Рекомендательные системы

Сергей Николенко

Mail.Ru, 19 апреля 2014 г.

Outline

Рекомендательные системы

- Рекомендательные системы анализируют интересы пользователей и пытаются предсказать, что именно будет наиболее интересно для конкретного пользователя в данный момент времени.
- Компании–лидеры в рекомендательных системах в основном делятся на две категории:
  1. мы «продаём» какие-то товары или услуги онлайн; у нас есть пользователи, которые либо явно оценивают товары, либо просто что-то покупают, а что-то нет; интересно порекомендовать товар, который данному покупателю максимально понравится; Netflix, Amazon;
  2. мы – портал, делаем деньги тем, что размещаем рекламу, надо разместить ссылки, по которым пользователи захотят переходить (и видеть ещё больше вкусной рекламы); Yahoo!, Google, Яндекс, большинство новостных сайтов.

Онлайн vs. оффлайн

- У рекомендательной системы есть два разных «уровня», на которых она должна работать:
  - глобальные оценки, медленно меняющиеся особенности и предпочтения, интересные страницы, зависимость от user features (география, пол etc.) и т.д.;
  - кратковременные тренды, hotness, быстрые изменения интереса во времени.

Онлайн vs. оффлайн

- Это очень разные задачи с разными методами, поэтому различают два класса моделей.
  - *Оффлайн-модели* выявляют глобальные закономерности (обычно это и называется коллаборативной фильтрацией). Цель зачастую в том, чтобы найти и рекомендовать человеку то, что ему понравится, из достаточно редких вещей, работать с «длинными хвостами» распределений интересов людей и веб-страниц.
  - *Онлайн-модели* должны реагировать очень быстро (поэтому там обычно подходы попроще, как правило, не индивидуализированные), они выявляют кратковременные тренды, позволяют рекомендовать то, что hot прямо сейчас.
- Мы будем говорить о коллаборативной фильтрации.

GroupLens

- Начнём краткий обзор разных рекомендательных систем с коллаборативной фильтрации.
- Обозначения:
  - индекс i всегда будет обозначать пользователей (всего пользователей будет N, $i = 1..N$);
  - индекс a – предметы (сайты, товары, фильмы...), которые мы рекомендуем (всего M, $a = 1..M$);
  - x_i – набор (вектор) признаков (features) пользователя, x_a – набор признаков предмета;
  - когда пользователь i оценивает предмет a, он производит отклик (response, rating) $r_{i,a}$; этот отклик – случайная величина, конечно.
- Наша задача – предсказывать оценки $r_{i,a}$, зная признаки x_i и x_a для всех элементов базы и зная некоторые уже расставленные в базе $r_{i',a'}$. Предсказание будем обозначать через $\hat{r}_{i,a}$.

GroupLens

- Начнём с небайесовских методов. Метод ближайших соседей: давайте введём расстояние между пользователями и будем рекомендовать то, что нравится вашим соседям.
- Расстояние можно считать просто как коэффициент корреляции (коэффициент Пирсона)

$$w_{i,j} = \frac{\sum_a (r_{i,a} - \bar{r}_a)(r_{j,a} - \bar{r}_a)}{\sqrt{\sum_a (r_{i,a} - \bar{r}_a)^2}\sqrt{\sum_a (r_{j,a} - \bar{r}_a)^2}},$$

где $\bar{r}_a$ – средний рейтинг продукта a среди всех пользователей.

GroupLens

- Простейший способ построить предсказание нового рейтинга $\hat{r}_{i,a}$ – сумма рейтингов других пользователей, взвешенная их похожестью на пользователя i:

$$\hat{r}_{i,a} = \bar{r}_a + \frac{\sum_j (r_{j,a} - \bar{r}_j)\, w_{i,j}}{\sum_j |w_{i,j}|}.$$

- Этот алгоритм называется GroupLens; так работает дедушка рекомендательных систем, сайт MovieLens.
- Чтобы не суммировать по всем пользователям, можно ограничиться ближайшими соседями:

$$\hat{r}_{i,a} = \bar{r}_a + \frac{\sum_{j \in \mathrm{kNN}(i)} (r_{j,a} - \bar{r}_j)\, w_{i,j}}{\sum_{j \in \mathrm{kNN}(i)} |w_{i,j}|}.$$

Item-item CF

- Симметричный подход – item-based collaborative filtering. Считаем похожесть между продуктами, выбираем похожие продукты.
- Amazon: customers who bought this item also bought...
- Преимущество – может быть эффективнее за счёт того, что похожесть продуктов всегда можно считать оффлайн, пара новых оценок не повлияет на неё совсем радикально.
- Считаем похожесть между парами продуктов, у которых есть общий оценивший пользователь.
- Есть масса разных модификаций, большое направление исследований, но нам уже пора обратиться к вероятностным моделям.

