

2. Коллаборативная фильтрация (Collaborative filtering)

Введение



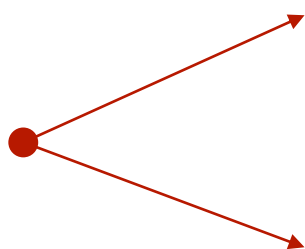
В этом уроке мы подробнее изучим коллаборативную фильтрацию.

Вторая часть урока будет посвящена проблемам, которые часто возникают при работе с коллаборативной фильтрацией – холодный старт и длинный хвост.

План

1

Что такое
коллаборативная
фильтрация?



User-based
подход

Item-based
подход

2

Метод
ближайших
соседей

3

Проблемы

**Что такое коллаборативная
фильтрация?**

Пример 1. Рекомендательная система по посещениям

U – users пользователи

I – items (сайты, страницы, документы, новости и т.п.)

r_{ui} = [пользователь u посетил (провел время) ресурс i]

Основная гипотеза Web Usage Mining:

- Действия (посещения) пользователя характеризуют его интересы, вкусы, привычки, возможности.

Задачи персонализации предложений:

- для пользователя u :
 - выдать *оценку* ресурса i ;
 - выдать ранжированный список рекомендуемых ресурсов;
- для ресурса i : выдать список ресурсов, близких к i .

Пример 2. Рекомендательная система по покупкам

U – users клиенты интернет-магазина

I – items (книги, видео, музыка, и т.п.)

r_{ui} = [клиент u купил (добавил в корзину) товар i]

Задачи персонализации предложений:

- выдать оценку товара i для клиента u
- выдать клиенту u список рекомендуемых товаров
- предложить скидку на совместную покупку (cross-selling)
- информировать клиента о новом товаре (up-selling)
- сегментировать клиентскую базу
- выделить интересы клиентов (найти целевые аудитории).

Пример 3. Рекомендательная система на основе рейтингов



U – users

I – items (книги, видео, музыка, и т.п.)

r_{ui} = рейтинг, который клиент u выставил товару i

Задачи персонализации предложений — те же.

Пример: конкурс Netflix [www.netflixprize.com]

Пример 4. Анализ текстов

U – текстовые документы (статьи, новости и т.п.)

I – ключевые слова и выражения

r_{ui} = частота встречаемости слова i в тексте u

Задачи тематического моделирования (topic modeling):

- кластеризовать тексты по темам
- определить число тем (кластеров) в коллекции;
- построить иерархический каталог тем
- определить, на какие подтемы разбивается каждая тема;
- по тексту u найти тексты близкой тематики; проследить динамику развития темы во времени
- найти публикации, авторов, организации, журналы, конференции по теме.

Пример 5. Анализ изображений



U – изображения

I – найденные на изображениях элементы

r_{ui} = [изображение u содержит элемент i]

Задачи тематического моделирования (topic modeling):

- кластеризовать изображения по темам
- построить иерархический каталог тем
- по изображению u найти схожие

Задачи связывания изображений и текстов:

- аннотировать (тегировать) изображение
- по изображению найти описание
- по описанию найти изображение

Обобщающий пример



U – субъекты

I – объекты

r_{ui} = [измеримая функция описывающая взаимодействие u и i]

Задача:

- Восстановить матрицу R
- Найти близкие («похожие») элементы по u
- Найти близкие («похожие») элементы по i

Пример рейтинга фильмов

	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля			3	6

Тривиальная рекомендательная система



Что первым приходит в голову?

Тривиальная рекомендательная система

Что первым приходит в голову?

1. Популярные объекты (товары, фильмы), «хиты продаж»
2. Недавно покупали, недавно бронировали, «последние»
3. Экспертное сопоставление
4. Самые полезные с точки зрения бизнеса
5. «Брошенные корзины», «предыдущие покупки»
6. Мэтч по «рейтингу/отклику»:
 - «Пользователи покупавшие этот товар также покупают...»
 - «С этим приложением часто устанавливают...»
7. Мэтч по признаку (это обсудим в рамках **content-base**):
 - «Пользователи из вашего города часто выбирают эту гостиницу...»
 - «Пользователи Premium сервиса часто пользуются...»

Популярные



1. Нужно вычислить (и придумать!) некоторую меру популярности.
2. Отсортировать по ней.
3. Profit!

Меры популярности:

1. Средний рейтинг
 - Мат. ожидание
 - Медиана
2. Взвешенный средний
 - По среднему баллу пользователя
 - По число голосов
3. Отсечения

Формула КиноПоиска

Формула расчета рейтинга Топ-250

$$Rating = \left(\frac{V}{V+M} \right) \times R + \left(\frac{M}{V+M} \right) \times C$$

V – количество голосов за фильм

M – порог голосов, необходимый для участия в рейтинге Топ-250 (сейчас: **500**)

R – среднее арифметическое всех голосов за фильм

C – среднее значение рейтинга всех фильмов (сейчас: **7.1382**)

Рейтинг фильма в Топ-250 отличается от рейтинга на странице фильма. Это происходит потому, что в топе используются специальные механизмы, препятствующие накрутке рейтинга недобросовестными пользователями.

<https://www.kinopoisk.ru/media/news/4003304/>

Недавние покупки/смотрели



1. Выбираем порог отсечения по времени.
2. Считаем частоту (возможно с учетом рейтинга).
3. Сортируем по ней.
4. Profit!

