hw06

March 1, 2021

0.1 REI602M Machine Learning - Homework 6

0.1.1 Due: *Sunday* 28.2.2021

Objectives: k-means clustering and recommender systems

Name: Alexander Guðmundsson, **email:** (alg35@hi.is), **collaborators:** (if any)

Please provide your solutions by filling in the appropriate cells in this notebook, creating new cells as needed. Hand in your solution on Gradescope, taking care to locate the appropriate page numbers in the PDF document. Make sure that you are familiar with the course rules on collaboration (encouraged) and copying (very, very, bad).

0.1.2 1. [Topic discovery via k-means, 30 points]

Here you are to use the k-means algorithm to cluster the Wikipedia data set from Homework 5 (file `wikipedia_corpus.npz`).

Run k-means with different values of k, e.g. $k = 2, 5, 8$ and investigate your results by looking at the words and article titles associated with each centroid. Feel free to visit Wikipedia if an article's content is unclear from its title. On the basis of your tests, select a final value of k and run k-means again. Give a short description of the topics your clustering discovered along with the 5 most common words from each topic. If the topics do not make sense pick another value of k.

Comments:

1) When you run the k-means implementation in `sklearn.cluster.KMeans` it initializes the centroids by randomly assigning the data points to k groups and taking the k representatives as the means of the groups. (This means that if you run the function twice, with the same data, you might get different results.) The cluster centers and labels can be accessed via the attributes `cluster_centers_` and `labels_`. The attribute `labels_` contains the index of each vector's closest centroid (labels start from zero), so if the 30th entry in `labels` is 7, then the 30th vector's closest centroid is the 7th entry in `centroids` (indexing starts from zero).

2) There are many ways to explore your results. For example, you could print the titles of selected articles in a cluster. Alternatively, you could find a topic's most common words by ordering `dictionary` by the size of its centroid's entries. A larger entry for a word implies it was more common in articles from that topic.

[45]:
```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
```

```python
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import pandas as pd

data = np.load('./../homework05/wikipedia_corpus.npz', allow_pickle=True)

dictionary = data["dictionary"]
article_titles = data["article_titles"]
article_histograms = data["article_histograms"] #Data matrix




k = 18 #number of clusters
max_words = 5

kmeans = KMeans(n_clusters = k,algorithm ="full", random_state = 42, max_iter =␣
↪2000)
kmeans.fit(article_histograms)
centers = kmeans.cluster_centers_
labels = kmeans.labels_

wordsInCluster = []
importanceIndex = []
#get the 5 top words for each cluster
for i in range(k):
    importance_sorted = np.argsort(centers[i])[::-1][:max_words]
    importanceIndex.append(importance_sorted)
    words = [dictionary[j] for j in importance_sorted]
    wordsInCluster.append(words)



#Let's find the titles belonging to each cluster, I will do this by
#creating a dictionary and placing each cluster with a label
titlesInCluster = {}
for i in range(len(labels)):
    if labels[i] in titlesInCluster:
        titlesInCluster[labels[i]].append(article_titles[i])
    else:
        titlesInCluster[labels[i]] = [article_titles[i]]


#finally, let's see if titles and words match with each cluster
for i in range(len(wordsInCluster)):
```

```
    #print titles in cluster i
    print("titles in cluster: " + str(i+1) + '\n')
    if i in titlesInCluster:
        for j in titlesInCluster[i]:
            print(str(j))
    else:
        print("No titles in cluster", str(i+1) + '\n')
    print('\n')


    #print top 5 words in cluster i
    print("Top 5 words in cluster" + str(i+1) +'\n')
    print(wordsInCluster[i])
    print('\n')

#Deleting data to preserve memory space

del data
del dictionary
del article_titles
del article_histograms

del kmeans
del centers
del labels

del wordsInCluster
del importanceIndex
```

```
titles in cluster: 1

Anemometer
High-pressure area
Jet stream
Low-pressure area
Sea breeze
Solar wind
Thunderstorm
Tornado
Wind chill
Wind direction
Windsock
Wind speed


Top 5 words in cluster1
```

['wind', 'pressure', 'air', 'direction', 'speed']

titles in cluster: 2

Amplitude modulation
Amplitude-shift keying
Analog signal
Carrier signal
Channel (communications)
Data transmission
Digital signal
Frequency-division multiplexing
Frequency modulation
Frequency-shift keying
Modulation
Multiplexing
NTSC
Phase-shift keying
Quadrature amplitude modulation
Time-division multiplexing

Top 5 words in cluster2

['signal', 'modulation', 'carrier', 'digital', 'frequency']

titles in cluster: 3

Brock (Pokémon)
Bulbasaur
Deoxys
Eevee
Game Boy line
Gameplay of Pokémon
Hey You, Pikachu!
Lapras
List of Pokémon characters
Meowth
Mew (Pokémon)
Mewtwo
Misty (Pokémon)
Nintendo
Pikachu
Pokémon
Pokémon (anime)
Pokémon Black and White

