KARPOV.COURSES >>>
КОНСПЕКТ

Конспект > 22 урок > Рекомендательные системы

Рекомендательные системы — программы, основанные на алгоритмах, предсказывающих на основе данных о пользователе, какие релевантные объекты (книги, фильмы, музыка, новости) ему будут интересны.

Пример рекомендательной системы - портал Кинопоиск, где на основе оценок пользователей осуществляется подбор наиболее подходящих по мнению алгоритма фильмов.

Обозначим набор пользователей как множество

$U = \{u_i\}_{i=1}^{n}$

Набор айтемов(фильмов)

$I = \{i_i\}_{i=1}^{m}$

Задача на основании данных наблюдений понять, какую рекомендацию фильма (i_i) выдать тому или иному пользователю (u_i).

Оценки фильмов пользователями можно представить в виде матрицы R размером nxm, где каждой ячейке будет соответствовать оценка от 0 до 10 того или иного фильма пользователем r_{ui}, для непросмотренных фильмов будет стоять прочерк.

>Контентная рекомендация

Рассмотрим общий алгоритм построения рекомендательной системы.

Шаг 1: По парам (u,i) построим пайплайн выделения l признаков

$x_{ui} = (d_1^{ui}, \ldots, d_l^{ui})$

Получаем пары объект-ответ: $(x_{ui}, r_{ui})_{j=1}^{n}$ на которых можно применить уже привычное нам обучение с учителем.

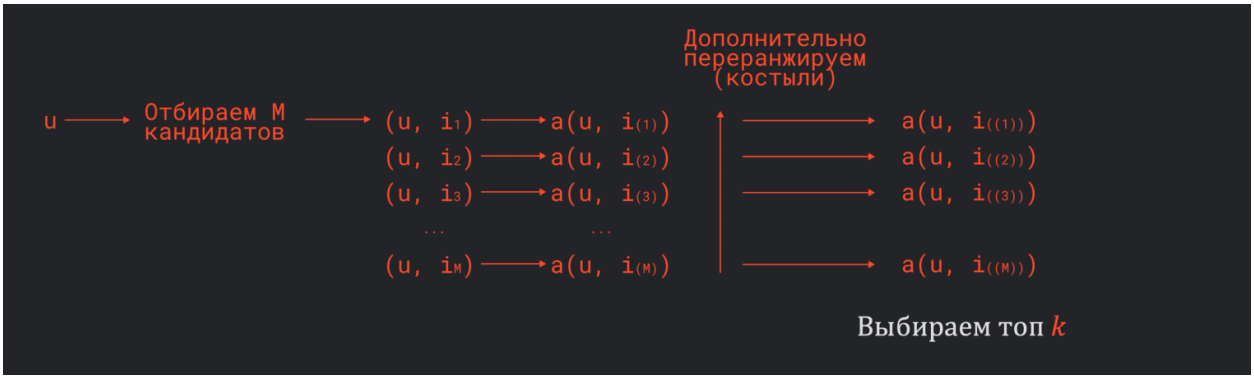
Шаг 2: Обучение выбранной модели

Шаг 3: Вывод рекомендаций

- Допустим обучили модель и получили некую оценку для каждой пары пользователь-фильм $a(u,i) \approx r_{ui}$
- Теперь наша задача дать некоторое количество (k) рекомендаций данному пользователю (u)
- Для этого прогоним модель по всем m айтемам и получим m прогнозов
- Отсортируем данные прогнозы в порядке убывания
- Выберем топ k выходов - это и будут рекомендации для пользователя u

Главный минус данного подхода - потребление огромного ресурса, так как для каждого пользователя будет необходимо прогнать модель на миллионах фильмов.

На практике данная процеура имеет несколько иной вид:



Действительно, если пользователь никогда не выбирает фильмы определенных жанров или на пределенном языке, можно сразу же их отсеять и не использовать в построении рекомендаций. То есть будем “скармливать” модели не весь датасет, а лишь его часть.

После этого мы дополнительно переранжируем выходы модели, добавив некие критерии, такие как новизна или известность фильма.

