

# Рекомендательные системы

Сергей Николенко

Computer Science Club, Екатеринбург, 2011

# Outline

## 1 Коллаборативная фильтрация

- Основные методы и модели
- PMF

## 2 Онлайн-модели

- Постановка задачи и бандиты
- DGP

# 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.

# 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:
  - 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;

- косинус угла между векторами рейтингов, выставленных  $i$  и  $j$ , т.е.

$$w_{i,j} = \frac{\sum_a r_{i,a} r_{j,a}}{\sqrt{\sum_a r_{i,a}^2} \sqrt{\sum_a r_{j,a}^2}}.$$

# GroupLens

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

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

- Это называется GroupLens algorithm – так работал дедушка рекомендательных систем GroupLens.
- Чтобы не суммировать по всем пользователям, можно ограничиться ближайшими соседями:

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



# GroupLens

- Естественно предположить, что продукты, которые любят или не любят практически все пользователи, не слишком полезны в определении ближайшего соседа.
- Поэтому естественно взвесить продукты по тому, как часто их уже оценивали пользователи; такая метрика называется iuf – inverse user frequency, обратная частота пользователей:  $f_a = \log \frac{N}{N_a}$ , где  $N$  – общее число пользователей,  $N_a$  – число оценивших продукт  $a$ . Получается

$$w_{i,j}^{\text{idf}} = \frac{\sum_a f_a \sum_a f_a r_{i,a} r_{j,a} - (\sum_a f_a r_{i,a}) (\sum_a f_a r_{j,a})}{\sqrt{\sum_a f_a (\sum_a f_a r_{i,a}^2 - (\sum_a f_a r_{i,a})^2)} \sqrt{\sum_a f_a (\sum_a f_a r_{j,a}^2 - (\sum_a f_a r_{j,a})^2)}},$$

а для косинуса

$$w_{i,j}^{\text{idf}} = \frac{\sum_a f_a^2 r_{i,a} r_{j,a}}{\sqrt{\sum_a (f_a r_{i,a})^2} \sqrt{\sum_a (f_a r_{j,a})^2}}.$$

# 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  $\mu$ :

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

# Вероятностные модели

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

$$r_{i,a} \sim \mu + b_i + b_a.$$

- 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 – вектором  $p_i \in \mathbb{R}^f$ , 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++, и эта модель была одним из основных компонентов модели, взявшей Netflix Prize.

# Вероятностное разложение матриц

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

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

- Вероятностно мы имеем правдоподобие

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

- Добавим гауссовские априорные распределения на  $U$  и  $V$ :

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



# Вероятностное разложение матриц

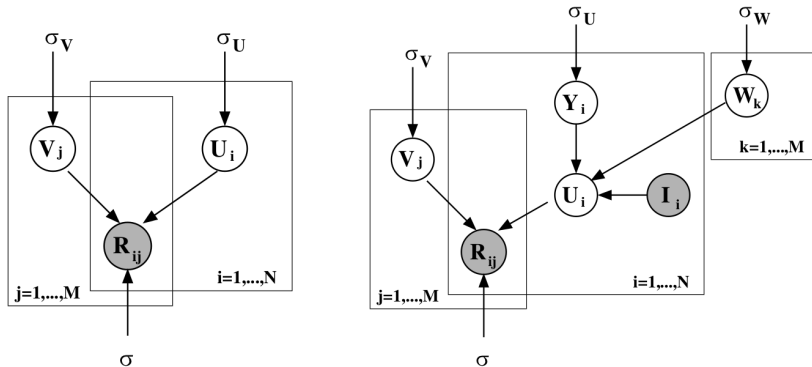
- Модификация: пользователи с малым числом оценок в PMF получают апостериорные распределения, очень похожие на «среднего пользователя».
- Чтобы на редких пользователей лучше обобщалось, добавим ещё факторы, которые меняют априорное распределение факторов у пользователя в зависимости от того, сколько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

$$p(W | \sigma_W^2) = \prod \mathcal{N}(W_i | \mathbf{0}, \sigma_W^2 I).$$

# Графическая модель



# Outline

## 1 Коллаборативная фильтрация

- Основные методы и модели
- PMF

## 2 Онлайн-модели

- Постановка задачи и бандиты
- DGP

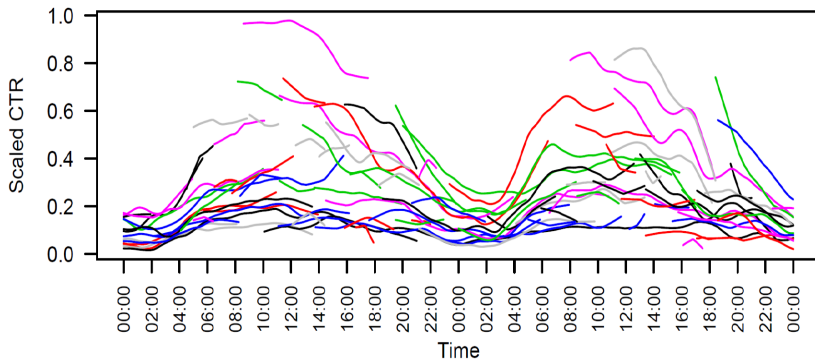
# Постановка задачи

- *Онлайн-модели* отличаются от оффлайн-моделей тем, что их главная цель – как можно быстрее «поймать» изменения популярности тех или иных продуктов.
- Данных тут недостаточно, чтобы такие изменения можно было поймать методами коллаборативной фильтрации.
- Поэтому онлайн-методы обычно не персонализированы.

# Постановка задачи

- Казалось бы, что может быть проще – есть набор продуктов/сайтов  $a_1, \dots, a_M$  со средними рейтингами  $\bar{r}_1, \dots, \bar{r}_M$ ; давайте упорядочим их по среднему рейтингу и будем рекомендовать пользователю продукты с наивысшим средним рейтингом.
- Однако такая система не будет достаточно чувствительной к быстрым изменениям истинного среднего рейтинга: представьте, что  $\bar{r}_i$  внезапно резко уменьшился – может понадобиться очень много новых показов, чтобы привести нашу его оценку в соответствие с новым значением.

# Пример



# Рекомендатели и бандиты

- Если мы хотим быстро оценивать средний рейтинг, то мы попадём в ситуацию обучения с подкреплением: есть набор продуктов, их надо рекомендовать, исход заранее неизвестен, и мы хотим оптимизировать суммарный рейтинг.

# Рекомендатели и бандиты

- Кратко пройдемся по основным стратегиям:
  - $\epsilon$ -жадная стратегия (вариант:  $\epsilon$ -начальная стратегия);
  - $\epsilon$ -убывающая стратегия: сделать так, чтобы  $\epsilon$  убывало со временем;
  - *softmax-стратегии*: выбираем ручки с вероятностями, связанными с уже успевшей накопиться информацией; вероятность  $p_k$  дёрнуть за ручку  $k$  равна

$$p_k = \frac{e^{\hat{\mu}_k / \tau}}{\sum_{i=1}^n e^{\hat{\mu}_i / \tau}},$$

где  $\hat{\mu}_k$  – наша текущая оценка среднего  $\mu_k$ , а  $\tau$  – параметр стратегии, называющийся *температурой* (это больцмановское распределение из статистической физики); температуру обычно постепенно понижают;



# Рекомендатели и бандиты

- Кратко пройдемся по основным стратегиям:
  - стратегия EXP3: если мы получили неожиданный результат – выбрали ручку с маленькой вероятностью, но получили при этом большой доход – стоит попробовать исследовать эту ручку дальше; вероятность выбрать на шаге  $t$  ручку  $k$  равна

$$p_k(t) = (1 - \gamma) \frac{w_k(t)}{\sum_{i=1}^n w_i(t)} + \frac{\gamma}{n}, \text{ где}$$

$$w_j(t+1) = w_j(t) e^{\gamma \frac{r_j(t)}{p_j(t)n}},$$

если ручку  $j$  дёрнули на шаге  $t$  с наблюдаемой наградой  $r_j(t)$ .

# Рекомендатели и бандиты

- Кратко пройдемся по основным стратегиям:
  - *Стратегия* **INTeSTIM**. Совершенно другой подход, более «честный» вероятно – подсчитывать для каждого автомата доверительный интервал с верхней границей  $(1 - \alpha)$  (в некоторых предположениях, конечно), где  $\alpha$  – параметр стратегии, а затем выбирать автомат, у которого максимальна *верхняя* граница доверительного интервала. Такой интервал легко подсчитать для (нашего) булевого случая испытаний Бернулли, когда награда фактически равна либо 0, либо 1 (понравилось или нет), он равен  $\left[ \bar{p} - z_\alpha \sqrt{\frac{\bar{p}(1-\bar{p})}{n}}, \bar{p} + z_\alpha \sqrt{\frac{\bar{p}(1-\bar{p})}{n}} \right]$ , где  $\bar{p} = \frac{\sum r_i}{n}$  – текущее среднее, а  $z_\alpha$  берётся из таблиц (специальных функций); например, для  $\alpha = 0.05$  будет  $z_\alpha = 1.96$ .

# Рекомендатели и бандиты

- Кратко пройдемся по основным стратегиям:
  - *Стратегия UCB1*. Учитывает неопределённость, «оставшуюся» в той или иной ручке, старается ограничить regret. Если мы из  $n$  экспериментов  $n_i$  раз дёрнули за  $i$ -ю ручку и получили среднюю награду  $\hat{\mu}_i$ , алгоритм UCB1 присваивает ей приоритет

$$\text{Priority}_i = \hat{\mu}_i + \sqrt{\frac{2 \log n}{n_i}}.$$

Дёргать дальше надо за ручку с наивысшим приоритетом.

# DGP

- Но это просто оценка статической ситуации, а мы помним, что надо двигаться быстро.
- Модель Dynamic Gamma–Poisson (DGP): фиксируем период времени  $t$  (небольшой) и будем считать показы и клики (рейтинги, отметки «like» и т.д.) за время  $t$ .
- Пусть мы в течение периода  $t$  показали продукт  $n_t$  раз и получили суммарный рейтинг  $r_t$  (если это ссылки на странице, например, то будет суммарное число кликов  $r_t \leq n_t$ ).
- Тогда нам в каждый момент  $t$  дана последовательность  $n_1, r_1, n_2, r_2, \dots, n_t, r_t$ , и мы хотим предсказать  $p_{t+1}$  (доля успешных показов в момент  $t + 1$ , CTR).

## DGP

- Вероятностные предположения модели DGP:
  - 1  $(r_t \mid n_t, p_t) \sim \text{Poisson}(n_t, p_t)$  (для данного  $n_t$  и  $p_t$ ,  $r_t$  распределено по пуассоновскому распределению).
  - 2  $p_t = \epsilon_t p_{t-1}$ , где  $\epsilon_t \sim \text{Gamma}(\mu = 1, \sigma = \eta)$  (средняя доля успешных показов  $p_t$  меняется не слишком быстро, а путём умножения на случайную величину  $\epsilon_t$ , которая имеет гамма-распределение вокруг единицы).
  - 3 Параметрами модели являются параметры распределения  $p_1 \sim \text{Gamma}(\mu = \mu_0, \sigma = \sigma_0)$ , а также параметр  $\eta$ , который показывает, насколько «гладко» может изменяться  $p_t$ .
  - 4 Соответственно, задача заключается в том, чтобы оценить параметры апостериорного распределения

$$(p_{t+1} \mid n_1, r_1, n_2, r_2, \dots, n_t, r_t) \sim \text{Gamma}(\mu = ?, \sigma = ?).$$

## DGP

- Можно пересчёт параметров в этой модели явно вычислить аналитически.
- Пусть на предыдущем шаге  $t - 1$  мы получили некоторую оценку  $\mu_t, \sigma_t$  для параметров модели:

$$(p_t \mid n_1, r_1, n_2, r_2, \dots, n_{t-1}, r_{t-1}) \sim \text{Gamma}(\mu = \mu_t, \sigma = \sigma_t),$$

а затем получили новую точку  $(n_t, r_t)$ .

- Тогда, обозначив  $\gamma_t = \frac{\mu_t}{\sigma_t^2}$  (эффективный размер выборки), сначала уточним оценки  $\mu_t, \sigma_t$ :

$$\gamma_{t|t} = \gamma_t + n_t,$$

$$\mu_{t|t} = \frac{\mu_t \gamma_t + r_t}{\gamma_{t|t}},$$

$$\sigma_{t|t}^2 = \frac{\mu_{t|t}}{\gamma_{t|t}}.$$

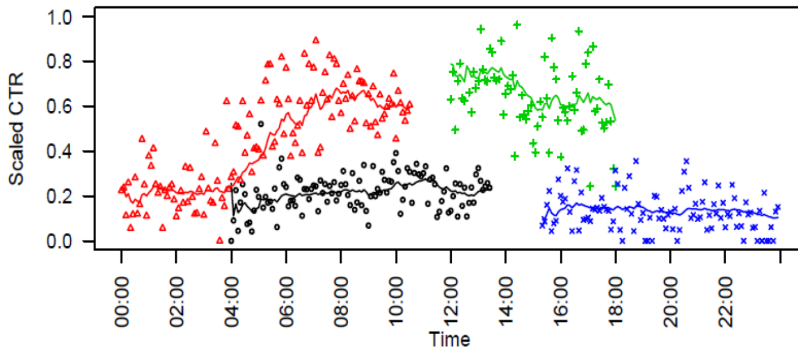
## DGP

- А затем породим новое предсказание для  $(p_{t+1} \mid n_1, r_1, \dots, n_t, r_t)$ :

$$\mu_{t+1} = \mu_{t|t},$$

$$\sigma_{t+1}^2 = \sigma_{t|t}^2 + \eta \left( \mu_{t|t}^2 + \sigma_{t|t}^2 \right).$$

# Пример





# Априорное распределение

- Тут интересный вопрос – чем инициализировать; поскольку надо всё делать быстро, хорошее априорное распределение очень важно.
- Пусть в тестовой выборке  $N$  записей, для которых известны показатели  $r_1^{(i)}$  и  $n_1^{(i)}$  (число показов и успешных показов за первый период времени); мы хотим получить оценку  $\mu_0$  и  $\sigma_0$  для нового, неизвестного сайта, которая должна хорошо аппроксимировать ожидаемое  $r_1$  и  $n_1$  для нового сайта.
- Тогда ответ такой – её нужно считать как

$$\arg \max_{\mu_0, \sigma_0} \left[ N \frac{\mu_0^2}{\sigma_0^2} \log \frac{\mu_0}{\sigma_0^2} - N \log \text{Gamma} \left( \frac{\mu_0^2}{\sigma_0^2} \right) + \sum_i \left( \log \text{Gamma} \left[ r_1^{(i)} + \frac{\mu_0^2}{\sigma_0^2} \right] - \left[ r_1^{(i)} + \frac{\mu_0^2}{\sigma_0^2} \right] \log \left[ n_1^{(i)} + \frac{\mu_0}{\sigma_0^2} \right] \right) \right].$$

Thank you!

**Спасибо за внимание!**