



Занятие 13. Ранжирование. Рекомендательные системы

Колмагоров Евгений
ml.hse.dpo@yandex.ru



20 января 2025

План лекции

1. Задача ранжирования
2. Существующие подходы
3. Рекомендации как частный случай ранжирования
4. Подходы к решению задачи
5. Метрики качества



Ранжирование. Пример

Яндекс машинное обучение   **Найти**

ПОИСК КАРТИНКИ ВИДЕО КАРТЫ МАРКЕТ НОВОСТИ ПЕРЕВОДЧИК ЕЩЁ

W **Машинное обучение** — Википедия
ru.wikipedia.org > Машинное обучение ▾
Машинное обучение (англ. Machine Learning) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи...

life **Что такое машинное обучение и почему оно может...**
lifehacker.ru > Лайфхакер > ...-mashinnoe-obuchenie ▾
Машинное обучение избавляет программиста от необходимости подробно объяснять компьютеру, как именно решать проблему.

Y **Курс «Машинное обучение» 2014 - YouTube**
youtube.com > playlist?list=..._b9zqEQiiBtC ▾
Курс "Машинное обучение" является одним из основных курсов Школы, поэтому он является обязательным для всех студентов ШАД.

P **Машинист электропоезда - обучение | Про профессии.ру**
proprof.ru > Машинист электропоезда ▾
Машинист электропоезда - **обучение**. И метрополитен, и РЖД приглашают на **обучение** в собственные учебно-производственные центры.

ng **Обучение - машина - Большая Энциклопедия Нефти...**
ngpedia.ru > id201843p1.html ▾
После **обучения машины** или в ходе его, смотря по алгоритму, проводится прогнозирование новых катализаторов...

Задача сортировки объектов
по их полезности для
пользователя называется
задачей ранжирования

Ранжирование. Пример

Яндекс

машинное обучение



Найти

ПОИСК

КАРТИНКИ

ВИДЕО

КАРТЫ

МАРКЕТ

НОВОСТИ

ПЕРЕВОДЧИК

ЕЩЁ

W **Машинное обучение** — Википедия

ru.wikipedia.org > Машинное обучение ▾

Машинное обучение (англ. Machine Learning) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи...

L Что такое **машинное обучение** и почему оно может...

lifehacker.ru > Лайфхакер > ...-mashinnnoe-obuchenie ▾

Машинное обучение избавляет программиста от необходимости подробно объяснять компьютеру, как именно решать проблему.

Y Курс «**Машинное обучение**» 2014 - YouTube

youtube.com > playlist?list=..._b9zqEQiiBtC ▾

Курс "**Машинное обучение**" является одним из основных курсов Школы, поэтому он является обязательным для всех студентов США.

P **Машинист электропоезда - обучение** | Про профессии.ру

proprof.ru > Машинист электропоезда ▾

Машинист электропоезда - обучение. И метрополитен, и РЖД приглашают на **обучение** в собственные учебно-производственные центры.

ng **Обучение - машина** - Большая Энциклопедия Нефти...

ngpedia.ru > id201843p1.html ▾

После **обучения машины** или в ходе его, смотря по алгоритму, проводится прогнозирование новых катализаторов...



Полезность объекта
определяется
относительно, того какой
был запрос

Формальное определение

- Дан набор запросов: $Q = \{q_1, q_2, \dots, q_m\}$
- Дан набор документов: $D = \{d_1, d_2, \dots, d_n\}$
- Необходимо для каждого запроса q_i получить упорядоченный набор документов $D = \{d_{i1}, \dots, d_{in}\}$ по убыванию полезности
- Как считать “полезность”?

Полезность документа

Так как в задаче ранжирования в первую очередь важен порядок объектов, а не точное значение их скоров, то будем использовать относительную полезность

- Рассматриваем пары “запрос-документ” (q, d)
- Для некоторых троек (q, d_1, d_2) известно, что для запроса q документ d_1 должен стоять раньше, чем d_2
- Пусть R – множество таких троек (q, d_1, d_2) для которых известен такой порядок

Построение модели

- Раньше: строим модель $a_w(x)$, которая приближает ответы y
- Сейчас: строим модель $a_w(q, d)$, которая правильно упорядочивает документы для запросов

$$(q, d_1, d_2) \in R \Rightarrow a_w(q, d_1) > a_w(q, d_2)$$

Пример

- Для запроса q и набора документов $\{d_1, d_2, d_3, d_4\}$ известно, что

$$(q, d_1) > (q, d_2) > (q, d_3) > (q, d_4)$$

- Какие наборы скоров для каждого из документов лучше?
- (3, 2, 4, 1)
- (2, 3, 4, 1)
- (3, 4, 2, 1)
- (13, 10, 20, 7)

Пример

- Для запроса q и набора документов $\{d_1, d_2, d_3, d_4\}$ известно, что

$$(q, d_1) > (q, d_2) > (q, d_3) > (q, d_4)$$

- Какие наборы прогнозов лучше?
- (3, 2, 4, 1)
- (2, 3, 4, 1)
- **(3, 4, 2, 1)**
- (13, 10, 20, 7)
- В задаче ранжирования важен порядок, а не точное значение!

Целевая переменная в ранжировании

- Объекты – пары “запрос-документ” $x_i = (q, d)$
- Ответы – числа y_i
- Требование – если есть объекты (q, d_1) и (q, d_2) , такие что $y_1 > y_2$, то должно быть $a(q, d_1) > a(q, d_2)$

Целевая переменная. Пример

- $(q_1, d_1) - 1$
- $(q_1, d_2) - 0.7$
- $(q_1, d_3) - 0$
- $(q_2, d_1) - 0$
- $(q_2, d_2) - 1$
- Для q_1 должны получить ранжирование (d_1, d_2, d_3)
- Для q_2 должны получить ранжирование (d_2, d_1)

Поточечный (pointwise) подход

- Обучим модель $a(q, d)$, чтобы она как можно точнее приближала ответы y_i
- Например, модель линейной регрессии:

$$\sum_{(q,d,y) \in R} (w^T \cdot x(q, d) - y)^2 \rightarrow \min_w$$

- $x(q, d)$ – признаки для пары “запрос-документ”

