Вопрос №1.

1. Какие показатели используются для оценки рекомендательных систем?

2. Метрики точности прогнозов (MAE, RMSE).

3. Показатели Top-N с учетом ранжирования (MRR, nDCG).

4. Пользовательская коллаборативная фильтрация.

5. Коллаборативная фильтрация по элементам.

6. Контентная рекомендательная система.

7. Экспертные рекомендательные системы.

8. Проблема холодного старта в рекомендательных системах.

9. Какие методы можно использовать для заполнения пропущенных данных?

10. В каких случаях используют MSE, а в каких MAE?

11. Что такое векторизация TF/IDF?

12. Типы рекомендательных систем.

13. Неперсонализированные рекомендации.

14. Коэффициент корреляции Пирсона на коэффициент Отиаи.

15. Совместная фильтрация и матрица оценок.

16. Примеры успешной реализации бизнес-моделей, основанных на использовании рекомендательных систем.

17. Недостатки неперсонализированных и слабо-персонализированных систем.

18. Способы получения и обработки информации о предпочтениях пользователя.

19. Алгоритм Slope One.

Вопрос 2.

1. Реализация рекомендательной системы на основе подбора элементов.

2. Метод k-средних. Как выбрать число k для алгоритма кластеризации?

3. Как работает алгоритм Funk SVD?

4. Как работает алгоритм SVD++?

5. Постановка задачи рекомендательной системы.

6. Гибридные рекомендательные системы.

7. SVD-разложение матрицы градиентным спуском.

8. Решение проблемы холодного старта алгоритмами.

9. Алгоритм кластеризации k-средних.

10. Процедура совместной фильтрации.

11. Факторизации с использованием SVD.

12. Как формировать рекомендации с помощью SVD.

13. Построение факторизации с помощью Funk SVD.

14. Генерация рекомендаций с помощью Funk SVD.

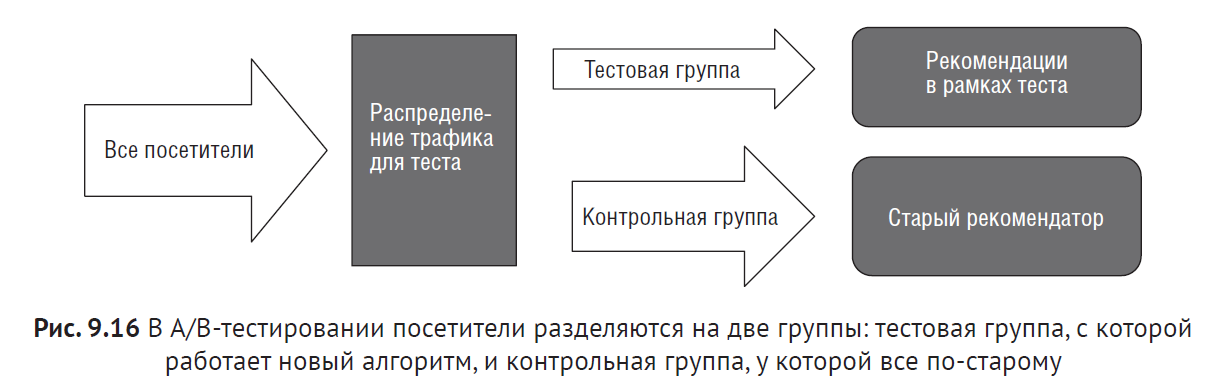
15. Построение и актуализация профилей пользователей, взвешивание/нормализация рейтингов.

16. Алгоритм «User-User» для построения рекомендации на основе оценок пользователей со схожими рейтингами.

17. Алгоритм «Item-Item»: рекомендации схожих товаров на основе связей между их рейтингами.

18. Двухшаговая реализация алгоритма «Item-Item»: определение сходства товаров на основе их рейтингов.

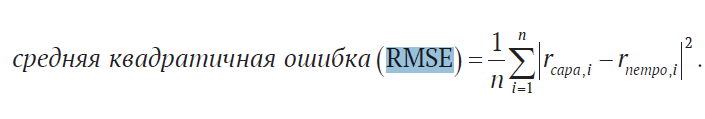
* 1. Оффлайн-эксперимент, мы используем обучающую и тестовую выборку, а затем оцениваем алгоритм с помощью MAE или MSE. Также можно оценивать эффективность ранжирования MRR nDCG. Также можно провести A/B тестирование.



1.2, 1.10

MAE – средняя абсолютная ошибка. R – оценка, p-прогноз.





MSE дает большие штрафы за большие ошибки и таким образом одна

большая ошибка будет значить гораздо больше, чем несколько маленьких.