Основной недостаток данного алгоритма - необходимость придумывать хорошие фичи, трудозатраты на переранжирование и отбор айтемов.

>Коллаборативный подход

Обойти недостатки предыдущих подходов позволяет коллаборативный подход, для реализации которого нужна только матрица R.

Рассмотрим две реализации данного подхода

>USER-BASED

Основан на схожести пользователей между собой. Если пользователи u и v похожи, то если u оценил высоко фильм i, то высока вероятность, что и v его оценит.

Как же мы будем оценивать схожесть пользователей? Посмотрим на оценки данные ими относительно просмотренных фильмов, отберем фильмы, для который есть оценки обоих юзеров:

$$I_{u,v} = \{i \in I : \exists r_{ui} \ \& \ \exists r_{vi}\}$$

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	10	5	-	3	4	5
u_2	-	7	8	-	2	6
u_3	3	-	6	8	-	5

Посчитаем для айтемов из $I_{u,v}$ коэффициенты корреляции $w_{u,v}$ для определения того, насколько один пользователь похож на другого

	i_2	i_5	i_6
u_1	5	4	5
u_2	7	2	6

$w_{u_1,u_2} \approx 0.98$

	i_1	i_4	i_6
u_1	10	3	5
u_3	3	8	5

$w_{u_1,u_3} \approx -0.93$

$$w_{u,v} = \frac{\sum_{i \in I_{uv}} (r_{ui} - r_u^{\text{ср}})(r_{ui} - r_v^{\text{ср}})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_u^{\text{ср}})^2} \sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_v^{\text{ср}})^2}}$$

Далее для кадого пользователя u_0 будем отбирать похожих на него по некоторому порогу коэффициента корреляции:

$$U(u_0) = v \in U | w_{u_0,v} > \alpha$$

Такое множество U похожих пользователей принято называть коллаборацией

Например если порог $\alpha = 0.5$
$U(u_1) = u_2$

Теперь мы можем на основе оценок пользователей из $U(u_0)$ прикинуть, какой рейтинг выставит пользователь u_0 для ряда непросмотренных фильмов.

Для этого обозначим $U(u_0)_i$ множество тех пользователей из $U(u_0)$, которые дали айтему i какую-либо оценку.

Спрогнозируем оценку, которую поставит пользователь u_0 фильму i:

$$a(u_0, i) = r_{u_0}^{\text{ср}} + \frac{\sum_{v\ in U(u_0)_i}(r_{vi} - r_v^{\text{ср}}) \cdot w_{u_o v}}{\sum_{v \in U(u_0)_i} |w_{u_o, v}|}$$

Данная формула описывает следующие действия:

-Вычислить среднюю оценку пользователя по всем фильмам, которые он смотрел.

-Вычислить разницу в оценках соседями пользователя данного фильма от их средних оценок, умноженную на коэффициент корреляции.

-Разделить на сумму корреляций по модулю.

Таким образом мы считаем предполагаемый “сдвиг оценки” и прибавляем его к средней оценке пользователя.

