Recommender system

Magda Kozajda, Filip Wolniewski

Kwiecień 2024

Spis treści

1 Wstęp teoretyczny

W świecie przepełnionym informacjami klasyfikacja i redukcja wymiaru są jednymi z kluczowych zagadnień analizy danych. Klasyfikacja przyporządkowuje obiekty do odpowiednich klas na podstawie ich cech, podczas gry redukcja to podejście polegające na zmniejszeniu ilości danych, tak aby jednocześnie zachować jak najwięcej istotnych informacji. Nasz projekt koncentruje się na zastosowaniu różnorodnych metod redukcji wymiaru w kontekście konkretnego problemu.

1.1 Problem

Naszym zadaniem jest stworzenie prostego systemu rekomendującego filmy dla użytkowników serwisu streamingowego. Dokładniej: chcemy przewidywać oceny użytkowników dla filmów na podstawie ich wcześniejszych ocen. Nie mamy żadnych konkretnych informacji o użytkownikach i ich filmowych upodobaniach. Filmy i użytkowników identyfikujemy jedynie na podstawie przypisanego ID.

1.2 Opis danych

Korzystamy z bazy danych o nazwie *ml-latest-small* stworzonej przez *GroupLens Research* zawierającej oceny filmów ze strony *MovieLens*. Baza zawiera 100836 ocen dla 9742 filmów. Filmy oceniane były według skali 0-5 przez 610 użytkowników od 29 marca 1996 do 24 września 2018.
Indeksy dla użytkowników przydzielane były losowo a każda z osób w bazie oceniła przynajmniej 20 filmów.

Korzystamy z pliku **ratings.csv**, gdzie dane zapisane są w następującym formacie:

```
userId,movieId,rating,timestamp
1,1,4.0,964982703
1,3,4.0,964981247
1,6,4.0,964982224
1,47,5.0,964983815
1,50,5.0,964982931
...,
```

gdzie `userID` to losowo przydzielony unikalny numer ID dla użytkownika, `movieID` jest unikalnym ID filmu, `rating` jest oceną filmu w skali 0-5. Dane z kolumny `timestamp` nie będą przez nas używane.

2 Wstęp praktyczny

Pracę zaczynamy od obróbki danych wejściowych i przekonwertowania ich do odpowiedniego formatu. Chcemy też podzielić dane na zbiór uczący się i zbiór testowy. Następnie przechodzimy do redukcji wymiaru metodami **NMF**, **SVD1**, **SVD2** oraz **SGD**. Ostatnim krokiem będzie walidacja, czyli sprawdzenie skuteczności modelu.

2.1 Obróbka danych

Dane wejściowe z pliku **ratings.csv** dzielimy na dwa zbiory: zbiór treningowy, którego używamy do trenowania naszego modelu i zbiór testowy, który pozwoli nam na jego walidację. Chcemy, żeby dane treningowe stanowiły około 90% wszystkich danych. Dane ze zbioru treningowego zapisujemy w pliku **train_ratings.csv** a dane testowe w **test_ratings.csv**. Format zapisu danych jest taki sam jak w pliku **ratings.csv**.

Dane z pliku **train_ratings.csv** wczytujemy do programu i konwertujemy do macierzy Z o rozmiarze $n \times d$, gdzie n to liczba użytkowników a d - liczba filmów. Przykładowa macierz Z wygląda następująco:

```
[4. , nan, 4. , ..., nan, nan, nan],
[nan, nan, nan, ..., nan, nan, nan],
[nan, nan, nan, ..., nan, nan, nan],
...,
[nan, nan, nan, ..., nan, nan, nan],
[nan, nan, nan, ..., nan, nan, nan],
[4.5, 2. , 4. , ..., nan, nan, nan]
```

Taki zapis oznacza, że użytkownik o $n-1$ indeksie w macierzy ocenił kolejne filmy na: $4.5, 2, 4, ...$ a film o indeksie 0 w macierzy dostał oceny: $4, .., 4.5$. W macierzy znajduje się dużo wartości **nan** - są to sytuacje, kiedy użytkownik nie wystawił oceny dla danego filmu lub ocena ta trafiła do zbioru testowego.

