Рекомендаційні системи

для інтернет-ресурсів, що використовують алгоритми машинного навчання

Об’єкт дослідження

- Статистичні дані, які містять оцінки вподобань користувачами деяких об’єктів

Предмет дослідження

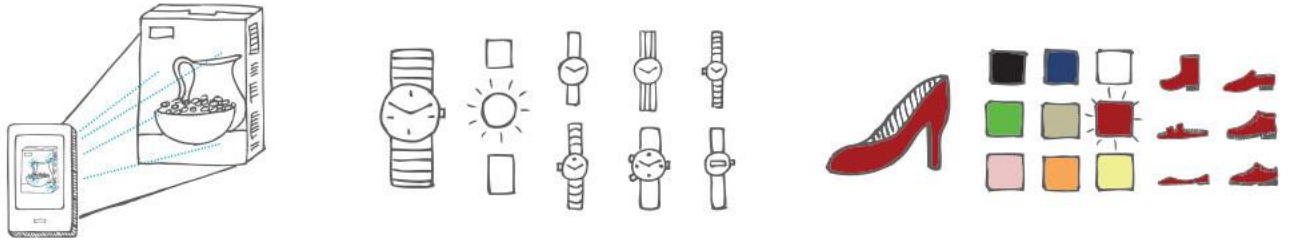
- Застосування алгоритмів машинного навчання для пошуку оптимальних значень кількісних оцінок вподобань користувачами об’єктів у рамках інтернет-ресурсів

Мета дослідження

- Проаналізувати предмет дослідження та дослідити ефективність застосування алгоритмів машинного навчання до задач рекомендаційних систем

Актуальність

Рекомендаційні системи — один з найбільш популярних додатків інтелектуального аналізу даних і машинного навчання в сфері інтернет-бізнесу. Вони аналізують поведінку користувачів інтернет-сервісу, після чого дають кількісну та якісну оцінку вподобання користувачем того чи іншого об’єкту. Об’єктами рекомендацій можуть бути товари в інтернет-магазині, набір розділів веб-сайту, медіа-контент, інші користувачі веб-сервісу.



В сучасних умовах нагромадження даних рекомендаційні системи є незамінним механізмом пошуку контенту.

Основні підходи до побудови рекомендаційних систем

Виділяють три основні підходи:

- Підхід на підставі ознакових описів (content-based filtering)

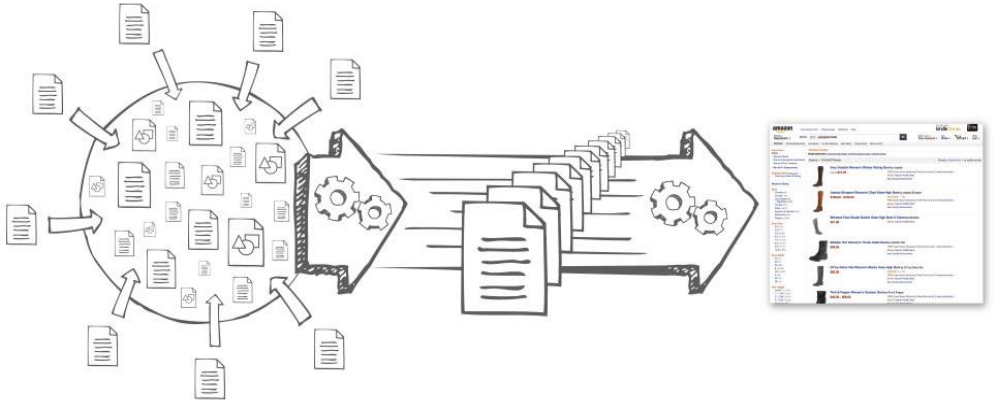
передбачає, що про користувачів і про рекомендовані об’єкти відомо досить багато інформації, по якій і будується набір рекомендацій.

- Колаборативна фільтрація (collaborative filtering)

набір рекомендацій будується виключно на підставі взаємодії користувачів з об’єктами.

- Гібридний підхід (hybrid filtering)

використовує композиції алгоритмів заснованих на ознакових описах і результатах колаборативної фільтрації.



Огляд моделей колаборативної фільтрації

Підходи до вирішення завдання колаборативної фільтрації умовно можна розділити на дві великі групи:

- Засновані на евристиках (memory / heuristic-based)

передбачається пошук оцінок вподобань за допомогою міри схожості між собою користувачів або об’єктів.

- Засновані на побудові моделі вподобання (model-based)

передбачається побудова моделі машинного навчання, яка враховує латентні (скриті) параметри користувачів та об’єктів.

Формалізація задачі колаборативної фільтрації

Нехай $\boldsymbol{U}$ — множина користувачів (users), $\boldsymbol{I}$ — множина об'єктів (items).

Інформація про відомі вподобання представлена у вигляді набору трійок:

$$\mathcal{D} = \{(u, i, r_{ui})\}_{(u,i)\in R},$$

де $r_{ui} \in \mathbb{R}$ — кількісна величина вподобання об'єкта $i \in I$ користувачем $u \in U$,

$\mathbf{R} \subseteq U \times I$ — множина пар (користувач, об'єкт), про які відома ступінь вподобання.

