

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ

Сергей Николенко

НИУ ВШЭ — Санкт-Петербург

29 сентября 2018 г.

Random facts:

- 29 сентября 1897 г. на премьере спектакля «Креольский суд» в театре «Олимпо» (Буэнос-Айрес) было впервые исполнено танго
- 29 сентября 1907 г. в Петербурге было торжественно открыто регулярное трамвайное движение
- 29 сентября 1922 г. из Петрограда отплыл «Обербургомистр Хакен», знаменитый «философский пароход»
- 29 сентября 1963 г. атомная подводная лодка «К-181» всплыла в полынье на Северном полюсе
- 29 сентября 1984 г. близ посёлка Куанда встретились и уложили последнее, «золотое» звено строители БАМа

МАТРИЧНЫЕ РАЗЛОЖЕНИЯ И ВЕРОЯТНОСТНЫЕ МОДЕЛИ

- Из чего складывается рейтинг пользователя i , который он выдал продукту a ?
- Вполне может быть, что пользователь добрый и всем подряд выдаёт хорошие рейтинги; или, наоборот, злой и рейтинг зажимает.
- С другой стороны, некоторые продукты попросту лучше других.
- Поэтому мы вводим так называемые *базовые предикторы* (baseline predictors) $b_{i,a}$, которые складываются из базовых предикторов отдельных пользователей b_i и базовых предикторов отдельных продуктов b_a , а также просто общего среднего рейтинга по базе μ :

$$b_{i,a} = \mu + b_i + b_a.$$

- Чтобы найти предикторы, уже нужен байесовский подход: надо добавить нормально распределённый шум и получить модель линейной регрессии

$$r_{i,a} \sim N(\mu + b_i + b_a, \sigma^2).$$

- Можно ввести априорные распределения и оптимизировать; или просто найти среднеквадратическое отклонение с регуляризатором:

$$b_* = \arg \min_b \sum_{(i,a)} (r_{i,a} - \mu - b_i - b_a)^2 + \lambda_1 \left(\sum_i b_i^2 + \sum_a b_a^2 \right).$$

- Как обучить такую модель?

- Правильно, это просто линейная регрессия!
- Можно обучать любым способом, которым мы умеем обучать линейную регрессию, например решить линейную систему:

$$b_* = (X^T X + \lambda I)^{-1} X^T r,$$

где r – вектор рейтингов, X – матрица из нулей и единиц (в каждой строке по три единицы).

- Часто у нас рейтинги бинарные: лайк/дизлайк, например.
- Тогда разумнее не пытаться приближать 0-1 данные линейной функцией, а рассмотреть их как распределение вероятностей.
- Давайте моделировать через логистический сигмоид:

$$b_{i,a} = \sigma(\mu + b_i + b_a), \quad \sigma(x) = \frac{1}{1 + e^{-x}}.$$

- Теперь это не линейная регрессия, а логистическая; её мы тоже умеем обучать.
- Часто это хороший метод, даже если рейтингов несколько.

- С тем, чтобы напрямую обучать оставшуюся матрицу предпочтений вероятностными методами, опять же есть проблема – матрица X , выражающая рейтинги, содержит $N \times M$ параметров, гигантское число, которое, конечно, никак толком не обучить.
- Более того, обучать их и не надо – как мы уже говорили, данные очень разреженные, и «на самом деле» свободных параметров гораздо меньше, проблема только с тем, как их выделить.
- Поэтому обычно число независимых параметров модели необходимо уменьшать.

- Метод SVD (singular value decomposition) – разложим матрицу X в произведение матриц маленького ранга.
- Зафиксируем некоторое число f *скрытых факторов*, которые так или иначе описывают каждый продукт и предпочтения каждого пользователя относительно этих факторов.
- Пользователь – вектор $p_i \in \mathbb{R}^f$, который показывает, насколько пользователь предпочитает те или иные факторы; продукт – вектор $q_a \in \mathbb{R}^f$, который показывает, насколько выражены те или иные факторы в этом продукте.

