■ **概要**

前回の内容では、メッセージという概念によって連鎖グラフにおける周辺分布を効率的に計算できることを確認した。今回の内容である「**8.4.3 因子グラフ**」「**8.4.4 積和アルゴリズム**」は、より一般のグラフに対しても効率的に周辺分布を計算するために必要なテクニックである。

因子グラフは同時分布の因数分解を直接グラフに反映させたグラフである。条件付き独立関係やクリークポテンシャルに対応する因子ノードを作成することで、有向グラフや無向グラフから因子グラフを形成できる。また、木構造であるグラフィカルモデルから構成される因子グラフも木構造となることを確認する。木構造であることが後の積和アルゴリズムで役立つ。

積和アルゴリズムは木構造である因子グラフ上でメッセージを伝搬することで、効率的に周辺分布を算出する方法である。連鎖構造の周辺分布を求めるときは、隣接する2ノードのみを考慮したが、ここでは一般的な木構造グラフで周辺分布を求める方法を学ぶ。今回は特にある一つの変数の周辺分布を求めるところまで扱い、すべての変数の周辺分布を求める方法は次回に回す。

■ **8.4.3 因子グラフ**

有向グラフや無向グラフでは、同時分布を因数分解できることを見てきた。因子グラフでは同時分布を因数分解したときの因子に対応するノードを追加することで、因数分解を陽に表現する。

同時分布を次のような因子の積で表せるとする。

$$p(\mathbf{x}) = \prod_{s} f_s(\mathbf{x}_s)$$

ここで $\mathbf{x}_s$ は変数の部分集合を表す。このとき因子 $f_s(\mathbf{x}_s)$ をノードで表現し、$\mathbf{x}_s$ とリンクを張ることで因子グラフを構築する。

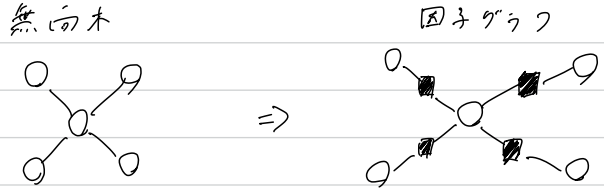
例えば次の因数分解が与えられた時、因子グラフは図8.40の形で表される。

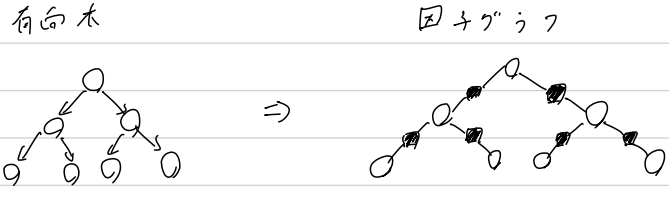
$$p(\mathbf{x}) = f_a(x_1, x_2)\, f_b(x_1, x_2)\, f_c(x_2, x_3)\, f_d(x_3)$$

図8.41、図8.42で無向グラフから因子グラフを構築する方法と有向グラフから因子グラフを構築する方法の説明があるので読み合わせとする。

次に因子グラフが木構造となるか否かの解釈を与える。結論から述べると、有向木、無向木、有向多重木からは木構造となる因子グラフを構築できる。また、有向グラフにおける局所的な閉路も木構造に変換できる。これを図8.43、図8.44で確認する。

（ただし、有向木、無向木から木構造の因子グラフを構築する例がない。図8.39から因子グラフを考えてみると確かにそうっぽい↓）





8.4.4 積和アルゴリズム

積和アルゴリズムは木構造の因子グラフから効率的に周辺分布を計算する方法である。

8.4.4 項では下記の2点を確かめるが、今回の内容は前半の (i) までとする。

(i) 周辺分布を求めるための効率良い厳密推論アルゴリズムを得る

(ii) 複数の周辺分布を計算したい場合に、計算の重複をなくして効率化する

まず、我々の目的は因子グラフ上のあるノード x の周辺分布を求めることである。

$$p(x) = \sum_{\boldsymbol{x} \setminus x} p(\boldsymbol{x}) \qquad (8.61)$$

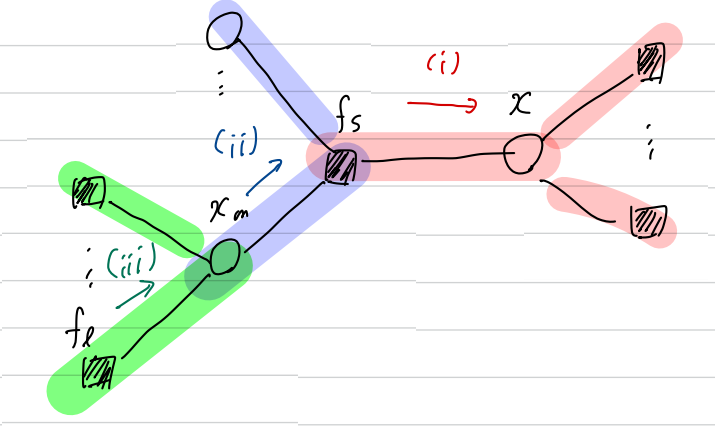
$\boldsymbol{x} \setminus x$: 変数集合から x を除いたもの

