Lista 5 (Lab) Termin wysłania na SVN do 3.05.2020

Przy pisaniu programów (funkcji) **NALEŻY** stosować się do ogólnie przyjętego stylu programowania w języku Python. Proszę dokładnie przeczytać PEP 8 (tutorial, google python style) i PEP 257.

1. (10pt) Napisz wykorzystując regresję liniową program, który na podstawie oceny filmów przez użytkowników będzie starał się przewidzieć ocenę innych użytkowników. Jako dane wykorzystamy zbiór MovieLens Latest Datasets. Dokładnie wybierzemy mniejszy zbiór, pobierz plik ml-latest-small.zip. Zadanie polega na wybraniu z pliku *ratings.csv* tych użytkowników (userId), którzy ocenili film 'Toy Story (1995)', który w tym pliku ma identyfikator '1' (patrz *movies.csv*). W tym pliku osób takich jest 215. Wtedy zgodnie z zapisem z wykładu x_{ij} będzie oceną i-tego użytkownika dla $i = 0,\ldots,214$, bo taki jest nasz zbiór osób, które oceniły 'Toy Story' oraz j będzie oceną j-tego filmu dla $j = 0,\ldots,m$. Jako j można wybrać $\mathrm{movieId}$ filmu, czyli np. film o $\mathrm{movieId} = 42$ oceniony przez użytkownika 5 (nie jest to userId, tylko piąta osoba ze zbioru 215 osób), który ocenił film jako np. 3.5 wpisujemy $x[5, 42] = 3.5$. Natomiast y_i to ocena 'Toy Story' przez i-tego użytkownika. Zatem tworzymy macierz $X = [x_{ij}]$ oraz wektor y_i gdzie $i = 0,\ldots,215$ oraz $j = 0,\ldots,m$. Dla tak przygotowanych danych wykonujemy:
   - regresje liniową na całym zbiorze użytkowników dla $m = 10, 1000, 10000$, czyli np. dla $m = 10$ ignorujemy filmy o $\mathrm{movieId} > 10$ i robimy regresje dla tak okrojonego zbioru ocen. Jakie dostajemy błędy. Pokaż na wykresie.
   - podziel zbiór osób na tzw. zbiór treningowy oraz zbiór testowy np. weźmy $i = 0,\ldots,200$ to będzie zbiór treningowy i na takim zbiorze osób wykonajmy regresje natomiast później sprawdzamy już dla całości (215). Zatem ostatnie 15 ocen będziemy chcieli przewidzieć (zbiór testowy). Zrób przewidywanie dla $m = 10, 100, 200, 500, 1000, 10000$. Wyświetl wynik predykcji i wynik prawidłowy dla tych 15 osób.

   Przykład danych do regresji jakie otrzymujemy dla $m = 10$ i $n = 215$.

```
X = [[0.  4.  0.  0.  4.  0.  0.  0.  0.  0. ]
     [0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
     [0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
     [0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
     [0.  0.  0.  0.  0.  0.  0.  0.  0.  0. ]
     [3.  0.  0.  0.  4.  0.  0.  0.  0.  0. ]
     [3.  3.  0.  0.  0.  2.  0.  0.  2.  0. ]
     [3.5 0.  0.  0.  0.  0.  0.  0.  5.  0. ]
     ...
y = [[4. ]
     [4. ]
     [4.5]
     [2.5]
     [4.5]
     [3.5]
     [4. ]
     [3.5]
     ...
```

2. (15pt) Napisz system rekomendacji filmów. Systemy takie są wykorzystywane przez różne firmy np. Netflix organizował konkurs na opracowanie algorytmu, który będzie przewidywał ocenę użytkownika (Netflix Prize). W zadaniu tym zaimplementujemy podobny system, który jednak zamiast przewidywania będzie na podstawie preferencji użytkownika rekomendował filmy, które najprawdopodobniej mu się spodobają. Istnieje wiele sposobów, aby taki system napisać, dla zainteresowanych bardziej tematem proponuje zobaczyć np. Recommendation Systems. W tym zadaniu wybierzemy w miarę prosty i łatwy do implementacji system rekomendacji. Sformalizujmy problem. Załóżmy, że mamy macierz oceny, gdzie wiersze będą reprezentować użytkowników a kolumny filmy np.

	Matrix	Star Wars IV
Alice	5	4
Bob	0	1
John	2	2
Ada	5	5

   Patrząc na powyższą macierz widać, że kolumny pierwsza i druga mają podobne oceny stąd można wywnioskować, że filmy Matrix i Star Wars IV są (według użytkowników) podobne do siebie, czyli jeśli komuś podobał się Matrix to jest duża szansa, że będzie podobał mu się Star Wars i odwrotnie. Dlatego "podobieństwo" sformalizujemy przez wykorzystanie podobieństwa cosinusowego, czyli jeśli $x = (x_1, x_2, \ldots, x_n)$ i $y = (y_1, y_2, \ldots, y_n)$, wtedy:

$$x_1y_1 + x_2y_2 + \cdots + x_ny_n = x \cdot y = \|x\|\|y\| \cos(\theta)$$

   Gdzie θ jest kątem między wektorami. Wygodniej będzie nam normalizować wektory. Wtedy

$$x_1y_1 + x_2y_2 + \cdots + x_ny_n = \cos(\theta).$$

   Wtedy $\cos(\theta)$ reprezentuje podobieństwo jednego wektora do drugiego, czyli jak bardzo jedne oceny są bliskie innym. Dla naszej macierzy otrzymujemy

```
>>> x = np.array([[5,0,2,5], [4,1,2,5]]).T
>>> x
array([[5, 4],
       [0, 1],
       [2, 2],
       [5, 5]])

# obliczamy normę wektorów kolumnowych
>>> np.linalg.norm(x, axis=0)
array([7.34846923, 6.78232998])

# obliczamy znormalizowaną macierz
>>> x/np.linalg.norm(x, axis=0)
array([[0.68041382, 0.58976782],
       [0.        , 0.14744196],
       [0.27216553, 0.29488391],
       [0.68041382, 0.73720978]])

# teraz weźmy własną ocenę filmów Matrix - 4, Star Wars IV - 3
# dla wygody zapisujemy jako wektor kolumnowy
>>> y = [[4],[3]]

# obliczamy podobieństwo cosinusowe z każdym użytkownikiem (skorzystamy z mnożenia macierzowego)
>>> z=np.dot(x/np.linalg.norm(x, axis=0), np.array(y)/np.linalg.norm(y))
>>> z
array([[0.89819175],
       [0.08846517],
       [0.39466277],
       [0.98665692]])

# normalizujemy otrzymany wektor (będzie on reprezentował nasz profil filmowy)
>>> z/np.linalg.norm(z)
array([[0.64422929],
       [0.06345177],
       [0.28307243],
       [0.70768107]])
```

   Teraz musimy obliczyć podobieństwo cosinusowe między naszym profilem a każdą kolumną macierzy, aby znaleźć takie filmy, które są podobne do naszego profilu. Sortujemy po otrzymanym podobieństwu i dostajemy rekomendacje. Wykorzystując nasz przykład, otrzymujemy

```
>>> X = x/np.linalg.norm(x, axis=0)
>>> Z = z/np.linalg.norm(z)
>>> np.dot(X.T, Z)
array([[0.99690104],
       [0.99448407]])
```

   Stąd dostaliśmy większą rekomendacje dla filmu Matrix (0.996) niż Star Wars IV (0.994). Co było oczywiście oczekiwane skoro nasza ocena była Matrix - 4, a Star Wars IV - 3. Możemy, teraz przejść do właściwej części zadania. Napisz system rekomendacji filmów który będzie wykorzystywał dane MovieLens Latest Datasets. Dokładnie mniejszy zbiór (który dodatkowo trochę jeszcze zmniejszymy). Pobierz plik ml-latest-small.zip. Interesować, będą nas głównie dwa pliki movies.csv oraz ratings.csv. W pliku ratings.csv mamy właściwie wszystkie dane, które będą nam potrzebne aby stworzyć macierz oceny. Ponieważ nawet w tym przypadku otrzymana macierz będzie dość duża dlatego w trakcie wczytywania danych z pliku proszę wziąć pod uwagę tylko te wiersze (z pliku ratings.csv) w których **movie_id** (druga kolumna) jest mniejsze od 10000 (resztę ignorujemy). Dla aktualnych danych ze strony otrzymamy, wtedy macierz mniej więcej 611x9019. Przykładowy wynik dla wybranego profilu filmowego:

```
# wektor my_ratings odpowiada wektorowi y z przykładu wyżej
my_ratings = np.zeros((9019,1))
my_ratings[2571] = 5      # patrz movies.csv  2571 - Matrix
my_ratings[32] = 4        # 32 - Twelve Monkeys
my_ratings[260] = 5       # 260 - Star Wars IV
my_ratings[1097] = 4
my_ratings_norm = my_ratings/np.linalg.norm(my_ratings)

...

# otrzymujemy wynik rekomendacji po posortowaniu
# (cos(θ), movies_id)
(0.8675507828468105, 260)
(0.8362098349303669, 2571)
(0.8227451213877744, 1196)
(0.8002349214247857, 1210)
(0.7458504689612442, 1097)
(0.7286029159733108, 32)
(0.7265369898748615, 1198)
(0.7095672455110477, 1240)
(0.7029872178855614, 1270)
```

   Otrzymane reprezentacje wyświetl w postaci nazw filmów, korzystając z movies.csv. Uwaga: w rzeczywistych danych otrzymamy dużą liczbę zer, nawet całe kolumny, wtedy dostaniemy wartości NaN przy dzieleniu! Rozwiąż ten problem wykorzystując np.nan_to_num(...).

3. (20pt)* Napisz rekomendacje dla pełnego zbioru danych ml-latest.zip. Wykorzystaj do tego macierze rzadkie (sparse matrix) np. z pakietu scipy (lub inny sposób, który pozwala na obróbkę tak dużej ilości danych).