Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki

Wiktor Zuba
Nr albumu: 320501

Metody ulepszania systemów rekomendacyjnych

Praca magisterska
na kierunku MATEMATYKA

Praca wykonana pod kierunkiem
prof. Hung Son Nguyen
Instytut Informatyki

??? 2016

Oświadczenie kierującego pracą

Potwierdzam, że niniejsza praca została przygotowana pod moim kierunkiem i kwalifikuje się do przedstawienia jej w postępowaniu o nadanie tytułu zawodowego.

Data Podpis kierującego pracą

Oświadczenie autora (autorów) pracy

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w wyższej uczelni.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Data Podpis autora (autorów) pracy

Streszczenie

??????????????????????????????????????

Słowa kluczowe

??????????????????????

Dziedzina pracy (kody wg programu Socrates-Erasmus)

11.0 Matematyka, Informatyka:????????????????????????
11.2 Statystyka??????????????????
11.4 Sztuczna inteligencja????????????

Klasyfikacja tematyczna

62-XX Statistics??????????????????????????????
68-XX Computer science??????????????????????????????????

Tytuł pracy w języku angielskim

Methods of recommendation systems improving

Spis treści

Wprowadzenie

Rozdział 1

Sformułowanie problemu

W każdym modelu danych wykorzystywanym do wyliczania możliwych ocen przedmiotów lub sporządzania list rekomendacji występują użytkownicy i przedmioty, oraz pewne niepełne informacje na temat ich powiązań ze sobą. Aby móc przewidzieć czy dany nie oceniony jeszcze przedmiot zainteresuje konkretnego użytkownika trzeba dla obu określić preferencje na podstawie załączonych opisów, znanych interakcji tego użytkownika z innymi przedmiotami i przedmiotu z innymi użytkownikami, lub też obu informacji na raz.

1.1. Podstawowe definicje

Definicja 1.1.1. Użytkownik – osoba korzystająca z serwisu, dla której staramy się stworzyć rekomendacje lub też, która tylko dostarcza informacji użytecznych przy ich sporządzaniu dla innych użytkowników.

Definicja 1.1.2. Przedmiot – rzecz która jest używana (czasem też oceniana) przez użytkowników.

Przedmiot musi być reużywalny (np. filmy do oglądnięcia, książki do przeczytania, strony internetowe do odwiedzenia), lub występujący w wielu identycznych egzemplarzach (przedmioty w sklepach), aby użycie przez innego użytkownika nie wyczerpało jego zasobu (wtedy rekomendacja nie miała by sensu, skoro przedmiot nie istnieje) i dało miarodajne informacje na jego temat. Przedmiot jest rzeczą, której ocenę przez użytkownika chcemy przewidzieć, lub którą chcemy mu zarekomendować.

Definicja 1.1.3. Użycie przedmiotu – zarejestrowana informacja o interakcji użytkownika z przedmiotem, dostępna w danych wykorzystywanych do tworzenia rekomendacji – niekoniecznie prawdziwe używanie przedmiotu (kupienie produktu nie oznacza jego używania a wypożyczenie książki jej przeczytania).

Definicja 1.1.4. Informacja explicite – wyraźne informacje dostarczone przez użytkownika odnośnie przedmiotu – głównie ocena, recenzja lub przypisanie tagów.

Definicja 1.1.5. Informacja implicite – bezwarunkowe informacje o użyciu przedmiotu przez użytkownika (kupienie towaru, obejrzenie filmu, wypożyczenie książki, odwiedzenie strony).

Informacje bezwarunkowe, których główną zaletą jest łatwość ich zbierania (dzieje się to

automatycznie, bez dodatkowych akcji ze strony użytkownika, a czasem i bez jego wiedzy), są niestety bardzo narażone na przekłamania odnośnie preferencji użytkownika. Po pierwsze zarejestrowanie uzycia przez system wcale nie musi oznaczać faktycznego użycia (przypadkowe kliknięcie linku, pomyłka we wpisywaniu nazwy, zrezygnowanie po przeczytaniu opisu), a po drugie z użycia nie musi wynikać pozytywny odbiór. W szczególności użycie przedmiotu wynikające z nietrafionych rekomendacji może pogłębiać zawodność systemu. Pomimo licznych wad informacje implicite często są jedynymi posiadanymi, a nawet przy dostępnych danych explicite ze względu na znacznie większą ilość znajdują istotne zastosowanie w systemach rekomendacyjnych. Dane o wielokrotnym użyciu przedmiotu są często spłaszczone do binarnych (przedmiot użyty/nie użyty), co pozwala na zmniejsznie rozmiaru pamięci potrzebnej do ich przechowywania, oraz użycie niektórych metod rekomendacji.

Definicja 1.1.6. Lista rekomendacji – (zazwyczaj uporządkowany) podzbiór przedmiotów nie użytych dotychczas przez użytkownika, które powinny się mu spodobać (być najwyżej ocenione, chętnie użyte).

