РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ II

Сергей Николенко

СПбГУ — Санкт-Петербург
1 декабря 2017 г.

Random facts:

- 1 декабря 1995 г. в Великом Новгороде открыто троллейбусное сообщение, а во Внутренней Монголии --- Цзитунская железная дорога, на которой 100% перевозок осуществляли паровозы
- с 1 декабря 2017 г. российские автовладельцы с шипованной резиной обязаны размещать знак «Ш», российские мясники вынуждены уменьшить закупки говядины и свинины из Бразилии из-за запрещенного в России рактопамина, а российские медицинские учреждения обязаны получать отдельную лицензию на проведение абортов

Матричные разложения и вероятностные модели

Вероятностные модели: базовые предикторы

- 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: базовые предикторы

- Чтобы найти предикторы, уже нужен байесовский подход: надо 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:

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

- Как обучить такую модель?

Вероятностные модели: базовые предикторы

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

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

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

Вероятностные модели: базовые предикторы

- 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.

Вероятностные модели: SVD

- 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.

ВЕРОЯТНОСТНЫЕ МОДЕЛИ: SVD

- 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: SVD

- 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?

Вероятностные модели: SVD

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

$$
\begin{aligned}
b_i &:= b_i + \gamma \left(e_{i,a} - \lambda_2 b_i \right), \\
b_a &:= b_a + \gamma \left(e_{i,a} - \lambda_2 b_a \right), \\
q_{a,j} &:= q_{a,j} + \gamma \left(e_{i,a} p_{i,j} - \lambda_2 q_{i,j} \right) \text{ для всех } j, \\
p_{i,j} &:= p_{i,j} + \gamma \left(e_{i,a} q_{a,j} - \lambda_2 p_{i,j} \right) \text{ для всех } j,
\end{aligned}
$$

где γ – скорость обучения.

Вероятностные модели: SVD

- 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.

Вероятностные модели: SVD

- 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.