Для подальшої зручності, введемо також позначення:

$\mathbf{R}(u) = \{i : (u, i) \in \mathbf{R}\}$ — множина об'єктів, суміжних з користувачем u,

$\mathbf{R}(i) = \{u : (u, i) \in \mathbf{R}\}$ — множина користувачів, суміжних з об'єктом i.

За відомою інформацією $\mathcal{D}$ потрібно вміти будувати передбачення вподобання $\hat{r}_{ui} \approx r_{ui}$ для нових пар $(u, i) \notin \mathbf{R}$.

Матрицею оцінок будемо називати матрицю $\mathbf{R} \in (\mathbb{R} \cup \emptyset)^{|U|\times|I|}$, рядки якої відповідають користувачам, стовпці — об'єктам, а елементи приймають значення r_{ui}, якщо $(u, i) \in \mathbf{R}$, інакше пропуск — $\emptyset$.

Структура матриці оцінок $\mathbf{R}$

В ході дослідження маємо велику розріджену матрицю, що має подібну структуру:

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	5	4	5			
User 2	4		5			
User 3		3	5		4	
User 4				3	4	
User 5			4	2	4	
User 6	3					5

Memory / heuristic-based алгоритм

Memory-based алгоритм колаборативної фільтрації являє собою зважування вподобань по користувачах (user-based) (1) і по об’єктах (item-based) (2).

$$\hat{r}_{ui} = \bar{r}_u + \frac{1}{\sum_{u' \in R(i)} |sim(u,u')|} \sum_{u' \in R(i)} sim(u,u')(r_{u',i} - \bar{r}_{u'}) \quad (1)$$

$$\hat{r}_{ui} = \bar{r}_i + \frac{1}{\sum_{i' \in R(u)} |sim(i,i')|} \sum_{i' \in R(u)} sim(i,i')(r_{u,i'} - \bar{r}_{i'}) \quad (2)$$

де $\bar{r}_u = \frac{1}{|R(u)|} \sum_{i \in R(u)} r_{ui}$, $\bar{r}_i = \frac{1}{|R(i)|} \sum_{u \in R(i)} r_{ui}$ — середні значення вподобань по користувачам і об’єктам, а $sim(u,u')$, $sim(i,i')$ — метрики схожості користувачів і об’єктів.

Міра схожості $sim(u,u')$ (і аналогічна для об’єктів) обчислюється за матрицею оцінок R. Найбільш загальновживані прості метрики схожості — кореляція Пірсона (3) і косинусна відстань (4) відповідних рядків (стовпців) матриці оцінок:

$$sim(u,u') = \frac{\sum_{i \in R(u) \cap R(u')} (r_{u,i} - \bar{r}_u)(r_{u',i} - \bar{r}_{u'})}{\sqrt{\sum_{i \in R(u) \cap R(u')} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in R(u) \cap R(u')} (r_{u',i} - \bar{r}_{u'})^2}} \quad (3)$$

$$sim(u,u') = \frac{\sum_{i \in R(u) \cap R(u')} r_{u,i} r_{u',i}}{\sqrt{\sum_{i \in R(u)} r_{u,i}^2 \sum_{i \in R(u')} r_{u',i}^2}} \quad (4)$$

SVD – розклад в машинному навчанні

SVD (Single Value Decomposition) — сингулярний розклад матриці. В теоремі про сингулярний розклад стверджується, що для будь-якої матриці $\mathbf{A}$ розміру $\boldsymbol{n \times m}$ існує розклад у добуток трьох матриць: $\boldsymbol{U}, \Sigma$ и $\boldsymbol{V^T}$ ($\boldsymbol{U}$ і $\boldsymbol{V}$ ортогональні, а Σ — діагональна):

$$\underset{n\times m}{\boldsymbol{A}} = \underset{n\times n}{\boldsymbol{U}} \times \underset{n\times m}{\Sigma} \times \underset{m\times m}{\boldsymbol{V^T}}$$

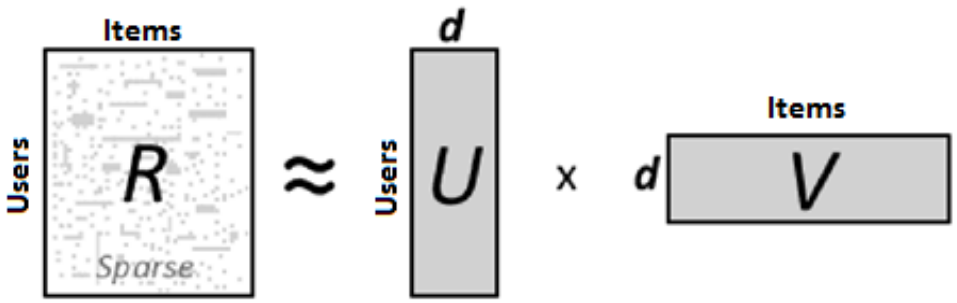
$$\boldsymbol{UU^T = I_n}, \qquad \boldsymbol{VV^T = I_m}, \qquad \Sigma = \boldsymbol{diag}\left(\lambda_1, \dots, \lambda_{\min(n,m)}\right),\ \boldsymbol{\lambda_1 \geq \cdots \geq \lambda_{\min(n,m)} \geq 0}$$

Крім звичайного розкладу буває ще зрізаний: $\underset{n\times m}{\boldsymbol{A'}} = \underset{n\times d}{\boldsymbol{U'}} \times \underset{d\times d}{\Sigma'} \times \underset{d\times m}{\boldsymbol{V'^T}}$, $\boldsymbol{\lambda_{d+1} = \cdots = \lambda_{\min(n,m)} = 0}$,

$\boldsymbol{d}$ — величина низькорангового наближення

Матрицю вподобань розкладаємо в добуток трьох матриць. Спростимо все, позначивши добуток перших двох матриць як одну матрицю.

