Towards a Better Understanding of Linear Models for Recommendation

Ruoming Jin
rjin1@kent.edu
Kent State University
USA

Dong Li
dli12@kent.edu
Kent State University
USA

Jing Gao
jgao@ilambda.com
iLambda
USA

Zhi Liu
zliu@ilambda.com
iLambda
USA

Li Chen
lchen@ilambda.com
iLambda
USA

Yang Zhou
yangzhou@auburn.edu
Auburn University
USA

ABSTRACT

Recently, linear regression models have shown to often produce rather competitive results against more sophisticated deep learning models. Meanwhile, the (weighted) matrix factorization approaches have been popular choices for recommendation in the past and widely adopted in the industry. In this work, we aim to theoretically understand the relationship between these two approaches, which are the cornerstones of model-based recommendations. Through the derivation and analysis of the closed-form solutions for two basic regression and matrix factorization approaches, we found these two approaches are indeed inherently related but also diverge in how they "scale-down" the singular values of the original user-item interaction matrix. We further introduce a new learning algorithm in searching (hyper)parameters for the closed-form solution and utilize it to discover the *nearby* models of the existing solutions. The experimental results demonstrate that the basic models and their closed-form solutions are indeed quite competitive against the state-of-the-art models, thus, confirming the validity of studying the basic models. The effectiveness of exploring the nearby models are also experimentally validated.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Learning linear models**; **Factorization methods**.

KEYWORDS

Recommender systems; linear model; low-rank regression; matrix factorization; hyper-parameter search

ACM Reference Format:
Ruoming Jin, Dong Li, Jing Gao, Zhi Liu, Li Chen, and Yang Zhou. 2021. Towards a Better Understanding of Linear Models for Recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467428

KDD '21, August 14–18, 2021, Virtual Event, Singapore

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467428

1 INTRODUCTION

Over the last 25 years, we have witnessed a blossom of recommendation algorithms being proposed and developed[1, 33]. Though the number of (top-n) recommendation algorithms is fairly large, the approaches can be largely classified as neighborhood approaches (including the regression-based approaches), matrix factorization (or latent factors) approaches, more recent deep learning based approaches, their probabilistic variants, and others[1, 33]. However, there have been some interesting debates on the results being reported by recent deep learning-based approaches: the experimental results show most of these methods seem to achieve sub-par results when compared with their simple/nonlinear counterparts [9]. This issue also relates to selecting and tuning baselines [26] as well as the choice of evaluation metrics [22], among others. But recent studies also seem to confirm the state-of-the-art linear models, such as SLIM [24] and EASE [29] do obtain rather remarkable results when compared to the more sophisticated counterparts [8].

Intuitively, SLIM and EASE search an item-to-item similarity matrix W so that the user-item interaction (denoted as matrix X with rows and columns corresponding to users and items, respectively) can be recovered by the matrix product: XW. In fact, they can be considered as simplified linear auto-encoders [30], where W serves as both encoder and decoder, denoted as L_W, such that $L_W(x_u) = x_u W$ can be used to recover x_u (x_u is the u-th row vector of user-item interaction matrix X). In the meantime, the matrix factorization methods, such as ALS [17] and SVD-based approaches [20] have been heavily favored and widely adopted in industry for recommendation. They aim to discover user and item embedding matrices P and Q, where p_u, q_i represents the latent factors of user u and item i, respectively, such that the user item interaction x_{ui} can be approximated by $q_i^T p_u$. Furthermore, there have been a list of *low-rank* regression approaches [5, 27, 30] which aim to factorize the similarity matrix W as AB^T, such that XAB^T can be used to recover X. Here, XA and B also introduces the user and item matrices, respectively (similar to matrix factorization).

Thus, on the surface, we can see the (reduced-rank) regression is like a special case of matrix factorization [18, 28], and it also has seemingly smaller number of parameters (the size of similarity matrix W is typically much smaller than the user and item latent

factor matrix as the number of users tend to be much larger than items). Therefore, the expectation is the regression models are more restricted, and thus less flexible (and expressive) than the matrix factorization approaches. However, the recent results seem to indicate the regression approaches tend to perform better than the matrix factorization approaches in terms of commonly used evaluation criteria, such as Recall and nDCG [8, 29, 30].

Is there any underlying factor/reason for the regression approaches to perform better than the matrix factorization approaches? If and how these two different factorization (low-rank regression vs matrix factorization) relate to one another? To seek the connection and be able to compare/analyze their inherent advantage/weakness, can we unify them under the same framework? As most of the deep learning, probabilistic and non-linear approaches all have the core representation from either factorization [15] or auto-encoders [23], the answer to these questions will not only help understand two of the (arguably) most important recommendation methodologies: neighborhood vs matrix factorization, but also help design more sophisticated deep learning based approaches. To our surprise, the theoretical analyses of these approaches are still lacking and the aforementioned questions remain unanswered.

In this paper, by analyzing two basic (low-rank) regression and matrix factorization models, we are able to derive and compare their closed-form solutions in a unified framework. We reveal that both methods essentially "scale-down" their singular values of user-item interaction matrix using slightly different mechanisms. The (low-rank) regression mechanism allows the use of more *principal components* (latent dimensions) of the user-item matrix X than the matrix factorization approaches. Thus, this potentially provides an inherent advantage to the former methods over the latter. Another surprising discovery is that although the matrix factorization seems to have more model parameters with both user and item latent factor matrices, its optimal solution for the simplified problem suggests that it is actually only dependent on the item matrix. Thus, it is actually more restricted and less flexible than the regression based approaches. This again indicates the potential disadvantage of the matrix factorization approaches.

To help further understand how the singular values of user-item interaction matrix can be adjusted (at individual level), we introduce a novel learning algorithm which can search through high dimensional continuous (hyper)parameter space. This learning algorithm also enables us to perform the post-model fitting exploration [12] for existing linear recommendation models. Our approach is to augment (existing) linear models with additional parameters to help further improve the model accuracy. The resulting models remaining as linear models, which can be considered as *nearby* models with respect to the existing models. This approach indeed shares the similar spirit of the our recently proposed "next-door analysis" from statistical learning [14] though our approaches and targets are quite different. To the best of our knowledge, this is the first work to study post-model fitting exploration for recommendation models. Such study can not only help better evaluate the optimality of the learned models, but also (potentially) produce additional boost for the learned models.

As pointed out by [8], a major problem in existing recommendation research is that the authors tend to focus on developing new methods or variants of recommendation algorithms, and then validate based on "hyper-focus on abstract metrics" with often weak or not-fully-tuned baselines to "prove" the progress. Though using better baselines, datasets, and evaluation metrics can help address of part of the problem, a better understanding of how, why, and where the improvement over existing are being made is equally important. We hope the theoretical analysis, the new learning tool for (hyper)parameter search, and the post-model analysis on linear recommendation models can be part of the remedy for the aforementioned problem.

