Data Analysis Mini-Project: Clustering Movies

Dr. Sivan Sabato

Fall 2018/9

1 The goal

Clustering is the process of taking a set of objects and partitioning them to disjoint sets (clusters), according to some criterion. In this project we are interested in clustering movies based on user preference. We would like the clusters to indicate which movies "go together" in people's preferences. For instance, probably a person who watched "Pulp Fiction" is more likely to also watch "Kill Bill", while a person who watched "The little mermaid" is more likely to also watch "The beauty and the beast", but it is less likely that someone who watched "The little mermaid" also watched "Pulp Fiction", so we would expect the first two movies to be in the same cluster, and the last two movies to be in another cluster.

To decide how to cluster a set of movies, we will define a **cost function**. This is a function that takes a clustering and calculates the clustering's cost. Our goal is to try to find a clustering in which this cost is as low as possible. This goal is usually computationally hard to achieve, so we will need to use a heuristic algorithm.

Let us now define the objective function for our movie problem. To do this we need to define some notations. Denote the number of movies by $k \geq 2$, and denote the different movies in our data set by $M = \{m_1, m_2, \ldots, m_k\}$. Every possible clustering of the movies can be described by a set of non-empty sets, $\mathcal{C} := \{C_1, \ldots, C_n\}$, where n is the number of clusters in this clustering, and we have $\cup_{i=1}^{n} C_i = M$, and for every $i \neq j$, $C_i \cap C_j = \emptyset$. We will not decide on the number of clusters n in advance, it will be decided by the clustering algorithm.

We consider a given population of movie-goers. For every movie $m \in M$, we can consider the **probability that a random person from our population watched this movie**, which we denote $p(m)$. For instance, if m is the movie "Taxi Driver", then $p(m) \in [0, 1]$ is the fraction of people in our population who watched this movie. We will assume that we have the list of all the people in our population, and all the movies each of these people watched, so that we can use this data to calculate $p(m)$ for any $m \in M$. We also consider pairs of movies $m_1, m_2 \subseteq M$, and denote by $p(m_1, m_2)$ the **probability that a random person from our population watched *both* of the movies**. For instance, if m_1 is "Taxi Driver" and m_2 is "The little mermaid", then $p(m_1, m_2)$ is the fraction of people in our population who watched both movies. Note that we cannot calculate $p(m_1, m_2)$ just from knowing $p(m_1)$ and $p(m_2)$, we have to know how often these two movies occur **together**.

The cost function we will try to minimize for our clustering is the following: For each clustering $\mathcal{C} = \{C_1, \ldots, C_n\}$, its cost is defined to be

$$\text{cost}(\mathcal{C}) := \sum_{i=1}^{n} \text{cost}(C_i),$$

where the cost of each cluster $C \subseteq M$ is defined by

$$\text{cost}(C) := \begin{cases} \sum_{m_1 \in C} \sum_{m_2 \neq m_1, m_2 \in C} \frac{1}{|C|-1} \log(1/p(m_1, m_2)) & |C| \geq 2 \\ \log(1/p(m)) & |C| = 1, C = \{m\}. \end{cases}$$

Note that all the probabilities are in $[0, 1]$ so all the expressions $\log(1/p)$ are non-negative. To get an intuition of the meaning of this cost function, consider a simple case there there are only two movies in our data set. If there are only two movies, $M = \{m_1, m_2\}$, we have only two options: either they are in the same cluster, so the clustering is

$\mathcal{C}_1 = \{C_1\}$ and $C_1 = \{m_1, m_2\}$, or they are in different clusters, so the clustering is $\mathcal{C}_2 = \{C_1, C_2\}$, with $C_1 = \{m_1\}$, $C_2 = \{m_2\}$. Which one has a lower cost? we have

$$\text{cost}(\mathcal{C}_1) = \log(1/p(m_1, m_2))$$
$$\text{cost}(\mathcal{C}_2) = \log(1/p(m_1)) + \log(1/p(m_2)) = \log(\frac{1}{p(m_1)p(m_2)}).$$

So:

$$\text{cost}(\mathcal{C}_1) \leq \text{cost}(\mathcal{C}_2) \iff p(m_1, m_2) \geq p(m_1)p(m_2).$$

In other words, if the two movies are *positively correlated*, meaning that people who watch one movie are more likely than random people to watch the other movie, then we would like them to be in the same cluster, but if they are *negatively correlated*, meaning that people who watch one movie are less likely to watch the other than random people, then we would like them to be in different clusters.

2 A Simple algorithm

Based on the example of two movies, we can use the following heuristic to find a clustering.