これを求めるために次の3つのステップに分解して計算を行う。

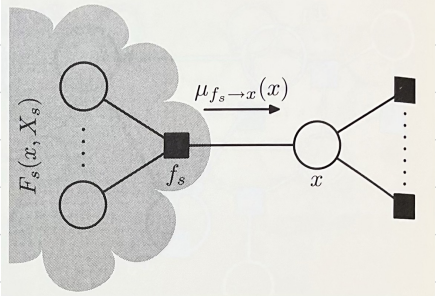
(i) 隣接因子 f_s から変数 x へのメッセージ

(ii) 隣接因子の隣接変数 x_m から隣接因子 f_s へのメッセージ

(iii) 隣接因子の隣接変数の隣接因子 f_l から隣接因子の隣接変数 x_m へのメッセージ



(i) 隣接因子 f_s から変数 x へのメッセージ



因子グラフの定義より、同時分布は全ての因子の積で表すことができる。これを変数 x の隣接因子ごとにグループ化すると次のように表すことができる。

$$p(x) = \prod_{s \in ne(x)} F_s(x, X_s) \qquad (8.62)$$

x の隣接ノード集合

隣接ノード f_s より先のノード集合

木構造であることより X_s を分離できる。

これを (8.61) に代入すると,

$$p(x) = \sum_{x \setminus x} \prod_{s \in ne(x)} F_s(x, X_s)$$

↓変数を分解

$$= \sum_{X_1} \sum_{X_2} \cdots \sum_{X_K} \prod_{s \in ne(x)} F_s(x, X_s)$$

f_1 より先のノード集合で集約化

F_s を掛ける。

$$= \sum_{X_1} \sum_{X_2} \cdots \sum_{X_K} F_1 F_2 \cdots F_K$$

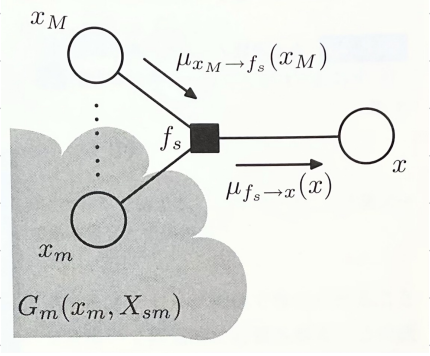
$$= \left(\sum_{x_1} F_1 \right) \left(\sum_{x_2} F_2 \right) \cdots \left(\sum_{x_K} F_K \right)$$

$$= \prod_{s \in ne(x)} \left[\sum_{x_s} F_s(x, X_s) \right]$$

$$= \prod_{s \in ne(x)} \mu_{f_s \to x}(x) \qquad (8.63)$$

ここで、$\mu_{f_s \to x}(x)$ を f_s から x へのメッセージとして解釈すると、求めたい周辺分布は隣接因子からのメッセージの積で表されることがわかる。

(ii) 隣接因子の隣接変数 x_m から隣接因子 f_s へのメッセージ



f_s から x へのメッセージ自体もさらに分割することができる。

$$\mu_{f_s \to x}(x) = \sum_{\sim x} F_s(x, X_s)$$

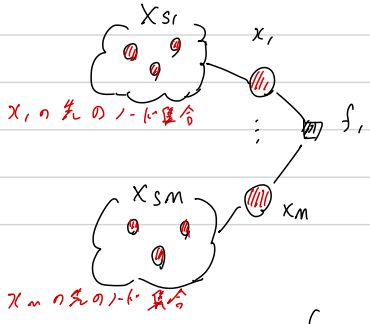
$$= \sum_{X_s} f_s(x, x_1, \cdots, x_m)\, G_1(x_1, X_{s1}) \cdots G_m(x_m, X_{sm})$$

f_s 自身の因子　　x_1 より先の因子の積 …

$$= \sum_{X_s} f_s(x, x_1, \cdots, x_m) \prod_{m \in ne(f_s)\setminus x} G_m(x_m, X_{sm})$$

↓ 変数を分解

$$= \sum_{x_1} \cdots \sum_{x_m} \sum_{X_{s1}} \cdots \sum_{X_{sm}} f_s(x, x_1, \cdots, x_m) \prod_{m \in ne(f_s)\setminus x} G_m(\cdot)$$



$$= \sum_{x_1} \cdots \sum_{x_m} f_s(x, x_1, \cdots, x_m) \sum_{X_{s1}} \cdots \sum_{X_{sm}} \prod_{m \in ne(f_s) \setminus x} G_m(x_m, X_{sm})$$

