

**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

December 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**

W pracy przedstawiono problemy rekomendacji przedmiotów oraz predykcji ocen, podstawowe techniki ich rozwiązania, jak i algorytmy je implementujące. Przedstawiono również kilka sposobów ich ulepszenia zaproponowanych na konferencji *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*. Do przedstawionych algorytmów należą CF (collaborative filtering), SVD (singular value decomposition), SVD++, gSVD++, BPR (Bayesian personalized ranking), SO (Slope One), oraz ulepszenia: CF ADV, MABPR (Metadata awareness Bayesian personalized ranking), Complex plus różne ich odmiany. Praca zawiera również przegląd różnych metod oceny systemów oraz szczegółowe porównanie wyżej wymienionych algorytmów na nich bazujące.

## **Słowa kluczowe**

rekomendacje, predykcja ocen, systemy rekomendacyjne, collaborative filtering, Slope One, SVD, BPR, Complex, porównanie algorytmów

## **Dziedzina pracy (kody wg programu Socrates-Erasmus)**

11.0 Matematyka, Informatyka:

11.4 Sztuczna inteligencja

## **Klasyfikacja tematyczna**

68–XX Computer science

68Txx Artificial Intelligence

68T05 Learning and adaptive systems

## **Tytuł pracy w języku angielskim**

Methods of recommendation systems improving



# Spis treści

<b>Wprowadzenie</b>	5
<b>1. Sformułowanie problemu</b>	7
1.1. Podstawowe definicje	7
1.2. Przeznaczenie systemów	8
1.3. Problemy stojące przed systemami rekomendacyjnymi	9
<b>2. Podstawowe techniki</b>	11
2.1. Techniki niespersonalizowane	11
2.2. Filtrowanie kolaboratywne	11
2.2.1. Collaborative Filtering	11
2.2.2. Slope One	13
2.2.3. Rozkład według wartości osobliwych	14
2.2.4. Spersonalizowany ranking Bayesa	16
2.3. Filtrowanie na podstawie opisów	17
2.4. Systemy hybrydowe	18
2.4.1. gSVD++	18
<b>3. Ulepszenia</b>	21
3.1. Standardowe ulepszenia algorytmów	21
3.1.1. ulepszenia CF	22
3.1.2. Wykorzystanie metadanych jako uzupełnienie danych implicite	24
3.2. Nowe algorytmy zbudowane na podstawie starych	26
3.2.1. Complex	27
<b>4. Ocena i porównanie systemów</b>	31
4.1. Opis danych	31
4.2. Załączona implementacja algorytmów	32
4.3. Ewaluacja predykcji ocen	34
4.4. Jakość list rekomendacyjnych	35
4.4.1. ROC i AUC	36
4.4.2. Precyzja i MAP	36
4.4.3. Porównanie systemów	37
4.4.4. Pokrycie	39
4.5. Porównanie czasowe	40
4.6. Podsumowanie	41
<b>Bibliografia</b>	43



# Wprowadzenie

W dzisiejszym świecie dzięki globalizacji, rozwojowi mediów i łatwiejszemu transportowi ludzie mają dostęp do coraz większej ilości produktów. Różnorodność towarów i mnogość dóbr intelektualnych dostępnych człowiekowi stała się tak duża, że nikt nie jest w stanie zapoznać się z wszystkimi możliwościami produktów, które mogłyby go zainteresować. W takiej sytuacji coraz bardziej przydatne stają się systemy które mogłyby pomóc użytkownikowi w znajdowaniu takich rzeczy, a do tego były szybkie i nie wymagały od użytkownika dokładnego precyzowania jego oczekiwań. Rozwiązaniem dla tego zapotrzebowania są systemy rekomendacyjne, które na podstawie znanych informacji o użytkowniku i historii jego zakupów lub użyć przedmiotów mogą znaleźć inne produkty, które powinny go zainteresować i dzięki temu ograniczyć czas i uciążliwość samodzielnego ich poszukiwania.

Systemy rekomendacyjne znajdują swoje zastosowanie w sklepach internetowych, reklamach przeglądarkowych (która reklama może zainteresować konkretnego użytkownika), stronach dających dostęp do filmów, książek czy utworów muzycznych w których to przypadkach dodatkowym plusem jest to, że użytkownik nie musi podejmować wysiłku związanego z załączaniem informacji o sobie ani o swoich upodobaniach, gdyż wystarczające są informacje o jego przeszłych działaniach na konkretnej stronie (choć załączenie takich informacji pozwala na jeszcze lepsze predykcje). Dodatkowo jeśli dostępne są informacje o odczuciach użytkownika na temat pewnych przedmiotów (oceny, komentarze) możliwe jest przewidzenie takiej oceny odnośnie nieznanego jeszcze produktu, co może pomóc użytkownikowi w podjęciu decyzji o jego kupnie czy użyciu.

Aby taka funkcjonalność przynosiła wymierne korzyści predykcje powinny być jak najlepsze. Oznacza to popyt na jak najlepsze algorytmy, zarówno te które dobrze radzą sobie w ogólnym przypadku jak i te stworzone dla konkretnego zastosowania.

Badania nad tworzeniem coraz lepszych systemów rekomendacyjnych w dużej mierze skupiają się nad ulepszaniem algorytmów znanych uprzednio, dodawaniem do nich dodatkowych części, wyspecjalizowywaniem do konkretnych zastosowań, lub dostosowaniem do takich w których wcześniej sobie nie radził.

Niniejsza praca ma na celu przedstawienie różnych algorytmów używanych w systemach rekomendacyjnych, pokazanie kilku ich ulepszeń oraz dokładne i możliwie obiektywne porównanie wyników przez nie osiągniętych (w przypadku nowszych algorytmów wcześniej niedostępne).

Praca składa się z czterech rozdziałów. W pierwszym rozdziale opisane jest sformułowanie problemu rekomendacji, podstawowe definicje, cele, oraz główne trudności z jakimi musi zmierzyć się dobry system rekomendacyjny. W rozdziale 2. zaprezentowany jest przegląd i klasyfikacja technik używanych w celach predykcji ocen oraz budowania list rekomendacji. Jest tam również przedstawione kilka wybranych, popularnych systemów rekomendacyjnych, w tym tych trywialnych i tych znanych już od dawna, ale też takich które zostały wymyślone

stosunkowo niedawno, jednak już teraz funkcjonują jako punkt odniesienia w pracach badawczych. Wśród przedstawionych znajdują się techniki niespersonalizowane, jak i algorytmy CF, SVD, SVD++, BPR oraz gSVD++.

W kolejnym rozdziale przybliżone są algorytmy z wybranych publikacji przedstawionych na konferencji *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, które stworzone zostały jako modyfikacje algorytmów przedstawionych we wcześniejszym rozdziale. Prace z których pochodzą przedstawione w tym rozdziale algorytmy wybrane zostały również pod kątem działania przy takim samym typie i rozkładzie danych jak podstawowe techniki dzięki temu możliwe jest przeprowadzenie porównania nie tylko pomiędzy algorytmem i jego ulepszeniem, ale także pomiędzy ulepszeniami przedstawionymi w różnych pracach.

W ostatnim rozdziale zademonstrowane są różne techniki oceny systemów rekomendacyjnych. W skład przedstawionych tam funkcji oceny wchodzi miary błędów przewidywanych ocen MAE, MSE, miary oparte na trafieniach przedmiotów z list rekomendacyjnych w te rzeczywiście preferowane przez użytkownika: krzywe ROC, ich AUC, oraz Precision i MAP, oraz miara różnorodności propozycji systemu – Coverage. W większości prac badawczych skupiono się na pojedynczych miarach oceny i zaburzeniach rozkładów danych, dzięki czemu ulepszenia mogą uzyskiwać wyraźną poprawę względem oryginalnego algorytmu. O ile jest to dobre podejście jeśli chce się uzyskać system dobry w jednej konkretnej sytuacji o tyle nie da się dobrze w tej sytuacji porównać algorytmów z różnych prac. Aby dokonać takiego porównania zaimplementowałem w języku R te algorytmy jak i wymienione wyżej metody oceny, a następnie przedstawiłem w pracy rezultaty ich użycia. Dzięki temu porównaniu przy pomocy użycia 5-krotnej walidacji krzyżowej na danych MovieLens100k [ML] możliwe staje się zweryfikowanie efektywnych zysków uzyskanych dzięki wprowadzaniu ulepszeń, oraz rozstrzygnięcie, który z algorytmów rzeczywiście zyskuje najlepsze wyniki w dość popularnie stosowanym przypadku testowym (w większości prac badawczych skupiono się na pojedynczych miarach oceny i zaburzeniach rozkładów danych, dzięki czemu ulepszenia mogą uzyskiwać wyraźną poprawę względem oryginalnego algorytmu). Przeanalizowana jest tam również szybkość tych systemów, co pozwala dobrać system również pod kątem rzeczywistych rozwiązań czasu rzeczywistego.

Do pracy dołączone są skrypty w języku R służące wygenerowaniu porównań użytych w ostatnim rozdziale. Mogą one służyć do wykonania innych badań, jednak są dosyć mocno dostosowane do rozpatrywanego typu danych, dlatego nie nadają się do użycia w prawdziwych systemach.



# 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. Dodatkowo w większości zastosowań zakłada się, że użycie/posiadanie przedmiotów nie obniży użyteczności innych (np. po kupnie mebla raczej nie będziemy potrzebować w najbliższym czasie drugiego o podobnym zastosowaniu, zaś w przypadku książki czy filmu po względnie krótkim czasie można użyć następnego w tym podobnego). 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 użycia 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 zmniejszenie 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 używane)
  - 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 systemem.

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 pomiędzy niezależnymi badaczami). W danych tych zazwyczaj nie występują skrajne warunki ani błędne informacje, ze względu na ciężkość porównania prędkości (badacze używają 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 pomiędzy 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 użytkownikom. 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, jednocześnie 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, że 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 inne 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 tych samych przedmiotów i podobnie je ocenili mają te same upodobania. Jeżeli ustalili 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. Collaborative Filtering

Najprostszym algorytmem realizującym ideę filtrwania 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:

$$S_{u,v} = \frac{\sum_{i \in R_{u,v}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in R_{u,v}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in R_{u,v}} (r_{v,i} - \bar{r}_v)^2}}$$

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

- podobieństwo kosinusowe:

$$S_{u,v} = \frac{\sum_{i \in R_{u,v}} r_{u,i} \cdot r_{v,i}}{\sqrt{\sum_{i \in R_{u,v}} r_{u,i}^2 \sum_{i \in R_{u,v}} r_{v,i}^2}}$$

Wyliczane dla par użytkowników, którzy ocenili co najmniej jeden wspólny przedmiot.

$S_{u,v}$  – podobieństwo użytkowników  $u$  i  $v$ ,  $R_{u,v}$  – zbiór przedmiotów, które ocenili zarówno użytkownik  $u$  jak i  $v$ ,  $r_{u,i}$  – ocena przedmiotu  $i$  wystawiona przez użytkownika  $u$ ,  $\bar{r}_u$  – 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  $S_{u,v} = \frac{0}{0}$ ). 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 pomiędzy 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:

$$\tilde{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N_u} (r_{v,i} - \bar{r}_v) S_{u,v}}{\sum_{v \in N_u} S_{u,v}}$$

$N_u$  – zbiór sąsiadów (najbardziej podobnych użytkowników)

W prawdziwych danych występuje zazwyczaj dysproporcja pomiędzy użytkownikami i przedmiotami objawiająca się poza różnicą ilości także ilością użyć oraz wielkością przecięć użyć (przedmioty ocenione przez dwóch konkretnych użytkowników, lub użytkownicy którzy użyli dwóch konkretnych przedmiotów). Jako że algorytm jest symetryczny ze względu na użytkowników i przedmioty ich rola może zostać zamieniona. Dzięki możliwości wyboru ról można uzyskać lepsze własności systemu jak szybkość przy dysproporcji rozmiarów macierzy użyć oraz zmniejszyć ilość przypadków w których algorytm sobie nie radzi (za mało sąsiadów o określonej własności).