Analogicznie postępujemy z danymi testowymi z pliku **train_ratings.csv**. Chcemy zapisać je w postaci macierzy T o rozmiarze n na d. Przykładowa macierz T to:

```
[nan, nan, nan, ..., nan, nan, nan],
[nan, nan, nan, ..., nan, nan, nan],
[nan, nan, 2. , ..., nan, nan, nan],
...,
[nan, nan, nan, ..., nan, nan, nan],
[nan, nan, nan, ..., nan, nan, nan],
[nan, nan, nan, ..., nan, nan, nan].
```

2.2 Wartości nan

Bardzo duża ilość wartości **nan** znacznie wpływa na optymalność projektu. Uzyskane wyniki będą przez to mniej dokładne a walidacja może dać gorsze efekty. Kolejnym problemem jest to, że niektóre z metod redukcji - przykładowo **NMF** działają jedynie dla kompletnych macierzy. Oznacza to, że zanim przejdziemy do następnego kroku musimy znaleźć sposób na zastąpienie **nan** wartościami liczbowymi. W naszej pracy spróbowaliśmy zrobić to na kilka sposobów. Wszystkie z metod opisane zostaną szczegółowo w poświęconym temu rozdziale.

2.3 Metody redukcji

Naszym celem jest dekompozycja uzupełnionej macierzy Z i redukcja jej wymiaru tak, aby zachować jak najwięcej informacji o danych. Redukcji dokonujemy testując różne techniki jak: Non-negative Matrix Factorization (**NMF**), Singular Value Decomposition (**SVD1**, **SVD2**) czy Stochastic Gradient Descent (**SGD**). Każda z tych metod także zostanie dokładnie omówiona w dedykowanym rozdziale.
Zmniejszając wymiar testujemy różne wielkości nowych macierzy w celu uzyskania najbardziej optymalnego wyniku. Skuteczność każdego z modeli mierzymy za pomocą miary **RMSE**.

2.4 Walidacja

Root-Mean-Square Error **RMSE** to popularna miara różnic między obserwowanymi a przewidywanymi wartościami. Idea dla **RMSE** jest następująca: trenujemy model na danych uczących z macierzy Z tak, aby dobrać jak najefektywniejsze parametry. Po skończonym treningu otrzymaliśmy estymowane oceny dla wszystkich par (użytkownik, film) a szczególnie dla tych, których prawdziwe wartości znajdują się w zbiorze testowym T. Możemy zatem porównać ze sobą wartości estymowane i wartości prawdziwe. Dzięki temu wiemy jak sprawny jest nasz model.

Ściślej wyrażamy to wzorem:

$$\mathbf{RMSE} = \sqrt{\frac{1}{\tau} \sum_{(u,m)\in\tau} (Z`[u,m] - T[u,m])^2}, \tag{1}$$

gdzie:
τ to zbiór par (u, m), których oceny zawarte są zbiorze testowym. $T[u, m]$ jest oceną filmu o indeksie m przez użytkownika z indeksem u. Dalej: $Z`$ to macierz zawierająca elementy $(u, m) \in \tau$ powstała po wytrenowaniu modelu na zbiorze uczącym Z.

Warto pamiętać, że losowy charakter podziału na dane testowe i treningowe może wpływać na otrzymane wyniki. Aby zmniejszyć wpływ losowości na rezultaty

wszystkie doświadczenia powtórzyliśmy stokrotnie. Dla każdego z eksperymentów niezależnie policzyliśmy wartość **RMSE** a wyniki zobrazowaliśmy na wykresach. Na tej podstawie dokonaliśmy wyboru najlepszych modeli i parametrów.

3 Uzupełnienie wartości nan

Przed skorzystaniem z metod redukcji wymiaru musimy zająć się problemem wartości **nan** w macierzy Z. Przetestowaliśmy kilka sposobów komplementacji danych:

1. **Cross mean**
   Dane uzupełniane są średnią z danej kolumny i wiersza: estymowana ocena filmu jest średnią ocen wszystkich użytkowników, którzy ocenili film i średnią ocen wszystkich filmów u danego użytkownika.

