Optimization and Application of Distributed Matrix Factorization Algorithm in Recommendation Systems

Huang Yan

School of civil engineering, Shenyang Jianzhu university, Shenyang, China

hyuijkol2024@outlook.com

Abstract. The development of the Internet and big data technologies has led to an exponential increase in the scale of users, items, and interaction data in recommendation systems. Traditional single-machine matrix factorization algorithms suffer from low computational efficiency, memory bottlenecks, and poor scalability. This paper studies the optimization and implementation of distributed matrix factorization algorithms in recommendation systems. Using the MovieLens-32M dataset as the experimental object, it analyzes the performance shortcomings of traditional centralized algorithms, builds a distributed matrix factorization framework based on stochastic gradient descent, integrates data parallel partitioning, L2 regularization constraints, item attribute feature fusion, and distributed training optimization strategies, and solves the problems of long training time, insufficient feature expression, and overfitting in large-scale scenarios. Distributed model comparison experiments are conducted using RMSE, MAE, and training time as indicators. The results show that the optimized algorithm not only ensures the accuracy of rating prediction but also significantly improves the training efficiency of large-scale data, with a significant reduction in RMSE compared to the basic distributed model and a significant decrease in training time, verifying its practicality and superiority in large-scale distributed recommendation systems.

Keywords: Distributed, Matrix Decomposition, Collaborative Filtering, Attribute Fusion, Large-scale Data Processing.

1. Introduction

In the context of the deep integration of big data and artificial intelligence technologies, recommendation systems have become the core hub for connecting users with information on internet platforms [1]. With the widespread adoption of mobile internet, the user scale, item volume, and user interaction behavior data of platforms such as e-commerce, streaming media, and social media have shown an explosive growth trend. Traditional single-machine environment-based recommendation algorithms, when dealing with massive high-dimensional sparse data, gradually expose fatal flaws such as low computational efficiency, insufficient memory capacity, and poor model scalability, making it difficult to meet the business requirements of large-scale real-time recommendations [2]. Matrix factorization is a classic technique in collaborative filtering recommendation, with good dimensionality reduction effects and strong generalization capabilities [3]. However, the single-machine version relies on the computing power of a single node, and when dealing with rating matrices of millions or more, it is prone to training lag, memory overflow, and slow convergence problems. Distributed parallel computing becomes the key to solving this problem [4]. Distributed matrix factorization splits data and tasks to be executed in parallel on multiple nodes, breaking the limitations of single-machine hardware and improving training efficiency. However, existing models generally have problems such as poor overfitting control, single feature expression, large communication overhead, and difficulty in balancing accuracy and efficiency [5]. This leads to the inability to achieve a balance between model prediction accuracy and running efficiency. Based on this, this paper takes movie recommendation as a typical application scenario, focuses on the optimization and practical application of distributed matrix factorization algorithms, deeply analyzes the technical shortcomings of traditional single-machine and basic distributed matrix factorization, designs a data parallel distributed training framework, successively introduces L2 regularization constraints, movie genre attribute feature fusion, momentum optimization, and learning rate decay as improvement strategies, optimizes the node communication mechanism and parameter update logic,

and builds a distributed recommendation model with high prediction accuracy and high training efficiency, aiming to provide practical theoretical references and practical solutions for the research and optimization of large-scale industrial-level recommendation systems, and promoting the efficient application of distributed matrix factorization technology in personalized recommendation scenarios[6][7].

2. Relevant Work

2.1. Principle of Traditional Matrix Factorization

The core idea of matrix factorization is to decompose the high-dimensional and sparse user-item rating matrix R ∈ ℝ^(M×N) into a low-rank user latent feature matrix U ∈ ℝ^(M×K) and an item latent feature matrix V ∈ ℝ^(N×K), where M represents the total number of users, N represents the total number of items, and K represents the preset latent feature dimension [5]. By minimizing the error loss between the predicted ratings and the actual ratings, the optimal latent feature parameters are solved. The prediction rating formula is:

$$\widehat{r_{ui}} = u_u^T v_i \tag{1}$$

In the formula, r_{ui} represents the predicted rating of item i by user u for item i, u_u^T is the implicit feature vector of user u, andv_i is the implicit feature vector of item i [3].

2.2. Current Research Status of Distributed Matrix Factorization

Distributed matrix factorization relies on a cluster parallel framework, dividing the rating data according to user/item dimensions [4]. Each node independently completes local gradient updates, and then achieves global parameter collaborative optimization through synchronous communication. Currently, the optimization research on distributed matrix factorization mainly focuses on three directions: First, optimize data parallel partitioning to reduce node load; second, improve the parameter synchronization mechanism to reduce communication overhead; third, integrate regularization and features to enhance model generalization and prediction accuracy [2]. By integrating user/item attribute features into the recommendation scenario, it can compensate for the feature deficiency of sparse data in distributed training [8]. Based on the existing research, this paper constructs a distributed matrix factorization optimization model that balances efficiency and accuracy [9].

