An Uncertainty-Aware Imputation Framework for Alleviating the Sparsity Problem in Collaborative Filtering

Sunghyun Hwang
Department of Artificial Intelligence
Hanyang University, South Korea
shhoff@hanyang.ac.kr

Dong-Kyu Chae*
Department of Artificial Intelligence
Hanyang University, South Korea
dongkyu@hanyang.ac.kr

ABSTRACT

Collaborative Filtering (CF) methods for recommender systems commonly suffer from the *data sparsity* issue. Data imputation has been widely adopted to deal with this issue. However, existing studies have limitations in the sense that both *uncertainty* and *robustness* of imputation have not been taken into account, where there is a high risk that the imputed values are likely to be far from the true values. This paper explores a novel imputation framework, named *Uncertainty-Aware Multiple Imputation* (UA-MI), which can effectively solve the sparsity issue. Given a (sparse) user-item interaction matrix, our key idea is to quantify *uncertainty* on each missing entry and then the cells with the *lowest uncertainty* are selectively imputed. Here, we suggest three strategies for measuring uncertainty in missing user-item interactions, each of which is based on *sampling*, *dropout*, and *ensemble*, respectively. They successfully obtain element-wise *mean* and *variance* on the missing entries, where the variance helps determine *where* in the matrix should be imputed and the corresponding *mean* values are imputed. Experiments show that our UA-MI framework significantly outperformed the existing imputation strategies.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**.

KEYWORDS

Collaborative filtering with implicit feedback, data sparsity, data imputation, uncertainty quantification

ACM Reference Format:
Sunghyun Hwang and Dong-Kyu Chae. 2022. An Uncertainty-Aware Imputation Framework for Alleviating the Sparsity Problem in Collaborative Filtering. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3511808.3557236

*Corresponding author

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557236

1 INTRODUCTION

The *data sparsity* [14] issue is one of the main challenges for collaborative Filtering (CF). *Data imputation* has been extensively studied in the CF research community [2, 14, 20, 21, 23, 24, 33–35], which infers the missing values of a given (sparse) user-item interaction matrix. The imputation strategies range from a simple zero imputation [5, 33] to inferring the missing values by using a machine learning model supervised by the observed interactions such as Random Forest [39], EM algorithm [1], Clustering based KNN [17], Generative Adversarial Network [2–4], and Autoencoder [22], to name a few.

However, we argue that the existing methods perform imputation without careful consideration of *where* and *what* to augment in the matrix; in other words, they naively decide which cells in the matrix to fill and impute a single value inferred by a deterministic model. This approach would not be reliable as well as robust due to a high risk that the chosen positions would be with high uncertainty and the inferred values are likely to be far from the true values.

For the viewpoint of robustness, *multiple imputation* instead of single imputation is necessary, which imputes the mean of multiple values on missing data obtained by multiple times of stochastic inference. Recently, multiple imputation methods have been studied outside the recommendation context [15, 25, 28, 30, 31, 36, 43, 44]. However, it is difficult to apply them directly to user-item matrix because they do not assume *Missing Not At Random* (MNAR) for most missing values; instead, most of them have been performed under the assumption of Missing Completely At Random (MCAR) or Missing At Random (MAR) [25, 30, 31, 43, 44]. But missing data is *not random* in the user-item interaction data used for recommendation systems [41]. Few studies were done on multiple imputation with the MNAR assumption [15, 28, 36], but they impute by calculating the mean from a deterministic probability distribution, which is essentially the same as unreliable single imputation method [27].

In this paper, we investigate a novel imputation framework, named *Uncertainty-Aware Multiple Imputation* (UA-MI). Its main idea is to consider *uncertainty*[1] in each empty cell and then *selectively* impute values only to matrix cells that are highly confident (i.e., low uncertainty). In addition, it performs multiple imputation on the chosen position rather than considering just a single value. Here, the question arises as to how to perform the uncertainty quantification and the multiple imputation on the given matrix. To answer this, we explore three methods based on sampling [40], dropout [37], and ensemble [10]. The sampling-based method exploits the re-parameterization trick [6, 18] in the latent

[1] Among the two types of uncertainty, i.e., *aleatoric uncertainty* and *epistemic uncertainty* [8], we focus on the *epistemic uncertainty*.

space. The dropout-based method creates implicit ensemble members of model parameters on the inference phase by applying the dropout to the trained model. The ensemble-based method creates explicit ensemble members by multiple training of models with different initializations. These methods eventually produce multiple predictive rates of the users, including their missing interactions. They enable us to know the *mean* and *variance* of each empty cell in the given matrix. The higher the variance is, the higher the uncertainty becomes. Therefore only the cells with the *lowest* variance are selectively imputed (we call this strategy as *uncertainty-aware positioning*). Finally, the imputed value is computed by the mean of user-item interaction probabilities drawn from each multinomial distribution parameterized by the obtained predictive rate (we call this step as *multiple imputation*). All the ideas are realized on top of the Variational Autoencoder model for CF (MultVAE) [26].