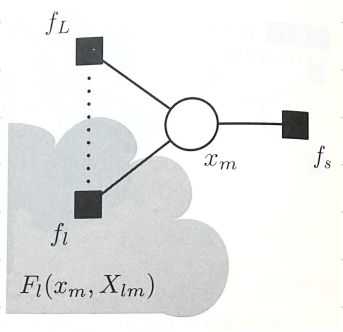
(8.63) 同様に入れ替え

$$= \sum_{x_1} \cdots \sum_{x_m} f_s(x, x_1, \cdots, x_m) \prod_{m \in ne(f_s) \setminus x} \left[\sum_{X_{sm}} G_m(x_m, X_{sm}) \right]$$

$$= \sum_{x_1} \cdots \sum_{x_m} f_s(x, x_1, \cdots, x_m) \prod_{m \in ne(f_s) \setminus x} \mu_{x_m \to f_s}(x_m)$$

これよりメッセージ $\mu_{f_s \to x}$ を計算するには、f_s の隣接変数から伝達されるメッセージ $\mu_{x_m \to f_s}$ のすべての積と自身の因子をかけ合わせ、隣接変数で周辺化すれば良い。

(iii) 隣接因子の隣接変数の隣接因子 f_l から隣接因子の隣接変数 x_m へのメッセージ



最後に、x_m から f_s へ送られるメッセージをさらに分解してみる。

$$\mu_{x_m \to f_s} = \sum_{X_{sm}} G_m(x_m, X_{sm})$$

$$= \sum_{X_{sm}} \prod_{l \in ne(x_m) \setminus f_s} F_l(x_m, X_{lm})$$

f_l より先の因子の積

$$= \text{(略)}$$

$$= \prod_{l \in ne(x_m) \setminus f_s} \left[\sum_{X_{lm}} F_l(x_m, X_{lm}) \right]$$

$$= \prod_{l \in ne(x_m) \setminus f_s} \mu_{f_l \to x_m}(x_m)$$

これより、x_m に隣接する因子から送られてくるメッセージの積として表すことができる。

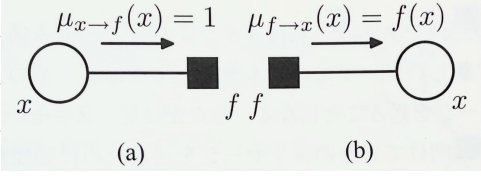
以上の (i), (ii), (iii) を繰り返すことにより再帰的に周辺分布を算出することができる。

ここで、葉ノードが変数ノードである場合、変数ノードからのメッセージは1となる。

$$\mu_{x \to f}(x) = 1$$

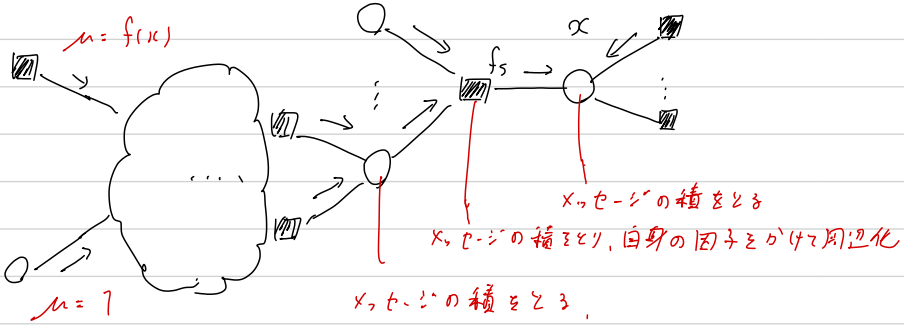
葉ノードが因子ノードである場合、送られるメッセージは自身の因子となる。

$$\mu_{f \to x}(x) = f(x)$$



よくわからないがこうすると上手く機能するっぽい。実際に上手く機能する様子は次回以降（図8.50、図8.51あたり）で確認できそう。

積和アルゴリズムの全体像を再度おさらいすると、次の図にまとめられる。



このようなメッセージの伝達は、因子グラフが木構造であることで可能となっていたことに注意する。すなわち、木構造であることでリンクより先のノード集合を隣接ノードごとに分離することができ、積と和を交換することが可能となっていた。

以上をまとめると、周辺分布を求めるための手順は次のようになる。

1. グラフィカルモデルに対応する因子グラフを（木構造になるように）構築する。
2. 葉ノードから順にメッセージを伝搬する。
3. 周辺分布を求めたいノードでメッセージの積を取る。