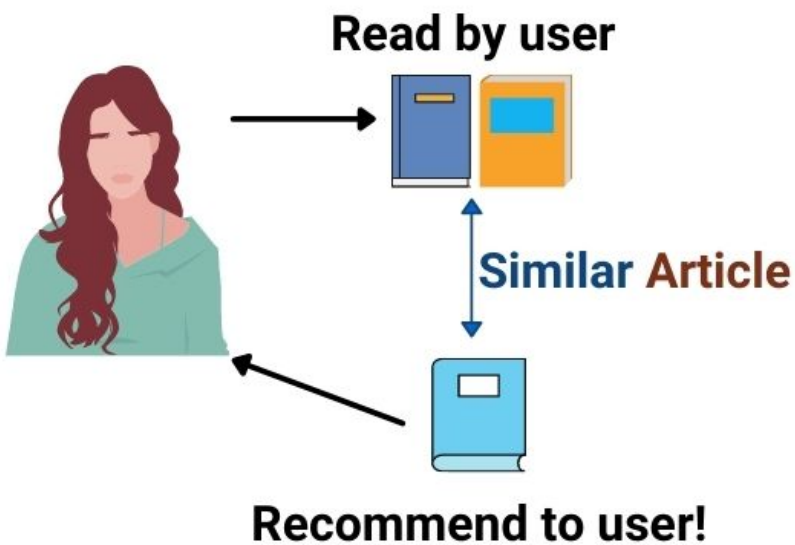
Рекомендательные системы

Виктор Китов

v.v.kitov@yandex.ru



Содержание

Рекомендательные системы

- Задача: рекомендовать пользователю приобрести новые товары/услуги по его интересам.
- Примеры рекомендаций:

сервис	предмет рекомендаций
YouTube, Netflix	видео
last.fm, pandora	музыка
amazon, ozon	товары
Яндекс.Дзен	новости
facebook	группы, друзья
LinkedIn	группы, друзья
TripAdvisor	достопримечательности

- Рекомендации повышают средний чек и user experience.
- 2018: 35% выручки Amazon и 75% выручки Netflix от рекомендованных товаров.

Доступная информация

Доступная информация:

- о пользователе: пол, возраст, интересы, логи предыдущих действий, социальные связи между пользователями и др.
- о товаре: категория, описание, характеристики, отзывы
- взаимодействие пользователей и товаров: матрица рейтингов (rating matrix) R
  - взаимодействие: например оценки товарам
  - $R \in \mathbb{R}^{\#\text{пользователей } x \ \#\text{товаров}}$
  - R большая и очень разреженная

Примеры матрицы рейтингов

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			5		2
U_2		5			4	
U_3	5	3		1		
U_4			3			4
U_5				3	5	
U_6	5		4			

	GLADIATOR	GODFATHER	BEN-HUR	GOODFELLAS	SCARFACE	SPARTACUS
U_1	1			1		1
U_2		1			1	
U_3	1	1		1		
U_4			1			1
U_5				1	1	
U_6	1		1			

Подходы к рекомендациям

Рекомендации могут быть основаны

- **по общей популярности** (summary-based, неперсональные)
- **на характеристиках пользователя и товара** (content-based)
  - считаем соответствие товара и пользователя (регрессия или классификация)

    F : (признаки пользователя, признаки товара) $\rightarrow$ соответствие
- **на матрице рейтингов** (collaborative filtering, matrix factorization)
  - задача: заполнение пустующих элементов матрицы по известным
- **на объединении всей доступной информации**
  - ансамбли повышают точность

Алгоритм общей популярности

- Неперсональные рекомендации, основанные на средней оценке товара
- Если оценок мало - доверие к ним меньше, поэтому можно сортировать по

$$\frac{1}{N}\sum_{i=1}^{N} r_i \to \frac{1}{N+\alpha}\sum_{i=1}^{N} r_i,\ \alpha = 20, 50, 100.$$

$$\bar{r} - \beta\sqrt{\mathrm{Var}[\bar{r}]}$$

Простейший алгоритм контентных рекомендаций

- Пример контентных рекомендаций:
  - если интересы пользователя известны:
    - выбрать товар, наиболее соответствующий по описанию интересам
    - например, используя косинусную меру близости
  - если интересы пользователя не известны - сгенерировать конкатенацией описаний ранее купленных товаров.

