Learning Discrete Matrix Factorization Models
Duc Minh Nguyen, Evaggelia Tsiligiani, and Nikos Deligiannis, Member, IEEE

Abstract—Matrix factorization is among the most popular approaches for matrix completion, with recent advances including gradient-based and deep-learning-based methods. Even though many applications involve matrices with discrete values, most of the existing matrix factorization models focus on the continuous domain. Discretization is applied as an additional step, often using a heuristic mapping that results in sub-optimal solutions, which either do not take into account the structure of the matrix or introduce significant quantization errors. In this letter, we propose a novel method that allows gradient-based and deep-learning-based methods to jointly learn both the matrix factorization model and a discretization operator. By introducing a loss function that accounts for the reconstruction error with respect to the discrete predictions, we obtain a discrete matrix completion algorithm with high reconstruction accuracy. Experiments using well-known datasets show the improvement obtained by the proposed algorithm over the state of the art.

Index Terms—Matrix Factorization, Continuous Approximation, Deep Neural Networks, Big Data.

I. INTRODUCTION

MATRIX completion concerns the problem of recovering the missing entries of a matrix when a small number of observed entries are given. Signal processing and machine learning problems employing matrix completion include image inpainting [1], phase retrieval [2], big data modelling [3], recommender systems [4], [5], classification [6] and clustering [7], [8]. Suppose we are given a matrix \( M \in \mathbb{R}^{n \times m} \) with observed entries \( M_{ij}, (i,j) \in \Omega \), where \( \Omega \) is the set of indices corresponding to the known entries. A matrix completion algorithm seeks for a solution \( R^* \in \mathbb{R}^{n \times m} \) such that

\[
R^* = \arg \min_{R} \| P_{\Omega}(R) - P_{\Omega}(M) \|_F, \tag{1}
\]

with \( R^* \) denoting the complete matrix, \( P_{\Omega} \) an operator that indexes the entries defined in \( \Omega \), and \( \| . \|_F \) the Frobenius norm, that is, \( \| P_{\Omega}(R) - P_{\Omega}(M) \|_F = (\sum_{(i,j) \in \Omega} (R_{ij} - M_{ij})^2)^{1/2} \).

Most approaches in the literature rely on the assumption that \( M \) is a low-rank matrix [9], [10], [11], [12]. Since the rank minimization problem is NP-hard, convex relaxation by the nuclear norm is often employed in the optimization [9], [13], [14], [15]. Matrix factorization (MF) methods [4], [16], [17], [18], on the other hand, approximate the unknown matrix by a product of two low-rank matrices \( U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}, r \ll \min(n,m) \). Commonly used algorithms to learn \( U \) and \( V \) include alternating least squares and gradient descent [4]. Recently proposed gradient-based MF models [5], [19], [20], [21] take advantage of deep learning, which has dominated several tasks in machine learning [22], [23] and signal processing [24], [25].

Most of the existing MF models focus on the reconstruction of matrices with continuous values. However, big data applications such as recommender systems deal with discrete valued matrices. A common solution to discrete matrix completion, namely, the problem of recovering a matrix with discrete entries, is an additional step that converts the continuous predictions to discrete ones. Baseline approaches include rounding, quantization, or application-specific mappings from the continuous to the discrete domain [26], [27]. A more sophisticated solution was proposed in [28], where the authors learn both an MF model and a quantizer. Nevertheless, this model does not account for the actual reconstruction error (see also Section IV-C). Direct optimization in the discrete domain has been proposed in [29], where a nuclear norm minimization problem with additional discrete constraints is formulated. However, the high computational cost of the obtained problem makes its application in a high-dimensional big data setting impractical. Similar to discrete MF, ordinal MF also deals with discrete matrix entries. The works of [30], [31], [32] employ a probabilistic model that outputs continuous predictions preserving the ordering and a probability distribution over the possible discrete values; discretization can be achieved in an additional step.

