Projekt MOW

Dokumentacja końcowa

Prowadzący: dr inż. Paweł Cichosz

Autorzy:

Mikołaj Chojnacki <M.Chojnacki@stud.elka.pw.edu.pl>

Maciej Gańko <M.Ganko@stud.elka.pw.edu.pl>

Temat: **Kolaboratywna filtracja**

**Szczegółowa interpretacja tematu projektu**

Celem projektu było wykonanie kolaboratywnej filtracji (*collaborative filtering* - CF), umożliwiającej zarekomendowanie (predykcję) wybranym użytkownikom pewnej liczby produktów - w tym przypadku będą to filmy.

Ideą stojącą za kolaboratywnym filtrowaniem jest następująca hipoteza:

*Jeżeli osoba A lubi przedmioty X i Y, a osoba B lubi przedmiot X, to prawdopodobieństwo, że osoba B również będzie lubić przedmiot Y jest większe niż w przypadku losowo wybranej osoby.*

W projekcie wykorzystany został zbiór danych MovieLens [1].

Pełny zbiór składa się z 20 milionów ocen 27 tysięcy filmów, dokonanych przez 138 tysięcy użytkowników. Oceny mają postać dyskretną, od 1 do 5 z dopuszczeniem ocen połówkowych.

Projekt wykonany został w środowisku R przy pomocy biblioteki recommenderlab [2, 3].

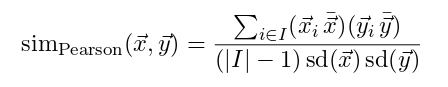
**Opis algorytmów**

W projekcie zespół wykonał dwa rodzaje kolaboratywnej filtracji:

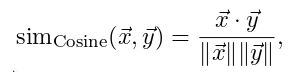
* *Item-based Collaborative Filtering* - ze względu na podobieństwo ocenianych przedmiotów (filmów)
* *User-based Collaborative FIltering* - ze względu na podobieństwo preferencji oceniających użytkowników

Wykorzystane zostały następujące miary podobieństwa:

* *Pearson similarity*:



* *Cosine similarity*:



**Implementacja**

Kod źródłowy składa się w następujących plików:

**collaborativeFiltering.R** - zawiera główną funkcję przeprowadzającą kolaboratywną filtrację:

collaborativeFiltering <- function (

ratings, zbiór danych o ocenach w postaci data.frame

trainDatasetSize, część zbioru danych brana do nauki (standardowo 0.9)

recommendationMethod, UBCF/IBCF

similarityMethod, Cosine/Pearson

trainMethod, split/bootstrap/cross-validation

kCount, liczba próbek bootstrapowych lub krotność walidacji krzyżowej

givenItems, ile ocen każdego użytkownika wykorzystać do nauki

nearestNeighbours, ilu najbliższych sąsiadów użytkownika wykorzystać do predykcji

(tylko UBCF)

goodRating, minimalna ocena uznawana za pozytywną (dla MovieLense 4)

predictType topNList (N najbardziej polecanych użytkownikowi produktów)

lub ratings (predykcja wszystkich brakujących ocen w zbiorze)

)

Funkcja:

- buduje macierz z podanego data.frame’a

- za pomocą funkcji evaluationScheme() dzieli ją na zbiór trenujący i testowy

- tworzy i trenuje rekomender

- przeprowadza predykcję dla zbioru testowego

- wylicza dokładność predykcji za pomocą funkcji calcPredictionAccuracy()

- dopisuje statystyki i czas wykonania jako wiersz do pliku (w zależnosci od parametru predictType) tests/topNList.csv lub tests/ratings.csv

- zwraca stworzony rekomender oraz statystyki zebrane w czasie testu

**utils.R** - zawiera funkcję:

preprocess <- function(dataset, maxMovieId, maxUserId, minRatings) - filtruje zbiór po wartości movieId, userId lub do użytkowników posiadających minimum podaną liczbę ocen

Oraz funkcje pomocnicze do wczytania i zapisu danych read(), write() i writeLine()

**tests.R** - zawiera listę wywołań głównej funkcji dla różnych metod kolaboratywnej filtracji i różnych wartości strojonych parametrów

**example.R** - zawiera pojedyncze wywołanie ww. funkcji dla przykładowych parametrów

**Testy**

Wyniki testów zamieszczono w plikach tests/topNList.csv oraz tests/ratings.csv.

Do testów wybrano początkowo użytkowników z min. 10 ocenami i dodatkowo w celach wydajnościowych z tego zbioru wybrano próbkę 10000 użytkowników i 5000 filmów. Wyniki były jednak niezadowalające, dlatego zdecydowano się na podniesienie progu do 100 ocen, motywując to tym, że użytkownicy regularnie oceniający filmy są tematyką bardziej zainteresowani i ich oceny będą bardziej miarodajne. Uzyskano w ten sposób próbę 1751 użytkowników z 455608 ocenami.

W celu pomiaru jakości predykcji oceny użytkowników ze zbioru testowego zostały podzielone na dwie grupy: znane i nieznane. Następnie dokonano predykcji tak, jak gdyby użytkownicy ci mieli wyłącznie oceny znane i porównano z ze zbiorem ocen nieznanych za pomocą funkcji calcPredictionAccuracy().