To sum, in this paper, we made the following contribution:

- (Section 3) We theoretically investigate the relationship between the reduced-rank regression (neighborhood) approaches and the popular matrix factorization approaches using the closed-form solutions, and reveal how they connect with one another naturally (through the lens of well-known principal component analysis and SVD). We also discover some potential factors which may provide a benefit for the regression based methods.
- (Section 4) We introduce a new learning algorithm to help search the high-dimension (hyper)parameter space for the closed-form solution from Section 3. We further apply the learning algorithm to perform post-model exploration analysis on the existing linear models by augmenting them with additional parameters (as nearby models).
- (Section 5) We experimentally validate the closed-form solution for the basic regression and matrix factorization models, and show their (surprising) effectiveness and accuracy comparing against the state-of-the-art linear models; we also experimentaly validate the effectiveness of the learning algorithms for the closed-form solutions and identifying nearby models. We show nearby models can indeed boost the existing models in certain datasets.

2 BACKGROUND

Let the training dataset consists of m users and $n = |I|$ items, where I is the entire set of items. In this paper, we will focus on the implicit setting for recommendation. Compared with the explicit settings, the implicit setting has more applications in ecommerce, content recommendation, advertisement, among others. It has also been the main subjects for recent top-n recommendation [9, 17, 24, 29, 33]. Here, the user-item interaction matrix X can be considered as a binary matrix, where $x_{ui} = 1$ represents there is an interaction between user u and item i. If there is no interaction between u and i, then $x_{ui} = 0$. Let $X_u^+ = \{j : x_{uj} > 0\}$ denote the item set that user u has interacted with, and $X_u^- = I - X_u^+$ to be the item set that u has not interacted with.

2.1 Regression Models for Neighborhood-based Recommendation

It is well-known that there are user-based and item-based neighborhood based collaborative filtering, and the item-based approach has shown to be more effective and accurate compared with user-based approaches [10]. Thus, most of the linear models are item-based collaborative filtering (ICF).

Intuitively, the model-based ICF aims to predict x_{ui} (user u's likelihood of interaction with and/or preference of item i) based on

user u's past interaction with other items X_u^+:

$$\hat{x}_{ui} = \sum_{j \in X_u^+} s_{ji} x_{uj}, \tag{1}$$

where s_{ji} denotes the similarity between item j and i.

The initial neighborhood approach uses the statistical measures, such as Pearson correlation and cosine similarity [1] between the two columns X_{*i} and X_{*j} from items i and j. The more recent approaches have been aiming to use a regression approach to directly learn the weight matrix W (which can be considered as the inferred similarity matrix) so that $||X - XW||_F^2$ ($||\cdot||_F$ denotes the Frobenius norm) is minimized. Clearly, in this formulation, the default solution $W = I$ should be avoided for generalization purpose, and the difference of different approaches lie in the constraints and regularization putting on W. Recent studies have shown these approaches achieve comparable or even better performance compared with the state-of-the-art deep learning based approaches [8, 30].

SLIM: SLIM [24] is one of the first regression-based approach to infer the weight matrix W. It considers W to be nonnegative, and regularizing it with L_1 and L_2 norm (thus *ElasticNet*) [35]. In addition, S require to be zero diagonal:

$$\begin{aligned} W = \arg\min_W \frac{1}{2}||X - XW||_F^2 + \lambda_1||W||_1 + \lambda_2||W||_F^2 \\ s.t.\ \ W \geq 0, diag(W) = 0, \end{aligned} \tag{2}$$

where $||\cdot||_1$ denotes the L_1 matrix norm, and $diag(\cdot)$ denotes the diagonal (vector) of the corresponding matrix. Since no closed-form solution for W, the solver of ElasticNet is used to optimize W, and λ_1 and λ_2 are the correspondingly regularization hyperparameters.

There are a quite few variants of SLIM being proposed, including HOLISM [6] which extends SLIM to capture higher-order relationship, and LRec [28] which considers a non-linear logistic loss (instead of squared loss) with no zero diagonal, no negative and $L1$ constraints, among others.

EASE: EASE [29] is a recent regression-based model which has shown to improve over SLIM with both speed and accuracy, and quite competitive again the state-of-the-art deep learning models. It simplifies the constraint and regularization enforced by SLIM by removing non-negative and $L1$ constraints:

$$\begin{aligned} W = \arg\min_W \frac{1}{2}||X - XW||_F^2 + \lambda||W||_F^2 \\ s.t. \quad diag(W) = 0 \end{aligned} \tag{3}$$

Empirical study [29] basically confirms that the non-negative constraint and L_1 norm on matrix W may not be essential (or have negative impact) on the performance. Particularly, EASE has a closed-form solution [29].

DLAE and EDLAE: The latest extension of EASE, the DLAE (Denoising linear autoencoder) [30] utilizes a drop-out induced the L_2 norm to replace the standard L_2 norm without zero diagonal constraints:

$$W = \arg\min_W \frac{1}{2}||X - XW||_F^2 + ||\Lambda^{1/2}W||_F^2 \tag{4}$$

where $\Lambda = \frac{p}{1-p} diagM(diag(X^TX))$ ($diagM(\cdot)$ denotes the diagonal matrix) and p is the dropout probability.

Another variant EDLAE would explicitly enforce the zero diagonal constraints:

$$\begin{aligned} W = \arg\min_W \frac{1}{2}||X - XW||_F^2 + ||\Lambda^{1/2}W||_F^2 \\ s.t.\ \ diag(W) = 0, \end{aligned} \tag{5}$$

Both DLAE and EDLAE have closed-form solutions [30].

2.1.1 Low-Rank Regression. There have been a number of interesting studies [18, 28, 30] on using low-rank regression to factorize the weight/similarity matrix W. The latest work [30] shows a variety of low-rank regression constraints which have been (or can be) used for this purpose:

$$\begin{aligned} &||X - XAB^T||_F^2 + \lambda(||A||_F^2 + ||B^T||_F^2) \\ &||X - XAB^T||_F^2 + \lambda||AB^T||_F^2 \\ &||X - XAB^T||_F^2 + ||(\Lambda + \lambda I)AB^T||_F^2 \end{aligned} \tag{6}$$

where $A_{n\times k}$, $B_{n\times k}$, and thus $rank(AB) \leq k$. The reduced-rank EDLAE [30] further enforces zero diagonal constraints for generalization purpose.

We note that interestingly, the reduced-rank regression solution naturally introduces a k-dimensional vector embedding for each user from XA ($m \times k$) , and a k-dimensional vector embedding for each item via B. This immediately leads to an important question: how such embedding differs from the traditional matrix factorization (MF) approaches which aim to explicitly decompose X into two latent factor matrices (for users and items). Note that in the past, the MF methods are more popular and widely used in industry, but the recent researches [8] seem to suggest an edge based on the regression-based (or linear autoencoder) approaches over the MF approaches. Before we formalize our question, let us take a quick review of MF approaches.

