata Masakazu[1]　Yoshitaka Kameya[1]　Taisuke Sato[1]　Shin-ichi Minato[2]

[1] 東京工業大学大学院情報理工学研究科
[1] Graduate School of Information Science and Engineering, Tokyo Institute of Technology
[2] 北海道大学大学院情報科学研究科
[2] Graduate School of Information Scince and Technology, Hokkaido University

Abstract: We propose two EM algorithms that work on binary decision diagrams (BDDs). They open a way to applying BDDs to statistical inference in general and machine learning in particular. We also present the complexity analysis of noisy-OR models based on BDDs.

1　はじめに

Binary Decision Diagram (BDD) は論理関数のコンパクトな表現として知られている [1, 2]. 本稿では BDD の新しい応用として, BDD 上で動作する Expectation-Maximization (EM) アルゴリズム [3] を提案する. EM アルゴリズムは不完全データに対する最尤推定を行うアルゴリズムとして広く知られている. 更に本稿ではより論理式をコンパクトに扱える手法である *decomposed BDD* を用いる. Decomposed BDD は, 論理関数 F をいくつかの互いに独立な論理関数に分割し, F を簡単化する一つの方法として VLSI 論理設計分野で知られている [4]. F と分割された論理関数はそれぞれ 1 つの BDD として表現される. 提案アルゴリズムはこれらの BDD をサブルーチンのように呼び出すことによって再帰的に確率計算を行う.

提案手法である BDD-EM アルゴリズムは一般的かつ効率的なアルゴリズムである. ここで一般的というのは, BDD-EM アルゴリズムは任意の命題論理式に対して適用可能であることを意味し, 効率的というのはその計算量が BDD の全節点数に比例することを意味する. これにより 4 章で示すように N 入力 noisy-OR モデルのパラメタ学習を $O(N)$ で行うことが可能となる.

本稿ではまず, 2 章において EM アルゴリズムと BDD の概要を述べ, 3 章で提案手法である BDD-EM アルゴリズムを紹介する. 4 章で N 入力 noisy-OR モデルにおける提案手法の計算量を評価し, 5 章では noisy-OR モデルのパラメタ学習の結果を示す. 最後に 6 章と 7 章にて関連研究とまとめを述べる.

*連絡先：東京工業大学大学院情報理工学研究科計算工学専攻
〒 152-8552 東京都目黒区大岡山 2-12-1
E-mail: ishihata@mi.cs.titech.ac.jp

2　準備

2.1　EM アルゴリズム

ここでは本稿が扱う問題を定義し, EM アルゴリズムの定式化を行う. 本稿では F を k 個の確率的命題変数 $X_1, X_2, \ldots, X_k$ からなる論理関数とし, F の値のみが観測可能で $X_1, X_2, \ldots, X_k$ の値は観測不可能とする. 確率的命題変数 $X_1, X_2, \ldots, X_k$ は互いに独立して 0 か 1 の値をとる. 以下, 記述を簡単にするために F を論理関数 F の値をとる確率的命題変数を表すことにし, F を観測命題と呼ぶ. これに対して $X_1, X_2, \ldots, X_k$ を基本命題と呼ぶ. 例えば, ある学生の遅刻に関する問題を考えてみる. 学生は普段は自転車で通学しているが, 雨の日はバスを利用する. このとき彼が遅刻することを $F = O \vee (R \wedge B)$ のように論理式で定式化できるとする. F, O, R, B はそれぞれ「彼は遅刻した」「彼は寝坊した」「雨が降っている」「バスは遅れた」という確率的命題である. 以上の設定において, 本稿では観測命題 F の観測値[1]から, EM アルゴリズムを用いて各基本命題 (O, R, B) が真を取る確率を推定する問題を扱う. また, いくつかの基本命題が i.i.d.(独立同時分布) であるような問題も扱うことにする. 例えば隠れマルコフモデルにおいて各状態は時刻 t と時刻 $t+1$ において i.i.d. である. i.i.d. を扱うために基本命題集合 $\mathbf{X} = \{X_1, X_2, \ldots, X_k\}$ に対して以下で定義される $\mathcal{S}$ を考える. $\mathcal{S}$ は $\mathbf{X}$ の分割の集合であり, $s \in \mathcal{S}$ は任意の $X, X' \in \mathbf{X}$ において, 「$X, X' \in s \Leftrightarrow X$ と X' は i.i.d.」を満たす. この s は基本命題 $X \in s$ が $X = x$ となる確率を表すパラメタ $\theta_{s,x}$ を持つ (すなわち $\sum_{x \in \{0,1\}} \theta_{s,x} = 1$ である). 以後,

[1] ここでは説明を簡単にするために 1 回のみ F の真偽を観測するものとする. 提案手法を複数回の観測値が与えられる問題に対して拡張するのは容易である.

パラメタベクトル $\langle\theta_{s,0},\theta_{s,1}\rangle$ を $\boldsymbol{\theta_s}$, すべてのパラメタのベクトルを $\boldsymbol{\theta}$ を用いて表す.

ϕ を基本命題集合 $\mathbf{X}$ に対する値の割り当ての一つとし, Φ をすべての ϕ の集合とする. すると F の値 $f\in\{0,1\}$ は ϕ に対して一意に決まるので, F は $\phi\in\Phi$ から $f\in\{0,1\}$ の関数として $F(\phi)=f$ と書ける. また, $F(\phi)=f$ となるようなすべての ϕ の集合を $F^{-1}(f)=\{\phi\in\Phi\mid F(\phi)=f\}$ と書くことにする. 更に, ある割り当て ϕ とあるパラメタ $\theta_{s,x}$ に対して $\sigma_{s,x}(\phi)=|\{X\in s\mid\phi(X)=x\}|$ と定義する. $\sigma_{s,x}(\phi)$ は $X\in s$ が ϕ 中で $X=x$ として現れる回数の総和である.

上で定義した問題における EM アルゴリズムは以下で定義される 2 つのステップからなる.

- **E-step**: $X\in s$ が $X=x$ として出現する回数の**条件付き期待値** $E_{\boldsymbol{\theta}}[\sigma_{s,x}(\cdot)\mid F=f]$ を計算する.

$$E_{\boldsymbol{\theta}}[\sigma_{s,x}(\cdot)\mid F=f]=\eta^x_{\boldsymbol{\theta}}[s]/P_{\boldsymbol{\theta}}(F=f)$$

$$\eta^x_{\boldsymbol{\theta}}[s]=\sum_{\phi\in F^{-1}(f)}\sigma_{s,x}(\phi)\prod_{s'\in\mathcal{S}}\prod_{x'\in\{1,0\}}\theta_{s',x'}^{\sigma_{s',x'}(\phi)}\quad(1)$$

$$P_{\boldsymbol{\theta}}(F=f)=\sum_{\phi\in F^{-1}(f)}\prod_{s\in\mathcal{S}}\prod_{x\in\{1,0\}}\theta_{s,x}^{\sigma_{s,x}(\phi)}\quad(2)$$

- **M-step**: $\boldsymbol{\theta}$ を $\hat{\boldsymbol{\theta}}$ に更新する. 固定した s に対し $\hat{\boldsymbol{\theta}}$ の各要素 $\hat{\theta}_{s,x}$ は $E_{\boldsymbol{\theta}}[\sigma_{s,x}(\cdot)\mid F=f]$ に比例するように決める.