W przypadku listy top N rekomendowanych produktów zwracanymi wartościami są:

TP - liczba produktów wypredykowanych i istniejących w zbiorze nieznanym

FP - liczba produktów wypredykowanych, choć nieistniejących w zbiorze nieznanym

FN - liczba produktów istniejących w zbiorze nieznanym, choć niewypredykowanych

TN - liczba produktów nieistniejących w zbiorze nieznanym i niewypredykowanych

Oraz policzone na ich podstawie precision, recall, TP-rate i FP-rate.

Drugi sposób predykcji to predykcja wszystkich brakujących par użytkownik-produkt w zbiorze ocen znanych i pomiar błędu pomiędzy wynikami a ocenami ze zbioru ocen nieznanych. Tutaj podstawową miarą jakości jest błąd średniokwadratowy (MSE), a oprócz niej w wynikach testów zamieszczono także pierwiastek z błędu średniokwadratowego (RMSE) oraz średni błąd bezwzględny (MAE).

Pierwszą fazą testów było strojenie parametrów. W tym celu przygotowano listę parametrów wejściowych, większość z nich powstała poprzez zmodyfikowanie jednego parametru w zestawie bazowym. Strojonymi parametrami były:

* Metoda filtracji (user-based/item-based)
* Miara podobieństwa (kosinusowe/Pearsona)
* Metoda wyboru zbioru trenującego (zwykły podział/próby bootstrapowe/k-krotna walidacja krzyżowa)
* Liczba ocen brana do oceny (5 lub 10)
* Liczba sąsiadów brana pod uwagę przy predykcji (5 lub 10)

Należy jednak pamiętać, że wyniki mają w sobie pewną dozę losowości z uwagi na niedeterministyczny podział na zbiór trenujący i testowy w każdym wywołaniu. Z tego też powodu niektóre testy powtarzaliśmy 2 lub 3-krotnie dla tego samego zbioru parametrów.

Ta faza testów wykazała m.in. że:

* Metoda UBCF daje wyraźnie lepsze wyniki niż IBCF, a przy tym ma wyraźnie krótszy czas wykonania
* Zmiana metody podobieństwa przynosiła na ogół pomijalne zmiany jakości predykcji z niewielkim wskazaniem na podobieństwo cosinusowe
* Zastosowanie 5 prób bootstrapowych i 5-krotnej walidacji krzyżowej powodowało na ogół niewielki wzrost jakości (dla podobieństwa Pearsona nawet niewielki spadek)
* Zwiększanie liczby branych do nauki produktów powodowało niewielki wzrost jakości
* Zwiększanie liczby najbliższych sąsiadów powodowało znaczny wzrost jakości

Ostatecznie wybrano parametry:

* UBCF
* Podobieństwo kosinusowe
* 5-krotna walidacja krzyżowa
* 10 ocen/użytkownik branych do nauki
* 100 sąsiadów

**Wnioski**

RecommenderLab pozwala na stworzenie systemu rekomendującego przy minimalnej ilości własnego kodu. Dostarcza też narzędzia do testów i oceny jakości stworzonych modeli, w tym do łatwego ich porównywania. Z drugiej strony liczba parametrów jest na tyle duża, że na ich strojenie należy poświęcić sporo wysiłku.

Biblioteka jest nie najgorszej udokumentowana, choć dla niektórych funkcji znajdowaliśmy tylko przykłady użycia zamiast listy wszystkich dostępnych parametrów. Co więce, niektóre przekazywało się do funkcji nie bezpośrednio, ale w postaci osobnej listy, co dodatkowo utrudniało ich użycie.

Problemy, na które się natknęliśmy, były głównie wydajnościowe. Uruchomiony dla dużych zbiorów rekomender alokuje w pamięci struktury o wielkości liczonej w gigabajtach, przez co nauka lub predykcja potrafiła zakończyć się niepowodzeniem na maszynie z 8 GB pamięci. Zaskoczył też nas długi czas działania niektórych metod, szczególnie IBCF. Nie jesteśmy pewni bez zagłębiania się w implementację, na ile funkcje biblioteki działają wielowątkowo, ale czas działania był naszym zdaniem miejscami zbyt długi w stosunku do wielkości zbiorów.

Przedstawiony system do rekomendacji można by ulepszyć, wykorzystując dodatkowe informacje o produktach, np. dla zbiór MovieLense posiada dodatkowy plik z przyporządkowaniem filmów do gatunków.

Zaczęliśmy również dla porównania testy rekomendera na drugim zbiorze - BookCrossing[4], lecz nie starczyło nam czasu na ich zakończenie.

**Literatura i zewnętrzne linki:**

1. <https://grouplens.org/datasets/movielens/>
2. Michael Hahsler (2017). recommenderlab: Lab for Developing and Testing Recommender Algorithms. R package version 0.2-2. <http://lyle.smu.edu/IDA/recommenderlab/>
3. <https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf>
4. <https://grouplens.org/datasets/book-crossing/>