CMP2003 Recommender System Project

Can Sarı, Batuhan Arda Bekar, Ferhan Çelik, Gencay Demir Ardıç

I. Introduction

This report presents a hybrid recommender system that combines Singular Value Decomposition (SVD) and Neural Collaborative Filtering (NCF) approaches. Initially developed with Item-Based Collaborative Filtering (IBCF), the system was optimized for performance, achieving an RMSE of 0.9041 with a runtime of 0.0793 seconds. The code was submitted on HackerRank.

II. Related Work

Recommender systems have become integral to various online platforms, enhancing user experience by providing personalized content suggestions. Traditional collaborative filtering methods, such as user-based and item-based approaches, have been widely adopted [1]. Matrix factorization techniques like SVD [2] have shown significant promise in capturing latent features underlying user-item interactions. More recently, neural network-based methods, such as Neural Collaborative Filtering (NCF) [3], have emerged, leveraging deep learning to model complex user-item relationships.

Hybrid recommender systems, which combine multiple recommendation techniques, aim to mitigate the limitations inherent in individual methods. By integrating SVD and NCF, our system leverages the strengths of both latent factor models and deep learning-based approaches to enhance prediction accuracy and computational efficiency.

III. Dataset Description

Each entry in the dataset includes:

- **User ID**: Unique identifier for each user.
- **Item ID**: Unique identifier for each item.
- **Rating**: User's rating for the item, typically on a scale from 1 to 5.

IV. System Architecture

The implemented system utilizes a hybrid approach with the following weight distribution:

- SVD Component: 70% weight
- NCF Component: 30% weight
- IBCF Component: 0% (disabled for performance)

V. Algorithm Components

A. Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a matrix factorization technique widely used in recommender systems to uncover the latent factors that explain observed user-item interactions. Mathematically, SVD decomposes the user-item interaction matrix R into three lower-dimensional matrices:

$$R = U\Sigma V^T \tag{1}$$

where:

- $R \in R^{m \times n}$ is the user-item interaction matrix with m users and n items.
- $U \in R^{m \times k}$ is the user latent factor matrix, where each row p_u represents the latent factors for user u.
- $\Sigma \in R^{k \times k}$ is a diagonal matrix containing the singular values, which indicate the strength of each latent factor.
- $V \in R^{n \times k}$ is the item latent factor matrix, where each row q_i represents the latent factors for item i.

In the context of recommender systems, the goal is to approximate the original interaction matrix R by learning the latent factor matrices U and V. This approximation enables the prediction of missing entries in R, thereby providing personalized recommendations.

a) Prediction Formula: The predicted rating $\hat{r}_{ui}$ for user u and item i is computed by combining the global mean rating μ, user and item biases, and the interaction of user and item latent factors:

$$\hat{r}_{ui} = \mu + b_u + b_i + \sum_{f=1}^{k} p_{uf} \cdot q_{if} \tag{2}$$

where:

- $\hat{r}_{ui}$: Predicted rating for user u and item i.
- μ: Global mean rating across all user-item interactions.
- b_u: Bias term for user u, capturing the user's tendency to rate items higher or lower than the global mean.
- b_i: Bias term for item i, capturing the item's inherent attractiveness or quality.
- p_{uf}: Element f of the user latent factor vector p_u for user u.
- q_{if}: Element f of the item latent factor vector q_i for item i.
- k: Number of latent factors (set to 5 in this implementation), determining the dimensionality of the latent space.

b) Matrix Representation: In the implementation, user and item biases are represented as vectors, while user and item latent factors are represented as matrices:

- $\mathbf{b_u} \in R^m$: User bias vector, where each element corresponds to a specific user's bias.
- $\mathbf{b_i} \in R^n$: Item bias vector, where each element corresponds to a specific item's bias.
- $U \in R^{m \times k}$: User latent factor matrix, where each row p_u is a k-dimensional vector representing user u's latent factors.