1.2. Przeznaczenie systemów

Istnieją dwa główne cele postawione przed systemami rekomendacyjnymi:

- przewidzenie niewprowadzonych ocen przedmiotów
- zbudowanie listy rekomendacyjnej konkretnej długości:
  - lista przedmiotów, które powinny być użyte (jak w przypadku informacji implicite)
  - lista przedmiotów, które powinny zostać najwyżej ocenione (niekoniecznie często uzywane)
  - lista przedmiotów, które powinny być użyte i jednocześnie pozytywnie ocenione (rozwiązanie pośrednie)

Oczywiście jeżeli system przewidzi oceny wszystkich nieocenionych przedmiotów, łatwo można je uszeregować malejąco i obciąć listę w odpowiednim miejscu otrzymując listę rekomendacji (typu najwyżej ocenione) pożądanej długości.
Konwersja w drugą stronę jest znacznie trudniejsza, a bez dodatkowych informacji praktycznie niemożliwa.
Posiadanie przewidzianych ocen przedmiotów może być bardzo przydatne – serwis proponujący przedmioty może przedstawiać użytkownikowi rekomendacje mocniejsze i słabsze w inny sposób, lub też obciąć listę rekomendacji tylko do przedmiotów o przewidzianej ocenie powyżej pewnego poziomu. Dodatkową motywacją do wyliczenia przewidywanych ocen jest możliwość podania tych informacji użytkownikowi wraz z rekomendacjami dla mocniejszego zainteresowania użytkownika przedmiotem, oraz zwiększenia satysfakcji z systemu.
Niestety w niektórych przypadkach na przykład przy posiadaniu jedynie informacji implicite określenie ocen przedmiotów jest niemożliwe (chyba że wprowadzimy sztuczne oceny na podstawie miejsc w liście rankingowej). Dodatkowo algorytm generowania rekomendacji którego wynikiem jest tylko lista rekomendacji (bez ocen) może posiadać pożądane przez nas zalety jak szybkość lub oszczędność pamięci, są więc one wykorzystywane w sytuacjach w których posiadanie przewidzianych ewaluacji nie ma dodatkowych zastosowań.

1.3. Problemy stojące przed systemami rekomendacyjnymi

Do czysto teoretycznych badań wykorzystuje się stabilne, nie zmieniające się dane aby uzyskać wiarygodne porównanie jakości algorytmów (również pomiedzy niezależnymi badaczami). W danych tych zazwyczaj nie występują skrajne warunki ani błędne informacje, ze względu na ciężkość porównania prędkości (badacze uzywają innego sprzętu do badań), algorytmy porównywane są głównie pod względem trafności ocen czy rekomendacji.
W warunkach w których systemy rekomendacyjne są najczęściej wykorzystywane – systemach proponujących różne rzeczy w internecie dane ulegają ciągłym zmianom oznacza to, że nie tylko mogą pojawić się różne nieregularne sytuacje, ale również cały rozkład danych może się zmienić po dłuższym czasie. Do głównych problemów jakie mogą napotkać systemy należą:

- zjawisko zimnego startu – pojawienie się nowego użytkownika lub przedmiotu.
  Gdy użytkownik nie zdążył użyć wystarczająco wielu przedmiotów i nie załączył o sobie dodatkowych informacji, większość algorytmów nie jest w stanie określić jego upodobań. W przypadku gdy użytkownik nie użył żadnego przedmiotu nie istnieje wręcz możliwość stworzenia spersonalizowanej rekomendacji
- dylemat hipstera – niektórzy użytkownicy (czasem umyślnie) nie wpasowują się w żadne proste schematy – nie są podobni do żadnego innego użytkownika (do zbyt małej ilości dla niektórych algorytmów bazujących na tym podobieństwie).
  Użytkownicy tacy wybierają rzadko używane przez ogół przedmioty, co tworzy jednocześnie w danych przedmioty, które mimo dostatecznej liczby użyć, by przekroczyć limity zabezpieczające algorytmy przed sytuacjami trudnymi nie dają się zaklasyfikować odpowiednio. Osobie która niejako działa przeciwko systemowi rekomendacyjnemu ciężko przyporządkować trafne propozycje. Dopasowanie parametrów algorytmu tak, żeby zoptymalizować funkcje miary jakości, obejmujące tych użytkowników może wpłynąć negatywnie również na rekomendacje dla standardowych użytkowników na normalnych przedmiotach.
- asymetria pomiedzy ilością przedmiotów ocenionych przez użytkowników – w przypadku danych explicite często zachodzi sytuacja, gdy kilku aktywnych użytkowników wystawia dużo ocen, zaś reszta zaledwie pojedyncze.
  W takiej sytuacji program rekomendujący również powinien być w stanie przydzielić propozycje mniej aktywnym użytownikom. Zasadniczym problemem w takiej sytuacji jest to, że ocena jakości w niektórych (zazwyczaj dobrych) miarach może być zawyżona przy dobrym dopasowaniu się do tych pojedynczych użytkowników, jendocześnie przypasowującej mniej aktywnym oceny bliskie losowym.
- sytuacja w której nie da się już nic zaproponować – jeśli użytkownik użył znaczną ilość przedmiotów, system nie może nic zaproponować albo ponieważ nie ma już wystarczająco dużo przedmiotów nieużytych, albo wszystkie pozostałe zostały zaklasyfikowane jako silnie niepasujące.
  Sytuacja ta jest dosyć łatwa do kontrolowania, jednak należy o niej pamiętać przy projektowaniu algorytmu i jego ocenie.