$|F^{-1}(f)|$ は基本命題の数に対して指数的に増加するので, E-step の計算量も指数的に増えることが問題になる. 本稿ではこの問題に対して, 論理関数をコンパクトに表現したデータ構造である BDD を用い, 動的計画法による条件付き期待値計算法を提案する.

2.2 Binary Decision Diagram

BDD は論理関数 F を表現する有向非循環グラフである. BDD はいくつかの変数節点と 1 定数節点, 0 定数節点を持つ. それぞれの定数節点を $\boxed{1}$, $\boxed{0}$ と表記する. 変数節点 n は F 中のある命題変数 X がラベル付けされており, これを $Label(n)$ と表現する. 異なる節点が同じラベルを持つこともある. また n は 2 つの子節点 1-child と 0-child を持ち, それぞれ $Ch_1(n)$,$Ch_0(n)$ と書く. n から $Ch_1(n)$ に至る辺のことを 1-枝と呼び実線で表現し, $Ch_0(n)$ に至る辺のことを 0-枝と呼び破線で表現する. 1-枝と 0-枝はそれぞれ $Label(n)$ の値が 1, 0 であることを意味する. 更に根節点から $\boxed{1}$($\boxed{0}$) に至るパスは $F=1$ ($F=0$) となる変数の割り当てを表現する.

図 1 の (a) と (b) はそれぞれ $F\Leftrightarrow(A\vee B)\wedge\bar{C}$ を表現した真理値表と BDD である. (b) のように, BDD が完全二分木であるとき, 特に二分決定木と呼ぶ. 例えば (b) の根節点から一番左の $\boxed{0}$ に至るパスは命題変数 A,B,C がそれぞれ 0 であるとき $F=0$ であることを表しており, これは (a) の真理値表の 1 行目に対応する.

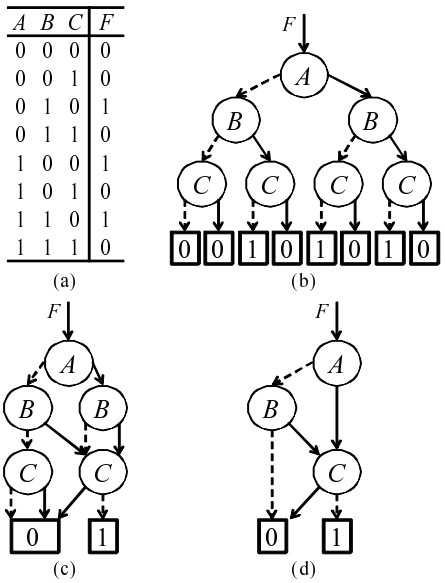

図 1: $F\Leftrightarrow(A\vee B)\wedge\bar{C}$ の (a) 真理値表, (b) 二分決定木, (c)BDD, (d)ROBDD

一般的に 1 つの論理式を表現する BDD は複数存在するが, reduced orderd BDD(ROBDD) は論理式に対して一意に決定される. ROBDD とはすべてのパスで現れる変数の順序が一意であり, 簡略化規則をそれ以上適用できない BDD のことである [2]. ここで変数が現れる順序のことを変数順序という. BDD には以下の 2 つの簡略化規則がある.

- 削除規則: 節点 n において, $Ch_1(n)=Ch_0(n)$ のとき n を削除する (図 2).
- 共有規則 : 節点 n,n' において, $Label(n)=Label(n')$, $Ch_1(n)=Ch_1(n')$, $Ch_0(n)=Ch_0(n')$ のとき n,n' を共有する (図 3).

図 1(c) の BDD は二分決定木に共有規則のみを施したものであり ROBDD ではない. 図 1(d) は (c) に更に削除規則を適用した ROBDD である.

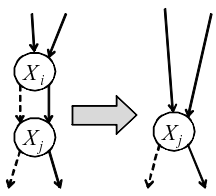

図 2: 削除規則

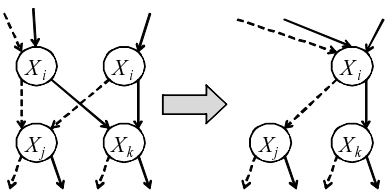

図 3: 共有規則

2.3 計算上の注意

ここでは BDD 上で動作する EM アルゴリズムである BDD-EM アルゴリズムを提案する前に, アルゴリズムを考える上で重要な 3 つのポイントを示す.

3 段階の確率/期待値計算

前節で述べたように, 同じラベルが付けられた BDD の節点は同じ基本変数を共有し, 更に互いに i.i.d. な基本命題は同じパラメタを共有する. よって確率/期待値計算を行うときはパラメタレベル, 変数レベル, 節点レベルの 3 段階を区別して考える必要がある. 式 (1) はパラメタと 1 対 1 に対応する $\mathcal{S}$ に沿って計算するため, パラメタレベルの記述である. ここで式 (1) を以下のように書きかえる.

$$\eta^x_{\boldsymbol{\theta}}[s] = \sum_{\phi\in F^{-1}(f)} \sum_{X\in\mathbf{X}:X\in s} \mathbf{1}_{\phi(X)=x} \prod_{Z\in\mathbf{X}} \theta^{\phi(Z)}_{[Z]}\theta^{1-\phi(Z)}_{[\bar{Z}]} \quad (3)$$

$[Z]$ は $Z\in s$ なる $s\in\mathcal{S}$ を表し, $\theta_{[Z]}=\theta_{s,1}=P_{\boldsymbol{\theta}}(Z{=}1)$, $\theta_{[\bar{Z}]}=\theta_{s,0}=P_{\boldsymbol{\theta}}(Z{=}0)$ である. また $\mathbf{1}_{\phi(X)=x}$ は $\phi(X)=x$ が成り立つときに 1 を取り, それ以外のときは 0 を取る関数である.

式 (3) の $\eta^x_{\boldsymbol{\theta}}[s]$ は, $F(\phi)=f$ なるすべての ϕ においてパラメタ $\boldsymbol{\theta}_s$ を持つ基本命題 X が $X=x$ で現れたとき, $P(\phi)=\prod_{Z\in\mathbf{X}}\theta^{\phi(Z)}_{[Z]}\theta^{1-\phi(Z)}_{[\bar{Z}]}$ を計算し, $\eta^x_{\boldsymbol{\theta}}[s]$ に加えることで計算される. この計算法は $\mathcal{S}$ に沿った変数レベルの計算であることがわかる.