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:

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

Вероятностные модели

- Чтобы найти предикторы, нужно просто приблизить базу рейтингов этой моделью; например, можно добавить нормально распределённый шум и получить модель линейной регрессии

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

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

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

Вероятностные модели

- С тем, чтобы напрямую обучать оставшуюся матрицу предпочтений вероятностными методами, есть одна очень серьёзная 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.

Вероятностные модели

- 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.

Вероятностные модели

- 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:

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

Вероятностные модели

- Можно добавлять и дополнительную информацию в эту модель. Например, введём дополнительный набор факторов для продуктов y_a, которые будут характеризовать пользователя на основе того, что он просматривал, но не оценивал.
- Модель после этого принимает вид

$$\hat{r}_{i,a} = \mu + b_i + b_a + q_a^\top \left(p_i + \frac{1}{\sqrt{|V(i)|}} \sum_{b \in V(i)} y_b \right),$$

где $V(i)$ – множество продуктов, которые просматривал этот пользователь ($\frac{1}{\sqrt{|V(i)|}}$ контролирует дисперсию).

- Это называется SVD++.

Вероятностное разложение матриц

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

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

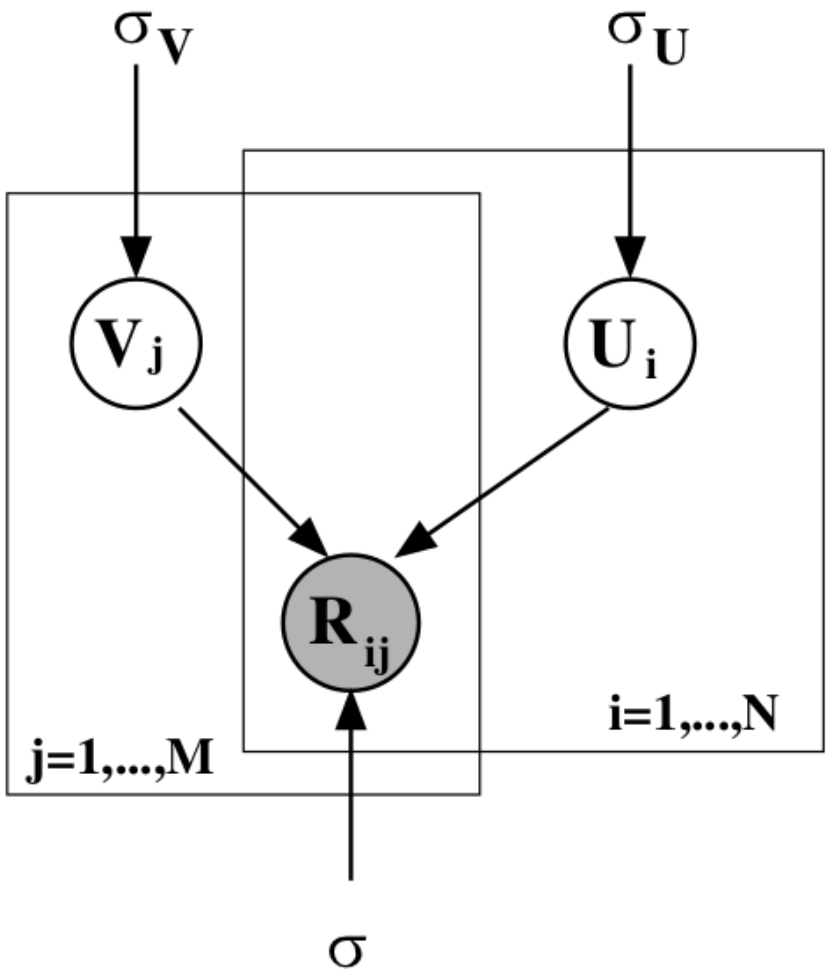
- Вероятностно мы имеем правдоподобие

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

- Добавим гауссовские априорные распределения на U и V:

$$p(U \mid \sigma_U^2) = \prod_i \mathcal{N}(U_i \mid \mathbf{0}, \sigma_U^2 \boldsymbol{I}), \quad p(V \mid \sigma_V^2) = \prod_a \mathcal{N}(V_a \mid \mathbf{0}, \sigma_V^2 \boldsymbol{I}).$$

