

РЕКОМЕНДАТЕЛЬНЫЕ СИСТЕМЫ

Сергей Николенко

СПбГУ — Санкт-Петербург

18 ноября 2020 г.

Random facts:

- 18 ноября в Латвии — День независимости, а в России — День рождения Деда Мороза
- 18 ноября 1626 г. папа Урбан VIII освятил собор Святого Петра в Риме
- 18 ноября 1723 г. был пущен в действие металлургический завод-крепость на реке Исети, названный в честь императрицы Екатеринбургом
- 18 ноября 1842 г. был издан указ о построении первого постоянного моста через Неву, Благовещенского
- 18 ноября 1870 г. между Туром и Парижем открылась первая официальная линия голубиной почты
- 18 ноября 1928 г. в Нью-Йорке прошла премьера первого звукового мультфильма «Пароходик Вилли» с Микки Маусом в главной роли

ВВЕДЕНИЕ

- Основные идеи:
 - (1) классика коллаборативной фильтрации: ближайшие соседи и как их масштабировать;
 - (2) матричные разложения: зачем они нужны, какие бывают, при чём тут рекомендации;
 - (3) расширения: что ещё можно добавить в рекомендательную систему; в частности, рекомендации с анализом контента;
 - (4) неперсонализированные рекомендации: что, если нам нужно быстро реагировать на меняющийся контекст, и персональная статистика не успевает?

- Рекомендательные системы анализируют интересы пользователей и пытаются предсказать, что именно будет наиболее интересно для конкретного пользователя в данный момент времени.
- Компании–лидеры в рекомендательных системах в основном делятся на две категории:
 - (1) мы «продаём» какие-то товары или услуги онлайн; у нас есть пользователи, которые либо явно оценивают товары, либо просто что-то покупают, а что-то нет; интересно порекомендовать товар, который данному покупателю максимально понравится; Netflix, Amazon;
 - (2) мы – портал, делаем деньги тем, что размещаем рекламу, надо разместить ссылки, по которым пользователи захотят переходить (и видеть ещё больше вкусной рекламы); Mail.Ru, Yahoo!, Google, Яндекс, контент-провайдеры, новостные сайты.

Close 

Other Movies You Might Enjoy

[Amélie](#)

Add



Not Interested

[Y Tu Mama Tambien](#)

Add



Not Interested

[Guys and Balls](#)

Add



Not Interested

[Mostly Martha](#)

Add



Not Interested

[Only Human](#)

Add



Not Interested

**Eiken has been added to your Queue at position 2.**

This movie is available now.

[Move To Top Of My Queue](#)[Continue Browsing](#)[Visit your Queue](#)**Witchblade (2006)**

In the wake of a catastrophe that virtually destroys Tokyo, police officer Masane Amaha acquires the legendary Witchblade – a mythical sword bestowed throughout history only to a chosen few – and assumes the identity of a mighty female warrior. With her young daughter's life to protect, Masane's mission is clear. But whether the Witchblade is a righteous weapon of God or a tool of the devil remains to be seen in this anime adventure series.

Starring: Akemi Kanda, Mamiko Noto

Director: Yoshimitsu Ohashi

Genre: Anime & Animation

Rating: TV-MA

**2.8**

Our best guess for Riyadh



3.7 Customer Average

[Witchblade](#)

Add All



Not Interested

amazon.com https://www.amazon.com/gp/yourstore/rate-this-asin/ref=pd_yv_qtk_general_recs_why?ie=UTF8&redirect

amazon.com

Recommended for You



Body by Science
 Our Price: **\$9.99**
Used & new from **\$9.99**

See all buying options

- I want to see more details
- Not interested

Because you purchased...



The Black Swan: Second Edition: The Impact of the Highly Improbable: With a new section: "On Robustness and Fragility" (Kindle Edition)



This was a gift

Frequently Bought Together



Price For All Three: **\$259.02**

Add all three to Cart

- This item:** The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics) by Trevor Hastie
- Pattern Recognition and Machine Learning (Information Science and Statistics)** by Christopher M. Bishop
- Pattern Classification (2nd Edition)** by Richard O. Duda

Customers Who Bought This Item Also Bought

| | | | | |
|---|--|---|---|---|
|  All of Statistics: A Concise Course in Statistics , by Larry Wasserman ★★★★★ (8) \$60.00 |  Pattern Classification (2nd Edition) by Richard O. Duda ★★★★★ (27) \$117.25 |  Data Mining: Practical Machine Learning Tools and Applications , by Ian H. Witten ★★★★★ (29) \$41.55 |  Bayesian Data Analysis, Second Edition (Texts in Applied Mathematics) , by Andrew Gelman ★★★★★ (10) \$56.20 |  Data Analysis Using Regression and Multilevel Modeling , by Andrew Gelman ★★★★★ (13) \$39.59 |
|---|--|---|---|---|