Экспертное сопоставление



- Часто хорошие пары товаров известны: наушники и чехлы к телефону, колонки к ресиверу, чипсы к пиву, картриджи к принтеру и т.д.
- Аналогично к фильмам: Гарри Поттер 1 + Гарри Поттер 2 и т.д.

Алгоритм:

1. Просим эксперта сделать сопоставление руками
2. Profit!

Для небольших интернет-магазинов это достаточно сильный baseline.

Самый ценный с точки зрения бизнеса



1. Выбираем метрику (прибыль)
2. Сортируем объекты по ней
3. Profit!

Про бизнес-метрики будем чуть позже говорить.

Брошенные корзины, последние покупки

- Часто в ретейле пользователи покупают одно и тоже.
- Часто пользователи в стриминговых платформах смотрят/играют в одно и тоже.
- Для рекомендации книг/кино это НЕ подойдет

Алгоритм:

1. Для каждого пользователя посчитать популярность (давность) действия
2. Отсортировать и сделать персональную рекомендацию.
3. Profit!

Это тоже достаточно сильный baseline.

Мэтч по рейтингу/отклику

Сделаем, персонализацию, которую использует знания о других пользователях.

Клиенты, которые покупают i_0 , также покупали $I(i_0)$

Алгоритм для пользователя u_0 :

1. $U(i_0) := \{u \in U \mid r_{ui_0} \neq \emptyset, u \neq u_0\}$ — выбираем множество всех пользователей, которые покупали/рейтинговали товар i_0 .
2. $I(i_0) := \{i \in I \mid \text{sim}(i, i_0) > s\}$ - находим все товары близкие к i_0 по мере sim на множестве $U(i_0)$.
Можно использовать «отсечение» s .
3. Отсортировать $I(i_0)$ по убыванию $\text{sim}(i, i_0)$
4. Profit!

Мэтч по рейтингу/отклику

Как выбрать sim ?

- «Классический подход»: $sim(i, i_o) = \frac{|U(i_o) \cap U(i)|}{|U(i_o) \cup U(i)|}$
- Можно придумать что-то посложнее

Недостатки:

- Рекомендации тривиальны
- Не учитываются интересы конкретного пользователя u_o (а только одна покупка/отклик)
- Надо хранить всю матрицу R

Преимущества:

- Для отдельных групп товаров сильный baseline

Два основных подхода в коллаборативной фильтрации



Memory-Based Collaborative Filtering

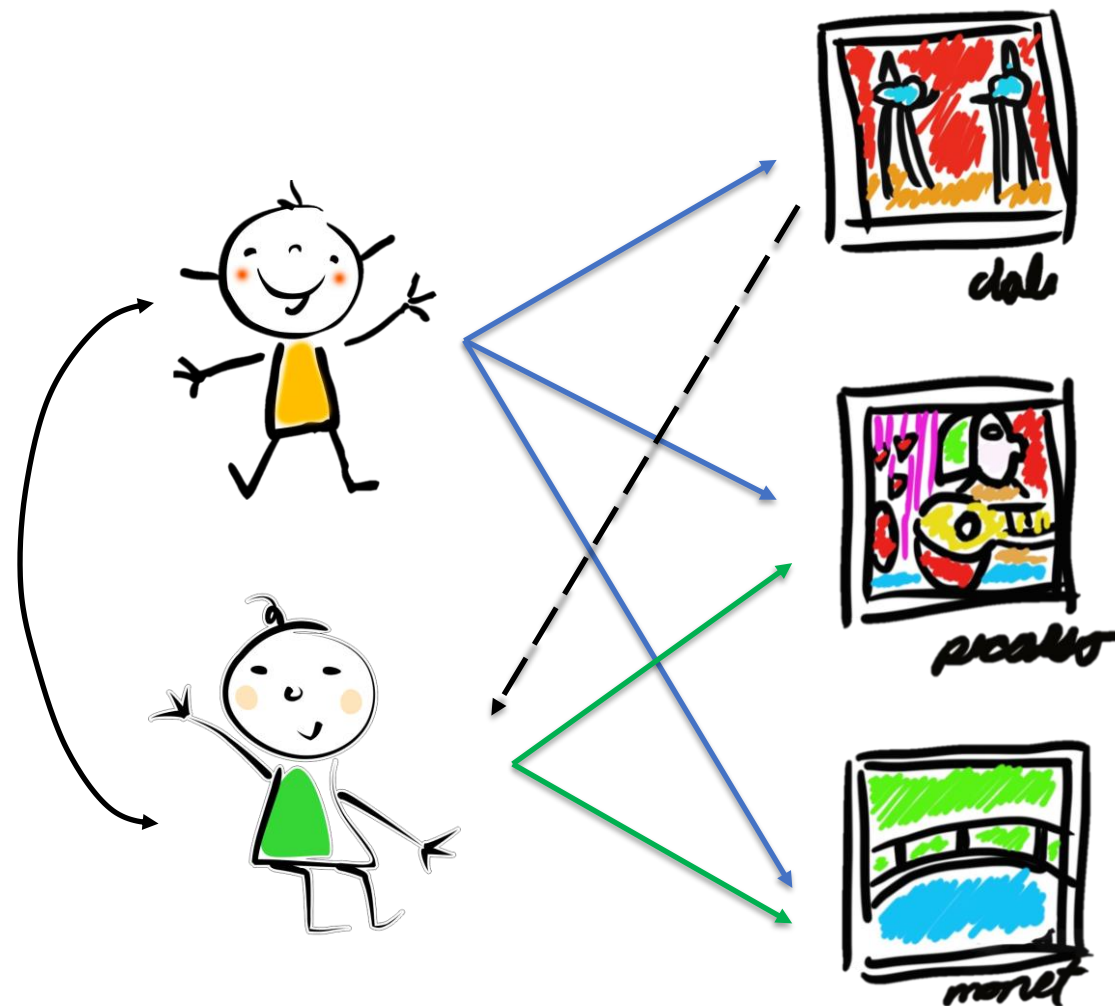
- хранение всей исходной матрицы данных R
- сходство клиентов — это корреляция* строк матрицы R
- сходство объектов — это корреляция* столбцов матрицы R

