Piotr Dzierza

System rekomendujący

1. Wstęp

Rozważamy problem przewidywania wystawionych ocen filmów przez użytkowników pewnej społeczności. Celem jest konstrukcja systemu rekomendującego, który przewidywałby, jaką ocenę wystawiłby dany użytkownik nieobejrzanemu jeszcze filmowi. Posiadając taką informację, system poleciłby dany film tylko takim użytkownikom, których prognozowana ocena byłaby wysoka.

Posiadamy następujący zbiór danych. Dane zawierają 100836 ocen 9724 różnych filmów wystawionych przez 610 różnych osób. Należy zauważyć, że najpewniej żadna osoba nie oceniła wszystkich filmów. Każda osoba i każdy film posiada jeden unikalny numer identyfikacyjny. Możliwe oceny mieszczą się w zbiorze $\{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$.

2. Wykorzystane metody

W celu przewidywania ocen, rozpatrywać będziemy następujące metody: NMF, SVD1, SVD2 i SGD.

2.1. NMF

Metoda *Non-negative matrix factorizaion* (NMF) przybliża pewną $n \times d$ nieujemną macierz Z za pomocą iloczynu dwóch nieujemnych macierzy W i H, rozmiarów odpowiednio $n \times r$ i $r \times d$. Pożądane jest, aby parametr r był jak najmniejszy, przy jednoczesnym minimalizowaniu utraty informacji. Macierze W i H uzyskuje się w wyniku rozwiązania problemu minimalizacji, który polega na znalezieniu dwóch takich nieujemnych macierzy przy parametrze r, że odległość Frobeniusa macierzy wyjściowej od ich iloczynu jest najmniejsza.

2.2. SVD1

Wiemy, że każdą macierz rzeczywistą Z możemy rozłożyć metodą SVD na iloczyn trzech pewnych macierzy U, Λ, V^T, takich że

- $Z = U\Lambda V^T$,
- U jest macierzą $n \times d$ o własności $U^T U = I$,
- Λ jest nieujemną macierzą diagonalną $d \times d$,
- V jest macierzą $d \times d$ o własności $V^T V = I$.

Metoda *Singular Value Decomposition 1* (SVD1), przy zadanym parametrze r, przybliża rzeczywistą macierz Z następująco

$$Z \approx U_r \Lambda_r V_r^T,$$

gdzie

- U_r jest macierzą $n \times r$ powstałą przez wzięcie r pierwszych kolumn macierzy U, powstałej w wyniku użycia metody rozkładu SVD,
- Λ_r jest macierzą $r \times r$ powstałą przez wzięcie r pierwszych kolumn i wierszy macierzy Λ, powstałej w wyniku użycia metody rozkładu SVD,
- V_r jest macierzą $r \times d$ powstałą przez wzięcie r pierwszych wierszy macierzy V, powstałej w wyniku użycia metody rozkładu SVD.

2.3. SVD2

Metoda SVD2 w pewnym sensie polega na n-krotnym wykonywaniu SVD1. Ponownie przyjmując r jako dany parametr, przybliżanie pewnej rzeczywistej macierzy Z za pomocą SVD2 możemy wyrazić rekurencyjnie, poprzez

$$Z \approx Z^{(n)} = SVD_r \left[\begin{cases} Z, & \text{gdy } z_{ij} > 0, \\ Z^{(n-1)} & \text{w przeciwnym przypadku.} \end{cases} \right],$$

gdzie $Z^{(0)} = SVD_r[Z]$.

