通信のための確率推論の基礎

林　和則（大阪市立大学）

2018年9月26日(水) – 28日(金)

ハトヤホテル（静岡県伊東市）

主催

電子情報通信学会　通信ソサイエティ

革新的無線通信技術に関する横断型研究会

MIKA (Multiple Innovative Kenkyu-kai Association for Wireless Communications)

林　和則(大阪市立大学)

1997年 3月	大阪大学　工学部通信工学科　卒業
1999年 3月	大阪大学　大学院工学研究科　博士前期課程　修了
2002年 3月	大阪大学　大学院工学研究科　博士後期課程　修了
2002年 5月	京都大学　大学院情報学研究科 助手
2007年 4月	京都大学　大学院情報学研究科 助教
2009年 4月	京都大学　大学院情報学研究科 准教授
2017年 4月	大阪市立大学　大学院工学研究科　教授
	現在に至る

目 次

第1章　はじめに

工学の多くの問題において，興味がある知りたい情報は直接観測することができず，それと関連がある（と期待される）観測可能なデータから推定する必要がある．通信の問題はその典型例であり，興味のある知りたい情報，すなわち送信信号は一般に直接観測できず，それが通信路を通ることで歪んだり雑音が付加されたりした受信信号のみが観測可能であり，受信信号から送信信号が何であったかを考えることが通信における重要な問題となる．このような問題に対処するための強力な理論的な枠組みが，ベイズの定理に基づく確率推論である．そこでは，観測可能な確率変数の実現値が与えられた下での，興味のある未知確率変数のそれぞれの条件付き確率分布（周辺事後分布）を求めることが主要な課題であり，周辺事後分布が求まればそれに基づいて様々な推定を行うことが可能になる．例えば，通信における信号検出の問題では，多くの場合ビット誤り率を最小にする信号検出法が望ましいが，そのような信号検出は周辺事後分布の最大値に対応する確率変数の値を推定値として採用する最大事後確率推定によって実現される．従来の通信システムにおける信号検出では計算量の制限などから主に線形の信号処理手法が採用されてきたが，誤り訂正符号の復号などでは最大事後確率推定やそれを近似するアルゴリズムがすでに実用化されており，今後さらなる通信品質の改善を目指すためには信号検出などの問題においてもこれらの統計的信号処理手法を用いることが必要不可欠である．また，C-RAN (Cloud-Radio Access Networks) に代表されるように，基地局機能をクラウド上で実現する動きも活発化しており，より複雑な非線形の信号処理手法を通信分野に導入するための土壌が整いつつある．このため，ベイズの定理に基づく確率推論はこれからの通信分野の全ての研究者が身につけるべき基礎知識であるといえる．本稿では，ベイズの定理に基づく確率推論の基礎として，時系列解析に広く用いられている状態推定とグラフ上のメッセージ伝搬アルゴリズムである確率伝搬法について解説する．周辺事後分布の評価では如何に計算量を低減するかが重要になるが，いずれの手法も確率変数間の条件付き独立性をうまく利用した方法になっており，前者は観測および状態のマルコフ性を，後者はより一般的な同時確率分布の因数分解を計算効率の改善に役立てている．本稿で説明する各手法の関係を図 1.1 に示す．

状態推定は状態空間モデルに基づく確率推論の枠組みであり，粒子フィルタやカルマンフィルタがその代表的なアルゴリズムである．これまでに多くの通信の問題に適用されており，通信分野においても基本的な数理的手法として既に定着している．何れも時系列のような逐次的に観測が得られる状況において，観測が得られる毎に過去の推定結果を利用して効率的に推定したいパラメータの分布を更新していくアルゴリズムとなっている．通常，粒子フィルタやカルマンフィルタは予測分布とフィルタ分布の更新を行なう手順のことを指し，周辺事後分布に相当する平滑化分布の計算は含まないが，平滑化分布は予測分布とフィルタ分布を用いて効率的に計算できる．また，特にカルマンフィルタは，線形・ガウス型の状態空間モデルで与えられる動的システムにおいて最小分散推定を行なうためのアルゴリズムという形で導入されることが多いが，通信分野の研究者にとってはマルコフ性からの状態空間モデルの導出も含めて，確率推論の文脈で説明するほうが理解しやすいと思われるため，本稿ではそのようなアプローチをとることにした．

一方，確率伝搬法（Belief Propagation, BP）は複数の確率変数の同時分布から各確率変数の周辺

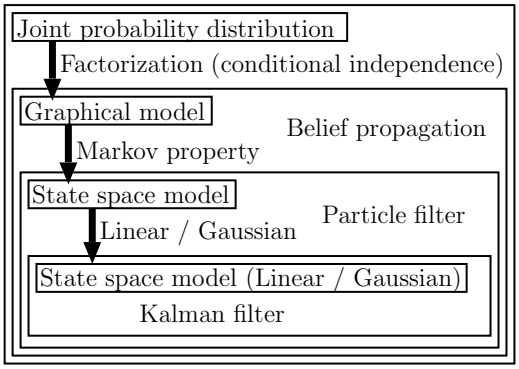


図 1.1: 本稿内容の関係

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のが最初とされている．確率伝搬法では，同時確率分布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ィカルモデルで記述し，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される．通信に関連する分野における応用としては既に実用化されている誤り訂正符号の復号に関するものが多く，比較的「枯れた手法」というイメージをもつ読者も多いかもしれないが，最近では大規模 MIMO 信号検出や圧縮センシングにおけるスパース再構成アルゴリズムなどにも応用されており，新しい数理的な手法や問題が生まれるたびに顔を出すという印象がある．このことは確率伝搬法の普遍性とその強力さの裏付けであると言える．本稿では，ファクターグラフ上の和-積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．

本稿の構成は以下の通りである．第 2 章では，本稿の内容を理解するために必要な最低限の確率に関する基礎事項の復習を行う．第 3 章では，本稿で考える確率推論の問題について説明する．第 4 章では，マルコフ性を用いて 3 種類の状態空間モデルの導出を行ない，粒子フィルタ及びカルマンフィルタのアルゴリズムを説明する．第 5 章では，確率伝搬法の基礎としてファクターグラフおよびベイジアンネットワーク上でのメッセージ伝搬アルゴリズムを解説する．第 6 章では，確率伝搬法のいくつかの応用例について説明する．最後に第 7 章で本稿のまとめを述べる．

第2章　基礎事項の確認

2.1　確率の基本法則

2つの離散値をとる確率変数 X, Y を考える．ただし，X は x_i $(i = 1, \ldots, M)$ の値を Y は y_j $(j = 1, \ldots, N)$ の値を, それぞれとるものとする[1]．このとき，X が x_i の値を取る確率 $\Pr(X = x_i)$ がある関数 $P(x)$ を用いて

$$\Pr(X = x_i) = P(x_i), \quad (i = 1, \ldots, M) \tag{2.1}$$

とかけるとき，$P(x)$ を確率変数 X の確率分布という．さらに，$X = x_i$ かつ $Y = y_j$ となる確率 (同時確率) がある関数 $P(x, y)$ を用いて

$$\Pr(X = x_i, Y = y_j) = P(x_i, y_j), \quad (i = 1, \ldots, M,\ j = 1, \ldots, N) \tag{2.2}$$

とかけるとき，$P(x, y)$ を確率変数 X, Y の同時確率分布という．また，$X = x_i$ である条件の下で $Y = y_j$ となる確率（条件付き確率）がある関数 $P(y|x)$ を用いて

$$\Pr(Y = y_j | X = x_i) = P(y_j|x_i), \quad (i = 1, \ldots, M,\ j = 1, \ldots, N) \tag{2.3}$$

とかけるとき，$P(y|x)$ を X が与えられた下での確率変数 Y の条件付き確率分布という．

以上の準備の下で，確率計算の最も基本的なルールは以下の通りである．

離散確率変数の加法定理:

$$P(x) = \sum_y P(x, y) \tag{2.4}$$

離散確率変数の乗法定理:

$$P(x, y) = P(y|x)P(x) \tag{2.5}$$

加法定理 (2.4) の操作は周辺化とも呼ばれる．また，乗法定理 (2.5) の式は条件付き確率の定義にもなっている．同時確率分布が

$$P(x, y) = P(x)P(y) \tag{2.6}$$

と書けるとき，確率変数 X と Y は独立であるといい，$X \perp\!\!\!\perp Y$ とかく．このとき，$P(y|x) = P(y)$ である．

[1]確率変数は大文字で，その実現値は小文字で表記するのが慣例である．本稿では，スカラーと多変量の確率変数を区別すること無く同一の表記を用いることに注意する．

よく用いられる他の計算ルールとして，ベイズの定理がある．これは同時確率の対称性と乗法定理から直ちに導かれる．すなわち，

$$P(x,y) = P(y,x) \tag{2.7}$$

より

$$P(y|x)P(x) = P(x|y)P(y) \tag{2.8}$$

となり，次式を得る．

ベイズの定理:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} = \frac{P(x|y)P(y)}{\sum_y P(x|y)P(y)} \tag{2.9}$$

次に連続の確率変数の場合について考える．実数値をとるスカラーの確率変数 X が任意の $c \in \mathbb{R}$ に対して $\Pr(X = c) = 0$ となるとき，X は連続の分布をもつ確率変数であるという．連続の確率変数 X に対して，ある関数 $p(x) \geq 0,\ -\infty < x < \infty$ が存在して，$a \leq X \leq b$ となる事象の確率が

$$\Pr(a \leq X \leq b) = \int_a^b p(x)dx \tag{2.10}$$

と書けるとき，$p(x)$ を確率変数 X の確率密度関数という[2]．確率密度関数は

$$p(x) \geq 0 \tag{2.11}$$

$$\int_{-\infty}^{\infty} p(x)dx = 1 \tag{2.12}$$

なる性質をもつが，$p(x)$ 自体は確率ではないため，ある x に対して $p(x) > 1$ となり得ることに注意する．

確率変数 X が区間 $(-\infty, a)$ の値をとる確率は累積分布関数と呼ばれ

$$\Pr(X \leq a) = c(a) = \int_{-\infty}^{a} p(x)dx \tag{2.13}$$

で与えられる．

2 つの連続確率変数 X, Y について考える．1 変数の場合の議論と同様にして，X, Y の同時分布の確率密度関数 $p(x,y)$ と条件付き分布の確率密度関数 $p(y|x)$ を定義すると，離散確率変数の場合と同様に加法と乗法の計算ルールが次で与えられる．

[2] これに対して，離散確率変数の場合に考えた $P(x)$ を確率質量関数ということがある．

連続確率変数の加法定理:

$$p(x) = \int p(x, y)dy \tag{2.14}$$

連続確率変数の乗法定理:

$$p(x, y) = p(y|x)p(x) \tag{2.15}$$

連続確率変数の場合は，同時分布の密度関数に対して $p(x, y) = p(x)p(y)$ が成り立つとき，X, Y は独立であるといい，$X \perp\!\!\!\perp Y$ とかく．

ここで，$\delta(x)$ を任意の連続関数 $f(x)$ について

$$\int_{-\infty}^{\infty} f(x)\delta(x)dx = f(0) \tag{2.16}$$

なる性質をみたす関数（ディラックのデルタ関数）として定義すると，$\delta(x)$ を用いることで確率密度関数と確率質量関数の重み付け和のような形で与えられるような分布も表現できる（一般化された確率密度関数と呼ばれることがある）．以下で離散確率変数と連続確率変数を同時に考える必要があるような場合には，特に明記することなくこの一般化された確率密度関数を仮定することにする．

2.2 条件付き独立

前節で議論したように，確率推論の問題では観測が得られたという条件の下での確率分布について考える必要がある．特に，周辺事後確率の計算の際に確率変数間の独立性を利用した計算量の削減を行なうが，その際に重要な概念が条件付き独立性である．２つの確率変数 X, Y が独立であることの定義は既に述べた．条件付き独立では，３つの確率変数について考えることになる．

確率変数 X, Y, Z の同時確率分布を $P(x, y, z)$ とすると（連続確率変数の場合には，同時分布の確率密度関数 $p(x, y, z)$ とする），乗法定理と周辺化により

$$P(x, y|z) = \frac{P(x, y, z)}{P(z)} \tag{2.17}$$

$$P(x|z) = \frac{P(x, z)}{P(z)} \tag{2.18}$$

$$P(y|z) = \frac{P(y, z)}{P(z)} \tag{2.19}$$

となる．このとき，全ての x, y, z の値について

$$P(x, y|z) = P(x|z)P(y|z) \tag{2.20}$$

が成り立つとき，確率変数 X と Y は Z を与えた下で条件付き独立といい，$X \perp\!\!\!\perp Y|Z$ とかく．例えば，$X \perp\!\!\!\perp Y|Z$ が成り立つとき，

$$P(x, y, z) = \frac{P(x, z)P(y, z)}{P(z)} \tag{2.21}$$

$$P(x|y, z) = P(x|z) \tag{2.22}$$

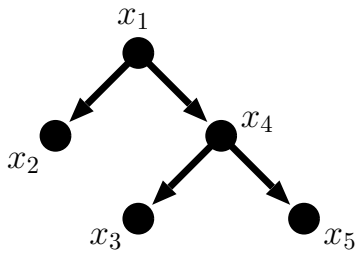


図 2.1: ベイジアンネットワークの例

などが示される．

複数の確率変数の同時確率分布が与えられたときに，どの確率変数とどの確率変数がどの確率変数について条件付き独立であるかを式の上から判定することは，それほど簡単な問題ではない．しかしながら，以下に述べるグラフィカルモデルを用いると条件付き独立性をグラフから直ちに読み取ることが可能である．

2.3　グラフィカルモデル

グラフィカルモデルとは多変量の確率変数の関係をグラフによって表現するものであり，様々な手法が提案されている．ここでは，確率推論の問題を扱う際に利用する代表的なグラフィカルモデルとして，ベイジアンネットワークとファクターグラフについて説明する．

確率変数 $X_1, X_2, \ldots, X_n$ の同時確率分布 $P(x_1, x_2, \ldots, x_n)$ 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）, すわなち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. 同時確率分布の因数分解が $P(x_1, x_2, \ldots, x_n) = P(x_1|S_1)P(x_2|S_2)\cdots P(x_n|S_n)$ 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, 同時確率分布が

$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)P(x_2|x_1)P(x_3|x_4)P(x_4|x_1)P(x_5|x_4) \tag{2.23}$$

で与えられる場合のベイジアンネットワークは図 2.1 のようになる．

また，確率変数 $X_1, X_2, \ldots, X_n$ のマルコフ連鎖の同時確率分布は

$$P(x_1, x_2, \ldots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_2)\cdots P(x_n|x_{n-1}) \tag{2.24}$$

で与えられるため，そのベイジアンネットワークは図 2.2 のようになる．

ベイジアンネットワークと条件付き独立の関係を理解するために，図 2.3 に示される確率変数 X, Y, Z についての 3 つのベイジアンネットワークについて考える．

図 2.3(a) に示されるベイジアンネットワークに対応する同時確率分布は

$$P(x, y, z) = P(x|z)P(y|z)p(z) \tag{2.25}$$

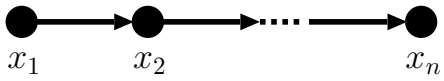


図 2.2: ベイジアンネットワークの例（マルコフ連鎖）

である．これより，

$$P(x,y) = \sum_z \{P(x|z)P(y|z)P(z)\} \tag{2.26}$$

は一般に $P(x)P(y)$ とならないため，X と Y は一般には独立ではないが，

$$P(x,y|z) = \frac{P(x,y,z)}{P(z)} = \frac{P(x|z)P(y|z)P(z)}{P(z)} = P(x|z)P(y|z) \tag{2.27}$$

となるため，X と Y は Z を与えた下で条件付き独立である．

次に，図 2.3(b) に示されるベイジアンネットワークに対応する同時確率分布は

$$P(x,y,z) = P(x)P(y|z)P(z|x) \tag{2.28}$$

であり，これより，

$$P(x,y) = \sum_z \{P(x)P(y|z)P(z|x)\} \tag{2.29}$$

は一般に $P(x)P(y)$ とならないため，X と Y は一般には独立ではないが，

$$P(x,y|z) = \frac{P(x)P(y|z)P(z|x)}{P(z)} = \frac{P(x,z)P(y|z)}{P(z)} = P(x|z)P(y|z) \tag{2.30}$$

となるため，X と Y は Z を与えた下で条件付き独立である．

最後に，図 2.3(c) に示されるベイジアンネットワークに対応する同時確率分布は

$$P(x,y,z) = P(x)P(y)P(z|x,y) \tag{2.31}$$

であり，これより，

$$P(x,y) = \sum_z \{P(x)P(y)P(z|x,y)\} = P(x)P(y) \tag{2.32}$$

となり，X と Y は独立であるが，

$$P(x,y|z) = \frac{P(x)P(y)P(z|x,y)}{P(z)} \tag{2.33}$$

は一般に $P(x|z)P(y|z)$ とならず，一般には X と Y は Z を与えた下で条件付き独立ではない．

以上のように，ベイジアンネットワークの構造から条件付き独立性を読み取ることが可能である例えば，図 2.1 の例では

$$X_2 \perp\!\!\!\perp X_4 | X_1$$
$$X_3 \perp\!\!\!\perp X_5 | X_4$$

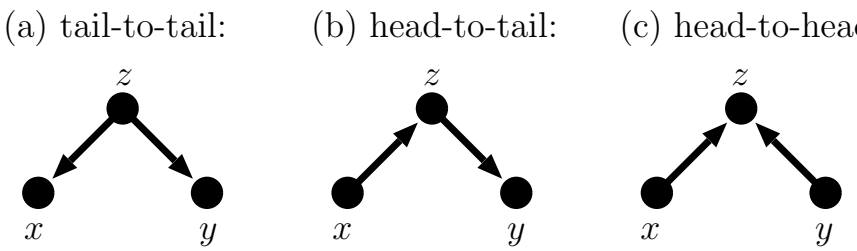


図 2.3: ベイジアンネットワークと条件付き独立

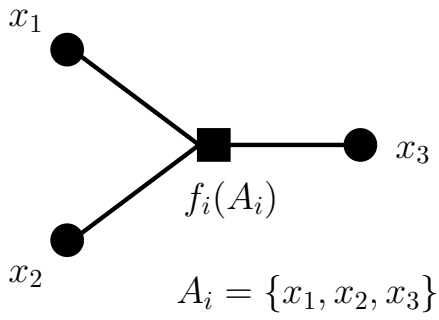


図 2.4: ファクターグラフの接続ルール

などが直ちに分かり，図 2.2 のマルコフ連鎖の例では

$$X_i \perp\!\!\!\perp X_{i+2} | X_{i+1}, \quad i = 1, \ldots, n-2$$

などが直ちに読み取れる．ベイジアンネットワークの他に，条件付き独立性からより直接的に構成されるようなグラフィカルモデル（マルコフ確率場）も存在する．

第 5 章で考える確率伝搬法のアルゴリズム（和-積アルゴリズム）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 ” から構成される．変数の集合を $\{x_1, x_2, \ldots, x_n\}$ とし, 多変数関数が

$$f(x_1, x_2, \ldots, x_n) = f_1(A_1) f_2(A_2) \cdots f_m(A_m) \tag{2.34}$$

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 $\{x_1, x_2, \ldots, x_n\}$ 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 2.4 参照）．ファクターグラフではベイジアンネットワークと異なり，2 種類のノードが存在する．

ファクターグラフの具体例として，同時確率分布が (2.23) で与えられる場合について考えてみる．

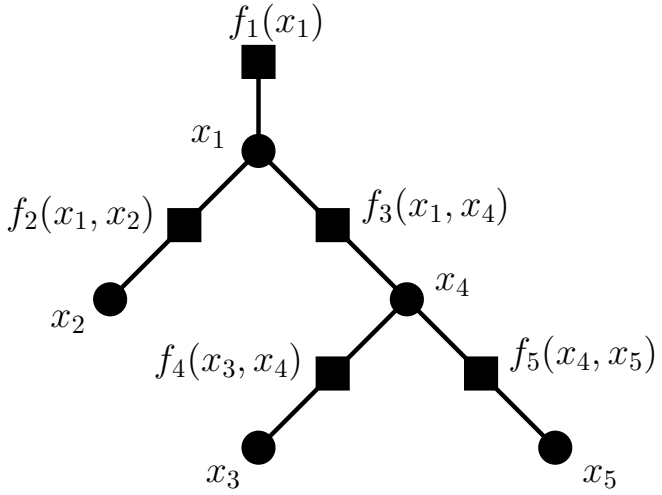


図 2.5: ファクターグラフの例

(2.23) の同時確率分布を，変数が $x_1, x_2, \ldots, x_5$ の多変数関数とみなし,

$$
\begin{aligned}
f_1(x_1) &= P(x_1) \\
f_2(x_1, x_2) &= P(x_2|x_1) \\
f_3(x_1, x_4) &= P(x_4|x_1) \\
f_4(x_3, x_4) &= P(x_3|x_4) \\
f_5(x_4, x_5) &= P(x_5|x_4)
\end{aligned}
$$

と定義して

$$
f(x_1, x_2, x_3, x_4, x_5) = f_1(x_1) f_2(x_1, x_2) f_4(x_3, x_4) f_3(x_1, x_4) f_5(x_4, x_5) \tag{2.35}
$$

ととみなすことで, ファクターグラフは図 2.5 のようになる.

