Video Game Recommendation System using Matrix Factorization

Problem Description:

The objective of this project is to build a video game recommendation system using matrix factorization techniques. The system will take in user preferences and provide recommendations for video games based on those preferences.

The dataset used for this project is sourced from the Steam platform, which is a digital distribution platform for video games. The dataset contains information about video games including their titles, genres, release dates, and user ratings. The dataset is available on Kaggle and can be accessed through the following link: https://www.kaggle.com/tamber/steam-video-games.

The project will involve the following deliverables:

1. Data preprocessing: cleaning and transforming the dataset to a suitable format for matrix factorization.
2. Matrix factorization: applying matrix factorization techniques such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS) to decompose the user-item rating matrix and obtain user and item factors.
3. Model evaluation: evaluating the performance of the matrix factorization models using appropriate metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
4. Recommendation generation: generating recommendations for video games based on user preferences and the learned user and item factors.

The goal of this project is to build a recommendation system that can accurately predict user preferences and provide relevant recommendations for video games. This will improve the overall user experience on the Steam platform and increase user engagement with the platform.

Possible Framework:

1. **Data preprocessing:**

- Load and clean the dataset.
- Transform the dataset to a suitable format for matrix factorization.
- Split the dataset into training and testing sets.

2. **Matrix factorization:**

- Apply Singular Value Decomposition (SVD) or Alternating Least Squares (ALS) to decompose the user-item rating matrix.
- Tune hyperparameters using cross-validation on the training set.
- Obtain user and item factors.

3. **Model evaluation:**

- Evaluate the performance of the matrix factorization models using appropriate metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
- Compare the performance of different matrix factorization models.

4. **Recommendation generation:**

- Generate recommendations for video games based on user preferences and the learned user and item factors.
- Evaluate the quality of the recommendations using appropriate metrics such as Precision and Recall.
- Implement the recommendation system in a web application.

5. **Deployment:**

- Deploy the recommendation system on a cloud platform such as AWS or GCP.
- Ensure scalability and reliability of the system.
- Monitor and optimize the system's performance.

6. **Future work:**

- Explore advanced matrix factorization techniques such as Non-negative Matrix Factorization (NMF) and Probabilistic Matrix Factorization (PMF).
- Incorporate contextual information such as time of day and user demographics into the recommendation system.
- Implement a feedback loop to continuously improve the quality of recommendations.

The framework outlined above provides a roadmap for building a video game recommendation system using matrix factorization techniques. By following this framework, we can ensure that the project is organized and that each step is executed effectively.

Code Explanation :

Here is the simple explanation for the code you can find at code.py file.

Section 1: Importing Libraries

In this section, we import the required libraries such as pandas, numpy, and surprise. Pandas and numpy are used for data manipulation, and surprise is used for building the recommendation system model.

Section 2: Loading Data

In this section, we load the data from the given dataset. We use the pandas library to read the CSV file and create a dataframe.

Section 3: Exploring Data

In this section, we explore the loaded data. We use the pandas library to view the first few rows of the dataframe and check if there are any missing values. We also check the distribution of ratings given by the users.

Section 4: Building the Model

In this section, we build the recommendation system model. We use the Surprise library which provides an easy-to-use interface to implement popular recommendation algorithms. We use the SVD (Singular Value Decomposition) algorithm to build the model as it has proven to be effective in predicting user ratings.

Section 5: Training the Model

In this section, we train the model on the loaded data. We split the data into training and testing sets using the Surprise library. We then train the model on the training set and test it on the testing set to evaluate its performance.

Section 6: Making Recommendations

In this section, we use the trained model to make recommendations for the given user. We first create a list of all games in the dataset and then remove the ones that the user has already played. We then use the trained model to predict ratings for the remaining games and recommend the top games with the highest predicted ratings.

We have used the SVD algorithm from the Surprise library to build our recommendation system. SVD is a matrix factorization technique that factorizes the user-item interaction matrix into two lower-dimensional matrices, one for users and another for items. We have used SVD because it has shown good performance in predicting user ratings.

To run the code, the following libraries need to be installed:

- pandas
- numpy
- surprise

The code can be executed in any Python environment such as Jupyter Notebook or Python IDE.

Future Work :

Future Work for Video Game Recommendation System using Matrix Factorization

The current implementation of the video game recommendation system using matrix factorization has shown promising results. However, there are a few areas where the system can be improved. In this section, we outline some potential future work for the project.

Section 1: Data Preprocessing

In this section, we can explore different techniques for data preprocessing such as handling missing values, removing outliers, and scaling the data. These techniques can improve the performance of the recommendation system by making the input data more reliable and consistent.