2. **Gl. mean/median/mode**
   Dane uzupełniane są średnią/medianą/modą wartości liczbowych z całej macierzy.

3. **M. mean/median/mode**
   Dane uzupełniane są średnią/medianą/modą wartości liczbowych w danej kolumnie.

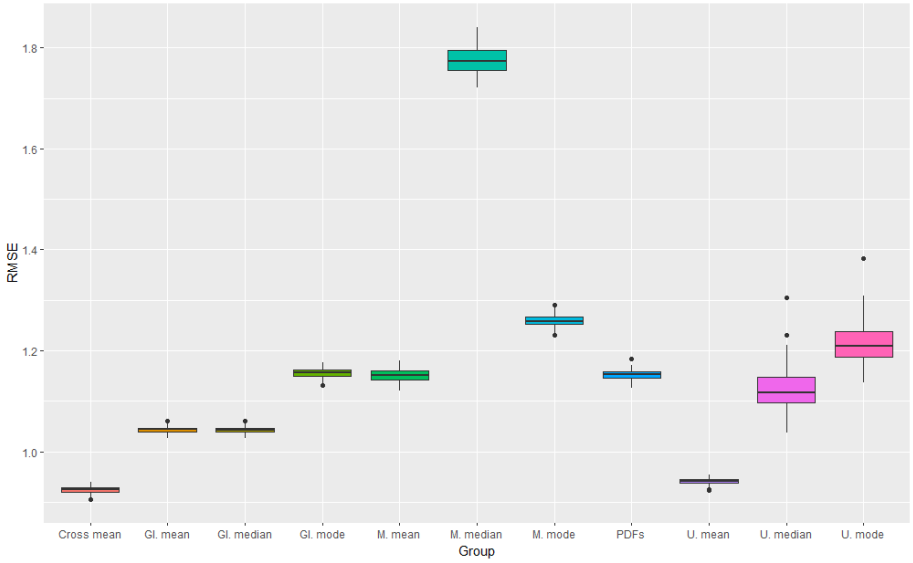
4. **U. mean/median/mode**
   Dane uzupełniane są średnią/medianą/modą wartości liczbowych w danym wierszu.

5. **PDFs**
   Dane uzupełniamy losowymi wartościami rozkładu empirycznego ocen z danej kolumny.

6. **Uzupełnienie zerami**
   Wszystkie wartości **nan** zastąpione są zerami (ta metoda okazała się najmniej skuteczna; wartość **RMSE** za każdym razem była większa od 3).

Na Rys. 1 przedstawiono wykresy pudełkowe **RMSE** dla każdej z metod otrzymane dla 100 losowych podziałów na zbiory treningowy i testowy. Najlepszymi z metod okazały się być średnia po wierszu i kolumnie (**Cross mean**) oraz średnia po wierszu (**U. mean**). W obu przypadkach otrzymaliśmy **RMSE** bliskie 0.9 co jest dobrym wynikiem. Na wykresie widzimy, że wariancja dla obu metod jest mała, co świadczy o ich stabilności. Mamy też mały rozstęp międzykwartylowy i nieliczne obserwacje odstające.

Rysunek 1: **RMSE** dla różnych metod uzupełniania macierzy Z



W dalszej części projektu decydujemy się na metodę **Cross mean**. Przykładowa macierz Z przed uzupełnieniem:

```
[4. , 0. , 4. , ..., 0. , 0. , 0. ],
[0. , 0. , 0. , ..., 0. , 0. , 0. ],
[0. , 0. , 0. , ..., 0. , 0. , 0. ],
...,
[2.5, 2. , 0. , ..., 0. , 0. , 0. ],
[3. , 0. , 0. , ..., 0. , 0. , 0. ],
[0. , 0. , 5. , ..., 3. , 3.5, 3.5]
```

i po uzupełnieniu metodą **Cross mean**:

```
[4.  , 4.17, 4.  , ..., 4.37, 4.37, 4.37],
[3.92, 3.51, 3.9 , ..., 3.92, 3.93, 3.93],
[3.72, 2.94, 3.52, ..., 2.53, 2.54, 2.54],
...,
[2.5 , 2.  , 3.21, ..., 3.13, 3.13, 3.13],
[3.  , 3.24, 3.72, ..., 3.24, 3.26, 3.26],
[3.73, 3.68, 5.  , ..., 3.  , 3.5 , 3.5 ].
```

Po przygotowaniu macierzy przechodzimy do metod redukcji wymiaru.