次に，マルコフ連鎖モデルについて考える．マルコフ連鎖の同時確率分布は (2.24) で与えられるため, これを変数が $x_1, x_2, \ldots, x_n$ の多変数関数とみなし,

$$
\begin{aligned}
f_1(x_1) &= P(x_1) \\
f_i(x_{i-1}, x_i) &= P(x_i|x_{i-1}), \quad i = 2, ..., n
\end{aligned}
$$

と対応づけることで同時確率分布は

$$
f(x_1, x_2, \ldots, x_n) = f_1(x_1) f_2(x_1, x_2) f_3(x_2, x_3) \cdots f_n(x_{n-1}, x_n) \tag{2.36}
$$

と書ける. これよりマルコフ連鎖のファクターグラフは図 2.6 のようになる.

2.4 サンプリング法

通信や信号処理の多くの問題において，確率変数の関数の期待値を計算したり，ある関数の積分を計算する必要があるが，その多くの場合，閉形式で解を得ることができない．そのような場合に有効

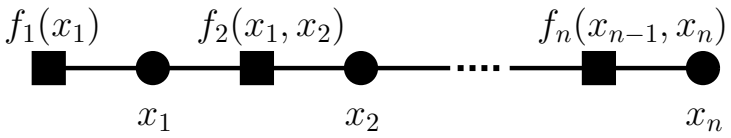


図 2.6: ファクターグラフの例（マルコフ連鎖）

な手法が，計算機によって生成した確率変数のサンプル（実現値）を用いてそれらの計算を行う方法である．ここでは，所望の確率分布からのサンプルを生成する様々な方法について述べる．計算機を用いれば，どんな確率分布からのサンプルも厳密に生成できると思われるかもしれないが実はそんなことはなく，多くの場合においては所望の分布に近似的に従うサンプルを生成することになる．本章で説明する SIR(sampling / importance resampling) は単にサンプルを生成する方法であるだけでなく，後述する逐次モンテカルロ法（粒子フィルタ）の理論的な基礎になっている．なお，本節で述べるアルゴリズムでは区間 $[0:1]$ の一様分布に従う確率変数のサンプルを生成できることを前提としているが，実際に使用される際には擬似乱数の性質に十分注意されたい．

2.4.1　逆関数法

逆関数法は最も素朴でかつ厳密なサンプリング法である．具体的には，区間 $[0:1]$ の一様分布に従う確率変数 Y からのサンプル y に対して，ある関数 f を用いて $x = f(y)$ と変数変換し，変換後の x が所望の確率分布からのサンプルになるようにするというものである．確率変数 Y の確率密度関数 $q(y)$ と変換後の確率変数 X の確率密度関数 $p(x)$ の関係は

$$p(x) = q(y) \left| \frac{dy}{dx} \right| \tag{2.37}$$

となるが，確率変数 Y の台において $q(y) = 1$ なので，これを両辺積分することで

$$y = \int_{-\infty}^{x} p(x')dx' \equiv h(x) \tag{2.38}$$

となる．ここで，$h(x)$ は確率変数 X の累積分布関数であるが，その逆関数を $h^{-1}(\cdot)$ として，$x = h^{-1}(y)$ なる関係が成り立つ．すなわち，逆関数法においては，変数変換を行う関数 f として，所望の確率変数の累積分布関数の逆関数を用いればよいことがわかる．したがって，所望の確率変数の累積分布関数の逆関数が求まる場合には，厳密に所望の確率分布に従うサンプルが生成できる．ただし，いつでもそのような逆関数が求まる訳ではないため逆関数法が適用可能な場合は限定的である．

所望の確率分布がパラメータ λ の指数分布の場合を考える．確率密度関数は

$$p(x) = \begin{cases} \lambda e^{-\lambda x} & 0 \leq x \leq \infty \\ 0 & \text{otherwise} \end{cases} \tag{2.39}$$

で与えられるため，累積分布関数は

$$\begin{aligned} y = h(x) &= \int_{-\infty}^{x} p(x')dx' \\ &= \int_{0}^{x} \lambda e^{-\lambda x'} dx' \\ &= 1 - e^{-\lambda x} \end{aligned} \tag{2.40}$$

となり，その逆関数を求めることで，変換式は

$$x = h^{-1}(y) = -\frac{\ln(1-y)}{\lambda} \tag{2.41}$$

となる．

所望の確率分布がコーシー分布の場合を考えると，確率密度関数は

$$p(x) = \frac{1}{\pi} \cdot \frac{1}{1+x^2} \tag{2.42}$$

で与えられるため，累積分布関数は

$$\begin{aligned} y = h(x) &= \int_{-\infty}^{x} \frac{1}{\pi} \cdot \frac{1}{1+x'^2} dx' \\ &= \frac{1}{\pi} \tan^{-1} x + \frac{1}{2} \end{aligned} \tag{2.43}$$

となり，変換式は

$$x = h^{-1}(y) = \tan\left(\pi y - \frac{\pi}{2}\right) \tag{2.44}$$

となる．

多変量の確率分布の場合には，変換前後の確率変数をそれぞれ $Y_1, Y_2, \ldots, Y_n$ と $X_1, X_2, \ldots, X_n$ として，それぞれの確率密度関数 $q(y_1, y_2, \ldots, y_n)$ と $p(x_1, x_2, \ldots, x_n)$ の間の関係

$$p(x_1, x_2, \ldots, x_n) = q(y_1, y_2, \ldots, y_n) \left| \frac{\partial(y_1, y_2, \ldots, y_n)}{\partial(x_1, x_2, \ldots, x_n)} \right| \tag{2.45}$$

を同様に用いればよい．ただし，$\left| \frac{\partial(y_1, y_2, \ldots, y_n)}{\partial(x_1, x_2, \ldots, x_n)} \right|$ は変換のヤコビアンである．

2.4.2 ガウス分布からのサンプル

ガウス分布には逆関数法が直接適用できないが，重要な分布であるため厳密にガウス分布に従うサンプルを生成する幾つかの方法が考えられている．ここではまず代表的なアルゴリズム 2 つを紹介する．

Marsaglia 法：

- 区間 $(-1:1)$ の一様分布に従う独立な確率変数 Y_1, Y_2 のサンプル y_1, y_2 を生成
- $y_1^2 + y_2^2 \leq 1$ を満足しないサンプルは破棄
- 保持されたサンプルのペア (y_1, y_2) から以下を計算

$$x_1 = y_1\sqrt{\frac{-2\ln(y_1^2+y_2^2)^2}{(y_1^2+y_2^2)^2}}$$
$$x_2 = y_2\sqrt{\frac{-2\ln(y_1^2+y_2^2)^2}{(y_1^2+y_2^2)^2}}$$

- 変換後の x_1, x_2 は独立な標準正規分布に従う

Marsaglia 法の 2 番目のステップで保持された (y_1, y_2) は，確率密度関数

$$p(y_1, y_2) = \begin{cases} \frac{1}{\pi} & y_1^2 + y_2^2 \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

に従って一様分布することから，変換後の確率変数を X_1, X_2 としたときにその同時分布の密度関数が

$$p(x_1, x_2) = \left(\frac{1}{\sqrt{2\pi}}e^{-\frac{x_1^2}{2}}\right)\left(\frac{1}{\sqrt{2\pi}}e^{-\frac{x_2^2}{2}}\right)$$

となることが容易に確認できる．

また，よく知られたアルゴリズムとして次の Box-Muller 法がある．

Box-Muller 法：

- 区間 $(0:1)$ の一様分布に従う独立な確率変数 Y_1, Y_2 のサンプル y_1, y_2 を生成
- サンプルのペア (y_1, y_2) から以下を計算

$$x_1 = \sqrt{-2\ln y_1}\cos 2\pi y_2$$
$$x_2 = \sqrt{-2\ln y_1}\sin 2\pi y_2$$

- 変換後の x_1, x_2 は独立な標準正規分布に従う

標準正規分布以外のガウス分布に従うサンプルを生成する必要がある場合は次のようにする．平均が μ で分散が σ^2 のガウス分布からのサンプルが必要であれば，標準正規分布からのサンプル x を前述のアルゴリズムによって生成し，それを用いて

$$z = \sigma x + \mu$$

と変数変換すればよい．実際，

$$E[Z] = E[\sigma X + \mu] = \sigma E[X] + \mu = \mu$$
$$E[(Z - E[Z])^2] = E[(Z - \mu)^2] = E[(\sigma X)^2] = \sigma^2 E[X^2] = \sigma^2$$

である．また，平均ベクトルが μ, 共分散行列が Σ の多変量ガウス分布からのサンプルが必要であれば，多変量標準正規分布からのサンプル x を前述のアルゴリズムから生成し，それを用いて

$$z = Lx + \mu$$

と変数変換すればよい．ただし，L は Σ のコレスキー分解 $\Sigma = LL^{\mathrm{T}}$ で与えられる．実際，

$$E[Z] = E[LX + \mu] = LE[X] + \mu = \mu$$
$$E[(Z - E[Z])(Z - E[Z])^{\mathrm{T}}] = E[LXX^{\mathrm{T}}L^{\mathrm{T}}] = LE[XX^{\mathrm{T}}]L^{\mathrm{T}} = LL^{\mathrm{T}} = \Sigma$$

である．

2.4.3 棄却サンプリング

逆関数法よりも広いクラスの確率分布に対してサンプルを生成することができる方法に棄却サンプリングがある．棄却サンプリングが適用可能な確率分布にも制限があるが，応用上極めて重要な重点サンプリングや SIR を理解する際に役に立つため，ここでその考え方について説明する．

サンプルを生成したい確率分布の密度関数 $p(x)$ があるが，$p(x)$ から直接その実現値を生成することが困難であるとする．ただし，応用上，密度関数はその値がある正規化定数 Z_p を除いてのみ分かることが多いため，$p(x) = \frac{1}{Z_p}\tilde{p}(x)$ としたとき，任意の x について $\tilde{p}(x)$ の値は分かるが，$p(x)$ の値は分からないと仮定する．棄却サンプリングと後述する重点サンプリングや SIR に共通する重要なアイデアは，実現値が生成可能な提案分布と呼ばれる確率密度関数 $q(x)$ の存在を仮定することである．提案分布 $q(x)$ は何でも良いが，$q(x)$ の台，すなわち，$q(x) \neq 0$ となる x の集合，が $p(x)$ の台を含む必要がある．さらに，棄却サンプリングでは，全ての x に対して，$kq(x) \geq \tilde{p}(x)$ となるような定数 k を設定する必要がある．以上の準備のもとで，棄却サンプリングの手順は以下のとおりである．

棄却サンプリングの手順：

- サンプルを抽出できる分布 $q(x)$ を用意（提案分布）
- 提案分布 $q(x)$ に従う確率変数 X_q のサンプル x_q を生成
- 区間 $[0 : kq(x_q)]$ の一様分布に従う確率変数 U のサンプル u を生成
- $u > \tilde{p}(x_q)$ ならば x_q を棄却，そうでなければ x_q を保持
- 保持された x_q は確率分布 $p(x)$ に従う確率変数のサンプル

棄却サンプリングによって得られたサンプルが，所望の確率分布 $p(x)$ に従うことを確認するために，最後に保持された確率変数 X_q の累積分布関数 $\Pr(X_q \leq a|\text{keep})$ を考えると

$$\begin{aligned}
\Pr(X_q \leq a|\text{keep}) &= \frac{\Pr(X_q \leq a, \text{keep})}{\Pr(\text{keep})} \\
&= \frac{\displaystyle\int_{-\infty}^{\infty}\int_{0}^{kq(x)} I(x \leq a, 0 \leq u \leq \tilde{p}(x))q(x)\frac{1}{kq(x)}dudx}{\displaystyle\int_{-\infty}^{\infty}\int_{0}^{kq(x)} I(0 \leq u \leq \tilde{p}(x))q(x)\frac{1}{kq(x)}dudx} \\
&= \frac{\displaystyle\int_{-\infty}^{a}\frac{\tilde{p}(x)}{k}dx}{\displaystyle\int_{-\infty}^{\infty}\frac{\tilde{p}(x)}{k}dx} \\
&= \frac{\displaystyle\int_{-\infty}^{a}\frac{\tilde{p}(x)}{Z_p}dx}{\displaystyle\int_{-\infty}^{\infty}\frac{\tilde{p}(x)}{Z_p}dx} \\
&= \int_{-\infty}^{a} p(x)dx
\end{aligned} \tag{2.46}$$

となり，これは $p(x)$ の累積分布関数に他ならない．ただし，$I(\cdot)$ は特性関数であり，引数として与えられた条件を満足する場合に 1，そうでない場合に 0 の値をもつ．

棄却サンプリングのアイデアは，前述の Marsaglia 法において区間 $(-1, 1)$ の一様分布に従う 2 つの独立な確率変数のサンプルから単位円上に一様分布する確率変数のサンプルを生成したときのそれと本質的に同一であるが，より一般の確率分布を対象とした棄却サンプリングでは「$p(x)$ の台を含む適切な提案分布 $q(x)$ が決められるか?」，「適切な k を決められるか？」が大きな問題となる．例えば，十分大きな台をもつ一様分布を考えて，k を十分大きな値に設定すれば，棄却サンプリングで求められる要請を満足することは可能であるが，棄却サンプリングでサンプルが保持される確率は

$$\begin{aligned}
\Pr(\text{keep}) &= \int_{-\infty}^{\infty}\int_{0}^{kq(x)} I(0 \leq u \leq \tilde{p}(x))q(x)\frac{1}{kq(x)}dudx \\
&= \frac{1}{k}\int_{-\infty}^{\infty}\tilde{p}(x)dx
\end{aligned}$$

となり，k が大きくなるにつれて小さくなるため，計算効率を考えるとこのアプローチは現実的ではない．多変量の確率分布の場合にはこの問題はさらに深刻になる．

2.4.4　重点サンプリング

所望の確率分布からのサンプリングを行う目的には様々なものが考えられるが，その多くの場合，確率変数 X に対するある関数 $f(X)$ の確率密度関数 $p(x)$ のもとでの期待値

$$E[f(X)] = \int f(x)p(x)dx \tag{2.47}$$

の計算である．$p(x)$ に従うサンプルを生成することができないが，与えられた x に対して $p(x)$ の値の計算は可能であるという場合に，(2.47) を近似的に計算する素朴な方法として，x の空間を均一な

グリッドで離散化し，それらの値 $x^{(1)}, \ldots, x^{(L)}$ を用いて

$$E[f(X)] \approx \frac{1}{L} \sum_{l=1}^{L} f(x^{(l)}) p(x^{(l)}) \tag{2.48}$$

とすることであるが，この方法では近似計算に必要とされる離散点の数が x の次元に対して指数的に増加してしまうという問題がある．$p(x)$ の値が大きいところ，あるいはもし可能であれば，$f(x)p(x)$ の値が大きいところから選択的にサンプルを生成できることが望ましい．

効率的に期待値の近似計算を行うことを目的に考えられたのが，重点サンプリングである（重点サンプリングの目的は確率変数の実現値のサンプリング自体ではないことに注意する）．重点サンプリングでは棄却サンプリングと同様に，サンプルを生成可能な確率分布の密度関数 $q(x)$ を用意し，$q(x)$ からの L 個のサンプル $x^{(1)}, \ldots, x^{(L)}$ を用いて期待値計算を以下のように近似する．

$$\begin{aligned} E[f(X)] &= \int f(x) p(x) dx \\ &= \int f(x) \frac{p(x)}{q(x)} q(x) dx \\ &\approx \frac{1}{L} \sum_{l=1}^{L} f(x^{(l)}) \frac{p(x^{(l)})}{q(x^{(l)})} \end{aligned} \tag{2.49}$$

ここで，$r_l \equiv \frac{p(x^{(l)})}{q(x^{(l)})}$ は重要度重みと呼ばれる．重点サンプリングの計算効率は，$q(x)$ の選択に大きく依存する．例えば，$q(x) = p(x)$ であれば，$p(x)$ の値が大きいところのサンプルを重点的に用いて期待値を評価できるため効率的な期待値の近似計算が可能となるが，$q(x)$ として一様分布を用いた場合には本質的に (2.48) と同じ計算を行うことになり，$p(x)$ が一様分布に近い形状をもつという特殊な場合を除いて，一般に計算効率が著しく低下する．

