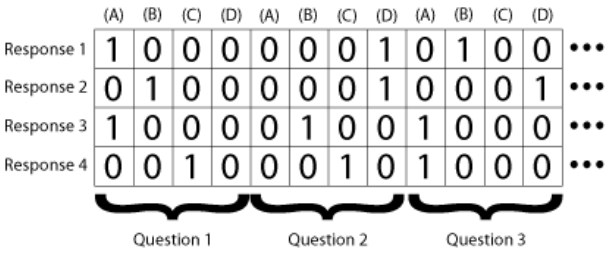
The Shape of Data
Exploring the geometry behind machine learning, data mining, etc.

K-modes

Posted on March 4, 2014

I recently read an interesting Wired story about Chris McKinlay (a fellow alum of Middlebury College), who used a clustering algorithm to understand the pool of users on the dating site OkCupid (and successfully used this information to improve his own profile.) The data involved was answers to multiple-choice questions, which is very similar to the categorical data that I discussed a few posts back. But, instead of translating the data into vectors, like I discussed in that post, McKinlay used an algorithm called K-modes that works directly on this type of data. So, I thought this would be a good excuse to write a post about K-modes and how it compares to translating the data into vectors and then running K-means.

First, I want to review how we could turn answers to multiple-choice questions into vector data, and what the K-means algorithm would do to it. To make things simple, lets say that we have a multiple-choice questionnaire with ten questions, each with four possible choices. A number of people fill out the questionnaire and we record their responses.



As I described in the post on token data, we can convert each of the filled-out questionnaires into a data point in a 40-dimensional space as follows: The first four dimensions/features will record the response to the first question. If they answered (A) to the first question, the first four values would be *(1,0,0,0)*. If they answered (B), it would be *(0,1,0,0)* and so on. Similarly, the next four features would record the response to the second question in the same way, and so on, as in the Figure to the right. So each questionnaire would give us a 40-dimensional vector with ten 1s and the remaining places all 0s.

Recall that K-means works by selecting a small number of special points called *centroids*, which are in the data space but not necessarily data points, and working out which of the centroids each data points is closest to. Then, it replaces each centroid with the new centroid/center of mass of the data points that were associated to it.

To understand how this works for the type of data that we get from a questionnaire, lets start by looking at the step where we find the new centroids. Given a collection of data points, we find their center of mass by adding them all together and then dividing by the number of data points. (This is basically an average, but we use the fancy term centroid because an average usually refers to a single number rather than a vector.) When we add the vectors together, we're just adding up all the 1s in a particular dimension/feature. So in the resulting vector, the value in the first spot will be equal to the number of questionnaires that answered (A) to the first question. The value in the second spot will be the number that answered (B) to the first question, etc. The value in the fifth spot will be the number who answered (A) to the second question and so on.

Next, we divide by the number of data points, which means we divide each entry of the vector by this number. The first spot in the resulting vector is the number of questionnaires that answered (A) to the first question, divided by the total number of questionnaires. This number will be between zero and one, and you can think of it as the percentage of questionnaires that answered (A) to the first question. (Technically, to get the percentage, you have to multiply the number by 100.) The number in the second spot is the percentage who answered (B), and so on. Think of this like a bar graph showing the responses to each question, where the value in each spot is the height of

the bar.

The step where we decide which centroid is closest to each data point is a little trickier. To find the standard Euclidean distance between a data point and a centroid, we subtract the number in each spot of the data vector from each spot of the centroid vector (and take the absolute value so that the resulting number is always positive.) Then we square each of these numbers, add them all up, then take the square root of the final number. (There are other possible types of distance to calculate, but that's a digression for another time.)

That's getting a bit more technical than I usually like, so let me just point out three key things to understand about this calculation: First, if a lot of the data points that were used to make the centroid agree with the new data point on a given question, this will make the distance lower (as we would hope.) Second, if a lot of the original data points agreed on a different answer than the new (Follow to a given question, then this will make the distance higher (again, as we would hope al data points disagree with the new data point, but are fairly evenly spread among t hen this will add a little bit to the distance, but not as much as if they all agreed on a s with the new data point.

