An experiment on recommender systems using Amazon reviews dataset

Tomás Pica del Canto
tomas.pica@gmail.com

Abstract: Here I present a collaborative filtering real-world application and tutorial of a recommender system using singular value decomposition techniques using python and python-recsys, over a dataset of real reviews exported from Amazon.com

1 Introduction

Amazon.com is regarded as some as the canonical example of a recommender system, they face a difficult but also very rewarding and lucrative problem: to predict what products their users will like more, knowing only what they like or dislike now.

My hypothesis is that this predictive model is reproducible with simple techniques and libraries, as I will demonstrate in this paper.

2 Collaborative Filtering Process

The dataset was obtained thanks to Stanford's SNAP[1], with permission of Julian McAuley[2] for educational purposes.

The dataset consists of 34,686,770 reviews, written by 6,643,669 users on 2,441,053 products. Each review contains a review score on a discrete scale of 1 to 5, as well as more data not relevant to this study.

2.1 Cleaning and Filtering

The first challenge is to obtain a clean dataset for use with our model. As our solution is based on a singular value decomposition approach, we need a clean dataset consisting just of user identifiers, product identifiers, and the review score linking them.

That is not hard to do and after parsing the dataset with a simple script[3], we have a dataset that python-recsys SVD model can understand.

---

[1] http://snap.stanford.edu/data/web-Amazon.html
[2] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. RecSys, 2013.
[3] https://github.com/tomaspdc/amazon-recsys/blob/master/src/clean/clean.py

2.2 Error Measuring and Preventing Overfitting

Once I obtained the full clean dataset, I kept a subset as a "true blind" to measure success on the final model used.

From the rest, I kept again a subset to serve as "second blind" to measure overfitting errors, and the remaining reviews were used as training/test subsets.

The remaining training/tests subset was sliced in several training/tests subsets, each of those composed of random samples of reviews from the original training/test subset, but of different sizes (from 100,000 reviews to 11,000,000, in steps of 100,000).

Each of those were trained and tested, measuring the average over five folds of the mean absolute errors and the root mean squared errors of the predictions[4][5].

After that, each of them were tested again the "second blind" subset and their errors measured in the same way[6][7].

The model chosen in the end minimized mean absolute errors against the "second blind" subset. Once compared against the "true blind" subsets, its mean absolute error was 0.766031.

3 Conclusions

Within a few simple steps, a viable predictive model was found, which demonstrates the viability of building a recommender system.

The disadvantages were obvious and expected as well: the "cold start" problem shows in the errors, although not too heavily, it's probably better in the long term to keep retraining your model as your data grows.

The other disadvantages are not big ones: (1) the amount of storage the model needs is huge, but storage is cheap, and (2) the amount of time it takes to make a prediction makes it non-viable for real-time prediction, but can be easily computed and stored in parallel, periodically, for real-time retrieval once needed.

---

[4] https://github.com/tomaspdc/amazon-recsys/blob/master/plots/mae-rmse-train-test.csv

[5] https://github.com/tomaspdc/amazon-recsys/blob/master/imgs/train-test.png

[6] https://github.com/tomaspdc/amazon-recsys/blob/master/plots/mae-rmse-second-blind.csv

[7] https://github.com/tomaspdc/amazon-recsys/blob/master/imgs/second-blind.png