2.4. SGD

Metoda *stochastic gradient descent* (SGD) bazuje na iteracjach, działa w następujący sposób. W odróżnieniu do poprzednich metod, w SGD nie musimy uzupełniać braków macierzy Z, choć niezmiennie poszukujemy dwóch macierzy W i H rozmiarów $n \times r$ i $r \times d$, odpowiednio. Interesujący nas problem przybliżenia ma postać

$$\arg\min_{W,H} \sum_{(i,j):z_{ij} \neq '?'} (z_{ij} - w_i^T h_j)^2 + \lambda (||w_i||^2 + ||h_j||^2)$$

(tutaj h_j to j-ta kolumna macierzy H, z kolei w_i^T to i-ta kolumna macierzy W, $\lambda > 0$ to parametr, przez $'?'$ oznaczamy wartość brakującą). SGD rozwiązuje to zagadnienie poprzez początkowe losowe wybranie macierzy W i H i przypisanie kolejno

$$e_{ij} = z_{ij} - w_i^T h_j,$$

$$w_i^T = w_i^T + \gamma(e_{ij} h_j + \lambda w_i^T),$$

$$h_j = h_j + \gamma(e_{ij} w_i^T + \lambda h_j),$$

przy parametrze $\gamma > 0$. Następnie, powyższe przypisania (bez ponownego losowania W i H) iteruje się wielokrotnie. Ostatecznym wynikiem jest iloczyn W i H.

3. Opis programu

Załączony program o nazwie `recom_system.py` przyjmuje następujące argumenty:

- --train to plik z danymi treningowymi,
- --test to plik z danymi testowymi,
- --alg to jeden z algorytmów NMF, SVD1, SVD1, SGD,
- --result to nazwa pliku, do którego zapisany zostanie wynik.

Program z danych treningowych tworzy macierz Z w taki sposób, że kolumny odpowiadają filmom, a wiersze użytkownikom. Wartościami tej macierzy są oceny wpisane w odpowiednie miejsca określone przez Id osoby i Id filmu jako współrzędne. W podobny sposób tworzy się macierz V z danych testowych. Należy zauważyć, że obie te macierze mają brakujące wartości. Dodatkowo, niech $\mathfrak{V}$ oznacza zbiór par $(u;m)$, dla których mamy oceny w zbiorze testowym – wówczas ocena jest podana przez $V[u;m]$.

Załóżmy, że algorytm, ucząc się na Z, wylicza Z', taką macierz, w której są elementy $Z'[u;m]$ dla $(u;m) \in \mathfrak{V}$. Skrypt ten pozwala na obliczenie błędu predykcji *RMSE* zdefiniowanego poprzez

$$RMSE = \sqrt{\frac{1}{|\mathfrak{V}|} \sum_{(u;m)\in\mathfrak{V}} (Z'[u;m] - V[u;m])^2}.$$

Program zapisuje wartość *RMSE* w pliku o nazwie podanej w argumencie *result_file*.

Skrypt ten zakłada, że podane argumenty spełniają poniższe wymagania. Pliki *train* i *test* są formatu csv oraz dane w nich zawarte muszą być w formie takiej samej jak plik wyjściowy `rating.csv` (nazwy kolumn i wierszy). Zbiór treningowy powinien zawierać przynajmniej jedną ocenę każdego użytkownika. Argument *alg* przyjmuje tylko jedną z możliwości: NMF, SVD1, SVD2, SGD. Argument *result* to tylko nazwa pliku, program sam utworzy plik z rozszerzeniem txt. Plik z wynikiem zostanie utworzony w miejscu, gdzie znajduje się skrypt i ważne jest, by nie było już innego pliku z wywoływaną w programie nazwą, ponieważ w takim przypadku program zwróci błąd (chyba, że poda się nazwę z wpisanym rozszerzeniem txt - wtedy, w wypadku istniejącego już takie go pliku, istniejący plik zostanie bez ostrzeżenia zastąpiony nowym). Program nie sprawdza poprawności zadanych argumentów. Ich błędne podanie może skutkować zwróceniem błędów z poziomu programu python lub prawdopodobnie błędnym wynikiem *RMSE*.

4. Szczegóły dotyczące poszczególnych metod

W tej części przedstawione zostaną szczegółowe informacje dotyczące działania skryptu.

