University of Athens

BIG DATA MANAGEMENT

Project Two

Ngomba Litombe – sn1199003
6-13-2020

PART ONE: CLUSTERING

Clustering is the process of examining a collection of "points," and grouping the points into "clusters" according to some distance measure. The goal is that points in the same cluster have a small distance from one another, while points in different clusters are at a large distance from one another. "points" here refer to objects belonging to some space.

TYPES

Hierarchical or agglomerative algorithms start with each point in its own cluster. Clusters are combined based on their "closeness," using one of many possible definitions of "close." Combination stops when further combination leads to clusters that are undesirable for one of several reasons. For example, we may stop when we have a predetermined number of clusters, or we may use a measure of compactness for clusters, and refuse to construct a cluster by combining two smaller clusters if the resulting cluster has points that are spread out over too large a region.

This algorithm is quite expensive as it requires several passes through the entire dataset and it is therefore not suitable for very large datasets.

point assignment. Points are considered in some order, and each one is assigned to the cluster into which it best fits. This process is normally preceded by a short phase in which initial clusters are estimated. Variations allow occasional combining or splitting of clusters, or may allow points to be unassigned if they are outliers (points too far from any of the current clusters). The likes of kmeans belong to this category.

NOTION OF DISTANCE

Cosine Similarity:

$$sim(A,B) = \cos(r_A, r_B)$$

Jaccard similarity:

$$sim(A,B) = | r_A \cap r_B | / | r_A \cup r_B |$$

As similarities are not distance metrics, the appropriate distance metric would be: distance = 1 – similarity (which is equivalent to dissimilarity). But in my implementation, I continued with similarity using a high similarity for closeness which is also the same thing. This implies all throughout my work, I have used a high similarity to define close proximity.

IMPLEMENTATION

DATA PREPARATION

I made four different datasets for the four different scenarios. It is worth noting that another alternative could have been to use one dataset for all but I didn't want to load things into memory that are not being used reason why I created four datasets.

D1: Jaccard similarity based on the genres of the movies

movieId	title	genres
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	Jumanji (1995)	Adventure Children Fantasy
3	Grumpier Old Men (1995)	Comedy Romance

D2: Jaccard similarity based on the tags of the movies

Tags were selected based on the relevance to the movie and the most relevant were combined to form the following.

movieId	title	tag
1	Toy Story (1995)	3d action adventure affectionate animal movie ...
2	Jumanji (1995)	action adventure animals based on book bullyin...
3	Grumpier Old Men (1995)	comedy destiny good great mentor original very...

D3: cosine similarity based on the ratings of the movies

movieId	title	1	2	3	4	5	6	7	8	9	...	7036	7037	7038	7039	7040	7041	7042	7043	7044	7045
1	Toy Story (1995)	0.0	3.5	4.0	3.0	4.0	0.0	0.0	4.0	0.0	...	0.0	0.0	0.0	5.0	3.5	4.0	4.5	0.0	0.0	4.5
2	Jumanji (1995)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	...	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Grumpier Old Men (1995)	0.0	0.0	0.0	0.0	0.0	0.0	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

D4: 0.3*d1 + 0.25*d2 + 0.45*d3

movieId	title	genres	tag	1	2	3	4	5	6	7	...	7036	7037	7038	7039	7040	7041	7042	7043	7044	7045
1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	3d action adventure affectionate animal movie ...	0.0	3.5	4.0	3.0	4.0	0.0	0.0	...	0.0	0.0	0.0	5.0	3.5	4.0	4.5	0.0	0.0	4.5
2	Jumanji (1995)	Adventure Children Fantasy	action adventure animals based on book bullyin...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	2.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	Grumpier Old Men (1995)	Comedy Romance	comedy destiny good great mentor original very...	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

Implementation procedure

Generally, a small chunk of the dataset (depending on the chosen operation) is loaded into memory at a time by buffering. Each chunk is processed and dropped, retaining only metadata that summarizes the result.

Dataset	Buffer size	Total size
D1	10KB	2.49MB
D2	200KB	6.65MB
D3	3MB	598MB
D4	3MB	371MB

Based on the nature of the produced datasets, I followed two approaches. Since in case d3 (cosine similarity with ratings) I had all numeric entries, I used the notion of centroids and added concepts of the BFR algorithm to the kmeans.