In this letter, we propose a novel matrix completion method that enables direct prediction of discrete values. Our method integrates a discretization operator into an MF model and allows joint training of the MF and the discretization model using gradient descent. The proposed discretization does not depend on the specific MF realization and is combined with the basic MF model and a deep neural network model [20], [21]. The joint matrix completion model is applicable to datasets with an arbitrary number of discrete values and can be employed to address high-dimensional problems in a big data setting. Our experimental studies involve well-known matrix completion benchmarks, and show that our method outperforms the state of the art in discrete matrix completion.

The letter is organized as follows: In Section II, we formulate the problem of joint learning the MF model and the discretization operator. The proposed discretization operator is presented in Section III, and the discrete MF models in Section IV. Section V describes our experimental results, while Section VI concludes our letter.

II. PROBLEM FORMULATION

In a linear MF model, \( M = UV^T, M \in \mathbb{R}^{n \times m}, U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{m \times r}, r \ll \min(n,m) \), the matrix entries \( M_{ij} \) are modelled by linearly combining the item factors \( U_i, V_j \). The idea behind such models is that the row and column samples in a matrix can be represented by unobserved factors learned.
in a latent space. In matrix completion, we solve for the factors $U$, $V$, by minimizing the reconstruction loss

$$
L_{\text{rec}} = \sum_{ij \in \Omega} (U_i^j V^T_j - M_{ij})^2,
$$

(2)

where $M_{ij}$, $(i, j) \in \Omega$ are the known discrete matrix entries.

A discrete MF model involves a discretization operator $Q$ mapping the continuous predictions $U_i^j V^T_j$ to the discrete values $Q(U_i^j V^T_j)$. High reconstruction accuracy can be achieved if the optimization algorithm minimizes the reconstruction error with respect to the discrete predictions; therefore, we propose to formulate the reconstruction loss as follows:

$$
L_{\text{rec}} = \sum_{ij \in \Omega} (Q(U_i^j V^T_j) - M_{ij})^2.
$$

(3)

A plausible approach to discretize the continuous predictions is the use of a quantizer. However, a major drawback of this solution is that minimization of $L_{\text{rec}}$ cannot be achieved with gradient descent algorithms. A quantizer is a piecewise constant function [see Fig. 1(a)], thus, the gradient with respect to $U_i$, $V_j$ is zero everywhere except being undefined at some points. Directly applying (3) in a gradient-based learning system results in the vanishing gradient problem. To overcome this obstacle, we propose to approximate the quantization operator $Q$ with a discretization function that exhibits a non-vanishing gradient. The proposed approach can be used to learn both the discretization operator and an MF model, and it falls within the category of parametric continuous relaxations of non-differentiable functions [33], [34].

### III. LEARNING A DISCRETIZATION OPERATOR

The process of mapping real quantities into a finite number of values in a set $\mathcal{I}$ is referred to as quantization [35]. Suppose a set of quantization levels $\mathcal{I} = \{I_1, \ldots, I_d\}$; a quantizer divides the real line into $d$ non-overlapping consecutive intervals, defined as the range $[b_{v-1}, b_v)$, $v = 1, \ldots, d$. A quantity will be mapped to $I_v \in \mathcal{I}$ if it falls within the interval $[b_{v-1}, b_v)$. A quantization operator $Q_b$ can be defined as [35]

$$
Q_b(x) = \sum_{v=1}^{d} I_v \mathbb{I}_{b_{v-1} \leq x < b_v},
$$

(4)

where $\mathbb{I}_s$ is an indicator function, which outputs 1 if the condition $s$ is satisfied and 0 otherwise, and $b = [b_0, \ldots, b_d]^T$ are the quantization boundaries. An example of a uniform quantizer with $d = 5$ is shown in Fig. 1(a). In discrete matrix completion, the set of quantization levels is the set of allowable discrete values of the matrix entries.

