Movie Recommendation System using Collaborative Filtering and Generative Adversarial Networks

Renish R (202252333), Prashant Bharti (202251102),
Amit Patel (20225014), Anubhav Singh (202251018)
CS/IT Department
Indian Institute of Information Technology, Vadodara
Gandhinagar, Gujarat, India

***Abstract*—This paper presents a novel approach to movie recommendation systems by combining collaborative filtering with generative adversarial networks (GANs). Collaborative filtering is widely used to predict user preferences, but it often suffers from sparsity and cold-start problems. To address these issues, we leverage GANs to generate synthetic user-item interaction data, thereby improving recommendation accuracy and diversity. Experimental results demonstrate that our model outperforms traditional collaborative filtering methods on benchmark datasets.**

***Index Terms*—Collaborative Filtering, Generative Adversarial Networks, Movie Recommendation, Data Sparsity, Personalization.**

I. Introduction

Recommendation systems are essential in modern digital platforms, enhancing user experience and business revenue by providing personalized content. Among the various methods, collaborative filtering (CF) is widely used for leveraging user-item interaction data to generate recommendations. However, CF often struggles with challenges like data sparsity, where limited user interactions reduce its predictive effectiveness.

Generative adversarial networks (GANs), originally designed for generating high-quality synthetic data, have shown potential in mitigating such issues. This work integrates GANs with CF to address sparsity, proposing a robust GAN-based framework for movie recommendations. Key contributions include developing the framework, utilizing GANs to generate synthetic interactions to combat sparsity, and conducting a comprehensive evaluation on standard datasets.

II. Collaborative Filtering Model

Collaborative filtering (CF) is categorized as follows:

- **User-Based CF:** Identifies user similarities from historical interactions. *Ex: Similar preferences for movies lead to shared recommendations from similar users.*
- **Item-Based CF:** Focuses on item relationships. *Ex: Movies frequently rated together are considered similar.*
- **Matrix Factorization-Based CF:** Techniques like SVD and ALS decompose the user-item matrix to uncover latent factors in a lower-dimensional space.

Advantages of CF

- No domain knowledge required.
- Accurate with sufficient data.
- Scalable with matrix factorization.

Limitations of CF

- Cold start for new users/items.
- Sparse matrices limit effectiveness.
- Scalability challenges in large datasets.

Recent GAN-based approaches enhance CF by generating synthetic data, addressing sparsity and cold-start issues, which this study leverages for robust movie recommendations.

Collaborative filtering is implemented using **Matrix Factorization**. The user-movie interaction matrix R is decomposed into two low-dimensional matrices:

$$R_{\text{pred}} = U \cdot V^T$$

where:

- $U \in \mathbb{R}^{m \times k}$: user feature matrix, where each row represents a user in k-dimensional latent space.
- $V \in \mathbb{R}^{n \times k}$: movie feature matrix, where each row represents a movie in the same latent space.
- k: number of latent factors.

A. Similarity Measures

1) Cosine Similarity (Scalar Form): Cosine similarity measures the cosine of the angle between two vectors A and B. It is computed as:

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

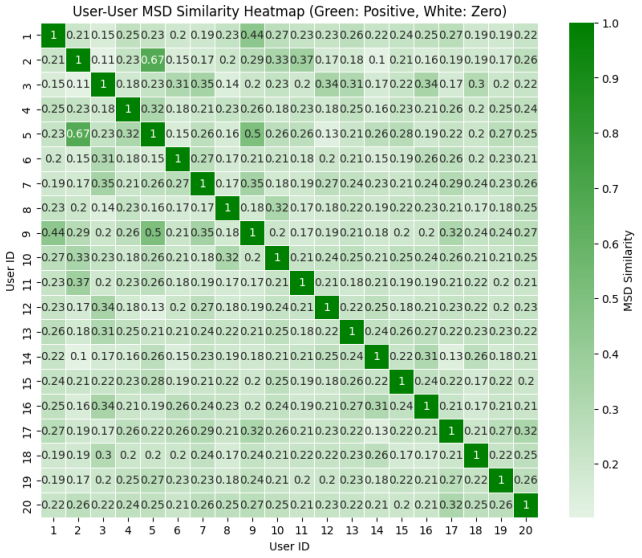
2) Pearson Correlation: Pearson correlation measures the linear relationship between two vectors A and B. The formula is:

$$\text{Pearson Correlation} = \frac{\sum_{i=1}^{n} (A_i - \overline{A})(B_i - \overline{B})}{\sqrt{\sum_{i=1}^{n} (A_i - \overline{A})^2} \sqrt{\sum_{i=1}^{n} (B_i - \overline{B})^2}} \tag{2}$$

where $\overline{A}$ and $\overline{B}$ are the means of A and B, respectively.

