数理解析研究所講究録
第1616巻 2008年 16-40

確率伝搬法とその応用

林和則（京都大学情報学研究科）

1 はじめに

確率伝搬法（Belief Propagation, BP）と呼ばれる手法が工学や統計物理など様々な分野で注目されている．これは複数の確率変数の結合 (同時) 密度関数から各確率変数の周辺事後分布を効率的に求めるための手法であり，1980年代に人工知能の分野でJ. Pearlによって提案された [1] のが最初とされている．確率伝搬法では，密度関数をベイジアンネットワークやファクターグラフなどのグラフで記述し，そのグラフ上での局所的なメッセージの交換及び処理を行うことで大域的な確率推論を行う．グラフが木構造のときにのみ厳密解が得られるが，ループが存在する場合にも多くの応用例において良好な結果が得られることが経験的に知られている．また，興味深いことは様々な分野でこれまでに提案され用いられている多くの手法が，確率伝搬法と同じ原理で説明されることである．例えば，シャノン限界に迫る符号として注目されているターボ符号や低密度パリティ検査（Low-Density Parity Check, LDPC）符号などの復号法や，隠れマルコフモデルに対する前向き後ろ向き（Bahl-Cocke-Jelinek-Raviv, BCJR）アルゴリズム，カルマンフィルタ，さらには高速フーリエ変換までもが確率伝搬法を用いて説明することが出来る．本稿では，ファクターグラフ上のsum-productアルゴリズム及びベイジアンネットワーク上でのPearlのBPアルゴリズムについて説明することで，確率伝搬法の基本的な考え方およびその原理を解説する．さらに，誤り訂正符号の最大事後確率復号アルゴリズムや高速フーリエ変換（fast Fourier transform, FFT）アルゴリズムなどが確率伝搬法を用いて説明できることを示すことで確率伝搬法の応用例についても紹介する．

2 確率伝搬法

本稿で取り扱う確率伝搬法は効率的に確率推論問題 [4] を解くための手法である．そこでまず，ここで考える確率推論問題の定式化を行ない，分配法則に基づく確率伝搬法の計算原理について説明する．

2.1 確率推論問題と確率伝搬法の原理

$X_1, \ldots, X_N$ をランダム変数とし，その結合（同時）密度関数を

$$p(x_1, \ldots, x_N) \stackrel{def}{=} \Pr\{X_1 = x_1, \ldots, X_N = x_N\} \tag{1}$$

と定義する. このとき本稿で考える確率推論問題は次のように定義される.

確率推論問題: ランダム変数 $X_1, \ldots, X_N$ のうち $X_{m+1}, \ldots, X_N$ の観測値 $a_{m+1}, \ldots, a_N$ が与えられたとき, 非観測値の周辺事後確率

$$\Pr(X_i = a | X_{m+1} = a_{m+1}, \ldots, X_N = a_N),\ i = 1, \ldots, m \tag{2}$$

を計算すること

(2) を直接計算して x_1 を評価しようとすると

$$\Pr(X_1 = a | X_{m+1} = a_{m+1}, \ldots, X_N = a_N) = \alpha \sum_{x_2, \ldots, x_m} p(a, x_2, \ldots, x_m, a_{m+1}, \ldots, a_N)$$

となるが (α は規格化定数), 各 X_i が q 通りの値を取り得る場合およそ q^{m-1} 回の演算が必要となり, m について指数オーダーの計算量となる. このことは, m が大きい問題 (例えば誤り訂正符号の復号問題では数千) では, (2) を直接評価することが困難であることを意味する. そこで結合密度関数を分解することを考える. ベイズ則を繰り返し用いることで, 結合密度関数は一般に

$$\begin{aligned} p(x_1, \ldots, x_N) &= p(x_1) p(x_2, \ldots, x_N | x_1) \\ &= p(x_1) p(x_2 | x_1) p(x_3, \ldots, x_N | x_1, x_2) \\ &\ \ \vdots \\ &= p(x_1) p(x_2 | x_1) p(x_3 | x_1, x_2) \cdots p(x_N | x_1, x_2, \ldots, x_{N-1}) \end{aligned} \tag{3}$$

のように書くことができる. これだけでは当然計算量の削減に繋がらないが, ここで注意すべき重要な点は"多くの工学的な応用において (3) の条件付き分布の条件は必ずしも全てが有効ではない (考慮すべきランダム変数の中に独立なのものも存在する)"ということである. この性質を利用することで, 周辺事後確率計算の演算量を削減することができる.

5つのランダム変数 $x_1, \ldots, x_5$ の結合密度関数を考え, x_2 と x_4, x_3 と x_1 及び x_2, x_5 と x_1 及び x_2 及び x_3 が独立であるとすると

$$\begin{aligned} &p(x_1, x_2, x_3, x_4, x_5) \\ &= p(x_1) p(x_2 | x_1) p(x_4 | x_1, x_2) p(x_3 | x_1, x_2, x_4) p(x_5 | x_1, x_2, x_3, x_4) \\ &= p(x_1) p(x_2 | x_1) p(x_4 | x_1) p(x_3 | x_4) p(x_5 | x_4) \end{aligned} \tag{4}$$

となるため, 確率推論問題の例として $X_3 = a_3$ が観測されたときの x_1 の周辺事後確

率の計算を考えると

$$
\begin{aligned}
&\Pr(X_1 = a | X_3 = a_3) \\
&= \alpha \sum_{x_2,x_4,x_5} p(a, x_2, a_3, x_4, x_5) \\
&= \alpha \sum_{x_2,x_4,x_5} p(x_1 = a) p(x_2|x_1 = a) p(x_3 = a_3|x_4) p(x_4|x_1 = a) p(x_5|x_4) \\
&= \alpha p(x_1 = a) \left(\sum_{x_2} p(x_2|x_1 = a) \right) \left(\sum_{x_4} p(x_3 = a_3|x_4) p(x_4|x_1 = a) \left(\sum_{x_5} p(x_5|x_4) \right) \right)
\end{aligned}
$$

となり, 直接計算した場合には q^3 の加算が必要であったのに対し最後の式を利用するとおよそ q^2 にまで計算量が削減されていることが分かる. ここでの計算量削減の肝は, 分配法則を利用してグローバルな周辺化計算をローカルな周辺化計算にすることであり, これが後述する確率伝搬法のアルゴリズムの計算原理になっている. 分配法則と確率伝搬法についての詳しい議論は [15] を参照されたい.

2.2 ファクターグラフ

本稿で考える確率伝搬法のアルゴリズム（sum-product アルゴリズム）は, ファクターグラフ上のメッセージパッシングアルゴリズムとして記述される. ファクターグラフは多変数関数の因子分解を表す無向グラフであり, 2 種類のノードすなわち "関数ノード" と "変数ノード" からなる. 変数の集合を $X = \{x_1, x_2, \ldots, x_n\}$ とし, 多変数関数が

$$
\begin{aligned}
f(X) &= f(x_1, x_2, \ldots, x_n) \\
&= f_1(A_1) f_2(A_2) \cdots f_m(A_m)
\end{aligned} \tag{5}
$$

のように因子分解されるとする. ただし, $A_1, A_2, \ldots, A_m$ は X の部分集合である. このとき, 関数ノードと変数ノードは, それぞれローカル関数 $f_i(A_i)$ と変数 x_j に対応する. そして, $f_i(A_i)$ に対応する関数ノードと A_i に含まれる変数に対応する変数ノードとがエッジで接続されることによりグラフが構成される（図 1 参照）.

ファクターグラフの具体例としてマルコフ連鎖モデルを考えてみる. マルコフ連鎖の結合密度関数は

$$
p(x_1, x_2, \ldots, x_n) = p(x_1) p(x_2|x_1) p(x_3|x_2) \cdots p(x_n|x_{n-1}) \tag{6}
$$

であるため, これを変数が $x_1, x_2, \ldots, x_n$ の多変数関数とみなし, $p(x_1)$ を $f_1(x_1)$, $p(x_i|x_{i-1})$ を $f_i(x_{i-1}, x_i)$, $i = 2, ..., n$ と対応づけることで結合密度関数は

$$
f(x_1, x_2, \ldots, x_n) = f_1(x_1) f_2(x_1, x_2) f_3(x_2, x_3) \cdots f_n(x_{n-1}, x_n) \tag{7}
$$

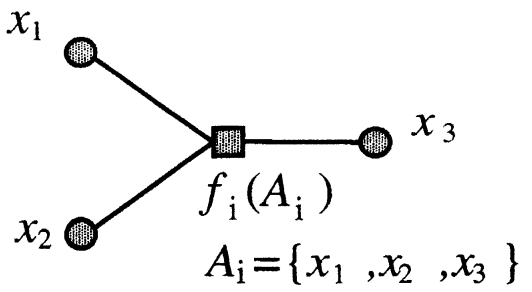


図 1: ファクターグラフの接続ルール

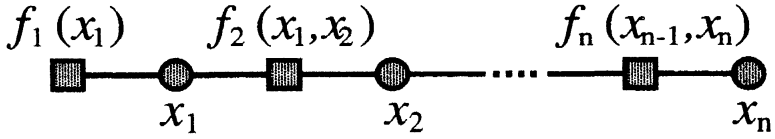


図 2: ファクターグラフの例（マルコフ連鎖）

書ける. これよりマルコフ連鎖のファクターグラフは図 2 のようになる.