2.2 Matrix Factorization Approaches

Matrix factorization has been been widely studied for recommendation and is likely the most popular recommendation methods (particularly showing great success in the Netflix competition [20]). The basic formula to estimate the rating is

$$\hat{x}_{ui} = p_u \cdot q_i = q_i^T p_u, \tag{7}$$

where p_u and q_i are the corresponding k-dimensional latent vectors of user u and item i, respectively. Below, we review several well-known matrix factorization approaches for implicit settings; they differ on how to treat known vs missing interactions and regularization terms, among others.

ALS: The implicit Alternating Least Square (ALS) method [17] is basically a weighted matrix factorization (WRMF):

$$\arg\min_{P,Q} ||C \odot (X - PQ^T)||_F^2 + \lambda(||P||_F^2 + ||Q||_F^2), \tag{8}$$

where $P_{m\times k}$ records all the k-dimensional latent vectors for users and $Q_{n\times k}$ records all the item latent vectors, and λ regularize the squared Frobenius norm. C is the weight matrix (for binary data, the known score in X typically has α weight and the missing value has 1), and $\odot$ is the element-wise product. For the general weight matrix, there is no closed-form solution; the authors thus propose using alternating least square solver to optimize the objective function.

PureSVD: The Matrix factorization approach is not only closely related to SVD (singular value decomposition), it is actually inspired by it [1]. In the PureSVD approach, the interaction matrix X is factorized using SVD (due to Eckart-Young theorem) [7]:

$$\arg \min_{U_k, \Sigma_k, V_k} ||X - U_k \Sigma_k V_k^T||_F^2, \tag{9}$$

where U_k is a $m \times k$ orthonormal matrix, V_k is a $n \times k$ orthonormal matrix, and Σ_k is a $k \times k$ diagonal matrix containing the first k singular values. Thus the user factor matrix can be defined as $P = U_k \Sigma_k$ and the item factor matrix is $Q = V_k$.

SVD++: SVD++ [19] is another influential matrix factorization approach which also integrate the neighborhood factor. It nicely combines the formulas of factorization and neighborhood approaches with generalization. It targets only positive user-item ratings and typically works on explicit rating prediction.

2.3 The Problem

As we mentioned earlier, a few recent studies [8, 30] seem to indicate the regression based approach (or linear autoencoder approach) seem to have better performance than the popular matrix factorization approach, such as ALS [16]. However, if we look at the reduced rank regression approach, we observe its solution can be considered a special case of matrix factorization. Another interesting question is on the regularization hyperparameter, λ: ALS typically use a much smaller regularization penalty compared with the one used in the regression based approach, such as EASE and low-rank version. The latter's λ value is typically very large, in the range of thousands and even tens of thousands or [29, 30]. Note that both aim to regularize the squared Frobenius matrix norm. What results in such discrepancy?

Another interesting problem is about model complexity of these two approaches. The regression-based (linear auto-encoder) approach uses the similarity matrix W (which has $n \times n$ parameters), and when using low-rank regression, its parameters will be further reduced to $O(n \times k)$ where k is the reduced rank. The MF has both user and item matrices, and thus has $O((m+n)k)$ parameters. This seems to indicate the MF approach should be more flexible than the regression approaches as it tends to have much more parameters (due to number of users is typically much larger than the number of items). But is this the case?

In this study, our focus is not to experimentally compare these two types of recommendation approaches, but instead to have a better theoretical understanding their differences as well their connections. Thus, we hope to understand why and how if any approach maybe more advantageous than the other and along this, we will also investigate why the effective range of their regularization hyper-parameters are so different. We will also investigate how to learn high dimensional (hyper)parameters and apply it to help perform post-model exploration to learn "nearby" models.

3 THEORETICAL ANALYSIS

In this section, we will theoretically investigate the regression and matrix factorization models, and explore their underlying relationships, model complexity, and explain their discrepancy on regularization parameters.

3.1 Low-Rank Regression (LRR) Models

To facilitate our discussion, we consider the following basic low-rank regression models:

$$W = \arg \min_{rank(W) \le k} ||X - XW||_F^2 + ||\Gamma W||_F^2, \tag{10}$$

where Γ Matrix regularizes the squared Frobenius norm of W. (This can be considered as the generalized ridge regression, or multivariant Tikhonov regularization) [31]. For the basic case, $\Gamma^T\Gamma = \lambda I$, and $\Gamma^T\Gamma = \Lambda = \frac{p}{1-p} diagM(diag(X^TX))$ for DLAE (eq 4) [30]. Note that this regularization does not include the zero diagonal requirement for W. As we will show in Section 5, enforcing it only provides minor improvement and thus the basic model can well capture the essence of (low-rank) regression based recommendation.

To help derive the closed-form solution for above problem, let us represent it as a standard regression problem.

$$\overline{Y} = \begin{bmatrix} X \\ 0 \end{bmatrix} \quad \overline{X} = \begin{bmatrix} X \\ \Gamma \end{bmatrix}$$

Given this, the original problem can be rewritten as:

$$\min_{rank(W) \le k} ||\overline{Y} - \overline{X}W||_F^2 =$$
$$\min_{rank(W) \le k} ||\overline{Y} - \overline{X}W^*||_F^2 + ||\overline{X}W^* - \overline{X}W||_F^2,$$

where $W^* = \arg\min ||\overline{Y} - \overline{X}W^*||_F^2$. Basically, the initial loss $||\overline{Y} - \overline{X}W||_F^2$ is decomposed into two parts: $||\overline{Y} - \overline{X}W^*||_F^2$ (no rank constraint), and $||\overline{X}W^* - \overline{X}W||_F^2$. Note this holds since the vector ($\overline{Y} - \overline{X}W^*$) is orthogonal to $\overline{X}W^* - \overline{X}W = \overline{X}(W^* - W)$ (The optimality of Ordinary Least-Square estimation [31]).

Now, the original problem can be broken into two subproblems:

(Subproblem 1:) item-weighted Tikhonov regularization:

$$\begin{aligned} W^* &= \arg\min_W ||\overline{Y} - \overline{X}W^*||_F^2 = \arg\min_W ||X - XW||_F^2 + ||\Gamma W||_F^2 \\ &= (\overline{X}^T\overline{X})^{-1}\overline{X}^T\overline{Y} \\ &= (X^TX + \Gamma^T\Gamma)^{-1}X^TX \end{aligned}$$

(Subproblem 2:) low-rank matrix approximation:

$$\begin{aligned} \hat{W} &= \arg \min_{rank(W) \le k} ||\overline{X}W^* - \overline{X}W||_F^2 \\ &= \arg \min_{rank(W) \le k} ||XW^* - XW||_F^2 + ||\Gamma(W^* - W)||_F^2 \end{aligned}$$