The strengths of our UA-MI framework can be summarized as follows: (1) While the existing methods naively decide which cells in the matrix to fill, our work chooses the cells much more carefully by considering only cells with low uncertainty (i.e., high confidence) and avoids imputing values in cells with high uncertainty. (2) By imputing the mean of multiple values obtained from the multinomial distribution, our imputation is more robust compared to the other existing methods that use a single value inferred by a deterministic model. (3) Our experimental results on four real-world datasets obtained from various domains (e.g., Amazon-Book, CiteULike, Epinions, and MovieLens) demonstrate that our imputation framework consistently and significantly improves the accuracy of recommendation compared to other imputation strategies. Our codes and data are available at: https://github.com/shhoff/UA-MI

2 DATA IMPUTATION FOR CF

This section briefly introduces existing data imputation methods. Specifically, we summarize them based on the following two perspectives: (1) how they decide which cells to impute in a matrix and (2) how they calculate the imputation values.

How to choose positions of the given matrix? From this perspective, we can first classify methods that impute *all empty cells* of the given matrix such as Preference Model [21], AllRank [38], RAGAN [2], PureSVD [5], among many others, and methods determining the positions for *partial imputation* [14, 20, 29, 34, 35]. When choosing the positions, neighborhood-based methods such as AutAI [34] and AdaM [35] heuristically find the key neighborhood users of a given user, and define the set of items that these key neighborhoods commonly rated as a key set. Only the ratings corresponding to this key set are predicted and imputed. In contrast, model-based methods employ a CF model to infer missing values, and then select the positions corresponding to some of the *lowest* predicted values [14, 20] assuming that lower values predicted by a model are more confident in showing the negative preference of users. However, none of the existing work considers uncertainty in the missing interactions, which makes their imputation not reliable due to a high risk that their chosen positions would be with high uncertainty.

How to compute the imputed values? From this point of view, we can classify existing methods into three categories: single value (non-computation) [5, 14, 33, 38], heuristic computation [21, 34, 35], and model-based computation [2, 23]. The methods in the first category include PureSVD [5] and Zero-Injection [14] that impute zero, and AllRank [38] that fills in two. AutAI [34] and AdaM [35] calculate values to impute with heuristic similarity-based metrics. Lastly, for model-based methods that infer and fill in the empty values, there are methods using Random Forest [39], Generative Adversarial Network [2], and Autoencoder [22], to name a few. However, all the existing work imputes values inferred by a deterministic model, which is not robust and likely to be biased.

3 UNCERTAINTY-AWARE IMPUTATION

This section introduces in detail the proposed *Uncertainty-Aware Multiple Imputation* (UA-MI) framework. Figure 1 illustrates the overview of our framework, which consists of three steps: uncertainty quantification, uncertainty-aware positioning, and multiple imputation. The following subsections elaborate on the process.

3.1 Uncertainty Quantification

This subsection explores three methods for quantifying uncertainty in unobserved user-item interactions. They are based on *sampling* [40], *dropout* [37], and *ensemble* [10], respectively. Note that they are not used simultaneously, but one of them gets chosen and used. Figure 2 shows the uncertainty quantification methods. As illustrated, they are realized on top of the Variational Autoencoder designed for the CF task, named Multinomial Variational Autoencoder (MultVAE) [26]. For concreteness (but without loss of generality), we present each method by assuming the MultVAE as the base model.

3.1.1 Base Model. We briefly introduce MultVAE before explaining the three uncertainty quantification methods. Let ϕ and θ denote the model parameters for the encoder and decoder of MultVAE, respectively. The encoder network encodes the posterior distribution $p_\phi(z_u|r_u)$ to generate a user u's latent variable vector z_u from her interaction vector r_u. However, since calculating the posterior distribution is intractable, an objective of approximating a relatively simple variational distribution $q_\phi(z_u|r_u)$ to $p_\phi(z_u|r_u)$ is used to train the encoder [11]. Next, the decoder network decodes multinomial distribution $p_\theta(r_u|z_u)$, which receives z_u generated from $q_\phi(z_u|r_u)$ as input and generates interaction vector r_u, which is a dense vector containing inferred probability of u's preference towards each item.