Щоб передбачити оцінку користувача $\boldsymbol{U}$ фільму $\boldsymbol{I}$, візьмемо деякий вектор $\boldsymbol{p_u}$ (набір параметрів для даного користувача і вектор для даного об'єкту (наприклад, фільму) $\boldsymbol{q_i}$. Їх скалярний добуток і буде потрібним передбаченням величини вподобання: $\boldsymbol{\hat{r}_{ui} = \langle p_u, q_i \rangle}$



Model-based алгоритм

Model-based алгоритми шукають оцінку $\hat{r}$, як функцію $\hat{r}(u, i; \theta)$ з деякого сімейства функцій параметру $\theta \in \Theta$. Модель машинного навчання реалізується шляхом мінімізації регуляризованого емпіричного ризику:

$$\mathcal{L}(\theta) = \sum_{(u,i)\in R} l(\hat{r}(u, i; \theta), r_{ui}) + \lambda\Omega(\theta) \to \min_{\theta\in\Theta},$$

де $l(\hat{r}, r)$ — функція втрат регресії, λ — сила регуляризації,

$\Omega(\theta)$ — регуляризатор на множині параметрів Θ.

Використовуючи результати SVD-розкладу, маємо таке представлення функції $\hat{r}(u, i; \theta)$:

$$\hat{r}(u, i; \theta) = p_u^T q_i, \ \theta = (\{p_u\}_{u\in U}, \{q_i\}_{i\in I}),$$

де $p_u \in \mathbb{R}^d$ — вектор прихованих (латентних) ознак користувача u,

$q_i \in \mathbb{R}^d$ — аналогічний вектор для об'єкта i,

d — розмірність латентних векторів (величина низькорангового наближення). Моделі, які будуть розглядатися далі, навчаються шляхом оптимізації квадратичних втрат $l(\hat{r}, r) = (\hat{r} - r)^2$ з квадратичною регуляризацією: $\Omega(\theta) = \sum_{u\in U}||p_u||^2 + \sum_{i\in I}||q_i||^2$ (евклідові норми).

Алгоритм машинного навчання SVD

Модель, що отримано на минулому кроці, зветься моделлю SVD та має вигляд:

$$\mathbf{J}(\boldsymbol{p_u}, \boldsymbol{q_i}) = \sum_{(\boldsymbol{u},\boldsymbol{i})\in\boldsymbol{\mathcal{D}}} (\boldsymbol{r_{ui}} - \boldsymbol{p_u^T q_i})^2 + \boldsymbol{\lambda}\left(\sum_{\boldsymbol{u}} ||\boldsymbol{p_u}||^2 + \sum_{\boldsymbol{i}} ||\boldsymbol{q_i}||^2\right) \longmapsto \min_{\boldsymbol{p_u},\boldsymbol{q_i}},$$

де $\boldsymbol{p_u}$ — вектор латентних ознак користувача $\boldsymbol{u}$, $\boldsymbol{q_i}$ — аналогічний вектор для об'єкта $\boldsymbol{i}$,

$\boldsymbol{\lambda}$ — сила регуляризації

Регуляризація потрібна для боротьби з перенавчанням — явищем, коли побудована модель добре пояснює приклади з навчальної вибірки, але погано працює на тестових прикладах. Регуляризація полягає в тому, що оптимізується помилка плюс деяка функція від параметрів (норма вектору параметрів). Це дозволяє обмежити розмір параметрів у розв'язку та зменшує ступінь свободи моделі.

Навчати параметри моделі будемо алгоритмом градієнтного спуску.

Візьмемо часткові похідні по кожній із змінних, що оптимізуються, та отримаємо прості правила для градієнтного спуску моделі:

$$\boldsymbol{p_{u,j}} = \boldsymbol{p_{u,j}} - \boldsymbol{\gamma}\left(\boldsymbol{e_{ui}q_{i,j}} + \boldsymbol{\lambda p_{u,j}}\right), \quad \boldsymbol{q_{i,j}} = \boldsymbol{q_{i,j}} - \boldsymbol{\gamma}\left(\boldsymbol{e_{ui}p_{u,j}} + \boldsymbol{\lambda q_{i,j}}\right), \forall \boldsymbol{j}$$

де $\boldsymbol{e_{ui}} = \boldsymbol{r_{ui}} - \hat{\boldsymbol{r}}_{\boldsymbol{ui}}$ — помилка на даному тестовому прикладі, $\boldsymbol{\gamma}$ — швидкість навчання.

