概要

前回は木構造における max-sum アルゴリズムを学んだ。 max-sumアルゴリズムは同時分布の最大値を求める方法であり、積和アルゴリズム同様のメッセージパッシングによって達成されることを確かめた。しかし、同時分布の最大値を得ることはできても、最大値を達成する変数の値を得るためには工夫が必要であることを確認した。

8.4.5 の続きでは、同時分布の最大値を与える変数を得るために、メッセージパッシングを拡張する。具体的には、各変数がどの値で最大状態を与えたかの情報を関数として保存するようにする。

8.4.6では、ループを持つグラフに対してもメッセージパッシングの枠組みを取り入れるため、ジャンクションツリーアルゴリズムによって（無理やり）木構造に変換する方法を紹介する。この方法ではループが発生する部分のノード群を一つのノードとして扱うことで、全体としては木構造としてメッセージパッシングを行うことができるようになる。

8.4.7では、ループを持つグラフに対して、木構造に変換することなくメッセージパッシングを実行する手続きを紹介する。この方法では、メッセージを飛ばすタイミングを決定するスケジュールが重要になる（が、結局どれが良いのかはわからない）。

8.4.8ではグラフ構造自体を学習する方法について触れているが、ここは読み合わせとしたい。

8.4.5 max-sum アルゴリズム（続き）

同時分布の最大値を与える値を、個々の変数について求める方法を扱う。前回は、メッセージパッシングを使い、そのノードに送られるメッセージの和が最大になる値を解とする方法を考えたが、これはうまくいかなかった。というのも、最大値を与える x の値が複数存在するときにこれらからランダムに選択すると必ずしも最大点とならないケースがあった。

最大値を与える変数の値

問題点：
最大の同時分布を与える変数値の組が複数通りある場合に、各変数について別々に考えていると、違う組に属する値を選んでしまい、全体として最大値を与える変数値の組にならない場合がある。

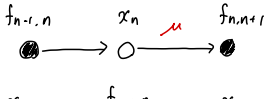
例として右の分布を考える。

	x=0	x=1
y=0	0.0	0.5
y=1	0.5	0.0

最大の同時分布を与える変数の組は (x,y) = (0,1)または(1,0) だが、xについてx=0を選び、yについてy=0を選んでしまうと、これらを合わせた p(x=0,y=0) は0になってしまう。

この問題に対して、各変数がどの値で最大状態を与えたかを表す情報を保持することで解決する。具体的に連鎖の例で考えてみると、メッセージの送り先 x_n の値に応じて、最大値を得る送り主 x_{n-1} の値を逐一保存すればよい。

まず連鎖のおさらいとして、メッセージは次のように送られるのであった。



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

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

自身の因子　　入ってくるメッセージ

ここで、第２式の $f_{n-1,n}$ から　x_n　に送られるメッセージは　x_n の関数であることに注意する。このメッセージを最大にする　x_{n-1}　の値を考えると次のように表される。

$$\phi_{n-1}(x_n) = \arg\max_{x_{n-1}} \left[\ln f_{n-1,n}(x_{n-1}, x_n) + \mu_{x_{n-1} \to f_{n-1,n}}(x_{n-1}) \right] \qquad (8.101)$$

繰り返しになるが、この式は　$f_{n-1,n}$　からのメッセージを最大にする　x_{n-1}　の値が、x_n　の値に依存して決定することを示している。（初め読んだときにこのイメージが掴めず苦戦したのであえて強調してます）

この情報を保持しながら順伝播を根ノードまで実行し、最後のノード x_N　の最も確からしい値が得られた後、(8.101) を再帰的に実行して全ての変数の値を取得すれば良い。すなわち、以下のように実行していく。

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

根ノードなので、ここで値が確定する

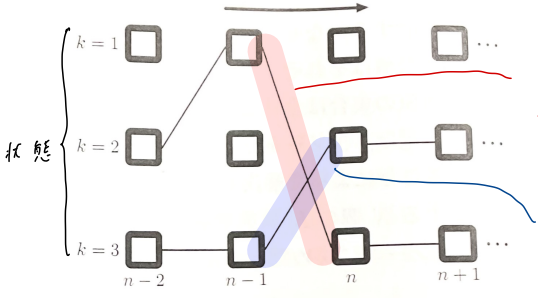
$$x_{N-1}^{max} = \phi_{N-1}(x_N^{max})$$

$$x_{N-2}^{max} = \phi_{N-2}(x_{N-1}^{max})$$

$$\vdots$$

$$x_1^{max} = \phi_1(x_2^{max})$$

これを格子図で説明したのが次の図である。この格子図は、各変数　x_n　がどの状態をとる時に最大同時分布を得るかを示している。

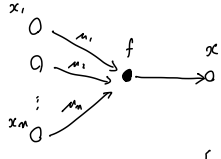


このリンクは，$x_n=3$ のときに，$x_{n-1}=1$ を選択するとメッセージが最大になることを示す，つまり，$\phi_{n-1}(x_n=3)=1$

同様に，このリンクは $\phi_{n-1}(x_n=2)=3$ を示す，

最終的に得られた　x_n^{max} を逆むきにたどることで、最大同時分布を達成する各変数の値が求まる。このときの操作をバックトラックと呼ぶ。

一般的な木構造についても同様の枠組みで解決できる。すなわち、因子ノードfからxに送られてくるメッセージを最大にする隣接変数集合　$x_1,\cdots,x_M$　の値を保持すれば良い。



