Embarrassingly Shallow Autoencoders for Sparse Data

Harald Steck
Netflix
Los Gatos, California
hsteck@netflix.com

ABSTRACT

Combining simple elements from the literature, we define a linear model that is geared toward sparse data, in particular implicit feedback data for recommender systems. We show that its training objective has a closed-form solution, and discuss the resulting conceptual insights. Surprisingly, this simple model achieves better ranking accuracy than various state-of-the-art collaborative-filtering approaches, including deep non-linear models, on most of the publicly available data-sets used in our experiments.

KEYWORDS

Recommender System; Collaborative Filtering; Autoencoder; Neighborhood Approach; Linear Regression; Closed-Form Solution

1 INTRODUCTION

Many recent improvements in collaborative filtering can be attributed to deep learning approaches, e.g., [6, 10–12, 17, 25, 30, 31]. Unlike in areas like computer vision, however, it was found that a *small* number of hidden layers achieved the best recommendation accuracy. In this paper, we take this to the extreme, and define a linear model *without* a hidden layer (see Figure 1). The (binary) input vector indicates which items a user has interacted with, and the model's objective (in its output layer) is to predict the best items to recommend to the user. This is done by reproducing the input as its output, as is typical for *auto*encoders.[1] We hence named it Embarrassingly Shallow AutoEncoder (in Reverse order: EASE^R).

This paper is organized as follows: we define EASE^R in the next section, using simple elements from the literature. In Section 3.1, we derive the closed-form solution of its convex training objective. This has several implications: (1) it reveals that the neighborhood-based approaches used in collaborative filtering are based on conceptually incorrect item-item similarity-matrices, while EASE^R may be considered a principled neighborhood model, see Sections 3.2 and 4.3;

---

[1]Note, however, that the proposed model does not follow the typical architecture of autoencoders, being comprised of an encoder and a decoder: one may introduce an implicit hidden layer, however, by a (full-rank) decomposition of the learned weight-matrix of this model.
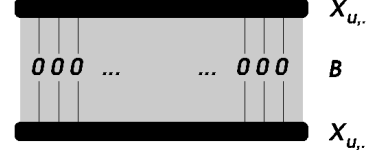

Figure 1: The self-similarity of each item is constrained to zero between the input and output layers.

(2) the code for training EASE^R is comprised of only a few lines, see Section 3.3 and Algorithm 1; (3) if the model fits into memory, the wall-clock time for training EASE^R can be several orders of magnitude less than for training a SLIM [20] model (see Section 3.4), which is the most similar model to EASE^R. Apart from that, we surprisingly found that EASE^R achieved competitive ranking accuracy, and even outperformed various deep, non-linear, or probabilistic models as well as neighborhood-based approaches on most of the publicly available data-sets used in our experiments (see Section 5).

2 MODEL DEFINITION

Like in many recommender papers that use implicit feedback data, we assume that the data are given in terms of a sparse (typically binary[2]) matrix $X \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$, regarding the sets of users $\mathcal{U}$ and items $\mathcal{I}$, where $|\cdot|$ denotes the size of a set. A positive value (typically 1) in X indicates that the user interacted with an item, while a value of 0 indicates that no interaction has been observed.

The parameters of the EASE^R model are given by the item-item weight-matrix $B \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{I}|}$. Note that this is also similar to neighborhood-based approaches, see Section 4.3. In this weight matrix, self-similarity of an item in the input-layer with itself in the output layer is forbidden as to force the model to generalize when reproducing the input as its output (see Figure 1): hence the diagonal of the weight-matrix is constrained to zero, $\text{diag}(B) = 0$. This constraint is crucial, and is discussed in detail in the remainder of this paper. This constraint was first introduced in the SLIM model [20].

The predicted score $S_{u,j}$ for an item $j \in \mathcal{I}$ given a user $u \in \mathcal{U}$ is defined by the dot product

$$S_{uj} = X_{u,\cdot} \cdot B_{\cdot,j}, \tag{1}$$

where $X_{u,\cdot}$ refers to row u, and $B_{\cdot,j}$ to column j.

3 MODEL TRAINING

We use the following convex objective for learning the weights B:

$$\min_B \quad ||X - XB||_F^2 + \lambda \cdot ||B||_F^2 \tag{2}$$

$$\text{s.t.} \quad \text{diag}(B) = 0 \tag{3}$$

---

[2]Non-binary matrices may also be used.

Several comments are in order:

- We choose the square loss ($\|\cdot\|_F$ denotes the Frobenius norm) between the data X and the predicted scores $S = XB$ over other loss functions because it allows for a closed-form solution (see next section). Training with other loss functions, however, might result in improved ranking-accuracy: in [17], it was observed that the multinomial likelihood resulted in better ranking-accuracy than training with the logistic likelihood (log loss) or the Gaussian likelihood (square loss). Directly optimizing a (surrogate) ranking loss might result in further accuracy gains–however, at possibly increased computational costs.
- We use L2-norm regularization of the weights B to be learned. The training objective hence has a single hyperparameter λ, to be optimized on a validation set.
- The constraint of a zero diagonal, $\mathrm{diag}(B) = 0$, is crucial as to avoid the trivial solution $B = I$ (self-similarity of items), where I is the identity matrix. It was introduced in SLIM [20].