重点サンプリングでは，所望の分布の密度関数 $p(x)$ と提案分布の密度関数 $q(x)$ の両方の正規化定数が分からない場合にも計算が可能である．$p(x) = \frac{\tilde{p}(x)}{Z_p}$，$q(x) = \frac{\tilde{q}(x)}{Z_q}$ として，$\tilde{p}(x)$ と $\tilde{q}(x)$ のみが分かっている場合には, $q(x)$ からの L 個のサンプル $x^{(1)}, \ldots, x^{(L)}$ を用いて

$$\begin{aligned} E[f(X)] &= \int f(x) p(x) dx \\ &= \int f(x) \frac{p(x)}{q(x)} q(x) dx \\ &= \frac{Z_q}{Z_p} \int f(x) \frac{\tilde{p}(x)}{\tilde{q}(x)} q(x) dx \\ &\approx \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^{L} f(x^{(l)}) \tilde{r}_l \end{aligned} \tag{2.50}$$

と近似する．ただし，$\tilde{r}_l = \frac{\tilde{p}(x^{(l)})}{\tilde{q}(x^{(l)})}$ である．また，正規化定数の比 $\frac{Z_p}{Z_q}$ は

$$\begin{aligned} \frac{Z_p}{Z_q} &= \frac{1}{Z_q} \int \tilde{p}(x) dx \\ &= \int \frac{\tilde{p}(x)}{\tilde{q}(x)} q(x) dx \\ &\approx \frac{1}{L} \sum_{l=1}^{L} \tilde{r}_l \end{aligned} \tag{2.51}$$

と近似できるため，重点サンプリングの手順は以下のようになる．

重点サンプリングの手順：

- サンプルを抽出できる分布 $q(x)$ を用意（提案分布）
- 提案分布 $q(x)$ から L 個のサンプル $x^{(1)}, \ldots, x^{(L)}$ を生成
- $x^{(1)}, \ldots, x^{(L)}$ から $\tilde{r}_l = \dfrac{\tilde{p}(x^{(l)})}{\tilde{q}(x^{(l)})}$，$(1 \leq l \leq L)$ を計算
- $\tilde{r}_1, \ldots, \tilde{r}_L$ から $w_l = \dfrac{\tilde{r}_l}{\sum_{n=1}^{L} \tilde{r}_n}$，$(1 \leq l \leq L)$ を計算
- 期待値の近似値を $E[f(X)] \approx \displaystyle\sum_{l=1}^{L} f(x^{(l)}) w_l$ によって計算

重点サンプリングでは棄却サンプリングとは異なり，生成されたすべてのサンプルが棄却されるに使用されるが，$f(x)p(x)$ の値が小さい領域のサンプルは期待値計算にはほとんど寄与しない．

2.4.5　SIR (sampling / importance resampling)

SIR (sampling / importance resampling) は前節の重点サンプリングによく似たアイデアを用いたサンプリング法であるが，重点サンプリングとは異なり，所望の確率分布からのサンプルを（近似的に）生成すること自体を目的としている．また，SIR は第 4 章で説明する粒子フィルタのアルゴリズムの根幹をなす重要なサンプリング法である．SIR のアルゴリズムを以下に示す．

SIR の手順：

- 容易にサンプルを抽出できる分布 $q(x)$ を用意（提案分布）
- 提案分布 $q(x)$ から L 個のサンプル $x^{(1)}, x^{(2)}, \ldots, x^{(L)}$ を生成
- $w_l = \dfrac{\tilde{p}(x^{(l)})/q(x^{(l)})}{\sum_{n=1}^{L} \tilde{p}(x^{(n)})/q(x^{(n)})}$ によって重み $w_1, w_2, \ldots, w_L$ を決定
- 確率質量 $w_1, w_2, \ldots, w_L$ の離散分布 $(x^{(1)}, x^{(2)}, \ldots, x^{(L)})$ に従う確率変数 X_r の L 個のサンプルを抽出する（リサンプリング）

最後のステップ（リサンプリング）においては，一様分布からサンプルが生成できれば，所望の離散分布からのサンプルが生成できる．具体的には，区間 $[0:1]$ をそれぞれの区間の幅が $w_1, w_2, \ldots, w_L$ であるような L 個の区間に分割し，区間 $[0:1]$ の一様分布から生成したサンプルが l 番目の区間に含まれていれば，$x^{(l)}$ が離散分布からのサンプルとして抽出されたとすれば良い．SIR では棄却サンプリングと同様に提案分布を用いるが，棄却サンプリングで必要であった定数 k の決定が不要であ

る．ただし，L が有限の場合には近似的なサンプルでしかないことに注意する．実際，リサンプリングされた確率変数 X_r の累積分布関数を計算すると，

$$\begin{aligned}\Pr(X_r \leq a) &= \sum_{l:x^{(l)} \leq a} w_l \\ &= \frac{\displaystyle\sum_{l:x^{(l)} \leq a} \frac{\tilde{p}(x^{(l)})}{q(x^{(l)})}}{\displaystyle\sum_{n=1}^{L} \frac{\tilde{p}(x^{(n)})}{q(x^{(m)})}} \\ &= \frac{\displaystyle\frac{1}{L}\sum_{l=1}^{L} I(x^{(l)} \leq a)\frac{\tilde{p}(x^{(l)})}{q(x^{(l)})}}{\displaystyle\frac{1}{L}\sum_{n=1}^{L} \frac{\tilde{p}(x^{(n)})}{q(x^{(n)})}}\end{aligned}$$

となり，$L \to \infty$ を考えると

$$\begin{aligned}\Pr(X_r \leq a) &\to \frac{\displaystyle\int I(x \leq a)\frac{\tilde{p}(x)}{q(x)}q(x)dx}{\displaystyle\int \frac{\tilde{p}(x)}{q(x)}q(x)dx} \\ &= \frac{\displaystyle\int I(x \leq a)\tilde{p}(x)dx}{\displaystyle\int \tilde{p}(x)dx} \\ &= \int I(x \leq a)p(x)dx \qquad (2.52)\end{aligned}$$

となる．これは $p(x)$ の累積分布関数に他ならない．

$p(x)$ に従う確率変数 X に対して，その関数 $f(X)$ の期待値を求めたい場合には，重点サンプリングと同様に

$$\begin{aligned}E[f(X)] &= \int f(x)p(x)dx \\ &= \frac{\displaystyle\int f(x)\frac{\tilde{p}(x)}{q(x)}q(x)dx}{\displaystyle\int \frac{\tilde{p}(x)}{q(x)}q(x)dx} \\ &\approx \sum_{l=1}^{L} w_l f(x^{(l)}) \qquad (2.53)\end{aligned}$$

とすればよい．

第3章　確率推論問題

2.1 節で説明したベイズ則はそれ自体条件付き確率の間の関係式に過ぎないが，そこに登場する確率変数に役割を与えることで，統計的信号処理や機械学習において中心的な役割を果たす式になる．具体的には，簡単のため離散確率変数を仮定し，ベイズ則

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \tag{3.1}$$

において，Y を観測可能な確率変数，X を興味がある未知の確率変数とすると，左辺の $P(x|y)$ は事後確率，$P(y|x)$ は尤度関数[1]，$P(x)$ は事前分布と呼ばれる．ここで，事後確率 $P(x|y)$ は Y の値が観測されたという事象の下での未知の興味のある確率変数 X の確率を表しており，これを求めることが確率推論の主題となる[2]．

確率推論問題: 確率変数 Y の観測 y に基づいて事後分布 $P(x|y)$ を求めること

例えば，確率変数 X が x_i $(i = 1, \ldots, M)$ の値をとり得るとし，その推定値を $\hat{x}$ としたとき，$Y = y$ が観測された下で推定値 $\hat{x}$ が正しい確率は

$$\Pr(X = \hat{x}, Y = y) = P(\hat{x}|y)P(y) \tag{3.2}$$

となる．したがって，推定値が正しい確率を最大化するため，言い換えると，誤り確率を最小化するためには

$$\hat{x}_{\mathrm{map}} = \arg\max_{x_i} P(x_i|y) \tag{3.3}$$

のように事後確率を最大にする x_i を推定値として選択すればよいことが分かる．この推定法を最大事後確率推定という．また，事前確率 $P(x)$ が x_i によらず一定の場合は，ベイズ則より

$$\hat{x}_{\mathrm{ml}} = \arg\max_{x_i} P(y|x_i) = \hat{x}_{\mathrm{map}} \tag{3.4}$$

となり，尤度関数を最大化する x_i と事後確率を最大化する x_i が一致する．尤度関数を最大化する推定法は，最尤推定と呼ばれる．ビット誤り率の最小化や誤識別率の最小化は通信や機械学習において基本的であり，この例から事後確率の評価が重要であることが分かる．ただし，得られた事後確率の使用方法にはここで述べた最大事後確率推定以外にも様々なものがあり，問題や目的に応じて処理方法を選択する必要がある．

---

[1] 確率変数 Y の実現値が観測として与えられた下では，$P(y|x)$ はもはや確率ではないため関数と呼ばれる．

[2] 典型的なディジタル通信では，送信信号 X は離散確率変数である一方，受信信号 Y は連続の確率変数である．この場合は，連続確率変数と離散確率変数についてのベイズ則 [1] $P(x|y) = \frac{p(y|x)P(x)}{p(y)}$ を考える

一般には観測される確率変数や未知の確率変数はいずれも複数存在する．このため，確率変数 $X_1,\ldots,X_n$ の同時確率分布を $P(x_1,\ldots,x_n)$ とし，$X_1,\ldots,X_n$ のうち $X_{m+1},\ldots,X_n$ の観測値が得られるとしたとき，最大事後確率推定の問題は次のように定義される．

最大事後確率推定（ブロック毎）： 確率変数 $X_1,\ldots,X_n$ 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, 事後分布 $P(x_1,\ldots,x_m|a_{m+1},\ldots,a_n)$ を最大にする $x_1,\ldots,x_m$ を求める問題

$$\{\hat{x}_1,\ldots,\hat{x}_m\} = \arg\max_{x_1,\ldots,x_m} P(x_1,\ldots,x_m|a_{m+1},\ldots,a_n) \tag{3.5}$$

これは，ブロック誤り確率を最小にする推定法であり，ブロック毎最大事後確率推定と呼ばれる．一方，周辺事後分布を考えることで，成分毎の誤り確率を最小にする推定法は成分毎最大事後確率推定（あるいは最大周辺事後確率推定）と呼ばれ，次で与えられる．

最大事後確率推定（成分毎）： 確率変数 $X_1,\ldots,X_n$ 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, 周辺事後分布 $P(x_i|a_{m+1},\ldots,a_n)$, $i=1,\ldots,m$ を最大にする $x_1,\ldots,x_m$ を求める問題

$$\hat{x}_i = \arg\max_{x_i} P(x_i|a_{m+1},\ldots,a_n) \tag{3.6}$$

最大事後確率推定は，ブロック毎の (3.5) も成分毎の (3.6) も，素朴に計算すると指数オーダーの計算量が必要となる．例えば，(3.6) を直接計算して x_1 を評価しようとすると

$$\begin{aligned}\Pr(X_1 = x_1|X_{m+1} = a_{m+1},\ldots,X_n = a_n) &= \frac{\Pr(X_1 = x_1, X_{m+1} = a_{m+1},\ldots,X_n = a_n)}{\Pr(X_{m+1} = a_{m+1},\ldots,X_n = a_n)} \\ &= \alpha \sum_{x_2,\ldots,x_m} P(x_1,x_2,\ldots,x_m,a_{m+1},\ldots,a_n)\end{aligned}$$

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, (3.5) や (3.6) を直接評価することが困難であることを意味する．しかしながら, (3.6) については，適切な確率モデルとその構造を利用することで，周辺事後確率の計算を効率的に行なうことが可能であり，その基本的な手法について第 4 章と第 5 章で説明する．

第4章　状態推定

通信や信号処理の多くの問題では，時系列データのような各時刻で得られる観測に基づいて逐次的に確率推論を行う必要がある．このような問題に対する強力な手法が，時系列をマルコフ性をもつ確率過程と仮定することで得られるモデル（状態空間モデル）に基づいて状態推定を行うアプローチである．本章では，状態空間モデルを導入し，状態推定の基本的な考え方について説明した後，代表的な状態推定法である粒子フィルタとカルマンフィルタについて説明する．

4.1　状態空間モデル

未知の興味のある離散時間確率過程（状態と呼ぶ）を $X_1, \ldots, X_T$ とし，そのそれぞれに対する観測データを $Y_1, \ldots, Y_T$ とする．$X_1, \ldots, X_T$ および $Y_1, \ldots, Y_T$ の同時分布の (一般化された) 確率密度関数 $p(x_1, \ldots, x_T, y_1, \ldots, y_T)$ は，確率の基本法則（積法則）を繰り返し用いることで以下のように変形することが出来る．

$$\begin{aligned}
p(x_1, \ldots, x_T, y_1, \ldots, y_T) &= p(y_T | x_1, \ldots, x_T, y_1, \ldots, y_{T-1}) p(x_1, \ldots, x_T, y_1, \ldots, y_{T-1}) \\
&= p(y_T | x_{1:T}, y_{1:T-1}) p(x_{1:T}, y_{1:T-1}) \\
&= p(y_T | x_{1:T}, y_{1:T-1}) p(x_T | x_{1:T-1}, y_{1:T-1}) p(x_{1:T-1}, y_{1:T-1}) \\
&= p(y_T | x_{1:T}, y_{1:T-1}) p(x_T | x_{1:T-1}, y_{1:T-1}) \\
&\quad \cdot p(y_{T-1} | x_{1:T-1}, y_{1:T-2}) p(x_{T-1} | x_{1:T-2}, y_{1:T-2}) p(x_{1:T-2}, y_{1:T-2}) \\
&\qquad \vdots \\
&= \prod_{t=1}^{T} p(y_t | x_{1:t}, y_{1:t-1}) p(x_t | x_{1:t-1}, y_{1:t-1}) \qquad (4.1)
\end{aligned}$$

ただし，$x_{1:T}$ は $x_1, \ldots, x_T$ を意味し，表記上 $p(y_1 | x_{1:1}, y_{1:0}) = p(y_1 | x_1)$ および $p(x_1 | x_{1:0}, y_{1:0}) = p(x_1)$ とする．ここまでの変形では何の仮定も導入しておらず，(4.1) は単に同時分布の密度関数の別表現を与えたに過ぎないが，ここで，マルコフ性を仮定することによって周辺事後分布の計算が極めて効率的に行なえる確率モデルが構築される．具体的には，X_t は X_{t-1} にのみ依存し，Y_t は X_t にのみ依存すると仮定すると，

$$p(x_t | x_{1:t-1}, y_{1:t-1}) = p(x_t | x_{t-1}) \qquad (4.2)$$

$$p(y_t | x_{1:t}, y_{1:t-1}) = p(y_t | x_t) \qquad (4.3)$$

となり，(4.1) は

$$p(x_{1:T}, y_{1:T}) = \prod_{t=1}^{T} p(y_t | x_t) p(x_t | x_{t-1}) \qquad (4.4)$$

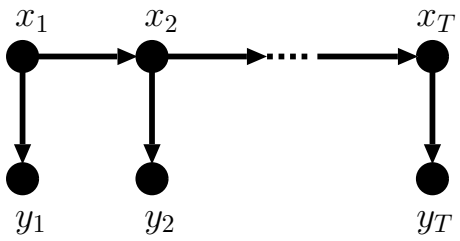


図 4.1: 一般状態空間モデルのベイジアンネットワーク

と書ける．ただし，表記上 $p(x_1|x_0) = p(x_1)$ としている．同時分布の密度関数がこのように因数分解されることを利用して効率的に状態 X_t の周辺事後確率密度関数を計算することが可能であり，粒子フィルタやカルマンフィルタはその代表的なアルゴリズムである．

(4.2) および (4.3) の仮定に注目すると，状態 X_t および観測 Y_t がどのように生成されるかについてのモデルとなっており，それぞれ，システムモデル，観測モデルと呼ばれ，これらをまとめて一般状態空間モデルという．

一般状態空間モデル：

$$\begin{aligned} x_t &\sim p(x_t|x_{t-1}) \quad &&: \text{システムモデル} \\ y_t &\sim p(y_t|x_t) &&: \text{観測モデル} \end{aligned}$$

(4.4) をベイジアンネットワークで表現すると図 4.1 に示すような鎖状の構造をもつため，鎖状グラフィカルモデルとも呼ばれる．また，隠れマルコフモデルも同様のグラフィカルモデルで表現されるが，これは状態と観測のいずれもが離散値をとる一般状態空間モデルの特別な場合となっている．

一般状態空間モデルの他の特殊ケースとして，非線形・非ガウス型状態空間モデルがある．非線形・非ガウス型状態空間モデルは，$f_t(\cdot,\cdot)$, $h_t(\cdot,\cdot)$ を非線形関数，w_t, v_t をそれぞれ白色雑音 W_t, W_t の実現値として，次で与えられる．

非線形・非ガウス型状態空間モデル：

$$\begin{aligned} x_t &= f_t(x_{t-1}, w_t) \quad &&: \text{システムモデル} \\ y_t &= h_t(x_t, v_t) &&: \text{観測モデル} \end{aligned}$$

さらに特殊なケースとして，上記の非線形関数を線形関数に，白色雑音を白色ガウス雑音に限定することで，線形・ガウス型状態空間モデルが得られる．線形・ガウス型状態空間モデルは，F_t, G_t, H_t を行列，w_t, v_t をそれぞれ白色ガウス雑音 W_t, W_t の実現値として，次で与えられる．

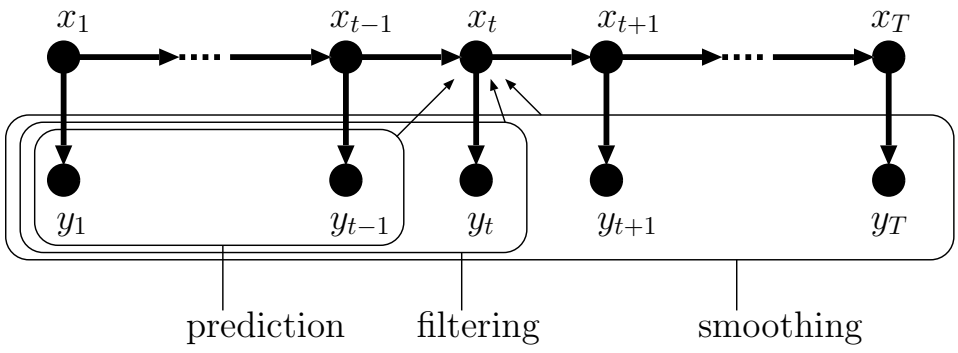


図 4.2: 重要な分布：予測分布，フィルタ分布，平滑化分布

線形・ガウス型状態空間モデル：

$$x_t = F_t x_{t-1} + G_t w_t \quad \text{：システムモデル}$$
$$y_t = H_t x_t + v_t \quad \text{：観測モデル}$$