Pokémon Diamond and Pearl
Pokémon Emerald
Pokémon FireRed and LeafGreen
Pokémon Gold and Silver
Pokémon HeartGold and SoulSilver
Pokémon Omega Ruby and Alpha Sapphire
Pokémon Platinum
Pokémon Red and Blue
Pokémon Ruby and Sapphire
Pokémon Trading Card Game
Pokémon universe
Pokémon Yellow
Pokémon 4Ever
Satoshi Tajiri
Team Rocket (anime)
Togepi
Zapdos

Top 5 words in cluster3

['pokemon', 'game', 'games', 'nintendo', 'player']

titles in cluster: 4

Alfred Sisley
Armand Guillaumin
Art Institute of Chicago
Berthe Morisot
Café Guerbois
Camille Pissarro
Claude Monet
Dutch Golden Age painting
Edgar Degas
Effets de soir
Eugène Boudin
Fauvism
Félix Fénéon
Gustave Caillebotte
Impressionism
Impressionism in music
Jan Steen
Johan Jongkind
Louis Leroy
Luncheon of the Boating Party
Macchiaioli
Mary Cassatt

```
Neo-impressionism
Paul Cézanne
Paul Signac
Pierre-Auguste Renoir
Post-Impressionism
Still life
The Child's Bath


Top 5 words in cluster4

['art', 'paintings', 'impressionism', 'painting', 'artists']


titles in cluster: 5

Albedo
Attenuation
Corona
Dew point
Dust storm
Effect of sun angle on climate
Hygrometer
Nephelometer
Plant
Pyranometer
Solarimeter
Stevenson screen
Sunlight
Temperature
Thermo-hygrograph
Thermometer
Ultraviolet


Top 5 words in cluster5

['temperature', 'radiation', 'solar', 'humidity', 'sun']


titles in cluster: 6

Extratropical cyclone
Famine
Flood
Hurricane Katrina
Natural disaster
Severe weather
```

```
Tropical cyclone
Typhoon


Top 5 words in cluster6

['tropical', 'cyclone', 'cyclones', 'storm', 'hurricane']


titles in cluster: 7

International Labour Organization
Kantō region
League of Nations
Paris Peace Conference, 1919


Top 5 words in cluster7

['league', 'labour', 'japan', 'war', 'conference']


titles in cluster: 8

Acid rain
Climate
Desertification
Dynamic Host Configuration Protocol
Erosion
Greenhouse gas
Human impact on the environment
Milankovitch cycles
United Nations Framework Convention on Climate Change


Top 5 words in cluster8

['climate', 'soil', 'emissions', 'erosion', 'greenhouse']


titles in cluster: 9

International Court of Justice
Organization for Security and Co-operation in Europe
Organization of American States
Secretary-General of the United Nations
United Nations
United Nations Charter
```

United Nations Economic and Social Council
United Nations General Assembly
United Nations Security Council
United Nations Trusteeship Council
World Tourism Organization

Top 5 words in cluster9

['council', 'nations', 'security', 'assembly', 'general']

titles in cluster: 10

Antenna (radio)
Audio power amplifier
Broadcasting
Communications satellite
Duplex (telecommunications)
Electrical telegraph
Fiber-optic communication
Flag signals
Guglielmo Marconi
Handset
Heliograph
Lightning detection
Microwave transmission
Optical communication
Optical fiber
Point-to-point (telecommunications)
Radio
Radio wave
Receiver (radio)
Repeater
Semaphore line
Smoke signal
Transmission medium
Transmitter
Wireless

Top 5 words in cluster10

['radio', 'signal', 'signals', 'telegraph', 'communication']

titles in cluster: 11

Anticyclone
Atmosphere
Atmosphere of Earth
Atmospheric pressure
Baroclinity
Barograph
Barometer
Chaos theory
Contrast effect
Coriolis effect
Dark adaptor goggles
Disdrometer
Field mill
Frontogenesis
Hadley cell
Lake
Lidar
Mesoscale meteorology
Meteorology
Microscale meteorology
Middle latitudes
Monsoon
Numerical weather prediction
Radiosonde
Rain gauge
Sounding rocket
Space weather
Sunshine recorder
Synoptic scale meteorology
Troposphere
Weather balloon
Weather forecasting
Weather front
Weather map
Weather modification
Weather radar