В то же время при использовании MAE большие ошибки в меньшей степени

влияют на результат. Если нужно, чтобы ни один из пользователей не получал

плохие рекомендации, нужно использовать MSE. Но если вы понимаете, что

всем не угодишь, то, вероятно, достаточно использовать MAE.

Изображение выглядит как стол

Автоматически созданное описание

1.3

Изображение выглядит как текст

Автоматически созданное описание

**Средний взаимный ранг** — это [статистическая](https://en.wikipedia.org/wiki/Statistic) мера для оценки любого процесса, который создает список возможных ответов на выборку запросов, упорядоченных по вероятности правильности. Обратный ранг ответа на запрос является [мультипликативным, обратным](https://en.wikipedia.org/wiki/Multiplicative_inverse) рангу первого правильного ответа: 1 для первого места, 1/2 для второго места, 1/3 для третьего места и так далее. Средний взаимный ранг — это среднее взаимных рангов результатов для выборки запросов Q

Изображение выглядит как текст, часы

Автоматически созданное описание

Например, предположим, что у нас есть следующие три примера запросов для системы, которая пытается перевести английские слова во множественное число. В каждом случае система делает три предположения, первое из которых, по ее мнению, является наиболее верным:

Изображение выглядит как стол

Автоматически созданное описание

Учитывая эти три выборки, мы могли бы рассчитать средний обратный ранг как (1/3 + 1/2 + 1)/3 = 11/18 или около 0,61.

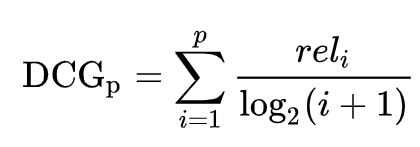
**Дисконтированный кумулятивный выигрыш** ( **DCG** ) является мерой качества ранжирования. В [информационном поиске](https://en.wikipedia.org/wiki/Information_retrieval) он часто используется для измерения эффективности [алгоритмов](https://en.wikipedia.org/wiki/Algorithm)[поисковых](https://en.wikipedia.org/wiki/World_Wide_Web)[систем](https://en.wikipedia.org/wiki/Search_engine) или связанных с ними приложений. Используя [градуированную шкалу релевантности](https://en.wikipedia.org/wiki/Relevance_(information_retrieval)) документов в наборе результатов поисковой системы, DCG измеряет полезность или *выгоду* документа на основе его позиции в списке результатов. Прибыль накапливается от верхней части списка результатов к нижней, при этом прибыль каждого результата дисконтируется на более низких рангах. [[1]](https://en.wikipedia.org/wiki/Discounted_cumulative_gain#cite_note-jarvelin_2002_cumulated-1)

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1.5

Наша задача – предсказать неизвестные элементы матрицы; если быть точным, наша задача – предсказать, какие из неизвестных элементов будут максимальными в своей строке, то есть какие продукты больше всего понравятся тому или иному пользователю.

1.6

Получаем метаданные о фильмах. Создаём словарь слов. С помощью LDA или TF-IDF ищем похожие фильмы.

## Контентная фильтрация

1. Получаем описания фильмов и другие метаданные оф ильмах (год съёмки, название)
2. Объединяем описание фильма, название фильмы и другие метаданные в одну строку (строка описания)
3. Разбиваем строку на слова
4. Удаляем незначимые слова (предлоги например)
5. Выполняем морфологическую обработку (чтобы слова супергеройский и супергерой считались одинаковыми, здесь это не реализовано)
6. С помощью алгоритма LDA получаем схожесть между строками описания (а также между фильмами)
7. Получаем топ фильмов по оценкам данного пользователя
8. Ищем фильмы которые имеют высокое соответствие с оценёнными фильмами пользователя
9. Предсказываем оценку этим фильмам изходя из оценки пользователя текущего фильма (фильмы для которого ищутся похожие фильмы), средней оценки пользователя и схожести фильмов.
10. Возвращаем топ по предсказанной оценке.

1.11

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст, человек, снимок экрана, документ

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1.13

**Неперсонализированные** **рекомендации** — **рекомендации** без учета личных предпочтений пользователя, когда система сортирует по средней оценке рейтинга популярные товары или фильмы.

Неперсонализировнные рекомендации – это топ 10, а также выборочные рекомендации, вместе с этим покупают то.

1.18

1.Явные оценки – попросить пользователя дать несколько оценок.

2. Неявные оценки – программное отслеживание пользователя: перемещение курсора мыши, направление взгляда, история поиска, анализ соц.сетей.

Обработка: Каждому действию добавляется коэффициент и вычисляется предполагаемая оценка пользователя элементу, можно использовать временное затухание чтобы начислять новым элементам больше баллов, также можно добавлять баллы редкими элементам, потому что они лучше показывают вкусы пользователя.