- Предпочтение в итоге будем подсчитывать просто как скалярное произведение $q_a^\top p_i = \sum_{j=1}^f q_{a,j} p_{i,j}$.
- Таким образом, добавляя теперь сюда baseline-предикторы, получаем следующую модель предсказаний рейтингов:

$$\hat{r}_{i,a} \sim \mu + b_i + b_a + q_a^\top p_i.$$

- Как обучить такую модель?

- SGD – стохастический градиентный спуск.
- Считаём градиент функции правдоподобия, идём по тренировочным примерам, апдейтим на каждом шаге:

$$b_i := b_i + \gamma (e_{i,a} - \lambda_2 b_i),$$

$$b_a := b_a + \gamma (e_{i,a} - \lambda_2 b_a),$$

$$q_{a,j} := q_{a,j} + \gamma (e_{i,a} p_{i,j} - \lambda_2 q_{a,j}) \text{ для всех } j,$$

$$p_{i,j} := p_{i,j} + \gamma (e_{i,a} q_{a,j} - \lambda_2 p_{i,j}) \text{ для всех } j,$$

где γ – скорость обучения.

- ALS – попеременные наименьшие квадраты.
- Заметим, что если в $\hat{r}_{i,a} \sim \mu + b_i + b_a + q_a^\top p_i$ фиксировать p_i , то по q_a будет опять линейная регрессия.
- ALS – повторять до сходимости:
 - фиксируем p_i , обучим q_a ;
 - фиксируем q_a , обучим p_i .
- Обычно более устойчив и быстрее работает, чем SGD.

- То же замечание про логистический вариант – при бинарных рейтингах можно рассмотреть

$$\hat{r}_{i,a} \sim \sigma(\mu + b_i + b_a + q_a^\top p_i).$$

- Тогда в SGD просто добавятся производные сигмоида $\sigma'(x) = \sigma(x)(1 - \sigma(x))$.
- А в ALS вместо линейной регрессии на каждой итерации надо будет обучать логистическую.

- Пусть мы хотим построить разложение матрицы рейтингов на матрицы меньшего ранга:

$$\hat{R} = U^T V.$$

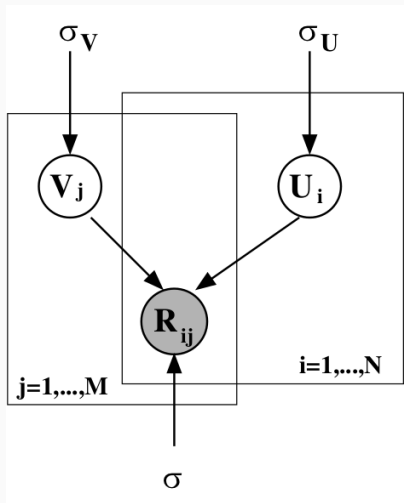
- Вероятностно мы имеем правдоподобие

$$p(R | U, V, \sigma^2) = \prod_i \prod_a (N(r_{i,a} | u_i^T v_j, \sigma^2))^{[i \text{ оценил } a]}.$$

- Добавим гауссовские априорные распределения на U и V :

$$p(U | \sigma_U^2) = \prod_i N(U_i | 0, \sigma_U^2 I), \quad p(V | \sigma_V^2) = \prod_a N(V_a | 0, \sigma_V^2 I).$$

ГРАФИЧЕСКАЯ МОДЕЛЬ



ВЕРОЯТНОСТНОЕ РАЗЛОЖЕНИЕ МАТРИЦ

- Если просто зафиксировать σ^2 , σ_V^2 и σ_U^2 , то они будут играть роль регуляризаторов, и нет никакого отличия от «обычного» SVD.
- Разница здесь в том, что теперь мы можем автоматически найти оптимальные $\sigma = (\sigma^2, \sigma_V^2, \sigma_U^2)$, максимизируя общее правдоподобие модели

$$\sigma^* = \arg \max_{\sigma} p(R | \sigma) = \arg \max_{\sigma} \int p(R, U, V | \sigma) dU dV$$

EM-алгоритмом:

- сначала зафиксируем σ и найдём

$$f(\sigma) = E_{U, V | R, \sigma} [\log p(R, U, V | \sigma)];$$

- потом максимизируем

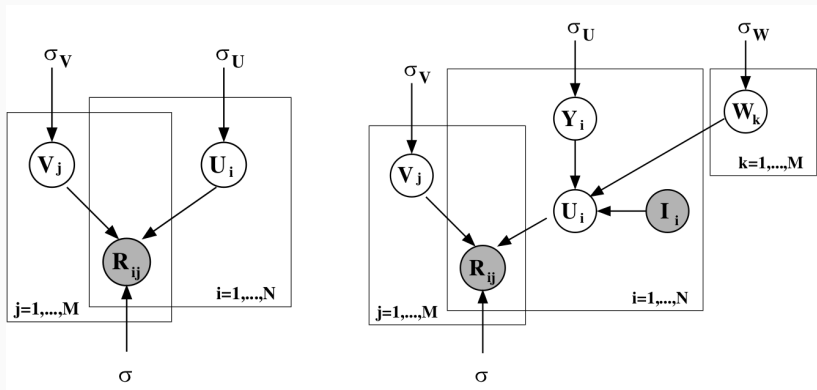
$$\sigma := \arg \max_{\sigma} f(\sigma).$$

- Модификация: пользователи с малым числом оценок в PMF получают апостериорные распределения, очень похожие на «среднего пользователя».
- Чтобы на редких пользователей лучше обобщалось, добавим ещё факторы, которые меняют априорное распределение факторов у пользователя в зависимости от того, сколько и чего он оценил:

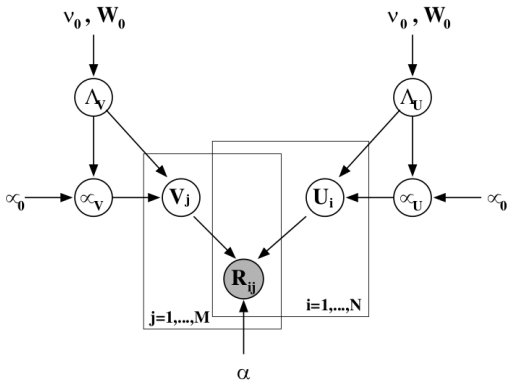
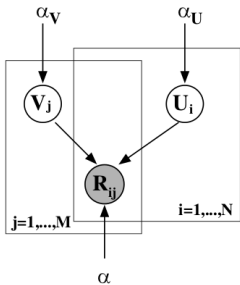
$$U_i = Y_i + \frac{\sum_a [i \text{ оценил } a] W_a}{\sum_a [i \text{ оценил } a]}.$$

- Матрица W показывает, как влияет на априорное распределение
- Тоже в качестве регуляризатора берём априорный гауссиан:
 $p(W | \sigma_W^2) = \prod_i N(W_i | 0, \sigma_W^2 I).$

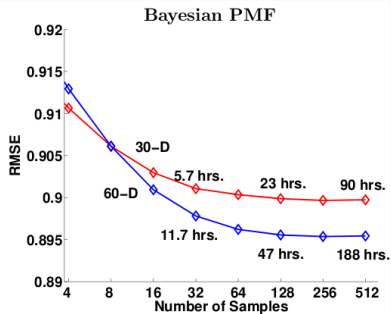
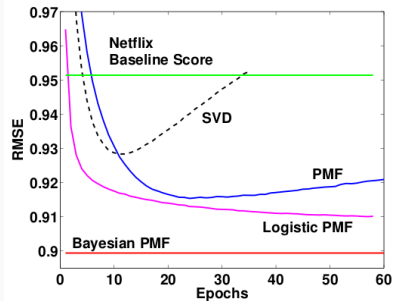
ГРАФИЧЕСКАЯ МОДЕЛЬ



ГРАФИЧЕСКАЯ МОДЕЛЬ



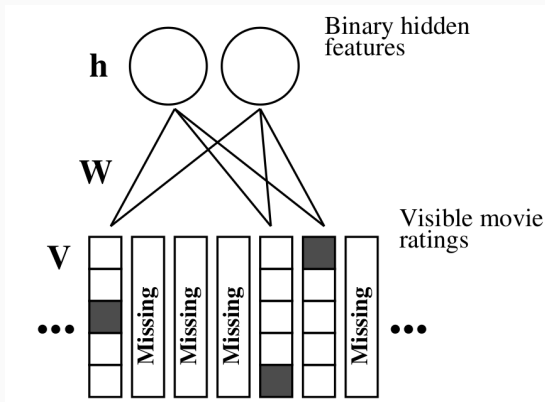
ГРАФИЧЕСКАЯ МОДЕЛЬ



- Ещё один метод вероятностного моделирования – *машины Больцмана* (restricted Boltzmann machine).
- Ненаправленная графическая модель, состоящая из двух уровней, видимого и скрытого.
- Машины Больцмана – один из основных блоков, на которых строят deep learning, так что о них мы поговорим гораздо подробнее позже.