Поточечный (pointwise) подход

- Простой в реализации
- В качестве модели можно использовать любую модель из задачи регрессии: линейные, деревья, ансамбли, нейронные сети и тд.
- Восстанавливает точные значения квазиоценки y_i , когда в задаче важен лишь порядок объектов

Попарный (pairwise) подход

- В ранжировании нужно правильно располагать пары документов друг относительно друга с учётом запроса q , поэтому формируем лосс:

$$\sum_{(q, d_i, d_j)} I[a(q, d_i) - a(q, d_j) < 0]$$

- Штрафуем, если второй документ из пары оказался раньше

Попарный (pairwise) подход

- Получили разрывной функционал – сложно оптимизировать
- Перейдем к гладкой верхней оценке (как в линейных классификаторах):

$$\sum_{(q, d_i, d_j)} I[a(q, d_i) - a(q, d_j) < 0] \leq \sum_{(q, d_i, d_j)} L(a(q, x_i) - a(q, x_j))$$

- Например, PairLogit loss: $L(z) = \log(1 + \exp(-z))$

Попарный (pairwise) подход

- Вычислительно сложнее поточечного, так как в сумме больше слагаемых
- Обычно даёт качество выше, чем поточечный
- Есть эффективные реализации: SVMLight, xgboost, catboost

Где можно встретить ранжирование

- Поиск
- Рекомендации
- Автодополнение
- Поиск синонимов

Задача рекомендации

Важным частным случаем задачи ранжирования является задача рекомендации.

В задаче рекомендации в качестве запроса q , выступает характеристика пользователя вместе с его историей взаимодействия с некоторой информационной системой.

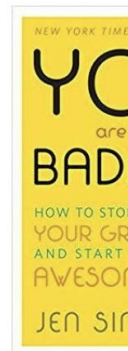
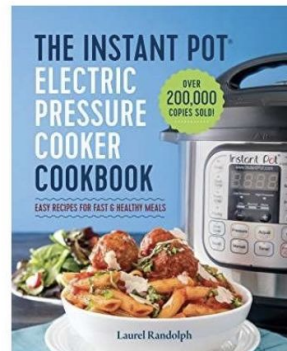
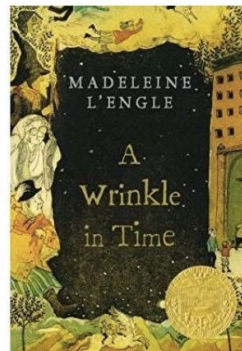
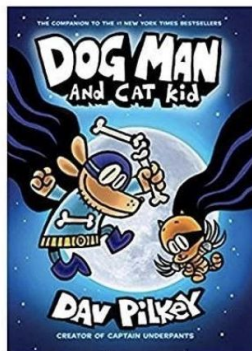
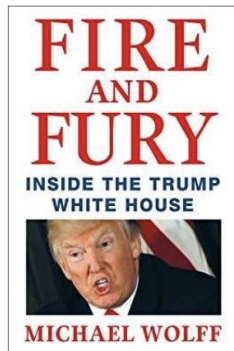
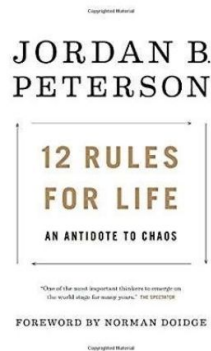
Где можно встретить рекомендательные системы

- Фильмы, видео
- Музыка
- Книги
- Товары
- Приложения
- Лента в социальных сетях
- Услуги (рестораны, отели, авиабилеты)
- Научные публикации

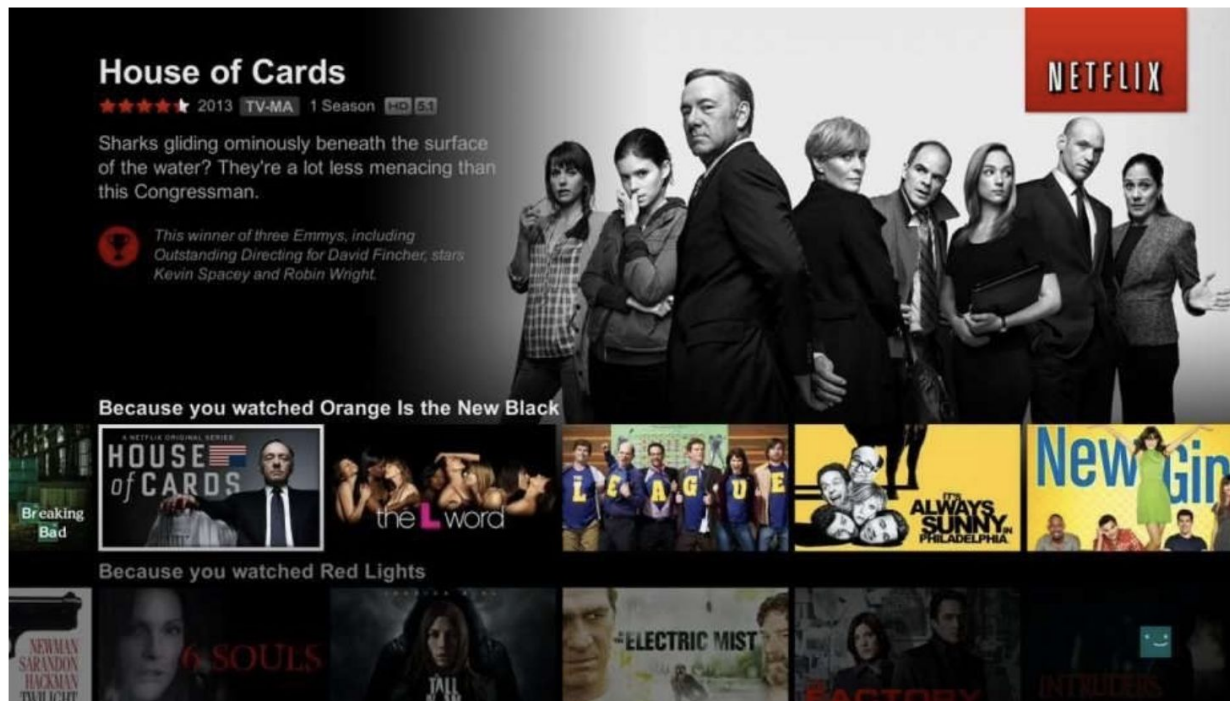


Примеры рекомендательных систем. Amazon

Books best sellers [See more](#)