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 $\boldsymbol{\sigma} = (\sigma^2, \sigma_V^2, \sigma_U^2)$, максимизируя общее правдоподобие модели:

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

EM-алгоритмом:

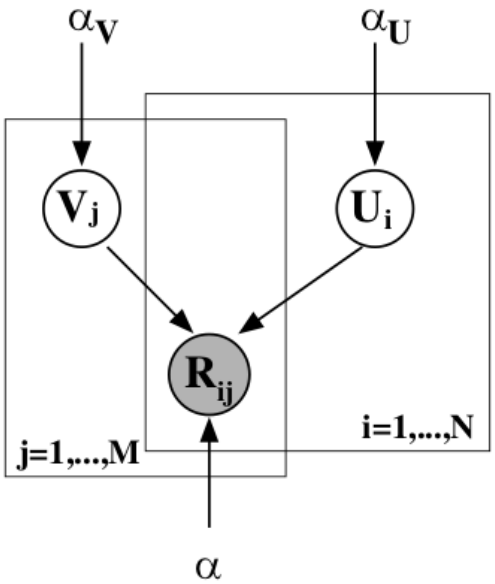
- сначала зафиксируем $\boldsymbol{\sigma}$ и найдём

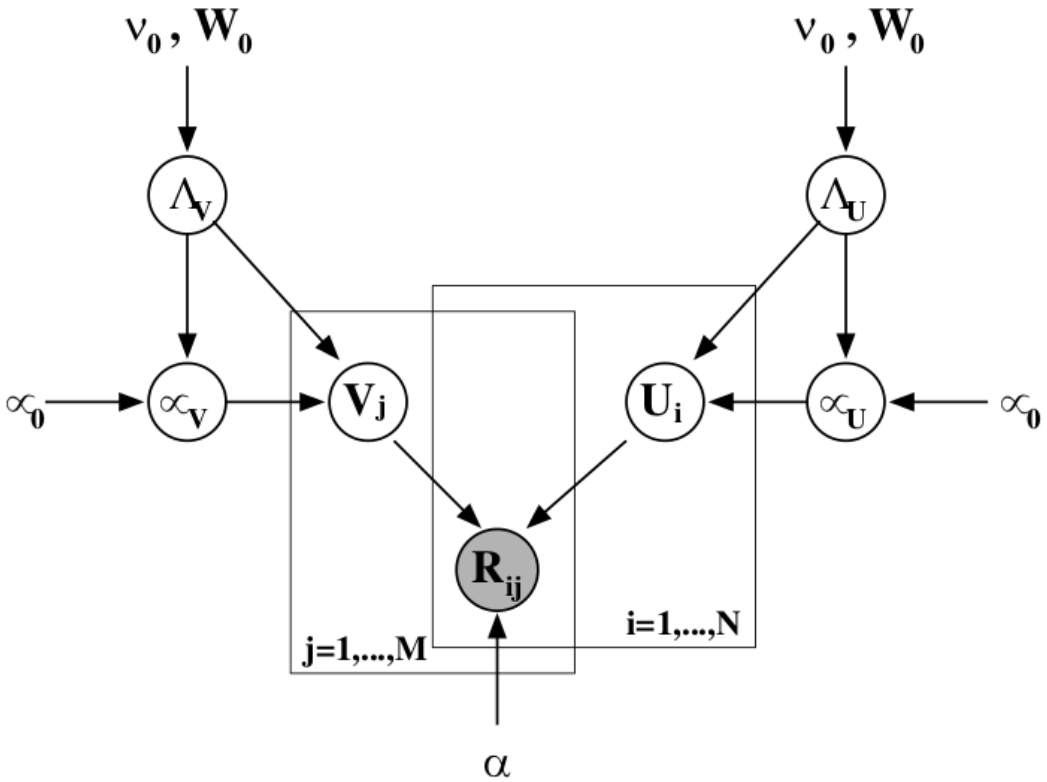
$$f(\boldsymbol{\sigma}) = \mathbb{E}_{U,V|R,\boldsymbol{\sigma}} \left[\log p(R, U, V \mid \boldsymbol{\sigma})\right];$$

