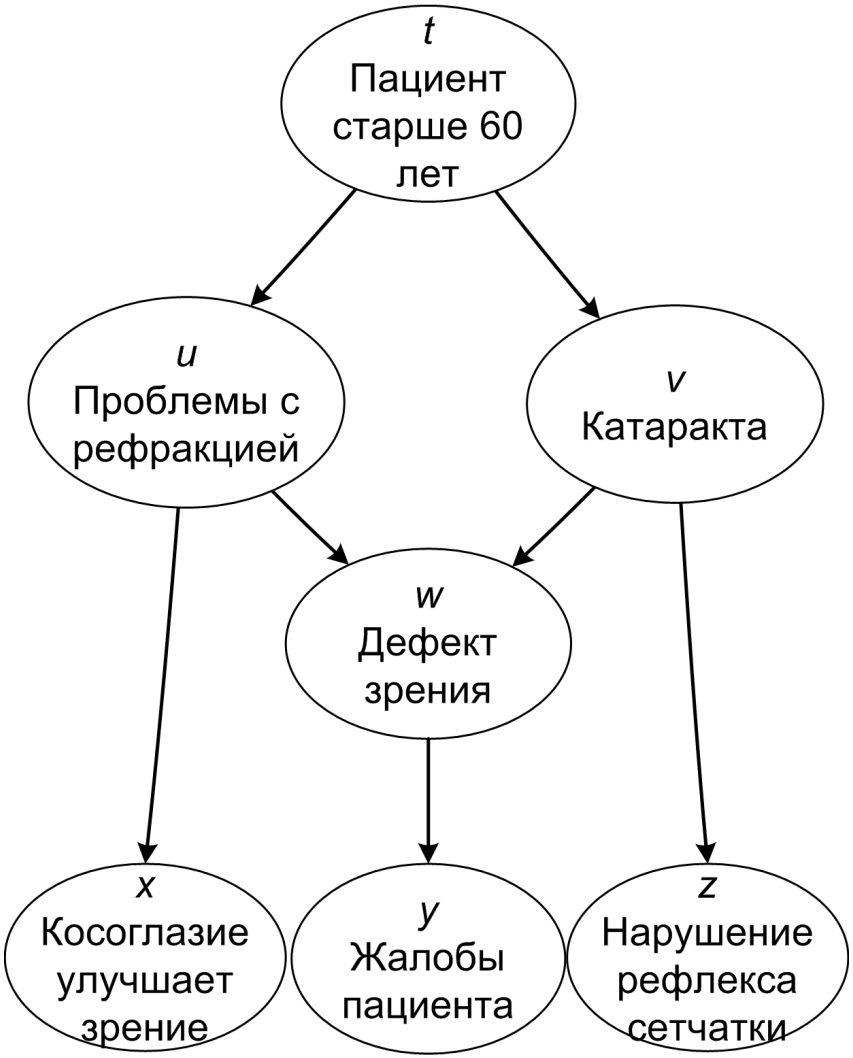
Деревья сочленений. Вывод в общем случае

Сергей Николенко

Академический Университет, весенний семестр 2011

Outline

Ненаправленные циклы



- На прошлых лекциях мы рассмотрели алгоритм вывода на полидереве.
- Но, конечно, на самом деле сплошь и рядом бывают графы с ненаправленными циклами.
- Что делать?

BBN и MRF

- Для этого мы преобразуем BBN к виду *марковского случайного поля* (Markov random field, MRF).
- MRF и BBN – фактически одно и то же, только BBN направленный, а MRF нет.

BBN и MRF

- В BBN заданы условные вероятности родителей при условии предков $p\left(s_i \mid \mathrm{pa}(s_i)\right)$, и общее совместное распределение равно

