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

Andrew Ross, Sophie Hilgard, Reiko Nishihara, Nick Hoernle

October 21, 2016

Introduction and Problem Formulation

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. Specifically we have chosen the dataset from the 'Yelp Dataset Challenge' (`https://www.yelp.com/dataset_challenge`)
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] method to enumerate the occurance and importance 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 review 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 or 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.

Another way to compute blank values is by performing UV-decomposition on the utility matrix. We initialize U and V and then iterate to find choices of U and V that minimize *root-mean-square-error* RMSE between values of UV and values of the actual utility matrix. Then, we can use entries of UV as a prediction for entries that are blank in the utility matrix. Intuitively, this method works off the assumption that there are only a small (comparatively) number of features that determine the final value in the utility matrix. Note that we again need to perform normalization for user rating scales and item quality before performing this algorithm. Additionally, it is possible to reach local minima in this algorithm, so we must consider random restarts.

Due to the large number of features, users and items that are present in any one dataset the computational overhead of constructing classifying matrices is vast [3]. Luckily, user preferences tend not to vary too drastically over time and thus these algorithms can be run periodically at a low frequency to update with new and relevant data.

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 then current 'CineMatch' algorithm's 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 movie).

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 slowly over time (e.g. as it becomes more or less culturally relevant). Thus, including time based data as a feature was a reasonable design choice. The team computed 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 [1].

The timescale over which these quantities vary is different: we expect movie baselines to shift slowly over months or years and user baselines to show a higher frequency change. Concretely, 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).

Another important term that the team factored 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 it on the same day. The intuitive justification they give for this design choice is that certain movies tend to be rated differently when a user is rating them individually vs. alongside 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 dislike it 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 confounding bias.

After obtaining the baseline factors, the team 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 used gradient boosted decision trees for some of the blend weight calculations.

We thus have seen a general structured layout for tackling ratings based problems from the 'Chapter 9' article, with a more introductory presentation of how to calculate TF.IDF scores from the 'Beginners Guide to Recommender Engines'. Lastly, the 'Netflix' specific case study has demonstrated how a deep analysis of the niche problem is required in order to use relevant data in making a useful prediction (i.e. certain features that were used here may not be applicable to other cases).

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