PRML 8.4.4~8.4.8

6/25/2018

酒井一徳

目次

積和アルゴリズム

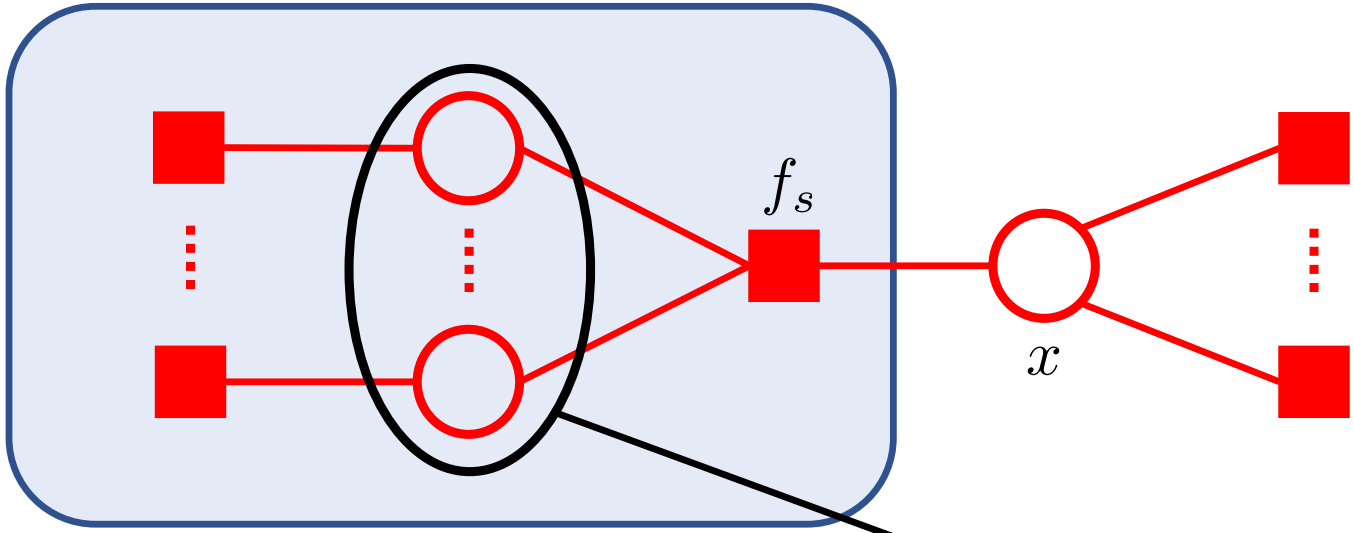
モチベーション

- 周辺分布を求めるための効率的な厳密推論アルゴリズムを得る.
- 複数の周辺分布を計算する際に，計算の重複をなくす.

積和アルゴリズム導出のための仮定

- 変数は全て離散
- グラフは因子グラフへ変換する
- 元々のグラフは有向木，無向木，多重木
- 全てのノードは観測されていない

周辺分布と同時分布



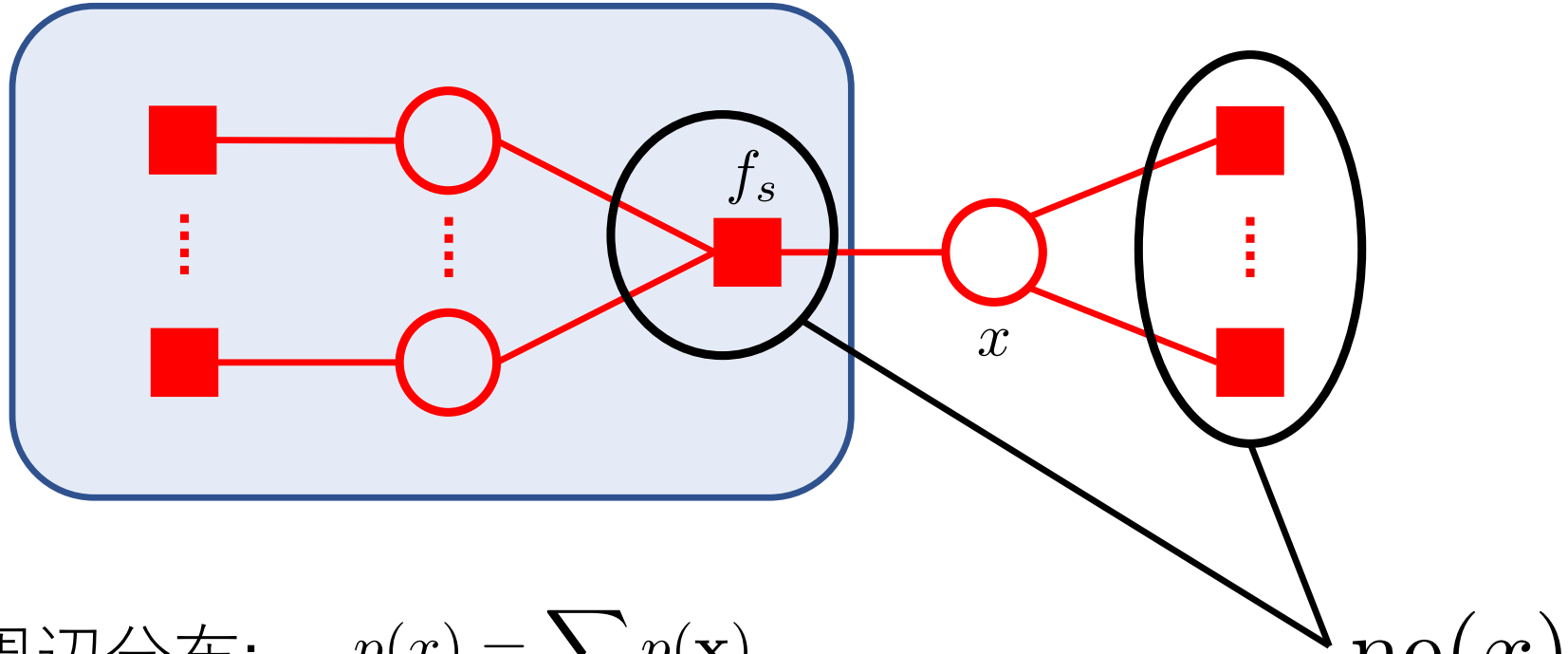
X_s

部分木に属する全ての変数集合

周辺分布: $p(x) = \sum_{\mathbf{x} \setminus x} p(\mathbf{x})$

同時分布: $p(\mathbf{x}) = \prod_{s \in \mathrm{ne}(x)} F_s(x, X_s)$

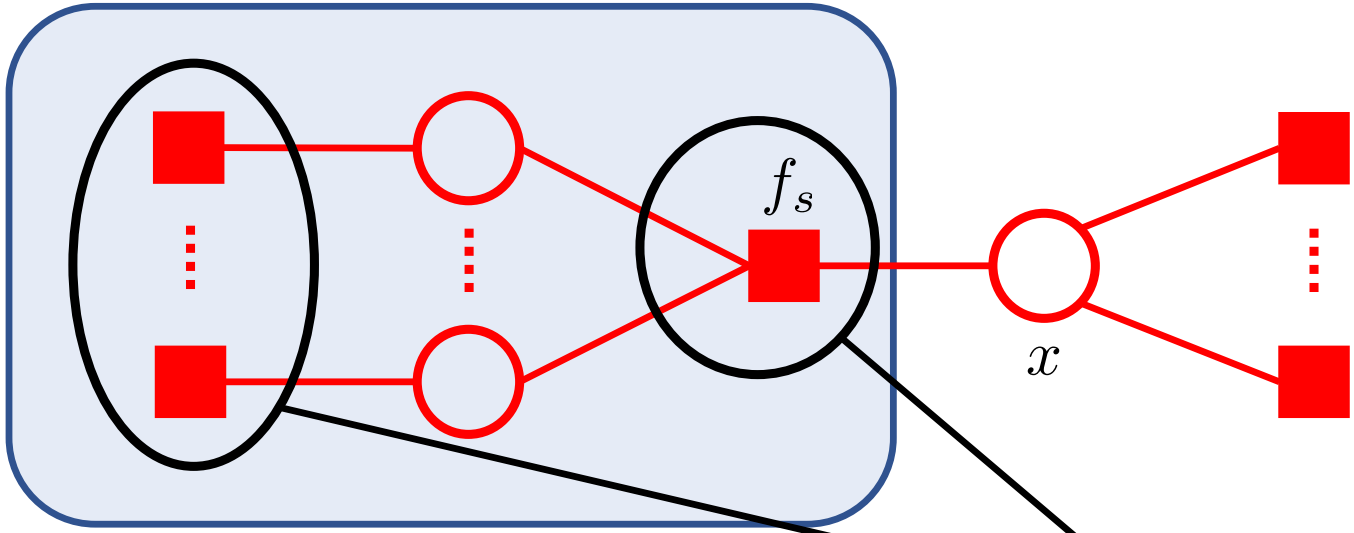
周辺分布と同時分布



周辺分布: $p(x) = \sum_{\mathbf{x}\setminus x} p(\mathbf{x})$

同時分布: $p(\mathbf{x}) = \prod_{s\in \mathrm{ne}(x)} F_s\left(x, X_s\right)$

周辺分布と同時分布



積: $F_s(x, X_s)$

周辺分布: $p(x) = \sum_{\mathbf{x}\setminus x} p(\mathbf{x})$

同時分布: $p(\mathbf{x}) = \prod_{s\in \mathrm{ne}(x)} F_s(x, X_s)$ ← 式(8.59)を
グループ分けした形

周辺分布の和と積の入れ替え

周辺分布の式に同時分布の式を代入,

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

① グループ内の因子の積
② 状態について総和

和と積を入れ替え,

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

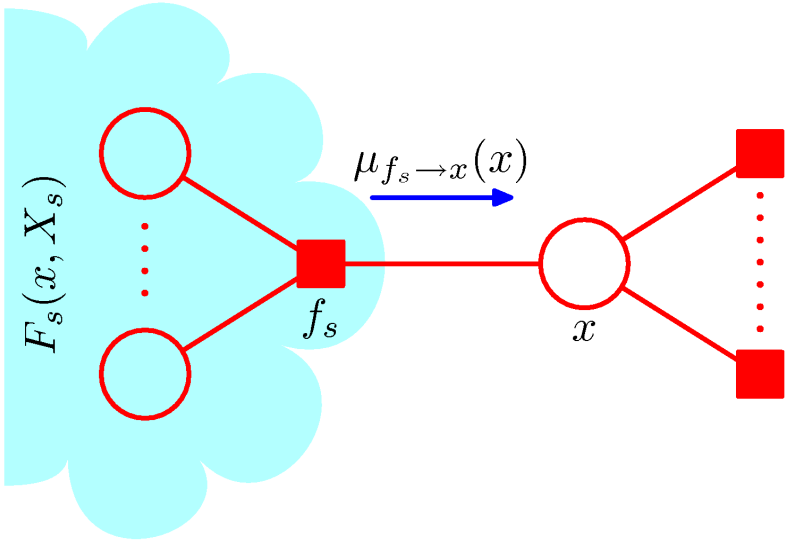
① グループ内で状態について総和
② 因子の積

因子ノードから変数ノードへのメッセージ

以下で定義される関数を導入,

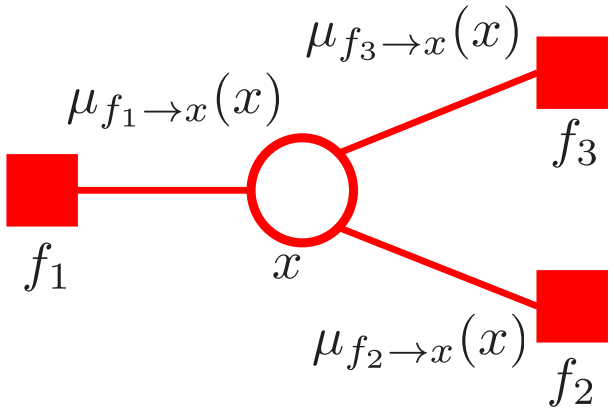
$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s)$$

周辺分布: $p(x) = \prod_{s \in ne(x)} \mu_{f_s \to x}(x)$



因子ノード f_s から変数ノード x へのメッセージと解釈できる.

