Графические модели и байесовский вывод на них

Сергей Николенко

Computer Science Club, Казань, 2014

Outline

В чём же проблема

- В предыдущих лекциях мы рассмотрели задачу байесовского вывода, ввели понятие сопряжённого априорного распределения, поняли, что наша основная задача – найти апостериорное распределение.
- Но если всё так просто – взяли интеграл, посчитали, всё получилось – о чём же здесь целая наука?
- Проблема заключается в том, что распределения, которые нас интересуют, обычно слишком сложные (слишком много переменных, сложные связи).
- Но, с другой стороны, в них есть дополнительная структура, которую можно использовать, структура в виде независимостей и условных независимостей некоторых переменных.

Графические модели с направленными графами

- Пример: рассмотрим распределение трёх переменных и запишем его по формуле полной вероятности:

$$p(x, y, z) = p(x \mid y, z)p(y \mid z)p(z).$$

- Теперь нарисуем граф, в котором стрелки указывают, какие условные вероятности заданы.
- Пока граф полносвязный, это нам ничего не даёт – любое распределение $p(x_1, \ldots, x_n)$ так можно переписать.
- Но если некоторых связей *нет*, это даёт нам важную информацию и упрощает жизнь.

Графические модели с направленными графами

- Рассмотрим направленный ациклический граф на вершинах $x_1, \ldots, x_k$ и зададим в каждой вершине распределения $p(x_i \mid \mathrm{pa}(x_i))$. Тогда будем говорить, что граф с этими локальными распределениями является графической моделью (байесовской сетью доверия) для совместного распределения вероятностей

$$p(x_1, \ldots, x_k) = \prod_{i=1}^{k} p(x_i \mid \mathrm{pa}(x_i)).$$

- Другими словами, если мы можем разложить большое совместное распределение в произведение локальных распределений, каждое из которых связывает мало переменных, это хорошо. :)

Графические модели с направленными графами

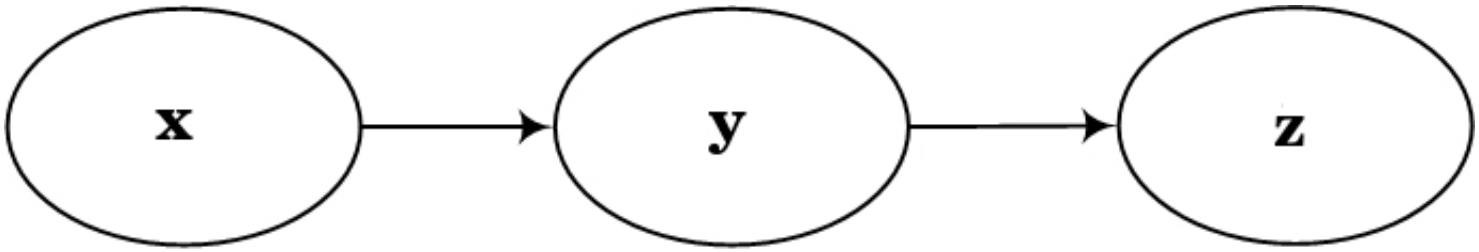
- Пример: обучение параметров распределения по нескольким экспериментам (плашки, можно нарисовать параметры явно):

$$p(x_1, \ldots, x_n, \theta) = p(\theta) \prod_{i=1}^{n} p(x_i \mid \theta).$$

- Что можно сказать о (не)зависимости случайных величин x_i и x_j?
- Задача вывода на графической модели: в некоторой части вершин значения наблюдаются, надо пересчитать распределения в других вершинах (подсчитать условные распределения). Например, из этой модели получатся и задача обучения параметров, и задача последующего предсказания.

Графические модели с направленными графами

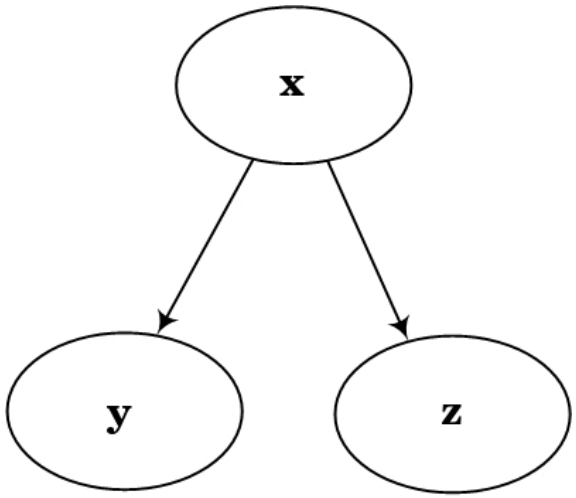
- d-разделимость – условная независимость, выраженная в структуре графа:
  - последовательная связь, $p(x, y, z) = p(x)p(y \mid x)p(z \mid y)$:
    - если y не наблюдается, то $p(x, z) = p(x) \int p(y \mid x)p(z \mid y)\mathrm{d}y = p(x)p(z \mid x)$;
    - если y наблюдается, то $p(x, z \mid y) = \frac{p(x,y,z)}{p(y)} = \frac{p(x)p(y|x)p(z|y)}{p(y)} = p(x \mid y)p(z \mid y)$, получили условную независимость.



Последовательная связь

Графические модели с направленными графами

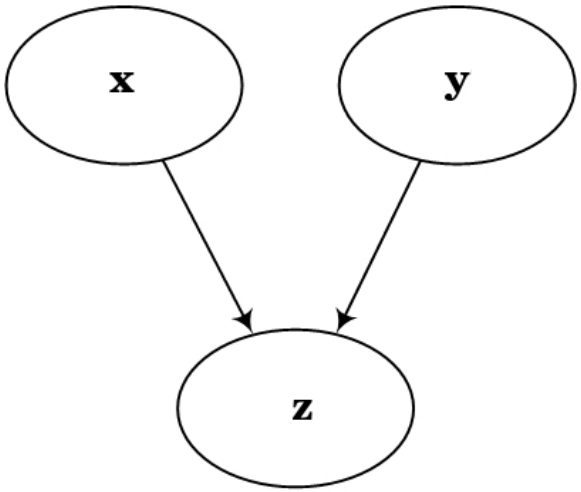
- расходящаяся связь, $p(x, y, z) = p(x)p(y \mid x)p(z \mid x)$, – так же:
  - если y не наблюдается, то
    $p(x, z) = p(x)p(z \mid x) \int p(y \mid x)\mathrm{d}y = p(x)p(z \mid x)$;
  - если y наблюдается, то
    $p(x, z \mid y) = \frac{p(x,y,z)}{p(y)} = \frac{p(x)p(y|x)p(z|x)}{p(y)} = p(x \mid y)p(z \mid y)$,
    получили условную независимость.