Let us denote as $H_c(x) \equiv H(x-c)$ the heaviside step function, where $H(x-c) = 1$ for a non-negative argument $(x \geq c)$ and zero otherwise. Without loss of generality, we assume that $I_{v+1} - I_v = \Delta$, for all $v = 1, \ldots, d-1$. We define $q = [q_1, q_2, q_3, \ldots, q_{d-1}, q_d] = [b_0, I_2, I_3, \ldots, I_{d-1}, b_d]$. Then, (4) can be written as

$$
Q_b(x) = \sum_{v=1}^{d-1} [I_v + \Delta \cdot H_{b_v}(x)] \mathbb{I}_{q_v \leq x < q_{v+1}}.
$$

(5)

To enable the use of a discretization operation in a gradient-based optimization algorithm, we need a discretization function exhibiting a non-vanishing gradient. We propose to replace the heaviside step function by a logistic function of the form

$$
\sigma_{\lambda,c}(x) = (1 + e^{-\lambda(x-c)})^{-1},
$$

where $\lambda$ is a scalar denoting the center of the sigmoid, and $\lambda$ is the scale controlling the slope of $\sigma$. Therefore, we obtain the piecewise smooth function

$$
G_{\lambda,b}(x) = \sum_{v=1}^{d-1} \left[ I_v + \Delta \cdot \sigma_{\lambda,b_v}(x) \right] \mathbb{I}_{q_v \leq x < q_{v+1}}.
$$

(6)

The logistic function is related to the heaviside step function by the following property: $\lim_{\lambda \to +\infty} \sigma_{\lambda,c}(x) = H_c(x)$; thus, $G_{\lambda,b}$ becomes arbitrarily close to $Q_b$ in (5), when $\lambda$ becomes arbitrarily large. Figure 1(b) shows $G_{\lambda,b}$ for different values of $\lambda$, defined for $d = 5$ quantization intervals. Replacing $Q$ with $G_{\lambda,b}$ in our cost function (3) results

$$
L_{\text{rec}} = \sum_{ij \in \Omega} (G_{\lambda,b}(U_i^j V^T_j) - M_{ij})^2.
$$

(7)

We propose to optimize the reconstruction loss (7) with respect to the quantization boundaries $b_1, \ldots, b_{d-1}$ to learn a discretization model that fits the data distribution. The optimization can be achieved with gradient descent as $G_{\lambda,b}$, and thus $L_{\text{rec}}$, is differentiable in $(q_v, q_{v+1})$, $v = 1, \ldots, d-1$. The partial derivatives of $L_{\text{rec}}$ with respect to the boundary parameters $b_v$, $v = 1, \ldots, d-1$, are given by

$$
\frac{\partial L_{\text{rec}}}{\partial b_v} = 2 \sum_{ij \in \Omega} \left[ G_{\lambda,b}(U_i^j V^T_j) - M_{ij} \right] \frac{\partial G_{\lambda,b}(U_i^j V^T_j)}{\partial b_v}.
$$

(8)

For $q_v < U_i^j V^T_j < q_{v+1}$, with $v = 1, \ldots, d-1$, (6) results in

$$
G_{\lambda,b}(U_i^j V^T_j) = G_{\lambda,b_v}(U_i^j V^T_j) := I_v + \Delta \sigma_{\lambda,b_v}(U_i^j V^T_j).
$$

It can be easily shown that

$$
\frac{\partial G_{\lambda,b}(U_i^j V^T_j)}{\partial b_v} = -\lambda \sigma_{\lambda,b_v}(U_i^j V^T_j)[1 - \sigma_{\lambda,b_v}(U_i^j V^T_j)],
$$

and, therefore,

$$
\frac{\partial G_{\lambda,b}(U_i^j V^T_j)}{\partial b_v} = -\lambda \Delta \sigma_{\lambda,b_v}(U_i^j V^T_j)[1 - \sigma_{\lambda,b_v}(U_i^j V^T_j)].
$$

(9)

Even though $G_{\lambda,b}$ is discontinuous at the points $q_v$, $v = 1, \ldots, d$, we find that using its sub-gradient is sufficient for optimization with gradient descent algorithms.