4 Metody redukcji wymiaru

W tej części testujemy różne sposoby przybliżania macierzy Z tak, aby zmniejszyć jej wymiar tracąc przy tym minimalną ilość informacji.

4.1 Nonnegative Matrix Factorization

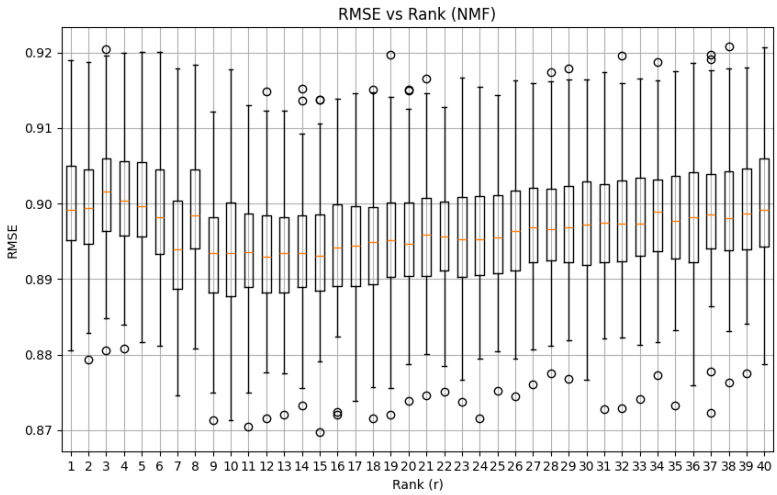
Metoda **NMF** polega na rozkładzie macierzy Z na dwie macierze W i H, z własnością, że wszystkie trzy macierze nie zawierają elementów ujemnych.

Dokładniej: dla macierzy Z rozmiaru $n \times d$ szukamy takich macierzy W $(n \times r)$ i H $(r \times d)$, żeby przybliżenie

$$Z \approx WH \tag{2}$$

było jak najlepsze. Naszym zadaniem jest wyznaczenie najbardziej optymalnej wartości parametru r.
Po obliczeniu **RMSE** dla stu różnych modeli otrzymaliśmy następujący wykres:

Rysunek 2: **RMSE** w zależności od parametru r



Zauważmy, że mediana **RMSE** dla większości modeli jest bliska 0.9, co oznacza, że metoda jest skuteczna. Po analizie wykresów pudełkowych uznaliśmy wartość $r = 9$ za najlepszą. Przekonały nas ku temu: niskie umiejscowienie pudełka; mały rozstęp w porównaniu z następnymi pudełkami; że jest to najmniejsza z dobrych rang (potencjalnym konkurentem mógłby być $r = 14$).

Przykładowy rozkład **NMF**:

```
Z =    [4.  , 4.17, 4.  , ..., 4.37, 4.37, 4.37],
       [3.92, 3.51, 3.9 , ..., 3.92, 3.93, 3.93],
       [3.72, 2.94, 3.52, ..., 2.53, 2.54, 2.54],
       ...,
       [2.5 , 2.  , 3.21, ..., 3.13, 3.13, 3.13],
       [3.  , 3.24, 3.72, ..., 3.24, 3.26, 3.26],
       [3.73, 3.68, 5.  , ..., 3.  , 3.5 , 3.5 ]

WH =  [4.22, 4.09, 4.54, ..., 4.38, 4.38, 4.38],
      [3.92, 3.53, 3.87, ..., 3.92, 3.93, 3.93],
      [3.55, 3.01, 3.39, ..., 2.51, 2.52, 2.52],
      ...,
      [3.43, 3.06, 3.34, ..., 3.12, 3.13, 3.13],
      [3.77, 3.28, 3.66, ..., 3.23, 3.25, 3.25],
      [4.12, 3.64, 4.01, ..., 3.67, 3.68, 3.68]]
```