更に式 (3) を節点レベルの記述に書きかえ, BDD 上での条件付き期待値の計算法を導出する. 今, 図 1(b) のような二分決定木を考えると, 二分決定木のパス π_ϕ はある値の割り当て ϕ と対応し, π_ϕ 上にはすべての基本命題が必ず一度ずつ現れる. よって式 (3) は以下のように書きかえることができる.

$$\eta^x_{\boldsymbol{\theta}}[s] = \sum_{\pi_\phi:\phi\in F^{-1}(f)} \sum_{n'\in\pi_\phi:L_{n'}\in s} \mathbf{1}_{\phi(L_{n'})=x} \prod_{n\in\pi_\phi} \theta^{\phi(L_n)}_{[L_n]}\theta^{1-\phi(L_n)}_{[\overline{L_n}]} \quad (4)$$

ここで $n\in\pi_\phi$ はパス π_ϕ 上の節点 n のことであり, $L_n=\mathit{Label}(n)$ である. n が $X\in s$ でラベル付けられるとき, n の x-枝は $X=x$ を表し $(x\in 0,1)$, これはパラメタ $\theta_{s,x}$ に対応する. すると式 (4) は, $F(\phi)=f$ なる ϕ に対応するパス π_ϕ 上で, パラメタ $\theta_{s,x}$ に対応する x-枝が現れる度に $P(\phi)$ を $\eta^x_{\boldsymbol{\theta}}[s]$ に加えることで計算できる. すなわち $P(\phi)$ は π_ϕ に沿って枝に対応するパラメタを乗ずることで計算される. 本稿ではこの計算法を簡略化規則が適用されている BDD に拡張する.

確率計算と期待値計算の違い

期待値計算を行う場合は確率計算では求められなかったような注意が新たに必要となる. 例えば図 1 の論理式 $F\Leftrightarrow(A\vee B)\wedge\bar{C}$ について考えると, 図 1(a) の真理値表から $P_{\boldsymbol{\theta}}(F{=}1)$ は $\theta_{[\bar{A}]}\theta_{[B]}\theta_{[\bar{C}]}+\theta_{[A]}\theta_{[\bar{B}]}\theta_{[\bar{C}]}+\theta_{[A]}\theta_{[B]}\theta_{[\bar{C}]}$ と計算される. その一方で, BDD から再帰的に $P_{\boldsymbol{\theta}}(F{=}1)$ を計算する方法 (3 章で説明する) を用いると, BDD の形によって計算過程が異なる. 例えば図 1 の (c) と (d) の BDD に対してこの方法を適用すると $P_{\boldsymbol{\theta}}(F{=}1)$ はそれぞれ $(\theta_{[\bar{A}]}\theta_{[B]}+\theta_{[A]}(\theta_{[\bar{B}]}+\theta_{[B]}))\theta_{[\bar{C}]}$ と $(\theta_{[\bar{A}]}\theta_{[B]}+\theta_{[A]})\theta_{[\bar{C}]}$ と計算される. ここですべての $X\in\mathbf{X}$ において $\theta_{[X]}+\theta_{[\bar{X}]}=1$ が成り立つので, この 2 つの確率値は明らかに等しい. これは BDD の削除規則が $P_{\boldsymbol{\theta}}(F{=}1)$ の計算値に影響しないことを意味する. しかし, 削除規則は期待値計算に影響を与える. 仮に, すべての $s\in\mathcal{S}$ において $|s|=1$ が成り立つ (すなわち, どの変数もパラメタを共有しない) 場合を考える. つまり X は $[X]$ の唯一の要素であり, すべての値の割り当て ϕ において $\phi(X)=x$ のとき $\sigma_{[X],x}(\phi)=1$, そうでなければ $\sigma_{[X],x}(\phi)=0$ が成り立つ. すると式 (3) は単純に以下のように書ける.

$$\begin{aligned}\eta^x_{\boldsymbol{\theta}}\big[[X]\big] &= \textstyle\sum_{\phi\in F^{-1}(f)} \mathbf{1}_{\phi(X)=x} \prod_{Z\in\mathbf{X}} \theta^{\phi(Z)}_{[Z]}\theta^{1-\phi(Z)}_{[\bar{Z}]} \\ &= P_{\boldsymbol{\theta}}(F{=}f, X{=}x) \end{aligned} \quad (5)$$

今, $F=1$ が観測されたときの $B=1$ の条件付き期待値 $\eta^1_{\boldsymbol{\theta}}\big[[B]\big]$ を計算する. 式 (5) より, 真理値表の $F=1$ かつ $B=1$ なる行を取り出すことで期待値を計算すると $\eta^1_{\boldsymbol{\theta}}\big[[B]\big]=\theta_{[\bar{A}]}\theta_{[B]}\theta_{[\bar{C}]}+\theta_{[A]}\theta_{[B]}\theta_{[\bar{C}]}$ となる. その一方で, 図 1(d) の ROBDD を見ると, 基本命題 A でラベル付けられた節点の 1-枝は直接 C でラベル付けされた節点を指している. つまり, この ROBDD には $\theta_{[A]}\theta_{[B]}\theta_{[\bar{C}]}$ を計算するための情報が明示されていないといえる. この情報の欠落は BDD の削除規則によって引き起こされ, 期待値を計算するときは削除された節点を「数値的に復元」する必要がある.

Decomposed BDD

問題によっては観測命題 F を *decomposed BDD* で表現する方がコンパクトになる場合がある. 本稿で考える decomposed BDD とは *BDD fragment* の集合であり, BDD fragment は基本命題と他の BDD fragment に対応する**中間命題**から構成される. 本稿では異なる BDD fragment 中に現れる変数は互いに独立であると仮定する.

F を構成する基本命題集合を $\mathbf{X}$ と書くのに対して, F を含めた基本命題でないすべての命題集合を $\mathbf{U}$ と書く. 同時に $\mathbf{U}$ に対して, $\mathit{Level}:\mathbf{U}\to\{1,2,\ldots,|\mathbf{U}|\}$ を定義し, これを $\mathbf{U}$ の順序関数とする. ここで Level は $\mathit{Level}(F)=|\mathbf{U}|$ を満たし, $U,U'\in\mathbf{U}$ において $\mathit{Level}(U)<\mathit{Level}(U')$ のとき U は U' より順序が早いという. すべての $U\in\mathbf{U}$ は $\mathbf{X}$ と自分より順序の早

い $\mathbf{U}$ の要素の集合 (すなわち $\mathbf{X} \cup \{U' \mid Level(U') < Level(U)\}$) 中のいくつかの変数からなる論理関数である. 各 U の論理式は BDD で表現され, 各々の BDD のことを BDD fragment と呼び, δ_U と書く. $\mathbf{U}\backslash\{F\}$ の要素のことを**中間命題**と呼ぶ. BDD fragment δ_U 中の全節点の集合を $\mathbf{N}(\delta_U)$, 全変数の集合を $\mathbf{V}(\delta_U)$ で表す. そして全 BDD fragment の集合を $\Delta_F = \{\delta_U \mid U \in \mathbf{U}\}$ と表し, これを *decomposed BDD* と呼ぶ[2].

