Netflix Prize

In 2006 *Netflix* was just a mail-based DVD rental company (they weren't streaming videos yet). Customers with a subscription could rent as many DVDs as they liked, and Netflix *wanted* to keep posting DVDs to their customers. People who weren't using the service would realize, and cancel their subscription. Netflix kept up demand by recommending movies their customers would like, and they had data showing that better personalized recommendations led to higher customer retention. To improve recommendations further, Netflix launched a challenge — with a million dollar prize — to improve the root mean square error of their recommendation engine by 10%.

The rules of the competition and FAQ are still online if you want more detail. Much of how the competition was set up was well thought out. The way the leaderboard works, and limitations on the number of submissions, was since adopted in much the same form by Kaggle's competitions. As we'll see, they didn't give enough thought to privacy.

Wikipedia has a good, short summary: `https://en.wikipedia.org/wiki/Netflix_Prize`

SVD approach

One of the most significant approaches to the competition was referred to as "SVD". Simon Funk, one of the competitors, beat Netflix's existing system early on with a short and simple C program, which performed stochastic gradient descent on a simple model.

The model stated that the $C \times M$ matrix of movie ratings for the C customers and M movies can be approximately decomposed into the product of a tall thin $C \times K$ matrix and a short wide $K \times M$ matrix. This low-rank approximation is like a standard *truncated SVD* approximation, but without an intermediate diagonal matrix, which can be absorbed into the other matrices. A conventional SVD routine finds an approximation with the minimum possible square error, summed over every element of the matrix. However, the Netflix data matrix isn't fully observed (no customer rates every movie), so we minimize the sum of the square differences between the observed matrix M and its approximation, only at the observed elements. A conventional SVD routine can't fit this cost function. However, we can apply stochastic gradient descent to the cost function, where the thin rectangular movie and customer matrices contain the parameters of the model. The resulting approximate matrix can then be evaluated at *any* cell, giving predictions for ratings that haven't been observed.

One of the fitted matrices contains K learned features about each customer. The other contains K learned features about each movie. The inner product of these features is used to predict the customer's rating of a movie. One of the K indexes might correspond to "romance": the corresponding customer feature will take on large values if the customer likes romantic movies, and the movie feature will take on large values if it is a romantic movie. In this model customers can like multiple genres of movie, and no other customer has to have the same combination of tastes for the system to work. No genre labels are required to fit the model though: the fitting procedure learns the features for itself.

Rather than expand further, some of the successful Netflix competitors wrote a great and accessible article, fleshing out the details: Matrix factorization techniques for recommender systems, Koren et al., IEEE Computer 42(8):30–37, 2009.

More methods and the winning ensemble

Restricted Boltzmann Machines (RBMs) are a type of neural network, although unlike the feedforward networks we have covered in this course, they have random values for hidden units. A prediction system based on RBMs, along with many other methods, were also tried on the Netflix challenge. They didn't beat tuned combinations of SVD models. However, they made different mistakes. Averaging predictions from RBM- and SVD-based systems did better than either system alone.

In fact, the final winning entry used a large ensemble (collection) of many different methods, and an elaborate way to form the ensemble. Netflix did adopt variants of SVD and RBMs, but the performance gains of the full ensemble system weren't worth adopting in their production system. Effort was better spent addressing the needs of their new streaming services.

The Netflix challenge nicely demonstrated how far you can get with the right baseline system. It also demonstrated that there is a lot of room for creativity in machine learning: there were many different approaches to the same problem, none of which are fundamentally and uniquely "right".

Competitions and well-defined shared tasks can be useful for validating methods: there was a lot of skepticism from some parts of the community about RBMs — but an independent competition showed they really worked. Although there are also limits to the usefulness of challenges. When reaching the limits of what's possible, highly engineered solutions are required to make small improvements, and the resulting large systems may not actually be practical.

Some commentators suggested that the challenge was a "failure" because Netflix didn't adopt the final winning entry. I doubt Netflix saw it that way — they rapidly got a lot of immediately useful "free" consulting and publicity. Although Netflix probably does have one regret...

The privacy fall-out

The Netflix data gave an anonymous unique identifier to each customer, and the real names and release dates of each movie. Apart from that, the data just defined a sparse matrix with ratings for a subset of customer-movie pairs. I'll admit that in 2006 I didn't see any reason why releasing this data could be a problem. This was what Netflix said in their FAQ:

> ''Is there any customer information in the dataset that should be kept private?
>
> No, all customer identifying information has been removed; all that remains are ratings and dates. This follows our privacy policy, which you can review here. Even if, for example, you knew all your own ratings and their dates you probably couldn't identify them reliably in the data because only a small sample was included (less than one-tenth of our complete dataset) and that data was subject to perturbation. Of course, since you know all your own ratings that really isn't a privacy problem is it?"
>
> — `http://www.netflixprize.com/faq.html`

I'll also admit that I didn't even understand why they were bothering to "perturb the dataset", which meant they randomly changed some of the ratings from their true values. How could this data possibly be deemed sensitive?

Firstly, anonymizing data doesn't work if an adversary might have access to other data that they can correlate with your data. In this case the publicly-available Internet Movie Database (IMDB) contains movie ratings. It was possible to identify some people in the Netflix prize data, by comparing patterns of ratings to those found in IMDB.

The IMDB ratings are public already though, so how has any harm been done? Well, users of IMDB know that it is public, and so may choose not to rate movies of a certain political or sexual persuasion. However, the Netflix dataset did contain ratings of political and pornographic movies. By matching users to IMDB, some of these sensitive ratings were attached to identifiable individuals. Not surprisingly, the relevant people were upset, and some of them took legal action. The wikipedia article on the Netflix Prize has more details and references.

Any scientific study (including your projects) should consider ethical issues, including privacy, *before* it begins. Anything involving potentially-sensitive data (amongst other issues) requires some thought and oversight. Companies also have a duty of care to their customers

and wider society. The Netflix prize shows how easy it is to incorrectly dismiss data as not possibly sensitive.

For keen students: A seminal paper on privacy is: *Differential privacy*, Cynthia Dwork, Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, 2006. — Differential privacy is a theoretical framework for data-releasing mechanisms that are resistant to all possible attacks from adversaries with arbitrary side-information. There is recent and ongoing work on applying it to machine learning. For those interested in theory, differential privacy is also interesting as a principled mechanism for regularization: if a system is provably insensitive to individual details in the training set, it can't have overfitted to them.

Whether Netflix could have really run their competition under the constraints of differential privacy is unclear. It's possible they could have explored social rather than technological solutions: getting people to donate their data. But that approach would not be without its challenges either.

Check your understanding

Work out the details for stochastic gradient descent for the low-rank approximation (or "SVD") approach to modelling the sparse customer–movie ratings matrix. Given a rating for a customer–movie pair, which parameters of the model get updated and how? Use the form of these updates to explain why it is important to randomly initialize the parameters of the model.

Is fitting the SVD model to a sparse customer–movie rations matrix a convex optimization problem? Would it be convex if fitting a dense customer–movie matrix, where every customer rated every movie?

To think about: What data do companies and governments have about you?[1] Are you comfortable with the organizations themselves having this data, and the predictions they might make from it? What if it was shared with other organizations, and might that happen? How much of this data would it be acceptable to be posted on a public website? How many of these organizations can be trusted to have good enough security that the data will always remain secret? How much consent do you feel you have given for this data collection and the use of the resulting data? How does the ability of these organizations to build useful services with this data trade-off against these privacy issues?

1. Starting points: do you use the web, a credit/debit or loyalty card, a smart-phone, or use any services?