Top 5 words in cluster11

['weather', 'pressure', 'atmosphere', 'air', 'temperature']

titles in cluster: 12

Convention on the Rights of Persons with Disabilities
International Centre for Settlement of Investment Disputes
International Civil Aviation Organization

```
International Maritime Organization
International Seabed Authority
International Tribunal for the Law of the Sea
Organisation for the Prohibition of Chemical Weapons
United Nations Convention on the Law of the Sea
United Nations Convention to Combat Desertification
United Nations Office on Drugs and Crime
World Intellectual Property Organization


Top 5 words in cluster12

['convention', 'international', 'nations', 'parties', 'member']


titles in cluster: 13

A Bar at the Folies-Bergère
A Sunday Afternoon on the Island of La Grande Jatte
Bal du moulin de la Galette
Complementary colors
Cubism
Édouard Manet
En plein air
Frédéric Bazille
Georges Seurat
Impasto
Impression, Sunrise
L'Absinthe
Landscape painting
Paris Street; Rainy Day
Portrait
Salon des Refusés
Wet-on-wet
Woman with a Parasol - Madame Monet and Her Son


Top 5 words in cluster13

['painting', 'manet', 'art', 'paris', 'monet']


titles in cluster: 14

Headquarters of the United Nations
International Refugee Organisation
Joint United Nations Programme on HIV/AIDS
Office of the United Nations High Commissioner for Human Rights
```

```
Outline of the United Nations
United Nations High Commissioner for Refugees
United Nations Human Settlements Programme
United Nations Office for Project Services
United Nations Secretariat
United Nations System
United Nations System Staff College
United Nations University
UN Women


Top 5 words in cluster14

['nations', 'general', 'assembly', 'secretary', 'international']


titles in cluster: 15

Ceiling balloon
Ceiling projector
Ceilometer
Cloud
Fog
Nephoscope


Top 5 words in cluster15

['clouds', 'cloud', 'fog', 'height', 'light']


titles in cluster: 16

Black ice
Freezing rain
Frost
Ice
Ice accretion indicator
Ice pellets
Little Ice Age
Precipitation
Winter storm


Top 5 words in cluster16

['ice', 'freezing', 'frost', 'snow', 'rain']
```

titles in cluster: 17

Food and Agriculture Organization
International Bank for Reconstruction and Development
International Development Association
International Finance Corporation
International Fund for Agricultural Development
International Monetary Fund
International Organization for Migration
International Telecommunication Union
International Trade Centre
Multilateral Investment Guarantee Agency
UNESCO
UNICEF
United Nations Capital Development Fund
United Nations Conference on Trade and Development
United Nations Development Programme
United Nations Environment Programme
United Nations Industrial Development Organization
United Nations International Strategy for Disaster Reduction
United Nations Population Fund
United Nations Relief and Works Agency for Palestine Refugees in the Near East
United Nations Volunteers
Universal Postal Union
World Bank Group
World Food Programme
World Health Organization
World Meteorological Organization
World Trade Organization

Top 5 words in cluster17

['countries', 'international', 'nations', 'member', 'bank']

titles in cluster: 18

Asynchronous Transfer Mode
GSM
Internet access
Internet protocol suite
Mobile phone
Multiprotocol Label Switching
Public switched telephone network
Telecommunications network
Telephone

```
Telephone exchange
Telepresence
Voice over IP
W-CDMA (UMTS)
Wide area network


Top 5 words in cluster18

['network', 'telephone', 'networks', 'ip', 'mobile']
```

0.1.3 2. [Recommender systems, 70 points]

Here you will build a recommendation system for movie ratings, using data from the MovieLens web site. We start with a ‘’small” data set with approx. 90,000 ratings of 3650 movies from 610 users and then move on to a larger set with one million ratings from 6000 users on 4000 movies.

Each user typically only rates a few movies but we want to be able to predict ratings for all the movies the user hasn't rated. We do this by basing predictions for the unseen movies on ratings from all the other users using a method based on matrix factorization. We construct a rank-k $n \times m$ user-movies matrix R of movie ratings where r_{ui} corresponds to the rating that user u would give movie i. The prediction $\hat{r}_{ui}$ is given by

$$\hat{r}_{ui} = w_u^T h_i$$

where $w_1^T, \ldots, w_n^T$ are row vectors with k elements and $h_1, \ldots, h_m$ are column vectors with k elements (see the article referenced below for an interpretation of these vectors).

The task is to "learn" the elements of the w and h vectors from the available ratings. This is done by minimizing the least squares error,

$$\sum_{(u,i)\in Z} (r_{ui} - w_u^T h_i)^2 + \lambda(\sum_{u=1}^{n} ||w_u||^2 + \sum_{i=1}^{m} ||h_i||^2)$$

where Z is the set of available ratings (the training set) and $\lambda > 0$ is a regularization parameter that is used to avoid overfitting.

