AC209a Data Science Project: Data Science with User Ratings and Reviews

Andrew, Sophie, Reiko, Nick

October 20, 2016

Introduction

Modern technologies unleash the possibilities for customising user experiences and thereby increase customer satisfaction by personalising content [2] based on the past preferences of a specific user. Content based recommendation systems can further be extended to not only provide customers with content that they are most likely to enjoy and associate with but also for advertisment purposes. It has been shown [1] that potential customers are more likely to respond to content that directly appeals to them. Such is the power of a system that is able to make these structured connections from an unstructured source of data, such as user reviews and ratings.

In this project, we aim to achieve the following:

1. Select and prepare a suitable database of user ratings and reviews from ... TODO
2. Qualitatively, and quantitatively evaluate features and metrics upon which to train a statistical learning model.
3. Implement and evaluate the following prediction and recommendation engines: simple user rating prediction, content based sentiment and recommendation, clustering of product and collaborative filtering.
4. Decide upon the most appropriate recommendation based system for a particular setting and discuss some ethical considerations to be evaluated when making content based recommendations.

Literature Review

Online systems encapsulate a 'long tail' effect [3] where many more products and servies are available than is typically seen in physically constrained retail stores. Thus the need for a recommendation system arises: to quickly and easily allow users to find products that are appealing to them.

Due to the text and qualitative nature of a product's content description and metadata, these systems use a TF.IDF [2] system to enumerate the occurance of certain key words in comparison to that word's occurance in other documents. Thus quantitative vectors can be constructed from the qualitative nature of these descriptions. Simple Content-Based filtering directly compares any two document vectors (using cosine angle comparisons or techniques such as the Jaccard distance) [3] whereas Collaborative-Filtering makes an assumption based on the similar likes and dislikes of users of the similarities among products. Collaborative-Filtering requires a deeper understanding of the subject matter of the content, as groups are not always explicitly applicable to these data [3]. Two examples of complications that arise in collaborative filtering are:

1. The process of generating recommendations from similar users does not translate directly into generating recommendations from sets of similar items.
2. Users may like multiple genres and have only one of those genres in common with a similar user. We need to avoid recommending the other genre that the second user may not care for.

For a given utility matrix, which contains feature data (usually including TF.IDF scores) for users or items we wish to predict empty values in that matrix to make the relevant recommendation [3]. To predict an empty value, (I,U) in the matrix, we can find n users most similar to user U, normalize their ratings for item I, average the difference for the users who have rated I, and add it to the average rating for U. We could also find m items most similar to I and take the average rating that U has given to these items. To generate recommendations, we seek to at least generate most of the entries for a user U that are blank but have a high estimated value. In general, item-item similarity will be more accurate, because of the complication of users who like multiple genres, but this requires more computation.

Note to us: possible inclusion of the computation overhead of this all and the need to store all of the given information in a prebuilt matrix before time. User preferences generally don't change over time and thus the need to change (and re-compute) the matrix is not too overwhelming.

We now turn our attention to the specific case study of the Netflix challenge where Netflix offerered a reward of $\$1,000,000$ for any algorithm that could beat the current 'CineMatch' algorithm's *root-mean-square-error* (RMSE) by more than 10% [3]. This challenge was successfully won in 2009. It is noted that the standard set by CineMatch only obtained a marginally (3%) lower RMSE than the average rating across all users for each movie and the average rating across all movies for each user.

The grand prize solution to the Netflix challenge [1] consists of multidimensional solution to the problem of recommendation. Techniques ranged from computing baseline ratings for movies (which measures their overall quality, independently from a specific viewer) and users (which measures how critically they tend to rate movies generally, independently from any specific movies).

The BellKor team had the insights to make the baselines time-dependent. It is reasonable to expect that a movie's general popularity might change over time (e.g. as it becomes more or less culturally relevant) or that a user's baseline might shift from day to day (e.g. depending on their mood). Thus, they compute baselines for every movie and every user as a function of time, although in practice they don't use continuous functions, but rather a set of discrete estimates for time bins spanning the total time range of the dataset.

The timescale over which these quantities vary is different; we expect movie baselines to shift slowly over months or years and user baselines to shift much more frequently. For movies, they bin their baselines in 10-week intervals. For users, because there isn't enough data to fully establish 1-day bins, they do a more complicated calculation that allows for gradual drift effects over long time periods as well as sudden "spikes" that are computed from day-specific data. This allows them to account for both long and short term changes in user baselines (which they note may even be caused by multiple people using the same user account, which many of us know from experience is a fairly common practice).

One other important term they factor into their (movie and user-specific) baseline rating calculation is an adjustment to the movie baseline based on the number of other ratings users who rated that movie also gave on the same day. The intuitive justification they give for this adjustment is that certain movies tend to be rated differently when a user is rating them individually vs. alongside a lot of other movies in bulk. Those two processes, from a user-psychological perspective, are very distinct, yet they appear commingled in the same dataset. In particular, the paper argues that when users are bulk-rating movies, they tend to only rate movies that are particularly memorable over a long period of time, either for positive or negative reasons. In these cases, it may be that 50% of the population will like the movie unmemorably, but another 50% will hate it very memorably, so that only the negative ratings show up during bulk-rating sessions (which may be the most common type of rating session). This can bias recommendations in an unhelpful way, and so the extra term they include which controls for frequency helps remove that bias.

Note that although most of this time-specific information is most useful for filling in predictions about the past, rather than the future (which we want to predict), the calculated baselines still help the model make better future predictions, on average.

So, finally, all of these baseline components are computed again by a (more complex) gradient descent solution to a least-squares problem, with additional λ terms to regularize the extra parameters (stil using effectively a Ridge penalty).

After obtaining the baseline factors, they trained a variety of models to predict $r_{ui} - b_{ui}$, i.e. the difference between the true rating and the baseline. Models they used include time-modified versions of matrix factorization, nearest neighbor algorithms, restricted Boltzmann machines, and more. Their prediction for a given rating was a blend (weighted average) of the predictions of each of these models, and the weights of that average were also user and movie-specific. The way they compute the blend weights for each user and movie is complex (and will require reading several more papers to start grasping), but they used gradient boosted decision trees for some of the blend weight calculations.

References

[1] The BellKor Solution to the Netflix Grand Prize, *Yehuda Koren, (2009)*

[2] Beginners Guide to learn about Content Based Recommender Engines,
`https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommende`

[3] Chapter 9: Recommendation Systems