2.2

Метод локтя

Метод силуэтов

**Преимущества K-Means**

* Простота реализации
* Масштабируемость до огромных наборов данных
* Метод очень быстро обучается на новых примерах
* Поддержка сложных форм и размеров.

**Недостатки K-Means**

* Чувствительность к выбросам
* Трудоемкость выбора k
* Уменьшение масштабируемости.

2.3

У каждого пользователя есть оценки-выбросы, они мешают понять его истинные вкусы.

Истинные вкусы – это как линия на графика, а оценки пользователя это облако точек. Отбросив часть точек мы получим линию.

Основная идея за-

ключается в том, что, глядя на данные о поведении пользователей, вы можете

найти категории или темы, которые могут объяснить вкусы пользователей менее

точно, чем по единичным фильмам, но более точно, чем по крупным жанрам.

Эти факторы позволяют нам расположить подобные фильмы ближе друг к другу.

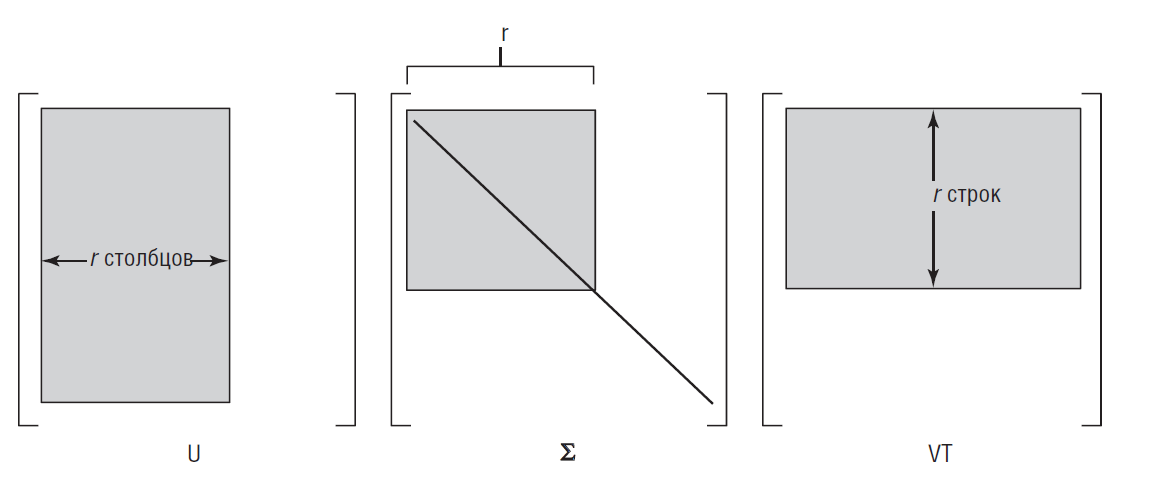
Алгоритм позволяющий сделать это называется SVD.

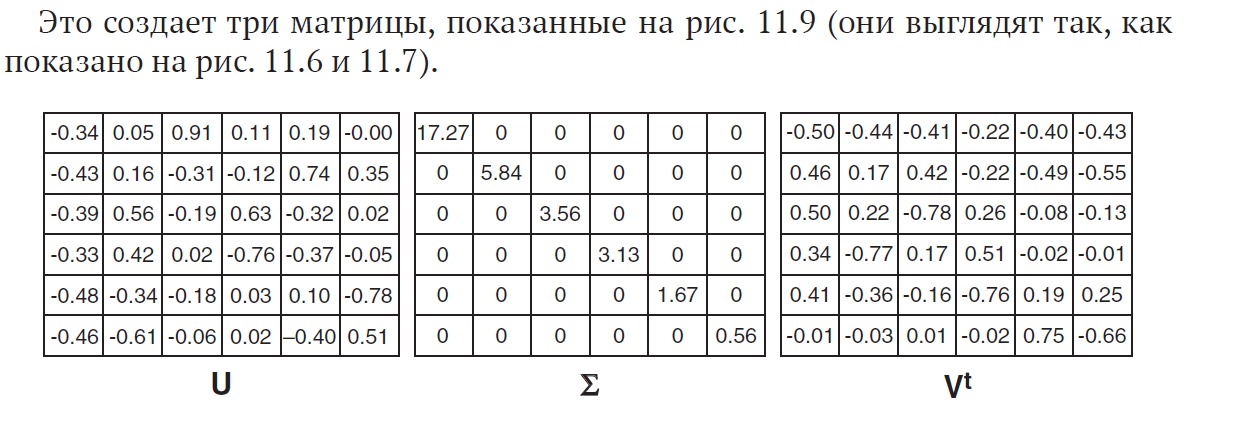
Изображение выглядит как текст

Автоматически созданное описание

SVD возвращает матрицу U (вкусы пользователей,матрицу сигма необходимую для отброски шумов, матрицу vt – обозначающаю профили элементов.

На вход SVD принмают матрицу оценок M.





Сумма всех весов равна 32,0, и 90 % от этого будет равняться 28,83. Если

сократить матрицу до четырех измерений, полученный вес будет равен 29,80,

так что четыре измерения – то что нужно. Чтобы сократить матрицы в коде,

выполните следующий листинг.def rank\_k(k):

## FunkSVD

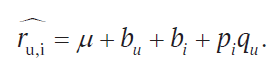
У алгортима SVD есть два серьёзных минуса:

1. Он медленный, а значит чтобы применить к реальной матрице пользователь-элемент в которой будут тычсичи (если не миллионы) строк и колонок, будет почти невозможно.
2. Проблема разряженности, в реальности у каждого пользователя будет максимум 1% оцененных товаров, и заполнять остальные товары среднеми значениями (используя даже продвинутую технику базисных предикатов) не очень правильно.

Чтобы решить эту проблем был разработан FunkSVD. FunkSVD - принимает на вход не матрицу пользователей-товаров.А список всех оценок, то есть он принимает только выставленные оценки, и ему не нужно заполнять пропуски. Во вторых мы работаем со списком, а не с бесконечной матрицей, это должно повысить производительность.

Метод FunkSVD возвращает: общее смещение, смещение пользователя, смещение элемента, факторы пользователей (q) и факторы элементов (p)

Прогноз оценки рассчитывается так:



Pi – это факторы элементы i

Qu – это факторы пользователя u

Pi\*qu – это число

Bu – смещение пользователя

Bi – смещение элемента

μ – общее смещение

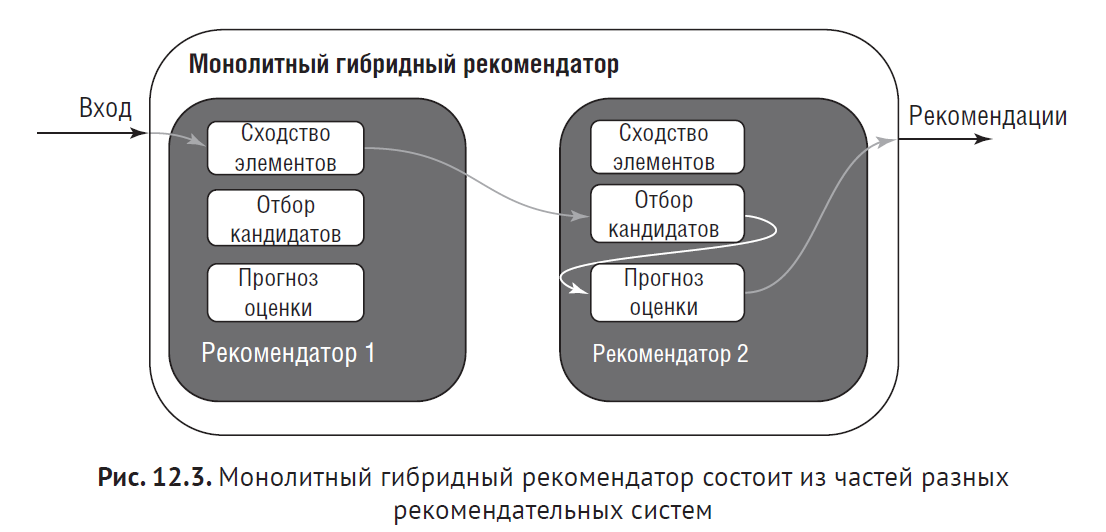
Метод FunkSVD,

Выполняет градиентный спуск чтобы найти факторы пользователя и элементов. Каждый новый фактор рассчитывается на основе всех предыдущих.

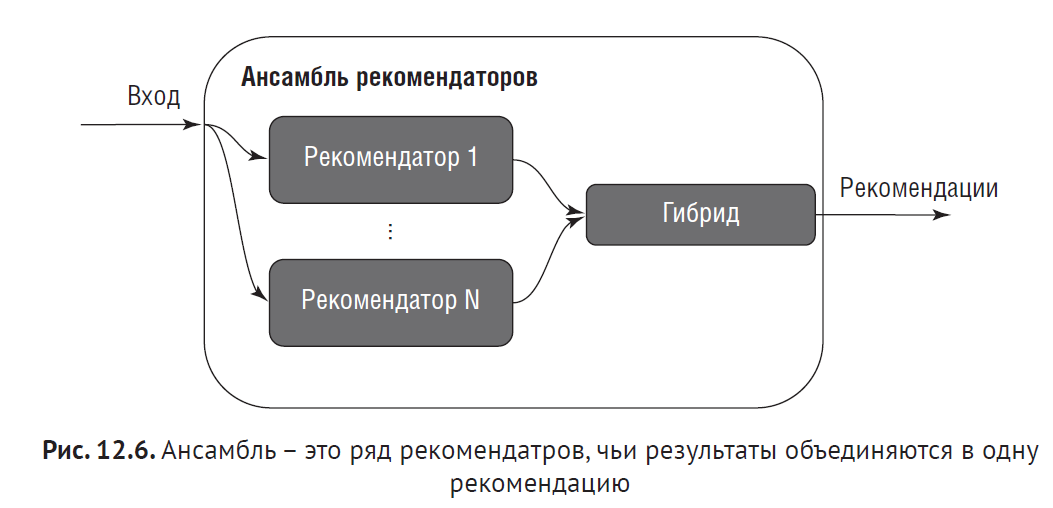
2.6

Изображение выглядит как текст

Автоматически созданное описание







2.7

Изображение выглядит как текст

Автоматически созданное описание

Как определить, в какой точке начать спуск? Вы можете начать в любом

месте, поскольку данная функция чашеобразная, как на рис. 11.14. Часто

в работе попадаются функции, у которых более одного локального миниму-

ма, как показано на рис. 11.15. Часто лучшее всего попробовать различные

стартовые точки и посмотреть, какой будет результат. Например, если вы

будете работать с описанными ранее функциями, как показано на рис. 11.14,

вы выбрали *x* = –5.

Чтобы найти направление вниз, нужно найти производную. Если вы не зна-

ете о дифференцировании, то просто примите на веру следующее:

Изображение выглядит как текст

Автоматически созданное описание

В стартовой точке *x* = –5 вы получите:

Изображение выглядит как текст

Автоматически созданное описание

что означает, что исходная функция в точке *x* = –5 наклонена вниз так, что

функция уменьшается на 125 при изменении аргумента на 1. Если произво-

дная функции положительная, вы должны двигаться вправо, чтобы найти *x*,

который даст минимальный результат.

Для проверки посмотрите на рис. 11.14, где видно, что при подстановке *x* = 2

в производную вы получите *dy / dx* = 43, т. е. двигаться нужно влево. Если име-

ется функция с несколькими переменными, как на рис. 11.13, это делается для

каждой из переменных.

Изображение выглядит как текст

Автоматически созданное описание

**Когда заканчивать?**

Сложно сказать, когда считать задачу выполненной (поэтому это *эвристи-*

*ческий* метод). Тут важно, чтобы ваша целевая функция, которую вы пытае-

тесь оптимизировать, становилась все меньше и меньше. Например, если вы

делаете шаг и функция уменьшается лишь на 0,0001, возможно, пора оста-

новиться. Или можно задать фиксированное количество итераций (шагов)

и закончить по их истечении.

FunkSVD выполняет градиентный последовательно добавля факторы и пересчитывая смещение пользователя

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Регуляризация – это штраф за ошибки

Параметры (скорость обучения и регуляризация) нужно подбирать чтобы RMSE была минимальной

2.8

Холодный старт для пользователей.

Мы получаем демографические данные о пользователе через мета данные запроса. С помощью API социальных сетей мы можем узнать уровень образования, социальный статус и прочие характеристики.

Существуют два основных подхода к применению в рекомендациях демографической информации о пользователе:  
— Экспертным образом составляются стереотипы для различных демографических категорий.

— Демографические категории определяются автоматически путем выявления кластеров пользователей со схожими интересами. Рекомендации строятся на основе того, какие рейтинги проставляли пользователи из той же категории, то есть того же возраста, пола, местоположения и т. д.

Для кластеризации по демографическим данным естественно использовать метод [*k*-средних](http://ru.wikipedia.org/wiki/K-means), так как в этом случае каждый кластер определяется точкой своего центра и, в следствии этого, хорошо интерпретируется.

После того, как кластеры пользователей получены и для каждого нового пользователя, указавшего свои демографические данные, мы знаем кластер, к которому он относится, мы можем улучшить рекомендации на холодном старте с помощью групповых рекомендаций или фильтр-ботов.

Наиболее естественным является метод **групповых рекомендаций (group recommendation to individual user)**, название которго говорит само за себя: мы подбираем новому пользователю такие рекомендации, которые нравятся большинству пользователей в его демографической категории.