ВЕРОЯТНОСТНОЕ РАЗЛОЖЕНИЕ МАТРИЦ

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

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

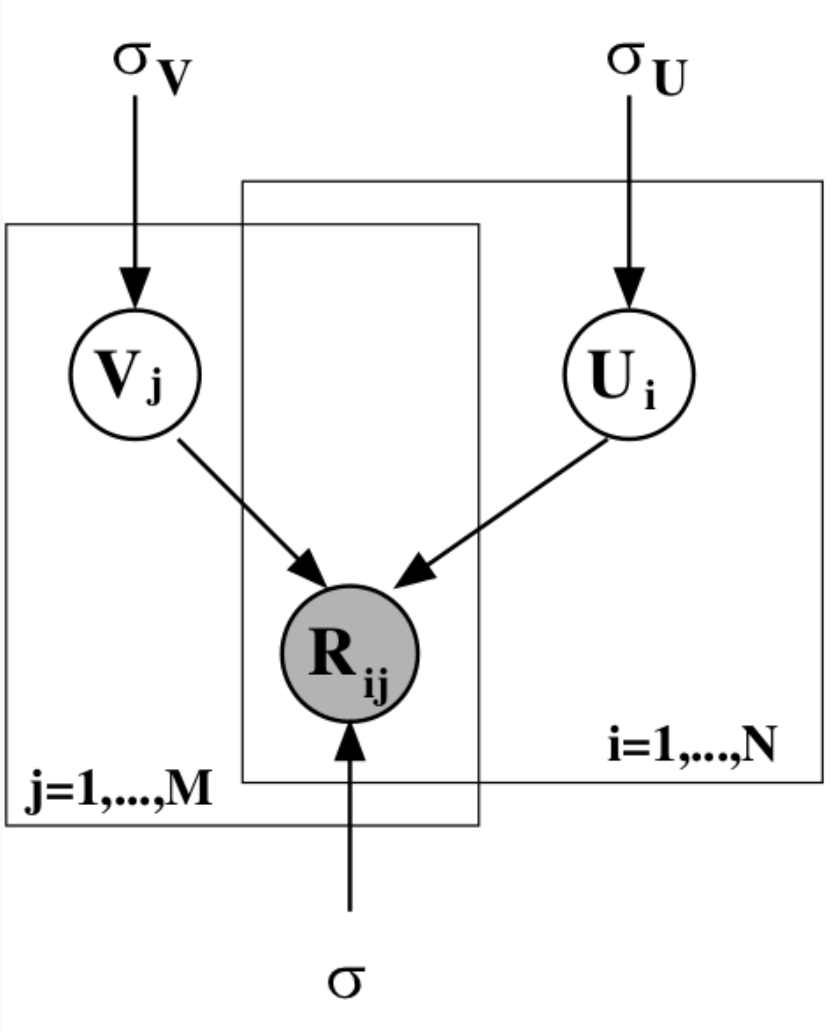
- Вероятностно мы имеем правдоподобие

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

- Добавим гауссовские априорные распределения на U и V:

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

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

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

EM-алгоритмом:

- сначала зафиксируем σ и найдём

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

- потом максимизируем

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

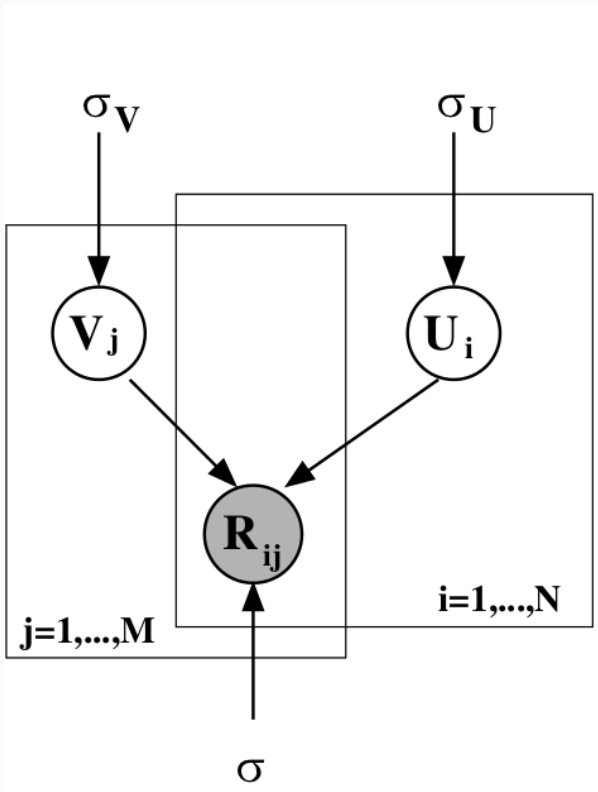
ВЕРОЯТНОСТНОЕ РАЗЛОЖЕНИЕ МАТРИЦ

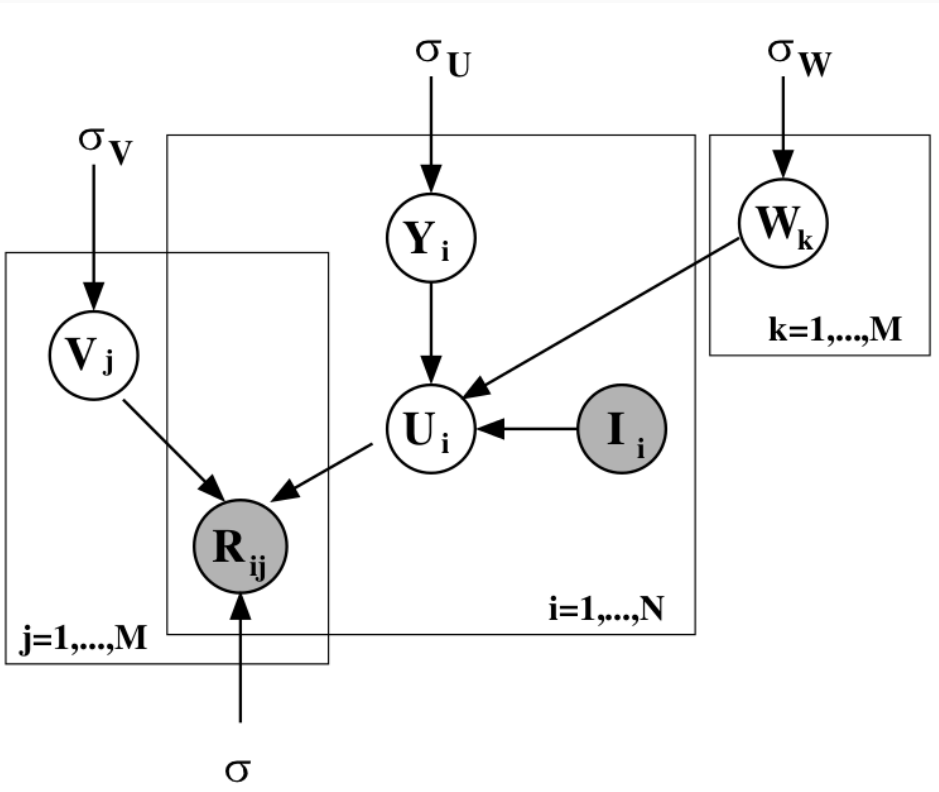
- Модификация: пользователи с малым числом оценок в PMF получат апостериорные распределения, очень похожие на «среднего пользователя».
