Обучение структуры байесовских сетей при помощи пакета **bnlearn** системы **R**

Ромов Петр

27 апреля 2012 г.

1 Введение

Вероятностные модели играют важную роль в анализе данных. Как правило, под вероятностной моделью понимают совместное распределение, в котором заключены зависимости между переменными:

$$p(x_1, x_2, \ldots, x_n). \tag{1}$$

Имея вероятностную модель (1), мы можем проводить рассуждения. Пусть, к примеру, в нашу модель входят переменные x, y, z, тогда $p(x|y) = \frac{1}{p(y)} \int_z p(x, y, z)dz$ — распределение переменной x с учетом знания значения y и без учета знания z. Наличие вероятностной модели естественным образом решает проблему пропусков в данных, существенно расширяет круг задач прогнозирования.

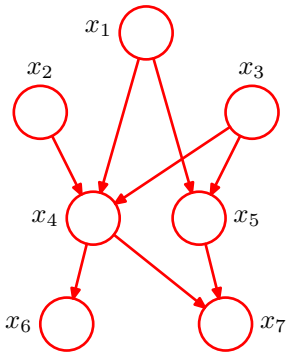


Рис. 1: Пример байесовской сети

Однако, проведение прогноза с использованием произвольной вероятностной модели (1) может привести к непреодолимым вычислительным трудностям, приходится упрощать модель. Задача обучения структуры байесовских сетей призвана сделать вероятностную модель как можно проще путем отказа от учета некоторых зависимостей между переменными.

Байесовские сети. Из теории вероятностей известно, что совместную плотность можно разложить на множители:

$$p(x_1, x_2, \ldots, x_n) = p(x_1|x_2, \ldots, x_n)p(x_2|x_3, \ldots, x_n) \ldots p(x_{n-1}|x_n)p(x_n). \tag{2}$$

Каждый множитель — условная плотность распределения одной переменной. Пусть плотность распределения $p(x_k|x_{k+1}, \ldots, x_n)$ меняется только при изменении набора переменных $\Pi_k = \{x_{\pi_1}, \ldots, x_{\pi_t}\} \subset \{x_{k+1}, \ldots, x_n\}$, т.е. $p(x_k|x_{k+1}, \ldots, x_n) = p(x_k|x_{\pi_1}, \ldots, x_{\pi_t}) = p(x_k|\Pi_k)$. Используя такие условные независимости можно выкинуть часть переменных в условных частях плотностей в выражении (2), тем самым сделав модель проще. Заметим, что порядок переменных $x_1, \ldots, x_n$ можно поменять произвольным образом.

Байесовская сеть — это ориентированный ациклический граф, вершинами которого являются переменные, а дуги определяют зависимости между переменными. Родителями вершины x_k будем называть множество вершин $\Pi_k = \{x_p | \exists \text{ дуга } x_p \to x_k\}$. Введенная таким образом, байесовская сеть однозначно определяет разложение совместной плотности на множители:

$$p(x_1, \ldots, x_n) = \prod_{k=1}^{n} p(x_k|\Pi_k). \tag{3}$$

Выражение (2) соответствует байесовской сети с полным графом (между двумя вершинами проходит ровно одна дуга). Изменение ориентации дуг в полной байесовской сети приводит к перестановке переменных в соответствующей факторизации совместной плотности (2). Избавляясь от

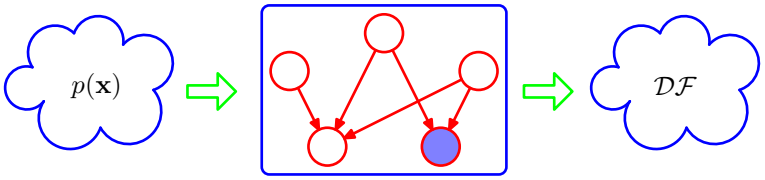


Рис. 2: На байесовские сети можно смотреть как на фильтр, который пропускает только те распределения $p(\mathbf{x})$, которые допускают разложение (3). Говорят, что такие распределения обладают свойством прямой факторизации *(direct factorization)*, образуют множество, обозначенное $\mathcal{DF}$.

