Алгоритм передачи сообщений

Сергей Николенко

Академический Университет, весенний семестр 2011

Outline

Введение

- Мы уже много лекций подряд рассматривали задачи маргинализации, потому что они являются основными для байесовского вывода.
- Рассматривали много частных случаев.
- Сегодня мы наконец-то обобщим всё то, чем занимались, до практически самой общей из применимых на практике ситуаций.

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.

Нормализованная функция в общем виде

- Мы написали p^*, потому что обычно, чтобы получилась вероятность, нужно ещё нормализовать:

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

где $Z = \sum_X \prod_{j=1}^{m} f_j(X_j)$.

Пример

$$p^*(X) = f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1, x_2)f_5(x_2, x_3),$$

где

$$f_1(x_1) = \begin{cases} 0.05, & x_1 = 0 \\ 0.95, & x_1 = 1 \end{cases}$$

$$f_2(x_2) = \begin{cases} 0.05, & x_2 = 0 \\ 0.95, & x_2 = 1 \end{cases}$$

$$f_3(x_3) = \begin{cases} 0.95, & x_3 = 0 \\ 0.05, & x_3 = 1 \end{cases}$$

$$f_4(x_1x_2) = \begin{cases} 1, & x_1 = x_2 \\ 0, & x_1 \neq x_2 \end{cases}$$

$$f_5(x_2x_3) = \begin{cases} 1, & x_2 = x_3 \\ 0, & x_2 \neq x_3 \end{cases}$$

- Что это за функция? Вспомните прошлую лекцию.

Пример

- Это функция апостериорной вероятности повторяющего кода, если вероятность ошибки равна 0.05.
- f_4 и f_5 утверждают, что изначально все биты были одинаковые, а f_1, f_2 и f_3 указывают на вероятность ошибки.
- Задачи мы тоже формулировали: это были задачи маргинализации в общем и маргинализации побитовой.
- Давайте их обобщим.

Задачи

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

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

- Задача нормализованной маргинализации: найти

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

- Также может понадобиться, например, $p_{i_1 i_2}$, но реже.

Задачи

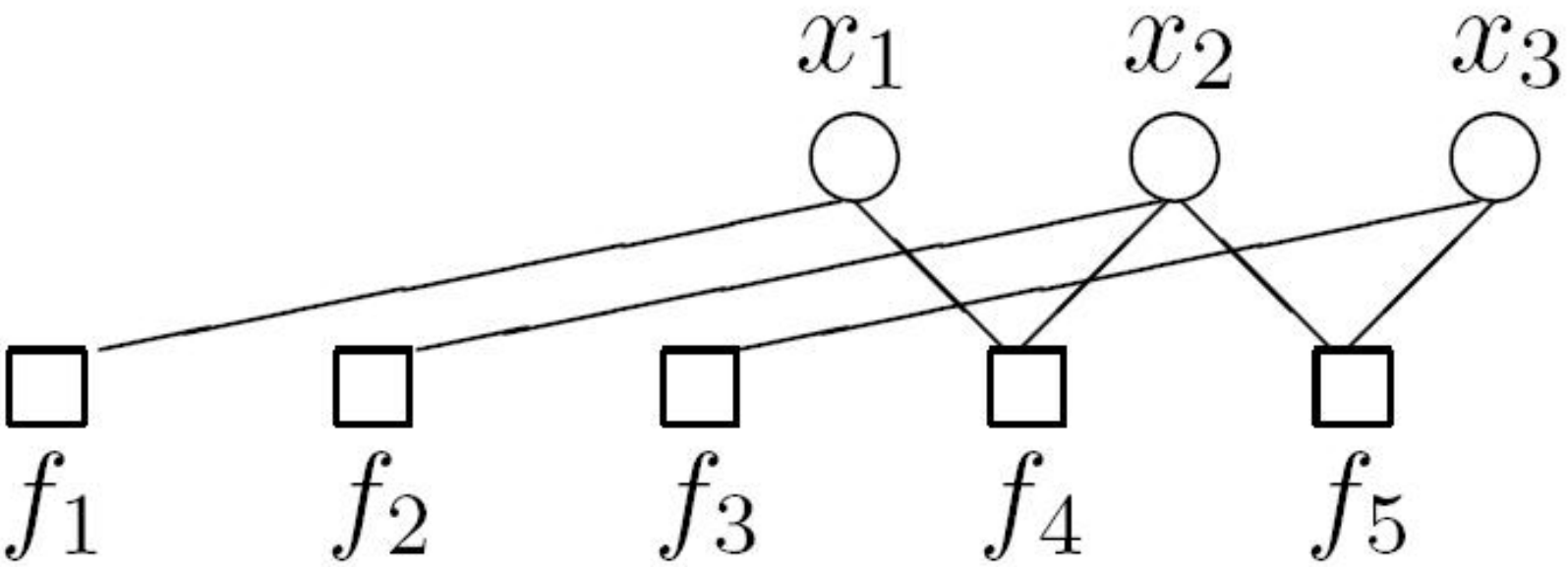
- Все эти задачи NP-трудные.
- То есть, если мир не рухнет, сложность их решения в худшем случае возрастает экспоненциально.
- Но можно решить некоторые частные случаи.

Структура графа

- Граф состоит из вершин, соответствующих переменным, и вершин, соответствующих функциям.
- Функции соединены с переменными, которых они описывают.

Структура графа

$$p^*(X) = f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1, x_2)f_5(x_2, x_3).$$



Идея алгоритма

- Как и прежде, идея в том, чтобы пересылать сообщения.
- Сообщения двух видов: от функций к переменным и от переменных к функциям.

Сообщения

- От переменной x_i к функции f_j:

$$q_{i\to j}(x_i) = \prod_{k\in \mathrm{nei}(i)\setminus j} r_{k\to i}(x_i).$$

- От функции f_j к переменной x_i:

$$r_{j\to i}(x_i) = \sum_{X_j\setminus x_i} \left(f_j(X_j) \prod_{k\in \mathrm{nei}(j)\setminus i} q_{k\to j}(x_k) \right).$$

Сообщения в листьях

- Отметим, что если у вершины только один сосед, то её сообщение можно вычислить, не зная входящих сообщений.
- Пустое произведение равно единице.
- Поэтому функция f_j, зависящая только от одной переменной x_i, передаёт своей переменной $r_{j\to i} = f_j(x_i)$, а переменная x_i, участвующая только в одной функции, передаёт ей 1.

Работа алгоритма I

- Предположим, что граф — дерево.
- Тогда можно начать с листьев и, постепенно вычисляя сообщения, обойти все вершины.
- При этом применяется стандартное правило передачи сообщений: сообщение можно передавать, только если его можно полностью построить.
- За количество шагов, равное диаметру графа, работа алгоритма закончится.