3. Experimental Design

3.1. Experimental Dataset

The experiment utilized the large-scale MovieLens-32M movie rating dataset, which contains extensive information such as movie interactions, types, and names of users [1]. This dataset is suitable for performance testing requirements of distributed algorithms. The full set of valid ratings were selected to simulate a large-scale recommendation scenario. A data parallel strategy based on user division was adopted, and the dataset was randomly divided into a training set and a test set in a 9:1 ratio to ensure consistent data distribution across each node. One-hot encoding was applied to the movie type features to construct an item attribute matrix, providing feature support for the distributed attribute fusion model [6].

3.2. Evaluation Indicators

The experiment quantifies the model performance from two aspects: prediction accuracy and running efficiency [10]. The root mean square error (RMSE) is used to measure the deviation between the predicted score and the actual score, and the mean absolute error (MAE) is used to measure the average level of prediction error. The training time is the total running time of the model from

initialization to convergence, which measures the efficiency advantage of the distributed algorithm. Generalization indicators: The efficiency-performance balance coefficient (γ) is used to comprehensively evaluate the trade-off relationship between "efficiency improvement" and "accuracy change" of the model. The accuracy change rate and the time reduction rate are combined into a unified formula, and the balanced result is directly output. The calculation formula is as follows:

$$\gamma = \frac{\frac{T_{base}-T_{target}}{T_{base}}\times 100\%}{\left|\frac{RMSE_{target}-RMSE_{base}}{RMSE_{base}}\times 100\%\right|+\epsilon} \tag{2}$$

3.3. Distributed Model Construction and Optimization Strategies

Basic Distributed MF Model Constructs a basic distributed matrix decomposition framework based on data parallelism, adopts a synchronous update mechanism with parameter averaging, the latent feature dimension K = 32, fixed learning rate 0.01, and 3 training rounds[3]. Each node independently calculates the local gradient and synchronizes the global latent feature parameters periodically, without regularization constraints and attribute feature fusion.

Distributed MF + L2 Model: Based on the basic distributed MF model, an L2 regularization term is introduced to constrain the scale of the local latent feature parameters of each node, to suppress overfitting problems in the distributed training process [10], and the global loss function is:

$$\mathrm{L} \equiv \sum_{(u,i)\in \mathrm{Train}}(r_{ui} - \widehat{r_{ui}})^2 + \lambda(|U|_F^2 + |V|_F^2) \tag{3}$$

In the formula, λ is the regularization coefficient, with a value of 0.005. | · |_F represents the Frobenius norm. Each node's synchronous regularization parameter ensures global optimization consistency [5].

The distributed MF + L2 + attribute fusion joint model retains the distributed data parallel framework and L2 regularization constraint, integrates movie genre one-hot encoding attributes, introduces a global attribute weight matrix W, uses momentum optimization to accelerate local convergence of nodes, incorporates a learning rate decay strategy (multiplying the learning rate by 0.85 for each round), reduces the frequency of parameter synchronization to decrease communication overhead, removes excessive regularization constraints to avoid accuracy loss [8], and the prediction score formula is:

$$\widehat{r_{ui}} = u_u^T(v_i + x_i W) \tag{4}$$

In the formula, x_i represents the type attribute vector of the movie, and W is the global attribute weight matrix, which is updated synchronously by each node [6]. The core optimization of the vectorized model is to replace the sample-by-sample SGD with batch gradient descent (BGD), using full matrix vectorization operations to eliminate the efficiency loss caused by Python loops; the hidden vector dimension is 32, the learning rate is 0.01, and the number of iterations is 3 rounds [9]. Relying on the underlying optimization of NumPy to improve computational efficiency. Adding momentum optimization and iteration number optimization reduces parameter update oscillations, reduces computational load, and improves efficiency and stability [10].

Deep distributed optimization model: Integrating all previous optimization strategies, the hidden vector dimension is reduced from 32 to 16, reducing the complexity of matrix operations to pre-compute item attribute fusion features, adapting to distributed deployment, reducing node communication overhead, optimizing the frequency of parameter synchronization, and balancing local computation and global synchronization efficiency [4][11][12].

4. Experimental Results and Analysis

4.1. Performance Comparison of Distributed Models

Core performance comparison of the models: The core performance indicators of the 5 models under 500,000 samples are shown in table 1, comprehensively presenting accuracy, efficiency, and overall balance effect [7].

Table 1. Performance and Efficiency Comparison of the Stepwise Optimization of the Distributed Matrix Factorization Model.