同様に考えて, 結合密度関数が

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2|x_1)p(x_3|x_4)p(x_4|x_1)p(x_5|x_4) \tag{8}$$

で与えられる場合は,

$$f(x_1, x_2, x_3, x_4, x_5) = f_1(x_1)f_2(x_1, x_2)f_4(x_3, x_4)f_3(x_1, x_4)f_5(x_4, x_5) \tag{9}$$

ととみなすことで, ファクターグラフは図 3 のようになる.

2.3 sum-product アルゴリズム

sum-product アルゴリズムは**木構造をもつ**ファクターグラフで表現された多変数関数

$$\begin{aligned} f(X) &= f(x_1, x_2, \ldots, x_n) \\ &= f_1(A_1)f_2(A_2)\cdots f_m(A_m) \end{aligned} \tag{10}$$

の周辺化関数

$$g_i(x_i) = \sum_{X \setminus x_i} f(X) \tag{11}$$

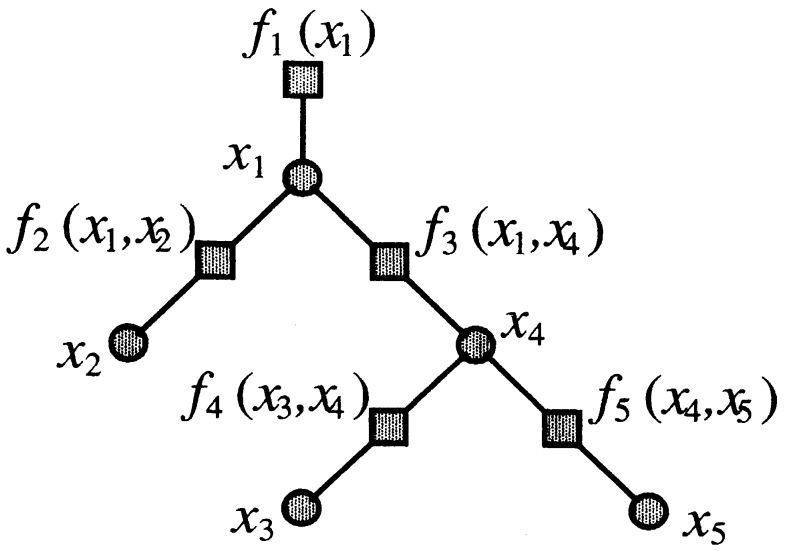


図 3: ファクターグラフの例

をファクターグラフ上で分散的に計算するメッセージパッシングアルゴリズムである. ただし, $X\backslash x_i$ は変数 x_i 以外について周辺化することを意味する. 以下では, ファクターグラフは木構造であるとし, ループが存在する場合については別途考察する. 周辺化関数計算の手順は以下のようである:

1. $f(X) = f(x_1, x_2, \ldots, x_n)$ からファクターグラフを作成する
2. 求めたい周辺化関数を $g_r(x_r)$ とするとき, 変数ノード x_r をルートとする木としてファクターグラフを描き直す
3. 木の下側のノードから上のノードの順に次に説明する各ノードでの処理ルールに従って計算する

変数ノードでの処理：変数ノード x_k から関数ノード f_i へ送るメッセージは次のように計算される（図 4）

$$M_{x_k,f_i}(x_k) = \prod_{a \in N(x_k)\backslash f_i} M_{a,x_k}(x_k) \tag{12}$$

ただし, $N(x_k)$ は変数ノード x_k に接続するノードの集合であり, $M_{a,x_k}(x_k)$ はノード a から x_k に届いたメッセージを表す. また, x_k が葉のときは

$$M_{x_k,f_i}(x_k) = 1 \tag{13}$$

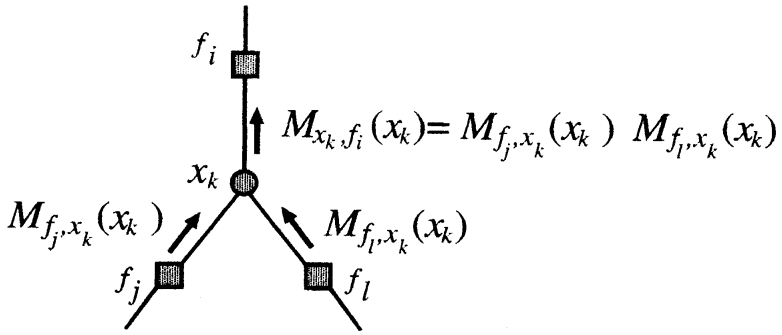


図 4: sum-product アルゴリズム（変数ノードでの処理）

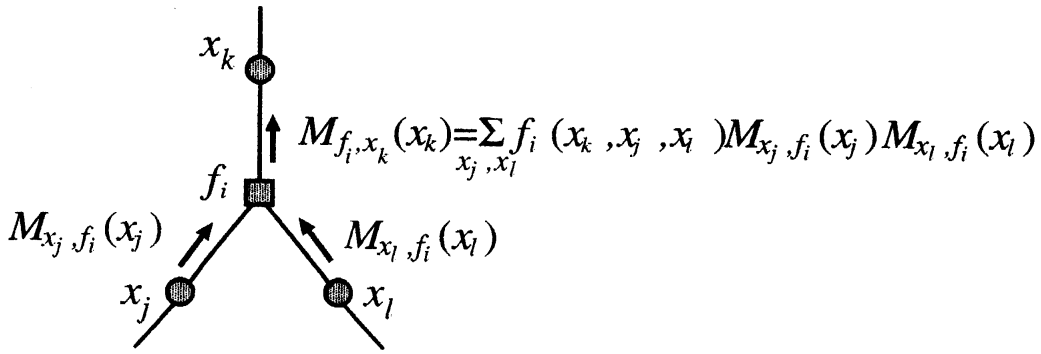


図 5: sum-product アルゴリズム（関数ノードでの処理）

関数ノードでの処理：関数ノード f_i から変数ノード x_k へのメッセージは

$$M_{f_i,x_k}(x_k) = \sum_{A_i \backslash x_k} f_i(A_i) \prod_{A_i \backslash x_k} M_{a,f_i}(a) \tag{14}$$

と計算される (図 5). f_i が葉のときは

$$M_{f_i,x_k}(x_k) = f_i(x_k) \tag{15}$$

ルートノードでの処理：最後にルートノードで

$$M_{x_r}(x_r) = \prod_{a \in N(x_r)} M_{a,x_r}(x_r) = g_r(x_r) \tag{16}$$

と計算することで, 所望の周辺化関数 $g_r(x_r)$ を得る

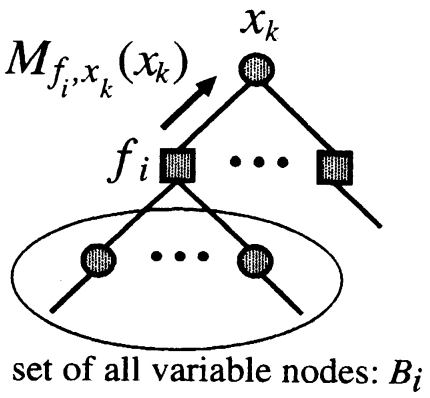


図 6: sum-product アルゴリズムのしくみ 1

ファクターグラフにループが存在しない場合, 上述の sum-product アルゴリズムによる分散的な処理によって厳密に周辺化関数を求めることができる. 以下ではそのしくみについて解説する.

求めたい周辺化関数を

$$g_k(x_k) = \sum_{X \setminus x_k} f(X) \tag{17}$$

とする. 関数ノード f_i を介して変数ノード x_k に繋がっている変数の集合を B_i とすると（図 6 参照）, 多変数関数 $f(X)$ は

$$f(X) = \prod_{i \in N(x_k)} F_i(x_k, B_i) \tag{18}$$

と書けることから

$$\begin{aligned} g_k(x_k) &= \sum_{X \setminus x_k} \prod_{i \in N(x_k)} F_i(x_k, B_i) \\ &= \prod_{i \in N(x_k)} \left[\sum_{B_i} F_i(x_k, B_i) \right] \\ &= \prod_{i \in N(x_k)} M_{f_i,x_k}(x_k) \end{aligned} \tag{19}$$

となる. ただし,

$$M_{f_i,x_k}(x_k) \stackrel{def}{=} \sum_{B_i} F_i(x_k, B_i) \tag{20}$$

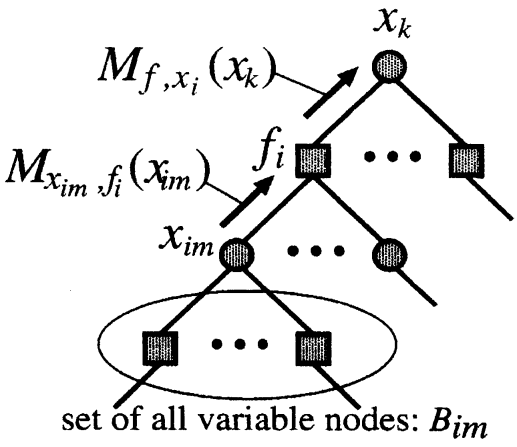


図 7: sum-product アルゴリズムのしくみ 2

であり, $M_{f_i,x_k}(x_k)$ は関数ノード f_i 以下の情報のみから計算される. つまり, $M_{f_i,x_k}(x_k)$ は関数ノード f_i から変数ノード x_k に送られるメッセージであると考えられ, (19) はルートノードでの処理を表していることに注意する.