Работа алгоритма II

- Что делать, если граф — не дерево, и не ясно, с чего начинать?
- Вариант — начать с того, что все переменные передают сообщение 1, а потом уже его модифицируют, когда до них доходят сообщения от функций.
- Такой алгоритм не всегда работает правильно и делает много лишнего, но всё же полезен на практике.

Упражнение. Привести пример, когда алгоритм sum–product на графе с циклами работает некорректно.

Результат работы

- Когда рассылка сообщений закончится, можно будет вычислить маргиналы:

$$p_i^*(x_i) = \prod_{j \in \mathrm{nei}(i)} r_{j \to i}(x_i).$$

- Ясно, что $Z = \sum_i p_i^*(x_i)$, и $p(x_i) = \frac{1}{Z} p_i^*(x_i)$.

Упражнение. Доказать, что действительно получится $p_i^*(x_i)$.

Нормализация on-the-fly

- Возможно, мы интересуемся только нормализованными маргиналами (настоящими вероятностями).
- Тогда можно просто на каждом шаге нормализовать сообщения от переменных к функциям (от функций к переменным сообщения те же):

$$q_{i\to j}(x_i) = \alpha_{ij} \prod_{k\in \mathrm{nei}(i)\setminus j} r_{k\to i}(x_i),$$

где α_{ij} подобраны так, чтобы

$$\sum_i q_{i\to j}(x_i) = 1.$$

Упражнение. Пошагово применить вышеописанный алгоритм к функции из примера (функции правдоподобия повторяющего кода).

Разложение функции p^*

- Что такое sum–product математически?
- Изначально было разложение

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

- А мы перераскладываем функцию в произведение

$$p^*(X) = \prod_{j=1}^{m} \phi_j(X_j) \prod_{i=1}^{n} \psi_i(x_i),$$

где ϕ_j соответствуют узлам–функциям, а ψ_i — узлам–переменным.

- Изначально, до передачи сообщений, $\phi_j(X_j) = f_j(X_j)$ и $\psi_i(x_i) = 1$.

Разложение функции p^*

- Каждый раз, когда приходит сообщение $r_{j\to i}$ из функции в переменную, ϕ и ψ пересчитываются:

$$\psi_i(x_i) = \prod_{j\in \mathrm{nei}(i)} r_{j\to i}(x_i),$$

$$\phi_j(X_j) = \frac{f_j(X_j)}{\prod_{i\in \mathrm{nei}(j)} r_{j\to i}(x_i)}.$$

- Очевидно, общее произведение от этого не меняется.

Упражнение. Доказать, что в результате этого процесса ψ_i действительно станет маргиналом $p^*(x_i)$. Это докажет корректность алгоритма.

Постановка задачи

- Маргинализация — не единственная возможная постановка задачи.
- Вот, например, задача максимизации: найти значения переменных из X, которые максимизируют функцию p^*.
- Мы уже решали её и видели, какие нужны для этого инструменты.

От sum–product к max–product

- Задача максимизации p^* решится, если в узлах операцию суммирования $\sum$ заменить на max.
- Тогда нормализационная константа $Z = \sum_X p^*(x)$ превратится в $\max_X p^*(x)$, а каждый маргинал $p^*(x_i)$ станет показывать максимальное значение p^*, которого можно достичь в зависимости от значения x_i; отсюда уже тривиально можно будет найти максимизирующий набор.

От max–product к min–sum

- Умножать дороже, чем складывать. Как бы нам начать складывать вместо того чтобы умножать?

От max–product к min–sum

- Умножать дороже, чем складывать. Как бы нам начать складывать вместо того чтобы умножать?
- Естественно, надо логарифмировать (а потом добавить знак «минус»); тогда из max–product получится min–sum. Мы это уже много раз делали.
- Такой алгоритм называется алгоритмом Витерби (Viterbi).

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 её экспоненциальному раздуванию (множество значений соответствующей переменной должно содержать все комбинации значений исходных переменных).
- О том, как это сделать формально и правильно, мы говорили на лекции по байесовским сетям.

Другие методы

- Есть приближённые методы, мы, возможно, позже рассмотрим некоторые из них.
- Например, метод Лапласа: взять и приблизить сложное непрерывное распределение гауссианом (для этого достаточно подсчитать ожидание и дисперсию).
- Но 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).

Outline

Байесовские сети

- В этой лекции мы на время отвлечёмся от обучения и рассмотрим основные понятия и алгоритмы теории байесовских сетей доверия.
- Кроме просто полезного аппарата, это даст хорошее применение дискретно-вероятностным вещам вроде [не]зависимости, а главное — более общее понимание задач байесовского вывода. Пригодится.

