

Алгоритм передачи сообщений

Сергей Николенко

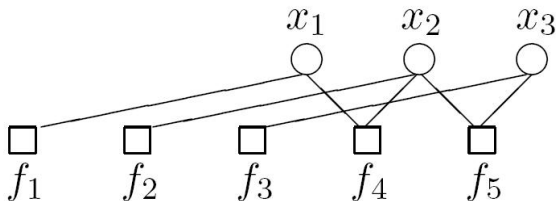
Казанский Федеральный Университет, 2014

Outline

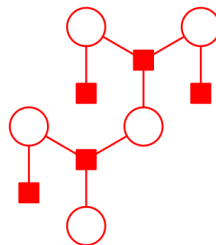
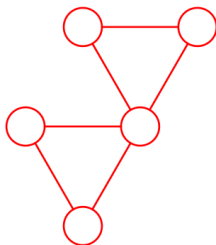
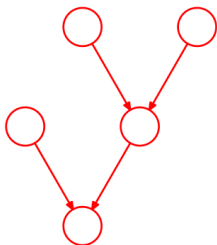
- 1 Алгоритм передачи сообщений
 - Фактор-графы
 - Sum-product и max-sum
- 2 Приближённый вывод
 - Сложная структура графа
 - Сложные факторы

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:

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

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:

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

- Мы хотим найти

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

Пример

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

$$\begin{aligned} \sum_{x_n} p(x_1, \dots, x_n) &= \\ &= \frac{1}{Z} \psi_{1,2}(x_1, x_2) \dots \psi_{n-2,n-1}(x_{n-2}, x_{n-1}) \sum_{x_n} \psi_{n-1,n}(x_{n-1}, x_n). \end{aligned}$$

- 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

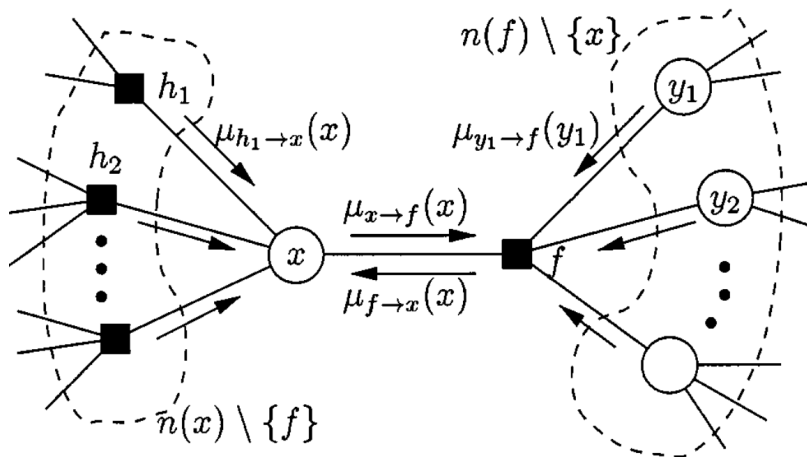
$$\mu_{\beta}(x_k) = \sum_{x_{k+1}} \psi_{k,k+1}(x_k, x_{k+1}) \left[\dots \left[\sum_{x_n} \psi_{n-1,n}(x_{n-1}, x_n) \right] \dots \right].$$

- 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 $p(x_1, \dots, x_n) = \prod_S f_S(X_S)$, 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 $p(x_1, \dots, x_n) = \prod_{s \in \text{ne}(x_k)} F_s(x_k, X_s)$, где X_s – переменные из поддерева с корнем в f_s . Тогда

$$\begin{aligned} p(x_k) &= \sum_{x_i \neq k} p(x_1, \dots, x_n) = \prod_{s \in \text{ne}(x_k)} \left[\sum_{X_s} F_s(x_k, X_s) \right] = \\ &= \prod_{s \in \text{ne}(x_k)} \mu_{f_s \rightarrow x_k}(x_k), \end{aligned}$$

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

Алгоритм передачи сообщений

- Чтобы найти $\mu_{f_s \rightarrow x_k}(x_k)$, 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

$$\begin{aligned} \mu_{f_s \rightarrow x_k}(x_k) &= \sum_{Y_s} f_s(x_k, Y_s) \prod_{y \in Y_s} \left(\sum_{X_{s,y}} G_y(y, X_{s,y}) \right) = \\ &= \sum_{Y_s} f_s(x_k, Y_s) \prod_{y \in Y_s} \mu_{y \rightarrow f_s}(y). \end{aligned}$$