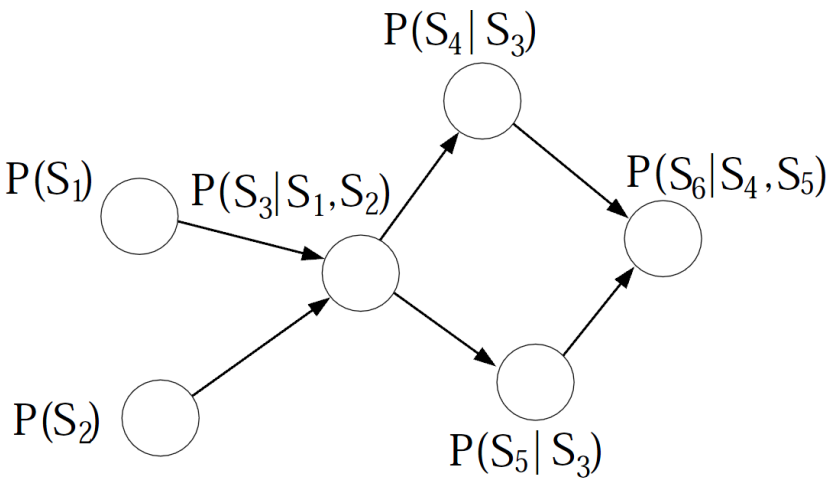
$$p(S) = \prod_{i=1}^{N} p\left(s_i \mid \mathrm{pa}(s_i)\right).$$

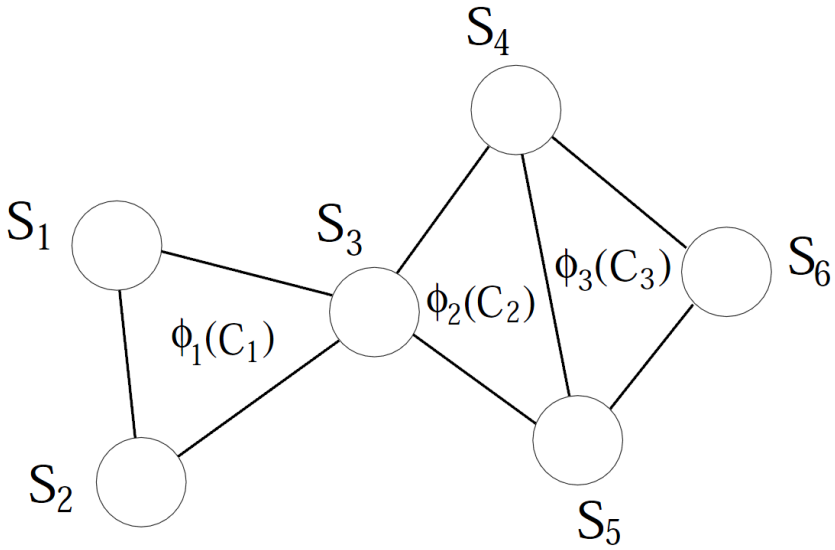
- В MRF заданы *потенциалы* ($\approx$ совместные распределения) на кликах графа $\phi(C_i)$, и общее совместное распределение равно

$$p(S) = \frac{1}{Z} \prod_{i=1}^{N} \phi_i(C_i).$$

BBN и MRF

Вот одна и та же структура в двух формализмах.





BBN и MRF

Вот очень общий алгоритм вывода на BBN с циклами.

1. Превратить BBN в MRF:
   - морализовать граф;
   - триангулировать граф.
2. Провести вывод на MRF:
   - построить дерево сочленений (junction tree);
   - запустить передачу сообщений на дереве сочленений.

Морализация

- Мы хотим, чтобы вершины, участвующие в одном маленьком распределении, были связаны друг с другом.
- При сходящейся связи это не так.
- *Морализация* – это просто добавление рёбер, связывающих родителей одного и того же ребёнка.
- После этого можно отбросить стрелочки, они больше не нужны.

Триангуляция

- Однако это ещё не всё. Полученный граф не обязательно будет подходить для MRF.
- Мы должны иметь возможность построить совместное распределение, последовательно перемножая распределения клик, причём последующие блоки должны зависеть только от предыдущих.
- Для цикла из четырёх вершин у нас появляются проблемы: мы напишем

$$p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_2)p(x_4 \mid x_3)$$

и упрёмся: оказывается, x_1 зависит от x_4, и цепочка нарушается.

Триангуляция

- Для цикла из трёх вершин всё ok, например,

$$p(x_1, x_2, x_3) = p(x_1)p(x_2, x_3 \mid x_1).$$

- Поэтому мы разобьём цикл из четырёх вершин на два треугольника, и с ним тоже всё станет хорошо:

$$p(x_1, x_2, x_3, x_4) = p(x_1)p(x_2, x_3 \mid x_1)p(x_4 \mid x_2, x_3).$$

Триангуляция

- Оказывается, что это достаточное условие.

Определение

Ненаправленный граф называется триангулярным, если у него нет циклов длины 4 без хорд.