Today's Recommendations For You

Here's a daily sample of items recommended for you. Click here to [see all recommendations](#)

| | | |
|--|---|---|
|  Principles of Data Mining, 1st Edition by David J. Hand ★★★★★ (17) \$52.00 |  Python in a Nutshell, Second Edition by Alex Martelli ★★★★★ (40) \$26.39 |  Introductory Statistics with Applications by Peter Dainoff ★★★★★ (20) \$48.96 |
|--|---|---|

The screenshot displays the Surfingbird web application interface. At the top, there is a navigation bar with a 'surf' logo, a heart icon, a plus sign, a star icon, and a dropdown menu for 'All interests'. To the right, there are icons for social media (Facebook, Twitter, VK) and a user profile icon labeled 'BroDude.ru'. Below the navigation bar, the main content area features a large article header with the text: 'Как тощему чуваку набрать массу: 8 простых шагов'. The interface includes a 'Newsfeed' section with a 'Popular' filter, and a sidebar with 'Your friends on Surfingbird' and social media connection options for Facebook and VK. The main feed contains several article cards, including one about cats and another about the Moon. A large, stylized blue and green bird logo is positioned on the right side of the page, with the text 'Surfingbird' below it.

surf

1 Like 0

All interests

4

f t vk

BroDude.ru

Как тощему чуваку набрать массу: 8 простых шагов

surf

Newsfeed Popular

Surfingbird

Available on the Google play App Store

Your friends on Surfingbird

Connect your account and we'll find your friends

f Connect

vk Connect

Быстрое чтение: котам наплеватель на своих хозяев

ТРП → Теории и практики

Они все понимают, но предпочитают игнорировать.

5 75 0

Нил Шубин о природе времени и образовании Луны

ТРП → Теории и практики

Образование Луны можно сравнить с гонимыми на выживание.

8 151 0

Surfingbird

- У рекомендательной системы есть два разных «уровня», на которых она должна работать:
 - глобальные оценки, медленно меняющиеся особенности и предпочтения, интересные страницы, зависимость от user features (география, пол etc.) и т.д.;
 - кратковременные тренды, hotness, быстрые изменения интереса во времени.

- Это очень разные задачи с разными методами, поэтому различают два класса моделей.
 - *Оффлайн-модели* выявляют глобальные закономерности (обычно это и называется коллаборативной фильтрацией). Цель зачастую в том, чтобы найти и рекомендовать человеку то, что ему понравится, из достаточно редких вещей, работать с «длинными хвостами» распределений интересов людей и веб-страниц.
 - *Онлайн-модели* должны реагировать очень быстро (поэтому там обычно подходы попроще, как правило, не индивидуализированные), они выявляют кратковременные тренды, позволяют рекомендовать то, что hot прямо сейчас.

КЛАССИЧЕСКАЯ КОЛЛАБОРАТИВНАЯ ФИЛЬТРАЦИЯ

- Начнём с коллаборативной фильтрации. Обозначения:
 - индекс i всегда будет обозначать пользователей (всего пользователей будет N , $i = 1..N$);
 - индекс a – предметы (сайты, товары, фильмы...), которые мы рекомендуем (всего M , $a = 1..M$);
 - когда пользователь i оценивает предмет a , он производит отклик (response, rating) $r_{i,a}$; этот отклик – случайная величина, конечно.
- Наша задача – предсказывать оценки $r_{i,a}$, зная признаки x_i и x_a для всех элементов базы и зная некоторые уже расставленные в базе $r_{i',a'}$. Предсказание будем обозначать через $\hat{r}_{i,a}$.