1. For every pair of movies, check if they are positively correlated or negatively correlated by calculating the values $p(m_1), p(m_2), p(m_1, m_2)$ from the data set and checking the inequality above.
2. Mark each pair by "$+$" if it's positively correlated, and by "$-$" otherwise.
3. Try to find a clustering in which as many possible "$+$" pairs in the same cluster, and as many possible "$-$" pairs in separate clusters.

The last step is called *correlation clustering*. Finding the optimal correlation clustering is computationally hard, but there is a simple algorithm that finds an approximate solution, see it here: `https://en.wikipedia.org/wiki/Correlation_clustering` (the pivot algorithm).

Your **first step** will be to implement the pivot algorithm for the movie clustering problem. This involves reading the data set of movies, finding for each pair of movies whether they should be marked $+$ or $-$, and implementing the pivot algorithm using these marks as input. This will give you a clustering, but it will not be the clustering that necessarily gets the lowest possible cost, since this problem is computationally hard, and also we only used $+$ and $-$ and not all the information about the probabilities.

3 Improving the algorithm

In the **second step**, you will design and implement an algorithm that improves the results of the previous algorithm. Note that you have access to a lot of information that the previous algorithm doesn't use, for instance the actual probabilities of each pair of movies, and additional information about each movie. Your goal is to suggest an improvement that will get better results than the correlation clustering implementation. The improvement can be a totally different algorithm, or it can take the output of the pivot algorithm and improve it, or it can change its input, or anything else you think might be helpful. Don't take an easy way out like clustering only based on the genre of the movie: you should use the information from the users as much as possible.

The way that we measure the improvement of the algorithm for a specific clustering problem is by comparing the cost function of the two output clusterings: if your output clustering has a lower cost, then it is considered better than the correlation clustering output. You should improve the algorithm in ways that you think might be helpful for the general movie-clustering problem. For instance, don't hard-code specific movies or users or genres into your algorithm.

Keep in your implementation the option to run also the algorithm from the first step, since you will need to compare the two versions.

4 The data set

We will use a data set of movies and users. The data set includes many movies, we will not try to cluster all of them at once: in each run of our algorithm, we will cluster a small number of movies (say a 100) from the data set, so that we can easily read the resulting clusters and see what they look like. So while the full data set has a set of movies M, in each run we will ask the algorithm to provide a clustering only for some subset $M' \subseteq M$ of the movies that will be given as input.

The data set can be downloaded from here: `https://grouplens.org/datasets/movielens/1m/`, it is the Movielens 1M dataset. There are several files in this data set, you should use the file with the movie ratings to construct the vectors of users. If user i rated movie j, you should set $v_i(j) = 1$, otherwise set it to zero. You should ignore the value of the rating itself. In other words, we assume for this project that a user watched a movie if and only if the user rated the movie.

- This data set includes only users who rated at least 20 movies.
- Your algorithm should ignore movie ids that have less than 10 ratings. If you have any such movie in the input set M', you should output to stderr the following message and continue without this movie (replace `<movieid>` and `<num>` with the proper values):
  `Movie <movieid> ignored because it has only <num> ratings`

In this project, we will run your algorithm only with this data set, so you can preprocess it any way you like, and save the results of the preprocessing into a file that your algorithm will read every time it runs. This can be useful to make the algorithm run faster.

5 Calculating the probabilities from the data set

Some users rate many more movies than others, and this can make it seem like movies are correlated when they are not really related to each other. To correct this issue, we will calculate the probabilities in a way that reduces the effect of users who watched many movies.

Denote the users in our data set by $v_1, \ldots, v_N$. Denote the number of movies that user i watched by n_i. Each user can be described by a vector $v_i \in \{0,1\}^k$, so that $v_i(j) = 1$ if and only if user i watched movie m_j, and $n_i = \sum_{j=1}^{k} v_i(j)$. We calculate the movie probabilities as follows:

$$p(m_j) := \frac{1}{N+1}\left(\frac{2}{k} + \sum_{i=1}^{N} \frac{2}{n_i} \cdot v_i(j)\right)$$

$$p(m_j, m_t) := \frac{1}{N+1}\left(\frac{2}{k(k-1)} + \sum_{i=1}^{N} \frac{2}{n_i(n_i-1)} \cdot v_i(j)v_i(t)\right).$$

This formula makes sure the probabilities are never zero so that we don't get infinite costs. Also, in our data set all users have watched at least two movies, so $n_i \geq 2$ always, which means the result of these formulas are also never more than 1.

Note: you should calculate $p(m_j), p(m_j, m_t)$ using the whole set of movies M, even though later you will only cluster a subset $M' \subseteq M$.

