

3.36pt

# РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ III

---

Сергей Николенко

СПбГУ — Санкт-Петербург

8 декабря 2017 г.

---

*Random facts:*

- 8 декабря --- День студента в Болгарии и Македонии
- 8 декабря 1854 г. папа Пий IX издал буллу *Ineffabilis Deus* о непорочном зачатии, в результате чего 8 декабря стало одним из центральных католических праздников
- 8 декабря 1987 г. Рон Хекстолл из «Philadelphia Flyers» стал первым вратарем в истории NHL, который забил гол

# 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  $\text{IDCG}_k$  – значение  $\text{DCG}_k$  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 (Top10 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 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.

- 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.

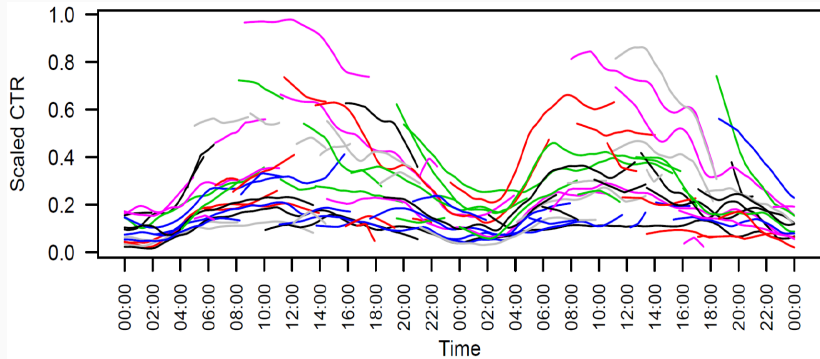
## Онлайн-модели

---

- *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 меньше персонализированы, индивидуальных данных не наберётся просто.

## 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.



## 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.
- Это называется задача о *многоруких бандитах* (multiarmed bandits).

- 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;

- 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  $r_j(t)$ .



- Кратко пройдёмся 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  $z_\alpha = 1.96$ .

- Кратко пройдёмся 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.

- 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).

- Вероятностные предположения модели DGP:

(1)  $(r_t | n_t, p_t) \sim \text{Poisson}(n_t, p_t)$

(для данных  $n_t$  и  $p_t$   $r_t$  имеет пуассоновское распределение);

(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) задача — оценить параметры апостериорного распределения

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

- 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  $(n_t, r_t)$ .

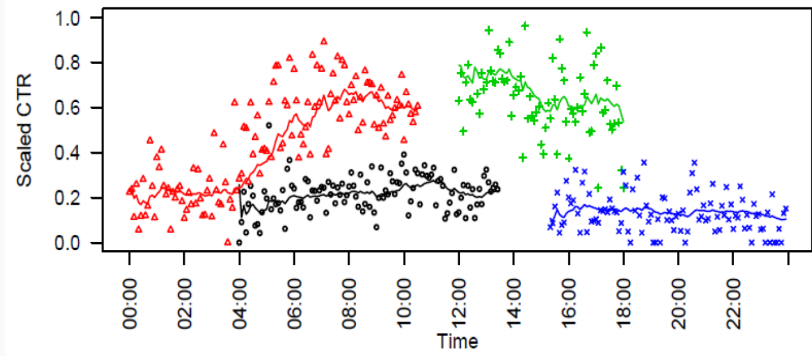
- Тогда, обозначив  $\gamma_t = \frac{\mu_t}{\sigma_t^2}$  (эффективный размер выборки), сначала уточним оценки  $\mu_t, \sigma_t$ :

$$\gamma_{t|t} = \gamma_t + n_t, \quad \mu_{t|t} = \frac{\mu_t \gamma_t + r_t}{\gamma_{t|t}}, \quad \sigma_{t|t}^2 = \frac{\mu_{t|t}}{\gamma_{t|t}}.$$

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

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

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



- 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

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



- Здесь тоже масса активно развивающихся направлений.
  1. Как совместно оптимизировать сразу много показываемых элементов? Представьте себе homepage большого портала.
  2. Какова на самом деле целевая метрика? CTR – это средство, а не цель. Для портала – user retention? проведённое на портале время? доход рекламодателей?
  3. Как именно персонализировать – кластеризовать пользователей? как именно сглаживать? каковы признаки пользователей?

1. Метод ближайших соседей: GroupLens. User-based и item-based.
2. SVD-разложение матриц. Градиентный спуск и ALS. Байесовская версия (PMF).
3. Машины Больцмана – модель пользователя. Другие разложения: NMF.
4. Метрики качества: NDCG, AUC, Top-N.
5. Холодный старт и дополнительная информация: время, контекст etc.
6. Онлайн-системы (поиск трендов): DGP.

Спасибо за внимание!