Алгоритм машинного навчання SVD++

Покращенням попередньої моделі є модель SVD++. Вона враховую користувачів, які оцінюють товари або тільки добре, або тільки погано; враховує товари, які кращі за інші (або більш прорекламовані), а також враховує загальний середній рейтинг по базі.

Тоді оцінка вподобання приймає вигляд:

$$\hat{r}_{ui} = \mu + b_u + b_i + p_u^T q_i \ ,$$

де b_u — базові предиктори окремих користувачів, b_i — базові предиктори окремих об'єктів,

μ — загальний середній рейтинг по базі

Модель SVD++ виглядає наступним чином:

$$J(\mu, b_u, b_i, p_u, q_i) = \sum_{(u,i)\in D} \left(r_{ui} - \mu - b_u - b_i - p_u^T q_i\right)^2 + \lambda \left(\sum_u b_u^2 + \sum_i b_i^2 + ||p_u||^2 + ||q_i||^2 \right) \longmapsto \min_{\mu, b_u, b_i, p_u, q_i}$$

Параметри будуть навчатися за такими правилами:

$$b_u = b_u - \gamma(e_{ui} + \lambda b_u), \qquad b_i = b_i - \gamma(e_{ui} + \lambda b_i),$$
$$p_{u,j} = p_{u,j} - \gamma\left(e_{ui} q_{i,j} + \lambda p_{u,j}\right), \qquad q_{i,j} = q_{i,j} - \gamma\left(e_{ui} p_{u,j} + \lambda q_{i,j}\right) \qquad \forall j,$$

де $e_{ui} = r_{ui} - \hat{r}_{ui}$ — помилка на даному тестовому прикладі, γ — швидкість навчання.

Вимірювання якості рекомендацій

Для поліпшення якості рекомендацій треба навчитися її вимірювати. Для цього алгоритм, навчений на одній вибірці — навчальній, перевіряється на іншій — тестовій. Запропоновано вимірювати якість рекомендацій за метрикою **RMSE**:

$$\mathbf{RMSE} = \sqrt{\frac{\mathbf{1}}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} (r_{ui} - \hat{r}_{ui}(\boldsymbol{\theta}))^2}$$

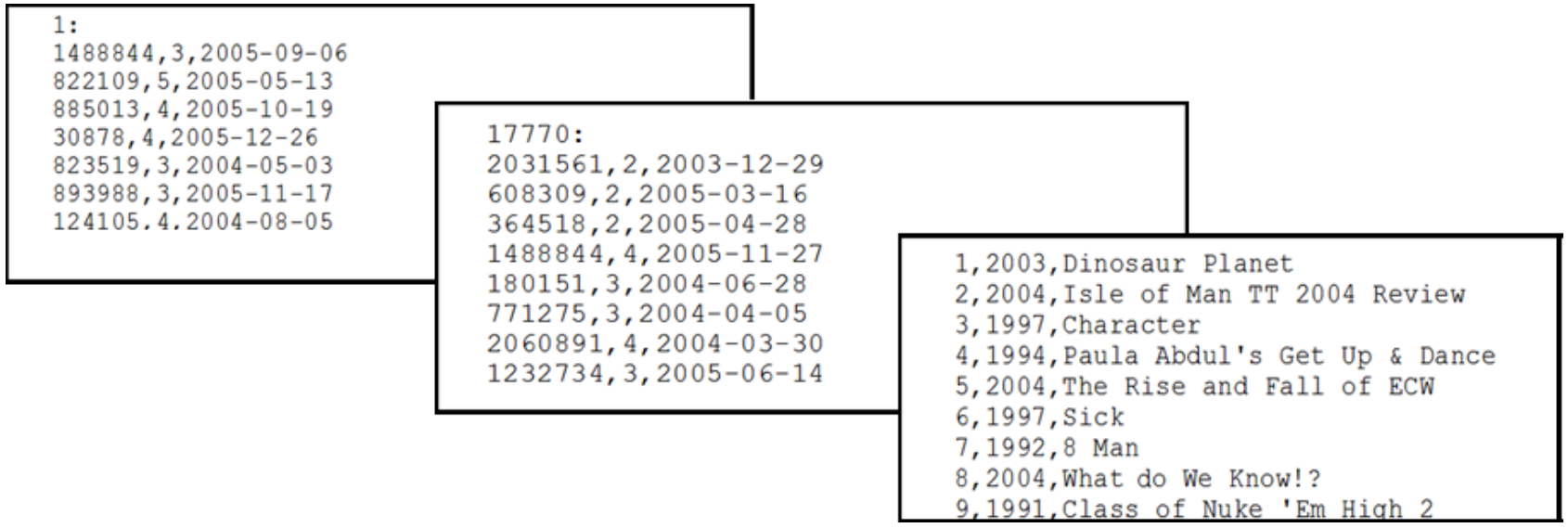
Це стандартна метрика для передбачення оцінки. Її недоліки:

- У кожного користувача своє уявлення про шкалу оцінок. Користувачі, у яких інтервал оцінок ширший, будуть більш впливати на значення метрики, ніж інші.
- Помилка в прогнозі високої оцінки має таку ж вагу, що і помилка в прогнозі низької оцінки.

