PDS Challenge 3:
Matrix Completion for Recommendation Systems

SR+ITL

November 1, 2023

Abstract

In this challenge, we delve into the critical role of Singular Value Decomposition (SVD) in matrix completion and imputation. SVD plays a pivotal role in predicting user preferences and filling in missing data in platforms like YouTube and Netflix, where recommendation systems heavily rely on its intricate workings. By applying SVD, we enable the accurate reconstruction of incomplete matrices, thereby enhancing user experiences and personalizing content recommendations.

1 Singular Value Decomposition (SVD) in Recommendation Systems

Singular Value Decomposition (SVD) serves as a fundamental technique in recommendation systems, facilitating personalized content suggestions and enhancing user experiences. In the context of recommendation systems, SVD operates through the following methodologies:

- **SVD in Collaborative Recommendation Systems:** By decomposing the user-item interaction matrix R, represented as $R = U\Sigma V^T$, SVD facilitates the identification of latent features that drive user preferences. Here, U represents the user feature matrix, Σ is a diagonal matrix containing singular values, and V^T denotes the item feature matrix.
- **SVD Integration in Hybrid Recommendation Systems:** SVD plays a crucial role in hybrid recommendation systems, combining collaborative and content-based filtering approaches. It captures both historical user behavior and item features, enabling more personalized and relevant suggestions to users.

1.1 Addressing Challenges with SVD in Recommendation Systems

Despite its effectiveness, collaborative filtering encounters challenges, notably the cold start and sparsity problems. Singular Value Decomposition (SVD) critically mitigates these challenges by utilizing latent features to make informed predictions for new users and addressing the scarcity of recorded user-item interactions.

2 Matrix Completion Using SVD for Recommendation Systems

In the context of collaborative filtering, matrix completion techniques are pivotal in handling the sparsity problem. Singular Value Decomposition (SVD) facilitates matrix completion in recommendation systems through the following steps:

1. **Low-rank Matrix Approximation:** Utilizing the assumption of a low-rank underlying user-item rating matrix, SVD efficiently represents data in a reduced-dimensional space, identifying latent factors that explain user-item interactions effectively.

2. **SVD Decomposition:** SVD decomposes the user-item rating matrix R into three constituent matrices, denoted as U, Σ, and V^T. The approximation of the original matrix R is achieved by retaining the top k singular values and corresponding columns of U and V, as given by:

$$R \approx \hat{R} = U_k \Sigma_k V_k^T$$

where U_k and V_k are the truncated matrices with the top k columns of U and V respectively.

Additionally, techniques like Probabilistic Matrix Factorization (PMF) can be integrated for further robustness and accuracy in predicting missing values in the user-item rating matrix.

3 Model Evaluation Metrics for SVD-based Recommendation Systems

Evaluation of model performance in SVD-based recommendation systems is crucial for assessing accuracy. Key evaluation metrics include:

- **Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE):** These metrics assess the accuracy of SVD-based models by measuring the differences between predicted and actual ratings.
- **Precision and Recall:** Crucial in assessing the precision and comprehensiveness of recommendations, especially for large datasets and diverse user preferences.
- **F1 Score:** Providing a comprehensive measure of the balance between precision and recall, offering holistic insight into recommendation quality.
- **Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC):** Vital tools for evaluating the performance of binary classifiers integrated into SVD-based recommendation systems, enabling nuanced evaluation of the model's discriminatory power.

4 Project Overview

For this challenge, your tasks will involve the following key objectives:

1. Delve into the intricate workings of recommender systems, particularly focusing on the role of Singular Value Decomposition (SVD) in matrix completion and imputation.
2. Design and develop a robust recommender system, ensuring to establish a baseline for measuring its effectiveness. Leverage the principles of SVD to enhance the accuracy and efficiency of the system.
3. Participate in the Kaggle competition, where you will have the opportunity to apply your insights and techniques developed during this challenge to solve real-world recommendation problems. Link: https://www.kaggle.com/t/ab55df79245b412abe77f7b059395ade

4.1 Deliverables

As with the past challenges, you are to deliver on E3 platform:

1. Code: as either notebook or python file. Make sure to add meaningful comments.
2. Report: a PDF file detailing what you did, and why. Delve into the mathematics and the reasoning behind your model.
3. Video: I suggest adding a link to your PDF report to your video. You can upload your video to the platform of your preference.

Please name youor files as follows: *yourgroupid_report-or-code.file-extension*

For example, if your group ID is "group10," and you wrote your code on a notebook, then your submitted code should be named *group10_code.ipynb*.

5 Spice

Let us re-interpret PCA as an iterative procedure as follows:

Algorithm 1 PCA as an iterative procedure

```
1: Input: Data matrix $X$ (centered), number of components $k$
2: Output: Principal components $U = [u_1, u_2, \dots, u_k]$
3: $X_{\text{temp}} = X$
4: for $i = 1$ to $k$ do
5:     Find $u_i$ such that:
           $u_i = \arg\max_u u^T X_{\text{temp}}^T X_{\text{temp}} u$    (1a)
           subject to $u^T u = 1$    (1b)
6:     Compute scores $t_i = X_{\text{temp}} u_i$
7:     Subtract contribution: $X_{\text{temp}} = X_{\text{temp}} - t_i u_i^T$
8: end for
9: return $U$
```

We can now reason on the various steps of Algorithm 1 and modify them accordingly.

Other projection measures: In step 5 of Algorithm 1, we have an inner product/projection operation induced by the L_2 norm. In order to define a norm from an inner product, we set

$$\|\mathbf{x}\| =< \mathbf{x}, \mathbf{x} > \tag{2}$$

how do we obtain an inner product from a norm? rather simply, we set

$$< \mathbf{x}, \mathbf{y} >= \|\mathbf{x} - \mathbf{y}\| \tag{3}$$

and a projection as

$$\mathbf{p} = \underset{\mathbf{v} \in \mathcal{S}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{v}\| \tag{4}$$

for some set $\mathcal{S}$. What would be the equivalent of the PCA if we use the L_1 norm, the Frobenius norm, or the infinity norm?

Non-negative PCA components: Very often, one is interested in a version of PCA components in which the components are positive defined. This is the case when the components have a physical meaning which is naturally "additive". Think at the example of the PCA for the face dataset: what is the meaning of a PCA component to be negative? this would mean that features are simultaneously added and subtracted to a face. Intuitively, this is counter intuitive, as we think about these portraits as being obtained by adding features in succession.

This can be obtained by imposing u_i in step 5 of Algorith 1 to have positive entries. If we impose this constraint, then we cannot enforce for components across iterations to be orthogonal. This is because the inner product between vector that have non negative entries is also non negative.

How can the algorithm be modified to obtain a result as close as possible to PCA under this additional constraint?

Regularized components: Another desirable property of the PCA components is that the components do not spread their "energy" across too many dimensions. In other words, we would want to minimize the number of small entries in u_i in step 5 of Algorithm 1. This is because these components are more affected by noise in the data matrix than the larger components. This property of the components can enforced by modifying (1) as

$$u_i = \arg\max_u u^T X_{\text{temp}}^T X_{\text{temp}} u + \lambda \max\{\tau, u\} \tag{5a}$$

$$\text{subject to } u^T u = 1 \tag{5b}$$

for some hyperparameter λ and λ and where $\max\{\tau, u\}$ is the operator setting $y_i = \max\{u_i, \tau$ for each coordinate. What is the algorithm resulting from this regularization?

NOTE these variation of the PCA algorithm are not going to be numerically efficient. Please feel free to reduce the size of the data matrix used for testing these variations.