Poza odpornością na wyżej wymienione sytuacje systemy używane przez ludzi powinny posiadać dodatkowe cechy zapewniające wygodę użycia. Podstawową różnicą między badaniem laboratoryjnym a systemem używanym przez ludzi jest szybkość użycia – użytkownik portalu internetowego nie jest skłonny czekać aż algorytm używany w rekomendacji przeanalizuje pełną bazę danych od początku (może to trwać od kilku minut do wielu godzin). Jednym z rozwiązań mogłoby być wyprodukowanie rekomendacji dla wszystkich użytkowników i pamiętanie ich dodatkowo w bazie danych jednak wymagałoby to przeliczania od początku po

wprowadzeniu kilku zmian, oraz nie dałoby szybkich rozwiązań nowym użytkownikom. W takich sytuacjach niektóre systemy mają możliwość przetworzenia danych i stworzenia modelu, który pozwoli wyprodukować rekomendacje wystarczająco szybko, a dodatkowo daje możliwość przypasowania nowego użytkownika w krótkim czasie. Niektóre algorytmy pozwalają jednak na bardzo szybkie modyfikacje modelu. Jeżeli jednak przetworzenie modelu zajmuje zbyt dużo czasu można pamiętać stary model, a po pewnej ilości zmian w bazie danych przeliczyć nowy model (nie zaburzając pracy portalu), Dodatkowymi zaletami takiego rozwiązania jest łatwa przenośność, oraz rozpraszalność systemu (mały model może być przechowywany w wielu miejscach, podczas gdy pełna baza danych znajduje się tylko w jednym).

Rozdział 2

Podstawowe techniki

2.1. Techniki niespersonalizowane

Najprostszą możliwą techniką tworzenia list rekomendacji jest stworzenie pojedynczej liczby najlepszych przedmiotów i jako rekomendacji zwrócenie użytkownikowi jej podlisty przedmiotów, których jeszcze nie użył. Wartością szeregującą przedmioty może być średnia ocen, popularność lub dowolna inna funkcja monotoniczna ze względu na oceny wystawione przedmiotowi. Główną wadą takiego podejścia jest to, ze tylko niewielki podzbiór popularnych przedmiotów zostanie zaproponowany ogółowi użytkowników, co pogłębi różnicę w popularności przedmiotów (przedmiot, który mógłby spodobać się użytkownikom nie będzie proponowany dlatego, że za mało osób go zna). Analogicznie, jako że gusta użytkowników są różne przedmiot oceniany jako dobry może obniżyć swoją średnią ponieważ będzie proponowany osobom do których nie pasuje, co może zaburzyć obraz danych.
Pomimo swoich wad niespersonalizowane rekomendacje oprócz dobrego punktu odniesienia stanowią jedyny sposób przydziału rekomendacji nowym użytkownikom (brak opisu i za mało użytych przedmiotów by zastosować ine algorytmy). Cecha ta powoduje, że często stanowią one fragment innych algorytmów używany w skrajnych przypadkach.

2.2. Filtrowanie kolaboratywne

Jedną z najbardziej podstawowych technik przewidywania ocen przedmiotów jest filtrowanie kolaboratywne polegające na wykorzystaniu wyłącznie informacji o użyciach przedmiotów (bez jawnych informacji o użytkownikach, czy przedmiotach). Predykcje tworzone są na podstawie założenia, że użytkownicy, którzy używali wych samych przedmiotów i podobnie je ocenili mają te same upodobania. Jeżeli ustali się na podstawie wspólnie ocenionych przedmiotów, że dwóch użytkowników jest bardzo podobnych można założyć, że przedmioty użyte tylko przez jednego z nich byłyby podobnie ocenione i przez drugiego.

2.2.1. Colaborative Filtering

Najprostszym algorytmem realizującym ideę filtorwania kolaboratywnego jest zdefiniowanie funkcji podobieństwa pomiędzy użytkownikami oraz wystawienie przedmiotowi oceny jako pewnej średniej z ocen najbardziej podobnych użytkowników, którzy ten przedmiot ocenili.
Do najpowszechniej stosowanych funkcji podobieństwa należą:

- Współczynnik korelacji liniowej Paersona:

$$S_{uv} = \frac{\sum_{i \in R_{uv}} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in R_{uv}} (r_{ui} - \overline{r_u})^2 \sum_{i \in R_{uv}} (r_{vi} - \overline{r_v})^2}}$$

Wyliczane dla par użytkowników, którzy ocenili po conajmniej dwa przedmioty, w tym conajmniej jeden wspólny.

- podobieństwo kosinusowe:

$$S_{uv} = \frac{\sum_{i \in R_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in R_{uv}} r_{ui}^2 \sum_{i \in R_{uv}} r_{vi}^2}}$$

Wyliczane dla par użytkowników, którzy ocenili conajmniej jeden wspólny przedmiot.