Примеры рекомендательных систем. Netflix



Рекомендательные системы

- Рекомендательные системы сокращают объём информации, необходимый для принятия решений
- Не нужно читать отзывы на 1000 фильмов – модель сама из предпочтений выберет подходящий
- Netflix: 80% просмотренных фильмов найдены через рекомендательную систему
- Amazon: 35% продаж через полки рекомендаций
- Youtube: 60% просмотров благодаря рекомендациям

Цели с точки зрения покупателя и продавца

Цели с точки зрения продавца:

- Продавать больше товаров
- Продавать больше редких товаров
- Повысить лояльность пользователя

Цели с точки зрения покупателя:

- Расширить представление об ассортименте
- Купить то, что потенциально представляет собой интерес
- Понять, что покупать вместе с данным товаром

Виды рекомендательных систем

Какие объекты хотим рекомендовать? Наиболее подходящие

Но что такое “наиболее подходящий”?

Можно придумать несколько вариантов:

- Предлагать популярное
 - Глобально популярное
 - по каким-то определённым категориям
 - у отдельных групп
- Похожий объект на то, с чем взаимодействовал пользователь
- Или то, с чем взаимодействуют похожие люди

Виды рекомендательных систем

Существует огромное множество различных алгоритмов рекомендательных систем, но все они делятся на три основных вида:

1. Рекомендации на основе контента (Content-based)

- Рекомендации строятся на основе объектов, которые похожи на те объекты, с которыми пользователь взаимодействовал
- Уровень подобия оценивается только по признакам самого объекта
- Можем рекомендовать объекты, которые находятся в одной предметной области

2. Коллаборативная фильтрация (Collaborative Filtering):

- Следующие рекомендации строятся на основе пользовательской истории взаимодействия и истории взаимодействия других пользователей
- Нет привязки к предметной области
- Не умеет работать с новыми пользователями

3. Гибридные подходы - учитывают при рекомендациях внутренние особенности объекта, так и историю взаимодействия с другими объектами

Content-based filtering

Основная идея — использовать характеристики объекта для поиска похожих объектов.

Функция подобия — как правило либо скалярное произведение или косинусное расстояние


























Процесс построения рекомендаций — ищем наиболее похожие на те объекты, с которыми пользователь уже взаимодействовал



Collaborative filtering

Основная идея – использовать историю взаимодействий пользователей с объектами для получения векторных представлений

Функция подобия – как правило либо скалярное произведение или косинусное расстояние

Neighbour-based collaborative filtering

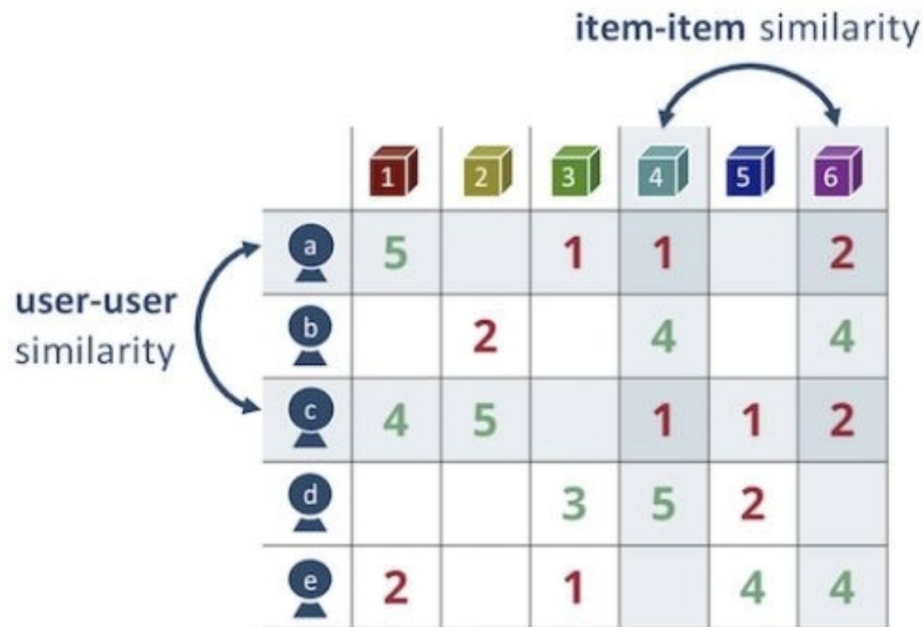
Основная идея – использовать строки или столбцы из матрицы оценок как векторное представление пользователя или объекта

Два подхода:

- Item-Item – матрица схожести объектов
- User-User – матрица схожести пользователей

Как использовать:

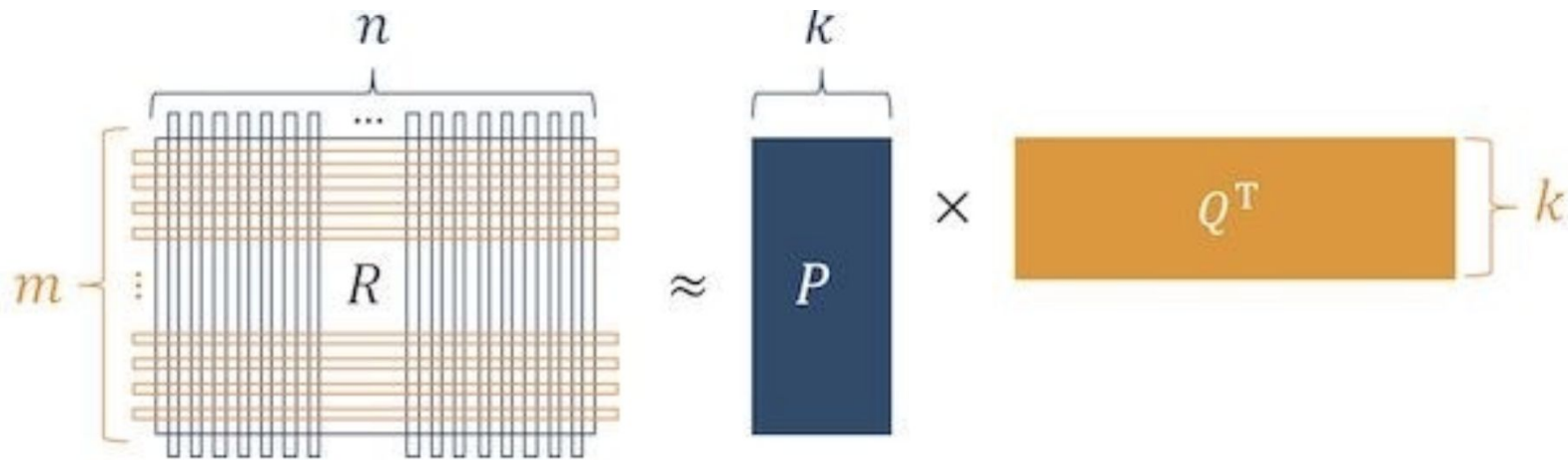
- Найти похожие объекты на то, с чем пользователь взаимодействовал
- Рекомендовать объекты из тех, с которыми взаимодействовали похожие пользователи