3.1 Closed-Form Solution

In this section, we show that the constrained convex optimization problem for learning the weight matrix B in Eqs. 2 and 3 can be solved in closed form.

We start by including the equality constraint in Eq. 3 into the objective function in Eq. 2 by forming the Lagrangian

$$L = \|X - XB\|_F^2 + \lambda \cdot \|B\|_F^2 + 2 \cdot \gamma^\top \cdot \mathrm{diag}(B),$$

where $\gamma = (\gamma_1, ..., \gamma_{|I|})^\top$ is the vector of Lagrangian multipliers. Its values will be chosen in Eq. 6 such that the constraint in Eq. 3 is fulfilled.

The constrained optimization problem in Eqs. 2 and 3 is solved by minimizing this Lagrangian. As a necessary condition, we hence set its derivative to zero, which yields the estimate $\hat{B}$ of the weight matrix after re-arranging terms:

$$\hat{B} = (X^\top X + \lambda I)^{-1} \cdot (X^\top X - \mathrm{diagMat}(\gamma)),$$

where $\mathrm{diagMat}(\cdot)$ denotes the diagonal matrix and I the identity matrix. Defining (for sufficiently large λ)

$$\hat{P} \triangleq (X^\top X + \lambda I)^{-1}, \tag{4}$$

this can be substituted into the previous equation:

$$
\begin{aligned}
\hat{B} &= (X^\top X + \lambda I)^{-1} \cdot (X^\top X - \mathrm{diagMat}(\gamma)) \\
&= \hat{P} \cdot (\hat{P}^{-1} - \lambda I - \mathrm{diagMat}(\gamma)) \\
&= I - \hat{P} \cdot (\lambda I + \mathrm{diagMat}(\gamma)) \\
&= I - \hat{P} \cdot \mathrm{diagMat}(\tilde{\gamma}) \tag{5}
\end{aligned}
$$

where we defined the vector $\tilde{\gamma} \triangleq \lambda \vec{1} + \gamma$ in the last line, with $\vec{1}$ denoting a vector of ones. The values of the Lagrangian multipliers γ, and hence $\tilde{\gamma}$, are determined by the constraint $\mathrm{diag}(\hat{B}) = 0$. It follows from Eq. 5:

$$0 = \mathrm{diag}(\hat{B}) = \vec{1} - \mathrm{diag}(\hat{P}) \odot \tilde{\gamma} \tag{6}$$

where $\odot$ denotes the elementwise product, and hence:

$$\tilde{\gamma} = \vec{1} \oslash \mathrm{diag}(\hat{P}),$$

where $\oslash$ denotes the elementwise division (which is well-defined given that $\hat{P}$ is invertible). Substituting this into Eq. 5 immediately results in the closed-form solution:

$$\hat{B} = I - \hat{P} \cdot \mathrm{diagMat}\left(\vec{1} \oslash \mathrm{diag}(\hat{P})\right). \tag{7}$$

In other words, the learned weights are given by:

$$\hat{B}_{i,j} = \begin{cases} 0 & \text{if } i = j \\ -\dfrac{\hat{P}_{ij}}{\hat{P}_{jj}} & \text{otherwise.} \end{cases} \tag{8}$$

This solution obviously obeys the constraint of a zero diagonal. The off-diagonal elements are determined by the matrix $\hat{P}$ (see Eq. 4), where the jth column is divided by its diagonal element $\hat{P}_{jj}$. Note that $\hat{B}$ is an asymmetric matrix in general, while $\hat{P}$ is symmetric (see Eq. 4).

Eqs. 4 and 8 show that the sufficient statistics for estimating B is given by the data Gram-matrix $G \triangleq X^\top X$, which is an item-item matrix. This is a consequence of using the square loss in Eq. 2, and is helpful for estimating B from sparse data X: if X is a sparse binary matrix, then $G = X^\top X$ is a co-occurrence matrix. The uncertainty of a co-occurrence count G_{ij} is (approximately) determined by the standard deviation of the Poisson distribution, which is $\sqrt{G_{ij}}$. As long as the co-occurrence counts G_{ij} are 'sufficiently large', G and hence B can be estimated with small error. An interesting fact is that the entries of $G = X^\top X$ can be increased by two different mechanisms: (1) a denser X (due to users with increased activity), or (2) an increased number of users in X. The latter is particularly useful, as an increased sparsity of X can be compensated by an increased number of users. In other words, the problem that there possibly is only a small amount of data available for *each* user (i.e., data sparsity), does not affect the uncertainty in estimating B if the number of users in the data matrix X is sufficiently large.