>ITEM-BASED

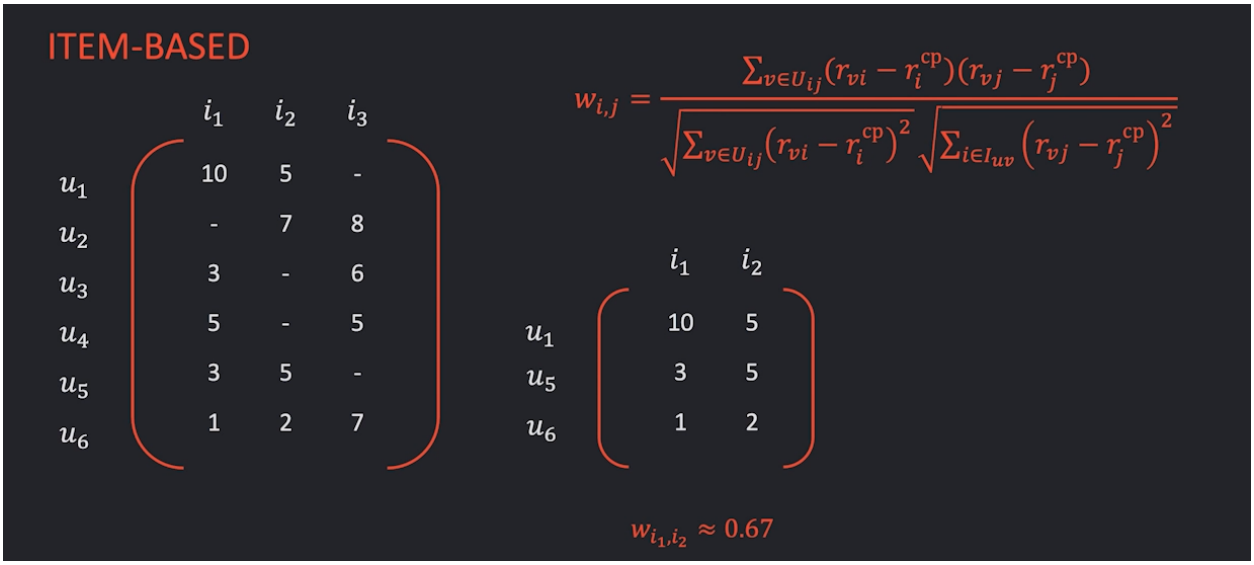
Другой подход, симметричный предыдущему. Основная идея - айтем понравится пользователю, если ему нравятся похожие айтемы. То есть в данном подходе рассматривается матрица оценок разных фильмов пользователем.

Отберем юзеров, у которых есть оценки для обоих айтемов:

$$U_{i,j} = \{v \in U : \exists r_{vi}\ \&\ \exists r_{vj}\}$$

Посчитаем коэффициент корреляции $w_{i,j}$ по оценкам обоих айтемов для юзеров $U_{i,j}$

Здесь используется тот же принцип и та же формула, что использовалась для юзеров выше.



Для каждого айтема i_0 будем отбирать похожих на него по некоторому порогу корреляции

$I(i_0) = j \in I | w_{i_0,j} > \alpha$

Такое множество также будет называться коллаборация

Далее на основании оценок айтемов из $I(i_0)$ можно дать оценку, какой рейтинг выставят айтему i_0 пользователи, еще не давшие оценки.

$I(i_0)_u$ - множество айтемов, которому пользователь u дал какую-то оценку.

Далее также корректируем сдвиги от средней оценки фильма.

>Модель со скрытыми переменными

Третий вариант коллаборативной фильтрации на основе матрицы R.

Идея - закодировать каждый айтем и каждого юзера в виде l-мерного вектора

Обозначим два вектора, для пользователя и айтема соответственно:

$p_u = (d_1^u, \ldots, d_l^u)$

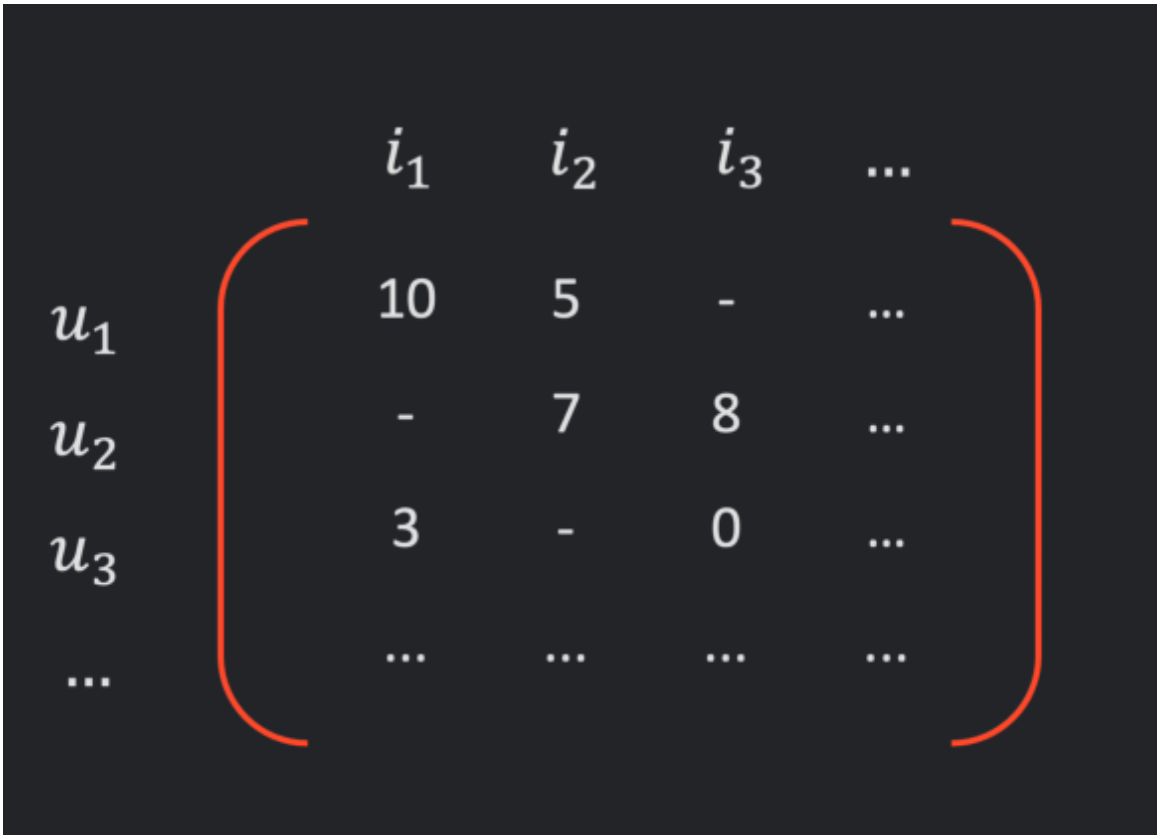
$q_i = (d_i^i, \ldots, d_l^i)$

Таким образом прогноз мождели можно представить в виде скалярного произведения между двумя векторами:

$a(u,i) = \langle p_u, q_i \rangle \approx r_{ui}$

Чтобы сгенерировать данные векторы из матрицы R используется следующая формула:

$\sum_{u,i,r_{ui} \in R} (r_{ui} - b_u - b_i - \langle p_u, q_i \rangle)^2 --> min_{b_u, b_i, p_u, q_i}$



Матрица R

Решая данную задачу получаем две матрицы - для пользователей и айтемов:

$P = (p_1 | \ldots | p_n)_{lxn}$

$Q = (q_1 | \ldots | q_m)_{lxm}$

При этом$(P^T \cdot Q)_{ui} = \langle p_u, q_i \rangle$

По факту строим такие эмбеддинги (кодировки l-мерных признаков), чтобы произведение двух матриц, состоящих из этих кодировок аппроксимировали нам изначальную матрицу R:

$P^T \cdot R \approx R$

Также в подобных задачах часто используется регуляризация.

>Минимизация модели со скрытыми переменными

1. Стохастический градиентный спуск. Позволяет быстро находить оптимумы в сложных задачах. Но на практике дает плохие минимумы.
2. ALS(Alternating least squares)

$\sum_{u,i,r_{ui} \in R} (r_{ui} - b_u - b_i - \langle p_u, q_i \rangle)^2 -- > min_{b_u, b_i, p_u, q_i}$

- Шаг 1: Инициализируем P, Q
- Шаг 2: Фиксируем значения в Q, ищем P:

$\sum_{u,i,r_{ui} \in R} (r_{ui} - b_u - b_i - \langle p_u, q_i \rangle)^2 -- > min_{p_u}$