Полезно взвешивать слова по важности (IDF):

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Вес слова x в описании товара y

$tf_{x,y}$ = частота слова x в описании товара y

df_x = количество товаров, содержащих слово x

N = общее количество товаров

Детали задачи

Типы рекомендаций:

- консервативные (из любимой категории пользователя)
- новые рекомендации (расширение кругозора пользователя)

Получение пользовательских предпочтений

- явное (посмотрел фильм, лайкнул товар, написал отзыв)
- неявное (долго читал описание, искал соотв. товар)

Прогноз рейтинга vs. рекомендация

- Не всегда рекомендуем пользователю товар с макс. предсказанным рейтингом
- Также важны:
  - текущий поисковый запрос пользователя
  - прогнозы др. рекомендательных моделей (summary-based, content-based, collaborative filtering)
  - р-ция не должна быть тривиальной ("масло к хлебу")
  - разнообразие р-ций: следующие д. быть непохожи на предыдущие
  - для некоторых товаров (фильмы, новости), важна временная свежесть

Сложности рекомендательных систем

- **Масштабируемость**
  - много пользователей и товаров
- **Проблема холодного старта (cold start)**
  - новый пользователь => сложно предсказать предпочтения
  - новый товар => сложно предсказать свойства
- **Фальшивые данные (shilling attacks)**
  - производители товаров могут искусственно накручивать оценку своим товарам.
- **Низкое разнообразие**
  - популярные товары рекомендуются, получают еще больший рейтинг и продвигаются больше
  - например, музыкальные хиты, о которых все и так знают
- **Проблема уникальных вкусов (gray sheep)**
  - отдельные люди могут не соответствовать типичным вкусам

Содержание

Матрица рейтингов

- Пользователи (users) дают рейтинги товарам (items).
- Матрица рейтингов R: (user,item)→rating
- Задача - восстановить пропущенные значения по известным:

	Batman Begins	*Alice in Wonderland*	*Dumb and Dumber*	*Equilibrium*
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?

Типы матрицы рейтингов

Типы значений матрицы рейтингов

- бинарные
  - купил/не купил товар
  - лайкнул/не лайкнул пост
  - посетил/не посетил сайт
- тернарные
  - нравится/не нравится/нет мнения
- целые
  - 1,2,3,4,5 звезды
- вещественные
  - время, потраченное на сайте
  - количество информации, скачанное по определенному ресурсу, тарифу
  - количество денег, уже потраченное на товар

Соревнование Netflix - предшественник kaggle

- Netflix - сервис по онлайн-аренде DVD и доступа к цифровым каналам.
- Октябрь 2006 - сентабрь 2009: выложил данные для рекомендаций фильмов клиентам.
  - коллаборативная фильтрация с
    - 480.189 пользователями
    - 17.770 фильмами
    - оценками: 1,2,3,4,5.
  - призовой фонд: 1.000.000 $
- Формат данных:

$$< \text{пользователь}, \text{фильм}, \text{датаоценки}, \text{оценка} >$$

Соревнование Netflix - предшественник kaggle

- Netflix - сервис по онлайн-аренде DVD и доступа к цифровым каналам.
- Октябрь 2006 - сентабрь 2009: выложил данные для рекомендаций фильмов клиентам.
  - коллаборативная фильтрация с
    - 480.189 пользователями
    - 17.770 фильмами
    - оценками: 1,2,3,4,5.
  - призовой фонд: 1.000.000 $
- Формат данных:

$$< \text{пользователь}, \text{фильм}, \text{датаоценки}, \text{оценка} >$$

- Были привлечены ученые со всего мира, лучший алгоритм - ансамбль большого количества хороших решений.

Обозначения

- U - множество пользователей (users)
- I - множество товаров (items)
- u - отдельный пользователь
- i - отдельный товар
- I_u - множество товаров, оцененных пользователем u
- U_i - множество пользователей, оценивших товар i.
- $R = \{r_{u,i}\}_{i \in I, u \in U}$ - матрица рейтингов
- $r_{u,i}$ - рейтинг товара i пользователем u
- $\widehat{r}_{u,i}$ - предсказанный рейтинг

Простейшие базовые алгоритмы

Простейшие базовые алгоритмы:

- $\widehat{r}_{u,i} = \mu$ $(\mu = \frac{1}{n}\sum_{u,i} r_{u,i},\ n = |\{(u,i):\ r_{u,i}\ \text{известен}\}|)$
- $\widehat{r}_{u,i} = \overline{r}_u = \frac{1}{|I_u|}\sum_{i\in I_u} r_{u,i}$
- $\widehat{r}_{u,i} = \overline{r}_i = \frac{1}{|U_i|}\sum_{u\in U_i} r_{u,i}$

Базовый алгоритм

- Прогноз базового алгоритма:

$$b_{u,i} := \widehat{r}_{u,i} = \mu + \Delta_u + \Delta_i$$

$$\Delta_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$\Delta_i = \frac{1}{|U_i|} \sum_{u' \in U_i} \left(r_{u',i} - \mu - \Delta_{u'}\right)$$

- Интуиция:
  - Δ_u насколько пользователь оценивает товары выше среднего
  - Δ_i насколько оценка товара i выше средней оценки пользователя.