- Чтобы на редких пользователей лучше обобщалось, добавим ещё факторы, которые меняют априорное распределение факторов у пользователя в зависимости от того, сколько и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

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

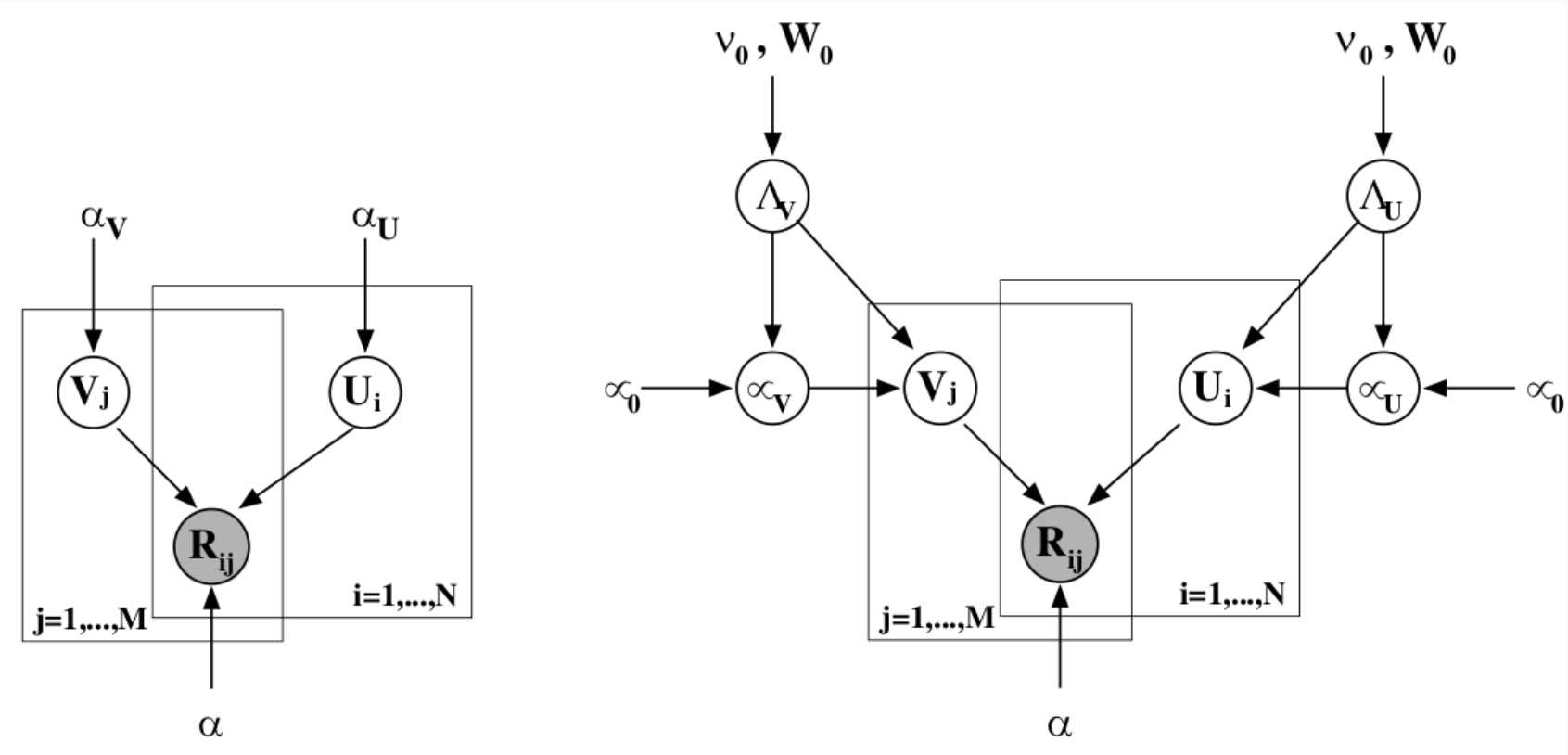
- Матрица W показывает, как влияет на априорное распределение
- Тоже в качестве регуляризатора берём априорный гауссиан: $p(W \mid \sigma_W^2) = \prod_i \mathcal{N}(W_i \mid 0, \sigma_W^2 I)$.