Опис вхідних даних

В якості дослідних даних розглянемо датасет, що був запропонований на змаганні **Netflix Prize** у 2006 році:

- Навчальні дані (training data set) містять **100.480.507** оцінок, які **480.189** клієнтів поставили **17.770** фільмів. Кожна оцінка являє собою набір **<номер клієнта, номер фільму, дата оцінки, оцінка>**. Номери клієнтів і фільмів — цілі числа, оцінка — ціле число **від 1 до 5** (нижча оцінка — 1, вища — 5). Таким чином, в середньому кожен клієнт поставив близько 200 оцінок, а кожен фільм отримав близько 5000. Кількість оцінок сильно варіюється: так, деякі фільми отримали всього 3 оцінки, а один клієнт оцінив понад 17.000 фільмів



Матриця оцінок вподобань

Складемо матрицю оцінок, яка матиме розмір 17.770×480.189 (кількість фільмів × кількість клієнтів). Щоб навчати наші моделі, приберемо половину оцінок із матриці довільним чином. Оскільки матриця має велику розмірність, демонструємо лише її частину:

	1	2	3	4	5	6	7
1	5	?	2	?	5	?	4
2	5	2	?	?	3	?	?
3	**?**	?	1	3	?	2	1
4	4	4	?	?	?	5	?
5	?	?	1	?	4	?	?
6	1	?	2	3	?	?	5
7	?	5	?	?	3	5	?
8	3	?	4	5	?	?	2
9	?	2	?	?	5	1	?
10	?	?	4	?	3	?	5

Розглянемо кожен з алгоритмів. В якості демонстрації порахуємо виділений елемент.

User-based та Item-based алгоритми

В рамках user-based алгоритму знайдемо міру схожості користувача №3 з іншими користувачами — попарну кореляцію між векторами оцінок фільмів від користувача №3 та вектором оцінок всіх інших 480.188 користувачів (враховуємо лише ті оцінки фільмів, які проставлені в обох користувачів) і скористаємося формулою зважування вподобань для користувачів — оцінка **3.9**:

	1	2	3	4	5	6	7
1	5	?	2	?	5	?	4
2	5	2	?	?	3	?	?
3	3.9	?	1	3	?	2	1
4	4	4	?	?	?	5	?
5	?	?	1	?	4	?	?

Item-based алгоритм шукає схожі між собою фільми за оцінками користувачів — попарну кореляцію між векторами оцінок від всіх користувачів фільму №1 та векторами оцінок всіх інших 17.769 фільмів. Використовуємо формулу зважування вподобань для об'єктів, отримуємо оцінку $\mathbf{4.4}$ — вподобання користувачем №3 фільму №1:

	1	2	3	4	5	6	7
1	5	?	2	?	5	?	4
2	5	2	?	?	3	?	?
3	4.4	?	1	3	?	2	1
4	4	4	?	?	?	5	?
5	?	?	1	?	4	?	?

Алгоритм SVD

r_{ui} — відомі оцінки з матриці вподобань **R**, яка має розмірність $\mathbf{17.770} \times \mathbf{480.189}$.

В свою чергу $\boldsymbol{P}$ — матриця скритих ознак користувачів розміру $\boldsymbol{d} \times \mathbf{17.770}$,

$\boldsymbol{Q}$—скриті ознаки фільмів, розмір $\boldsymbol{d} \times \mathbf{480.189}$, де $\boldsymbol{d}$ — величина низькорангового наближення і $\boldsymbol{d} \leq \boldsymbol{min}(\mathbf{n}, \boldsymbol{m})$.

Будемо варіювати величину $\boldsymbol{d}$.

В прикладі розглянемо $\boldsymbol{d} = \mathbf{2}$.

Одразу зауважимо, вектори латентних ознак $\boldsymbol{p_u}$ та $\boldsymbol{q_i}$ є строками матриць латентних вподобань $\boldsymbol{P}$ та $\boldsymbol{Q}$ відповідно.

Результати навчання параметрів:

```
Read 17770 users and 489189 films
Iteration 1:     RMSE=4.21306489635098
Iteration 2:     RMSE=4.18968548673874
Iteration 3:     RMSE=4.14335325525564
...
Iteration 1098492:     RMSE=0.9742745242478784
Iteration 1098493:     RMSE=0.9742694274828472
User features
  user 1:       0.9542  0.7742
  user 2:       1.3245  0.9823
  user 3:       0.8096  0.8555
  user 4:       0.2339  0.7811
  user 5:       0.9095  0.6347
  ---------------------------
Item features:
  item 1:       0.8368  4.2511
  item 2:       4.9101  0.6321
  item 3:       1.3159  2.4454
  item 4:       4.3130  0.8635
  item 5:       1.7408  1.9205
  ---------------------------
```

Шукана оцінка буде рахуватися так: $\mathbf{0.8096} * \mathbf{0.8368} + \mathbf{0.8555} * \mathbf{4.2511} = \mathbf{4.31428933}$

	1	2	3	4	5	6	7
1	5	?	2	?	5	?	4
2	5	2	?	?	3	?	?
3	4.31429	?	1	3	?	2	1
4	4	4	?	?	?	5	?

Алгоритм SVD++

b_u, b_i та μ — базові предиктори.

Нагадаємо, що ці параметри взаємопов'язані, тому розраховувати їх будемо разом.