What this process does is to picl onses that are most similar to each centroid, and thus to each other. In other words, w points that are closest to any given centroid will have a lot of responses in common wi we calculate the centroids the next time, there will be a stronger majority on each que

So, we run K-means by repeating this process over and over until the centroids stabilize/converge to places that define our new clusters. We can then read off the "typical" response to the first question in each cluster by looking at the values of the first four spots in each centroid, and choosing the one that's the highest. We can read off the "typical" response to the second question by looking at the next four entries, and so on. If the data set has well defined clusters, and we pick our *K* correctly, we would expect the values defining these "typical" responses to be pretty close to 1 (i.e. 100%).

The K-modes algorithm is based on a very similar idea, but as I mentioned above, it skips the intermediate step of transforming the questionnaire data into vectors. It consists of the same two steps, but they look slightly different. For the step in which we compute the centroids, we again start by adding up the number of questionnaires that responded with each possible answer to each of the questions. So far, this isn't too different; it's essentially the same as adding up the 40-dimensional vectors like we did with K-means.

However, instead of dividing by the number of questionnaires like we did with K-means, the K-modes algorithm simply records which answer to each question got the most votes. This is the *mode* of the responses -the most common answer – which is where the name *K-modes* comes from. So each centroid is in the same form as the original questionnaire data – a set of responses to the different questions – rather than a 40-dimensional vector.

The next step is again to calculate the "distance" from each data point to each centroid. For K-means, we were able to use the standard Euclidean distance, since we were working with vector data. For K-modes, we're going to have to come up with a notion of distance from scratch, but this turns out not to be too hard. The most obvious notion of distance is as follows: For each data point and each centroid, we can define the distance to be the number questions they disagree on. As with the Euclidean distance we used in K-means, when they agree on a question, this will make the distance lower, and when they disagree on a question, it will make the distance higher.

Note, however, that the third point about the Euclidean distance doesn't hold here: The K-modes centroids don't keep track of how close the margin was between the answer that that got the most responses and the second most. So, if the data point disagrees with the most popular response among the original data points, it gets the same penalty whether or not the original data points strongly agreed on this answer.

This difference is a trade-off rather than a deficiency. The problem with the way K-means calculates distances is that it can lead to centroids where no one answer is much higher than the others. Essentially, K-means never makes a strong decision about which data points to abandon. Since K-modes forces the centroids to make this decision, it can lead to much better defined clusters. Of course, for data where there aren't strong correlations to be found, having to make this decision (especially in the early rounds of K-means/K-modes) could make things worse.

As usual, the question of which algorithm is better depends entirely on the data set and the goals of the project. Both algorithms rely heavily on picking the right value for *K*, and the Wired article does a nice job of describing how McKinlay did this through trial and error. (I may have to borrow the lava lamp analogy.) McKinlay's analysis was complicated somewhat by the fact that each profile in the data set that he looked at answered a slightly different set of multiple choice questions. While there was a lot of overlap between the sets, each question would have had a lot of no-answer data. There are a number of ways one could deal with this, and I don't know which one McKinlay used, but as always the best solution to a problem like this depends on the particular data set.

Related

Clusters and DBScan
In "Clustering"

Statistics vs. Heuristics
In "Software engineering"

Modularity - Measuring cluster separation
In "Clustering"

This entry was posted in Clustering, Feature extraction. Bookmark the permalink.

4 Responses to *K-modes*

rrenaud *says:*
March 5, 2014 at 10:42 am

What I'd love to see is a discussion or characterization of problems when you expect K-modes will outperform K-means and vice versa. Likewise, mentioning particular problems where the K-means averaging step doesn't really make any sense and so it's not even really a consideration, compared to K-modes.

Reply

Jesse Johnson *says:*
March 5, 2014 at 12:32 pm

That's a good idea, though perhaps easier said than done. I'll have to think about this and see if I can come up with some good examples. Thanks!

Reply

outputlogic *says:*
February 12, 2015 at 7:26 am

Simple 2-d or 3-d visualization would be helpful to understand k-means and k-modes approaches

Reply

Pingback: *Sam svoj Kupid | Udomačena statistika*