メッセージはベクトル



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

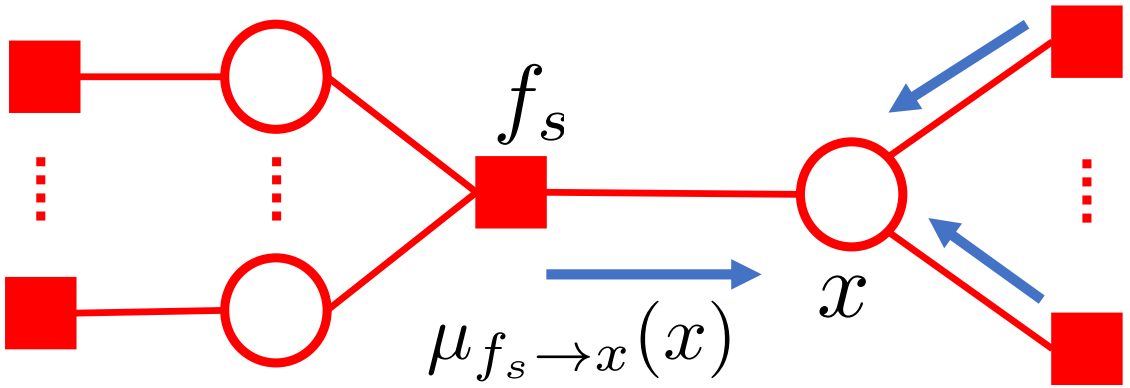
$$p(x) = \mu_{f_1\to x}(x)\mu_{f_2\to x}(x)\mu_{f_3\to x}(x)$$

状態数が３つの時,

$$p(x) = \begin{pmatrix} p(x=1) \\ p(x=2) \\ p(x=3) \end{pmatrix} = \begin{pmatrix} \mu_{f_1\to x}(x=1) \\ \mu_{f_1\to x}(x=2) \\ \mu_{f_1\to x}(x=3) \end{pmatrix} \odot \begin{pmatrix} \mu_{f_2\to x}(x=1) \\ \mu_{f_2\to x}(x=2) \\ \mu_{f_2\to x}(x=3) \end{pmatrix} \odot \begin{pmatrix} \mu_{f_3\to x}(x=1) \\ \mu_{f_3\to x}(x=2) \\ \mu_{f_3\to x}(x=3) \end{pmatrix}$$

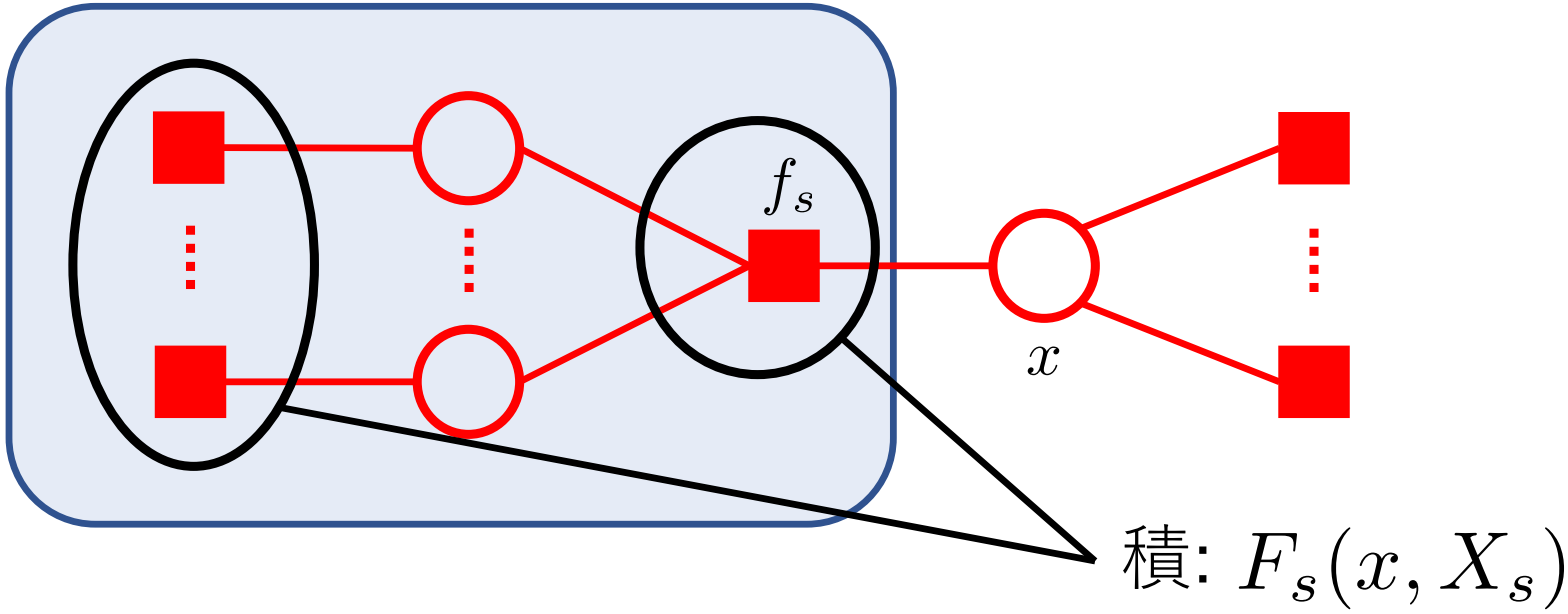
$\odot$ はアダマール積？

周辺分布と変数ノードへのメッセージ



$$p(x) = \prod_{s \in \mathrm{ne}(x)} \left[\sum_{X_s} F_s(x, X_s) \right] = \prod_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x)$$

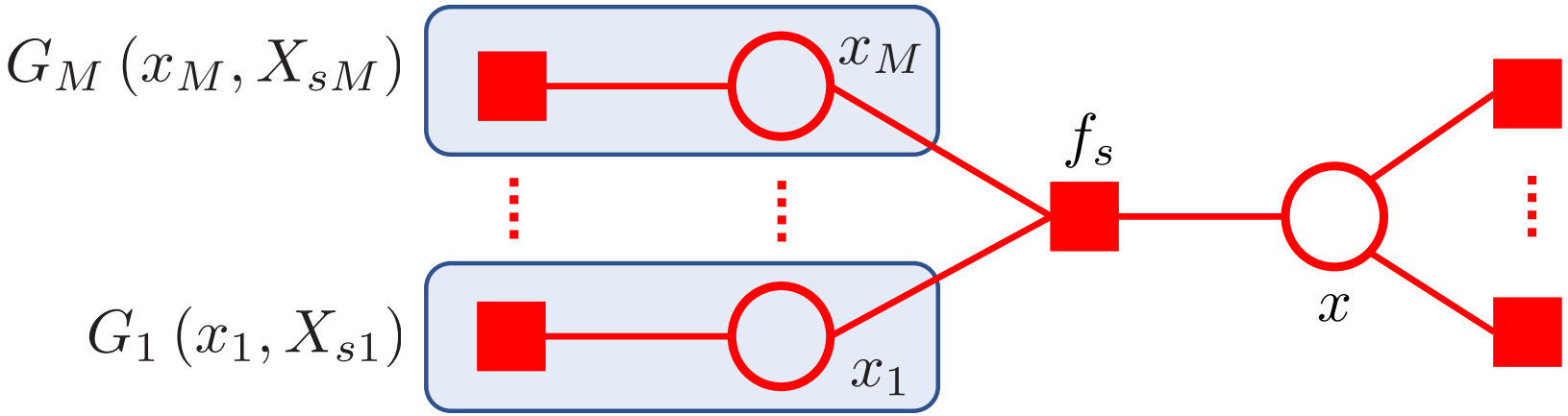
$F_s\,(x, X_s)$ の因数分解



$F_s\,(x, X_s)$ も因子(部分)グラフで記述される.

因数分解できる.

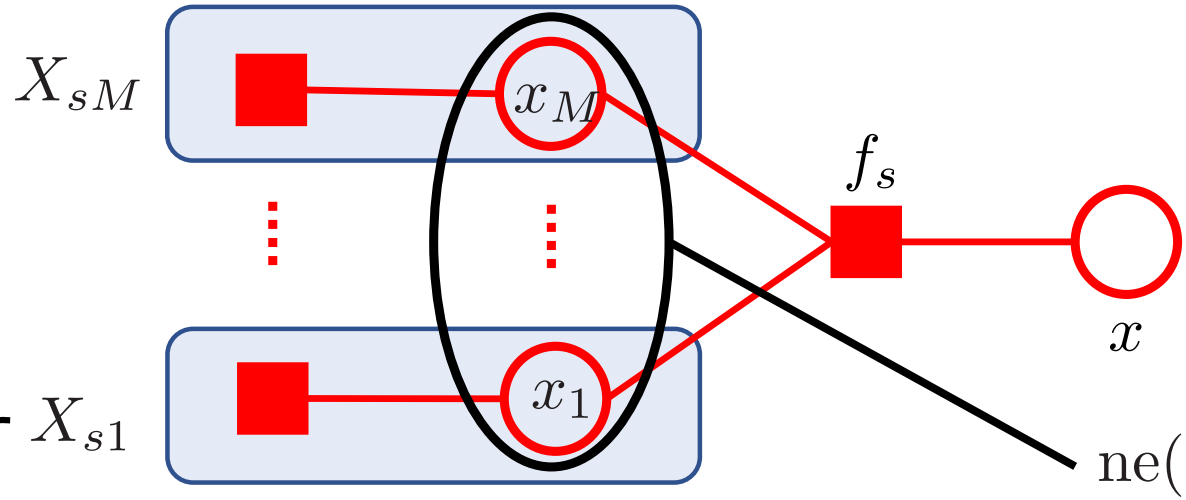
$F_s\left(x, X_s\right)$ の因数分解



$$F_s(x, X_s) = f_s(x, x_1, \ldots, x_M) G_1\left(x_1, X_{s1}\right) \ldots G_M\left(x_M, X_{sM}\right)$$

因子(根)　　変数ノードのグループ(部分木)

文字の意味



x_1を通してf_sへ接続される部分木に
属する全ての変数の集合.

xを除くf_sの
隣接変数ノード集合.

変数ノードから因子ノードへのメッセージ

$$F_s(x, X_s) = f_s(x, x_1, \ldots, x_M) \prod_{m \in \mathrm{ne}(f_s) \setminus x} G_m(x_m, X_{sm}) \qquad (8.65)$$

$$\mu_{f_s \to x}(x) \equiv \sum_{X_s} F_s(x, X_s) \qquad (8.64)$$

(8.65)を(8.64)へ代入.

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

変数ノードから因子ノードへのメッセージ

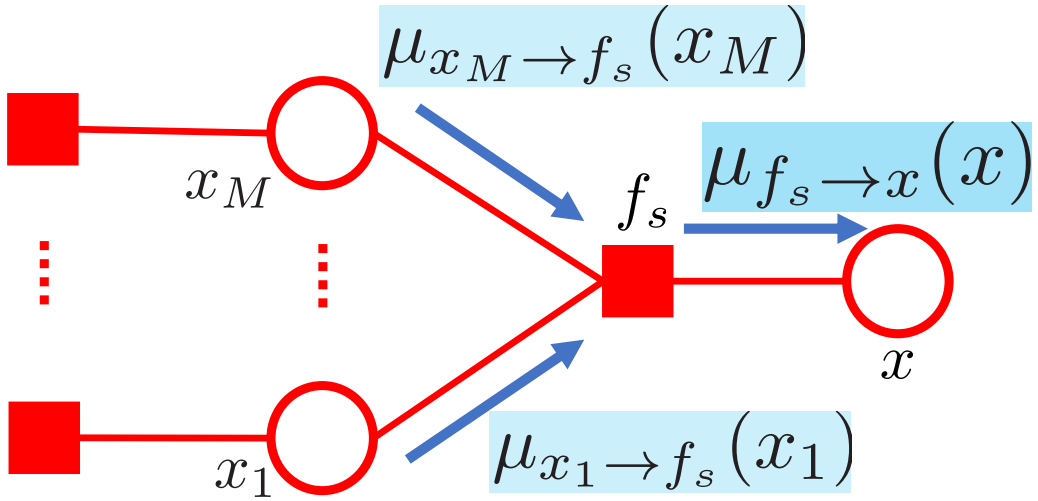
以下の因数ノードへのメッセージを定義.

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

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

変数ノードへのメッセージの計算手順

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



変数ノードへのメッセージの計算手順

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

ベクトル

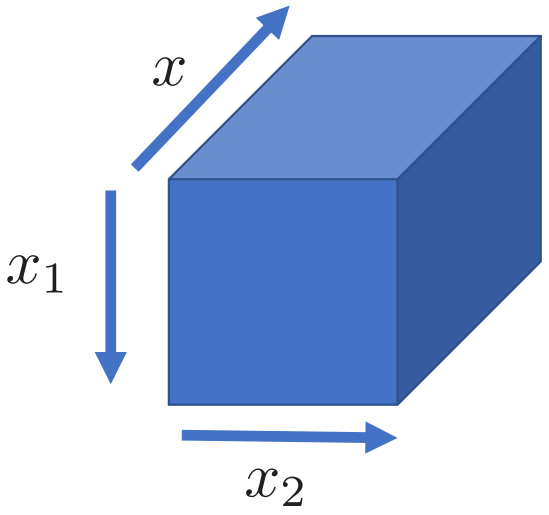
(状態数)^(M+1)次元のテンソル

状態数次元のベクトル

(状態数)=3, M=2のとき,

$f_s(x, x_1, x_2)$ は

$3 \times 3 \times 3$ のテンソル



変数ノードへのメッセージの計算手順

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

状態数が3つ，M＝3の場合の，メッセージの要素(x=1)の具体例が以下.

ただし，$f_s(x = i, x_1 = j, x_2 = k) = f_{ijk}$ として記述した.