- Метод ближайших соседей: давайте введём расстояние (похожесть) между пользователями и будем рекомендовать то, что нравится вашим соседям (похожим на вас).
- Расстояние:
 - коэффициент корреляции (коэффициент Пирсона)

$$w_{i,j} = \frac{\sum_a (r_{i,a} - \bar{r}_a)(r_{j,a} - \bar{r}_a)}{\sqrt{\sum_a (r_{i,a} - \bar{r}_a)^2} \sqrt{\sum_a (r_{j,a} - \bar{r}_a)^2}},$$

где \bar{r}_a – средний рейтинг продукта a среди всех пользователей;

- косинус угла между векторами рейтингов:

$$w_{i,j} = \frac{\sum_a r_{i,a} r_{j,a}}{\sqrt{\sum_a r_{i,a}^2} \sqrt{\sum_a r_{j,a}^2}}.$$

- Простейший способ построить предсказание нового рейтинга $\hat{r}_{i,a}$ – сумма рейтингов других пользователей, взвешенная их похожестью на пользователя i :

$$\hat{r}_{i,a} = \bar{r}_a + \frac{\sum_j (r_{j,a} - \bar{r}_j) w_{i,j}}{\sum_j |w_{i,j}|}.$$

- Это называется GroupLens algorithm – так работал дедушка рекомендательных систем GroupLens.
- Чтобы не суммировать по всем пользователям, можно ограничиться ближайшими соседями:

$$\hat{r}_{i,a} = \bar{r}_a + \frac{\sum_{j \in \text{knn}(i)} (r_{j,a} - \bar{r}_j) w_{i,j}}{\sum_{j \in \text{knn}(i)} |w_{i,j}|}.$$

- Естественно предположить, что продукты, которые любят или не любят практически все пользователи, не слишком полезны в определении ближайшего соседа.
- Поэтому естественно взвесить продукты по тому, как часто их уже оценивали пользователи; такая метрика называется iuf – inverse user frequency, обратная частота пользователей: $f_a = \log \frac{N}{N_a}$, где N – общее число пользователей, N_a – число оценивших продукт a . Получается

$$w_{i,j}^{\text{iuf}} = \frac{\sum_a f_a \sum_a f_a r_{i,a} r_{j,a} - (\sum_a f_a r_{i,a}) (\sum_a f_a r_{j,a})}{\sqrt{\sum_a f_a (\sum_a f_a r_{i,a}^2 - (\sum_a f_a r_{i,a})^2)} \sqrt{\sum_a f_a (\sum_a f_a r_{j,a}^2 - (\sum_a f_a r_{j,a})^2)}},$$

а для косинуса

$$w_{i,j}^{\text{iuf}} = \frac{\sum_a f_a^2 r_{i,a} r_{j,a}}{\sqrt{\sum_a (f_a r_{i,a})^2} \sqrt{\sum_a (f_a r_{j,a})^2}}.$$

- Симметричный подход – item-based collaborative filtering. Считаем похожесть между продуктами, выбираем похожие продукты.
- Amazon: customers who bought this item also bought...
- Преимущество – может быть эффективнее за счёт того, что похожесть продуктов всегда можно считать оффлайн, пара новых оценок не повлияет на неё совсем радикально.
- Считаем похожесть между парами продуктов, у которых есть общий оценивший пользователь.

КАК МАСШТАБИРОВАТЬ БЛИЖАЙШИХ СОСЕДЕЙ

- Искать ближайших соседей довольно сложно алгоритмически (k-d-деревья в больших размерностях плохо работают).
- В действительно больших рекомендательных системах используют приближённые методы.
- Например, LSH (locality sensitive hashing) на min-hash:
 - заведём несколько хеш-функций, посчитаем их от каждого продукта;
 - для каждого пользователя вычислим минимальное значение хеш-функций на его продуктах;
 - будем соседей искать только среди тех пользователей, у которых есть одинаковые значения хотя бы в одном хеше.

ЧТО ДЕЛАТЬ С ЛАЙКАМИ?

- Заметим, что мы до сих пор рассматривали только системы, в которых есть явно поставленные рейтинги, некоторые хорошие, некоторые плохие.
- Но в реальной ситуации часто есть только множества «потреблённых» продуктов I и J для пользователей i и j :
 - лайки (дизлайки мало кто ставит);
 - купленные товары без явных рейтингов.
- Это называется implicit feedback. Что делать?

ЧТО ДЕЛАТЬ С ЛАЙКАМИ?

- Надо определить расстояние между множествами; похожесть по Жаккару

$$w_{i,j} = \text{Jaccard}(I, J) = \frac{|I \cap J|}{|I \cup J|}.$$

- Можно так ввести веса между пользователями и дальше рекомендовать по GroupLens.

ЧТО ДЕЛАТЬ С ЛАЙКАМИ?

- Но ещё чаще похожесть по Жаккару используется для поиска других продуктов, «похожих на».
- Определим похожесть между a и b через множества потребивших их пользователей A и B :

$$w_{a,b} = \text{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

- Часто хорошо работает, но есть тонкие эффекты.
- Например, что если продукт a редкий, а b более популярный, и так получилось, что почти все $i \in A$ также потребили и b ? (это очень реальная ситуация)

ЧТО ДЕЛАТЬ С ЛАЙКАМИ?