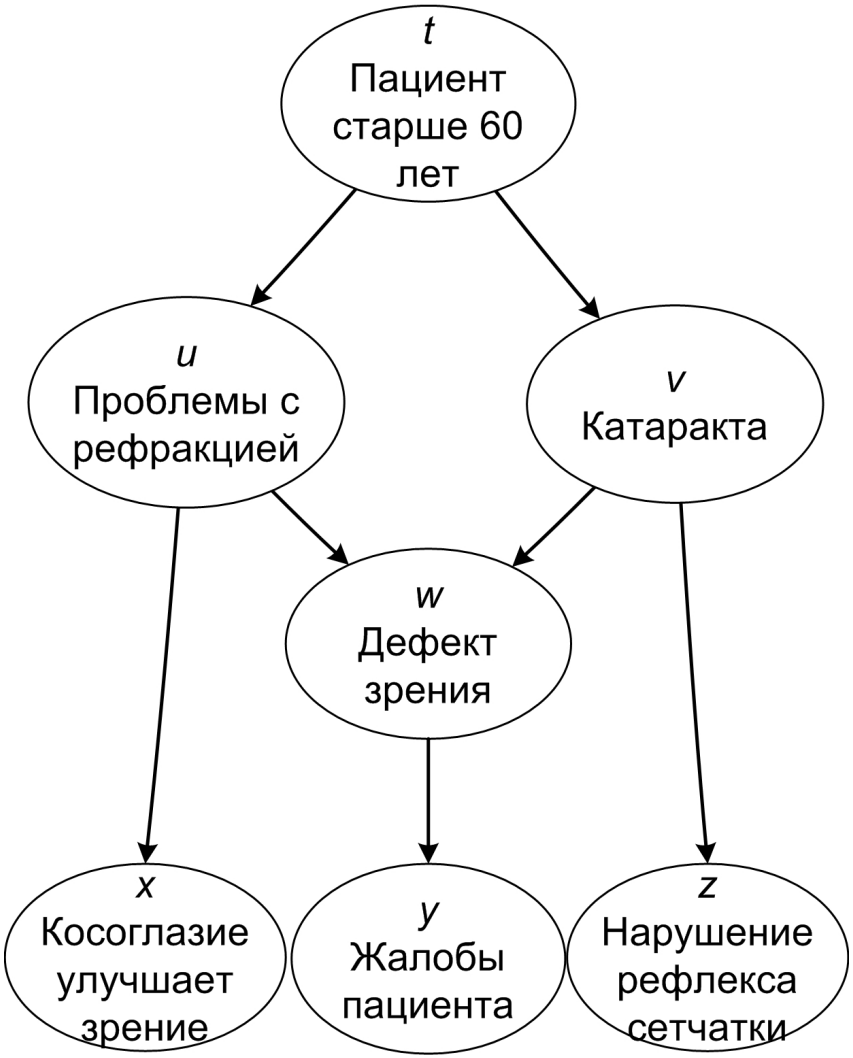
Наивный байесовский классификатор

- Наивный байесовский классификатор часто хорошо работает.
- Но он основывается на очень серьёзном предположении, а именно на условной независимости атрибутов при условии данного целевого значения.
- Зачастую такое предположение делает аппарат неприменимым.
- Что делать?

От байесовского классификатора к байесовским сетям

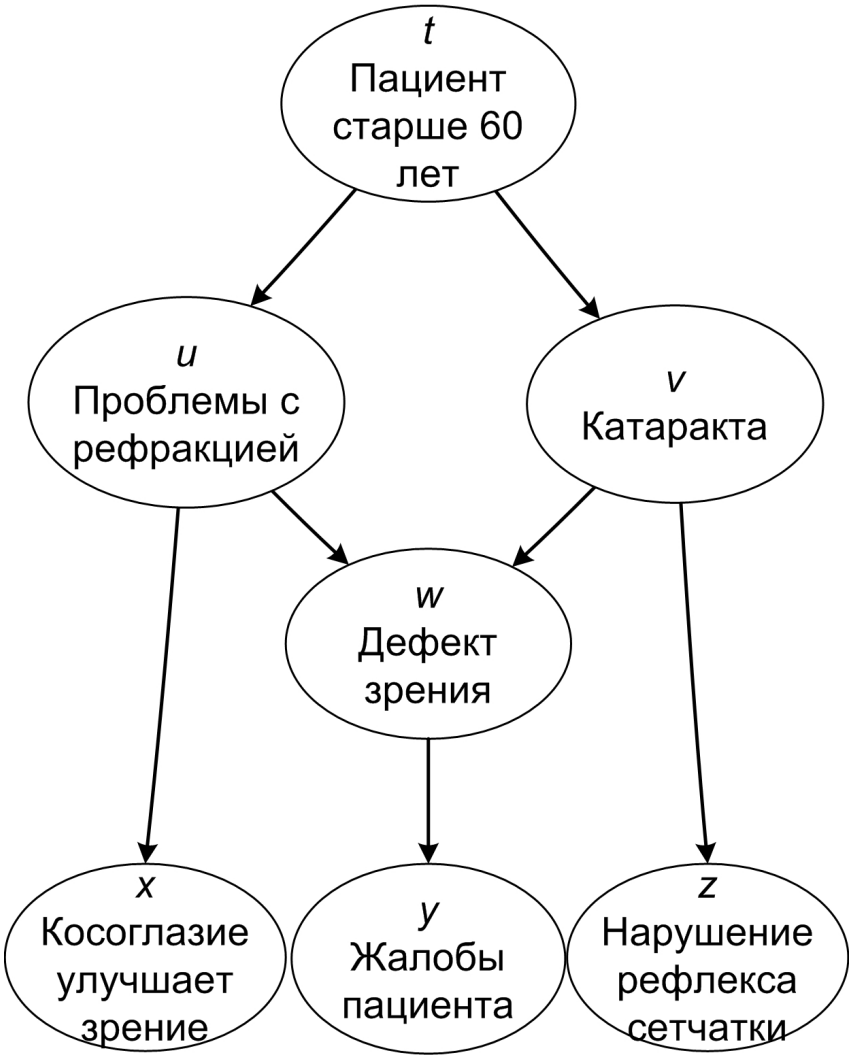
- Нужно научиться представлять множество (не)зависимостей между имеющимися переменными.
- Достаточно естественная идея: направленный граф, в котором стрелки показывают причинно-следственную связь.

Пример



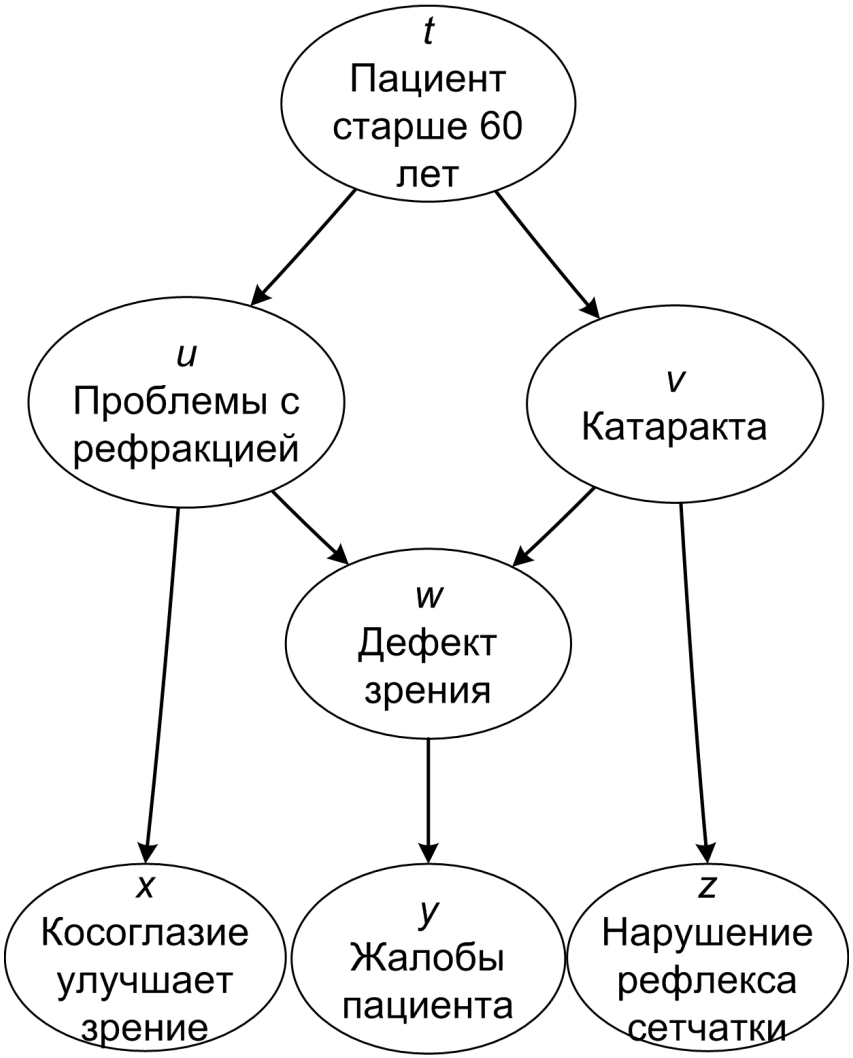
Пример того, как может быть задан граф байесовской сети. Истоки — причины, стоки — следствия. Корни полидерева — первопричины, листья — симптомы (как правило, именно их наблюдают).