Section 2: Algorithm Selection

In this section, we can explore different algorithms for building the recommendation system such as content-based filtering, collaborative filtering, and hybrid approaches. These algorithms can provide different perspectives on the data and can lead to better recommendations.

Section 3: Hyperparameter Tuning

In this section, we can explore different hyperparameters for the selected algorithm such as the number of latent factors, learning rate, and regularization strength. Tuning these hyperparameters can improve the performance of the recommendation system by optimizing the algorithm to the specific dataset.

Section 4: Integration with Gaming Platforms

In this section, we can explore ways to integrate the recommendation system with gaming platforms such as Steam and GOG. This can provide a more seamless user experience by allowing users to receive recommendations directly on the gaming platform.

Section 5: User Interface

In this section, we can explore ways to improve the user interface of the recommendation system. This can include adding user profiles, search functionality, and personalized recommendations. These features can make the recommendation system more user-friendly and increase user engagement.

Implementation Guide

To implement the future work for the project, the following steps can be followed:

1. Preprocess the data using techniques such as handling missing values, removing outliers, and scaling the data.
2. Explore different algorithms for building the recommendation system such as content-based filtering, collaborative filtering, and hybrid approaches.
3. Tune the hyperparameters of the selected algorithm to optimize its performance on the dataset.
4. Integrate the recommendation system with gaming platforms such as Steam and GOG.
5. Improve the user interface of the recommendation system by adding user profiles, search functionality, and personalized recommendations.

These steps can be implemented using the same libraries and tools as the current implementation. The code can be modified to incorporate the new features and techniques as required. It is important to evaluate the performance of the system after each modification to ensure that it is improving the recommendations.

Exercise Questions :

1. **How can you improve the accuracy of the recommendation system?**
   Answer: We can improve the accuracy of the recommendation system by using other advanced techniques such as content-based filtering or hybrid filtering. Additionally, we can use more advanced matrix factorization algorithms, such as SVD++ or NMF, to further improve the accuracy.

2. **How can you handle cold-start problem in this recommendation system?**
   Answer: Cold-start problem can be handled by using content-based filtering for new users or items. This means that we can recommend items to new users based on the attributes of the item or profile information of the user. We can also use popularity-based recommendation until we have enough data to use collaborative filtering.

3. **Can you recommend more than one item to a user at a time?**
   Answer: Yes, we can recommend more than one item to a user at a time. In fact, we can recommend any number of items to a user based on the algorithm's output. We can also set a threshold on the predicted rating or similarity score and recommend only items that pass the threshold.

4. **How can you evaluate the performance of the recommendation system?**
   Answer: We can evaluate the performance of the recommendation system by using different evaluation metrics such as RMSE, MAE, precision, recall, F1 score, and AUC. We can also use cross-validation techniques such as k-fold cross-validation or leave-one-out cross-validation to measure the accuracy of the model.

5. **Can you use this recommendation system for other types of products, such as books or music?**
   Answer: Yes, this recommendation system can be used for other types of products, such as books or music, by modifying the dataset and retraining the model. We can also use the same model architecture and algorithm for other types of recommendation systems such as content-based filtering, collaborative filtering, or hybrid filtering.

Concept Explanation :

Imagine you're a boss and you have a big team of employees working on different projects. You want to know which employees are good at which projects so you can assign them accordingly. One way to do this is to ask all of your employees to rate each project on a scale of 1 to 10. But this could be a lot of work for your employees and there's a chance they might not be honest with their ratings.

Instead, you can use matrix factorization to figure out which employees are good at which projects based on their past performance. You can create a matrix where the rows represent the employees and the columns represent the projects. Each cell in the matrix contains the rating that the employee gave to that project. But there might be missing values in this matrix if an employee hasn't worked on a certain project yet.

Matrix factorization allows us to break down this matrix into two smaller matrices - one for the employees and one for the projects. Each row in the employee matrix represents the "skill" of that employee, and each column in the project matrix represents the "difficulty" of that project. The product of a row in the employee matrix and a column in the project matrix gives us the predicted rating that the employee would give to that project.

But how do we actually find these smaller matrices? We use a process called gradient descent to iteratively update the matrices until we reach a minimum error. Essentially, we start with random values in the employee and project matrices, calculate the error between the predicted ratings and the actual ratings, and adjust the matrices to decrease this error. We keep doing this until the error is minimized.

Once we have these smaller matrices, we can use them to make predictions for new employees or projects. For example, if a new employee joins your team, you can use their "skill" vector and the project matrix to predict which projects they would excel at.

In summary, matrix factorization is a powerful technique for finding latent factors in a large dataset, such as the skills of employees and the difficulty of projects. It allows us to make predictions for missing values and can be used for recommendation systems, like in this video game recommendation project.