Технічні науки

УДК 004.021

Прасолов Андрій Павлович

студент

Національного технічного університету України

«Київський політехнічний інститут імені Ігоря Сікорського»

Прасолов Андрей Павлович

студент

Национального технического университета Украины

«Киевский политехнический институт имени Игоря Сикорского»

Prasolov Andriy

Student of the

National Technical University of Ukraine

"Igor Sikorsky Kyiv Polytechnic Institute"

МЕТОДИ КОЛАБОРАТИВНОЇ ФІЛЬТРАЦІЇ У РЕКОМЕНДАЦІЙНИХ СИСТЕМАХ
МЕТОДЫ КОЛЛАБОРАТИВНОЙ ФИЛЬТРАЦИИ В РЕКОМЕНДАЦИОННЫХ СИСТЕМАХ
METHODS OF COLLABORATIVE FILTERING IN RECOMMENDER SYSTEMS

Анотація. *Розглянуто рекомендаційні алгоритми, які оцінюють схожість користувача (продукту) на інших користувачів (інші продукти).*

Ключові слова: *рекомендаційні системи, колаборативна фільтрація.*

Аннотация. *Рассмотрены рекомендательные алгоритмы, которые оценивают сходство пользователя (продукта) на других пользователей (другие продукты).*

Ключевые слова: *рекомендательные системы, коллаборативная фильтрация.*

Summary. *Recommendation algorithms that evaluate the similarity of the user (product) to other users (other products) are considered.*

Key words: *recommender systems, collaborative filtering.*

У сучасному світі часто доводиться стикатися з проблемою рекомендації товарів або послуг користувачам якоїсь інформаційної системи. У старі часи для формування рекомендацій обходилися зведенням найбільш популярних продуктів. Але з часом такі рекомендації стали витіснятися цільовими пропозиціями: користувачам рекомендуються не просто популярні продукти, а ті продукти, які напевно сподобаються саме їм. Отже, перед нами стоїть проблема, як краще надати користувачу потрібний саме йому товар і зробити це найкращим співвідношенням витрачених на це ресурсів та результату (якості рекомендації).

На вході у нас є дуже розріджена матриця, що складається з рейтингів (лайків, фактів покупки і т.п.), які користувачі (рядки матриці) присвоїли продуктам (стовпці матриці):

$$R = (r_{i,a})_{i=1,a=1}^{N,M} \qquad (1)$$

Наше завдання - прогнозувати оцінки $r_{i,a}$, знаючи деякі вже розставлені в матриці оцінки $r_{i',a'}$. Наше найкраще передбачення будемо позначати через $\bar{r}_{i,a}$. Якщо ми зможемо отримати такі передбачення, потрібно буде просто вибрати один або кілька продуктів, для яких $\bar{r}_{i,a}$ максимальні.

Розглянемо два підходи: або шукати схожих користувачів - це називається «рекомендації, засновані на користувачів» (user-based collaborative filtering), або шукати схожі продукти - це, що логічно, називається «рекомендації, засновані на продуктах» (item-based collaborative

filtering). Власне, основний алгоритм в обох випадках зрозумілий. Знайти, наскільки інші користувачі (продукти) в базі даних схожі на даного користувача (продукт).

1. Знайти, наскільки інші користувачі (продукти) в базі даних схожі на даного користувача (продукт).
2. За оцінками інших користувачів (продуктів) передбачити, яку оцінку дасть даний користувач даному продукту, враховуючи з великою вагою тих користувачів (продукти), які більше схожі на даний..

По-перше, потрібно визначити, що означає «схожий». Нагадую, що все, що у нас є - це вектор вподобань для кожного користувача (рядки матриці R) і вектор оцінок користувачів для кожного продукту (стовпці матриці R). Перш за все залишимо в цих векторах тільки ті елементи, для яких нам відомі значення в обох векторах, тобто залишимо тільки ті продукти, які оцінили обидва користувачі, або тільки тих користувачів, які обидва оцінили даний продукт. В результаті нам просто потрібно визначити, наскільки схожі два вектора дійсних чисел. Це, звичайно, відома задача, і класичне її рішення - підрахувати коефіцієнт кореляції: для двох векторів переваг користувачів i і j коефіцієнт кореляції Пірсона дорівнює

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

де $\bar{r}_i$ — середній рейтинг, виставлений користувачем i. Можна користоватися так званою «косинусной схожістю», використовуючи косинус кута між векторами:

$$w_{i,j} = \frac{\sum_a r_{i,a} - r_{j,a}}{\sqrt{\sum_a r_{i,a}^2} \sqrt{\sum_a r_{j,a}^2}} \qquad (3)$$

Але для того, щоб косинус добре працював, бажано все одно спочатку відняти середнє по кожному вектору, так що в реальності це та ж сама метрика.

Для прикладу розглянемо якусь матрицю оцінок.

Таблиця 1

	Фільм1	**Фільм2**	**Фільм3**	**Фільм4**	**Фільм5**
Вова	?	3	4	5	2
Діма	3	5	2	2	5
Катя	5	3		4	3
Оля	5	5	5		4
Вітя	2	3		2	2

Для user-based рекомендацій кореляцію між вектором переваг Вови і інших учасників системи.