In the other three case where the use of centroids in not possible (presence of non-Euclidean space), I used the notion clustroids.

Procedure for d3 (centroid approach)

1. Cluster the first 1000 movies loaded into memory using the full kmeans. That is:
    a. Randomly select centroids and assign other points to the different clusters depending on the highest cosine similarity to the centroids. Update centroids by averaging.
    b. Do a. until clusters don't change or maximum iteration is reached (=100).
2. Summarize clusters by retaining the sum, the squared sum, the number of movies in the cluster and the movies in the cluster. This chunk of the data is then dropped from memory.
3. For the subsequent movies that are loaded in to memory, the similarity is compared to the current centroids and they are assigned to the closest cluster and the cluster metadata is updated then the points are dropped.
4. Do 3. Until the entire dataset has gone in and out of memory and then write centroids and clusters to file.

It is important to note that this is essentially a one pass algorithm of kmeans similar to the BFR.

Example of centroids for 10 clusters

	0	1	2	3	4	5	6	7	8	9	...	7035	7036	7037	7038	7039	
0	0.000000	0.293269	0.081731	0.000000	0.817308	0.019231	0.423077	0.913462	1.201923	0.201923	...	1.528846	0.086538	0.048077	0.312500	0.254808	0
1	0.000000	0.000000	0.000000	0.000000	0.010169	0.000000	0.000000	0.023729	0.016949	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0
2	0.001335	0.018279	0.073424	0.007291	0.008215	0.001232	0.000000	0.022181	0.022387	0.008215	...	0.176012	0.001848	0.015712	0.025467	0.000000	0
3	0.037551	0.008745	0.009774	0.000000	0.011317	0.002058	0.034979	0.007202	0.000000	0.003086	...	0.013374	0.014403	0.000000	0.007202	0.000000	0
4	0.000000	0.091549	0.000000	0.000000	0.422535	0.000000	0.000000	0.204225	0.542254	0.007042	...	0.038732	0.000000	0.000000	0.066901	0.000000	0
5	0.000000	0.150350	0.052448	0.013986	0.503497	0.000000	0.000000	0.594406	0.216783	0.020979	...	0.405594	0.020979	0.000000	0.087413	0.020979	0
6	0.023826	0.084307	0.396564	0.158534	0.018557	0.016495	0.002978	0.040092	0.029095	0.018786	...	0.519129	0.019244	0.175258	0.126002	0.016609	0
7	0.001686	0.010750	0.002951	0.000843	0.000000	0.000000	0.000000	0.001265	0.000000	0.000000	...	0.019393	0.000000	0.000000	0.017917	0.000000	0
8	0.026665	0.014991	0.012902	0.002826	0.003932	0.005407	0.000000	0.008356	0.022610	0.006636	...	0.029983	0.007373	0.000000	0.020767	0.004055	0
9	0.001528	0.010389	0.051334	0.015176	0.002241	0.000815	0.000000	0.003463	0.024852	0.001222	...	0.124975	0.000000	0.034732	0.007130	0.000815	0

10 rows × 7045 columns

Procedure for d1, d2 and d4(Clustroid approach)

1. For the first chunk of movies loaded into memory, random clustroids are selected and clusters are assigned.
   - Compute intracluster distances between a point and every other point in the cluster to ensure that the clustroid has the smallest distance (largest similarity) to every other point. Stop If this is the case otherwise recompute new clustroids.
   - Retain clustroids and clusters then drop chunk
2. For every other subsequent movie entry loaded into memory, assign to the cluster with the closest clustroid.
3. After the last chunk of movies is loaded into memory, print out clustroids and clusters to output files

D1 clustroids for 6 clusters (genre)

	Cluster id	Centroids
0	0	Comedy
1	1	Drama Thriller War
2	2	Comedy Drama
3	3	Drama Thriller
4	4	Comedy Romance
5	5	Action Adventure Sci-Fi

D2 clustroids for 10 clusters(tags)

	Cluster id	Centroids
0	0	clever comedy dialogue funny good soundtrack g...
1	1	bullying catastrophe childhood destiny feel-go...
2	2	action chase cool destiny dialogue entertainin...
3	3	based on book excellent excellent script good ...
4	4	affectionate destiny friendship great great en...
5	5	adventure animal movie animals animated animat...
6	6	based on book clever cool dialogue entertainin...
7	7	action catastrophe chase cool dialogue fast pa...
8	8	adventure based on book childhood children dis...
9	9	original runaway brutality beautifully filmed ...

