# Metody Odkrywania Wiedzy

Projekt 2018 Z

## Predykcja ocen książek

Igor Markiewicz
Aleksander Droszcz

Prowadzący – dr inż. Paweł Cichosz

# Spis treści

# 1 Wstęp

## 1.1 Opis tematu

Celem projektu jest predykcja (i jej ocena) treści dopasowywanych użytkownikom przy użyciu różnych algorytmów filtracji kolaboratywnej.

## 1.2 Opis zbioru danych

Jako zbiór danych został wybrany zestaw zawierający oceny książek wprowadzone przez różnych użytkowników [1] składający się z trzech plików:

- BX-Book-Ratings.csv – 493813 rekordy bez wartości pustych o następujących atrybutach:
  - User.ID – ID użytkownika (liczba naturalna) ,
  - ISBN – ID książki (liczba naturalna),
  - Book.Rating – ocena pośrednia (0) lub bezpośrednia książki (całkowita liczba nieujemna z przedziału $[1, 10]$). Projekt skupia się na ocenach bezpośrednich, dlatego po przefiltrowaniu pozostaje 176019 rekordów z ocenionymi książkami.

## 1.3 Opisy badanych algorytmów

### 1.3.1 User–based Collaborative Filtering

Niech $\boldsymbol{r}_x$ oznacza wektor ocen użytkownika $x$, analogicznie $\boldsymbol{r}_y$, zaś $I_{xy}$ zbiór produktów, które zostały ocenione zarówno przez użytkownika $x$, jaki i przez użytkownika $y$. Dwie najpopularniejsze miary podobieństwa to:

- współczynnik korelacji liniowej Pearsona:

$$sim_{Pearson}(x, y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r}_x)(r_{y,i} - \overline{r}_y)}{(|I_{xy}| - 1)\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \overline{r}_x)^2}\sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \overline{r}_y)^2}}, \tag{1}$$

- podobieństwo cosinusowe

$$sim_{Cosine}(x, y) = \frac{\boldsymbol{r}_x \cdot \boldsymbol{r}_y}{\|\boldsymbol{r}_x\|_2 \|\boldsymbol{r}_y\|_2} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2}\sqrt{\sum_{i \in I_y} r_{y,i}^2}}. \tag{2}$$

Po obliczeniu podobieństw, w następnym kroku dla danego użytkownika $a$ wybieramy zbiór $\mathcal{N}(a)$ innych użytkowników dla których współczynniki podobieństwa z nim były największe. Zazwyczaj dokonujemy tego posługując się odpowiednim progiem decyzyjnym lub metodą najbliższych sąsiadów. W dalszej części realizacji projektu zostanie podjęta decyzja która z tych metod jest skuteczniejsza.

W ostatnim kroku agregujemy dostępne oceny od wybranych użytkowników dla pewnego przedmiotu $j$ (wybierając tych którzy go ocenili) i przypisujemy użytkownikowi $a$ ocenę $\hat{r}_{a,j}$:

$$\hat{r}_{a,j} = \frac{1}{\sum_{i \in \mathcal{N}(a)} s_{a,i}} \sum_{i \in \mathcal{N}(a)} s_{a,i} r_{i,j}, \tag{3}$$

gdzie $s_{a,i}$ oznacz współczynnik podobieństwa między użytkownikiem $a$ oraz $i$.
Często wykorzystywaną metodą jest normalizacja ocen użytkowników mająca zapobiegać obciążeniu związanemu ze stosowaniem przez niektóre osoby ciągłego zaniżania lub zawyżania oceny w porównaniu z innymi użytkownikami. Przykładem takiej normalizacji dla pewnego przedmiotu $i$ jest:

$$r_{a,i} := r_{a,i} - \overline{r}_a, \tag{4}$$

gdzie $\overline{r}_a$ oznacza średnią arytmetyczną wszystkchi ocen dla użytkownika $a$.