μ — це число, b_u — вектор довжиною 17.770,

b_i— вектор довжиною 480.189.

Результати навчання параметрів:

```
Read 17770 users and 489189 films
Iteration 1:     RMSE=3.92845291697284
Iteration 2:     RMSE=3.28481736445124
Iteration 3:     RMSE=3.14159530044051
...
Iteration 20390208:     RMSE=0.9280161117835789
Iteration 20390209:     RMSE=0.9280091123741253
       mu:      2.54559533638261
User base:      0.7139  0.1626  0.7271  1.9097  -0.9677 ---
Item base:      0.8450  0.6593  0.2731  0.7328  0.0354 ---
User features
  user 1:       -0.9509 0.2792
  user 2:       1.0220  1.2826
  user 3:       -0.5087 -0.8326
  user 4:       0.1031  -0.4814
  user 5:       0.6095  0.0557
  ----------------------------
Item features:
  item 1:       -0.8368 0.2511
  item 2:       1.1101  0.4120
  item 3:       -0.4159 -0.4073
  item 4:       -0.3130 -0.9115
  item 5:       0.6408  1.2205
  ----------------------------
```

Шукана оцінка рахується так: $\mathbf{2.5456} + \mathbf{0.7271} + \mathbf{0.8450} + (\mathbf{-0.5087}) * (\mathbf{-0.8368}) + (\mathbf{-0.8326}) * \mathbf{0.2511} = \mathbf{4.3343143}$

	1	2	3	4	5	6	7
1	5	?	2	?	5	?	4
2	5	2	?	?	3	?	?
3	4.334314	?	1	3	?	2	1

Порівняння результатів

Отже, оцінка, яку ставить користувач №3 фільму №1, відрізняється в залежності від алгоритму.

- User-based : 3.9 RMSE = 1.203469
- Item-based : 4.4 RMSE = 1.212423
- Model-based SVD : 4.31428933 RMSE = 0.974269
- Model-based SVD++ : 4.3343143 RMSE = 0.928009

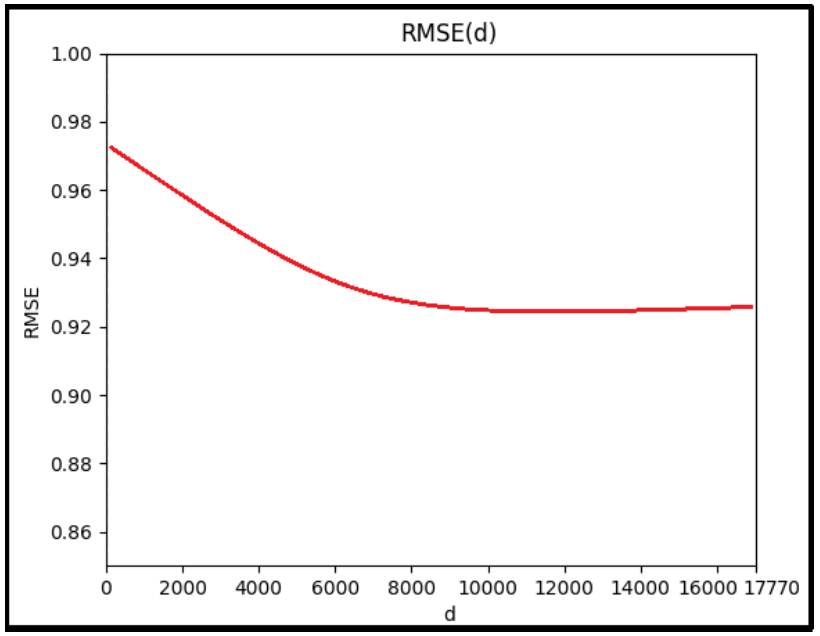
Як видно з результатів, алгоритм **SVD++** має найменший рейтинг **RMSE**.

Слід зазначити, якщо кількість користувачів і об’єктів занадто велика, кожна тисячна рейтингу **RMSE** має значення.

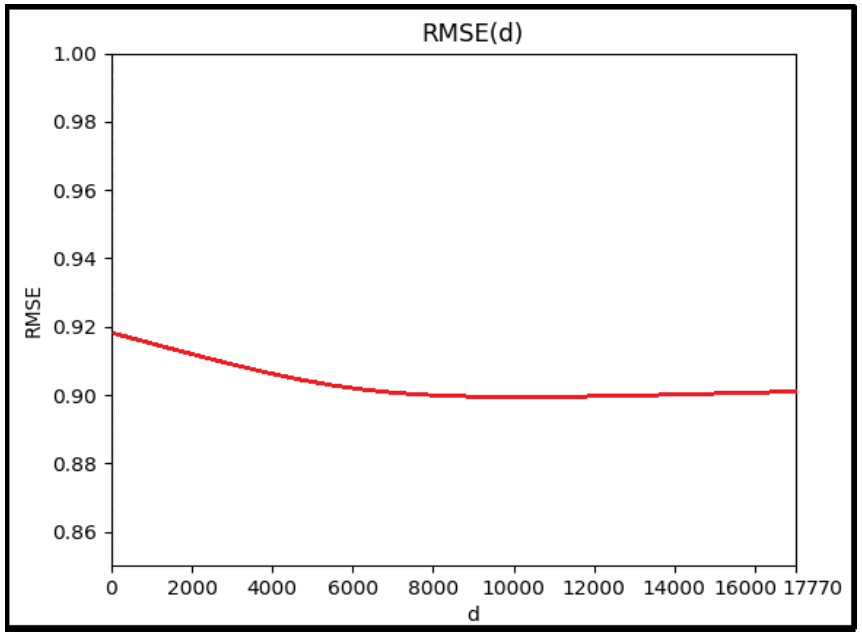
Наведемо залежності похибки **RMSE** від параметрів моделей.