さらに, 関数ノード f_i に接続する x_k 以外の変数ノードを $x_{i1}, ..., x_{iM}$ とし, x_{im} を介して f_i に繋がっている変数の集合を B_{im} とすると（図 7 参照）

$$F_i(x_k, B_i) = f_i(x_k, x_{i1}, \ldots, x_{iM})G_1(x_{i1}, B_{i1}) \cdots G_M(x_{iM}, B_{iM}) \tag{21}$$

と書くことができる. これより,

$$\begin{aligned} M_{f_i,x_k}(x_k) &= \sum_{B_i} f_i(x_k, x_{i1}, \ldots, x_{iM})G_1(x_{i1}, B_{i1}) \cdots G_M(x_{iM}, B_{iM}) \\ &= \sum_{B_i} f_i(x_k, x_{i1}, \ldots, x_{iM}) \prod_{m \in N(f_i)\backslash x_k} G_m(x_{im}, B_{im}) \\ &= \sum_{x_{i1},\ldots,x_{iM}} f_i(x_k, x_{i1}, \ldots, x_{iM}) \prod_{m \in N(f_i)\backslash x_k} \sum_{B_{im}} G_m(x_{im}, B_{im}) \\ &= \sum_{x_{i1},\ldots,x_{iM}} f_i(x_k, x_{i1}, \ldots, x_{iM}) \prod_{m \in N(f_i)\backslash x_k} M_{x_{im},f_i}(x_{im}) \end{aligned} \tag{22}$$

となる. ここで,

$$M_{x_{im},f_i}(x_{im}) \stackrel{def}{=} \sum_{B_{im}} G_m(x_{im}, B_{im}) \tag{23}$$

であり, これは x_{im} から f_i に送られるメッセージである. (22) は関数ノード f_i での処理を表している. 変数ノード x_{im} に f_j, $(j \neq i)$ を介して接続している変数ノードの集合を B_{imj} とすると（図 8

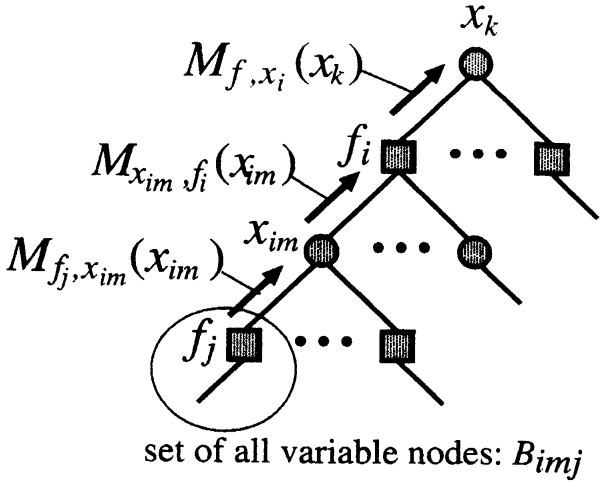


図 8: sum-product アルゴリズムのしくみ 3

$$G_m(x_{im}, B_{im}) = \prod_{j \in N(x_{im}) \setminus f_i} F_j(x_{im}, B_{imj}) \tag{24}$$

と書けるので

$$\begin{aligned} M_{x_{im},f_i}(x_{im}) &= \sum_{B_{im}} \prod_{j \in N(x_{im}) \setminus f_i} F_j(x_{im}, B_{imj}) \\ &= \prod_{j \in N(x_{im}) \setminus f_i} \sum_{B_{imj}} F_j(x_{im}, B_{imj}) \\ &= \prod_{j \in N(x_{im}) \setminus f_i} M_{f_j,x_{im}}(x_{im}) \end{aligned} \tag{25}$$

となる. これは変数ノード x_{im} での処理を表している.

以上より, ファクターグラフにループが存在しない場合には, 上述の変数ノード及び関数ノードにおける分散的な処理によって所望の周辺化関数を計算できることが示された.

2.4 グラフにループがある場合の確率伝搬法

確率伝搬法はファクターグラフにループが無い場合にのみ厳密解を与えるが, ループがある場合にはどのような解が得られるかはこれまでのところ理論的には十分に分かっていない. しかしながら大変興味深いことに, ループが存在する場合においても多くの応用例で良好な結果が得られることが経験的に知られており, ループが存在する場合の確率伝搬法の振る舞いの理論的な解明が実用の面からも求められている. これまでに, 行なわれている Loopy BP に対する収束特性の解析や理論的な説明

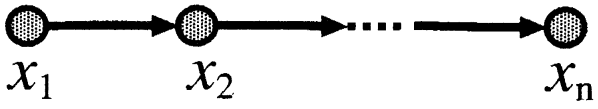


図 9: ベイジアンネットワークの例（マルコフ連鎖）

付けの試みの例としては, 密度発展法 [17], ガウス近似法 [18], EXIT チャート法 [19], 情報幾何に基く解釈 [10, 12], ベーテ自由エネルギーと sum-product アルゴリズムの停留点の関連の指摘 [20] などが挙げられる.

2.5 Pearl の BP アルゴリズム

確率伝搬法のアルゴリズムとして, 2.3 ではファクターグラフ上で定義される sum-product アルゴリズムを説明した. これは sum-product アルゴリズムが簡潔で理解しやすく, また確率密度関数の周辺分布の計算だけでなく多変数関数の周辺化関数の計算というより一般的な目的に使用できるからである. しかしながら, 確率伝搬法の基本的なアイデアは J. Pearl によって提案されたアルゴリズム [1] が最初とされている. そこで本節ではベイジアンネットワーク上でのメッセージパッシングアルゴリズムである Pearl の BP アルゴリズムについて説明し, 確率密度関数の周辺分布の計算に sum-product アルゴリズムを適用したものと等価なアルゴリズムであることを簡単な例を用いて示す.

Pearl の BP アルゴリズムについて説明する前に, ベイジアンネットワークの導入を行なう. 同時確率 $p(x_1, x_2, \ldots, x_n)$ に対応するベイジアンネットワークは次の性質をみたす有向非巡回グラフ（Directed acyclic graph, DAG）, 矢印の向きにたどって同じノードに戻ることがない有向グラフ, である.

1. 確率変数が各ノードに対応
2. 同時分布の因数分解が $p(x_1, x_2, \ldots, x_n) = p(x_1|S_1)p(x_2|S_2)\cdots p(x_n|S_n)$ で与えられるときに, S_i が x_i の親ノードの集合（すなわち, $x_j \in S_i \to x_i$ に対応する有向エッジが存在）

ベイジアンネットワークの具体例として, ファクターグラフのときと同様にマルコフ連鎖 (6) を考えると図 9 のようになる. また, 確率密度関数が (8) で与えられる場合のベイジアンネットワークは図 10 のようである. ベイジアンネットワークではファクターグラフと異なり, 1 種類のノード（変数ノード）のみが存在する.

sum-product アルゴリズムでは変数ノードと関数ノードで処理ルールが異なっていたのに対し, Pearl の BP アルゴリズムでは各ノードが親ノードにメッセージを送る場合と子ノードにメッセージを送る場合とで処理ルールが異なる. 図 11 のように

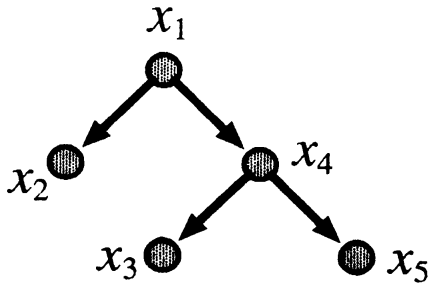


図 10: ベイジアンネットワークの例

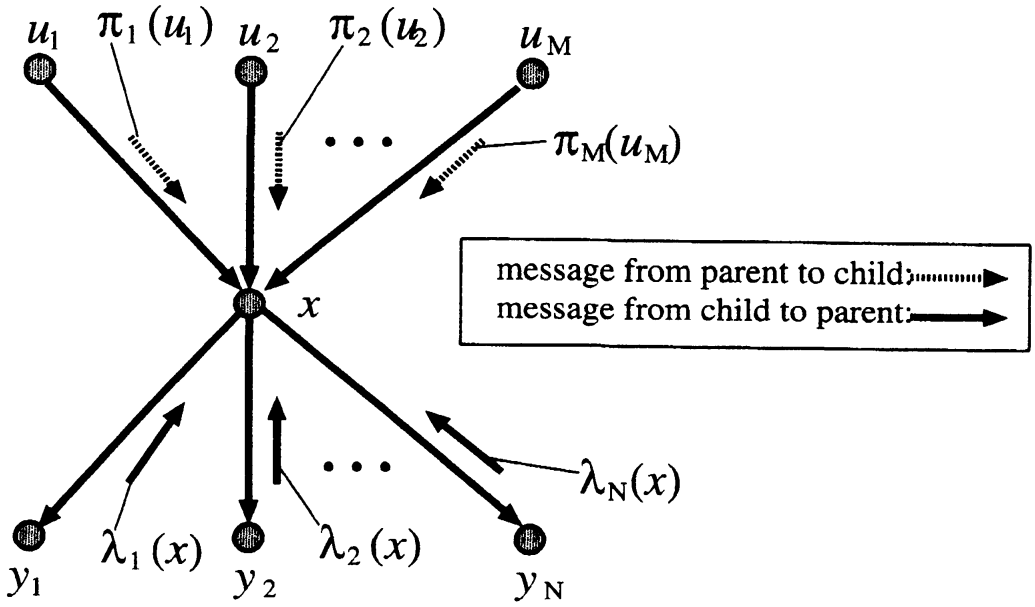


図 11: Pearl の BP アルゴリズム

複数の親 $\mathbf{u} = \{u_1, ..., u_M\}$ 及び子 $\{y_1, ..., y_N\}$ が存在するノード x での Pearl の BP アルゴリズムによる処理ルールを次に示す. ただし, 親 u_i から x に届いているメッセージを $\pi_i(u_i)$ とし, 子 y_j から x に届いているメッセージを $\lambda_j(x)$ とする.