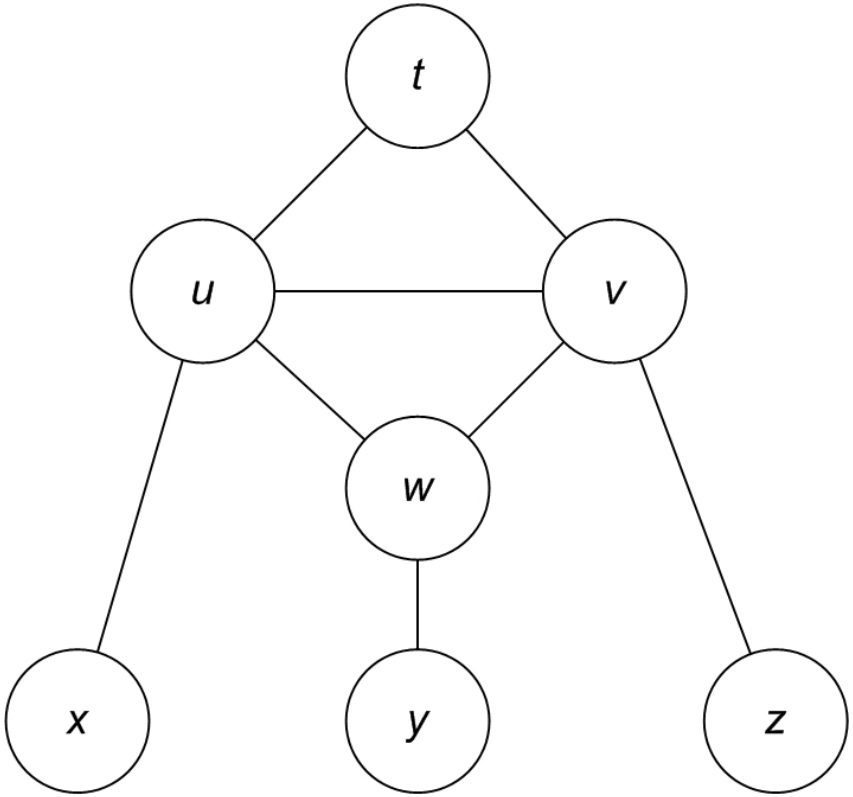
- Процесс такого разбиения называется *триангуляцией*.

Триангуляция

- Отдельная наука – алгоритмы *оптимальной триангуляции* (минимизирующие количество добавленных рёбер); это, конечно, NP-трудная задача, со своими приближёнными алгоритмами etc.
- Мы этим заниматься не будем, но приведём важную для наших целей переформулировку свойства триангулированности.

Outline

Моральный граф

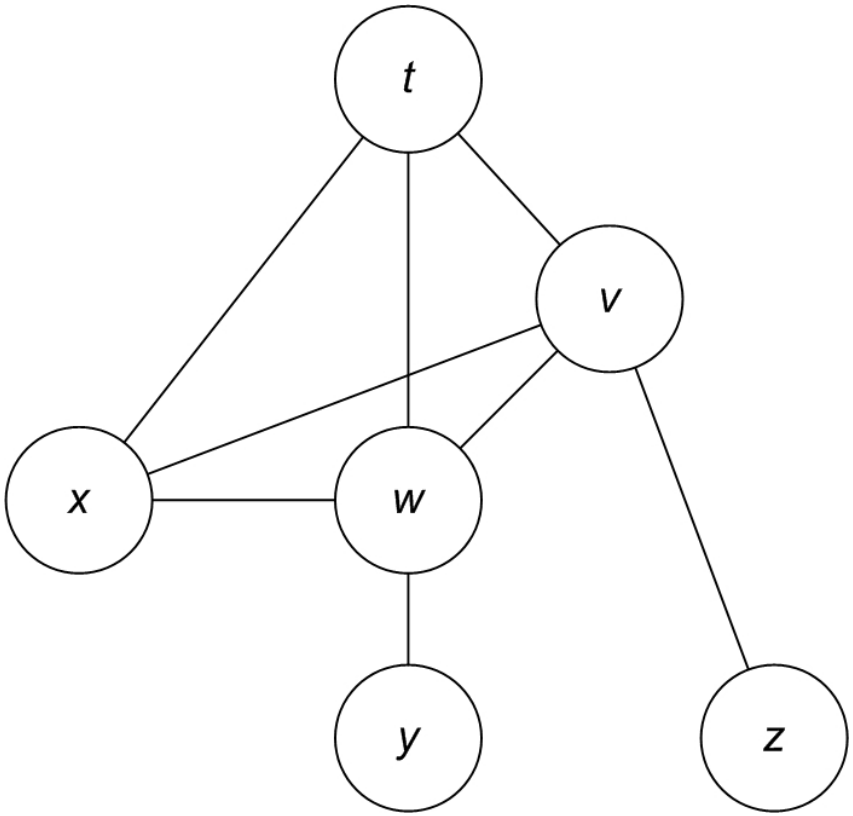


Добавилось ребро между u и v — общими предками w.