In order to penalize the discretization functions deviating significantly from a uniform quantizer, we enforce the boundary values $b_v$ to be in $[I_v, I_{v+1}]$, $v = 1, \ldots, d-1$ [see Fig. 1(a)], by introducing a regularization term of the form

$$
\mathcal{L}_{\text{boundary}} = \| \mathbf{b} - \mathbf{b}^* \|^2_2, \text{ with } \mathbf{b} = [b_0, b_1, \ldots, b_d].
$$

Assuming $U_i^j V^T_j \in (-\infty, +\infty)$, we set $b_0 = -\infty$, $b_v = I_v + \Delta/2,$
\(v = 1, \ldots, d - 1\), and \(\tilde{b}_d = +\infty\). We note that \(b_0\) and \(b_d\) are set to \(-\infty\) and \(+\infty\), respectively, and are not updated during training. Therefore, the loss function for optimizing the discretization operator takes the form

\[ L_{\text{discrete}} = \sum_{i,j \in \Omega} (G_{\lambda,b}(U_iV_j^T) - M_{ij})^2 + \gamma \|b - \tilde{b}\|_2^2, \]

with \(\gamma\) being a regularization weight.

The accuracy of approximating a real discretization function with \(G_{\lambda,b}\) depends on \(\lambda\), which needs to be properly set during training. If \(\lambda\) is small, \(G_{\lambda,b}\) will be far from a real discretization function, whereas the larger \(\lambda\) is, the closer \(G_{\lambda,b}\) is to a real discretization function, yet, the more likely it is to get into the gradient vanishing problem. We employ the continuation method [36], [37] to solve this problem. At the beginning of the training, \(\lambda\) is set to a small value to avoid vanishingly small gradients, allowing efficient update of the model parameters. While training proceeds, we increase \(\lambda\), making the outputs closer to those of the real discretization function\(^1\).

At testing, the learned boundaries \(b\) are used to construct a real discretization function \(Q_b(x) = \lim_{\lambda \to +\infty} G_{\lambda,b}(x)\). If \(\lambda\) is increased to a large enough value during training, then the approximation error \(|G_{\lambda,b}(x) - Q_b(x)|\) is close to zero.

IV. DISCRETE MATRIX FACTORIZATION MODELS

Having described the details of the realization of a discretization model, we are now in the position to build the proposed discrete MF models, combining different gradient-based MF models with our discretization approach.

A. Discrete Basic Matrix Factorization model (BMF-D)

The first model is obtained from the basic factorization model (BMF) given by (2), and directly solves for \(U\), \(V\) and the discretization parameters \(b\) using the proposed reconstruction loss (7). We refer to this model as discrete BMF (BMF-D). The complete loss function of BMF-D, after integrating the constraints on the boundary values and the parameters of the employed MF model, takes the form

\[ L = \sum_{i,j \in \Omega} (G_{\lambda,b}(U_iV_j^T) - M_{ij})^2 + \frac{1}{2} \gamma_1 (\|U\|_2^2 + \|V\|_2^2) + \gamma_2 \|b - \tilde{b}\|_2^2, \]

where \(\gamma_1, \gamma_2\) are the regularization weights. The proposed loss is differentiable with respect to \(U\), \(V\) and the discretization parameters \(b\), and can be optimized with gradient descent, resulting in a jointly learned discrete MF model. The partial derivatives of the reconstruction loss with respect to \(b_v, v = 1, \ldots, d - 1\) are given by (8), (9). Denoting \(U_i = [u_{i1}, u_{i2}, \ldots, u_{ir}], V_j = [v_{j1}, v_{j2}, \ldots, v_{jr}], U_iV_j^T = \sum_{k=1}^r u_{ik}v_{jk}\), the partial derivatives of the reconstruction loss with respect to \(u_{i\ell}, \ell = 1, \ldots, r\) are obtained according to

\[ \frac{\partial L_{\text{rec}}}{\partial u_{i\ell}} = 2 \sum_{j \in \Omega} \left[ G_{\lambda,b}(U_iV_j^T) - M_{ij} \right] \frac{\partial}{\partial u_{i\ell}} G_{\lambda,b}(U_iV_j^T). \]