	User-base кореляція
Діма	-0.8944
Катя	0.9449
Оля	0.8660
Вітя	-0.1890

Ми зараз привели формули для user-based рекомендацій. У item-based підході ситуація схожа, але є один нюанс: різні користувачі по-різному ставляться до оцінок, хтось ставить всім підряд по п'ять зірочок («лайкати» все поспіль), а хтось, навпаки, ставить всім по дві-три зірочки (часто тисне «дізлайк»). Для першого користувача низький рейтинг («дізлайк») буде набагато більш інформативний, ніж високий, а для другого - навпаки. У user-based підході про це автоматично дбає коефіцієнт кореляції. А в item-based рекомендаціях, щоб це врахувати, можна, наприклад, відняти від кожної оцінки середній рейтинг того чи іншого користувача, а потім вже підрахувати кореляцію або косинус кута між векторами. Тоді у формулі для косинуса вийде

$$w_{i,j} = \frac{\sum_i (r_{i,a} - \bar{r}_i)(r_{i,b} - \bar{r}_i)}{\sqrt{\sum_i (r_{i,a} - \bar{r}_i)^2}\sqrt{\sum_i (r_{i,b} - \bar{r}_i)^2}} \qquad (4)$$

де $\bar{r}_i$ позначає середній рейтинг, виставлений користувачем *i*. У нашому прикладі, якщо відняти від кожного вектора оцінок середнє, вийде ось що:

Таблиця 2

	Фільм1	**Фільм2**	**Фільм3**	**Фільм4**	**Фільм5**
Вова	?	-0.5	0.5	1.5	-1.5
Діма	-0.4	1.6	-1.4	-1.4	1.6
Катя	1.25	-0.75		0.25	-0.75
Оля	0.25	0.25	0.25		-0.75
Вітя	-0.25	0.75		-0.25	-0.25

І тоді кореляція між вектором оцінок фільму «Фільм1» і оцінками інших фільмів складе (зауважимо, що з «Фільм3» склалася вироджена ситуація, тому що оцінок, які перетинаються, було занадто мало)

	Item-base кореляція
Фільм2	-0.9545
Фільм3	0.7870
Фільм4	0.7870
Фільм5	-0.6689

У цих заходів схожості є свої недоліки і різноманітні варіації на тему, але давайте для ілюстрації методів ними обмежимося. Як скористатися цими оцінками схожості (весами $w_{i,j}$)?

Простий і логічний підхід - наблизити новий рейтинг як середній рейтинг даного користувача плюс відхилення від середнього рейтингів інших користувачів, зважених цими самими вагами:

$$\bar{\bar{r}}_{i,j} = \bar{\bar{r}}_i + \frac{\sum_i (r_{i,a} - \bar{r}_j) w_{i,j}}{\sum_j |w_{i,j}|} \qquad (5)$$

Цей підхід іноді ще називають GroupLens algorithm. У випадку з Вовою і «Термінатором» за цим методом очікується оцінка близько 4.1, так що можна сміливо дивитися.

Для item-based рекомендацій все абсолютно еквівалентно - потрібно просто знайти зважене середнє вже оцінених користувачем продуктів:

$$\bar{\bar{r}}_{i,j} = \bar{\bar{r}}_i + \frac{\sum_b (r_{i,b} - \bar{r}_i) w_{a,b}}{\sum_b |w_{a.b}|} \qquad (6)$$

Item-based метод в нашому прикладі передбачає, що Вова поставить «фільму 1» аж 4.4.

Звичайно, теоретично все це добре, але в реальності навряд чи вийде для кожної рекомендації підсумувати дані від мільйонів користувачів. Тому цю формулу зазвичай спрощують до k найближчих сусідів - k користувачів, максимально схожих на даного користувача і вже оцінили цей продукт:

$$\bar{\bar{r}}_{i,a} = \bar{\bar{r}}_i + \frac{\sum_{j \in kNN(i)} (r_{j,a} - \bar{r}_j) w_{i,j}}{\sum_{j \in kNN(i)} |w_{i,j}|} \qquad (7)$$

Залишається тільки зрозуміти, як швидко шукати найближчих сусідів. Два основні методи: в невеликих размерностях можна користуватися k-d-деревами (k-d-trees), а в великих размерностях - локально-чутливе хешування (locally sensitive hashing).

Отже, ми розглянули рекомендаційні алгоритми, які оцінюють схожість користувача (продукту) на інших користувачів (інші продукти), а потім рекомендують, виходячи з думки найближчих сусідів по цій метриці.

Література

1. Item-based collaborative filtering – Режим доступа: http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html – Дата доступа: 15.03.2018.
2. Коллаборативная фильтрация — WitologyWiki – Режим доступа: http://wiki.witology.com/index.php/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D1%84%D0%B8%D0%BB%D1%8C%D1%82%D1%80%D0%B0%D1%86%D0%B8%D1%8F – Дата доступа: 15.03.2018.