各 BDD は独自の変数順序を持ち,BDD fragment δ_U の変数順序は $Ord_U : \mathbf{V}(U) \to \{1, 2, \ldots, |\mathbf{V}(U)|\}$ によって定義される[3]. $Ord_U(X) < Ord_U(X')$ であるとき, これを $X \prec_U X'$ と書くことにする. 更に, 文脈より U が明らかである場合は U を省略して単純に $\prec$ と書く. また $Ord_U(n) = 1$ となるのは, n が根節点であるときでありそのときに限る.

図 4 は decomposed BDD Δ_F である. 図より $\mathbf{U} = \{F, X, Y\}$, $\mathbf{X} = \{A, B, C, D, E\}$ であるとわかる. また, BDD fragment δ_Y に注目してみれば, $\mathbf{N}(\delta_Y) = \{D, E, E\}$, $\mathbf{V}(\delta_Y) = \{D, E\}$ であることがわかる. 提案手法である BDD-EM アルゴリズムは, $F = f$ を観測したときの $X \in \mathbf{X}$ のパラメタを推定する.

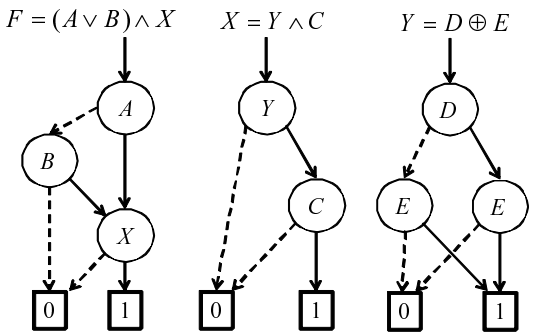

図 4: F を表現した decomposed BDD

3 提案方法

ここでは BDD 上で動作する EM アルゴリズムである **BDD-EM アルゴリズム**を提案する. BDD-EM() は F の観測データ f が与えられたとき, BDD-EM アルゴリズムを用いて各基本命題のパラメタを学習する疑似コードである. IterateEM() 中に現れる GetBackward(U) は**後ろ向き確率** $\mathcal{B}^x_{\boldsymbol{\theta}}[n]$ を, GetForward(U) は**前向き確率** $\mathcal{F}_{\boldsymbol{\theta}}[n]$ を, GetInside(U) は**内側確率** $\mathcal{P}^u_{\boldsymbol{\theta}}[U]$ を計算し, GetOutsideExpe(U) は**外側確率** $\mathcal{Q}^x_{\boldsymbol{\theta}}[X]$ と**条件付き期待値** $\eta^y_{\boldsymbol{\theta}}[Y]$ を更新する. ここで $U \in \mathbf{U}$, $n \in \mathbf{N}(\delta_U)$, $X \in \mathbf{V}(\delta_U) \cap \mathbf{X}$, $Y \in \mathbf{V}(\delta_U) \cap \mathbf{U}$, $u, x, y \in \{1, 0\}$ である. InitializeAll() はこれら 4 つの確率と条件付き期待値を初期化する.

```
1: Procedure: BDD-EM()
2:    Initialize all parameters θ;
3:    repeat
4:       IterateEM();
5:    until the parameters θ converge;
6: end
```

```
1: Procedure: IterateEM()
2:    // E-step
3:    InitializeAll()
4:    U = U;
5:    while U ≠ φ do
6:       U = argmin_{U'∈U} level(U');
7:       GetBackward(U);
8:       GetForward(U);
9:       GetInside(U);
10:      U = U\{U};
11:   end while
12:   U = U;
13:   while U ≠ φ do
14:      U = argmax_{U'∈U} level(U');
15:      GetOutsideExpe(U);
16:      U = U\{U};
17:   end while
18:   // M-step
19:   for each s ∈ S do
20:      θ_{s,1} ∝ η^1_θ[s]/P^f_θ[F];
21:      θ_{s,0} ∝ η^0_θ[s]/P^f_θ[F];
22:   end for
23: end
```

3.1 後ろ向き確率と前向き確率

GetBackward(U) は BDD fragment δ_U 中の全節点 $n \in \mathbf{N}(\delta_U)$ において**後ろ向き確率**を計算する. 後ろ向き確率 $\mathcal{B}^1_{\boldsymbol{\theta}}[n]$ は節点 n から $\boxed{1}$ に至る全パスの確率の総和, $\mathcal{B}^0_{\boldsymbol{\theta}}[n]$ は n から $\boxed{0}$ に至る全パスの確率の総和である. よって $\mathcal{B}^1_{\boldsymbol{\theta}}[\boxed{1}] = 1, \mathcal{B}^1_{\boldsymbol{\theta}}[\boxed{0}] = 0$ とし, 同様に $\mathcal{B}^0_{\boldsymbol{\theta}}[\boxed{1}] = 0, \mathcal{B}^0_{\boldsymbol{\theta}}[\boxed{0}] = 1$ とする. 後ろ向き確率は定数節点から根節点に向かって順に計算していく. それとは対照的に GetForward(U) は根節点から定数節点に向かって順に**前向き確率**を計算する. 前向き確率 $\mathcal{F}_{\boldsymbol{\theta}}[n]$ は根節点から節点 n に至るすべてのパスの確率の総和である. ここで $\mathcal{F}_{\boldsymbol{\theta}}[n]$ は InitializeAll() において $\mathcal{F}_{\boldsymbol{\theta}}[root] = 1, \mathcal{F}_{\boldsymbol{\theta}}[n] = 0$ $(n \neq root)$ に初期化される. $N = |\mathbf{N}(\delta_U)|$ とすると GetBackward(U) と GetForward(U) の計算量は $O(N)$ となる[4].

[2]異なる BDD fragment に同名の変数が現れる場合, それらは i.i.d. であるとする.

[3]*Ord* は各 BDD fragment の変数順序を決定する関数であるのに対して, *Level* は BDD fragment に順序を与える関数である.

[4]疑似コードの GetBackward(U) と GetForward(U) の計算量は $O(N^2)$ である. しかし, あらかじめ δ_U の全節点をトポロジカルソートによって整列しておくことで, これらの計算は $O(N)$ で実行することが可能である.

```
Procedure: GetBackward(U)
  B^1_θ[1] = 1, B^1_θ[0] = 0;
  B^0_θ[1] = 0, B^0_θ[0] = 1;
  N = Par(1) ∪ Par(0);
  // Par(n): the set of parents of n.
  while N ≠ φ do
    n = argmax_{n'∈N} Ord_U(n')
    X = Label(n);
    B^1_θ[n] = θ_[X] B^1_θ[Ch_1(n)] + θ_[X̄] B^1_θ[Ch_0(n)];
    B^0_θ[n] = θ_[X] B^0_θ[Ch_1(n)] + θ_[X̄] B^0_θ[Ch_0(n)];
    N = N\{n} ∪ Par(n);
  end while
end
```

```
Procedure: GetForward(U)
  N = {root}; // F_θ[root] = 1
  while N ≠ φ do
    n = argmin_{n'∈N} Ord_U(n');
    X = Label(n);
    F_θ[Ch_1(n)] += F_θ[n]θ_[X];
    F_θ[Ch_0(n)] += F_θ[n]θ_[X̄];
    N = N\{n} ∪ {Ch_1(n), Ch_0(n)};
  end while
end
```