### przykład

Rozważmy CF z podobieństwem korelacji Paersona między użytkownikami na danych:

	i1	i2	i3
u1	?	3	1
u2	2	5	?
u3	4	4	1

Wtedy  $\bar{r}_{u1} = 2, \bar{r}_{u2} = 3.5, \bar{r}_{u3} = 3$

$$S_{u1,u2} = \frac{1 \cdot 1.5}{\sqrt{1^2 \cdot (1.5)^2}} = 1$$

$$S_{u1,u3} = \frac{1 \cdot 1 + (-1) \cdot (-2)}{\sqrt{(1^2 + (-1)^2) \cdot (1^2 + (-2)^2)}} = \frac{3}{\sqrt{10}}$$

$$S_{u2,u3} = \frac{-1.5 \cdot 1 + 1.5 \cdot 1}{\sqrt{((1.5)^2 + (-1.5)^2) \cdot (1^2 + 1^2)}} = 0$$

$$\tilde{r}_{u1,i1} = 2 + \frac{-1.5 \cdot 1 + 1 \cdot \frac{3}{\sqrt{10}}}{1 + \frac{3}{\sqrt{10}}} = 1.72$$

$$\tilde{r}_{u2,i3} = 3.5 + \frac{-1 \cdot 1}{1} = 2.5$$

### 2.2.2. Slope One

Popularnym algorytmem wykorzystującym podobieństwo pomiędzy parami przedmiotów w nieco inny sposób jest Slope One (opisane w pracy [1]). Algorytm zamiast stosować wyszukiwane miary podobieństwa statystycznego wektorów dla każdej pary przedmiotów zapamiętuje jedynie ilość użytkowników 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:

$$\tilde{r}_{u,i} = \frac{\sum_{j \in R_u} (\text{diff}_{i,j} + r_{u,j}) \cdot |R_{i,j}|}{\sum_{j \in R_u} |R_{i,j}|}$$

$R_{i,j}$  – zbiór użytkowników, którzy ocenili zarówno przedmiot  $i$  jak i  $j$ ,

$\text{diff}_{i,j}$  – średnia różnica ocen pomiędzy przedmiotami  $i$  oraz  $j$  (w przeciwieństwie do wszystkich podobieństw antysymetryczna)

W przeciwieństwie do algorytmu CF który tak naprawdę do predykcji ocen wykorzystuje regresję liniową ( $f(x) = ax + b$ ), SO wykorzystuje prostszą regresję jednoparametrową ( $f(x) = x + b$ ). Użycie tak prostego modelu czyni algorytm o wiele bardziej odpornym na przeuczenie.

### przykład

Na tych samych danych:

$$\text{diff}_{i1,i2} = -\text{diff}_{i2,i1} = -1.5, \text{diff}_{i1,i3} = -\text{diff}_{i3,i1} = 3, \text{diff}_{i2,i3} = -\text{diff}_{i3,i2} = 2.5$$

$$\tilde{r}_{u1,i1} = \frac{(-1.5+3) \cdot 2 + (3+1) \cdot 1}{2+1} = \frac{7}{3}$$

$$\tilde{r}_{u2,i3} = \frac{(-3+2) \cdot 1 + (-2.5+5) \cdot 2}{1+2} = \frac{4}{3}$$

### 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  $m \times m$ ,  $\Sigma$  – 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  $\Sigma$ .

Zastosowanie jednej permutacji do kolumn macierzy  $U$ , wierszy macierzy  $V^\perp$ , oraz wartości własnych w macierzy  $\Sigma$  nie zmienia ich iloczynu.

Po zastosowaniu tych przekształceń dla  $r = \text{rank}(M)$  otrzymujemy rozkład  $M = U'\Sigma'V'^\perp$ , gdzie macierz  $U'$  ma wymiary  $m \times r$ ,  $\Sigma'$  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:  $\|M\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |m_{i,j}|^2}$

**Twierdzenie 2.2.2.1.** (Eckhart-Young 1936) Dla dowolnego  $r < \text{rank}(M)$  macierz  $\tilde{M} = U\Sigma_r V^\perp$ , (gdzie  $\Sigma_r$ , to macierz  $\Sigma$  z pozostawionymi jedynie  $r$  wartościami własnymi o największych wartościach bezwzględnych) jest najlepszym przybliżeniem macierzy  $M$  pod względem normy Frobeniusa, wśród macierzy o rzędzie  $\leq r$ .

Zastosowanie rozkładu SVD i Twierdzenia Eckharta-Younga pozwala na stratną kompresję macierzy.

### Model SVD

Algorytmy rekomendacyjne z grupy SVD mają na celu wygenerowaniu możliwie najlepszych macierzy  $U'$  i  $V'^\perp$  dla zadanego rzędu  $r$  (od teraz oznaczanego jako  $f$  – factor) 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  $i$

$\mu$  – średnia wszystkich ocen wystawionych wszystkim przedmiotom

$K = \{(u, i) \mid \text{użytkownik } u \text{ wystawił ocenę przedmiotowi } i\}$

$\lambda_*$  – różne stałe użyte do regularyzacji (ustawiane w celu optymalizacji pożądanych miar jakości predykcji)

$r_{u,i}$  – prawdziwa ocena wystawiona przedmiotowi  $i$  przez użytkownika  $u$

$\tilde{r}_{u,i}$  – przewidziana przez system ocena przypisana użytkownikowi  $u$  i przedmiotowi  $i$

$\alpha$  – prędkość uczenia (zazwyczaj mała stała rzędu 0.01)



$$\tilde{r}_{u,i} = \mu + b_u + b_i + p_u \cdot q_i$$

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  $f = \text{rank}(M)$  można znaleźć optymalną wartość dla wektorów  $p$  i  $q$  będących wierszami macierzy  $U$  i  $V$  (macierz  $\Sigma$  wmnóżona 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  $\text{rank}(M)$  uniemożliwia modelowi przeuczenie się znanych ocen i powodując uproszczenie modelu ustala wartości  $\tilde{r}_{u,i}$  na najmniej komplikujące model. Ustawienie stałej  $\Lambda$  na dodatnią 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  $\mu, b_u, b_i$  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  $(u, i) \in K$  {
     $err = \tilde{r}_{u,i} - r_{u,i}$ 
     $x += \alpha \cdot (err \cdot \frac{\partial err}{\partial x} - \lambda_x \cdot x)$ 
}
```

Gdzie uaktualnienie  $x$  oznacza zmianę każdej zmiennej zawartej we wzorze na błąd ( $err$ ) – dotyczy to również dodatkowych zmiennych dodanych przez rozszerzenia podstawowego algorytmu.

Warto wspomnieć 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 używanym rozszerzeniem modelu SVD jest algorytm SVD++ (opisany w pracy [2]), 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:

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

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 znajdują się poprzez optymalizację modelu w procesie iteracyjnym opisanym pseudokodem:

```

for  $I$  iterations{
  foreach  $(u, i) \in K$ {
     $py = p_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j$ 
     $\tilde{r}_{u,i} = \mu + b_u + b_i + q_i \cdot py$ 
     $err = \tilde{r}_{u,i} - r_{u,i}$ 
     $b_u = b_u + \alpha \cdot (err - \lambda_b \cdot b_u)$ 
     $b_i = b_i + \alpha \cdot (err - \lambda_{b_2} \cdot b_i)$ 
     $p_u = p_u + \alpha \cdot (err \cdot q_i - \lambda_p \cdot p_u)$ 
     $q_i = q_i + \alpha \cdot (err \cdot py - \lambda_q \cdot q_i)$ 
    foreach  $j \in N(u)$ {
       $y_j = y_j + \alpha \cdot (err \cdot |N(u)|^{-\frac{1}{2}} \cdot q_i - \lambda_y \cdot y_j)$ 
    }
  }
}

```

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

stałe  $\lambda_*$  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 różnicować i będą redundantne)

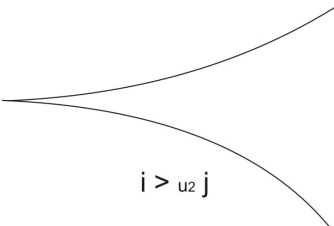
Rozszerzony algorytm jest częściej używany od czystego SVD nawet przy danych wyłącznie explicite, jednak bez dodatkowych informacji implicite nie uzyskuje lepszych wyników.

## 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 nieznanymi model BPR (Bayesian personalized ranking, opisany w pracy [3]) stara się oszacować dla danego użytkownika i każdej pary przedmiotów prawdopodobieństwo, że preferuje on pierwszy z nich. Aby uzyskać początkowe dane używane do filtrowania kolaboratywnego pomiędzy użytkownikami przyjmujemy, że te już użyte są bardziej preferowane od nieznanymi i dla każdego wiersza macierzy tworzymy nową macierz kwadratową.

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



	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  $\Theta$

jest modelem wygenerowanym przez algorytm. Podobnie jak w przypadku modelu SVD  $\Theta$  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:

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

gdzie  $D_K = \{(u, i, j) : i \in N(u) \& j \notin N(u)\}$ ,  $\tilde{s}_{u,i,j} = \tilde{r}_{u,i} - \tilde{r}_{u,j}$ ,  $\|\Theta\|^2 = (b_i^2 + \|p_u\|^2 + \|q_i\|^2)$ ,  $\sigma(x) = \frac{1}{1+e^{-x}}$

Zaś przewidywana ocena przedmiotu jest równa  $\tilde{r}_{u,i} = b_i + p_u \cdot q_i$ .

Warto wspomnieć tu, że uzyskana w ten sposób wartość nie ma wiele wspólnego z używaną skalą ocen (w przypadku informacji wyłącznie implícite nie występującą) i jest używana wyłącznie do ustawienia przedmiotów w odpowiedniej kolejności dla każdego użytkownika (dlatego też używane w SVD wartości  $\mu$  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  $D_K$ 
   $\tilde{s} = b_i - b_j + p_u \cdot (q_i - q_j)$ 
   $err = \frac{e^{-\tilde{s}}}{1+e^{-\tilde{s}}}$ 
   $b_i = b_i + \alpha \cdot (err - \lambda_b \cdot b_i)$ 
   $b_j = b_j + \alpha \cdot (-err - \lambda_{b_2} \cdot b_j)$ 
   $p_u = p_u + \alpha \cdot (err \cdot (q_i - q_j) - \lambda_p \cdot p_u)$ 
   $q_i = q_i + \alpha \cdot (err \cdot p_u - \lambda_q \cdot q_i)$ 
   $q_j = q_j + \alpha \cdot (-err \cdot p_u - \lambda_q \cdot q_j)$ 
}
```

W tym algorytmie trójki  $(u, i, j) \in D_K$  są wybierane losowo, gdyż jest ich na ogół o wiele więcej 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 wywnioskować 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 niepersonalizowanego, a jednocześnie daje lepsze wyniki (choć dzieli też większość jego wad). 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

Dane przypisane do użytkowników i przedmiotów rzadko są wystarczające, aby zbudować dobry system oparty 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). Systemy używające zarówno filtrowania kolaboratywnego i opisów a czasem również danych demograficznych (proponowanie produktów/obiektów dostępnych w okolicy użytkownika lub preferencje regionalne) zwane są hybrydowymi. Łączenie kilku algorytmów (lub typów informacji) dokonywane jest standardowo na jeden z siedmiu sposobów:

- Ważone – wyniki rekomendacji lub przewidywania ocen są średnią ważoną z wyników podsystemów
- Przełączane – wybierany jest podsystem najlepiej pasujący do szczególnego przypadku w danych
- Mieszane – wyniki różnych rekomendacji są zwracane razem
- Połączenie cech – cechy z różnych źródeł wiedzy są wykorzystywane razem w jednym algorytmie
- Uzupełnienie cech – dodatkowy system generuje dane, które wzbogacają wejście do głównego algorytmu
- Kaskadowy – dodatkowy system rozstrzyga sytuacje, z którymi główny system nie może sobie dobrze poradzić
- Meta-poziom – jeden system generuje model, który stanowi wejście drugiego systemu

### 2.4.1. gSVD++

Jednym z podstawowych hybrydowych systemów rekomendacyjnych (typu połączenie cech) jest algorytm gSVD++ (generalized singular value decomposition, opisany w pracy [4]), będący wzbogaceniem podstawowego modelu SVD o informacje na temat kategorii do których

należą przedmioty. Algorytm zakłada, stałą niewielką liczbę kategorii, do której zakwalifikowane są przedmioty, przy czym jeden przedmiot może być zakwalifikowany do wielu kategorii na raz. Na przykład dla danych w których przedmioty są filmami jeden film może być jednocześnie sensacyjny, przygodowy oraz science-fiction. Dokładniej zakładane, jest że do każdego przedmiotu dołączona jest informacja o zbiorze kategorii przedmiot należy, a do których nie. Ocena przewidziana dla użytkownika i przedmiotu opisana jest wzorem

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

Gdzie  $G(i)$  oznacza zbiór kategorii do których należy przedmiot, zaś  $x_g \in \mathbb{R}^f$  dodatkowym wektorem przypisanym kategorii (w przypadku, gdy zbiór kategorii jest pusty wartość  $|G(i)|^{-1} \sum_{g \in G(i)} x_g$  traktowana jest jako 0). Zmienne występujące we wzorach na przewidziane

oceny optymalizowane są przez standardowy algorytm opisany pseudokodem:

```

for  $I$  iterations{
  foreach  $(u, i) \in K$ {
     $py = p_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j$ 
     $qx = q_i + |G(i)|^{-1} \cdot \sum_{g \in G(i)} x_g$ 
     $\tilde{r}_{u,i} = \mu + b_u + b_i + qx \cdot py$ 
     $err = \tilde{r}_{u,i} - r_{u,i}$ 
     $b_u = b_u + \alpha \cdot (err - \lambda_b \cdot b_u)$ 
     $b_i = b_i + \alpha \cdot (err - \lambda_{b_2} \cdot b_i)$ 
     $p_u = p_u + \alpha \cdot (err \cdot qx - \lambda_p \cdot p_u)$ 
     $q_i = q_i + \alpha \cdot (err \cdot py - \lambda_q \cdot q_i)$ 
    foreach  $j \in N(u)$ {
       $y_j = y_j + \alpha \cdot (err \cdot |N(u)|^{-\frac{1}{2}} \cdot qx - \lambda_y \cdot y_j)$ 
    }
    foreach  $g \in G(i)$ {
       $x_g = x_g + \alpha \cdot (err \cdot |G(i)|^{-1} \cdot py - \lambda_x \cdot x_g)$ 
    }
  }
}

```



## Rozdział 3

# Ulepszenia

Aby stworzyć system odnoszący porządane rezultaty w rzeczywistych zastosowaniach warto wybrać odpowiedni do konkretnego celu algorytm. Znane algorytmy pozwalają uzyskać w miarę dobre predykcje lub listy rekomendacji jednak ich rezultaty wciąż odbiegają mocno od optymalnych, plasując się często niewiele wyżej od prostych technik losowych i niespersonalizowanych. Dodatkowo niektóre algorytmy nie radzą sobie dobrze w przypadku niektórych rozkładów danych, dodatkowo ograniczając możliwość wyboru odpowiedniego. Jedną z możliwości pokonania tych niedoskonałości jest wymyślenie zupełnie nowego, lepszego algorytmu, to jednak jest wyjątkowo trudne. Inną opcją jest wzięcie jednego ze znanych algorytmów i dokonanie na nim modyfikacji, pozwalającej uzyskać lepsze rezultaty lub użycie niektórych jego fragmentów i stworzenie zupełnie nowego systemu.