дуг в полной байесовской сети мы тем самым убираем зависимости между переменными и упрощаем модель совместного распределения.

На Рис. 1 приведен пример байесовской сети, ей соответствует следующая факторизация совместной плотности:

$$p(x_1, \ldots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5). \tag{4}$$

Алгоритмы обучения структуры анализируют наборы значений переменных, входящих в вероятностную модель, строят байесовскую сеть, убирая часть зависимостей между переменными.

Байесовская сеть — вероятностная графическая модель *(Probabilistic Graphical Model, PGM)*, детальный обзор теории байесовских сетей и других графических моделей можно найти в книге [1]. Тема дальнейшего применения байесовских сетей: обучение параметров *(parameter learning)*, вывод *(inference)* раскрывается на спецкурсе ММП Д. П. Ветрова, Д. А. Кропотова «Байесовские методы машинного обучения».

2 Пакет bnlearn

Пакет `bnlearn` предоставляет пользователю класс `bn`, который содержит описание байесовской сети. В частности, `bn$learning` — информация о методе обучения структуры, при помощи которого была получена сеть, `bn$nodes` — описание переменных, `bn$arcs` — описание дуг в графе (связей между переменными).

Пакет позицианирует себя как инструмент работы с байесовскими сетями и предоставляет возможности обучения параметров, вывода, однако эти возможности достаточно скудны и рассмотрены не будут. Основная ценность `bnlearn` — обучение структуры.

2.1 Алгоритмы обучения структуры байесовской сети

Пакет `bnlearn` содержит набор алгоритмов. Далее в скобках будут указаны библиотечные названия алгоритмов и методов. Каждый алгоритм имеет собственную `R`-функцию, название которой соответствует указанному в скобках.

Все алгоритмы работают только в случае, когда либо все переменные дискретны (представлены факторами в `R`), либо все переменные вещественны (представлены числовыми массивами). На вход алгоритму обучения следует подавать наборы значений переменных на обучающей выборке в виде фрейма данных *(data frame)*.

Алгоритмы, основанные на анализе условной независимости (constraint-based).

- *Grow-shrink* (`gs`) [3].
- *Incremental association* (`iamb`) [5].
- *Fast incremental association* (`fast.iamb`) [7].
- *Interleaved incremental association* (`inter.iamb`) [5].
- *Max-min parents and children* (`mmpc`) [6].

Эти алгоритмы выбирают структуру байесовской сети на основании статистический и теоретико-информационных тестов на условную независимость. В результате работы алгоритма некоторые ребра в графе могут остаться неориентированными.

Функции имеют аргумент `test`, который позволяет выбрать метод определения условной независимости из следующих:

- Для сетей с дискретными переменными:
  1. По *взаимной информации* (`mi`, `mc-mi`), ускоренный но менее точный вариант (`fmi`).
  2. *Критерий согласия Пирсона* χ^2 (`xi2`, `mc-x2`).
  3. *Информационный критерий Акаике* (`aict`).
- Для сетей с вещественными переменными:
  1. По *коэффициенту корреляции* (`cor`, `mc-cor`).
  2. *Z-преобразование Фишера* (`zf`, `mc-zf`) — модификация коэффициента корреляции, которая используется в коммерческом ПО.
  3. Вещественный аналог *взаимной информации* (`mi-g`).

Некоторые тесты реализованы в двух вариантах: асимптотически нормальный (без префикса) и проведенный методом перестановок Монте-Карло [2] (префикс `mc-`). Тесты на условную независимость можно проводить отдельно от алгоритма обучения, для этого есть функция `ci.test`.

Ограничения условной независимости не всегда позволяют однозначно построить байесовскую сеть, поэтому на выходе у алгоритмов данного класса могут быть неориентированные ребра.

Вычислительная сложность этих алгоритмов зависит линейно от размера входных данных, обычно $O(N^2)$ ($O(N^4)$ в худшем случае) в зависимости от N — числа переменных.