Базовый алгоритм с регуляризацией

- Базовый алгоритм с регуляризацией (with damping):

$$b_{u,i} := \widehat{r}_{u,i} = \mu + \Delta_u + \Delta_i$$

$$\Delta_u = \frac{1}{|I_u| + \alpha} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$\Delta_i = \frac{1}{|U_i| + \beta} \sum_{u' \in U_i} \left(r_{u',i} - \mu - \Delta_{u'}\right)$$

- $\alpha > 0, \beta > 0$ - сила регуляризации, $\alpha = \beta \approx 25$.
- Интуиция: доверяем Δ только когда выборка велика.

$$\Delta = \frac{1}{N + \alpha} \sum_{n=1}^{N} z_n = \begin{cases} \approx 0 & \text{для малых } N \\ \approx \frac{1}{N} \sum_{n=1}^{N} z_n & \text{для больших } N \end{cases}$$

Мотивация базового подхода

Мотивация базового подхода:

- сравнивать точность с более продвинутыми методами
- заполнить пропуски базовым прогнозом
  - например, с рекомендациях сингулярным разложением
- предсказывать $r_{u,i} - \widehat{r}_{u,i}$ вместо $r_{u,i}$
  - базовый прогноз:$\widehat{r}_{u,i}$, $r_{u,i} - \widehat{r}_{u,i}$ предсказывается продвинутой моделью
    - которая будет концентрироваться только на проблемных случаях

User-based алгоритм

Определим функцию близости между пользователями $s(u_1, u_2)$.

Построения прогноза $\widehat{r}_{u,i}$:

1. Найдем подмножество пользователей U_i, оценивших товар i.
2. Используя $s(u_1, u_2)$ найдем похожих на u пользователей N_u
3. Прогноз-средний рейтинг среди пользователей $U_i \cap N_u$.

User-based алгоритм

Базовый user-based прогноз:

$$\widehat{r}_{u,i} = \frac{\sum_{u' \in U_i \cap N_u} s(u,u') r_{u',i}}{\sum_{u' \in U_i \cap N_u} |s(u,u')|}$$

User-based алгоритм

Базовый user-based прогноз:

$$\widehat{r}_{u,i} = \frac{\sum_{u' \in U_i \cap N_u} s(u,u') r_{u',i}}{\sum_{u' \in U_i \cap N_u} |s(u,u')|}$$

+учет пользовательских смещений (оптимисты/пессимисты):

$$\widehat{r}_{u,i} = \mu_u + \frac{\sum_{u' \in U_i \cap N_u} s(u,u') \left(r_{u',i} - \mu_{u'}\right)}{\sum_{u' \in U_i \cap N_u} |s(u,u')|}$$

User-based алгоритм

Базовый user-based прогноз:

$$\widehat{r}_{u,i} = \frac{\sum_{u' \in U_i \cap N_u} s(u,u') r_{u',i}}{\sum_{u' \in U_i \cap N_u} |s(u,u')|}$$

+учет пользовательских смещений (оптимисты/пессимисты):

$$\widehat{r}_{u,i} = \mu_u + \frac{\sum_{u' \in U_i \cap N_u} s(u,u') \left(r_{u',i} - \mu_{u'}\right)}{\sum_{u' \in U_i \cap N_u} |s(u,u')|}$$

+учет пользовательских разбросов
(эмоциональные/стабильные)

$$\widehat{r}_{u,i} = \bar{r}_u + \sigma_u \frac{\sum_{u' \in U_i \cap N_u} s(u,u') \left(r_{u',i} - \bar{r}_{u'}\right)/\sigma_{u'}}{\sum_{u' \in U_i \cap N_u} |s(u,u')|}$$

μ_u,σ_u - среднее и стд. отклонение пользователя u.