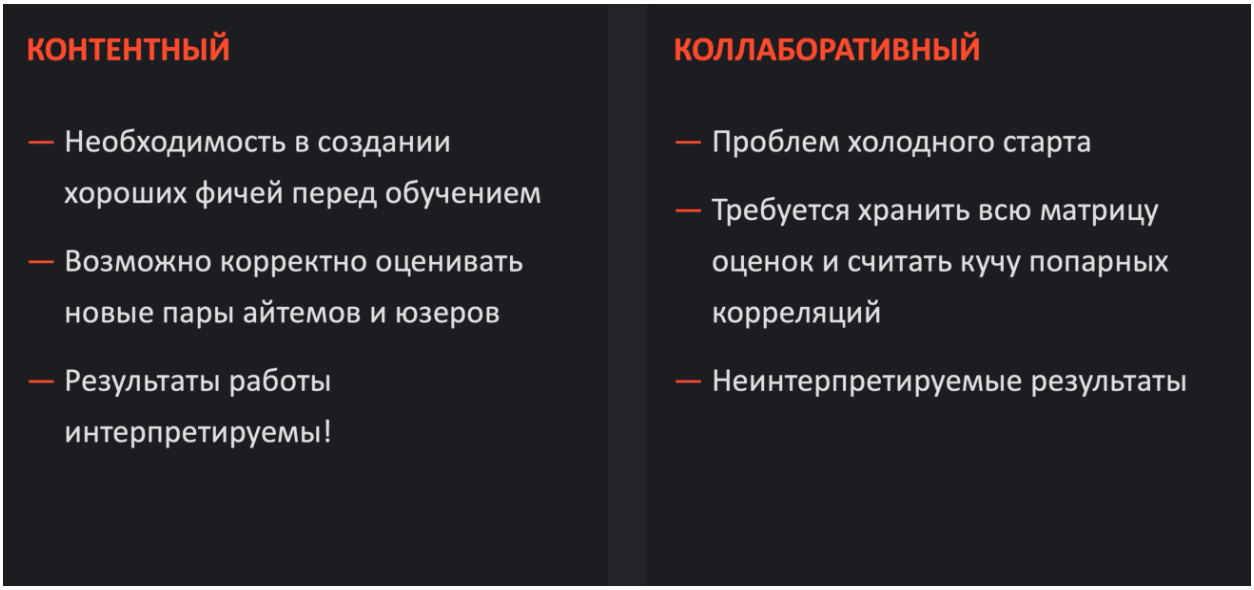
- Шаг 3: После того, как оптимальная матрица P найдена, фиксируем и ищем Q :

$\sum_{u,i,r_{ui} \in R} (r_{ui} - b_u - b_i - \langle p_u, q_i \rangle)^2 -- > min_{q_i}$

-Повторяем шаг 2,3 до сходимости

По сути это то же самое, что мы делали, когда изучали линейную регрессию!

Сравнение контентного и коллаборативного подходов



>Оценка качества работы рекомендательных системы и валидация

В целом для оценки качества работы рекомендательных системы можно использовать классические метрики метрики регрессии или классификации.

Рассмотрим также несколько новых метрик.

$$hitrate@k = [R_u(k) \bigcap L_u \neq \emptyset]$$

$$presicion@k = \frac{|R_u(k) \bigcap L_u|}{K}$$

$$recall@k = \frac{|R_u(k) \bigcap L_u}{L_u},$$

где $R_u(k)$ - топ k рекомендаций пользователю u

L_u - айтемы, которые понравились пользователю

Однако такие метрики обладают существенным недостатком - если первые рекомендации будут плохими, пользователь может проигнорировать оставшиеся рекомендации и прогноз получится провальным.

“Хитрые” метрики

Отсортируем айтемы i для пользователей по выходам модели $a_{ui} = a(u, i)$ и выберем топ k:

$i_1, \dots, i_k : a_{u_1} \geq \dots \geq a_{u_k}$

$DCG@k = \sum_{p=1}^{k} g(r_u i_p) \cdot d(p)$

где p - порядок айтема, на который его поставили в отранжированном списке

$g(r_{ui_p})$ - функция релевантности айтема

$d(p)$ - штраф за позицию

То есть такая модель, в отличие от предыдущих, учитывает то, в какой последовательности даются рекомендации (сначала - наиболее подходящие, затем менее)

Валидация

Есть несколько подходов для валидации рекомендательных систем на исторических данных

Допустим у нас есть данные от том, каким образом в прошлом пользователи оценивали те или иные фильмы/товары.

Можем последний небольшой отрезок времени отрезать в качестве теста, а на остальных обучать модель.