Дальнейший план

- Мы хотим в дальнейшем по одной удалять вершины из морального графа, при этом проецируя общее распределение на то, что останется.
- Каждый раз, когда мы удаляем вершину, нам приходится объединять её соседей, потому что нужно объединить распределение вероятностей.

Пример



Вот результат элиминации вершины u.

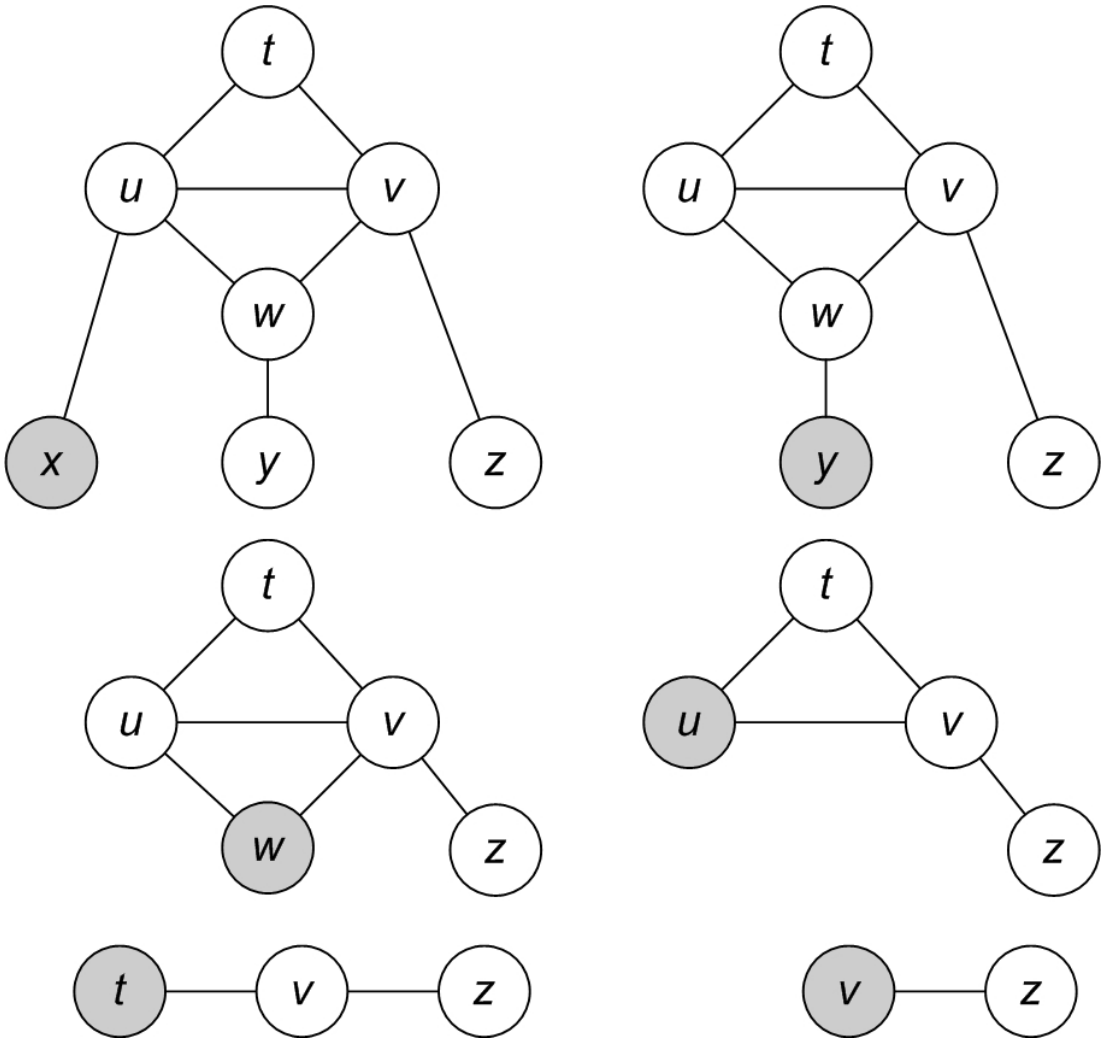
Идеальная элиминирующая последовательность

Цель понятна — нужно постараться избегать добавления новых рёбер при элиминировании переменных.