Алгоритмы, основанные на мере качества сети (score-based). Пакет содержит только один алгоритм основанный на мере качества:

- *Hill-Climbing* (`hc`): Жадный поиск в пространстве ориентированных графов.

Алгоритм может сходиться к плохому локальному оптимуму, для предотвращения этого функция `hc` имеет аргументы `restart` и `perturb`, позволяющие устанавливать число случайных перезапусков алгоритма и перемешиваний структуры по ходу обучения, аргумент `start` позволяет задать байесовскую сеть в качестве начального приближения.

Алгоритм может оптимизировать различные меры качества, выбор меры происходит путем передачи аргумента `score`:

1. *Правдоподобие* (`lik`) и *логарифм правдоподобия* (`loglik`).
2. *Критерий Акаике* (`aic`) и *Байесовский информационный критерий* (`bic`).
3. Логарифм *Bayesian Dirichlet equivalent score* (`bde`).
4. Логарифм *K2 score* (`k2`).

Гибридные алгоритмы. Алгоритмы используют ограничения условной независимости, но в то же время оптимизируют меру качества сети. Они имеют аргументы `test` и `score`, аналогичные негибридным методам, а также аргумент `restrict`, позволяющий выбрать один из constraint-based методов для ограничения пространства байесовских сетей.

- *Max-Min Hill-Climbing* (`mmhc`): алгоритм Hill-Climbing (hc), примененный к ограниченному пространству байесовских сетей, ограничения формируются алгоритмом Max-Min Parents and Children (mmpc).
- *Restricted Maximization* (`rsmax2`): общий алгоритм, позволяет выбирать метод ограничения пространства байесовских сетей, выбирает лучшую при помощи Hill-Climbing.

Ручное введение ограничений на связи. Все алгоритмы в библиотеке имеют аргумент `whitelist` и `blacklist`, которые позволяют вручную установить ограничения на добавление или не-добавление некоторых связей между переменными. Эти ограничения могут быть известны эксперту, их добавление может помочь алгоритму найти лучшую структуру для сети (например сократить множество поиска для алгоритма hill-climbing).

2.2 Рекомендации по освоению

Данное руководство знакомит читателя с возможностями библиотеки. После освоения, его можно будет использовать как справочник, т.к. здесь представлен список всех алгоритмов и их опций с библиотечными названиями.

Более детальные теоретические сведения со ссылками приведены в статье [4]. Много полезной информации, примеры использования, наборы данных, руководство по установке можно найти на сайте библиотеки: `http://www.bnlearn.com/`. Наконец, технические детали и спецификации функций можно найти в технической документации к пакету, которую можно вызвать в `R`:

```
> help(nblearn)
```

3 Пример использования

3.1 Модельный пример

Будем рассматривать переменные a, b, c, d, e, f, g. Сгенерируем случайную выборку размера $N = 30$ следующим образом:

```
> N = 30;
> a = runif(N)                            # a не зависит от других переменных
> b = 3*a + rnorm(N, sd=1)                # b зависит только от a
> c = a*a + rnorm(N, sd=0.3)              # c зависит только от a, не линейно
> d = b+c + runif(N, min=-0.5, max=0.5)   # d зависит от b,c
> e = 5 - 2*d + rnorm(N)                  # e зависит от d
> f = 2*c*runif(N, min=0.9, max=1.1)      # f зависит от c, шум не гауссовский
> g = f + rnorm(N, sd=0.3)                # g зависит от f
```

Для простоты, почти все зависимости между переменными линейны с добавлением гауссовского шума:

$$x = Ay + B + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma),$$
$$p(x|y) = \mathcal{N}(Ay + B, \sigma)$$

где A, B — константы, σ — уровень шума.