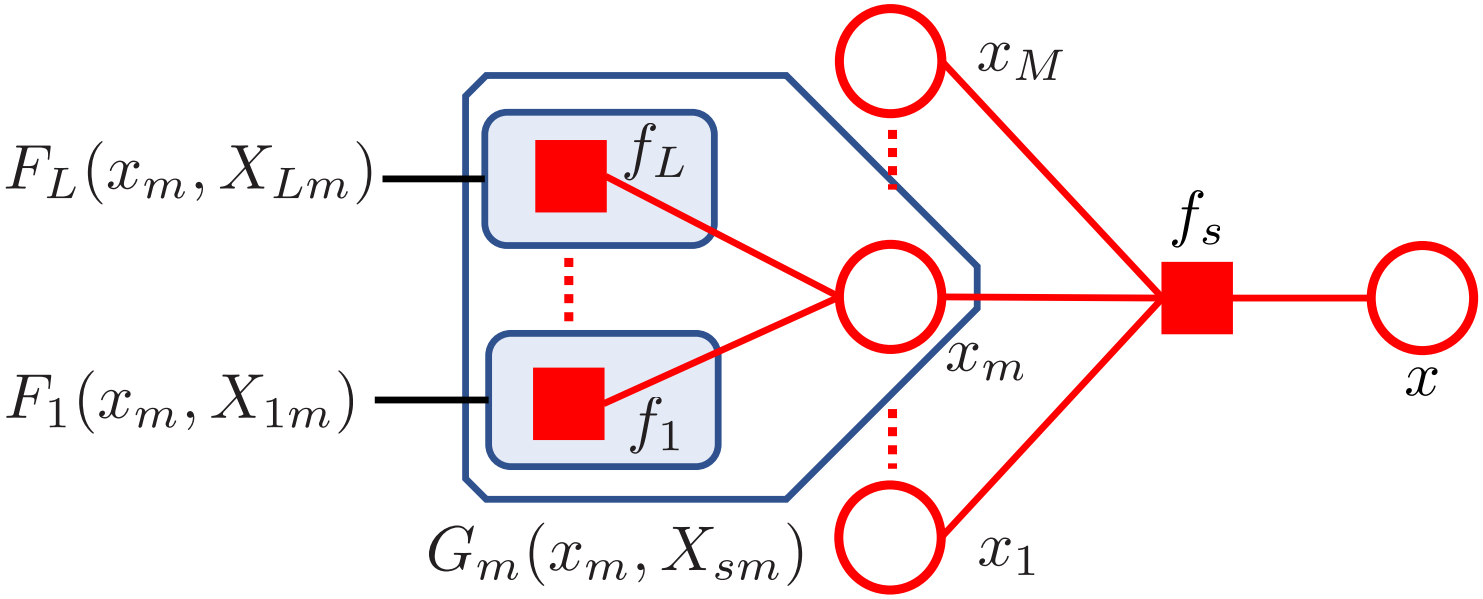
$$\mu_{f_s \to x}(x = 1) = \sum_{x_1} \sum_{x_2} \begin{pmatrix} f_{111} & f_{112} & f_{113} \\ f_{121} & f_{122} & f_{123} \\ f_{131} & f_{132} & f_{133} \end{pmatrix} \odot \begin{pmatrix} \mu_{x_1 \to f_s}(x_1 = 1) \\ \mu_{x_1 \to f_s}(x_1 = 2) \\ \mu_{x_1 \to f_s}(x_1 = 3) \end{pmatrix} \odot \begin{pmatrix} \mu_{x_2 \to f_s}(x_2 = 1) \\ \mu_{x_2 \to f_s}(x_2 = 2) \\ \mu_{x_2 \to f_s}(x_2 = 3) \end{pmatrix}$$

スカラー

⊙ はアダマール積ではない？

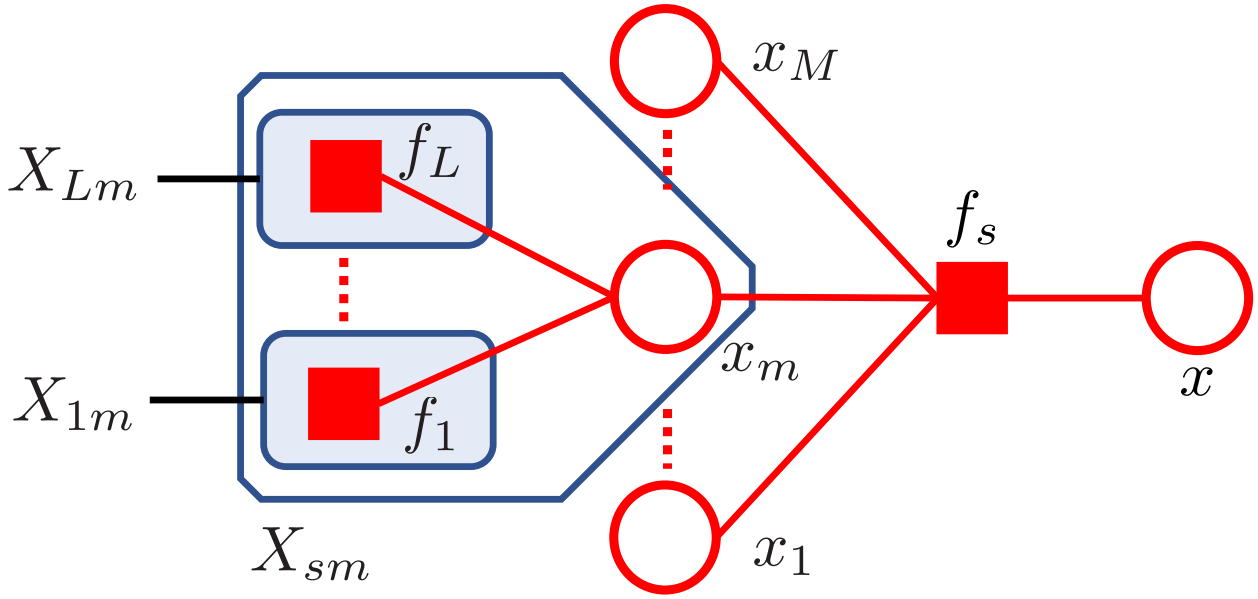
アダマール積は同じサイズの二つの行列に対し定義されるため

$G_m(x_m, X_{sm})$ の因数分解



$$G_m\left(x_m, X_{sm}\right) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l\left(x_m, X_{lm}\right)$$

文字の意味



X_{sm}： x_mを通して因子ノードf_sへ接続する変数集合

X_{lm}： f_l を通して変数ノードx_mへ接続する変数集合

因子ノードへのメッセージ

$$G_m(x_m, X_{sm}) = \prod_{l \in \mathrm{ne}(x_m) \setminus f_s} F_l(x_m, X_{lm}) \qquad (8.68)$$

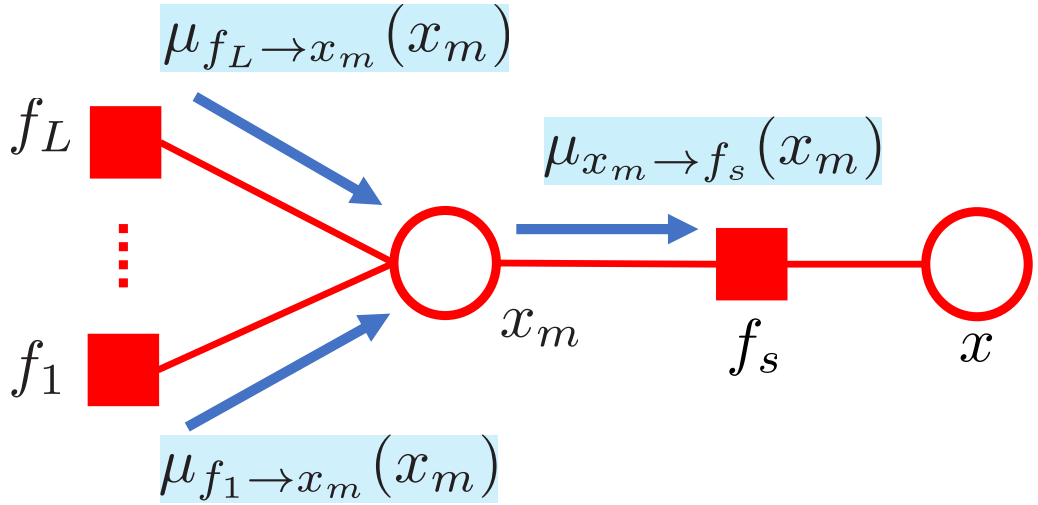
$$\mu_{x_m \to f_s}(x_m) \equiv \sum_{X_{sm}} G_m(x_m, X_{sm}) \qquad (8.67)$$

(8.68)を(8.67)へ代入.

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

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

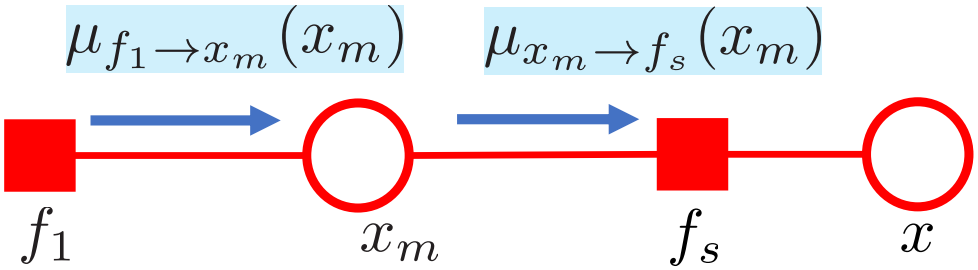
因子ノードへのメッセージの計算手順



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

変数ノードの隣接因子ノードが2つの場合

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

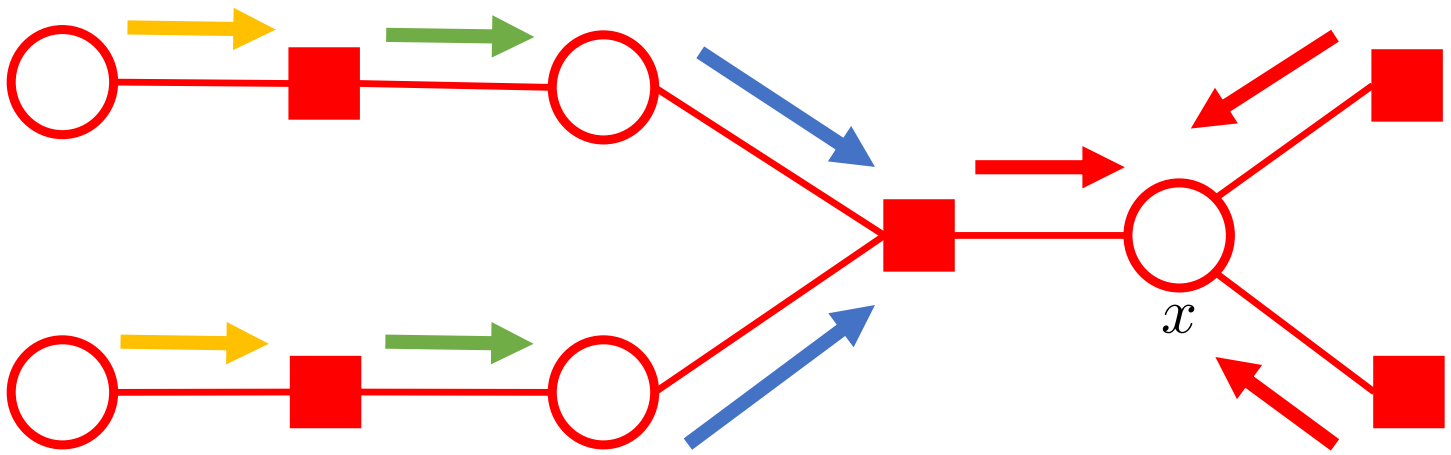


メッセージを変えずにそのまま伝達するだけ.

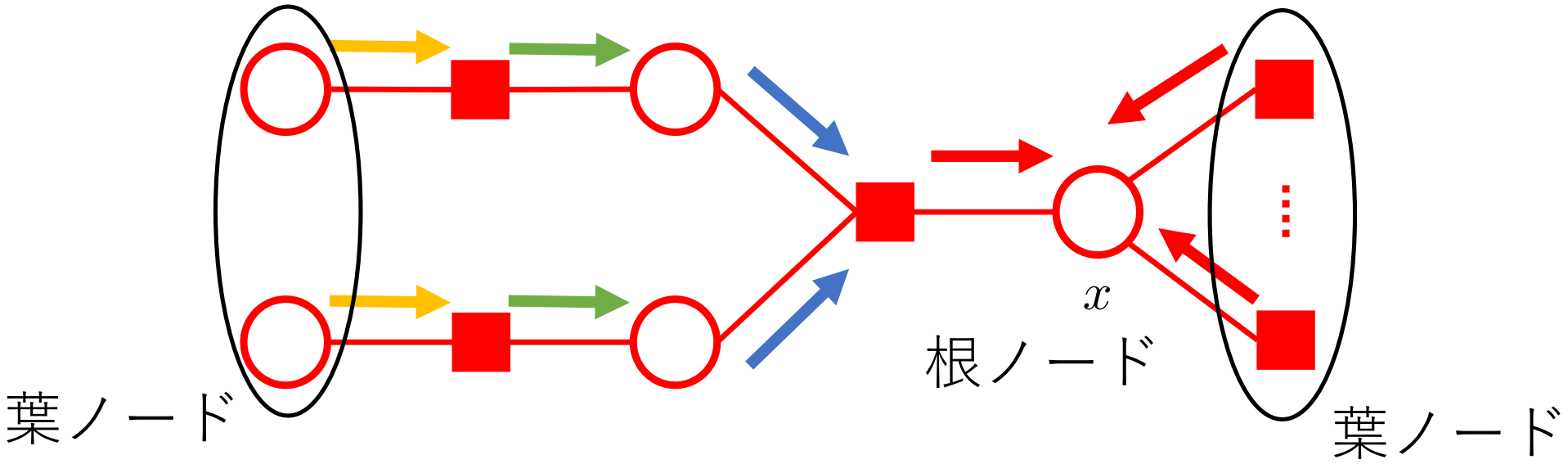
モチベーションと方針

- 周辺分布を求める.