3.2 Interpretation

In this section, we outline that the closed-form solution in Eq. 8 does not come as a complete surprise. To this end, let us consider the following special case throughout this section: let the training data X be an i.i.d. sample of $|\mathcal{U}|$ data points regarding a vector of $|I|$ random variables $x \sim \mathcal{N}(0, \Sigma)$ that follows a Gaussian distribution with zero mean and covariance matrix $\Sigma \in \mathbb{R}^{|I| \times |I|}$.

Then the estimate of the covariance matrix is $\hat{\Sigma} = X^\top X / |\mathcal{U}|$. If we further drop the L2-norm regularization in Eq. 4 and assume invertibility, then $\hat{P} = \hat{\Sigma}^{-1}$ is the estimate of the so-called precision (or concentration) matrix. Estimating the precision matrix from given data is a main objective in the area of graphical models (e.g., see [15]).

It is well known [5, 15] that the (univariate) *conditional distribution* of the random variable x_j given the vector of *all* the other variables, denoted by $x_{-j} \triangleq (x_k)_{k \in I \setminus \{j\}}$, is a Gaussian distribution with variance $\mathrm{var}(x_j | x_{-j}) = 1/P_{j,j}$ and mean

$$
\begin{aligned}
\mu_{j|-j} \triangleq \mathbb{E}[x_j | x_{-j}] &= -x_{-j} \cdot P_{-j,j} / P_{j,j} \\
&= x_{-j} \cdot B_{-j,j} \\
&= x \cdot B_{\cdot,j}
\end{aligned}
$$

where the dot in the first line denotes the dot-product between the (row) vector x_{-j} and the jth column of the precision matrix

$P = \Sigma^{-1}$, omitting the jth element. The second line follows from Eq. 8, and the last line from $B_{jj} = 0$. Note that this is identical to the prediction rule of the EASE$^\text{R}$ model in Eq. 1. This shows that EASE$^\text{R}$ makes a principled point-prediction that user u will like item j *conditional* on the fact that the user's past interactions with *all* items are given by $X_{u,\cdot} = x$.

A more well-known fact (e.g., see [19]) is that the absence of an edge between the random variables x_i and x_j in a Markov network corresponds to the *conditional independence* of the random variables x_i and x_j given the vector of *all* the other variables $(x_k)_{k \in I \setminus \{i,j\}}$, which is also equivalent to a zero entry in the precision matrix. Note that this is different from a zero entry in the covariance matrix Σ, which signifies *marginal independence* of x_i and x_j. This shows that the precision matrix is the conceptually correct similarity-matrix to be used, rather than the covariance matrix, which (or some rescaled variant thereof) is typically used in state-of-the-art neighborhood-based approaches (see Section 4.3).

Learning the graph structure in Markov networks corresponds to learning a sparse precision matrix from data. Approaches developed in that field (e.g., see [2, 7, 8, 13, 19, 23, 26, 29]) might be useful for improved learning of a *sparse* matrix $\hat{B}$. This is beyond the scope of this short paper.

While the interpretation we outlined in this section is limited to the special case of normally distributed variables with *zero mean*, note that the derivation of Eq. 8 does not require that each column of the data matrix X has zero mean. In other words, in EASE$^\text{R}$ any definition of the Gram matrix $G \triangleq X^\top X$ may be used, e.g., G may be the co-occurrence matrix (if X is a binary matrix), proportional to the covariance matrix (if X is pre-processed to have zero mean in each column), or the correlation matrix (if each column of X is pre-processed to have zero mean and unit variance). After training EASE$^\text{R}$ on these transformed matrices X and then transforming the predicted scores back to the original space (as defined by the binary matrix X), we found in our experiments that the differences in the obtained ranking accuracies were of the order of the standard error, and we hence do not separately report these results in Section 5.

3.3 Algorithm

The Python code of the resulting learning algorithm is given in Algorithm 1. Note that the training of EASE$^\text{R}$ requires only the item-item matrix $G = X^\top X$ as input, instead of the user-item matrix X, and hence is particularly efficient if the size of G (i.e., $|I| \times |I|$) is smaller than the number of user-item-interactions in X. In this case, the expensive computation of $G = X^\top X$ can be done on a big-data pre-processing system, prior to the actual model training.

3.4 Computational Cost

The computational complexity of Algorithm 1 is determined by the matrix inversion of the data Gram-matrix $G \triangleq X^\top X \in \mathbb{R}^{|I| \times |I|}$, which is $O(|I|^3)$ when using a basic approach, and about $O(|I|^{2.376})$ when using the Coppersmith-Winograd algorithm. Note that this is independent of the number of users as well as the number of user-item-interactions, as G can be computed in the pre-processing step.