Niezależnie która z metod: NMF, SVD1, SVD2 czy SGD zostanie wywołana w argumencie, program zaczyna swoje działanie od załadowania niezbędnych bibliotek i wczytania argumentów. Następnie program zamienia numery Id filmów tak, by były one kolejnymi liczbami naturalnymi. Sposób zamiany numerów Id zachowuje ich porządek. Oznacza to, że kolejność filmów wyznaczona przez numer ID się nie zmieni. Zamiana numerów id filmów odbywa się jednocześnie na zbiorach treningowym i testowym. Następnie tworzona jest macierz Z ze zbioru treningowego, korzystając z jednej z rozważanych metod uzupełnia brakujących danych (szczegóły poniżej). Kolejnym krokiem jest stworzenie macierzy V, która w miejsce braków zawsze będzie przyjmowała 0. Potem na macierzy Z wykonywany jest jeden z algorytmów przy pewnym wybranym parametrze r. Następnie obliczane jest *RMSE* i zapisywane w nowo utworzonym pliku tekstowym.

4.1. Dobór parametrów i wyniki algorytmu NMF

NMF posiada dwa "parametry" - liczbę r opisującą jeden z wymiarów macierzy W i H oraz sposób uzupełniania wartości brakujących macierzy Z. Tutaj zawsze sprawdzano $r = 10, 20, 50, 100$. W przypadku bardzo dobrych wyników, sprawdzano jeszcze inne wartości r, niekoniecznie w sposób regularny. Macierz Z próbowano uzupełniać poprzez: wstawianie 0,

wstawianie losowych liczb ze zbioru $\{0.5, 1, 1.5, \ldots, 4.5, 5\}$ oraz wstawianie średniej z danego wiersza i uzupełnianie nią braków w tym wierszu (średnia ocena użytkownika).

Poniżej przedstawione zostaną wyniki symulacji dla algorytmu NMF przy różnych wartościach r i metodach uzupełniania. Tutaj zawsze liczono 75 wartości *RMSE* dla różnych zbiorów testowych i treningowych. Następnie wynik uśredniano. Każda kombinacja parametrów testowana była na tych samych zbiorach testowych i treningowych.

Tabela 1: Wyniki średnich *RMSE* metody NMF dla różnych wartości parametru r i różnych sposobów wypełniania braków danych w macierzy Z.

r	zera	losowo	średnie wierszy
10	2.9260	1.1885	0.9095
20	2.9874	1.2001	0.9078
50	3.1321	1.2370	0.9127
100	3.3110	1.2877	0.9224

Z tabeli 1 możemy odczytać, że metoda zastępowania średnimi w wierszach jest zdecydowanie lepsza od pozostałych na każdym sprawdzonym poziomie parametru r. Okazało się jednak, że sprawdzając niektóre wartości r między 10 i 20 oraz między 20 i 50, da się uzyskać trochę lepszy rezultat. Uzupełnianie średnią dla $r = 15$ dało średnie *RMSE* na poziomie 0.9064 i to te parametry zostały wpisane do kodu załączonego skryptu.

4.2. Dobór parametrów i wyniki algorytmu SVD1

SVD1 posiada te same parametry, co NMF i sprawdzane one będą w ten sam sposób.

Poniżej przedstawione zostaną wyniki symulacji dla algorytmu SVD1 przy różnych wartościach r i metodach uzupełniania. Tutaj zawsze liczono 75 wartości *RMSE* dla różnych zbiorów testowych i treningowych. Następnie wynik uśredniano. Każda kombinacja parametrów testowana była na tych samych zbiorach testowych i treningowych.