  目的ノードへ到達する全てのメッセージの積.



方針とアルゴリズムの始点

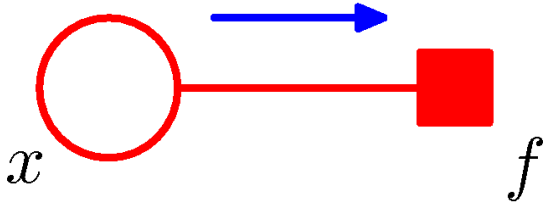


目的ノードを根と見て葉ノードからメッセージを送る.

葉ノードからのメッセージ

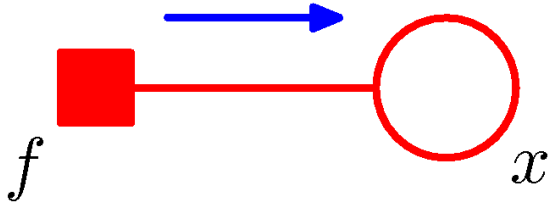
葉ノードが
変数ノードの時

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



葉ノードが
因子ノードの時

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

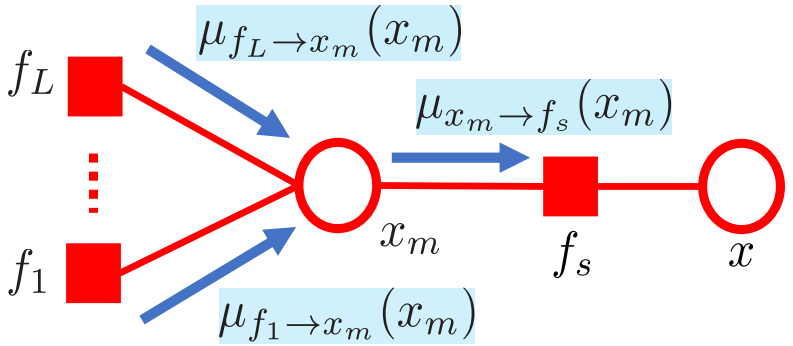


上記２パターンでメッセージ初期化

メッセージ伝達

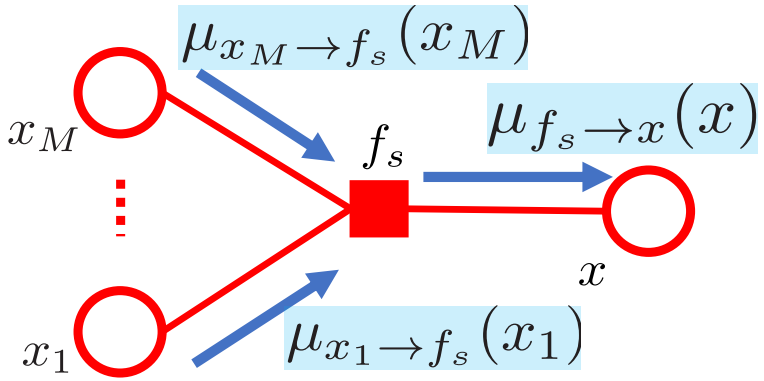
変数ノードから
因子ノード

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



因子ノードから
変数ノード

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



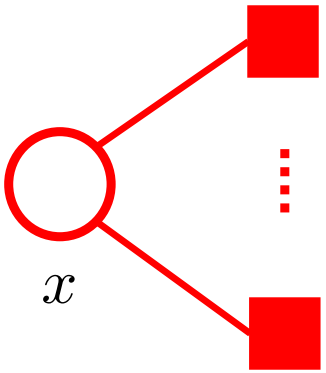
積和アルゴリズムまとめ

1. 根ノードを決定(求めたい周辺分布の変数ノード).
2. 葉ノードからのメッセージ初期化.
3. 葉ノードから根ノードに到達するまで再帰的にメッセージ計算.
4. 根ノードに伝達された全てのメッセージの積を計算.

メッセージを必ず受け取れるか

- メッセージを送信するには，送りたいノード以外からのメッセージを全て受け取る必要がある．

自明な例



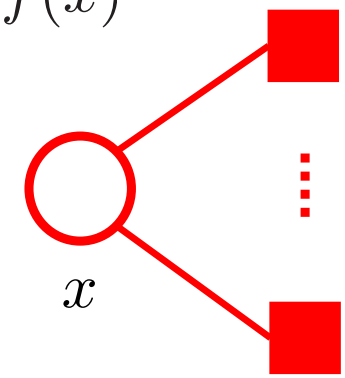
$$\mu_{f \to x}(x) = f(x)$$

ノードを一つずつ付け足して一般のグラフで成立することを証明する．

メッセージを必ず受け取れるか　証明

仮定　あるグラフに対して適切なアルゴリズムがある.

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

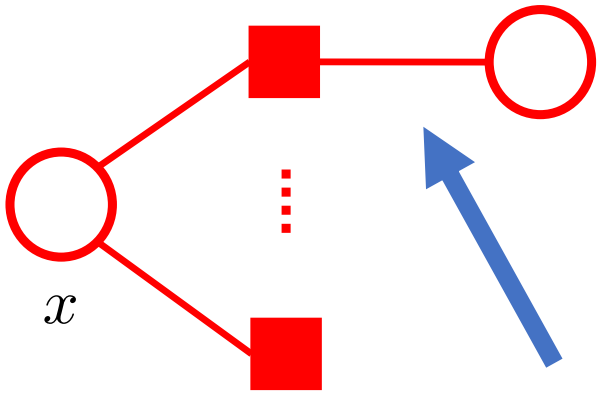


メッセージを必ず受け取れるか　証明

仮定　あるグラフに対して適切なアルゴリズムがある.

一つのノードを加える.

新しいノードは必ず葉ノードになる.

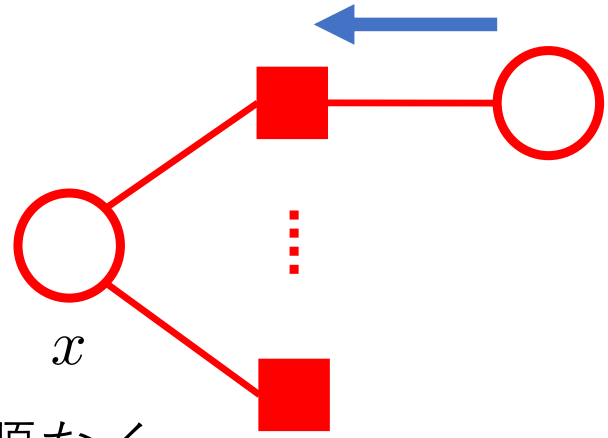


木構造を保持するため,
リンクは一つのみ.

メッセージを必ず受け取れるか　証明

仮定　あるグラフに対して適切なアルゴリズムがある.

接続されたノードはメッセージを受け取れるため，送信できる.

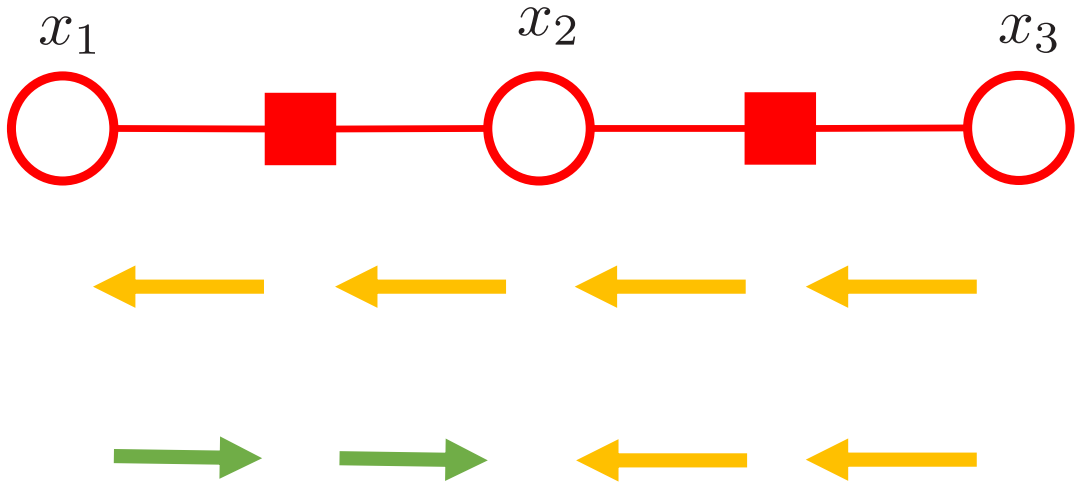


仮定から，他のノードは問題なくメッセージを伝達できる.

適切なアルゴリズムが得られた.

各変数について周辺分布を計算する

一から計算し直すことは非効率

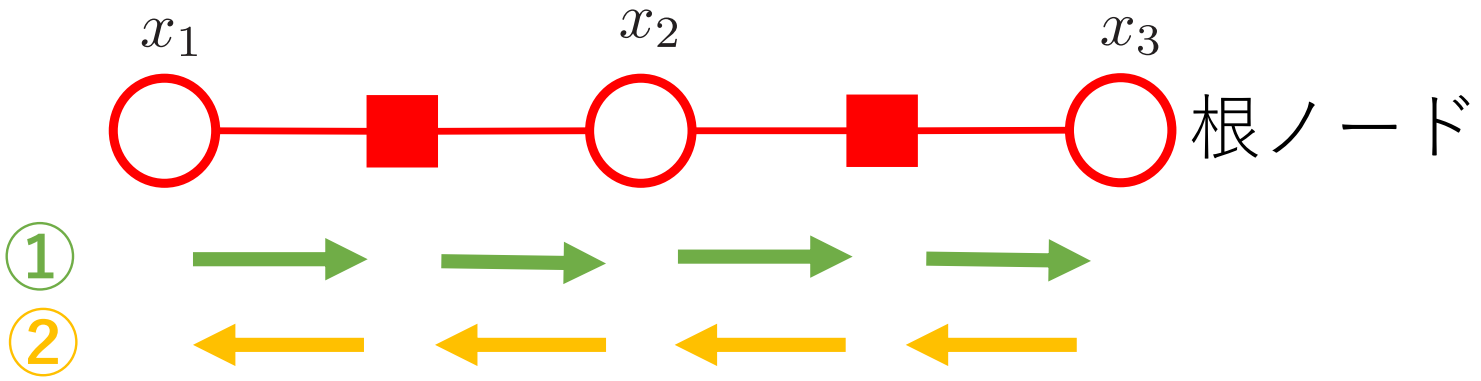


同じ計算は省こう.

各変数について周辺分布を計算する

任意のノードを根ノードとして選択.

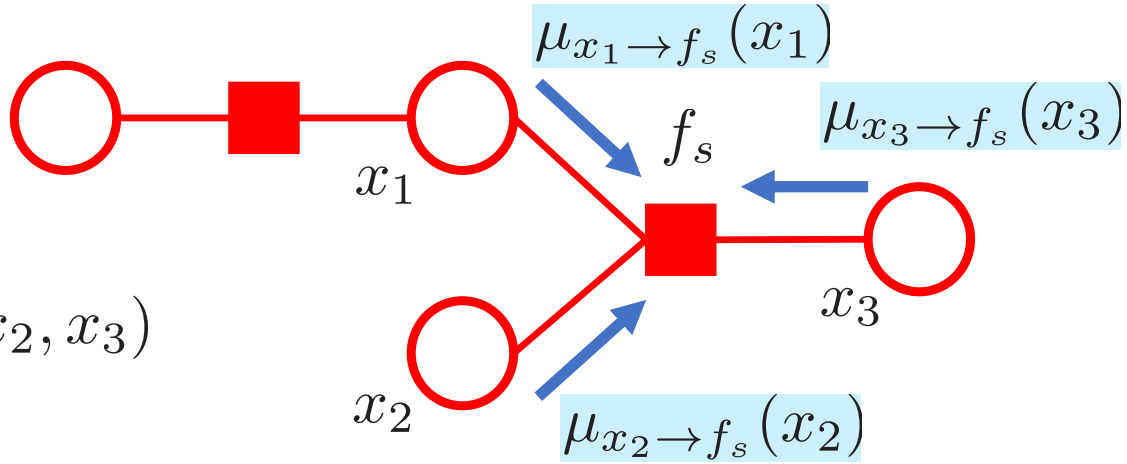
① 葉から根へメッセージ伝達.
② 根から葉へメッセージ伝達.



全てのリンクについて双方向のメッセージを計算し，保持する.
メッセージ数はリンクの２倍.

1つの因子に関連する変数集合全体上の周辺分布

- 一つの因子 f_s に関連する変数集合全体上の周辺分布 $p(\mathbf{x}_s)$ を求める.