このモデルは制御工学の講義等で目にした読者も多いと思われるが，多くの講義や教科書では背景についての説明無しに導入されることが多いため,「なぜこのようなモデルを考えるのか？」,「なぜこのモデルで良いのか？」など疑問に感じたかもしれない．実は，状態空間モデルは「現在の状態は一つ前の時刻の状態にのみ依存する」,「現在の観測は現在の状態にのみ依存する」という 2 つの仮定から自然に導かれる確率モデルなのである．

4.2 予測，フィルタ，平滑化

4.2.1 予測分布

状態 x_{t-1} のフィルタ分布 $p(x_{t-1}|y_{1:t-1})$ から，状態 x_t の 1 ステップ予測分布 $p(x_t|y_{1:t-1})$ を求める問題を考える．予測分布は

$$\begin{aligned} p(x_t|y_{1:t-1}) &= \int p(x_t, x_{t-1}|y_{1:t-1})dx_{t-1} \\ &= \int p(x_t|x_{t-1}, y_{1:t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \\ &= \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \end{aligned} \tag{4.5}$$

と書ける．ただし，2 行目から 3 行目への式変形において，マルコフ性（システムモデル）$p(x_t|x_{t-1}, y_{1:t-1}) = p(x_t|x_{t-1})$ を用いている．これより，状態 x_t の予測分布 $p(x_t|y_{1:t-1})$ が状態 x_{t-1} のフィルタ分布 $p(x_{t-1}|y_{1:t-1})$ とシステムモデル $p(x_t|x_{t-1})$ によって表現されることが分かる．

4.2.2　フィルタ分布

次に，状態 x_t の 1 ステップ予測分布 $p(x_t|y_{1:t-1})$ と時刻 t における観測 y_t から，状態 x_t のフィルタ分布 $p(x_t|y_{1:t})$ を求める問題を考える．フィルタ分布は

$$\begin{aligned} p(x_t|y_{1:t}) &= p(x_t|y_{1:t-1}, y_t) \\ &= \frac{p(x_t, y_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \\ &= \frac{p(x_t, y_t|y_{1:t-1})}{\int p(x_t, y_t|y_{1:t-1})dx_t} \\ &= \frac{p(y_t|x_t, y_{1:t-1})p(x_t|y_{1:t-1})}{\int p(y_t|x_t, y_{1:t-1})p(x_t|y_{1:t-1})dx_t} \\ &= \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t} \end{aligned} \tag{4.6}$$

と書ける．ただし，4 行目から 5 行目への式変形において，マルコフ性（観測モデル）$p(y_t|x_t, y_{1:t-1}) = p(y_t|x_t)$ を用いている．これより，時刻 t におけるフィルタ分布 $p(x_t|y_{1:t})$ が時刻 $t-1$ における予測分布 $p(x_t|y_{1:t-1})$ と観測モデル $p(y_t|x_t)$ によって表現されることが分かる．

4.2.3　平滑化分布

最後に，状態 x_{t+1} の平滑化分布 $p(x_{t+1}|y_{1:T})$ から，状態 x_t の平滑化分布 $p(x_t|y_{1:T})$ を求める問題を考える．平滑化分布は

$$\begin{aligned} p(x_t|y_{1:T}) &= \int p(x_t, x_{t+1}|y_{1:T})dx_{t+1} \\ &= \int p(x_t|x_{t+1}, y_{1:T})p(x_{t+1}|y_{1:T})dx_{t+1} \end{aligned} \tag{4.7}$$

と書ける．ここで，$p(x_t, y_{1:T}|x_{t+1})$ に対して確率の積法則を用いると

$$p(x_t, y_{1:T}|x_{t+1}) = p(x_t|x_{t+1}, y_{1:T})p(y_{1:T}|x_{t+1}) \tag{4.8}$$

となるが，一方，$x_t, y_{1:t}$ と $y_{t+1:T}$ の x_{t+1} を与えた下での条件付き独立性を用いると

$$\begin{aligned} p(x_t, y_{1:T}|x_{t+1}) &= p(x_t, y_{1:t}, y_{t+1:T}|x_{t+1}) \\ &= p(x_t, y_{1:t}|x_{t+1})p(y_{t+1:T}|x_{t+1}) \\ &= p(x_t|x_{t+1}, y_{1:t})p(y_{1:t}|x_{t+1})p(y_{t+1:T}|x_{t+1}) \\ &= p(x_t|x_{t+1}, y_{1:t})p(y_{1:t}, y_{t+1:T}|x_{t+1}) \\ &= p(x_t|x_{t+1}, y_{1:t})p(y_{1:T}|x_{t+1}) \end{aligned} \tag{4.9}$$

となる．従って，

$$p(x_t|x_{t+1}, y_{1:T}) = p(x_t|x_{t+1}, y_{1:t}) \tag{4.10}$$

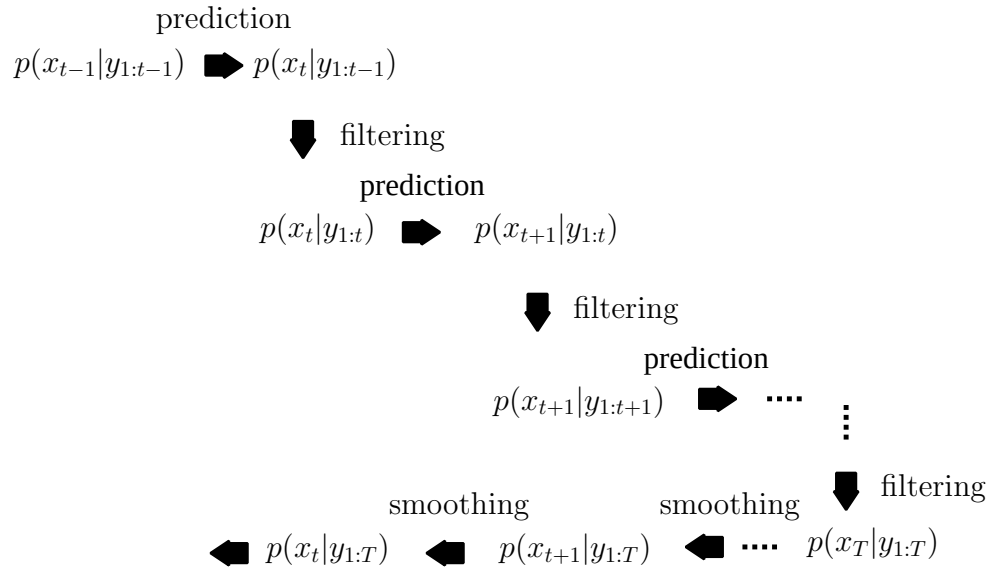


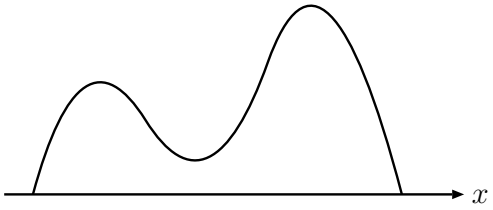
図 4.3: 時系列の状態推定の流れ

が成り立つ (d-separation と呼ばれる) ので，これを用いると平滑化分布は

$$\begin{aligned} p(x_t|y_{1:T}) &= \int p(x_t|x_{t+1}, y_{1:t})p(x_{t+1}|y_{1:T})dx_{t+1} \\ &= \int \frac{p(x_t, x_{t+1}|y_{1:t})}{p(x_{t+1}|y_{1:t})}p(x_{t+1}|y_{1:T})dx_{t+1} \\ &= \int \frac{p(x_{t+1}|x_t, y_{1:t})p(x_t|y_{1:t})}{p(x_{t+1}|y_{1:t})}p(x_{t+1}|y_{1:T})dx_{t+1} \\ &= p(x_t|y_{1:t}) \int \frac{p(x_{t+1}|x_t)}{p(x_{t+1}|y_{1:t})}p(x_{t+1}|y_{1:T})dx_{t+1} \end{aligned} \tag{4.11}$$

と書ける．ただし，最後の行への式変形で，マルコフ性（システムモデル）$p(x_{t+1}|x_t, y_{1:t}) = p(x_{t+1}|x_t)$ を用いている．これより，状態 x_t の平滑化分布 $p(x_t|y_{1:T})$ は，状態 x_{t+1} の平滑化分布 $p(x_{t+1}|y_{1:T})$ と状態 x_t のフィルタ分布 $p(x_t|y_{1:t})$，状態 x_{t+1} の予測分布 $p(x_{t+1}|y_{1:t})$，システムモデル $p(x_t|x_{t-1})$ によって表現されることが分かる．

以上のように，各時刻で観測を得る毎に 1 時刻前の状態の予測分布を用いて現時刻の状態のフィル分布を求め，さらに，そのフィルタ分布を用いて次の時刻の状態の予測分布を求める，というのが基本的な状態推定の流れである．そして，必要があれば，全ての観測が終わった後に，時間軸を遡って各時刻の平滑化分布，すなわち事後周辺分布，を求めることになる．これらの処理の流れのまとめを図 4.3 に示す．この方法によって，各時刻で予測分布やフィルタ分布，平滑化分布を求める際に周辺化が必要な確率変数の数が大幅に削減されるため，それぞれの分布をそれまでの計算結果を使用せずに毎回計算する素朴な方法に比べると，計算量が大きく軽減されている．しかしながら，更新式には積分計算が含まれており，各分布が解析的に与えられない場合や計算量に強い制限がある場合などにはそのままこのアプローチを適用することは困難である．以下で述べる粒子フィルタやカルマンフィルタはそのような場合の現実的な状態推定法として知られている．



(a)original probability density function

(b)approximated probability density function

図 4.4: 粒子による密度関数の近似

4.3　粒子フィルタ

粒子フィルタは，非線形・非ガウス型の状態空間モデルにおいて予測分布とフィルタ分布の近似を逐次的に更新していくアルゴリズムである．重要な点は，確率密度関数を複数の「粒子」と呼ばれるサンプルを用いて近似することと，フィルタリングのステップにおいてサンプリング法の一つである SIR のアイデアを利用することである．以下ではその詳細について説明する．

粒子フィルタでは，一般に連続確率変数の条件付き確率密度関数である $p(x_t|y_{1:t-1})$ や $p(x_t|y_{1:t})$ を，粒子と呼ばれるそれぞれの分布からの実現値（サンプル）を用いて近似する．図 4.4 にその例を示す．元の確率密度関数と粒子を用いた確率質量関数を比較すると，粒子による表現が元の確率密度関数の近似になっているように見えないが，累積分布関数では近似になっているという点が重要である．

非線形・非ガウス型の状態空間モデル

$$x_t = f_t(x_{t-1}, w_t)$$
$$y_t = h_t(x_t, v_t)$$

を考える．時刻 t の状態の予測分布 $p(x_t|y_{1:t-1})$ を近似する粒子を $\hat{x}_t^{(i)}$, $i = 1, \ldots, L$ とし，時刻 t の状態のフィルタ分布 $p(x_t|y_{1:t})$ を近似する粒子を $x_t^{(i)}$, $i = 1, \ldots, L$ とする．式 (4.5) より，時刻 t の状態の予測分布は

$$\begin{aligned}
p(x_t|y_{1:t-1}) &= \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \\
&= \int \left\{ \int p(x_t, w_t|x_{t-1})dw_t \right\} p(x_{t-1}|y_{1:t-1})dx_{t-1} \\
&= \int \left\{ \int p(x_t|x_{t-1}, w_t)p(w_t|x_{t-1})dw_t \right\} p(x_{t-1}|y_{1:t-1})dx_{t-1} \\
&= \int \left\{ \int \delta\left(x_t - f_t(x_{t-1}, w_t)\right) p(w_t)dw_t \right\} p(x_{t-1}|y_{1:t-1})dx_{t-1} \qquad (4.12)
\end{aligned}$$

と書くことができる．ただし，$\delta(\cdot)$ はディラックのデルタ関数である．したがって，粒子フィルタにおける予測分布の計算は次で与えられる．

粒子フィルタによる予測分布の計算：

- 時刻 $t-1$ の状態のフィルタ分布 $p(x_{t-1}|y_{1:t-1})$ を近似する粒子 $x_{t-1}^{(i)}$, $i=1,\ldots,L$ を用意
- システム雑音の分布 $p(w_t)$ からの L 個のサンプル $w_t^{(i)}$, $i=1,\ldots,L$ を生成
- 時刻 t の状態の予測分布 $p(x_t|y_{1:t-1})$ を近似する粒子を次で計算

$$\hat{x}_t^{(i)} = f_t(x_{t-1}^{(i)}, w_t^{(i)}), \quad i=1,\ldots,L \tag{4.13}$$

次に，フィルタ分布の導出について考える．式 (4.6) より，時刻 t の状態のフィルタ分布は

$$\begin{aligned} p(x_t|y_{1:t}) &= \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t} \\ &\propto p(y_t|x_t)p(x_t|y_{1:t-1}) \end{aligned} \tag{4.14}$$

と書くことができる．ここで，計算したいものは左辺の $p(x_t|y_{1:t})$ に従うサンプルであるが，右辺の $p(x_t|y_{1:t-1})$ のサンプルはすでにもっていることに注意する．すなわち，$p(x_t|y_{1:t})$ がサンプルを生成したい所望の確率分布であり，$p(x_t|y_{1:t-1})$ がサンプルを生成可能な（ここではすでにサンプルをもっている）提案分布，観測モデルで与えられる $p(y_t|x_t)$ がそれらの値の比（重み）であると考えると，SIR のアルゴリズムがそのまま適用できる．したがって，粒子フィルタにおけるフィルタ分布の計算は次で与えられる．

粒子フィルタによるフィルタ分布の計算：

- 時刻 t の状態の予測分布 $p(x_t|y_{1:t-1})$ を近似する粒子 $\hat{x}_t^{(i)}$, $i=1,\ldots,L$ を用意
- 確率質量が

$$\frac{p(y_t|\hat{x}_t^{(i)})}{\sum_{n=1}^{L} p(y_t|\hat{x}_t^{(n)})} \tag{4.15}$$

である，$\hat{x}_t^{(i)}$, $i=1,\ldots,L$ 上の離散分布から L 回サンプリングすることで時刻 t の状態のフィルタ分布 $p(x_t|y_{1:t})$ を近似する粒子 $x_t^{(i)}$, $i=1,\ldots,L$ を得る．

2 番目のステップでは，一般状態空間モデルの観測モデルの式を用いて計算することになるが，非

線形・非ガウス型状態空間モデルの観測モデルの式を用いる場合には，

$$\begin{aligned} p(y_t|\hat{x}_t^{(i)}) &= \int p(y_t, v_t|\hat{x}_t^{(i)})dv_t \\ &= \int p(y_t|v_t, \hat{x}_t^{(i)})p(v_t|\hat{x}_t^{(i)})dv_t \\ &= \int \delta\left(y_t - h_t(\hat{x}_t^{(i)}, v_t)\right)p(v_t)dv_t \end{aligned}$$

となることから，$p(v_t)$ に従うサンプルを用いて $p(y_t|\hat{x}_t^{(i)})$ の値を計算すれば良い．なお，重点サンプリングや SIR では提案分布と興味のある分布が近い場合にその計算効率が改善されるが，粒子フィルタにおいてはそれらはそれぞれ時刻 t の状態の予測分布とフィルタ分布であり，いずれも時刻 t の状態の分布となっており，これらは近い分布であることが期待される．

4.4　カルマンフィルタ

カルマンフィルタでは，線形・ガウス型の状態空間モデル

$$\begin{aligned} x_t &= F_t x_{t-1} + G_t w_t \\ y_t &= H_t x_t + v_t \end{aligned}$$

を考える．ただし，システム雑音 W_t と観測雑音 V_t はいずれも白色ガウス雑音であり，$E[W_t] = 0$, $E[W_t W_t^T] = Q_t$, $E[V_t] = 0$, $E[V_t V_t^T] = R_t$ とする．初期状態がガウス分布に従うと仮定するとすべての状態 x_t および観測 y_t はガウス分布に従う．また，予測分布 $p(x_t|y_{1:t-1})$ とフィルタ分布 $p(x_t|y_{1:t})$ のいずれもガウス分布となるため，線形・ガウス型の状態空間モデルでは状態 x_t の条件付き期待値と条件付き共分散行列のみを追いかければよい．すなわち，条件付き期待値と条件付き共分散行列を

$$E[X_t|y_{1:s}] = \bar{x}_{t|s} \tag{4.16}$$

$$E[(X_t - \hat{x}_{t|s})(X_t - \bar{x}_{t|s})^T|y_{1:s}] = P_{t|s} \tag{4.17}$$

と定義すると

$$p(x_t|y_{1:t-1}) = \mathcal{N}(\bar{x}_{t|t-1}, P_{t|t-1}) \tag{4.18}$$

$$p(x_t|y_{1:t}) = \mathcal{N}(\bar{x}_{t|t}, P_{t|t}) \tag{4.19}$$

$$\tag{4.20}$$

であり，予測のステップでは $\bar{x}_{t-1|t-1}$, $P_{t-1|t-1}$ から $\bar{x}_{t|t-1}, P_{t|t-1}$ を，フィルタリングのステップでは $\bar{x}_{t|t-1}$, $P_{t|t-1}$ から $\bar{x}_{t|t}, P_{t|t}$ を，それぞれ更新することになる．具体的な更新の手続きを以下に示す．

カルマンフィルタ：
観測 y_t を受け取ったときのカルマンフィルタの処理は以下のとおり.

