RainFM: A Stratified Hybrid Model for Enhanced Predictive Accuracy in Recommender Systems

Rainer Feichtinger, Rongxing Liu, Justin Lo, Ruben Schenk
Group: Recommenders
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—This work presents an innovative approach, RainFM, to enhance prediction accuracy in a specific recommender system dataset given by the Computational Intelligence Lab 2023 at ETH Zurich. RainFM combines strengths from a range of established models, making use of traditional and advanced methodologies, to create a refined interpretation of the underlying data. Notably, it adopts a stratified approach, dividing the dataset into distinct statistical groups and treating these groups individually.**

In comparison against several widely-used models such as matrix factorization through Singular Value Decomposition, generalized matrix factorization, Multi-Layer Perceptron, NeuFM, and Bayesian Factorization Machines, RainFM consistently outperformed all, showcasing superior predictive accuracy. However, the method is associated with certain challenges, including computational cost and careful hyperparameter tuning.

I. Introduction

Recommender systems have long become a significant component of many digital platforms, effectively tailoring user experiences across various sectors. The strength of these systems lies in their ability to accurately predict the preferences of users, a task often achieved through Collaborative Filtering (CF). Our work focuses on comparing, innovating, and improving existing CF strategies, particularly for user ratings on items based on a 1-5 star scale. In other words, we are concerned with the explicit feedback case, meaning that users directly rate items. This problem has famously been subject to the Netflix Prize competition [1].

Using a dataset of ratings from 10'000 users across 1'000 items, we aim to refine the prediction accuracy of these systems. Our initial model, based on the straightforward and well-researched Singular Value Decomposition (SVD) and the Alternating Least Squares (ALS) technique served as a benchmark. Recognizing the need for a more nuanced approach, we expanded our toolkit to include Factorization Machines (FM) [2], specifically Bayesian Factorization Machines (BFM) [3] and Neural Collaborative Filtering systems [4]. We developed our final model based on the strengths and weaknesses of those different models.

II. Models and Methods

Matrix Factorization (MF) techniques are the foundation of many CF approaches to solve matrix completion problems. These techniques take a high-dimensional dataset and factorize it into a product of low-dimensional matrices, thereby reducing the dimensionality of the data.

Consider we are given a sparse matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m = 10'000$ and $n = 1'000$, where entry $(\mathbf{A})_{ij}$ denotes the rating user i gave item j. The aim of MF techniques is to find a factorization of $\mathbf{A}$. We know that for each matrix there exists the SVD; in other words, there exists orthogonal matrices $\mathbf{U} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{m \times m}$ such that $\mathbf{A}$ can be expressed as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \quad \mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_{\min\{m,\,n\}}), \quad \sigma_i \geq \sigma_{i+1}\,(\forall i). \tag{1}$$

In the context of CF, the two low-rank matrices $\mathbf{U}$ and $\mathbf{V}$ represent the latent factors of users and items, respectively. Here, row $(\mathbf{A})_{i,:}$ corresponds to a latent factor representation of user i, and similarly, the column $(\mathbf{V})_{:,j}$ to a latent factor representation of item j. The singular values in matrix $\mathbf{\Sigma}$ determine the significance of each basis vector, which in turn means that by keeping only the top k singular values and setting the remaining ones to 0, we can get a k-dimensional approximation of our data.

A. Baseline

While SVD can give a good approximation of the data, it is prone to overfitting when subjected to noise. In our baseline approach we therefore make use of the iterative shrinkage SVD algorithm [5]. In this approach, we shrink the singular values by some amount in each iteration, clipping them at zero if they become negative. The code for the iterative shrinkage SVD algorithm is presented in Algorithm 1.

The resulting matrix $\mathbf{A}_\tau$ of Algorithm 1 is a low-rank, noise-reduced approximation of our initial matrix $\mathbf{A}$, reconstructing the missing data entries through SVD matrix factorization. The parameters we use for our dataset are $\tau = 50$, $n = 20$, and $\eta = 0.2$. This provides a good initial approximation, which we further improve through Alternating Least Squares (ALS).