$S_{u,v}$ – podobieństwo użytkowników u i v, R_{uv} – zbiór przedmiotów, które ocenili zarówno użytkownik u jak i v, r_{ui} – ocena przedmiotu i wystawiona przez użytkownika u, $\overline{r_u}$ – średnia ocen wystawionych przez użytkownika u

Najczęściej stosowany jest współczynnik korelacji Paersona, jako że jest trafny w większości przypadków. Odjęcie średniej od oceny pozwala poradzić sobie z często spotykanym zjawiskiem spłaszczenia ocen (używanie przez użytkownika ocen tylko po jednej stronie skali), może jednak spowodować nieważność funkcji podobieństwa w przypadku gdy użytkownik ocenił tylko jeden przedmiot, lub wszystkie wspólnie ocenione przedmioty mają ocenę równą średniej (wtedy $S_{uv} = \frac{0}{0}$). Podobieństwo kosinusowe jest bardziej odporne na szczególne przypadki w danych, dlatego jest często wykorzystywane przy bardzo rzadkich danych (kiedy takie przypadki często występują). Oba podobieństwa nie sprawdzają się zbyt dobrze w przypadku gdy użytkownik wystawia wszystkim przedmiotom takie same oceny, oraz premiują podobieństwo pomiedzy użytkownikami z pojedynczymi wspólnie ocenionymi przedmiotami (przy jednym wspólnym przedmiocie podobieństwo kosinusowe = 1), dlatego, stosowane są często dodatkowe wagi ze względu na ilość wspólnie ocenionych przedmiotów lub odgraniczenia na ich minimalną ilość. Jeśli założenia funkcji nie są spełnione (za mało wspólnych przedmiotów), lub wartość jest bez sensu (np. $\frac{0}{0}$) funkcja przyjmuje wartość odpowiadającą brakowi podobieństwa (zazwyczaj 0).

Do wyliczenia przewidzianej oceny przedmiotu używa się ważonej średniej z ocen wystarczająco podobnych użytkowników (obcięcie listy użytkowników według wartości funkcji podobieństwa, ilości najbliższych sąsiadów, lub obu), zadanej wzorem:

$$r'_{ui} = \overline{r_u} + \frac{\sum_{v \in N_u} (r_{vi} - \overline{r_v}) S_{uv}}{\sum_{v \in N_u} S_{uv}}$$

N_u – zbiór sąsiadów (najbardziej podobnych użytkowników)

2.2.2. Slope One

Popularnym algorytmem wykorzystującym podobieństwo pomiędzy parami przedmiotów w nieco inny sposób jest Slope One. Algorytm zamiast stosować wyszukane miary podobieństwa statystycznego wektorów dla każdej pary przedmiotów zapamiętuje jedynie ilość użytkowniów którzy ocenili oba i sumaryczną różnicę w ich ocenach.
Predykcja ocen przedmiotów nieocenionych przebiega w tej metodzie również trochę inaczej. Zamiast wyciągania średniej z podobnych przedmiotów model przyjmuje, że skoro rozważany

przedmiot był oceniany wyżej od podobnego (takiego, który użyty był przez wielu wspólnych użytkowników), to i użytkownik, który go jeszcze nie ocenił przypisze mu ocenę wyższą o podobną wartość. Takie podejście do problemu daje następujący wzór:

$$r'_{ui} = \frac{\sum\limits_{j \in R_u} (\mathrm{diff}_{ij} + r_{uj}) \cdot |R_{ij}|}{\sum\limits_{j \in R_u} |R_{ij}|}$$

R_{ij} – zbiór użytkowników, którzy ocenili zarówno przedmiot i jak i j,
diff_{ij} – średnia różnica ocen pomiedzy przedmiotami i oraz j (w przeciwieństwie do wszystkich podobieństw antysymetryczna)
W przeciwieństwie do algorytmu CF który tak na prawdę do predykcji ocen wykorzystuje regresję liniową ($f(x) = ax + b$), SO wykorzysutje prostszą regresję jedno parametrową ($f(x) = x + b$). Użycie tak prostego modelu czyni algorytm o wiele bardziej odpornym na przeuczenie.

2.2.3. Rozkład według wartości osobliwych

Rozkład SVD (Singular Value Decomposition) macierzy $m \times n$ przedstawiony jest wzorem: $M = U\Sigma V^{\perp}$
gdzie U – macierz unitarna $m \times m$, Σ – macierz diagonalna zawierająca wartości własne macierzy M, V – macierz unitarna $n \times n$
(w przypadku macierzy nad ciałem rzeczywistym macierze unitarne są ortogonalne)

Rozkład SVD stosowany jest między innymi do kompresji macierzy:
W przypadku gdy $m > n$ ($m < n$) $m - n$ ostatnich kolumn macierzy U ($n - m$ ostatnich wierszy macierzy $V^{\perp}$) jest nieistotne, gdyż są mnożone zawsze przez wektor 0, więc można ich nie przechowywać w pamięci.
Podobnie dzieje się w przypadku wierszy (kolumn) przypadających na zerowe wartości własne w macierzy Σ.
Zastosowanie jednej permutacji do kolumn macierzy U, wierszy macierzy $V^{\perp}$, oraz wartości własnych w macierzy Σ nie zmienia ich iloczynu.
Po zastosowaniu tych przekształceń dla $r = \mathrm{rank}(M)$ otrzymujemy rozkład $M = U'\Sigma'V'^{\perp}$, gdzie macierz U' ma wymiary $m \times r$ Σ' jest macierzą diagonalną rozmiaru $r \times r$, zaś $V'^{\perp}$ macierzą rozmiaru $r \times n$ (a więc rozkład pozwala na przechowywanie mniejszej ilości danych dla $r < \frac{mn}{m+n+1}$).
Norma Frobeniusa macierzy: $\|M\|_F = \sqrt{\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} |m_{ij}|^2}$

Twierdzenie 2.2.2.1. (Eckhart-Young 1936) Dla dowolnego $r < \mathrm{rank}(M)$ macierz $\tilde{M} = U\Sigma_r V^{\perp}$, (gdzie Σ_r, to macierz Σ z pozostawionymi jedynie r wartościami własnymi o największych wartościach bezwzględnych) jest najlepszym przybliżeniem macierzy M pod względem normy Frobeniusa, wśród macierzy o rzędzie $\leqslant r$.
Zastosowanie rozkładu SVD i Twierdzenia Eckharta-Younga pozwala na stratną kompresję macierzy.

