

Рекомендательные системы

МАКСИМОВСКАЯ
АНАСТАСИЯ

Рекомендательные системы

- Предсказываем, что будет интересно пользователю на основе информации о его профиле
- Уже знакомые нам Netflix, Кинопоиск, ЯндексМузыка, Youtube

Рекомендательные системы

беру! [Каталог товаров](#)

Профиль ⁵ Беру Бонусы Корзина







Маркетплейс от Сбербанка и Яндекса Скидки? Беру! Бесплатная доставка Удобная оплата Лёгкий возврат Город: Москва

Ваш город — Москва?
[Да, верно](#) [Нет, не он](#)

3 дня особых скидок с Беру Бонусами
Только с 10 по 12 июля

Скидка 20% на покупки на Беру

Стоит приглядеться

					
4 038 Р от 146 Р / мес.	10 990 Р от 397 Р / мес.	1 200 Р ★★★★★ 1548 отзывов	1 790 Р ★★★★★ 931 отзыв	10 690 Р 13 990 Р -24% от 386 Р / мес.	889 Р ★★★★★ 25

Коллаборативная фильтрация

Матрица предпочтений

	Товар 1	Товар 2	Товар 3	Товар 4	Товар 5
Клиент 1		3		5	
Клиент 2	1		1	1	
Клиент 3	2			3	2
Клиент 4		4			5
Клиент 5	5		2	3	4

Memory-based

- Пусть есть 2 пользователя u и v . Множество товаров, для которого известны оценки обоих пользователей:

$$I_{uv} = \{i \in I \mid \exists r_{ui} \ \& \ \exists r_{vi}\}$$

- Тогда сходство двух пользователей можно вычислить через корреляцию Пирсона:

$$w_{uv} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}}$$

Memory-based

- Пусть есть 2 пользователя i и j . Множество пользователей, для которого известны оценки этих товаров:

$$U_{ij} = \{u \in U \mid \exists r_{ui} \ \& \ \exists r_{uj}\}$$

- Тогда сходство двух товаров можно вычислить через корреляцию Пирсона:

$$w_{ij} = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

User-based

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in U_i} \text{sim}(u, v) (r_{vi} - \bar{r}_v)}{\sum_{v \in U_i} \text{sim}(u, v)}$$

- \bar{r}_u — средняя оценка пользователя u
- \bar{r}_v — средняя оценка пользователя v

Item-based

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in I_u} \text{sim}(i, j) (r_{uj} - \bar{r}_j)}{\sum_{j \in I_u} \text{sim}(i, j)}$$

Комментарий по подходу

Преимущества:

- Меньше размерность матрицы расстояний
- Легче вычислять
- Модель более устойчива к переобучению
- Можно реже обновлять
- Меньше подвержены изменению предпочтений со временем

Недостатки:

- Проблема холодного старта
- Плохие предсказания для новых/нетипичных пользователей/объектов
- Тривиальность рекомендаций
- Ресурсоемкость вычислений

Модели со скрытыми переменными

- Хотим построить для каждого пользователя u и товара i векторы, которые будут отражать их «категории интересов»
- Например, каждую компоненту такого вектора можно интерпретировать как степень принадлежности данного товара к определённой категории или степень заинтересованности данного пользователя в этой категории
- По сути, векторы пользователей и товаров являются представлениями (embeddings), позволяющими свести эти сущности в одно векторное пространство.

Latent Factor Model (LFM)

- Сходство пользователя и товара будем вычислять через скалярное произведение их представлений:

$$r_{ui} \approx \langle p_u, q_i \rangle$$

- Мы можем записать функционал ошибки, исходя из способа вычисления сходства:

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 \rightarrow \min_{P, Q}$$

Latent Factor Model (LFM)

➤ Данный функционал можно регуляризовать:

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 + \lambda \sum_{u \in U} \|p_u\|^2 + \mu \sum_{i \in I} \|q_i\|^2 \rightarrow \min_{P, Q}$$

SGD

- Для решения данной задачи популярно 2 подхода
- Первый — стохастический градиентный спуск, который на каждом шаге случайно выбирает пару (u, i) :

$$\begin{aligned} p_{uk} &:= p_{uk} + \eta q_{ik} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle), \\ q_{ik} &:= q_{ik} + \eta p_{uk} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle). \end{aligned}$$

Alternating Least Squares (ALS)