Latent Models for Collaborative Filtering

- оценивание профилей клиентов и объектов (профиль — это вектор скрытых характеристик). Получение эмбедингов.
- хранение профилей вместо хранения R
- сходство клиентов и объектов — это сходство их профилей

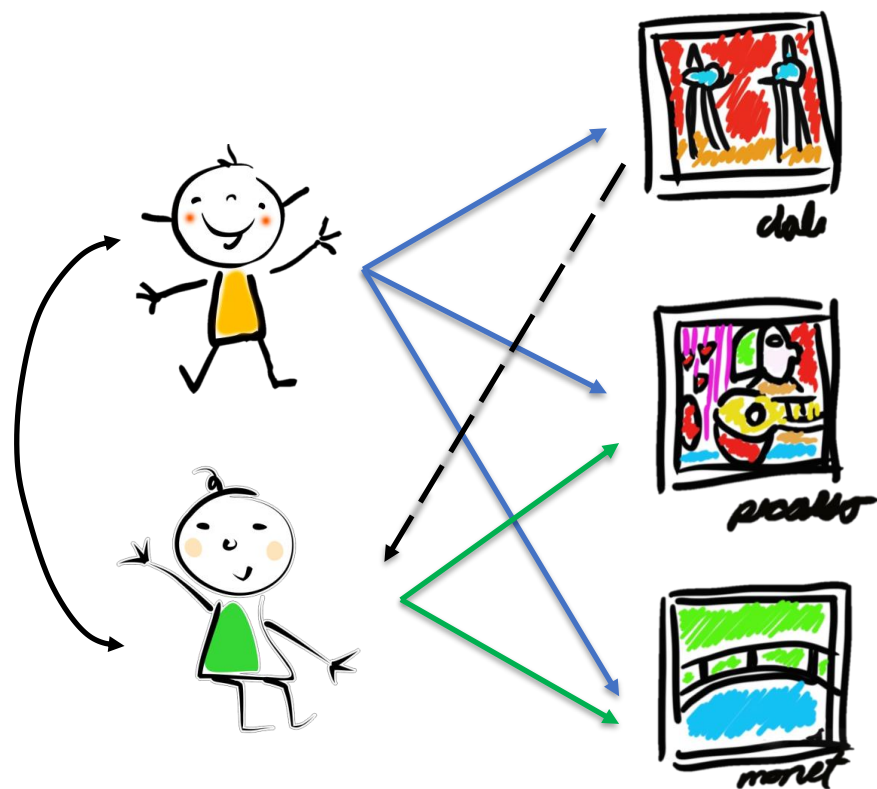
Коллаборативная фильтрация

Рекомендации для пользователя строятся на основе **оценок похожих пользователей**.

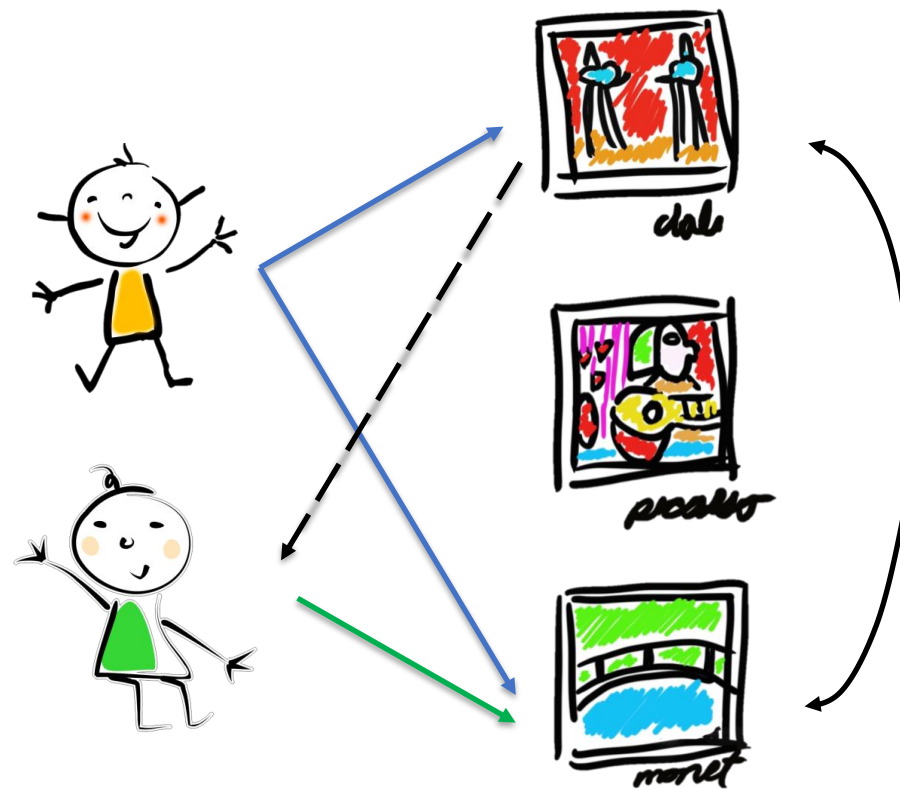


Коллаборативная фильтрация

1 User-based



2 Item-based



User-based подход



Алгоритм:

1. Найдем пользователей похожих на u_0
2. Среди этих пользователей найдем наиболее популярные (рейтинговые) товары
3. Сортируем по рейтингу.
4. Profit!