Comments:

1) The MovieLens datasets are taken from https://grouplens.org/datasets/movielens/
2) Start with the `ml-latest-small` data set. Once your code is working, you may want to switch to the `ml-1m` data set to obtain a more accurate model.
3) Code for reading the MovieLens data and performing some cleanup, is given below.
4) The rank k is user defined. For the NetFlix dataset, a value of $k = 40$ worked well (feel free to experiment).
5) The recommendation systems studied here are based on an article by the winners of the NetFlix prize in 2009. https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf

6) Minimizing the mean square error does not necessarily translate to better business

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.96&rep=rep1&type=pdf

a) [Baseline model, 20 points] Start by exploring the data briefly, e.g. by looking at the number of ratings behind each movie and the number of ratings per user (histograms are useful here).

It is by no means guaranteed that a fancy machine learning model performs better than a simple model in the real world. Here we construct a simple baseline model which we use to gauge the quality of the matrix factorization model below. The baseline model is

$$r_{ui} = \mu + c_u + d_i$$

where $\mu \in \mathbb{R}$ is the average rating over all movies, $c \in \mathbb{R}^n$ is a vector representing the deviation of individual users from the average. If e.g. $c_u = -0.5$ then user u tends to rate films 0.5 lower than the average. Element i of the vector $d \in \mathbb{R}^m$ represents the deviation of film i from the average. A positive d_i indicates that movie i is better than an average movie.

Estimate μ, c and d from the ratings data using least squares, i.e. by minimizing

$$\sum_{(u,i)\in Z} (r_{ui} - \mu - c_u - d_i)^2.$$

This can be done by solving a standard least squares problem on the form $Ax \approx b$. The vector b contains the movie ratings, the vector $x = (c, d, \mu)$ is an $n + m + 1$ vector of unknowns. If rating j is (u, i, r) then $b_j = r$ and row j of A is as follows. All the elements of row j are zero except $A[j, u-1] = 1$, $A[j, n+i-1] = 1$ and $A[j, n+m] = 1$ (adjustments for zero-based indexing in numpy). The least squares problem is most conveniently solved using 'scipy.sparse.lsqr' after constructing the matrix with 'scipy.sparse.lil_matrix' and/or 'scipy.sparse.csc_matrix'. It is also possible to use stochastic gradient descent to obtain the parameter values.

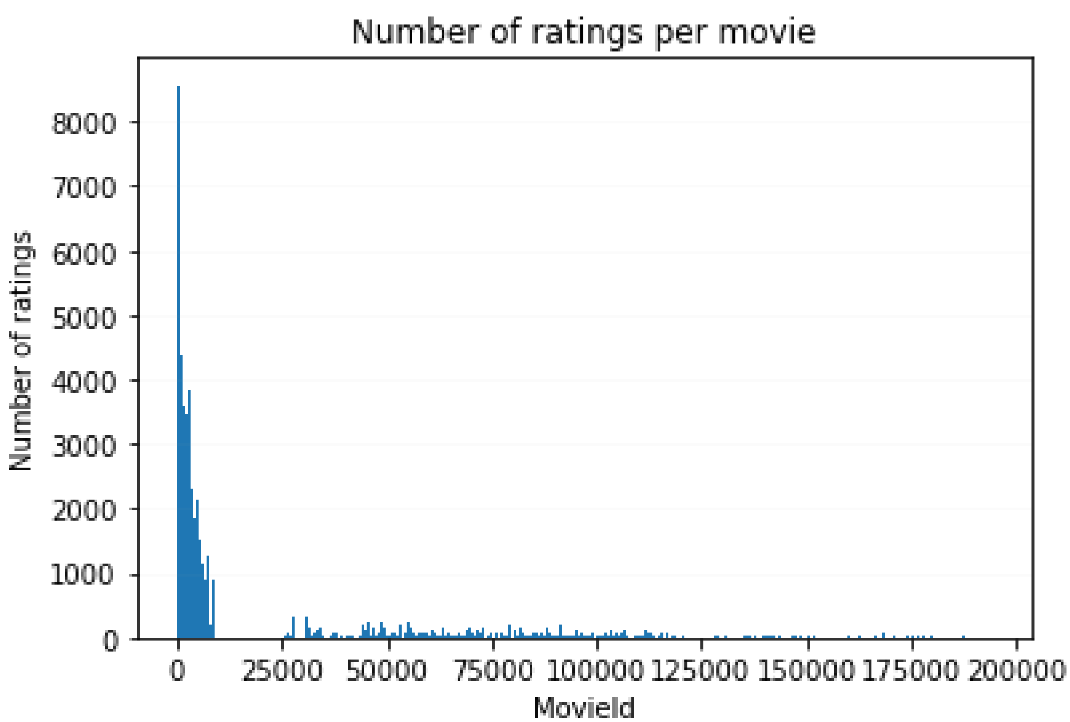
When you have obtained estimates of the model parameters, use the model to compute the *root mean square error* (RMSE) on the test set. Report the error, μ and the first 10 elements of both the c and d vectors.