Пример



Изначально заданы условные вероятности потомков при условии предков. Это уже позволит определить совместную априорную вероятность любой комбинации событий в сети.

Пример



Суть рассуждений в байесовской сети — *пропагация свидетельств*. На вход поступают данные типа «Зрение пациента улучшилось из-за косоглазия» и «Пациент старше 60 лет», а задача — оценить, как изменилась вероятность других узлов (например, того, что «У пациента нарушение рефлекса сетчатки»).

Применение

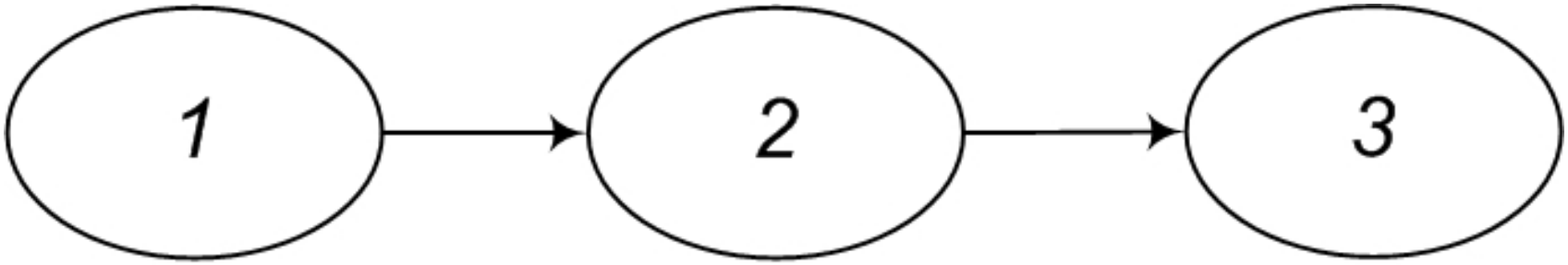
- Байесовские сети применяются в основном для решения *диагностических* задач. Например, их часто используют в медицине и вообще для оценки рисков. Т.е. пропагация обычно идёт снизу вверх, от следствий к причинам. Но можно и наоборот.

Зависимость в БСД

- Какие узлы (события) в байесовской сети зависимы?
- Понятно, что те, которые соединены ребром.
- Но не только...

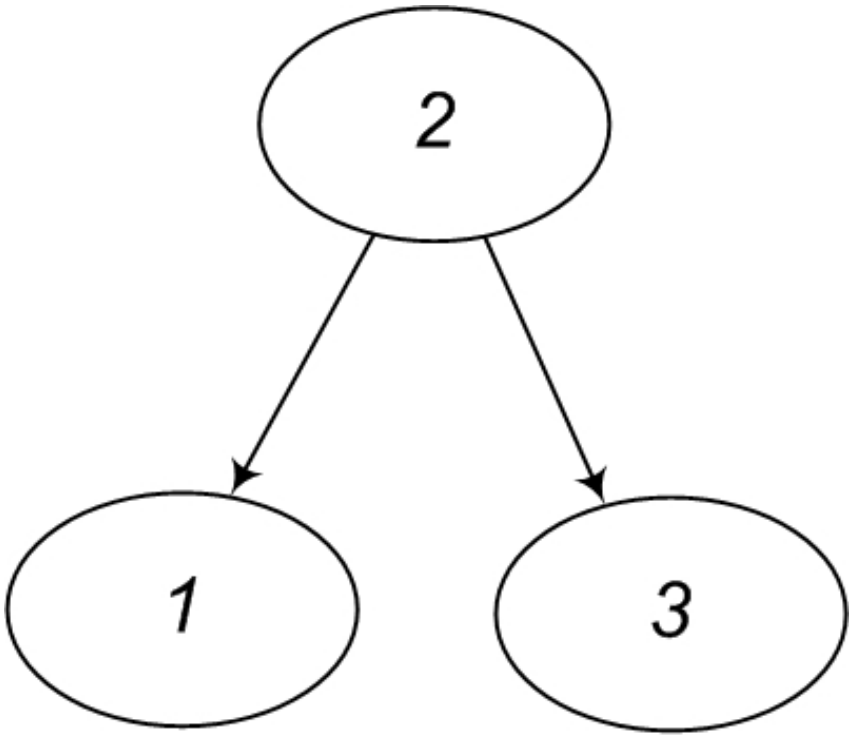
Последовательная связь

Прямая связь между узлами сети. В этом случае 1 влияет на 2, а 2, в свою очередь, влияет на 2, и узлы 1 и 3 получаются связанными. Однако, если в 2 поступило свидетельство, связь между 1 и 3 нарушается.



Последовательная связь

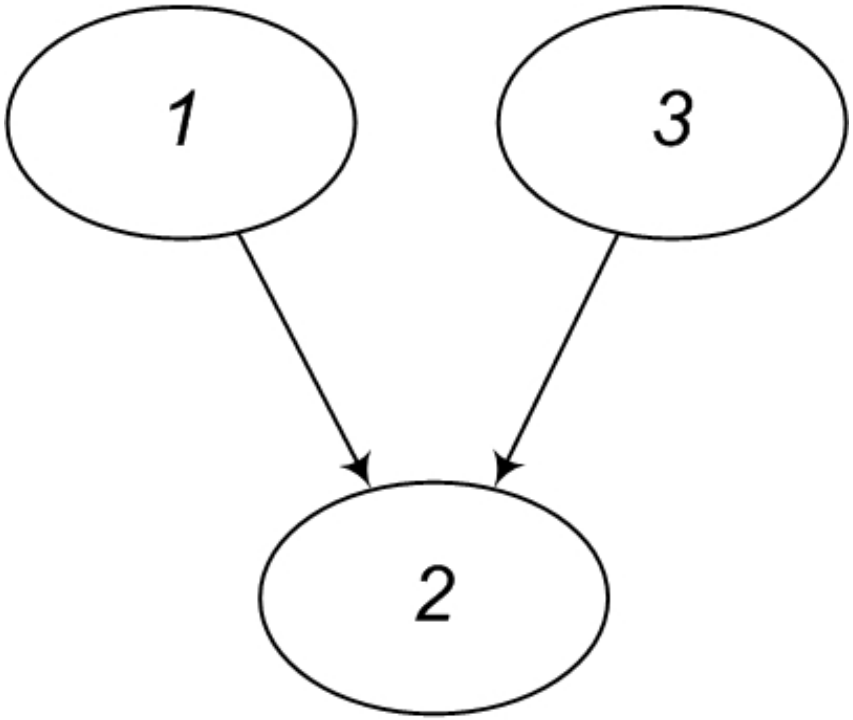
Расходящаяся связь



Расходящаяся связь

Информация об одном из потомков может повлиять на вероятность другого потомка одного и того же узла. Это связано с тем, что не только информация о произошедшей причине повышает вероятность следствия, но и случившееся следствие повышает вероятность причины. Если общий предок уже получил означивание, то связь нарушается.