3.2 内側確率

GetInside(U) は U の**内側確率**を計算する．内側確率 $\mathcal{P}^1_{\boldsymbol{\theta}}[U]$ は中間命題 U が真を取る確率, $\mathcal{P}^0_{\boldsymbol{\theta}}[U]$ は偽を取る確率である (すなわち $\mathcal{P}^1_{\boldsymbol{\theta}}[U] + \mathcal{P}^0_{\boldsymbol{\theta}}[U] = 1$)．ここで U が真となる確率は δ_U の根節点から $\boxed{1}$ に至るすべてのパスの確率の総和に等しいので, $\mathcal{P}^1_{\boldsymbol{\theta}}[U] = \mathcal{B}^1_{\boldsymbol{\theta}}[root]$ である．また, $\mathcal{P}^f_{\boldsymbol{\theta}}[F]$ はパラメタ $\boldsymbol{\theta}$ における $F = f$ の確率, すなわち尤度となる．中間命題 U の値はすべての基本命題の値が決まれば一意に決まるので, U のパラメタは存在しない．しかし, BDD-EM アルゴリズムでは U の内側確率 $\mathcal{P}^1_{\boldsymbol{\theta}}[U], \mathcal{P}^0_{\boldsymbol{\theta}}[U]$ を U のパラメタ $\theta_{[U]}, \theta_{[\bar{U}]}$ のように扱うことにする．

```
Procedure: GetInside(U)
  P^1_θ[U] = B^1_θ[root];
  P^0_θ[U] = 1 − B^1_θ[root];
end
```

3.3 外側確率と条件付き期待値

GetOutsideExpe(U) はすべての $X \in \mathbf{V}(\delta_U) \cap \mathbf{X}$ において $\eta^x_{\boldsymbol{\theta}}[[X]]$ を, すべての $Y \in \mathbf{V}(\delta_U) \cap \mathbf{U}$ において $\mathcal{Q}^y_{\boldsymbol{\theta}}[Y]$ を更新する．$\eta^x_{\boldsymbol{\theta}}[[X]]$ は 2.1 の式 (1) で定義したものであり, $\mathcal{Q}^y_{\boldsymbol{\theta}}[Y]$ は $\eta^x_{\boldsymbol{\theta}}[[X]]$ を計算するための一時的な値で**外側確率**と呼ばれる．ここで $\eta_{\boldsymbol{\theta}}$ と $Q_{\boldsymbol{\theta}}$ は InitializeAll() において, すべての $s \in \mathcal{S}$ に対して $\eta^x_{\boldsymbol{\theta}}[s] = 0$, すべての $Y \in \mathbf{U}$ に対して $Y = F$ のとき $\mathcal{Q}^f_{\boldsymbol{\theta}}[F] = 1, \mathcal{Q}^{\bar{f}}_{\boldsymbol{\theta}}[F] = 0$, $Y \neq F$ のとき $\mathcal{Q}^y_{\boldsymbol{\theta}}[Y] = 0$ と初期化される．ここで $f = \{0, 1\}$ は観測命題 F の観測値である．ここで BDD の削除規則によって削除された節点を記述するために $Del^1_{\delta_U}(n)$ と $Del^0_{\delta_U}(n)$ を導入する．$Del^x_{\delta_U}(n)$ $(x \in \{1, 0\})$ は δ_U 中の節点 n と $Ch_x(n)$ の間で削除されたすべての節点のラベルの集合である．つまり $Del^x_{\delta_U}(n) = \{X \in \mathbf{V}(\delta_U) \mid Label(n) \prec X \prec Label(Ch_x(n))\}$ である．

```
Procedure: GetOutsideExpe(U)
  for each n ∈ N(δ_U) do
    X = Label(n);
    e^1_n = Q^1_θ[U] F_θ[n] B^1_θ[Ch_1(n)] θ_[X]
      + Q^0_θ[U] F_θ[n] B^0_θ[Ch_1(n)] θ_[X];
    e^0_n = Q^1_θ[U] F_θ[n] B^1_θ[Ch_0(n)] θ_[X̄]
      + Q^0_θ[U] F_θ[n] B^0_θ[Ch_0(n)] θ_[X̄];
    if X ∈ U then
      Q^1_θ[X] += e^1_n/θ_[X], Q^0_θ[X] += e^0_n/θ_[X̄];
    else if X ∈ X then
      η^1_θ[[X]] += e^1_n, η^0_θ[[X]] += e^0_n;
    end if
    for each Z ∈ Del^1_U(n) do
      if Z ∈ U then
        Q^1_θ[Z] += e^1_n, Q^0_θ[Z] += e^1_n;
      else if Z ∈ X then
        η^1_θ[[Z]] += e^1_n θ_[Z], η^0_θ[[Z]] += e^1_n θ_[Z̄];
      end if
    end for
    for each Z ∈ Del^0_U(n) do
      if Z ∈ U then
        Q^1_θ[Z] += e^0_n, Q^0_θ[Z] += e^0_n;
      else if Z ∈ X then
        η^1_θ[[Z]] += e^0_n θ_[Z], η^0_θ[[Z]] += e^0_n θ_[Z̄];
      end if
    end for
  end for
end
```