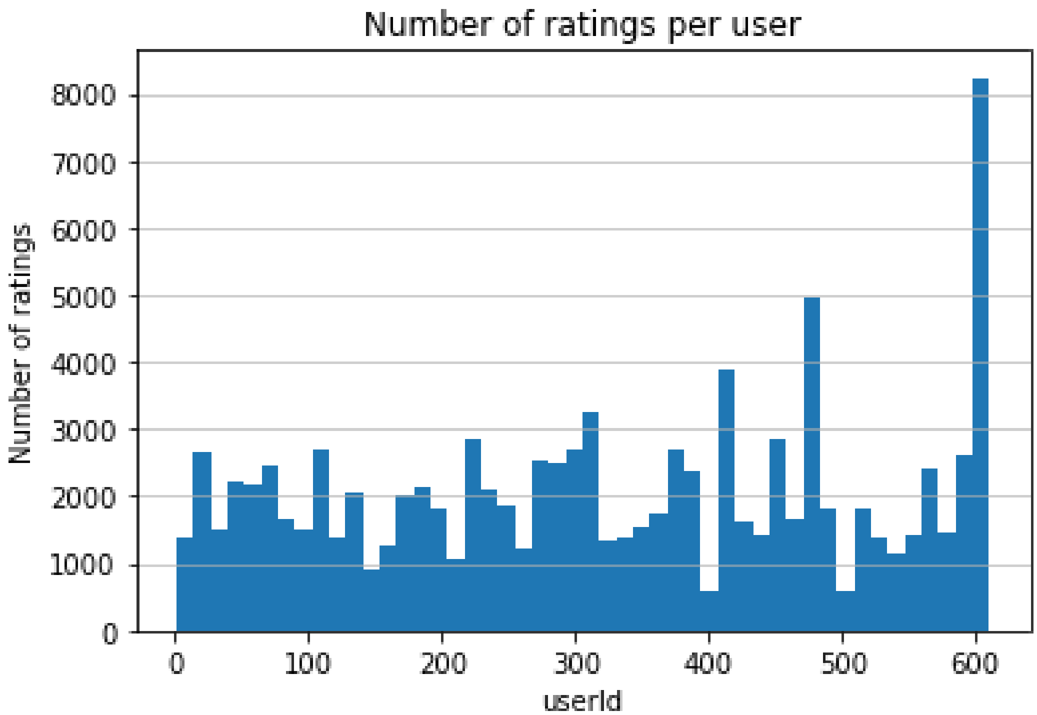
[46]:
```
# a)
#first we will import the small data.
import operator
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.sparse.linalg import lsqr
from scipy.sparse import csc_matrix




path = 'ml-latest-small' # 100K ratings
#path = 'ml-1m' # 1 million
df_ratings = pd.read_csv(path + '/ratings.csv')
print(df_ratings.head())
userId = df_ratings['userId']
movieId = df_ratings['movieId']
```

```
rating = df_ratings['rating']

print(rating)

userUnique = np.unique(userId)

#number of ratings per movie
hist = plt.hist(movieId, bins='auto')
plt.grid(axis='y', alpha=0.05)
plt.ylabel('Number of ratings')
plt.xlabel('MovieId')
plt.title('Number of ratings per movie')
plt.show()

#number of ratings per user
hist = plt.hist(userId, bins='auto')
plt.grid(axis='y', alpha=0.75)
plt.ylabel('Number of ratings')
plt.xlabel('userId')
plt.title('Number of ratings per user')
plt.show()


#let's find average ratings over all movies
avg = sum(rating)/len(rating)

#let's find the c value
userIdUnique, countRatings = np.unique(userId, return_counts=True) #counts of␣
↪rating for each user
avgUsers = np.ones(len(userIdUnique)) #avg rating per user
c = np.ones(len(userIdUnique)) #c value

#first we will create a vector with avg rating per user
j = 0
for i in range(0,len(rating),countRatings[j]):
    temp = rating[i:i+countRatings[j]]
    avgUser = sum(temp)/len(temp)
    avgUsers[j] = avgUser
    j = j+1

#now we can find the c value

for i in range(len(avgUsers)):
    c[i] = "{:.2f}".format(avgUsers[i] - avg)


#let's find the d value
```

```python
#since the matrix is sorted by user Id I need to find a way to sort it by movie␣
↪Id
#for that I will create a dictionary

movieDict = {}
movieRating = {}

print(len(movieRating))

for i in range(len(movieId)):
    if movieId[i] in movieDict:
        movieDict[movieId[i]].append(rating[i])
    else:
        movieDict[movieId[i]] = [rating[i]]

#now we have all ratings for each movie, let's create movieRating
for i in range(max(movieId)):
    if i in movieDict:
        movieRating[i] = sum(movieDict[i])/len(movieDict[i])


d = np.ones(len(movieRating))
#now we can create d
j = 0
for i in movieRating:
    d[j] = movieRating[i] - avg
    j += 1


#now let's create the model
model = np.ones(len(c)*len(d)).reshape(len(c),len(d)) #where It's (number of␣
↪users)x(number of movies)
print(model.shape)
for i in range(len(c)):
    for j in range(len(d)):
        model[i,j] = 1*(avg + c[i] + d[j])





del movieId
del userId
del rating
```

```
   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
0         4.0
1         4.0
2         4.0
3         5.0
4         5.0
         …
100831    4.0
100832    5.0
100833    5.0
100834    5.0
100835    3.0
Name: rating, Length: 100836, dtype: float64
```






```
0
(610, 9723)
```