User-based

«клиенты, похожие на u_0 , также покупали $I(u_0)$ »

1 $U(u_0) := \{u \in U \mid \text{corr}(u_0, u) > a\}$ — коллаборация, где $\text{corr}(u_0, u)$ одна из возможных мер близости u к u_0 .
 a порог отсечения (параметр).

2 $I(u_0) := \left\{ i \in I \mid B(i) = \frac{|U(u_0) \cap U(i)|}{|U(u_0) \cup U(i)|} > 0 \right\}$,
где $U(i) := \{u \in U \mid r_{ui} \neq \emptyset\}$

3 отсортировать $i \in I(u_0)$ по убыванию $B(i)$

User-based подход



Недостатки:


- — рекомендации тривиальны
- — не учитываются интересы конкретного пользователя u_g
- надо хранить всю матрицу R
- нечего рекомендовать нетипичным/новым пользователям

Item-based подход



- 1** $I(u_0) := \{i \in I \mid \exists i_0: r_{u_0 i_0} \neq \emptyset \text{ и } B(r) = \text{sim}(i_0, i) > b\}$, где $\text{sim}(i_0, i)$ - одна из возможных мер близости к i к i_0
- 2** сортировка $i \in I(u_0)$ по убыванию $B(i)$

Item-based



Недостатки:

- рекомендации тривиальны (нет коллаборативности)
- надо хранить всю матрицу R
- — нечего рекомендовать нетипичным пользователям

Восстановление пропущенных значений (рейтингов)

Непараметрическая регрессия Надарайя-Ватсона:

$$\widehat{r_{ui}} = \bar{r}_u + \frac{\sum_{u' \in U_a(u)} K(u, u') (r_{u'i} - \bar{r}_{u'})}{\sum_{u' \in U_a(u)} K(u, u')}$$

где $\bar{r}_u = \frac{1}{|I(u)|} \sum_{i \in I(u)} r_{ui}$ - средний рейтинг пользователя u

$I(u)$ – множество объектов, которые пользователь u оценил

$K(u, u')$ – сглаживающее ядро (или мера близости u и u')

$U_a(u) = \{u' \mid K(u', u) > a\}$ – коллаборация клиента u (с параметром a)

Недостатки:

- проблема «холодного старта»
- надо хранить всю матрицу R

Восстановление пропущенных значений (рейтингов)



Недостатки:

- проблема «холодного старта»
- надо хранить всю матрицу R

Преимущества:

- «Шкалирует»
- Сглаживает

Резюме по Memory-based моделям



Преимущества для бизнес-приложений:

- Легко понять
- Легко реализовать

Недостатки:

- Не хватает теоретического обоснования:
 - придумано много способов оценить сходство. . .
 - придумано много гибридных (item-user-based) методов. . .
 - . . . и не ясно, что лучше
- Все методы требуют хранения огромной матрицы R
- Проблема «холодного старта»

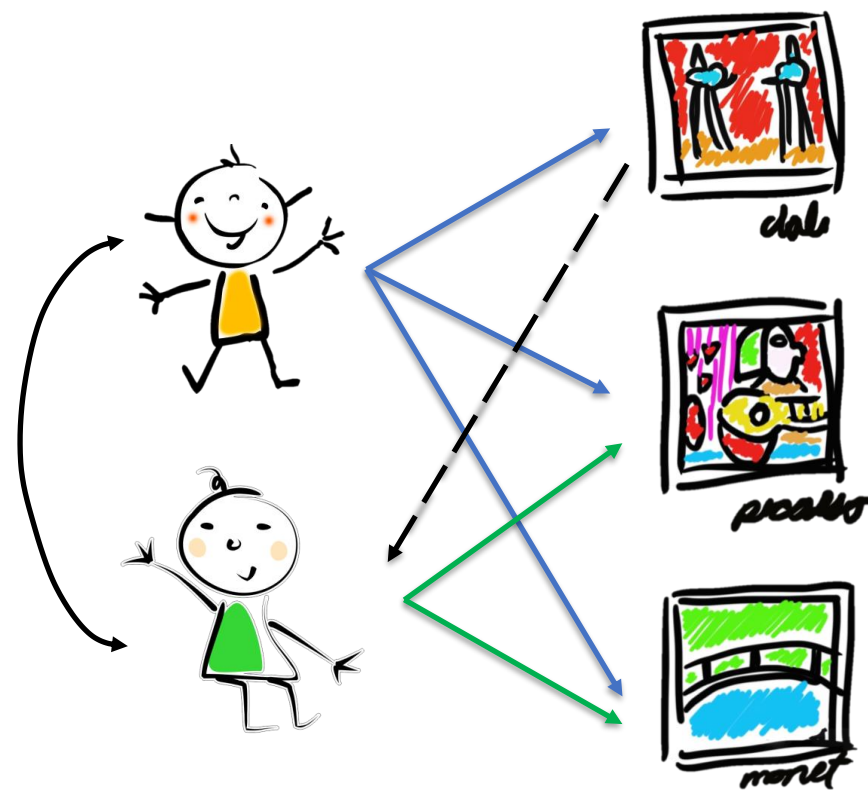
Далее:

Латентные модели — лишены этих недостатков (но об этом в следующий раз 😊)

Метод ближайших соседей

User-based коллаборативная фильтрация

Суть заключается в выборе подмножества пользователей на основе их сходства, после чего взвешенная комбинация из рейтингов используется для предсказания рейтинга, который поставит отдельный пользователь.