МАШИНЫ БОЛЬЦМАНА

- В коллаборативной фильтрации мы строим машиной Больцмана модель предпочтений пользователя.



- В результате на скрытом слое обучается модель пользователя.
- Метод обучения – contrastive divergence (приближение к максимальному правдоподобию).
- RBM не то чтобы *лучше* SVD, но часто ошибается в *других местах*, поэтому комбинация этих двух моделей даёт значительное улучшение.

- Кстати, что значит «комбинация моделей»?
 1. Просто линейная комбинация (байесовское усреднение или регрессия).
 2. *Бустинг*: метод комбинации простых классификаторов; в качестве простых классификаторов можно брать результаты сложных моделей (оценки вероятности успеха или ожидаемый рейтинг).
- Современные рекомендательные системы – обычно большие *ансамбли* моделей.
- Два уровня: обучение отдельных моделей в ансамбле (*bootstrapping* и т.п.) и обучение комбинации.

- Мы только что описали метод SVD (singular value decomposition) – разложение матрицы X в произведение матриц маленького ранга.
- Это не единственное матричное разложение, и все они интересны по-своему.

- PCA (анализ главных компонент) – один из самых популярных методов сокращения размерности.
- PCA пытается объяснить как можно больше дисперсии исходного датасета.
- Однако часто направления на кластеры в данных не ортогональны, а признаки PCA трудно интерпретировать.

- SVD (сингулярное разложение) мы уже рассматривали.
- Оно делает ровно то, что нам нужно, когда доступны рейтинги:
 - максимизирует правдоподобие имеющихся рейтингов (минимизирует ошибку предсказания рейтингов);
 - умеет работать с разреженными матрицами (минимизирует только по имеющимся рейтингам).
- Но что делать, когда рейтингов никаких нет, а есть только факт использования? SVD тогда ничего хорошего не сделает...

- NMF (неотрицательное разложение): раскладываем по-прежнему в произведение

$$X \approx UV^T,$$

где U размера $n \times f$, V размера $m \times f$, и f гораздо меньше n и m .

- Но теперь требуем, чтобы элементы U и V были неотрицательны.
- Результат – признаки, которые лучше интерпретируются и могут иметь больше смысла.

- NMF можно использовать для рекомендаций, когда рейтингов нет.
- Берём матрицу X , в которой единицы – продукты, использованные пользователем, остальное нули.
- И раскладываем по NMF (методом ALS, нужно итеративно решать системы размерности $n \times f$ и $m \times f$).
- Задача перестаёт быть разреженной, но кое-что сделать можно.

МЕТРИКИ КАЧЕСТВА И РАСШИРЕНИЯ

- Ещё одна важная тема, которую в рекомендательных системах часто замалчивают.
- Как оценить качество рекомендаций? Какая должна быть метрика качества?
- Когда мы обучаем SVD (максимизируем правдоподобие), мы оптимизируем среднеквадратичную ошибку отклонения.
- И Netflix Prize, например, так и был сформулирован: надо было оптимизировать RMSE, среднеквадратичное отклонение предсказаний от истинного рейтинга пользователя.
- Но что нам надо на самом деле? Что у нас в тестовом множестве?

- В тестовом множестве отложены рейтинги некоторых продуктов, которым пользователь дал оценку.
- Наша задача – выдать пользователю топ-рекомендации; не предсказать все рейтинги, а найти, у каких продуктов рейтинг будет самым большим.
- То есть на самом деле это задача *ранжирования*! И метрики качества лучше брать из *information retrieval*; там, когда оценивают качество выдачи, вовсе не пытаются минимизировать отклонение предсказания функции релевантности.
- Дальше для простоты будем рассматривать бинарный случай (лайк-дизлайк).