3) Mean Squared Difference (MSD): MSD measures the average squared difference between two vectors A and B. The similarity is given by:

$$\text{MSD Similarity} = \frac{1}{1 + \frac{\sum_{i=1}^{n} (A_i - B_i)^2}{n}} \tag{3}$$



B. Vectorization

1) Bag of Words (BoW) Representation: The *tag* field of each movie is treated as a document d, and the entire dataset forms the corpus $\mathcal{D} = \{d_1, d_2, ..., d_N\}$, where N is the total number of movies. The BoW model represents each document d_i as a vector in a high-dimensional space based on the vocabulary of the corpus.

Let $\mathcal{V}$ denote the vocabulary of all unique words in $\mathcal{D}$, where $|\mathcal{V}| = V$. Each document d_i is represented as:

$$\mathbf{v}_i = [t_1^{(i)}, t_2^{(i)}, ..., t_V^{(i)}] \in \mathbb{R}^V, \tag{4}$$

where $t_k^{(i)}$ is the term frequency of word w_k in document d_i.

2) Count Vectorizer: The *Count Vectorizer* implementation in *sklearn* automates the BoW process by:

- Tokenizing text into words.
- Mapping each unique word w_k in the vocabulary $\mathcal{V}$ to a unique index.
- Creating a sparse matrix $\mathbf{M} \in \mathbb{R}^{N \times V}$, where each entry M_{ij} represents the frequency of the j-th word in the i-th document.

3) Cosine Similarity (Vector form): To measure the similarity between two movies i and j, the cosine similarity of their respective vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ is computed as:

$$\text{Cosine Similarity}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}, \tag{5}$$

where:

- $\mathbf{v}_i \cdot \mathbf{v}_j$ is the dot product of the two vectors.
- $\|\mathbf{v}i\| = \sqrt{\sum k = 1^V (t_k^{(i)})^2}$ is the Euclidean norm of vector $\mathbf{v}_i$.

Cosine similarity ranges from -1 to 1, where 1 indicates identical vectors and 0 indicates orthogonal (dissimilar) vectors.

C. Aggregate Functions

1) Average: The average of a set of ratings R is computed as:

$$\text{Average} = \frac{\sum_{i=1}^{n} R_i}{n} \tag{6}$$

2) Weighted Average: Weighted average considers the similarity scores Sim_i for each rating R_i. It is given by:

$$\text{Weighted Average} = \frac{\sum_{i=1}^{n} \text{Sim}_i R_i}{\sum_{i=1}^{n} \text{Sim}_i} \tag{7}$$

3) Difference From Mean: To adjust for mean rating differences between two users A and B, the predicted rating is:

$$\text{Predicted Rating} = \overline{A} + \frac{\sum_{i=1}^{n} \text{Sim}_i (R_i - \overline{B})}{\sum_{i=1}^{n} \text{Sim}_i} \tag{8}$$

where $\overline{A}$ and $\overline{B}$ are the mean ratings of users A and B, respectively.

III. Generative Adversarial Networks

GANs use two neural networks, a generator and a discriminator, competing in a zero-sum game. The generator creates synthetic data, while the discriminator distinguishes real from generated data, making GANs effective for augmenting sparse recommendation datasets.

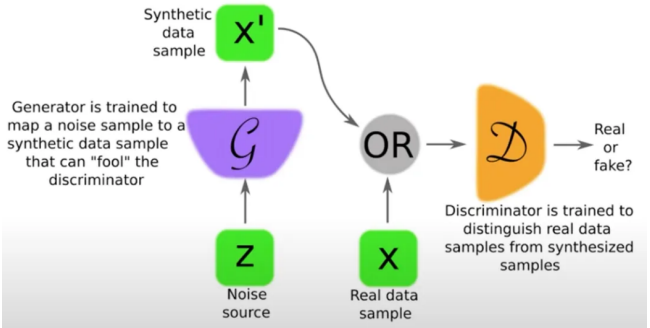


Fig. 1. GAN Structure

Key Components

- **Generator:** Converts random noise into synthetic data.
- **Discriminator:** Classifies data as real or generated.
- **Adversarial Training:** Generator aims to fool the discriminator, while the discriminator minimizes classification errors.