Let $\overline{Y}^* = \overline{X}W^*$, and based on the well-known *Eckart-Young* theorem [11], we have the best rank k approximation of $\overline{Y}^*$ in Frobenius norm is best represented by SVD. Let $\overline{Y}^* = P\Sigma Q^T$ (P, Q are orthogonal matrices and Σ is the singular value diagonal matrix, and then the best rank k approximation of $\overline{Y}^*$, denoted as $\overline{Y}^*(k)$ is

$$\overline{Y}^*(k) = P_k \Sigma_k Q_k^T, \tag{11}$$

where M_k takes the first k rows of matrix M. We also have the following equation:

$$P\Sigma Q^T(Q_k Q_k^T) = P_k \Sigma_k Q_k^T$$

Given this, we notice that

$$\overline{Y}^*(k) = P_k\Sigma_k Q_k^T = P\Sigma Q^T(Q_kQ_k^T)$$
$$= \overline{X}W^*(Q_kQ_k^T) = \overline{X}W$$

Thus, we have

$$\widehat{W} = W^*(Q_kQ_k^T) = (X^TX + \Gamma^T\Gamma)^{-1}X^TX(Q_kQ_k^T),$$

and the complete estimator for XD (interaction/rating inference) is written as:

$$\boxed{\widehat{W} = (X^TX + \Gamma^T\Gamma)^{-1}X^TX(Q_kQ_k^T)} \tag{12}$$

Next, let us further simplify it using SVD which can better reveal its "geometric" insight.

3.1.1 Eigen Simplification. First, let the SVD of X as

$$X = U\Sigma V$$

When $\Gamma = \Lambda^{1/2}V^T$ where Λ is a diagonal matrix, we can observe:

Proposition 3.1.

$$Q_k = V_k$$

$$\overline{Y}^* = \overline{X}W^* = \overline{X}V(\Sigma^2 + \Lambda)^{-1}\Sigma^2V^T$$

Then, from

$$(\overline{Y}^*)^T\overline{Y}^* = V\Sigma^{-2}(\Sigma^2 + \Lambda)V^T\overline{X}^T\overline{X}V(\Sigma^2 + \Lambda)^{-1}\Sigma^2V^T$$
$$= V(\Sigma^2 + \Lambda)V^T$$

Then we have the following:

$$\widehat{W} = (X^TX + \Gamma^T\Gamma)^{-1}X^TX(Q_kQ_k^T)$$
$$= V(\Sigma^2 + \Lambda)^{-1}\Sigma^2V^T(V_kV_k^T)$$
$$= Vdiag(\frac{\sigma_1^2}{\sigma_1^2+\lambda_1}, \ldots, \frac{\sigma_k^2}{\sigma_n^2+\lambda_k})V^T(V_kV_k^T)$$

Thus, we have the following closed-form solution:

$$\boxed{\widehat{W} = V_kdiag(\frac{\sigma_1^2}{\sigma_1^2+\lambda_1}, \ldots, \frac{\sigma_k^2}{\sigma_k^2+\lambda_k})V_k^T} \tag{13}$$

Now, if $\lambda_i = \lambda$, we have:

$$\boxed{\widehat{W} = V_kdiag(\frac{\sigma_1^2}{\sigma_1^2+\lambda}, \cdots, \frac{\sigma_k^2}{\sigma_k^2+\lambda})V_k^T} \tag{14}$$

Note that this special case $\Gamma^T\Gamma = \lambda I$ has indeed been used in [27] for implicit recommendation. However, the authors do not realize that it actually has a closed-form solution.

We also note that using the matrix factorization perspective, we obtain the user (P) and item (Q) matrices as:

$$\boxed{\begin{aligned} P &= XV_kdiag(\frac{\sigma_1^2}{\sigma_1^2+\lambda}, \ldots, \frac{\sigma_k^2}{\sigma_k^2+\lambda}) = U_kdiag(\frac{\sigma_1}{1+\lambda/\sigma_1^2}, \ldots, \frac{\sigma_k}{1+\lambda/\sigma_k^2}) \\ Q &= V_k \end{aligned}} \tag{15}$$

3.2 Matrix Factorization

To study the relationship between the regression approach and matrix factorization, we consider the basic regularized SVD [34]:

$$\arg\min_{P,Q} ||X - PQ^T||_F^2 + \lambda'(||P||_F^2 + ||Q||_F^2), \tag{16}$$

The solution for this type problem is typically based on Alternating Least Square, but authors in [34] have found a closed-form solution.

Let $X = U\Sigma V^T$, and then let

$$\boxed{\begin{aligned} P &= U_kdiag(\sigma_1 - \lambda', \cdots, \sigma_k - \lambda') \\ &= U_kdiag(\sigma_1(1 - \lambda'/\sigma_1), \cdots, \sigma_k(1 - \lambda'/\sigma_k)) \\ Q &= V_k \end{aligned}} \tag{17}$$

Before we further analyze the relationship between them, we ask the following interesting question: Can matrix factorization be represented as a linear encoder? In other words, we seek if there is an W such that $XW = PQ$ (defined by matrix factorization). Let the Moore-Penrose inverse $X^+ = V\Sigma^{-1}U^T$, then we have $\widehat{W} = X^+PQ$,

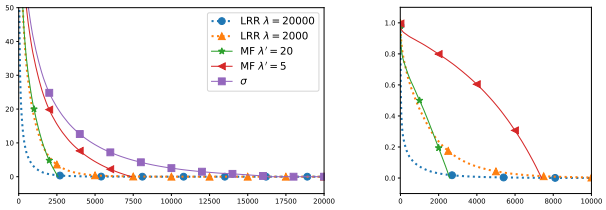
$$\boxed{\widehat{W} = V_kdiag(1 - \lambda'/\sigma_1, \cdots, 1 - \lambda'/\sigma_k)V_k^T} \tag{18}$$

3.3 Model Comparison and Analysis

When $\lambda = \lambda' = 0$ (no regularization), then both approaches (using the matrix factorization) correspond to the standard SVD decomposition, where $P = U_k\Sigma_k$ and $Q = V_k^T$. Further, both approaches will also have $\widehat{W} = V_kV_k^T$. Then, let us consider $X\widehat{W} = XV_kV_k^T$, which is indeed our standard principal component analysis (PCA), where V_k serves as a linear map which transforms each row vector x_i^T to the new coordinate under the principal component basis. Clearly, when $\lambda = \lambda' \neq 0$, both models start to diverge and behave differently, and results in their difference in terms of regularization penalties, model complexities and eventually model accuracy.

The Geometric Transformation From the matrix factorization perspective, the Formulas 15 and 17 essentially tells us that these two different approaches both scale-down the singular values of the user-item interaction matrix (binary) with slightly different manner: $\frac{1}{1+\lambda/\sigma_i^2}$ for low-rank regression and $1 - \lambda/\sigma_i$ for matrix factorization. Figure 1a illustrates the compressed singular value for *ML-20M* dataset with LRR corresponds to the low-rank regression and MF corresponds to the matrix factorization. Figure 1b illustrates the compression ratios, which also has a direct geometric explanation: if we consider both approaches as the linear auto-encoder (or regression), as being described in Formulas 14 and 18, then the comprehension ratios directly scale down the coordinates (XV_k) for the principal component basis in V_k^T.