[47]:
```
from scipy.sparse.linalg import lsqr
from scipy.sparse import csc_matrix



r = np.ones(len(movieRating))
j = 0
for i in movieRating:
    r[j] = movieRating[i]
    j += 1
r = np.array(r)
A = csc_matrix(model.T, dtype=float)

print(len(r))
print(A.shape)
print(len(movieRating))
#It is possible that I need to flip A

x, istop, itn, normr = lsqr(A, r)[:4]

print("the value of x in Ax = b is: \n",x)
```

```
#now we can put x in the form Ax = b

#I was having problem running this where my computer kept freezing,
#Even for the smaller data.

#b = np.dot(A,x)
#print(b)
```

```
9723
(9723, 610)
9723
the value of x in Ax = b is:
 [0.00291792 0.00258389 0.0021195  0.0022743  0.00197285 0.00229059
 0.00220912 0.0023965  0.00263277 0.00238021 0.00138626 0.001427
 0.00138626 0.00253501 0.00212765 0.001981   0.00146773 0.00209506
 0.00200544 0.00185065 0.00238836 0.0025513  0.00220912 0.00204618
 0.00228245 0.001981   0.00226615 0.00255945 0.0023965  0.00250242
 0.00187509 0.00177732 0.00254315 0.0022743  0.00315419 0.00224171
 0.00200544 0.00282015 0.00275498 0.00264092 0.00234762 0.00244539
 0.00234762 0.00243724 0.00215209 0.00177732 0.00181806 0.00183435
 0.00217653 0.00234762 0.00260018 0.00242095 0.00266536 0.0021195
 0.00260833 0.00207062 0.00233947 0.00243724 0.00245353 0.00191583
 0.00274683 0.00212765 0.00209506 0.00253501 0.00209506 0.0025513
 0.00260018 0.00251871 0.00251871 0.00209506 0.0028446  0.00257574
 0.00286904 0.00259204 0.00209506 0.00183435 0.00200544 0.00204618
 0.0021358  0.00207877 0.00286089 0.00293421 0.00288533 0.00250242
 0.00233947 0.0025513  0.00163068 0.00205433 0.00221727 0.00202174
 0.00189138 0.00218468 0.00223356 0.00216024 0.002258   0.00230689
 0.00205433 0.00130479 0.00262462 0.00242909 0.00186694 0.001427
 0.00255945 0.00244539 0.00266536 0.00289348 0.00204618 0.0025513
 0.00205433 0.00200544 0.00224171 0.0025513  0.00192397 0.00234762
 0.00224171 0.00206247 0.00219283 0.00230689 0.00250242 0.00232318
 0.00255945 0.00235577 0.00260018 0.00186694 0.00234762 0.00229059
 0.00242095 0.00272239 0.00247798 0.00262462 0.00259204 0.00238021
 0.00251056 0.00192397 0.00163882 0.00161438 0.0019973  0.00171215
 0.00260018 0.00236392 0.0023965  0.00189138 0.00265721 0.00220097
 0.00220097 0.00249427 0.00246168 0.0021195  0.00174474 0.00196471
 0.00216024 0.00216024 0.00264092 0.00277942 0.00269795 0.00260833
 0.00233133 0.00255945 0.00222542 0.00238836 0.00243724 0.00253501
 0.00265721 0.00242909 0.00240465 0.00233947 0.00207877 0.0021358
 0.00220912 0.00237206 0.00209506 0.00196471 0.00174474 0.00186694
 0.00211136 0.0026898  0.00251056 0.00243724 0.00251871 0.00265721
 0.00252686 0.00181806 0.001981   0.00163882 0.00197285 0.00212765
 0.00270609 0.00260833 0.00162253 0.00195656 0.00167141 0.00162253
 0.0018588  0.00123961 0.00123147 0.00100335 0.00235577 0.0025513
 0.00268165 0.00254315 0.00238836 0.00172029 0.00187509 0.00143515
```