- フィルタリング：

$$K_t = P_{t|t-1}H_t^T(H_tP_{t|t-1}H_t^T + R_t)^{-1} \tag{4.21}$$
$$\bar{x}_{t|t} = \bar{x}_{t|t-1} + K_t(y_t - H_t\bar{x}_{t|t-1}) \tag{4.22}$$
$$P_{t|t} = P_{t|t-1} - K_tH_tP_{t|t-1} \tag{4.23}$$

- 予測：

$$\bar{x}_{t+1|t} = F_t\bar{x}_{t|t} \tag{4.24}$$
$$P_{t+1|t} = F_tP_{t|t}F_t^T + G_tQ_tG_t^T \tag{4.25}$$

第5章　確率伝搬法

確率伝搬法は前章で考えた状態空間モデルよりも広いクラスの確率モデルにおける確率推論問題を解くことが可能な手法である．ここでは離散の確率変数を仮定し，確率推論問題の定式化および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．具体的な確率伝搬法のアルゴリズムとして，ファクターグラフ上のメッセージ伝搬アルゴリズムである和-積アルゴリズムと，ベイジアンネットワーク上のアルゴリズムである Perl の BP アルゴリズムの両方について述べ，それらの関係について簡単な例を用いて説明する．

5.1　確率伝搬法の原理

離散確率変数 $X_1, X_2, \ldots, X_n$ の同時確率分布は，状態空間モデルの導出のときと同様に確率の積法則を繰り返し用いることで,

$$\begin{aligned} P(x_1, \ldots, x_n) &= P(x_1)P(x_2, \ldots, x_n|x_1) \\ &= P(x_1)P(x_2|x_1)P(x_3, \ldots, x_n|x_1, x_2) \\ &\vdots \\ &= P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\cdots P(x_n|x_1, x_2, \ldots, x_{n-1}) \end{aligned} \tag{5.1}$$

と書くことができる．この変形だけでは周辺事後確率を求める際の計算量の削減に繋がらないが, 多くの工学の問題において (5.1) の条件付き分布の条件は必ずしもその全てが有効というわけではない（考慮すべき確率変数の中に独立なのものも存在する）ため，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．前章の状態空間モデルでは，その特別な場合として状態と観測のマルコフ性を利用したが，ここではより一般の依存関係を仮定したときの確率推論について考える．なお，状態空間モデルでは観測されない確率変数 (状態) と観測される確率変数が一対一対応になっていたため，それらを区別して X と Y という記号をそれぞれの確率変数に使用していたが，本章ではそのような対応関係は仮定しないため，確率変数の記号として X のみを使用し，それらのうちのいずれかが観測されるという状況を考える．

簡単な例として 5 つの離散確率変数 $X_1, \ldots, X_5$ の同時分布を考える．仮に 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 が条件付き独立であるとすると，これらの確率変数の同時分布は

$$\begin{aligned} &P(x_1, x_2, x_3, x_4, x_5) \\ &= P(x_1)P(x_2|x_1)P(x_4|x_1, x_2)P(x_3|x_1, x_2, x_4)P(x_5|x_1, x_2, x_3, x_4) \\ &= P(x_1)P(x_2|x_1)P(x_4|x_1)P(x_3|x_4)P(x_5|x_4) \end{aligned} \tag{5.2}$$

となる．ここで，確率推論問題の具体例として $X_3 = a_3$ が観測されたときの確率変数 X_1 の周辺事

後分布の計算を考えると

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

となる．ただし，α は定数 $\alpha = \frac{1}{\Pr(X_3=a_3)}$ である．これより，事後周辺分布を同時分布から直接計算した場合には，各 X_i が取りうる値が q 通りあるとして 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, これが以下で述べる和-積アルゴリズムや Perl の BP アルゴリズムなどの確率伝搬法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 [34] を参照されたい．

5.2　和-積 アルゴリズム

和-積 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

$$f(x_1, x_2, \ldots, x_n) = f_1(A_1)f_2(A_2)\cdots f_m(A_m) \tag{5.3}$$

の周辺化関数

$$g_r(x_r) = \sum_{x_1,\ldots,x_n \backslash x_r} f(x_1, \ldots, x_n) \tag{5.4}$$

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, $A_1, A_2, \ldots, A_m$ は $\{x_1, \ldots, x_n\}$ の部分集合であり，$x_1, \ldots, x_n \backslash x_r$ は変数 x_r 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. $f(x_1, x_2, \ldots, x_n)$ 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 を根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側のノードの順に，以下に説明する各ノードでの処理ルールに従って計算する

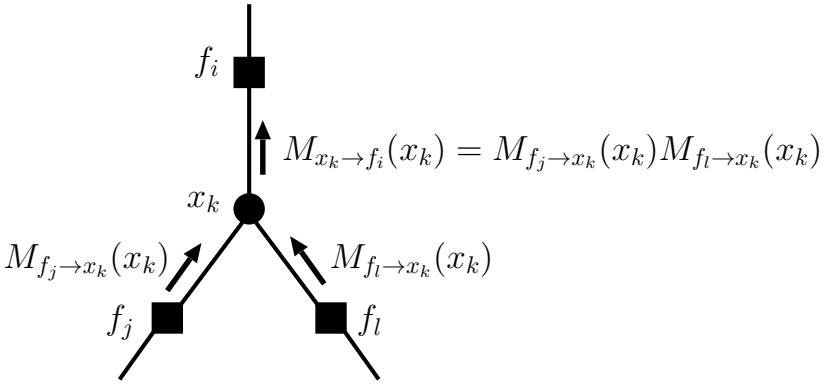


図 5.1: 和-積 アルゴリズム（変数ノードでの処理）

変数ノードでの処理：
変数ノード x_k からそれに直接つながった関数ノード f_i へ送るメッセージは次のように計算される（図 5.1）

$$M_{x_k \to f_i}(x_k) = \prod_{a \in N(x_k) \setminus f_i} M_{a \to x_k}(x_k) \tag{5.5}$$

ただし, $N(x_k)$ は変数ノード x_k に直接つながった関数ノードの集合であり, $M_{a \to x_k}(x_k)$ は関数ノード a から x_k に届いたメッセージを表す. x_k が葉のときは

$$M_{x_k \to f_i}(x_k) = 1 \tag{5.6}$$

とする.

関数ノードでの処理：
関数ノード f_i からそれに直接つながった変数ノード x_k へのメッセージは

$$M_{f_i \to x_k}(x_k) = \sum_{A_i \setminus x_k} \left(f_i(A_i) \prod_{a \in A_i \setminus x_k} M_{a \to f_i}(a) \right) \tag{5.7}$$

と計算される (図 5.2). f_i が葉のときは

$$M_{f_i \to x_k}(x_k) = f_i(x_k) \tag{5.8}$$

とする.

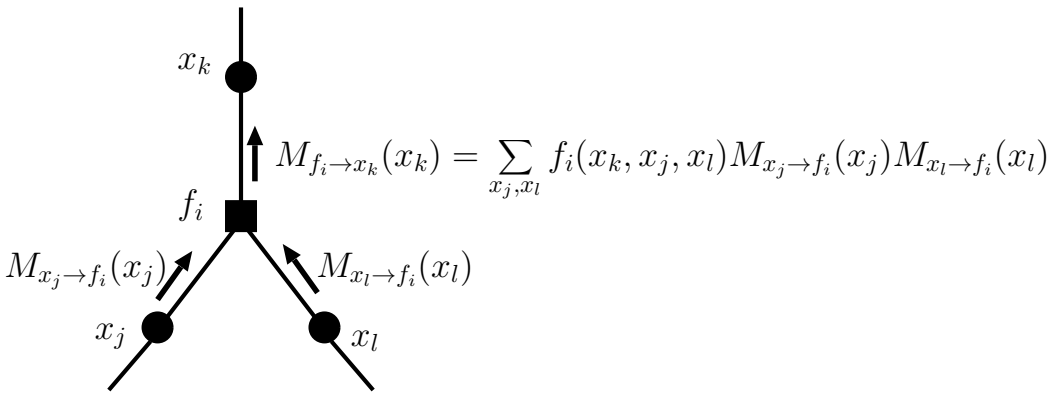


図 5.2: 和-積 アルゴリズム（関数ノードでの処理）

ルートノードでの処理：
最後にルートノードで

$$M_{x_r}(x_r) = \prod_{a \in N(x_r)} M_{a \to x_r}(x_r) = g_r(x_r) \tag{5.9}$$

と計算することで, 所望の周辺化関数 $g_r(x_r)$ を得る

ファクターグラフにループが存在しない場合, 上述の和-積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 [4] にならって解説する.

求めたい周辺化関数を

$$g_r(x_r) = \sum_{x_1, \ldots, x_n \setminus x_r} f(x_1, \ldots, x_n) \tag{5.10}$$

とする．変数ノード x_r に直接つながっている関数ノードを $a \in N(x_r)$ とし，関数ノード a を介して変数ノード x_r につながっている全ての変数の集合を B_a とすると（図 5.3 参照）, 多変数関数 $f(x_1, \ldots, x_n)$ は

$$f(x_1, \ldots, x_n) = \prod_{a \in N(x_r)} F_a(x_r, B_a) \tag{5.11}$$

という因数分解の形で書ける．ファクターグラフが木構造をもつことから，$F_a(x_r, B_a)$, $a \in N(x_r)$ は x_r のみを共通の引数としてもつことに注意する．このとき，

$$\begin{aligned} g_r(x_r) &= \sum_{x_1, \ldots, x_n \setminus x_r} \prod_{a \in N(x_r)} F_a(x_r, B_a) \\ &= \prod_{a \in N(x_r)} \left(\sum_{B_a} F_a(x_r, B_a) \right) \\ &= \prod_{a \in N(x_r)} M_{a \to x_r}(x_r) \end{aligned} \tag{5.12}$$

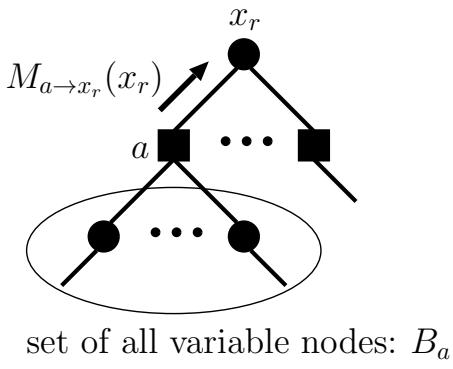


図 5.3: 和-積 アルゴリズムのしくみ 1

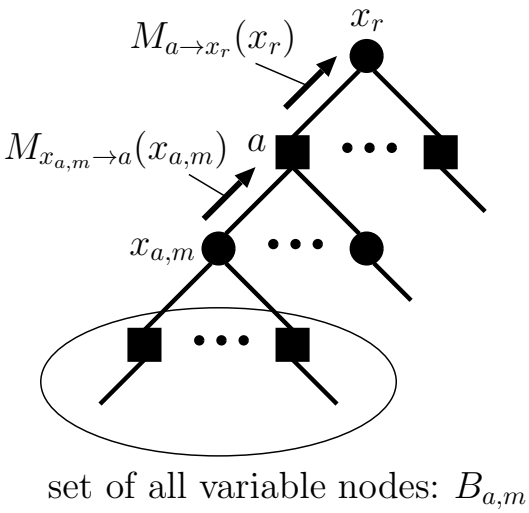


図 5.4: 和-積 アルゴリズムのしくみ 2

となる. ただし,

$$M_{a\to x_r}(x_r) \stackrel{def}{=} \sum_{B_a} F_a(x_r, B_a) \tag{5.13}$$

であり, $M_{a\to x_r}(x_r)$ は関数ノード a 以下の情報のみから計算される. つまり, $M_{a\to x_r}(x_r)$ は関数ノード a から変数ノード x_r に送られるメッセージであると考えられ, (5.12) はルートノードでの処理を表している.

さらに, 関数ノード a に直接接続する x_r 以外の変数ノードを $x_{a,1}, ..., x_{a,M}$ とし, $x_{a,m}$ を介して a につながっている全ての変数の集合を $B_{a,m}$ とすると（図 5.4 参照）

$$F_a(x_r, B_a) = a(x_r, x_{a,1}, \ldots, x_{a,M}) G_{a,1}(x_{a,1}, B_{a,1}) \cdots G_{a,M}(x_{a,M}, B_{a,M}) \tag{5.14}$$

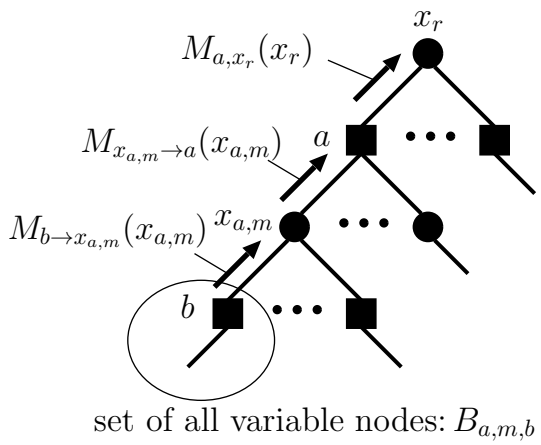


図 5.5: 和-積 アルゴリズムのしくみ 3

と書くことができる. これより,

$$\begin{aligned} M_{a\to x_r}(x_r) &= \sum_{B_a} a(x_r, x_{a,1}, \ldots, x_{a,M}) G_{a,1}(x_{a,1}, B_{a,1}) \cdots G_{a,M}(x_{a,M}, B_{a,M}) \\ &= \sum_{B_a} a(x_r, x_{a,1}, \ldots, x_{a,M}) \prod_{m=1}^{M} G_{a,m}(x_{a,m}, B_{a,m}) \\ &= \sum_{x_{a,1},\ldots,x_{a,M}} a(x_r, x_{a,1}, \ldots, x_{a,M}) \prod_{m=1}^{M} \sum_{B_{a,m}} G_{a,m}(x_{a,m}, B_{a,m}) \\ &= \sum_{x_{a,1},\ldots,x_{a,M}} a(x_r, x_{a,1}, \ldots, x_{a,M}) \prod_{x_{a,m}\in N(a)\setminus x_r} M_{x_{a,m}\to a}(x_{a,m}) \end{aligned} \tag{5.15}$$

となる. ここで,

$$M_{x_{a,m}\to a}(x_{a,m}) \stackrel{def}{=} \sum_{B_{a,m}} G_{a,m}(x_{a,m}, B_{a,m}) \tag{5.16}$$

であり, これは変数ノード $x_{a,m}$ から関数ノード a に送られるメッセージである. (5.15) は関数ノード a での処理を表している.

変数ノード $x_{a,m}$ に関数ノード b, $(b \neq a)$ を介して接続している変数ノードの集合を $B_{a,m,b}$ とすると（図 5.5 参照）

$$G_{a,m}(x_{a,m}, B_{a,m}) = \prod_{b\in N(x_{a,m})\setminus a} F_{a,b}(x_{a,m}, B_{a,m,b}) \tag{5.17}$$

と書けるので

$$
\begin{aligned}
M_{x_{a,m}\to a}(x_{a,m}) &= \sum_{B_{a,m}} \prod_{b\in N(x_{a,m})\backslash a} F_{a,b}(x_{a,m}, B_{a,m,b}) \\
&= \prod_{b\in N(x_{a,m})\backslash a} \sum_{B_{a,m,b}} F_{a,b}(x_{a,m}, B_{a,m,b}) \\
&= \prod_{b\in N(x_{a,m})\backslash a} M_{b\to x_{a,m}}(x_{a,m}) \qquad (5.18)
\end{aligned}
$$

となる. これは変数ノード $x_{a,m}$ 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.

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完全には分かっていない. 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付けの試みおよび対応策の例としては, 密度発展法 [36], ガウス近似法 [37], EXIT チャート法 [38], 情報幾何に基く解釈 [23, 25], ベーテ自由エネルギーと和-積 アルゴリズムの停留点の関連の指摘 [39], 一般化確率伝搬法 [41] などが挙げられる.

5.3 Pearl の BP アルゴリズム

確率伝搬法のアルゴリズムとして, 5.2 節ではファクターグラフ上で定義される和-積アルゴリズムを説明した. これは和-積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 [2] 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和-積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.

和-積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 5.6 のように複数の親 $u_1, ..., u_M$ 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 に届くメッセージを $\pi_{u_i\to x}(u_i)$ とし, 子 y_j から x に届くメッセージを $\lambda_{y_j\to x}(x)$ とする.

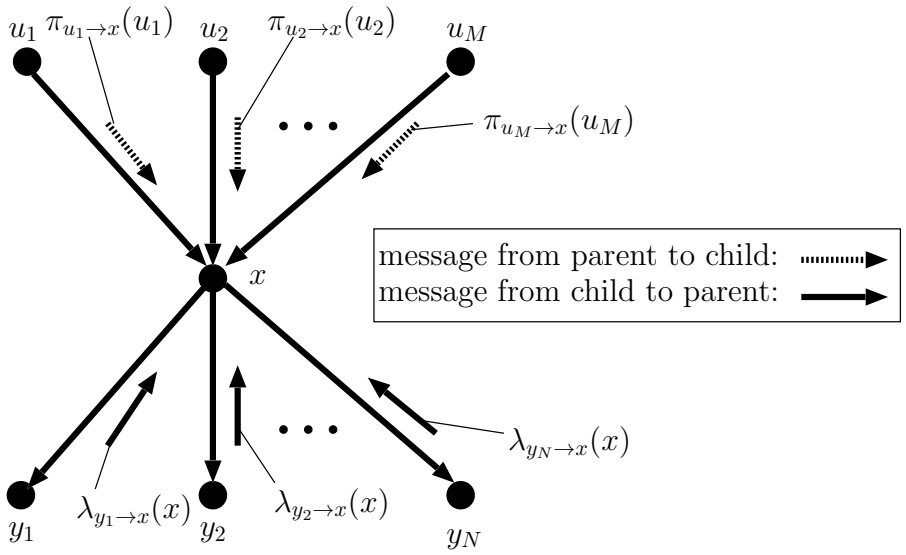


図 5.6: Pearl の BP アルゴリズム

x が親 u_i に送るメッセージ：

$$\lambda_{x\to u_i}(u_i) = \sum_{u_1,\ldots,u_M\setminus u_i}\left(\sum_x \left(P(x|u_1,\ldots,u_M)\prod_{k=1}^{N}\lambda_{y_k\to x}(x)\right)\prod_{k=1\setminus i}^{M}\pi_{u_k\to x}(u_k)\right) \tag{5.19}$$

x が子 y_j に送るメッセージ：

$$\pi_{x\to y_j}(x) = \left(\prod_{k=1\setminus j}^{N}\lambda_{y_k\to x}(x)\right)\sum_{u_1,\ldots,u_M}\left(P(x|u_1,\ldots,u_M)\prod_{k=1}^{M}\pi_{u_k\to x}(u_k)\right) \tag{5.20}$$

Pearl の BP アルゴリズムと和-積アルゴリズムの関係を確認するために, 図 5.7 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 $\lambda_{x\to u_1}(u_1)$, $\pi_{x\to y_1}(x)$ 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

$$\begin{aligned}\lambda_{x\to u_1}(u_1) &= \sum_{u_2}\left(\sum_x P(x|u_1,u_2)\lambda_{y_1\to x}(x)\lambda_{y_2\to x}(x)\right)\pi_{u_2\to x}(u_2)\\ &= \sum_x \lambda_{y_1\to x}(x)\lambda_{y_2\to x}(x)\sum_{u_2}P(x|u_1,u_2)\pi_{u_2\to x}(u_2)\end{aligned} \tag{5.21}$$

及び

$$\pi_{x\to y_1}(x) = \lambda_{y_2\to x}(x)\sum_{u_1,u_2}P(x|u_1,u_2)\pi_{u_1\to x}(u_1)\pi_{u_2\to x}(u_2)$$

