Predictive Recommendation Engine
Using MovieLens Dataset

Final Project
Milestone #5

By:

Timur Zambalayev
Joshua Coffie

Harvard University

CS-109A, Fall 2016
Introduction to Data Science

Proposal of Future Work

Through milestone #4 we determine the baseline model performance was best determined by decoupling the item and user variables from our dataset and then taking into account both item (movie) and user effects independently as predictors. This method uses a regularization parameter, lambda, which we tuned to provide the highest R^2 possible and it yielded a baseline $R^2$of .28. For this reason, any model that we use for the final stage of our project should yield a better R^2 than that baseline model, and with reasonably acceptable performance in proportion to the return of accuracy.

Other components that we will want to address in the final project are the following measure of the performance of a recommendation engine, taken from the "Recommender Systems Handbook:" [1]

1. *User Preference* (having users test out the models to determine how they would rate it)
2. *Prediction Accuracy* (discussed above, but also taking into account % classified correctly)
3. *Coverage* (the percentage of all items that can ever be recommended)
4. *Confidence* (the system's trust in its recommendations or predictions, which can grow with the amount of data it has to work with)
5. *Trust* (the user's trust in the system's recommendations, which may lead us to introduce prior beliefs to lead the system to recommend "safer" options for users so that it is "right" more often than not)
6. *Novelty* (recommendations for items that the user did not know about)
7. *Serendipity* (how surprising the successful recommendations are to users)
8. *Diversity* (a measure of item-to-item similarity – recommending the same movie a user likes, for instance, isn't particularly helpful)
9. *Utility* (the expected utility of the recommendation – does this recommendation engine result in the service being used more, for instance)
10. *Risk* (minimizing risk in recommending items)
11. *Robustness* (the stability of the recommendation in the presence of fake information)
12. *Privacy* (no disclosure of private information from the user)
13. *Adaptivity* (how well the model adapts to changing title availability and trends in preference)

Using these additional variables in measuring the effectiveness of our model apart from simply a measure of R^2 will allow us to build the most usable model possible. By using this approach, however, we will need to establish a weighting system for each of the above variables to determine how important they are in developing our model.

As for our approach in building the best model for our recommendation engine, we'd like to explore both content-based recommendations and collaborative filtering, using a variety of techniques for each.

1. Content based recommendations

    a. Leverage item representation based upon actors, director, genre, year of release, etc.
2. Collaborative filtering
    a. Measuring similarity using Jaccard or Cosine Distance
    b. Clustering users and items, as the utility matrix is scarce
    c. Dimensionality reduction using UV-decomposition

Some other methods and approaches that we can explore to optimize our model include:

1. *Incorporating context into the recommender* system (by gathering data on the task that the user hopes to accomplish). This would mean presenting a variety of options that illicit feedback from the user to gain insight into what they're looking to view during their session. For instance, if we're approaching the holidays, we may recommend some holiday films simply to determine is that's the reason the user is looking for a recommendation for a movie on December 25th.
2. *Incorporating trends into the recommender system*. By reviewing the user's history, especially in the case of a long-term relationship, we can determine the user's desire to be presented with modern trends and items. For instance, if we determine that the user prefers to watch new releases and doesn't like to watch older movies (maybe they don't like the graphics and lack of realism, for instance) we can skew our recommendation to always incorporate newer releases and never recommend a title that is beyond X years old.
3. *Testing the recommender system on another external dataset*. To truly judge the effectiveness of the recommendation engine, we can use another dataset to measure its performance. For instance, we're using MovieLens review data to train and perform precursory testing on its performance. When the model is optimized for our current dataset, we can try it out on Rotten Tomatoes or Netflix data or a similar provider to determine the suitability of the model given a different dataset. The "Recommender Systems Handbook" also suggests using a "test set [that] contains many more ratings by users that do not rate much and are therefore harder to predict. In a way, this represents real requirements for a CF system, which needs to predict new ratings from older ones, and to equally address all users, not just the heavy raters." [1, pg. 150]

Sources

1. Ricci, Francesco, Lior Rokach, Bracha Shapira, and Paul B. Kantor. Recommender Systems Handbook. New York: Springer, 2011.

Github repo: https://github.com/starbuck10/CS109a_DataScience_UserRatings_Team_Project