0.00132923	0.00164697	0.00180991	0.00224986	0.00222542	0.00200544
0.00201359	0.00238836	0.00237206	0.00242095	0.00246168	0.00231503
0.00234762	0.00218468	0.00216839	0.00185065	0.00242909	0.00242095
0.00226615	0.00212765	0.00224171	0.00245353	0.00200544	0.00259204
0.00278757	0.00246168	0.00240465	0.00254315	0.00269795	0.00259204
0.0025513	0.00194841	0.00295051	0.00167141	0.00222542	0.00189138
0.00153291	0.00216024	0.00189953	0.00201359	0.00251056	0.00247798
0.00236392	0.00216024	0.00228245	0.00262462	0.00250242	0.00232318
0.00221727	0.00210321	0.00194027	0.00174474	0.00264092	0.00187509
0.00167141	0.00204618	0.00249427	0.00220912	0.00226615	0.00216024
0.00233133	0.00266536	0.00282015	0.00231503	0.00273868	0.00210321
0.00210321	0.002258	0.00194027	0.00205433	0.00201359	0.00209506
0.002258	0.00202988	0.00205433	0.00249427	0.00277942	0.0023965
0.00270609	0.00202988	0.00233133	0.00234762	0.0014107	0.00232318
0.00238836	0.00245353	0.00283645	0.00222542	0.00219283	0.00220097
0.00253501	0.00242909	0.00196471	0.00180991	0.00221727	0.00165512
0.00176918	0.00137812	0.00136997	0.00118259	0.0026898	0.00295866
0.00233133	0.00233947	0.00247798	0.00198915	0.00237206	0.00260833
0.00237206	0.00229874	0.00245353	0.00217653	0.00230689	0.00198915
0.001704	0.00189138	0.00194841	0.00220912	0.00222542	0.00210321
0.00269795	0.00233133	0.00228245	0.00246168	0.00216024	0.00224171
0.00189138	0.00198915	0.0023965	0.00216839	0.00220097	0.00248612
0.00255945	0.00247798	0.0019973	0.00164697	0.00169585	0.00188324
0.00254315	0.00245353	0.00247798	0.00216839	0.00223356	0.00195656
0.00221727	0.00274683	0.00197285	0.00176918	0.00132923	0.00232318
0.00240465	0.0025513	0.00229059	0.00207877	0.00233947	0.00272239
0.00274683	0.00248612	0.00236392	0.00287718	0.00246983	0.00230689
0.00189138	0.0024128	0.00230689	0.00209506	0.00221727	0.00232318
0.00224986	0.00216839	0.00229059	0.00211136	0.00249427	0.0021195
0.00244539	0.0021195	0.00215209	0.00222542	0.00247798	0.00268165
0.00238836	0.00219283	0.0025513	0.00273868	0.00285274	0.00268165
0.00224986	0.00192397	0.00180991	0.00202174	0.0021195	0.00251056
0.00189953	0.00245353	0.0023965	0.00258389	0.00133738	0.00229059
0.00158179	0.00171215	0.0012885	0.001704	0.00169585	0.00141885
0.00157365	0.00152476	0.00160623	0.00201359	0.001704	0.00226615
0.00216024	0.00279571	0.0022743	0.00216839	0.00212765	0.00192397
0.00232318	0.00224986	0.00230689	0.00236392	0.00233947	0.00203803
0.00183435	0.00191583	0.00228245	0.00268165	0.00215209	0.00234762
0.00221727	0.00233947	0.00251871	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049
0.00018049	0.00018049	0.00018049	0.00018049	0.00018049	0.00018049

```
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049 0.00018049 0.00018049
0.00018049 0.00018049 0.00018049 0.00018049]
```

b) [Matrix factorization model, 20 points] Use stochastic gradient descent to minimize

$$\sum_{(u,i)\in Z} (r_{ui} - w_u^T h_i)^2 + \lambda(\sum_{u=1}^{n} ||w_u||^2 + \sum_{i=1}^{m} ||h_i||^2)$$

with e.g. $k = 20$. You can start with e.g. $\lambda = 0.01$ and step-size $\alpha = 0.05$ (some adjustments may be needed). To initialize the w_u and h_i vectors use normally distributed random values, e.g. with mean zero and standard deviation 0.02. You can perform the updates on the w_u and h_i vectors separately.

Monitor the root mean square error on both the training set and the test set. Keep track of the best set of parameter values (w and h vectors) and stop training when the test error starts to increase.

Report the root mean square error on the test set for the best parameters. How does this model compare to the one you found in a) in terms of RMSE?

Comment: You can generate normally distributed random variables with `np.random.randn`.

[10]:
```python
# Insert code here
# ...
```

c) [Matrix factorization model with bias, 10 points] Expand your model from b) by adding user-movie bias on the form

$$b_{ui} = \mu + c_u + d_i.$$

where μ is the global average of the ratings,

$$\mu = \sum_{(u,i)\in Z} r_{ui}/|Z|$$

(fixed throughout the iterations) but the vectors c and d are estimated along with the w and h vectors. The predictive model becomes

$$\hat{r}_{ui} = \mu + c_u + d_i + w_u^T h_i$$

and the least squares error criteria

$$\sum_{(u,i)\in Z} (r_{ui} - \mu - c_u - d_i - w_u^T h_i)^2 + \lambda(\sum_{u=1}^{n} ||w_u||^2 + \sum_{i=1}^{m} ||h_i||^2 + ||c||^2 + ||d||^2)$$

Test different values of k and λ. Report the RMSE on the test set for the best model that you obtain. How does it compare to the models from a) and b)?