- $V \in R^{n \times k}$: Item latent factor matrix, where each row q_i is a k-dimensional vector representing item i's latent factors.

c) Training Objective: The objective of training the SVD model is to minimize the prediction error between the observed ratings and the predicted ratings. This is typically achieved by minimizing the following loss function:

$$\mathcal{L} = \sum_{(u,i) \in \kappa} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(\|\mathbf{b_u}\|^2 + \|\mathbf{b_i}\|^2 + \|U\|^2 + \|V\|^2 \right) \tag{3}$$

where:

- κ is the set of observed user-item interactions.
- r_{ui} is the actual rating given by user u to item i.
- λ is the regularization parameter that controls the complexity of the model to prevent overfitting.

B. Neural Collaborative Filtering (NCF)

Neural Collaborative Filtering (NCF) is an advanced recommendation technique that leverages neural network architectures to model complex user-item interactions. Unlike traditional matrix factorization methods, NCF can capture non-linear relationships between users and items, enhancing the system's ability to make accurate predictions. In this implementation, we utilize a component of NCF known as Generalized Matrix Factorization (GMF), which combines the strengths of matrix factorization with neural network-based approaches.

a) Generalized Matrix Factorization (GMF): The Generalized Matrix Factorization (GMF) model makes predictions mathematically as follows:

User and Item Embeddings Let:

- $\mathbf{p}_u \in R^K$: Latent embedding vector for user u.
- $\mathbf{q}_i \in R^K$: Latent embedding vector for item i.
- K: Dimensionality of the embedding space.

Element-wise Multiplication GMF combines the user and item embeddings using an element-wise (Hadamard) product:

$$\mathbf{h}_{ui} = \mathbf{p}_u \odot \mathbf{q}_i \tag{4}$$

Here, $\mathbf{h}_{ui} \in R^K$ is the resulting vector representing the interaction between user u and item i.

b) Prediction Formula: The predicted rating $\hat{r}_{ui}$ for user u and item i is computed by taking the inner product of the user and item embeddings:

$$\hat{r}_{ui} = \mathbf{p}_u \cdot \mathbf{q}_i = \sum_{d=1}^{K} p_{u,d} \times q_{i,d} \tag{5}$$

where:

- $\mathbf{p}_u$: User embedding vector for user u.
- $\mathbf{q}_i$: Item embedding vector for item i.
- K: Dimensionality of the embedding vectors (set to 3 in this implementation).
- $p_{u,d}$ and $q_{i,d}$: The d-th component of the user and item embedding vectors, respectively.

c) Mathematical Framework: In GMF, each user u and item i is associated with an embedding vector $\mathbf{p}_u$ and $\mathbf{q}_i$, respectively. These embeddings are learned during the training process and are designed to capture the latent features that influence user preferences and item characteristics.

d) Training Objective: The objective of training the GMF model is to minimize the discrepancy between the actual ratings and the predicted ratings. This is achieved by minimizing the following loss function, which includes both the prediction error and a regularization term to prevent overfitting:

$$\mathcal{L} = \sum_{(u,i) \in \kappa} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 \right) \tag{6}$$

where:

- κ: Set of observed user-item interactions.
- r_{ui}: Actual rating given by user u to item i.
- λ: Regularization parameter that controls the complexity of the model.

e) Optimization: The model parameters $\mathbf{p}_u$ and $\mathbf{q}_i$ are optimized using Stochastic Gradient Descent (SGD). During each iteration, the gradients of the loss function with respect to each embedding vector are computed and the embeddings are updated accordingly:

$$\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta \left((r_{ui} - \hat{r}_{ui}) \cdot \mathbf{q}_i - \lambda \mathbf{p}_u \right)$$
$$\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta \left((r_{ui} - \hat{r}_{ui}) \cdot \mathbf{p}_u - \lambda \mathbf{q}_i \right)$$

where:

- η: Learning rate.

f) Matrix Representation: In the implementation, user and item embeddings are represented as matrices to facilitate efficient computations:

- $\mathbf{P} \in R^{m \times K}$: User embedding matrix, where each row $\mathbf{p}_u$ corresponds to a user's embedding vector.
- $\mathbf{Q} \in R^{n \times K}$: Item embedding matrix, where each row $\mathbf{q}_i$ corresponds to an item's embedding vector.

