Implementation of a UV decomposition algorithm for Yelp business recommendation system

Gustaw Andrzejewski

June 13, 2023

1 Introduction

Recommender systems have become a key part of our online experiences, guiding us through an ever-growing range of content and services. These systems help to personalize our digital world, tailoring it to our individual preferences. In this project, I've set out to build a recommender system from scratch, aiming to predict and recommend businesses to users based on the rich dataset provided by Yelp.

Yelp is a platform where users can share reviews and ratings of various businesses, making it a comprehensive and valuable resource for developing a recommender system. For this project, I focus specifically on the data contained in the "yelp academic dataset review.json" file, which is part of the broader Yelp dataset available on Kaggle and released under the CC-BY-SA 4.0 license. The file is formatted as a set of JSON-encoded documents, easily transformable into a pandas dataframe, perfect for manipulation and analysis. While the entire Yelp dataset is expansive, for practicality's sake, I've chosen to work with a suitable subset of the data.

To construct the recommender system, I've chosen to implement UV decomposition. This method of matrix factorization decomposes a utility matrix into two lower-rank matrices - a user matrix (U) and an item matrix (V). This technique enables effective representation of the interactions between users and businesses, allowing me to predict user preferences for unrated businesses, and thus, create personalized recommendations.

This report will delve into the theoretical underpinnings of UV decomposition, outline the preprocessing steps taken with the Yelp dataset, and detail the implementation of the UV decomposition algorithm. I will also present the results derived from the recommender system and draw conclusions from the findings. The goal of this project is to gain a deeper understanding of the UV decomposition technique and its effectiveness for business recommendations within the context of the Yelp dataset.

2 YELP Dataset

The cornerstone of this project is the comprehensive Yelp dataset, rich user-generated content spanning a multitude of businesses across the globe. The Yelp dataset's multi-faceted structure and vast scale provide an excellent basis for robust data analysis, particularly in the context of developing a recommender system. For this project, I primarily use three components of the Yelp dataset: business data, user data, and review data.

The business data encapsulates a plethora of business-related details, offering a perspective on various businesses listed on Yelp. It combines funda- mental identifiers such as the business ID and name with more elaborate data points such as the geographical location and detailed operational attributes. This dataset component serves as a pivotal source of context for understanding and interpreting user reviews and ratings.

The user data component provides insights about Yelp's user base. Beyond providing basic identifying information, this data set uncovers a breadth of user activity metrics on the Yelp platform, thereby shedding light on the user's engagement level. A distinguishing feature of the user data com- ponent lies in its inclusion of user voting patterns and received compliments, thereby reflecting the user's standing within the Yelp community. Correlating this information with users' reviews and ratings may yield intriguing insights into their behavior and preferences.

The review data, the key of this project, records the interaction between users and businesses. Each review, alongside its unique identifier, maintains a direct link to the respective user and business, thereby meticulously recording the user-business interaction landscape. The textual content of the review and the star rating stand as direct reflections of the user's sentiment towards a particular business. Moreover, the votes a review garners from the Yelp community offer an external perspective on the review's reception and impact.

To summarize, the Yelp dataset presents an intricate web of interrelated data points encapsulating user sentiment, business details, and community reception. Its vastness and complexity provide a fertile ground for meaningful data analysis and pattern discovery. For employing the UV decomposition technique, this rich array of user-business interactions can be harnessed by simple utilizing business and user interactions expressed by the ratings. It allows to create business recommendations, thereby elevating the user experience on Yelp and contributing significantly to the platform's utility.

3 UV Decomposition for populating the Utility Matrix

UV decomposition, also known as matrix factorization, is a method primarily used in the field of recommendation systems. The underlying principle of UV decomposition is the approximation of a given matrix by the product of two lower-rank matrices, referred to as the user matrix (U) and the item matrix

(V). Each row of the user matrix represents the latent features of a user, while each column of the item matrix represents the latent features of an item. The multiplication of these two matrices yields a utility matrix that approximates the original matrix, where each element represents a predicted rating.

There are several compelling reasons for choosing UV decomposition over other techniques for this project. First, the UV decomposition method has an inherent ability to deal with sparsity, a common issue in the context of recommendation systems where the majority of users have not rated most businesses. The ability to handle missing values and provide reliable estimates in such sparse scenarios is a distinct advantage of UV decomposition.

Moreover, UV decomposition brings forth the concept of latent features, which enable the model to capture the complex, often implicit factors influencing a user's rating of a business. By representing both users and items (in this case, businesses) in terms of these latent features, UV decomposition can predict a user's rating for a business they have not yet reviewed.

While collaborative filtering methods based on Jaccard Distance or Cosine Distance are popular, they are not the best fit for this project. These methods primarily measure similarity based on the intersection of rated items between users (Jaccard) or the cosine of the angle between user vectors (Cosine), which can lead to a number of limitations. For example, they often struggle with the problem of sparsity and do not deal well with new users or businesses with little or no rating history (cold-start problem).

Moreover, these methods are typically more suited to binary ratings, such as "like" or "dislike". However, the Yelp dataset involves a more complex rating system, ranging from 1 to 5 stars. Therefore, capturing the intricate nuances of these ratings calls for a more sophisticated technique like UV decomposition, which can model the latent factors underlying these ratings.

4 Data Preprocessing

The initial phase of the recommendation system development was focused on detailed data preprocessing and exploratory analysis. I have worked with three key DataFrames: Users, Businesses, and Reviews, with an ultimate goal to build a utility matrix.

4.1 Users DataFrame

The original Users DataFrame consisted of about 1,987,897 unique users. However, for the optimization of the recommendation system, it was crucial to focus on 'relevant' users, identified based on certain parameters such as unique businesses a user reviewed and the variance in their ratings. Using these metrics and setting appropriate thresholds, I successfully filtered out approximately 1,963,398 'less irrelevant' users, leaving us with 24,499 'relevant' users.

4.2 Businesses DataFrame

The Businesses DataFrame originally had around 150,346 unique businesses. To increase computational efficiency and improve user relevance, I decided to focus on businesses classified as 'Restaurants' in New Jersey and Pennsylvania. This decision was based on the high concentration of businesses in these areas. After applying further filtering criteria such as a minimum of 36 reviews and a minimum rating of 3 stars, I narrowed down the 'relevant' businesses to 4,874.

4.3 Reviews DataFrame

The preprocessing of the Reviews DataFrame involved retaining only the 'relevant' users and businesses from the prior steps. This resulted in a significant reduction in the number of reviews considered for the model, down to 18,465 reviews.

4.4 Construction of the Utility Matrix

Constructing the utility matrix was a crucial part of the data preprocessing phase. This matrix encapsulates user-business interactions, with each entry representing the average ratings given by a user to a business. This process involved:

1. Calculating the average rating for each user-business pair.
2. Indexing of User IDs and Business IDs into numeric indices.
3. Constructing the utility matrix in chunks to handle the large volume of data.

The completion of the utility matrix marked the end of the data preprocessing phase and set the stage for the next step of recommendation system development. The application of rigorous filtering and selection criteria ensured I worked with the most relevant data, optimizing the quality of the recommendations.

5 Implementation

The UV-Decomposition method implemented here is based on the principles elucidated in the book "Mining of Massive Datasets" by Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. The algorithm is central to the development of recommendation systems and finds global optimum decomposition of a utility matrix M.

The implementation comprises four key phases:

1. Preprocessing
2. Initialization

3. Gradient Descent Optimization
4. Convergence Determination

5.1 Preprocessing

The method implements a data preprocessing step underscored in the book to handle the inherent biases present in raw user ratings. This is carried out in the _normalize method, which adjusts each user's ratings by subtracting the mean of their non-zero ratings. These means are stored in the self.mean dictionary for each user, serving as a key component in the prediction stage. The _denormalize method reverses this process, adding back the mean to the product of U and V matrices for a user-item pair to produce a final rating.

5.2 Initialization

To align with the randomness aspect proposed in the book, the U and V matrices are initialized with randomly distributed random values in the _initialize_U_and_V method. The randomness plays a crucial role in avoiding local minima during the optimization process, leading to a more satisfactory global optimum solution.

5.3 Gradient Descent Optimization

To optimize U and V, the implementation features three types of gradient descent: Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD), and Mini-batch Gradient Descent (MBGD) within the _train_step method. This is a significant interpretation of the optimization strategies discussed in the book. For BGD, the gradients for all non-zero elements in M are computed. SGD selects a single random non-zero element, while MBGD chooses a random batch of non-zero elements. The choice of gradient descent type dictates the pace and stability of convergence.

5.4 Convergence Determination

Consistent with the principles outlined in the book, the algorithm employs the Root Mean Square Error (RMSE) metric to measure the error between the product of the U and V matrices and the original utility matrix M. The _calculate_RMSE method encapsulates this computation. The train method supervises the overall training process. Here, the algorithm cycles through the maximum specified number of iterations, refining U and V with each iteration using gradient descent. However, if the RMSE ceases to improve (i.e., increases) for a predetermined number of iterations (dictated by the 'patience' parameter), the training halts prematurely. This approach allows the algorithm to prevent unnecessary computations when the solution is no longer improving significantly.

The UVDecomposition class also encompasses the _RMSE method, which provides a general-purpose function for calculating the root mean square error between two arrays. Additionally, the predict method estimates the rating for a user-item pair by multiplying the corresponding vectors from the U and V matrices, followed by denormalization of the result.

Certainly! Here's the updated LaTeX paragraph with placeholders for the training history charts:

6 Results and Discussion

In my project, I conducted three distinct experiments employing different types of gradient descent: Batch Gradient Descent (BGD), Mini Batch Gradient Descent (MBGD), and Stochastic Gradient Descent (SGD). I applied these methods to matrix factorization in the context of recommendation systems, a common approach in collaborative filtering. For each experiment, the model's training was tracked using the Root Mean Square Error (RMSE) measure.

In the first experiment, I utilized Batch Gradient Descent, training the model for a maximum of 200 iterations at a learning rate of 0.005. The RMSE steadily decreased over the iterations, eventually converging to a low value of 0.0167. While this suggests a good fit to the training data, when making random predictions on unseen data from the test set, it was clear that the model was overfitting. Despite the low RMSE, it failed to generalize well to unseen data, a crucial aspect of effective machine learning models.

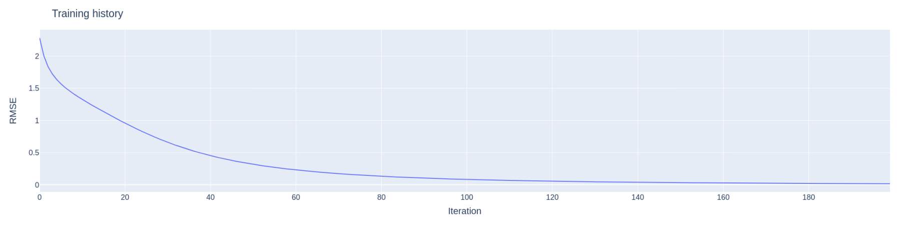

Figure 1: Batch Gradient Descent Training History

In the second experiment, I applied Mini Batch Gradient Descent with a batch size of 512 and a higher learning rate of 0.05. The model trained for a maximum of 500 iterations, reaching an RMSE of 0.1332. Although higher than the BGD approach, this method was quicker due to the utilization of mini batches. Similar to the first experiment, the model showed reasonable accuracy on the training set but struggled with predictions on the unseen test set, indicating an overfitting issue.

Finally, in the third experiment, I utilized Stochastic Gradient Descent. This model was trained at the same learning rate as the MBGD but had a smaller dimensionality ($d = 5$). After 500 iterations, the RMSE was significantly higher at 1.6548. This high error rate suggests a poor fit to the data, and similar to the previous methods, the model had difficulty generalizing to unseen data.

In conclusion, although the Batch Gradient Descent model demonstrated the

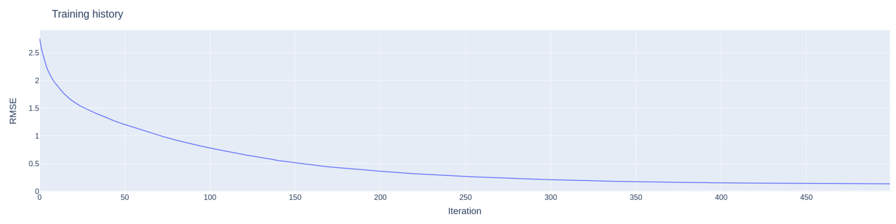

Figure 2: Mini Batch Gradient Descent Training History

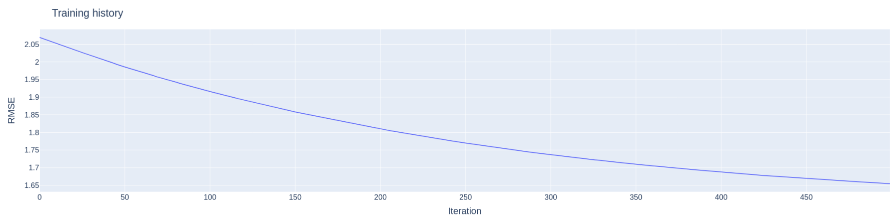

Figure 3: Stochastic Gradient Descent Training History

lowest RMSE value, none of the tested models were able to effectively generalize to unseen data. This indicates a substantial overfitting issue across all experiments. Despite the faster performance of the Mini Batch Gradient Descent and the variation introduced by the Stochastic Gradient Descent, all models showed a similar inability to make accurate predictions on new data. These findings highlight the importance of addressing overfitting when designing and evaluating machine learning models.