Tabela 2: Wyniki średnich *RMSE* metody SVD1 dla różnych wartości parametru r i różnych sposobów wypełniania braków danych w macierzy Z.

r	zera	losowo	średnie wierszy
10	2.8742	1.1995	0.9056
20	2.9413	1.2075	0.9086
50	3.1091	1.2434	0.9187
100	3.3236	1.3099	0.9290

Z tabeli 2 możemy odczytać, że metoda zastępowania średnimi w wierszach jest ponownie zdecydowanie lepsza od pozostałych na każdym sprawdzonym poziomie parametru r. Sprawdzane były również różne wartości r mniejsze od 10 oraz między 10 i 20, ale nie znaleziono średnie *RMSE* zawsze było wyższe od tego dla $r = 10$. Zaimplementowano zatem metodę uzupełniania średnimi z $r = 10$, która uzyskała średni *RMSE* na poziomie 0.9056.

4.3. Dobór parametrów i wyniki algorytmu SVD2

SVD2 posiada dodatkowy parametr w porównaniu do poprzednich dwóch metod. Tym parametrem jest m, czyli liczba iteracji wykonania uciętego SVD. Tutaj strategia sprawdzania parametrów uległa zmianie. Najpierw sprawdzano wszystkie metody uzupełniania braków dla $r = 5, 10, 20, 50$. Potem wybrano najlepszą metodę uzupełniania i sprawdzono kombinacje $m = 15, 20, 50$ i $r = 5, 10, 20, 50$. Pierwszą przyczyną zmian sprawdzania jest znaczący wzrost czasu wykonania programu. Drugą jest zauważenie, że w poprzednich przypadkach wartości *RMSE* zaczęły rosnąć przy dużych r.

Poniżej przedstawione zostaną wyniki symulacji dla algorytmu SVD2 przy różnych wartościach r i metodach uzupełniania oraz $m = 50$. Tutaj zawsze liczono 75 wartości *RMSE* dla różnych zbiorów testowych i treningowych. Następnie wynik uśredniano. Każda kombinacja parametrów testowana była na tych samych zbiorach testowych i treningowych.

Tabela 3: Wyniki średnich *RMSE* metody SVD2 dla różnych wartości parametru r i różnych sposobów wypełniania braków danych w macierzy Z przy $m = 50$.

r	zera	losowo	średnie wierszy
5	1.3849	0.9463	0.8881
10	1.4674	0.9647	0.9325
20	1.7271	1.0135	0.9990
50	2.2301	1.1207	0.9888

Z tabeli 3 możemy odczytać, że metoda zastępowania średnimi w wierszach jest lepsza od pozostałych na każdym sprawdzonym poziomie parametru r. Przy dalszym optymalizowaniu będziemy sprawdzać tylko ten sposób tworzenia Z.

Tabela 4: Wyniki średnich *RMSE* metody SVD2 dla różnych wartości parametru r i różnych wartości m przy uzupełnianiu Z średnimi po w wierszach.

r	$m = 15$	$m = 20$	$m = 50$
5	0.8805	0.8816	0.8881
10	0.9015	0.9070	0.9325
20	0.9392	0.9562	0.9990
50	0.9736	0.9884	0.9888

Z tabeli 4 możemy odczytać, że najlepszy rezultat wśród sprawdzanych parametrów daje połączenie $r = 5$, $m = 15$ i uzupełniania braków średnimi w wierszach. Średni *RMSE* w takich warunkach wynosi 0.8805. Parametry te zostały wpisane do skryptu.

4.4. Dobór parametrów i wyniki algorytmu SGD

Metoda SGD bardzo różni się od 3 pozostałych. Jej parametrami są r i m oznaczające to samo, co wcześniej oraz γ i λ będące współczynnikami uczenia się algorytmu. Brano pod uwagę $r = 5, 10, 15, 20$, $m = 25, 50$, $\lambda = 0.01, 0.1$ oraz $\gamma = 0.001, 0.01$. Dość skromne zróżnicowanie parametrów wynika z ich liczby. Wartość średniego *RMSE* liczono tutaj na podstawie 30 różnych zbiorów treningowych i testowych.