Model-based collaborative filtering

Основная идея – построить внутренние векторные представления для пользователей и объектов на основе матрицы оценок

Основной подход – матричные разложения



Матричное разложение

Предположим, что существует некоторое признаковое пространство U и I размерности d , как для пользователей, так и для фильмов.

Теперь каждый пользователь описывается некоторым вектором признаков: $u = (u_1, \dots, u_d)$ и каждый предмет описывается своим вектором $i = (i_1, \dots, i_d)$

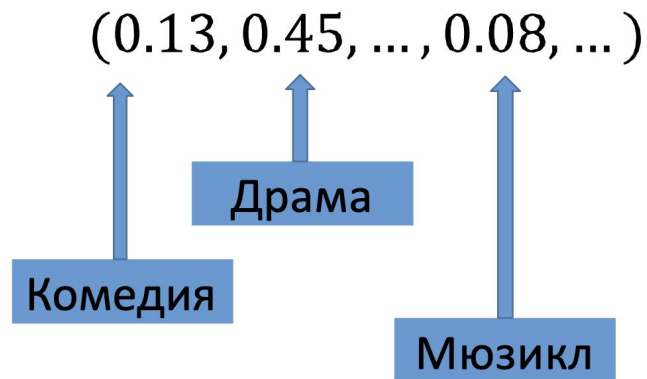
Тогда будем предсказывать оценку $r_{u,i}$ некоторой предмета i пользователем u по следующей формуле $r_{u,i} = u * i^T$

Векторы пользователя и объекта

Каждый стоящий на j -ой позиции признак как в пользовательском вектре u так и векторе объекта i несёт одинаковую семантику.

Например, для случая рекомендации фильмов j -й признак:

- Для пользователя – насколько он интересуется каждым жанром
- Для фильма – насколько он относится к тому или иному жанру



Рейтинг

Предположение: заинтересованность определяется как скалярное произведение векторов пользователя и фильма



$$(0.1, 0.5, 0.01, 0.92) \times (0, 0, 0.1, 0.95) = 0.875$$















$$(0.1, 0.5, 0.01, 0.92) \times (0.9, 0, 0, 0.1) = 0.182$$

Пользователь

Фильм

Получение рейтинга

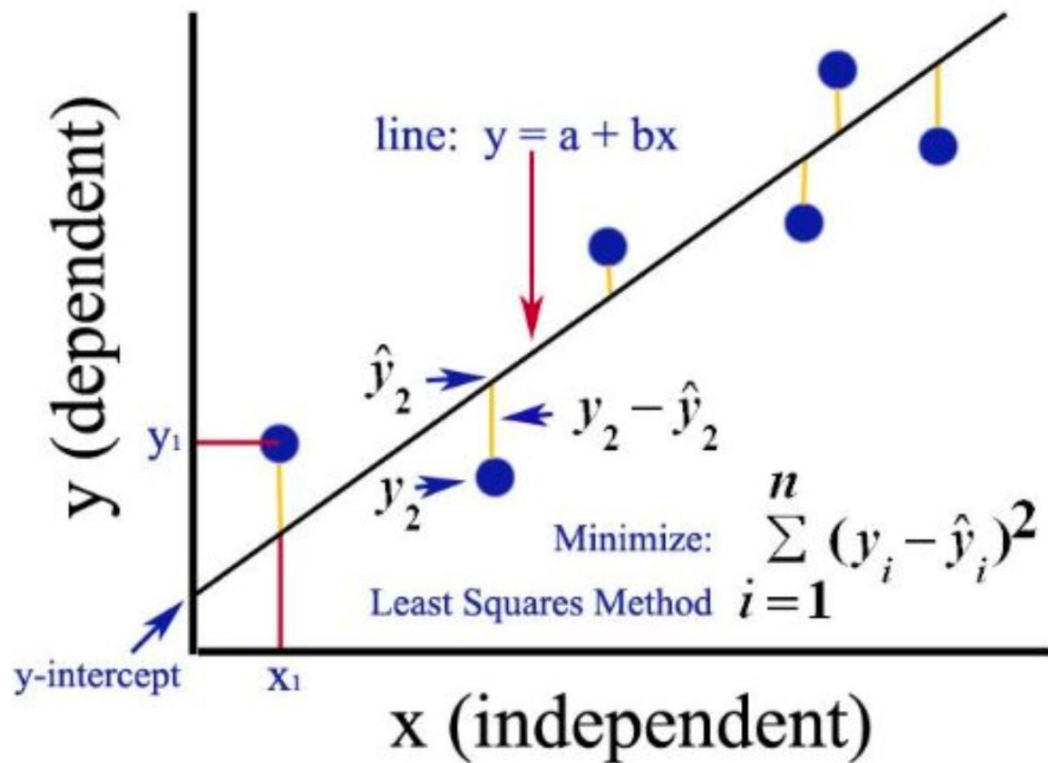
	 Comedy	 Action
M1	3	1
M2	1	2
M3	1	4
M4	3	1
M5	1	3

	 Comedy	 Action
		
		
		
		