### 1.3.2 Item–based Collaborative Filtering

Jest to algorytm podobny do User–based Collaborative Filtering. Różnica polega na tym, że zamiast wyszukiwania dla danego użytkownika innych użytkowników podobnie oceniających przedmioty w celu agregacji ich wyników, wyszukujemy dla danego użytkownika i danego przedmiotu inne przedmioty podobne do niego na bazie ocen użytkowników a następnie agregujemy wyniki na podstawie ocen tych przedmiotów zgłoszonych przez użytkownika docelowego. W szczególności w mocy pozostają zależności (1) – (2) przy czym należy zamienić rolę użytkowników i przedmiotów (obowiązuje także wzór (4)). Wzór na zagregowaną ocenę wyznaczamy jako:

$$\hat{r}_{a,i} = \frac{1}{\sum_{j \in \mathcal{S}(i) \cap \{l : r_{a,l} \neq ?\}} s_{i,j}} \sum_{j \in \mathcal{S}(i) \cap \{l ; r_{a,l} \neq ?\}} s_{i,j} r_{a,j}, \tag{5}$$

gdzie

- $\mathcal{S}(i)$ – wyznaczony zbiór najbliższych sąsiadów przedmiotu $i$,
- $r_{a,l} \neq ?$ – istniejąca ocena użytkownika $a$ przedmiotu $l$,
- $s_{i,j}$ – podobieństwo przedmiotu $i$ i $j$,
- $r_{a,j}$ – ocena przez użytkownika $a$ przedmiotu $j$.

### 1.3.3 Funk SVD

Mając niepełną macierz ocen $\overline{R} \in \mathbb{R}^{users \times items}$ bedącą obserwowanym fragmentem prawdziwej macierzy$R$, staramy się dokonać takiej dekompozycji na dwie składowe $M \in \mathbb{R}^{users \times latent\_factors}$ oraz $U \in \mathbb{R}^{items \times latent\_factors}$ by $\hat{R} \approx R$, gdzie $\hat{R} = MU^T$, przy czym macierz $\hat{R}$ posiada już wszystkie elementy. Założono jako rozkład $MU^T$ zamiast $M\Sigma U^T$ ze względu na to, że macierz $\Sigma$ jest diagonalna, więc działa jako czynnik skalujący. Jako $latent\_factors$ rozumiemy liczbę składowych "utajonych"modelu którą możemy dobierać w celu optymalizacji wyników (zwiększając

liczbę $latent\_factors$ zwiększamy potencjalna siłę reprezentacji, ale jednocześnie za duża wartość tego parametru może spowodować nadmierne dopasowanie). Ze względu na to, iż macierz $\overline{R}$ jest przeważnie rzadka, bezpośrednie metody faktoryzacji SVD oparte o wektory własne bardzo często zawodzą. Innym rozwiązaniem tego problemu jest znalezienie takich wektorów $p_u$ i $q_i$ (gdzie $p_u$ jest wierszem w macierzy $M$ a $q_i$ kolumną w macierzy $U^T$), że:

- $r_{u,i} = p_u \cdot q_i$ dla każdego $u$ i $i$,
- wszystkie wektory $p_u$ są wzajemnie ortogonalne, podobnie jak wektory $q_i$.

Ze względów praktycznych często opuszczane jest wymaganie ortogonalności wektorów, dlatego jako funkcję błędu można przyjąć:

$$f(p_*, q_*) = \sum_{(u,i)\in k} (r_{u,i} - p_u \cdot q_i)^2 = \sum_{(u,i)\in k} f_{u,i}(p_u, q_i), \tag{6}$$

gdzie $k$ to zbiór wszytkich możliwych pozycji $r$, zaś $p_*, q_*$ to zbiór wszystkich możliwych wektorów $p$ i $q$. Wtedy zadanie sprowadza się do minimalizacji funkcji $f$ co można osiągnąć stosując metodę gradientową (ignorując jednocześnie przykłady z pustymi $r_{u,i}$) i otrzymując w efekcie (po uwzględnieniu dodatkowych czynników regularyzacyjnych karzących model za nadmierne dopasowanie):

- aktualną predykcję:
$$\hat{r}_{u,i} = p_u \cdot q_i, \tag{7}$$
- aktualny błąd wykorzystywany w metodzie gradientowej:
$$e_{u,i} = r_{u,i} - \hat{r}_{u,i}, \tag{8}$$
- aktualiację współczynników:
$$q_i := q_i + \alpha(e_{u,i} \cdot p_u - \beta \cdot q_i), \tag{9}$$
$$p_u := p_u + \lambda(e_{u,i} \cdot q_i - \gamma \cdot p_u), \tag{10}$$
  - $\alpha$ – współczynnik szybkości uczenia (ang. *learning rate),*
  - $\beta$ – *waga czynnika regularyzacyjnego (ang. regularizing coefficient),*
  - $\lambda$, $\gamma$ – *dodatkowe współczynniki.*