Алгоритм передачи сообщений

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

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

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

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

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

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

Алгоритм передачи сообщений

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

$$p(x_k) = \prod_{f \in \text{ne}(x_k)} \mu_{f \rightarrow x_k}(x_k).$$

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

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

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

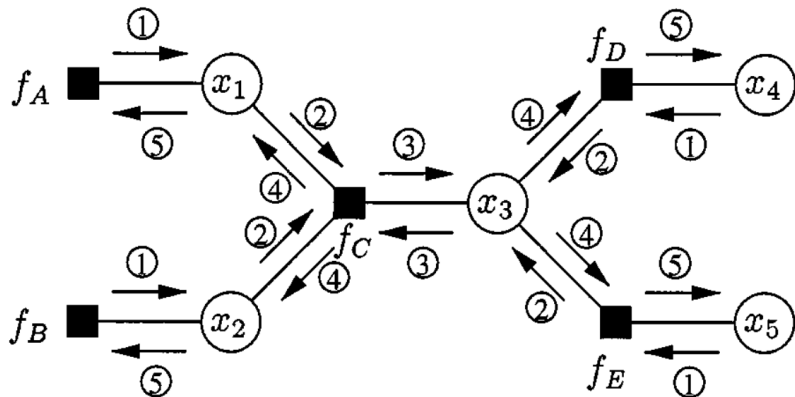
Алгоритм передачи сообщений

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

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

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

Передача сообщений



Так что же делать с байесовской сетью?

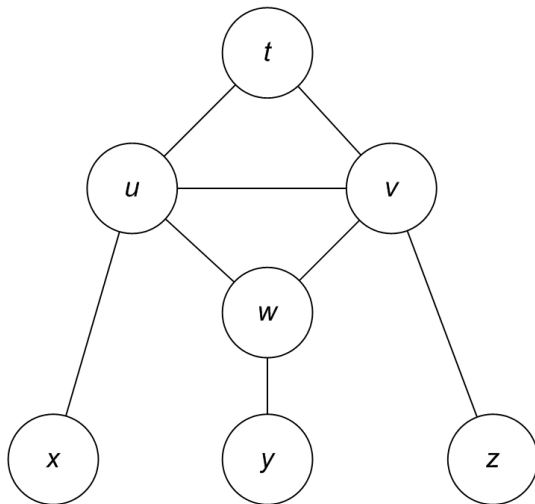
Для модели не в виде фактор-графа надо просто представить её в виде фактор-графа тем или иным способом.

Для байесовской сети это значит, что её сначала нужно представить в виде марковской сети, а потом добавить факторы в явном виде.

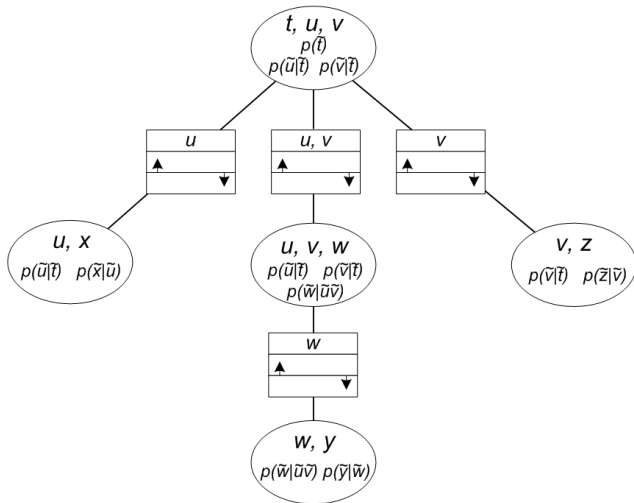
Так что же делать с байесовской сетью?



Так что же делать с байесовской сетью?



Так что же делать с байесовской сетью?



Outline

- 1 Алгоритм передачи сообщений
 - Фактор-графы
 - Sum-product и max-sum
- 2 Приближённый вывод
 - Сложная структура графа
 - Сложные факторы

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.

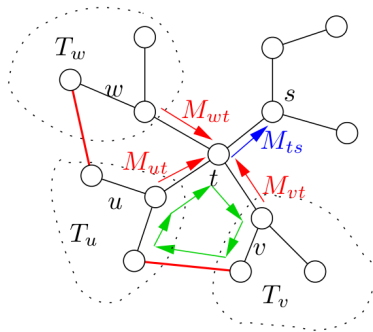
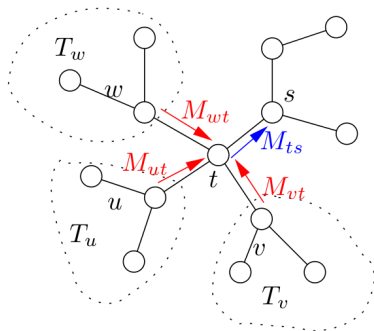
Деревья сочленений — неформально

- 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

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

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.

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

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

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

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

- 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 $\mu_{f_s \rightarrow x_k}(x_k)$ считать его приближение $\hat{\mu}_{f_s \rightarrow x_k}(x_k)$ из какого-нибудь простого семейства;
 - 2 повторять передачу сообщений, пока не сойдётся.

Thank you!

Спасибо за внимание!