=

	M1	M2	M3	M4	M5
	3	1	1	3	1
	1	2	4	1	3
	3	1	1	3	1
	4	3	5	4	4

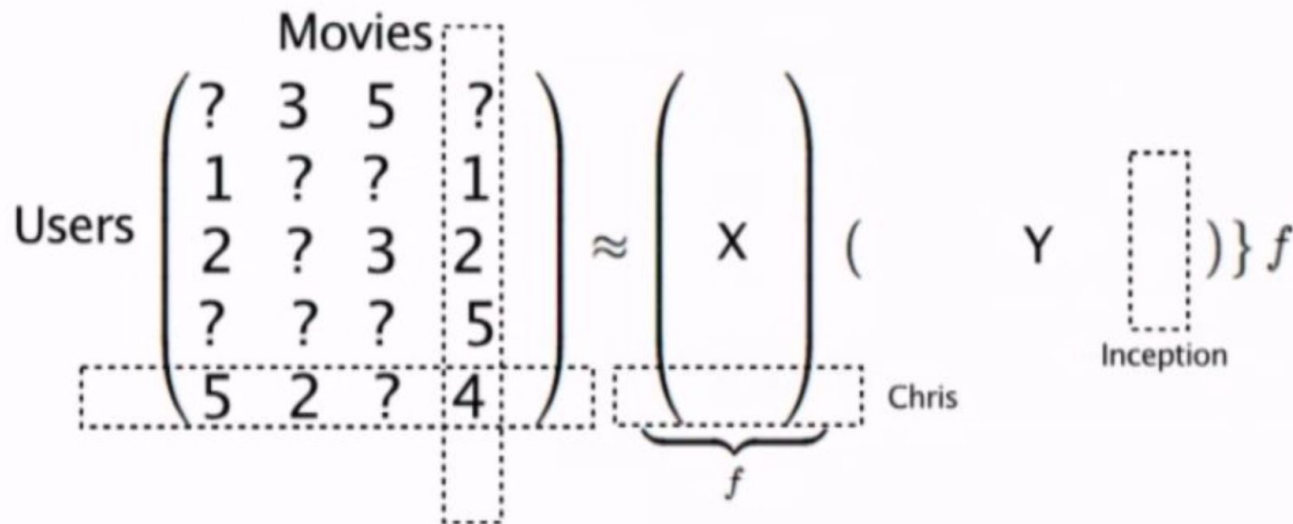
Напоминание линейная регрессия



Метрика, которую оптимизируем

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (p_i - y_i)^2}{n}}$$

ALS – Alternating Least Squares



$$\min_{x,y} \sum_{u,i} (r_{ui} - x_u^T y_i - \beta_u - \beta_i)^2 + \lambda (\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

- r_{ui} = user u 's rating for movie i
- x_u = user u 's latent factor vector
- x_i = item i 's latent factor vector
- β_u = bias for user u
- β_i = bias for item i
- λ = regularization parameter

Ищем матрицы U и I методом ALS

Будем искать матрицы U и I итеративным алгоритмом - поочерёдно фиксируя матрицу пользователей и матрицу предметов

Таким образом задача превращается в последовательность задач наименьших квадратов

$$\min_{x,y} \sum_{u,i} (r_{ui} - x_u^T y_i - \beta_u - \beta_i)^2 + \lambda (\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$$

- r_{ui} = user u 's rating for movie i
- x_u = user u 's latent factor vector
- y_i = item i 's latent factor vector

- β_u = bias for user u
- β_i = bias for item i
- λ = regularization parameter

Шаг градиентного спуска

Оптимизируемый функционал:

$$Q = \sum_{u,i} (r_{u,i} - \langle x_u, y_i \rangle)^2 \rightarrow \min_{x_u, y_i}$$

Посчитаем для данного функционала градиент:

$$\begin{aligned} \frac{\partial Q}{\partial x_u} &= \sum_{u,i} \frac{\partial}{\partial x_u} (r_{u,i} - \langle x_u, y_i \rangle)^2 = \sum_i 2(r_{u,i} - \langle x_u, y_i \rangle) \frac{\partial -\langle x_u, y_i \rangle}{\partial x_u} = \\ &= \sum_i -2(r_{u,i} - \langle x_u, y_i \rangle) y_i = \sum_i 2(\langle x_u, y_i \rangle - r_{u,i}) y_i \end{aligned}$$

Алгоритм ALS

SGD Algorithm for MF

Input: training matrix V , the number of features K , regularization parameter λ , learning rate ϵ

Output: row related model matrix W and column related model matrix H

- 1: Initialize W, H to $UniformReal(0, \frac{1}{\sqrt{K}})$
 - 2: **repeat**
 - 3: **for random** $V_{ij} \in V$ **do**
 - 4: $error = W_{i*}H_{*j} - V_{ij}$
 - 5: $W_{i*} = W_{i*} - \epsilon(error \cdot H_{*j}^\top + \lambda W_{i*})$
 - 6: $H_{*j} = H_{*j} - \epsilon(error \cdot W_{i*}^\top + \lambda H_{*j})$
 - 7: **end for**
 - 8: **until** convergence
-

Виды пользовательских фидбеков

Чтобы построить рекомендательную систему, необходимо учитывать пользовательские отклики на тот или иной объект рекомендации.

- **Явный отклик (Explicit feedback)** - когда можем собрать прямую оценку/мнение пользователя об объекте, например, в случае с рекомендацией фильмов - это может быть поставленный пользователем рейтинг данному фильму
- **Неявный отклик (Implicit feedback)** - более часто встречаемый вид откликов, когда нет прямой оценки от пользователя об объекте, поэтому оценка аппроксимируется исходя совершенных действий: клики, просмотры, добавление в корзину и тд.