となる. 一方, 図 5.7 の同時確率分布は

$$P(u_1,u_2,x,y_1,y_2) = P(x|u_1,u_2)P(y_1|x)P(y_2|x)P(u_1)P(u_2) \tag{5.22}$$

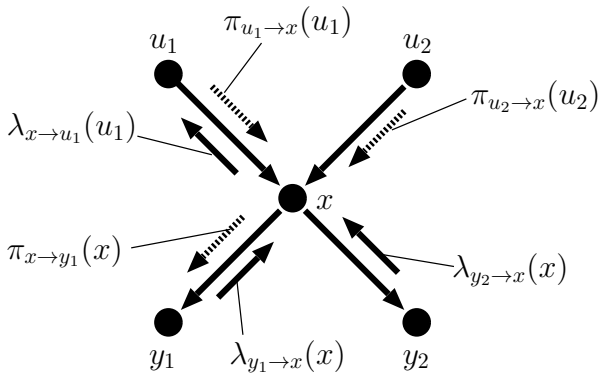


図 5.7: Pearl の BP アルゴリズムの例

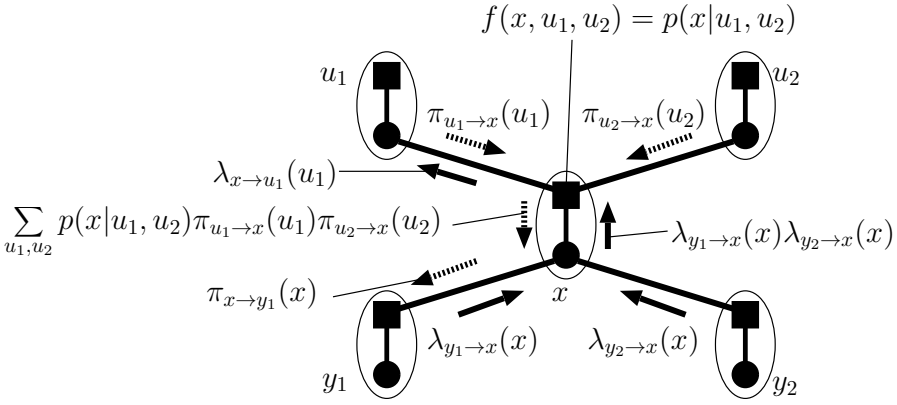


図 5.8: Pearl の BP アルゴリズムの例（ファクターグラフ表現）

となるのでこれをファクターグラフ表現したものが図 5.8 である. 和-積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 $\lambda_{y_1\to x}(x)\lambda_{y_2\to x}(x)$ 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 $\sum_{u_1,u_2} P(x|u_1,u_2)\pi_{u_1\to x}(u_1)\pi_{u_2\to x}(u_2)$ となる. よって図 5.8 における $\lambda_{x\to u_1}(u_1)$ 及び $\pi_{x\to y_1}(x)$ はそれぞれ

$$\begin{aligned}\lambda_{x\to u_1}(u_1) &= \sum_{x,u_2} P(x|u_1,u_2)\lambda_{y_1\to x}(x)\lambda_{y_2\to x}(x)\pi_{u_2\to x}(u_2)\\ &= \sum_{x} \lambda_{y_1\to x}(x)\lambda_{y_2\to x}(x)\sum_{u_2} P(x|u_1,u_2)\pi_{u_2\to x}(u_2) \qquad (5.23)\end{aligned}$$

$$\pi_{x\to y_1}(x) = \lambda_{y_2\to x}(x)\sum_{u_1,u_2} P(x|u_1,u_2)\pi_{u_1\to x}(u_1)\pi_{u_2\to x}(u_2) \qquad (5.24)$$

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.

第6章　確率伝搬法の応用例

6.1　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和-積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 [17]. しかしながら, その後 1990 年代に D. J. C. Mackay によって再発見 [33] （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

$$\mathbf{c} = \begin{bmatrix} c_1 & \cdots & c_{K+N} \end{bmatrix}^T = \mathbf{Gu} \tag{6.1}$$

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

$$\mathbf{c}' = \mathbf{c} + \mathbf{e} \tag{6.2}$$

となるが, これに検査行列をかけると

$$\mathbf{Hc}' = \mathbf{Hc} + \mathbf{He} = \mathbf{HGu} + \mathbf{He} = \mathbf{He} \tag{6.3}$$

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

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{6.4}$$

で与えられるとすると $\mathbf{Hc} = \mathbf{0}$ より, この符号は図 6.1 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 6.2 となる. ここに, 前節で説明した和-積

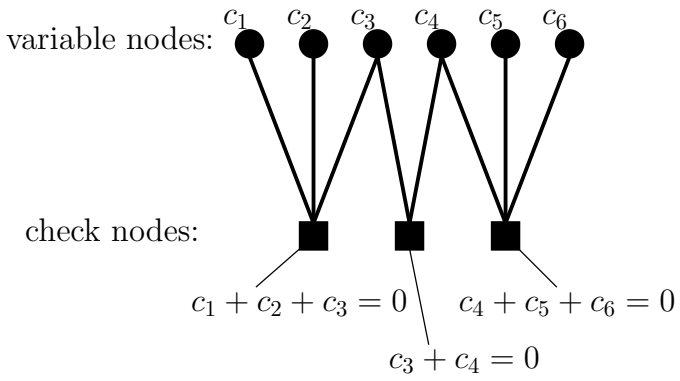


図 6.1: LDPC 符号の例（タナーグラフ表現）

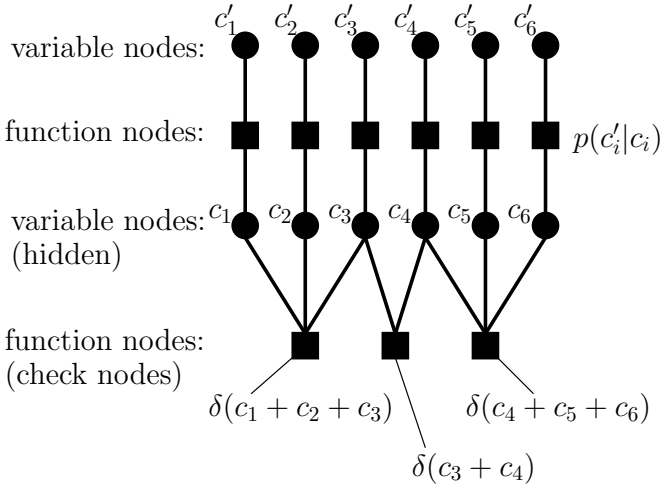


図 6.2: LDPC 符号の例（ファクターグラフ表現）

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 [10, 11]. 一方, タナーグラフに短いループが存在する場合には, 和-積アルゴリズムによって良い特性が得られないが, LDPC 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, 和-積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 [33].

6.2　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

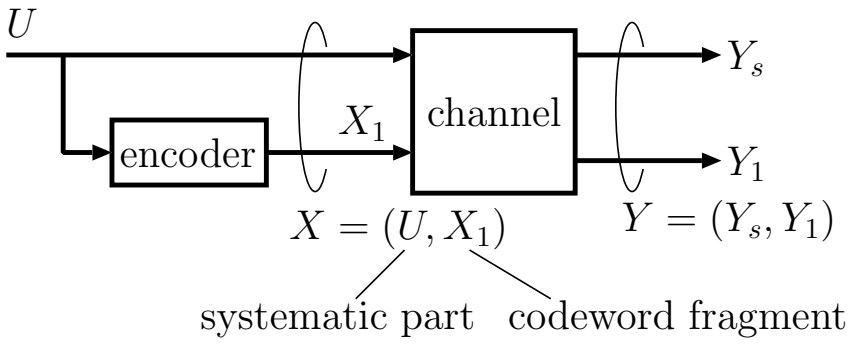


図 6.3: 組織符号

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 [19].

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.

$U = (U_1 \dots U_K)$ をシンボルからなる長さ K の情報語（を表す確率変数）とし, これを符号化することで得られる符号語（を表す確率変数）を 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 $X = (U, X_1)$ という形に書ける. 以下では, U と X_1 をそれぞれ符号語 X の組織部, 非組織部と呼ぶ. 符号語 X が通信路を通って受信されたものを $Y = (Y_s, Y_1)$ とする（図 6.3 参照）. ここで，Y_s 及び Y_1 はそれぞれ，X の組織部及び非組織部に対応する受信語である.

$P(y|x) \stackrel{def}{=} \Pr(Y = y|X = x)$ と定義し, 通信路を無記憶[1]と仮定すると条件付き密度関数は

$$\begin{aligned} P(y|x) &= P(y_s, y_1|u, x_1) \\ &= P(y_s|u)P(y_1|x_1) \\ &= \left(\prod_{i=1}^{K} P(y_{si}|u_i)\right) \cdot P(y_1|x_1) \end{aligned} \tag{6.5}$$

と書ける. ただし, y_{si} は y_s の i 番目の成分である.

Y_s, Y_1 の観測値 y_s, y_1 を得たときのシンボル毎最大事後確率復号は周辺事後確率

$$\mathrm{BEL}_i(a) \stackrel{def}{=} \Pr\{U_i = a|Y_s = y_s, Y_1 = y_1\} \tag{6.6}$$

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,

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

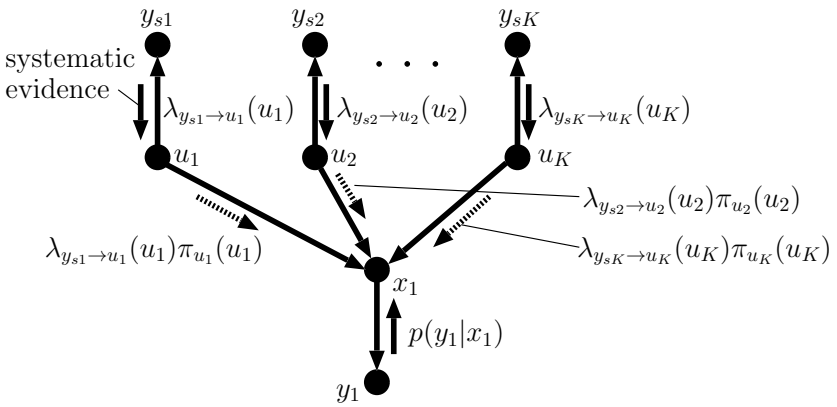


図 6.4: 組織符号のベイジアンネットワーク表現

尤度 $P(y_{si}|u_i)$ を $\lambda_{y_{si}\to u_i}(u_i)$, 事前確率 $\Pr(U_i = a)$ を $\pi_{u_i}(a)$ とし, (6.5) を用いると

$$\begin{aligned}\mathrm{BEL}_i(a) &= \alpha \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1) \prod_{j=1}^{K} \lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}(u_j) \\ &= \alpha \lambda_{y_{si}\to u_i}(a)\pi_{u_i}(a) \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1) \prod_{\substack{j=1 \\ j\neq i}}^{K} \lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}(u_j) \end{aligned} \tag{6.7}$$

となる. ただし, α は Pearl の α 表記と呼ばれ [2], 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. (6.7) の $\lambda_{y_{si}\to u_i}(a)$ 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, $\pi_{u_i}(a)$ 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 6.4 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 $\lambda_{y_{si}\to u_i}(u_i) = P(y_{si}|u_i)$ 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 y_1 から x_1 に $P(y_1|x_1)$ 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 $\lambda_{y_{si}\to u_i}(u_i)$ に自身が持つ事前確率 $\pi_{u_i}(u_i)$ を乗算したものをメッセージとして x_1 に送る. 次に x_1 が各 u_i に送るメッセージは,

$$\sum_{u_1,\ldots,u_K \setminus u_i = a} (P(x_1|u)P(y_1|x_1)) \prod_{\substack{j=1 \\ j\neq i}}^{K} \lambda_{ysj\to u_j}(u_j)\pi_{u_j}(u_j) = \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1) \prod_{\substack{j=1 \\ j\neq i}}^{K} \lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}(u_j) \tag{6.8}$$

となる. ここで, u と 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

$$\alpha \lambda_{y_{si}\to u_i}(a)\pi_{u_i}(a) \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1) \prod_{\substack{j=1 \\ j\neq i}}^{K} \lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}(u_j) \tag{6.9}$$

を得る. これは所望の事後周辺分布 (6.7) に一致している.

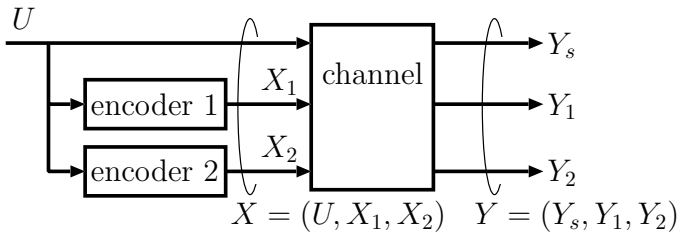


図 6.5: 並列連接組織符号（ターボ符号）

次に, 図 6.5 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. X_1, X_2 は 2 つの符号器の出力を表す確率変数であり, 符号語 $X = (U, X_1, X_2)$ の非組織部分である. また, Y_1, Y_2 は X_1, X_2 に対応する受信語であるとする. 無記憶通信路を仮定することで

$$\begin{aligned} P(y|x) &= P(y_s, y_1, y_2|u, x_1, x_2) \\ &= P(y_s|u)P(y_1|x_1)P(y_2|x_2) \\ &= \left(\prod_{i=1}^{K} P(y_{si}|u_i)\right) P(y_1|x_1)P(y_2|x_2) \end{aligned} \tag{6.10}$$

と書ける. さらに, 組織符号のときと同様に $\lambda_{y_{si}\to u_i}(u_i)$, $\pi_{u_i}(u_i)$ 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

$$\begin{aligned} \mathrm{BEL}_i(a) &= \alpha \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1)P(y_2|x_2) \prod_{j=1}^{K} \lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}(u_j) \\ &= \alpha\lambda_{y_{si}\to u_i}(a)\pi_{u_i}(a) \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1)P(y_2|x_2) \prod_{\substack{j=1 \\ j\neq i}}^{K} \lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}(u_j) \end{aligned} \tag{6.11}$$

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 (U, X_1) 及び (U, 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 $P(u_i) = \pi_{u_i}^{(2m-2)}(u_i)$ として以下の周辺事後確率に基づく符号語 (U, X_1) の最大事後確率復号を行なう

$$\begin{aligned} \Pr\{U_i = a|Y_s = y_s, Y_1 = y_1\} &= \alpha \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1) \prod_{j=1}^{K} \lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}^{(2m-2)}(u_j) \\ &= \alpha\lambda_i(a)\pi_{u_i}^{(2m-2)}(a) \sum_{u_1,\ldots,u_K \setminus u_i = a} P(y_1|x_1) \prod_{\substack{j=1 \\ j\neq i}}^{K} \lambda_{y_{sj}\to}(u_j)\pi_{u_j}^{(2m-2)}(u_j) \end{aligned} \tag{6.12}$$

ここで $\pi_{u_i}^{(2m-2)}(u_i)$ は 2m-2 番目の (U, 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. (6.12) の外部値（(6.12) 右辺の $\alpha\lambda_i(a)\pi_{u_i}^{(2m-2)}(a)$) 以外の部分）を

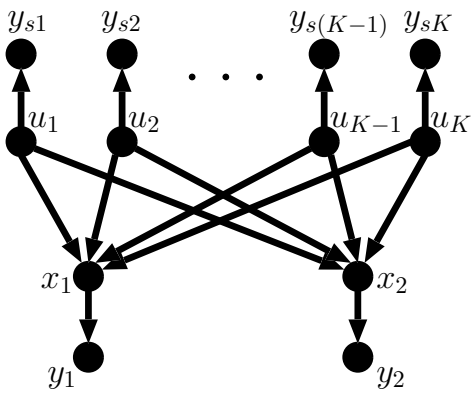


図 6.6: ターボ符号のベイジアンネットワーク表現

$\pi_{u_i}^{(2m-1)}(u_i)$ として次のステップに受け渡す. 2m 番目のステップでは $P(u_i) = \pi_{u_i}^{(2m-1)}(u_i)$ として (U, X_2) の周辺事後確率

$$\Pr(U_i = a|Y_s = y_s, Y_2 = y_2) = \alpha\lambda_{y_{si}\to u_i}(a)\pi_{u_i}^{(2m-1)}(a) \sum_{u_1,\ldots,u_K\setminus u_i=a} P(y_2|x_2)\prod_{\substack{j=1\\ j\neq i}}^{K}\lambda_{y_{sj}\to u_j}(u_j)\pi_{u_j}^{(2m-1)}(u_j) \tag{6.13}$$

を計算し, その外部値を次のステップでの事前確率 $P(u_i) = \pi_{u_i}^{(2m)}(u_i)$ 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 (U, X_1) 及び (U, 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 6.6 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 x_1 からノード u_i に届くメッセージを $\lambda_{x_1\to u_i}(u_i) = 1$, ノード x_2 からノード u_i に届くメッセージを $\lambda_{x_2\to u_i}(u_i) = 1$ とする. 一方, ノード y_{si} 及び y_1, y_2 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 $\lambda_{y_{si}\to u_i}(u_i)$, $P(y_1|x_1)$ 及び $P(y_2|x_2)$ をそれぞれ送る. また, ノード u_i が持つ事前確率 $\pi_{u_i}(u_i)$ 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 x_1 及び x_2 に送信する (ノード u のアクティベーションという). ここで, u_i から x_1 及び x_2 へのメッセージはいずれも $\alpha\lambda_{y_{si}\to u_i}(u_i)$ である. この時点でノード x_1 と 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 x_1 のアク

ティベーションを行なうことにすると, x_1 から u_i へのメッセージは

$$\alpha \sum_{u_1,\ldots,u_K \backslash u_i} \left(\sum_{x_1} P(x_1|u) P(y_1|x_1) \right) \prod_{\substack{j=1 \\ j \neq i}}^{K} \lambda_{y_{sj} \to u_j}(u_j) = \alpha \sum_{u_1,\ldots,u_K \backslash u_i} P(y_1|x_1) \prod_{\substack{j=1 \\ j \neq i}}^{K} \lambda_{y_{sj} \to u_j}(u_j) \tag{6.14}$$

と計算される. これを $\pi_{u_i}^{(1)}$ とする（実際, ターボ復号アルゴリズムの外部値 $\pi_{u_i}^{(1)}$ に一致している）. これより, ノード u に届くメッセージの一部が更新されたので次は u のアクティベーションを行なう. 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 x_2 宛のものだけであり, $\alpha \lambda_{y_{si} \to u_i}(u_i) \pi_{u_i}^{(1)}$ となる. 続いて, x_2 のアクティベーションを行なうと, x_2 から u_i へのメッセージは

$$\alpha \sum_{u_1,\ldots,u_K \backslash u_i} \left(\sum_{x_2} P(x_2|u) P(y_2|x_2) \right) \prod_{\substack{j=1 \\ j \neq i}}^{K} \lambda_{y_{sj} \to u_j}(u_j) \pi_{u_j}^{(1)} = \alpha \sum_{u_1,\ldots,u_K \backslash u_i} P(y_2|x_2) \prod_{\substack{j=1 \\ j \neq i}}^{K} \lambda_{y_{sj} \to u_j}(u_j) \pi_{u_j}^{(1)} \tag{6.15}$$

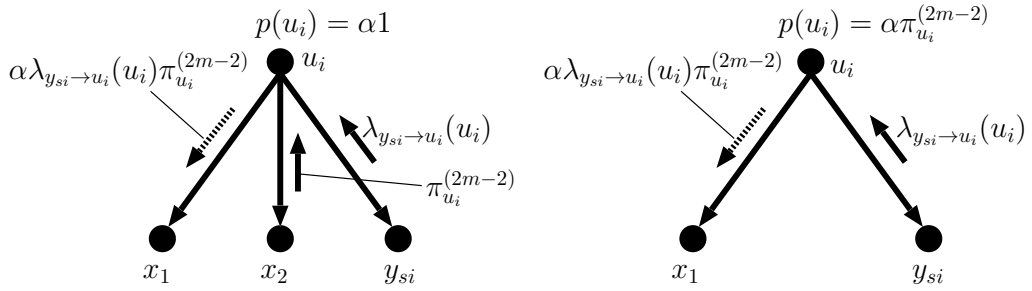
と計算され, これは外部値 $\pi_{u_i}^{(2)}$ に一致していることが分かる. 以下同様に, ノードのアクティベーションを $\mathbf{u} \to \mathbf{x}_1 \to \mathbf{u} \to \mathbf{x}_2 \to \cdots$ の順に繰り返していくと上記の初期化及び順序による Pearl の BP アルゴリズムはターボ復号のアルゴリズムそのものになっている.