Выбор пользователей, похожих на u

Выбор пользователей N_u, похожих на u:

- использовать всех: $U \backslash \{u\}$
- использовать K самых похожих на u (обычно $K \in [20, 50]$)
- использовать $\{u' : s(u', u) \geq \text{threshold}\}$

Корреляция Пирсона

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{v,i} - \bar{r}_v)^2}}$$

- Учитывает только линейную связь (можно считать ранговую корреляцию),
- Если есть нейтральная оценка: $\bar{r} \to r_{neutral}$
- Может давать завышенную корреляцию для пользователей с несколькими рейтингами
  - решение: использовать $s'(u,v) = s(u,v) \min\{|I_u \cap I_v|/50, 1\}$

$$s(u,v) = \frac{\sum_{i \in I_u \cap I_v} r_{u,i} r_{v,i}}{\sqrt{\sum_{i \in I_u \cap I_v} r_{u,i}^2} \sqrt{\sum_{i \in I_u \cap I_v} r_{v,i}^2}}$$

Item-based алгоритм

Определим похожесть товаров $s(i_1, i_2)$.
Алгоритм определения $\widehat{r}_{u,i}$:

1. Определим подмножество товаров I_u, оцененных u.
2. Используя $s(i_1, i_2)$, определим подмножество товаров S_i, похожих на i.
3. Прогноз=средний рейтинг u по товарам $I_u \cap S_i$:

$$\widehat{r}_{u,i} = \frac{\sum_{i' \in I_u \cap S_i} s(i, i') r_{u,i'}}{\sum_{i' \in I_u \cap S_i} |s(i, i')|} \tag{1}$$

Можем делать поправку на среднее & разброс:

$$\widehat{r}_{u,i} = \mu_i + \sigma_i \frac{\sum_{i' \in I_u \cap S_i} s(i, i') \left(r_{u,i'} - \mu_{i'}\right) / \sigma_{i'}}{\sum_{i' \in I_u \cap S_i} |s(i, i')|}$$

Особенность item-based алгоритма

- Необходимо быстро пересчитывать рекомендации по динамически наполняемой корзине товаров в магазине.
- Использовать user-based или item-based алгоритм?

Особенность item-based алгоритма

- Необходимо быстро пересчитывать рекомендации по динамически наполняемой корзине товаров в магазине.
- Использовать user-based или item-based алгоритм?
- Профиль пользователя динамически меняется.
  - User-based: нужно пересчитывать похожих пользователей, долго.
  - Item-based: $s(i, i') \approx const$, предпосчитаем их вместе с S_i $\forall i$. Меняется только I_u и $r_{u,i'}$, поэтому (1) можно быстро пересчитать.

Похожесть товаров

$$s(i,j) = \frac{\langle r_i, r_j \rangle}{\|r_i\| \, \|r_j\|} = \frac{\sum_{u \in U_i \cap U_j} r_{u,i} r_{u,j}}{\sqrt{\sum_{u \in U_i \cap U_j} r_{u,i}^2} \sqrt{\sum_{u \in U_i \cap U_j} r_{u,j}^2}}$$

$$s(i,j) = \frac{\sum_{u \in U_i \cap U_j} (r_{u,i} - \overline{r}_i)(r_{u,j} - \overline{r}_j)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{u,i} - \overline{r}_i)^2} \sqrt{\sum_{u \in U_i \cap U_j} (r_{v,j} - \overline{r}_j)^2}}$$

Можем также использовать корреляцию между рангами.

Технические улучшения

Коррекция на оптимистов/пессимистов:

- Пользователи-оптимисты завышают оценки, а пессимисты-занижают.
- Наличие оптимистов и пессимистов делает похожесть $s(i,j)$ выше, чем она есть на самом деле.
- Чтобы избежать завышения похожести $s(i,j)$, его можно считать по $R' = \{r_{u,i} - b_{u,i}\}_{u,i}$.
  - товары похожи, если скоррелированы не их оценки, а отклонения от ожидаемых оценок

Более сильный учет одинаковости предпочтений для пользователей, оценивших мало товаров:

- Для этого можно нормализовать $r_u \leftarrow r_u / \|r_u\|$, $\uparrow r_u$ при малом $\|r_u\|$.

Рекомендации - сокращенное сингулярное разложение

Прогноз для имеющихся пользователей r_u:

$$R \approx U_K \Sigma_K V_K^T = \widehat{R};\ \ \widehat{r}_u = \widehat{R}_{u:}$$

Прогноз для новых пользователей r_u:

- строки V_K^T - K "тем" предпочтений (главные компоненты).