Расходящаяся связь

Графические модели с направленными графами

- Интересный случай – сходящаяся связь, $p(x, y, z) = p(x)p(y)p(z \mid x, y)$:
  - если z не наблюдается, то $p(x, y) = p(x)p(y)$, независимость есть;
  - если z наблюдается, то $p(x, y \mid z) = \frac{p(x,y,z)}{p(z)} = \frac{p(x)p(y)p(z|x,y)}{p(z)}$, и условной независимости нету.



Сходящаяся связь

Обобщение: если наблюдается хотя бы один из потомков z, уже может не быть независимости между x и y.

Графические модели с направленными графами

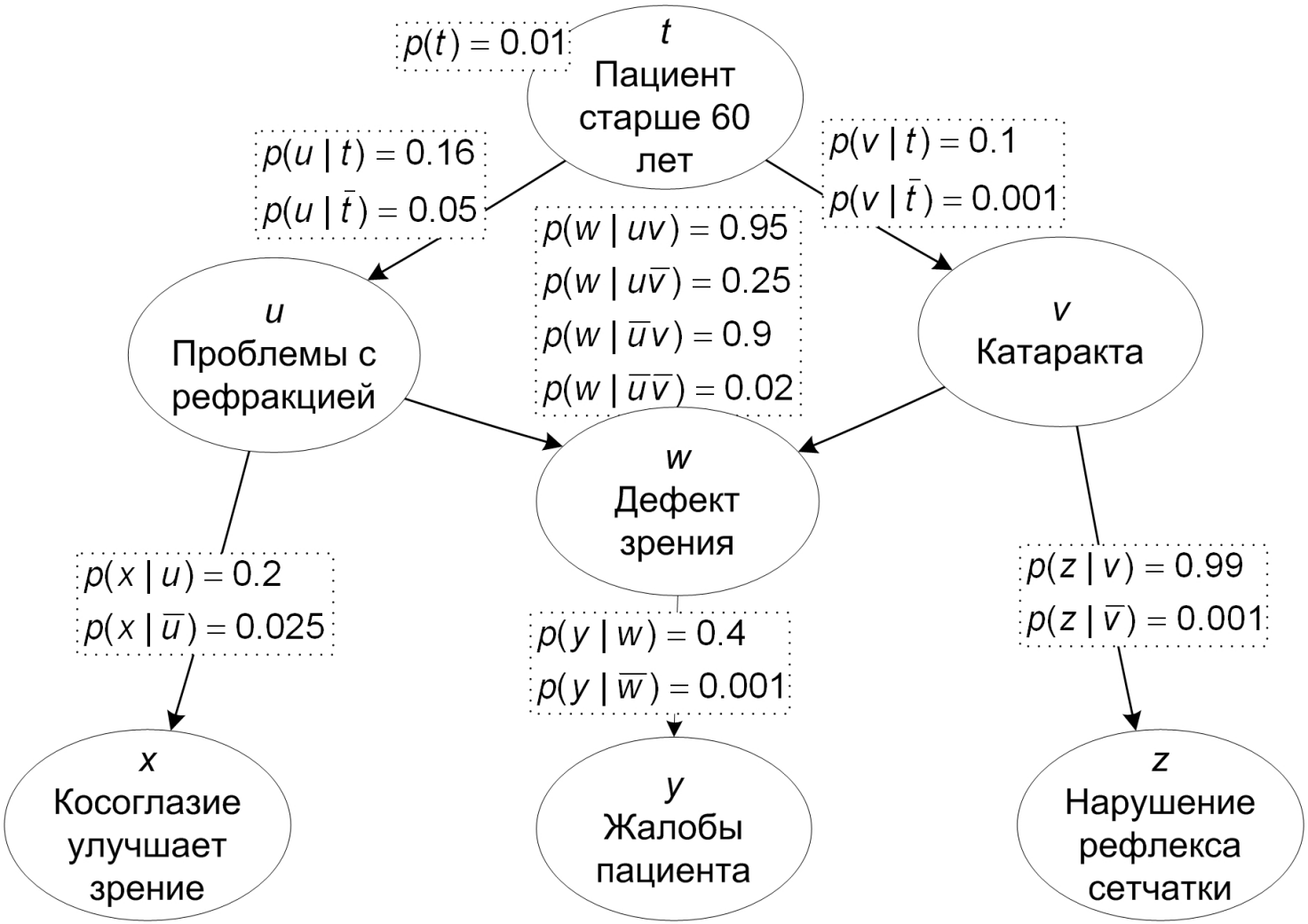
- Можно сформулировать, как структура графа соотносится с условной независимостью: в графе, где вершины из множества Z получили означивания (evidence), две ещё не означенные вершины x и y условно независимы при условии множества означенных вершин Z, если любой (ненаправленный) путь между x и y:
  - либо проходит через означенную вершину $z \in Z$ с последовательной или расходящейся связью;
  - либо проходит через вершину со сходящейся связью, в которой ни она, ни её потомки не получили означиваний.

Графические модели с направленными графами

- Можно сказать, что граф задаёт некоторое семейство распределений – не все распределения на вершинах графа будут соответствовать тем ограничениям по условной независимости, которые накладывает структура графа.
- Теорема (без доказательства): это семейство распределений в точности совпадает с семейством тех распределений, которые можно разложить в произведение

$$p(x_1, \ldots, x_k) = \prod_{i=1}^{k} p(x_i \mid \mathrm{pa}(x_i)).$$

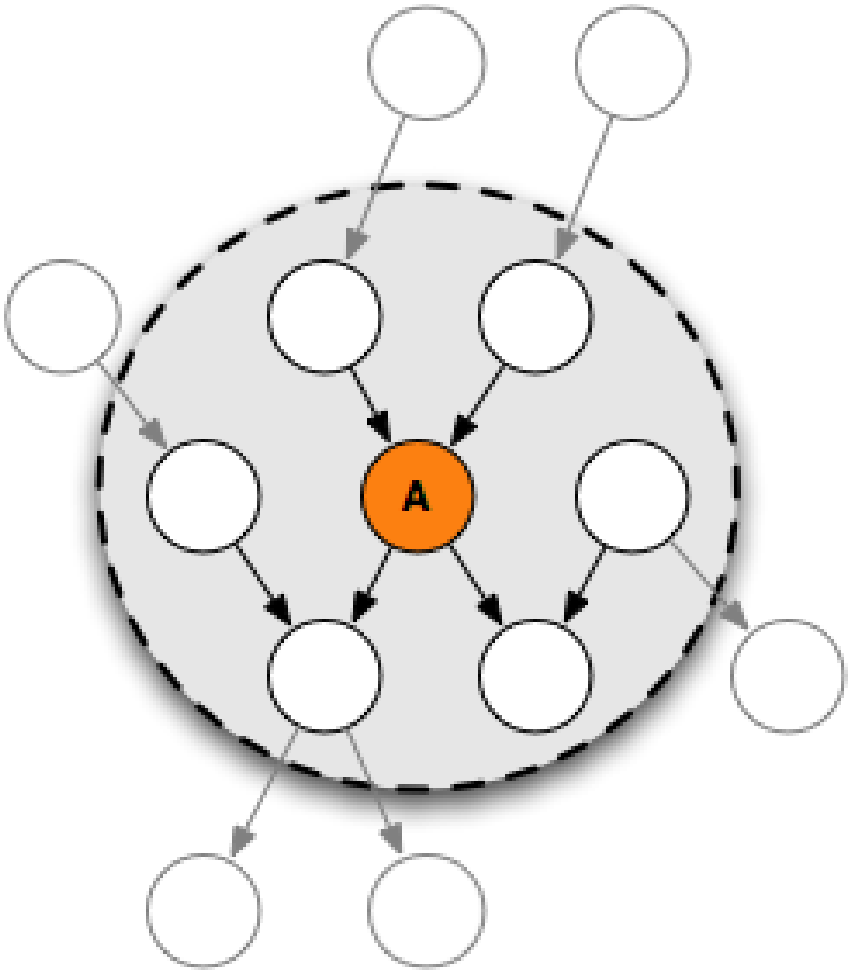
Пример байесовской сети



Markov blanket

- Интересный вопрос: какие вершины нужно означить, чтобы наверняка «отрезать» одну вершину (Markov blanket)?
- Иначе говоря, для какого минимального множества вершин X $p(x_i \mid x_{j \neq i}) = p(x_i \mid X)$?

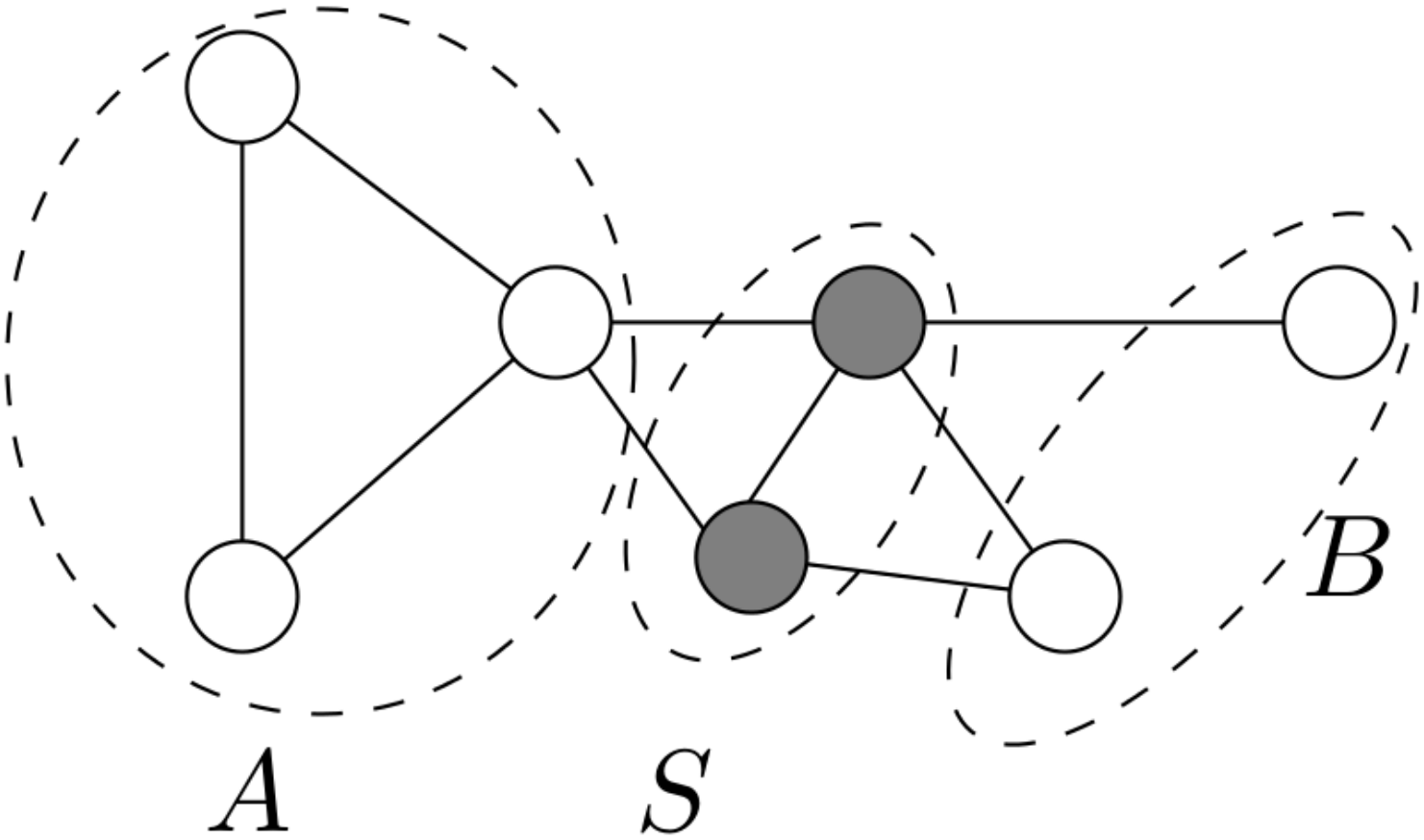
Markov blanket



Графические модели с ненаправленными графами

- Можно сделать и так, чтобы условие независимости было (более) локальным.
- Для этого нужно задавать модели ненаправленными графами. В них условие совсем естественное: множество вершин X условно независимо от множества вершин Y при условии множества вершин Z, если любой путь от X к Y проходит через Z.
- В частности, очевидно, $p(x_i, x_j \mid x_{k\neq i,j}) = p(x_i \mid x_{k\neq i,j})p(x_j \mid x_{k\neq i,j})$ тогда и только тогда, когда x_i и x_j не соединены ребром.
- Такие модели называются *марковскими сетями* (Markov random fields).

Условная независимость в ненаправленных моделях



Графические модели с ненаправленными графами

- Поэтому в ненаправленных моделях локальные распределения соответствуют кликам в графе, и факторизация получается в виде

$$p(x_1, \ldots, x_k) = \frac{1}{Z} \prod \psi_C(x_C),$$

где C – максимальные клики, ψ_C – неотрицательные функции (*потенциалы*), а Z – нормировочная константа (partition function).

- Поскольку $\psi_C \geq 0$, их обычно представляют как экспоненты:

$$\psi_C(x_C) = \exp\left(-E_C(x_C)\right),$$

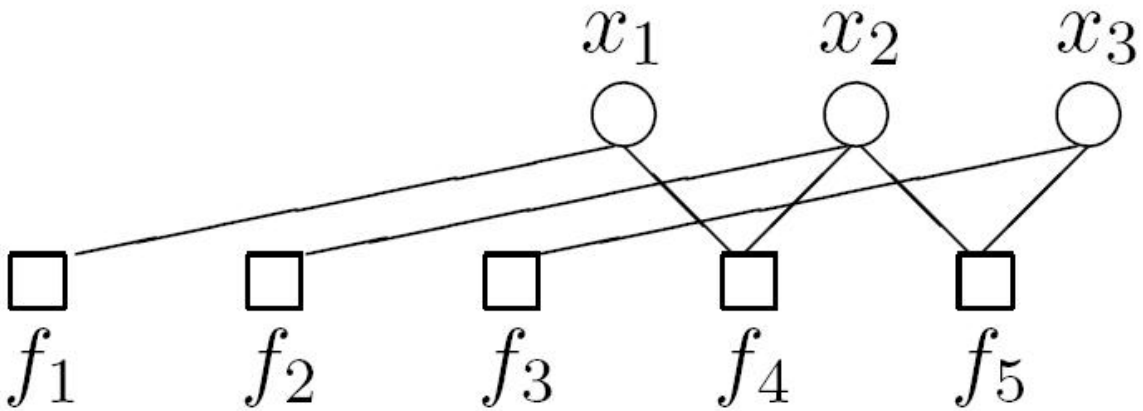
E_C – функции энергии, они суммируются в полную энергию системы (это всё похоже на статистическую физику, отсюда и терминология).

Графические модели с ненаправленными графами

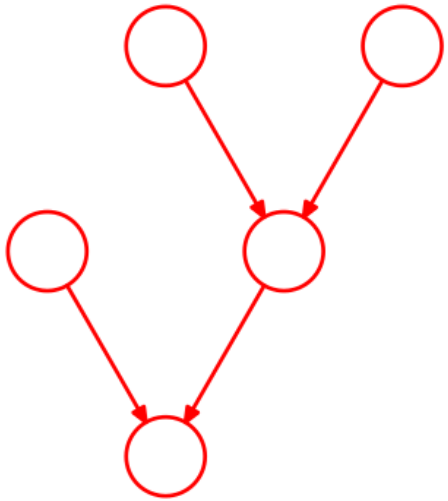
- Интересный факт: назовём *идеальной картой* (perfect map) распределения D графическую модель G, если все условные независимости, присутствующие в D, отображены в G, и наоборот (ничего лишнего). Тогда идеальные карты в виде направленных моделей существуют не у всех распределений, в виде ненаправленных тоже не у всех, и эти множества существенно различаются (бывают распределения, которые нельзя идеально выразить направленной моделью, но можно ненаправленной, и наоборот).

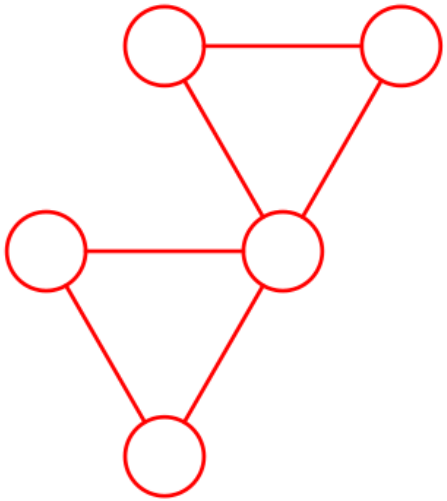
Фактор-графы

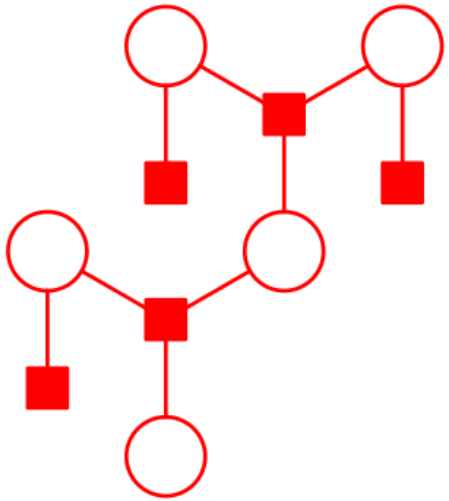
- Важная для вывода модификация – *фактор-граф* (можно построить и по направленной модели, и по ненаправленной).
- Фактор-граф – двудольный граф функций и переменных.
- Функция, соответствующая графу, – произведение всех входящих в него функций (т.е. то самое разложение и есть).
- Пример: $p(x_1, x_2, x_3) = f_1(x_1) f_2(x_2) f_3(x_3) f_4(x_1, x_2) f_5(x_2, x_3)$.



Три представления







Функция в общем виде

- Чтобы поставить задачу в общем виде, рассмотрим функцию

$$p^*(X) = \prod_{j=1}^{m} f_j(X_j),$$

где $X = \{x_i\}_{i=1}^{n}$, $X_j \subseteq X$.

- Т.е. мы рассматриваем функцию, которая раскладывается в произведение нескольких других функций.

Задачи

- Задача нормализации: найти $Z = \sum_X \prod_{j=1}^{m} f_j(X_j)$.
- Задача маргинализации: найти

$$p_i^*(x_i) = \sum_{k \neq i} p^*(X).$$

  Также может понадобиться, например, $p_{i_1 i_2}$, но реже.
- Поиск гипотезы максимального правдоподобия:

$$\mathbf{x}^* = \arg\max_X p(X).$$

Задачи

- Все эти задачи NP-трудные.
- То есть, если мир не рухнет, сложность их решения в худшем случае возрастает экспоненциально.
- Но можно решить некоторые частные случаи.

Пример

- Давайте начнём с графа в виде (ненаправленной) цепи:

$$p(x_1, \ldots, x_n) = \frac{1}{Z}\psi_{1,2}(x_1, x_2)\ldots\psi_{n-1,n}(x_{n-1}, x_n).$$

- Мы хотим найти

$$p(x_k) = \sum_{x_1} \ldots \sum_{x_{k-1}} \sum_{x_{k+1}} \ldots \sum_{x_n} p(x_1, \ldots, x_n).$$

Пример

- Очевидно, тут можно много чего упростить; например, справа налево:

$$\sum_{x_n} p(x_1, \ldots, x_n) =$$

$$= \frac{1}{Z}\psi_{1,2}(x_1, x_2) \ldots \psi_{n-2,n-1}(x_{n-2}, x_{n-1}) \sum_{x_n} \psi_{n-1,n}(x_{n-1}, x_n).$$

- Эту сумму можно вычислить отдельно и продолжать в том же духе справа налево, потом аналогично слева направо.

Пример

- В итоге процесс сойдётся на узле x_k, куда придут два «сообщения»: слева

$$\mu_\alpha(x_k) = \sum_{x_{k-1}} \psi_{k-1,k}(x_{k-1}, x_k) \left[\dots \sum_{x_2} \psi_{2,3}(x_2, x_3) \left[\sum_{x_1} \psi_{1,2}(x_1, x_2) \right] \dots \right],$$

справа

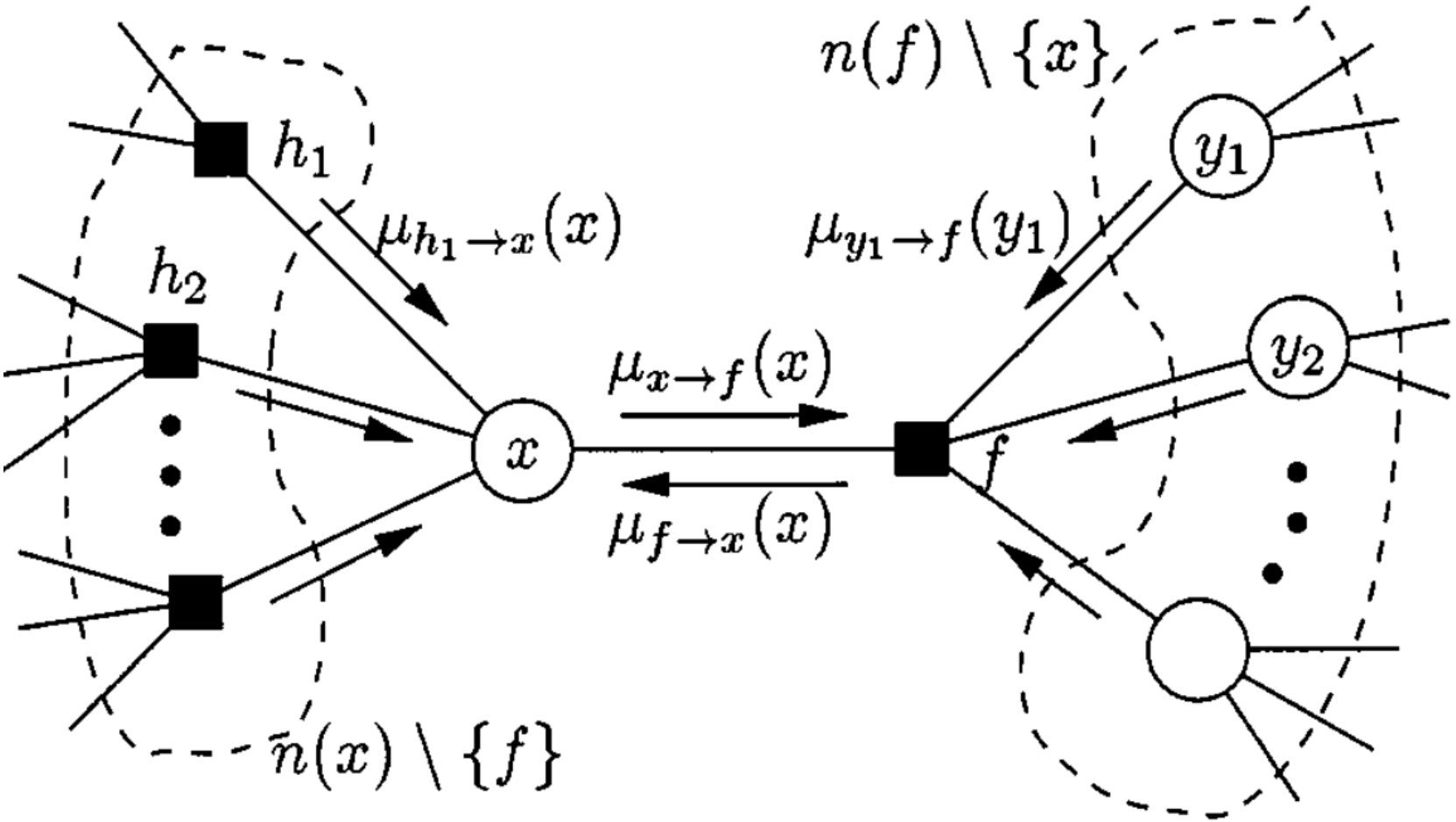
$$\mu_\beta(x_k) = \sum_{x_{k+1}} \psi_{k,k+1}(x_k, x_{k+1}) \left[\dots \left[\sum_{x_n} \psi_{n-1,n}(x_{n-1}, x_n) \right] \dots \right].$$

- Каждую частичную сумму можно рассматривать как «сообщение» от узла к своему соседу, причём это сообщение – функция от соседа.

Алгоритм передачи сообщений

- Чтобы обобщить, удобно рассмотреть опять фактор-граф.
- Предположим, что фактор-граф – дерево (если не дерево, так просто не сработает).
- Алгоритм передачи сообщений решает задачу маргинализации для функции вида $p(x_1, \ldots, x_n) = \prod_s f_s(X_s)$, заданной в виде фактор-графа.
- Передаём сообщения по направлению к нужному узлу от переменных к функциям и наоборот.

Передача сообщений



Алгоритм передачи сообщений

- Чтобы найти $p(x_k)$, запишем $p(x_1, \ldots, x_n) = \prod_{s \in \mathrm{ne}(x_k)} F_s(x_k, X_s)$, где X_s – переменные из поддерева с корнем в f_s. Тогда

$$p(x_k) = \sum_{x_{i \neq k}} p(x_1, \ldots, x_n) = \prod_{s \in \mathrm{ne}(x_k)} \left[\sum_{X_s} F_s(x_k, X_s) \right] = \\ = \prod_{s \in \mathrm{ne}(x_k)} \mu_{f_s \to x_k}(x_k),$$

где $\mu_{f_s \to x_k}(x_k)$ – сообщения от соседних функций к переменной x_k.

Алгоритм передачи сообщений

- Чтобы найти $\mu_{f_s \to x_k}(x_k)$, заметим, что $F_s(x_k, X_s)$ тоже можно разложить по соответствующему подграфу:

$$F_s(x_k, X_s) = f_s(x_k, Y_s) \prod_{y \in Y_s} G_y(y, X_{s,y}),$$

где Y_s – переменные, непосредственно связанные с f_s (кроме x_k), $X_{s,y}$ – соответствующие поддеревья.

- Итого получаем

$$\mu_{f_s \to x_k}(x_k) = \sum_{Y_s} f_s(x_k, Y_s) \prod_{y \in Y_s} \left(\sum_{X_{s,y}} G_y(y, X_{s,y}) \right) = \\ = \sum_{Y_s} f_s(x_k, Y_s) \prod_{y \in Y_s} \mu_{y \to f_s}(y).$$

Алгоритм передачи сообщений

- Можно аналогично подсчитать, что $\mu_{y \to f_s}(y) = \prod_{f \in \mathrm{ne}(y) \setminus f_s} \mu_{f \to y}(y)$.

Алгоритм передачи сообщений

- Итак, получился простой и понятный алгоритм:
  - как только узел получил сообщения от всех соседей, кроме одного, он сам начинает передавать сообщение в этого соседа;
  - сообщение по ребру между функцией и переменной является функцией от этой переменной;
  - узел-переменная x передаёт сообщение

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

  - узел-функция $f(x, Y)$ передаёт сообщение

$$\mu_{f \to x}(x) = \sum_{y \in Y} f(x, Y) \prod_{y \in Y} \mu_{y \to f}(y);$$

  - начальные сообщения в листьях $\mu_{x \to f}(x) = 1$, $\mu_{f \to x}(x) = f(x)$.

Алгоритм передачи сообщений

- Когда сообщения придут из всех соседей в какую-то переменную x_k, можно будет подсчитать

$$p(x_k) = \prod_{f \in \mathrm{ne}(x_k)} \mu_{f \to x_k}(x_k).$$

- Когда сообщения придут из всех соседей в какой-то фактор $f_s(X_s)$, можно будет подсчитать совместное распределение

$$p(X_s) = f_s(X_s) \prod_{y \in \mathrm{ne}(f_s)} \mu_{y \to f_s}(y).$$

- За два прохода (по каждому ребру туда и обратно) можно будет подсчитать маргиналы во всех узлах.

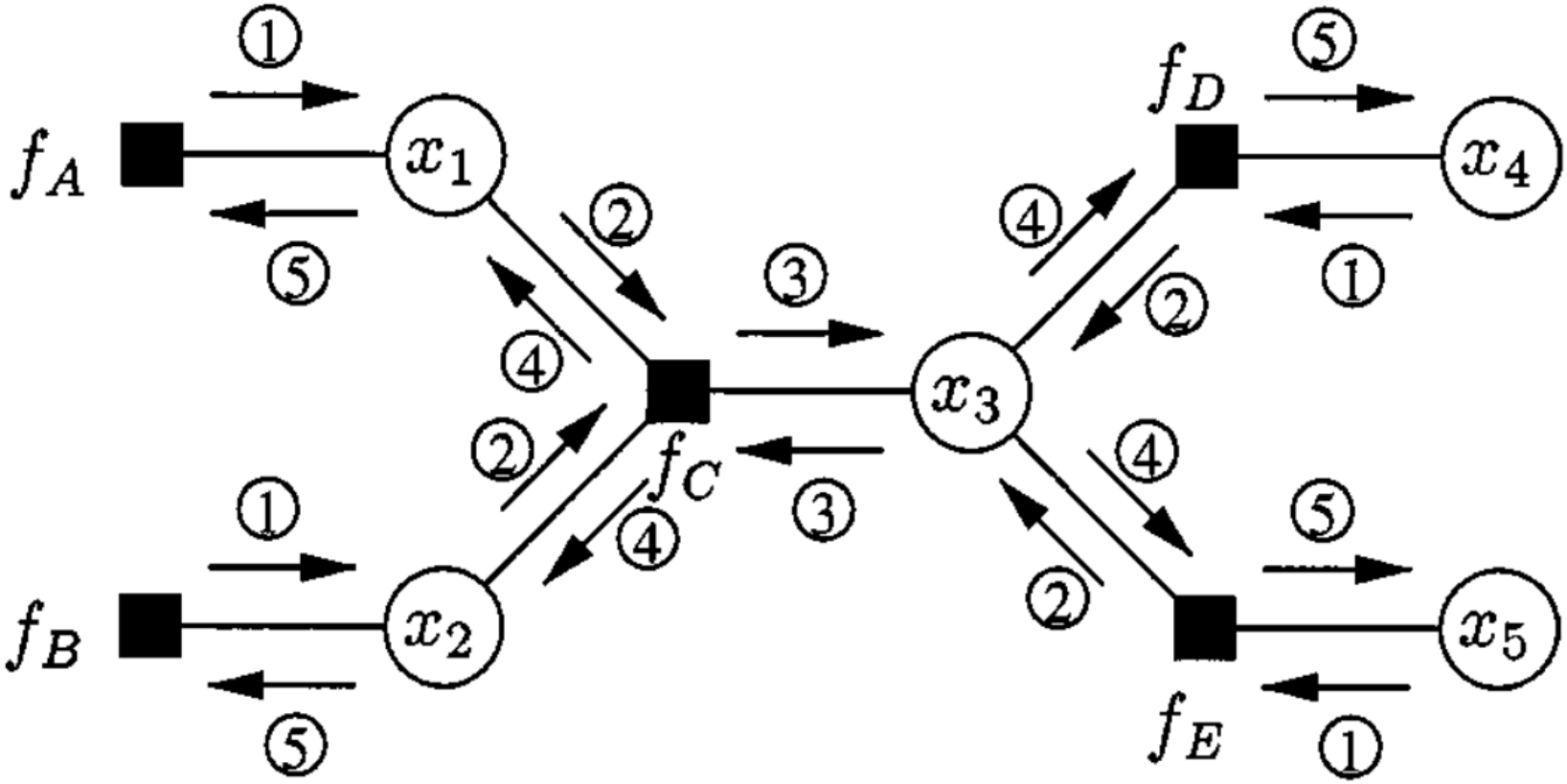
Алгоритм передачи сообщений

- Это называется алгоритм sum-product, потому что сообщение вычисляется как

$$\mu_{f\to x}(x) = \sum_{y\in Y} f(x, Y) \prod_{y\in Y} \mu_{y\to f}(y).$$

- Задача максимизации $\arg\max_x p(x_1, \ldots, x_n)$ решается так же, но алгоритмом max-sum: сумма заменяется на максимум, а произведение на сумму.

Передача сообщений

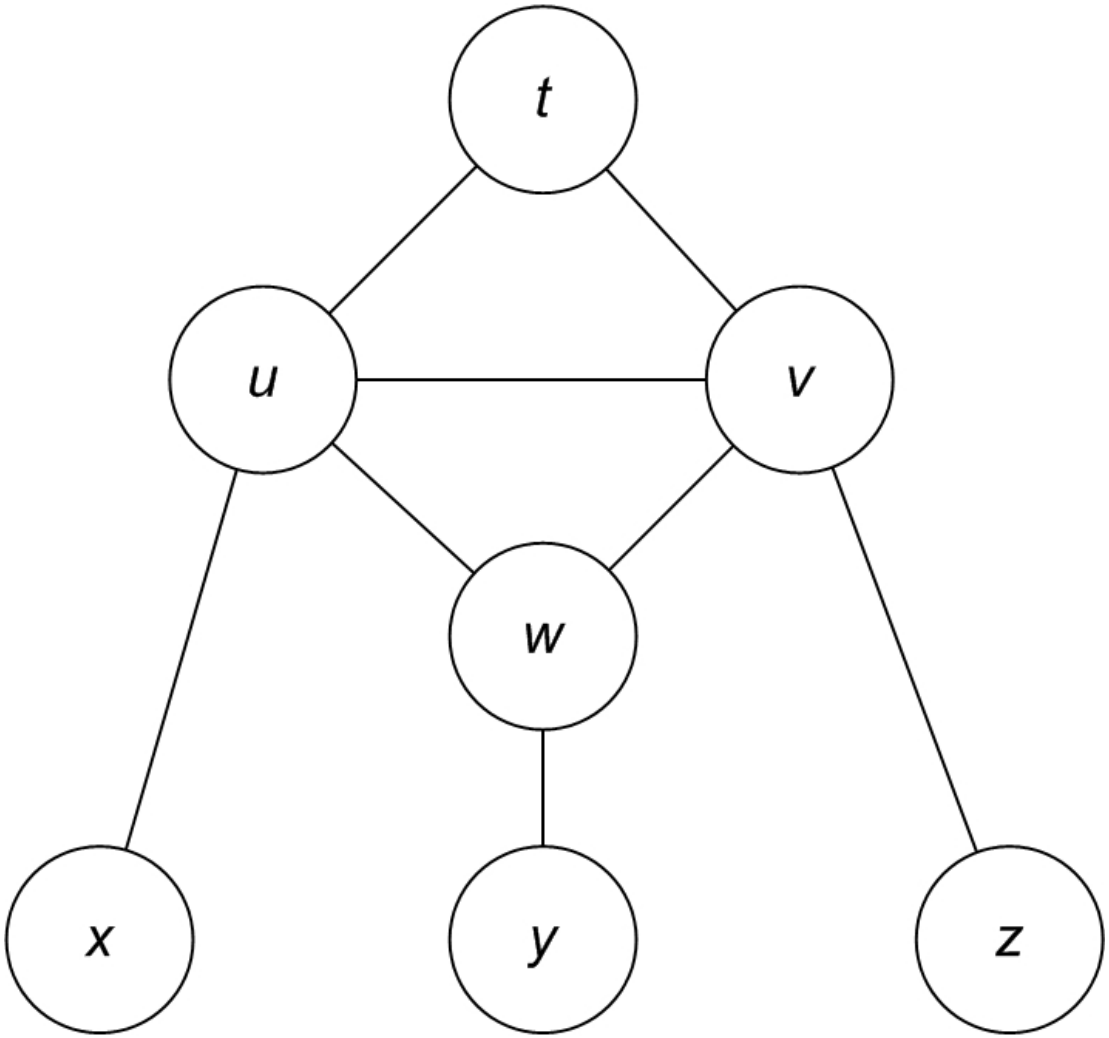


Так что же делать с байесовской сетью?

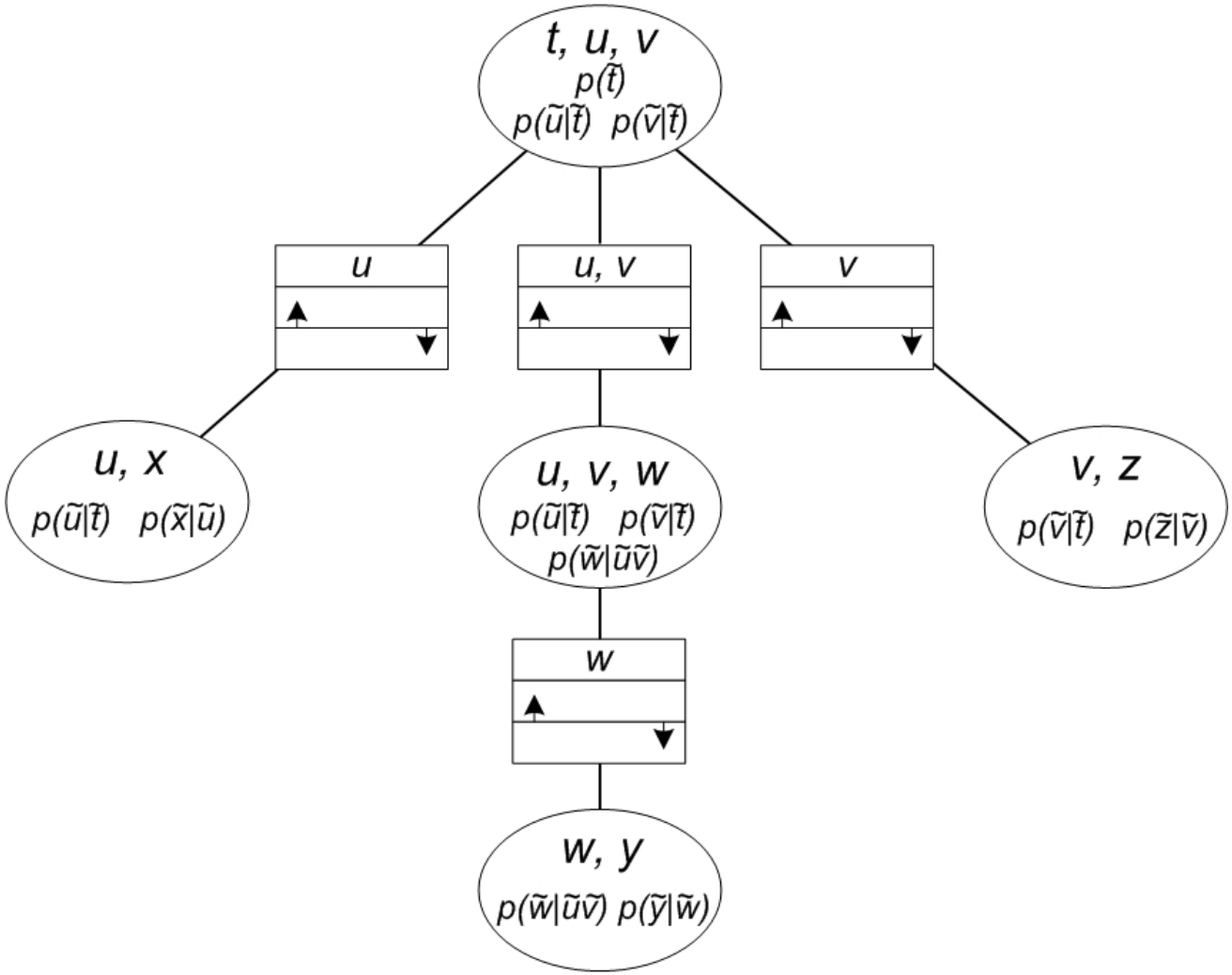
Для модели не в виде фактор-графа надо просто представить её в виде фактор-графа тем или иным способом.
Для байесовской сети это значит, что её сначала нужно представить в виде марковской сети, а потом добавить факторы в явном виде.

Так что же делать с байесовской сетью?

Так что же делать с байесовской сетью?



Так что же делать с байесовской сетью?



Outline

Приближённый вывод

- Когда граф – дерево, и в каждом узле всё считается явно и аналитически, можно посчитать быстро и точно.
- Но что делать, когда зубная щётка недоступна?
- Могут быть две проблемы:
  1. сложная структура графа, с циклами;
  2. сложные факторы – результат маргинализации в отдельном узле неудобный.

Суть

- Sum–product работает корректно, только если граф — дерево (ну, разве что скрестить пальцы и помолиться...).
- Что делать, когда граф содержит циклы?
- Нужно использовать деревья сочленений.

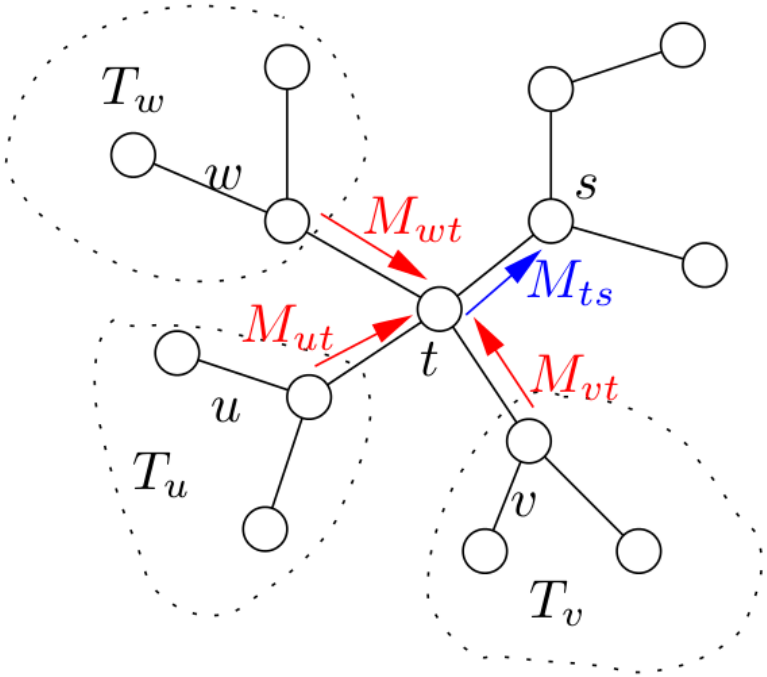
Деревья сочленений — неформально

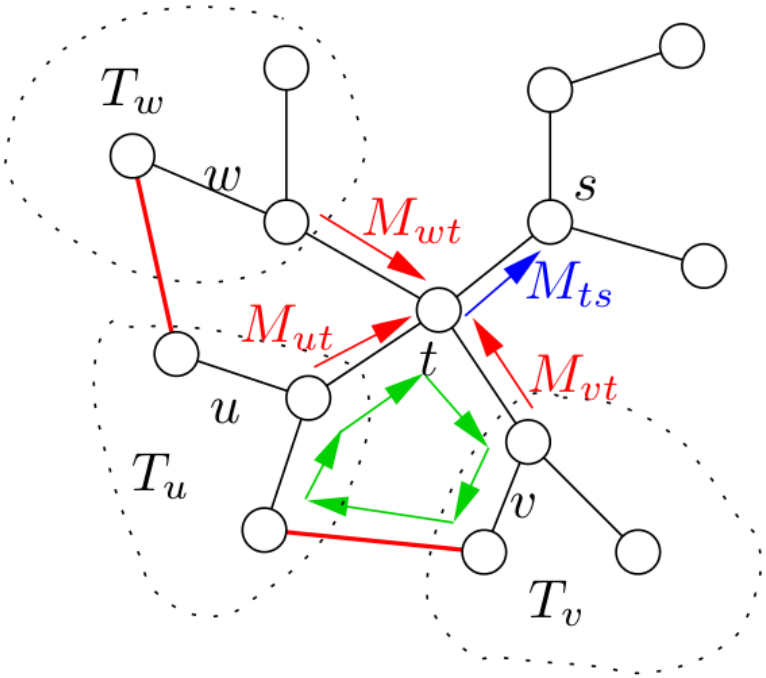
- Если цикл не сдаётся, его уничтожают, то есть заменяют весь цикл на одну вершину.
- Получается дерево, в котором уже можно работать обычным sum–product'ом; но при этом, конечно, замена нескольких вершин одной приводит к экспоненциальному раздуванию соответствующего фактора (множество значений соответствующей переменной должно содержать все комбинации значений исходных переменных).

Другие методы

- Если цикл всё-таки большой, то есть хороший общий метод, который применяют, когда нельзя применять sum–product.
- Метод заключается в том, чтобы применять sum–product. :)
- Он работает довольно часто даже тогда, когда в принципе работать не обязан (когда есть циклы).

Передача сообщений с циклами





Вариационные методы

- Если факторы простые, а структура сложная, можно приближать сложное распределение более простой формой, разрывая связи в графе: *вариационные приближения* (из матфизики).
- Т.е. надо будет выбрать распределение из некоторого более простого семейства, которое лучше всего приближает сложное распределение.
- «Похожесть» можно определить по расстоянию Кульбака–Лейблера

$$d(p, q) = \int p(x) \ln \frac{p(x)}{q(x)} \mathrm{d}x.$$

Вариационные методы

- Например: давайте искусственно разорвём связи, оставив только рёбра внутри подмножеств вершин X_i.
- Иначе говоря, будем предполагать, что любой фактор $q(Z)$ представляется в виде

$$q(Z) = \prod q_i(Z_i), \text{ где } Z_i = Z \cup X_i.$$

- Затем оптимизируем параметры, минимизируя расстояние между исходным распределением и таким факторизованным; это соответствует методу самосогласованного поля (mean field theory) в физике.
- Более подробно мы рассматривать вариационные методы сейчас не будем.

Сэмплирование по Гиббсу

- Ещё один важный метод – сэмплирование по Гиббсу.
- Постепенно сэмплируем одни переменные при условии других.
- Это вариант MCMC-приближений (Markov chain Monte Carlo), про них мы подробно говорить не будем.

Expectation propagation

- Если структура простая, но сложные факторы (результат не представляется в виде распределения нужной формы), можно его приближать простыми распределениями. Если в нашем факторизованном распределении

$$p(\theta \mid D) = \frac{1}{p(D)} \prod_i f_i(\theta)$$

слишком сложные факторы f_i, мы хотим их заменить на какие-нибудь простые (из экспоненциального семейства, например, гауссианы):

$$q(\theta \mid D) = \frac{1}{Z} \prod_i \hat{f}_i(\theta).$$

- И тоже минимизировать расстояние Кульбака–Лейблера между p и q.

Expectation propagation

- Для одного фактора всё это очень просто было бы – посчитать среднее и дисперсию (moment matching).
- Для многих факторов надо приближать все $\hat{f}_i$ одновременно. Можно доказать (мы не будем), что это можно делать последовательно, приближая фактор за фактором и итерируя, пока
- Таким образом, алгоритм Expectation Propagation на самом деле очень простой:
  1. запустить алгоритм передачи сообщений, но на каждом шаге вместо сообщения $\mu_{f_s \to x_k}(x_k)$ считать его приближение $\hat{\mu}_{f_s \to x_k}(x_k)$ из какого-нибудь простого семейства;
  2. повторять передачу сообщений, пока не сойдётся.

Thank you!

Спасибо за внимание!