- Похожесть по Жаккару плохо подходит в несбалансированных случаях, потому что она симметричная.
- Давайте сделаем несимметрично:

$$w_{a,b} = \frac{|A \cap B|}{|A|}, \quad w_{b,a} = \frac{|A \cap B|}{|B|}.$$

- Теперь с предыдущим примером всё хорошо, но и тут есть тонкие эффекты.
- Что если один из продуктов очень популярен, и почти все его видели? Banana trap.

ЧТО ДЕЛАТЬ С ЛАЙКАМИ?

- Ещё одна вариация – метод ассоциаций:

$$w_{a,b} = \frac{|A \cap B| / |A|}{|\bar{A} \cap B| / |\bar{A}|},$$

где \bar{A} – дополнение A .

- На практике обычно стоит просто попробовать все эти варианты и выбрать лучший.

МЕТРИКИ КАЧЕСТВА

- Ещё одна важная тема, которую в рекомендательных системах часто замалчивают.
- Как оценить качество рекомендаций? Какая должна быть метрика качества?
- Когда мы обучаем SVD (максимизируем правдоподобие), мы оптимизируем среднеквадратичную ошибку отклонения.
- И Netflix Prize, например, так и был сформулирован: надо было оптимизировать RMSE, среднеквадратичное отклонение предсказаний от истинного рейтинга пользователя.
- Но что нам надо на самом деле? Что у нас в тестовом множестве?

- В тестовом множестве отложены рейтинги некоторых продуктов, которым пользователь дал оценку.
- Наша задача – выдать пользователю топ-рекомендации; не предсказать все рейтинги, а найти, у каких продуктов рейтинг будет самым большим.
- То есть на самом деле это задача *ранжирования*! И метрики качества лучше брать из *information retrieval*; там, когда оценивают качество выдачи, вовсе не пытаются минимизировать отклонение предсказания функции релевантности.
- Дальше для простоты будем рассматривать бинарный случай (лайк-дизлайк).

- Классические метрики:
 - (1) точность (precision) – количество «хороших» (релевантных) запросу в случае поисковой выдачи, отмеченных высокой оценкой в случае рекомендательной системы) документов в выдаче, делённое на общее количество документов в выдаче;
 - (2) полнота (recall) – количество «хороших» документов в выдаче, делённое на общее количество релевантных документов в базе поисковой системы.
- Однако здесь те же проблемы; эти параметры не зависят от ранжирования выдачи, надо знать заранее, сколько потребуется рекомендаций.

- Метрики качества ранжирования:
 - NDCG, Normalized Discounted Commulative Gain; выберем топ- k рекомендаций (k может быть заведомо больше нужного числа) и посчитаем:

$$\text{DCG}_k = \sum_{i=1}^k \frac{2^{\hat{r}_i} - 1}{\log_2(1 + i)},$$
$$\text{NDCG}_k = \frac{\text{DCG}_k}{\text{IDCG}_k},$$

где \hat{r}_i – наша оценка рейтинга продукта на позиции i , а IDCG_k – значение DCG_k при ранжировании по истинным значениям (рейтингам из валидационного набора);

- NDCG от 0 до 1, но ей трудно придумать естественную интерпретацию (как вероятность чего-нибудь, например).

- Метрики качества ранжирования:
 - AUC, Area Under (ROC) Curve; можно считать по всей выдаче сразу;
 - AUC – вероятность того, что случайно выбранная пара продуктов с разными оценками будет отранжирована правильно (понравившийся будет выше в выдаче, чем не понравившийся);
 - в бинарном случае можно посчитать в замкнутом виде:

$$\hat{A} = \frac{S_0 - n_0(n_0 + 1)/2}{n_0 n_1},$$

где n_0, n_1 – число понравившихся и не понравившихся пользователю объектов, $S_0 = \sum p_i$ – сумма номеров позиций понравившихся объектов в выдаче.

- Метрики качества ранжирования:
 - но на самом деле простые метрики тоже важны, потому что обычно пользователь успеет увидеть только несколько самых верхних рекомендаций;
 - WTA (winner takes all) – эта метрика равна 1, если топ-рекомендация (с самым большим предсказанным рейтингом) из просмотренных пользователем получила положительную оценку, и 0 в противном случае;
 - Top k – доля положительных оценок среди топ- k рекомендаций (Top10 часто называют MAP – mean average precision).

Спасибо за внимание!