4.2 Singular Value Decomposition 1

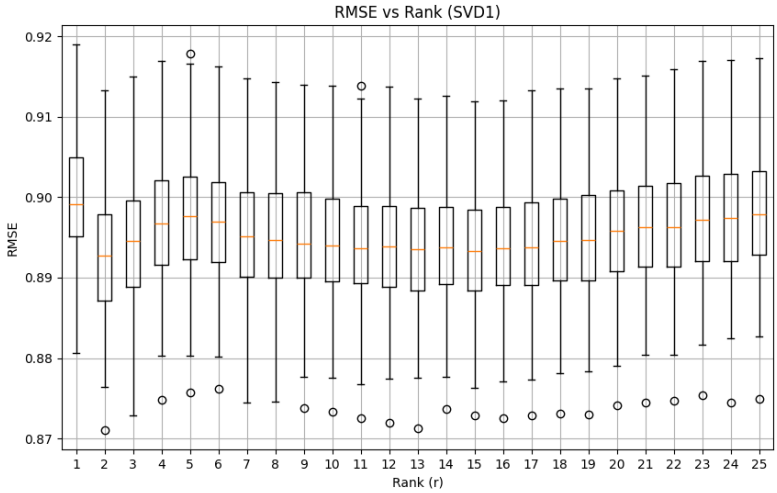
Rozkład według wartości osobliwych (**SVD**) to dekompozycja macierzy Z na iloczyn trzech specyficznych macierzy zgodnie ze wzorem:

$$Z \approx U_r \Lambda_r V_r^T. \tag{3}$$

Dla utrzymania ciągłości oznaczeń przyjmijmy $W = U_r$ oraz $H = \Lambda_r V_r^T$.

W tym przypadku także wykonujemy doświadczenie 100 razy aby znaleźć najlepsze r i otrzymujemy następujące wykresy pudełkowe:
Na rysunku 3. widzimy wartości **RMSE** w zależności od różnych wielkości r. Niezależnie od wartości parametru mediana **RMSE** utrzymuje się w przedziale $(0.89, 0.9)$ a wartość górnego kwantyla oscyluje wokół 0.9, co świadczy o udanym zastosowaniu metody. Dla metody **SVD1** wybieramy $r = 2$ ze względu na najmniejsze wartości kwantyli i regularność rozkładu.

Rysunek 3: **RMSE** w zależności od parametru r



Przykładowy rozkład **SVD1** dla $r = 2$ to:

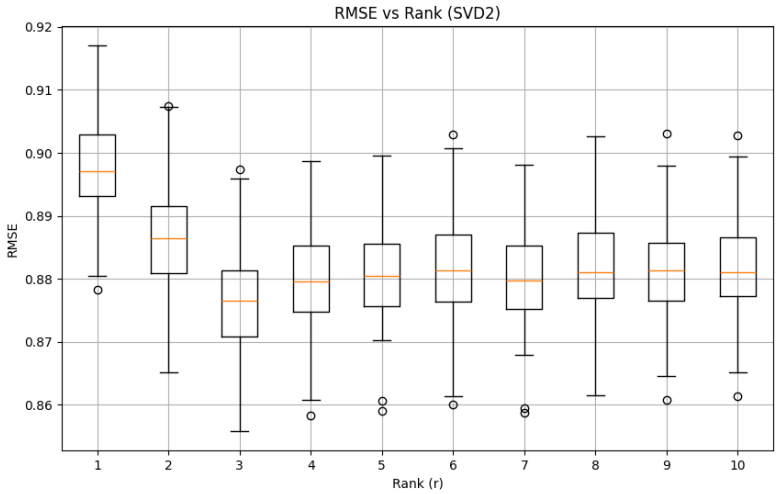
```
Z =    [4.  , 4.17, 4.  , ..., 4.37, 4.37, 4.37],
       [3.92, 3.51, 3.9 , ..., 3.92, 3.93, 3.93],
       [3.72, 2.94, 3.52, ..., 2.53, 2.54, 2.54],
       ...,
       [2.5 , 2.  , 3.21, ..., 3.13, 3.13, 3.13],
       [3.  , 3.24, 3.72, ..., 3.24, 3.26, 3.26],
       [3.73, 3.68, 5.  , ..., 3.  , 3.5 , 3.5 ]

W =    [ 0.05, -0.01],
       [ 0.04, -0.03],
       [ 0.03,  0.12],
       ...,
       [ 0.03,  0.02],
       [ 0.04,  0.05],
       [ 0.04,  0.03]

H =    [95.51, 85.82, 93.85, ..., 90.67, 90.93, 90.93],
       [ 7.2 ,  3.75,  5.86, ..., -1.15, -1.11, -1.11]

WH =   [4.49, 4.08, 4.43, ..., 4.38, 4.39, 4.39],
       [3.89, 3.57, 3.86, ..., 3.91, 3.92, 3.92],
       [3.64, 2.95, 3.44, ..., 2.52, 2.53, 2.53],
       ...,
       [3.49, 3.07, 3.4 , ..., 3.12, 3.13, 3.13],
```

```
        [3.79, 3.28, 3.67, ..., 3.24, 3.25, 3.25],
        [4.16, 3.64, 4.05, ..., 3.67, 3.69, 3.69].
```

4.3 Singular Value Decomposition 2

Dla metody **SVD2** wykonujemy analogiczne doświadczenie i generujemy wykresy pudełkowe.

Rysunek 4: **RMSE** w zależności od parametru r



W tym przypadku (Rys. 4) udało nam się osiągnąć mediany bliskie 0.88, co świadczy o otrzymaniu lepszego modelu niż w przypadku **SVD1**. Wybór $r = 3$ jest oczywisty: wartości kwantyli, najwyższych wyników i obserwacji odstających są tutaj najmniejsze.

Przykładowy rozkład **SVD2** dla $r = 3$ to:

```
Z =     [4.  , 4.17, 4.  , ..., 4.37, 4.37, 4.37],
        [3.92, 3.51, 3.9 , ..., 3.92, 3.93, 3.93],
        [3.72, 2.94, 3.52, ..., 2.53, 2.54, 2.54],
        ...,
        [2.5 , 2.  , 3.21, ..., 3.13, 3.13, 3.13],
        [3.  , 3.24, 3.72, ..., 3.24, 3.26, 3.26],
        [3.73, 3.68, 5.  , ..., 3.  , 3.5 , 3.5 ]

W =     [ 0.05, -0.01, -0.05],
        [ 0.04, -0.03,  0.03],
        [ 0.03,  0.11, -0.01],
```

```
       ...,
       [ 0.03,  0.05,  0.06],
       [ 0.04,  0.04,  0.01],
       [ 0.04,  0.06,  0.08]

H =    [96.58, 85.51, 94.85, ..., 90.61, 90.91, 90.91],
       [ 7.29,  3.37,  5.92, ..., -1.14, -1.03, -1.03],
       [ 0.57, -2.3 ,  0.16, ...,  0.26,  0.48,  0.48]

WH =   [4.  , 4.18, 4.  , ..., 4.37, 4.37, 4.37],
       [3.96, 3.49, 3.91, ..., 3.92, 3.93, 3.93],
       [3.61, 2.89, 3.41, ..., 2.53, 2.55, 2.55],
       ...,
       [2.5 , 2.  , 3.58, ..., 3.11, 3.14, 3.14],
       [3.  , 3.23, 3.69, ..., 3.24, 3.26, 3.26],
       [4.45, 3.52, 5.  , ..., 3.  , 3.5 , 3.5 ].
```

4.4 Podsumowanie

Udało nam się przeprowadzić dekompozycję macierzy Z każdą z trzech analizowanych metod redukcji wymiaru. W każdym z przypadków otrzymaliśmy wartości **RMSE** bliskie 0.9, co świadczy o skuteczności uzyskanych modeli. Najmniejsze wartości uzyskaliśmy dla metody **SVD2** z parametrem $r = 3$. Ten sposób modelowania pozwolił nam na dobrą estymację brakujących ocen filmów.