This matrix-based representation allows for scalable and efficient computation of inner products during prediction and gradient updates during training.

C. Item-Based Collaborative Filtering (IBCF)

Item-Based Collaborative Filtering (IBCF) is a recommendation technique that leverages the similarity between items to predict user preferences. Unlike User-Based Collaborative Filtering, which focuses on finding similar users, IBCF identifies items that are similar to those a user has previously interacted with and recommends them accordingly.

a) Cosine Similarity: The similarity between two items i and j is computed using cosine similarity, which measures the cosine of the angle between their respective feature vectors. Mathematically, it is defined as:

$$\text{similarity}(i,j) = \frac{\sum_{f=1} i_f \cdot j_f}{\sqrt{\sum_{f=1} i_f^2} \sqrt{\sum_{f=1} j_f^2}} \tag{7}$$

where:

- i_f and j_f: The f-th feature of items i and j, respectively.

b) Prediction Formula: The predicted interaction $\hat{p}_{ui}$ for user u and item i is calculated as a weighted average of the binary interactions of the user for items similar to i. The weights are the cosine similarities between item i and each similar item j:

$$\hat{p}_{ui} = \frac{\sum_{j \in N(i)} \text{similarity}(i,j) \cdot b_{uj}}{\sum_{j \in N(i)} \text{similarity}(i,j)} \tag{8}$$

where:

- $N(i)$: Set of top-K similar items to item i.
- b_{uj}: Binary interaction indicator (1 if user u interacted with item j, 0 otherwise).

VI. Implementation Details

A. Hyperparameters

The system uses the following optimized hyperparameters:

- SVD Learning Rate: 0.0178
- NCF Learning Rate: 0.04
- Regularization Factor: 0.052
- Training Epochs: 44
- SVD Factors: 5
- NCF Embedding Dimension: 3

B. Training Process

The training process employs a stochastic gradient descent approach with the following steps:

1) Initialize model parameters with small random values (normal distribution: $\mu = -0.01$, $\sigma = 0.01$)
2) For each epoch:
   - Compute hybrid predictions
   - Calculate prediction error
   - Update SVD components (biases and factors)
   - Update NCF embeddings
3) Apply regularization to prevent overfitting

C. Gradient Updates

The system updates parameters using the following formulas:
For SVD components:

$$\begin{aligned} b_u &\leftarrow b_u + \eta(e_{ui} \cdot w_{SVD} - \lambda b_u) \\ b_i &\leftarrow b_i + \eta(e_{ui} \cdot w_{SVD} - \lambda b_i) \\ p_{uf} &\leftarrow p_{uf} + \eta(e_{ui} \cdot w_{SVD} \cdot q_{if} - \lambda p_{uf}) \\ q_{if} &\leftarrow q_{if} + \eta(e_{ui} \cdot w_{SVD} \cdot p_{uf} - \lambda q_{if}) \end{aligned}$$

For NCF embeddings:

$$\begin{aligned} E_u &\leftarrow E_u + \eta(e_{ui} \cdot w_{NCF} \cdot E_i - \lambda E_u) \\ E_i &\leftarrow E_i + \eta(e_{ui} \cdot w_{NCF} \cdot E_u - \lambda E_i) \end{aligned}$$

where:

- η: Learning rate
- e_{ui}: Prediction error
- w_{NCF}: Component weights
- λ: Regularization factor

VII. Experimental Setup

A. Evaluation Metrics

The performance of the recommender system was evaluated using the following metrics:

- **Root Mean Square Error (RMSE)**: Measures the average magnitude of the prediction errors.
- **Runtime Performance**: Time taken to generate predictions for the test set.

VIII. Results

A. Runtime Performance

The optimized hybrid system demonstrated a runtime of 0.0799 seconds for generating predictions on the test set. The removal of the IBCF component contributed significantly to the reduced computational overhead.