ALS is another matrix factorization technique which alternately fixes one factor matrix, either $\mathbf{U}$ or $\mathbf{V}$, and optimizes the other. We define the approximation error for which we

Algorithm 1 Iterative Shrinkage SVD

Input: $\mathbf{A}$, τ, n, η

$\mathbf{X} \leftarrow \mathbf{0}$
for $i \leftarrow 1$ to n **do**
  $\mathbf{U}, \mathbf{s}, \mathbf{V}^\top \leftarrow \text{SVD}(\mathbf{X})$ ▷ Perform SVD on X
  $\mathbf{s} \leftarrow \max(\mathbf{s} - \tau, 0)$ ▷ Shrink singular values
  $\mathbf{A}_\tau \leftarrow \mathbf{U} \cdot \mathbf{s} \cdot \mathbf{V}^\top$ ▷ Reconstruction
  $\mathbf{A}_{\text{diff}} \leftarrow \Pi_\Omega(\mathbf{A} - \mathbf{A}_\tau)$ ▷ Get masked difference
  $\mathbf{X} \leftarrow \mathbf{X} + \eta \cdot \mathbf{A}_{\text{diff}}$ ▷ Update X
end for
return $\mathbf{A}_\tau$

optimize the factor matrices as

$$l(\mathbf{U},\, \mathbf{V}) = \frac{1}{2}||\Pi_\Omega(\mathbf{A} - \mathbf{U}\mathbf{V})||_F^2 + \lambda(||\mathbf{U}||_F^2 + ||\mathbf{V}||_F^2), \quad (2)$$

with $\lambda > 0$. Given this objective, we can define the ALS algorithm as

$$\begin{aligned} \mathbf{V}^{t+1} &\leftarrow \arg\min_{\mathbf{V}} l(\mathbf{U}^t,\, \mathbf{V}), \\ \mathbf{U}^{t+1} &\leftarrow \arg\min_{\mathbf{U}} l(\mathbf{U},\, \mathbf{V}^{t+1}). \end{aligned} \quad (3)$$

The implementation of ALS is shown in Algorithm 2. The resulting matrix $\mathbf{A} = \mathbf{U}\mathbf{V}^\top$ provides one of our baselines for our CF problem. Through experimenting we found the parameters $n = 10$ and $\lambda = 0.15$ to be providing the best results for our specific dataset.

Algorithm 2 Alternating Least Squares

Input: $\mathbf{A}$, λ, n
$\mathbf{U}, \mathbf{s}, \mathbf{V} \leftarrow \text{SVD}(\mathbf{A})$
for $t \leftarrow 1$ to n **do**
  for rows $\mathbf{u}_i$ of $\mathbf{U}$ **do**
    $\mathbf{u}_i^* \leftarrow (\sum_j \omega_{ij}\mathbf{v}_j\mathbf{v}_j^T + 2\lambda\mathbf{I})^{-1}(\sum_j \omega_{ij}a_{ij}\mathbf{v}_j)$
  end for
  for columns $\mathbf{v}_j$ of $\mathbf{V}$ **do**
    $\mathbf{v}_j^* \leftarrow (\sum_i \omega_{ij}\mathbf{u}_i\mathbf{u}_i^T + 2\lambda\mathbf{I})^{-1}(\sum_i \omega_{ij}a_{ij}\mathbf{u}_i)$
  end for
end for
return $\mathbf{A} \leftarrow \mathbf{U}\mathbf{V}^T$

B. Neural Collaborative Filtering

Given the significant influence that deep neural networks have in a variety of computational disciplines, we decided to explore this approach as a comparative measure against our existing methodologies. The main focus lies on the three methods presented in [4]. Let us consider a factorization of $\mathbf{A} = \mathbf{U}\mathbf{V}^\top$ where $\mathbf{U} \in \mathbb{R}^{n\times k}$ and $\mathbf{V} \in \mathbb{R}^{m\times k}$, symbolizing the latent factor matrices for users and items, in that order. Additionally, we denote $\mathbf{v}_i^U$ and $\mathbf{v}_j^I$ as the feature vectors that characterize user i and item j, respectively.