Определение

Идеальной элиминирующей последовательностью называется такая последовательность переменных, что при их элиминировании в этой последовательности ни разу ни одного ребра в моральный граф добавлено не будет.

Идеальная элиминирующая последовательность: пример



Симплициальные вершины

- Если $x_1, \ldots, x_i, \ldots, x_l$ — идеальная элиминирующая последовательность, и множество соседей x_i образует полный подграф, то $x_i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_l$ — также идеальная элиминирующая последовательность. Такие вершины называются *симплициальными*.
- Рассмотрим множество клик, которые появятся в моральном графе в процессе последовательной элиминации переменных по какой-либо последовательности. Выбросим подграфы других клик. Это множество всегда есть множество клик морального графа сети.

Упражнение. Докажите эти два утверждения.

Триангулярные графы

Определение

Ненаправленный граф называется триангулярным, если у него существует идеальная элиминирующая последовательность.

Мы не будем подробно доказывать, что это то же самое, что отсутствие циклов длины 4, только идею дадим.

Триангулярные графы и симплициальные вершины

Теорема

У неполного триангулярного графа, содержащего по крайней мере три вершины, всегда есть две несоседних симплициальных вершины.

- Следствие: Ненаправленный граф триангулярен iff все его вершины можно элиминировать, последовательно элиминируя симплициальные вершины.

Упражнение. Доведите эту идею до формального доказательства.