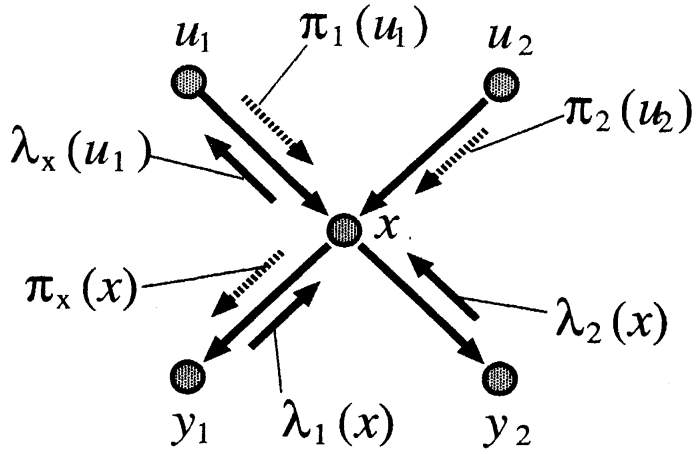


図 12: Pearl の BP アルゴリズムの例

x が親 u_i に送るメッセージ：

$$\lambda_x(u_i) = \sum_{\mathbf{u}\backslash u_1} \left(\sum_x p(x|\mathbf{u}) \prod_{k=1}^{N} \lambda_k(x) \right) \prod_{k=1\backslash i}^{M} \pi_k(u_k) \tag{26}$$

x が子 y_j に送るメッセージ：

$$\pi_x(x) = \left(\prod_{k=1\backslash j}^{N} \lambda_k(x) \right) \sum_{\mathbf{u}} p(x|\mathbf{u}) \prod_{k=1}^{M} \pi_k(u_k) \tag{27}$$

Pearl の BP アルゴリズムと sum-product アルゴリズムの関係を確認するために，図 12 の例でノード x から親ノード u_1 及び子ノード y_1 に送られるメッセージ $\lambda_x(u_1)$, $\pi_x(x)$ を Pearl の BP アルゴリズムの処理ルールに従って計算すると

$$\begin{aligned} \lambda_x(u_1) &= \sum_{u_2} \left(\sum_x p(x|u_1,u_2)\lambda_1(x)\lambda_2(x) \right) \pi_2(u_2) \\ &= \sum_x \lambda_1(x)\lambda_2(x) \sum_{u_2} p(x|u_1,u_2)\pi_2(u_2) \end{aligned} \tag{28}$$

及び

$$\pi_x(x) = \lambda_2(x) \sum_{u_1,u_2} p(x|u_1,u_2)\pi_1(u_1)\pi_2(u_2)$$

となる．一方，図 12 の同時密度関数は

$$p(u_1,u_2,x,y_1,y_2) = p(x|u_1,u_2)p(y_1|x)p(y_2|x)p(u_1)p(u_2) \tag{29}$$

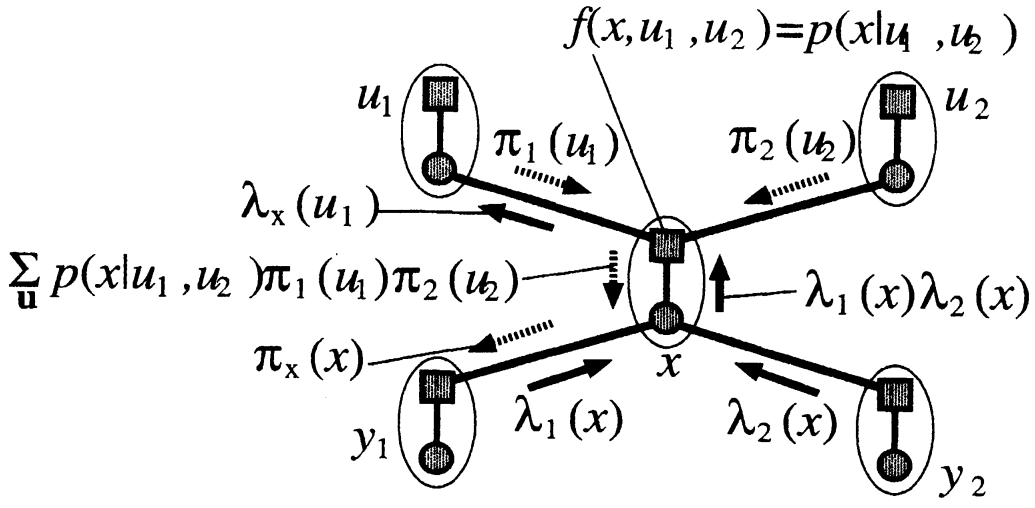


図 13: Pearl の BP アルゴリズムの例（ファクターグラフ表現）

となるのでこれををファクターグラフ表現したものが図 13 である. sum-product アルゴリズムのルールに従うと, 変数ノード x から関数ノード $f(x,u_1,u_2)$ へのメッセージは $f(x,u_1,u_2)$ 以外の関数ノードからきたメッセージの積なので $\lambda_1(x)\lambda_2(x)$ である. 一方, 関数ノード $f(x,u_1,u_2)$ から変数ノード x へのメッセージは x 以外から届いたメッセージと自身の関数を乗算して, x 以外の変数で周辺化したものであるため $\sum_{u_1,u_2} p(x|u_1,u_2)\pi_1(u_1)\pi_2(u_2)$ となる. よって図 13 における $\lambda_x(u_1)$ 及び $\pi_x(x)$ はそれぞれ

$$\begin{aligned}\lambda_x(u_1) &= \sum_{x,u_2} p(x|u_1,u_2)\lambda_1(x)\lambda_2(x)\pi_2(u_2) \\ &= \sum_{x} \lambda_1(x)\lambda_2(x) \sum_{u_2} p(x|u_1,u_2)\pi_2(u_2) \end{aligned} \tag{30}$$

$$\pi_x(x) = \lambda_2(x) \sum_{u_1,u_2} p(x|u_1,u_2)\pi_1(u_1)\pi_2(u_2) \tag{31}$$

となり, Pearl の BP アルゴリズムによるメッセージと一致することが確認できる.

3 確率伝搬法の応用

3.1 低密度パリティ検査（LDPC）符号とその復号アルゴリズム

低密度パリティ検査（LDPC）符号はその復号アルゴリズムである sum-product アルゴリズムとともに 1960 年代初頭に R. G. Gallager によって提案された符号であ

る [6]. しかしながら, その後1990年代に D. J. C. Mackay によって再発見 [14] （この論文の中で, 通信の論文で初めて"belief propagation" という言葉が使われた）されるまで長い間注目されなかった. これは, LDPC 符号が本領を発揮するのは符号長が十分に長い場合であり, そのような符号長の特性を評価する計算機シミュレーションが当時の計算機では困難であったためと考えられる. 現在では, LDPC 符号は符号長や符号化率によってはターボ符号よりも特性が良いことが知られており, 様々なシステムで採用されている.

線形符号の符号語は, 長さ K の情報語を $\mathbf{u} = [u_1 \dots u_K]^T$ としたとき

$$\mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_{K+N} \end{bmatrix}^T = \mathbf{Gu} \tag{32}$$

で生成される. ここで $\mathbf{G}$ は $(K+N) \times K$ の行列であり, 情報語に N 個分の冗長を付加している. $\mathbf{G}$ は生成行列と呼ばれ, $K/(K+N)$ を符号化率という. これに対して検査行列 $\mathbf{H}$ は, $\mathbf{G}$ の列ベクトルが張る空間の直交補空間をその行ベクトルが張るように定義される $N \times (K+N)$ の行列である. 受信語に誤り $\mathbf{e}$ が存在すると

$$\mathbf{c}' = \mathbf{c} + \mathbf{e} \tag{33}$$

となるが, これに検査行列をかけると

$$\mathbf{Hc}' = \mathbf{Hc} + \mathbf{He} = \mathbf{HGu} + \mathbf{He} = \mathbf{He} \tag{34}$$

となり, $\mathbf{Hc}'$ から誤りの有無の検出や訂正ができる.

低密度パリティ検査（LDPC）符号は非零要素の割合が小さい疎な検査行列によって定義される線形符号である. 例えば, 検査行列が

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{35}$$

で与えられるとすると $\mathbf{Hc} = \mathbf{0}$ より, この符号は図14ようなグラフで表現される. これはタナーグラフと呼ばれる2部グラフであり, $\mathbf{H}$ の各列に対応する変数ノードと $\mathbf{H}$ の各行に対応するチェックノードからなる. $\mathbf{H}$ の1に対応するノードがエッジで結ばれるため, タナーグラフの枝の数は $\mathbf{H}$ の1の数に等しい.

