Intelligent Data Mining - Exercise 7

Michael Debono Mrđen

16 January 2018

1 Assignment 1: Content-based recommendation

Three computers, A, B, and C, have the numerical features listed below:

Feature	A	B	C
Processor Speed	3.06	2.68	2.92
Disk Size	500	320	640
Main-Memory Size	6	4	6

We may imagine these values as defining a vector for each computer; for instance, A's vector is [3.06,500,6]. We can compute the cosine distance between any two of the vectors, but if we do not scale the components, then the disk size will dominate and make differences in the other components essentially invisible. Let us use 1 as the scale factor for processor speed, α for the disk size, and β for the main memory size.

a. In terms of α and β, compute the cosines of the angles between the vectors for each pair of the three computers.

- A $= [3.06, 500\alpha, 6\beta]$
- B $= [2.68, 320\alpha, 4\beta]$
- C $= [2.92, 640\alpha, 6\beta]$

$$\cos(AB) = \frac{(3.06)(2.68) + (500\alpha)(320\alpha) + (6\beta)(4\beta)}{\sqrt{3.06^2 + (500\alpha)^2 + (6\beta)^2}\sqrt{2.68^2 + (320\alpha)^2 + (4\beta)^2}}$$

$$\cos(AC) = \frac{(3.06)(2.92) + (500\alpha)(640\alpha) + (6\beta)(6\beta)}{\sqrt{3.06^2 + (500\alpha)^2 + (6\beta)^2}\sqrt{2.92^2 + (640\alpha)^2 + (6\beta)^2}}$$

$$\cos(CB) = \frac{(2.92)(2.68) + (640\alpha)(320\alpha) + (6\beta)(4\beta)}{\sqrt{2.92^2 + (640\alpha)^2 + (6\beta)^2}\sqrt{2.68^2 + (320\alpha)^2 + (4\beta)^2}}$$

b. What are the angles between the vectors if $\alpha = \beta = 1$?

	cos	$\angle$
AB	0.999997	0.1403°
AC	0.9999953431	0.1749°
CB	0.9999878534	0.2824°

c. What are the angles between the vectors if $\alpha = 0.01$ and $\beta = 0.5$?

	cos	$\angle$
AB	0.990882	7.743°
AC	0.991555	7.451°
CB	0.969178	14.26°

d. A certain user has rated the three computers as follows: A: 4 stars, B: 2 stars, C: 5 stars. Normalize the ratings for this user.

Average rating:

$$\frac{4+2+5}{3} = \frac{11}{3}$$

Normalized ratings:

A	B	C
1/3	−5/3	4/3

e. Compute a user profile for the user, with components for processor speed, disk size, and main memory size.

- Processor Speed = $[3.06, 2.68, 2.92]$
- Disk Size = $[500, 320, 640]$
- Main-Memory Size = $[6, 4, 6]$

Processor Speed component:

$$\frac{(1/3)(3.06) + (-5/3)(2.68) + (4/3)(2.92)}{3} = 0.1489$$

Disk Size component:

$$\frac{(1/3)(500) + (-5/3)(320) + (4/3)(640)}{3} = 162.2$$

Main-Memory Size component:

$$\frac{(1/3)(6) + (-5/3)(4) + (4/3)(6)}{3} = 1.111$$

User profile:

$$[0.1489, 162.2, 1.111]$$

2 Assignment 2: Singular Value Decomposition

Use the SVD form Fig. 11.7. Suppose Leslie assigns rating 3 to Alien and rating 4 to Titanic, giving us a representation of Leslie in "movie space" of [0,3,0,0,4]. Find the representation of Leslie in concept space. What does that representation predict about how well Leslie would like the other movies appearing in our example data?

$$\mathbf{q} = [0, 3, 0, 0, 4]$$

$$\mathbf{q}V = [0, 3, 0, 0, 4] \begin{bmatrix} 0.58 & 0 \\ 0.58 & 0 \\ 0.58 & 0 \\ 0 & 0.71 \\ 0 & 0.71 \end{bmatrix}$$

$$\mathbf{q}V = [1.74, 2.84]$$

This shows that Leslie would like romance movies slightly more than science fiction movies.

3 Assignment 3: Singular Value Decomposition with Mahout

Use the data "netflix_data.txt" uploaded in StudIP which contains real Netflix user ratings of 100 different movies. Pick 3 different users of your choice and print out the movie titles of the first 4 recommendations of each of the three following Mahout recommenders:

- SVDRecommender using the SVDPlusPlusFactorizer
- SVDRecommender using the ALSWRFactorizer
- GenericUserBasedRecommender using and appropriate UserSimilarity and UserNeighborhood