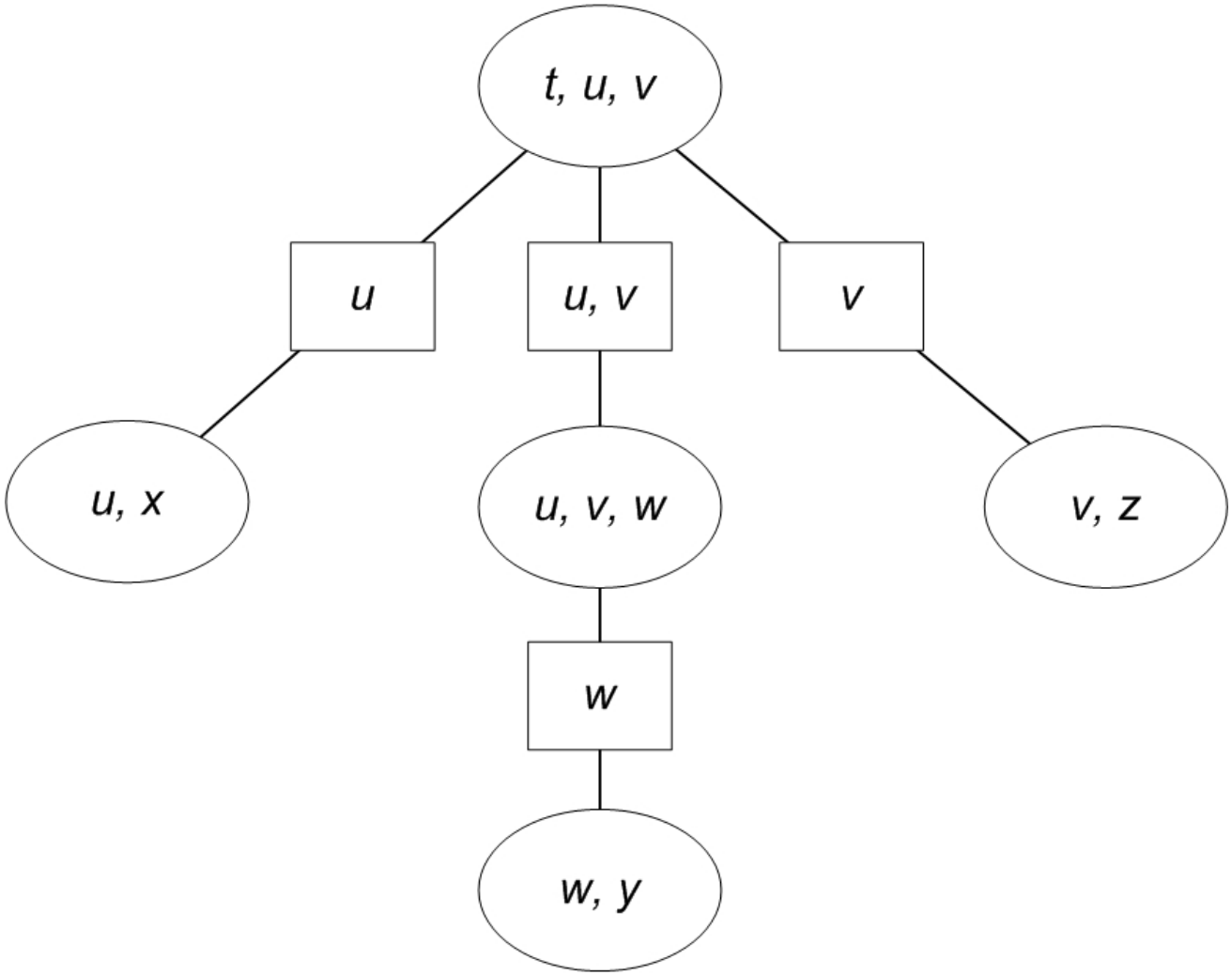
Дерево смежности

- Мы хотим действовать по идеальной элиминирующей последовательности.

Определение

Дерево смежности (join tree) — это дерево, вершинами которого служат элементы $\mathrm{Clique}(G)$ *(клики графа* G*), а ребра организованы так, что для любых двух вершин дерева смежности* C_1*,* C_2 *любая вершина на пути от* C_1 *к* C_2 *содержит их пересечение* $C_1 \cap C_2$ *(running intersection property).*

Дерево смежности: пример



Дерево смежности и триангуляция

Теорема

Ненаправленный граф триангулярен тогда и только тогда, когда для него существует дерево смежности.

Упражнение. Докажите.