受信語 $c_1' \dots c_6'$ と送信符号語 $c_1' \dots c_6'$ を変数ノードとし, $\mathbf{H}$ の各行によるパリティチェックを関数ノードとするとこのLDPC 符号のファクターグラフは図15となる. ここに, 前節で説明したsum-product アルゴリズムを適用することで, LDPC 符号の復号ができる. 正確な事後周辺分布が計算されるのは, ファクターグラフ（あるいはタナーグラフ）にループが存在しない場合のみであるが, そのような符号はループを含む符号に比べて最尤復号特性が劣る [4]. 一方, タナーグラフに短いループが存在する場合には, sum-product アルゴリズムによって良い特性が得られないが, LDPC

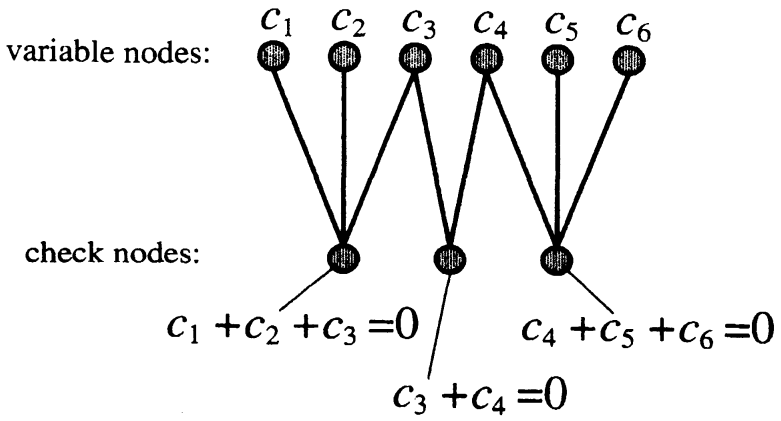


図 14: LDPC 符号の例（タナーグラフ表現）

符号の場合は符号長が十分に長ければそのタナーグラフは短いループをほとんど含まず, sum-product アルゴリズムによって良好な特性が得られることが経験的に知られている [14].

3.2 ターボ符号とその復号アルゴリズム

ターボ符号及びその復号アルゴリズムは, C. Berrou らによって 1993 年に提案され, 現実的な計算量でシャノン限界に迫る誤り率特性が得られることから大変注目を集めた. ターボ符号は並列連接組織符号による符号化とターボ復号と呼ばれる 2 つの復号器による繰り返し処理を特徴とする. その後しばらく, 何故ターボ復号のアルゴリズムで良好な特性が得られるのか理由が分かっていなかったが, 1990 年代後半に R. J. McEliece らによってターボ復号アルゴリズムは確率伝搬法として解釈できることが示された [8].

ここでは, 組織符号とその最大事後確率復号について説明した後, 並列連接組織符号すなわちターボ符号とその最大事後確率復号およびターボ復号アルゴリズムについて Pearl の BP アルゴリズムを用いて説明する.

$\mathbf{u} = (u_1 \ldots u_K)$ をシンボルからなる長さ K の情報語とし, これを符号化することで得られる符号語を $\mathbf{x}$ とする. 組織符号では情報語がそのままの形で符号語に現れるため, その符号語は $\mathbf{x} = (\mathbf{u}, \mathbf{x}_1)$ と書ける. 以下では, $\mathbf{u}$ と $\mathbf{x}_1$ をそれぞれ符号語 $\mathbf{x}$ の組織部, 非組織部と呼ぶ. 符号語 $\mathbf{x}$ が通信路を通って受信されたものを $\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_1)$ とする（図 16 参照）. $\mathbf{y}_s$ 及び $\mathbf{y}_1$ は $\mathbf{x}$ の組織部及び非組織部に対応する受信信号である.

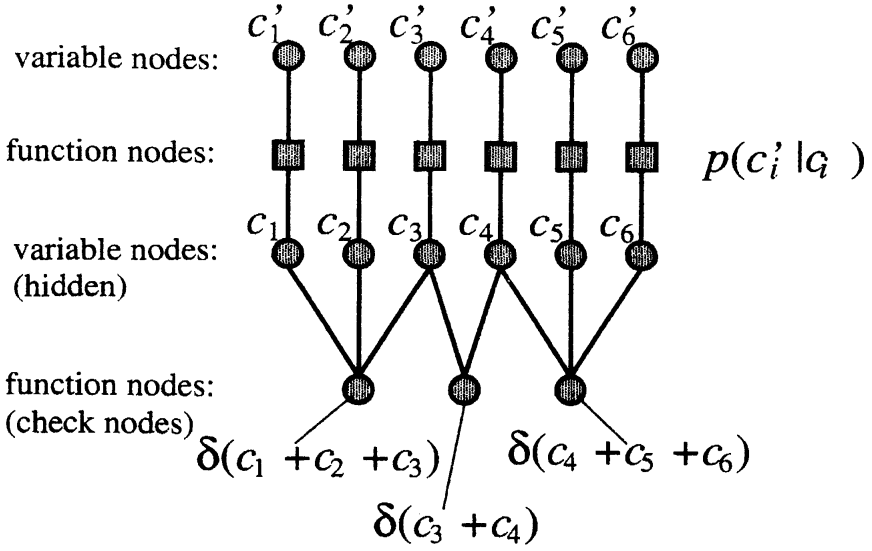


図 15: LDPC 符号の例（ファクターグラフ表現）

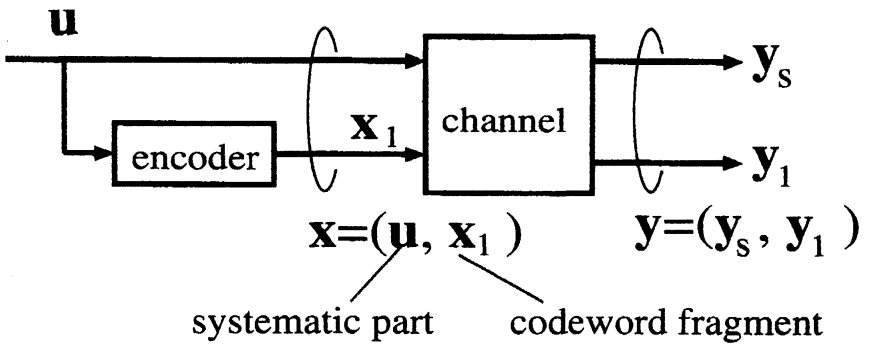


図 16: 組織符号

通信路を無記憶[1]と仮定すると条件付き密度関数は

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= p(\mathbf{y}_s, \mathbf{y}_1|\mathbf{u}, \mathbf{x}_1) \\ &= p(\mathbf{y}_s|\mathbf{u})p(\mathbf{y}_1|\mathbf{x}_1) \\ &= \left(\prod_{i=1}^{K} p(y_{si}|u_i)\right) \cdot p(\mathbf{y}_1|\mathbf{x}_1) \end{aligned} \tag{36}$$

と書ける．ただし，y_{si} は $\mathbf{y}_s$ の i 番目の成分である．

[1]送受信信号をそれぞれ $(x_1 \ldots x_n)$, $(y_1 \ldots y_n)$ としたとき，$i = 1, \ldots, n$ に対して x_i が与えられたときに y_i が $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ 及び $(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n)$ とは独立である場合に，この通信路は無記憶であるといわれる

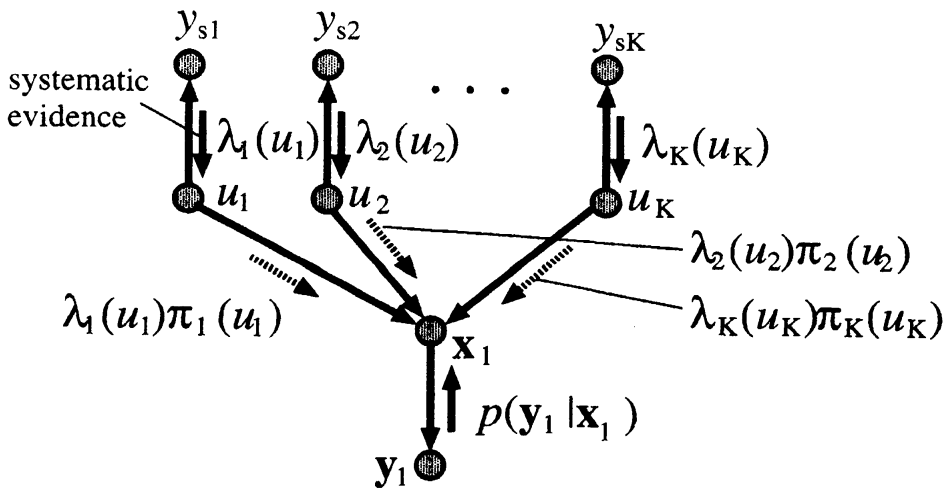


図 17: 組織符号のベイジアンネットワーク表現

$\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_1)$ を受信したときのシンボル毎最大事後確率復号は周辺事後確率

$$\mathrm{BEL}_i(a) \stackrel{def}{=} \Pr\{u_i = a|\mathbf{y}_s, \mathbf{y}_1\} \tag{37}$$