Model SVD

Algorytmy rekomendacyjne z grupy SVD mają na celu wygenerowaniu mozliwie najlepszych macierzy U' i $V'^{\perp}$ dla zadanego r reprezentujących pełną macierz ocen użytkownik–przedmiot

(wiersz zawiera oceny wystawione przez jednego użytkownika, zaś kolumna oceny wystawione jednemu przedmiotowi) na podstawie rzadkiej macierzy znanych już ocen.
Oznaczenia:
$p_u \in \mathbb{R}^f$ – wektor związany z użytkownikiem u
$b_u \in \mathbb{R}$ – podstawa oceny związana z użytkownikiem u
$q_i \in \mathbb{R}^f$ – wektor związany z przedmiotem i
$b_i \in \mathbb{R}$ – podstawa oceny związana z przedmiotem i
μ – średnia wszystkich ocen wystawionych wszystkim przedmiotom
$K = \{(u,i)|$ użytkownik u wystawił ocenę przedmiotowi $i\}$
λ_* – różne stałe użyte do regularyzacji (ustawiane w celu optymalizacji pożądanych miar jakości predykcji)
r_{ui} – prawdziwa ocena wystawiona przedmiotowi i przez użytkownika u
$\tilde{r}_{ui}$ – przewidziana przez system ocena przypisana użytkownikowi u i przedmiotowi i
α – prędkość uczenia (zazwyczaj mała stała rzędu 0.01)

$$\tilde{r}_{ui} = \mu + b_u + b_i + p_u \cdot q_i$$

zaś parametry (b_*, p_*, q_*) równań wyliczone są poprzez minimalizację funkcji kwadratowej:

$$\sum_{(u,i)\in K} (r_{ui} - \mu - b_u - b_i - p_u \cdot q_i)^2 + \lambda \cdot (b_u^2 + b_i^2 + \|p_u\|^2 + \|q_i\|^2)$$

Dla f =rank(M) można znaleźć optymalną wartość dla wektorów p i q będących wierszami macierzy U i V (macierz Σ wmnożona w pozostałe) z rozkładu SVD macierzy M (z dowolnymi wartościami zastępującymi nieznane). Ustawienie f na odpowiednią wielkość pozwala uzyskać z jednej strony model wystarczająco prosty do wyliczenia, oraz wystarczająco skomplikowany, aby wektory p_* i q_* mogły oddać profile użytkowników i przedmiotów. Wartość f odpowiednio mniejsza od rank(M) uniemożliwia modelowi przeuczenie się znanych ocen i powodując uproszczenie modelu ustala wartości $\tilde{r}_{ui}$ na najmniej komplikujące model. Ustawienie stałej λ na niezerową pozwala uniknąć optymalizacji funkcji w skrajnych, oddalonych wartościach wektorów (co mogłoby wynikać z zaburzeń w danych).
Ze względu na brakujące wartości w macierzy nie mogą zostać zastosowane standardowe metody rozkładu dlatego też stosowane jest schodzenie po gradiencie iterowane po znanych wartościach macierzy. Wartości μ, b_u, b_i stosowane są aby macierz będąca iloczynem macierzy złożonych z wektorów p_* i q_* była możliwie bliska zerowej dzięki czemu można uzyskać większą dokładność oraz uniknąć osiągania dużych wartości mogących wpłynąć na stabilność zbieżności.
Zgodnie z określonymi wytycznymi zmienne w algorytmach klasy SVD optymalizowane są poprzez wielokrotne przeprowadzenie procesu opisanego pseudokodem:

foreach $(u,i) \in K\{$
    $err = \tilde{r}_{ui} - r_{ui}$
    $x+ = \alpha \cdot (err \cdot \frac{\partial err}{\partial x} - \lambda_x \cdot x)$
$\}$

Gdzie uaktualnienie x oznacza zmianę każdej zmiennej zawartej we wzorze na błąd (err) – dotyczy to również dodatkowych zmiennych dodanych przez rozszerzenia podstawowego algorytmu.
Warte wspomnienia jest również to, że w przypadku pełnej macierzy M algorytmy te przy odpowiedniej liczbie iteracji i odpowiednio małej prędkości uczenia w założeniu symulują proces dojścia do rozwiązania takiego jak w kompresji SVD.

Algorytm SVD++

Najczęściej uzywanym rozszerzeniem modelu SVD jest algorytm SVD++, korzystający również z informacji implicite. Do modelu dodane zostają wektory $y_i \in \mathbb{R}^f$ reprezentujące informacje o użyciach przedmiotów. Równanie docelowe przyjmuje postać:

$$\tilde{r}_{ui} = \mu + b_u + b_i + q_i \cdot \left(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

gdzie $N(u)$ oznacza zbiór przedmiotów użytych przez użytkownika u (zarówno tych ocenionych jak i nie).