### 3.1. Standardowe ulepszenia algorytmów

Najprostszym sposobem stworzenia lepszego systemu rekomendacyjnego jest zmodyfikowanie istniejącego już systemu.

Ulepszenie można uzyskać poprzez zaaplikowanie modyfikacji polegającej na:

- zaadaptowaniu znanego algorytmu do innego typu danych (jak w przypadku zmiany  $SVD \rightarrow BPR$ )
- poprzedzeniu algorytmu preprocessingiem modyfikującym otrzymane dane na pasujące do wejścia normalnego systemu
- użycie algorytmu bez zmian a jedynie z inną interpretacją wyników
- dodanie dodatkowego elementu do modelu (jak  $SVD \rightarrow SVD++ \rightarrow gSVD++$ ) – zazwyczaj wynikające z posiadania dodatkowych danych niewykorzystywanych w bazowym systemie
- porzucenie części algorytmu, która nie polepsza rezultatów, jednocześnie upraszczając model
- zamianę części na inną o tej samej funkcji (inna miara podobieństwa czy funkcja celu)

Niektóre systemy jako wejście oprócz danych o przedmiotach i użytkownikach przyjmują dodatkowe parametry zmieniające ich pracę i o ile wprowadzenie innych stałych regularyzacyjnych w SVD czy wybór pomiędzy podobieństwami użytkowników i przedmiotów w CF nie może być uznany jako istotna zmiana algorytmu, o tyle wprowadzenie własnej funkcji podobieństwa lub inne schodzenie po gradiencie (np. zmniejszająca się szybkość uczenia) pozwala uzyskać istotnie inny algorytm.

Porzucenie części algorytmu o ile raczej nie usprawni wyników samo w sobie, o tyle pozwala

użyć go na większych danych, lub wykorzystać zyskany czas w inny sposób (np. stosując dodatkowe iteracje).

### 3.1.1. ulepszenia CF

Największą niedoskonałością algorytmu Collaborative Filtering (poza złożonością czasową) jest słaba możliwość predykcji ocen w przypadku gdy użytkownik nie ma sąsiadów którzy użyli danego przedmiotu, takich sąsiadów jest mało, lub też mimo podobieństwa w wybranej mierze użytkownik nie jest dobrym wyznacznikiem oceny.

W pracy [7] zaproponowano szereg ulepszeń klasycznego algorytmu CF mających na celu zminimalizowanie tych wad.

#### alternatywne podobieństwo

Najbardziej standardową zmianą jaką można dokonać w algorytmie CF jest zdefiniowanie własnej funkcji podobieństwa.

Największą wadą podobieństw korelacji Paersona oraz kosinusowego jest niedokładność lub wręcz brak możliwości określenia podobieństwa pomiędzy użytkownikami w przypadku małej ilości wspólnych przedmiotów użytych lub spłaszczenia ocen. Aby pokonać te ograniczenia zaproponowana została nowa miara podobieństwa pomiędzy dwoma użytkownikami:

$$S_{u,v} = \frac{\sum_{i \in R_{u,v}} f(r_{u,i} - r_{v,i})}{|R_{u,v}|}$$

$$f(r_{u,i}, r_{v,i}) = \begin{cases} \frac{1}{|r_{u,i} - r_{v,i}| + 1} & \text{dla } (r_{u,i} > \bar{r}) \wedge (r_{v,i} > \bar{r}) \\ \frac{1}{|r_{u,i} - r_{v,i}| + 1} & \text{dla } (r_{u,i} \leq \bar{r}) \wedge (r_{v,i} \leq \bar{r}) \\ \frac{0.5}{|r_{u,i} - r_{v,i}| + 1} & \text{w p.p.} \end{cases}$$

$\bar{r}$  – średnia ocena dostępna w systemie (neutralna)

W ten sposób otrzymane podobieństwo przyjmuje wartości w przedziale  $[0, 1]$  i daje w miarę dobre informacje o podobieństwie dwóch użytkowników już dla małych ilości wspólnie użytych przedmiotów oraz spłaszczonych rankingach, w pozostałych przypadkach przegrywa jednak z korelacją Paersona.

Biorąc pod uwagę te cechy proponowanym sposobem użycia alternatywnego podobieństwa jest używanie korelacji Paersona w normalnych przypadkach, zaś tego podobieństwa gdy ilość wspólnie ocenionych przedmiotów jest niższa od określonej wartości lub dla zbyt małego zróżnicowania ocen jednego z użytkowników dla których oceniane jest podobieństwo.

#### współczynnik podobieństwa

Głównym zagrożeniem wynikającym z obliczania standardowych funkcji podobieństwa dla małych ilości wspólnie ocenionych przedmiotów jest możliwość zwrócenia dużego podobieństwa w sytuacji gdy tak naprawdę użytkownicy nie są wcale podobni (w przypadku PC i kosinusowego dla jednego wspólnego przedmiotu wartość będzie zawsze równa 1). W takim przypadku przy predykcji oceny wartości wystawione przez tych sąsiadów mają duże znaczenie, zaburzając działanie systemu. Sposobem poradzenia sobie z takimi przypadkami niewymagającym trudnych do zdefiniowania funkcji podobieństwa jest zmiana wag wykorzystywanych do wyliczania ocen na zależne od ilości wspólnie ocenionych przedmiotów. Zdefiniowany zostaje współczynnik podobieństwa (similarity factor – SF):



$$SF_{u,v} = \begin{cases} |R_{u,v}| \cdot S_{u,v} & \text{dla podobieństwa symetrycznego (PC lub kosinusowe)} \\ |R_{u,v}| \cdot (S_{u,v} - 0.1) & \text{dla podobieństwa alternatywnego} \end{cases}$$

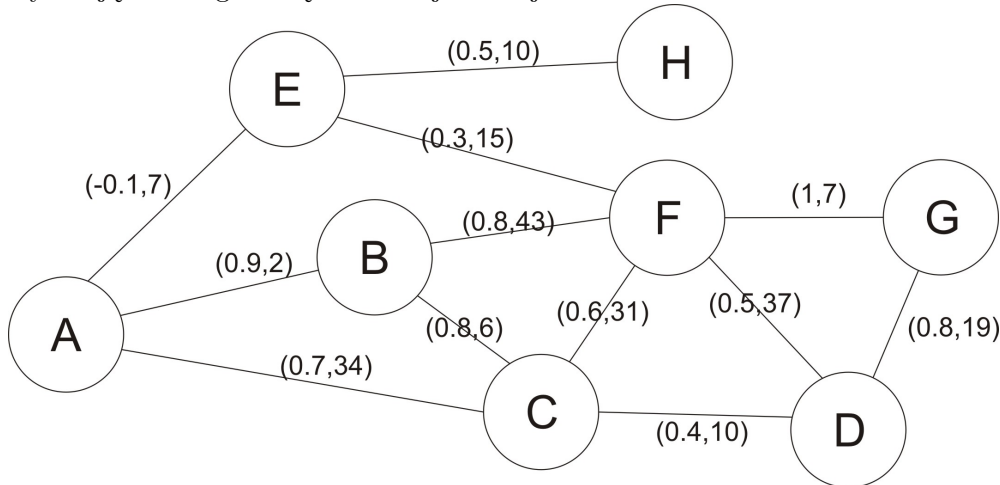
Zaś ocena wystawiona przez algorytm zdefiniowana jest wzorem:

$$r'_{u,i} = \bar{r}_u + \frac{\sum_{v \in N_u} (r_{v,i} - \bar{r}_v) SF_{u,v}}{\sum_{v \in N_u} SF_{u,v}}$$

Zbiór sąsiadów jest jednak nadal wyliczany na podstawie normalnych podobieństw między użytkownikami.

### dalsze podobieństwo

W przypadku gdy pewien użytkownik nie posiada sąsiadów, którzy użyli danego przedmiotu można odnieść się do dalszych sąsiadów – użytkowników bliskich według podobieństwa przechodniego (sąsiedzi sąsiadów), które już dla ścieżek pomiędzy użytkownikami długości 2 powinny znacznie poprawić pokrycie przypadków występujących w danych. Dalsze podobieństwo można odczytać z grafu w którym wierzchołkami są użytkownicy, zaś krawędziami wartości podobieństwa między nimi z odciętymi krawędziami ujemnymi. W ten sposób podobieństwo można zdefiniować jako po pierwsze odległość w grafie, a w dalszej kolejności podobieństwo wynikające z wag krawędzi na najkrótszej ścieżce.



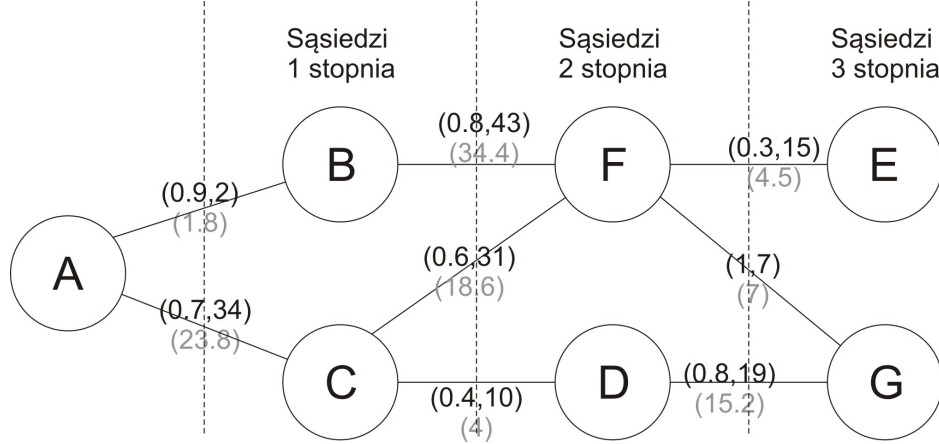
### połączenie ulepszeń

Łącząc przedstawione usprawnienia algorytmu CF w pracy [7] zaproponowany został system CF-ADV.

Dla zadanego użytkownika  $u$  oraz przedmiotu  $i$  jeżeli według podobieństwa zdefiniowanego wcześniej (korelacja Paersona w przypadku zwykłym, alternatywne podobieństwo w skrajnych) użytkownik ma sąsiadów, którzy ocenili ten przedmiot, to zwracana jest wartość otrzymana z ich ocen ważona współczynnikami podobieństwa.

W przeciwnym przypadku sąsiedzi wyznaczani są poprzez przeszukiwanie grafu ze źródłem w użytkowniku  $u$  – użytkownicy ustawiani są w kolejności zgodnej z przeszukiwaniem BFS (najpierw ci z najkrótszą odległością od źródła), aż do zadanej odległości maksymalnej  $H$ . Sąsiedzi użytkownika  $u$  na podstawie których wystawiana jest ocena dla przedmiotu  $i$  wybierani są jako początek tej listy ograniczonej do użytkowników, dla których znana jest ocena

tego przedmiotu.



Aby móc zastosować średnią ważoną pozostaje jeszcze zdefiniować wagi – w tym przypadku dla użytkownika  $v$  o najkrótszej ścieżce od źródła  $u$  w grafie użytkowników długości  $m$  na podstawie średniej z wag jego bezpośrednich poprzedników na wszystkich ścieżkach tej długości:

$$SF_{u,v} = \frac{\sum_{w \in P_{u,v}} (SF_{u,w} \cdot SF_{w,v})}{\sum_{w \in P_{u,v}} SF_{u,w}}$$

( $P_{u,v}$  – zbiór wierzchołków oddalonych od  $u$  o  $m - 1$ , połączonych krawędzią z  $v$ )

W przypadku z rysunku wartości współczynników wyniosą odpowiednio:

$$\begin{aligned} SF_{A,D} &= \frac{SF_{A,C} \cdot SF_{C,D}}{SF_{A,C}} = 4 \\ SF_{A,F} &= \frac{SF_{A,B} \cdot SF_{B,F} + SF_{A,C} \cdot SF_{C,F}}{SF_{A,B} + SF_{A,C}} = \frac{61.92 + 442.68}{25.6} = 19.71 \\ SF_{A,G} &= \frac{SF_{A,D} \cdot SF_{D,G} + SF_{A,F} \cdot SF_{F,G}}{SF_{A,D} + SF_{A,F}} = \frac{60.8 + 67.97}{23.71} = 5.43 \\ SF_{A,E} &= \frac{SF_{A,F} \cdot SF_{F,E}}{SF_{A,F}} = 4.5 \end{aligned}$$

Widać tutaj, że nawet jeśli bezpośrednie podobieństwo między użytkownikami jest ujemne, to mogą być w relacji dalszego podobieństwa. Dlatego też ważne jest aby najpierw patrzeć na długość ścieżek, a na wartości współczynników podobieństwa dopiero w drugiej kolejności.

### 3.1.2. Wykorzystanie metadanych jako uzupełnienie danych implicite

Przykład ulepszenia które można wprowadzić posiadając dodatkowe dane przedstawiony został w pracy [5], w której zajęto się poprawą algorytmu BPR.

W metodzie spersonalizowanego rankingu Bayesa wykorzystywane jest założenie, że przedmiot użyty ma dla użytkownika większą wartość od takiego, którego nie użył i na podstawie tych dwóch klas znajdowane są wartości odpowiednich wektorów z rozkładu SVD jak najbardziej uwidaczniające tą różnicę. Tak wielkie spłaszczenie wejścia (2 klasy) jest spowodowane tym, że dostępne są dane wyłącznie implicite, możliwe jest jednak uzyskanie większej różnorodności dzięki użyciu dodatkowych danych o przedmiotach nie wymagając równocześnie bardziej kosztownego zbierania informacji od użytkowników. W pracy [5] zaproponowane zostały dwa algorytmy obudowujące standardowy algorytm BPR poprzez wprowadzenie rozróżnienia pomiędzy użytymi przedmiotami uzyskanemu dzięki danym o należności przedmiotów do kategorii (takimi jak w algorytmie gSVD++).

## MABPR

Pierwszy algorytm zakłada użycie standardowego BPR z rozszerzonym zbiorem  $D_K$ , z którego próbkowane są trójki (użytkownik, przedmiot preferowany, przedmiot mniej lubiany). Aby wprowadzić rozróżnienie na dwóch użytych przedmiotach stosowane są dane o tym do jakich kategorii należą, a następnie sprawdzić, który zestaw kategorii powinien bardziej przypaść do gustu użytkownikowi. Pozostaje jednak problem ustalenia tego które kategorie są przez użytkownika lubiane. Ponieważ nie posiadamy o użytkowniku danych ponad to jakie przedmioty zostały przez niego użyte, to właśnie z tych danych należy uzyskać pożądane informacje. Przeprowadzony zostaje preprocessing mający na celu dla każdego użytkownika  $u$  oraz kategorii  $g$  uzyskać wartość rzeczywistą  $w_{u,g}$ , obrazującą jak bardzo dana kategoria jest przez użytkownika lubiana. Zgodnie z podstawowym założeniem o większej wartości przedmiotu użytego wagi  $w$  powinny zostać dobrane tak, aby dla użytego przedmiotu  $i$  oraz nieużytego  $j$  zmaksymalizować wartość

$$\tilde{s}_{u,i,j} = \tilde{r}_{u,i} - \tilde{r}_{u,j} = \sum_{g \in G} w_{u,g} a_{i,g} - \sum_{g \in G} w_{u,g} a_{j,g} = \sum_{g \in G} w_{u,g} (a_{i,g} - a_{j,g})$$

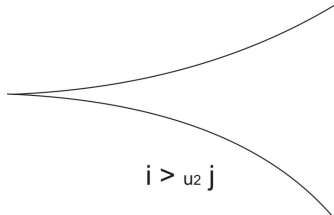
gdzie  $G$  to zbiór kategorii, zaś  $a_{i,g}$  to binarna wartość przynależności przedmiotu  $i$  do kategorii  $g$

Jak w pozostałych takich przypadkach wartości te są otrzymane poprzez zaaplikowanie algorytmu iteracyjnego zejścia po gradiencie dla tej funkcji:

```
LearnBPR{
  for  $I$  iterations{
    wylosuj  $(u, i, j)$  z  $D_K$ 
     $\tilde{s} = w_u \cdot (a_i - a_j)$ 
     $err = \frac{e^{-\tilde{s}}}{1 + e^{-\tilde{s}}}$ 
     $w_u = w_u + \alpha \cdot (err \cdot (a_i - a_j) - \lambda_w \cdot w_u)$ 
  }
}
```

gdzie  $w_u, a_i, a_j$  to wektory długości  $|G|$  (wartości  $w_{u,g}$  oraz binarne przynależności do kategorii  $a_{i,g}$ )

Dzięki tak wyliczonym wagom można utworzyć tabelkę porównań przedmiotów dla użytkownika z mniejszą ilością wartości nieznanych

	i1	i2	i3	i4	i5			j1	j2	j3	j4	j5
u1	+	?	?	?	+		i1	X	-	-	?	-
u2	?	+	+	?	+		i2	+	X	$\delta_{2,3}$	+	$\delta_{2,5}$
u3	+	+	+	?	?		i3	+	$\delta_{3,2}$	X	+	$\delta_{3,5}$
u4	?	?	?	?	+		i4	?	-	-	X	-
							i5	+	$\delta_{5,2}$	$\delta_{5,3}$	+	X

$$\delta_{i,j} = \begin{cases} + & \text{jeśli } \varphi(u, i) > \varphi(u, j) \\ - & \text{jeśli } \varphi(u, i) < \varphi(u, j) \\ ? & \text{w p.p} \end{cases}$$

$$\varphi(u, i) = \frac{1}{|G(i)|} \sum_{g \in G(i)} w_{u,g}$$

Po przeprowadzeniu takiego preprocessingu i ustaleniu nowego zbioru  $D_K$  pseudo oceny ustalane są już przy pomocy niezmiennego BPR.

## MABPR gSVD++

Kolejny algorytm również zakłada wyliczenie rozszerzonego zbioru  $D_K$  (w ten sam sposób) przed wykonywaniem właściwej pracy, jednak korzysta z danych o należności przedmiotów

do kategorii dodatkowo i w właściwej swojej części. MABPR gSVD++ łączy w sobie dwa ulepszenia jakimi są przejście z SVD do gSVD++ oraz z BPR do MABPR generując sztuczną ocenę przypisaną użytkownikowi i przedmiotowi przy pomocy wzoru:

$$\tilde{r}_{u,i} = b_i + \left( q_i + |G(i)|^{-1} \sum_{g \in G(i)} x_g \right) \cdot \left( p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

czyli takiego jak w przypadku gSVD++, uproszczonego o nieistotne w BPR przesunięcia. W takim przypadku maksymalizowane wartości różnic dla przedmiotów preferowanego  $i$  i mniej lubianego  $j$  przyjmują postać:

$$\tilde{s}_{u,i,j} = \tilde{r}_{u,i} - \tilde{r}_{u,j} = b_i - b_j + \left( q_i - q_j + |G(i)|^{-1} \sum_{g \in G(i)} x_g - |G(j)|^{-1} \sum_{g \in G(j)} x_g \right) \cdot \left( p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j \right)$$

Po połączeniu wszystkich tych komponentów uzyskuje się system MABPR gSVD++ w którym wartości zmiennych znajdowane są za pomocą iteracyjnego algorytmu:

```

LearnBPR()
for  $I$  iterations{
  wylosuj  $(u, i, j)$  z  $D_K$ 
   $py = p_u + |N(u)|^{-\frac{1}{2}} \cdot \sum_{j \in N(u)} y_j$ 
   $qx_1 = q_i + |G(i)|^{-1} \cdot \sum_{g \in G(i)} x_g$ 
   $qx_2 = q_j + |G(j)|^{-1} \cdot \sum_{g \in G(j)} x_g$ 
   $\tilde{s} = b_i - b_j + (qx_1 - qx_2) \cdot py$ 
   $err = \frac{e^{-\tilde{s}}}{1 + e^{-\tilde{s}}}$ 
   $b_i = b_i + \alpha \cdot (err - \lambda_{b_2} \cdot b_i)$ 
   $b_j = b_j + \alpha \cdot (-err - \lambda_{b_2} \cdot b_j)$ 
   $p_u = p_u + \alpha \cdot (err \cdot (qx_1 - qx_2) - \lambda_p \cdot p_u)$ 
   $q_i = q_i + \alpha \cdot (err \cdot py - \lambda_{q_1} \cdot q_i)$ 
   $q_j = q_j + \alpha \cdot (-err \cdot py - \lambda_{q_2} \cdot q_j)$ 
  foreach  $k \in N(u)$ {
     $y_k = y_k + \alpha \cdot (err \cdot |N(u)|^{-\frac{1}{2}} \cdot (qx_1 - qx_2) - \lambda_y \cdot y_k)$ 
  }
  foreach  $g \in G(i)$ {
     $x_g = x_g + \alpha \cdot (err \cdot |G(i)|^{-1} \cdot py - \lambda_{x_1} \cdot x_g)$ 
  }
  foreach  $g \in G(j)$ {
     $x_g = x_g + \alpha \cdot (-err \cdot |G(j)|^{-1} \cdot py - \lambda_{x_2} \cdot x_g)$ 
  }
}

```

### 3.2. Nowe algorytmy zbudowane na podstawie starych

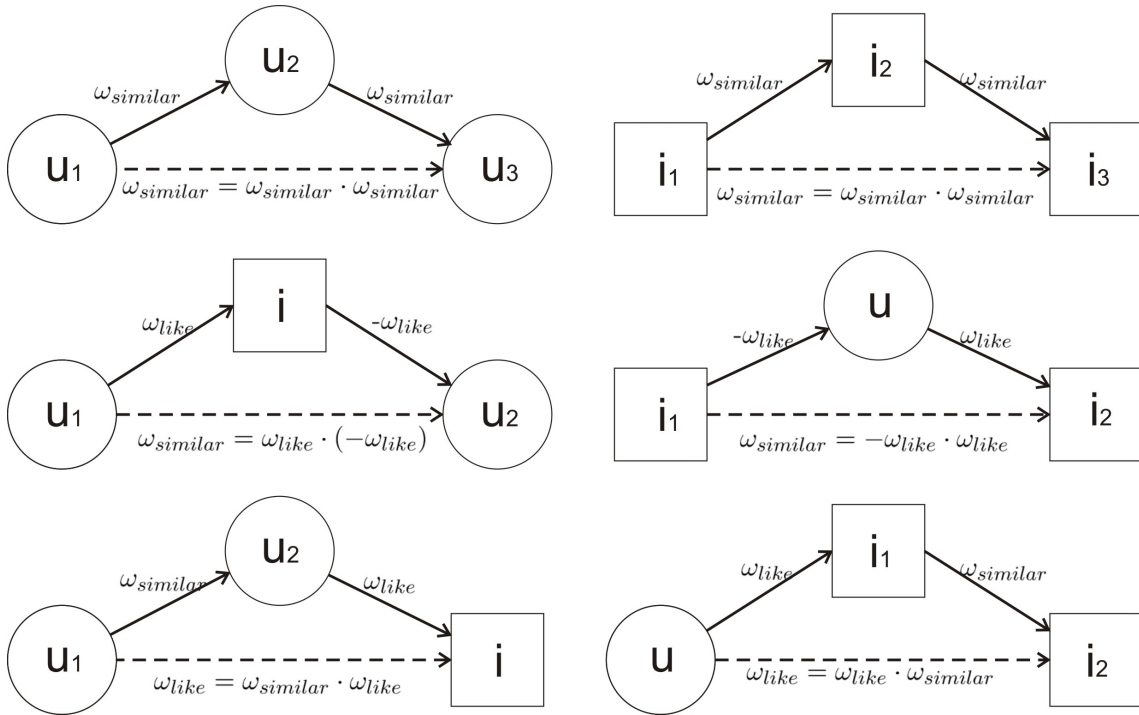
Inną metodą skonstruowania ulepszenia jest przeanalizowanie głównych cech pewnego algorytmu i stworzenie nowego pozornie zupełnie innego algorytmu, który jednak również zawiera te cechy a jednocześnie lepiej radzi sobie z postawionym przed nim celem.

Jednym z takich ulepszeń jest algorytm Complex bazujący na kolaboratywnym filtrowaniu i otrzymany poprzez znaczne uproszczenie algorytmów CF i SO oraz dostosowanie do prostych danych postaci  $\{-1, 0, 1\} = \{\text{negatywna ocena, brak oceny, pozytywna ocena}\}$ .

### 3.2.1. Complex

Standardową reprezentacją danych rozważanych w filtrowaniu 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 oznaczają powiązania między 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-użytkownik. 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 macierzy 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  $\Omega(m^2n)$  dla macierzy o rozmiarach  $m \times n$  badając tylko bardzo krótkie ścieżki, przy dłuższych ten czas ulegałby potęgowaniu (jest to częściowo naprawiane przez podejście jak w pracy [7], nie jest to jednak całkowite rozwiązanie problemu), dlatego aby uzyskać lepsze wyniki w rozsądnym czasie model podobieństwa musi zostać uproszczony. Jeden z modeli wykorzystujących dalsze podobieństwo w praktyce został zaproponowany w [6]. 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:



Ze schematu można wywnioskować następujące reguły:

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

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 krawędziach (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  $I + M + \frac{1}{2}M^2 + \dots = e^M$ , jednak ponieważ ścieżki większych długości są mniej istotne dla prawdziwych wartości (nawet po zniwelowaniu oddziaływania ilościowego) w praktyce lepiej stosować wagi, które bardziej premią krótkie ścieżki. Jeśli model danych nie posiada żadnych krawędzi pomiędzy użytkownikami lub przedmiotami macierz grafu (dwudzielnego) można zapisać w uproszczony sposób:

$$M = \begin{bmatrix} M_{UU} & M_{UI} \\ M_{IU} & M_{II} \end{bmatrix} = \begin{bmatrix} 0 & M_{UI} \\ M_{IU} & 0 \end{bmatrix} = \begin{bmatrix} 0 & jB \\ -jB^\perp & 0 \end{bmatrix} \Rightarrow$$

$$M^{2k} = \begin{bmatrix} (BB^\perp)^k & 0 \\ 0 & (B^\perp B)^k \end{bmatrix}, M^{2k+1} = \begin{bmatrix} 0 & j(BB^\perp)^k B \\ -j(B^\perp B)^k B^\perp & 0 \end{bmatrix}$$

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). Ze względu na drastycznie malejącą użyteczność wiedzy coraz dłuższych ścieżek wagi dla tych ścieżek powinny być na tyle mniejsze od tych dla krótkich. W przypadku wag dobranych do prawdziwych zastosowań suma ogonowa ze wzoru  $\sum_{k=0}^{\infty} a_k (BB^\perp)^k B$  jest zaniedbywalna, na tyle, że można wzór ograniczyć do pierwszych kilku elementów.

Dzięki tym wszystkim uproszczeniom oraz temu, że mnożenie macierzy jako bardzo standardowa operacja jest w większości języków programowania bardzo dobrze zoptymalizowana czasowo algorytm jest o wiele szybszy od pozostałych przedstawionych w tej pracy (na poziomie szybkości algorytmów niespersonalizowanych).

## regularyzacje

Dodatkową metodą używaną do poprawienia osiągnięć systemu jest zastosowanie regularyzacji. W przypadku gdy pewien użytkownik ocenił bardzo dużo przedmiotów lub dany przedmiot został użyty przez większość użytkowników taki węzeł stwarza znaczne większą ilość ścieżek przechodzących przez niego – ma dużo większy wpływ na oceny niż pozostałe wierzchołki. Aby zapobiec sytuacji w których wpływ jednego użytkownika lub przedmiotu na system przeważałby nad pozostałymi oraz rozszerzyć grono istotnych czynników często stosowane jest dzielenie wartości odpowiadających pewnej instancji przez pierwiastek ilości tych wartości (podobnie jak w przypadku wektorów  $y$  w SVD++). W przypadku algorytmu Complex regularyzacja taka sprowadza się do podzielenia wartości w wierszach (lub kolumnach) macierzy przez pierwiastek liczby niezerowych elementów w tym wierszu (tej kolumnie). Dokładniej wartości występujące w macierzy  $M$  w różnych wariantach przyjmują odpowiednio wartości:

	brak reg.	reg. przedmiotów	reg. użytkowników	reg. uż. i przed.
lubi	i	$\frac{i}{\sqrt{N_i(u)}}$	$\frac{i}{\sqrt{N_u(i)}}$	$\frac{i}{\sqrt{N_u(i) \cdot N_i(u)}}$
nie lubi	-i	$-\frac{i}{\sqrt{N_i(u)}}$	$-\frac{i}{\sqrt{N_u(i)}}$	$-\frac{i}{\sqrt{N_u(i) \cdot N_i(u)}}$

Algorytm Complex działa dobrze również w przypadku danych implicite, oczywiście w tym przypadku nie skupia się na znajdowaniu obiektów najlepszych na podstawie pozytywnych i negatywnych ocen, a jedynie na znajdowaniu wszystkich przedmiotów które powinny zostać użyte.





## Rozdział 4

# Ocena i porównanie systemów

Aby dokonać wyboru jednego z algorytmów do prawdziwego systemu lub aby sprawdzić czy modyfikacja systemu rzeczywiście go poprawia należy określić jakąś miarę jakości przewidywanych ocen i rekomendacji względem której będą one porównywane.

Jakość poszczególnych systemów może się znacznie różnić w zależności od specyfiki danych:

- gęstości – dla rzadszych danych ciężiej znaleźć podobnych użytkowników, istotne jest mniej wartości własnych macierz
- rozmiaru – dla dużych danych niektóre algorytmy są zbyt wolne, lub wymagają zbyt dużej pamięci – należy użyć algorytmów szybszych lub przybliżonych
- ilości nowych użytkowników i przedmiotów (z małą liczbą ocen) – niektóre algorytmy są lepiej dostosowane do zjawiska zimnego startu
- różnorodności ocen – jeśli użytkownicy wystawiają takie same oceny wszystkim przedmiotom wiele funkcji podobieństwa przestaje sobie radzić

Dlatego też do wyliczenia tych miar należy użyć prawdziwych danych dla których przewidziany jest system, lub też okrojonej ich części, która zachowa możliwie wiele własności pełnych danych.

### 4.1. Opis danych

Dane użyte do testowania i oceny algorytmów pochodzą z ogólnodostępnego i popularnego wśród prac naukowych zbioru zebranego przez GroupLens ([ML]) nazwanego MovieLens (ML) i dotyczą danych filmowych.

Dokładniej dane składają się z listy ocen wystawionych przez użytkowników filmom w skali 1–5, binarnych danych o przynależności przedmiotów do 19 gatunków filmowych oraz kilku podstawowych informacji o użytkownikach. Dodatkowo uwzględnione są tam nieużywane w opisanych algorytmach dane o datach wydania filmu i wystawienia oceny. W pełnym zbiorze danych występuje 1682 filmów oraz 943 użytkowników, z których każdy wystawił co najmniej 20 ocen. Łącznie wystawionych jest 100 000 ocen, co przy 1 486 126 brakujących ocenach daje poziom rzadkości danych równy 93.7% ( $1 - \frac{\text{ilość ocen}}{\text{ilość wszystkich możliwych}}$ ), dosyć mały przy porównaniu z większością innych baz danych. Oznacza to też, że użytkownicy ocenili średnio po 106 przedmiotów, zaś średnia popularność filmów wyniosła 59. Do danych dołączone jest dodatkowo 5 zbiorów treningowych i 5 testowych stworzonych do badania systemów przy pomocy 5-krotnej cross-walidacji (całe dane zostają podzielone na 5 równych co do wielkości części, zbiór testowy to jedna z tych części, zaś odpowiedni treningowy to reszta danych). W moich ocenach algorytmów korzystam właśnie z tego podziału (wartości odpowiednich

miar dla systemu jest średnią po 5 wartościach otrzymanych przez uczenie modelu na zbiorze treningowym i ocenianiu na podstawie odpowiedniego testowego).

## 4.2. Załączona implementacja algorytmów

Aby przeprowadzić miarodajne porównanie algorytmów zaimplementowałem je oraz skrypt umożliwiający zbadanie ich wyników w języku R (implementacje są dołączone do pracy). Wybór języka dostosowanego do data miningu oraz brak nietrywialnych optymalizacji czasowych pozwolił na bezpośrednie przerobienie podanych w pracy pseudokodów na działającą implementację a więc pozwala na zajrzenie do kodu w przypadku wątpliwości co do dokładnego mechanizmu działania algorytmów. Ze względu na to, że implementacja jest dostosowana do testowania algorytmów na konkretnej, stałej bazie danych może nie działać dobrze na danych innego typu, w szczególności nie jest dostosowana do szybkiej zmiany wyliczonego modelu po wprowadzeniu nowej oceny. W celu stworzenia prawdziwego systemu lepiej skorzystać z implementacji dostępnej w bibliotece [8], w której zawarta jest większość przedstawionych tu algorytmów.

W skład zaimplementowanych i porównywanych algorytmów wchodzi:

- algorytmy niespersonalizowane:
  - ocena jako połowa (średniej oceny użytkownika i średniej oceny przedmiotu)
  - propozycja najpopularniejszych przedmiotów
  - losowe oceny
  - optymalne oceny (skonstruowane na podstawie zbioru testowego tylko do celów porównawczych)
- algorytmy CF z funkcjami podobieństwa:
  - korelacja Paersona (kowariancja)
  - podobieństwo kosinusowe
  - podobieństwo mieszane z pracy [7] (alternatywne gdy zawodzi korelacja Paersona)
  - zaawansowane (dalsze podobieństwo mieszane)

i dodatkowo z wyborem podobieństwa użytkowników lub przedmiotów oraz zastosowania współczynnika podobieństwa lub nie.
- algorytm SO
- algorytmy predykcji ocen przy pomocy rozkładu macierzy:
  - SVD
  - SVD++
  - gSVD++
- algorytmy predykcji list rekomendacyjnych przy pomocy rozkładu macierzy:
  - BPR
  - MABPR
  - MABPR gSVD++
- algorytm Complex na wejściu explicite i implicate + możliwe regularyzacje

Jako, że dla każdej dostępnej opcji w algorytmach grupy SVD i BPR (ilość iteracji, długość wektorów, szybkość uczenia, stałe regularyzacyjne) ilość możliwości jest zbyt duża do przetestowania każdej sensownej kombinacji dobrałem ilość iteracji na 40 (w przypadku BPR

40·(rozmiar zbioru treningowego)) aby czas działania algorytmów był podobny do tych z grupy CF, długość wektorów na 5 (dosyć mała w porównaniu z używanymi w pracach do których się odnoszę, jednak z eksperymentów wynikało, że jej zwiększanie w testowanym przypadku pogarsza wyniki). Pozostałe opcje dobrane zostały w sposób eksperymentalny (ze względu na zbyt duży czas działania algorytmów automatyczne metody ustalania optymalnych parametrów nie mogły zostać poprawnie zastosowane), tak aby w przypadku algorytmów predykcji ocen minimalizować błędy, zaś w przypadku algorytmów z grupy *BPR*, aby maksymalizować precyzję krótkich list rankingowych (szczegóły dotyczące ocen systemów znajdują się w następnych podrozdziałach). Wartości szybkości uczenia zostały ustalone na 0.005 w przypadku algorytmów *SVD*, oraz 0.02 dla algorytmów z grupy *BPR*. Stałe regularyzacyjne używane w porównaniach przyjmują natomiast wartości:

dla algorytmów *SVD*:

$$\lambda_b = \lambda_{b_2} = 0.01, \lambda_p = \lambda_q = \lambda_y = 0.3, \lambda_x = 0.15$$

dla algorytmów *BPR*:

$$\lambda_w = 0.075, \lambda_{b_2} = 0.005, \lambda_p = \lambda_{q_1} = 0.025, \lambda_{q_2} = 0.0025, \lambda_y = \lambda_{x_1} = \lambda_{x_2} = 0.006$$

ze względu jednak na występujący w algorytmach czynnik losowy oraz ograniczoną przestrzeń przeszukiwania mogą one jednak mocno odbiegać od optymalnych.

W algorytmach z grupy CF jako sąsiedzi uznawani są tacy inni użytkownicy (inne przedmioty), dla których funkcja podobieństwa wynosi więcej niż 0, a w przypadku dużej ich ilości do wyliczania oceny używanych jest 30 z nich, dla których to podobieństwo jest największe. Pojęcie spłaszczonego oceniania przez użytkownika używane w algorytmach z pracy [7] do określenia, kiedy w mieszanej funkcji podobieństwa użyć korelacji Paersona, a kiedy alternatywnego podobieństwa określiłem przy pomocy wariancji tych ocen. Podobieństwo alternatywne używane jest albo kiedy ilość wspólnych ocen jest mniejsza niż 5, albo wariancja ocen jednego z użytkowników jest mniejsza niż 0.15.

Z ważniejszych uwag dotyczących implementacji algorytmów niezawartych w pseudokodach:

- przy używaniu algorytmów z grupy *SVD* (schodzenie po gradiencie) należy rozpocząć od wektorów z wartościami wylosowanymi. W przypadku wartości odpowiednich współrzędnych liniowo powiązanych we wszystkich wektorach nie ma możliwości pozbycia się tej zależności, co skutkuje efektywnie zmniejszeniem długości wektorów bez uzyskania korzyści czasowych.
- należy używać małych stałych uczenia (rzędu 0.01), gdyż przy dużych wartościach w wektorach mogą w trakcie wykonywania algorytmu iteracyjnego rozbiegać zamiast stabilizować się, skutkując skrajnie wielkimi wartościami.
- odpowiednio dobrane, dodatnie stałe regularyzacyjne pozwalają nie tylko uzyskiwać lepsze wyniki, ale również częściowo uodporniają algorytm na problem z poprzedniego podpunktu pozwalając użyć większych szybkości uczenia.
- w przypadku algorytmów CF można wyliczyć i zapamiętać macierz podobieństwa pomiędzy użytkownikami (lub przedmiotami) i w ten sposób zmniejszyć złożoność czasową zwiększając jednocześnie złożoność pamięciową.
- w wielu wzorach pojawiają się przypadki gdy pewna wartość jest wyliczana jako  $\frac{0}{0}$ , należy to zazwyczaj traktować jako 0.

### 4.3. Ewaluacja predykcji ocen

Jednym z celów do których możemy chcieć użyć algorytmu jest przewidzenie jaką ocenę powinien wystawić użytkownik pewnemu filmowi, którego jeszcze nie obejrzał. W tym przypadku stosunkowo łatwo jest dokonać oceny systemu, skoro zbiorem testowym jest lista niektórych takich właśnie ocen. Wystarczy policzyć jak bardzo oceny przewidziane przez algorytm różnią się od tych jakie zostały wystawione w rzeczywistości. W tym celu najczęściej używane są dwie miary – średni błąd bezwzględny (mean average error – MAE) i błąd średniokwadratowy (mean square error – MSE), o następujących definicjach:

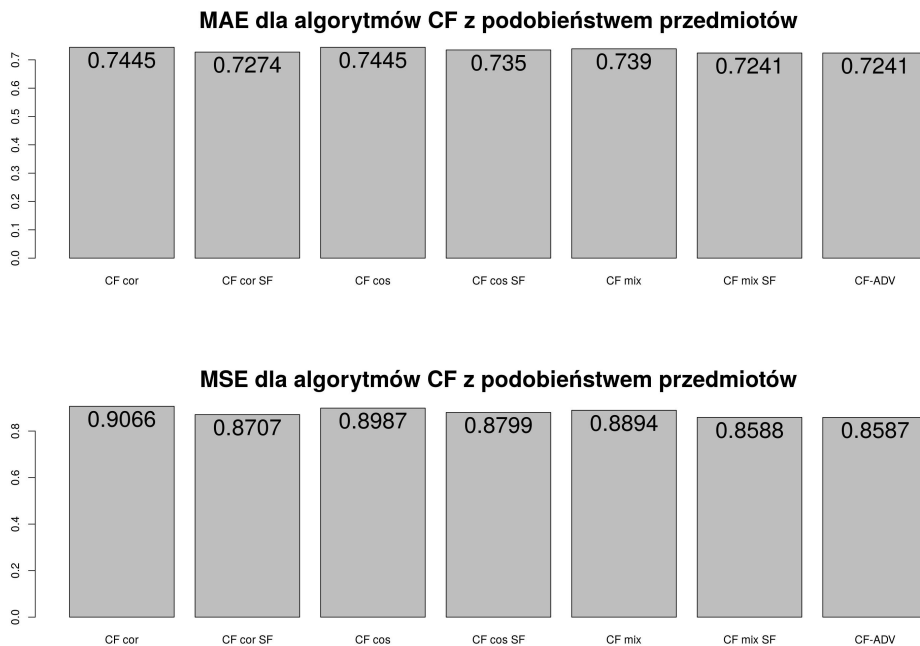
$$MAE = \frac{1}{|T|} \sum_{(u,i) \in T} |\tilde{r}_{u,i} - r_{u,i}| \quad \quad \quad MSE = \frac{1}{|T|} \sum_{(u,i) \in T} (\tilde{r}_{u,i} - r_{u,i})^2$$

gdzie  $T$  to zbiór par (użytkownik, przedmiot) ze zbioru testowego,  $r_{u,i}$  to prawdziwe oceny z tego zbioru, zaś  $\tilde{r}_{u,i}$  oceny przewidziane przez system

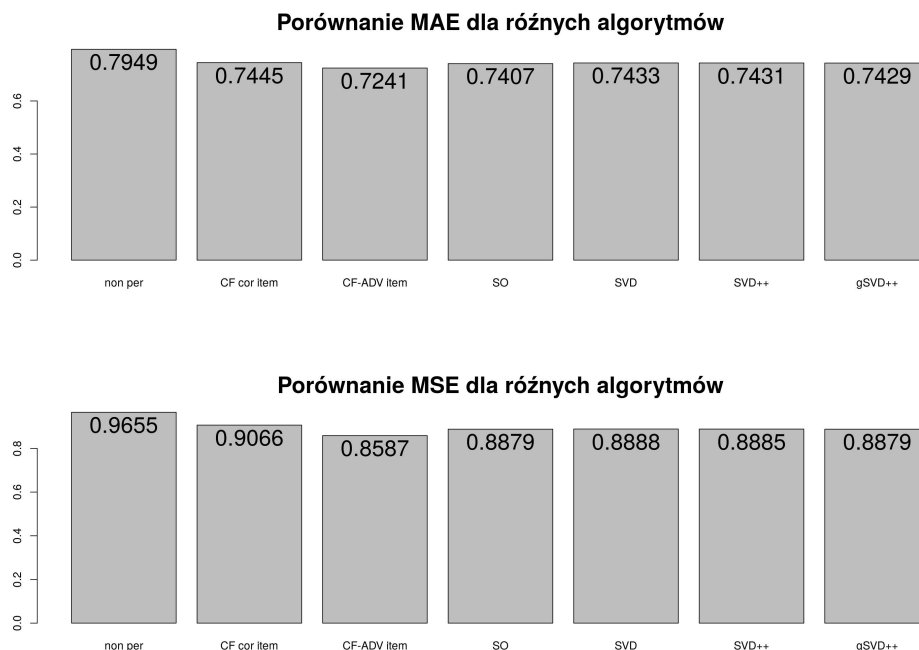
Przed wyliczeniem miar jakości warto obciąć wartości ocen do dostępnego przedziału (w przypadku danych ML [1,5]). Większość algorytmów tu przedstawionych może zwracać wartości spoza przedziału, przy zwracaniu ocen lepiej zwrócić wartość z krańca przedziału, jednak przy konstruowaniu list rekomendacyjnych lepiej nie spłaszczać wyników systemu.

Przykładem takiej sytuacji jest użycie algorytmu SO na macierzy użyć  $\begin{bmatrix} 4 & 5 \\ 5 & ? \end{bmatrix}$ , kiedy to wartość nieznana zostanie wyliczona na  $\text{diff}_{i,j} + r_{u,j} = 1 + 5 = 6$ .

W porównaniu wartości błędów dla algorytmów CF zastosowanie podobieństwa między przedmiotami dało według obu miar zawsze mniejsze błędy niż zastosowanie podobieństwa między użytkownikami, średnio o 1.6% w przypadku MAE i 3.5% w przypadku MSE, ta prawidłowość wynika jednak głównie z własności danych, w szczególności tego, że filmów w bazie jest prawie 2 razy więcej niż użytkowników. W dalszych porównaniach aby uzyskać większą przejrzystość CF będzie reprezentowane przez wartości otrzymane przy podobieństwie przedmiotów (pozostałe miary jakości w większości też preferują podobieństwo przedmiotów). Wartości dla CF przy różnych funkcji podobieństwa można odczytać z wykresów:



Jak widać użycie współczynnika podobieństwa poprawia wyniki, podobnie jak uzupełnienie podobieństwa korelacji Paersona przez alternatywne podobieństwo. Dodanie dalszego podobieństwa w przypadku podobieństwa przedmiotów daje nieistotne ulepszenie, przy podobieństwie użytkowników poprawa jest większa lecz dalej ledwo zauważalna (poniżej 0.5% zmniejszenia błędów), wynika to z tego, że dalsze podobieństwo używane jest dopiero gdy nie znajdzie się żaden bezpośredni sąsiad który użył danego przedmiotu (został użyty przez danego użytkownika), a dla rozpatrywanych danych jest to rzadkie zjawisko. Można teraz porównać CF z innymi algorytmami tworzącymi predykcje ocen:



Jak widać po poprawieniu CF potrafi uzyskiwać lepsze wyniki niż algorytmy z grupy SVD (które jednak bez dużej straty można przyspieszyć zmniejszając liczbę iteracji). Widać, że zmiana SVD na SVD++ nie daje żadnej poprawy, jednak jest to związane z tym, że w bazie danych nie ma dodatkowego wejścia implicite. Dodanie wiedzy o kategoriach przedmiotów (używając gSVD++) też nie dodaje tutaj wiele wartości predykcjom systemu. Każdy z algorytmów tu przedstawionych zwraca pewne wartości dla każdej pary użytkownik przedmiot, można poprzez afiniczne przeniesienie przedziału w jakim znajdują się te oceny na  $[1,5]$  otrzymać odpowiednie wartości dla algorytmów o wejściu implicite, nie otrzymują one jednak tak dobrych wyników. Algorytmy te otrzymują wartości od  $MAE = 0.9$ ,  $MSE = 1.34$  do  $MAE = 1.61$ ,  $MSE = 3.54$  a więc porównywalne do losowych ocen o wartościach  $MAE = 1.38$ ,  $MSE = 2.88$

#### 4.4. Jakość list rekomendacyjnych

Ocena rekomendacji zwracanych użytkownikom jest bardziej skomplikowana – ważne jest to, żeby jak najwięcej z przedmiotów proponowanych rzeczywiście trafiało w gusta użytkownika, jednak w zależności od zastosowań używane mogą być listy różnych długości. Jedne algorytmy mogą bardzo dobrze radzić sobie z wyborem pojedynczego najlepszego przedmiotu, jednocześnie nie radząc sobie z generowaniem większych ilości propozycji, dlatego też stosowana jest większa ilość miar badających różne własności generowanych list.

#### 4.4.1. ROC i AUC

Ocenianą uporządkowaną listę przedmiotów przypisaną przez algorytm użytkownikowi można znając zbiór testowy uprościć do postaci binarnej poprzez przypisanie 1 dla trafionej rekomendacji (para (użytkownik, przedmiot) znajdowała się w zbiorze testowym), oraz 0 w przeciwnym przypadku. Jako, że każdy z zaprezentowanych tu algorytmów tak naprawdę zwraca dla każdej pary (użytkownik, przedmiot) jakąś ocenę, to listy rekomendacyjne wygenerowane poprzez wzięcie dla użytkownika przedmiotów których nie użył w zbiorze treningowym o najlepszych ocenach mogą być dowolnych długości (aż do ilości wszystkich przedmiotów nie użytych w zbiorze testowym). Dla dwóch list tej samej długości i z taką samą ilością 1 (odpowiadającym jednemu użytkownikowi i dwóm algorytmom) lepsza jest ta, dla której te jedynki znajdują się wcześniej, nie jest to jednak dobrze sprecyzowane pojęcie. Jedną z metod oceny takich list jest krzywa ROC (receiver operating characteristic), która dla każdego prefiksu listy zlicza osobno jaki procent liczby wszystkich 1 i 0 już się pojawił, otrzymując dwie wartości:

$$TPR(n) = \frac{TP(n)}{TP(\infty)} \quad FPR(n) = \frac{FP(n)}{FP(\infty)}$$

gdzie  $TP(n)$ , to ilość 1 w prefiksie długości  $n$ ,  $FP(n)$  to ilość 0, zaś długość prefiksu  $\infty$  oznacza całą listę. Dla każdej długości prefiksu do układu współrzędnych wprowadzany jest punkt  $(FPR(n), TPR(n))$ , tworząc wykres "funkcji" (mogą występować punkty o tej samej współrzędnej  $x$ ) niemalejącej. Krzywą ROC dla systemu otrzymujemy poprzez uśrednienie wyników po wszystkich użytkownikach (średnia wszystkich "funkcji"), ignorując użytkowników, którzy w zbiorze testowym nie użyli żadnego przedmiotu, lub użyli wszystkie, aby uniknąć wartości  $\frac{0}{0}$ . Otrzymane w ten sposób wykresy można porównać ze sobą – miejsca dla których jeden wykres przyjmuje większą wartość od drugiego pokazują dla list jakiej długości względnej algorytm zwraca więcej trafnych przedmiotów. W przypadku przecinających się wykresów aby wybrać jeden algorytm niezależnie od długości list często wyliczane jest AUC (area under curve) – powierzchnia pod wykresem krzywej ROC wyliczana jako całka (być może dyskretna) z "funkcji" opisującej tę krzywą. Warto jeszcze dodać, że w zależności od celu systemu można jako trafienie uznać tylko predykcje, których oceny w zbiorze testowym są powyżej pewnego poziomu, lub też zamiast wartości 1 użyć prawdziwych ocen ze zbioru testowego (jako  $TP(n)$  używając sumy wartości z prefiksu) i dzięki temu uzyskać dodatkowy efekt rozróżnienia jakości. Warto jeszcze wspomnieć, że sensowny system powinien mieć wykres powyżej krzywej  $y = x$  (AUC=0.5) odpowiadającej losowej kolejności przedmiotów oraz że nie da się przekroczyć optymalnej krzywej  $y = 1$  (AUC=1).

#### 4.4.2. Precyzja i MAP

W przypadku krzywej ROC badanie wartości dla krótkich list mija się z celem – jeśli lista rekomendacji ma długość rzędu 1% liczby wszystkich przedmiotów (a w rzeczywistości listy są o wiele krótsze), to wartości dla kilku porównywanych algorytmów odczytywane z krzywej nie zdążą się jeszcze rozejść od startowego punktu (0, 0).

Miarą dającą lepsze informacje na temat jakości krótkich list rekomendacyjnych jest precyzja zdefiniowana dla pojedynczego użytkownika jako stosunek ilości trafionych rekomendacji do wszystkich:

$$Precision(n) = \frac{TP(n)}{n}$$

zaś dla systemu jako średnia tych wartości po wszystkich użytkownikach (tym razem nie trzeba żadnych użytkowników pomijać).

Dzięki tej mierze można dobrać algorytm do generowania list rekomendacyjnych w prawdziwych systemach proponujących przedmioty użytkownikom. Jeżeli chcemy dodatkowo zadbać żeby początkowe rekomendacje na liście były jak najlepsze (początkowe rekomendacje są ważniejsze, a końcówka listy służy jako dodatek) wtedy warto rozpatrzyć wartości otrzymywane według zmodyfikowanej precyzji która bierze tę kolejność pod uwagę. AveP (average precision) jest zdefiniowane dla użytkownika następującym wzorem:

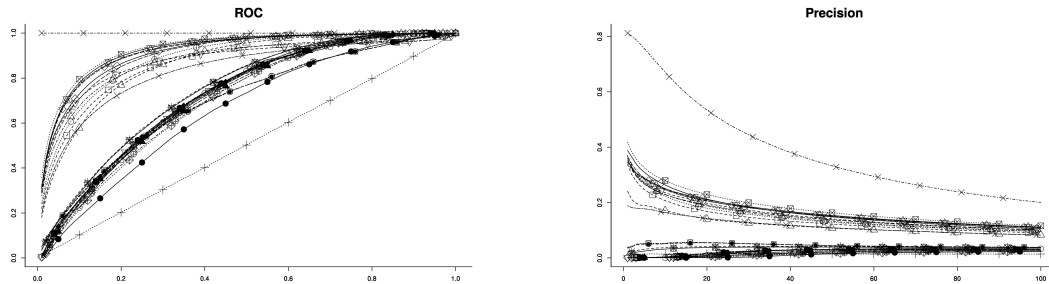
$$AveP(n) = \sum_{i=1}^n Precision(i) * V[i]$$

gdzie  $V[i]$ , to wartość wektora trafień (to czy  $i$ -ty przedmiot zarekomendowany użytkownikowi był trafiony)

Zaś wartość MAP (mean average precision) jest wektorem wartości dla systemu zdefiniowanym jako średnia AveP po użytkownikach.

#### 4.4.3. Porównanie systemów

Porównanie jakości list rankingowych można rozpocząć od spojrzenia na wykresy ROC i precyzji (dla list długości od 1 do 100) dla wszystkich porównywanych systemów.

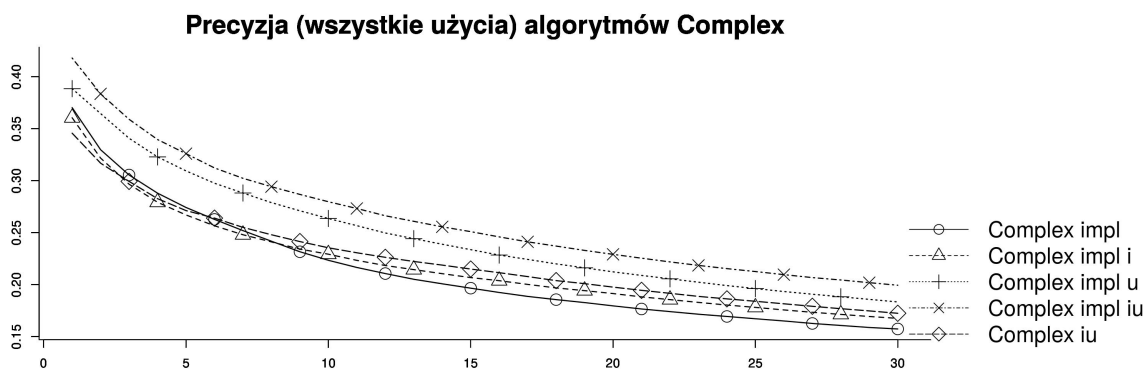
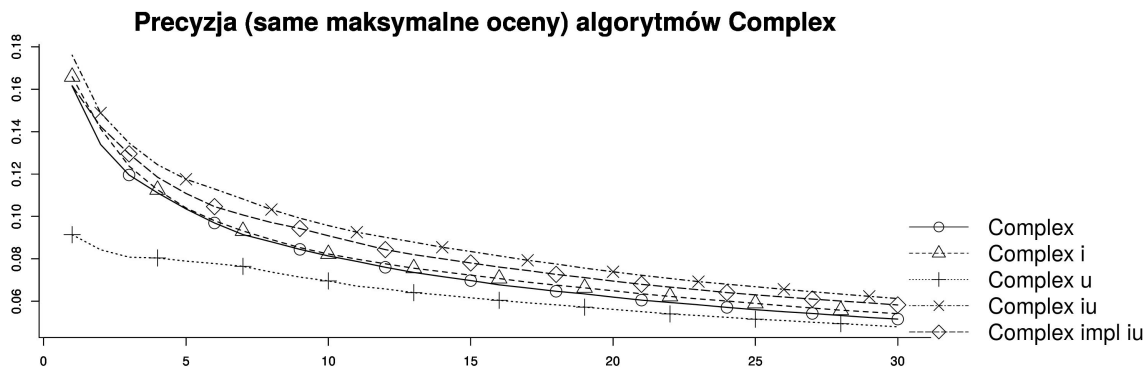


O ile ciężko z nich odczytać wartości dla konkretnych algorytmów, o tyle daje się zauważyć wyraźny podział na 4 grupy (w przypadku precyzji dwie pierwsze grupy się łączą) - w kolejności od najgorszych rekomendacje losowe, systemy z wejściem explicite (bez complex), algorytmy z wejściem implicate (wraz z complex), rekomendacje optymalne (sztucznie wygenerowane w celu porównania). Wartości AUC dla środkowych grup mieszczą się w przedziałach  $[0.696, 0.751]$  oraz  $[0.838, 0.937]$ , (losowy – 0.5, optymalny – 1), a więc rozgraniczenie jest wyraźne również po wzięciu pod uwagę różnic wartości wewnątrz grup. Taki sam rozdział występuje przy porównywaniu tych miar po uwzględnieniu rozróżnienia pomiędzy ocenami ze zbioru testowego (wyższa ocena - ważniejsza predykcja lub tylko najlepsze oceny się liczą), oraz przy używaniu miary MAP (zależy ona niemałąco od prefiksowych wartości precyzji = może dać inną decyzję tylko w przypadku przecinających się krzywych). Oznacza, to że miary te dają odwrotne porównanie między tymi grupami niż miary jakości ocen zwracanych przez systemy, daje to wniosek, że ciężko jest znaleźć pojedynczy algorytm dobry w obu sytuacjach i lepiej dobrać jeden do konkretnego problemu (lub używać dwóch niezależnie, jeśli chcemy żeby system miał obie funkcjonalności). Można teraz skupić się wyłącznie na algorytmach z grupy otrzymującej lepsze wyniki.

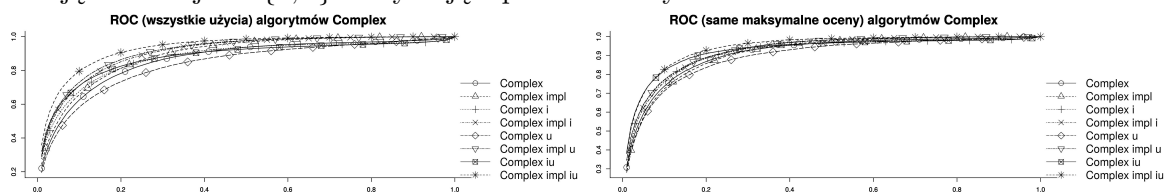
#### Complex

W przypadku porównań wewnątrz różnych odmian algorytmu Complex (regularyzacje, rodzaj wejścia) wyniki porównania zależą od tego czy chce się uzyskać listy rekomendacyjne oceniane przy pomocy wszystkich użyć ze zbioru testowego (jak w przypadku danych implicate), czy też tylko tych z najwyższą oceną (podejście rozważane w pracy [6]). Lepsze wyniki

uzyskuje się przy użyciu danych zgodnych z typem danych testowych ( $\{0,1\}$  dla implicite,  $\{-1,0,1\}$  dla poszukiwania wyłącznie najlepszych). W obu przypadkach dodanie regularyzacji poprawia wyniki, regularyzacja przedmiotów uzyskuje lepsze wyniki niż regularyzacja użytkowników, zaś użycie obu daje jeszcze lepsze rezultaty, wyjątkiem jest użycie samej regularyzacji użytkowników na danych w formacie  $\{-1,0,1\}$ , która to kombinacja uzyskuje wyniki gorsze od siedmiu pozostałych w obu przypadkach testowych.

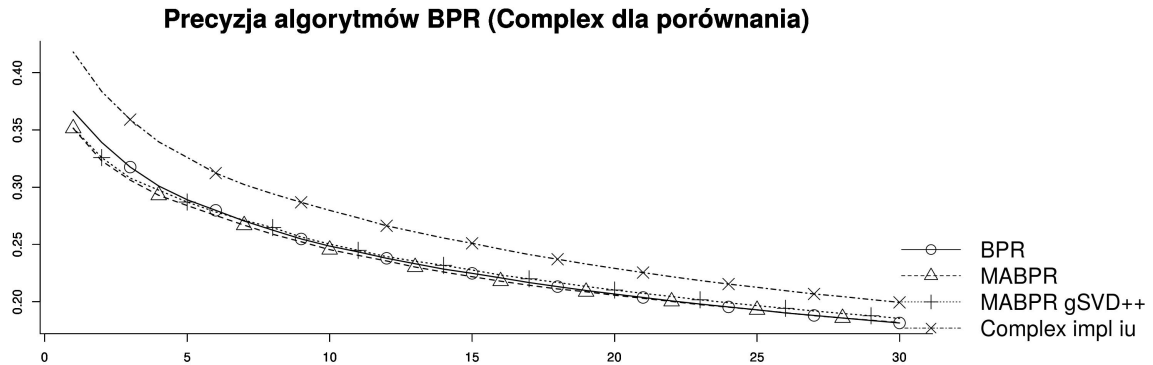


Bardzo podobne porównanie dotyczy długich list rekomendacyjnych – krzywe ROC pomiędzy różnymi regularyzacjami układają się w tych samych kolejnościach. Jedyną różnicą jest to, że tym razem nawet przy obcięciu zbioru testowego wyłącznie do maksymalnych ocen algorytmy bazujące na wejściu  $\{0,1\}$  otrzymują lepsze rezultaty.

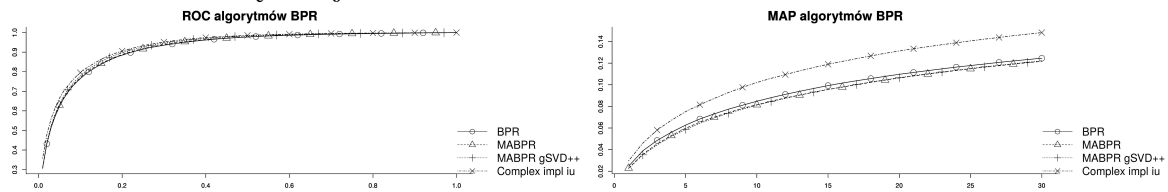




## BPR/MABPR



Ulepszenia Algorytmu BPR podobnie jak analogiczne ulepszenia SVD nie dają istotnej poprawy na rozważanym zbiorze danych. Przy wielokrotnym testowaniu daje się jednak zauważyć tendencję, że czyste BPR daje lepsze rezultaty dla list długości 1–3, zaś MABPR gSVD++ odrobinę (prawie niezauważalne na wykresach) lepsze dla list dłuższych niż 10 ( $AUC(BPR)=0.9244$ ,  $AUC(MABPR)=0.9263$ ,  $AUC(MABPR\ gSVD++)=0.9304$ ). AUC jest właśnie miarą optymalizowaną w pracy [5], algorytmy te radzą sobie według niej lepiej niż większość przypadków Complex (poza wersją z oboma regularyzacjami na wejściu implicate). Cóż więcej ze względu na losowość w wielu próbach porównania te dawały odwrotne wyniki (tutaj przedstawiłem jedno z bardziej typowych). Co ciekawe we wszystkich próbach algorytm MABPR gSVD++ uzyskuje sporą mniejsze błędy w predykcji ocen (ok. 4% dla MAE, 8% dla MSE) i mimo, że wciąż jest pod tym względem dużo gorszy od algorytmów działających na wejściu explicate, to pozwala zauważyć, że korelacja pomiędzy kategoriami i ocenami wystawianymi przez użytkowników jest niezaniebywalna. Jeżeli chce się wymagać jednak dodatkowo malejącą kolejność istotności wraz z pozycjami w listach (miara MAP), to zwykle BPR wypada lepiej. Tak małe różnice pomiędzy tymi algorytmami wynikają najpewniej z tego, że ilość trójek losowanych w algorytmach, w których oba przedmioty zostały użyte jest znikoma w przypadku rzadkich macierzy użyć (a tak jest prawie zawsze), dodatkowo różnice w ich wartościach są mniej istotne.