Залежність RMSE від величини низькорангового наближення

На графіках наведена залежність значення **RMSE** від $\boldsymbol{d}$ — величини низькорангового наближення $\boldsymbol{d} \leq \mathbf{min}(\mathbf{n}, \boldsymbol{m})$, де $\boldsymbol{d} \leq \mathbf{17.770}$.



Залежність **RMSE** від $\boldsymbol{d}$ для алгоритму **SVD**

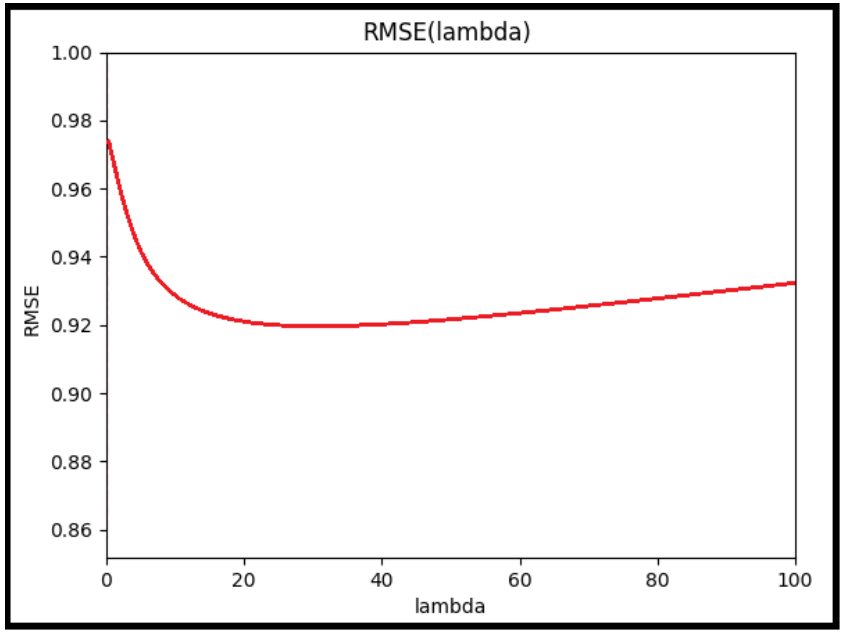


Залежність **RMSE** від $\boldsymbol{d}$ для алгоритму **SVD++**

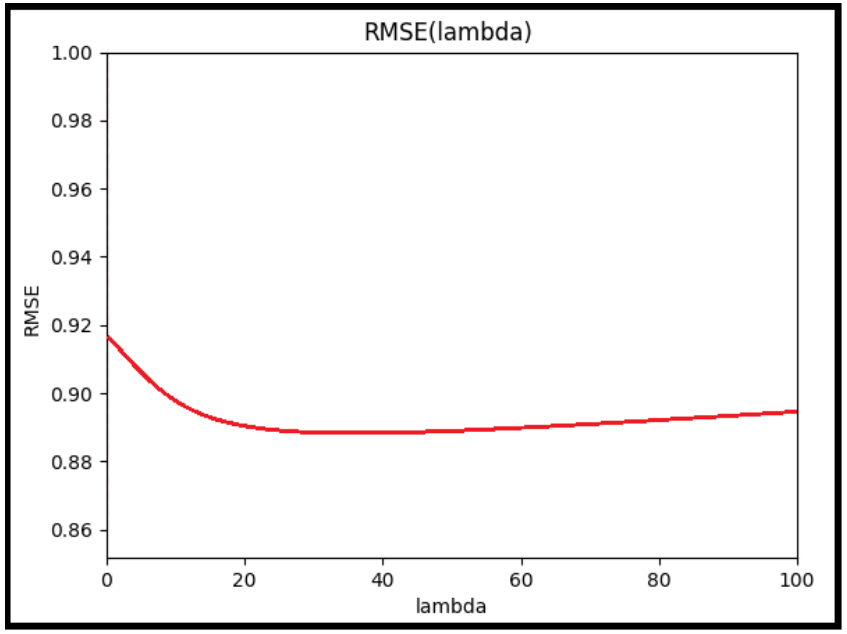
Така поведінка **RMSE** пояснюється тим, що найкращим низькоранговим наближенням, з точки зору середньоквадратичного відхилення, є деяке конкретне значення $\boldsymbol{d}$, після якого похибка майже не змінюється. Якщо буде обрано потрібне $\boldsymbol{d}$, час навчання параметрів зменшиться.

Залежність RMSE від параметра регуляризації λ

На графіках наведена залежність значення **RMSE** від параметра регуляризації λ.