(dodawanie wektorów oznacza dodanie odpowiadających współrzędnych, zaś mnożenie iloczyn skalarny)

Parametry równań w tym przypadku znajdowane są poprzez optymalizację modelu w procesie iteracyjnym opisanym pseudokodem:

for I iterations{
  foreach $(u, i) \in K${
    $py = p_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j$
    $\tilde{r}_{ui} = \mu + b_u + b_i + q_i \cdot py$
    $err = \tilde{r}_{ui} - r_{ui}$
    $b_u = b_u + \alpha \cdot (err - \lambda_1 \cdot b_u)$
    $b_i = b_i + \alpha \cdot (err - \lambda_2 \cdot b_i)$
    $p_u = p_u + \alpha \cdot (err \cdot q_i - \lambda_3 \cdot p_u)$
    $q_i = q_i + \alpha \cdot (err \cdot py - \lambda_4 \cdot q_i)$
    foreach $j \in N(u)${
      $y_j = y_j + \alpha \cdot (err \cdot |N(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_5 \cdot y_j)$
    }
  }
}

stała α oznacza szybkość uczenia (powinna więc zależeć od ilości przewidzianych iteracji = dostępnego czasu na działanie algorytmu uczącego)

stałe λ_* służą lepszemu dostosowaniu algorytmu do problemu i powinny zostać ustalone tak, by optymalizowały funkcje oceny na zbiorze testowym

wartości początkowe wektorów p_* i y_* powinny zostać wylosowane np. z rozkładu normalnego (jeśli wszystkie wektory początkowe mają wartość 0 na tej samej współrzędnej to tak pozostanie, jeśli wszystkie wektory mają te same wartości na określonych współrzędnych, to nie będą mogły się zróżnicować i będą redundantne)

2.2.4. Spersonalizowany ranking Bayesa

W przypadku dostępu do danych wyłącznie typu implicite popularną metodą tworzenia spersonalizowanych list rekomendacyjnych jest faktoryzacja macierzy oparta o analizę bayesowską prawdopodobieństw.

Mając dostępną binarną macierz użyć przedmiotów traktujemy negatywne wartości bardziej jako wartość nieznaną niż jako stwierdzenie, że dany przedmiot nie podobał się użytkownikowi. Aby uzyskać więcej niż dwie wartości, a przez to i rozróżnienie w obrębie przedmiotów nieznanych model BPR (Bayesian personalized ranking) stara się oszacować dla danego użytkownika i każdej pary przedmiotów prawdopodobieństwo, że preferuje on pierwszy z nich. Aby uzystać początkowe dane używane do filtrowania kolaboratywnego pomiedzy użytkow-

nikami przyjmujemy, że te już użyte są bardziej preferowane od nieznanych i dla każdego wiersza macierzy tworzymy nową macierz kwadratową.

	i1	i2	i3	i4	i5
u1	+	?	?	?	+
u2	?	?	+	?	+
u3	+	+	+	?	?
u4	?	?	?	?	+

$i >_{u_2} j$

	j1	j2	j3	j4	j5
i1	X	?	-	?	-
i2	?	X	-	?	-
i3	+	+	X	+	?
i4	?	?	-	X	-
i5	+	+	?	+	X

Decyzję o tym czy przedmiot i jest bardziej pasujący do użytkownika u od przedmiotu j jest podejmowana na podstawie prawdopodobieństwa warunkowego $\mathbb{P}(i >_u j|\Theta)$, gdzie Θ jest modelem wygenerowanym przez algorytm. Podobnie jak w przypadku modelu SVD Θ składa się z wektorów $p_u, q_i \in \mathbb{R}^f$ oraz wartości $b_i \in \mathbb{R}$, oraz powstaje poprzez iterowaną optymalizację funkcji celu:

$$\sum_{(u,i,j)\in D_K} \ln\left(\sigma(\tilde{s}_{uij})\right) - \lambda\|\Theta\|^2$$

gdzie $D_K = \{(u,i,j) : i \in N(u) \& j \notin N(u)\}, \tilde{s}_{u,i,j} = \tilde{r}_{ui} - \tilde{r}_{uj}, \|\Theta\|^2 = (b_i^2 + \|p_u\|^2 + \|q_i\|^2)$, $\sigma(x) = \frac{1}{1+e^{-x}}$
Zaś przewidywana ocena przedmiotu (opisująca wyłącznie kolejność preferencji) jest równa $\tilde{r}_{ui} = b_i + p_u \cdot q_i$ (μ oraz b_u nie mają tutaj sensu)
Algorytm iterowanego poprawiania modelu wygląda podobnie do tego z SVD:

for I iterations{
  wylosuj (u,i,j) z D_K
  $\tilde{s} = b_i - b_j + p_u \cdot (q_i - q_j)$
  $err = \frac{e^{-\tilde{s}}}{1+e^{-\tilde{s}}}$
  $b_i = b_i + \alpha \cdot (err - \lambda_2 \cdot b_i)$
  $b_j = b_j + \alpha \cdot (-err - \lambda_2 \cdot b_j)$
  $p_u = p_u + \alpha \cdot (err \cdot (q_i - q_j) - \lambda_3 \cdot p_u)$
  $q_i = q_i + \alpha \cdot (err \cdot p_u - \lambda_4 \cdot q_i)$
  $q_j = q_j + \alpha \cdot (-err \cdot p_u - \lambda_4 \cdot q_j)$
}

W tym algorytmie trójki $(u,i,j) \in D_K$ są wybierane losowo, gdyż jest ich na ogół o wiele wiecej niż par $(u,i) \in K$ i przez to już jedna iteracja po wszystkich mogła by być zbyt droga obliczeniowo.