Сходящаяся связь



Сходящаяся связь

Связи между узлами 1 и 3 нет: если произошло 1, то это повлияет на вероятность события 2, но вероятность 3 измениться не должна. Однако ситуация меняется, если свидетельство 2 уже получено: если мы знаем, что одна из причин произошла, это должно понизить вероятность другой причины, ведь следствие уже объяснено.

d-разделимость

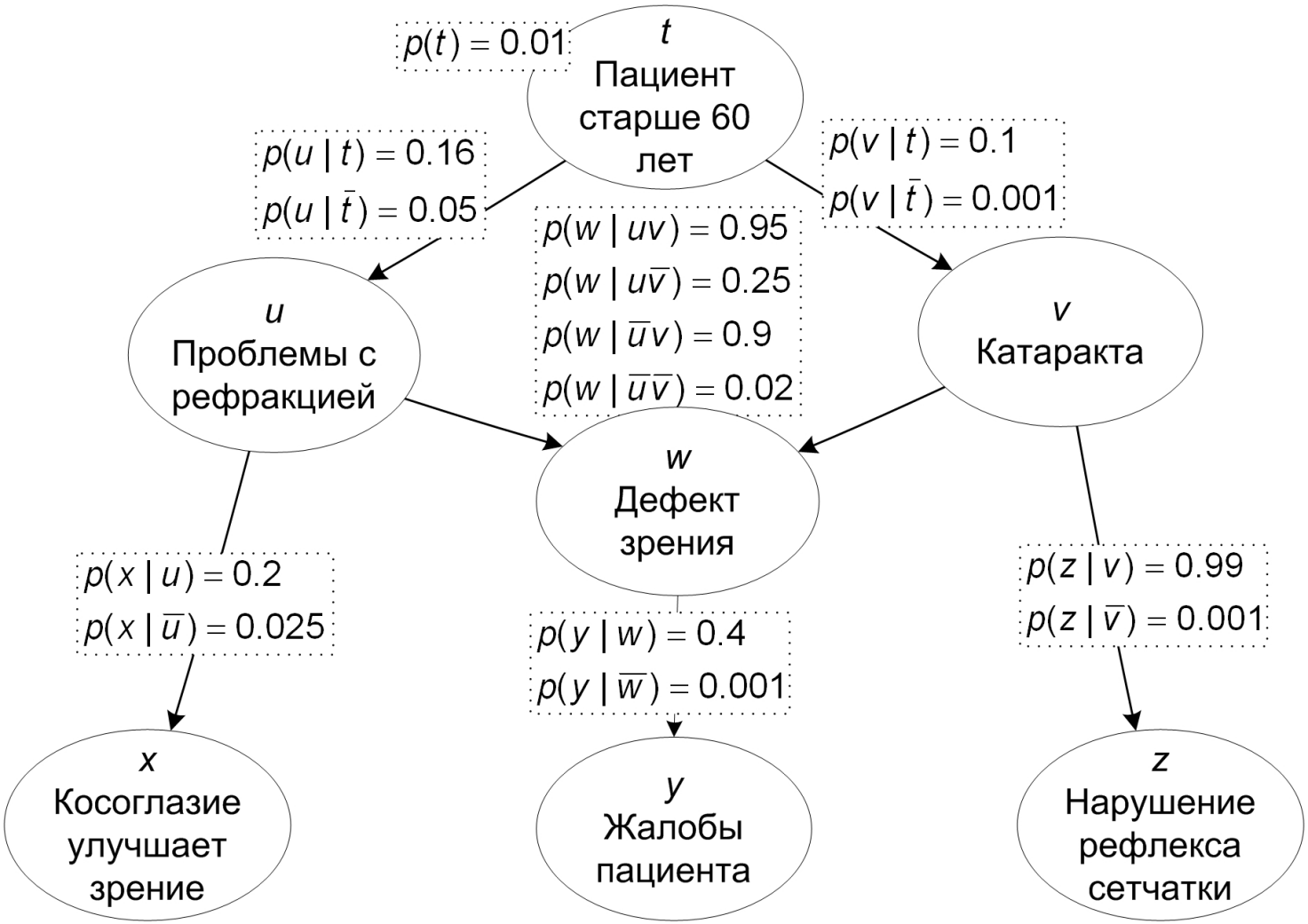
Определение

Два узла направленного графа x и y называются d-разделёнными, если для всякого пути из x в y (здесь не учитывается направление рёбер) существует такой промежуточный узел z (не совпадающий ни с x, ни с y), что либо связь в пути в этом узле последовательная или расходящаяся, и узел z получил означивание, либо связь сходящаяся, и ни узел z, ни какой-либо из его потомков означивания не получил.
В противном случае узлы называются d-связанными.

Вероятности

- До сих пор были только графы и рассуждения «на пальцах».
- Теперь пора перейти к заданию вероятностей и прочим численным примерам.
- В вершинах заданы условные вероятности при условии всего множества предков. Если предков нет, вероятности не условные (а маргинальные).

