MITOCW | MIT15_071S17_Session_6.2.13_300k

In this video we'll use hierarchical clustering to cluster the movies in the movie lens data set by genre.

After we make our clusters, we'll see how they can be used to make recommendations.

There are two steps to hierarchical clustering.

First we have to compute the distances between all data points.

And then we need to cluster the points.

To compute the distances we can use the dist function.

We only want a cluster movies on the genre variable, not on the title variable, so we'll cluster on columns two through 20.

So let's call the output distances, and we'll use the dist function, where the first argument is moviesmovies[2:20], this is what we want to cluster on.

And the second argument is method="euclidean", meaning that we want to use euclidean distance.

Now let's cluster our movies using the hclust function for hierarchical clustering.

We'll call the output clusterMovies, and use hclust where the first argument is distances, the output of the dist function.

And the second argument is method="ward".

The ward method cares about the distance between clusters using centroid distance, and also the variance in each of the clusters.

Now let's plot the dendrogram of our clustering algorithm by typing plot, and then in parentheses clusterMovies.

This dendrogram might look a little strange.

We have all this black along the bottom.

Remember that the dendrogram lists all of that data points along the bottom.

But when there are over 1,000 data points it's impossible to read.

We'll see later how to assign our clusters to groups so that we can analyze which data points are in which cluster.

So looking at this dendrogram, how many clusters would you pick?

It looks like maybe three or four clusters would be a good choice according to the dendrogram.

But let's keep our application in mind, too.

We probably want more than two, three, or even four clusters of movies to make recommendations to users.

It looks like there's a nice spot down here where there's 10 clusters.

This is probably better for our application.

We could select even more clusters if we want to have very specific genre groups.

If you want a lot of clusters it's hard to pick the right number from the dendrogram.

You need to use your understanding of the problem to pick the number of clusters.

Let's stick with 10 clusters for now, combining what we learned from the dendrogram with our understanding of the problem.

Now back in our R console we can label each of the data points according to what cluster it belongs to using the cutree function.

So let's type clusterGroups=cutree(clusterMovies, k=10).

Now let's figure out what the clusters are like.

We'll use the tapply function to compute the percentage of movies in each genre and cluster.

So let's type tapply, and then give us the first argument, movies$Action-- we'll start the action genre-- and then clusterGroups, and then mean.

So what does this do?

It divides our data points into the 10 clusters and then computes the average value of the action variable for each cluster.

Remember that the action variable is a binary variable with value 0 or 1.

So by computing the average of this variable we're computing the percentage of movies in that cluster that belong in that genre.

So we can see here that in cluster 2, about 78% of the movies have the action genre label, whereas in cluster 4 none of the movies are labeled as action movies.

Let's try this again, but this time let's look at the romance genre.

Here we can see that all of the movies in clusters six and seven are labeled as romance movies, whereas only 4% of the movies in cluster two are labeled as romance movies.

We can repeat this for each genre.

If you do you can create a large table to better analyze the clusters, which I saved to a spreadsheet.

Lets take a look.

Here we have in each column the cluster, and in each row the genre.

I highlighted the cells that have a higher than average value.

So we can see here in cluster 2, as we saw before, that cluster 2 has a high number of action movies.

Cluster 1 has a little bit of everything, some animation, children's, fantasy, musicals, war and westerns.

So I'm calling this the miscellaneous cluster.

Cluster 2 has a lot of the action, adventure, and sci-fi movies.

Cluster 3 has the crime, mystery, thriller movies.

Cluster 4 exclusively has drama movies.

Cluster 5, exclusively has comedies.

Cluster 6 has a lot of the romance movies.

Cluster 7 has movies that are comedies and romance movies.

So I'm calling these the romantic comedies.

Cluster 8 has the documentaries.

Cluster 9 has the movies that are comedies and dramas, so the dramatic comedies.

And cluster 10 has the horror flicks.

Knowing common movie genres, these cluster seem to make a lot of sense.

So now, back in our rconsole, let's see how these clusters could be used in a recommendation system.

Remember that Amy liked the movie Men in Black.

Let's figure out what cluster Men in Black is in.

We'll use the subset function to take a subset of movies and only look at the movies where the Title="Men in Black (1997)".

Close the quotes in the parentheses.

I knew that this is the title of Men in Black because I looked it up in our data set.

So it looks like Men in Black is the 257th row in our data.

So which cluster did the 257th movie go into?

We can figure this out by typing clusterGroupsclusterGroups[257].

It looks like Men in Black went into cluster 2.

That make sense since we just saw that cluster 2 is the action, adventure, sci-fi cluster.

So let's create a new data set with just the movies from cluster two.

We'll call it cluster two, and use the subset function to take a subset of movies only taking the observations for which clusterGroups is equal to 2.

Let's look at the first 10 titles in this cluster.

We can do this by typing cluster2$Titlecluster2$Title[1:10].

So it looks like good movies to recommend to Amy, according to our clustering algorithm, would be movies like Apollo 13 and Jurassic Park.

In this video we saw how clustering can be applied to create a movie recommendation system.

In the next video, we'll conclude by learning who ended up winning the million dollar Netflix prize.