Implicit matrix factorization

Добавим различный вес на те или иные действия пользователя:

$$\sum_{u,i} w_{u,i} (r_{u,i} - \langle x_u, y_i \rangle) \rightarrow \min_{x_u, y_i}$$

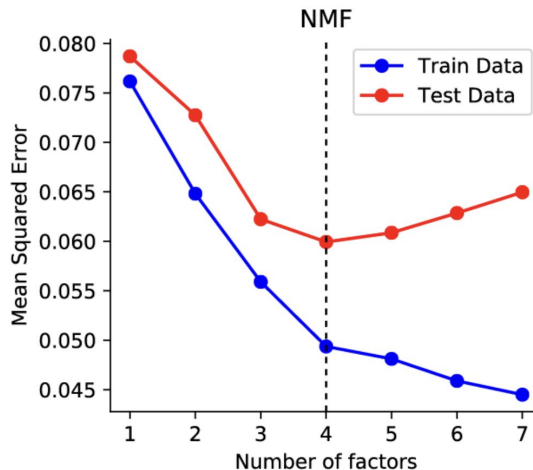
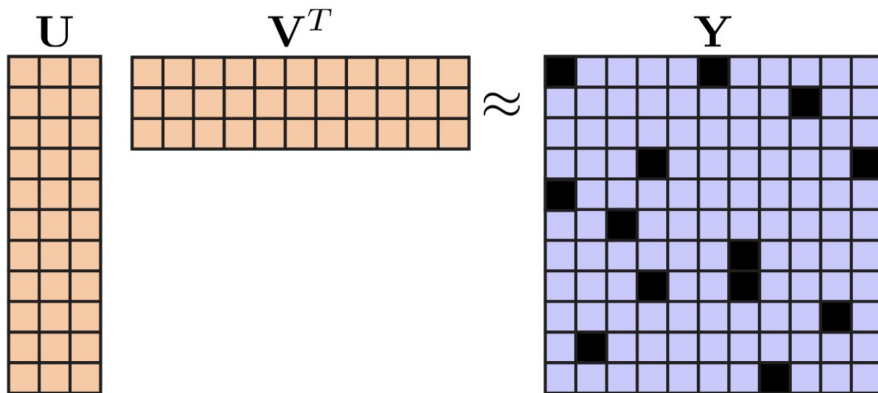
- $w_{u,i}$ принимает большие значения для $r_{u,i} \neq 0$ и значительно меньшие для $r_{u,i} = 0$
- Применяют следующую формулу для $w_{u,i} = 1 + \alpha |r_{u,i}|$, где $\alpha = 1, 10, 1000$

Как подобрать число d - размерность признаков?

Число d играет важное значение в качестве работы алгоритма, малые значения d будут приводить к тому, что можем пропустить в модели учёт важных параметров, большие же значения будут приводить к переобучению. Можно доказать, что при $d = \min(m, n)$, произведение $U \cdot V$ будет в точности равно целевой матрице R .

Кросс-валидация для поиска оптимального значения

Используем отложенную выборку, на которой будем запускать алгоритм с различными значениями d , и выберем то, где RMSE ошибка будет минимальна.



Вопрос: почему не стоит использовать для кросс валидации сразу всю строку/столбец?

Способы оценки качества

При построение любой системы важно уметь правильно оценивать качество предлагаемого решения. В случае с задачей рекомендации существует два подхода в оценке алгоритмов.

- **Online-evaluation** - самый точный способ оценки качества системы — прямая проверка на пользователях в контексте бизнес-метрик. Это может быть CTR, время, проведенное в системе, или количество покупок. Но эксперименты на пользователях дороги, а выкатывать плохой алгоритм даже на малую группу пользователей не хочется, поэтому до онлайн-проверки пользуются оффлайн метриками качества.
- **Offline-evaluation** - оценка качества производится на отложенной выборке, которая как правило собирается из истории взаимодействия пользователей с системой. Как правило, для оценки применяют те же метрики, что и для задач поиска и ранжирования: $MAP@K$, $nDCG@K$ и тд.

Метрики в задаче рекомендации

Метрики в задаче рекомендаций можно поделить на следующие группы:

- Регрессионные
- Классификационные
- Ранжирующие

Регрессионные метрики

Регрессионные метрики применяются для оценки качества предсказанных моделью значений:

- Mean Absolute Error:

$$MAE = \frac{1}{N} \sum_{i=1}^N |y - \hat{y}|$$

- Mean Squared Error:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y - \hat{y})^2$$

- Rooted Mean Squared Error:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (y - \hat{y})^2}$$

Метрики классификации

Классификационные метрики оценивают качество топ-N рекомендаций с точки зрения бинарной классификации:

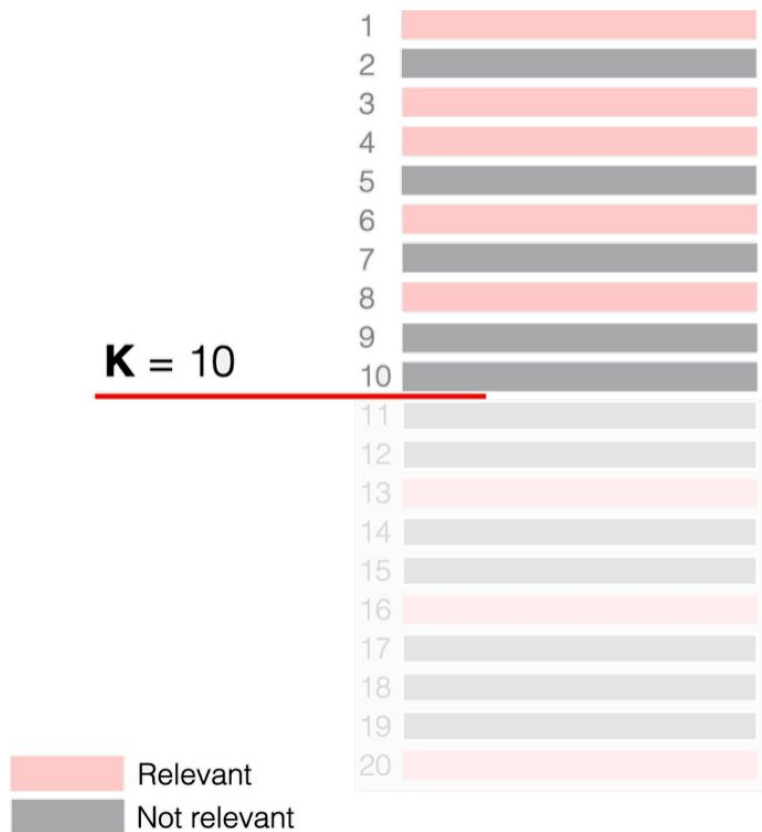
- **True Positive (TP)** – модель рекомендовала объект, с которым пользователь взаимодействовал
- **False Positive (FP)** – модель рекомендовала объект, с которым пользователь не взаимодействовал
- **True Negative (TN)** – модель не рекомендовала объект, с которым пользователь не взаимодействовал
- **False Negative (FN)** – модель не рекомендовала объект, с которым пользователь взаимодействовал