- Можно показать, что этот функционал не является выпуклым в совокупности по P и Q , но при это становится выпуклым, если зафиксировать либо P , либо Q .
- Оптимальное значение P при фиксированном Q (и наоборот) можно выписать аналитически, — но оно будет содержать обращение матрицы:

$$p_u = \left(\sum_{i:\exists r_{ui}} q_i q_i^T \right)^{-1} \sum_{i:\exists r_{ui}} r_{ui} q_i;$$
$$q_i = \left(\sum_{u:\exists r_{ui}} p_u p_u^T \right)^{-1} \sum_{u:\exists r_{ui}} r_{ui} p_u;$$

Hierarchical alternating least squares (HALS)

- Чтобы избежать сложной операции обращения, будем фиксировать всё, кроме одной строки p_k матрицы P или одной строки q_k матрицы Q . В этом случае можно найти оптимальное значение для p_k и q_k :

$$p_k = \frac{q_k(R - \sum_{s \neq k} p_s q_s^T)^T}{q_k q_k^T},$$
$$q_k = \frac{p_k(R - \sum_{s \neq k} p_s q_s^T)}{p_k p_k^T}.$$

Учёт неявной информации

- Введём показатель неявного интереса пользователя к товару:

$$s_{ui} = \begin{cases} 1, & \exists r_{ui}, \\ 0, & \text{иначе.} \end{cases}$$

- Введём веса, характеризующие уверенность в показателе интереса:

$$c_{ui} = 1 + \alpha r_{ui}.$$

- Коэффициент α позволяет регулировать влияние явного рейтинга на уверенность в интересе.

Implicit ALS (iALS)

➤ Теперь мы можем задать функционал:

$$\sum_{(u,i) \in D} c_{ui} (s_{ui} - \bar{s}_u - \bar{s}_i - \langle p_u, q_i \rangle)^2 + \lambda \sum_u \|p_u\|^2 + \mu \sum_i \|q_i\|^2 \rightarrow \min_{P,Q}$$

➤ Как и раньше, обучать его можно с помощью стохастического градиентного спуска, ALS или HALS.

Факторизационные машины

- Допустим, что целевая переменная зависит от парных взаимодействий между признаками. В этом случае представляется разумным строить полиномиальную регрессию второго порядка:

$$a(x) = w_0 + \sum_{j=1}^d w_j x_j + \sum_{j_1=1}^d \sum_{j_2=j_1+1}^d w_{j_1 j_2} x_{j_1} x_{j_2}.$$

- Данная модель состоит из $d(d-1)/2 + d + 1$ параметров.

Факторизационные машины

- Предположим, что вес взаимодействия признаков j_1 и j_2 может быть аппроксимирован произведением низкоразмерных скрытых векторов v_{j_1} и v_{j_2} , характеризующих эти признаки. Получим:

$$a(x) = w_0 + \sum_{j=1}^d w_j x_j + \sum_{j_1=1}^d \sum_{j_2=j_1+1}^d \langle v_{j_1}, v_{j_2} \rangle x_{j_1} x_{j_2}.$$

- Благодаря описанному трюку число параметров снижается до $dr + d + 1$, где r — размерность скрытых векторов.

Факторизационные машины

➤ Нашу задачу

$$\sum_{(u,i) \in R} (r_{ui} - \bar{r}_u - \bar{r}_i - \langle p_u, q_i \rangle)^2 \rightarrow \min_{P, Q}$$

можно сформулировать как задачу построения регрессии с двумя категориальными признаками: идентификатором пользователя и идентификатором товара и целевым признаком – рейтингом товара.

➤ После бинаризации признаков получим, что каждый объект x описывается $|U| + |I|$ признаками, причём ненулевыми являются ровно два из них: один соответствует номеру пользователя u , второй — номеру товара i

Факторизационные машины

- Тогда факторизационная машина примет следующий вид:

$$a(x) = w_0 + w_u + w_i + \langle v_u, v_i \rangle.$$

- По сути, факторизационная машина позволяет строить рекомендательные модели на основе большого количества категориальных и вещественных признаков.
- Существует несколько методов настройки факторизационных машин, из которых наиболее совершенным считается метод Монте-Карло на основе марковских цепей; реализацию можно найти в библиотеке libFM.

Контентные модели

- В коллаборативной фильтрации используется информация о предпочтении пользователей и об их сходствах, но при этом никак не используются свойства самих пользователей или товаров
- **Идея:** все товары описываются с помощью векторов (представлений, embeddings), и затем измеряется сходство между вектором нового товара и векторами товаров из истории пользователя.