Залежність **RMSE** від λ для алгоритму **SVD**

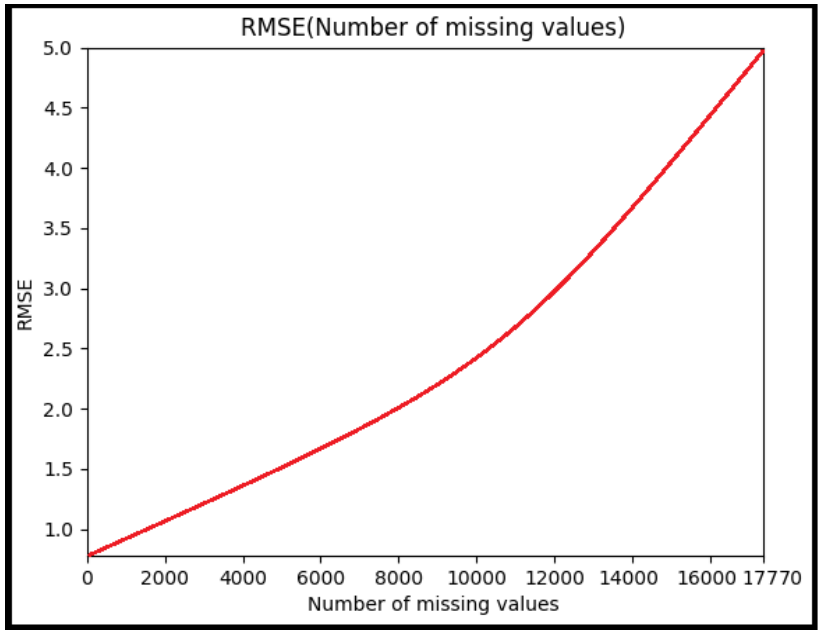


Залежність **RMSE** від λ для алгоритму **SVD++**

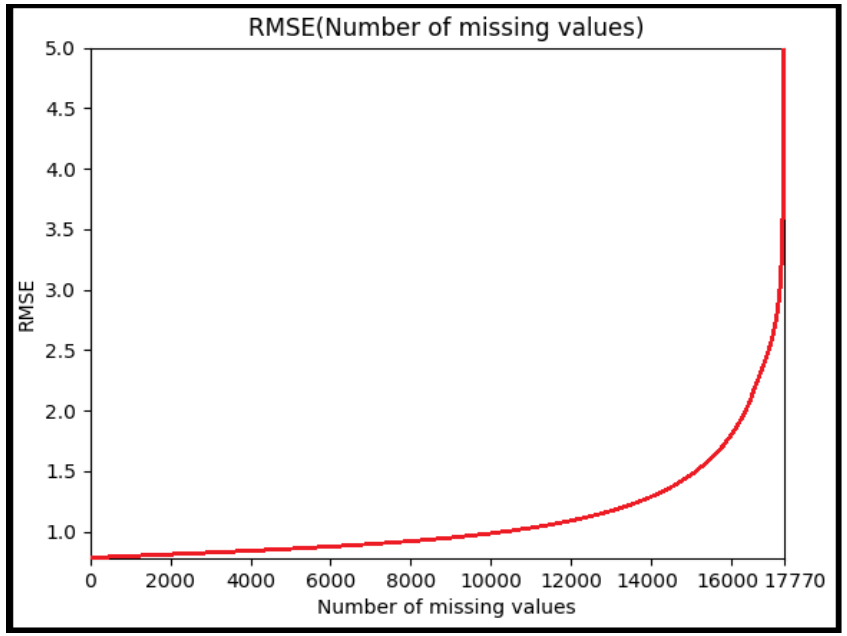
Очевидно, що при значенні $\lambda \approx 30$, похибка **RMSE** в нашому досліді має мінімальне значення. За графіком видно, що необхідно відповідально підбирати значення параметра регуляризації.

Залежність RMSE від ступеня розрідженості матриці

На графіках наведена залежність значення **RMSE** від ступеня розрідженості матриці вподобань для **memory-based** та **model-based** алгоритмів:



Залежність **RMSE** від кількості відсутніх значень в матриці вподобань за **memory-based** алгоритмом



Залежність **RMSE** від кількості відсутніх значень в матриці вподобань за **model-based** алгоритмом

Такий результат є передбачуваним, адже алгоритми SVD та SVD++ покликані боротися з тим, щоб відсутність більшості оцінок не так суттєво впливала на якість передбачення вподобань користувачів.

Висновки

- Проаналізовано базові поняття та підходи до побудови рекомендаційних систем
- Розкрита проблема колаборативної фільтрації
- Наведено дослідні дані, описано і програмно реалізовано алгоритми навчання memory-based та model-based моделей
- Продемонстровано процес підрахунку оцінки вподобання конкретним користувачем конкретного фільму за всіма колаборативними алгоритмами
- Проаналізовано залежності похибки передбачення від алгоритму та його параметрів

Перспективи подальших досліджень

- Впровадження додаткових методів групування користувачів та об’єктів за їх описами — кластеризація
- Пошук способів імплементації предикторів в моделі машинного навчання, що відповідають за час та дату виставлення оцінок користувачами.