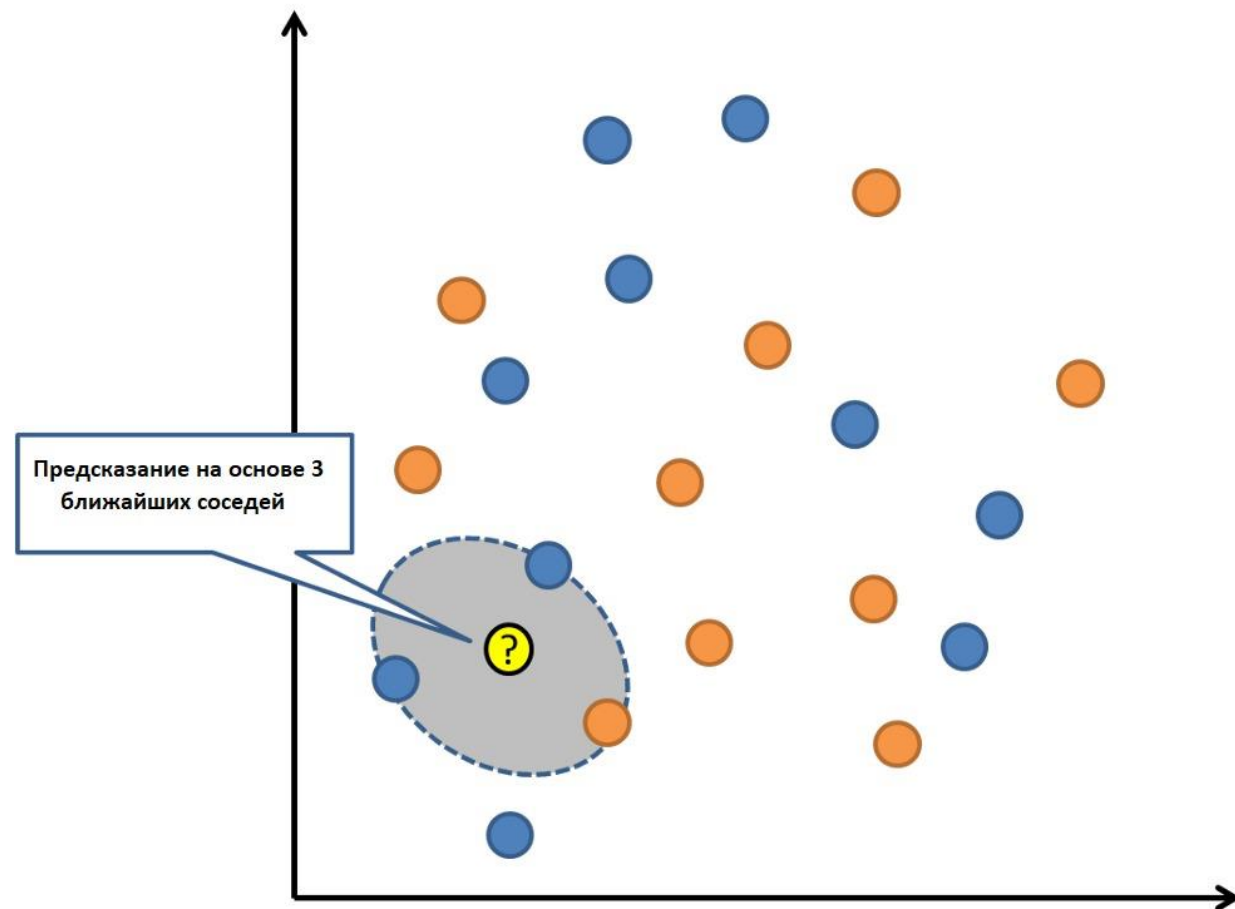


Метод ближайших соседей (kNN)

kNN (k Nearest Neighbor или k Ближайших Соседей)

Метод ближайших соседей:

Давайте все-таки введём расстояние между пользователями и будем рекомендовать то, что нравится k-соседям.



