Collaborative Filtering Recommender Systems

-Rahul Makhijani, Saleh Samaneh, Megh Mehta

ABSTRACT - Aim to implement sparse matrix completion algorithms and principles of recommender systems to develop a predictive user-restaurant rating model. In particular, we implement the two primary forms of collaborative filtering - neighborhood and latent factor models to our Yelp data set. In the k-neighborhood models, we are looking to exploit the similarity of our items and users so as to group them in clusters and make predictions based on the available ratings. Contrastingly, in the latent factor model, we use matrix factorization techniques to derive the hidden similarity layer between the users and the items. All our model comparisons and evaluations are done using the RMSE metric.

1/ Introduction

High-quality recommender systems are the main drivers for success of many online services such as Netflix, Yelp, eBay and Amazon among others. The 5-star rating parameter in Yelp is a critical parameter in determining whether a user will click to find out more about a particular business, or just scroll on. In fact, a Harvard study states "a one star increase in Yelp rating leads to a 5-9% increase in revenue." Hence it is apparent why there is such a strong commercial incentive for good user recommender systems. They have a direct impact on a growing number of businesses.

Over the years, Yelp has managed to greatly increase the amount of clientele information in its database and our goal is to evaluate the predictive performance of various collaborative filtering techniques on this data set. This paper will take on the following structure:

- Short explanation of the data set used for testing our recommendation systems.
- Brief discussion on the performance metric used in our model evaluation.
- Description and evaluation of our k-neighborhood and latent-factor models against the initial baseline model.
- Conclusion and future prospects.

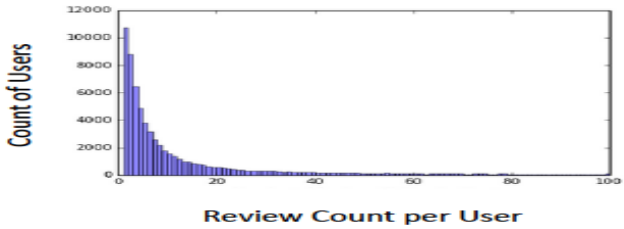
2/ Data set description

We implement our various models on the Yelp Dataset publicly available online for all at: http://www.yelp.com/dataset_challenge. It contains actual user, business and users' review data from Phoenix, Las Vegas, Madison, Waterloo and Edinburgh. To put into context, we have data for 42 153 businesses, 252 898 users with 1 125 458 reviews. The data was given in the following format:

review

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

For the purpose of this paper, we essentially make use of the review.json and business.json files from which we extract the relevant data in order to create our user-business ratings matrix.



Despite the large amount of data available to us, as we notice in the graph above, the distribution of reviews per user is heavily skewed to the low end and so this presents a challenge to our collaborative filtering methods that rely heavily on a high number of shared-ratings between users or restaurants in order to create reliable vector-based similarity metrics. To a certain extent, we try to circumvent this

problem by restricting our attention to restaurants in the city of Las Vegas only. This gave us a slightly denser matrix on which we could better perform our collaborative filtering techniques.

3/ Error metrics

The error metric we look at throughout this paper is probably the most popular one when evaluating recommender systems. The Root mean squared error is a prediction accuracy metric that tries to answer the question of how close the prediction ratings are to the true ratings. It uses squared deviations and so larger errors tend to get amplified.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}}$$

p_i = predicted rating

a_i = actual rating

We evaluated all of our models through a 5-cross validation RMSE, where we train the model on 80% of the data and then test its accuracy on the remaining 20%.

4/ Predictive methods

Given our user-business ratings matrix for restaurants in Las Vegas, we aim to fill in the missing ratings in our sparse matrix. We will look at both nearest neighbor and matrix factorization methods in order to assess their predictive performance. However, since most collaborative filtering methods start off with a simple baseline model, we will begin by implementing the following model to our data:

$$r_{ui} = \mu + \beta_u + \beta_i$$

μ = global average rating

β_u = user bias

β_i = restaurant bias

In this model, we take into account the mean rating given by the user and the mean of the restaurants in order to remove the presence of any possible bias when predicting. (i.e. a user has the tendency to rate everything higher than it's "true" rating). We get an RMSE of 1.204 on our predictions.

An issue we encounter is notably with outliers, such as very low "sandbag" ratings that bring the average down, especially when dealing with sparse data such as ours. One solution to this problem would be to assign weights to the different ratings instead of just taking the means. This is where we would look to incorporate our k-nearest neighbor collaborative filtering methods, which exploit any similarity existing between users/items.

There are many possible ways of measuring the similarity between users/items and in this case the Pearson r correlation would be the most reliable metric.

Similarity between users u & v

$$sim(u,v) = \frac{\sum_{I \in u,v}(r_{u,i} - \hat{r}_u)(r_{v,i} - \hat{r}_v)}{\sqrt{\sum_{I \in u,v}(r_{u,i} - \hat{r}_u)^2 (r_{v,i} - \hat{r}_v)^2}}$$

$\hat{r}_u$ = mean rating user u

$r_{u,i}$ = rating user u assigns to item i

For example, the Jacard distance metric would not be the most effective, as it wouldn't take into account the 1 to 5 ratings assigned to each restaurant by each user. Once we calculate the Pearson r similarity correlation metric, we make clusters of the k most similar users/items.

Method 1: User-User k-neighborhood model

In order to implement this initial method, for any given user i, we begin by calculating the similarity between this user and all other users. Then we select its k-nearest neighbors based on this similarity measure and finish by computing the predicted rating for i based on a weighted combination of the k-nearest neighbor ratings.

$$p_{u,i} = \bar{r}_u + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \bar{r}_{u'})}{\sum_{u' \in N} |s(u,u')|}$$

$p_{u,i}$: user u's prediction for restaurant j

$\bar{r}_u$: mean prediction for user u

$s(u,u')$: similarity between user u and user u'

$r_{u',i}$: rating of user u' for restaurant i

N: neighborhood of user u

This model gives us an RMSE of 1.193 on our predictions, which is a slight improvement on our baseline model.

Issues we encounter while running this model is that at times we find users with no ratings history except for the one rating we are trying to predict, or we find users whose neighbors had never rated the restaurant for which we want to predict. This model

also suffers from scalability problems, as the user base grows. Hence, it might be more beneficial to focus our attention towards the item space instead of the user space.

Method 2: Item-Item k-neighborhood model

Item–item CF uses similarities between the rating patterns of items. If two items tend to have the same users like and dislike them, then they are similar and users are expected to have similar preferences for similar items. Again, as in the user case, we look at the k-nearest neighbours of any given restaurant from the set of restaurants each user has already rated, using the Pearson r correlation similarity metric and then take a weighted average of the selected k restaurants to compute a prediction for the rating of the targeted restaurant.

$$\text{Pred}_{mn} = \frac{\sum_{j \in N_u^K(m)} sim(m,j) R_{ju}}{\sum_{j \in N_u^K(m)} |sim(m,j)|}$$

$N_u^K(m) = \{j : j \text{ belongs to the K most similar restaurants m that user u has rated}\}$

Pred_{mn} = prediction of restaurant m by user u

sim(m,j) = adjusted pearson correlation for restaurants m and j

We get an RMSE of 1.157 which is a marked improvement compared to the user-user k-neighborhood model.

Considering the item-space rather than the user space takes care of several crucial problems. Namely, the search for similar users in the high-dimensional space of user is computationally more expensive and restricting our attention to the item-space takes into account the problem of new user profiles having comparatively fewer ratings for businesses. However, sparseness of the matrix is a recurring challenge we encounter, since the reliability of vector-based similarity matrices depends on the number of shared ratings.

Method 3: Item-Item collaborative filtering (baseline model included)

This method is identical to the previous one except for the fact that we include the baseline model this time. We manage to correct the negative predictions by averaging the distance from the baseline predictor. It also helps extend collaborative filtering to large user bases. Since we subtract the baseline estimate from the ratings matrix, we get a normalized matrix to which we just need to apply the similarities.

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

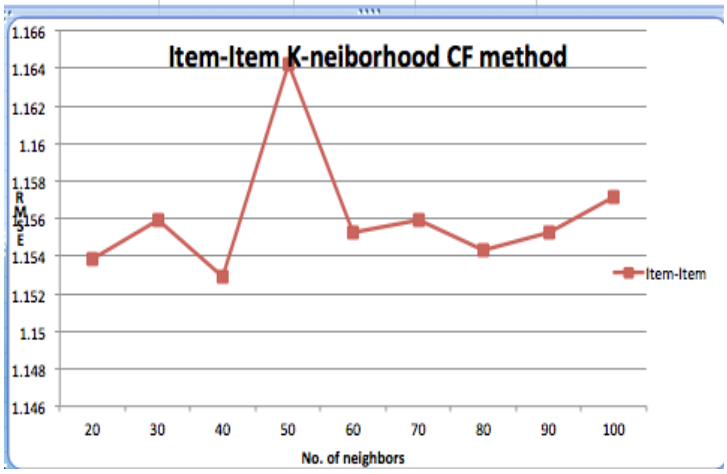
$$b_{xi} = \mu + b_x + b_i$$

- μ = overall mean movie rating
- b_x = rating deviation of user x = (*avg. rating of user x*) – μ
- b_i = rating deviation of movie i

We get an RMSE of 1.155, which is a minor improvement compared to our previous model.

We must note that in this method it is possible for some of the ratings averaged together to compute the predictions to be negative after the weightings, as the similarity scores could be negative whilst the ratings are constrained to be non-negative.

Whole Set	Training Set %	Testing Set %	K-nearest Neighbor	RMSE (Item Based) $b_{u,i} = \mu + b_u + b_i$
City: Las Vegas Business: Restaurants	80	20	20	1.153878716
			30	1.155957094
			40	1.152898163
			50	1.164204698
			60	1.155261052
			70	1.155957094
			80	1.154272312
			90	1.155261052
			100	1.157154278



Above, we see a summary of the RMSE results for various k-neighborhood parameters when training and testing for the Item-Item collaborative filtering model just discussed. We observe that the optimal RMSE is achieved for a neighborhood of 60.

However, given the sparsity of our ratings matrix, it would be interesting to have a closer look at some matrix factorization techniques. Recommender systems rely on many different types of input data

and since our "explicit feedback" (ratings given by users for the various businesses) is sparse, it would be useful to implement matrix factorization algorithms, which would infer user preferences using "implicit feedback". The intuition behind using matrix factorization to solve this problem is that there should be some latent features that determine how a user rates an item. For example, two users would give high ratings to a certain restaurant if they both like that type of food, or if the restaurant has a nice atmosphere for example. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the restaurant. Basically, these models assume a similarity layer between users and restaurants is induced by a hidden lower-dimensional structure latently present in the data.

Method 4: Stochastic Gradient Descent (SVD)

At first we were looking to find the singular value decomposition in order to perform matrix factorization on our ratings matrix. However this wouldn't be a very good idea as we have a very high number of unknowns in our matrix and computing the SVD would require us to set these unknowns to 0. This would hamper our predictions, as a user not having a rating for a given restaurant would be equivalent to them giving the restaurant the lowest possible rating. Hence, instead of directly factorizing our ratings matrix, we can use stochastic gradient descent in order to derive a matrix P containing the latent factors of the user and a matrix Q containing those of the restaurants without having to set any of the unknown entries to 0. The algorithm for this method is given below:

To get the prediction of a rating of an item q_j by p_i, we can calculate the dot product of the two vectors corresponding to q_j and p_i :

$r_{ij} = p_i^T q_j$ (fitted rating)

$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2$ (squared-error)

We want to minimize the error and so using stochastic gradient descent, we get the following update rule:

$p_{ik} = p_{ik} + 2\alpha e_{ij} q_{kj}$

$q_{kj} = q_{kj} + 2\alpha e_{ij} p_{ik}$

This method greatly improves our predictions with our RMSE going down to 1.128.

However, one of the issues with this method is that it can lead to some serious over-fitting of our data and in order to solve this problem, we can try and regularize our SVD stochastic gradient descent method.

Method 5: Regularized Stochastic Gradient Descent (SVD)

In this method, we add a new regularization parameter to the model above in order to avoid over-fitting the data. In order to learn the factor vector p_i and q_j, the system minimizes the regularized squared error on the set of known errors. The system learns the model by fitting the previously observed ratings, so as to predict future unknown ratings. The constant β controls the extent or regularization as we try and regularize the learned parameters whose magnitudes are penalized.

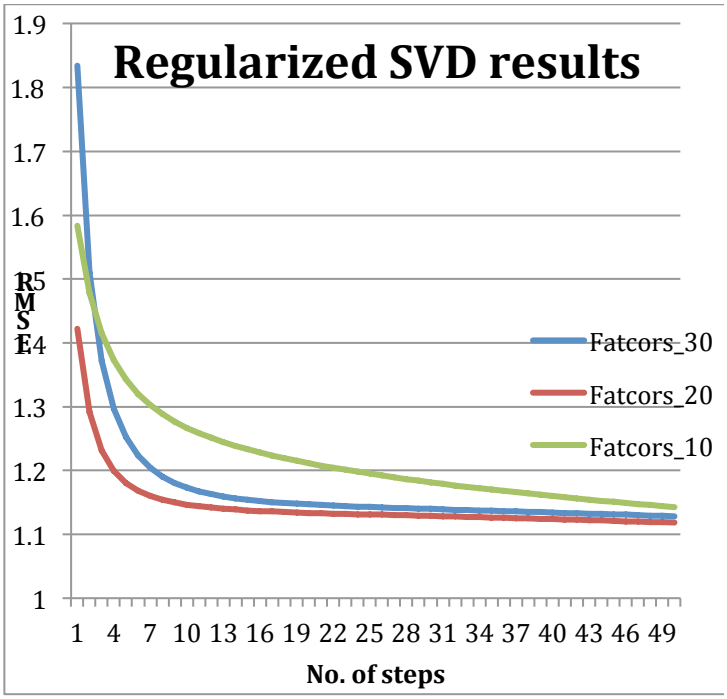
$e_{ij}^2 = (r_{ij} - \hat{r}_{ij})^2 + \frac{\beta}{2}\sum_{k=1}^{K}(\|P\|^2 + \|Q\|^2)$ where P and Q give a good approximation of R.

The new update rule is as follows:

$p_{ik} = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik})$

$q_{kj} = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj})$

Our new RMSE goes down to 1.119 as we get our best result till now. We see below that the number of iterations before which our regularized SVD model minimizes the RMSE is 50. Here the ideal number of latent factors would be 20.



In order to improve our predictions, we add biases to the regularized SVD model, one parameter c_i for each user and one d_j for each movie:

$\hat{r}_{ij} = c_i + d_j + u_i^T v_j$ (Improved RSVD model)

We train the weights c_i and d_j are trained simultaneously with u_{ik} and v_{jk} :

$c_i = c_i + \alpha * (r_{ij} - \lambda_1 (c_i + d_j - global\_mean))$

$d_j = d_j + \alpha * (r_{ij} - \lambda_2 (c_i + d_j - global\_mean))$

α is the learning rate

$\hat{r}_{ij}$ is the predicted ratings matrix

This method gives us a slightly improved RMSE of 1.117 on our predictions.

Method 6: Alternative Least Squares

The previous two methods lead us to the Alternative Least Squares optimization problem, which works in the same way as the SVD stochastic gradient descent algorithm excluding the fact that it keeps rotating between fixing the q_{kj} and the p_{ik} . Since the system computes each q_i independently of the other item (restaurant) factors and computes each p_j independently of the other users and other user factors, this gives rise to the possibility of massive parallelization of the algorithm. Also, since the training set is large and sparse, instead of looping over every single training case like gradient descent, ALS provides a more computationally cost efficient method. However, we see here the ALS gives us an RMSE of 1.131, which is poorer than our previous SVD models because at times stochastic gradient descent converges faster and is easier to compute.

5/ Summary of results

Model	RMSE
Baseline	1.204
Baseline + user -user CF + kNN	1.193
item – item + kNN	1.157
Baseline+ item - item CF + kNN	1.155
SVD with stochastic gradient descent	1.128
Regularized SVD (RSVD)	1.119
Improved RSVD	1.117
Alternating Least Squares	1.131

The results we got were in line with what we expected to get, as the different collaborative filtering techniques performed better than the baseline model. Also, given the sparseness of the data, it was no surprise that the matrix factorization techniques performed a lot better than the vanilla collaborative filtering ones. Our best result was achieved, in our improved RSVD model. Interestingly, the Alternating least squares method performed worse than the RSVD. However, we must note that the ALS is only more efficient in certain cases when parallelization is required or when stochastic gradient descent is very slow.

6/ Future

The two main issues faced by collaborative filters were the sparseness of data and the cold-start problem (item cannot be recommended unless it hasn't been rated before). Given the amount the data available to Yelp, a possible solution would be to combine collaborative recommendation systems with content-based filters to exploit the information of the items already rated. On its own, content-based filters wouldn't provide great insight as collaborative filter have certain advantages over them: (1) they have the ability to provide recommendations for items both relevant and non-relevant to the user's profile (2) they can also perform in domains where not much content is available about the users/items. A "content-boosted collaborative filtering model" would help us get the best of both methods. Also, implementing a combination of our models should help improve our predictions' RMSE.

7/ References:

- **Recommendation System Based on Collaborative Filtering** - Zheng Wen
- **Improving regularized singular value decomposition for collaborative filtering** - Arkadiusz Paterek
- **Data Mining Methods for Recommender Systems** - Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M. Pujol
- **Mining of Massive Datasets** - Stanford lectures
- **Matrix factorization techniques for Netflix** – Koren
- **Collaborative Filtering Recommender Systems**- Michael D. Ekstrand, John T. Riedl and Joseph A. Konstan
- **Content-Boosted Collaborative Filtering for Improved Recommendations** - Melville and Mooney and Nagarajan
- http://officialblog.yelp.com/2011/10/harvard-study-yelp-drives-demand-for-independent-restaurants.
- http://ucersti.ieis.tue.nl/files/papers/4.pdf
- **Collaborative filtering models for recommendations systems** -Nikhil Johri, Zahan Malkani, and Ying Wang