2.3. Filtrowanie na podstawie opisów

Gdy do danych o przedmiotach i/lub użytkownikach dołączone są dodatkowe informacje podobieństwo można wywnioskowac na podstawie właśnie tych opisów.
W przypadku przedmiotów najczęściej spotykanymi formami opisów są:

- przypasowanie do kategorii (często dany przedmiot może należeć do kilku kategorii na raz jak na przykład film może być jednocześnie komedią i science fiction)
- słowa opisujące – tagi (w przeciwieństwie do kategorii tagi mogą nie mieć stałej struktury)
- opis tekstowy (streszczenie/specyfikacja)

W przypadku użytkowników:

- kategorie oznaczone przez użytkownika jako ulubione
- opis (rzadko samego użytkownika, częściej upodobań związanych z przedmiotami w serwisie)
- powiązania z innymi użytkownikami (częstość kontaktów, informacje z innych źródeł jak serwisy społecznościowe)

Systemy generujące rekomendacje oparte o dane wyczerpujących opisów mogą dawać dobre wyniki już przy najprostszych użytych technikach. Algorytm zaproponuj najpopularniejsze przedmioty z ulubionych kategorii użytkownika nie jest wiele bardziej skomplikowany od niespersonalizowanego, a jednocześnie daje lepsze wyniki (choć dzieli też większość jego wad). Dane zamieszczone są jednak rzadko wystarczające, aby zbudować dobry system tylko na opisach, które są często niekompletne (szczególnie w przypadku użytkowników – wielu nie wprowadza informacji o sobie) niedokładne lub wręcz błędne. Dlatego częściej wykorzystywane są w połączeniu z danymi o użyciach przedmiotu. Dane te mogą posłużyć jako miara podobieństwa: przedmioty z tych samych kategorii powinny być podobne do siebie (w przypadku przypisań do wielu kategorii można również definiować podobieństwa przechodnie, a w przypadku wielu kategorii użyć również podobieństw pomiędzy nimi). Użytkownicy przypisujący przedmiotom takie same tagi, czy tworzący o nich podobne komentarze również mogą zostać zakwalifikowani jako sąsiedzi w grafie podobieństw. W przypadku danych tekstowych zbieżności można wyciągnąć dzięki zastosowaniu technik text miningowych, co może prowadzić do nietrywialnych wniosków na temat przedmiotów i pomóc stworzyć lepszą ich klasyfikację.
Posiadanie informacji o przedmiotach pozwala również na stworzenie systemów rekomendacyjnych współpracujących z użytkownikiem. W przypadku usystematyzowanych danych o parametrach przedmiotów (cena, lokalizacja, dostępność, itp.) można pozwolić użytkownikowi na wybranie które cechy są dla niego najbardziej istotne i albo zawęzić wyszukiwanie albo też dostosować różne stałe w algorytmach (aby podobieństwo pod danym względem miało większą wagę).

2.4. Systemy hybrydowe

Rozdział 3

TODO - ulepszenia

3.1. TODO - sekcja o ulepszeniach standardowych algorytmów

3.1.1. BPR

3.2. TODO - sekcja o ulepszeniach - zupełnie nowe algorytmy

3.2.1. Complex

Standardową reprezentacją danych rozważanych w filtorwaniu kolaboratywnym jest macierz użyć przedmiotów (lub lista ocen pozwalająca zaoszczędzić pamięć). Inną formą wizualizacji danych jest graf w którym wierzchołkami są użytkownicy i przedmioty, zaś krawędzie między nimi oznaczają powiązania miedzy nimi – oceny wystawione przez użytkowników przedmiotom lub znane podobieństwo. Zazwyczaj informacje zawarte w krawędziach mogą zostać spłaszczone do rzeczywistej wartości liczbowej oznaczającej upodobanie użytkownika do przedmiotu lub podobieństwo przedmiot-przedmiot użytkownik-uzytkownik, w przypadku braku informacji na temat interakcji krawędź nie istnieje. Klasyczne algorytmy bazujące na podobieństwie pomiędzy użytkownikami lub przedmiotami skupiają się w większości na wyliczeniu tych podobieństw dla każdej pary (sporządzeniu maciery podobieństwa) i w celu predykcji nowej oceny aplikują pewną funkcję agregującą oceny podobnych instancji. Przy spojrzeniu na nie z perspektywy grafu algorytmy te skupiają się na stworzeniu brakującej krawędzi pomiędzy użytkownikiem i przedmiotem na podstawie ścieżek długości 3 między nimi.
Podstawowym problemem tego podejścia jest to, że w przypadku rzadkich macierzy użyć przedmiotów (w zastosowaniach macierze te są zazwyczaj bardzo rzadkie) ścieżek takich może być bardzo mało, co oznacza małą wiarygodność predykcji, zaś brak ścieżek uniemożliwia jakąkolwiek predykcję. Jednym z rozwiązań problemu jest posłużenie się również dłuższymi ścieżkami, takie podejście może jednak napotkać barierę złożonościową (algorytmy CF i SO do wyliczenia wszystkich ocen potrzebują czasu $O(m^2n^2)$ dla macierzy o rozmiarach $m \times n$ badając tylko bardzo krótkie ścieżki, przy dłuższych ten czas ulegałby potęgowaniu), dlatego aby uzyskać lepsze wyniki w rozsądnym czasie model podobieństwa musi zostać uproszczony. Jeden z modeli wykorzystujących dalsze podobieństwo w praktyce został zaproponowany w [205]. Algorytm Complex upodobanie użytkownika do przedmiotu predykuje jako sumę wartości po ścieżkach wiodących od użytkownika do tego przedmiotu aplikując do nich wagi zmniejszające się wraz z rosnącą ich długością. Twórcy algorytmu wykorzystują model grafowy w którym upodobanie oznaczone jest poprzez krawędzie skierowane od użytkownika do przedmiotu z pojedynczą wartością rzeczywistą i uogólniają na dłuższe ścieżki mnożąc