The objective function to train MultVAE is as follows:

$$\mathcal{L}_\beta(r_u;\theta,\ \phi) \equiv \mathbb{E}_{q_\phi(z_u|r_u)}[\log p_\theta(r_u|z_u)] \\ -\beta \cdot \mathrm{KL}[q_\phi(z_u|r_u)|p(z_u)] \tag{1}$$

where the right-hand side is the well-known Evidence Lower Bound (ELBO) that consists of the reconstruction term and the Kullback-Leibler (KL) divergence term with a free regularization parameter β [26]. The model parameters ϕ and θ are trained to maximize ELBO.

3.1.2 Sampling-based Method. The method introduced exploits sampling with the re-parameterization trick. The re-parameterization trick was originally used to help in the back-propagation process when training a VAE [6, 18], which we also have utilized in training MultVAE. Additionally, we rethink the re-parameterization trick as a way of generating a re-parameterized latent variable z_u. In this way, we can *indirectly* generate multiple samples of z_u as follows:

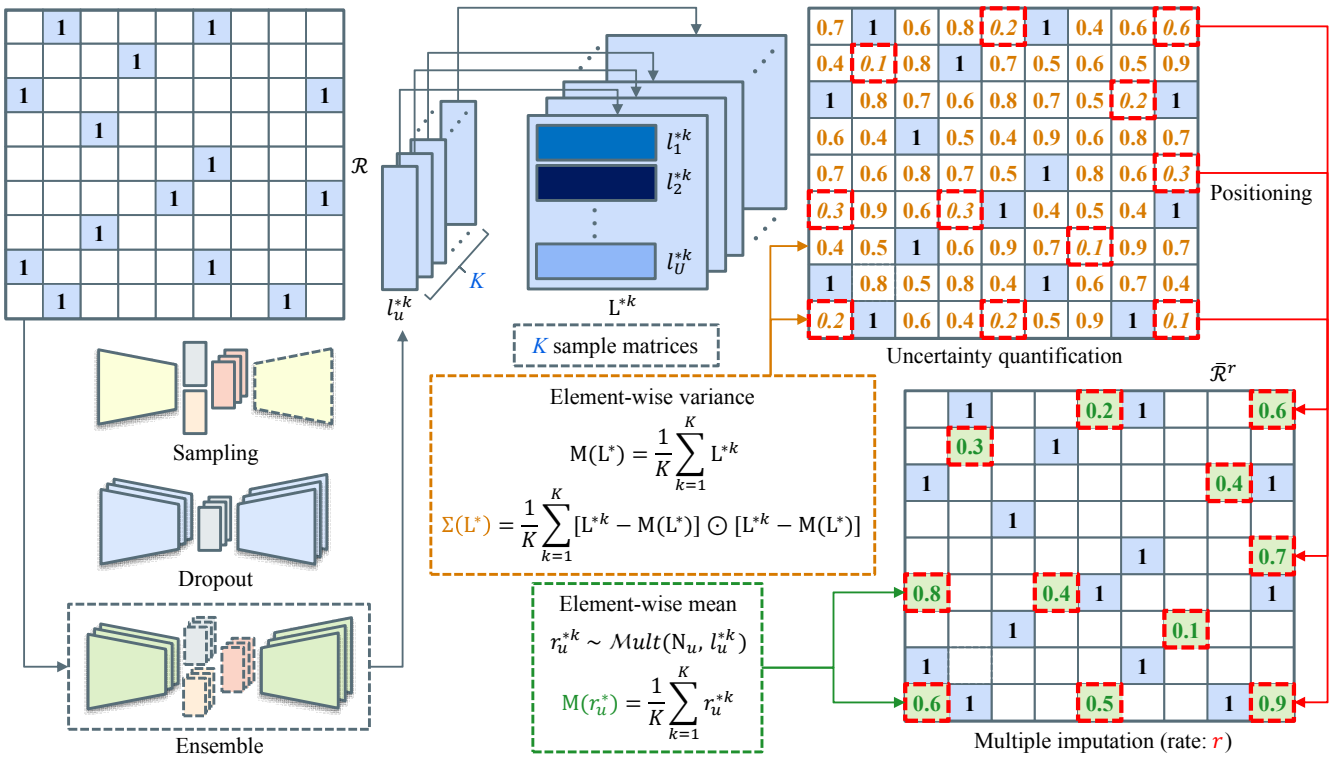


Figure 1: The overview of our UA-MI framework. It first selects one of the three methods (sampling-based, dropout-based, and ensemble-based) shown on the left. Let's assume that the ensemble-based method was chosen. Then, it learns the given implicit feedback matrix and generates K (dense) sample matrices. Based on them, it computes the element-wise variance to select the positions to impute. Finally, the mean of K interaction probabilities obtained from the multinomial distribution is imputed to the corresponding position. The input matrix and imputed matrix with the imputation rate r is denoted as $\mathcal{R}$ and $\bar{\mathcal{R}}^r$, respectively.