Пример


$p(t) = 0.01$
t
Пациент старше 60 лет
$p(u \mid t) = 0.16$
$p(u \mid \bar{t}) = 0.05$
$p(v \mid t) = 0.1$
$p(v \mid \bar{t}) = 0.001$
$p(w \mid uv) = 0.95$
$p(w \mid u\bar{v}) = 0.25$
$p(w \mid \bar{u}v) = 0.9$
$p(w \mid \bar{u}\bar{v}) = 0.02$
u
Проблемы с рефракцией
v
Катаракта
w
Дефект зрения
$p(x \mid u) = 0.2$
$p(x \mid \bar{u}) = 0.025$
$p(y \mid w) = 0.4$
$p(y \mid \bar{w}) = 0.001$
$p(z \mid v) = 0.99$
$p(z \mid \bar{v}) = 0.001$
x
Косоглазие улучшает зрение
y
Жалобы пациента
z
Нарушение рефлекса сетчатки

Численный пример

Вычислим совместные вероятности цепочки $\tilde{u}\tilde{v}\tilde{w}$:

$$
\begin{aligned}
p(uvw) &= p(w|uv)\sum_{\tilde{t}} p(u|\tilde{t})p(v|\tilde{t})p(\tilde{t}) = 0.000199025,\\
p(uv\bar{w}) &= p(\bar{w}|uv)\sum_{\tilde{t}} p(u|\tilde{t})p(v|\tilde{t})p(\tilde{t}) = 0.000010475,\\
p(u\bar{v}w) &= p(w|u\bar{v})\sum_{\tilde{t}} p(u|\tilde{t})p(\bar{v}|\tilde{t})p(\tilde{t}) = 0.012722625,\\
p(u\bar{v}\bar{w}) &= p(\bar{w}|u\bar{v})\sum_{\tilde{t}} p(u|\tilde{t})p(\bar{v}|\tilde{t})p(\tilde{t}) = 0.038167875,\\
p(\bar{u}vw) &= p(w|\bar{u}v)\sum_{\tilde{t}} p(\bar{u}|\tilde{t})p(v|\tilde{t})p(\tilde{t}) = 0.00160245,\\
p(\bar{u}v\bar{w}) &= p(\bar{w}|\bar{u}v)\sum_{\tilde{t}} p(\bar{u}|\tilde{t})p(v|\tilde{t})p(\tilde{t}) = 0.00017805,\\
p(\bar{u}\bar{v}w) &= p(w|\bar{u}\bar{v})\sum_{\tilde{t}} p(\bar{u}|\tilde{t})p(\bar{v}|\tilde{t})p(\tilde{t}) = 0.01894239,\\
p(\bar{u}\bar{v}\bar{w}) &= p(\bar{w}|\bar{u}\bar{v})\sum_{\tilde{t}} p(\bar{u}|\tilde{t})p(\bar{v}|\tilde{t})p(\tilde{t}) = 0.92817711.
\end{aligned}
$$

Важные замечания

- Если у узла n предков, нужно задавать 2^n условных вероятностей.
- Если предков у узла x нет, нужно задавать маргинальные вероятности $p(x)$.
- В графе запрещены направленные циклы.
- Вся эта информация в сумме даст возможность вычислять любую вероятность в сети, т.е. единственным образом задаст распределение.

Теорема о декомпозиции

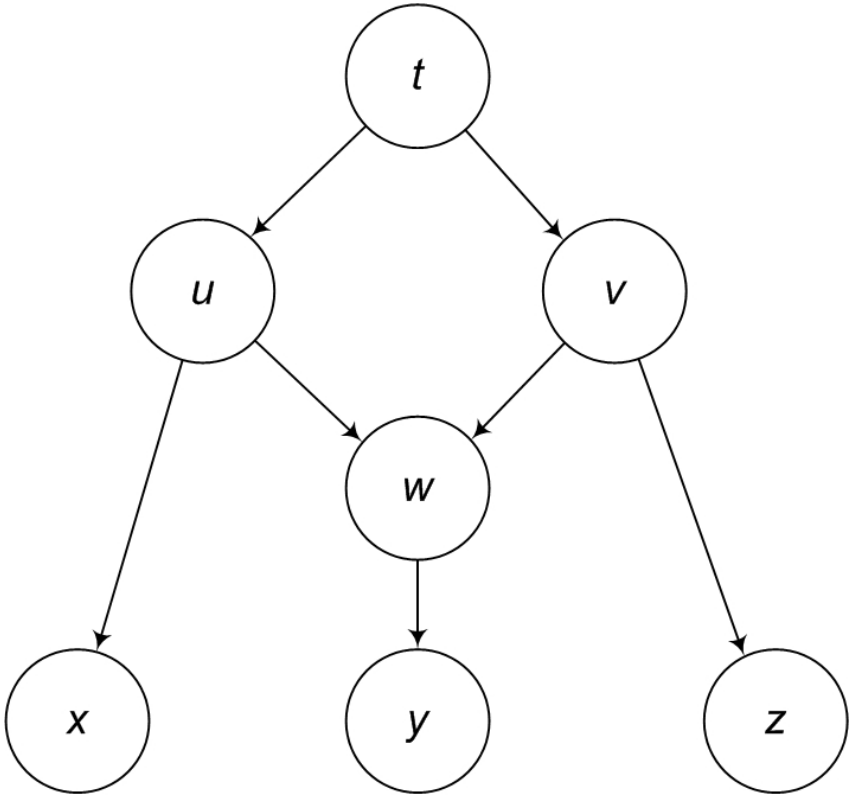
Теорема

Для БСД, построенной на множестве переменных $\mathcal{S} = \{x_1, x_2, \ldots, x_n\}$, *общее распределение вероятностей* $p(\tilde{x}_1 \tilde{x}_2 \ldots \tilde{x}_n)$, *индуцирующее заданные условные и маргинальные вероятности и согласованное с условной независимостью, вытекающей из d-разделимости узлов, существует, единственно и представляет собой произведение всех тензоров, заданных в байесовской сети доверия:*

$$p(\tilde{\mathcal{S}}) = \prod_{x \in \mathcal{S}} p(\tilde{x} | \widetilde{\mathrm{pa}(x)}),$$