Метрики классификации

Наиболее популярными метриками являются:

- Precision@K
 - Формула: $TP / (TP + FP)$
 - Можно заметить, что под positives понимается рекомендованные объекты, то есть топ-K, значит $TP + FP = K$
 - Итоговая формула: TP / K
 - Интерпретируется как доля релевантных рекомендаций
- Recall@K
 - Формула: $TP / (TP + FN)$
 - $TP + FN$ это количество известных релевантных объектов для пользователя
 - Интерпретируется как доля релевантных объектов, попавших в рекомендации

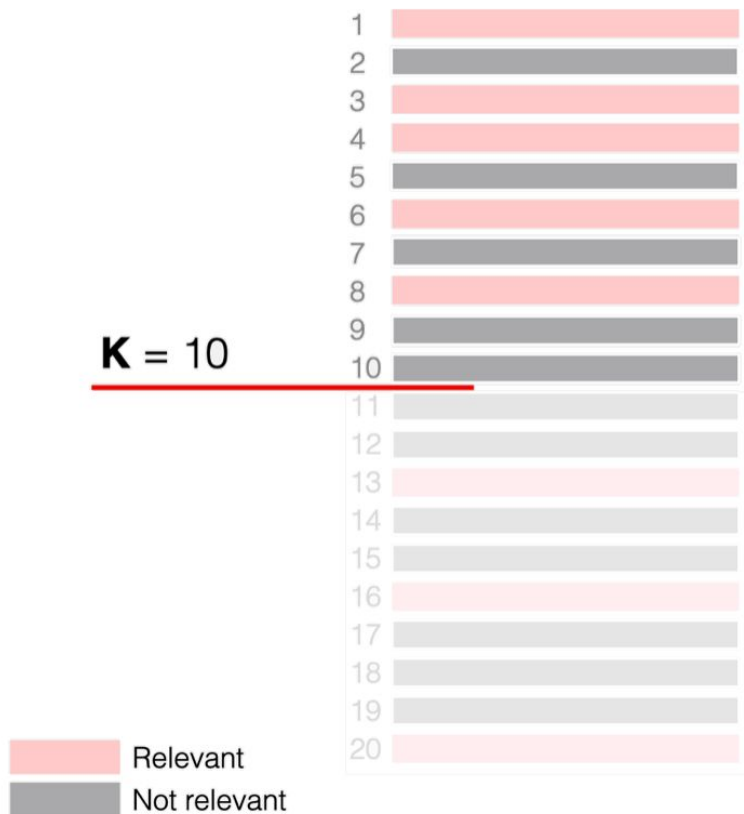
Пример расчёта Precision@K



$$\text{Precision@K} = \frac{\text{\# of relevant items at K}}{K}$$

$$\text{Precision@10} = \frac{5}{10} = 0.5$$

Пример расчёта Recall@K



$$\text{Recall@K} = \frac{\text{\# of relevant items at K}}{\text{\# of relevant items}}$$

$$\text{Recall@10} = \frac{5}{8} = 0.625$$







Average Precision

Чтобы иметь представление в целом о точности выдачи всей выдачи до К-ой позиции применяют усреднение $Precision@K$ по всем релевантным объектам

$$AP@K = \frac{1}{N} \sum_{k=1}^K Precision@k \times rel(k)$$

- N – количество релевантных объектов
- $rel(k)$ – равен 1 если k -й объект релевантный, и 0 если нет

Пример расчёта

Rank	Item	Precision@k
1		$1/1 = 1$
2		$1/2 = 0.5$
3		$1/3 = 0.33$
4		$2/4 = 0.5$
5		$3/5 = 0.6$
6		$3/6 = 0.5$

Average precision

$$AP@6 = \frac{1 + 0.5 + 0.6}{3} = 0.7$$

 Not relevant  Relevant

Метрики ранжирования

Mean Reciprocal Rank – средний обратный ранг

$$MRR = \frac{1}{N} \sum_{i=1}^N \frac{1}{rank_i}$$

- N – количество пользователей
- $rank_i$ – позиция первой релевантной рекомендации для пользователя i
- Если для некоторого пользователя не нашлось ничего релевантного, то дробь $1/rank_i$ зануляется

Метрики ранжирования

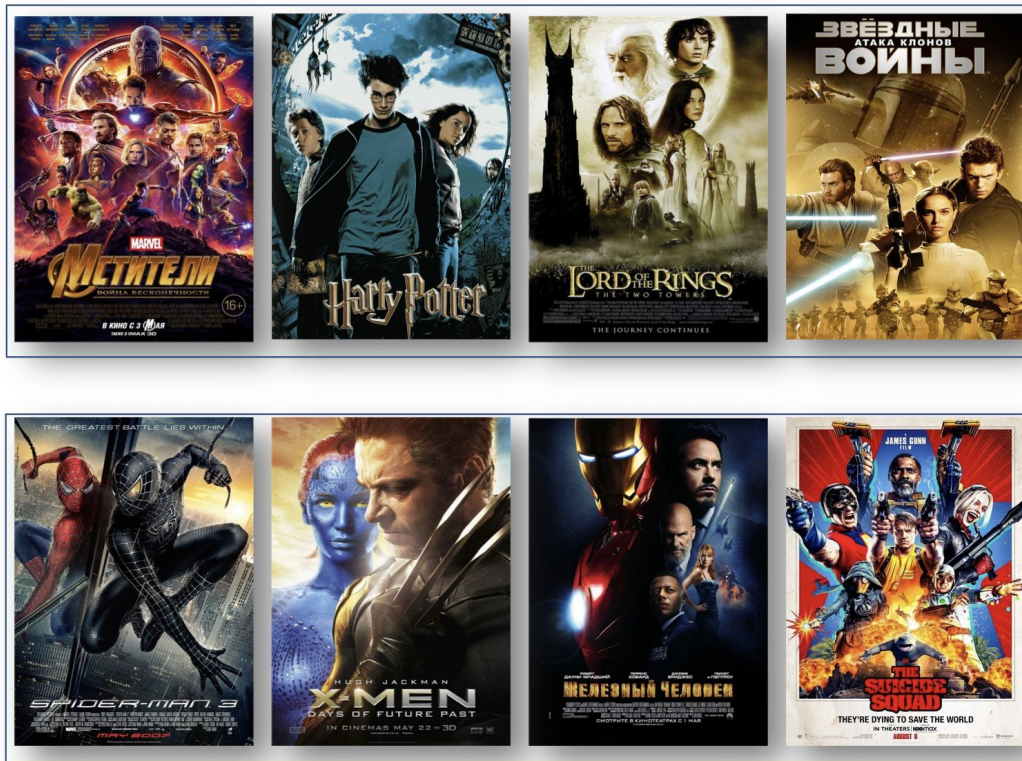
Mean Average Precision – средняя точность по пользователям

$$MAP@K = \frac{1}{U} \sum_{i=1}^U AP@K(user_i)$$

- U – количество пользователей
- $AP@K(user_i)$ – средняя точность на отдельном пользователе

Ещё немного про метрики

Чего не хватает этим рекомендациям?