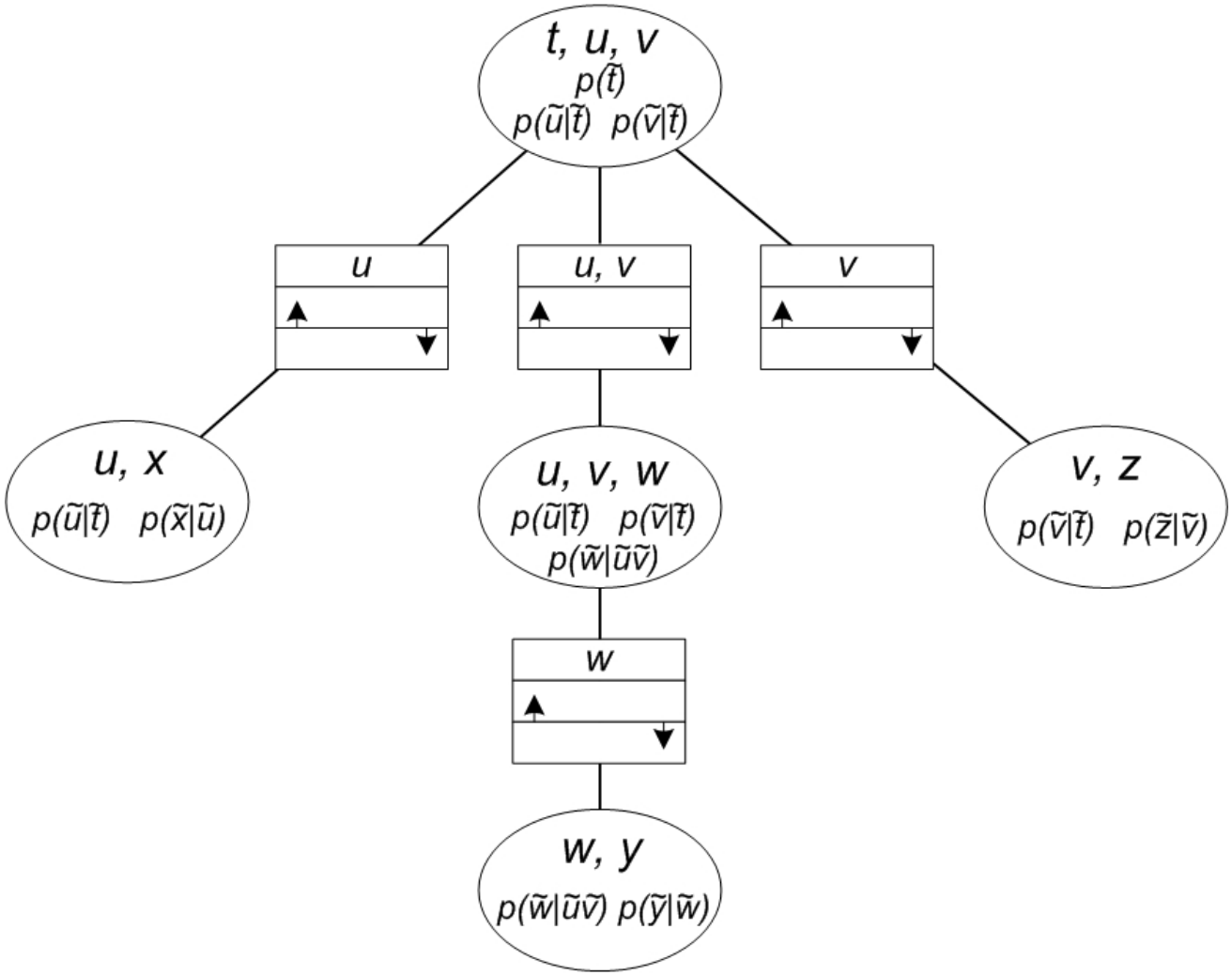
Дерево сочленений

Определение

Дерево сочленений (junction tree) для BBN — это дерево смежности её морального графа, где каждой вершине дополнительно приписаны условные вероятности исходной BBN, области определения которых полностью содержатся в соответствующей клике морального графа, а каждый из сепараторов содержит два почтовых ящика, предназначенных для передачи пересчитанных распределений в одну и другую сторону.

Конечно, реально дерево смежности и дерево сочленений – это один и тот же граф.

Дерево сочленений: пример



Идеальная нумерация

Определение

Фиксируем граф $G = (V, E)$. *Нумерация*

$$\sigma : \{1, \ldots, |V|\} \longrightarrow V$$

называется идеальной, если для всякого i *множество вершин*

$$\mathrm{Fam}(\sigma(i)) \cap \sigma(\{1, \ldots, i\})$$

индуцирует полный подграф G.

Алгоритм построения идеальной нумерации

- Для всех i от 1 до $|V|$:
  - найти еще не пронумерованную вершину x, для которой $|\mathrm{Fam}(x) \cap \sigma(1, \ldots, i-1)|$ максимальна (если таких вершин несколько, выбрать любую произвольным образом);
  - присвоить ей номер i, т.е. положить $\sigma(i) = x$.
- Выдать полученную нумерацию.

Алгоритм триангуляции

Кстати, простейший алгоритм триангуляции — запустить алгоритм, получить «идеальную» нумерацию, а потом сделать её идеальной:

- Построить нумерацию σ при помощи предыдущего алгоритма.
- Для всех i от $|V|$ до 1:

$$E = E \cup \{(x, y) \mid \\ \mid x, y \in \mathrm{Fam}(\sigma(i)) \cap \sigma(\{1, \ldots, i-1\}), (x, y) \notin E\}.$$

- Выдать граф (V, E).

Алгоритм построения дерева смежности

- $\boldsymbol{C} = \emptyset$.
- $\boldsymbol{E} = \emptyset$.
- Для всех i от 1 до $|V|$:
  - найти еще не пронумерованную вершину x, для которой $|\mathrm{Fam}(x) \cap \sigma(1, \ldots, i-1)|$ максимальна (если таких вершин несколько, выбрать любую произвольным образом);
  - присвоить ей номер i, т.е. положить $\sigma(i) = x$;
  - $C_i = \{\mathrm{Fam}(\sigma(i)) \cap \sigma(\{1, \ldots, i\})\}$;
  - $\boldsymbol{C} = \boldsymbol{C} \cup \{C_i\}$;
  - $\boldsymbol{E} = \boldsymbol{E} \cup (C_i, C_j)$, где $j = \max_{k<i, (i,k) \in E} k$.
- Для всех i от 1 до $|V|$, если C_i полностью содержится в одном из своих соседей $(\exists j : \ C_i \subseteq C_j \wedge (C_i, C_j) \in \boldsymbol{E})$, то удалить C_i из $\boldsymbol{C}$ и соединить соседей C_i напрямую.
- Выдать граф $(\boldsymbol{C}, \boldsymbol{E})$.