に基づいて行なわれる（$a \in A$ で A は情報シンボルのアルファベット）．周辺事後確率 $\mathrm{BEL}_i(a)$ は，尤度 $p(y_{si}|u_i)$ を $\lambda_i(u_i)$，事前確率 $\Pr\{u_i = a\}$ を $\pi_i(a)$ とし，(36) を用いると

$$\begin{aligned}\mathrm{BEL}_i(a) &= \alpha \sum_{\mathbf{u}:u_i=a} p(\mathbf{y}_1|\mathbf{x}_1) \prod_{j=1}^{k} \lambda_j(u_j)\pi_j(u_j) \\ &= \alpha\lambda_i(a)\pi_i(a) \sum_{\mathbf{u}:u_i=a} p(\mathbf{y}_1|\mathbf{x}_1) \prod_{\substack{j=1 \\ j\neq i}}^{k} \lambda_j(u_j)\pi_j(u_j)\end{aligned} \tag{38}$$

となる．ただし，α は Pearl の α 表記と呼ばれ [1]，確率に（合計が 1 に）なるようにするための規格化定数である．(38) の $\lambda_i(a)$ は組織部分の受信信号から得られる尤度であり systematic evidence と呼ばれ，$\pi_i(a)$ は事前確率，残りは外部値と呼ばれる．後に説明するターボ符号では外部値が極めて重要な役割を果たす．

組織符号の復号問題をベイジアンネットワーク表現すると図 17 のようになる．これに Pearl の BP アルゴリズムを適用することを考える．受信信号の組織部分に対応するノード y_{si} からは systematic evidence $\lambda_i(u_i) = p(y_{si}|u_i)$ が情報語に対応するノード u_i にメッセージとして送られ，同様に受信信号の非組織部 $\mathbf{y}_1$ から $\mathbf{x}_1$ に $p(\mathbf{y}_1|\mathbf{x}_1)$ がメッセージとして送られる．u_i は子ノードのみがあるノードなので，y_{si} から届いたメッセージ $\lambda_i(u_i)$ に自身が持つ事前確率 $\pi_i(u_i)$ を乗算したものをメッセージとし

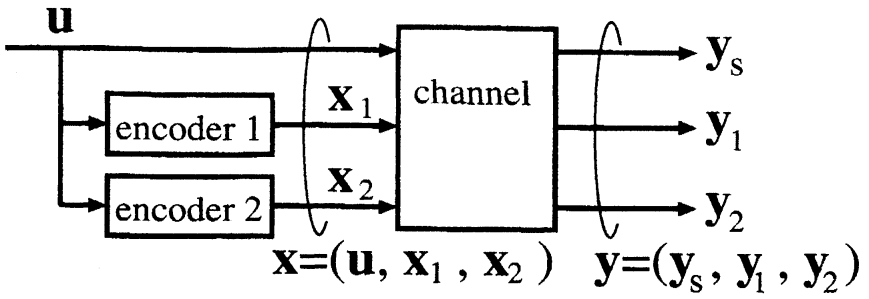


図 18: 並列連接組織符号（ターボ符号）

て $\mathbf{x}_1$ に送る. 次に $\mathbf{x}_1$ が各 u_i に送るメッセージは,

$$\sum_{\mathbf{u}:u_i=a}(p(\mathbf{x}_1|\mathbf{u})p(\mathbf{y}_1|\mathbf{x}_1))\prod_{\substack{j=1\\j\neq i}}^{K}\lambda_j(u_j)\pi_j(u_j)=\sum_{\mathbf{u}:u_i=a}p(\mathbf{y}_1|\mathbf{x}_1)\prod_{\substack{j=1\\j\neq i}}^{k}\lambda_j(u_j)\pi_j(u_j) \tag{39}$$

となる. ここで, $\mathbf{u}$ と $\mathbf{x}_1$ は確定的な関係であることに注意する. 最後にノード u_i では, 届いた全てのメッセージと自身の持つ事前確率を乗算し正規化することで

$$\alpha\lambda_i(a)\pi_i(a)\sum_{\mathbf{u}:u_i=a}p(\mathbf{y}_1|\mathbf{x}_1)\prod_{\substack{j=1\\j\neq i}}^{k}\lambda_j(u_j)\pi_j(u_j) \tag{40}$$

を得る. これは所望の事後周辺分布 (38) に一致している.

次に, 図 18 の並列連接組織符号（ターボ符号）とその最大事後確率復号について考える. $\mathbf{x}_1$, $\mathbf{x}_2$ は 2 つの符号器の出力であり, 符号語 $\mathbf{x}=(\mathbf{x}_s,\mathbf{x}_1,\mathbf{x}_2)$ の非組織部分である. また, $\mathbf{y}_1$, $\mathbf{y}_2$ は $\mathbf{x}_1$, $\mathbf{x}_2$ に対応する受信信号であるとする. 無記憶通信路を仮定することで

$$\begin{aligned}p(\mathbf{y}|\mathbf{x})&=p(\mathbf{y}_s,\mathbf{y}_1,\mathbf{y}_2|\mathbf{u},\mathbf{x}_1,\mathbf{x}_2)\\&=p(\mathbf{y}_s|\mathbf{u})p(\mathbf{y}_1|\mathbf{x}_1)p(\mathbf{y}_2|\mathbf{x}_2)\\&=\left(\prod_{i=1}^{K}p(y_{si}|u_i)\right)p(\mathbf{y}_1|\mathbf{x}_1)p(\mathbf{y}_2|\mathbf{x}_2)\end{aligned} \tag{41}$$

と書ける. さらに, 組織符号のときと同様に $\lambda_i(u_i)$, $\pi_i(u_i)$ を定義するとシンボル毎最大事後確率復号をするための事後周辺確率は

$$\begin{aligned}\mathrm{BEL}_i(a)&=\alpha\sum_{\mathbf{u}:u_i=a}p(\mathbf{y}_1|\mathbf{x}_1)p(\mathbf{y}_2|\mathbf{x}_2)\prod_{j=1}^{k}\lambda_j(u_j)\pi_j(u_j)\\&=\alpha\lambda_i(a)\pi_i(a)\sum_{\mathbf{u}:u_i=a}p(\mathbf{y}_1|\mathbf{x}_1)p(\mathbf{y}_2|\mathbf{x}_2)\prod_{\substack{j=1\\j\neq i}}^{k}\lambda_j(u_j)\pi_j(u_j)\end{aligned} \tag{42}$$

となる.

ターボ復号では事後周辺分布（の近似）を得るために, 符号語 $(\mathbf{x}_s, \mathbf{x}_1)$ 及び $(\mathbf{x}_s, \mathbf{x}_2)$ に対応する最大事後確率復号を交互に何度も行ない, 復号結果（正確には外部値）を次のステップでの事前確率に取り込むという形で互いの情報を交換するアルゴリズムを用いる. 具体的には, 奇数番目のステップすなわち2m-1番目のステップでは, 事前確率を $p(u_i) = \pi_i^{(2m-2)}(u_i)$ として以下の周辺事後確率に基づく符号語 $(\mathbf{x}_s, \mathbf{x}_1)$ の最大事後確率復号を行なう

$$\begin{aligned}\Pr\{u_i = a|\mathbf{y}_s, \mathbf{y}_1\} &= \alpha \sum_{\mathbf{u}:u_i=a} p(\mathbf{y}_1|\mathbf{x}_1) \prod_{j=1}^{k} \lambda_j(u_j)\pi_j^{(2m-2)}(u_j) \\ &= \alpha\lambda_i(a)\pi_i^{(2m-2)}(a) \sum_{\mathbf{u}:u_i=a} p(\mathbf{y}_1|\mathbf{x}_1) \prod_{\substack{j=1 \\ j\neq i}}^{k} \lambda_j(u_j)\pi_j^{(2m-2)}(u_j) \qquad (43)\end{aligned}$$

ここで $\pi_i^{(2m-2)}(u_i)$ は2m-2番目の $(\mathbf{x}_s, \mathbf{x}_2)$ の復号結果から得られた外部値であり, m=1の場合には通常一様分布が用いられる. (43) の外部値（(43) 右辺の $\alpha\lambda_i(a)\pi_i^{(2m-2)}(a)$）以外の部分）を $\pi_i^{(2m-1)}(u_i)$ として次のステップに受け渡す. 2m 番目のステップでは $p(u_i) = \pi_i^{(2m-1)}(u_i)$ として $(\mathbf{x}_s, \mathbf{x}_2)$ の周辺事後確率

$$\Pr\{U_i = a|\mathbf{Y}_s, \mathbf{Y}_2\} = \alpha\lambda_i(a)\pi_i^{(m)}(a) \sum_{\mathbf{u}:u_i=a} p(\mathbf{y}_2|\mathbf{x}_2) \prod_{\substack{j=1 \\ j\neq i}}^{k} \lambda_j(u_j)\pi_j^{(m)}(u_j) \qquad (44)$$

を計算し, その外部値を次のステップでの事前確率 $p(u_i) = \pi_i^{(2m)}(u_i)$ として受け渡す. これをある終了条件を満たすまで繰り返し, 最後に得られている周辺事後確率から復号結果を出力する.