1) Generalized Matrix Factorization: The first strategy, known as Generalized Matrix Factorization (GMF), leverages the proven MF method, elaborated on in Section II-A. Within this approach, the input layers constitute to one-hot encoded representations of user i and item j. The resulting embedding vector then serves as the latent representation for the corresponding user or item.

We can define the latent vector for user $\mathbf{p}_i$ as $\mathbf{U}^\top\mathbf{v}_i^U$ and for item $\mathbf{q}_j$ as $\mathbf{V}^\top\mathbf{v}_j^I$. Based on this, the mapping function for the first neural CF layer can be expressed as

$$\phi_1(\mathbf{p}_i,\, \mathbf{q}_j) = \mathbf{p}_i \odot \mathbf{q}_j, \quad (4)$$

where $\odot$ denotes the element-wise multiplication of vectors.

Subsequently, we can project the resultant vector onto the output layer as follows:

$$\hat{y}_{ij} = a_{\text{out}}(\mathbf{h}^\top(\mathbf{p}_i \odot \mathbf{q}j)), \quad (5)$$

where a_{out} signifies the activation function and $\mathbf{h}$ represents the weights of the output layer edges.

2) Multi-Layer Perceptron: With a dual-pathway model at our disposal, a straightforward approach would involve concatenating the feature vectors derived from both pathways. However, this overlooks the potential interactions between the latent characteristics of users and items, a critical aspect for capturing the CF effect. To address this, a Multi-Layer Perceptron (MLP) is utilized, facilitating the capture of interactions between user and item latent features, as suggested by [4]. The MLP model can be defined as follows:

$$\begin{aligned} \mathbf{z}_1 &= \phi_1(\mathbf{p}_i,\, \mathbf{q}_j) = \begin{bmatrix}\mathbf{p}_i \\ \mathbf{q}_j\end{bmatrix}, \\ \phi_2(\mathbf{z}_1) &= a_2(\mathbf{W}_2^\top\mathbf{z}_1 + \mathbf{b}_2), \\ &\dots \\ \phi_L(\mathbf{z}_{L-1}) &= a_L(\mathbf{W}_L^\top\mathbf{z}_{L-1} + \mathbf{b}_L), \\ \hat{y}_{ij} &= \sigma(\mathbf{h}^\top\phi_L(\mathbf{z}_{L-1})). \end{aligned} \quad (6)$$

Here, $\mathbf{W}_x$, $\mathbf{b}_x$, and a_x represent the weight matrix, bias vector, and activation function of the x-th layer's perceptron, respectively. By introducing hidden layers on top of the concatenated vector, the model gains enhanced flexibility and non-linearity, empowering it to learn the interactions between $\mathbf{p}_i$ and $\mathbf{q}_j$.

3) GMF and MLP Blend: Our third method within the neural network approach involves the NeuFM model as proposed by [4]. This model effectively blends the strengths of both the linear kernel model (GMF) and the non-linear kernel model (MLP). The NeuFM model can be expressed as:

$$\hat{y}_{ij} = \sigma(\mathbf{h}^\top a(\mathbf{p}_i \odot \mathbf{q}_j + \mathbf{W}\begin{bmatrix}\mathbf{p}_i \\ \mathbf{q}_j\end{bmatrix} + \mathbf{b})) \quad (7)$$

This design accommodates datasets where optimal embeddings have varying sizes by allowing the model to learn separate embeddings, enhancing its flexibility. The formulation

we used for our comparisons is:

$$\begin{aligned} \phi^{\text{GMF}} &= \mathbf{p}_i^G \odot \mathbf{q}_j^G, \\ \phi^{\text{MLP}} &= a_L(\mathbf{W}_L^\top \psi + \mathbf{b}_L), \end{aligned} \tag{8}$$

where

$$\psi = a_{L-1}(\cdots a_2(\mathbf{W}_2^\top \begin{bmatrix} \mathbf{p}_i^M \\ \mathbf{q}_j^M \end{bmatrix} + \mathbf{b}_2)\cdots), \tag{9}$$

and

$$\hat{y}_{ij} = \sigma(\mathbf{h}^\top \begin{bmatrix} \phi^{\text{GMF}} \\ \phi^{\text{MLP}} \end{bmatrix}). \tag{10}$$