6 Code

You can write your code in any (reasonable) language, e.g., Python, Java, Matlab, C++. If you want to use a different language email me to make sure it's OK. Note that Java or C++ implementations will usually run much faster than Python or Matlab. You should make sure your code runs in a reasonable time. It shouldn't usually take more than

several seconds. Your submission should include the code as well as an executable (or a script, e.g. can be a script that runs Java with the jar file). The command line to run your code should be in the following format:

```
moviecluster <datasetfolder> <1/2> <moviesubsetfilepath>
```

1. `moviecluster`: name of script. No changes to the name are allowed. No extensions to the script name are allowed. If you need to run a file with an extension, put a script called `moviecluster` that runs it.

2. `<1/2>`: if "1", run correlation clustering algorithm. If "2", run the new and improved algorithm.

3. `moviesubsetfilepath`: the path to a file that lists the numerical ids of movies to cluster. This will be a text file, with one number in each line. Make sure to check for legality of this file so that you don't crash on bad input. The file path can be a full path or a relative path from the current directory.

4. `datasetfolderpath`: the path to a folder that has the Movielens 1M data files, with their original names. You do not have to read the files in this folder - you may assume that the algorithm works only with the Movielens 1M dataset, and prepare your own data files that will be submitted with the code and the program will read them. This may be useful to make your algorithm run faster. The folder path can be a full path or a relative path from the current directory.

Your code should run on a standard Linux machine in the lab. It will be tested on the machine **vmcicero3** so make sure it works there before you submit.

The output of your algorithm should be printed to the standard output. It should be a list of the clusters you found: in each line, list one cluster. Each cluster should look as follows:

```
<movie id> <movie title>, <movie id> <movie title>, ...
```

After outputting the full clustering, output in a separate line the cost of the clustering you found. Print only a number, no text.

Do not output anything else to the standard output!

7 Report

In your submission you should include a pdf file with your report. The report should have your answers to the following questions:

1. Describe your suggested improvement to the algorithm. Explain why you chose it and why it should be helpful.

2. Give an example of a movie subset where your improvement indeed improved the results significantly, show the clustering before and after your improvement, along with the costs before and after.

3. Describe the main challenges you had in the implementation of this project, and how you handled them.

4. Select 3 movie subsets that produce interesting results in your clustering algorithm. Each subset should include at least 20 movies. For these subsets, list the output clustering that you got with your algorithm, with and without your improvement. Also list in a table the cost of the clustering your algorithm obtained, before and after the improvement. You are allowed to share movie subset files with other teams so that each team can find interesting subsets for its own report. Explain for each movie subset what is interesting about the results for this case.

5. Generate 20 random movie subsets of size 100, by selecting 100 random (legal) movies from the data set for each subset. Run your algorithm on each of them with and without your improvement, and report in a table the cost before and after the improvement. Also report the average cost of each algorithm on these 20 subsets. Which cost is better?

8 Submission

You should submit a zip file named by your id numbers, e.g. `01234567_7654321.zip`. Include the following files **in the root of the submission archive file**:

- `moviecluster` - the exectuable or script that runs the algorithm.
- `report.pdf` - the project report.
- `subset1.txt,...,subset3.txt` - the subset files used for producing interesting clusterings in the report. These files may be shared between teams.
- `randomsubset1.txt`,...,`randomsubset20.txt` - the random subset files used for the report.
- Your source code (can be in a subfolder)
- The binaries, including data files processed from the Movielens 1M dataset, if your algorithm uses any such files.

Before you submit, make sure that if you unzip the submission file on a different computer and in a different folder than where you created it, and then runs the command line, it works OK. Submit the zip file through the submission system, make sure it is not larger than the size limit. If your submission is larger than the submission system size limit because of large data files, do the following:

- Put the data files in a location from which they can be downloaded using a simple link
- Put in the root of the submission archive file a script to download the files and save them in the correct location and with the correct name for your project to run properly. The command line for the script should be:

  ```
  getmoviefiles <the path where your submission was unzipped>
  ```

  You can use the command line "wget" to download the files.
- The data files should be available to download for the entire time until you get your grade.

Important: For grading, the project will be executed as follows:

- The submission file will be unzipped **on vmcicero3**.
- From the same directory, `getmoviefiles` will be executed **on vmcicero3**, if it exists
- From the same directory, `moviecluster` will be executed **on vmcicero3**, possibly several times.

Make sure that this process runs smoothly on vmcicero3, even in someone else's account. If you need a package that is not installed on vmcicero3, please email me. **Note: it could take a while to install packages, so don't do this last minute before the deadline!**

IMPORTANT: SOME OF THE TESTS WILL BE DONE BY SCRIPTS. ANY DEVIATION FROM THE INSTRUCTIONS ON OUTPUT OR SUBMISSION MIGHT CAUSE POINT DEDUCTION.

9 Addional details

- Submission deadline: 30.1.19, 23:59.
- You may submit alone or in pairs.
- If you would like to ask me anything, please email first. If you would like to come to reception hour, please coordinate this by email first.