Статистические признаки

- Важны и более простые типы факторов: конверсия просмотра данного товара в покупку за всю историю магазина, число покупок данного пользователя в категории данного товара, число покупок данного пользователя и т.д.
- Если товар или пользователь уже набрали достаточно статистики, то зачастую такие признаки оказываются самыми главными при принятии решения, поскольку уже содержат в себе достаточно информации о предпочтениях.

Метрики

Качество предсказаний

- **Предсказание рейтингов.** Если модель предсказывает рейтинг или другую вещественную величину (например, длительность просмотра), то качество может измеряться через MSE, RMSE, MAE или другие регрессионные метрики.
- **Предсказание событий.** Если модель предсказывает вероятность некоторого события (клика, покупки, просмотра, добавления в корзину), то качество можно измерять с помощью метрик качества классификации — доля правильных ответов, точность, полнота, F-мера, AUC-ROC, AUC-PR, log-loss и т.д.

mAP @ N

➤ **mAP @ N** (mean average precision). N – длина ленты рекомендаций. P = число релевантных рекомендаций / общее число рекомендаций. AP @ N считается для каждого пользователя, после чего считается среднее и получаем mAP @ N

$$AP@N = \frac{1}{m} \sum_{k=1}^N (P(k) \text{ if } k^{th} \text{ item was relevant}) = \frac{1}{m} \sum_{k=1}^N P(k) \cdot rel(k).$$

mAP @ N

	Position	Movie Recommendation	Did user watch recommended movie
k=1 {	1	Interstellar	True positive
	2	Inception	True positive
	3	Avengers	False positive
	4	Harry Potter	True positive
	5	The Imitation Game	False positive

$P(k=1) = 1/1 \longrightarrow p @ k = \text{proportion of all examples above that rank which are from the positive class}$

Calculating precision up to cutoff $k = 1$

mAP @ N



Precision Represented at each cut off
'k' (1 to 5)

Recommendations represented in terms of
relevance for the user (1 if user watched the
recommended movie, 0 otherwise)

mAR @ N

- **mAR @ N** (mean average recall). r = число релевантных рекомендаций / общее число возможных релевантных рекомендаций. В примере ниже рекомендовано 5 товаров из 10. метрика считается для каждого пользователя, после чего считается среднее и получаем mAR @ N
- **F1-score** = $2 * (mAR * mAP) / (mAR + mAP)$

mAR @ N

Расчет:

1. Для каждого пользователя собираем таблицу вида рекомендация (id товара) и купил ли он ее (1 – купил, 0 – нет)
2. Считаем $r(k)$: сколько из первых k рекомендаций релевантны / N (общее число рекомендаций). Пример: Пусть у нас есть 5 товаров и список (1 1 0 1 0), где 1 означает, что пользователь купил товар, а 0 – нет. Тогда: $r(k=1) = 1/5$, $r(k=2)=2/5$, $r(k=3)=2/5$, $r(k=4)=3/5$, $r(k=5)=3/5$.
3. Перемножаем $r(k)$ на 1, если товар куплен, и 0, если нет. В примере выше: $1/5*1 + 2/5*1 + 2/5*0 + 3/5*1 + 3/5*0 = 1.2$ Это мы посчитали AR @N для одного пользователя.
4. Считаем метрику для всех пользователей по схеме выше и усредняем. Ответ – итоговое среднее число

Personalisation

➤ **personalization** = $1 - \text{средняя cosine similarity}$

Расчет:

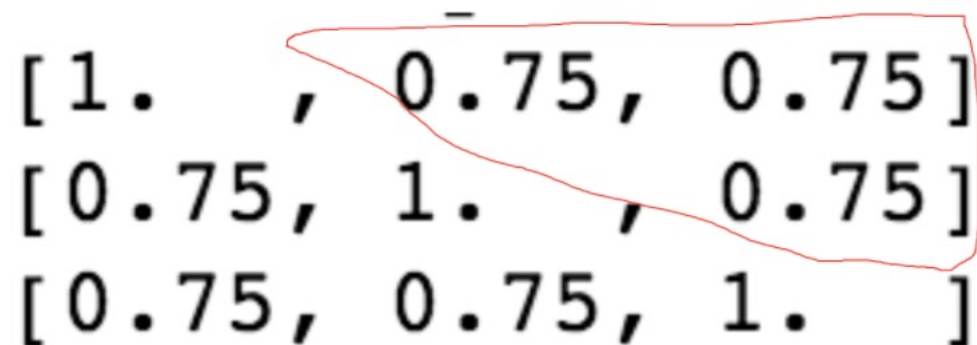
1. Пусть у нас есть таблица с 3 строчками (3 пользователями). Первому мы порекомендовали товары А, В, С, D, второму – А, В, С, Х, третьему – А, В, С, Z. Составим новую таблицу где число строчек равно числу пользователей, а число столбцов равно числу товаров, которые были в списке с рекомендациями (в нашем случае это товары А, В, С, Х, Z – 5 столбцов). Заполняет таблицу 1, если пользователю в строке i был порекомендован товар в столбце j , и 0 иначе. Для нашего примера она будет выглядеть так:

Personalisation

	A	C	B	D	X	Z
0	1	1	1	1	0	0
1	1	1	1	0	1	0
2	1	1	1	0	0	1

Personalisation

2. Создаем cosine similarity matrix, где будет число строчек и число столбцов равно числу пользователей. Для примера выше строчка 1 будет заполняться как (cosine similarity между рядом 0 и 0, cosine similarity между рядом 0 и 1, cosine similarity между рядом 0 и 2). Матрица на этом шаге будет выглядеть так:



The image shows a 3x3 cosine similarity matrix. The diagonal elements are 1.0, and the off-diagonal elements are 0.75. A red hand-drawn circle highlights the off-diagonal elements, indicating the similarity between different rows.

1.0	0.75	0.75
0.75	1.0	0.75
0.75	0.75	1.0

Personalisation

3. Видим, что по диагонали 1 (такое происходит, тк мы считали похожесть векторов самих на себя), считаем среднее по треугольнику, что выше этой диагонали (выделено красным). В нашем примере среднее = 0.75
4. Считаем итоговый показатель $1 - \text{среднее из пункта 3} = 1 - 0.75 = 0.25$.

Качество ранжирования

- Обозначим через a_{ui} предсказание модели для пользователя u и товара i . Отсортируем все товары по убыванию предсказания a_{ui} . Тогда для товара i_p на позиции p можно вычислить его полезность $g(r_{ui_p})$ и штраф за позицию $d(p)$.
- Введем метрику DCG (Discounted cumulative gain):

$$\text{DCG@k}(u) = \sum_{p=1}^k g(r_{ui_p})d(p)$$

Качество ранжирования

- Примерами конкретных функций могут служить такие $g(r)$ и $d(p)$:

$$g(r) = 2^r - 1 \qquad d(p) = \frac{1}{\log(p+1)}.$$

- Чтобы значение метрики легче было интерпретировать, её можно поделить на значение DCG при идеальном ранжировании — в этом случае получим метрику nDCG (normalized DCG):

$$\text{nDCG@k}(u) = \frac{\text{DCG@k}(u)}{\max \text{DCG@k}(u)}.$$

Покрытие

- **Покрытие товаров.** В качестве простейшей метрики можно использовать покрытие каталога, которое вычисляется как доля товаров, порекомендованных хотя бы один раз.
- Пусть $p(i)$ — доля показа товара $i \in I$ среди всех показов для данной рекомендательной системы. Тогда разнообразие можно определить как энтропию такого распределения:

$$H(p) = - \sum_{i \in I} p(i) \log p(i).$$

- **Покрытие пользователей.** Имеет смысл вычислять долю пользователей, для которых не рекомендуется ни одного товара, чтобы отслеживать проблемы с покрытием в модели рекомендаций

Новизна

Под новизной понимается доля новых для пользователя товаров среди рекомендованных. Можно предложить несколько подходов к измерению новизны:

- Для каждого рекомендованного товара добавить в интерфейсе возможность сообщить о том, что этот товар пользователь уже видел.
- Удалить из обучающей выборки часть товаров, которые пользователь купил или просмотрел. Далее будем оценивать новизну на основе того, как часто эти удалённые товары попадают в рекомендации.
- Можно вычислять как долю угаданных рекомендательной системой товаров, где каждый товар имеет вес, обратно пропорциональный популярности этого товара.

Прозорливость (serendipity)

- Под прозорливостью понимается способность рекомендательной системы предлагать товары, которые отличаются от всех купленных пользователем ранее.
- Прозорливость можно измерять как долю рекомендаций, которые далеки от всех оценённых пользователем товаров.
- Расстояние между новым товаром b и множеством уже оцененных товаров B :

$$d(b, B) = \frac{1 + c_B - c_{B,w(b)}}{1 + c_B}$$

Разнообразие

- Под разнообразием понимается степень сходства товаров внутри одной пачки рекомендаций (т.е. тех товаров, которые одновременно рекомендуются пользователю).
- Можно её задавать как, например, среднее попарное расстояние между товарами в одной пачке.