- Классические метрики:
 - (1) точность (precision) – количество «хороших» (релевантных) запросу в случае поисковой выдачи, отмеченных высокой оценкой в случае рекомендательной системы) документов в выдаче, делённое на общее количество документов в выдаче;
 - (2) полнота (recall) – количество «хороших» документов в выдаче, делённое на общее количество релевантных документов в базе поисковой системы.
- Однако здесь те же проблемы; эти параметры не зависят от ранжирования выдачи, надо знать заранее, сколько потребуется рекомендаций.

- Метрики качества ранжирования:
 - NDCG, Normalized Discounted Commulative Gain; выберем топ- k рекомендаций (k может быть заведомо больше нужного числа) и посчитаем:

$$\text{DCG}_k = \sum_{i=1}^k \frac{2^{\hat{r}_i} - 1}{\log_2(1 + i)},$$
$$\text{NDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k},$$

где \hat{r}_i – наша оценка рейтинга продукта на позиции i , а IDCG_k – значение DCG_k при ранжировании по истинным значениям (рейтингам из валидационного набора);

- NDCG от 0 до 1, но ей трудно придумать естественную интерпретацию (как вероятность чего-нибудь, например).

- Метрики качества ранжирования:
 - AUC, Area Under (ROC) Curve; можно считать по всей выдаче сразу;
 - AUC – вероятность того, что случайно выбранная пара продуктов с разными оценками будет отранжирована правильно (понравившийся будет выше в выдаче, чем не понравившийся);
 - в бинарном случае можно посчитать в замкнутом виде:

$$\hat{A} = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1},$$

где n_0, n_1 – число понравившихся и не понравившихся пользователю объектов, $S_0 = \sum p_i$ – сумма номеров позиций понравившихся объектов в выдаче.

- Метрики качества ранжирования:
 - но на самом деле простые метрики тоже важны, потому что обычно пользователь успевает увидеть только несколько самых верхних рекомендаций;
 - WTA (winner takes all) – эта метрика равна 1, если топ-рекомендация (с самым большим предсказанным рейтингом) из просмотренных пользователем получила положительную оценку, и 0 в противном случае;
 - Top_k – доля положительных оценок среди топ- k рекомендаций (Top_{10} часто называют MAP – mean average precision).

- Проблема: холодный старт.
- Надо как-то инициализировать; если вообще ничего не знаем, сделать ничего нельзя, конечно.
- Но так не бывает; обычно есть набор признаков – можно пытаться предсказывать значения факторов:
 - просто регрессией по признакам;
 - (обычно для продуктов) выделяя темы при помощи topic modeling.

- Получается, что для признаков пользователя x_i и продукта x_a мы рассматриваем модель

$$r_{i,a} \sim \mu + b_{\text{user}}(x_i) + b_{\text{item}}(x_a) + q_a^\top p_i(t),$$

где

$$b_{\text{user}}(x_i) \sim N(u(x_i), \sigma_u^2),$$

$$b_{\text{item}}(x_i) \sim N(v(x_i), \sigma_v^2),$$

и в качестве u и v может выступать любая регрессия [Agarwal, Chen, 2009].

- Ещё один вариант, через контент:
 - выделить темы из продуктов (LDA), получится распределение $z_{a,k}$ для каждого a ;
 - обучить факторы $s_{i,k}$ того, насколько пользователю “нравятся” эти темы;
 - затем для нового продукта оценить темы $\hat{z}_{a,k}$ по контенту, а потом добавлять в модель слагаемое

$$r_{i,a} \sim \dots + \sum_k s_{i,k} \hat{z}_{a,k},$$

что помогает для холодного старта по продуктам.

- Есть модели, связанные с тем, как обучить не абы какие темы, а хорошо выражающие предпочтения (fLDA).