The Effective Range of λ and λ' and Latent Dimension k For the regression based approach, we note that when $\lambda < \sigma_i^2$, then there is relatively small effect to scale down σ_i (or any other singular values larger than σ_i). Given this, λ tends to be quite big, and it has close to binary effect: let $\sigma_i^2 \geq \lambda_i > \sigma_{i+1}^2$, then any $\sigma_j < \sigma_i$ has small effect, and for any $\sigma_j > \sigma_i$, then, the reduction will become more significant (proportionally). Typically, λ is within the range of the first few singular values (in other words, i is very small, and λ is quite large).



(a) scaling-down singular values (b) compression ratios

Figure 1: Geometric Transformation for ML-20M.

For the matrix factorization, assuming we consider the k dimensional latent factors, then we note that $\sigma_k > \lambda'$, which effectively limit the range of λ'. Furthermore, we notice that since each singular value is reduced by the same amount λ, which makes the latent dimensions with smaller singular values are even less relevant than their original value. Thus, this leads to the matrix factorization typically has a relatively small number of dimensions to use (typically less than 1000).

For the low-rank regression, as the singular value reduced, its proportion ($\frac{1}{1+\lambda/\sigma_i^2}$) will also be reduced down, the same as the matrix factorization. However, unlike the regularized matrix factorization approach whose absolute reduction may vary:

$$\Delta_i = \sigma_i - \frac{\sigma_i}{1+\lambda/\sigma_i^2} = \frac{\lambda}{\sigma_i + \lambda/\sigma_i} \tag{19}$$

Interesting, when σ_i is both very large or very small, its absolute reduction are fairly small, but it may reduce those in between more. Thus, this effectively enables the low-rank approaches to use more latent dimensions, thus larger k (typically larger than 1000).

Now, an interesting *conjecture* of an optimal set of λ_i in Equation 13, is that they should help (relatively) scale-down those large principal components and help (relatively) scale-up those smaller principal components for better performance. However, how can we search the optimal set of λ_i for a large number of k? We will introduce a learning algorithm for this purpose in Subsection 4, and then utilize that to study the conjecture (through an experimental study in Section 5). This help provide a better understanding of the adjustment of singular values.

Model Complexity For both types of models, we observe that the gram matrix X^TX serves as the sufficient statistics. This indeed suggest that the complexities of both models (number of effective parameters [14]) will have no higher than X^TX. In the past, we typically consider P and Q (together) are defined as the model parameters for matrix factorization. Thus, the common assumption is that MF has model complexities ($O(mk+nk)$). However, the above analysis based on linear autoencoder/regression perspective, shows that both models essentially only have V_k (together with the scaled principal components). (See Equations 14 and 18) for W estimation. Thus, their model complexity are both $O(nk)$ (but with different k for different models).

Now relating to the aforementioned discussion on the latent dimension factor, we can immediately observe the model complexity of the basic low-rank regression actually have higher complexity than its corresponding matrix factorization model (as the former can allow larger k than the latter).

Note that for the real-world recommendation models, the matrix factorization will utilize the weight matrix C (equation 8) to increase model complexity (which does not have closed-form solution [17]). However, due to the alternating least square solution, its number of effective parameters will remain at $O(nk)$. Thus, it is more restricted and less flexible than the regression based approaches.

Algorithm 1 Hyperparameter Search for Formula 14

INPUT: Hyperparameter candidate lists: λ_l, k_l, user-item binary matrix X.
OUTPUT: Model performance for all hyperparameter λ_l, k_l combinations.

```
X^T X = V Σ^T Σ V^T  (Eigen Decomposition)
for all λ ∈ λ list do
    Δ := (Σ^T Σ + λI)^{-1} Σ^T Σ = diag(d_1^2/(d_1^2+λ), ..., d_n^2/(d_n^2+λ))
    for all k ∈ k list do
        Δ_k ← first k columns and rows of Δ
        V_k ← first k columns of V
        W_k ← V_k Δ_k V_k^T
        evaluate(W_k) based on nDCG and/or Recall@K
    end for
end for
```

4 PARAMETER SEARCH AND NEARBY MODELS

In this section, we first aim to identify a set of *optimal* (hyper)parameters $\{\lambda_i : 1 \le i \le k\}$ for the closed-form solution 13. Clearly, when the parameter number is small, we can deploy a grid search algorithm as illustrated in Algorithm 1 for search k and λ for the closed-form in Equation 14. However, for a large number of (hyper)parameters, we have to resort to new approaches (Subsection 4.1). Furthermore, once the parameter searching algorithm is available, we consider to utilize it for searching *nearby models* for an existing model (Subsection 4.2). As we mentioned before, this can be considered as a post-model fitting exploration in statistical learning [12].

4.1 Parameter Search

In this subsection, we assume the closed-form solution in Equation 14 ($\widehat{W} = V_k diag(\frac{\sigma_1^2}{\sigma_1^2+\lambda}, \ldots, \frac{\sigma_n^2}{\sigma_k^2+\lambda}) V_k^T$) has optimized hyperparameters λ and k through the grid search algorithm Algorithm 1. Our question is how to identify optimized parameter $\lambda_1, \cdots \lambda_k$ in Equation 13: $\widehat{W} = V_k diag(\frac{1}{1+\lambda_1/\sigma_1^2}, \ldots, \frac{1}{1+\lambda_n/\sigma_n^2}) V_k^T$.

The challenge is that the dimension (rank) k is fairly large and the typical (hyper)parameter search cannot work in such high dimensional space [3]. Also, as we have the closed-form, it does not make sense to utilize the (original) criterion such as Equation 10 for optimization. Ideally, we would like to evaluate the accuracy of any parameter setting (such as in Algorithm 1) based on nDCG or AUC [1]. Clearly, for this high dimensional continuous space, this is too expensive. To deal with this problem, we consider to utilize the *BPR* loss function which can be considered as a continuous analogy

of AUC [25], and parameterize λ_i with a search space centered around the optimal λ discovered by Algorithm 1:

$$\lambda_i(\alpha_i) = \lambda + c \times tanh(\alpha_i),$$

where c is the search range, which is typically a fraction of λ and α_i is the parameter to be tuned in order to find the right λ_i. Note that this method effectively provides a bounded search space $(\lambda - c, \lambda + c)$ for each λ_i.

Given this, the new objective function based on BPR is:

$$\mathcal{L} = \sum_{u,i \in X_u^+, j \in X_u^-} -\log(\delta(tx_u(W(\alpha_1, \cdots, \alpha_k)_{*i} - W(\alpha_1, \cdots, \alpha_k)_{*j}))$$

where $W(\alpha_1, \cdots, \alpha_k) = V_k diag(\frac{\sigma_1^2}{\sigma_1^2+\lambda_1(\alpha_i)}, \ldots, \frac{\sigma_n^2}{\sigma_n^2+\lambda_n(\alpha_n)}) V_k^T$, and $W(\alpha_1, \cdots, \alpha_k)_{*i}$ is the i-th column of W matrix, x_u is the $u-th$ row of matrix X and t is a scaling constant. Here t and c are hyper-parameters for this learning procedure.

