Recommendations

Final Report

Holubiec, Maciej
Romero, Maria Jimena
Von Chamier, Paul

1. Problem Statement and Motivation

The project's objective is to create a recommendation system for restaurants using regressions. We will use regressions to predict ratings (number of stars) per user and business. In specific, we will make two Baselines, one with sample averages and another with a Regularized Regression, which we will compare to our Matrix Factorization, according to the Root Mean Square Error. The data used comes from Yelp reviews, where we can identify users and business, among other characteristics like type of business, geographic region, among others.

2.Introduction and Description of Data

The project is relevant in the consumption industry, as currently modern consumers are inundated with choices, and matching consumers consumers with the most appropriate products is key to enhancing user satisfaction and loyalty[1]. However, there are more sectors that can benefited from this method. In public policy, for instance, there is wide potential to better match citizens preferences to services and policies. Citizens can benefit from better information on school, public health locations, doctors, among other. Also, the Matrix Factorization can be used not only when dealing with consumers, but also when trying to fill any missing values between two different parties (like even implied costs between companies).

The challenge comes from the fact that on average, users rate only a few places and places are rated by only a few users. According to the EDA, we found out that we see that the average rating issued per user 2.5 and that the average number of ratings received per business is 12.5. This large amount of missing values can create difficulties when doing regressions, and also create a matrix with more number of variables than observations. To fix this, we might only select the most frequented restaurants and users.

The Yelp data is distributed among 5 different json files, out of which 3 are being used: business.json, review.json and user.json. In total, we started with 4.7 MM reviews. To build our models, we will need to transform review database to create a user_id - business_id matrix (filled with ratings). With our baseline model, we will fill the missing data with averages or predictions from a multiple regression. With the latent matrix model, we will try to predict using matrix factorization backed up by alternating least squares regularization.

3. Literature Review/Related Work

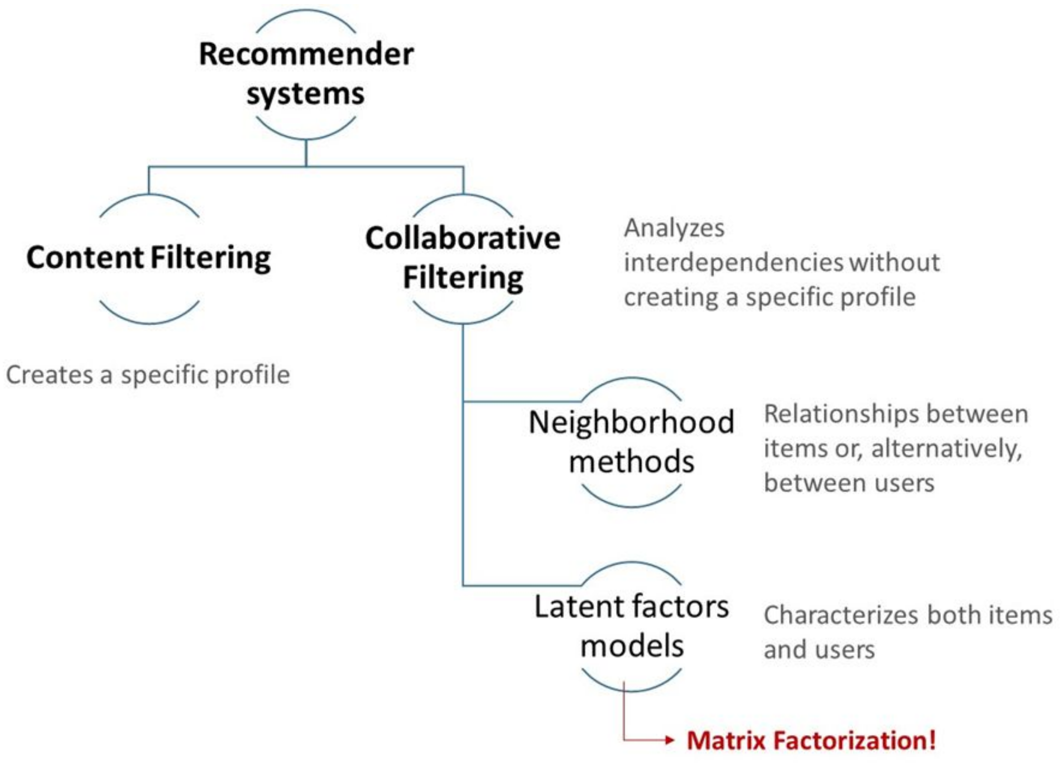
Matrix factorization for recommender system is a recent method that became popular in 2006 when Netflix launched a competition to improve its recommendations.[2] During the following years, Matrix Factorization methods were the most important class of techniques for winning the Netflix Prize[3].

[1] Matrix factorization for recommender systems https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf
[2] Ensembling for the Netflix Prize http://web.stanford.edu/~lmackey/papers/netflix_story-nas11-slides.pdf
[3] How the Netflix prize was won http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/

Matrix Factorization belongs to one of two main strategies in recommender systems: *collaborative filtering.* In general, recommender systems are based either on *content filtering* (creates a profile for each user or product to characterize its nature), or collaborative filtering (relies only on past user behavior without requiring the creation of explicit profiles and analyzes relationships between users and interdependencies among products to identify new user-item associations).[4] On its own, collaborative filtering has two main areas: *neighborhood methods* and *latent factor models*. Neighborhood methods focus on relationships between items or, alternatively, between users, while latent factor models focus on characterizing both items and users.

In other words, latent factor models transforms both items and users to the same latent factor space. This latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback.[5] In this area, Matrix Factorization is one of the most popular methods.

Recommender Systems Methods

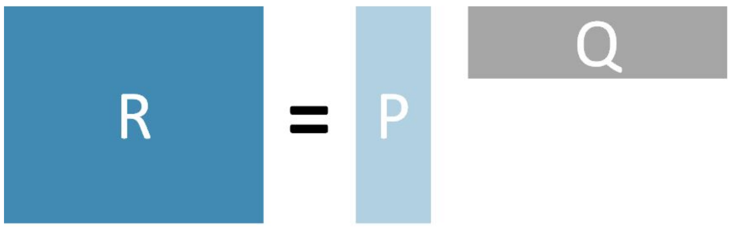

Recommender systems
Content Filtering
Creates a specific profile
Collaborative Filtering
Analyzes interdependencies without creating a specific profile
Neighborhood methods
Relationships between items or, alternatively, between users
Latent factors models
Characterizes both items and users
Matrix Factorization!

A matrix factorization can be performed with a singular value decomposition (SVD) on the ratings matrix (using SGD or ALS and regularizing the weights of the factors). Matrix factorization is particularly advantageous because it allows the incorporation of additional information. For instance, when explicit feedback is not available (ie. direct ratings on a business), one can infer user's

[4] Matrix factorization for recommender systems,
https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf
[5] Advances in Collaborative Filtering fromthe Netflix prize
https://datajobs.com/data-science-repo/Collaborative-Filtering-%5BKoren-and-Bell%5D.pdf

preferences using implicit feedback by observing behavior like purchase history, search patterns, etc. [6]

Matrix Factorization model



Some SVD-inspired methods for matrix factorization include: **i) Standard SVD** (you can dot product the user's vectors with the movies or business' vectors to predict each individual rating); **ii) Asymmetric SVD** (instead of users having their own notion of factor vectors, we can represent users as a bag of items they have rated); **iii) SVD++** (incorporates both the standard SVD and the asymmetric SVD model by representing users both by their own factor representation and as a bag of item vectors)[7]. For purposes of this project, we will focus on the Standard SVD.

4. Modeling Approach and Project Trajectory

4.0 Assumptions

We experienced significant difficulties while doing the modeling: dataset was too big and our laptops could not handle the operations (either kernel died or memory error showed up). Therefore, we decided to narrow down the observations to users with 150+ reviews given and restaurants with 350+ reviews received. Later, we sampled 15% of those observations and reached a review dataset with ~40,000 observations, which translated into ~(16,000x1500) latent factor matrix, which our laptops could handle. While doing the Regularized regression, we had to get dummy variables for all the users and restaurants, so our review dataset's dimension became ~(40,000x17,000), since we had ~1500 distinct users and ~15,500 distinct restaurants.

4.1. Baseline model

The baseline will be estimated with two models: Sample Averages and Regularized Regression. With the residuals of these models, we will estimate the matrix factorization that will allow to predict more accurately the recommendations.

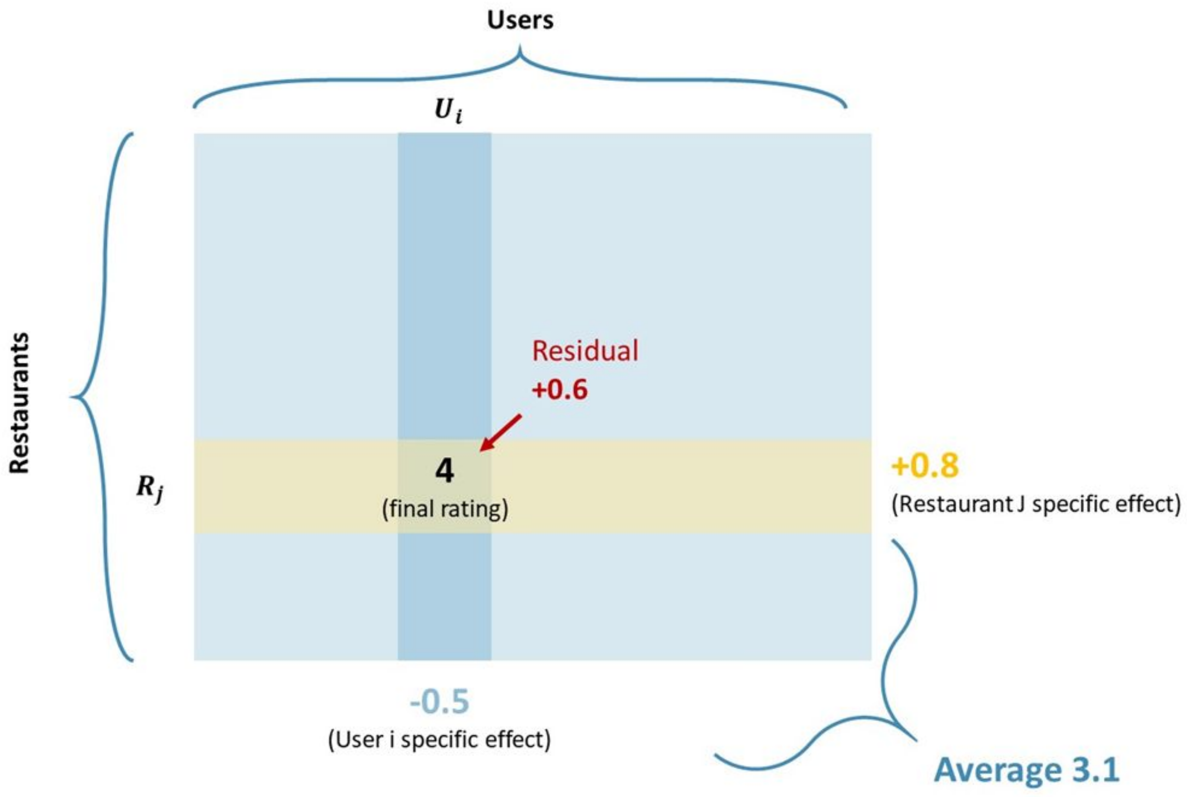
a. Sample Averages

Suppose an user rates a restaurant with 4 stars[8]. This can be decomposed in:

[6] Matrix factorization for recommender systems, https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf
[7] How the Netflix prize was won http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/
[8] Example inspired from How the Netflix prize was won http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/

- A baseline rating (average of all reviews) = 3.1.
- The effect of the user = - 0.5
- The effect of the restaurant = 0.8
- Specific interaction = 0.6

Sample Averages Baseline Example



The effect of the specific interaction between the restaurant and the users is the residual, which will be estimated with the matrix factorization.

From the Yelp data, we estimated an average of 3.7 stars. With this information, we calculated the averages per user and restaurant, to predict the missing values.

The Root Mean Square Error of the predicted stars with the sample average baseline is **0.89** (in other words, less than one star of error).

b. Regularized Regression

Another way to estimate the baseline is through a regularized regression. To achieve this, we chose Lasso with cross validation (5 fold) to find the alpha that minimizes the Root Mean Square Error. The alpha was 0.001. This alpha gave us a RMSE of **0.935**.

c. Matrix Factorization

The matrix factorization was done with the residuals from the regularization regression. To select the best number of iterations of ALS, the best learning rate and the best number of latent factors we performed a pseudo cross validation (we did not split data into train and validation hence "pseudo", but we iterated over all the parameters). We found that we should choose a lambda of 0.01 and a number of latent factors of 100. This gave us an RMSE of **0.87,** less than the RMSE from the baseline regression and from the sample average.

5. Results, Conclusions, and Future Work

At the very bottom of the notebook we included a get_recommendations function which gives recommendations for a random username (included in the sample chosen by us). We included results for the baseline average model, baseline regression model and latent factors model. As we see, all the models seem to narrow down the

As a result, the Factorization Matrix can be an effective way for Latent Factors Models, as it reduced significantly the RMSE from the baseline regressions.

Our results could further be improved by:

- Find a better way to split the data intro train, validation and train sets so we can choose parameters for both baseline models and matrix factorization model using the same data sample (each model requires a different approach: baseline average and matrix factorization operate on latent matrix, whereas baseline regression uses original reviews dataframe)
- Run the model on some virtual machine/on the cloud in order to account for the entire dataset, and not only users who gave 150+ reviews and restaurants with 350+ reviews.
- Do an ensemble model, taking into consideration the sample average baseline and the regression baseline. Also, adding biases and additional input sources can have a positive effect in the optimization.