ГРАФИЧЕСКАЯ МОДЕЛЬ

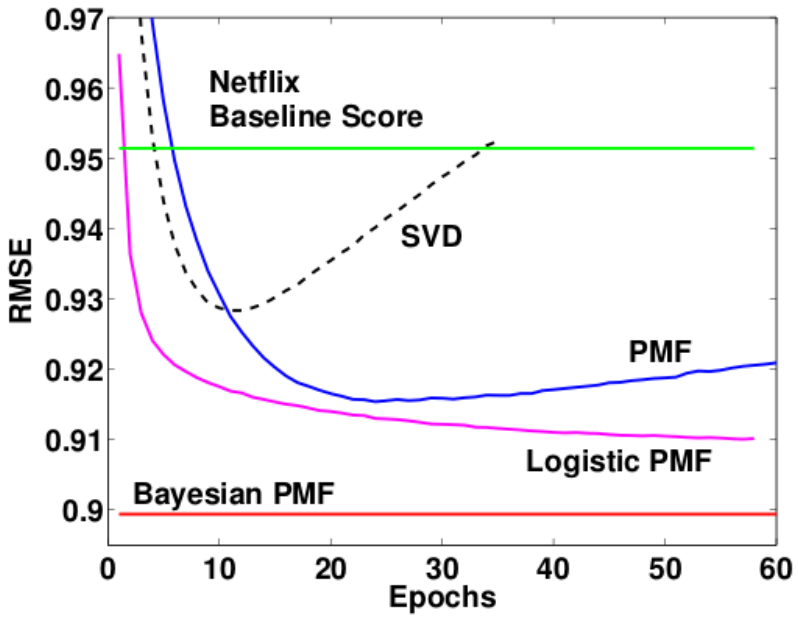


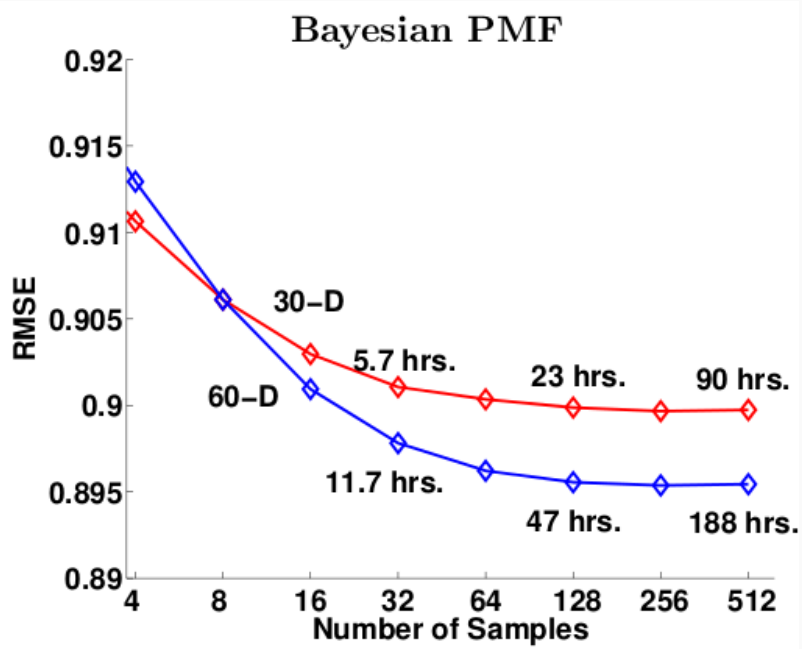


ГРАФИЧЕСКАЯ МОДЕЛЬ



ГРАФИЧЕСКАЯ МОДЕЛЬ



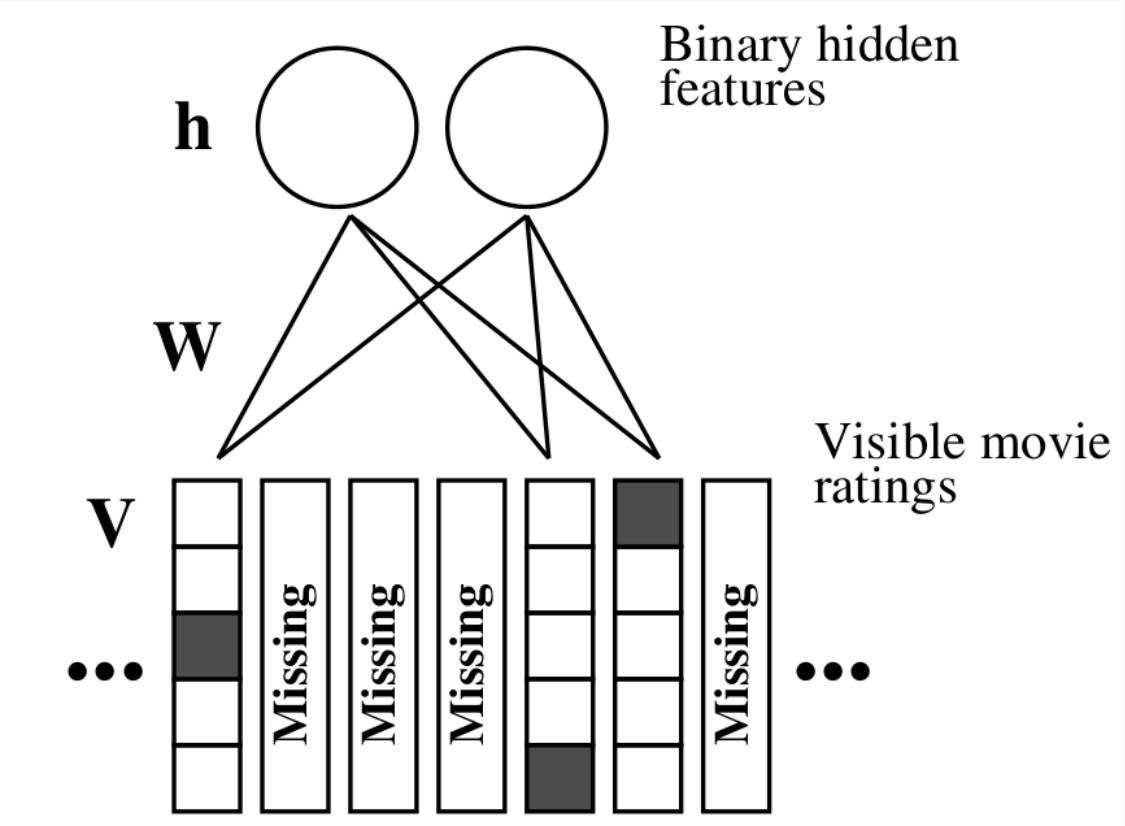


Машины Больцмана

- Ещё один метод вероятностного моделирования – *машины Больцмана* (restricted Boltzmann machine).
- Ненаправленная графическая модель, состоящая из двух уровней, видимого и скрытого.
- Машины Больцмана – 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.



Машины Больцмана

- 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.

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 – обычно большие *ансамбли* моделей.
- Два уровня: обучение отдельных моделей в ансамбле (bootstrapping и т.п.) и обучение комбинации.

Матричные разложения

- 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.

Матричные разложения

- 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.

МАТРИЧНЫЕ РАЗЛОЖЕНИЯ

- 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...

МАТРИЧНЫЕ РАЗЛОЖЕНИЯ

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

$$X \approx UV^\top,$$

  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 для рекомендаций

- 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

Метрики качества

- 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?

Метрики качества

- 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).

Метрики качества

- 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

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

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

$$\mathrm{NDCG}_k = \frac{\mathrm{DCG}_k}{\mathrm{IDCG}_k},$$

    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

- 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

- 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).

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

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

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 (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

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

- 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.