GetOutsideExpe(U) によって計算される $\eta^x_{\boldsymbol{\theta}}[[X]]$ が式 (5) の値 $P_{\boldsymbol{\theta}}(X{=}x, F{=}f)$ と一致することは付録 A で説明する．GetOutsideExpe(U) はすべての $n \in \mathbf{N}(U)$ において $\eta^x_{\boldsymbol{\theta}}[[X]]$ $(X = Label(n))$ を更新し, 更に n のすべての $Y \in Del^x_U(n)$ において $\eta^y_{\boldsymbol{\theta}}[[Y]]$ を更新する．よって GetOutsideExpe(U) の計算量は $O(N(D_1 + D_0))$ となる．ここで $N = |\mathbf{N}(\delta_U)|$, $D_x = \max_{n \in \mathbf{N}(\delta_U)} |Del^x_U(n)|$ である．しかし, GetOutsideExpe(U) を GetOutsideExpe*(U) に置きかえることで計算量を $O(N{+}V)$ に減らすことが可能である．ここで $V = |\mathbf{V}(U)|$ である．GetOutsideExpe*(U) が GetOutsideExpe(U) と等価であることは付録 A で説明する．これより BDD-EM アルゴリズムのループ 1 回分の計算量を考えると, IterateEM() は GetOutsideExpe*(U) を $|\mathbf{U}|$ 回呼び出しているので, その計算量は $O(U_{\text{size}}(N_{\max}{+}V_{\max}))$ となる．ここで $U_{\text{size}} = |\mathbf{U}|$, $N_{\max} = \max_{Y \in \Delta_F} |\mathbf{N}(\delta_Y)|$, $V_{\max} = \max_{Y \in \Delta_F} |\mathbf{V}(\delta_Y)|$ である．

```
1: Procedure: GetOutsideExpe*(U)
2:   for each n ∈ N(δ_U) do
3:     X = Label(n);
4:     e^1_n = Q^1_θ[U]F_θ[n]B^1_θ[Ch_1(n)]θ_[X]
5:       +Q^0_θ[U]F_θ[n]B^0_θ[Ch_1(n)]θ_[X];
6:     e^0_n = Q^1_θ[U]F_θ[n]B^1_θ[Ch_0(n)]θ_[X̄]
7:       +Q^0_θ[U]F_θ[n]B^0_θ[Ch_0(n)]θ_[X̄];
8:     if X ∈ U then
9:       Q^1_θ[X] += e^1_n/θ_[X], Q^0_θ[X] += e^0_n/θ_[X̄];
10:    else if X ∈ X then
11:      η^1_θ[[X]] += e^1_n, η^0_θ[[X]] += e^0_n;
12:    end if
13:    X' : Ord_U(X') = Ord_U(X) + 1
14:    ζ[X'] += e^1_n + e^0_n;
15:    ζ[Label(Ch_1(n))] −= e^1_n;
16:    ζ[Label(Ch_0(n))] −= e^0_n;
17:  end for
18:  V = V(δ_U);
19:  X = argmin_{X'∈V} Ord_U(X');
20:  z = ζ[X], V = V\{X};
21:  while V ≠ φ do
22:    X = argmin_{X'∈V} Ord_U(X');
23:    if X ∈ U then
24:      Q^1_θ[X] += z, Q^0_θ[X] += z;
25:    else if X ∈ X then
26:      η^1_θ[[X]] += zθ_[X], η^0_θ[[X]] += zθ_[X̄];
27:    end if
28:    z += ζ[X], V = V\{X};
29:  end while
30: end
```

4 noisy-OR モデルにおける計算量

一般的に BDD の構築計算量は与えられた論理式のサイズに対して NP-hard であることが知られている [5]. しかしその一方で, BDD をなるべく少ない計算量で構築する技術として, BDD 同士の論理演算により新しい BDD を構築する *Apply* 演算 [2] や, 有効な変数順序決定法 [5, 6] が知られている. 本稿では提案アルゴリズムの評価するために noisy-OR モデルを取り上げ, BDD の構築計算量と BDD-EM アルゴリズムの実行計算量を合わせて評価する.

noisy-OR モデルは複数の原因と 1 つの結果の関係を表現する. 原因を観測命題 F で表し, F が真となる N 個の原因を基本命題 $C_1, C_2, \ldots, C_N$ で表す. 原因と結果が論理的 OR 関係のとき, F と $C_1, C_2, \ldots, C_N$ は論理式 $F \Leftrightarrow C_1 \vee C_2 \vee \cdots \vee C_N$ で表現できる. これに対して noisy-OR 関係は原因 C_i が真にも関わらず確率的に F が偽となる. C_i が F を導くことを妨害する事象を妨害変数 I_i で表現すると, 妨害変数のパラメタは $\theta_{[I_i]} = P(F{=}0 \mid C_i = 1, C_j = 0)$ $(j \neq i)$ と書くことができ, これらをノイズパラメタと呼ぶ. これより N 入力 noisy-OR モデルは妨害変数を用いて以下の論理式で表現することができる.

$$F \Leftrightarrow (C_1 \wedge \bar{I_1}) \vee (C_2 \wedge \bar{I_2}) \vee \ldots (C_N \wedge \bar{I_N}) \qquad (6)$$

図 5 は N 入力 noisy-OR を表現した BDD である. ただし変数順序 Ord_F は $C_i \prec C_j$, $I_i \prec I_j$ $(i < j)$, $C_i \prec I_k$ $(i < k)$ と決めた. 図 5 の BDD は式 (6) から Apply 演算によって構築可能である. Apply$(\delta_X, \delta_Y, \langle \mathrm{op} \rangle)$ は論理関数 X, Y を表現した BDD δ_X, δ_Y から論理関数 $X \langle \mathrm{op} \rangle Y$ を表現する BDD を構築する操作である. 一般に, Apply$(\delta_X, \delta_Y, \langle \mathrm{op} \rangle)$ の計算量は $N_X = |\mathbf{N}(\delta_X)|$, $N_Y = |\mathbf{N}(\delta_Y)|$ とすると $O(N_X N_Y)$ となる. しかし, N 入力 noisy-OR モデルでは変数順序の遅い変数から順に Apply$(\cdot)$ を適用することで, 1 回の Apply$(\cdot)$ の計算量を $O(1)$ に削減することが可能である. これより, 図 (5) の BDD は $O(N)$ で構築可能である. 更に図 (5) からわかるように $|\mathbf{N}(\delta_F)| = |\mathbf{V}(\delta_F)| = 2N$, $|\mathbf{U}| = 1$ なので, BDD-EM アルゴリズムの実行計算量は $O(N)$ である. これより, N 入力 noisy-OR モデルは BDD 構築と BDD-EM アルゴリズムの実行を合わせて $O(N)$ で実行できる.

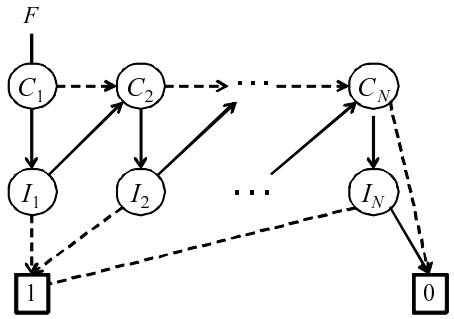

図 5: N 入力 noisy-OR を表現した BDD

5 実験結果

BDD-EM アルゴリズムが正しく動作することを確認するために, 10 入力 noisy-OR のパラメタ推定を行う. 推定するモデルのパラメタはランダムに決定し, モデルから F の値を 100 回観測し, そのうち $F = 1$ が 60 回で $F = 0$ が 40 回であった. ここで 100 個の観測データは i.i.d. であると仮定する.

表 1 の 1 行目は IterateEM() の実行回数, 2 行目と 3 行目はその時点でのパラメタにおける F の事後確率と対数尤度である. この表より, IterateEM() を繰り返すごとに対数尤度が増加していることが確認できる.

