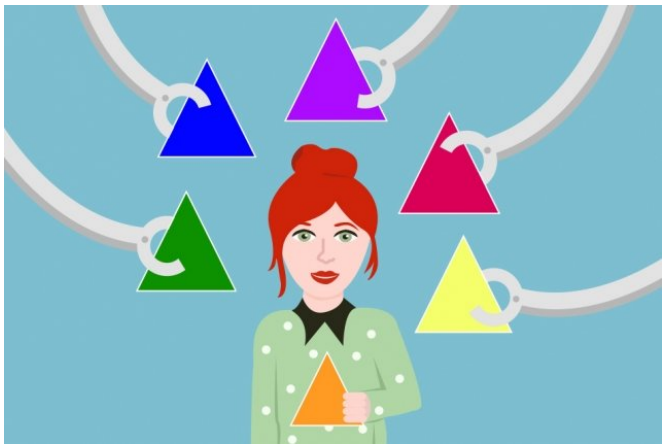
Try this! Researchers devise better recommendation algorithm

6 December 2017, by Larry Hardesty



To determine what products a given customer might like, recommendation systems at websites such as Amazon and Netflix look for other customers who have assigned similar ratings to a similar range of products, and extrapolate from there. Credit: Chelsea Turner/MIT

The recommendation systems at websites such as Amazon and Netflix use a technique called "collaborative filtering." To determine what products a given customer might like, they look for other customers who have assigned similar ratings to a similar range of products, and extrapolate from there.

The success of this approach depends vitally on the notion of similarity. Most recommendation systems use a measure called cosine similarity, which seems to work well in practice. Last year, at the Conference on Neural Information Processing Systems, MIT researchers used a new theoretical framework to demonstrate why, indeed, cosine similarity yields such good results.

This week, at the same conference, they are reporting that they have used their framework to construct a new recommendation algorithm that should work better than those in use today, particularly when ratings data is "sparse"—that is, when there is little overlap between the products reviewed and the ratings assigned by different customers.

The algorithm's basic strategy is simple: When trying to predict a customer's rating of a product, use not only the ratings from people with similar tastes but also the ratings from people who are similar to those people, and so on.

The idea is intuitive, but in practice, everything again hinges on the specific measure of similarity.

"If we're really generous, everybody will effectively look like each other," says Devavrat Shah, a professor of electrical engineering and computer science and senior author on the paper. "On the other hand, if we're really stringent, we're back to effectively just looking at nearest neighbors. Or putting it another way, when you move from a friend's preferences to a friend of a friend's, what is the noise introduced in the process, and is there a right way to quantify that noise so that we balance the signal we gain with the noise we introduce? Because of our model, we knew exactly what is the right thing to do."

All the angles

As it turns out, the right thing to do is to again use cosine similarity. Essentially, cosine similarity represents a customer's preferences as a line in a very high-dimensional space and quantifies similarity as the angle between two lines.

Suppose, for instance, that you have two points in a Cartesian plane, the two-dimensional coordinate system familiar from high school algebra. If you connect the points to the origin—the point with coordinates (0, 0)—you define an angle, and its cosine can be calculated from the point coordinates themselves.

If a movie-streaming service has, say, 5,000 titles in its database, then the ratings that any given user has assigned some subset of them defines a single point in a 5,000-dimensional space. Cosine similarity measures the angle between any two sets of ratings in that space.

When data is sparse, however, there may be so little overlap between users' ratings that cosine similarity is essentially meaningless. In that context, aggregating the data of many users becomes necessary.

The researchers' analysis is theoretical, but here's an example of how their algorithm might work in practice. For any given customer, it would select a small set—say, five—of those customers with the greatest cosine similarity and average their ratings. Then, for each of those customers, it would select five similar customers, average their ratings, and fold that average into the cumulative average. It would continue fanning out in this manner, building up an increasingly complete set of ratings, until it had enough data to make a reasonable estimate about the rating of the product of interest.

Filling in blanks

For Shah and his colleagues—first author Christina Lee PhD '17, who is a postdoc at Microsoft Research, and two of her Microsoft colleagues, Christian Borgs and Jennifer Chayes—devising such an algorithm wasn't the hard part. The challenge was proving that it would work well, and that's what the paper concentrates on.

Imagine a huge 2-D grid that maps all of a movie-streaming service's users against all its titles, with a number in each cell that corresponds to a movie that a given user has rated. Most users have rated only a handful of movies, so most of the grid is empty. The goal of a recommendation engine is to fill in the empty grid cells as accurately as possible.

Ordinarily, Shah says, a machine-learning system learns two things: the features of the data set that are useful for prediction, and the mathematical function that computes a prediction from those features. For purposes of predicting movie tastes, useful features might include a movie's genre, its box office performance, the number of Oscar nominations it received, the historical box-office success of its leads, its distributor, or any number of other things.

Each of a movie-streaming service's customers has his or her own value function: One might be inclined to rate a movie much more highly if it fits in the action genre and has a big budget; another might give a high rating to a movie that received numerous Oscar nominations and has a small, arty distributor.

Playing the odds

In the new analytic scheme, "You don't learn features; you don't learn functions," Shah says. But the researchers do assume that each user's value function stays the same: The relative weight that a user assigns to, say, genre and distributor doesn't change. The researchers also assume that each user's function is operating on the same set of movie features.

This, it turns out, provides enough consistency that it's possible to draw statistical inferences about the likelihood that one user's ratings will predict another's.

"When we sample a movie, we don't actually know what its feature is, so if we wanted to exactly predict the function, we wouldn't be able to," Lee says. "But if we just wanted to estimate the difference between users' functions, we can compute that difference."

Using their analytic framework, the researchers showed that, in cases of sparse data—which describes the situation of most online retailers—their "neighbor's-neighbor" algorithm should yield more accurate predictions than any known algorithm.

Translating between this type of theoretical algorithmic analysis and working computer systems, however, often requires some innovative engineering, so the researchers' next step is to try to apply their algorithm to real data.

"The algorithm they present is simple, intuitive, and elegant," says George Chen, an assistant professor

at Carnegie Mellon University's Heinz College of Public Policy and Information Systems, who was not involved in the research. "I'd be surprised if others haven't tried an algorithm that is similar, although Devavrat and Christina's paper with Christian Borgs and Jennifer Chayes presents, to my knowledge, the first theoretical performance guarantees for such an algorithm that handles the sparse sampling regime, which is what's most practically relevant in many scenarios."

More information: Thy Friend is My Friend: Iterative Collaborative Filtering for Sparse Matrix Estimation: papers.nips.cc/paper/7057-thy- … se-matrix-estimation

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

Provided by Massachusetts Institute of Technology