Output:

```
SVDRecommender using the SVDPlusPlusFactorizer
Recommended for user 699878:
Maya Lin: A Strong Clear Vision
Rudolph the Red-Nosed Reindeer
I Love Lucy: Season 2
The Bad and the Beautiful
Recommended for user 835810:
I Love Lucy: Season 2
The Bad and the Beautiful
```

```
ABC Primetime: Mel Gibson's The Passion of the Christ
The Love Letter
Recommended for user 1290128:
Screamers
Justice League
Davy Crockett: 50th Anniversary Double Feature
Jade

SVDRecommender using the ALSWRFactorizer
Recommended for user 699878:
Ken Burns' America: Empire of the Air
Elfen Lied
Aqua Teen Hunger Force: Vol. 1
Lord of the Rings: The Return of the King: Extended Edition: Bonus Material
Recommended for user 835810:
Lord of the Rings: The Return of the King: Extended Edition: Bonus Material
Aqua Teen Hunger Force: Vol. 1
ABC Primetime: Mel Gibson's The Passion of the Christ
Invader Zim
Recommended for user 1290128:
Lord of the Rings: The Return of the King: Extended Edition: Bonus Material
Aqua Teen Hunger Force: Vol. 1
Zatoichi's Conspiracy
The Killing

GenericUserBasedRecommender
Recommended for user 699878:
The Battle of Algiers: Bonus Material
Rudolph the Red-Nosed Reindeer
Richard III
What the #$*! Do We Know!?
Recommended for user 835810:
Recommended for user 1290128:
Record of Lodoss War: Chronicles of the Heroic Knight
I Love Lucy: Season 2
Immortal Beloved
Mostly Martha
```

NetflixFileDataModel.java:

```
package com.intelligent.data.management.Exercise7Assignment1;

import java.io.File;
import java.io.IOException;

import org.apache.mahout.cf.taste.impl.model.MemoryIDMigrator;
```

```
public class NetflixFileDataModel extends
↪  org.apache.mahout.cf.taste.impl.model.file.FileDataModel {
        private static final long serialVersionUID =
        ↪  -3337785425912797856L;
        private static MemoryIDMigrator idMigrator = new
        ↪  MemoryIDMigrator();

        public NetflixFileDataModel(File dataFile) throws
        ↪  IOException {
                super(dataFile);
        }

        @Override
        protected long readItemIDFromString(String value) {
            long result = idMigrator.toLongID(value);
            idMigrator.storeMapping(result, value);
            return result;
        }

        public String getMovieNameFromID(long itemID) {
                return idMigrator.toStringID(itemID);
        }
}
```

App.java:

```
package com.intelligent.data.management.Exercise7Assignment1;

import java.io.File;
import java.io.IOException;
import java.util.List;

import org.apache.mahout.cf.taste.common.TasteException;
import
↪  org.apache.mahout.cf.taste.impl.neighborhood.NearestNUserNeighborhood;
import
↪  org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedRecommender;
import
↪  org.apache.mahout.cf.taste.impl.recommender.svd.ALSWRFactorizer;
import
↪  org.apache.mahout.cf.taste.impl.recommender.svd.SVDPlusPlusFactorizer;
import
↪  org.apache.mahout.cf.taste.impl.recommender.svd.SVDRecommender;
import
↪  org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;
```

```
import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.Recommender;
import org.apache.mahout.cf.taste.similarity.UserSimilarity;

public class App
{
    public static void main(String args[]) throws IOException,
    ↪  TasteException {
            NetflixFileDataModel netflixData = new
            ↪  NetflixFileDataModel(new
            ↪  File("data/netflix_data.txt"));
            DataModel data = netflixData;

            long[] users = {699878, 835810, 1290128};

            System.out.println("SVDRecommender using the
            ↪  SVDPlusPlusFactorizer");
            for (long u : users) {
                    System.out.println(String.format("Recommended
                    ↪  for user %d:", u));
                        SVDPlusPlusFactorizer factorizer = new
                        ↪  SVDPlusPlusFactorizer(data, 10, 5);
                        Recommender recommender = new
                        ↪  SVDRecommender(data, factorizer);
                        List<RecommendedItem> recommendedItems =
                        ↪  recommender.recommend(u, 4);
                        for(RecommendedItem r : recommendedItems)
                        ↪  {
                                System.out.println(netflixData.getMovieNameFromID(r.getItem
                        }
            }
            System.out.println();

            System.out.println("SVDRecommender using the
            ↪  ALSWRFactorizer");
            for (long u : users) {
                    System.out.println(String.format("Recommended
                    ↪  for user %d:", u));
                    float lambda = new Float(0.02);
                        ALSWRFactorizer factorizer = new
                        ↪  ALSWRFactorizer(data, 10, lambda, 5);
                        Recommender recommender = new
                        ↪  SVDRecommender(data, factorizer);
                        List<RecommendedItem> recommendedItems =
                        ↪  recommender.recommend(u, 4);
```

```
                    for(RecommendedItem r : recommendedItems)
                    ↪ {
                            System.out.println(netflixData.getMovieNameFromID(r.getItemI
                    }
            }
            System.out.println();

            System.out.println("GenericUserBasedRecommender");
            for (long u : users) {
                    System.out.println(String.format("Recommended
                    ↪ for user %d:", u));
                    UserSimilarity userSimilarity = new
                    ↪ PearsonCorrelationSimilarity(data);
                    UserNeighborhood neighborhood = new
                    ↪ NearestNUserNeighborhood(100,
                    ↪ userSimilarity, data);
                    Recommender recommender = new
                    ↪ GenericUserBasedRecommender(data,
                    ↪ neighborhood, userSimilarity);
                        List<RecommendedItem> recommendedItems =
                        ↪ recommender.recommend(u, 4);
                        for(RecommendedItem r : recommendedItems)
                        ↪ {
                                System.out.println(netflixData.getMovieNameFromID(r.getItemI
                        }
            }
            System.out.println();
    }
}
```