Альтернативным подходом являются **фильтр-боты (filterbots)**, которые генерируют дефолтные рейтинги для нового пользователя. То есть при регистрации фильтр-боты автоматически сгенерируют несколько рейтингов для пользователя на основе его демографических данных, которые будут использованы алгоритмами коллаборативной фильтрации на холодном старте.

Холодный старт для элементов

Алгоритм **relevance feedback** составляет профиль тегов (то есть ключевых слов) для каждого пользователя по контент-профилям страниц, которые лайкал этот пользователь.  
Новые элементы рекомендуются тем пользователям, прифили тегов которых наиболее коррелируют с контент-профилем вновь добавленного элемента.

Альтернативой может быть алгоритм LDA. Это алгоритм кластеризации – который разбивает все элементы на кластеры, после этого выясняется предпочтения пользователей по этим кластерам. Новый элемент добавляется в существующий кластер и рекомендуется тем пользователям, которые предпочитают данный кластер.

2.9

Алгоритм кластеризации k-средних находит такие точки k, называемые цен-

троидами, которые удовлетворяют утверждению о том, что сумма расстояний

между всеми элементами и присвоенными ими центроидами будет настолько

мала, насколько это возможно. Пока пишу этот текст, смотрю как на участке

работает поливалка для цветов. Задача примерно похожая – надо крепко подумать о том, где разместить поливалку, потому что вода в Дании дорогая, и нужно найти такую точку, чтобы поливать как можно больше цветов, а воды тратить

поменьше. Кластеризация k-средних подойдет и для этой задачи, так как этот

алгоритм сгенерирует центроиды, в которые можно разместить поливалку.

Алгоритм выполняет следующие этапы:

1. Выбирается k центров кластеров.
2. Дальше в цикле делается следующее:
   1. для каждой точки данных в наборе находится центроид с кратчай-

шим расстоянием;

* 1. когда все точки будут назначены центроиду, вычисляет сумму всех

расстояний между элементом и его центроидом.

1. Если расстояние оказалось не меньше, чем в предыдущей итерации,

возвращаются кластеры.

1. Каждый центроид перемещается в центр назначенного кластера.

Существует много способов выбрать начальные положения центроидов,

и это очень важный шаг, от которого сильно зависят результаты всей кла-

стеризации.

2.10 Процедура совместной фильтрации

Простыми словами, совместная фильтрация выводит вам список рекомендуемых элементов. Список создается на основе данных о людях, чьи вкусы схожи с вашими и которые видели что-то, чего еще не видели вы.

Совместная фильтрация (CF, Collaborative filtering):

1. Item-Item
2. User-user

То есть мы находим пользователей, схожих с активным пользователем, а за-

тем рекомендуем ему фильмы, которые понравились схожим пользователям.

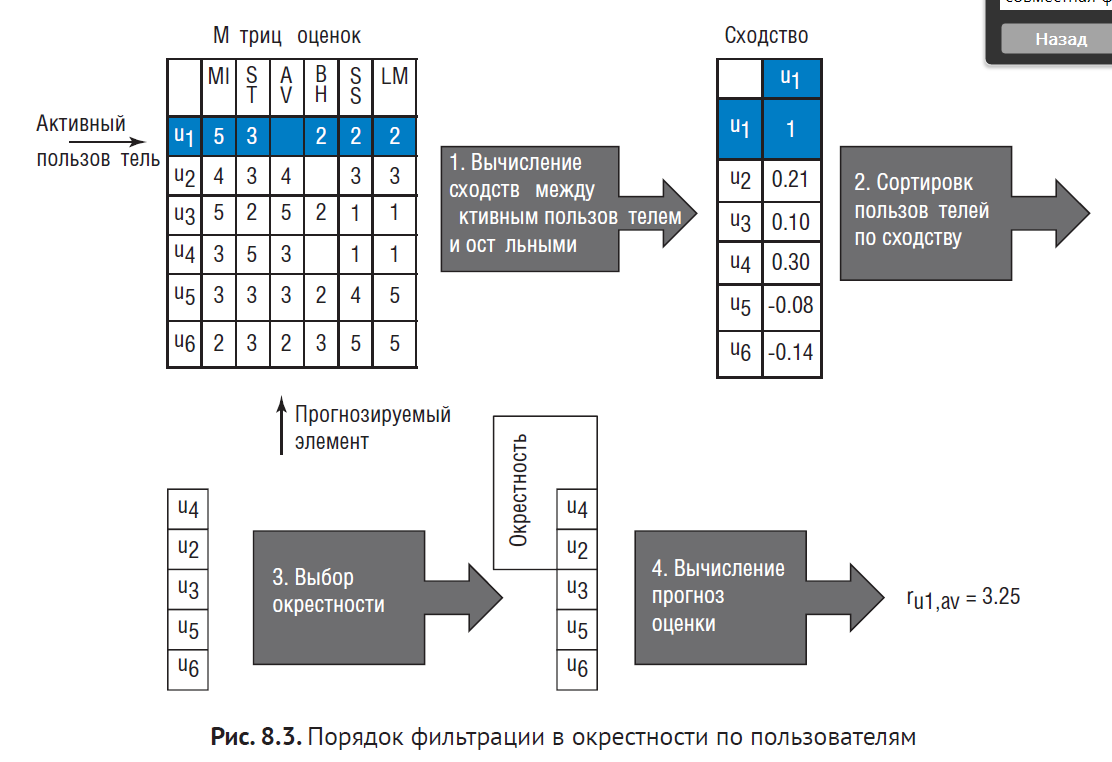
И второй способ – мы находим элементы, схожие с теми, которые пользователю понравились ранее. Чтобы сделать все это, нам снова понадобятся матрицы оценок, где описаны предпочтения пользователей.

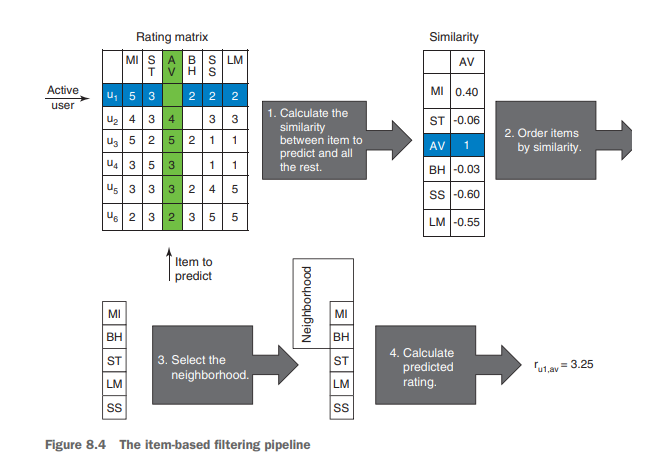
Процедура совместной фильтрации (пользователь-пользователь):

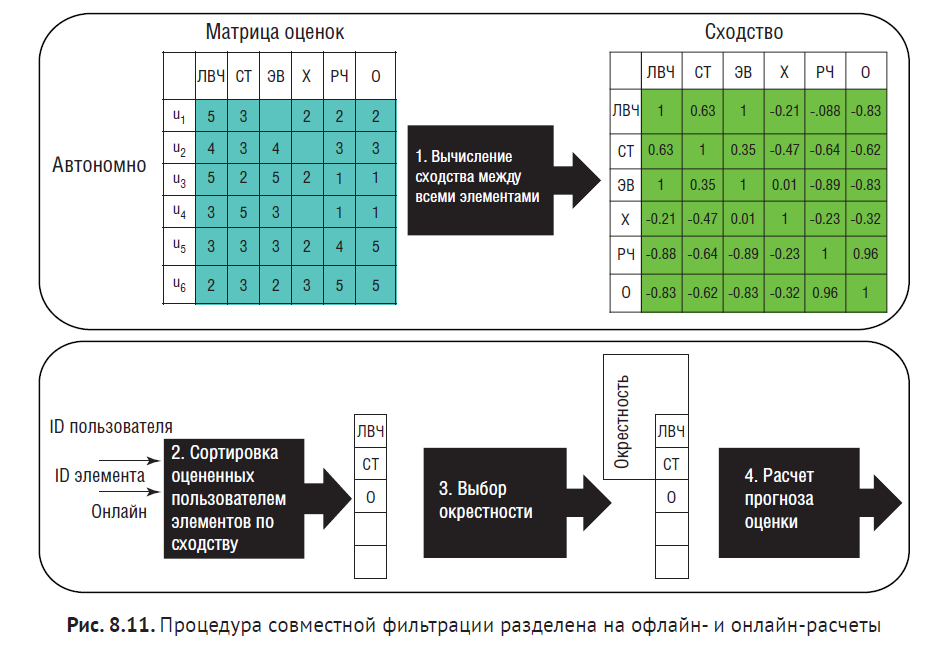
1. Вычисляем сходство между текущим пользователем и остальными (коэффициент Пирсона, Отиаи или косинусное расстояние)
2. Сортируем пользователей по сходству
3. Выбираем окрестность (k-mean,топ-N, пороговое значение)
4. Делаем прогноз

Процедура совместной фильтрации (элемент-элемент):

1. Вычисляем сходство между текущим элементом и остальными (коэффициент Пирсона, Отиаи или косинусное расстояние)
2. Сортируем элементы по сходству
3. Выбираем окрестность
4. Делаем прогноз

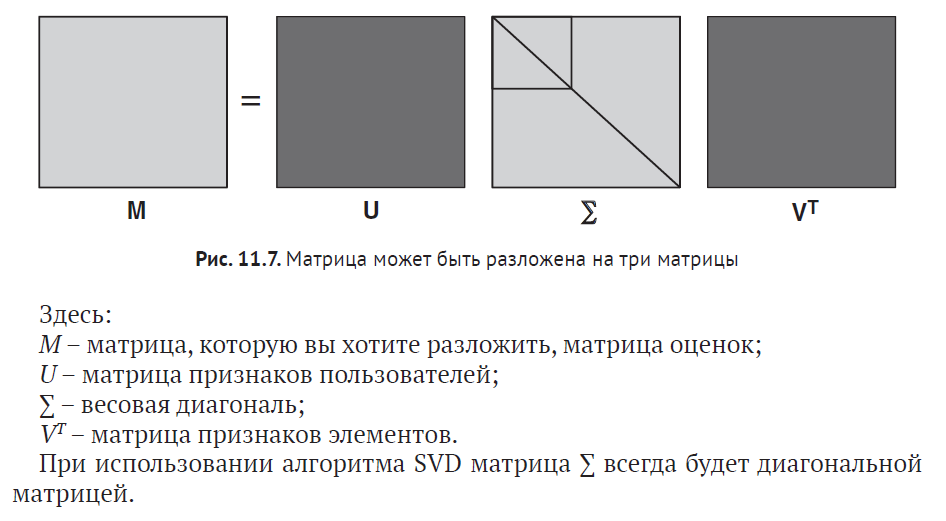






2.11 Факторизация – это разложение матрицы на множества. Методы разложения матрицы SVD, FuncSVD,SVD++.Факторизация нужна, чтобы уменьшить размер матрицы, потеряв небольшую часть данных.

Формула SVD



Но идея состоит в том, что центральная диагональная матри-

ца Σ содержит элементы, которые отсортированы от самых больших до самых

маленьких. Эти элементы – *особые значения*, показывающие, сколько инфор-

мации признак производит для набора данных.

В таком небольшом примере сокращение матрицы не имеет большого значе-

ния. Как правило, нам нужно сохранить 90 % информации. Если сложить все веса –

это будет 100 %, и теперь нужно продолжать подсчет весов, пока не получим 90 %

информации.

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описание

Сокращение матрицы для числа 4:

Изображение выглядит как стол

Автоматически созданное описание

После выполнения факторизации становится легко предсказать оценки поль-

зователей. Для этого достаточно посмотреть на новую матрицу M\_hat, которая

содержит все предсказанные оценки.

2.12

Есть два способа генерировать рекомендации: рассчитать все прогнозы оценок и взять самые большие из них либо перебрать все элементы и найти подобные в сокращенном пространстве. Третий способ – с помощью новых матриц вы можете выполнить совместную фильтрацию в окрестности, как вы это делали в главе 8. Но это не очень хорошая идея, так как у матрицы нет нулевых ячеек (после нормализациии). В таком плотном пространстве у вас гораздо больше шансов найти подобные элементы или пользователей.

2.13

Тоже что и 2.3

2.14

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

2.17 Алгоритм «Item-Item»: рекомендации схожих товаров на основе связей между их рейтингами.

Совместная фильтрация по элементам , или item-item , или элемент-элемент — это форма совместной фильтрации для рекомендательных систем, основанная на сходстве между элементами, рассчитанном с использованием оценок этих элементов людьми. Совместная фильтрация элементов была изобретена и использована Amazon.com в 1998 году.

Более ранние системы совместной фильтрации, основанные на схожести оценок между пользователями (известная как совместная фильтрация пользователей ), имели несколько проблем:

1. системы работали плохо, когда у них было много элементов, но сравнительно мало оценок
2. вычисление сходства между всеми парами пользователей было дорогим
3. профили пользователей быстро менялись, и всю модель системы приходилось пересчитывать

Модели «элемент-элемент» решают эти проблемы в системах, в которых пользователей больше, чем элементов. В моделях «элемент-элемент» используется распределение рейтинга по элементу , а не по пользователю . При большем количестве пользователей, чем элементов, каждый элемент, как правило, имеет больше оценок, чем каждый пользователь, поэтому средний рейтинг элемента обычно не меняется быстро. Это приводит к более стабильным распределениям оценок в модели, поэтому модель не нужно перестраивать так часто. Когда пользователи потребляют, а затем оценивают элемент, аналогичные элементы этого элемента выбираются из существующей модели системы и добавляются в рекомендации пользователя.