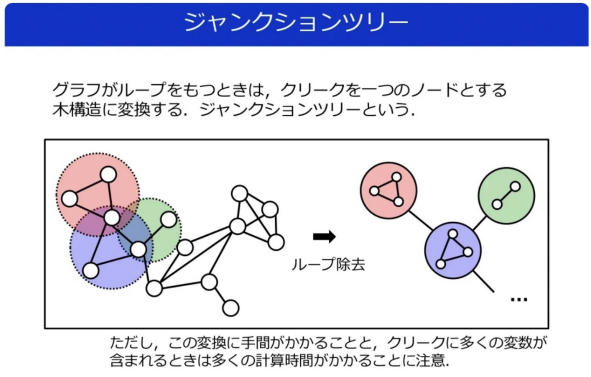
$$\phi(x) = \underset{x_1,\cdots,x_M}{\arg\max}\left[\ln f + \sum_{i=1}^{M}\mu_i\right]$$

最大値を与える $x_1,\cdots,x_M$ ベクトル

131頁の「証拠」の話とICMとの比較はよくわからなかったがそこまで重要でもなさそうなのでパス

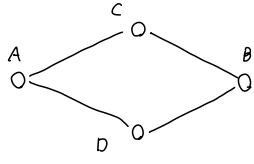
8.4.6 一般のグラフにおける厳密推論

これまで木構造の因子グラフが得られた元での議論をしてきたが、ループを持つグラフについてもメッセージパッシングを適用する方法がある。ジャンクションツリーアルゴリズムは、ループを含むグラフの一部を一つの変数として扱うことで、マクロな視点では木構造上のメッセージパッシングを実行できるようにするグラフ変換方法である。

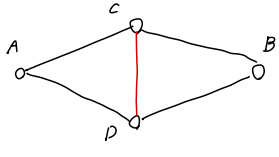


参照：https://www.slideshare.net/Kawamoto_Kazuhiko/ss-35483453

具体的な方法を、次の閉路 A-C-B-D-A を持つグラフで考える。



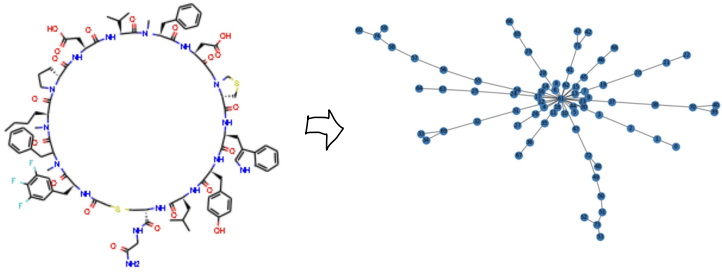
まず、4つ以上のノードを含む弦のない閉路を見つけたら、リンクを追加して弦のない閉路をなくす。



次に極大クリークを対応する単一のノードに置き換え、共通変数を持つクリーク同士にリンクを張る。

以上のようにしてジャンクションツリーを作成する。

詳細は追えてないが、分子構造をグラフとしてジャンクションツリーを構成している記事があった。
https://udnp.hatenablog.com/entry/2019/10/14/130215



クリークを1ノードとして扱うことでメッセージパッシングを扱えるようになるメリットはあるが、実は極大クリークが大きすぎる場合には計算量の問題が生じる。というのも、1ノード内の扱いとしてはクリーク内部の同時分布を扱う必要があり、周辺化や最大同時分布を求める際に計算量が指数的に増加する。これは、K状態をとる離散変数について周辺化をしようとすると、 $K^{\prime\prime}$ の計算量がかかってししまうというp.109と同様の問題がクリーク内部で発生するのである。

そのため、極大クリークが持つ変数の数の最大値（＝木幅＋1）が大きい場合にはジャンクションツリーがうまく機能しなくなる。

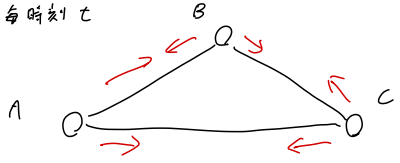
木幅の定義は下記URLに詳しくあるが、「木幅＝極大クリークが持つ変数の最大値－1」という理解さえしていれば参照する必要はない：
https://mizuwater0.hatenablog.com/entry/2019/01/05/185327#f-aaab66c1

8.4.7 ループあり確率伝播

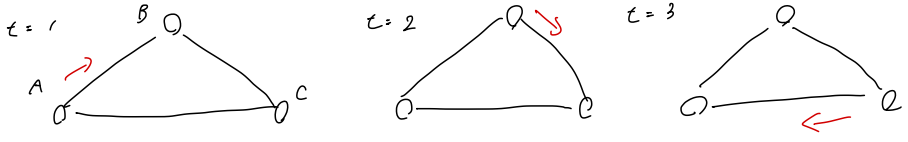
前節では木構造への変換を使いループのあるグラフへの対処をしたが、本節ではループのあるグラフのままメッセージパッシングを実施する手法について扱う。これまで扱ってきたメッセージパッシングの枠組みは、隣接ノード間でのやりとりのみに閉じた計算であるので、ループがあっても適用可能である。この方法はモデルによっては収束するが、収束しない場合もある。

ループあり確率伝播を行うための手続きは、メッセージパッシングのスケジュールに依存する。その方法の例として、フラッディングスケジュールと、直列スケジュールがある。

■ 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
各時刻に一つのメッセージしか送信しない。



このようなメッセージパッシングを、値が収束するまで実施すれば良い。

また、保留メッセージという概念を考慮すると、必ずしもメッセージを送らなくてもよいタイミングが存在することもある。あるノードがメッセージを受け取ると、そのノードが発するメッセージが変化するので、そのノードは隣接ノードに新たにメッセージを送る必要が出る。これが保留メッセージである。逆に保留「でない」メッセージについては、メッセージの内容が変化していないため送る必要がない。

8.4.8 グラフ構造の学習

（読み合わせ）

第2段落、「その分布に関する平均を計算することによって予測分布を求めるのが理想である」がよくわからなかった。