- потом максимизируем

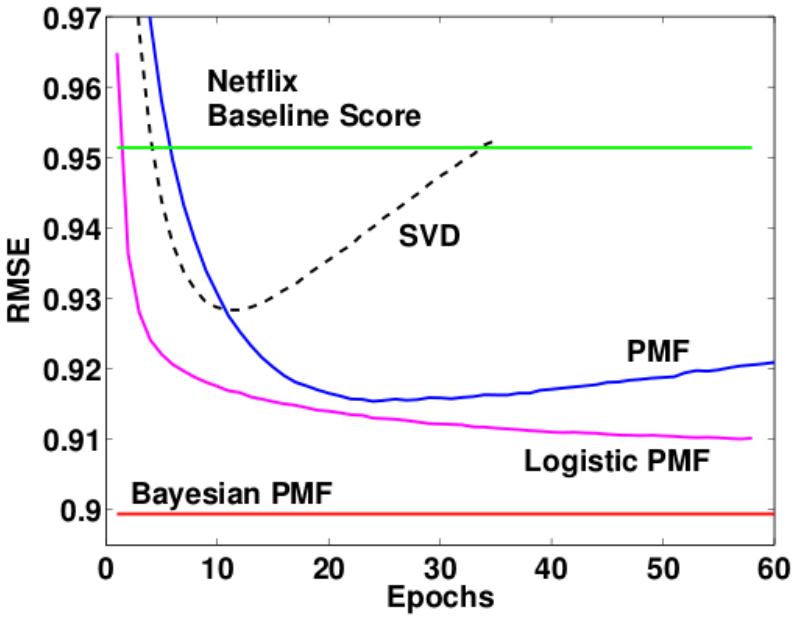
$$\boldsymbol{\sigma} := \arg\max_{\boldsymbol{\sigma}} f(\boldsymbol{\sigma}).$$