wartości z krawędzi zgodnie ze schematem:
TODO obrazek jak w pracy TODO

Ze schematu można wywnioskować następujące reguły:

- $\omega_{similar} = -\omega_{like}^2$
- $\omega_{like} = \omega_{similar} \cdot \omega_{like}$
- $\omega_{similar} = \omega_{similar}^2$

Z reguł wynika, że ograniczenie do wartości rzeczywistych nie wystarcza, jednak stosując wartości zespolone otrzymuje się wygodny model, w którym wartości podobieństwa wyrażone są liczbami rzeczywistymi, zaś upodobanie wartościami czysto zespolonymi (stąd też pochodzi nazwa algorytmu). Dzięki takiej interpretacji sumy wartości ścieżek długości l mogą zostać wyliczone dzięki zwykłemu przemnożeniu przez siebie macierzy zawierającej wartości dla pojedynczych krawędzi l krotnie.
Aby zastosować algorytm do prawdziwych danych należy uwzględnić kilka rzeczy – doprowadzić macierz ocen do odpowiedniego stanu, można też przy okazji dokonać kilku uproszczeń. Jako, że aby dodawanie ścieżek różnych długości miało jakąkolwiek możliwość utrzymania wiadomości o prawdziwych ocenach wystawionych przez użytkowników przedmiotom trzeba by zastosować bardzo skomplikowane wagi na konkretnych krawędzich (jeżeli użytkownik ocenił dużo przedmiotów lub przedmiot był licznie używany, związane z nim oceny mają rosnący wpływ na dłuższe ścieżki) algorytm sprawdza się dobrze tylko w celu stworzenia list rekomendacji. Jeżeli unormalizować wartości podobieństw w macierzy tak aby, zrównoważyć wykładniczo szybko rosnącą ilość ścieżek wraz ich długością (dzieląc w podstawowej macierzy wartości przez rozmiar macierzy) narzucającą się macierzą wynikową było by $I + A + \frac{1}{2}A^2 + ... = e^A$, jednak ponieważ ścieżki większych długości są mniej istotne dla prawdziwych wartości (nawet po zniwelowaniu oddziaływania ilościowego) w praktyce lepiej stosować wagi, które bardziej premiują krótkie ścieżki. Jeśli model danych nie posiada żadnych krawędzi pomiędzy użytkownikami lub przedmiotami macierz grafu (dwudzielnego) można zapisać w uproszczony sposób:
$$A = \begin{bmatrix} A_{UU} & A_{UI} \\ A_{IU} & A_{II} \end{bmatrix} = \begin{bmatrix} 0 & A_{UI} \\ A_{IU} & 0 \end{bmatrix} = \begin{bmatrix} 0 & jB \\ -jB^{\perp} & 0 \end{bmatrix} \Rightarrow$$
$$A^{2k} = \begin{bmatrix} (BB^{\perp})^k & 0 \\ 0 & (B^{\perp}B)^k \end{bmatrix}, A^{2k+1} = \begin{bmatrix} 0 & j(BB^{\perp})^k B \\ -j(B^{\perp}B)^k B^{\perp} & 0 \end{bmatrix}$$
Ponieważ przy odczytywaniu wyników interesujące są jedynie ścieżki od użytkownika do przedmiotu (prawa górna podmacierz) macierz przechowująca wynik przyjmuje postać $C = \sum_{k=0}^{\infty} a_k (BB^{\perp})^k B$, gdzie a_k to odpowiednie wagi, zaś B to odpowiednio zmodyfikowana macierz ocen wystawionych przez użytkowników przedmiotom. Autorzy pracy w której przedstawiony został algorytm przyjęli model w którym negatywna ocena wystawiona przez użytkownika oznacza mniejsze upodobanie w przedmiocie niż brak jego użycia, jednocześnie celując w rekomendację tylko przedmiotów, które użytkownik powinien ocenić pozytywnie (podejście pośrednie pomiędzy predykcją ocen przy użyciu informacji explicite i tworzenia list na podstawie informacji implicite) stąd oceny można w macierzy spłaszczyć do 1 w przypadku pozytywnej i -1 negatywnej i 0 w przypadku braku oceny/użycia (neutralną najlepiej zakwalifikować jako pozytywną – bardziej lubiane niż w przypadku braku użycia).

Bibliografia

[191] Marcelo G. Manzato, Marcos A. Domingues, Solange O. Rezende, "Optimizing Personalized Ranking in Recommender Systems with Metadata Awareness" *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pp. 191 - 197

[205] Feng Xie, Zhen Chen, Jiaxing, Xiaoping Feng, Wenliang Huang, Jun Li, "A Link Prediction Approach for Item Recommendation with Complex Number" *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pp. 205 - 212