Tabela 5: Wyniki średnich *RMSE* metody SGD dla różnych wartości parametrów i $m = 25$.

r	$\lambda = 0.1$ & $\gamma = 0.001$	$\lambda = 0.01$ & $\gamma = 0.001$	$\lambda = 0.01$ & $\gamma = 0.01$	$\lambda = 0.1$ & $\gamma = 0.01$
5	0.9515	0.9342	0.9099	0.9197
10	0.9104	0.9114	0.9331	0.8919
15	0.9105	0.9168	0.9485	0.8861
20	0.9194	0.9287	0.9638	0.8911

Tabela 6: Wyniki średnich *RMSE* metody SGD dla różnych wartości parametrów i $m = 50$.

r	$\lambda = 0.1$ & $\gamma = 0.001$	$\lambda = 0.01$ & $\gamma = 0.001$	$\lambda = 0.01$ & $\gamma = 0.01$	$\lambda = 0.1$ & $\gamma = 0.01$
5	0.9225	0.9072	0.9628	0.9191
10	0.8944	0.8974	0.9629	0.8888
15	0.8942	0.9030	0.9848	0.8809
20	0.9002	0.9099	1.0094	0.8854

Porównując tabele 5 i 6, widzimy, że najlepszy wynik średniego *RMSE* równy 0.8809 dała kombinacja $m = 50$, $r = 15$, $\lambda = 0.1$ i $\gamma = 0.01$. Ponadto, można zaobserwować, że kombinacja $\lambda = 0.01$ & $\gamma = 0.001$ miała zawsze gorsze wyniki dla przy wzroście m. Jest to dość zaskakujące, zwiększenie liczby powtórzeń przeważnie powoduje polepszenie wyników, tak jak we wszystkich innych rozpatrywanych przypadkach w tej metodzie.

Na podstawie powyższych wyników postanowiono jeszcze wyrywkowo sprawdzić średnie *RMSE* dla $r = 10, 15$, $\lambda = 0.1$ i $\gamma = 0.01$ oraz $m = 100$. Takie ograniczenie badań ponownie usprawiedliwiamy rosnącym czasem trwania programu oraz skupieniem się na miejscu najbardziej obiecującym pod względem polepszenia rezultatów. Dla $r = 10$ uzyskano wynik średniego *RMSE* równy 0.8885, a dla $r = 15$ wynik to 0.8799. Widzimy zatem, że nastąpiło niewielkie polepszenie obu wyników, w porównaniu z tabelą 6. Zaimplementowane zostały zatem $m = 100$, $\lambda = 0.1$, $\gamma = 0.01$ i $r = 15$.

5. Podsumowanie

Wśród rozważanych metod widać, że najlepsze wyniki uzyskuje się przy doborze niskich wartości r. Wszystkie metody, przy przyjęciu wybranych tutaj najlepszych parametrów, dają podobne wyniki średniego *RMSE* będącego w przedziale $[0.87, 0.91]$. Należy mieć świadomość, że przedstawione w tym raporcie rezultaty mogą okazać się słabsze, w porównaniu z innymi parametrami, które nie zostały sprawdzone. Taki potencjał widać przede wszystkim w algorytmie SGD, gdzie można wziąć inne wartości m oraz inne parametry uczenia.

6. Dodatek - zaokrąglanie Z'

Jednym z innych pomysłów podejścia do tematu było zaokrąglenie wartości obliczonej już macierzy Z' do najbliższego 0.5. Miało to na celu doprowadzenie przewidywanych wartości

do formy oceny, jaką wystawia rzeczywisty użytkownik portalu. Niestety, pomysł ten został porzucony ze względu na gorsze wyniki i zdecydowanie dłuższy czas działania programu. Obliczone w ten sposób zostały rezultaty dla algorytmów NMF i SVD1, gdzie średnie wyniki *RMSE* zawsze okazywały się o około 0.2 wyższe od wersji bez zaokrąglenia.