$$z_u^{*k} = \mu_\phi(r_u) + \epsilon \cdot \sigma_\phi(r_u) \qquad \epsilon \sim \mathcal{N}(0,\ \mathbf{I}_D) \tag{2}$$

where z_u^{*k} indicates k^{th} sampled latent variable. $\mu_\phi(\cdot)$ and $\sigma_\phi(\cdot)$ are the encoder mean and encoder standard deviation, respectively. ϵ is an D-dimensional random variable sampled from a standard multivariate Gaussian. Then each sampled z_u^{*k} can be used for estimating u's predictive rate for items, l_u^{*k}, by:

$$l_u^{*k} = \text{softmax}(f_\theta(z_u^{*k})) \tag{3}$$

where $f_\theta(\cdot)$ is the decoder output. As a result, for each user u, we have K number of predictive rates.

3.1.3 Dropout-based Method. In general, the internal logic of a standard neural network to make an output is deterministic. Applying dropout to the internal layers (except the output layer) enables the network to produce a set of stochastic outputs. For example, an input vector can be passed through the dropout network K times, where each time uses a different dropout mask (so-called Monte-Carlo dropout) [9]. Then we get K stochastic outputs. This can be also considered as an implicit ensemble of networks.

Motivated by this property, we developed an uncertainty quantification method based on Monte-Carlo dropout. First, we put a user u's interaction vector r_u to the trained MultVAE model, where we apply K different sets of dropout masks for the internal layers with a pre-defined dropout rate of p:

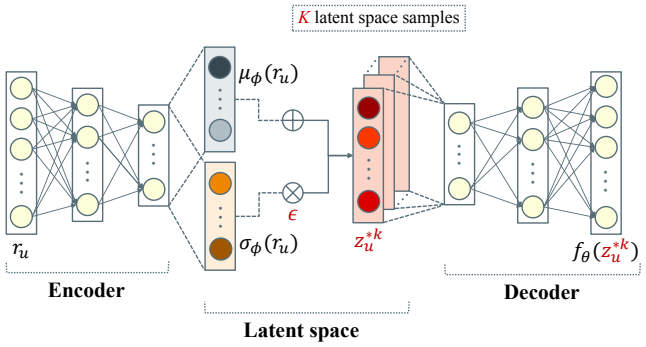
$$\{\phi_{*k,\,p},\ \theta_{*k,\,p}\} = \text{dropout}(\{\phi,\ \theta\},\ p) \tag{4}$$

where $\phi_{*k,\,p}$ and $\theta_{*k,\,p}$ are k^{th} set of parameters of the encoder and decoder, respectively, within the ensemble size K. In this way, we have K stochastic outputs where each one is computed by $f_{\theta_{*k,\,p}}(\mu_{\phi_{*k,\,p}}(r_u))$. Finally, we collect a user's K predictive rates, each computed by:

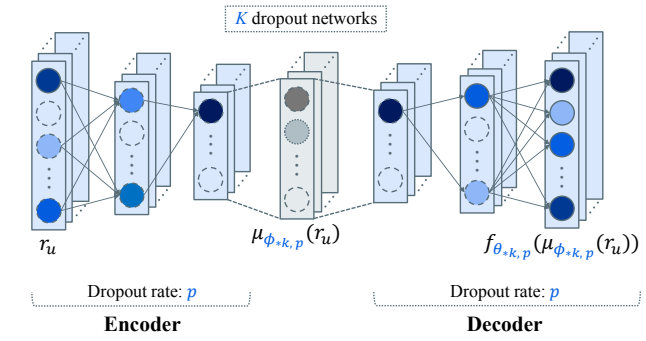
$$l_u^{*k} = \text{softmax}(f_{\theta_{*k,\,p}}(\mu_{\phi_{*k,\,p}}(r_u))) \tag{5}$$

3.1.4 Ensemble-based Method. While the dropout-based method generates an implicit ensemble of networks, the ensemble-based method introduced here constructs an explicit ensemble of networks [19]. Following the stacking ensemble approach [10], we train the MultVAE model multiple times with different initializations, where we expect each model converges to one of the local optima of the loss surface and the ensemble of them yields better uncertainty estimates [7, 32].

