CSCI 485 (Machine Learning)

Sami Al-Qusus

Spring 2019 - Assignment 3
Submit deadline: 7 March 2019, Thursday

Task:
Write a document explaining how to build a similarity-based model to recommend movie(s) to any of the 1000 users, using the following data as an example:

- The data set contains 12,672 ratings that 1000 users gave to 100 movies in twenty days. Each data item is a quadruplet of the form <user, movie, date of grade, grade>. The user and movie fields are integer IDs, the date of grade takes the format of "yyyy-mm-dd", and grades are from 1 to 5 (integral) stars.
- The full data set contains 12,672 instances and can be accessed at (http://csci.viu.ca/~liuh/485/assignments/A3-data.csv).

Intro:
Recommendation systems are used on many products and services like Netflix shows/movies, amazon books, spotify music, google search and youtube videos.

The idea is to use collected data of users past behavior to make predictions and recommend items similar to the ones that a user has liked/purchased/interacted with in the past or to make the recommendation based on that of the closest similar users.

Similarity is measured in the range 0 to 1.

- Similarity = 1 if X = Y
- Similarity = 0 if X ≠ Y

User-User Collaborative filtering:
For this example we will use a similarity-based model that uses a collaborative filtering algorithm. The collaborative filtering algorithm uses "User Behaviour" for recommending items. This algorithm first finds the similarity score between users. Based on this similarity score, it then picks out the most similar users and recommends movies that similar users have rated well previously.

How To Calculate the Similarity:

There are different ways of deciding on similarity. One way is to use distance formula to measure distance between two users. One way is to simply measure the distance between attributes using the common distance formulas, a Minkowski-based metric.

The Minkowski distance between two variabes *X* and *Y* is defined as

$$\left(\sum_{i=1}^{n} |X_i - Y_i|^p\right)^{1/p}$$

The case where *p* = 1 is equivalent to the Manhattan distance and the case where *p* = 2 is equivalent to the Euclidean distance.

Another way is to measure correlation, measures the linear relationship between objects. For this example I will use cosine similarity, it is a similarity measure that finds the similarity by finding the cosine of the angle between each two user's set of movie ratings vector.

The cosine is better because it takes into account the whole picture, as in, instead of deciding on similarity by only measuring the distance between movies they have in common ignoring the fact that the movies they didn't both rate might be way bigger than the ones they both did, rather take the weight of the common movies compared to ones they didn't match on. For example if they both only have 1 movie in common and they both rated it 5 measuring the distance between them we get 0 and that would mean they're exactly similar. But what if they each watched 90 different other movies that they don't have in common, then common sense says they shouldn't be considered exactly similar.

The cosine takes that into consideration by dividing over the length of each vector that way the bigger the length compared to the common movies the smaller the cosine of the angle and the bigger the angle of difference! While the bigger the cosine of the angle, the bigger the similarity since the angle would be small meaning they are very similar in similarity.

The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \cos\theta$$ We get => $$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

- An example of calculating the distance between two users, lets take:
- user 2: <movie12: 1, movie60: 3, movie73: 3, movie86: 3, movie99: 3>
- user 286: <movie12: 2, movie27: 2, movie29: 5, movie44: 5, movie46: 5, movie61: 3, movie63: 3, movie78: 2, movie80: 1, movie95: 3>
- sim(user 2, user 286)= cos(θ)=
- $=(1\times2)/[(\sqrt{1^2+3^2+3^2+3^2+3^2})+(\sqrt{2^2+2^2+5^2+5^2+5^2+3^2+3^2+3^2+2^2+1^2+3^2})]=$
- =2/(6.08+11.14)=2/17.22=0.12
- $\theta = \cos^{-1}(0.12) = 83.10\,°$ <-degrees between user 2 and 286

- This shows that the similarity is closer to 0 than one and therefore the users are not that similar.

✓ An advantage of cosine similarity is its low-complexity, especially for sparse vectors: only the non-zero dimensions need to be considered.