User-based kNN

	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля			3	6

User-based kNN



	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля			3	6

User-based kNN

	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	?
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля			3	6

User-based kNN

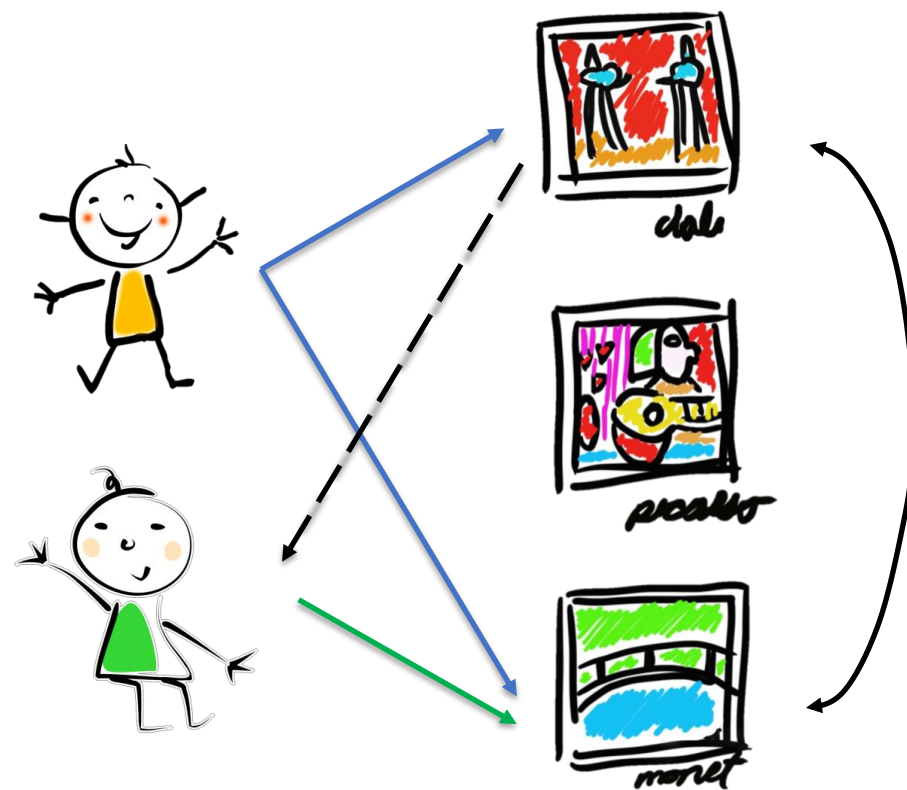
	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	?
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля			3	6

2 ближайших
соседа по
предпочтениям

Следовательно, оценка Бориса
на фильм «Легенда №17»
будет где-то 6,5

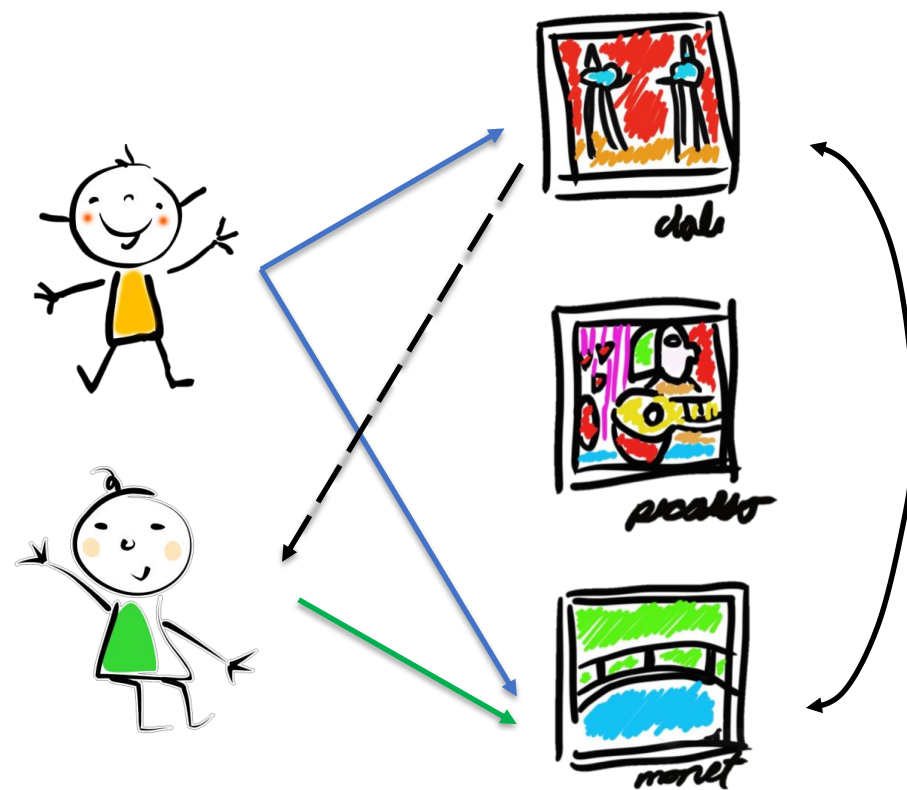
Item-based коллаборативная фильтрация

По мере роста системы количество пользователей увеличивается, и соответственно сложность поиска похожих пользователей возрастает. Потому был предложен новый подход, который вместо похожих пользователей ищет **похожие элементы**.



Item-based коллаборативная фильтрация

Подход предполагает, что сходство между пользователями и элементами вызвано некоторой скрытой низкоразмерной структурой данных. В данном подходе одна предварительная модель разрабатывается на основе имеющихся данных. Когда появляется запрос от пользователя, этот подход дает быстрый ответ о предпочтениях пользователя.



Item-based kNN

	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	?
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля			3	6

Item-based kNN

	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	?
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля			3	6

Как оценить близость соседей?

Ключевым в алгоритме kNN – является расстояние (близость). От того как ее задать зависит результат.

Примеры, расстояний:

1. Число совпавших оценок
2. Корреляция Пирсона
3. Косинусную расстояние



Корреляция Пирсона

Корреляция Пирсона — классический коэффициент, который вполне применим и при сравнении векторов.