Note that this is a non-linear loss function for a linear model and the entire loss function can be directly implemented as a simple neural network, and ADAM (or other gradient descent) optimization procedure can be utilized for optimization. We can also add other optimization such as dropout and explicitly enforcing the zero diagonal of W [29].

4.2 Nearby Linear Models

In this subsection, we consider how to further leverage the new learning procedure for other linear models to help identify the (hyper)parameters. Inspired by the recent efforts of post-model fitting exploration [12], we consider to augment the existing learned W from any existing models (or adding on top of the aforementioned closed-form solution) with two types of parameters:

$$\begin{aligned} W_{HT} &= diagM(H) \cdot W \cdot diagM(T) \\ W_S &= S \odot \widehat{W} \odot (\widehat{W} \geq t) \end{aligned} \tag{20}$$

where $H = (\delta(h_1), \cdots \delta(h_n))$ and $T = (\delta(t_1), \cdots, \delta(t_n))$ are the *head* and *tail* vectors with values between 0 and 1 (implemented through sigmoid function). We also refer to the diagonal matrices $diagM(H)$ and $diagM(T)$ as the head and tail matrices. Basically, these diagonal matrices $diagM(H)$ and $diagM(T)$ help re-scale the row and column vectors in W. Furthermore, $S = (\delta(s_{ij}))$ is a matrix with values between 0 and 1 (implemented through sigmoid function). Finally, $\widehat{W} \geq t$ is a boolean matrix for sparsification: when $\widehat{W}_{ij} > t$, its element is 1, otherwise, it it zero. Thus, this augmented model basically consider to sparsify the learned similar matrix W and re-scale its remaining weights. Note that both W_{HT} and W_S can be considered as the *nearby* models for the existing models with learned $\widehat{W}$. Note that studying these models can also help us understand how close these available learner models are with respect to their limit for recommendation tasks. Since the optimization is more close to the "true" objective function, it helps us to squeeze out any potential better models near these existing models. In Section 5, we will experimentally validate if there is any space for improvement based on those simple augmented learning models.

5 EXPERIMENTAL RESULTS

In this section, we experimentally study the basic linear models as well as the (hyper)parameter search algorithms and its applications

ML-20M	EASE λ=400	LRR k =2K, λ = 10K	MF k = 1K, λ = 50	WMF(ALS) k = 100, C = 10, λ = 1e2
Recall@20	**0.39111**	0.37635	0.36358	0.36327
Recall@50	**0.52083**	0.51144	0.50069	0.50232
nDCG@100	**0.42006**	0.40760	0.39187	0.39314

Table 1: ML-20M: Basic Model Evaluation

Netflix	EASE λ= 1000	LRR k =3K, λ = 40K	MF k=1K, λ=100	WMF(ALS) k=100, C = 5, λ=1e2
Recall@20	**0.36064**	0.3478	0.33117	0.3213
Recall@50	**0.44419**	0.4314	0.41719	0.40629
nDCG@100	**0.39225**	0.38018	0.36462	0.35548

Table 2: Netflix: Basic Model Evaluation

to the nearby models. Note that our goal here is not to demonstrate the superiority of these basic/closed-form solutions, but to show they can fare well against the state-of-the-art linear models. This can thus help validate using these basic models to study these advanced linear models [17, 29, 30]. Specifically, we aim to answer:

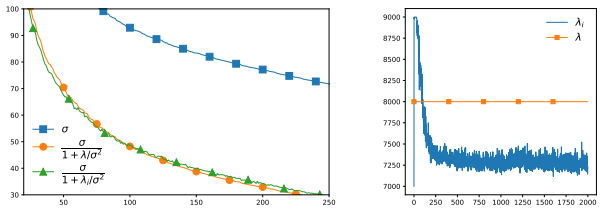
- (Question 1) How do the basic regression and matrix factorization based models (and their closed-form solutions) compare against the state-of-the-art linear models? Also we hope to compare the two basic models (using their closed-form solutions) to help provide evidence if the matrix factorization approaches have inherently disadvantage for the implicit recommendation task.
- (Question 2) How can the learning algorithm to help search the optimal parameter for the closed-form solution of Equation 13 as well as its augmented models (adding both head and tail matrices)? How does the (augmented) closed-form solution perform against the state-of-the-art methods? We are also interested in understanding how the learned $\{\lambda_i\}$ parameters look like with respect to the constant λ.
- (Question 3) How does the nearby models based on the head and tail matrices W_{HT} and sparsification W_S introduced in Subsection 4.2 perform? Can any existing state-of-the-art linear models be boosted by searching through the augmented nearby models?

Experimental Setup: We use three commonly used datasets for recommendation studies: MovieLens 20 Million (ML-20M) [13], Netflix Prize (Netflix) [2], and the Million Song Data (MSD)[4]. The characteristics of first two datasets are in the bottom of Table 3. The characteristics of the third dataset and its results is in Appendix.

For the state-of-the-art recommendation algorithms, we consider the following: ALS [17] for matrix factorization approaches, SLIM [24], EASE [29], and EDLAE [30] for regression models, CDAE [32] and MultiVAE [23] for deep learning models. For most of the experiment settings, we follow [23, 29, 30] for the *strong generalization* by splitting the users into training, validation and tests group. Also following [23, 29, 30], we report the results using metrics *Recall*@20, *Recall*@50 and *nDCG*@100.

Finally, note that our code are openly available (see Appendix).

Basic Model Evaluation: In this experiment, we aim to evaluate the the closed-form (Formulas 14, referred to as *LRR* and 18, referred to as *MF*) of the two basic models (Equations 10 and 16). We compare them against the state-of-the-art regression model EASE [29] and ALS [17]. Since this is mainly for evaluating their prediction capacity (not on how they perform on the real world environment), here we utilize the leave-one-out method to evaluate these models. Note that this actually provides an advantage



(a) optimal adjusted singular value

(b) optimized (hyper)parameters λ_i

Figure 2: Optimization Results for Formula 13

to the matrix factorization approaches as they prefer to learn the embeddings (latent factors) before its prediction.

Tables 1 and 2 show the results for these four linear models on the ml-20m and netflix datasets, respectively. We perform a grid search for each of these models (the grid search results are reported in Appendix), and report their better settings (and results) in these tables. From these results, we observe: (1) Both basic models *LRR* and *MF* have very comparable performances against their advanced version. Note that *LRR* does not have the zero diagonal constraint and use reduced rank regression compared with EASE; and *MF* does not have the weighted matrix in ALS [17]. This helps confirm the base models can indeed capture the essence of the advanced models and thus our theoretical analysis on these models can help (partially) reflect the behaviors from advanced models. (2) Both regression models are consistently and significantly better than the matrix factorization based approaches. This helps further consolidate the observations from other studies [8] that the regression methods have the advantage over the matrix factorization methods.