ОНЛАЙН-МОДЕЛИ

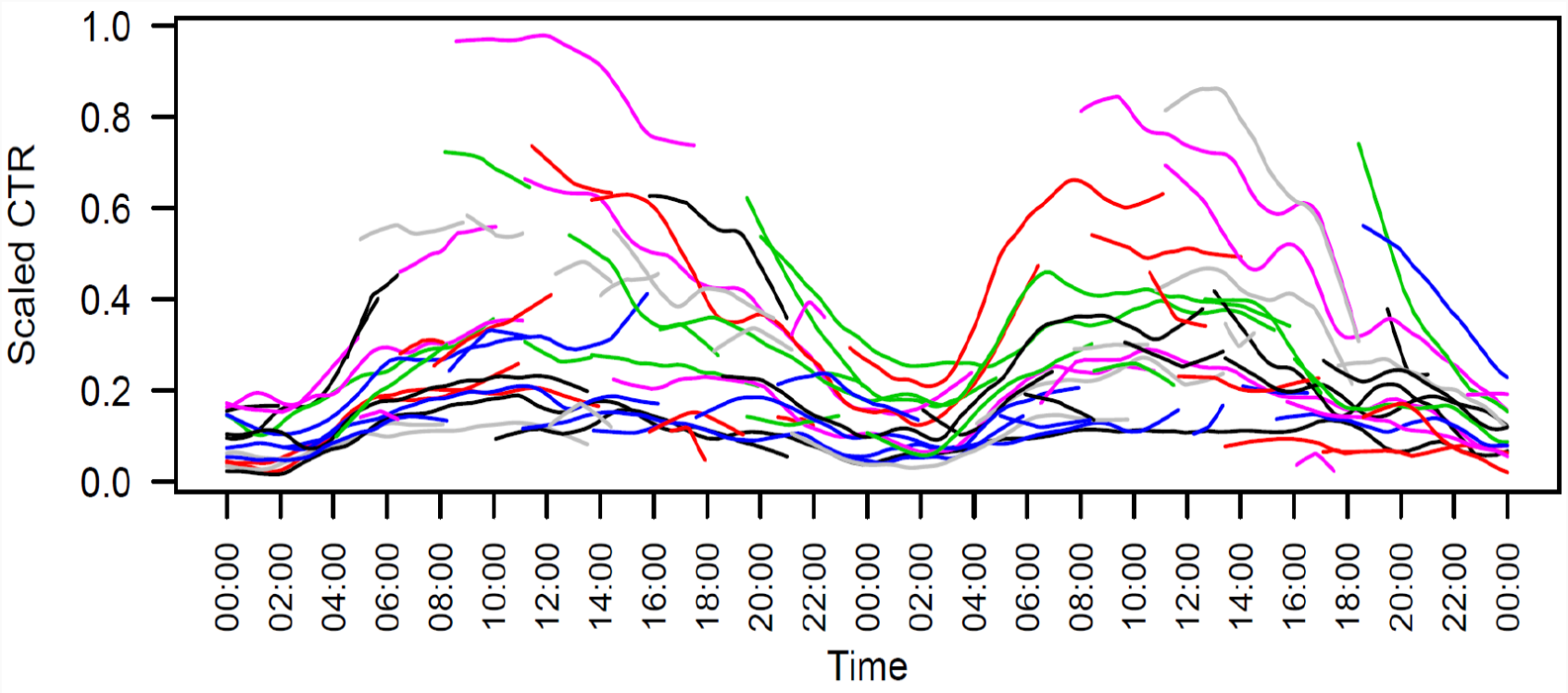
ПОСТАНОВКА ЗАДАЧИ

- *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 меньше персонализированы, индивидуальных данных не набёрется 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.

ПРИМЕР


Scaled CTR
1.0
0.8
0.6
0.4
0.2
0.0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
00:00
Time

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

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

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

- Кратко пройдёмся 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 дёргали на шаге t с наблюдаемой наградой $r_j(t)$.

- Кратко пройдёмся по основным стратегиям:
  - *Стратегия* IntEstim. Совершенно другой подход, более «честный» вероятностно – подсчитывать для каждого автомата доверительный интервал с верхней границей $(1-\alpha)$ (в некоторых предположениях, конечно), где α – 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_α 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

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

    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) $(r_t \mid n_t, p_t) \sim \mathrm{Poisson}(n_t, p_t)$
  (для данных n_t и p_t r_t имеет пуассоновское распределение);
  (2) $p_t = \epsilon_t p_{t-1}$, где $\epsilon_t \sim \mathrm{Gamma}(\mu = 1, \sigma = \eta)$
  (средняя доля успешных показов p_t меняется не слишком быстро, а путём умножения на случайную величину ϵ_t, которая имеет гамма-распределение вокруг единицы);
  (3) параметрами модели являются параметры распределения $p_1 \sim \mathrm{Gamma}(\mu = \mu_0, \sigma = \sigma_0)$, а также параметр η, который показывает, насколько «гладко» может изменяться p_t;
  (4) задача — оценить параметры апостериорного распределения

