CS 229 Final Project - Using machine learning to enhance a collaborative filtering recommendation system for Yelp

Chris Guthrie

Abstract

In this paper I present my investigation of machine learning as a tool to augment collaborative filtering in a recommendation system learned on a sparse dataset. Although methodological flaws and scaling difficulties limited the conclusiveness of my results, I believe lessons learned will enable me to successfully extend the project on my own in the future.

Introduction

Yelp

Yelp offers users high-quality local search and reviews for businesses, in particular restaurants and retail outlets. Users engage with the app by searching, writing reviews, rating businesses, connecting with other users and "checking in" at businesses. Two factors make Yelp an ideal product for a recommendation system analysis: first, Yelp's product depends heavily on search and business discovery, making ratings prediction valuable from a business standpoint; and second, Yelp users generate a large amount of relevant, highly structured data.

Project objectives

The goal of the project was to implement a ratings predictor for Yelp (i.e. a machine capable of taking a user and a business and predicting how the user would rate that business). A good ratings predictor potentially would have product implications for Yelp - for example, it could form the basis for a recommendation system or more personalized search results rankings.

The standard method for recommendation systems is collaborative filtering (explained below in detail). I wanted to see if I could enhance collaborative filtering predictions by incorporating:

1. Multiple collaborative filtering methods
2. Signals derived from structured content

into a sophisticated, multi-input predictor.

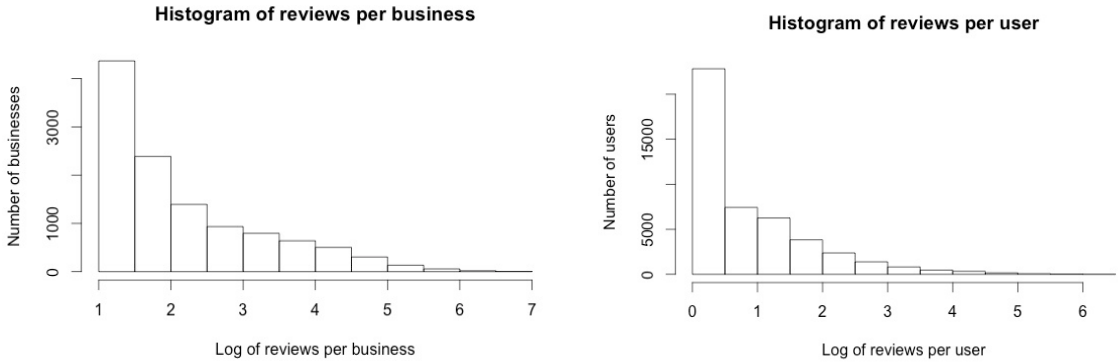
I implemented the predictor using a selection of out-of-the-box regressors, where features are drawn both from the outputs of a collaborative filtering process and from structured Yelp data.

I measured the ratings predictor through the mean square error metric. I chose MSE because it is a highly balanced metric that requires little explanation to understand. Additionally, all standard regressors efficiently support MSE. For Yelp's specific purposes, it could be interesting to evaluate different metrics that perhaps have different business implications - e.g. finding the ROC curve on the proportion of predictions that are off by more than 1 rating star or adjusting the error metric to punish false 4\5 star ratings more severely. Experimenting with error metrics is something that I'd like to explore in the future.

The Yelp dataset

Yelp offers a freely available “challenge” dataset consisting of data for the Phoenix, AZ area, with highly structured data on businesses, reviews, users, and user checkins. Since we focus our efforts on collaborative filtering, the review data has the most implications for us.

The dataset has 229906 reviews spanning 11537 businesses and 45981 users. The distribution of reviews per user and reviews per business is heavily skewed toward the low end.



This presents challenges for collaborative filtering, since reliable vector-based similarity metrics depend on high numbers of shared ratings. For users who have rated less than 3 businesses, for example, collaborative filtering becomes impossible using the metric I chose (explained in the next section).

Features

Collaborative filtering

Collaborative filtering typically attempts to predict ratings based on a ratings matrix indexed by user and business, which we’ll refer to as the ‘user-business matrix’. The collaborative filtering algorithm is very simple - it uses the matrix to compute a similarity score between any two users or businesses. The similarity metric can vary, and in my case I chose to compute the Pearson correlation of the corresponding row/column vectors weighted by a significance factor based on the Jaccard similarity of the vectors. This metric was a poor choice, since the vectors were very sparse and the majority of vectors had Jaccard similarity and Pearson correlation of zero.

Independent of the metric used, typically the top k similar users or businesses (that have been rated by the user or business in question) are selected, and the output of CF is a similarity-weighted average of the ratings given. I computed collaborative filtering averages under a user-user CF scheme as well as a business-business CF scheme and used both outputs as inputs to my regressors.

Content-based features

I had planned to integrate a number of content-based signals from the Yelp data, but the only feature I had a chance to analyze was the user’s average rating for businesses that share a category with the business in question, due to time constraints. Other content-based features that could be analyzed are suggested under the ‘Future Areas of Exploration’ section.

Summary features

I computed average rating scores for each business and user, the distribution of user rating scores, and the number of ratings for each business and by each user. For every rating, I added these features as signals to the regressors.