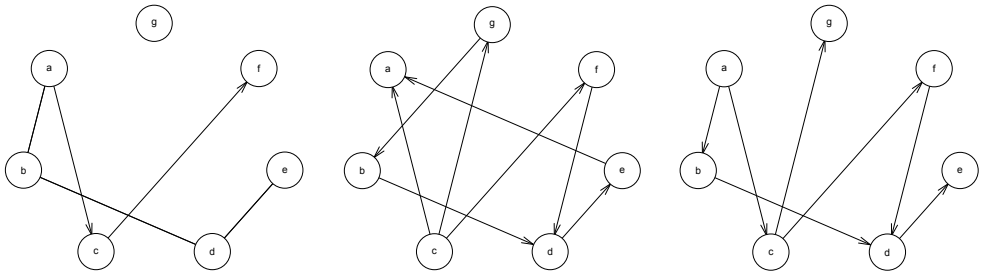
Обучим структуру байесовской сети различными методами:

```
> input = data.frame(a, b, c, d, e, f, g)
> bn.gs = gs(input)
> bn.iamb = iamb(input)
> bn.mmpc = mmpc(input)
> plot(bn.gs)             # нарисовать полученный граф структуры сети
```

Попробуем повлиять на обучение, введя ограничение: переменная c зависит исключительно a. Занесем в «белый список» связь $a \to c$, в «черный список» остальные связи, ведущие в вершину c.

```
> wl = data.frame(from=c('a'), to=c('c'))
> bl = data.frame(from=c('b', 'd', 'e', 'f', 'g'), to=c('c', 'c', 'c', 'c', 'c'))
> bn.gs = gs(input, whitelist=wl, blacklist=bl)
> bn.hc = hc(input, whitelist=wl, blacklist=bl, restart=10)
```

На Рис. 3 можно видеть графы зависимостей, полученные в результате обучения. Структура сети полученная методом Hill-Climbing с ограничениями для переменной c (Рис. 3c) отражает верные зависимости между переменными, незначительно отличается от модели, при помощи которой данные порождались.



(a) Метод Grow-Shrink (b) Метод Hill-Climbing (c) Улучшение метода HC за счет ограничения

Рис. 3: Примеры обученных байесовских сетей для модельного примера

3.2 Система мониторинга пациентов «ALARM»

Для создания системы мониторинга пациентов, позволяющей предупреждать врача о неблагоприятных изменениях состояния пациента, экспертами была построена байесовская сеть, которая связывала набор дискретных переменных. Пример переменной: «центральное венозное давление», принимает одно из трех значений «низкое», «нормальное», «высокое». Экспертная сеть используется для сравнения методов обучения структуры.

Модель содержит 37 дискретных переменных, 20'000 наборов значений для обучения. Набор данных устанавливается в `R` вместе с пакетом `bnlearn`, заргузим их и обучим структуру сети различными методами:

```
> data(alarm)
> alarm.gs <- gs(alarm)
> alarm.iamb <- iamb(alarm)
> alarm.fast.iamb <- fast.iamb(alarm)
> alarm.inter.iamb <- inter.iamb(alarm)
> alarm.mmpc <- mmpc(alarm)
> alarm.hc <- hc(alarm, score = "bic")
```

Обученные сети представлены на Рис. 4.

Список литературы

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2007.

[2] P.I. Good. *Permutation, parametric and bootstrap tests of hypotheses*. Springer Verlag, 2005.

[3] D. Margaritis. Learning bayesian network model structure from data. Technical report, DTIC Document, 2003.

[4] Marco Scutari. Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[5] I. Tsamardinos, C.F. Aliferis, A. Statnikov, and E. Statnikov. Algorithms for large scale markov blanket discovery. In *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference*, pages 376–381, 2003.

[6] I. Tsamardinos, L.E. Brown, and C.F. Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[7] S. Yaramakala and D. Margaritis. Speculative markov blanket discovery for optimal feature selection. In *Data Mining, Fifth IEEE International Conference on*, pages 4–pp. IEEE, 2005.