This computational complexity is orders of magnitude lower than the cost of training a SLIM model and its variants [16, 20, 24]:

ALGORITHM 1: Training in Python 2 using numpy

Input: data Gram-matrix $G := X^\top X \in \mathbb{R}^{|I| \times |I|}$,
  L2-norm regularization-parameter $\lambda \in \mathbb{R}^+$.
Output: weight-matrix B with zero diagonal (see Eq. 8).
$diagIndices$ = numpy.diag_indices(G.shape[0])
$G[diagIndices]$ += λ
P = numpy.linalg.inv(G)
B = P / (-numpy.diag(P))
$B[diagIndices]$ = 0

---

those approaches take advantage of the fact that the optimization problem regarding $|I|$ items can be decomposed into $|I|$ independent (and hence embarrassingly parallel) optimization problems involving $|I| - 1$ items each, due to the identity $||X - XB||_F^2 = \sum_{j \in I} |X_{\cdot,j} - XB_{\cdot,j}|_2^2$. If each of the $|I|$ independent problems is solved based on an item-item matrix, the total computational cost is hence $O(|I|(|I| - 1)^{2.376})$. Note that the computational cost of solving those $|I|$ problems is larger by almost a factor of $|I|$ than training EASE$^\text{R}$, which requires only a *single* regression problem to be solved. In practice, however, SLIM and its variants are trained on the user-item-interactions in X, which may incur additional computational cost. This explains the vastly reduced training-times of EASE$^\text{R}$ observed in our experiments in Section 5.

In practice, the wall-clock time depends crucially on the fact if the number of items $|I|$ is sufficiently small such that the weight matrix fits into memory, so that the matrix inversion can be computed in memory. This was the case in our experiments in Section 5.

4 RELATED WORK

EASE$^\text{R}$ can be viewed as an autoencoder, as a modified version of SLIM, and a neighborhood-based approach. We discuss each of the three related approaches in the following.

4.1 Deep Learning and Autoencoders

While the area of collaborative filtering has long been dominated by matrix factorization approaches, recent years have witnessed a surge in deep learning approaches [6, 10–12, 17, 25, 30, 31], spurred by their great successes in other fields. Autoencoders provide the model architecture that fits exactly the (plain-vanilla) collaborative filtering problem. While various network architectures have been explored, it was found that deep models with a large number of hidden layers typically do not obtain a notable improvement in ranking accuracy in collaborative filtering, compared to 'deep' models with only one, two or three hidden layers, e.g., [10, 17, 25, 31], which is in stark contrast to other areas, like computer vision. A combination of deep and shallow elements in a single model was proposed in [6].

In contrast, EASE$^\text{R}$ has no hidden layer. Instead, the self-similarity of each item in the input and output layer is constrained to zero, see also Figure 1. As a consequence, the model is forced to learn the similarity of an item in the output layer in terms of the *other* items in the input layer. The surprisingly good empirical results of EASE$^\text{R}$ in Section 5 suggest that this constraint might be more effective than using hidden layers with limited capacity as to force the model to generalize well to unseen data.

4.2 SLIM and Variants

While the SLIM model [20] has shown competitive empirical results in numerous papers, it is computationally expensive to train, e.g., see [17, 20] and Section 3.4. This has sparked follow-up work proposing various modifications. In [16], both constraints on the weight matrix (non-negativity and zero diagonal) were dropped, resulting in a regression problem with an elastic-net regularization. While competitive ranking results were obtained in [16], in the experiments in [17] it was found that its performance was considerably below par. The square loss in SLIM was replaced by the logistic loss in [24], which entailed that both constraints on the weight matrix could be dropped, as argued by the authors. Moreover, the L1-norm regularization was dropped, and a user-user weight-matrix was learned instead of an item-item matrix.

All these approaches take advantage of the fact that the optimization problem decomposes into independent and embarrassingly parallel problems. As discussed in the previous section, however, this is several orders of magnitudes more costly than training EASE^R if the weight matrix fits into memory.

Most importantly, while those modifications of SLIM dropped the constraint of a zero diagonal in the weight matrix, it is retained in EASE^R. In fact, we found it to be the most crucial property for achieving improved ranking accuracy (see Section 5). As we showed in Section 3.1, this constraint can be easily included into the training objective via the method of Lagrangian multipliers, allowing for a closed-form solution.

Compared to SLIM [20], we dropped the constraint of non-negative weights, which we found to greatly improve ranking accuracy in our experiments (see Table 1 and Figure 2). Moreover, we did not use L1-norm regularization for computational efficiency. We also did not find sparsity to noticeably improve ranking accuracy (see Section 5). The learned weight matrix $\hat{B}$ of EASE^R is dense.

Also note that extensions to SLIM like cSLIM [21], can be turned into an analogous extension of EASE^R.

4.3 Neighborhood-based Approaches

Numerous neighborhood-based approaches have been proposed in the literature (e.g., see [27, 28] and references therein). While model-based approaches were found to achieve better ranking accuracy on some data sets, neighborhood-based approaches dominated on others, e.g., the Million Song Data Competition on Kaggle [1, 18]. Essentially, the co-occurrence matrix (or some modified variant) is typically used as item-item or user-user similarity matrix in neighborhood-based methods. These approaches are usually heuristics, as the similarity matrix is not learned by optimizing an objective function (loss function or likelihood). More importantly, the closed-form solution derived in Eqs. 4 and 8 reveals that the *inverse* of the data Gram matrix is the conceptually correct similarity matrix,[3] see Section 3.2 for more details. This is in contrast to the typical neighborhood-based approaches, which use the data Gram-matrix without inversion. The use of the conceptually correct, inverse matrix in EASE^R may explain the improvement observed in Table 2 compared to the heuristics used by state-of-the-art neighborhood approaches.

---

[3]In fact, inverse matrices are used in many areas, for instance, the inverse covariance matrix in the Gaussian density function or in the Mahalanobis distance.

5 EXPERIMENTS

In this section, the proposed EASE^R model is empirically compared to several state-of-the-art approaches, based on two papers that provided publicly available code for reproducibility of results [17, 28]. Both papers together cover linear, non-linear, deep and probabilistic models, as well as neighborhood-based approaches.

5.1 Experimental Set-up

We will only summarize the experimental set-ups used in these papers, and refer the reader to these papers for details.

Summary of Set-up in [17]: This paper considers the following models:

- Sparse Linear Method (SLIM) [20]. Besides the original model, also a computationally faster approximation (which drops the constraints on the weights) [16] was considered, but its results were not found to be on par with the other models in the experiments in [17].
- Weighted Matrix Factorization (WMF) [14, 22], a linear model with a latent representation of users and items.
- Collaborative Denoising Autoencoder (CDAE) [30], a non-linear model with one hidden layer.
- denoising autoencoder (MULT-DAE) and variational autoencoder (MULT-VAE ^{PR}) [17], both trained using the multinomial likelihood, which was found to outperform the Gaussian and logistic likelihoods. Best results were obtained in [17] for the MULT-VAE ^{PR} and MULT-DAE models that were rather shallow 'deep models', namely with a 200-dimensional latent representation, as well as a 600-dimensional hidden layer in both the encoder and decoder. Both models are non-linear, and MULT-VAE ^{PR} is also probabilistic.

Three data sets were used in the experiments in [17], and were pre-processed and filtered for items and users with a certain activity level, resulting in the following data-set sizes, see [17] for details:[4]

- MovieLens 20 Million (*ML-20M*) data [9]: 136,677 users and 20,108 movies with about 10 million interactions,
- Netflix Prize (*Netflix*) data [3]: 463,435 users and 17,769 movies with about 57 million interactions,
- Million Song Data (*MSD*) [4]: 571,355 users and 41,140 songs with about 34 million interactions.

The evaluation in [17] was conducted in terms of *strong generalization*, i.e., the training, validation and test sets are disjoint in terms of users. This is in contrast to *weak generalization*, where the training and test sets are disjoint in terms of the user-item interaction-pairs, but not in terms of users. Concerning evaluation in terms of ranking metrics, Recall@k for $k \in \{20, 50\}$ as well as Normalized Discounted Cumulative Gain, NDCG@100 were used in [17].

Summary of Set-up in [28]: Their paper focuses on neighborhood-based approaches, and the authors publicly shared code[5] regarding the experiments in their table 2 in [28], albeit only for the data-split where the training data was comprised of (at most) 30% of each user's interactions (and the remainder was assigned to the test data), which restricted our experimental comparison to this single

---

[4]The code regarding *ML-20M* in [17] is publicly available at https://github.com/dawenl/vae_cf. The authors kindly provided the code for the other two data sets upon request.
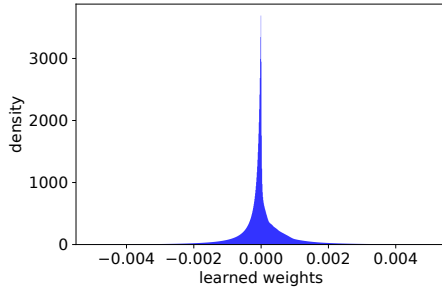[5]http://www.cs.toronto.edu/~mvolkovs/SVD_CF.zip

Figure 2: Histogram of the weights learned on *Netflix* data.

split in Table 2. They used the MovieLens 10 Million (*ML-10M*) data [9], which was binarized in [28] and is comprised of 69,878 users and 10,677 movies with 10 million interactions. Their evaluation was done in terms of *weak generalization*, and NDCG@10 was used as ranking metric for evaluation in [28].

5.2 Results

Despite the simplicity of EASE^R, we observed that EASE^R obtained considerably better ranking accuracy than any of the competing models on most of the data sets. This remarkable empirical result is discussed in detail in the following.

Comparison to SLIM: Table 1 shows that EASE^R achieved notably increased accuracy compared to SLIM on all the data sets. This suggests that dropping the L1-norm regularization as well as the non-negativity constraint on the learned weights is beneficial. Our analysis indicates that the latter is especially important: as illustrated in Figure 2 on the *Netflix* data (the histograms for *ML-20M* and *MSD* data look almost identical up to re-scaling, and are omitted), the learned weights in EASE^R are distributed around 0. Interestingly, it turns out that about 60% of the learned weights are negative on all the data sets in our experiments (regarding both papers [17, 28]). This indicates that it is crucial to learn also the dissimilarity (negative weights) between items besides their similarity (positive weights). Moreover, when we simply set the negative weights to zero (see EASE^R ≥ 0 in Table 1), which obviously is not the optimal non-negative solution, the resulting accuracy drops and is very close to the one of SLIM. Apart from that, note that EASE^R ≥ 0 is still quite dense (40% positive weights) compared to SLIM, which indirectly indicates that the sparsity of SLIM (due to L1-norm regularization) did not noticeably improve the ranking accuracy of SLIM in our experiments.

Regarding regularization, the optimal L2-norm regularization parameter (λ) for EASE^R is about 500 on *ML-20M*, 1,000 on *Netflix*, and 200 on *MSD*. These values are much larger than the typical values used for SLIM, which often are of the order of 1, see [20]. Note that SLIM additionally uses L1-norm regularization, and hence has much fewer (non-zero) parameters than EASE^R.

As expected based on Section 3.4, we also found the (wall-clock) training-time of EASE^R to be smaller by several orders of magnitude compared to SLIM : [17] reports that parallelized grid search for SLIM took about two weeks on the *Netflix* data, and the *MSD* data was 'too large for it to finish in a reasonable amount of time' [17]. In contrast,

Table 1: Ranking accuracy (with standard errors of about 0.002, 0.001, and 0.001 on the *ML-20M*, *Netflix*, and *MSD* data, respectively), following the experimental set-up in [17].

(a) *ML-20M*	Recall@20	Recall@50	NDCG@100
popularity	0.162	0.235	0.191
EASE^R	0.391	0.521	0.420
EASE^R ≥ 0	0.373	0.499	0.402
results reproduced from [17]:			
SLIM	0.370	0.495	0.401
WMF	0.360	0.498	0.386
CDAE	0.391	0.523	0.418
MULT-VAE ^PR	0.395	0.537	0.426
MULT-DAE	0.387	0.524	0.419
(b) *Netflix*			
popularity	0.116	0.175	0.159
EASE^R	0.362	0.445	0.393
EASE^R ≥ 0	0.345	0.424	0.373
results reproduced from [17]:			
SLIM	0.347	0.428	0.379
WMF	0.316	0.404	0.351
CDAE	0.343	0.428	0.376
MULT-VAE ^PR	0.351	0.444	0.386
MULT-DAE	0.344	0.438	0.380
(c) *MSD*			
popularity	0.043	0.068	0.058
EASE^R	0.333	0.428	0.389
EASE^R ≥ 0	0.324	0.418	0.379
results reproduced from [17]:			
SLIM	— did not finish in [17] —		
WMF	0.211	0.312	0.257
CDAE	0.188	0.283	0.237
MULT-VAE ^PR	0.266	0.364	0.316
MULT-DAE	0.266	0.363	0.313

training EASE^R on the *Netflix* data took less than two minutes, and on the *MSD* data less than 20 minutes on an AWS instance with 64 GB RAM and 16 vCPUs in our experiments. These times have to be multiplied by the number of different hyperparameter-values to be grid-searched. Note, however, that EASE^R only has a single hyperparameter (regarding L2-norm regularization), while SLIM has two hyperparameters (concerning L1 and L2 norms) to be jointly optimized.

Comparison to linear and deep non-linear models in [17]: Table 1 shows that EASE^R was consistently outperformed on only the *ML-20M* data, and only by a small margin by the best competing model (MULT-VAE ^PR). On the *Netflix* and *MSD* data, EASE^R obtained significantly better ranking results than any of the competing linear, non-linear, deep or probabilistic models evaluated in [17]. On the *MSD* data, EASE^R even improved over the best competing model by 25%, 17% and 23% regarding Recall@20, Recall@50, and NDCG@100, respectively. This is consistent with the results of the Million Song Data Challenge on Kaggle [18], where neighborhood-based approaches were found to vastly outperform model-based
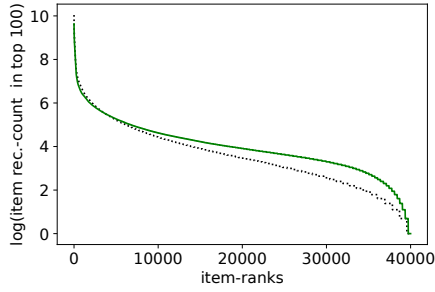
Figure 3: EASE^R (green) recommends long-tail items more often in the top-100, compared to Mult-vae ^PR (dotted), on *MSD* data.

Table 2: Comparison to the neighborhood-approaches in [28]: EASE^R considerably improves over 'ii-SVD-500' [28].

| | EASE^R | EASE^R ≥ 0 | reproduced from [28]: | | |
			ii-SVD-500	item-item	WMF
NDCG@10	0.6258	0.6199	0.6113	0.5957	0.5969

approaches [1]. As discussed in Section 4.3, EASE^R may also be viewed as a principled neighborhood approach.

As to explain EASE^R's relative improvements from *ML-20M* via *Netflix* to *MSD* data, various properties of the data sets may be considered. As shown by table 1 in [17], the number of user-item interactions, and the sparsity of the data sets do not appear well correlated with EASE^R's relative performance in Table 1. Only the number of users correlates well with the improvements of EASE^R over the competing models, which, however, appears to be spurious.

The explanation can be understood in terms of the tradeoff between recommending generally popular items vs. personally relevant items to each user, which is supported by two empirical findings: (1) we evaluated the popularity model in Table 1 as an additional baseline, where the items are ranked by their popularities (i.e., the number of users who interacted with an item). These unpersonalized recommendations obviously ignore the personalized relevance to a user. Table 1 shows that this popularity model obtains better accuracy on the *ML-20M* data than it does on the *Netflix* data, while its accuracy is considerably reduced on the *MSD* data. This suggests that good recommendations on the *MSD* data have to focus much more on personally relevant items rather than on generally popular items, compared to the *ML-20M* and *Netflix* data. (2) When counting how often an item was recommended in the top-100 across all test-users, and then ranking the items by their counts, we obtained Figure 3 for the *MSD* data: it shows that EASE^R recommended long-tail items more often than Mult-vae ^PR did. In contrast, there was almost no difference between the two approaches on either of the data sets *ML-20M* and *Netflix* (figures omitted due to page limit).

The notable improvement of EASE^R over the other models on the *MSD* data suggests that it is able to better recommend personally relevant items on this data set. On the other hand, EASE^R's results on the *ML-20M* and *Netflix* data suggest that it is also able to make recommendations with an increased focus on popular items. We suspect that EASE^R's large number of parameters, combined with its constraint regarding self-similarity of items, provides it with sufficient flexibility to adapt to the various data sets. In contrast, the model architectures based on hidden layers with limited capacity seem to be unable to adapt well to the increased degree of personalized relevance in the *MSD* data.

Comparison to neighborhood-based approaches in [28]: Considerable improvements were obtained in [28] by first predicting the scores for all user-item interactions with a neighborhood-based approach ('item-item' in Table 2) that was followed by a low-rank singular value decomposition of the predicted user-item score-matrix ('ii-SVD-500' in Table 2): an increase in NDCG@10 by 0.0156 and 0.0144 compared to the baseline models 'item-item' and WMF, respectively, as reproduced in Table 2. In comparison, EASE^R obtained increases of 0.0301 and 0.0289 over the baseline models 'item-item' and WMF, respectively, see Table 2. This is about twice as large an improvement as was obtained by the approach 'ii-SVD-500' proposed in [28].

Given the small size of this training data-set, a large L2-norm regularization was required ($\lambda = 3000$) for EASE^R. Like in the previous experiments, also here about 60% of the learned weights were negative in EASE^R, and setting them to zero (EASE^R ≥ 0 in Table 2) resulted in a small decrease in NDCG@10, as expected. In terms of wall-clock time, we found that training EASE^R was about three times faster than computing merely the factorization step in the ii-SVD-500 approach.

6 CONCLUSIONS

We presented a simple yet effective linear model for collaborative filtering, which combines the strengths of autoencoders and neighborhood-based approaches. Besides enabling efficient training (with savings of up to several orders of magnitude if the model fits into memory), the derived closed-form solution also shows that the conceptually correct similarity-matrix to be used in neighborhood approaches is based on the *inverse* of the given data Gram-matrix. In contrast, state-of-the-art neighborhood approaches typically use the data Gram-matrix directly. In our experiments, we found that allowing the learned weights to also take on negative values (and hence learn dissimilarities between items, besides similarities), was essential for the obtained ranking accuracy. Interestingly, the achieved ranking accuracy in our experiments was on par or even (notably) better than those of various state-of-the-art approaches, including deep non-linear models as well as neighborhood-based approaches. This suggests that models where the self-similarity of items is constrained to zero may be more effective on sparse data than model architectures based on hidden layers with limited capacity. We presented a basic version as to illustrate the essence of the idea, and leave various modifications and extensions for future work.