Optimizing Closed-Form Solutions: In this and next experiment, we will follow the strong generalization setting by splitting the users into training, validation and testing groups. The top section of Table 3 shows the experimental results of using the closed-form solution (Formula 13). Here, (1) $LRR(closed-form)$ is the starting point for λ being constant; (2) $LRR+\lambda_i$ utilizes the BPR learning algorithm in Subsection 4 to search the hyperparameter space; (3) $LRR+\lambda_i+HT$ uses $diagM(H)\cdot W\cdot diagM(T)$ (as the targeted similarity matrix), where W is defined in Formula 13 (here the optimization will simultaneously search hyperparameters $\{\lambda_i\}$ and head (H), tail (T) vectors; (4) finally, $LRR+\lambda_i+HT+RMD$ further enforces the zero diagonal constraints. We also add dropout (with dropout rate 0.5) for the model training for models $(2-4)$.

We observe the variants of the closed-form solutions are comparable against the state-of-the-art linear models and deep learning models. For instance, on ML-20M, $LRR+\lambda_i+HT+RMD$ reports 0.522 *Recall*@50, is among the best for the existing linear models (without additional boosting from the augmented nearby models).

Finally, Figure 2 illustrates the parameter search results for Formula 13 from the learning algorithm. Specifically, Figure 2 (a) shows how the singular value are adjusted vs the compressed singular value for a constant $\lambda = 8000$ (Formula 14). We provide $c = 1000$ to allow each individual λ_i search between 7000 to 9000. Figure 2 (b) shows the search results for the parameters λ_i. As we conjectured, we can see that the initial λ_i is quite large which leads to smaller singular values compared with adjusted singular value from Formula 14. Then the parameters λ_i reduces which make the smaller singular values reduced less. This can help more (smaller) singular values to have better presence in the final prediction.

Nearby Models In this experiment, we augment the latest regression models EDLAE (full rank and reduced rank) [30] with additional parameters and apply the parameter learning algorithm to optimize the parameters: (1) *EDLAE* is the original reduced rank regression with rank $k = 1000$; (2) $EDLAE+HT$ corresponds to the augmented model with head and tail matrices, W_{HT} from Formula 20; (3) *EDLAE Full Rank* is the original full rank regression; (4) $EDLAE\ Full\ Rank+HT$ applies the head and tail matrices on the learned similarity matrix from *EDLAE Full Rank*; (5) $EDLAE\ Full\ Rank{+}Sparsification$ applies the W_S from Formula 20, which sparsifies the similarity matrix of *EDLAE Full Rank* with additional parameters in matrix S to further adjust those remaining entries in the similarity matrix.

The experimental results on ML-20M and Netflix of these augmented (nearby) models are listed in the middle section in Table 3. We can see that on the ML-20M dataset, the *Recall*@50 has close to 1% boost while other metrics has small improvement. This indeed demonstrates the nearby models may provide non-trivial improvement over the existing models. On the Netflix dataset, the nearby models only have minor changes and indicates the originally learned model may already achieve the local optimum.

6 CONCLUSION AND DISCUSSION

In this work, we provide a thorough investigation into the relationship between arguably two of the most important recommendation approaches: the neighborhood regression approach and the matrix factorization approach. We show how they inherently connect with each other as well as how they differ from one another. However, our study mainly focuses on the implicit setting: here the goal is not to recover the original ratings (like in the explicit setting), but to recover a "likelihood" (or a preference) of the interaction. Thus, the absolute value/rating is not of interests. In fact, for most of the linear regression models, the predicted value can be very small (more close to zero than one). What matters here is the relative rank of the predicted scores. Thus it helps to use more latent factors to express the richness of user-item interactions. This can be rather different from the rating recovery, which requires the original singular values to be preserved. Especially, the current approaches of explicit matrix factorization which often consider only the positive values and thus the methodology developed in this work cannot be immediately applied in this setting. Indeed, Koren and Bell in [21] has analyzed the relationship between neighborhood and factorization models under explicit settings. It remains to be seen whether the insights gained here can be applied to the explicit setting.

Also, we would like to point out that this is the first work to investigate the nearby linear models. We consider two basic models which utilize limited additional parameters to help explore the additional models. An interesting question is whether we can explore more nearby models.

Finally, we note that the theoretical models need eigen decomposition which makes them infeasible for the real-world datasets with millions of items. But our purpose here is to leverage such models to help understand the tradeoffs and limitations of linear models, not to replace them. We hope what being revealed in this work can help design better linear and nonlinear models for recommendation.

Model		ML-20M			Netflix		
		Recall@20	Recall@50	nDCG@100	Recall@20	Recall@50	nDCG@100
LRR (closed-form)		0.376	0.513	0.406	0.347	0.432	0.380
LRR + λ_i		0.380	0.515	0.410	0.348	0.433	0.381
LRR + λ_i + HT		0.386	0.520	0.418	0.351	0.435	0.384
LRR + λ_i + HT + RMD		0.386	0.522	0.418	0.351	0.435	0.384
EDLAE		0.389	0.521	0.422	0.362	0.446	0.393
EDLAE + HT		0.394	0.527	0.424	0.361	0.446	0.393
EDLAE Full Rank		0.393	0.523	0.424	0.364	0.449	0.397
EDLAE Full Rank + HT		0.395	0.527	0.426	0.364	0.449	0.396
EDLAE Full Rank + Sparsification		0.394	0.526	0.423	0.365	0.450	0.397
SLIM		0.370	0.495	0.401	0.347	0.428	0.379
ALS/WMF		0.363	0.502	0.393	0.321	0.406	0.355
EASE		0.391	0.521	0.420	0.360	0.444	0.392
CDAE		0.391	0.523	0.418	0.343	0.428	0.376
MULT-DAE		0.387	0.524	0.419	0.344	0.438	0.380
MULT-VAE		0.395	0.537	0.426	0.351	0.444	0.386
dataset statistics	# items	20108			17769		
	# users	136677			463435		
	# interactions	10 millions			57 millions		

Table 3: The performance comparison between different models. λ_i:learned (hyper)parameters; HT: augmented models with head and tail parameter matrix; RMD: with removing the diagonal matrix (enforcing zero diagonal). For more details of the experimental set-up and model, please refer to the appendix.