We start with a small \(\lambda\) and use an exponential scaling to update it so that it reaches a large value at the end of training. The detailed schedule update depends on the factorization model used and its hyperparameter settings. We found experimentally that an exponential scaling produces more stable results, compared to the step-wise and linear schedules [see also Sec. V].

For \(q_v < U_i V_j^T < q_{v+1}, v = 1, \ldots, d - 1\), it can be shown that

\[ \frac{\partial G_{\lambda,b}(U_iV_j^T)}{\partial u_{i\ell}} = \lambda v_{j\ell} \Delta \sigma_{\lambda,b}(U_iV_j^T)[1 - \sigma_{\lambda,b}(U_iV_j^T)]. \]

In a similar way, we obtain the partial derivatives of \(L_{\text{rec}}\) with respect to \(v_{j\ell}, \ell = 1, \ldots, r\).

B. Discrete Deep Matrix Factorization model (DMF-D)

The second model is obtained from the deep MF model (DMF) [20], [21]. In this model, the latent space vectors \(U_i, V_j\) are obtained from the corresponding input row and column vectors \(X_i, Y_j\), with \(X_iV_j^T = M_{ij}\), according to \(U_i = h_X(U_i)\) and \(V_j = h_Y(V_j)\); the non-linear differentiable functions \(h_X, h_Y\) are realized by a two-branch deep neural network architecture with parameters \(W_X, W_Y\), respectively. We refer to the discrete DMF as DMF-D. The loss function can be written as

\[ L = \sum_{i,j \in \Omega} (G_{\lambda,b}(h_X(U_i[h_Y(V_j)^T]) - M_{ij})^2 + \frac{1}{2} \gamma_1 (\|W_X\|_2^2 + \|W_Y\|_2^2) + \gamma_2 \|b - \tilde{b}\|_2^2, \]

and is differentiable with respect to \(W_X, W_Y\) and the discretization parameters \(b\), thus, it can be optimized with gradient descent. We omit the explicit formulas for the gradient due to space limitations.

C. Comparison with Existing Work

Next, we highlight the difference of the proposed approach against a discretization algorithm presented in [28], which also employs the BMF model. Assuming that the discrete matrix values are the integer values \(\mathcal{I} = \{1, 2, \ldots, d\}\), a quantizer is defined in [28] as follows: A continuous prediction is mapped to \(I_v \in \mathcal{I}, v = 1, \ldots, d\), if it belongs to the quantization interval \([I_v - (1 - t_{i_v,i_{v-1}}), I_v + t_{i_v,i_{v+1}}]\), where \(t_{i_v,i_{v-1}}, t_{i_v,i_{v+1}} \in [0, 1]\) are learned thresholds. The reconstruction loss employed in [28] has the form

\[ L_{\text{rec}} = \frac{1}{2} \sum_{i,j \in \Omega} \left( \max\{U_iV_j^T - M_{ij} - t_{(M_{ij} - M_{ij})}, 0\}^2 \right. \]

promoting solutions that satisfy \(U_iV_j^T - M_{ij} \leq t_{(M_{ij} - M_{ij})}\) or \(M_{ij} - U_iV_j^T \leq 1 - t_{(M_{ij} - M_{ij})}\); thus, during the optimization the error \(|U_iV_j^T - M_{ij}|\) is upper bounded. Denoting with \(Q_t\) the quantizer of [28], the reconstruction error with respect to the discrete predictions is \(|Q_t(U_iV_j^T) - M_{ij}|\), and upper bounding \(|U_iV_j^T - M_{ij}|\) does not necessarily minimize the reconstruction error as can be seen by the inequality

\[ |Q_t(U_iV_j^T) - M_{ij}| \leq |Q_t(U_iV_j^T) - U_iV_j^T| + |U_iV_j^T - M_{ij}|. \]

Clearly, the major advantage of the loss proposed in (7) is that it directly minimizes the reconstruction error with respect to the discrete predictions, and, consequently, our algorithm leads to higher reconstruction accuracy (see also Sec. V).

V. EXPERIMENTAL RESULTS

We carry out experiments on the ML100K and ML1M MovieLens datasets [38], involving users’ ratings of movies, with 100K and 1M known ratings in \{1, 2, \ldots, 5\}, respectively. We also employ the Book-Crossing dataset [39], after
selecting the most active users and books, resulting in a sub-matrix of size $2000 \times 2000$, with around 35.4K known entries in $\{1, 2, \ldots, 10\}$. For each dataset and each run, we randomly keep 75% of the known entries for training while the remaining 25% entries are reserved for testing. For evaluation, we use the Root Mean Square Error, $\text{RMSE} = \sqrt{\frac{\sum_{i,j \in \Omega_{\text{eval}}} (R_{ij} - \hat{M}_{ij})^2}{|\Omega_{\text{eval}}|}}$, and the Mean Absolute Error, $\text{MAE} = \frac{\sum_{i,j \in \Omega_{\text{eval}}} |R_{ij} - \hat{M}_{ij}|}{|\Omega_{\text{eval}}|}$, calculated over the entries reserved for testing ($\Omega_{\text{eval}}$). The reported results are averaged over 5 runs. As baseline models, we select the discrete models [28], [29]. We also compare with the well-known matrix completion methods in the continuous domain [40], [41], converting the continuous predictions to discrete ones by performing an additional step as in [28].

For the low-rank BMF-D model, we set the rank to $r = 25$. We search for the best hyper-parameter settings for the BMF-D and DMF-D on a separate validation set on the ML100K dataset. For the BMF-D model, the sigmoid scale is initialized to $\lambda = 0.5$ and increases by a factor of 1.01 every epoch. The model is trained in a batch-based fashion for 300 iterations (epochs). To ensure $\lambda$ reaches a large value at the end of training, we also scale $\lambda$ by a factor 1000 every 100 epochs. The regularization weights are set to $\gamma_1 = 0.00025$ and $\gamma_2 = 0.3$. For the DMF-D model, we put two hidden layers into each branch of the network, with 2048 and 1024 hidden units, respectively. The sigmoid scale is initialized to $\lambda = 0.1$ and increases by a factor of 1.015 every 10 training iterations, while we train the models for 10000 iterations. The regularization weights are set to $\gamma_1 = 0$ and $\gamma_2 = 0.1$.

Table I presents a comparison between the proposed BMF-D and DMF-D models with existing methods on the ML100K and ML1M datasets. On the ML100K dataset, the BMF-D model performs better than [41] and [29], with similar performance to [28] (same RMSE, lower MAE), and slightly worse than [40]. With more training data becoming available, the BMF-D model experiences less overfitting and generalizes better; therefore, on the ML1M dataset, it performs significantly better than [28], [40]. The prediction accuracy is improved when the proposed discretization is combined with the expressive power of deep neural networks; the DMF-D model outperforms all other models on both datasets by large margins$^2$. The results on the Book-Crossing dataset$^3$ presented in Table II are similar, with DMF-D and BMF-D outperforming [28] and DMF-D performing the best.

To further compare the efficiency of the proposed discretization method with the one presented in [28], independently of the MF model, we perform the following experiment: We train the BMF and DMF models employing the proposed reconstruction loss (7) and the reconstruction loss presented in [28], while we evaluate both the continuous $U_iV_j^T$ and the discrete $Q(U_iV_j^T)$ predicted values. The configurations for the BMF and DMF models are as presented earlier, and we tune for the best parameters for the loss in [28] on a separate validation set. The results on the ML1M dataset are shown in Table III. Concerning the predictions before applying the learned discretization function (continuous predictions), BMF and DMF models trained with the proposed loss function consistently outperform the ones trained with the loss in [28]. After discretization, the performance differences become even more significant, showing the efficiency of the proposed approach in learning better discretization operators than [28] as discussed in Sec. IV-C. We also provide results for the DMF model trained with the square error function (2). While the results in the continuous domain are similar, the discrete predictions of the DMF-D model are more accurate than the quantized versions of the former model.

We also tested the effect of the penalty term on the boundary values $b$ in (11), training BMF-D with $\gamma_2 = 0$. The performance was worse and some of the obtained boundaries $b_v$, $v = 1, \ldots, d - 1$, jumped out of the range $[I_v, I_{v+1})$. This justifies the employment of the proposed penalty on the boundary values.

### Table I: Results on the ML100K and ML1M datasets [38].

<table>
<thead>
<tr>
<th></th>
<th>ML100K</th>
<th>ML1M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RMSE</td>
<td>MAE</td>
</tr>
<tr>
<td>RankK [41]</td>
<td>1.004</td>
<td>0.706</td>
</tr>
<tr>
<td>OPTSPACE [40]</td>
<td>0.962</td>
<td>0.679</td>
</tr>
<tr>
<td>RMDC [29]</td>
<td>0.971</td>
<td>0.712</td>
</tr>
<tr>
<td>ODMC [28]</td>
<td>0.968</td>
<td>0.703</td>
</tr>
<tr>
<td>BMF-D</td>
<td>0.968</td>
<td>0.684</td>
</tr>
<tr>
<td>DMF-D</td>
<td>0.941</td>
<td>0.656</td>
</tr>
</tbody>
</table>

### Table II: Results on the Book-Crossing dataset [39].

<table>
<thead>
<tr>
<th></th>
<th>RMSE</th>
<th>MAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODMC [28]</td>
<td>1.741</td>
<td>1.282</td>
</tr>
<tr>
<td>BMF-D</td>
<td>1.096</td>
<td>1.254</td>
</tr>
<tr>
<td>DMF-D</td>
<td>1.544</td>
<td>1.120</td>
</tr>
</tbody>
</table>

### Table III: Results for continuous and discrete predictions on the ML1M dataset [38].

<table>
<thead>
<tr>
<th>Model</th>
<th>Discretization model</th>
<th>Continuous value</th>
<th>Discrete value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMF</td>
<td>[28]</td>
<td>0.887</td>
<td>0.707</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>0.884</td>
<td>0.700</td>
</tr>
<tr>
<td>DMF</td>
<td>[28]</td>
<td>0.859</td>
<td>0.685</td>
</tr>
<tr>
<td></td>
<td>Proposed</td>
<td>0.852</td>
<td>0.681</td>
</tr>
</tbody>
</table>

$^2$On the ML100K dataset, for the BMF-D and DMF-D, the corresponding margins of error are 0.0060 and 0.0045 for the RMSE, and 0.0048 and 0.0041 for the MAE with 95% confidence. On the ML1M dataset, the respective values are 0.0037 and 0.0017 for the RMSE, and 0.0095 and 0.0015 for the MAE.

$^3$On the Book-Crossing dataset, for the BMF-D and DMF-D models, the corresponding margins of error are 0.0351 and 0.0153 for the RMSE, and 0.0184 and 0.0177 for the MAE with 95% confidence.

### VI. Conclusions

We proposed a novel method to learn discrete MF models. In order to build a discretization model that can be trained with gradient descent, we approximate a quantization function with a combination of logistic functions to overcome the gradient vanishing problem. The proposed discretization model can be integrated into any gradient-based MF model allowing joint training of both models. Experimental studies with two different MF models show the effectiveness of our approach. Specifically, a discrete deep MF model, which combines the proposed discretization with a deep neural network model, outperforms the state of the art by large margins.