To improve the performance of the NeuFM blend, we performed pretraining by first separately training on the GMF and MLP models. These weights were then saved and loaded into the NeuFM architecture, where the model was further trained using Stochastic Gradient Descent. This served to stabilise training and improve the eventual performance of the NeuFM model.

C. (Bayesian) Factorization Machines

Factorization Machines (FM) [2] are an advancement of linear regression and MF models. Rather than applying traditional MF methods that leverage a user-item interaction matrix as input, FMs utilize tuples of real-valued feature vectors and numeric target variables to model user-item interactions. An n-way FM is employed to model interactions of n variables for a target y using p explanatory variables $\mathbf{x} \in \mathbb{R}^p$. Specifically, the n-way FM is formulated as:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{n=2}^{N} \sum_{j_1=1}^{p} \cdots \sum_{j_n > j_{n-1}}^{p} w_{j_1,\cdots j_d} \prod_{l=1}^{n} x_{jl} \tag{11}$$

Here, $\mathbf{V} \in \mathbb{R}^{d \times k}$ denotes the feature embeddings and k represents the dimensionality of latent factors.

In practical applications, a 2-way FM is most commonly used, defined as:

$$y(\mathbf{x}) = w_0 + \sum_{j=1}^{p} w_j x_j + \sum_{i=1}^{p} \sum_{j=i+1}^{p} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \tag{12}$$

This model is effective for handling large dimensional sparse inputs, thanks to an efficient optimization method associated with FMs that reduces the computational time from polynomial to linear.

Bayesian FM (BFM) is a further development of FM which integrates Bayesian inference. The Bayesian theory recommends incorporating hierarchical hyperpriors for regulating the model parameters. In a typical FM model, the standard hyperparameters μ, λ, α are augmented by hyperpriors $\Theta_0 = \{\alpha_0, \beta_0, \mu_0, \gamma_0, \alpha_\lambda, \beta_\lambda\}$.

To sample from the posterior, we use Gibbs sampling, leveraging the multilinear nature of FM. This approach defines the relationship between the target and explanatory variables as:

$$y(\mathbf{x}|\Theta) = g_\theta(\mathbf{x}) + \theta h_\theta(\mathbf{x}) \quad \forall \theta \in \Theta \tag{13}$$

BFM, given its Bayesian approach, further generates conditional posterior distributions for each parameter and hyperparameter. While we omit the specific mathematical expressions for these distributions for brevity, it's essential to understand that these distributions allow the model parameters to adjust more effectively based on the data. This mechanism enhances the model's flexibility, thus improving its ability to capture complex data patterns while preventing overfitting. Finally, the algorithm works by iteratively drawing samples for hyperparameters and model parameters.

D. Adaptation of Baseline Approaches

Due to the promising baseline score of the BFMs, we initially explored methods that could improve on its performance. One benefit of the BFMs is its ability to produce probabilistic outputs. Consequently, we experimented with using the BFM outputs as a pretraining for other methods. As an initial experiment, we utilized the BFM model to predict the output of all user-item pairs. We then augmented our original training data with a subset of these predictions and used the augmented dataset to perform various methods such as the SVD or K-Nearest-Neighbours (KNN) models. Although this did not directly improve the results, it inspired further experiments into how we could combine methods together.

An effective strategy involved applying a KNN model with $k = 30$ clusters to the initial training dataset. This model allowed us to assign each datapoint to a cluster and obtain a corresponding weighted rating for the datapoint. The KNN pretraining was followed by augmenting the original training dataset by evenly sampling datapoints from each cluster, thus expanding the training dataset. This enriched dataset was then used to train a BFM model, where we noted an improvement in the Mean Absolute Error (MAE) over the baseline BFM model.