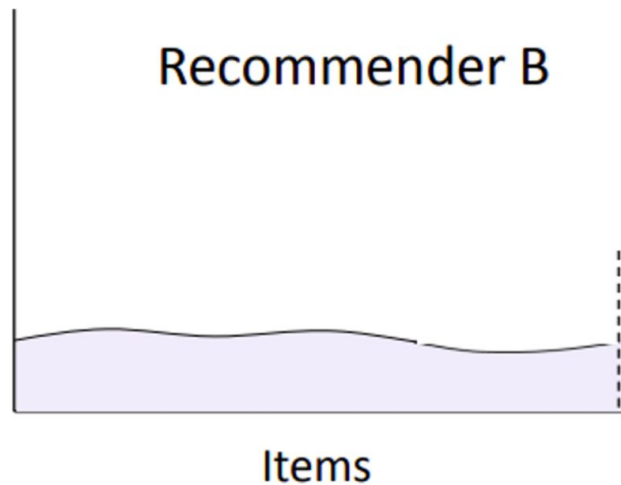
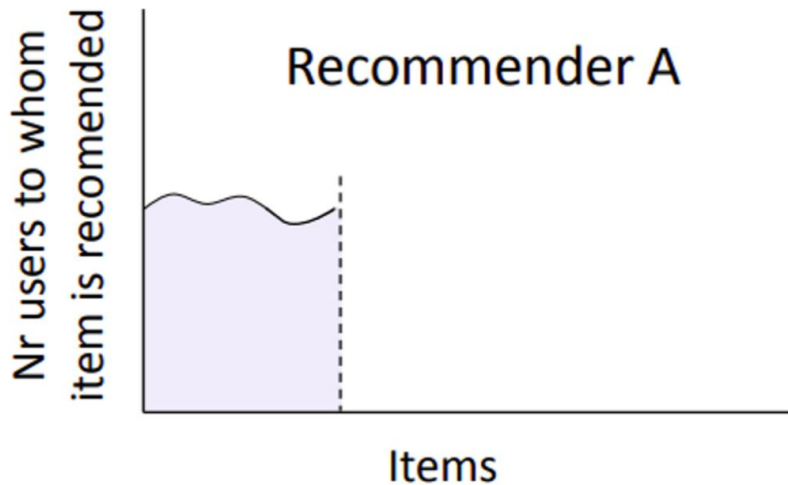


Общее разнообразие

Aggregate diversity – количество товаров, в совокупности рекомендуемых алгоритмом всем пользователям

$$AggDiv = \left| \bigcup_{u \in Users} R_u \right|$$

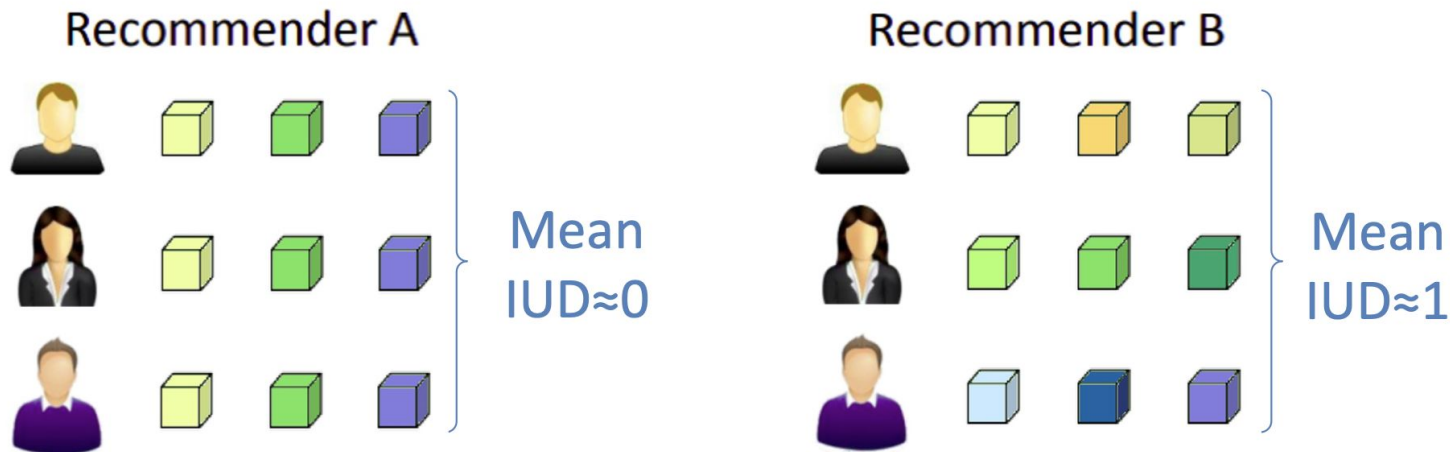
- Где R_u – множество товаров, порекомендованных пользователю u



“Персональность” рекомендаций

Inter-User Diversity – средняя доля пересечений рекомендаций для пользователя с рекомендациями для остальных пользователей

$$IUD = \frac{1}{|U|-1} \sum_{u \in U} \frac{|R - R_u|}{|R|}$$



Резюме

- Рекомендации – широкая задача с большим количеством коммерческих применений
- Модели: коллаборативная фильтрация, контентный подход
- Наиболее популярный подход: модели со скрытыми переменными
- Обилие метрик качества