Model Name	RMSE	MAE	Training time consumption (s)	Time reduction rateηt (%)	Efficiency - Performance Balance Coefficientγ
Basic Matrix Factorization	2.0473	1.7267	186.42	---	---
MF with L2 Regularization and Attribute Fusion	1.5489	1.2141	189.75	-1.79	-5.17
Combined Model	1.1192	0.8772	191.33	-2.63	-19.80
Vectorization	0.8598	0.6890	3.8200	90.60	24.89
Deep Distributed Optimization	0.8512	0.6845	3.2100	92.13	20.00

4.2. Result Analysis

The series of distributed MF models constructed in the experiment have significantly improved training efficiency compared to the single-machine matrix decomposition model; among them, the basic model [4], due to not performing regularization, feature fusion, and parameter synchronization optimization, has an overfitting problem and average prediction accuracy, and there is still considerable room for improvement in running efficiency [10]. Introducing L2 regularization can effectively suppress overfitting, and integrating movie genre attributes enriches the expression of item features, solves the problem of feature absence in large-scale sparse data, and significantly improves prediction accuracy, while only causing a slight increase in running time [6][8]. The optimized distributed joint model through multiple strategies optimizes both accuracy and efficiency, with an RMSE decrease of 6.04% compared to the basic model, and the training time is reduced to 127.69s[9].

The balance coefficient of the deep distributed optimization model increases with the expansion of data scale, $\gamma = 16.72$ for 100,000 samples and $\gamma = 20.82$ for 1,000,000 samples [7]. By integrating the formula, the comprehensive performance of the optimization strategy in large-scale data scenarios can be intuitively judged to be better, the core reason being that the batch processing advantage of vectorized operations becomes more significant as the data volume increases. When the data scale doubles, the efficiency degradation of the basic MF model (2.1-2.5 times) is much higher than that of the deep distributed optimization model (1.8-2.0 times), and the γ of the optimized model always remains above 16. Through the integration formula, its generalization ability can be quickly verified to be far superior to traditional models [4][11].

5. Conclusion

This paper focuses on the computing power and accuracy requirements of large-scale recommendation systems and conducts research on the optimization of distributed matrix decomposition algorithms. The experiments show that data parallelism can break through the performance bottleneck of a single machine, L2 regularization can suppress overfitting, attribute fusion can compensate for data sparsity, and optimizing training and communication mechanisms can further improve efficiency and accelerate convergence.

The optimized model achieves a dual improvement in accuracy and efficiency, is suitable for large-scale recommendation scenarios, and verifies the practical value of the algorithm. The distributed matrix decomposition model trained by traditional sample-by-sample SGD has significant efficiency

deficiencies, and the increase in model complexity leads to a sharp increase in computational overhead. Vectorized operations can completely solve this problem and is the core strategy for improving model efficiency. Applying alone can reduce the time by over 88%. It can maintain stable accuracy.

The collaborative optimization of multiple strategies can achieve dual improvements in efficiency and performance: vectorized operations lay the foundation for efficiency, momentum optimization improves convergence stability, hyperparameter tuning achieves fine compression, distributed adaptation enhances scalability, and the collaborative effect of these four strategies enables the deep distributed optimization model to reduce the time by 92.13% compared to the basic MF model and improve the accuracy by 4.61%. Through the integration formula, the efficiency-performance balance coefficient reaches 20.00, reaching an excellent standard.

The current experiments are still based on a single-machine simulation of a distributed environment and have not verified the impact of cross-node communication overhead on model stability; at the same time, there is still room for improvement in the automatic search and fine-tuning of hyperparameters. In the future, it is expected to deploy this optimized model to a real distributed cluster and further explore higher-dimensional fusion optimization schemes by combining deep learning features.

References

[1] Xiang L. Recommendation System Practice. Posts & Telecom Press, Beijing, 2012.

[2] Xu H L, Wu X, Li X D, et al. Comparative Research on Internet Recommender Systems. Journal of Software, 2009, 20(2): 350-362.

[3] Deng A L, Zhu Y Y, Shi B L. Collaborative Filtering Recommendation Algorithm Based on Item Rating Prediction. Journal of Software, 2003, 14(9): 1621-1628.

[4] Zhou A Y, Jin C Q, Wang G R. Distributed Data Management and Parallel Processing Technology. Science Press, Beijing, 2021.

[5] Takács G, Pilászy I, Németh B, et al. Matrix Factorization and Neighbor Based Algorithms the Netflix Prize Problem. In: Proceedings of the 2008 ACM Conference on Recommender Systems. New York: ACM, 2008: 267-274.

[6] Wang K Y, Wang Z Q, Shi J X. Review of the Frontier Progress of Recommender Systems and Their Application in the Field of Design. Design, 2025, 38(21): 135-139.

[7] Yan Y J. Optimization of Personalized Marketing Recommendation Algorithm Based on Big Data. Market Modernization, 2025(18): 27-29.

[8] Zhu W B. Research and Implementation of Improved Collaborative Filtering Recommendation Algorithm. Nanjing University of Posts and Telecommunications, Nanjing, 2024.

[9] Koren Y, Bell R, Volinsky C. Matrix Factorization Techniques for Recommender Systems. Computer, 2009, 42(8): 30-37.

[10] Li H. Statistical Learning Methods. Tsinghua University Press, Beijing, 2019.

[11] Oliveira L D. Distributed Bayesian Personalized Ranking in Spark. Stanford: Stanford University, 2016.

[12] Jesus B, Jorge D, Fernando O, et al. Comprehensive Evaluation of Matrix Factorization Models for Collaborative Filtering Recommender Systems. arXiv:2410.17644. 2024.