6 関連研究

ベイジアンネットワークの確率計算に ZBDD を用いる手法が提案されている [7]. しかし, (Z)BDD に対するパラメタ推定手法は提案されていない. PRISM[8] は

step	$P_{\boldsymbol{\theta}}(F=1)$	対数尤度	増加量
1	0.8214140	-80.71108	
5	0.6181515	-67.37056	1.961360e-01
10	0.6006246	-67.30125	2.327827e-04
15	0.6000213	-67.30117	2.702984e-07
20	0.6000007	-67.30117	3.136284e-10

表 1: 10 入力 noisy-OR の学習結果

Prolog を基にしたモデル記述言語であり, decomposed BDD と似たデータ構造である説明グラフを用いて論理和標準形の論理式を表現する. 現在の PRISM は扱う論理式の各選言子同士が互いに排反であるという排反性条件を仮定しているが, BDD を導入することによって PRISM の排反性条件を取り除くことが可能となる. ProbLog は BDD によって確率計算を行う論理形式である [9]. ProbLog のプログラムは観測式を説明する独立な確率的命題 (基底アトム) からなる DNF 式を BDD に変換し, 確率の和積計算を適用することで観測式の確率を計算する.

広い視野でみると BDD-EM アルゴリズムは, 命題論理式を用いて確率モデルの確率計算や学習を行う命題化確率計算の一例であると考えられる. これの例として BN[7, 10, 11] や Markov random field[12, 13], PRISM[8] が挙げられる.

7 まとめ

本稿は確率的命題変数からなる decomposed BDD 上で EM アルゴリズムを実行する BDD-EM アルゴリズムを提案した. BDD-EM アルゴリズムは汎用の学習アルゴリズムであり, PCFG における Inside-Outside アルゴリズムと Forward-Backward 確率計算の動的計画法により効率的に実行可能である. 本稿では N 入力 noisy-OR モデルにおいてその計算量が BDD 構築を含めて $O(N)$ であることを示し, noisy-OR モデルのパラメタ推定において BDD-EM アルゴリズムが正しく動作していることを確認した. また本稿では省略したが, BDD-EM アルゴリズムを ZBDD に拡張し, decomposed ZBDD と組みわせることで PCFG を扱うことも可能となる. 今後は BDD-EM アルゴリズムを変分ベイズ推定に拡張させることを検討している.

参考文献

[1] S. B. Akers. Binary decision diagrams. *IEEE Trans. on Computers*, 27(6):509–516, 1978.

[2] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Computers*, 35(8):677–691, 1986.

[3] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society B*, 39:1–38, 1977.

[4] J. Jain, A. Narayan, C. Coelho, S.P. Khatri, A. Sangiovanni-Vincentelli, R.K. Brayton, and M. Fujita. Decomposition techniques for efficient ROBDD construction. *Int'l Conf. on Formal Methods in Computer-Aided Design*, 1996.

[5] R. Drechsler and D. Sieling. Binary decision diagrams in theory and practice. *Int'l J. on Software Tools for Technology Transfer*, 3:112–136, 2001.

[6] S. Minato, N. Ishiura, and S. Yajima. Shared binary decision diagram with attributed edges. *Proc. of ACM/IEEE Design Automation Conf.*, pages 52–57, 1990.

[7] S. Minato, K. Satoh, and T. Sato. Compiling Bayesian networks by symbolic probability calculation based on Zero-suppressed BDDs. In *Proc. of IJCAI'07*, pages 2550–2555, 2007.

[8] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *J. of Artificial Intelligence Research*, 15:391–454, 2001.

[9] L. De Raedt, K. Angelika, and H. Toivonen. ProbLog: A probabilistic Prolog and its application in link discoverry. In *Proc. of IJCAI'07*, pages 2468–2473, 2007.

[10] M. Chavira and A. Darwiche. Compiling Bayesian networks with local structure. In *Proc. of IJCAI'05*, pages 1306–1312, 2005.

[11] R. Mateescu and R. Dechter. The relationship between AND/OR search spaces and variable elimination. In *Proc. of UAI'05*, pages 380–387, 2005.

[12] D. McAllester, M. Collins, and F. Pereira. Case-factor diagrams for structured probabilistic modeling. In *Proc. of UAI'04*, pages 382–391, Arlington, Virginia, 2004.

[13] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

付録 A GetOutsideExpe(U) と GetOutsideExpe*(U) の正当性

GetOutsideExpe(U) によって計算される $\eta^x_{\boldsymbol{\theta}}[[X]]$ が式 (5) の値 $P_{\boldsymbol{\theta}}(X{=}x, F{=}f)$ と一致することを簡単に示す．ここでも先ほどと同様，すべての $s \in \mathcal{S}$ において $|s| = 1$ が成り立つ (どの変数もパラメタを共有しない) と考える．まず $U = F$ の場合を考える．$\mathcal{Q}^f_{\boldsymbol{\theta}}[F] = 1, \mathcal{Q}^{\bar{f}}_{\boldsymbol{\theta}}[F] = 0$ より，GetOutsideExpe(F) 中の e^1_n, e^0_n は単純に $e^1_n = \mathcal{F}_{\boldsymbol{\theta}}[n]\mathcal{B}^f_{\boldsymbol{\theta}}[Ch_1(n)]\theta_{[X]}$, $e^0_n = \mathcal{F}_{\boldsymbol{\theta}}[n]\mathcal{B}^f_{\boldsymbol{\theta}}[Ch_0(n)]\theta_{[\bar{X}]}$ と書ける．ここで $X = Label(n)$ である．$\mathcal{F}_{\boldsymbol{\theta}}[n]$ と $\mathcal{B}^f_{\boldsymbol{\theta}}[n]$ の定義より，e^1_n は根節点から節点 n の 1-枝を通って f 定数節点に至るすべてのパスの確率の総和であることがわかる．したがって $\Phi^f_{n,x}$ を $\{\phi \in \Phi \mid F(\phi) = f\}$ の要素のうち，ϕ に対応するパスが節点 n の x-枝を通るものすべての集合とすると，$e^1_n = \sum_{\phi \in \Phi^1_{n,1}} P_{\boldsymbol{\theta}}(\phi)$ と書ける．その一方で，GetOutsideExpe(F) が計算する $\eta^x_{\boldsymbol{\theta}}[[X]]$ の値は以下のように書ける．

$$\begin{aligned}\eta^1_{\boldsymbol{\theta}}[[X]] &= \textstyle\sum_{n \in \mathbf{N}_X(\delta_U)} e^1_n \\ &\quad + \textstyle\sum_{x \in \{1,0\}} \sum_{n \in Del^x_U(X)} e^x_n \theta_{[X]} \qquad (7)\\ \eta^0_{\boldsymbol{\theta}}[[X]] &= \textstyle\sum_{n \in \mathbf{N}_X(\delta_U)} e^0_n \\ &\quad + \textstyle\sum_{x \in \{1,0\}} \sum_{n \in Del^x_U(X)} e^x_n \theta_{[\bar{X}]} \qquad (8)\end{aligned}$$