1. Получаем предпочтения пользователя в K "темах" $\widehat{p}_u$:

$$\begin{aligned}\widehat{p}_u &= \arg\min_p \|r_u - V_K p\|^2 = \{\text{решение МНК}\} = \\ &= \left(V^T V\right)^{-1} V^T r_u = \{\text{ортогональность } V\} \\ &= V^T r_u \text{ - вектор скалярных произведений на темы}\end{aligned}$$

2. Восстанавливаем все рейтинги через тематические предпочтения: $\widehat{r}_u = V\widehat{p}_u$

Недостатки подхода

Недостаток: 0-отстутсвие мнения, но он же - минимально доступная оценка.

- возможное решение: $0 \to \bar{r}_u$ либо заменяем прогнозом базовой модели.
  - не точное решение, т.к. прогнозы учитываются с таким же весом, как истинные рейтинги.

Хотим использовать рекомендации, но опираясь только на истинные рейтинги.

Переформулировка подхода через оптимальность разложения

Сокращенное сингулярное разложение - оптимальная низкоранговая аппроксимация:

$$\widehat{R} = \underbrace{U_K \Sigma_K}_{:=P} \underbrace{V_K^T}_{:=Q} = arg \min_{A \in \mathbb{R}^{MxN}, \text{rank } B \leq K} \|R - B\|_F^2$$

Решим:

$$\|R - PQ\|_F^2 \to \min_{P,Q} \tag{2}$$

$P = [p_1, ...p_{|U|}]^T \in \mathbb{R}^{|U|xK}, \quad Q = [q_1, ...q_{|I|}] \in \mathbb{R}^{Kx|I|}.$
Прогноз: $\widehat{R} = PQ$.

Оценка разреженного сингулярного разложения[1]

p_u:=u-ая строка P, q_i:=i-й столбец Q, $\{PQ\}_{ui} = \langle p_u, q_i \rangle$

Функция потерь:

$$\sum_{(u,i)\in D} \left(\underbrace{r_{ui} - \{PQ\}_{ui}}_{\varepsilon_{ui}} \right)^2 = \sum_{(u,i)\in D} \left(\underbrace{r_{ui} - \langle p_u, q_i \rangle}_{\varepsilon_{ui}} \right)^2 \to \min_{P,Q}$$

Оценка:

- повторять до сходимости:
  сэмлируем (u, i), такую, что r_{ui} известна,
  уменьшаем ε_{ui}^2, смещая P, Q в направлении $-\eta\nabla(\varepsilon_{ui}^2)$:

$$\begin{cases} p_u := p_u - \eta \frac{\partial \varepsilon_{ui}^2}{\partial p_u} = p_u + 2\eta\varepsilon_{ui} q_i & k \in \{1, 2, ..K\} \\ q_i := q_i - \eta \frac{\partial \varepsilon_{ui}^2}{\partial q_i} = q_i + 2\eta\varepsilon_{ui} p_u & k \in \{1, 2, ..K\} \end{cases}$$

[1] Др. название - latent factor model.

Преимущества подхода

- Используются только известные r_{ui}.
  - нет смещения из-за заполнения пропусков.
- Метод работает при динамическом появлении новых данных
  - нового пользователя u
  - нового товара i
  - нового рейтинга r_{ui}
- За счет SGD - масштабируется и быстро считается на больших данных.

Возможные модификации

- Легко добавить регуляризацию (против переобучения)[2]:

$$\varepsilon_{ui}^2 + \lambda \|p_u\|_2^2 + \mu \|q_i\|_2^2 \to \min_{P,Q}$$

- Можно добавить ограничения $p_{tu} \geq 0$, $q_{ti} \geq 0$
  - методом проекции градиента (перепроецируем на область после каждого шага)

Смещения на оптимистов/пессимистов в товарах может обработать композиция с базовой модели B:

$$\|R - B - PQ\|_F^2 \to \min_{P,Q}$$

$$\text{прогноз: } \widehat{R} = B + PQ$$

[2]Как изменится шаг обновления p_u, q_i?

Заключение

- Рекомендации могут быть основаны:
  - на характеристиках пользователя и товара (content-based)

    F : (признаки пользователя, признаки товара) $\rightarrow$ соответствие

  - на матрице рейтингов (collaborative filtering)
    - базовый алгоритм (когда данных совсем мало)
    - user-based
    - item-based (работает с динамическими рейтингами)
    - основанный на SVD (требуется заполнение пропусков)
    - основанный на разреженном SVD (использует только известные $r_{u,i}$)