Steps to build the similarity based model from data given:

1. First step is to decide what are the descriptive features and transform the data so a user's descriptive features can be represented in a vector form. We don't need the date its irrelevant, we will assume the users preference didn't change with time and that the similarity between two users doesn't depend on when they rated the movie since it's a very short period of time for this data. Therefore the descriptive features are the movies and their ratings. We transform the data so all the ratings of a user can be represented in a vector form.
   - All users will have the same vector number of elements to hold all movies even ones they didn't rate; the ones they didn't rate will be filled with 0, if rated it will hold be an integer between 1 and 5.
   - To find the max number of elements the vector will hold we will do a SQL run on the data to find the number of unique movies, i.e. vector size.
     - `Select count (distinct movie) from T;`
     - `We get 101`
   - So the data representation should be changed from this format

user	movie	date_of_grade	grade
1	8	2015-10	5
:	:	:	:
1000	98	2015-10-11	4

   To this matrix
   ||
   V

User	Movie:0	Movie: 1	..Movie:8..	..Movie:10..	Movie: 100
1	0	0	5	2	0
:	:	:	:	:	:
1000	0	0	2	0	0

   - That can be done with this algorithm

```
1. Import data
2. Select count (distinct movie) from T;
   //unique movies=101
3. Select count (distinct user) from T;
   //unique users=1000
4. Data_matrix = (1000,101) //create matrix 1000 by 101
5. For each line in original data matrix with same user id
       Put grade in position movie
```

2. Second, we will use the cosine algorithm to compare all the users, then pick neighbors for all the users. The neighbors allow for us to store only the top k

similar neighbors, thus we save space and we can average the neighbors to find a general recommendation instead of just recommending the most similar user. The most similar user alone might not have enough movies that the user didn't watch yet, so it's better to for the recommendation to be based off more than one user. This allows us to have more movies to recommend with other advantages.

- Calculating similarity algorithm:

```
1  For each user create an array of size 1000-1
     //size is (max users - minus user him or herself)
2  For i from 1 to 1000
       For j from 1 to 1000
           If ( i ≠j )
               ▪ sum the multiplication of each
                 element in same column at row i and j
               ▪ then divide it over
                 (square root of summation of all
                 elements in row i squared)
                 plus
                 (square root of summation of all
                 elements in row j squared)
               ▪ put answer in array user_i [j-1]
```

//this should do the cosine similarity between each two users as long as it's not the same user and stores all the similarities in a user array.

```
3  Sort each user array and find biggest 50 numbers those are
   the neighbors.
   //used most similar 5% users to the user, but could've use
   another percentage.
4  For each movie the user didn't watch i.e. there is a 0 in the
   matrix, average out the neighbors ratings
5  Order the ratings from largest to smallest and recommend to
   them to user from largest to smallest.
```

Flaws:

- Supervised machine learning is based on the stationary assumption, which states that the data doesn't change – remains stationary – over time.
- Its not a proper distance metric as it does not have the triangle inequality property or, more formally, the Schwarz inequality and it violates the coincidence axiom; to repair the triangle inequality property while maintaining the same ordering, it is necessary to convert to angular distance as follows:

When the vector elements may be positive or negative:

$$\text{angular distance} = \frac{\cos^{-1}(\text{cosine similarity})}{\pi}$$

$$\text{angular similarity} = 1 - \text{angular distance}$$

Or, if the vector elements are always positive:

$$\text{angular distance} = \frac{2 \cdot \cos^{-1}(\text{cosine similarity})}{\pi}$$

$$\text{angular similarity} = 1 - \text{angular distance}$$

- Treats the blank attribute as 0, 0 means user has disliked the product, although it because it was not rated.
- Have to be careful because cosine similarity is a judgement of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

<u>Possible improvement</u>

- Final step is to compare recommendations with secret testing data.
    - Algorithm:

        1. Search matrix for not watched movies in provided data and where rated in testing data.
        2. For each movie calculate percentage of accuracy
        3. Then average out all accuracies
        4. Result is the overall accuracy of the recommendation
        5. If its less than 10 percent its good enough for this example.

- If overall recommendation is not accurate then use another algorithm and check for accuracy. Go with algorithm that gives best accuracy.
- Another way to improve is maybe to consider including other descriptive features. Can also try play around with weights given for each descriptive feature. As some features have greater say on the recommendation from others. Though as mentioned earlier, because time is very short, it probably wont be an improvement if considered.