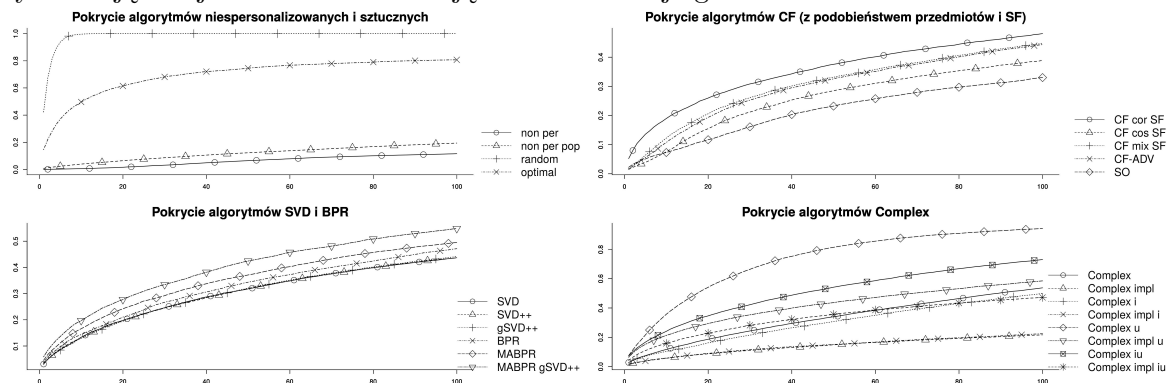


### 4.4.4. Pokrycie

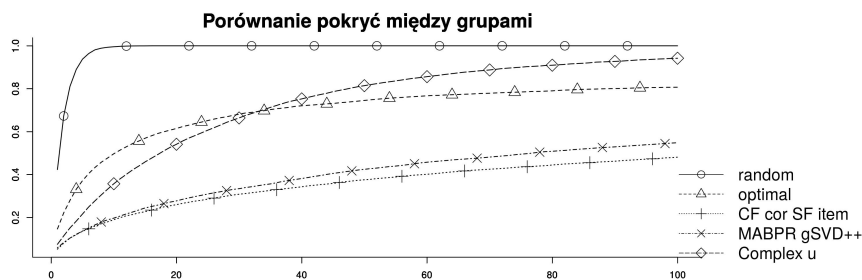
W przypadku gdy mamy do wyboru kilka algorytmów, o podobnej jakości rekomendacji możemy do stworzenia prawdziwego systemu wybrać taki, który posiada dodatkowe cechy. Jedną z takich pożądaných własności jest różnorodność proponowanych przedmiotów – proponowanie mało popularnych (mała ilość użyć), ale pasujących przedmiotów daje szansę na to że użyje je większa ilość użytkowników i dzięki temu uzyskamy na ich temat liczniejszy a więc i bardziej rzetelny odzew. Miarą badającą predykcje pod tym kątem jest pokrycie zdefiniowane dla zbioru list rekomendacji tej samej długości dla wszystkich użytkowników:

$$Coverage(n) = \frac{|\bigcup_{u \in U} \bigcup_{i=1}^n V_u[i]|}{\#Items}$$

Miara ta szczególnie dobrze pokazuje jak bardzo spersonalizowane są rekomendacje (techniki niespersonalizowane dają bardzo niskie wyniki w tej mierze), jednak nie należy kierować się nią jako głównym kryterium, gdyż bardzo faworyzuje ona losowe rekomendacje. Dodatkowo z racji nie korzystania ze zbioru testowego przy wyliczaniu miary łatwo stworzyć listy ją maksymalizujące a jednocześnie zwracające rekomendacje gorsze niż losowanie.



W przypadku algorytmów z grupy CF podobnie jak w przypadku błędów przewidywania ocen użycie podobieństwa przedmiotów oraz współczynnika podobieństwa poprawia wyniki w każdym przypadku. Porównanie pomiędzy różnymi miarami podobieństwa wypada jednak inaczej i dodanie alternatywnego podobieństwa oraz dalszych sąsiedztw ogranicza różnorodność. Pokrycie w algorytmach z grup SVD i BPR wzrasta wraz z ilością ulepszeń i daje dużo wyraźniejsze rozróżnienie niż wcześniejsze miary, pokazuje więc, że pomimo nikłych zysków z ulepszeń, ich zastosowanie może mieć pozytywny wpływ na działanie systemu na dłuższą metę. Dla algorytmu Complex regularyzacje dla użytkowników dają większą różnorodność, zaś dla przedmiotów ją zmniejszają (ich połączenie wypada trochę powyżej zwykłego). Używanie wejścia w formacie  $\{-1, 0, 1\}$  dodatkowo poprawia wartość tej miary, czyniąc algorytm Complex z regularyzacją użytkowników (ten który najslabiej radzi sobie z pozostałymi miarami spośród Complex) najlepszym po losowym (i na równi z sztucznym optymalnym) algorytmem jeżeli chodzi o różnorodność rekomendacji.



## 4.5. Porównanie czasowe

W zastosowaniach praktycznych równie ważną albo i ważniejszą od miar jakości rzeczą jest czas potrzebny do wyliczenia przewidywanych wartości czy sporządzenia list rekomendacji. Aby porównać tę własność algorytmów można podać czasy potrzebne do wyliczenia ocen (lub pseudo ocen) dla wszystkich par użytkownik – przedmiot na jednym zbiorze treningowym. Dzięki podobnemu stopniowi optymalizacji we wszystkich implementacjach takie porównanie powinno być miarodajne, jednak zastosowanie prawdziwych optymalizacji pozwala dać te same wyniki dużo szybciej (aby pokazać to przyspieszenie dodałem czas przy użyciu wbudowanego w R mnożenia macierzy w algorytmach w których ono używane). Algorytmy do

przeliczenia potrzebowały następujących czasów:

$CF_{\text{user sim}} - 370s$ ,  $CF_{\text{item sim}} - 586s$  dla zwykłych podobieństw

$CF_{\text{user sim}} - 558s$ ,  $CF_{\text{item sim}} - 877s$  dla podobieństwa mieszanego

$CF_{\text{user sim}} - 872s$ ,  $CF_{\text{item sim}} - 1627s$  dla CF ADV z odległością maksymalną 2

$CF_{\text{user sim}} - 663s$ ,  $CF_{\text{item sim}} - 1188s$  gdy użyje się do tego wbudowanego mnożenia macierzy  
SO – 577s

$SVD_{40 \text{ iter}} - 106s$ ,  $SVD_{20 \text{ iter}} - 57s$

$SVD++_{40 \text{ iter}} - 433s$ ,  $SVD++_{20 \text{ iter}} - 220s$

$gSVD++_{40 \text{ iter}} - 627s$ ,  $gSVD++_{20 \text{ iter}} - 348s$

$BPR_{40 \text{ iter}} - 442s$ ,  $BPR_{20 \text{ iter}} - 235s$

$MABPR_{40 \text{ iter}} - 710s$ ,  $MABPR_{20 \text{ iter}} - 396s$

$MABPR \ gSVD++_{40 \text{ iter}} - 1241s$ ,  $MABPR \ gSVD++_{20 \text{ iter}} - 718s$

Complex – 104s/136s/142s dla ścieżek maksymalnej długości 3/5/7

Complex – 4.2s/5.3s/5.7s gdy użyje się do tego wbudowanego mnożenia macierzy

Algorytmy SVD i BPR mogą zostać przyspieszone poprzez używanie mniejszej ilości iteracji, jednocześnie zwiększając odpowiednio szybkość uczenia, w przypadku użycia 20 iteracji zamiast 40 wyniki nie są wiele gorsze (plasują się tak samo na tle pozostałych). Nie należy jednak z tym przesadzać, ponieważ dla dużych szybkości uczenia wartości potrafią nie tylko mocno odbiegać od właściwych, ale również rozbiegać się do nieskończoności. Plusem algorytmów CF jest to, że może być zaaplikowana do każdego użytkownika osobno (jednak czas sumaryczny po wszystkich użytkownikach wtedy wzrasta). Wystarczy wtedy wyliczać tylko podobieństwa między tym jednym użytkownikiem i pozostałymi uzyskując system szybszy od pozostałych, do których nie da się zaaplikować podobnej operacji. Własność ta jest niestety tracona jeśli używa się dalszego sąsiedztwa, gdyż wtedy potrzeba wyliczyć podobieństwo pomiędzy prawie wszystkimi parami użytkowników.

## 4.6. Podsumowanie

Z porównań wyszło, że dla danych [ML] w rozważanych przypadkach testowych z przewidywaniem ocen wstawianym filmom, przez użytkowników najlepiej poradził sobie algorytm CF ADV, przewaga ta nie jest jednak duża. Z tworzeniem list rekomendacji natomiast najlepiej poradził sobie algorytm Complex z obiema regularyzacjami, który wykazał przewagę we wszystkich miarach oceniających jakość rekomendacji (często ze sporą przewagą).

Wyniki tu zamieszczone mogą odstawać od tych przedstawionych w pracach z których pochodzą przedstawione w tej pracy ulepszenia głównie ze względu na inne dobranie zbiorów treningowych i testowych.

W pracy [6] algorytm Complex badany był poprzez wyłączenie 10% danych jako zbioru testowego, który następnie został obciążony tylko do wpisów z maksymalną oceną (5). Ten przypadek testowy jest bardzo podobny do rozważanego u mnie, stąd też może wynikać wyjątkowa skuteczność tego algorytmu. Badania przeprowadzone w niniejszej pracy wykazały nadto, że algorytm Complex radzi sobie również bardzo dobrze dla danych implícite.

W pracy [7] ulepszenia CF testowane były głównie pod kątem minimalizacji błędów na dodatkowo rozrzedzonych danych (duża ilość użytkowników z zimnym startem), w której to właśnie sytuacji alternatywne podobieństwo zyskuje przewagę nad korelacją Paersona i podobieństwem kosinusowym. W takiej sytuacji również częściej zdoła się algorytmowi CF ADV korzystać z dalszych sąsiadów, co było rzadkie w przypadkach testowych rozważanych przeze mnie. Z racji na te cechy różnice w moich porównaniach są o wiele mniejsze.

Algorytmy MABPR i MABPR gSVD++ opisane w pracy [5] przyjmują podobny przypadek testowy do mojego (20% danych wyłączone do zbioru testowego). Ulepszenia te badane są tam głównie pod kątem AUC dla zwykłych krzywych ROC. Jest to też miara która u mnie pokazała największy zysk z tych ulepszeń i jednocześnie plasowała je bardzo wysoko na tle pozostałych algorytmów (przegrywając wyłącznie z Complex z obiema regularyzacjami).

Po wynikach widać, że bardzo ciężko wybrać algorytm który byłby dobry zarówno w predykcji ocen i tworzeniu rekomendacji – warto wybrać odpowiedni algorytm do konkretnego zastosowania i konkretnych danych. Z racji świeżości przedstawionych tu algorytmów można spodziewać się w najbliższym czasie kolejnych ulepszeń oraz nowych algorytmów, które pozwolą na jeszcze lepsze rezultaty otrzymywane przez systemy rekomendacyjne.

# Bibliografia

[ML] <http://grouplens.org/datasets/movielens/100k/>

- [1] Daniel Lemire, Anna Maclachlan, "Slope One Predictors for Online Rating-Based Collaborative Filtering" *SIAM Data Mining (SDM'05), Newport Beach, California, April 21-23, 2005*
- [2] Yehuda Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model" *KDD '08 Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* Pages 426 – 434
- [3] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback" *UAI '09 Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* Pages 452 – 461
- [4] Marcelo Garcia Manzato, "gSVD++: supporting implicit feedback on recommender systems with metadata awareness" *SAC '13 Proceedings of the 28th Annual ACM Symposium on Applied Computing* Pages 908 – 913
- [5] 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)*, Pages 191 – 197
- [6] Feng Xie, Zhen Chen, Jiaying, 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)*, Pages 205 – 212
- [7] Anna Satsiou and Leandros Tassioulas, "Propagating Users' Similarity towards improving Recommender Systems" *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, Pages 221 – 228
- [8] Zeno Gantner, Steffen Rendle, Lucas Drumond, and Christoph Freudenthaler *MyMediaLite* <http://www.mymedialite.net/index.html>
- [9] [https://en.wikipedia.org/wiki/Recommender\\_system](https://en.wikipedia.org/wiki/Recommender_system)