D4 clustroids (10)

	0	1	2	3	4	5	6	7	8	9	...	7037	7038	7039	7040	7041	7042	7043	7044	7045	7046
0	Adventure Children Fantasy	adventure based on book childhood children fam...	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
1	Drama	based on book original runaway adaptation base...	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
2	Comedy Drama	affectionate dialogue original suprisingly cle...	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	Comedy Drama	dialogue good soundtrack nostalgic original qu...	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
4	Drama	dialogue good soundtrack original drinking rel...	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
5	Animation Children Drama Musical Romance	adventure animal movie animals animated animat...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	3.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0
6	Drama	dialogue family original relationships parenth...	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
7	Comedy Romance	classic comedy cute cute! feel-good friendship...	0.0	0.0	0.0	0.0	2.0	0.0	0.0	3.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
8	Action Sci-Fi Thriller	computer animation mentor original technology ...	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
9	Comedy Drama	affectionate good good soundtrack original run...	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

Sample Run

```
Movie Clustering Application
Please input the number of clusters k=5
Supported distance metrics
         d1: jaccard similarity based on the genres of the movies
         d2: jaccard similarity based on the tags of the movies
         d3: cosine similarity based on the ratings of the movies
         d4 = 0.3*d1 + 0.25*d2 + 0.45*d3
Please input selection from above (eg d1):d2
Running...Please Wait
Done..Please check output files
```

PART TWO: MOVIE RECOMMENDATION

Types of recommendation systems

Content based

Similarity of items is determined by measuring the similarity in their properties. An item profile is created consisting of important features of that item. However, choosing these features is not a trivial task. From the item profiles, a user profile is generated. This is used to make recommendations for the user.

Advantages

- No need for data on other users
- Ability to recommend to users with unique taste
- Ability to recommend new and unpopular items
- Recommendations are explainable

Disadvantages

-Finding appropriate features is a difficult task

- overspecialization to specific user profile
- Cold start problem for new users (how do you build a user profile for a new user?)

Collaborative Filtering

Types of collaborative filtering

User-based collaborative filtering

The basic idea here is to find a group of users similar to the target user and predict user ratings based on the ratings of these similar users.

Item based
The basic idea here is that the rating of a specific user for a particular item(movie) is based on the ratings of that user for similar movies (recall that similarity here refers to closeness).

Definition of similarity

This applies to both user based and item based collaborative filtering

Jaccard similarity (as seen above in part one)

Limitation: Jaccard similarity does not take into account the actual ratings of the two components.

Cosine similarity (as seen above in part one)

Limitation: Treats missing ratings as negatives

Centered Cosine similarity (aka Pearson correlation)

I did this by subtracting the mean value from the values of the concerned vectors before using normal cosine similarity.

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

Pros

- Handles “tough raters” and “easy rates” well.
- Missing ratings are treated as “average”.

Prediction of Ratings

User based

- Let ***rx*** be the vector of user ***x***’s ratings
- Let ***N*** be the set of ***k*** users most similar to ***x*** who have rated item ***i.*** the predicted rating of item I by user x is given by:
  - Average of similar ratings

    $$r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$$
  - Weighted average of similar ratings (weighted by their similarities).

    $$r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$$
  - Other options also exist.

For item based, the rating prediction is the same but for the fact that *rx* would be the vector of an item’s (movie) ratings and N the set of k movies most similar to x that have been rated.