Applications

- Image generation, super-resolution, and style transfer.
- Data augmentation and domain adaptation.
- Enhancing collaborative filtering by generating sparse user-item interactions.

Challenges

- **Mode Collapse:** Limited output diversity.
- **Vanishing Gradients:** Weak gradients slow generator learning.
- **Convergence Issues:** Stability in adversarial training is hard to achieve.

The proposed approach integrates a **Generative Adversarial Network (GAN)** to generate synthetic user-movie interactions. The GAN consists of:

A. The Generator

The generator's objective is to confuse the discriminator, making it label generated data as true. The generator's loss function is:

$$L_G = \text{Error}(D(G(z)), 1) \tag{9}$$

This ensures that the generator minimizes the difference between the discriminator's output and the label for true data.

B. The Discriminator

The goal of the discriminator is to correctly label real data as true and generated (fake) data as false. The loss function for the discriminator can be expressed as:

$$L_D = \text{Error}(D(x), 1) + \text{Error}(D(G(z)), 0) \tag{10}$$

Here, Error is a generic function representing the difference between the predicted and true labels. It can be implemented using various metrics, such as cross-entropy or Kullback-Leibler divergence.

C. Noise Function

A common choice for the Error function in binary classification tasks is binary cross entropy. The general formula for cross entropy is:

$$H(p, q) = \mathbb{E}_{x \sim p(x)}[-\log q(x)] \tag{11}$$

In discrete cases, this can be expressed as a summation:

$$H(p, q) = -\sum_{x \in \chi} p(x) \log q(x) \tag{12}$$

For binary classification, where there are only two labels (0 and 1), the binary cross entropy becomes:

$$H(y, \hat{y}) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right] \tag{13}$$

This binary cross entropy is used as the Error function in the GAN loss functions.

D. Applying Binary Cross Entropy to GAN Loss Functions

Using binary cross entropy, the discriminator's loss function can be rewritten as:

$$L_D = -\sum_{x \in \chi} \log(D(x)) - \sum_{z \in \zeta} \log(1 - D(G(z))) \tag{14}$$

Similarly, the generator's loss function becomes:

$$L_G = -\sum_{z \in \zeta} \log(D(G(z))) \tag{15}$$

E. Min-Max Game Formulation

In the original GAN formulation by Goodfellow et al., the discriminator seeks to maximize its ability to distinguish real data from generated data, while the generator seeks to minimize the same objective. This is framed as a min-max game:

$$\min_G \max_D \left\{\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]\right\} \tag{16}$$

This compact representation highlights the adversarial nature of GANs. In practice, separate loss functions for the generator and discriminator are defined, as described above, to improve training stability and convergence.

F. Key Observations

The gradient of the function $y = \log x$ is steeper near $x = 0$ than that of $y = \log(1-x)$. Thus, minimizing $-\log(D(G(z)))$ leads to quicker and more substantial improvements in the generator's performance than minimizing $\log(1 - D(G(z)))$.

IV. Training GAN Components

A. Training the Discriminator

During discriminator training, the generator remains fixed. The discriminator maximizes the value function:

$$V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{17}$$

The optimal discriminator $D^*(x)$ is derived as:

$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)} \tag{18}$$

This ensures $D^*(x)$ approaches 1 for real samples ($p_{\text{data}}(x)$ dominates) and 0 for generated samples ($p_g(x)$ dominates).

B. Training the Generator

With the discriminator fixed, the generator minimizes the Jensen-Shannon divergence between p_{data} and p_g. Substituting $D^*(x)$ into the value function:

$$V(G, D^*) = -\log 4 + 2 \cdot J(p_{\text{data}} \parallel p_g) \tag{19}$$

Here, $J(p_{\text{data}} \parallel p_g)$ represents the Jensen-Shannon divergence, encouraging p_g to closely approximate p_{data}.

V. Collaborative Filtering Generative Adversarial Network (CFGAN)

Collaborative Filtering Generative Adversarial Network (CFGAN) is a model used in recommendation systems to generate synthetic user-item interaction data that closely resembles real data. It consists of two main components, the Generator (G) and the Discriminator (D), which work together in an adversarial process.