ACKNOWLEDGMENTS

REFERENCES

[1] F. Aiolli. 2013. Efficient top-N Recommendation for very large scale binary rated Datasets. In *ACM Conference on Recommender Systems (RecSys)*.

[2] O. Banerjee, L.E. Ghaoui, and A. d'Aspremont. 2008. Model Selection Through Sparse Maximum Likelihood Estimation for Multivariate Gaussian or Binary Data. *Journal of Machine Learning Research* 9 (2008).

[3] J. Bennet and S. Lanning. 2007. The Netflix Prize. In *Workshop at SIGKDD-07, ACM Conference on Knowledge Discovery and Data Mining*.

[4] T. Bertin-Mahieux, D.P.W. Ellis, B. Whitman, and P. Lamere. 2011. The Million Song Dataset. In *International Society for Music Information Retrieval Conference (ISMIR)*.

[5] J. Besag. 1975. Statistical analysis of non-lattice data. *The Statistician* 24 (1975), 179–95. Issue 3.

[6] H.T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS)*. 7–10.

[7] J. Friedman, T. Hastie, and R. Tibshirani. 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics* 9 (2008). Issue 3.

[8] D. Guillot, B. Rajaratnam, B. Rolfs, A. Maleki, and I. Wong. 2012. Iterative Thresholding Algorithm for Sparse Inverse Covariance Estimation. In *Advances in Neural Information Processing Systems (NIPS)*.

[9] F. M. Harper and J. A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5 (2015). Issue 4.

[10] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. 2017. Neural Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[11] B. Hidasi and A. Karatzoglou. 2017. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *International Conference on Information and Knowledge Management (CIKM)*. arXiv:1706.03847.

[12] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. arXiv:1511.06939.

[13] C.-J. Hsieh, M.A. Sustik, I.S. Dhillon, P.K. Ravikumar, and R. Poldrack. 2013. Sparse Inverse Covariance Estimation for a Million Variables. In *Advances in Neural Information Processing Systems (NIPS)*.

[14] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *IEEE International Conference on Data Mining (ICDM)*.

[15] S. L. Lauritzen. 1996. *Graphical Models*. Oxford University Press.

[16] M. Levy and K. Jack. 2013. Efficient Top-N Recommendation by Linear Regression. In *RecSys Large Scale Recommender Systems Workshop*.

[17] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[18] B. McFee, T. Bertin-Mahieux, D. P. Ellis, and G. R. Lanckriet. 2012. The Million Song Dataset Challenge. In *International World Wide Web Conference (WWW)*. http://www.kaggle.com/c/msdchallenge.

[19] N. Meinshausen and P. Bühlmann. 2006. High-dimensional graphs and variable selection with the Lasso. *Annals of Statistics* 34 (2006). Issue 3.

[20] X. Ning and G. Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *IEEE International Conference on Data Mining (ICDM)*. 497–506.

[21] X. Ning and G. Karypis. 2012. Sparse Linear Methods with Side Information for Top-N Recommendations. In *ACM Conference on Recommender Systems (RecSys)*.

[22] R. Pan, Y. Zhou, B. Cao, N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-Class Collaborative Filtering. In *IEEE International Conference on Data Mining (ICDM)*.

[23] M. Schmidt. 2011. *Graphical Model Structure Learning with L1-Regularization*. Ph.D. Dissertation. University of British Columbia, Vancouver, Canada.

[24] S. Sedhain, A. K. Menon, S. Sanner, and D. Braziunas. 2016. On the Effectiveness of Linear Models for One-Class Collaborative Filtering. *AAAI* (2016).

[25] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. 2015. Autorec: Autoencoders meet Collaborative Filtering. In *International World Wide Web Conference (WWW)*.

[26] E. Treister and J.S. Turek. 2014. A Block-Coordinate Descent Approach for Large-scale Sparse Inverse Covariance Estimation. In *Advances in Neural Information Processing Systems (NIPS)*.

[27] K. Verstrepen and B. Goethals. 2014. Unifying Nearest Neighbors Collaborative Filtering. In *ACM Conference on Recommender Systems (RecSys)*.

[28] M. N. Volkovs and G. W. Yu. 2015. Effective Latent Models for Binary Feedback in Recommender Systems. In *ACM Conference on Research and Development in Information Retrieval (SIGIR)*.

[29] A. Wiesel and A.O. Hero. 2012. Distributed Covariance Estimation in Gaussian Graphical Models. *IEEE Transactions on Signal Processing* 60 (2012). Issue 1.

[30] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. 2016. Collaborative Denoising Auto-Encoders for top-N Recommender Systems. In *ACM Conference on Web Search and Data Mining (WSDM)*.

[31] Y. Zheng, B. Tang, W. Ding, and H. Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *International Conference on Machine Learning (ICML)*.