ここで $\mathbf{N}_X(\delta_U) = \{n \in \mathbf{N}(\delta_U) \mid Label(n) = X\}$, $Del^x_U(X) = \{n \in \mathbf{N}(\delta_U) \mid X \in Del^x_U(n)\}$ である．BDD の削除規則では節点 n の両枝が同じ節点を指しているとき，n を削除する．よって，節点 n' と $Ch_x(n')$ の間に $Label(n) = X$ なる節点 n が削除されているとき，削除を適用する前は n' の x-枝の先に n があり，その両枝が $Ch_x(n')$ を指していたことが分かる．すると n' の x-枝を通るパスは n の 1-枝を通るパスと 0-枝を通るパスを一つにまとめたものであることがわかる．つまり n' の $e^x_{n'}$ は $e^x_{n'} = e^x_{n'}\theta_{[X]} + e^x_{n'}\theta_{[\bar{X}]}$ と二つに分けることができ，$e^x_{n'}\theta_{[X]} = e^x_n$, $e^x_{n'}\theta_{[\bar{X}]} = e^0_n$ であるといえる．これより，式 (7)(8) の第 2 項は削除された節点 n に対しても e^1_n を足し合わせていることがわかる．よって δ^X_U を δ_U の削除された $Label(n) = X$ なる節点をすべて復元した BDD とすると，式 (7)(8) は以下のように書きかえられる．

$$\begin{aligned}\eta^x_{\boldsymbol{\theta}}[[X]] &= \textstyle\sum_{n \in \mathbf{N}_X(\delta^X_U)} e^x_n = \sum_{n \in \mathbf{N}_X(\delta^X_U)} \sum_{\phi \in \Phi^f_{n,x}} P_{\boldsymbol{\theta}}(\phi) \\ &= P_{\boldsymbol{\theta}}(F{=}f, X{=}x) \qquad (9)\end{aligned}$$

これより，GetOutsideExpe(F) は実際には削除された節点を復元することなく，式 (5) と同じ値を計算していることがわかる．$X \in \mathbf{U}$ のときは $\eta^x_{\boldsymbol{\theta}}[[X]]$ の代わりに $\mathcal{Q}^x_{\boldsymbol{\theta}}[X]$ を更新する．ここで $\mathcal{Q}^x_{\boldsymbol{\theta}}[X]$ は $\eta_{\boldsymbol{\theta}}$ の更新式を自身のパラメタで割って更新しているだけなので，$\mathcal{Q}^x_{\boldsymbol{\theta}}[X] = P_{\boldsymbol{\theta}}(F{=}f, X{=}x)/\theta_{[X]}$ であるといえる．よって以下が成り立つ．

$$\mathcal{Q}^x_{\boldsymbol{\theta}}[X] = P_{\boldsymbol{\theta}}(F{=}f \mid X{=}x) \qquad (10)$$

次に GetOutsideExpe(U) ($U \neq F$) について考える．ここで $\mathcal{Q}^u_{\boldsymbol{\theta}}[U]$ はすでに計算されており，式 (10) より $\mathcal{Q}^u_{\boldsymbol{\theta}}[U] = P_{\boldsymbol{\theta}}(F{=}f \mid U{=}u)$ を満たす．また先と同様に $\mathcal{F}_{\boldsymbol{\theta}}[n]$ と $\mathcal{B}^u_{\boldsymbol{\theta}}[X]$ の定義より $\mathcal{F}_{\boldsymbol{\theta}}[n]\mathcal{B}^u_{\boldsymbol{\theta}}[Ch_1(n)]\theta_{[X]} = \sum_{\phi \in \Phi^u_{n,1}} P_{\boldsymbol{\theta}}(\phi)$, $\mathcal{F}_{\boldsymbol{\theta}}[n]\mathcal{B}^u_{\boldsymbol{\theta}}[Ch_0(n)]\theta_{[\bar{X}]} = \sum_{\phi \in \Phi^u_{n,0}} P_{\boldsymbol{\theta}}(\phi)$ と書ける．これより以下が成り立つ．

$$\begin{aligned}e^x_n &= P_{\boldsymbol{\theta}}(F{=}f \mid U{=}1) \textstyle\sum_{\phi \in \Phi^1_{n,1}} P_{\boldsymbol{\theta}}(\phi) \\ &\quad + P_{\boldsymbol{\theta}}(F{=}f \mid U{=}0) \textstyle\sum_{\phi \in \Phi^0_{n,1}} P_{\boldsymbol{\theta}}(\phi) \qquad (11)\end{aligned}$$

更に先と同様に式 (7)(8) に式 (11) を代入して．

$$\begin{aligned}\eta^x_{\boldsymbol{\theta}}[[X]] &= P_{\boldsymbol{\theta}}(F{=}f \mid U{=}1) \textstyle\sum_{n \in \mathbf{N}_X(\delta^X_U)} \sum_{\phi \in \Phi^1_{n,x}} P_{\boldsymbol{\theta}}(\phi) \\ &\quad + P_{\boldsymbol{\theta}}(F{=}f \mid U{=}0) \textstyle\sum_{n \in \mathbf{N}_X(\delta^X_U)} \sum_{\phi \in \Phi^0_{n,x}} P_{\boldsymbol{\theta}}(\phi) \\ &= P_{\boldsymbol{\theta}}(F{=}f \mid U{=}1) P_{\boldsymbol{\theta}}(U{=}1 \mid X = x) \\ &\quad + P_{\boldsymbol{\theta}}(F{=}f \mid U{=}0) P_{\boldsymbol{\theta}}(U{=}0 \mid X = x) \\ &= P_{\boldsymbol{\theta}}(F{=}f, X{=}x) \qquad (12)\end{aligned}$$

を得る．また同様に $X \in \mathbf{U}$ のとき

$$\mathcal{Q}^x_{\boldsymbol{\theta}}[X] = P_{\boldsymbol{\theta}}(F{=}f, X{=}x) \qquad (13)$$

が成り立つことがわかる．式 (9)(12) よりすべての $U \in \mathbf{U}$ において GetOutsideExpe(U) は式 (5) と同じ値を計算することが確認できた．

次に GetOutsideExpe*(U) によって計算される $\eta^x_{\boldsymbol{\theta}}[[X]]$ が GetOutsideExpe(U) によって計算されるそれと等しいことを説明する．GetOutsideExpe(U) ではすべての節点とすべての削除された節点において $\eta_{\boldsymbol{\theta}}$ を更新する．これに対して GetOutsideExpe*(U) ではすべての節点に対して $\eta^x_{\boldsymbol{\theta}}[[X]]$ と $\zeta[X]$ を更新する．ここで $\zeta[X]$ は以下を満たすように定義された値である．

$$\textstyle\sum_{Y \in \mathbf{V}(\delta_U): Y \prec X} \zeta[Y] = \sum_{x \in \{1,0\}} \sum_{n \in Del^x_U(X)} e^x_n$$

つまり $\sum_{Y \in \mathbf{V}(\delta_U): Y \prec X} \zeta[Y]\theta_{[X]}$ は式 (7) の第 2 項と一致する．これより GetOutsideExpe(U) が計算する $\eta^x_{\boldsymbol{\theta}}[[X]]$ と GetOutsideExpe*(U) が計算する $\eta^x_{\boldsymbol{\theta}}[[X]]$ は一致することがわかる．