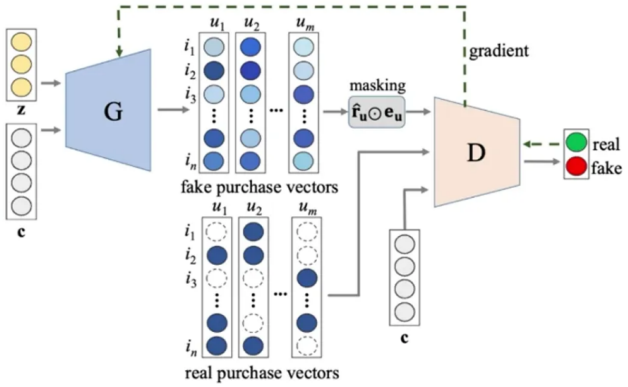


Fig. 2. CFGAN Structure

A. Components and Flow

- **Input to Generator** (G):
  - **Latent Vector** (z): A random vector sampled from a distribution, which introduces randomness in the generation process.
  - **Conditional Vector** (c): Represents user-specific information (such as preferences or demographic data), conditioning the generation of personalized recommendations.
- **Generator Output**:
  - **Fake Purchase Vectors**: The generator produces synthetic user-item interaction vectors (representing fake purchases) based on the latent vector z and conditional vector c. These vectors mimic real user preferences.
- **Masking Operation**:
  - The generated fake purchase vectors ($\hat{u}$) are combined with real purchase vectors u using a masking operation ($f_u \circ e_u$), where f_u is a user-specific mask and e_u represents the generated data. This maintains the sparsity seen in real data.
- **Discriminator** (D):
  - The discriminator evaluates both real and masked fake purchase vectors, classifying them as either **real** (green) or **fake** (red). This helps it learn to distinguish between actual and synthetic user-item interactions.
- **Gradient Flow**:
  - The feedback from the discriminator (real or fake classification) is used to adjust the generator's weights, guiding it to generate more realistic user-item interactions over time.

B. Evaluation Metrics

The model is evaluated using:

- **Precision:** Accuracy of recommended movies.
- **Recall:** Proportion of relevant movies recommended.
- **RMSE:** Root mean square error for rating predictions.

VI. Conclusion

In conclusion, Collaborative Filtering (CF) is a widely used technique for generating personalized recommendations based on user-item interactions. However, traditional CF methods often struggle with sparse data and cold-start problems. The introduction of Generative Adversarial Networks (GANs) offers a promising solution by leveraging adversarial training to generate synthetic data that mimics real-world patterns. Collaborative Filtering Generative Adversarial Network (CFGAN) combines the strengths of both CF and GAN, providing an effective approach to generating realistic user-item interactions and overcoming data sparsity. By employing adversarial learning, CFGAN enhances the quality of recommendations and facilitates the generation of more diverse and personalized datasets, ultimately improving the performance of recommendation systems.

Acknowledgements

I would like to express my sincere gratitude to Dr. Ravi Nahta for providing me with the opportunity to work on this research. Their guidance and support were invaluable throughout the process. I am deeply appreciative of the insights and resources they shared, which helped shape this work. Thank you for your constant encouragement and for fostering an environment of learning and growth.

References

1) S. Bag *et al.*, "A noise correction-based approach to support a recommender system in a highly sparse rating environment," *Decision Support Syst.*, 2019.
2) J. Liu *et al.*, "Cofigan: collaborative filtering by generative and discriminative training for one-class recommendation," *Knowl. Based Syst.*, 2020.
3) H. Bharadhwaj, H. Park, and B. Y. Lim, "Recgan: recurrent generative adversarial networks for recommendation systems," in *Proceedings of the 2018 Conference on Artificial Intelligence*, 2018.
4) D. Chae, J. Kang, S. Kim, and J. Choi, "Rating augmentation with generative adversarial networks towards accurate recommendation," in *Proceedings of the 2019 ACM Conference on Recommender Systems*, 2019.
5) D. Chae, J. Kang, S. Kim, and J. Lee, "CFGAN: a generic collaborative filtering framework based on generative adversarial networks," in *Proceedings of the 2020 IEEE International Conference on Artificial Intelligence*, 2020.