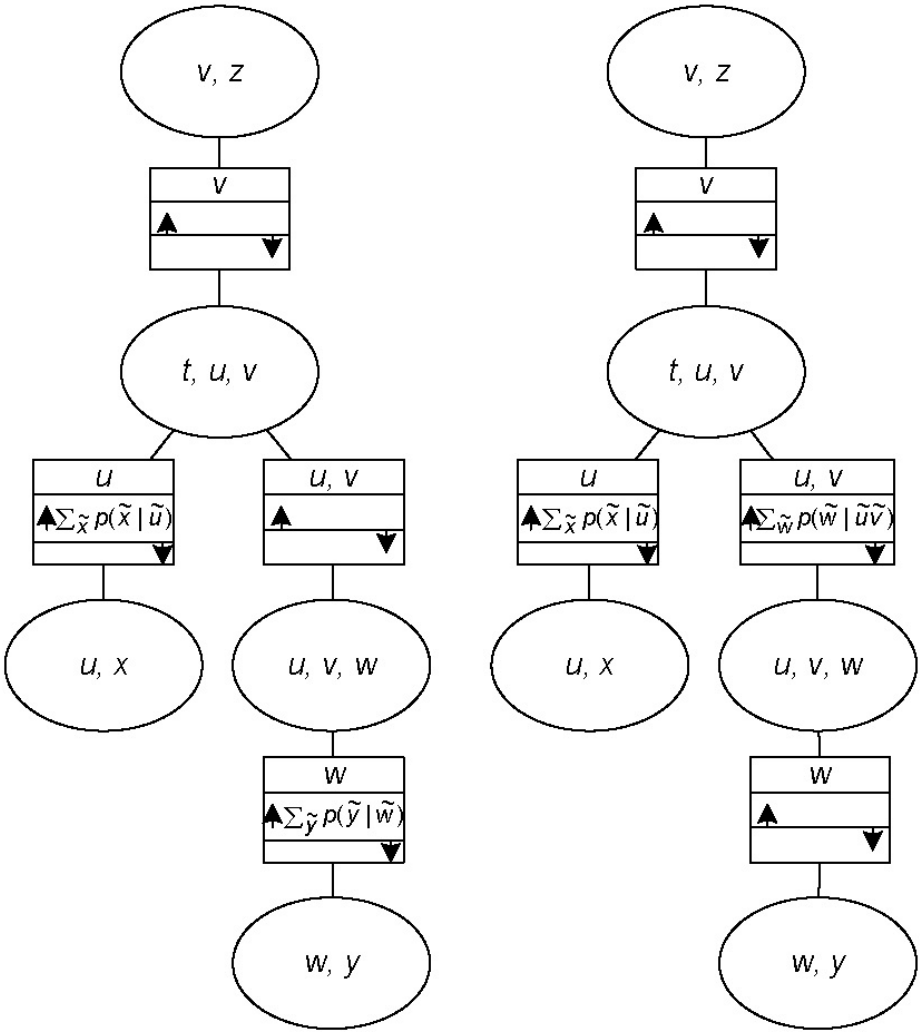
Схема алгоритма

- Итак, вот что у нас пока есть:
  1. Морализовать базовый граф БСД.
  2. Триангулировать моральный граф.
  3. Построить дерево смежности (оно же дерево сочленений).
  4. Выполнять собственно пропагацию.
- Первые три шага мы уже рассмотрели; четвёртый шаг – это уже знакомый нам алгоритм передачи сообщений.

Сбор сведений

- Предположим, что мы хотим найти z.
- Сначала нужно найти вершину дерева сочленений, содержащую z. В данном случае такая клика одна — $\{v, z\}$.
- Эта клика становится временным корнем дерева, а остальные вершины, начиная с листьев, посылают к временному корню сообщения, в которых записан результат суммирования по переменным, не содержащимся в ближайшем сепараторе (частичная маргинализация).
- Каждая вершина посылает сообщение «наверх» тогда, когда она получила все сообщения «снизу».

Первые два шага



Алгоритм

- На этапе сбора сведений условие выполняется снизу вверх, и узлы передают сообщения наверх.
- Потом, когда мы добрались до корня и пересчитали там, мы начинаем двигаться обратно; условие не меняется, но теперь оно работает для всех исходящих сепараторов.
- Алгоритм заканчивает работу, когда пустых почтовых ящиков больше нет.

Thank you!

Спасибо за внимание!