Clustering Movies

November 20 and 21, 2019

Instructions

Upon completion of all tasks, a TA will give you credit for today’s studio. If you do not manage to complete all the assigned work during the studio period, do not worry. You can continue to work on this assignment until Monday, November 25 at 2 PM. Come by TA hours any time before then to show us your completed work and get credit for today’s studio, or show us your work during Monday’s class period.

Objectives

To practice clustering with a variety of similarity metrics.

Overview (Leftover from the HW!)

This assignment has two parts. In the first, you’ll be exploring movie data by clustering them in various ways. In the second, you’ll be using a technique called collaborative filtering to recommend movies to users (including yourself!), similar to how Netflix makes movie recommendations.

More specifically, in the first part, you will be clustering movies based on their content; you won’t be making any recommendations. You should note, however, that if you can cluster movies effectively this way, then you could recommend new movies to users with content that is similar to other movies they are known to have liked.

Data

The movie data found here contains movies and some information about their content, namely their genres.

Data Wrangling

Load the “movies.csv” file into an R data frame called movies. Be sure to use stringsAsFactors = FALSE, so that every single movie title is not assigned its own level.

Take a glimpse of the data. You’ll notice that the “genre” variable concatenates all genres into a single string. You can/should split this string up into a list using strsplit(movies$genres, "|”).

Next, use the duplicates predicate to find duplicate movie titles: e.g., movies[duplicated(movies$title), ]. Note: Movies produced and then
reproduced (with a different cast, etc.) again later should not be treated as duplicates!

*Hint:* You can use the `match` function to find the index of a particular title: e.g.,
`match("Hoop Dreams (1994)", movies$title)`.

For all duplicate titles, choose one of the two observations to retain, deleting the other from the data frame. Delete the one that seems to you to be a less accurate description of the movie’s genres. Feel free to watch a trailer to help you decide!

*Hint:* There are two duplicate titles (same title; same year), and in both cases, their corresponding genres differ.

Your next goal is to build a matrix of movies by genres for eventual content filtering by genre. This matrix should have `nrow(movies)` rows, and the number of columns should be the number of genres (including no genre, in case some movies have no genres associated with them).

Here’s a command to extract all the unique genres, and hence the number of them:
`genres <- unique(unlist(movies$genres))`. Note the use of the `unlist` function. It is necessary because, for each movie in the data frame, the `genres` variable contains a list of that movie’s genres.

With this information on hand, you can now create a matrix of the required size:
`mat <- matrix(0, nrow(movies), length(genres))`. The first parameter, 0, is the default value for entries in this matrix, which we will take to mean that a given movie is not of a given genre. The next two parameters describe its dimensions.

So far so good!

The next step is to populate this matrix; that is, to assign 1s rather than 0s to all rows (movies) and columns (genres) where a movie is indeed of that genre. To do so, you should loop over all movies, and inside that loop, you should include a second loop where you loop over all genres associated with each movie. Inside these nested loops, you should set the entries in the matrix corresponding to that movie-genre combination to 1 rather than 0.

*Hint:* Use two for loops, and in the inner loop, loop over all genres in `movies[i, "genres"]`: i.e., loop over all of the i-th movie’s genres.

*Hint:* Consider using the `match` function again, matching each of a movie’s genres to the `genres` vector to find the columns in the matrix that should be set to 1 for that row (i.e., that movie).

We are almost there. Next, use `as.data.frame(mat)` to convert your matrix to a data frame. And then finally, use the `names` function to name the variables to be the appropriate genres (which, conveniently, you have already stored in the vector `genres`), and the `row.names` function to set the row names to be the movies’ titles.
At this point, you should have successfully created a matrix of movies by genres, where each observation represents a movie and its corresponding genres. How many movies are classified as romance? How many movies are classified as Westerns? How many movies are classified as romance and Westerns?

*Hint:* Use the `table` function to answer these questions.

**Similarity**

A **similarity measure** measures how alike two observations are. In data science, we sometimes describe similarity in terms of distance; if distance is small, then there is a high degree of similarity, whereas if distance is large, then there is a low degree of similarity.

Popular similarity metrics include Euclidean distance, cosine similarity, Pearson correlation, and Jaccard similarity. Refer to the class slides for a refresher on these different metrics.

Install the following package, which contains a variety of similarity metrics, including the ones mentioned above.

```
{r, eval = FALSE} install.packages('proxy') library(proxy)
```

You can now use the `dist(x, method = "..." )` function in R to calculate distances, passing in `x`, the data frame of observations that you want to compare, and the name of a distance/similarity metric (e.g., “euclidean”, “jaccard”, etc.).

*Note:* The `proxy` package includes a `simil` function as well as a `dist` function. All measures can be computed either as similarities or distances using one of these two functions. It is a simple matter to transform one to the other; check out (this documentation)[https://www.rdocumentation.org/packages/proxy/versions/0.4-19/topics/dist] for more details.

Report the similarity between the movies “Star Wars: Episode IV - A New Hope (1977)” and “Stargate (1994)” using the four metrics mentioned above. Similarly, report the similarity between “Toy Story (1995)” and “Die Hard (1988)” using the four metrics. Do the different metrics behave similarly? Which ones seem most to work best on these examples?

**Hierarchical clustering (Time Permitting)**

Recall the steps in (agglomerative) hierarchical clustering: 1. Initialize the clustering with all data points in their own cluster 2. Compute distances between all clusters 3. Identify the two closest clusters, and combine them 4. Repeat until only one cluster remains

Choose 15 to 30 movies of interest to you in the dataset, and create a data frame that contains only these movies. Then calculate the distances between these movies using any two similarity metrics that seem to you to work best.
Create a hierarchical clustering of your choice movies using \texttt{hclust}. Save this clustering as, say, \texttt{hc}, and then truncate the result so that it contains no more than ten clusters using the \texttt{cutree} function: e.g., \texttt{cutree(hc, k = 7)}.

The output of \texttt{hclust} is a vector of clusters: e.g., 1 1 2 1 2 3 2 3 1. In this sample output, cluster 1 consists of the first, second, fourth, and ten movies; cluster 2 consists of the third, fifth, and eighth movies; and cluster 3 consists of the sixth, seventh, and ninth movies. That’s all good and well, but it would be nice to know the names of these movies!

The function \texttt{which} can be used to extract the indices corresponding to each cluster from the output of \texttt{hclust}, and then you can find out the names of the movies in each cluster by looking up these indices in the \texttt{name} column of your data frame (say, \texttt{my_movies}) with this line of code: \texttt{my_movies$name[indices]}.

To extract the indices corresponding to the $i$th cluster from \texttt{hc}, the output of \texttt{hclust}, you can use \texttt{which(hc %in% rep(i, num_movies))}. Here, \texttt{num_movies} is the number of movies in your data frame, \texttt{rep(i, num_movies)} is a vector of all $i$'s of length \texttt{num_movies}, and the \texttt{which} function returns the indices of \texttt{hc} whose value matches $i$.

Extract the movie names corresponding to the output of \texttt{hclust}. Does \texttt{hclust} do a good job of clustering movies by content? Which similarity metric(s) seems to work best? What happens when you set $k$ too high or too low?

\textbf{End of Studio}

When you are done please call over a TA to review your work, and check you off for this studio. If you do not finish within the two hour studio period, remember to come to TA office hours to get checked off.