5 Stochastic Gradient Descent

Gradient Descent to proces optymalizacji, który poszukuje optymalnej wartości funkcji straty. Gradient to wektor wskazujący na kierunek największego wzrostu funkcji w punkcie. Algorytm stopniowo zbliża się do niższych wartości funkcji, poruszając się w przeciwnym kierunku do gradientu, aż do osiągnięcia minimum funkcji. Stochastic Gradient Descent (**SGD**) to wariant **GD**, który losowo wybiera pojedyncze próbki danych zamiast korzystać ze wszystkich danych treningowych przy każdej iteracji.

W części projektu dotyczącej gradientu chcemy przeformułować problem. Nie musimy już modyfikować macierzy Z przed uruchomieniem procedury redukcji. Metoda **SGD** sama uzupełni wartości **nan** najlepszymi przybliżeniami ocen.

Nowy problem brzmi następująco:
Dla danej macierzy Z o rozmiarze $n \times d$ chcemy znaleźć macierze W rozmiaru $n \times r$ i H rozmiaru $r \times d$ tak aby zminimalizować funkcję straty:

$$\sum_{(i,j):z_{ij} \neq \textbf{nan}} (z_{ij} - w_i^T h_j)^2 \tag{4}$$

Zacznijmy od uproszczenia problemu. Wyznaczmy najpierw najbardziej optymalne r dla filmów, które mają przynajmniej jedną ocenę. Pozwoli nam to na zmniejszenie rozmiaru macierzy Z i na dokładniejszą estymację brakujących ocen.

Po 10−krotnym powtórzeniu eksperymentu otrzymaliśmy następujące wyniki:

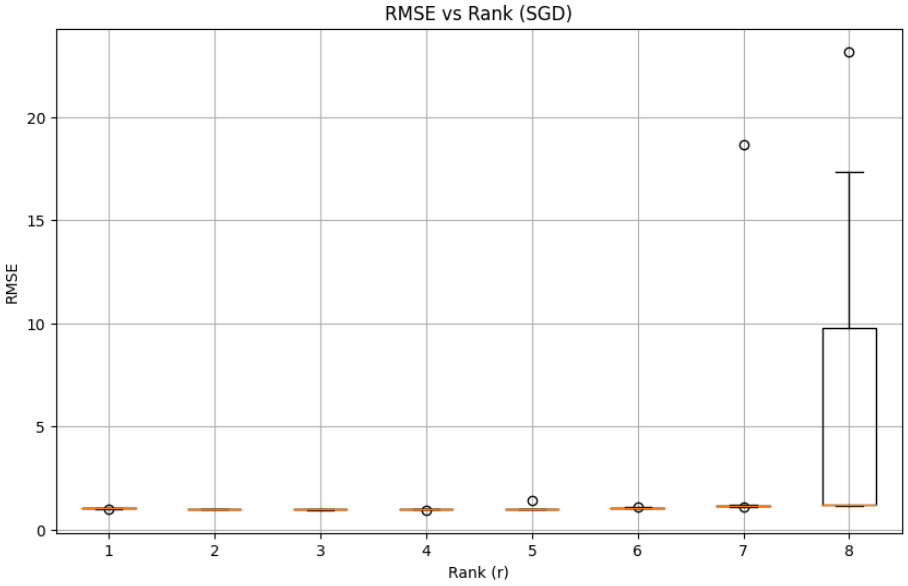
Dla $r = 8$ wariancja jest bardzo duża - podobnie jak rozstęp między-kwantylowy. Rozkład danych jest mocno niesymetryczny. Przy $r = 7$ także pojawiła się duża wartość odstająca. Dla zwiększenia czytelności wykonaliśmy też wykres bez uwzględniania tych wielkości parametru. Widzimy na nim, że dla $r = 2, 3, 4$ udało nam się uzyskać medianę **RMSE** mniejszą niż 1. Wybieramy wartość $r = 3$, ponieważ rozkład danych dla tego parametru jest najbardziej symetryczny a wartości kwantyli są najmniejsze.