ВРЕМЯ В КОЛЛАБОРАТИВНОЙ ФИЛЬТРАЦИИ

- Пример: давайте добавим время, т.е. будем рассматривать базовые предикторы и характеристики пользователя как функции от времени:

$$\hat{r}_{i,a} = \mu + b_i(t) + b_a(t) + q_a^\top p_i(t),$$

где

$$b_a(t) = b_a + b_{a, \text{Bin}(t)},$$

$$b_i(t) = b_i + \alpha_i \text{dev}_i(t) + b_{i,t},$$

$$p_{i,f}(t) = p_{i,f} + \alpha_{i,f} \text{dev}_i(t) + p_{i,f,t} + \frac{1}{\sqrt{|V(i)|}} \sum_{b \in V(i)} y_b,$$

$$\text{dev}_i(t) = \text{sign}(t - t_i) |t - t_i|^\beta.$$

- Это называется timeSVD++, и эта модель была одним из основных компонентов модели, взявшей Netflix Prize.

- Предположим, что пользователи приходят из социальной сети.
- Т.е. есть друзья, есть социальный граф (его часть) и т.д. Это тоже можно добавить в рекомендательную модель:
 - фильтр/перевзвешивание в методе ближайших соседей;
 - дополнительные слагаемые в разложение типа SVD;
 - разложение матрицы доверия (из социального графа) вместе с матрицей рейтингов, меняем априорное распределение для PMF и т.д.

- Filter bubble: как вывести человека за его привычный круг.
- Можно до конца жизни рекомендовать одно и то же;
метрики:
 - diversity – разнообразие, мера схожести элементов списка;
 - novelty – новизна для пользователя, распространённость продукта, доля его рейтингов;
 - serendipity – неожиданность, сюрприз, схожесть на историю пользователя.
- Для всего этого нужно уметь распознавать схожесть контента рекомендованных товаров.

- CARS (context-aware recommender systems) – мы рекомендуем в контексте:
 - временном;
 - ситуативном;
 - географическом;
 - предшествующего поведения пользователей и т.д.

- Формально контекст – это новые измерения в матрице предпочтений.
- Получается “гиперкуб” данных, есть методы тензорного разложения, аналогичного SVD.
- Но часто не хуже работают простые решения – отфильтровать контексты и обучить модели только по этим данным, добавить полученные модели и сам контекст как факторы в бленд.

ГРАФИЧЕСКИЕ МОДЕЛИ

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.

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

$$p(x, y, z) = p(x | y, z)p(y | z)p(z).$$

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

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

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

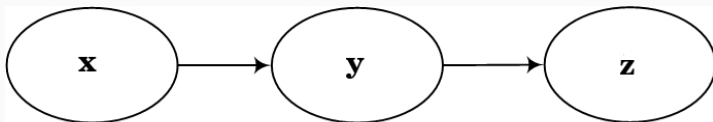
- Другими словами, если мы можем разложить большое совместное распределение в произведение локальных распределений, каждое из которых связывает мало переменных, это хорошо. :)

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

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

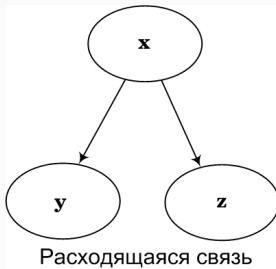
- Что можно сказать о (не)зависимости случайных величин x_i и x_j ?
- Задача вывода на графической модели: в некоторой части вершин значения наблюдаются, надо пересчитать распределения в других вершинах (подсчитать условные распределения). Например, из этой модели получатся и задача обучения параметров,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