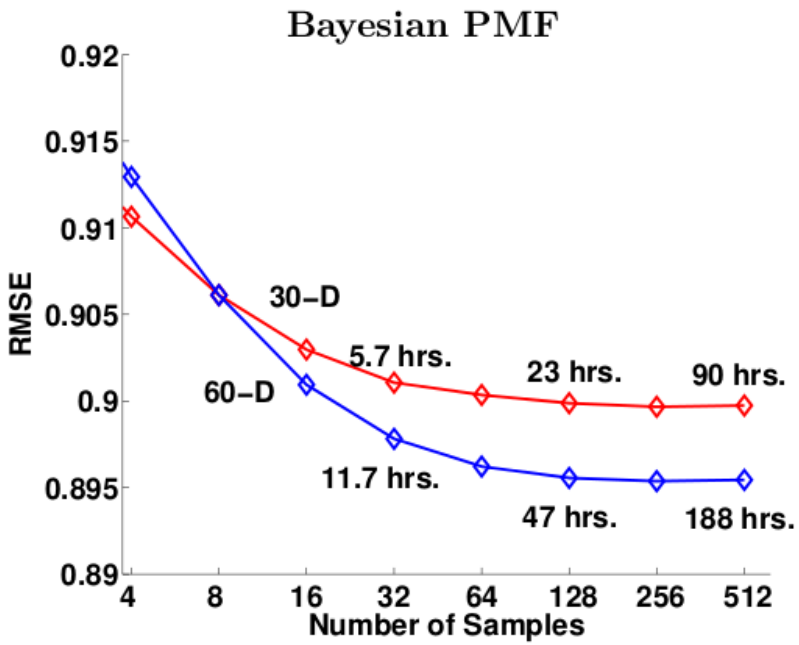
Графическая модель





Графическая модель



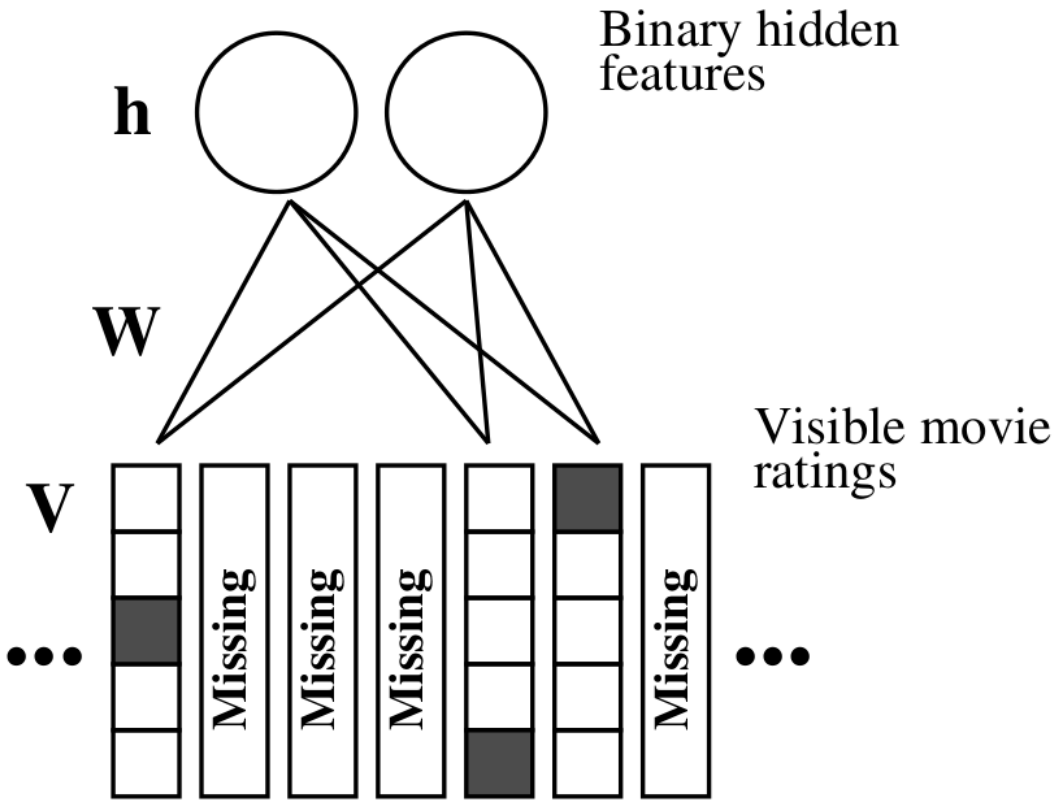


Машины Больцмана

- Ещё один метод вероятностного моделирования – *машины Больцмана* (restricted Boltzmann machine).
- Ненаправленная графическая модель, состоящая из двух уровней, видимого и скрытого.
- Машины Больцмана – основа для deep learning, одного из наших текущих представлений о том, как устроен мозг.

Машины Больцмана

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



Машины Больцмана

- В результате на скрытых нейронах 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.

Комбинация моделей

- 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 – всегда большие *ансамбли* моделей. Два уровня: обучение отдельных моделей в ансамбле (bootstrapping и т.п.) и обучение комбинации.
- Однако до абсурда, как в итоге получилось в Netflix Prize, доводить на практике не обязательно.

Outline

Регрессия по признакам

- 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.

Регрессия по признакам

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

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

где

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

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

Регрессия по признакам

- 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

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

  что помогает для холодного старта по продуктам.
- Есть модели, связанные с тем, как обучить не абы какие темы, а хорошо выражающие предпочтения.

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

$$\begin{aligned}
b_a(t) &= b_a + b_{a,\mathrm{Bin}(t)}, \\
b_i(t) &= b_i + \alpha_i \mathrm{dev}_i(t) + b_{i,t}, \\
p_{i,f}(t) &= p_{i,f} + \alpha_{i,f}\mathrm{dev}_i(t) + p_{i,f,t} + \frac{1}{\sqrt{|V(i)|}} \sum_{b \in V(i)} y_b, \\
\mathrm{dev}_i(t) &= \mathrm{sign}(t - t_i)\,|t - t_i|^\beta .
\end{aligned}$$

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

Социальные сети

- 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.

Метрики разнообразия

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

Контекстно-зависимые рекомендации

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

Контекстно-зависимые рекомендации

- 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.

Summary

1. Метод ближайших соседей: GroupLens.
2. SVD-разложение матриц градиентным спуском, его байесовская версия (PMF).
3. Машины Больцмана – модель пользователя.
4. Холодный старт.
5. Дополнительная информация: время, контекст etc.

Thank you!

Спасибо за внимание!