Основной его минус — когда пересечение по оценкам низкое, корреляция может быть высокой просто случайно.

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Косинусное расстояние

Основная идея, на которой базируется расчет косинусного расстояния, заключается в том, что строку из символов можно преобразовать в числовой вектор. Если проделать эту процедуру с двумя сравниваемыми строками, то меру их сходства можно оценить через косинус между двумя числовыми векторами.

$$similarity = \cos(\theta) = \frac{XY}{\|X\| \|Y\|} = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2}}$$

Из курса школьной математики известно, что если угол между векторами равен 0 (то есть векторы полностью совпадают), то косинус равен 1.

Пример расчета

	Item 1	Item 2	Item 3	Item 4	Средн.
User 1	4	4	5	3	4
User 2	2	1	3	1	1,75

Пирсон

Косинус

$$\frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2 \sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}}$$
$$\rho = \frac{0 + 0 + 1 * 1,25 - 1 * (-0,75)}{\sqrt{(1^2 + 1^2) * (0,25^2 + 0,75^2 + 1,25^2 + 0,75^2)}}$$
$$= \frac{2}{\sqrt{5,5}} \approx 0,85$$

$$\frac{\sum_{i=1}^m r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^m r_{a,i}^2 \sum_{i=1}^m r_{u,i}^2}}$$
$$\cos(\theta) = \frac{4 * 2 + 4 * 1 + 5 * 3 + 3 * 1}{\sqrt{4^2 + 4^2 + 5^2 + 3^2} * \sqrt{2^2 + 1^2 + 3^2 + 1^2}}$$
$$= \frac{30}{\sqrt{66}} \approx 0,95$$

Пример расчета

	Item 1	Item 2	Item 3	Item 4	Средн.
User 1	4	?	5	?	
User 2	4	1	?	1	

Пирсон

$$\frac{\sum_{i \in I} (r_{a,i} - \bar{r}_a) * (r_{u,i} - \bar{r}_u)}{\sqrt{\sum_{i \in I} (r_{a,i} - \bar{r}_a)^2 \sum_{i \in I} (r_{u,i} - \bar{r}_u)^2}}$$
$$\rho = \frac{0 + 0 + 1 * 1,25 - 1 * (-0,75)}{\sqrt{(1^2 + 1^2) * (0,25^2 + 0,75^2 + 1,25^2 + 0,75^2)}}$$
$$= \frac{2}{\sqrt{5,5}} \approx 0,85$$

Косинус

$$\frac{\sum_{i=1}^m r_{a,i} r_{u,i}}{\sqrt{\sum_{i=1}^m r_{a,i}^2 \sum_{i=1}^m r_{u,i}^2}}$$
$$\cos(\theta) = \frac{4 * 2 + 4 * 1 + 5 * 3 + 3 * 1}{\sqrt{4^2 + 4^2 + 5^2 + 3^2} * \sqrt{2^2 + 1^2 + 3^2 + 1^2}}$$
$$= \frac{30}{\sqrt{66}} \approx 0,95$$

Как оценить близость соседей?

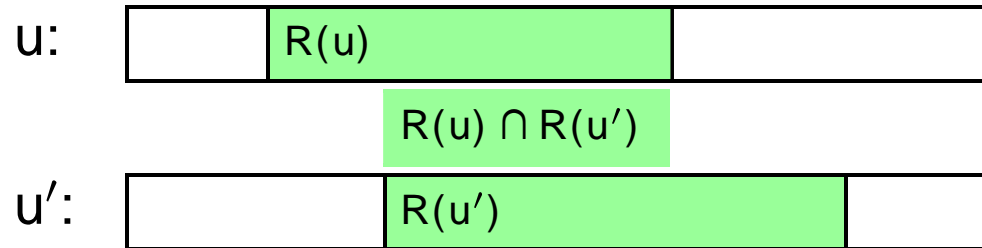


Есть много других метрик.

Давайте придумаем «свои» метрики.

Функция близости на основе точного теста Фишера (FET)

Пользователи u и u' совершают свой выбор независимо.
Какова вероятность того, что пересечение их оценок оказалось случайным?



Вероятность случайной реализации r совместных выборов

$$p(i) = P\{|I(u) \cap I(u')| = i\} = \frac{C_{|I(u)|}^i C_{|I|-|I(u)|}^{|I(u')|-i}}{C_{|I|}^{|I(u')|}}$$

Функция близости $K(u, u') = -\log p(|I(u) \cap I(u')|)$

3. Проблемы

Проблема холодного старта

Проблема холодного старта

Новый пользователь еще не успел проявить активность в системе и о нём практически ничего неизвестно, а новый объект еще никто не оценил. Возникает проблема “холодного старта”.

Решение проблемы:

- Использовать средние оценки или взвешенное среднее всех "предсказанных" результатов и т.п.
- Создавать специальную сущность «новый пользователь» и строим оценки для нее.
- Применяется принцип суперпозиции рекомендательных систем или гибридные рекомендательные системы (см. следующие лекции).



Холодный старт

	1+1	Три мушкетер а	12 стульев	Легенда №17
Алексей	10	9	1	7
Борис		9	2	
Вова	1		6	
Коля	3		4	10
Петя		1		
Юля				

Что рекомендовать Юле?

