
Tackling the Netflix Challenge

Marc Pickett I, Doctoral Candidate
with Adam Anthony, Masters Student
and Donald Miner, Undergraduate Student
and Tim Oates, Assistant Professor

Cognition, Robotics, and Learning,
Department of Computer Science and Electrical Engineering,
College of Engineering and Information Technology,
University of Maryland, Baltimore County

Earlier this year, Netflix announced a \$1,000,000 prize for the first person or group who could outperform its own recommender system, “Cinematch”, by a 10% reduction in error. The goal isn’t to make recommendations explicitly, but to accurately *predict* what rating a given user will give for a particular movie. (The prediction doesn’t need to be a whole number.)

The “training” dataset given out by Netflix is simultaneously overwhelming and spare of details. It has over 480,000 users, 17,000 movies, and 100,000,000 “data-points” each consisting of a user-number, a movie-number, and the rating (a whole number from 1 to 5) that the user gave the movie. (The dataset also includes the date the rating was given, but neither our system nor Cinematch make use of this information.) That’s a lot of data compared to some of the “toy” problems our lab usually tests our algorithms on. Our typical datasets are ones like zoological data, or electronic books, that typically have less than a few million data-points. The Netflix data forces us to address less theoretical issues such as memory efficiency: 100,000,000 data-points won’t fit in your memory if you’re sloppy. At the same time, the data’s lacking in information that might be useful. All we know about the users is an arbitrarily assigned number. We don’t have any of the users’ demographic data: age, gender, where they live, etc.. The data about the movies is also spare. Netflix *does* provide a list of which movie number with the title of the movie, but we’re on our own if we want to get other features about the movie: genre, actors, director, language, etc., which could all be useful.

The “standard” technique for recommender systems is called “Collaborative Filtering”. Suppose you want to know what user, Alice, will rate film #10212. One way to do this would be to find all the other users who have

rated this film, and get the average rating. This technique isn’t so bad (only 10% worse than Cinematch), but you can do better. Suppose another user, Westy, gave the movie a 5 (the highest score). One thing we could do is look at the set of movies that both Alice and Westy have seen, and compute a “similarity”, and give a higher weight to Westy’s recommendation if their overlap is high. This is the core idea of collaborative filtering. We tried our own naive collaborative filtering approach just to get a “feel” for the problem, and, without too much tweaking, did only a little worse than Cinematch. With more tweaking, we figured we could do perhaps 1% better than that, but we decided that to make a substantial improvement, we’d need a more fundamental change.

This is where knowing a movie’s genre might come in handy. Suppose that we somehow knew that Alice tends share Westy’s taste for Westerns, but that she shares Romeo’s taste for Romances. If we know that a certain movie that Alice might want to watch is a Romance, then we should give more weight to Romeo’s opinion than Westy’s. This is where standard collaborative filtering fails: If Alice has rated about as many Westerns as Romances, she’d had roughly the same “similarity” to both Westy and Romeo, and standard collaborative filtering would give them both the same weight.

But Netflix didn’t provide us the genres of the movies. One thing we could do would be to grab data from the Internet Movie Database (IMDB), which has all the information we need. The first problem here is that Netflix gives no guarantee that the movie titles they give will match the title in the IMDB. So this would be a lot of “leg work”, sorting through the 17,000 titles and making sure they match properly. Our lab is more interested in general purpose learning algorithms. If

it's hacking, it's not interesting for us, even if hacking is the most direct approach for the prize. This is because hacking tends to be brittle. If you add a new movie to the dataset, you'll need to add their IMDB information as well. Another problem with grabbing the data off the IMDB is that the "genres" might not match up to how people's tastes work. For example, the movie *Plan 9 from Outer Space* by Ed Wood, might be classified as a Horror or a Sci-Fi movie, but people who like Horror or Sci-Fi typically don't like this movie. This is because it's considered by some critics to be "The Worst Movie Ever Made". For this reason, it appeals more to film buffs as a classic example of bad movie making. A more appropriate genre for a recommendation system for this movie might be "Movies-For-Film-Buffs".

This is where an algorithm that Tim and I developed last year, called "The Cruncher" (Pickett & Oates, 2005), comes in. The Cruncher takes in "raw" (or unlabeled) data and, by finding patterns, produces an "ontology", which is a hierarchical (or "heterarchical", technically) set of (possibly overlapping) categories. It's kind of like a taxonomy except items can be in multiple categories. So, when we applied The Cruncher to a standard zoological dataset, it created categories that roughly correspond to Mammals, Birds, and Fish, even though the dataset didn't explicitly contain these categories. They were *implicit* in the data. Biologists originally developed those categories *because* of the similarities of the animals.

We thought that useful "genres" could likewise be implicit in the Netflix data. Because it doesn't rely on people for labeling and "knowledge engineering", a downside to The Cruncher is that it needs *a lot* of raw data, which is what Netflix gave us. So the Netflix Challenge was a fairly straightforward application of The Cruncher (aside from the sheer size of the dataset). The genres that we found only roughly correspond to the genres in the IMDB (and they don't have labels, of course), but we think that these "organic genres" could be more useful than the "standard genres" for making recommendations. A point to note is that The Cruncher's genres simultaneously categorize movies and users. For example, a common category was the users that rated all three Lord of the Rings movies a 5. (Note that The Cruncher did this without us ever mentioning anything about Peter Jackson, Fantasy, Elijah Wood, or Tolkien.) This both tells you that the movies have something in common, but also that those users have something in common.

Another downside to The Cruncher is the time it takes. The algorithm is "quadratic-time complexity" in the

size of the data, which means that the time needed to process the data is proportional to the *square* of the number of users we process. So "Crunching" just 5% of the data (5,000,000 data-points) takes a week. 10% of the data would take a month. That's the bad news. The good news is that we don't need to Crunch the entire dataset to get a good idea of what the Ontology will look like. Despite this, We still have more work to do. The size of the dataset makes for slow progress. Trying out a new idea or a new strategy for the The Cruncher can take 2 weeks of computer time alone.

References

- Pickett, M., & Oates, T. (2005). The cruncher: Automatic concept formation using minimum description length. *proceedings of the 6th International Symposium on Abstraction, Reformulation and Approximation (SARA 2005)*. Springer Verlag.