E. Our Method (RainFM)

Observing the possible success in combining models, our proposed RainFM approach leverages gradient boosting and a variety of baseline models to create a combination of linear models for heightened prediction accuracy. We found, empirically, that linear models delivered superior outcomes for our specific dataset.

Given the enormity of potential model combinations – equating to $2^{|Models|}$ – we had to exclude the exhaustive testing of all possible combinations due to computational constraints. Instead, we applied a forward selection approach based on the models detailed in Table I. In each step, we incorporated the model that provided the most substantial reduction in test Root Mean Squared Error (RMSE), continuing this process until no further improvements were noted.

We also adopted a stratified approach by partitioning the dataset into distinct subsets. This strategy stemmed from the observation that certain models might demonstrate better

performance within specific data groups than over the entire dataset. We experimented with a variety of grouping strategies, exploring different group numbers and characteristics such as the number of ratings a movie received, average user ratings, and average movie ratings. The optimal outcome was achieved by grouping based on the number of user ratings and utilizing quantiles to define these groups. Within each group, forward selection was used to identify the most effective models. These were then linearly combined to create a group-specific model.

Our final RainFM model is a linear amalgamation of the group-specific models and the model based on the entire dataset, achieving an improved balance between specificity and generalization.

III. Results

We performed extensive evaluations on our dataset, separating it into a training and validation set using an 80/20 split. The resulting performance metrics, specifically RMSE and MAE, allowed us to compare and contrast the efficiency of the various models tested. Table I encapsulates these results.

Method	*RMSE*	*MAE*
Item average	1.0309	0.8398
SVD & ALS	0.9921	0.7896
Generalized MF	1.0822	0.8795
Multi-Layer Perceptron	1.0029	0.8105
NeuFM (Pretrained)	1.0041	0.8092
BFM Baseline	0.9777	0.7809
BFM (augmented with KNN)	0.9840	0.7808
RainFM	**0.9695**	**0.7714**

Table I
Results of the different methods on our specific dataset. Bold entries mark the best performing method.

IV. Discussion

Our RainFM approach demonstrates substantial improvements in predictive accuracy, as confirmed by superior RMSE and MAE scores in Table I. This approach, specifically developed with our dataset in mind, leverages the strengths of various models, enabling a nuanced understanding of the data. One of the key attributes of RainFM is the use of partitioning, which caters to the multifaceted nature of user behavior. By breaking down the dataset into distinct statistical groups, we manage to make the recommendations more personalized and more effective.

However, the strengths of RainFM come with certain complexities. The combination of models and the tuning of multiple hyperparameters can increase computational demands. Additionally, identifying the optimal characteristics for data partitioning could be challenging in certain cases. Yet, its application can lead to significant enhancements in recommendation systems.

V. Summary

This work introduced RainFM, an innovative approach tailored to our specific dataset, with the aim to enhance prediction accuracy in recommendation systems. Our approach merges strengths from various models, making use of both traditional techniques and advanced models. In addition, we have employed a stratified approach to the data, treating different groups differently, based on statistical measures such as the number of ratings per user.

Our evaluations showed that RainFM outperforms several well-established models, underscoring its effectiveness. However, the complexity of the model does present certain challenges, such as computational cost and the need for meticulous tuning of hyperparameters.

Acknowledgements

The authors thank André Minder for his careful reading and helpful suggestions.

References

[1] J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of the KDD Cup Workshop 2007*. ACM, 2007, pp. 3–6.

[2] S. Rendle, "Factorization machines," in *2010 IEEE International conference on data mining*. IEEE, 2010, pp. 995–1000.

[3] ——, "Bayesian factorization machines," in *Proceedings of the NIPS Workshop on Sparse Representation and Low-rank Approximation*, 2011.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," 2017.

[5] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on Optimization*, vol. 20, no. 4, pp. 1956–1982, Jan. 2010.