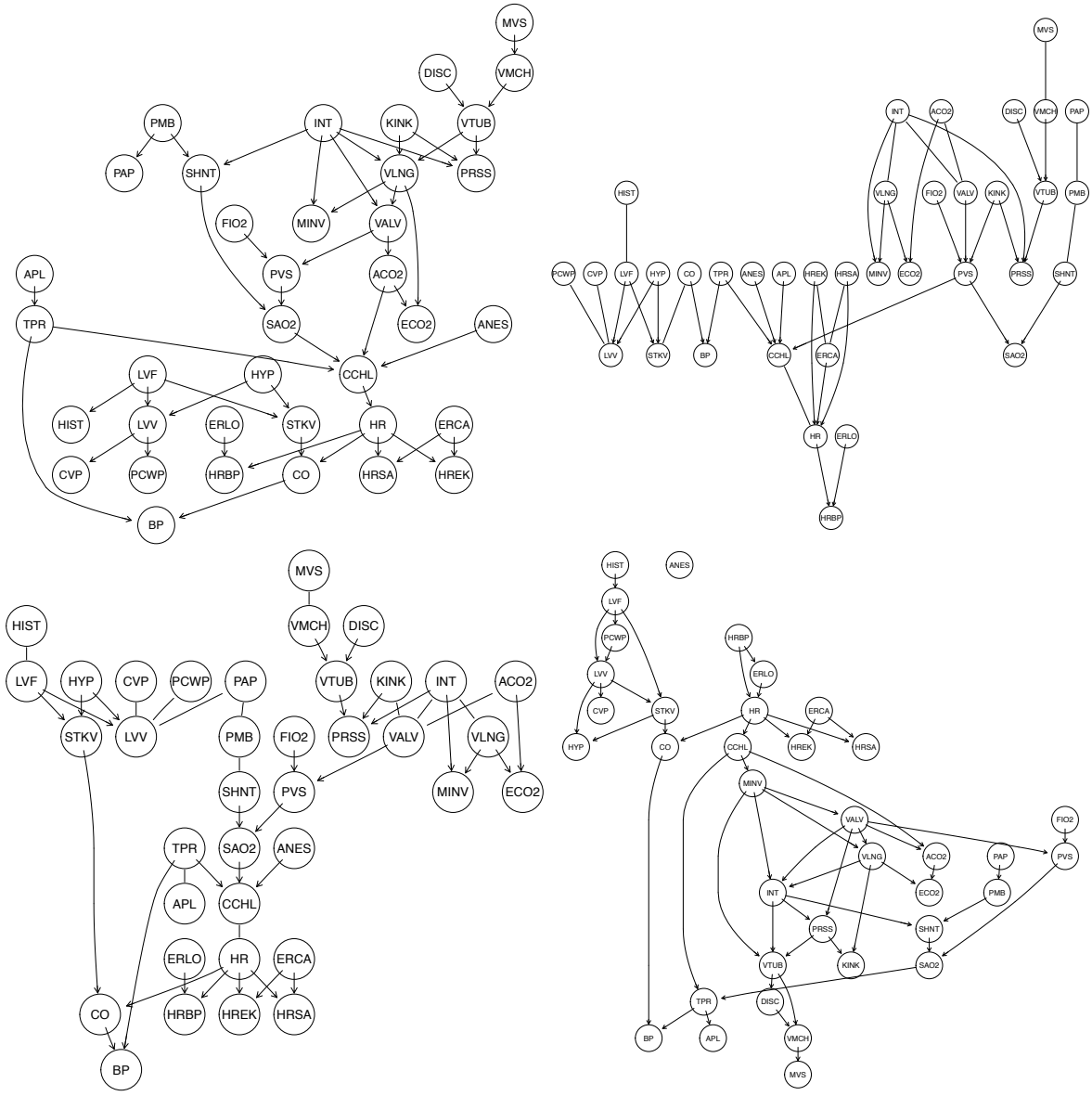


Рис. 4: Байесовские сети для набора ALARM: построенная экспертами (слева вверху), методом Grow-Shrink (справа вверху), Interleaved IAMB (слева внизу), Hill-Climbing (справа внизу).