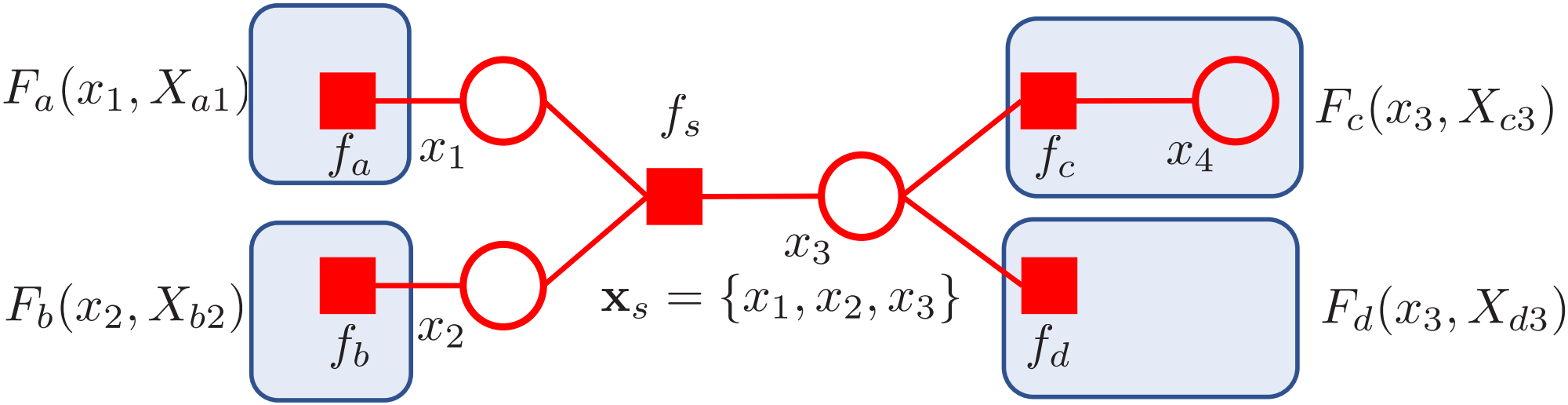


$$p(\mathbf{x}_s) = p(x_1, x_2, x_3)$$

$$p(\mathbf{x}_s) = f_s(\mathbf{x}_s) \prod_{i \in \mathrm{ne}(f_s)} \mu_{x_i \to f_s}(x_i)$$

演習8.21

$$p(\mathbf{x}_s) = \sum_{\mathbf{x}\setminus\mathbf{x}_s} p(\mathbf{x}) \qquad p(\mathbf{x}) = f_s(\mathbf{x}_s) \prod_{i\in \mathrm{ne}(f_s)} \prod_{j\in \mathrm{ne}(x_i)\setminus f_s} F_j(x_i, X_{ij})$$



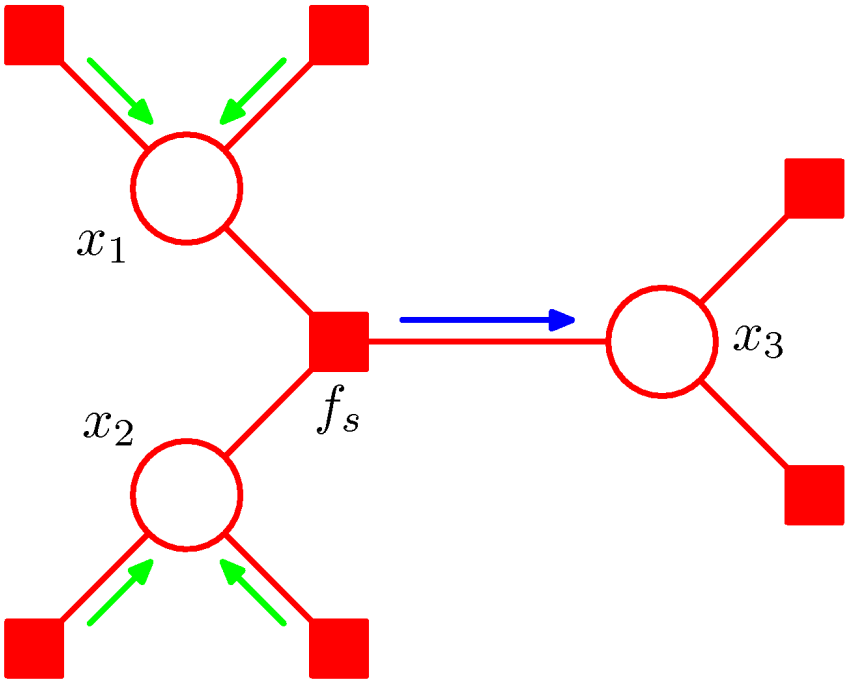
演習8.21

$$
\begin{aligned}
p(\mathbf{x}_s) &= \sum_{\mathbf{x}\setminus\mathbf{x}_s} f_s(\mathbf{x}_s) \prod_{i\in \mathrm{ne}(f_s)} \prod_{j\in \mathrm{ne}(x_i)\setminus f_s} F_j(x_i, X_{ji}) \\
&= f_s(\mathbf{x}_s) \prod_{i\in \mathrm{ne}(f_s)} \sum_{\mathbf{x}\setminus\mathbf{x}_s} \prod_{j\in \mathrm{ne}(x_i)\setminus f_s} F_j(x_i, X_{ji}) \\
&= f_s(\mathbf{x}_s) \prod_{i\in \mathrm{ne}(f_s)} \sum_{\mathbf{x}\setminus\mathbf{x}_s} G_i(x_i, X_{si}) \\
&= f_s(\mathbf{x}_s) \prod_{i\in \mathrm{ne}(f_s)} \mu_{x_i\to f_s}(x_i)
\end{aligned}
$$

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

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

積和アルゴリズムの異なる解釈



青のメッセージは緑のメッセージの積に因子f_sをかけて
x_1とx_2について周辺化することで得られる.

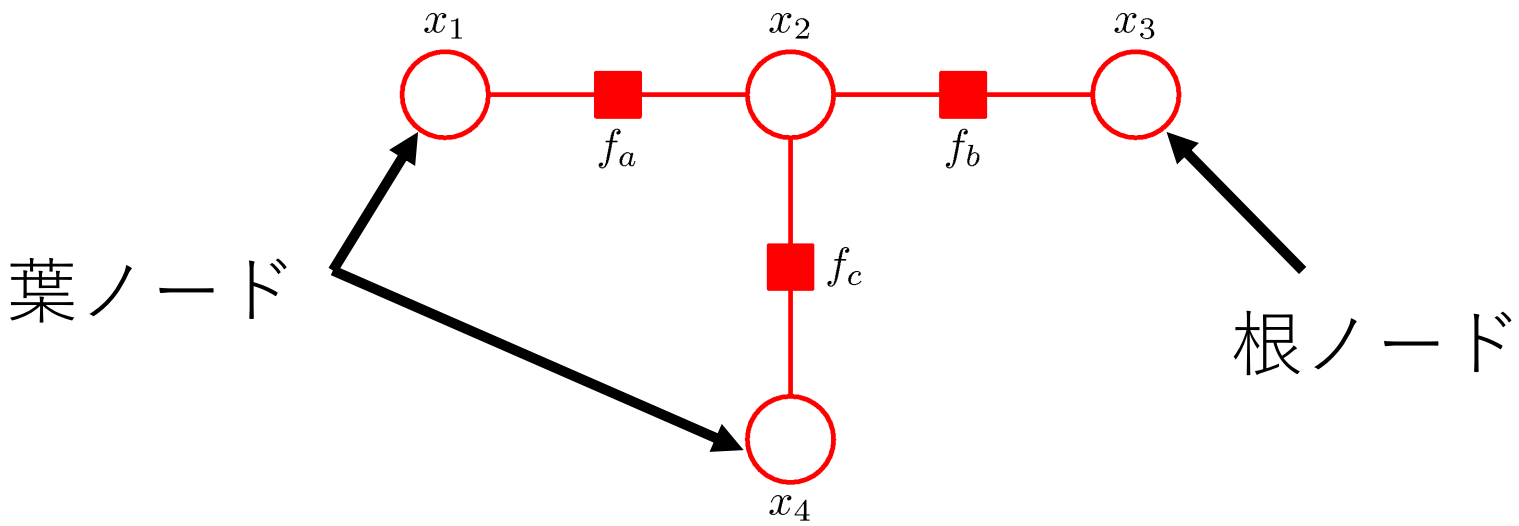
規格化

因子グラフの導出元が…

- 有向グラフなら問題なし.
- 無向グラフの場合，規格化定数が存在.

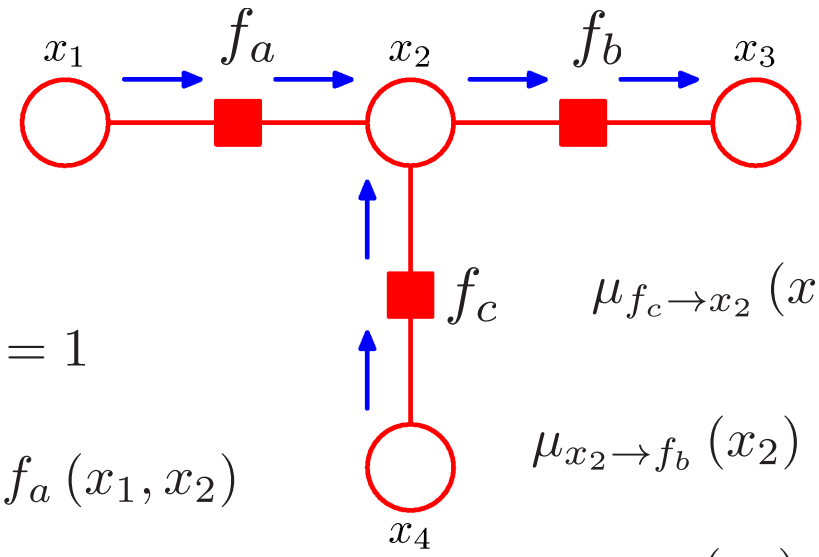
積和アルゴリズムを実行し，得られた周辺分布 $\tilde{p}(x_i)$について
規格化すればいい.

具体例



$$\tilde{p}(\mathbf{x}) = f_a\left(x_1, x_2\right) f_b\left(x_2, x_3\right) f_c\left(x_2, x_4\right)$$

具体例



$$\mu_{x_1 \to f_a}(x_1) = 1$$

$$\mu_{f_a \to x_2}(x_2) = \sum_{x_1} f_a(x_1, x_2)$$

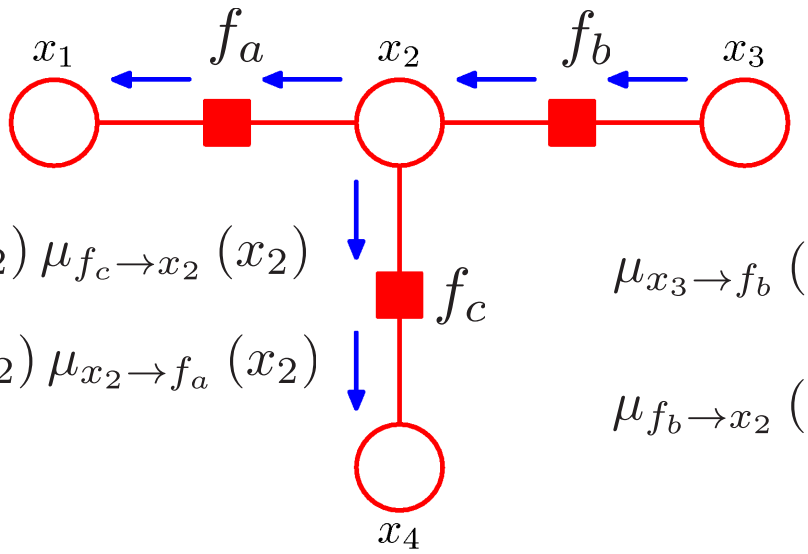
$$\mu_{x_4 \to f_c}(x_4) = 1$$

$$\mu_{f_c \to x_2}(x_2) = \sum_{x_4} f_c(x_2, x_4)$$

$$\mu_{x_2 \to f_b}(x_2) = \mu_{f_a \to x_2}(x_2)\, \mu_{f_c \to x_2}(x_2)$$

$$\mu_{f_b \to x_3}(x_3) = \sum_{x_2} f_b(x_2, x_3)\, \mu_{x_2 \to f_b}(x_2)$$

具体例



$$\mu_{x_2 \to f_a}(x_2) = \mu_{f_b \to x_2}(x_2)\,\mu_{f_c \to x_2}(x_2)$$

$$\mu_{f_a \to x_1}(x_1) = \sum_{x_2} f_a(x_1, x_2)\,\mu_{x_2 \to f_a}(x_2)$$

$$\mu_{x_3 \to f_b}(x_3) = 1$$

$$\mu_{f_b \to x_2}(x_2) = \sum_{x_3} f_b(x_2, x_3)$$

$$\mu_{x_2} \to f_c(x_2) = \mu_{f_a \to x_2}(x_2)\,\mu_{f_b \to x_2}(x_2)$$

$$\mu_{f_c \to x_4}(x_4) = \sum_{x_2} f_c(x_2, x_4)\,\mu_{x_2 \to f_c}(x_2)$$

具体例

$$
\begin{aligned}
\tilde{p}\left(x_2\right) &= \mu_{f_a \to x_2}\left(x_2\right) \mu_{f_b \to x_2}\left(x_2\right) \mu_{f_c \to x_2}\left(x_2\right) \\
&= \left[\sum_{x_1} f_a\left(x_1, x_2\right)\right] \left[\sum_{x_3} f_b\left(x_2, x_3\right)\right] \left[\sum_{x_4} f_c\left(x_2, x_4\right)\right] \\
&= \sum_{x_1} \sum_{x_3} \sum_{x_4} f_a\left(x_1, x_2\right) f_b\left(x_2, x_3\right) f_c\left(x_2, x_4\right) \\
&= \sum_{x_1} \sum_{x_3} \sum_{x_4} \tilde{p}(\mathbf{x})
\end{aligned}
$$

一部の変数が観測済みの場合

- 変数$\mathbf{x}$を隠れ変数$\mathbf{h}$と観測変数$\mathbf{v}$に分割.
  - $\mathbf{v}$の観測値を$\hat{\mathbf{v}}$とする.

$$p(\mathbf{h}, \mathbf{v} = \hat{\mathbf{v}}) = p(\mathbf{x}) \prod_i I(v_i, \hat{v}_i)$$

$$I(v, \hat{v}) = \begin{cases} 1 & (v = \hat{v}) \\ 0 & (otherwise) \end{cases}$$

$$p(\mathbf{h}|\mathbf{v} = \hat{\mathbf{v}}) = \frac{p(\mathbf{h}, \mathbf{v} = \hat{\mathbf{v}})}{p(\mathbf{v} = \hat{\mathbf{v}})}$$

max-sum アルゴリズム

max-sumのモチベーション

積和アルゴリズムによって，周辺分布を効率よく
計算できるようになった．しかし…

確率最大となる変数の組とその確率値も欲しい．

max-sum アルゴリズム

- 動的計画法のグラフィカルモデルへの応用．

max-sum アルゴリズムの対象

目標 最大の確率値を**同時に**達成する変数集合とその値を得ること.

$$\mathbf{x}^{\max} = \arg\max_{\mathbf{x}} p(\mathbf{x})$$

$$p\left(\mathbf{x}^{\max}\right) = \max_{\mathbf{x}} p(\mathbf{x})$$

個別の周辺分布を最大にする変数値 $x_i^\star$ の集合と一般に一致しない.