[11]:
```
# Insert code here
# ...
```

d) [Model evaluation in the real world - open ended, 10 points] Use the best model from a) - c) to generate movie recommendations for a user that reflects your own taste in movies, or a user that has strong preference for particular genre(s) (e.g. a horror fan). Many recommender systems suffer ‘’popularity bias”, i.e. they tend to focus on popular items. Does your model have this tendency? Discuss briefly. Do you think that your model is useful in the real world?

Popularity bias in recomender systems and evaluation metrics are discussed in some detail here: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.465.96&rep=rep1&type=pdf

[4]:
```
# Insert code here
# ...
```

The code below is for reading the MovieLens data. You can place it wherever you want in this workbook or hide it away in a function that you import.

[5]:
```
# Read and preprocess the MovieLens data

import numpy as np
import pandas as pd

path = 'ml-latest-small' # 100K ratings
#path = 'ml-1m' # 1 million
df_ratings = pd.read_csv(path + '/ratings.csv')
print("Number of rows in original ratings matrix:", df_ratings.shape[0])

# Remove movies and users with few reviews
min_ratings = 10 # (feel free to change this value)
df_ratings = df_ratings.groupby('movieId').filter(lambda x : len(x) >=␣
↪min_ratings)
print("Number of rows in ratings matrix after removing movies with few ratings:
↪", df_ratings.shape[0])
df_ratings = df_ratings.groupby('userId').filter(lambda x : len(x) >=␣
↪min_ratings)
```

```
print("Number of rows in ratings matrix after removing users with few reviews:
 ↪", df_ratings.shape[0])
print("Number of users:", len(df_ratings['userId'].unique()))
print("Number of movies:", len(df_ratings['movieId'].unique()))

# THINK: Koren et al. used the timestamp to good effect
df_ratings.drop('timestamp', axis=1, inplace=True)
df_ratings.head()
```

```
Number of rows in original ratings matrix: 100836
Number of rows in ratings matrix after removing movies with few ratings: 81116
Number of rows in ratings matrix after removing users with few reviews: 81109
Number of users: 609
Number of movies: 2269
```

[5]:

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0

[6]:

```
def convert_ids(values):
    # Convert Ids to consecutive integers
    newIds = {}
    count = 1
    for i in values:
        if i not in newIds:
            newIds[i] = count
            count += 1
    inv_Ids = dict(map(reversed, newIds.items()))
    return newIds, inv_Ids

# Read movie descriptions from file
df_movies = pd.read_csv(path + '/movies.csv')

print("Number of movies read from file:", df_movies.shape[0])
df_movies = df_movies[df_movies['movieId'].isin(df_ratings['movieId'].values)]
print("After filtering:", df_movies.shape[0])

# Make sure that userIds and movieIds are consecutive integers
userIds, inv_userIds = convert_ids(df_ratings['userId'].values)
df_ratings['userId'] = df_ratings['userId'].apply(lambda x: userIds[x])

movieIds, inv_movieIds = convert_ids(df_ratings['movieId'].values)
df_ratings['movieId'] = df_ratings['movieId'].apply(lambda x: movieIds[x])
df_movies['movieId'] = df_movies['movieId'].apply(lambda x: movieIds[x])
```

```
print("Number of users:", len(df_ratings['userId'].unique()))
print("Number of movies:", len(df_ratings['movieId'].unique()))

df_movies.head()
```

```
Number of movies read from file: 9742
After filtering: 2269
Number of users: 609
Number of movies: 2269
```

[6]:

```
   movieId                              title  \
0        1                   Toy Story (1995)
1      406                     Jumanji (1995)
2        2            Grumpier Old Men (1995)
4      407  Father of the Bride Part II (1995)
5        3                        Heat (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
4                                       Comedy
5                        Action|Crime|Thriller
```

[7]:

```
# Create random train/test split
from sklearn.model_selection import train_test_split

df_train, df_test = train_test_split(df_ratings, test_size=0.2,␣
 ↪random_state=42) # 10% might be sufficient
ratings_train = df_train[['userId','movieId','rating']].values.astype(np.int32)
ratings_test = df_test[['userId','movieId','rating']].values.astype(np.int32)
print("Training set size: ", ratings_train.shape[0])
print("Test set size: ", ratings_test.shape[0])
```

```
Training set size:  64887
Test set size:  16222
```

[ ]:

[ ]: