ALTERNATING AUTOENCODERS FOR MATRIX COMPLETION

Kiwon Lee, Yong H. Lee, and Changho Suh

School of Electrical Engineering, KAIST, Korea

ABSTRACT

We consider autoencoders (AEs) for matrix completion (MC) with application to collaborative filtering (CF). It is observed that for a given sparse user-item rating matrix, denoted $\boldsymbol{M}$, an AE consisting of a nonlinear encoder followed by a linear decoder can perform matrix factorization so that $\widehat{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^T$ and sequentially estimates the feature matrices $\boldsymbol{U}$ and $\boldsymbol{V}$, where $\widehat{\boldsymbol{M}}$ is a completed version of $\boldsymbol{M}$. For the item-based AE (I-AE) whose inputs are the columns of $\boldsymbol{M}$, the AE's encoder first estimates $\boldsymbol{V}^T$ and then its decoder estimates $\boldsymbol{U}$ using the output of the encoder. Similarly, the user-based AE (U-AE) whose inputs are the columns of $\boldsymbol{M}^T$ first estimates $\boldsymbol{U}^T$ and then $\boldsymbol{V}$. This *sequential* estimation can degrade the performance of the MC, because the decoder depends on the output of the encoder. To enhance MC performance, we propose alternating AEs (AAEs) that employ both I-AE and U-AE and use them *alternately*. We apply the AAE to synthetic, MovieLens 100k and 1M data sets. The results demonstrate that AAE can outperform all existing MC/CF methods.

Index Terms— Matrix factorization, Collaborative filtering, Autoencoder, Recommendation systems

1. INTRODUCTION

The objective of matrix completion (MC) is to complete a low-rank matrix from its incomplete version with many missing entries. A conventional approach to tackle an MC problem is to solve a rank minimization (RM) problem that minimizes the rank of the completed matrix while maintaining the observed entries. Since an RM problem is computationally intractable (NP-hard), heuristic algorithms that solve the RM problems approximately but efficiently have been proposed [1]. In [2], the authors consider such a heuristic algorithm, which minimizes the trace of the completed matrix and derives a *lower bound* on the number of observed entries for an exact MC. In [3], the lower bound is improved with an efficient singular value decomposition (SVD) based algorithm, called OptSpace. Both the lower bounds are derived under the assumption that the pattern of missing entries is uniformly random. Although these results are interesting and provide guidelines for perfect MC, their use for practical applications is rather limited. This is true because the number of observed entries is often less than the lower bounds and the pattern of missing entries can be non-uniform. In applications such as recommendation systems, alternative methods for MC, which are referred to as collaborative filtering (CF), have been proposed.

In CF, most algorithms are based on the rank factorization theorem [4]. To be specific, let $\boldsymbol{M} \in \mathbb{R}^{n_1 \times n_2}$ be an incomplete rating matrix, and $\widehat{\boldsymbol{M}} \in \mathbb{R}^{n_1 \times n_2}$ be its completed version. Under the assumption that $\boldsymbol{M}$ and $\widehat{\boldsymbol{M}}$ have rank $r << \min(n_1, n_2)$, the matrix $\widehat{\boldsymbol{M}}$ is given by

$$\widehat{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{V}^T \tag{1}$$

where $\boldsymbol{U} \in \mathbb{R}^{n_1 \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{n_2 \times r}$ represent the user and item feature matrices, respectively. A popular approach to such feature extraction is to estimate $\boldsymbol{U}$ and $\boldsymbol{V}$ alternatively in the least-squares sense [5]. This type of LS estimate, called alternative LS (ALS), formed a major component of the winning entry in the Netflix Challenge [6].

As an alternative to the LS approach, deep learning techniques have been applied to CF. It is shown in [7] that CF based on a restricted Boltzmann machine (RBM) can perform slightly better than the LS methods. In addition, autoencoders (AEs), called AutoRec [8], and their modifications [9, 10] are used for CF. More recently, a neural autoregressive architecture, called CF-NADE [11], and geometric deep learning on user/item graphs [12] have been proposed for CF. Experimental results indicate that the AE-based techniques can outperform the LS- and RBM- based techniques, and CF-NADE performs the best.

In this paper, we analyze an AE consisting of a nonlinear encoder followed by a linear decoder and observe that the AE estimates the features matrices sequentially. For I-AE, the item feature matrix $\boldsymbol{V}^T$ is estimated first and the user feature matrix $\boldsymbol{U}$ is obtained as a function of $\boldsymbol{V}^T$. Similarly, for U-AE, $\boldsymbol{U}^T$ is estimated first and then $\boldsymbol{V}$ is obtained. This sequential estimation of feature matrices can cause some performance degradation, because one of the estimated feature matrices always depends on the other. To improve CF performance, we propose alternating autoencoders (AAEs) for CF that employ both I-AE and U-AE and use them alternately. We applied the AAEs to synthetic, MovieLens 100k and 1M data sets. The results demonstrate that AAE can outperform all existing MC/CF methods.

Notations : Matrices and vectors are denoted by boldfaced uppercase and lowercase letters, respectively. The column space and the rank of a matrix $\boldsymbol{A}$ are denoted by col($\boldsymbol{A}$)

and rank($\boldsymbol{A}$). In addition, the (i,j)-th entry of a matrix $\boldsymbol{A}$ is denoted as $\boldsymbol{A}_{ij}$, and $\|\boldsymbol{A}\|_F$ is the Frobenius norm of $\boldsymbol{A}$. The orthogonal projection of a vector $\boldsymbol{x} \in \mathbb{R}^m$ onto the space spanned by columns of $\boldsymbol{A} \in \mathbb{R}^{m\times r}$ is denoted as $P_{\boldsymbol{A}}(\boldsymbol{x})$.

2. RANK FACTORIZATION BY AUTOENCODERS

We shall show that an AE with the following characteristics can perform matrix factorization: i) Layers of the encoder, with the exception of the center layer, employ the rectified linear unit (ReLU). ii) For the center layer, either sigmoid or hyperbolic tangent ($tanh$) activation functions are used, and the number of units in this layer is equal to the rank r which is assumed to be known[1]. iii) The decoder employs linear activation functions. In this section, it is assumed that the pattern of missing entries of $\boldsymbol{M}$ is uniformly random.

Fig. 1 illustrates I-AE with 3 hidden layers ($n_L = 3$). Suppose that $\boldsymbol{M}$ is filled with a default rating of 0 for $\boldsymbol{M}_{ij}$s without rating observations. For the input $\boldsymbol{m}_i$ which is the i-th column of $\boldsymbol{M}$, the output of I-AE, $\hat{\boldsymbol{m}}_i$ is represented as $\hat{\boldsymbol{m}}_i = \boldsymbol{W}\boldsymbol{h}_i$, where $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_r] = \boldsymbol{W}_4\boldsymbol{W}_3 \in \mathbb{R}^{n_1\times r}$; $\boldsymbol{W}_3$ and $\boldsymbol{W}_4$ are the weighting matrices for the decoder, and $\boldsymbol{h}_i \in \mathbb{R}^r$ is the output of the encoder given by $\boldsymbol{h}_i = \sigma_S(\boldsymbol{W}_2\sigma_{RL}(\boldsymbol{W}_1\boldsymbol{m}_i + \boldsymbol{b}_1) + \boldsymbol{b}_2)$. Here $\sigma_S(\cdot)$ and $\sigma_{RL}(\cdot)$ are sigmoid and rectified linear units, respectively; $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are weighting matrices for the encoder; and $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are biases. The weights and biases of I-AE are determined by solving the following optimization problem via back-propagation:

$$\min_{\boldsymbol{W}_i, \boldsymbol{b}_i} \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} \boldsymbol{I}_{ij}\left(\boldsymbol{Y}_{ij} - \boldsymbol{M}_{ij}\right)^2 + \lambda \sum_{i=1}^{n_L+1} \|\boldsymbol{W}_i\|_F^2, \quad (2)$$

where $\boldsymbol{I}_{ij}$ is the indicator function that is equal to 1 if user i rated item j and equal to 0 otherwise, and λ is the regularization coefficient.

Suppose that I-AE parameters converge to their optimal values after a certain number of epochs. Denoting the resulting output matrices of the encoder and decoder by $\boldsymbol{H} = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{n_2}] \in \mathbb{R}^{r\times n_2}$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_2}] \in \mathbb{R}^{n_1\times n_2}$, respectively, the input-output relation can be written as

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{H}. \quad (3)$$

The matrices in (3) have rank r, as shown below.

Observation 1. After convergence, the matrices $\boldsymbol{Y}$, $\boldsymbol{W}$ and $\boldsymbol{H}$ for I-AE satisfy

$$\text{rank}(\boldsymbol{H}) = \text{rank}(\boldsymbol{W}) = \text{rank}(\boldsymbol{Y}) = r \quad (4)$$

In what follows, we justify this observation via simulation. The rank equalities in (4) holds if $\boldsymbol{W}^T\boldsymbol{W}$ is nonsingular,

$$\text{col}(\boldsymbol{Y}) = \text{col}(\boldsymbol{W}), \text{ and } \text{col}(\boldsymbol{Y}^T) = \text{col}(\boldsymbol{H}^T) \quad (5)$$

[1]Use of ReLU for the center layer is prohibited, because the dimension of output vectors from the center layer should be equal to r and remain constant.

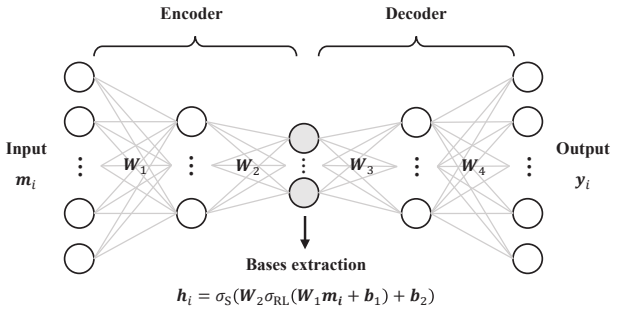

Fig. 1. An I-AE with 3 hidden layers ($n_L = 3$), where the activation functions for the encoder are nonlinear, while those for the decoder are linear.

The equality in (3) indicates that col($\boldsymbol{Y}$) $\subseteq$ col($\boldsymbol{W}$) and col($\boldsymbol{Y}^T$) $\subseteq$ col($\boldsymbol{H}^T$) [13]. Therefore (5) can be proved by showing that

$$\text{col}(\boldsymbol{Y}) \supseteq \text{col}(\boldsymbol{W}), \text{ and } \text{col}(\boldsymbol{Y}^T) \supseteq \text{col}(\boldsymbol{H}^T). \quad (6)$$

The subspace relations in (6) can be proved by showing that

$$\|P_{\boldsymbol{W}}(\boldsymbol{y}_i)\|/\|\boldsymbol{y}_i\| = 1 \text{ and } \|P_{\boldsymbol{H}^T}(\bar{\boldsymbol{y}}_i)\|/\|\bar{\boldsymbol{y}}_i\| = 1 \quad (7)$$

for all i, where $\bar{\boldsymbol{y}}_i$ is the i-th column of $\boldsymbol{Y}^T$. Next, we show through computer simulation that (7) holds for all i.

In the simulation, we first generate 100×100 matrices of rank two ($r = 2$) by multiplying two 100×2 matrices consisting of independent identically distributed (i.i.d.) Gaussian random variables. Then from each matrix an incomplete matrix $\boldsymbol{M} \in \mathbb{R}^{100\times 100}$ is obtained by sampling 15% of its entries uniformly at random positions. The results in this section are obtained through 100 generations of such $\boldsymbol{M}$ matrices.

We use AEs with 3 hidden layers, as shown in Fig. 1: since the inputs to the center layer can be either positive or negative, $tanh(\cdot)$ functions are employed in the center layer, and the number of units for this layer is set at $r = 2$, while those for the 1st and 3rd layers are set at 15. In this case, the rank r is known and there is no noise, and thus the regularization coefficient λ in (2) is set at $\lambda = 0$. The initial weights of AE are determined by the initialization method in [14] and the biases are initialized by zero. For back-propagation, the gradient-based optimization algorithm, called *Adam* [15], is used and the learning rate is 0.0001.

Fig. 2 shows the empirical means and variances of the normalized norms of projections against the training epoch. It is seen that the means and variances of the normalized norms converge to 1 and 0, respectively, after about 8,000 epochs. Furthermore, the maximum and minimum values of all the normalized norms after 10,000 epochs are 0.9997 and 0.9982, respectively. These results indicate that the column space relations in (6) hold true. During the simulation, we also observe that the products $\boldsymbol{W}^T\boldsymbol{W}$ and $\boldsymbol{H}\boldsymbol{H}^T$ are nonsingular, and that $\boldsymbol{W}$ satisfies (9), after about 8,000 epochs. These results show that the rank equalities in (4) are valid.

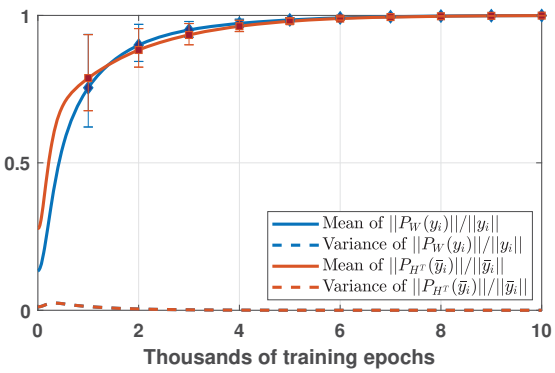

Fig. 2. Means and variances of the normalized norms in (7)

Next we show the rank factorization property of I-AE.

Observation 2. For I-AE,

$$\boldsymbol{Y} = \widehat{\boldsymbol{M}},\ \boldsymbol{W} = \boldsymbol{U} \text{ and } \boldsymbol{H} = \boldsymbol{V}^T \tag{8}$$

for the matrices in (1) and (3). Furthermore, the i-th row of $\boldsymbol{W}$, denoted as $\boldsymbol{w}_i^{row}$, can be represented as

$$\boldsymbol{w}_i^{row} = \boldsymbol{M}(i, \mathcal{I}_i)\boldsymbol{H}_{\mathcal{I}_i}^T(\boldsymbol{H}_{\mathcal{I}_i}\boldsymbol{H}_{\mathcal{I}_i}^T)^{-1} \tag{9}$$

where $\mathcal{I}_i$ denotes the set of items that user i rated, and $\boldsymbol{H}_{\mathcal{I}_i}$ denotes the sub-matrix of $\boldsymbol{H}$ where columns $j \in \mathcal{I}_i$ are selected. $\boldsymbol{M}(i, \mathcal{I}_i)$ is the row vector where columns $j \in \mathcal{I}_i$ of the i-th row of $\boldsymbol{M}$ is taken.

Proof. (8) holds true because of (4) and the fact that $\text{rank}(\boldsymbol{U}) = \text{rank}(\boldsymbol{V}) = \text{rank}(\widehat{\boldsymbol{M}}) = r$. The expression in (9) has been derived in [5] by solving a regularized LS problem for obtaining $\boldsymbol{W}$ when $\boldsymbol{H}$ is given. During the simulation for Fig. 2, we observe the validity of (9). □

This observation indicates that I-AE's encoder estimate $\boldsymbol{V}^T$ and then its decoder estimates $\boldsymbol{U}$ using the estimate of $\boldsymbol{V}^T$ (Fig 3).

In a similar manner, we can show that U-AE performs matrix factorization. In this case U-AE's input is $\boldsymbol{M}^T$, and the desired output is given by $\widehat{\boldsymbol{M}}^T = \boldsymbol{V}\boldsymbol{U}^T$. Denoting U-AE's output, the output of the decoder and that of the encoder, by $\overline{\boldsymbol{Y}}$, $\overline{\boldsymbol{W}}$ and $\overline{\boldsymbol{H}}$, respectively, we can show that

$$\overline{\boldsymbol{Y}} = \widehat{\boldsymbol{M}}^T,\ \overline{\boldsymbol{H}} = \boldsymbol{U}^T \text{ and } \overline{\boldsymbol{W}} = \boldsymbol{V}. \tag{10}$$

The equalities in (10) can be shown following the approach in Observations 1 and 2. U-AE's encoder estimates $\boldsymbol{U}^T$ and decoder obtains $\boldsymbol{V}$ using the estimates of $\boldsymbol{U}^T$ (Fig. 3).

3. ALTERNATING AUTOENCODERS

Fig. 3 shows the proposed AAE employing both I-AE and U-AE and their alternate use. After each training epoch, AAE evaluates the root mean square error (RMSE) of the current

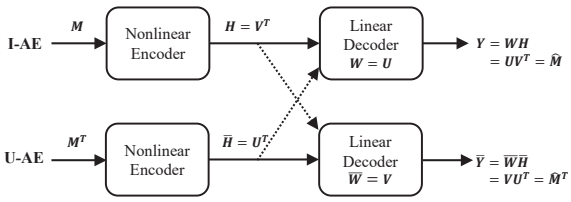

Fig. 3. I-AE, U-AE and AAE. Whenever alternation occurs in the AAE, the current AE's encoder output becomes the weighting matrix of the other AE's decoder.

AE, and alternates with the other AE if the RMSE becomes less than a threshold value. Whenever alternation occurs, the current AE's encoder output becomes the weighting matrix of the next AE's decoder. To be specific, we introduce the following notations: e_k and τ_k denote the RMSE for the k-th epoch and threshold, respectively. The index of training epoch at which alternation occurs is denoted as k_{al}. The threshold τ_k is adaptive and given by $\tau_k = e_{k_{al}} - \delta$ for a small constant δ.

Algorithm 1 illustrates an AAE that starts with I-AE (of course, the AAE can be started with U-AE). In steps 3-7, I-AE's encoder updates $\boldsymbol{H}_k$, while fixing the decoder weight $\boldsymbol{W}_k$ at $\boldsymbol{H}_{k_{al}}$. This update for $\boldsymbol{H}_k$ is repeated until the RMSE $e_k < \tau_{k-1}$. Since $\tau_0 = \infty$, at the beginning alternation occurs after a single epoch ($k_{al} = 1$). The threshold τ_k and alternation time k_{al} are updated in step 8. In a similar manner, the parameters for U-AE are updated in steps 9-14, and the alternation continues until the number of epochs becomes

Algorithm 1 Alternating autoencoders.

1: **Input:** Incomplete matrix $\boldsymbol{M}$, error tolerance δ and maximum training epoch $k_{\max}$.
2: **Initialize** $\overline{\boldsymbol{H}}_0$ and all weighting matrices of AEs, $k = 0$, $\tau_0 = \infty$ and $k_{al} = 0$.
3: **repeat**
4: $\quad k = k + 1$
5: $\quad$ Update $\boldsymbol{H}_k$ of I-AE, while fixing $\boldsymbol{W}_k = \overline{\boldsymbol{H}}_{k_{al}}^T$, via back-propagation.
6: $\quad$ Evaluate e_k.
7: **until** $e_k < \tau_{k-1}$
8: **Set** $\tau_k = e_k - \delta$ and $k_{al} = k$.
9: **repeat**
10: $\quad k = k + 1$
11: $\quad$ Update $\overline{\boldsymbol{H}}_k$ of U-AE, while fixing $\overline{\boldsymbol{W}}_k = \boldsymbol{H}_{k_{al}}^T$, via back-propagation.
12: $\quad$ Evaluate e_k.
13: **until** $e_k < \tau_k$
14: **Set** $\tau_k = e_k - \delta$ and $k_{al} = k$
15: **Go** to step 3 and repeat the process until $k = k_{\max}$.
16: **Output**: Estimated low-rank matrix $\widehat{\boldsymbol{M}} = \overline{\boldsymbol{H}}_{k_{\max}}^T \boldsymbol{H}_{k_{\max}}$

$k = k_{\max}$. The input-output relations of AE after the k-th epoch, are given by $\boldsymbol{Y}_k = \boldsymbol{W}_k \boldsymbol{H}_k$ for I-AE and $\overline{\boldsymbol{Y}}_k = \overline{\boldsymbol{W}}_k \overline{\boldsymbol{H}}_k$ for U-AE; the final output in step 16 is given by the product of final outputs from U-AE's and I-AE's encoders: $\widehat{\boldsymbol{M}} = \overline{\boldsymbol{H}}_{k_{\max}}^T \boldsymbol{H}_{k_{\max}}$.

4. EXPERIMENTS

Both synthetic and MovieLens [16] data sets are used to compare the performances of the proposed and existing MC/CF techniques. As in the simulation for Fig. 2, the initial weights of the AE are determined by the method in [14] and the initial biases are set at zero; for back-propagation Adam [15] is used. For the AAE, the error margin $\delta = 0.0005$.

4.1. Synthetic data

We generate 500×500 matrices of rank 10 ($r = 10$), denoted as $\boldsymbol{M}_0$, and obtain incomplete matrices $\boldsymbol{M}$ by choosing $|\boldsymbol{M}|$ entries of $\boldsymbol{M}_0$, where $|\boldsymbol{M}|$ is the number of observed entries of $\boldsymbol{M}$. Following the approach in [3], the normalized reconstruction error, $E = \|\boldsymbol{M}_0 - \widehat{\boldsymbol{M}}\|_F / \|\boldsymbol{M}_0\|_F$, is evaluated for each pair of $(\boldsymbol{M}_0, \boldsymbol{M})$, and it is declared that $\boldsymbol{M}$ is successfully reconstructed if the error is less than 10^{-4}. We obtain the reconstruction rate which is given by the ratio between the number of successfully reconstructed matrices and the total number of generated $\boldsymbol{M}$ matrices (50 in this simulation).

The parameters of I-AE/U-AE are as follows: 3 hidden layers with 100, 10 and 100 units. Since the inputs of the center layer can take either positive or negative values, $tanh$ functions are employed in the center layer. The learning rate for the back-propagation is piecewise constant: 0.01 for $k \leq 40,000$, 0.001 for $40,000 < k \leq 80,000$ and 0.0001 for $80,000 < k \leq 100,000 = k_{max}$. AAE employs these I-AE and U-AE, and alternately uses them.

Fig. 4 compares the reconstruction rates of I-AE, U-AE, AAE and OptSpace [3]. In this case I-AE and U-AE exhibit identical performance, because the item- and user-vectors have identical distribution. I-AE/U-AE perform better than OptSpace, and AAE outperforms the others.

4.2. Movielens data

We evaluate and compare the AAE with OptSpace [3], ALS [5], LLORMA [17], AutoRec [8], and CF-NADE [11]. For MovieLens data, at each trial, we randomly choose 90% of the data and use them for training; the remaining data are used for testing and the root mean square error (RMSE) is evaluated. We repeat this trial procedure 5 times and report the average RMSE. For all I-AE/U-AE/AAE, the center layer employs the sigmoid activation functions, because the inputs of the center layer can take only non-negative values; the learning rate is set at 0.0001.

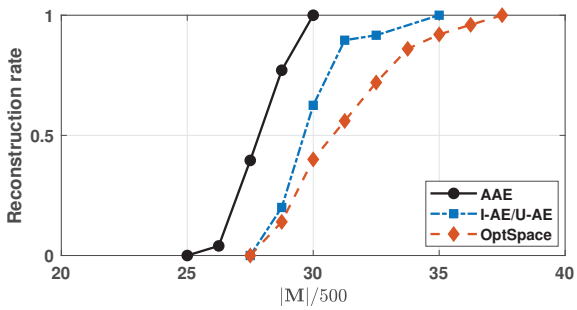

Fig. 4. Reconstruction rates for randomly generated 500×500 matrix with rank 10. Here $|\boldsymbol{M}|$ is the number of observed entries of $\boldsymbol{M}$.

The parameters of I-AE/U-AE for ML-100k data are as follows: 3 hidden layers with 100, 50 and 100 units. The regularization ratio λ is set at 0.00032. The parameters of I-AE/U-AE employed in AAE are the same except that $\lambda = 0.005$. For ML-1M data, we use I-AE and U-AE with different numbers of layers/units, because distribution of item-vectors is considerably different from that of user-vectors. The parameters of I-AE are: 3 hidden layers with 400, 500, 400 units; $\lambda = 0.00105$. The parameters of U-AE are: 5 hidden layers with 600, 400, 500, 400, 600 units; $\lambda = 0.0009$. These I-AE/U-AE are employed in AAE.

Table 1 compares the RMSE performance. AAE outperform all the other techniques. It is interesting to note that using alternation, the AAE behavior exceeds those of individual I-AE and U-AE.

Table 1. RMSE comparison

	ML-100k	ML-1M
Optspace [3]	0.911	0.873
ALS-WR [5]	0.913	0.843
LLORMA [17]	0.898	0.833
CF-NADE [11]	-	0.829
U-AutoRec [8]	-	0.874
I-AutoRec [8]	-	0.831
U-AE	0.905	0.841
I-AE	0.884	0.829
AAE	**0.877**	**0.826**

5. CONCLUSION

Based on the observation that an AE consisting of a nonlinear encoder and a linear decoder can perform matrix factorization and sequentially estimate the feature matrices, we proposed an AAE that employs both I-AE and U-AE and alternately uses them. Experimental results with synthetic, MovieLens 100k and 1M data sets demonstrated that the AAE can outperform all existing techniques.

6. REFERENCES

[1] M. Fazel, H. Hindi, and S. Boyd, "Rank minimization and applications in system theory," in *Proceedings of American Control Conference*. Boston, Massachusetts, 2004, pp. 3273–3278.

[2] Emmanuel J. Candés and Benjamin Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9(6), pp. 717–772, 2009.

[3] Raghunandan H. Keshavan, Andrea Montanari, and Sewoong Oh, "Matrix completion from a few entries," *IEEE Transactions on Information Theory*, vol. 56(6), pp. 2980–2998, 2010.

[4] Karim M. Abadir and Jan R. Magnus, *Matrix Algebra*, Cambridge University Press, Cambridge, 2005.

[5] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan, "Large-scale parallel collaborative filtering for the netflix prize," in *Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management*, 2008, pp. 337–348.

[6] Yehuda Koren, "The belkor solution to the netflix grand prize," *Netflix prize documentation*, vol. 81, pp. 1–10, 2009.

[7] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 2007, pp. 791–798.

[8] Suvash Sedhain, Menon Krishna, Scott Scanner, and Lexing Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 111–112.

[9] F. Strub and J. Mary, "Collaborative filtering with stacked denoising autoencders and sparse inputs," *in NIPS Workshop on Machine Learning for eCommerce*, 2015.

[10] Li Sheng, Jaya Kawale, and Yun Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, 2015, pp. 811–812.

[11] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou, "A neural autoregressive approach to collaborative filtering," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016, pp. 764–773.

[12] Federico Monti, Michael M. Bronstein, and Xavier Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," *Advances in Neural Information Processing Systems (NIPS)*, pp. 3700–3710, 2017.

[13] Jin Ho Kwak and Sungpyo Hong, *Linear algebra*, Birkhäuser, Boston, 2012.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.

[15] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[16] F. Maxwell Harper and Joseph A. Konstan, "The movielens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5(4), pp. 19, 2015.

[17] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer, "Local low-rank matrix approximation," in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013, pp. 295–303.