REFERENCES

[1] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer, 1st edition, 2016.
[2] James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, and Domonkos Tikk. Kdd cup and workshop 2007. 2007.
[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, February 2012.
[4] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.
[5] Evangelia Christakopoulou and George Karypis. HOSLIM: higher-order sparse linear method for top-n recommender systems. In *PAKDD*, 2014.
[6] Evangelia Christakopoulou and George Karypis. Hoslim: Higher-order sparse linear method for top-n recommender systems. In *Advances in Knowledge Discovery and Data Mining*, 2014.
[7] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys'10*, 2010.
[8] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Trans. Inf. Syst.*, 39(2), January 2021.
[9] Maurizio Ferrari Dacrema, P. Cremonesi, and D. Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *RecSys'19*, 2019.
[10] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 2004.
[11] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.
[12] Leying Guan and Robert Tibshirani. Post model-fitting exploration via a "next-door" analysis, 2018.
[13] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 2015.
[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.
[15] Xiangnan He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua. Neural collaborative filtering. In *WWW'17*, 2017.
[16] Binbin Hu, C. Shi, W. X. Zhao, and P. S. Yu. Leveraging meta-path based context for top- n recommendation with a neural co-attention model. In *KDD'18*, 2018.
[17] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM'08*, 2008.
[18] Santosh Kabbur, Xia Ning, and George Karypis. Fism: Factored item similarity models for top-n recommender systems. KDD '13, 2013.
[19] Yehuda Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *KDD'08*, 2008.
[20] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.
[21] Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 77–118. Springer, 2015.
[22] Walid Krichene and Steffen Rendle. *On Sampled Metrics for Item Recommendation*, page 1748–1757. 2020.
[23] Dawen Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. Variational autoencoders for collaborative filtering. In *WWW'18*, 2018.
[24] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. ICDM '11, 2011.
[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. UAI '09, 2009.
[26] Steffen Rendle, Li Zhang, and Yehuda Koren. On the difficulty of evaluating baselines: A study on recommender systems. 2019.
[27] Suvash Sedhain, Hung Bui, Jaya Kawale, Nikos Vlassis, Branislav Kveton, Aditya Krishna Menon, Trung Bui, and Scott Sanner. Practical linear models for large-scale one-class collaborative filtering. IJCAI'16, 2016.
[28] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Darius Braziunas. On the effectiveness of linear models for one-class collaborative filtering. AAAI'16, 2016.
[29] Harald Steck. Embarrassingly shallow autoencoders for sparse data. *WWW'19*, 2019.
[30] Harald Steck. Autoencoders that don't overfit towards the identity. In *NIPS*, 2020.
[31] Wessel N. van Wieringen. Lecture notes on ridge regression, 2020.
[32] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. WSDM '16, 2016.
[33] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 2019.
[34] Shuai Zheng, Chris Ding, and Feiping Nie. Regularized singular value decomposition and application to recommender system, 2018.
[35] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005.

7 ACKNOWLEDGMENTS

The research was partially supported by a sponsorship research agreement between Kent State University and iLambda, Inc.

A REPRODUCIBILITY

Generally, We follow the (strong generalization) experiment set-up in [23, 30] and also the pre-processing of the three public available datasets, MovieLens 20 Million (ML-20M) [13], Netflix Prize (Netflix) [2], and the Million Song Data (MSD)[4].

A.1 Experimental Set-up for Table 1 and 2

For the experiment of table 1 and 2, we utilize the strong generalization protocol for EASE [29] and LRR methods. For Matrix Factorization based methods (MF and WMF/ALS), they are trained for with data (except the items to be evaluated in validation and test sets) Note that this actually provides an advantage to the matrix factorization approaches as they prefer to learn the embeddings (latent factors) before its prediction. The experiment results present in table 1 and 2 are obtained by parameter grid search over the validation set according to *nDCG*@100, the same as [23]. The searching results are listed as following : table 4, table 5, table 6 and table 7.

		λ						
		0	10	**50**	100	200	500	1000
k	128	0.29874	0.30951	0.36488	0.37826	0.30121	0.1901	0.1901
	256	0.22911	0.25104	0.37504	0.37826	0.30121	0.1901	0.1901
	512	0.1546	0.19782	0.39682	0.37826	0.30121	0.1901	0.1901
	1000	0.09177	0.18242	**0.39893**	0.37826	0.30121	0.1901	0.1901
	1500	0.06089	0.20776	0.39893	0.37826	0.30121	0.1901	0.1901

Table 4: ML-20M, MF, parameter search

		λ						
		0	10	50	**100**	200	500	1000
k	128	0.31297	0.31542	0.32607	0.34103	0.34132	0.25831	0.17414
	256	0.25036	0.25536	0.28659	0.33521	0.34132	0.25831	0.17414
	512	0.17485	0.18314	0.26081	0.35157	0.34132	0.25831	0.17414
	1000	0.12036	0.13766	0.28868	**0.36414**	0.34132	0.25831	0.17414
	1500	0.09147	0.12449	0.32103	0.36414	0.34132	0.25831	0.17414

Table 5: Netflix, MF, parameter search

		λ						
		8000	9000	**10000**	11000	12000	13000	14000
k	1000	0.41063	0.41273	0.41432	0.41476	0.41515	0.41513	0.41478
	2000	0.41332	0.41469	**0.41533**	0.41509	0.41499	0.41455	0.41394
	3000	0.41282	0.41397	0.41473	0.4146	0.41452	0.41413	0.41347

Table 6: ML-20M, LRR, parameter search

		λ					
		10000	20000	30000	**40000**	50000	60000
k	2000	0.33632	0.37139	0.37856	0.37942	0.37828	0.37644
	3000	0.34905	0.37441	0.37934	**0.37949**	0.37807	0.37617
	4000	0.35184	0.37468	0.37919	0.37931	0.37786	0.37601

Table 7: Netflix, LRR, parameter search

A.2 Experimental Set-up for Table 3

In table 3, for LRR (closed-form) model (described in equation 14). For ML-20M dataset, we set $k = 2000$, $\lambda = 8000$, $c = 1000$ (used to control range of weighted λ_i). For Netflix dataset the $\lambda = 8000$, $\lambda = 40000$, $c = 5000$. Noting that these hyper-parameters are not set as optimal ones (described in table 6, table 7), which won't affect our claims. For EDLAE (including full rank) model, we obtain the similarity matrix by running the code from [30]. For WMF/ALS model and EASE model, we set the hyper-parameters as table 1 and table 2. Other models' results are obtained form [23], [29] and [30].

For fast training augmented model, we sample part of training data. Generally, it takes 2.5 minutes per 100 batch (batch size is 2048) for training.

A.3 MSD Dataset Results

The table 8 shows our experiment results carried out on the MSD dataset. Baseline models' results are obtained form [23], [29] and [30].

		MSD		
		Recall@20	Recall@50	nDCG@100
LRR		0.24769	0.33509	0.30127
LRR + λ_i		0.25083	0.33902	0.30372
EDLAE		0.26391	0.35465	0.31951
EDLAE Full Rank		0.33408	0.42948	0.39151
EDLAE Full Rank+HT		0.33423	0.43134	0.38851
SLIM		did not finished in [24]		
WMF		0.211	0.312	0.257
EASE		0.333	0.428	0.389
CDAE		0.188	0.283	0.237
MULT-DAE		0.266	0.363	0.313
MULT-VAE		0.266	0.364	0.316
dataset statistics	# items	41140		
	# users	571355		
	# interactions	34 millions		

Table 8: The performance comparison between models on MSD dataset.