周辺分布を最大にする変数値

	$x=0$	$x=1$
$y=0$	0.3	0.4
$y=1$	0.3	0.0

最大値演算の性質

次の条件下で，以下の法則が成立する．

$$a \geq 0$$

法則 1： $\max(ab, ac) = a \max(b, c)$ 積演算との入れ替え

法則 2： $\max(a + b, a + c) = a + \max(b, c)$ 和演算との入れ替え

max-product アルゴリズム導出

目標　同時分布の最大値とその時の変数を効率よく求める.

$$\max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_1} \dots \max_{x_M} p(\mathbf{x})$$

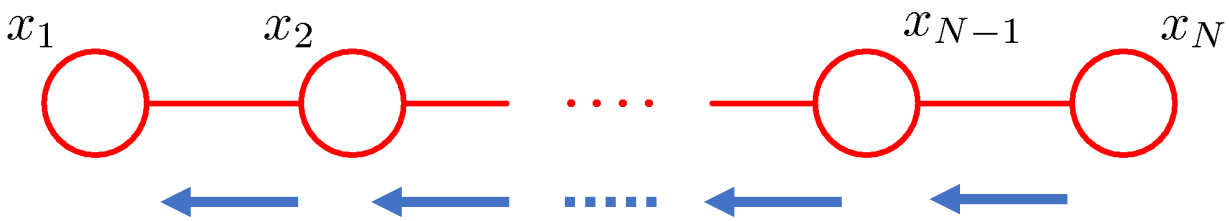
連鎖における同時分布(以下)を代入.

$$p(\mathbf{x}) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\,\psi_{2,3}(x_2, x_3)\cdots\psi_{N-1,N}(x_{N-1}, x_N)$$

max-product アルゴリズム導出

$$\max_{\mathbf{x}} p(\mathbf{x}) = \frac{1}{Z} \max_{x_1} \cdots \max_{x_N} \left[\psi_{1,2}\left(x_1, x_2\right) \cdots \psi_{N-1,N}\left(x_{N-1}, x_N\right)\right]$$

$$= \frac{1}{Z} \max_{x_1} \left[\max_{x_2} \left[\psi_{1,2}\left(x_1, x_2\right) \left[\cdots \max_{x_N} \psi_{N-1,N}\left(x_{N-1}, x_N\right)\right] \cdots\right]\right]$$



max-product アルゴリズム

- メッセージパッシング

$$\mu_{x\to f}(x) = \prod_{l\in \mathrm{ne}(x)\setminus f} \mu_{f_l\to x}(x)$$

$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_M} \left[f\left(x, x_1, \ldots, x_M\right) \prod_{m\in \mathrm{ne}(f)\setminus x} \mu_{x_m\to f}\left(x_m\right) \right]$$

- 葉ノード初期化

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

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

アンダーフロー対策

小さな値の積を取り続けるためアンダーフローを起こしやすい.

- 対策: 対数をとる.

$a > b$ ならば $\ln a > \ln b$

$$\ln\left(\max_{\mathbf{x}} p(\mathbf{x})\right) = \max_{\mathbf{x}} \ln p(\mathbf{x})$$

max-sum アルゴリズム

- メッセージパッシング

$$\mu_{x\to f}(x) = \sum_{l\in \mathrm{ne}(x)\setminus f} \mu_{f_l\to x}(x)$$

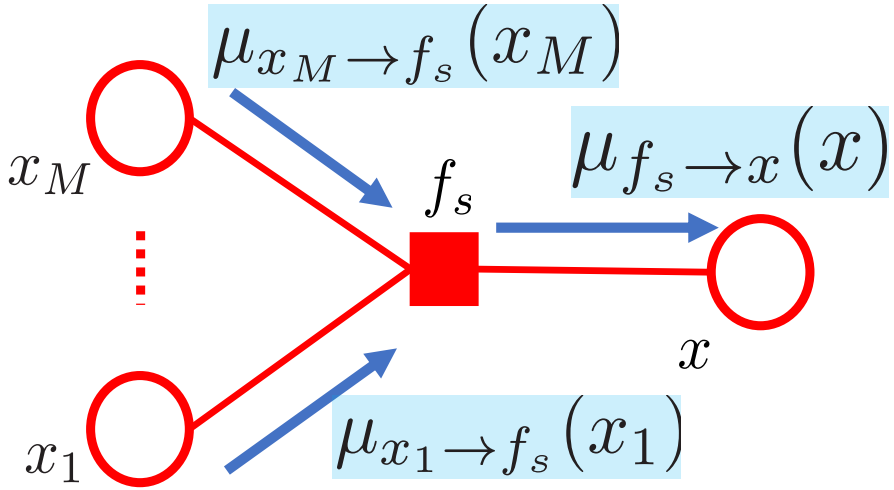
$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_M} \left[\ln f\left(x, x_1, \ldots, x_M\right) + \sum_{m\in \mathrm{ne}(f)\setminus x} \mu_{x_m\to f}\left(x_m\right)\right]$$

- 葉ノード初期化

$$\mu_{x\to f}(x) = 0$$

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

最大確率



$$\ln p^{\max} = \max_{x} \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]$$

最大確率を与える根ノードの変数値

$$x^{\max} = \arg\max_{x} \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]$$

- 最大値を与える変数の組が複数存在する場合，失敗する．
  - 上記式は変数集合を扱わないため．

バックトラック(back-tracking)一般式

$$x^{\max} = \arg\max_{x} \left[\sum_{s \in \mathrm{ne}(x)} \mu_{f_s \to x}(x) \right]$$

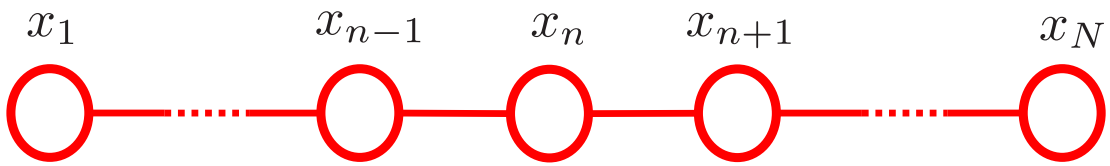
$$\phi(x) = \arg\max_{x_1, \ldots, x_M} \left[\ln f(x, x_1, \ldots, x_M) + \sum_{m \in \mathrm{ne}(f) \setminus x} \mu_{x_m \to f}(x_m) \right]$$

最大確率を与える状態が複数存在する場合は，ランダムに選択.

連鎖の場合のバックトラック

$$\mu_{x_1 \to f_{1,2}}(x_1) = 0$$

$$\mu_{x_n \to f_{n,n+1}}(x_n) = \mu_{f_{n-1,n} \to x_n}(x_n)$$

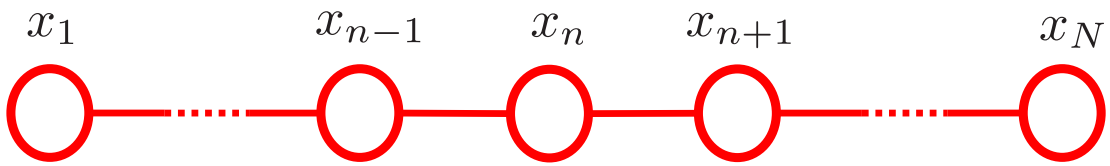


$$\mu_{f_{n-1,n} \to x_n}(x_n) = \max_{x_{n-1}} \left[\ln f_{n-1,n}(x_{n-1}, x_n) + \mu_{x_{n-1} \to fn-1,n}(x_{n-1}) \right]$$

メッセージを計算する際に，x_{n-1} のどの状態で x_n の各状態に対して最大確率を与えたか保存する．

$$\phi(x_n) = \arg\max_{x_{n-1}} \left[\ln f_{n-1,n}(x_{n-1}, x_n) + \mu_{x_{n-1} \to f_{n-1,n}}(x_{n-1}) \right]$$

連鎖の場合のバックトラック



$$x_N^{\max} = \arg\max_{x_N} \left[\mu_{f_{N-1,N} \to x_N}(x_N)\right]$$

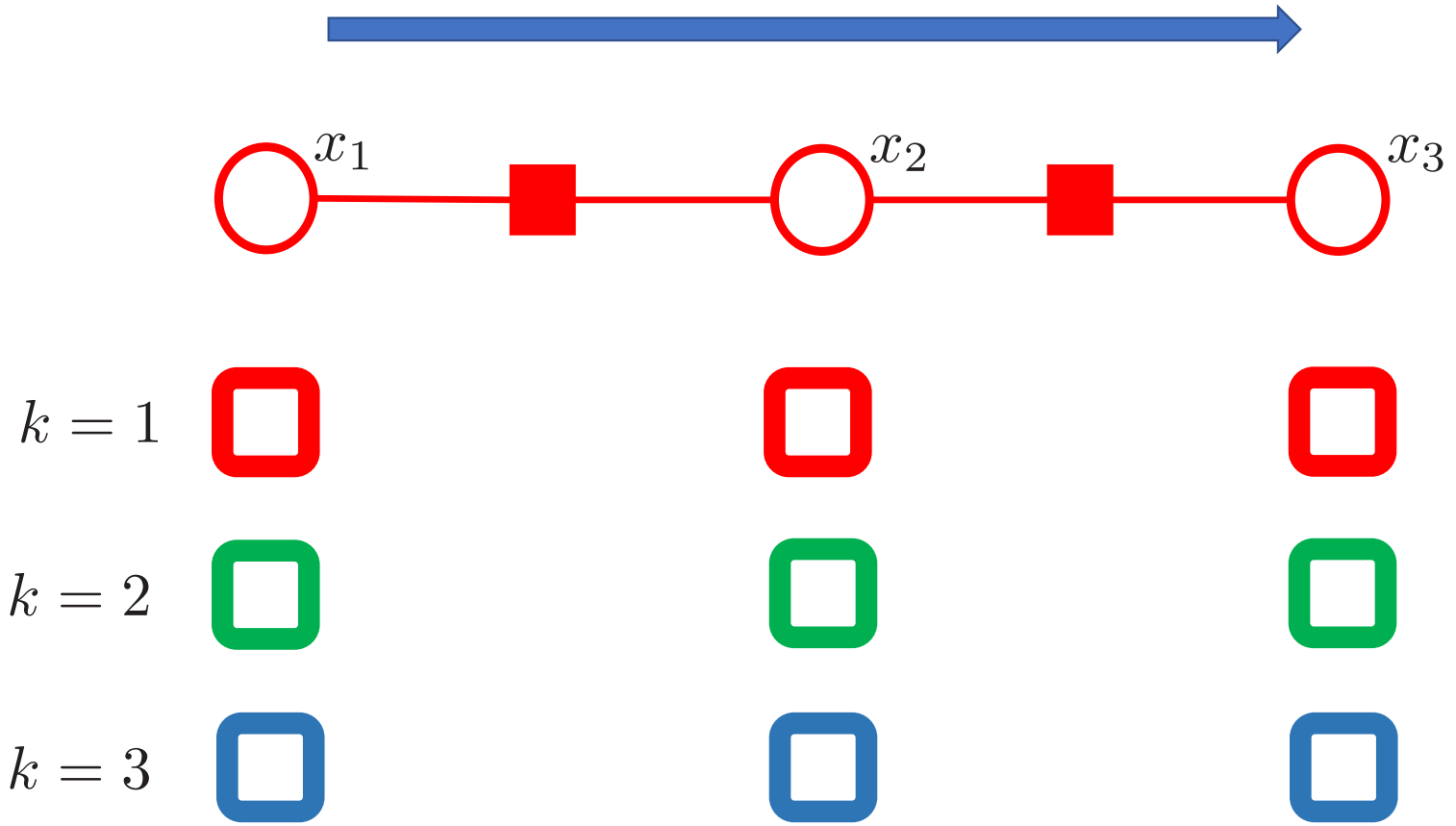
送られてきたメッセージの要素で最大となるときの状態を選ぶだけ.

以下の式を使って，以前のノードの状態をたどる.