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 6.6(a) ではなく図 6.6(b) 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 6.7 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 (U, X_1) の復号の際には x_2 見えておらず, ネットワークは図 6.7 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 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 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 x_2 のノードが省略されたネットワークでは x_2 から届くはずであったメッセージ $\pi_{u_i}^{(2m-2)}$ 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.

6.3 高速フーリエ変換（FFT）

次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）

$$W_k = \sum_{n=0}^{N-1} w(n) e^{-j \frac{2\pi}{N} nk} \tag{6.16}$$



(a) original Bayesian network　　(b) equivalent Bayesian network

図 6.7: 外部値を次段の事前値とする理由

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 (6.16) 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

$$\begin{aligned} W_k &= \sum_{n=0}^{\frac{N}{2}-1} w(2n)e^{-j\frac{2\pi}{N}2nk} + \sum_{n=0}^{\frac{N}{2}-1} w(2n+1)e^{-j\frac{2\pi}{N}(2n+1)k} \\ &= \sum_{n=0}^{\frac{N}{2}-1} w(2n)e^{-j\frac{2\pi}{\frac{N}{2}}nk} + e^{-j\frac{2\pi}{N}k}\sum_{n=0}^{\frac{N}{2}-1} w(2n+1)e^{-j\frac{2\pi}{\frac{N}{2}}nk} \end{aligned} \tag{6.17}$$

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 [35] 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 [34], [32]. 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

$$\begin{aligned} W_k &= \sum_{n=0}^{7} w(n)e^{-j\frac{2\pi}{8}nk} \\ &= \sum_{x_0,x_1,x_2} w(4x_2+2x_1+x_0)e^{-j\frac{2\pi}{8}(4x_2+2x_1+x_0)(4y_2+2y_1+y_0)} \\ &= \sum_{x_0,x_1,x_2} w(4x_2+2x_1+x_0)(-1)^{x_2y_0}(-1)^{x_1y_1}(-1)^{x_0y_2}(j)^{-x_0y_1}(j)^{-x_1y_0}e^{-j\frac{2\pi}{8}(x_0y_0)} \\ &= \sum_{x_0}(-1)^{x_0y_2}(j)^{-x_0y_1}e^{-j\frac{2\pi}{8}x_0y_0}\sum_{x_1}(-1)^{x_1y_1}(j)^{-x_1y_0}\sum_{x_2} w(4x_2+2x_1+x_0)(-1)^{x_2y_0} \end{aligned} \tag{6.18}$$

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図 6.8 のようなるがこれにはループが含まれているため, 和-積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

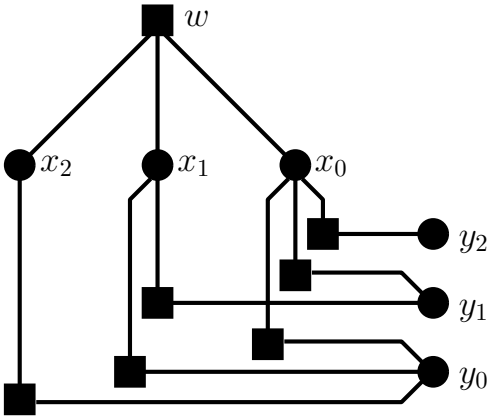


図 6.8: DFT のファクターグラフ表現

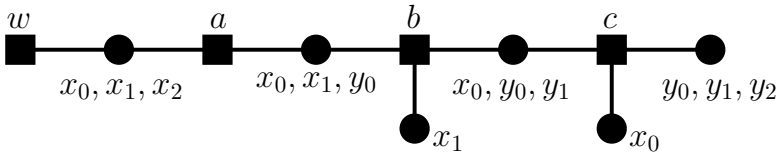


図 6.9: DFT のファクターグラフ表現（修正版）

とができない. そこで, 図 6.8 からスパニング木を構成し,

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

と関数を定義（クラスタリング）することで, 図 6.9 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 [32] などを参照). 変数ノード中の 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. これに和-積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.

6.4 近似メッセージ伝搬法（AMP）

本節では，ℓ_1 最適化 [45] と SOAV（Sum-of-Absolue-Value）最適化 [46] に対する近似メッセージ伝搬法 (Approximate Message Passing, AMP）のアルゴリズム [44], [47] の導出を行う．基本的な流れは，まず最適化問題に対する確率分布を構成し，対応するファクターグラフにおける確率伝搬法の和-積アルゴリズム [2], [32] を考える．そして，そのアルゴリズムに対して大システム極限をとった後にさらに近似を行うことで AMP アルゴリズムが得られる．

ℓ_1 最適化は，スパースなベクトル $\boldsymbol{b} = [b_1 \cdots b_N]^{\mathrm{T}} \in \mathbb{R}^N$ をその線形観測

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{b}, \tag{6.19}$$

から再構成するための手法である．ここで，$\boldsymbol{y} = [y_1 \cdots y_M]^{\mathrm{T}} \in \mathbb{R}^M$ および

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{M,1} & \cdots & a_{M,N} \end{bmatrix} \in \mathbb{R}^{M \times N} \tag{6.20}$$

である．ℓ_1 最適化では，最適化問題

$$\hat{\boldsymbol{b}} = \arg\min_{\boldsymbol{s} \in \mathbb{R}^N} \|\boldsymbol{s}\|_1 \quad \text{subject to} \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{s} \tag{6.21}$$

を解くことによりスパースベクトルの推定値を計算する．

一方，SOAV 最適化は，離散値をとるベクトル $\boldsymbol{b} = [b_1 \cdots b_N]^{\mathrm{T}} \in \{q_1, \ldots, q_L\}^N \subset \mathbb{R}^N$ を再構成するための手法である．簡単のため以下では二値ベクトルの場合を考え，$\boldsymbol{b} \in \{1, -1\}^N$ かつ各成分は独立に $\Pr(b_j = 1) = \Pr(b_j = -1) = 1/2$ $(j = 1, \ldots, N)$ に従うとする．$\boldsymbol{b} - \boldsymbol{1}$ や $\boldsymbol{b} + \boldsymbol{1}$ がおおよそ $N/2$ 個の零成分を持つことを利用して，SOAV 最適化では $\boldsymbol{b}$ の推定値

$$\hat{\boldsymbol{b}} = \arg\min_{\boldsymbol{s} \in \mathbb{R}^N} \frac{1}{2}\|\boldsymbol{s} - \boldsymbol{1}\|_1 + \frac{1}{2}\|\boldsymbol{s} + \boldsymbol{1}\|_1 \quad \text{subject to} \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{s} \tag{6.22}$$

を計算する．

ℓ_1 最適化と SOAV 最適化に対する AMP アルゴリズムは，最適化問題 (6.21) と (6.22) に対応する確率分布に対する確率伝搬法をそれぞれ近似することにより得られる．ℓ_1 最適化に対しては確率分布

$$\mu(\boldsymbol{s}) \propto \prod_{j=1}^{N} \exp\left(-\beta|s_j|\right) \prod_{\theta=1}^{M} \delta\left(y_\theta = \sum_{i=1}^{N} a_{\theta,i} s_i\right), \tag{6.23}$$

SOAV 最適化に対しては確率分布

$$\mu(\boldsymbol{s}) \propto \prod_{j=1}^{N} \exp\left\{-\beta\left(\frac{1}{2}|s_j - 1| + \frac{1}{2}|s_j + 1|\right)\right\} \prod_{\theta=1}^{M} \delta\left(y_\theta = \sum_{i=1}^{N} a_{\theta,i} s_i\right) \tag{6.24}$$

を考える．ここで，$\beta > 0$ であり，$\delta\left(y_\theta = \sum_{i=1}^{N} a_{\theta,i} s_i\right)$ は超平面 $y_\theta = \sum_{i=1}^{N} a_{\theta,i} s_i$ 上の Dirac 測度を表す．$\beta \to \infty$ の極限で，(6.23) と (6.24) の分布はそれぞれ最適化問題 (6.21) と (6.22) の解を台とする一様分布となるので，各変数 s_j $(j = 1, \ldots, N)$ に関する周辺化を確率伝搬法 [2] を用いて行うことにより (6.21) や (6.22) の解を得られる．例えば確率分布 (6.24) のファクターグラフは図 6.10 のようになる．よって，和-積アルゴリズム [32] におけるメッセージ計算は，ℓ_1 最適化の場合は

$$\nu_{j \to \theta}^{t+1}(s_j) \propto \exp\left(-\beta|s_j|\right) \prod_{\zeta \neq \theta} \hat{\nu}_{\zeta \to j}^{t}(s_j), \tag{6.25}$$

$$\hat{\nu}_{\theta \to j}^{t}(s_j) \propto \int \delta\left(y_\theta = \sum_{i=1}^{N} a_{\theta,i} s_i\right) \prod_{k \neq j} \nu_{k \to \theta}^{t}(s_k) \mathrm{d}\boldsymbol{s}_{\setminus j}, \tag{6.26}$$

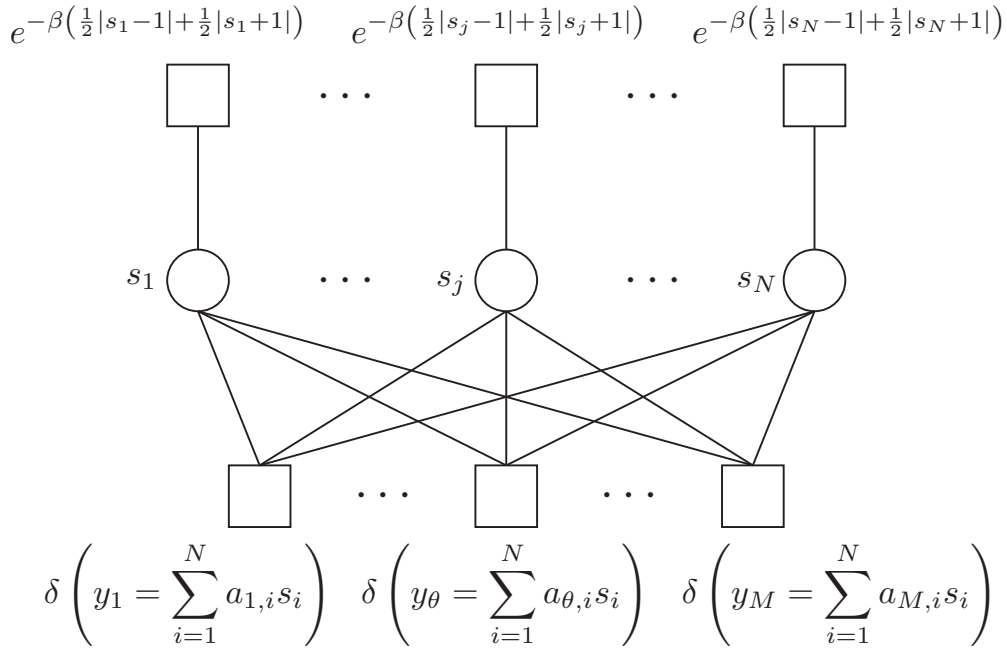


図 6.10: 確率分布 (6.24) のファクターグラフ：N 個の円は変数ノード $s_1,\ldots,s_N$ である．上側の N 個の四角は関数ノード $\exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)\right\}$ $(j=1,\ldots,N)$ を表し，下側の M 個の四角は関数ノード $\delta\left(y_\theta=\sum_{i=1}^N a_{\theta,i}s_i\right)$ $(\theta=1,\ldots,M)$ を表す．

SOAV 最適化の場合は

$$\nu_{j\to\theta}^{t+1}(s_j)\propto\exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)\right\}\prod_{\zeta\neq\theta}\hat{\nu}_{\zeta\to j}^{t}(s_j), \tag{6.27}$$

$$\hat{\nu}_{\theta\to j}^{t}(s_j)\propto\int\delta\left(y_\theta=\sum_{i=1}^N a_{\theta,i}s_i\right)\prod_{k\neq j}\nu_{k\to\theta}^{t}(s_k)\mathrm{d}\boldsymbol{s}_{\setminus j}, \tag{6.28}$$

と書ける．$\nu_{j\to\theta}^{t+1}(s_j)$ は変数ノード s_j から関数ノード $\delta\left(y_\theta=\sum_{i=1}^N a_{\theta,i}s_i\right)$ へのメッセージであり，$\hat{\nu}_{\theta\to j}^{t}(s_j)$ は関数ノードから変数ノードへのメッセージである．また，t はアルゴリズム中の繰り返しのインデックスを表し，$\int(\cdot)\mathrm{d}\boldsymbol{s}_{\setminus j}$ はすべての s_i $(i\neq j)$ に関して積分（周辺化）を行うことを意味する．

次に，大システム極限（$M,N\to\infty, M/N=\Delta$）を考えメッセージ計算 (6.25), (6.26)（(6.27), (6.28)）を近似する．以下では，計算を簡略化するために $a_{\theta,j}\in\{1/\sqrt{M},-1/\sqrt{M}\}$ とする．行列 $\boldsymbol{A}$ の成分が i.i.d. で平均 0，分散 $1/M$ であれば，同様のアルゴリズムが導出可能である．[47] の Lemma III.1 にあるように，式 (6.26)（(6.28)）の $\hat{\nu}_{\theta\to j}^{t}(s_j)$ に対応する確率分布関数は平均 $z_{\theta\to j}^{t}/a_{\theta,j}$，分散

$\hat{\tau}^t_{\theta\to j}/\beta a^2_{\theta,j}$ のガウス分布で近似できる．ただし，

$$z^t_{\theta\to j} = y_\theta - \sum_{i\neq j} a_{\theta,i} x^t_{i\to\theta} \tag{6.29}$$

$$\hat{\tau}^t_{\theta\to j} = \sum_{i\neq j} a^2_{\theta,i} \tau^t_{i\to\theta} \tag{6.30}$$

であり，$x^t_{i\to\theta}$ と $\tau^t_{i\to\theta}/\beta$ はそれぞれ $\nu^t_{i\to\theta}$ の平均と分散を表す．これを利用して

$$\hat{\nu}^t_{\theta\to j}(s_j) \approx \hat{\phi}^t_{\theta\to j}(s_j) \tag{6.31}$$

$$:= \sqrt{\frac{\beta a^2_{\theta,j}}{2\pi\hat{\tau}^t_{\theta\to j}}} \exp\left\{-\frac{\beta}{2\hat{\tau}^t_{\theta\to j}}(a_{\theta,j}s_j - z^t_{\theta\to j})^2\right\} \tag{6.32}$$

とすると，$\nu^t_{i\to\theta}(s_i)$ $(i\neq j)$ の平均と分散のみから $\hat{\nu}^t_{\theta\to j}(s_j)$ の近似値を計算できる．そこで，次に $\nu^{t+1}_{j\to\theta}(s_j)$ の平均と分散について考える．ℓ_1 最適化の場合は以下の確率分布

$$f_\beta(s;u,c) \propto \exp\left\{-\beta|s| - \frac{\beta}{2c}(s-u)^2\right\}, \tag{6.33}$$

SOAV 最適化の場合は以下の確率分布

$$f_\beta(s;u,c) \propto \exp\left\{-\beta\left(\frac{1}{2}|s-1| + \frac{1}{2}|s+1|\right) - \frac{\beta}{2c}(s-u)^2\right\} \tag{6.34}$$

に従う確率変数の平均と分散をそれぞれ $\mathsf{F}_\beta(u,c), \mathsf{G}_\beta(u,c)$ とおく．このとき，以下の補題が成り立つ．

Lemma 1. 式 (6.30) の $\hat{\tau}^t_{\theta\to j}$ が θ, j によらず一定であり，$\hat{\tau}^t_{\theta\to j} = \hat{\tau}^t$ と書けるとする．関数ノードから変数ノードへのメッセージが $\hat{\nu}^t_{\theta\to j}(s_j) = \hat{\phi}^t_{\theta\to j}(s_j)$ であったとすると，変数ノードから関数ノードへのメッセージは

$$\nu^{t+1}_{j\to\theta}(s_j) = \phi^{t+1}_{j\to\theta}(s_j)\left\{1 + \mathcal{O}\left(\frac{s_j^2}{M}\right)\right\} \tag{6.35}$$

と書ける．ここで，

$$\phi^{t+1}_{j\to\theta}(s_j) := f_\beta\left(s_j, \sum_{\zeta\neq\theta} a_{\zeta,j} z^t_{\zeta\to j}, \hat{\tau}^t\right) \tag{6.36}$$

である．特に，$\nu^{t+1}_{j\to\theta}(s_j)$ の平均と分散はそれぞれ

$$x^{t+1}_{j\to\theta} = \mathsf{F}_\beta\left(\sum_{\zeta\neq\theta} a_{\zeta,j} z^t_{\zeta\to j}, \hat{\tau}^t\right) \tag{6.37}$$

$$\tau^t_{j\to\theta} = \beta\mathsf{G}_\beta\left(\sum_{\zeta\neq\theta} a_{\zeta,j} z^t_{\zeta\to j}, \hat{\tau}^t\right) \tag{6.38}$$