このようにターボ復号は符号語 $(\mathbf{x}_s, \mathbf{x}_1)$ 及び $(\mathbf{x}_s, \mathbf{x}_2)$ の復号を交互に行なうことで外部値を更新していくアルゴリズムであるが, 復号結果ではなく外部値を次段の事前値とする理由が明らかではない. そこで, ターボ符号をベイジアンネットワーク表現してみると図 19 のようになる. このベイジアンネットワークに Pearl の BP アルゴリズムを適用することを考える. 組織符号の場合と異なりネットワークにループが存在するため, BP アルゴリズムによって真の事後周辺分布は得られないことに注意されたい. また, オリジナルの手順ではメッセージを送信するエッジ以外の全てのエッジからのメッセージが届いて初めてメッセージの計算及び送信ができるが, ループが存在する場合にはそのようなルールではアルゴリズムが進まない箇所が発生するためメッセージの人為的な初期化が必要となる. また, ループが存在する場合にはメッセージを処理, 伝搬する順序によって異なる結果が得られるため, BP アルゴリズムを適用するためにはそのスケジューリングを決定する必要がある. そこで, まず次のような初期化を行なう. ノード $\mathbf{x}_1$ からノード u_i に届くメッセージを $\lambda_{x_1,u_i(u_i)} = 1$, ノード $\mathbf{x}_2$ からノード u_i に届くメッセージを $\lambda_{x_2,u_i(u_i)} = 1$ とする. 一

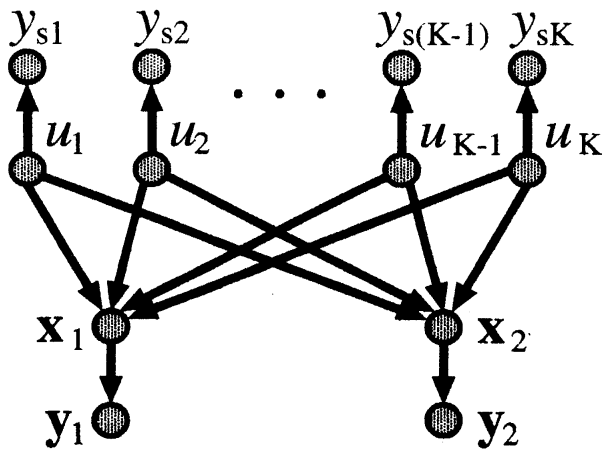


図 19: ターボ符号のベイジアンネットワーク表現

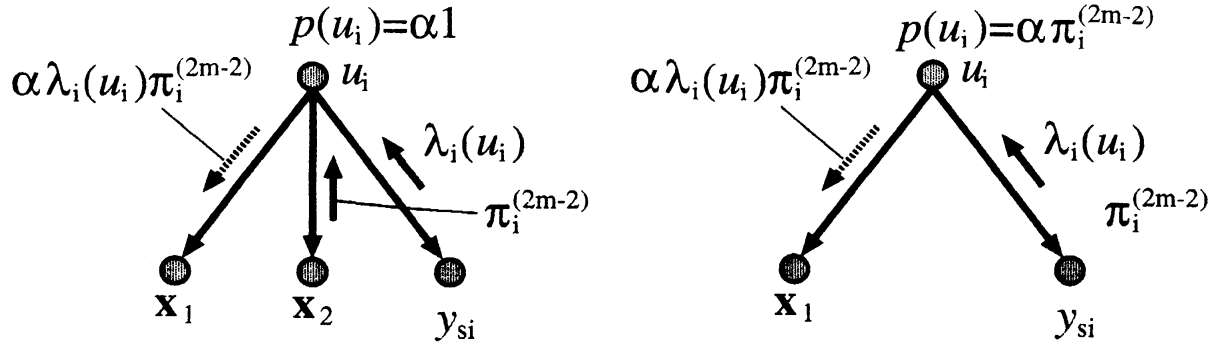
方, ノード y_{si} 及び $\mathbf{y_1}$, $\mathbf{y_1}$ は葉のノードなので繰り返し処理に関係なく常に固定のメッセージ $\lambda_i(u_i)$, $p(\mathbf{y}_1|\mathbf{x}_1)$ 及び $p(\mathbf{y}_2|\mathbf{x}_2)$ をそれぞれ送る. また, ノード u_i が持つ事前確率 $\pi_i(u_i)$ も固定で一様分布とする. 次にノード $u_1 \cdots u_K$ でメッセージを計算し $\mathbf{x}_1$ 及び $\mathbf{x}_2$ に送信する (ノード $\mathbf{u}$ のアクティベーションという). ここで, u_i から $\mathbf{x}_1$ 及び $\mathbf{x}_2$ へのメッセージはいずれも $\alpha\lambda_i(u_i)$ である. この時点でノード $\mathbf{x}_1$ と $\mathbf{x}_2$ は全てのエッジからメッセージが届いているのでどちらもメッセージの更新が可能であるが, まず $\mathbf{x}_1$ のアクティベーションを行なうことにすると, $\mathbf{x}_1$ から u_i へのメッセージは

$$\alpha \sum_{\mathbf{u}:u_i} \left(\sum_{\mathbf{x}_1} p(\mathbf{x}_1|\mathbf{u}) p(\mathbf{y}_1|\mathbf{x}_1) \right) \prod_{\substack{j=1 \\ j\neq i}}^{k} \lambda_j(u_j) = \alpha \sum_{\mathbf{u}:u_i} p(\mathbf{y}_1|\mathbf{x}_1) \prod_{\substack{j=1 \\ j\neq i}}^{k} \lambda_j(u_j) \qquad (45)$$

と計算される. これを $\pi_i^{(1)}$ とする (実際, ターボ復号アルゴリズムの外部値 $\pi_i^{(1)}$ に一致している). これより, ノード $\mathbf{u}$ に届くメッセージの一部が更新されたので次は $\mathbf{u}$ のアクティベーションを行なう. $\mathbf{x}_1$ から届くメッセージのみが更新されているので, 各ノード u_i が更新すべきメッセージは $\mathbf{x}_2$ 宛のものだけであり, $\alpha\lambda_i(u_i)\pi_i^{(1)}$ となる. 続いて, $\mathbf{x}_2$ のアクティベーションを行なうと, $\mathbf{x}_2$ から u_i へのメッセージは

$$\alpha \sum_{\mathbf{u}:u_i} \left(\sum_{\mathbf{x}_2} p(\mathbf{x}_2|\mathbf{u}) p(\mathbf{y}_2|\mathbf{x}_2) \right) \prod_{\substack{j=1 \\ j\neq i}}^{k} \lambda_j(u_j)\pi_j^{(1)} = \alpha \sum_{\mathbf{u}:u_i} p(\mathbf{y}_2|\mathbf{x}_2) \prod_{\substack{j=1 \\ j\neq i}}^{k} \lambda_j(u_j)\pi_j^{(1)} \qquad (46)$$

と計算され, これは外部値 $\pi_i^{(2)}$ に一致していることが分かる. 以下同様に, ノードのアクティベーションを $\mathbf{u} \to \mathbf{x}_1 \to \mathbf{u} \to \mathbf{x}_2 \to \cdots$ の順に繰り返していくと上記の初期化及び順序による Pearl の BP アルゴリズムはターボ復号のアルゴリズムそのものになっている.



(a) original bayesian network　　(b) equivalent bayesian network

図 20: 外部値を次段の事前値とする理由

さて, 改めて外部値を次段の事前値とする理由について考えてみる. ターボ復号ではそれぞれの要素符号の復号を交互に行うため, それぞれの復号アルゴリズムからはベイジアンネットワークが図 19 ではなく図 19 のように見えている. このときどうすればもう 1 つの符号語の持つ情報をうまく取り込めるだろうか？図 20 にこれを説明するためのベイジアンネットワークを示す. 図中左側はターボ符号の元のベイジアンネットワークの u_i に関わる箇所を抽出したものである. 一方, ターボ復号における要素符号 $(\mathbf{x}_s, \mathbf{x}_1)$ の復号の際には $\mathbf{x}_2$ 見えておらず, ネットワークは図 20 の右側のようになっている. その際, 両方のネットワークでノード $\mathbf{x}_1$ に届けられるメッセージを同じにすることを考えると, u_i は子ノードのみを持つノードであることから $\mathbf{x}_1$ に送るメッセージはそれ以外のノードからのメッセージと事前確率 $p(u_i)$ の積となるため, 右の $\mathbf{x}_2$ のノードが省略されたネットワークでは $\mathbf{x}_2$ から届くはずであったメッセージ $\pi_i^{(2m-2)}$ を事前確率 $p(u_i)$ に含んでしまえば良いことが分かる. これは外部値を次段の事前確率にしていることに他ならない. このようにターボ符号および復号アルゴリズムは Pearl の BP アルゴリズムを用いて解釈することが可能であり, これによりその仕組みを見通し良く理解することができる.

3.3 高速フーリエ変換（FFT）

最後に確率伝搬法として解釈される応用例として高速フーリエ変換（FFT）を取り上げる. FFT は離散フーリエ変換（discrete Fourier transform, DFT）

$$W_k = \sum_{n=0}^{N-1} w(n) e^{-j\frac{2\pi}{N}nk} \tag{47}$$

を効率的に演算するためのアルゴリズムである. その基本的なアイデアは (47) の直接的な計算では N^2 のオーダーの計算量が必要であるが

$$\begin{aligned} W_k &= \sum_{n=0}^{\frac{N}{2}-1} w(2n)e^{-j\frac{2\pi}{N}2nk} + \sum_{n=0}^{\frac{N}{2}-1} w(2n+1)e^{-j\frac{2\pi}{N}(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} w(2n)e^{-j\frac{2\pi}{\frac{N}{2}}nk} + e^{-j\frac{2\pi}{N}k}\sum_{n=0}^{\frac{N}{2}-1} w(2n+1)e^{-j\frac{2\pi}{\frac{N}{2}}nk} \end{aligned} \tag{48}$$

と変形することで N 点の DFT 計算を 2 つの $N/2$ 点 DFT 計算で表現し, さらにそれぞれの $N/2$ 点 DFT の離散周波数領域における周期性（周期 $N/2$）を利用することで, 結果的に演算量が $(N/2)^2 \times 2 = N^2/2$ のオーダーに削減できるというものである. N が 2 の整数巾乗のときこれを 2 点 DFT になるまで繰り返していくことで最終的に $N \log N$ の演算量にまで削減される.