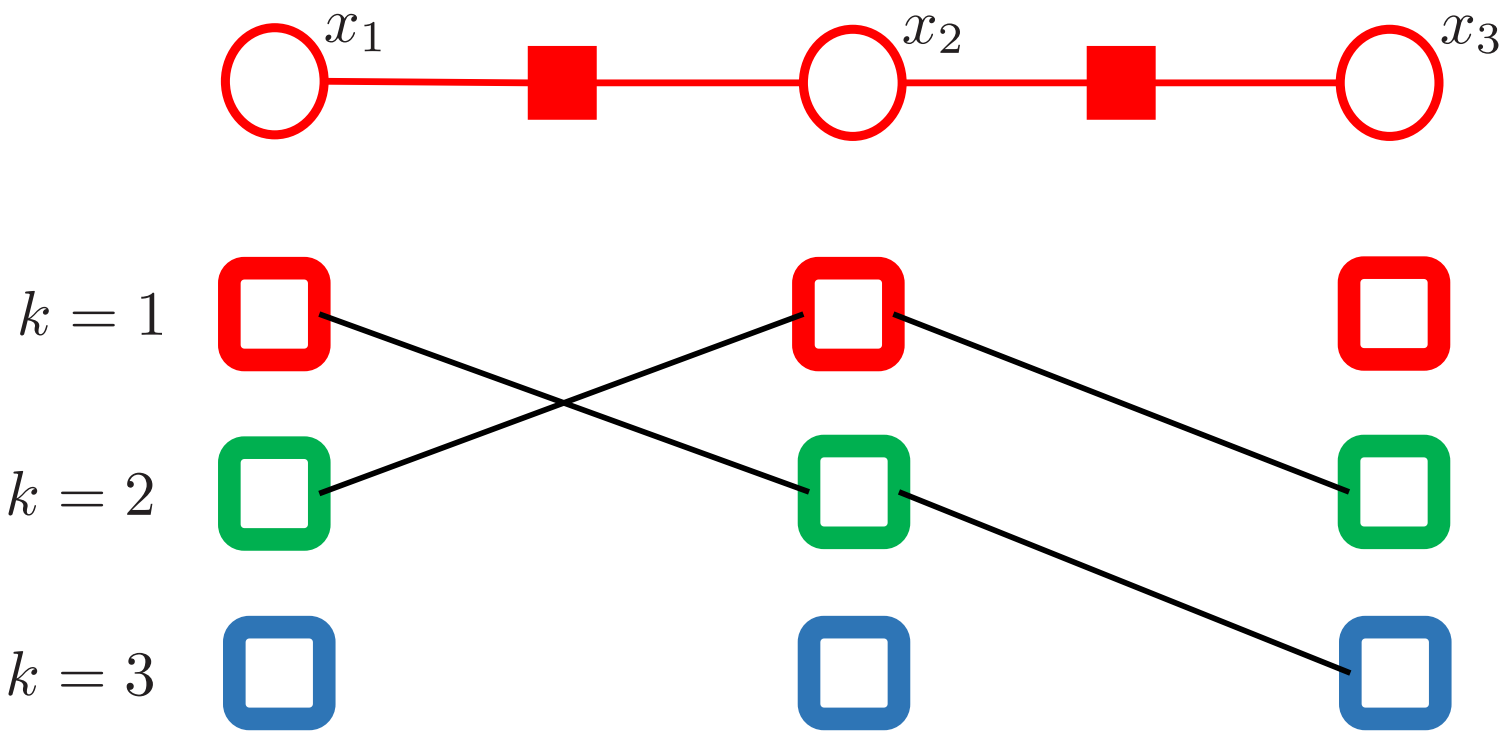
$$x_{n-1}^{\max} = \phi\left(x_n^{\max}\right)$$

$$\phi(x_n) = \arg\max_{x_{n-1}} \left[\ln f_{n-1,n}(x_{n-1}, x_n) + \mu_{x_{n-1} \to f_{n-1,n}}(x_{n-1})\right]$$

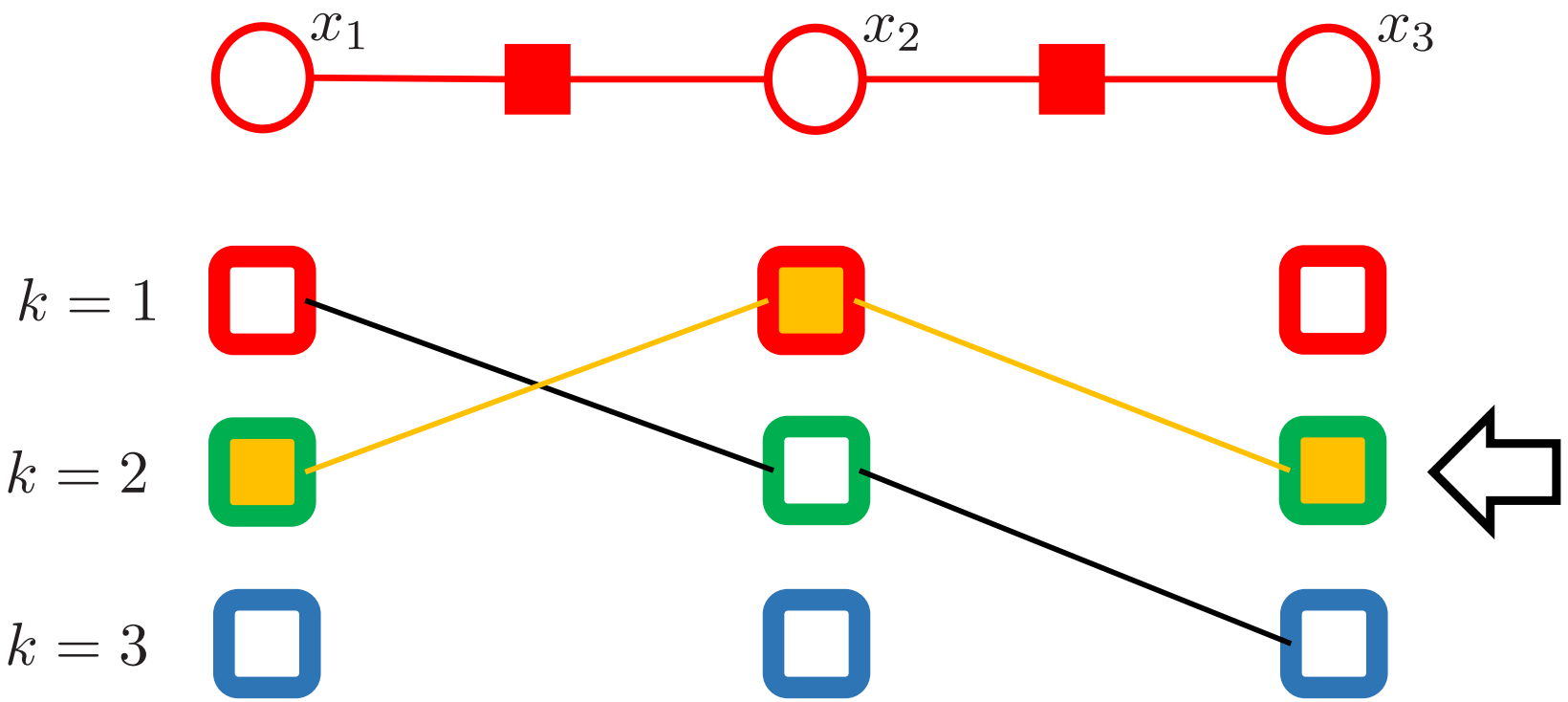
連鎖の場合のバックトラックの様子



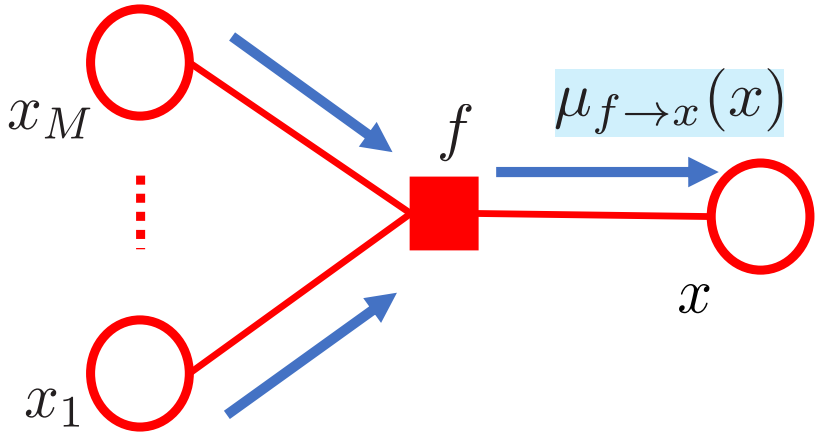
連鎖の場合のバックトラックの様子



連鎖の場合のバックトラックの様子



木構造因子グラフに対する拡張



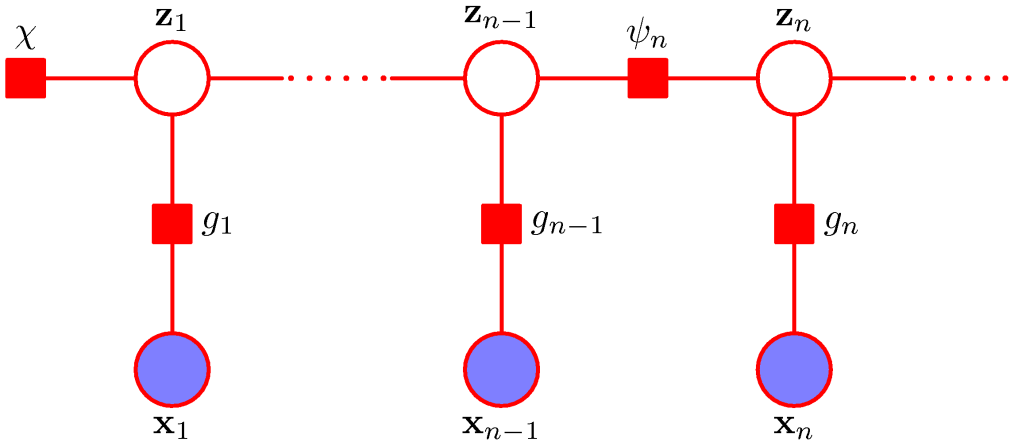
$$x^{\max} = \arg\max_{x} \left[\sum_{s\in \mathrm{ne}(x)} \mu_{f_s\to x}(x) \right]$$

$$\mu_{f\to x}(x) = \max_{x_1,\ldots,x_M} \left[\ln f\left(x, x_1, \ldots, x_M\right) + \sum_{m\in \mathrm{ne}(f)\setminus x} \mu_{x_m\to f}\left(x_m\right) \right]$$

$$\phi\left(x\right) = \arg\max_{x_1,\ldots,x_M} \left[\ln f\left(x, x_1, \ldots, x_M\right) + \sum_{m\in \mathrm{ne}(f)\setminus x} \mu_{x_m\to f}\left(x_m\right) \right]$$

Viterbi アルゴリズム

- HMMにおいてもっとも確からしい隠れ状態の系列を推定するアルゴリズム
- max-sum アルゴリズムの応用.
  - 詳しくは13.2節.



観測変数の利用

積和と同じ.

ICMとの比較

- ICMの各ステップの計算は max-sum アルゴリズムよりも簡単.

  それぞれのメッセージは…

  - ICM: 送信側ノードの条件付き分布が最大となる状態(数値)だけ.
  - max-sum: 受信ノード変数の関数.

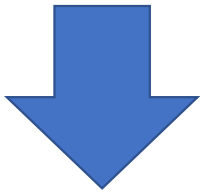
- ICMは max-sum アルゴリズムと異なり，木構造グラフに対して大域的最大点を得る保証はない.

一般のグラフにおける厳密推論

一般のグラフへ拡張

前節までの内容

- 木構造グラフであれば，効率よく厳密推論できる．



一般のグラフへ拡張．

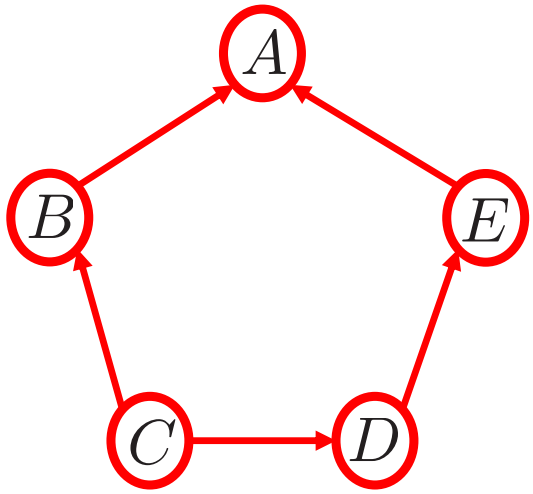
ジャンクションツリーアルゴリズム

- 任意の形のグラフを木構造グラフへ変換．
- 得た木構造グラフで厳密推論する．

ジャンクションツリー構築

1. 有向グラフはモラル化により無向グラフへ変換.
2. グラフを三角形分割する.
   - 4つ以上のノードを含む弦のない閉路があれば，リンクを追加して弦のない閉路をなくす.
3. 三角形分割したグラフからジャンクションツリーを構築.
   - ジャンクションツリーの各ノードは三角形分割グラフの各極大クリークに対応.
   - リンクは共通の変数を持つクリークの組に対して選択.
   - 極大全域木を与えるリンクが選ばれる.