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

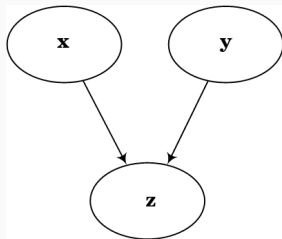


Последовательная связь

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



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



Сходящаяся связь

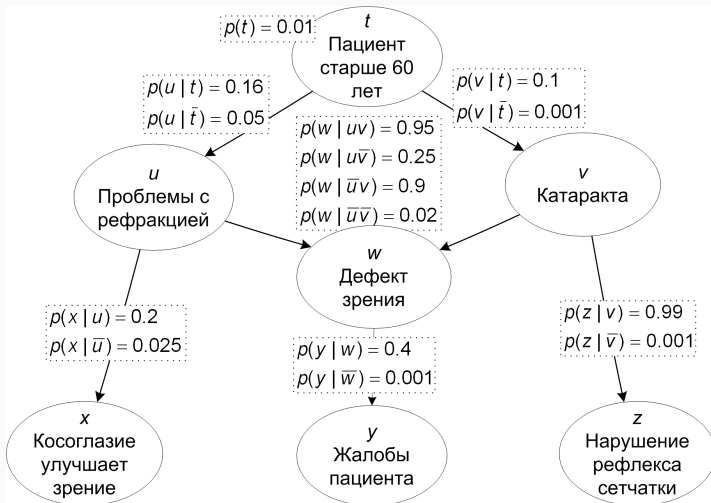
Обобщение: если наблюдается хотя бы один из потомков z , уже может не быть независимости между x и y .

- 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.

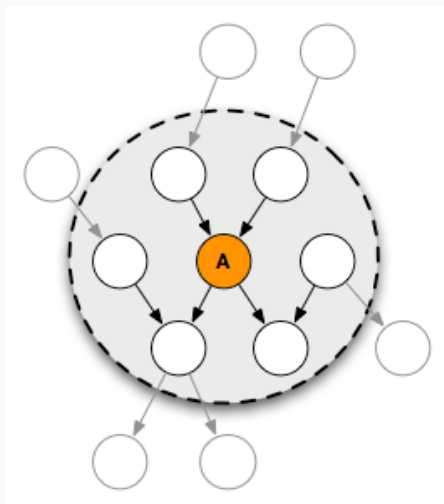
- Можно сказать, что граф задаёт некоторое семейство распределений – не все распределения на вершинах графа будут соответствовать тем ограничениям по условной независимости, которые накладывает структура графа.
- Теорема (без доказательства): это семейство распределений в точности совпадает с семейством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

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

ПРИМЕР БАЙЕСОВСКОЙ СЕТИ



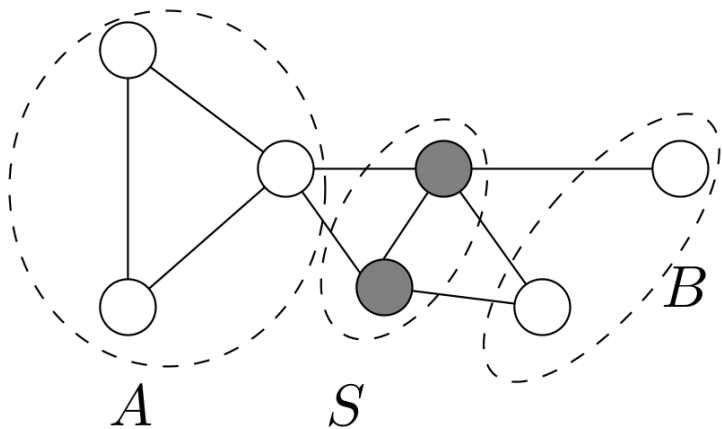
- Интересный вопрос: какие вершины нужно означить, чтобы наверняка «отрезать» одну вершину (Markov blanket)?
- Иначе говоря, для какого минимального множества вершин X $p(x_i | x_{j \neq i}) = p(x_i | X)$?



ДРУГИЕ ГРАФИЧЕСКИЕ МОДЕЛИ

- 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,
$$p(x_i, x_j | x_{k \neq i, j}) = p(x_i | x_{k \neq i, j})p(x_j | x_{k \neq i, j})$$
 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



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

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

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

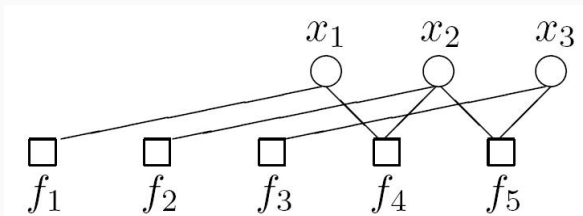
- Поскольку $\psi_C \geq 0$, их обычно представляют как экспоненты:

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

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

- 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).

- 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: $p(x_1, x_2, x_3) = f_1(x_1)f_2(x_2)f_3(x_3)f_4(x_1, x_2)f_5(x_2, x_3)$.



Спасибо за внимание!