Regressors

I chose to use two different classes of regressors in my exploration, since multiple regressors increased the chance of positive results and could have given me better insight into the most effective ways machine learning can enhance collaborative filtering.

Linear regression

I chose to use a linear regressor for two reasons:

1. There was an a priori reason to think that most of the features I chose (such as average user rating or collaborative filtering guess) should be linearly related to the output rating
2. Linear regression allows for easy interpretability of features, so I could easily separate the explanatory power of collaborative filtering features as opposed to content-based features or summary statistics

Decision tree regression

Decision trees are a topic I had heard about before entering 229, and I was looking for an opportunity to learn more about them and explore their effectiveness as regressors. I chose to evaluate three regressors based on decision trees (all from scikit-learn) - a simple decision tree regressor, a random forest regressor (which randomly generates a set of trees and averages over the results to create a final prediction), and an AdaBoost regressor, which combines multiple decision trees and uses the statistical technique of boosting to reduce variance.

Method

Tools

I implemented the project entirely in Python, representing datasets and the user-business matrices in sparse SciPy matrices. I used scikit-learn for learning and model optimization.

Dataset creation

For the entire course of my investigation, I created training and testing datasets by randomly splitting the set of review data. This was inadvertent. I had intended to select test and training sets that were complete sub-matrices of the user-business matrix - i.e. by including ratings of the form (user, business) where both user and business were in a specified set of values, but I made a technical error. Because I randomly split the data based on individual reviews, I significantly undermined the effectiveness of the regressors and of the collaborative filtering step, by systematically biasing many key features and making collaborative filtering less statistically significant.

Feature extraction

I built a sparse user-business matrix (stored in multiple ways to optimize access), in addition to similarity matrices and appropriate methods to efficiently query these matrices in parallel. I performed CF on

these matrices, and computed some summary statistics for users and businesses that I used as inputs to the regressors.

Challenges with data size

My largest struggle with scaling my working small-scale regression model to the entire Yelp dataset was with computing similarity scores between users and between businesses. Despite using highly efficient representations of the data, my exact algorithm was too computationally complex to feasibly run on such a large dataset. Instead of trying to optimize speed, memoize every computation I made, and parallel program through multithreading, I should have instead searched out algorithms that would've made it easier to approximate a similarity metric. Alternatively, I should have seeked out computing resources with enough memory to support my computations across multiple training iterations.

The result of these struggles was that I limited the number of reviews, users, and businesses I included in my training/testing sets. Unfortunately, I mis-implemented this limitation and inadvertently made the review data significantly more sparse, compounding my troubles.

Optimizing collaborative filtering parameters

I tested across different values for k_{user} and $k_{business}$, the basic parameters for collaborative filtering, with cross-validation. I used a single scale for both (based on difference from median reviews per user and median reviews per business).

Model optimization

For the tree-based regressors, I ran cross-validation tests to optimize for the maximum tree depth hyperparameter. Ultimately, due to lack of results for reasons relating to feature selection, I did not further explore model optimizations.

Results

Baseline

I found that regressors based solely on summary features performed at an average cross-validated MSE of 0.32 on a dataset with 20k ratings samples. A simple linear regression model performed best with cross-validated MSE of 0.27.

Effect of collaborative filtering

In sum, I found that on data sizes of less than 60k (for given user and business sizes), collaborative filtering had practically no effect on the regression fit compared to the 20k baseline. This was confirmed by an analysis of the coefficients and significance of the linear regression fit on the collaborative filtering features - the coefficients were consistently 5-10 orders of magnitude smaller than comparably scaled features, and the fit found that the CF features explained no variance in the data.

I attribute the absolute ineffectiveness of collaborative filtering to sparseness of the data and the lack of significant correlations between users or between businesses.

Effect of sample size

Due to a lack of regularization in my fits, as data sizes increased up to 60k, cross-validated MSE increased as well (since the regressors were based almost entirely on summary statistics).

Conclusion

Getting positive results on a machine learning project has more hurdles than I had anticipated.

Future Areas of Exploration

Other methodologies

Faster similarity metrics

Instead of computing an exact similarity metric for every user-user and business-business pair in every dataset, it could be productive to use nearest-neighbor search via locality-sensitive hashing or clustering to find similar users/businesses.

Graph-based collaborative filtering

I saw another student who had solved the sparseness problem I encountered by constructing a bipartite user-item (or in my case, a user-business) graph with edges representing a review, then performing network analysis on this graph to find similar users, even if the users have no overlap in rated businesses.

Weight ratings by votes and recency

Since these signals are in the dataset and give a good idea of the reliability of a rating, some sort of weighting mechanism would likely improve overall performance.

Add an SVR regressor and optimize it

This would make the analysis more complete.

Other features to explore

Review text as a content signal

It would be interesting to mine review text to extract content features for businesses (and possibly for users) to construct a more complete content profile. There are a number of interesting algorithms (dimensionality reduction algorithms, NLP algorithms) that would be very cool to investigate.

Review text as a similarity metric

It would be possible to build a collaborative filtering mechanism around review text in addition to one built around ratings. Using review text as a similarity metric would help solve some of the sparseness problems I encountered, since you can come up with a metric between any two users that have written a review or any two businesses that have been reviewed.

Business location as a similarity metric

As with review text, business location solves the sparseness problems I encountered and could be a strong signal for user preference.