J. W. Cooley と J. W. Tukey の 1960 年代の発明 [16] とされているこの有名な FFT アルゴリズムもまたファクターグラフを用いた確率伝搬法として解釈できる [15], [13]. FFT アルゴリズムのファクターグラフ表現をするために 8 点 DFT の場合を考える. 離散時間関数 $w(n)$ の 8 点 DFT は n 及び k を 2 進数表現 $n = 4x_2+2x_1+x_0$, $k = 4y_2 + 2y_1 + y_0$ することで

$$\begin{aligned} W_k &= \sum_{n=0}^{7} w(n)e^{-j\frac{2\pi}{8}nk} \\ &= \sum_{x_0,x_1,x_2} w(4x_2+2x_1+x_0)e^{-j\frac{2\pi}{8}(4x_2+2x_1+x_0)(4y_2+2y_1+y_0)} \\ &= \sum_{x_0,x_1,x_2} w(4x_2+2x_1+x_0)(-1)^{x_2y_0}(-1)^{x_1y_1}(-1)^{x_0y_2}(j)^{-x_0y_1}(j)^{-x_1y_0}e^{-j\frac{2\pi}{8}(x_0y_0)} \\ &= \sum_{x_0}(-1)^{x_0y_2}(j)^{-x_0y_1}e^{-j\frac{2\pi}{8}x_0y_0}\sum_{x_1}(-1)^{x_1y_1}(j)^{-x_1y_0}\sum_{x_2} w(4x_2+2x_1+x_0)(-1)^{x_2y_0} \end{aligned} \tag{49}$$

と変形することができる. これをそのままファクターグラフ表現すると となるがこれにはループが含まれているため, sum-product アルゴリズムをそのまま適用しても厳密な周辺化関数は求めることができない. そこで, 図 21 からスパニング木を構成し,

$$\begin{aligned} a(x_2, y_0) &= (-1)^{x_2y_0} \\ b(x_1, y_0, y_1) &= (-1)^{x_1y_1}(-j)^{x_1y_0} \\ c(x_0, y_0, y_1, y_2) &= (-1)^{x_0y_2}(-j)^{x_0y_1}e^{j\frac{2\pi}{8}x_0y_0} \end{aligned}$$

と関数を定義（クラスタリング）することで, 図 22 のファクターグラフを得る (ファクターグラフのスパニング木の詳細については [13] などを参照). 変数ノード中の

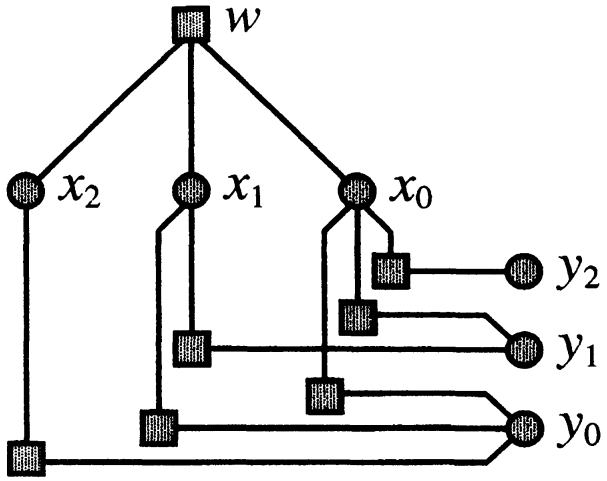


図 21: DFT のファクターグラフ表現

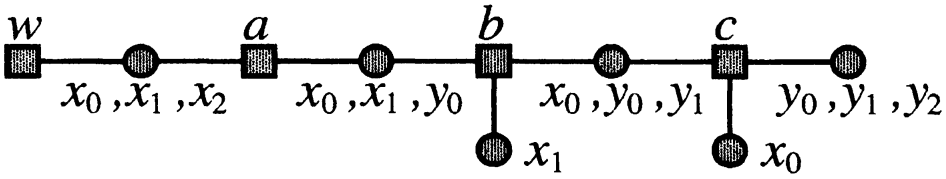


図 22: DFT のファクターグラフ表現（修正版）

x_0, x_1 は冗長であるが, 元のグラフ中での繋がりを明示するために追加してあることに注意する. これに sum-product アルゴリズムを適用したものは FFT のアルゴリズムに等しいことが確認できる. 実際, 図の左側から計算を進めると x_2, x_1, x_0 の順に 3 段階で周辺化が行なわれ $w(n)$ から W_k が得られる.

4 まとめ

本稿では近年様々な分野で注目されている確率伝搬法についてその基本的な考え方及びその原理を説明し, 具体的なアルゴリズムや応用例に対する確率伝搬法に基づく解釈を示した. J. Pearl によるベイジアンネットワーク上の BP アルゴリズムがオリジナルの確率伝搬法であると思われるが, その処理ルールの簡単さと理解のし易さから主にファクターグラフ上の sum-product アルゴリズムを確率伝搬法として詳細に解説し, Pearl の BP アルゴリズムは簡単な例を用いて sum-product アルゴリズムと等価であることを示した. また確率伝搬法の応用例, より正確には確率伝搬法として解釈が可能な応用例, に LDPC 符号とターボ符号の復号アルゴリズム及び FFT アルゴリズムを取り上げ, sum-product アルゴリズムあるいは Pearl の BP アル

ゴリズムによって記述することが可能であることを確認した.

確率伝搬法の原理は一言でいうと分配法則に尽き非常に単純なアルゴリズムであるが, 理論上も応用の面からも大変興味深いのはループが存在するグラフでの確率伝搬法の振る舞いであり, このような場合には理論的に未解決な問題が多く存在する. 確率伝搬法は既に述べたように複数の分野における重要なアルゴリズムの基礎をなす原理に基づいており, その振る舞いを理論的に解明することは学術的にも応用上も非常に大きなインパクトをもつことは間違いない.

参考文献

[1] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann Publishers Inc., 1988.

[2] D. J. C. MacKay, Information Theory, Inference, and Learning Algorithms, Cambridge University Press, 2003.

[3] C. M. Bishop, Pattern Recognition and Machine Leaning, Springer, 2006

[4] 和田山正, 低密度パリティ検査符号とその復号法, トリケップス, 2002.

[5] 荻原春生, ターボ符号の基礎, トリケップス, 1999.

[6] R. G. Gallager, Low-Density Parity-Check Codes, MIT Press, 1963, (http://web.mit.edu/gallager/www/pages/ldpc.pdf)

[7] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding," *IEEE International Conference on Communications' 93*, pp. 1064-1070, 1993.

[8] R. J. McEliece, D. J. C. MacKay and J. Cheng, "Turbo Decoding as an Instance of Pearl's Belief Propagation Algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 140-152, Feb. 1998.

[9] 井坂元彦, "LDPC符号・ターボ符号と反復復号法," 応用数理, vol. 14, no. 3, pp.12-23, Sept. 2004.

[10] 池田思朗, 田中利幸, 甘利俊一, "確率伝搬法の情報幾何-符号理論, 統計物理, 人工知能の接点-," 応用数理, vol. 14, no. 3, pp. 24-35, Sept. 2004.

[11] 田中和之, 樺島祥介, 西森秀稔, "ベイズ統計と統計力学を用いた確率的情報処理技術講義ノート," 若手研究者・学生向けに最新技術をわかりやすく紹介する講演会, Mar. 2002.

[12] 池田思朗，田中利幸，岡田真人，“「確率的情報処理としての移動体通信技術」講義ノート，”若手研究者・学生向けに最新技術をわかりやすく紹介する講演会, Dec. 2002.

[13] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, “ Factor Graphs and the Sum-Product Algorithm,” *IEEE Trans. on Information Theory*, vol. 47, no. 2, pp.498-519, Feb. 2001.

[14] D. J. C. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Trans. on Information Theory*, vol. 45, no. 2, pp.399-431, March, 1999.

[15] S. M. Aji and R. J. McEliece, “The generalized distributive law,” *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp.325-343, March, 2000.

[16] J. W. Cooley and J. W. Tukey, ”An algorithm for the machine calculation of complex Fourier series,” *Mathematics of Computation*, vol. 19, no. 90, pp. 297-301, Apr. 1965.

[17] T. J. Richardson and R. L. Urbanke, “The capacity of low-density parity check codes under message-passing decoding,” *IEEE Trans. on Information Theory*, vol.47, pp. 599-618, 2001

[18] S. Y. Chung, T. J. Richardson and R. L. Urbanke, “Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation,” *IEEE Trans. on Information Theory*, vol.47, pp. 657-686, 2001

[19] S. Brink, “Convergence behavior of iteratively decoded parallel concatenated codes,” *IEEE Trans. on Communications*, vol. 48, pp. 1727-1737, 2001.

[20] J. S. Yedidia, W. T. Freeman and Y. Weiss, “Bethe free energy, Kikuchi approximations, and belief propagation algorithms,” (http://www.merl.com/papers/TR2001-16).