ジャンクションツリー構築

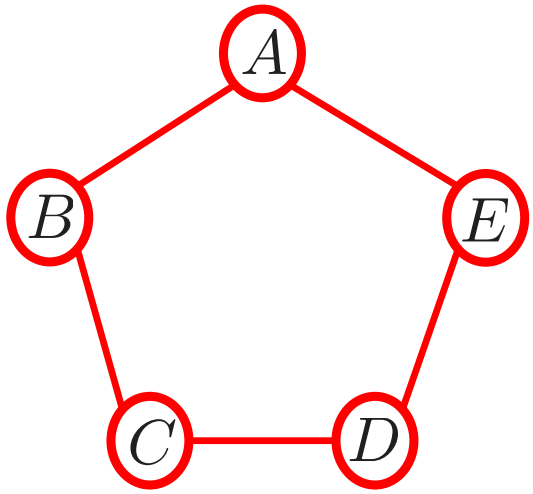


1. 有向グラフはモラル化により
   無向グラフへ変換.

2. グラフを三角形分割する.

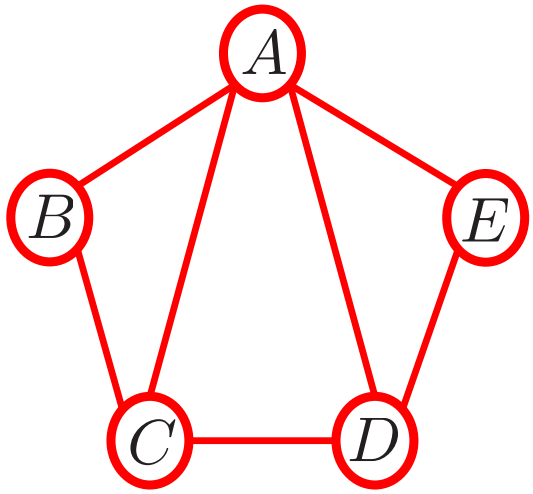
3. 三角形分割したグラフから
   ジャンクションツリーを構築.

ジャンクションツリー構築



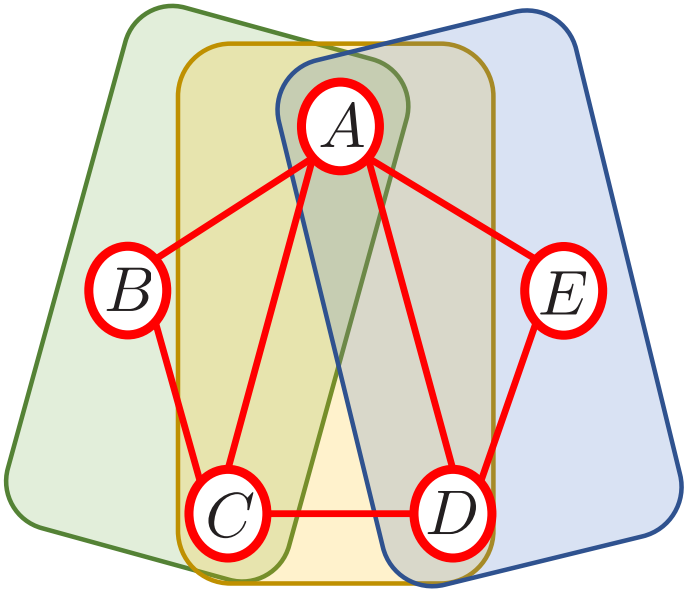
1. ~~有向グラフはモラル化により~~
   無向グラフへ変換.
2. グラフを三角形分割する.
3. 三角形分割したグラフから
   ジャンクションツリーを構築.

ジャンクションツリー構築



1. ~~有向グラフはモラル化により無向グラフへ変換.~~
2. ~~グラフを三角形分割する.~~
3. 三角形分割したグラフからジャンクションツリーを構築.

ジャンクションツリー構築



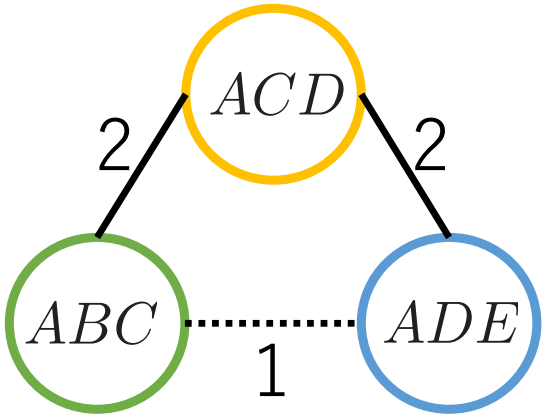
1. ~~有向グラフはモラル化により無向グラフへ変換.~~
2. ~~グラフを三角形分割する.~~
3. 三角形分割したグラフからジャンクションツリーを構築.

ジャンクションツリー構築

1. 有向グラフはモラル化により無向グラフへ変換.
2. グラフを三角形分割する.
3. 三角形分割したグラフからジャンクションツリーを構築.

ジャンクションツリーの性質

- 木構造無向グラフ.
- 可能な木の中で，重み最大の木.
  - リンクの重みはクリークの共有するノード数.
- 連結横断特性.
  - ある変数が2つのクリークに含まれていれば，その変数は両者クリーク間の経路上の全てのクリークに含まれる.
  - 変数に関する推論がグラフ全体で整合することを保証する.



ジャンクションツリーアルゴリズムの性質

- 任意のグラフに厳密解を与える.
- 一般にこれよりも軽い計算法がない.

- 計算量は極大クリークに含まれる変数の数に依存.
  - 離散変数の場合，指数的に増大.
  - グラフの木幅が重要.（「木幅」=極大クリークの含む変数の数)
  - 木幅が大きいと実行不可能.

- グラフ上の操作だけで正確で効率的な計算手順を構成している.

ループあり確率伝播

実用のために

厳密推論が困難であることが多い.

- 変分近似(10章)
- 確率的なサンプリング(11章)
  - サンプリング法
  - モンテカルロ法

ループあり確率伝播

- 積和アルゴリズムのメッセージパッシングルールは局所的
  - ループがあっても適用はできる

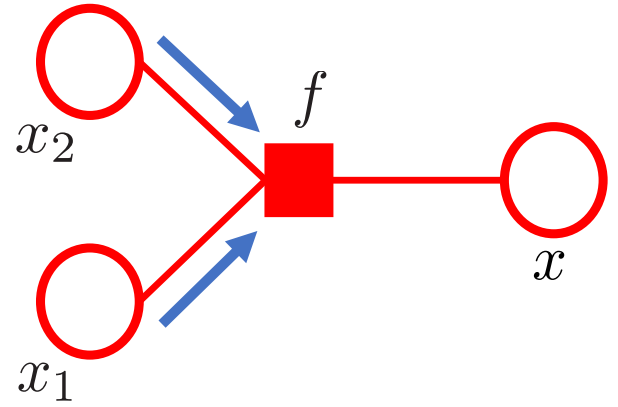
- グラフは閉路を持つので情報はグラフを何度も伝わる
  - 収束するか否かはモデル次第

- **課題**
  - メッセージパッシングのスケジュール
  - メッセージの初期化方法

メッセージの初期化

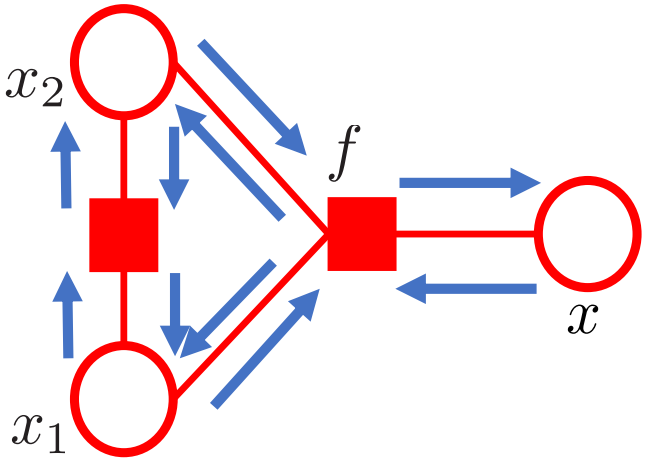
ループなし

葉ノードを初期化すれば良い.



ループあり

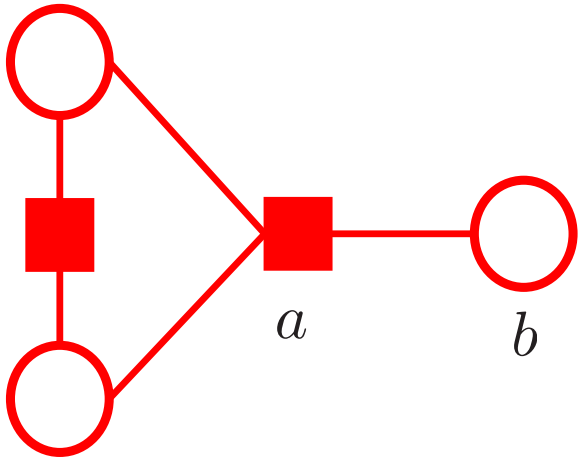
初めに全てのリンクで初期メッセージが
両方向に伝達されていると仮定.



メッセージパッシングのスケジュール

- **フラッディングスケジュール**
  - 各時刻において全てのリンクに沿って両方向に同時にメッセージが送信される.
- **直列スケジュール**
  - 各時刻において1つのメッセージしか送信しない.
- **保留メッセージだけ送信**
  - ループがあるグラフでは保留メッセージが常に存在し得る.

保留メッセージ



ノードaがノードbに送信したのちに，他ノードからメッセージを受信した時，aからbへ向かうリンクに対して**保留メッセージ**を持っているという．
木構造グラフは保留メッセージを持たない(演習8.29)．

ループあり確率伝播が有効な例

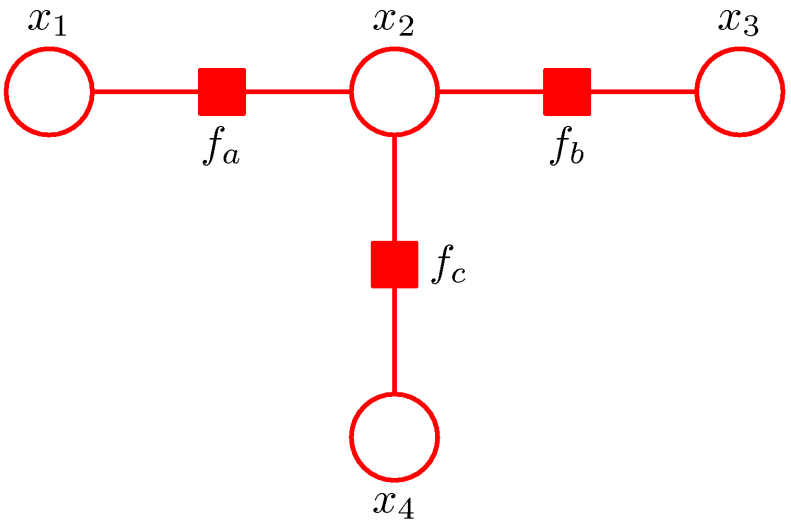
- 良い結果を与えないものもある.

- 良い結果を与える例
  - ある種の誤り訂正符号の復号のための最新鋭(?)アルゴリズム.

グラフ構造の学習

グラフから推論

グラフィカルモデル



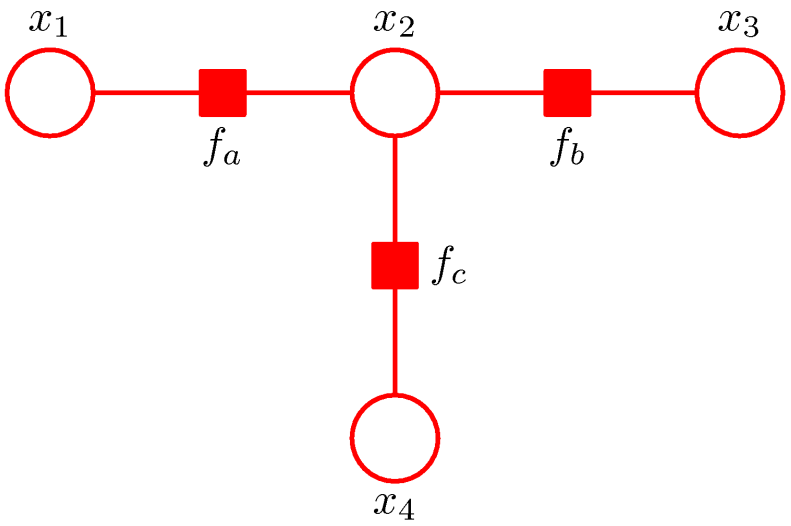
$$\mathbf{x}^{\max} = \arg\max_{\mathbf{x}} p(\mathbf{x})$$

$$p\left(\mathbf{x}^{\max}\right) = \max_{\mathbf{x}} p(\mathbf{x})$$

グラフあり き

データからグラフ構造

観測データ集合 $\mathcal{D}$



課題

- 可能なグラフ構造の空間の定義.
- グラフ構造のスコアとしての尺度の定義.

ベイズ的なグラフ構造(モデル)学習

$$p(m|\mathcal{D}) \propto p(m)p(\mathcal{D}|m)$$

グラフ構造mの事後分布

グラフ構造mの事前分布

モデルエビデンス（スコア）

課題

- モデルエビデンスの計算には全ての潜在変数の周辺化が必要.
- 可能なグラフ構造の数はノード数に対して指数的に増大.

各ノード間について，リンクを張る/張らない