と表せる．

Proof. ℓ_1 最適化の場合は [47] の Lemma III.2 により成り立つ．以下では SOAV 最適化の場合の証明を同様の方法で行う．式 (6.27) と式 (6.32) より，$\hat{\nu}^t_{\theta\to j}(s_j) = \hat{\phi}^t_{\theta\to j}(s_j), \hat{\tau}^t_{\theta\to j} = \hat{\tau}^t$ のとき

$$\nu^{t+1}_{j\to\theta}(s_j) \propto \exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)\right\}\prod_{\zeta\neq\theta}\hat{\nu}^t_{\zeta\to j}(s_j) \tag{6.39}$$

$$\propto \exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)-\sum_{\zeta\neq\theta}\frac{\beta}{2\hat{\tau}^t}\left(a_{\theta,j}s_j - z^t_{\zeta\to j}\right)^2\right\} \tag{6.40}$$

$$\propto \exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)-\frac{\beta}{2\hat{\tau}^t}\sum_{\zeta\neq\theta}\left(a^2_{\zeta,j}s_j^2 - 2a_{\zeta,j}s_j z^t_{\zeta\to j}\right)\right\} \tag{6.41}$$

$$\propto \exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)-\frac{\beta}{2\hat{\tau}^t}\left(\frac{M-1}{M}s_j^2 - 2s_j\sum_{\zeta\neq\theta}a_{\zeta,j}z^t_{\zeta\to j}\right)\right\} \tag{6.42}$$

となる．これを式 (6.36) の

$$\phi^{t+1}_{j\to\theta}(s_j) \propto \exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)-\frac{\beta}{2\hat{\tau}^t}\left(s_j-\sum_{\zeta\neq\theta}a_{\zeta,j}z^t_{\zeta\to j}\right)^2\right\} \tag{6.43}$$

$$\propto \exp\left\{-\beta\left(\frac{1}{2}|s_j-1|+\frac{1}{2}|s_j+1|\right)-\frac{\beta}{2\hat{\tau}^t}\left(s_j^2-2s_j\sum_{\zeta\neq\theta}a_{\zeta,j}z^t_{\zeta\to j}\right)\right\} \tag{6.44}$$

と比較すると式 (6.35) が成り立つことが分かる． □

$x^t_{j\to\theta}$ と $\tau^t_{j\to\theta}/\beta$ はそれぞれ $\nu^t_{j\to\theta}(s_j)$ の平均と分散なので，以下のようなメッセージ更新式が得られる．

$$x^{t+1}_{j\to\theta} = \mathsf{F}_\beta\left(\sum_{\zeta\neq\theta}a_{\zeta,j}z^t_{\zeta\to j}, \hat{\tau}^t\right), \tag{6.45}$$

$$z^t_{\theta\to j} = y_\theta - \sum_{i\neq j}a_{\theta,i}x^t_{i\to\theta}, \tag{6.46}$$

$$\hat{\tau}^{t+1}_{\theta\to j} = \frac{\beta}{M}\sum_{i\neq j}\mathsf{G}_\beta\left(\sum_{\zeta\neq\theta}a_{\zeta,i}z^t_{\zeta\to i}, \hat{\tau}^t\right). \tag{6.47}$$

さらに，$\hat{\tau}^{t+1}_{\theta\to j}$ を θ, j に依存しない値

$$\hat{\tau}^{t+1} = \frac{\beta}{M}\sum_{i=1}^{N}\mathsf{G}_\beta\left(\sum_{\zeta=1}^{M}a_{\zeta,i}z^t_{\zeta\to i}, \hat{\tau}^t\right) \tag{6.48}$$

で近似する．

式 (6.23), (6.24) の確率分布の最頻値がそれぞれ $\beta\to\infty$ の極限で最適化問題 (6.21), (6.22) の解となるため，$\beta\to\infty$ としたときの更新式 (6.45), (6.46), (6.48) の形を求める．具体的には，$\mathsf{F}_\beta(u,c)$

と $\mathsf{G}_\beta(u,c)$ の値を計算する．$\beta\to\infty$ のとき，F_β の値は式 (6.34) の $f_\beta(s;u,c)$ の指数部分の最大値に支配され，ℓ_1 最適化の場合は

$$\begin{aligned}&\mathsf{F}_\beta(u,c)\\ &\to \arg\min_{s\in\mathbb{R}}\left\{|s|+\frac{1}{2c}(s-u)^2\right\} &(6.49)\\ &=\eta(u,c) &(6.50)\end{aligned}$$

となる．ここで

$$\eta(u,c)=\begin{cases}u+c & (u<-c)\\ 0 & (-c\le u\le c)\\ u-c & (c<u)\end{cases} \tag{6.51}$$

は弱しきい値関数である．

SOAV 最適化の場合も同様にして

$$\begin{aligned}&\mathsf{F}_\beta(u,c)\\ &\to \arg\min_{s\in\mathbb{R}}\left\{\left(\frac{1}{2}|s-1|+\frac{1}{2}|s+1|\right)+\frac{1}{2c}(s-u)^2\right\} &(6.52)\\ &=\eta(u,c) &(6.53)\end{aligned}$$

となる．ここで

$$\eta(u,c)=\begin{cases}u+c & (u<-1-c)\\ -1 & (-1-c\le u\le -1)\\ u & (-1<u<1)\\ 1 & (1\le u\le 1+c)\\ u-c & (1+c<u)\end{cases} \tag{6.54}$$

は SOAV 最適化 (6.22) に対する弱しきい値関数と考えられる（図 6.11 参照）．分散 $\mathsf{G}_\beta(u,c)$ は平均 $\eta(u,c)$ の近傍での $f_\beta(s;u,c)$ の値を調べることにより得られる．例えば，SOAV 最適化に対する弱しきい値関数において $-1<u<1$ で $\eta(u,c)=u$ の場合，u の近傍の s に対しても $-1<s<1$ が成り立つ．よって式 (6.34) から

$$\begin{aligned}f_\beta(s;u,c)&\propto\exp\left\{-\beta-\frac{\beta}{2c}(s-u)^2\right\} &(6.55)\\ &\propto\exp\left\{-\frac{\beta}{2c}(s-u)^2\right\} &(6.56)\end{aligned}$$

となる．したがって $-1<u<1$ の場合は $f_\beta(s;u,c)$ はガウス分布として近似でき，その分散は $\mathsf{G}_\beta(u,c)=c/\beta$ となる．一方 $1\le u\le 1+c$ で $\eta(u,c)=1$ の場合 $f_\beta(s;u,c)$ はラプラス分布 $\exp\left(-\frac{\beta}{2}|s-1|\right)$ で近似でき，その分散は $\mathsf{G}_\beta(u,c)=8/\beta^2$ となる．同様に考えると，ℓ_1 最適化の場

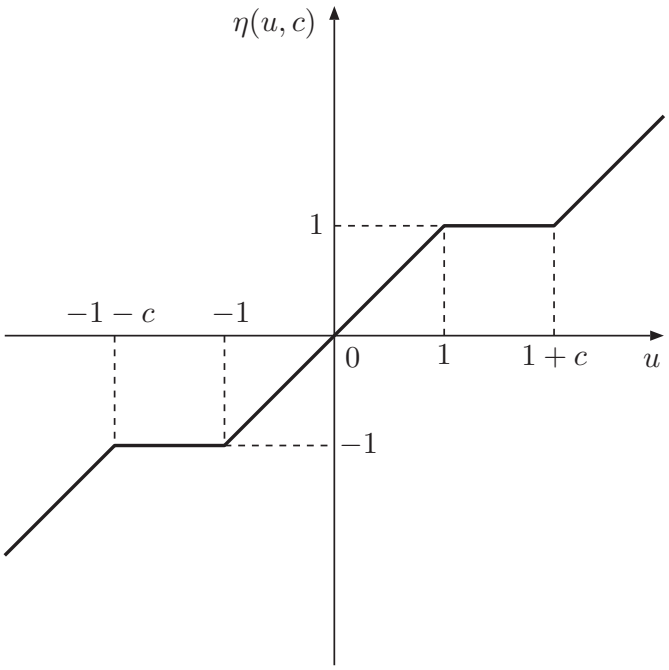


図 6.11: SOAV 最適化に対する弱しきい値関数 $\eta(u,c)$

合は

$$\lim_{\beta\to\infty} \mathsf{F}_\beta(u,c) = \begin{cases} u+c & (u<-c) \\ 0 & (-c\le u\le c)\,, \\ u-c & (c<u) \end{cases} \tag{6.57}$$

$$\lim_{\beta\to\infty} \beta\mathsf{G}_\beta(u,c) = c\,\eta'(u,c) \tag{6.58}$$

$$= \begin{cases} c & (u<-c) \\ 0 & (-c\le u\le c)\,, \\ c & (c<u) \end{cases} \tag{6.59}$$

SOAV 最適化の場合は

$$\lim_{\beta\to\infty}\mathsf{F}_\beta(u,c)=\begin{cases}u+c & (u<-1-c)\\ -1 & (-1-c\le u\le -1)\\ u & (-1<u<1)\\ 1 & (1\le u\le 1+c)\\ u-c & (1+c<u)\end{cases}, \tag{6.60}$$

$$\lim_{\beta\to\infty}\beta\mathsf{G}_\beta(u,c)=c\,\eta'(u,c) \tag{6.61}$$

$$=\begin{cases}c & (u<-1-c)\\ 0 & (-1-c\le u\le -1)\\ c & (-1<u<1)\\ 0 & (1\le u\le 1+c)\\ c & (1+c<u)\end{cases} \tag{6.62}$$

となり，更新式 (6.45), (6.46), (6.48) は以下のように書ける．

$$x_{j\to\theta}^{t+1}=\eta\left(\sum_{\zeta\neq\theta}a_{\zeta i}z_{\zeta\to i}^{t},\hat{\tau}^{t}\right), \tag{6.63}$$

$$z_{\theta\to j}^{t}=y_\theta-\sum_{i\neq j}a_{\theta,i}x_{i\to\theta}^{t}, \tag{6.64}$$

$$\hat{\tau}^{t+1}=\frac{\hat{\tau}^{t}}{M}\sum_{i=1}^{N}\eta'\left(\sum_{\zeta=1}^{M}a_{\zeta,i}z_{\zeta\to i}^{t},\hat{\tau}^{t}\right). \tag{6.65}$$

さらに近似を行い，SOAV 最適化問題に対する AMP アルゴリズムを導く．まず，伝搬されるメッセージが

$$x_{j\to\theta}^{t}=x_j^{t}+\partial x_{j\to\theta}^{t}+\mathcal{O}\left(\frac{1}{N}\right) \tag{6.66}$$

$$z_{\theta\to j}^{t}=z_\theta^{t}+\partial z_{\theta\to j}^{t}+\mathcal{O}\left(\frac{1}{N}\right) \tag{6.67}$$

と近似されるとする．ここで，$\partial x_{j\to\theta}^{t},\partial z_{\theta\to j}^{t}=\mathcal{O}(1/\sqrt{N})$ である．このとき，[47] の Lemma III.4 にあるように次の補題が成り立つ．

Lemma 2. 式 (6.66), (6.67) が成り立つとすると，

$$x_j^{t+1}=\eta\left(\sum_{\zeta=1}^{M}a_{j,\zeta}z_\zeta^{t}+x_j^{t},\hat{\tau}^{t}\right)+o_N(1) \tag{6.68}$$

$$z_\theta^{t}=y_\theta-\sum_{i=1}^{N}a_{\theta,i}x_i^{t}+\frac{1}{M}z_\theta^{t-1}\sum_{k=1}^{N}\eta'\left(\sum_{\zeta}a_{\zeta,k}z_\zeta^{t-1}+x_k^{t-1},\hat{\tau}^{t-1}\right)+o_N(1) \tag{6.69}$$

となる．ここで，$o_N(1)$ は $N\to\infty$ のとき $o_N(1)\to 0$ となる項である．

式 (6.68), (6.69) をベクトルを使って書き直し，$o_N(1)$ の項を無視すると

$$\boldsymbol{x}^{t+1} = \eta(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{z}^t + \boldsymbol{x}^t, \hat{\tau}^t) \tag{6.70}$$

$$\boldsymbol{z}^t = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^t + \frac{1}{\Delta}\boldsymbol{z}^{t-1}\langle \eta'(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{z}^{t-1} + \boldsymbol{x}^{t-1}, \hat{\tau}^{t-1})\rangle \tag{6.71}$$

となる．また，同様にして

$$\hat{\tau}^t = \frac{\hat{\tau}^{t-1}}{\Delta}\langle \eta'(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{z}^{t-1} + \boldsymbol{x}^{t-1}, \hat{\tau}^{t-1})\rangle \tag{6.72}$$

が得られる．以上の式 (6.70)–(6.72) が AMP アルゴリズムの更新式である．ベクトル同士の和および行列とベクトルの積からなっているため，その計算量は $\mathcal{O}(MN)$ となる．

式 (6.70)–(6.72) はパラメータのない更新式となっているが，$\hat{\tau}$ を最適化すべきパラメータとみなすアプローチも考えられる [44]．例えば $\lambda > 0$ と $\sigma_t^2 := \|\boldsymbol{x}^t - \boldsymbol{b}\|_2^2/N$ を用いて $\hat{\tau}$ を $\lambda\sigma_t$ と置き換えれば，式 (6.70), (6.71) は

$$\boldsymbol{x}^{t+1} = \eta(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{z}^t + \boldsymbol{x}^t, \lambda\sigma_t) \tag{6.73}$$

$$\boldsymbol{z}^t = \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^t + \frac{1}{\Delta}\boldsymbol{z}^{t-1}\langle \eta'(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{z}^{t-1} + \boldsymbol{x}^{t-1}, \lambda\sigma_{t-1})\rangle \tag{6.74}$$

となる．実際に計算を行う際には $\boldsymbol{b}$ は未知であるから，例えば SOAV 最適化の場合は $\boldsymbol{b}$ の代わりに $\mathrm{sign}(\boldsymbol{x}^t)$ などを用いて $\sigma_t^2 = \|\boldsymbol{x}^t - \mathrm{sign}(\boldsymbol{x}^t)\|_2^2/N$ とする必要がある．

第7章　まとめ

本稿では，ベイズの定理に基づく確率推論の基礎として，状態空間モデルに基づく時系列解析の手法と，確率伝搬法の基本的なアルゴリズムについて解説した．カルマンフィルタや粒子フィルタ，ターボ符号の復号アルゴリズムといった一見するとあまり関連がなさそうなアルゴリズムが全て，条件付き独立性を利用した周辺事後分布の効率的な計算という文脈で統一的に説明される点が重要である．

従来の通信分野の問題における信号処理では，最大事後確率推定や最尤推定等の非線形のアプローチのほうが特性が良いことが分かった上で，様々な制約によって主に線形の手法が採用されてきたが，近年の様々な確率推論手法の進展による計算効率の向上，絶え間ない通信特性改善への要求，基地局機能のクラウド上での実現などを背景に，今後は確率推論に基づく手法が通信分野にも次々に導入されていくと考えられる．本稿が，通信分野の読者が確率推論に興味をもつ切っ掛けになれば幸いである．なお，各手法を解説する都合上，一部で確率論に関する内容についても説明を行なったが，数学的な厳密性はかなり犠牲にしてあるため，さらに勉強したい読者は巻末の参考文献を参考にされたい．例えば，[8], [9] などは，工学系の読者や初学者を意識して書かれており，大変お勧めである．

関連図書

[1] D. P. Bertsekas and J. N. Tsitsiklis, Introduction to Probability, Athena Scientific, 2008.

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

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

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

[5] A. Papoulis and S. U. Pillai, Probability, Random Variables and Stochastic Process, MacGraw-Hill, 2002.

[6] 片山徹，新版応用カルマンフィルタ，朝倉書店，2000.

[7] 片山徹，非線形カルマンフィルタ，朝倉書店，2011.

[8] 渡辺澄夫，村田昇，確率と統計–情報学への架橋–，コロナ社，2005.

[9] 高橋幸雄，基礎数理講座 2 確率論，朝倉書店，2008.

[10] 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.

[11] 和田山正，誤り訂正技術の基礎，森北出版，2010.

[12] 汪金芳，田栗正章，手塚集，樺島祥介，上田修功，統計科学のフロンティア 11 計算統計 I，岩波書店，2003.

[13] 伊庭幸人，種村正美，大森裕浩，和合肇，佐藤整尚，高橋明彦，統計科学のフロンティア 12 計算統計 II，岩波書店，2005.

[14] A. F. M. Smith and A. E. Gelfand, “Bayesian statistics without tears: a sampling-resampling perspective,” Amer. Stat., vol. 46, no. 2, pp. 84–88, May, 1992.

[15] W. H. Press et al., Numerical Recipes, 3rd Edition, Cambridge University Press, 2007.

[16] 荻原春生，ターボ符号の基礎，トリケップス，1999.

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

[18] 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.

[19] 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.

[20] N. J. Gordon, D. J. Salmond and A. F. M. Smith, “Novel approach to nonlinear/ non-Gaussian Bayesian state estimation,” IEE Proceedings-F, vol. 140, no. 2, pp. 107–113, Apr. 1993.

[21] A. Doucet, N. de Freitas and N. Gordon, Sequential Monte Carlo Methods in Practice, Springer, 2001.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 樋口知之，“粒子フィルタ，”電子情報通信学会誌，vol. 88, no. 12, pp. 989–994, Dec. 2005.

[27] 中妻照雄，入門ベイズ統計学，朝倉書店，2007.

[28] 豊田秀樹，マルコフ連鎖モンテカルロ法，朝倉書店，2008.

[29] 伊庭幸人，講座物理の世界 ベイズ統計と統計物理，岩波書店，2003.

[30] 小西貞則，越智義道，大森裕浩，予測と発見の科学 計算統計学の方法，朝倉書店，2008.

[31] 宮川雅巳，グラフィカルモデリング，朝倉書店，1997.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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

[37] 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

[38] S. Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. on Communications*, vol. 48, pp. 1727–1737, 2001.

[39] 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).

[40] Y. Kabashima and D. Saad, "Belief propagation vs. TAP for decoding corrupted messages," Europhysics Letters, vol. 44, no. 5, pp. 668–674, Dec. 1998.

[41] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free energy approximations and generalized belief propagation algorithms," IEEE Trans. Inf. Theory, vol. 51, no. 7, pp. 2282–2312, July 2005.

[42] Y. Yoshida, K. Hayashi, H. Sakai, "Marginalized Particle Filter for Blind Signal Detection with Analog Imperfections," IEICE Transactions on Communications, vol. E93-B, No. 2, pp. 336–344, Feb. 2010.

[43] T. Wo and P. A. Hoeher, "Low-Complexity Gaussian Detection for MIMO Systems," Journal of Electrical and Computer Engineering, vol. 2010, Article ID 609509, 12 pages doi:10.1155/2010/609509.

[44] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," Proc. of the National Academy of Sciences (PNAS) , vol. 106, no. 45, pp. 18914–18919, Nov. 2009.

[45] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[46] M. Nagahara, "Discrete signal reconstruction by sum of absolute values," *IEEE Signal Process. Lett.*, vol. 22, no. 10, pp.1575–1579, Oct. 2015.

[47] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction.", in *Proc. IEEE Inf. Theory Workshop*, pp. 1–5, Jan. 2010.