B. Performance Study

A performance study was conducted to assess the impact of each component on the overall performance. Disabling the IBCF component resulted in a marginal decrease in accuracy but a substantial improvement in runtime, validating the decision to exclude it for performance gains.

IX. Explanations

The hybrid approach effectively combines the strengths of SVD and NCF, resulting in improved prediction accuracy and efficient runtime performance. SVD captures global patterns in user-item interactions, while NCF models complex, non-linear relationships through neural embeddings. The decision to disable IBCF was justified by the negligible impact on accuracy and the significant reduction in computational time.
One notable observation is the relatively low embedding dimension used in NCF (dimension = 3), which sufficed in capturing essential interaction features without introducing excessive computational complexity. Future experiments could explore varying embedding dimensions to further optimize performance.

X. Implementation Optimizations

Key optimizations include:

- Efficient data structures (vectors and unordered sets)
- Pre-computed global mean
- Optimized hyperparameters through experimentation
- Disabled synchronization with C++ streams
- Vector operations without unnecessary allocations
- Fixed-size dimensions for embeddings and factors

XI. Code

To provide a comprehensive understanding of the implementation, the core components of the Hybrid Recommender system are presented below using the `listings` package for syntax highlighting.

Listing 1. Hybrid Recommender System Class Definition

```
class HybridRecommender {
private:
    // SVD components
    std::vector<std::vector<float>> userFactors;
    std::vector<std::vector<float>> itemFactors;
    std::vector<float> userBiases;
    std::vector<float> itemBiases;
    float globalMean = 0.0f;

    // NCF components
    std::vector<std::vector<float>>
        userEmbeddings;
    std::vector<std::vector<float>>
        itemEmbeddings;

    // User-Item interactions for IBCF
    std::vector<std::unordered_set<int>>
        userInteractions; // Stores items each
        user has interacted with

    // Hyperparameters
    const int SVD_FACTORS = 5;
    const int NCF_EMBEDDING_DIM = 3;
    const float SVD_LR = 0.0178f;
    const float NCF_LR = 0.04f;
    const float REG = 0.052f;
    const int EPOCHS = 44;
    const float WEIGHTSVD = 0.7f;
    const float WEIGHTNCF = 0.3f; // 1.f -
        WEIGHTSVD
    const float WEIGHTIBCF = 0; // IBCF not used
        , therefore 0 weight
    const int IBCF_TOP_K = 10; // Number of top
        similar items to consider

    std::mt19937 rng{ 10486 };

public:
    void train(const std::vector<std::tuple<int,
         int, float>>& data);
    float predictRating(int userId, int itemId)
        const;
    void processTrainingData(std::istream& input
        );
};
```

XII. Experimental Results

A. Comparison with/without IBCF

While using IBCF provided a lower RMSE score, it drastically increased the runtime of the code. That is why, for runtime, we disabled it in the final model.

B. Impact of Hyperparameters

An analysis of hyperparameter sensitivity was conducted. Varying the number of SVD factors and NCF embedding dimensions revealed that a balance between model complexity and performance is crucial. Overfitting was observed with higher dimensions, while too low dimensions compromised the model's ability to capture intricate patterns.

XIII. Conclusion

The implemented hybrid recommender system demonstrates effective performance through the combination of SVD and NCF approaches. The removal of IBCF and careful optimization of hyperparameters resulted in both accurate predictions and efficient runtime performance.

Acknowledgments

We would like to thank Doç. Dr. Tevfik Aytekin and Simge Akay for their support and guidance throughout this project.

References

[1] F. Ricci, L. Bellucci, and J. Foscarini, *Introduction to Recommender Systems Handbook*, 1st ed. Springer, 2011.

[2] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, 2008, pp. 1257–1264.

[3] X. He, L. Dai, X. Yuan, Y. Sun, and X. Zhang, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 173–182.

[4] D. A. Harper and M. A. Konstan, "The MovieLens Datasets," *GroupLens Research*, University of Minnesota, 2015.