Przykładowa macierz Z uzupełniona metodą **SGD** dla $r = 3$:

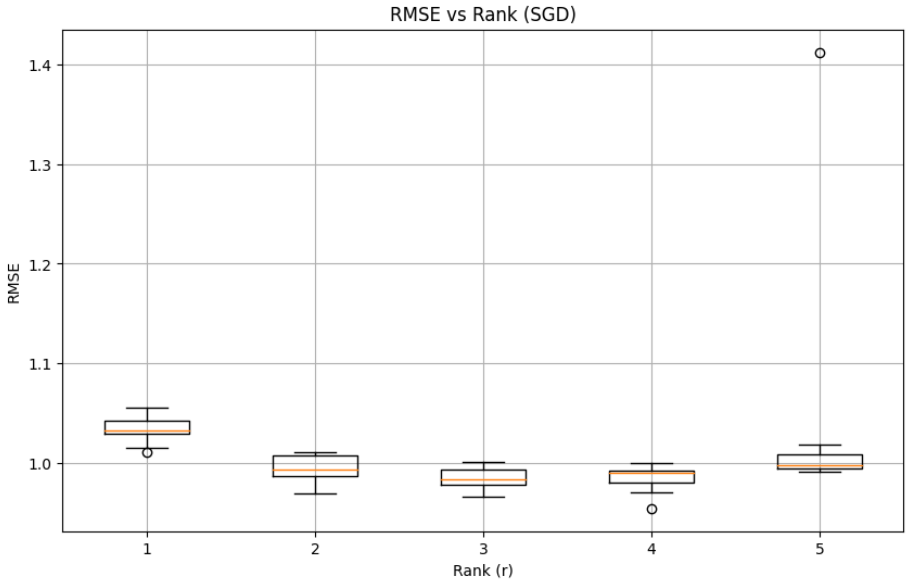
```
Z =    [4. , nan, 4. , ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       [nan, nan, nan, ..., nan, nan, nan],
       ...,
       [2.5, 2. , 2. , ..., nan, nan, nan],
       [3. , nan, nan, ..., nan, nan, nan],
       [5. , nan, nan, ..., nan, 4. , nan]
```

```
WH =   [4.67, 4.24, 3.87, ..., 0.66, 4.41, 3.6 ],
       [3.65, 3.97, 3.45, ..., 0.53, 4.12, 1.76],
       [2.86, 1.86, 2.34, ..., 0.38, 2.11, 2.57],
       ...,
       [3.58, 3.31, 2.97, ..., 0.51, 3.43, 2.72],
       [3.73, 3.63, 3.64, ..., 0.53, 3.91, 1.8 ],
       [4.02, 3.62, 3.34, ..., 0.57, 3.78, 3.1 ].
```

Rysunek 5: **RMSE** w zależności od parametru r



Rysunek 6: **RMSE** w zależności od parametru r



Dla właściwego problemu - bez usuwania z macierzy nieocenionych filmów, nie udało nam się skutecznie zoptymalizować metody **SGD**. Początkowa wartość funkcji straty jest zbyt duża, żeby zminimalizować ją przez obliczenie gradientu 200−krotnie przy learning rate równym 0.0001. Osiągane przez nas wartości **RMSE** oscylowały wokół 3 dla $r = 3$.

Uzyskana macierz Z wygląda następująco:

```
Z =     [4.73, 4.44, 3.5 , ..., 4.47, 2.78, 1.01],
        [3.46, 3.27, 2.94, ..., 3.3 , 2.23, 1.15],
        [2.31, 2.15, 1.38, ..., 2.16, 1.13, 0.13],
        ...,
        [4.3 , 4.09, 3.93, ..., 4.12, 3.02, 1.73],
        [1.59, 1.54, 1.96, ..., 1.56, 1.37, 1.19],
        [5.38, 5.07, 4.4 , ..., 5.1 , 3.21, 1.63].
```

Spis rysunków