IMPLEMENTATION

PHASE 1: DATA GENERATION

I merged the “movies” and the “ratings” datasets to form a new dataset which I called “item_based”. the columns here are the user_ids and the rows represent the movies.

userId	1	2	3	4	5	6	7	8	9	10	...	7036	7037	7038	7039	7040	7041	7042	7043	7044	7045
movieId																					
1	NaN	3.5	4.0	3.0	4.0	NaN	NaN	4.0	NaN	3.5	...	NaN	NaN	NaN	5.0	3.5	4.0	4.5	NaN	NaN	4.5
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	...	2.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 7045 columns

PHASE 2: CLUSTERING

Reason:

The motivation behind my inclusion of clustering here is that the procedure for finding the k most similar items/users (neighborhood) is quite expensive as it usually entails a pass through the entire rated items for every unrated item. In order to reduce the time complexity of this procedure, I precomputed clusters, so the search of k most similar items is only within one cluster which contains similar items so one can say with some degree of certainty that the most similar movies to another movie would be within its cluster.

Implementation of clustering

Clustering here is performed as discussed in part one above except:

- The similarity metric used here in the centered cosine (Pearson’s correlation) for reasons explained earlier.
- The clusters are written to different files (this is to ensure that only points from one cluster are present in main memory at a time, also respecting the disk-based policy in phase three as well).

```
Please Enter the number of clusters k :10
Please wait
Generating Clusters........
Done...Please check output files
```

PHASE 3:

ITEM BASED HIGH LEVEL ALGORITHM

For every cluster in disk perform the following:

- Load in memory by buffering using a buffer of size 4MB.
- Consider user x. For every unrated movie perform the following:
  Find the k (=3) most similar movies that have been rated by that user (centered cosine similarity)
  Predict rating using weighted average of the user x 's ratings of the k most similar movies. (simple averaging is also used in the rare case of false positives found in the cluster, penalized by subtracting 1)
- Retained the top 20 highly rated movies for that cluster and proceed to reading the next cluster from the disk while still buffering and do same.
- The top 20 of the combined top 20s of all clusters is computed at the end and printed.

USER BASED HIGH LEVEL ALGORITHM

For every cluster in disk perform the following:

- Load in memory by buffer with size 4MB.
- Consider user x. For every unrated movie y perform the following.:
  Find the k (=5) most similar users to x that have rated movie y (centered cosine similarity)
  Predict rating using weighted average of the ratings of the k most similar users.
  (simple averaging is also used in the rare case of false positives, penalized by subtracting one)
- Retain the top 20 highly rated movies for that cluster and proceed to reading the next cluster from the disk and do same.

HYBRID

- Compute top 20 recommendations using user-based collaborative filtering
- Compute top 20 recommendations using item_based collaborative filtering
- Return the top 20 of the 40 recommendations obtained so far based on their ratings.

Sample Run

```
Recommendation System
Please select operation
    1. Item based
    2. User based
    3. Hybrid
Choice:1
Please enter the number of clusters:10
Enter user id:123
Please wait
Generating Recommendations......
```

```
TOP 20 MOST RECOMMENDED MOVIES FOR CHOSEN USER
Movie title                     Rating
('Paris  France (1993)', 5.0)
('Butterfly Kiss (1995)', 5.0)
('They Made Me a Criminal (1939)', 5.0)
("Lilian's Story (1995)", 5.0)
('Heart Condition (1990)', 5.0)
('Belizaire the Cajun (1986)', 5.0)
('Last Legion  The (2007)', 5.0)
('Outpost (2008)', 5.0)
('Urgh! A Music War (1981)', 5.0)
('Babylon 5: The Legend of the Rangers: To Live and Die in Starlight (2002)', 5.0)
('Tango (1981)', 5.0)
('Absentia (2011)', 5.0)
('Pod (2015)', 5.0)
('Conceiving Ada (1997)', 5.0)
('Fire Within  The (Feu follet  Le) (1963)', 4.9573)
('Split Second (1992)', 4.9361)
('Maps to the Stars (2014)', 4.9183)
('Raw Deal (1948)', 4.9146)
('Low Down  The (2000)', 4.8728)
('Jackpot (2001)', 4.8728)
```

Remarks

- Selecting the right neighborhood size for k nearest neighbors is important as it affects the ratings. I had to try several values before sticking to 3 and 5 which gave quite good results.
- In the clustering phase the selection of the centroids is random and it is possible different clusters are generated if this is run again.