где $\mathrm{pa}(x)$ *— множество родителей узла* x *в базовом графе сети.*

Пример

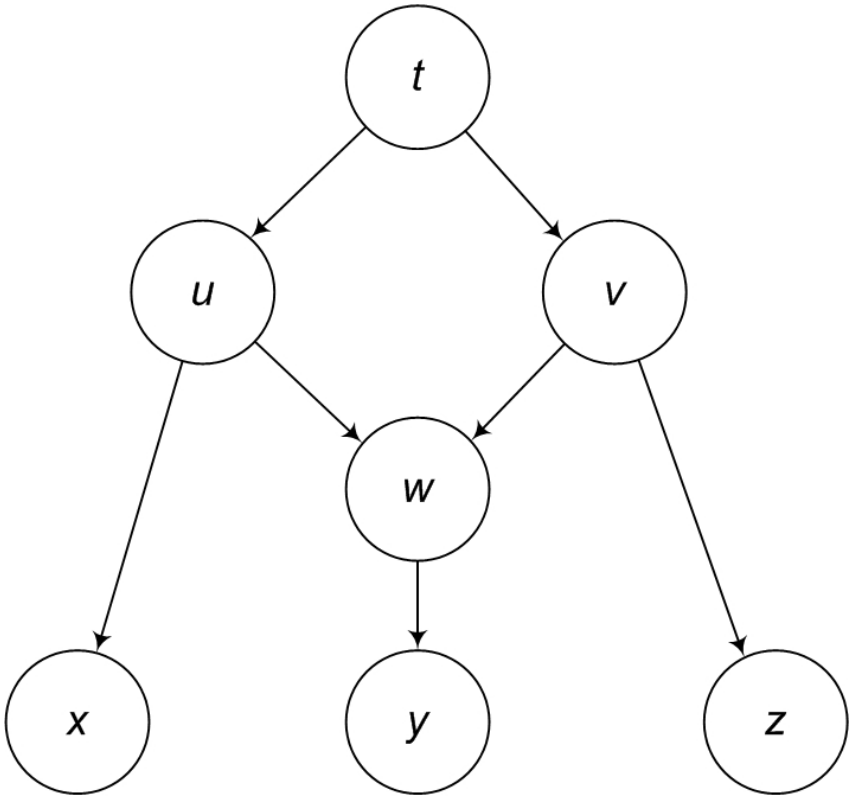


Попробуем в нашем примере вычислить $p(z)$. Для этого надо просуммировать по всем остальным вероятностям:

$$p(z) = \sum_{\tilde{t}\tilde{u}\tilde{v}\tilde{w}\tilde{x}\tilde{y}} p(\tilde{t}\,\tilde{u}\,\tilde{v}\,\tilde{w}\,\tilde{x}\,\tilde{y}\,z).$$

Тут безумное количество вычислений.

Пример

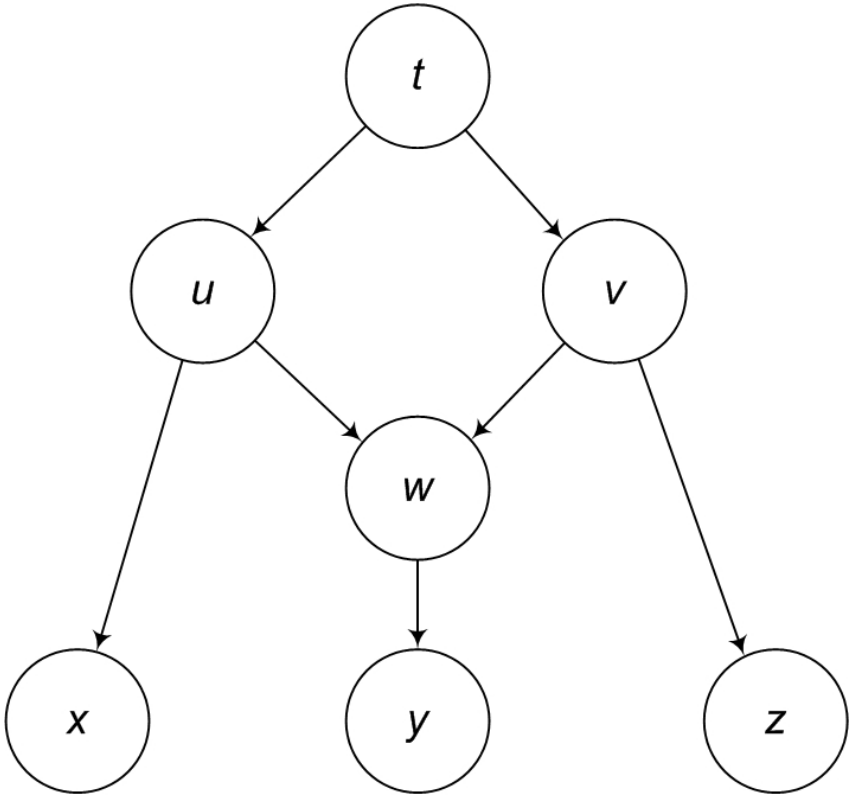


После применения правила декомпозиции получается:

$$p(\tilde{z}) = \sum_{\tilde{t}} p(\tilde{t}) \sum_{\tilde{v}} p(\tilde{v}|\tilde{t}) p(\tilde{z}|\tilde{v})$$
$$\sum_{\tilde{u}} p(\tilde{u}|\tilde{t}) \sum_{\tilde{x}} p(\tilde{x}|\tilde{u})$$
$$\sum_{\tilde{w}} p(\tilde{w}|\tilde{u}\tilde{v}) \sum_{\tilde{y}} p(\tilde{y}|\tilde{w}),$$

и вычислений уже гораздо меньше.

Пример



В этом и заключается смысл байесовских сетей доверия (и других алгоритмов байесовского вывода) — разложить большое распределение на произведение маленьких.

Свидетельства

- Свидетельства — утверждения вида «событие в узле x произошло». Например: «У пациента дефект зрения», т.е. w.
- Главная наша задача: научиться пересчитывать вероятности при поступлении свидетельств.

Пересчёт вероятностей в одном узле

Пусть поступило свидетельство «Дефект зрения». Давайте рассчитаем апостериорную вероятность $p(u|w)$.
Сначала нужно приравнять нулю несовместимые со свидетельством случаи в таблице совместных вероятностей:

$$\begin{aligned} p(uvw \wedge w) &= 0.000199025, & p(uv\bar{w} \wedge w) &= 0, \\ p(u\bar{v}w \wedge w) &= 0.012722625, & p(u\bar{v}\bar{w} \wedge w) &= 0, \\ p(\bar{u}vw \wedge w) &= 0.00160245, & p(\bar{u}v\bar{w} \wedge w) &= 0, \\ p(\bar{u}\bar{v}w \wedge w) &= 0.01894239, & p(\bar{u}\bar{v}\bar{w} \wedge w) &= 0. \end{aligned}$$

Пересчёт вероятностей в одном узле

И нормировать то, что получилось:

$$p(u|w) = \frac{\sum_{\tilde{v}\tilde{w}} p(u\tilde{v}\tilde{w}\wedge w)}{p(w)} = 0.386107\ldots,$$
$$p(\bar{u}|w) = \frac{\sum_{\tilde{v}\tilde{w}} p(\bar{u}\tilde{v}\tilde{w}\wedge w)}{p(w)} = 0.61389288\ldots.$$

Пропагация в сетях без циклов

- Мы сейчас запрещаем не только направленные, но и ненаправленные циклы. Т.е. граф байесовской сети — полидерево.
- Чтобы не запрещать ненаправленные циклы, нужно рассматривать алгоритмы построения морального графа, триангуляции и построения дерева смежности. Этим мы займёмся на следующей лекции.
- Мы уже поняли, как пропагировать свидетельство через один узел. Осталось формализовать это и собрать алгоритм целиком.

Идея алгоритма

- Мы делим поступившие свидетельства на две части — те, что выше данного узла x (E_x^+) и те, что ниже (E_x^-). Наша задача — найти

$$p(x|E) = p(x|E_x^-, E_x^+).$$

- Затем вычисляем их действие на x по отдельности и складываем это всё вместе.

Обозначения

- Evid(x) возвращает 1, если x инстанциирован нашим свидетельством.
- Normalize(Pr) нормализует распределение Pr.
- $E_{x \setminus Y}$ означает свидетельства, связанные с x, кроме тех, путь к которым пролегает через элементы множества Y.

Вывод

$$p(x|E_x^-, E_x^+) = \frac{p(E_x^-|x, E_x^+)p(x|E_x^+)}{p(E_x^-|E_x^+)}.$$

Но в полидереве x d-отделяет E_x^+ от E_x^-, поэтому $p(E_x^-|x, E_x^+) = p(E_x^-|x)$. Кроме того, вероятности должны в сумме давать 1, поэтому можно забыть про $\frac{1}{p(E_x^-|E_x^+)}$, а потом просто нормализовать результат. Итого нужно вычислить

$$p(E_x^-|x)p(x|E_x^+).$$

Начнём с $p(x|E_x^+)$.

Вывод

Рассмотрим все конфигурации родителей x $\tilde{U} = \widetilde{\mathrm{pa}(x)}$. Тогда

$$p(x|E_x^+) = \sum_{\tilde{U}} p(x|\tilde{U}, E_x^+) p(\tilde{U}|E_x^+).$$

- U d-отделяет x от E_x^+, поэтому первый сомножитель — это просто $p(x|\tilde{U})$, и это дано нам в таблицах условных вероятностей.
- E_x^+ d-отделяет каждый $u \in U$ от других, и

$$p(\tilde{U}|E_x^+) = \prod_{u \in U} p(\tilde{u}|E_X^+).$$

Вывод

Рассмотрим все конфигурации родителей x $\tilde{U} = \widetilde{\mathrm{pa}(x)}$. Тогда

$$p(x|E_x^+) = \sum_{\tilde{U}} p(x|\tilde{U}, E_x^+) p(\tilde{U}|E_x^+).$$

- Осталось заметить, что $E_x^+ = \bigcup_{u \in U} E_{u \setminus x}$, все они не пересекаются, и $E_{u \setminus x}$ d-отделяет u от всех остальных свидетельств в E_x^+. Итого получается:

$$p(x|E_x^+) = \mathrm{Norm}\left(\sum_{\tilde{U}} p(x|\tilde{U}) \prod_{u \in U} p(\tilde{u}|E_{u \setminus x}) \right).$$

Вывод

Рассмотрим все конфигурации родителей x $\tilde{U} = \widetilde{\text{pa}(x)}$. Тогда

$$p(x|E_x^+) = \sum_{\tilde{U}} p(x|\tilde{U}, E_x^+) p(\tilde{U}|E_x^+).$$

$$p(x|E_x^+) = \text{Norm}\left(\sum_{\tilde{U}} p(x|\tilde{U}) \prod_{u \in U} p(\tilde{u}|E_{u \setminus x})\right).$$

- Это уже похоже на рекурсивный алгоритм: $p(\tilde{u}|E_{u \setminus x})$ — рекурсивный вызов исходной процедуры.

Вывод $p(E_x^-|x)$

- Здесь всё то же самое, но посложнее, потому что нужно учитывать возможных других предков потомков узла x. Распишем по конфигурациям детей $\widetilde{\mathrm{ch}}(x)$:

$$p(E_x^-|x) = \sum_{\widetilde{\mathrm{ch}}x} \prod_{u \in \mathrm{ch}(x)} p(E_u^-|u)p(u|\mathrm{pa}(u) \setminus x).$$

- Переставим местами сумму и произведение и распишем $p(u|pa(u) \setminus x)$, исходя из уже полученных результатов:

$$p(u|Z) = \sum_{\tilde{Z}} p(\tilde{u}|x, \tilde{Z}) \prod_{z \in Z} p(\tilde{z}|E_{z \setminus u}), \; Z = pa(u) \setminus x.$$

Вывод $p(E_x^-|x)$

- В этой сумме узел x фиксирован. В итоге получаем:

$$p(E_x^-|x) = \prod_{u \in \mathrm{ch}(x)} \sum_{\tilde{u}} \Big\{ p(E_u^-|\tilde{u}) \times \\ \times \sum_{\widetilde{\mathrm{pa}(u)}} p(\tilde{u}|x, \widetilde{\mathrm{pa}(u)}) \prod_{z \in \mathrm{pa}(z)} p(\tilde{z}|E_{z \setminus u}) \Big\}.$$

  В этой формуле $p(E_u^-|x)$ также вычисляется рекурсивно.
- Осталось только не забыть нормировать.

Алгоритм

$\mathrm{Belief}(x)$:

- Вернуть $\mathrm{BeliefExcept}(x, \emptyset)$.

$\mathrm{BeliefExcept}(x, V)$:

- Если $\mathrm{Evid}(x)$, то вернуть уже имеющееся распределение x.
- Вычислить $p(E^-_{x \setminus V}|x) = \mathrm{EvidenceExcept}(X, V)$.
- $U = \mathrm{pa}(x)$.
- Если $U = \emptyset$ вернуть $\mathrm{Norm}\left(p(E^-_{x \setminus V}|x)p(x)\right)$.
- Иначе для каждого $u \in U$ вычислить и сохранить $p(\tilde{u}|E_{u \setminus x}) = \mathrm{BeliefExcept}(\tilde{u}, x)$.
- Вернуть $\mathrm{Norm}\left(p(E^-_{x \setminus V}|x) \sum_{\tilde{U}} p(x|\tilde{U}) \prod_{u \in U} p(\tilde{u}|E_{u \setminus x})\right)$.

Алгоритм

$\mathrm{EvidenceExcept}(x, V)$:

- $U = \mathrm{ch}(x) \setminus V$.
- Если $(U == \emptyset)$ вернуть равномерное распределение.
- Иначе для каждого $u \in U$:
  - Вычислить $p(E_u^-|\tilde{u}) = \mathrm{EvidenceExcept}(\tilde{u}, \emptyset)$.
  - $Z = \mathrm{pa}(u) \setminus \{x\}$.
  - Для каждого $z \in Z$ вычислить

$$p(\tilde{Z}|E_{z\setminus u}) = \mathrm{BeliefExcept}(\tilde{w}, u).$$

- Вернуть

$$p(E_x^-|x) = \mathrm{Norm}\left(\prod_{u\in\mathrm{ch}(x)}\sum_{\tilde{u}} p(E_u^-|\tilde{u})\sum_{\tilde{Z}} p(\tilde{u}|x,\tilde{Z})\prod_{z\in Z} p(\tilde{z}|E_{z\setminus u})\right).$$

Thank you!

Спасибо за внимание!