### 1.3.4 Popular

Ten algorytm zostanie przetestowany jedynie w wypadku, gdy zostanie użyta binaryzacja problemu (opisana w podpunkcie 1.4) zamiast predykcja dokładnych ocen. Jest to prosty algorytm heurystyczny, który tworzy listę najpopularniejszych przedmiotów. Popularność danego przedmiotu jest mierzona w liczbie użytkowników, którzy ocenili ten przedmiot. Przy wykonaniu listy wszystkich przedmiotów można taką listę podzielić na dwie części: górną i dolną. Część górną zawierającą $N$ elementów można potraktować jako przedmioty godne polecenia (logiczne 1), natomiast część dolną jako przedmioty niegodne polecenia (logiczne 0). Badanie skuteczności takiego algorytmu ma na celu umożliwienie porównania prostych algorytmów heurystycznych względem bardziej wyrafinowanych metod przedstawionych w poprzednich podpunktach. W przypadku tego algorytmu każdy użytkownik będzie otrzymywał takie same predykcje binarne danego przedmiotu.

### 1.3.5 Średnia arytmetyczna przedmiotu

Ostatnim przedstawionym algorytmem jest również prosty algorytm heurystyczny. Przewidywana ocena danego przedmiotu $j$ dla użytkownika $a$ jest wyznaczany na podstawie średniej arytmetycznej dotychczasowych ocen tego przedmiotu:

$$\hat{r}_{a,j} = \frac{1}{|A_j|} \sum_{i \in A_j} r_{i,j} \tag{11}$$

gdzie $A_j$ oznacza zbiór użytkowników, którzy ocenili przedmiot $j$. W przypadku tego algorytmu również każdy użytkownik będzie uzyskiwał takie same predykcje oceny danego przedmiotu.

## 1.4 Plan badań

Wstępnie planuje się podział zbioru danych na część testową – 20 % oraz część trenującą – 80 %, przy czym część trenująca będzie podlegać 10 krotnej walidacji krzyżowej, zaś testowa zostanie użyta na samym końcu po selekcji modeli. Jako potencjalne testy rozważa się:

- rodzaj normalizacji danych (*row center*, *Z – Score*)
- dla UBCF i IBCF rodzaj współczynnika podobieństwa (Pearsona, cosinusowy) oraz liczbę najbliższych sąsiadów
- dla modelu FunkSVD – liczbę latent_factors oraz wartości $\alpha$ oraz $\beta$

Jako metryki oceny modeli planowane jest użycie (gdzie $f(x)$ to wartość rzeczywista, a $h(x)$ odpowiedź modelu):

$$RMSE = \sqrt{\frac{\sum_{x \in S}(f(x) - h(x))^2}{|S|}}, \tag{12}$$

$$MAE = \frac{1}{|S|} \sum_{x \in S} |f(x) - h(x)|. \tag{13}$$

Do rozważenia pozostają następujące kwestie:

- sposób wstępnej analizy danych np: podstawowe statystyki, wykresy, histogramy,
- sposób oceny modeli – prosty, czyli bezpośrednie porównanie wartości metryk czy bardziej złożony np: testy statystyczne,
- binaryzacja problemu w celu wprowadzenia macierzy pomyłek i wyliczenia metryk charakterystycznych dla klasyfikacji.

# 2 Bibliografia

[1] https://grouplens.org/datasets/book-crossing/

[2] https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf

[3] https://cran.r-project.org/web/packages/recommenderlab/recommenderlab.pdf

[4] https://en.wikipedia.org/wiki/Collaborative_filtering

[5] https://cran.r-project.org/web/packages/rrecsys/rrecsys.pdf

[6] https://en.wikipedia.org/wiki/Matrix_factorization_(recommender_systems)#Funk_SVD

[7] http://nicolas-hug.com/blog/matrix_facto_1

[8] http://nicolas-hug.com/blog/matrix_facto_2

[9] http://nicolas-hug.com/blog/matrix_facto_3

[10] https://www.slideshare.net/DKALab/collaborativefilteringfactorization