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

DGP

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

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

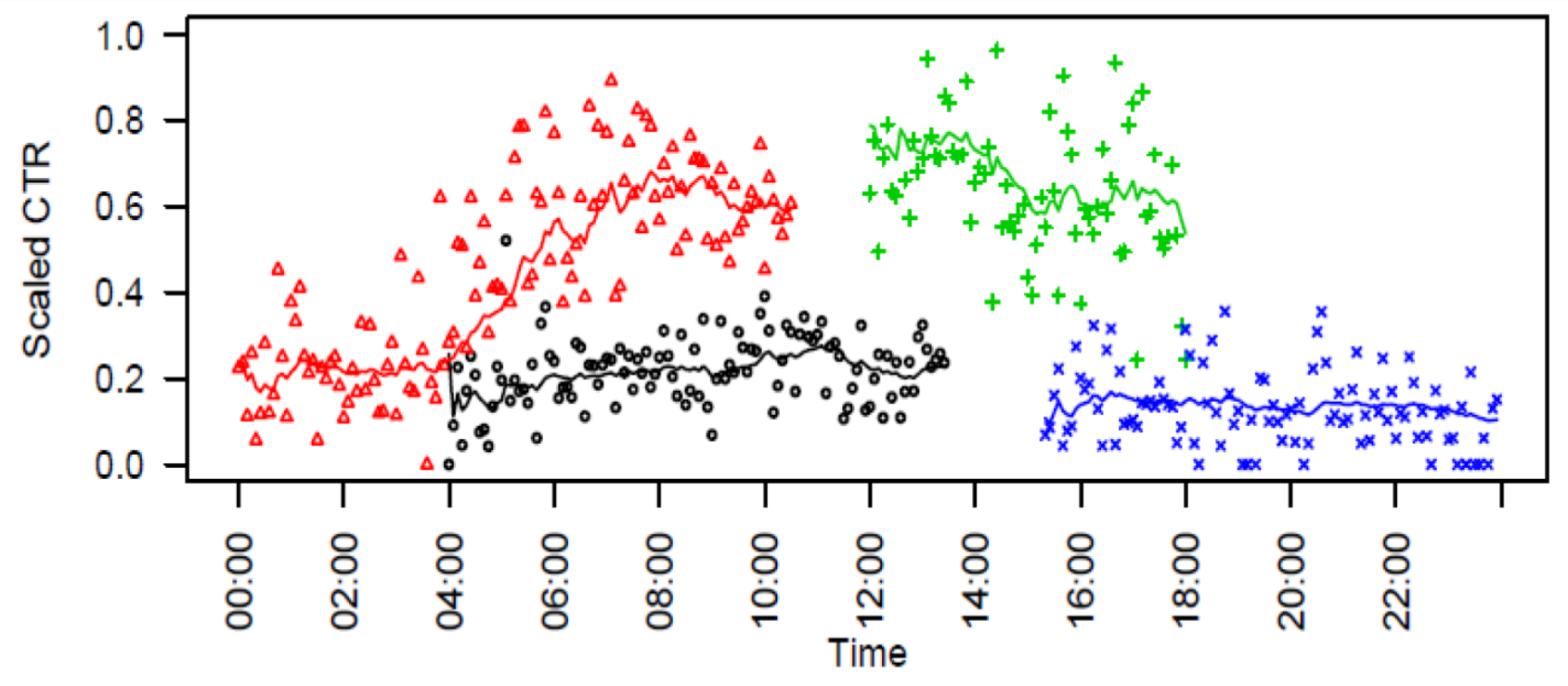
  а затем получили новую точку (n_t, r_t).
- Тогда, обозначив $\gamma_t = \frac{\mu_t}{\sigma_t^2}$ (эффективный размер выборки), сначала уточним оценки μ_t, σ_t:

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

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

$$\begin{aligned}\mu_{t+1} &= \mu_{t|t},\\ \sigma^2_{t+1} &= \sigma^2_{t|t} + \eta\left(\mu^2_{t|t} + \sigma^2_{t|t}\right).\end{aligned}$$

ПРИМЕР


Scaled CTR
1.0
0.8
0.6
0.4
0.2
0.0
00:00
02:00
04:00
06:00
08:00
10:00
12:00
14:00
16:00
18:00
20:00
22:00
Time

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 μ_0 и σ_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

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

ЧТО ДАЛЬШЕ

- 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?

КРАТКОЕ РЕЗЮМЕ

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!

Спасибо за внимание!