Проблема холодного старта



Проблема холодного старта по пользователю

возникает при обнаружении
в системе новых или редко
появляющихся пользователей

Проблема холодного старта по объекту

возникает при наличии новых
объектов либо таких, *свойства*
которых могут измениться
со временем.

Что значит меняются свойства?



Пример 1. Сотовый телефон.

Никакие его физические характеристики не меняются. Тем не менее 1гб оперативной памяти в 2010 году не тоже самое, что в 2020.

То есть свойство «осталось на месте», но отношение к нему изменилось.

Что значит меняются свойства?



Пример 1. Сотовый телефон.

Никакие его физические характеристики не меняются. Тем не менее 1гб оперативной памяти в 2010 году не тоже самое, что в 2020.

То есть свойство «осталось на месте», но отношение к нему изменилось.

Пример 2. Одежда.

Цвет одежды не меняется. Но меняются тренды.

Что значит меняются свойства?



Пример 1. Сотовый телефон.

Никакие его физические характеристики не меняются. Тем не менее 1гб оперативной памяти в 2010 году не тоже самое, что в 2020.

То есть свойство «осталось на месте», но отношение к нему изменилось.

Пример 2. Одежда.

Цвет одежды не меняется. Но меняются тренды.

Нужно иметь какой-то инвариант свойств (нормировку относительно времени/пространства и т.д.), что на практике почти не возможно.

Проблема холодного старта



Проблема холодного старта по пользователю

возникает при обнаружении
в системе новых или редко
появляющихся пользователей

Лучшее решение:

- метод «контекстного многорукого бандита»
- алгоритм для решения проблемы пользовательской волатильности

Проблема холодного старта по объекту

возникает при наличии новых
объектов либо таких, свойства
которых могут измениться
со временем.

Лучшее решение:

комбинирование коллаборативной
фильтрации и контентных методов
при предположении, что «похожим»
пользователям понравятся
«похожие объекты»

Как валидировать проблему холодного старта?

Валидировать проблему «холодного старта» оффлайн довольно сложно, поэтому самый простой вариант ее тестировать онлайн на реальных пользователях и смотреть какой результат получится.



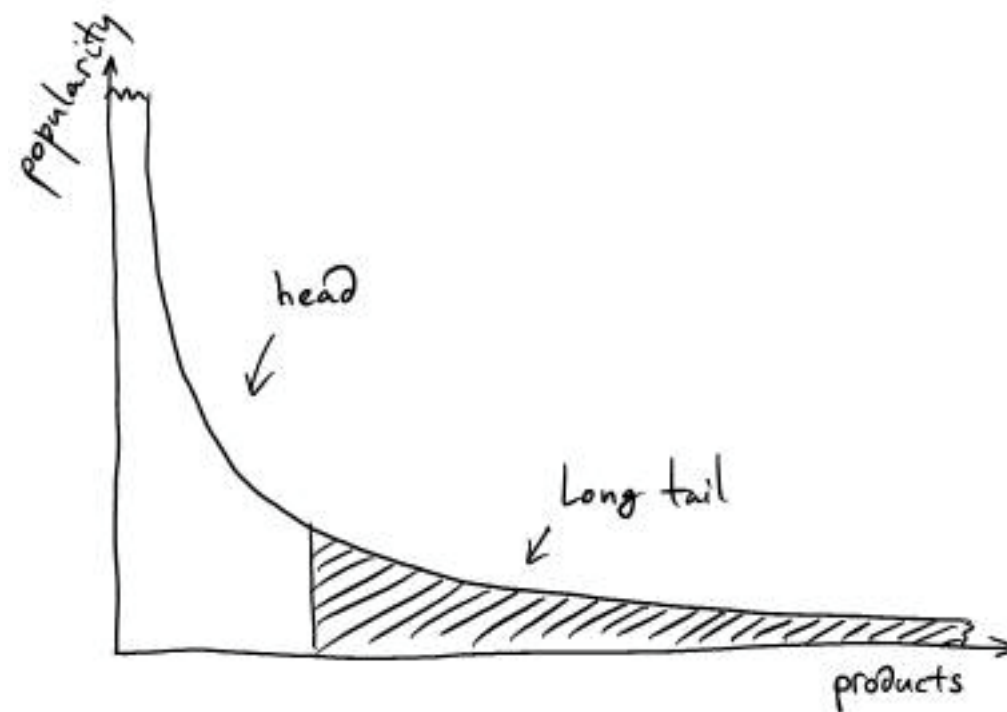
Проблема длинного хвоста

Проблема длинного хвоста

Длинный хвост — розничная концепция, описывающая явление больших суммарных продаж товаров, ставших в своё время классикой, по сравнению с товарами, которые в настоящее время считаются модными.

Концепцию «Длинный хвост» впервые сформулировал Крис Андерсон.

Одно из решений - объединять товары, например, для которых мало статистики, в кластеры или в какую-то категорию.



Проблема длинного хвоста



Давайте обсудим эту проблему, для

1. Фильмов

2. Товаров

Проблема первых оценщиков

Проблема первых оценщиков



Пусть в наших сервис приходят первые оценщики, которые сильно отличаются от «среднего». Из-за этого мы начинаем делать плохие рекомендации, и пользователи не узнают, что у нас есть хорошие контент.

Что делать?

Заключение



- Коллаборативная фильтрация – наиболее используемый вид рекомендательной системы
- Однако, часто компаниям приходится бороться с проблемами «длинного» хвоста и «холодного» старта

Вопросы

Семинар knn рекомендациям