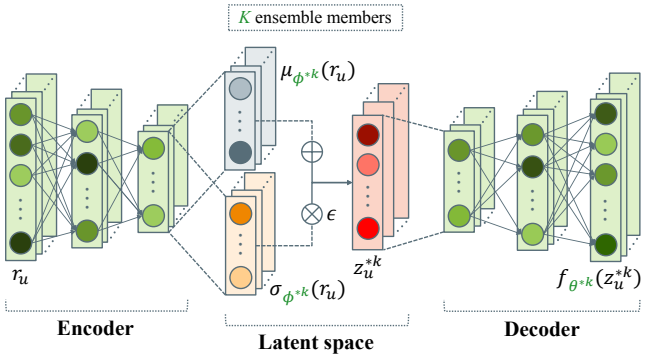
Formally, let ϕ^{*k} and θ^{*k} be the k^{th} trained model parameters of the encoder and decoder, respectively. Given the input r_u, the



(a) Sampling-based method



(b) Dropout-based method



(c) Ensemble-based method

Figure 2: The three methods for uncertainty quantification at a glance. (a) The latent space variable is sampled K times by indirect re-parameterization. (b) Dropout is performed on each internal layer to construct an implicit ensemble of K networks. (c) The network is trained in K times with different initialization to construct an explicit ensemble of networks.

corresponding latent variable is computed by:

$$z_u^{*k} = \mu_{\phi^{*k}}(r_u) + \epsilon \cdot \sigma_{\phi^{*k}}(r_u) \qquad \epsilon \sim \mathcal{N}(0,\ \mathbf{I}_D) \tag{6}$$

After then, we get the corresponding decoder output by $f_{\theta^{*k}}(z_u^{*k})$. We finally obtain k^{th} predictive rate of the user u by:

$$l_u^{*k} = \text{softmax}(f_{\theta^{*k}}(z_u^{*k})) \tag{7}$$

In this way, we get u's collection of K predictive rates, each obtained from the trained MultVAE model.

Finally, we now introduce how to measure uncertainty (i.e., variance) in each user-item interaction based on the predictive rates produced by each method explained so far. We collect all U users' k^{th} predictive rates and concatenate them to construct the k^{th} sample matrix, denoted as L^{*k}, by:

$$\mathrm{L}^{*k} = [l_1^{*k},\ l_2^{*k},\ \dots,\ l_u^{*k},\ \dots,\ l_U^{*k}]^T \tag{8}$$

Then, we collect K matrices $(\mathrm{L}^{*1}, \mathrm{L}^{*2},\ \dots, \mathrm{L}^{*K})$ and perform the following element-wise operations on them to obtain a variance matrix $\Sigma(\mathrm{L}^*)$:

$$\begin{aligned} \mathrm{M}(\mathrm{L}^*) &= \frac{1}{K}\sum_{k=1}^{K} \mathrm{L}^{*k} \\ \Sigma(\mathrm{L}^*) &= \frac{1}{K}\sum_{k=1}^{K} [\mathrm{L}^{*k} - \mathrm{M}(\mathrm{L}^*)] \odot [\mathrm{L}^{*k} - \mathrm{M}(\mathrm{L}^*)] \end{aligned} \tag{9}$$

where $\mathrm{M}(\mathrm{L}^*)$ includes element-wise mean of the matrices. Each value in the variance matrix $\Sigma(\mathrm{L}^*)$ that corresponds to the missing interaction is used as a measure of its uncertainty.

3.2 Positioning and Imputation

After quantifying the uncertainty, we now select where to impute in the given matrix $\mathcal{R}$ by referring to the variance matrix $\Sigma(\mathrm{L}^*)$. We sort all the variance values in $\Sigma(\mathrm{L}^*)$ corresponding to the missing interactions. Then, the positions are determined by imputation rate r in the order of the lowest variance (i.e., the lowest uncertainty) where we define the r as a hyper-parameter that determines what percentage of the total missing cells in $\mathcal{R}$ will be filled.

The values that will be imputed in the chosen positions are computed by the mean of K interaction probabilities drawn from the multinomial distribution. Formally, a user's k^{th} (dense) interaction probability on the items, r_u^{*k}, is obtained from the following multinomial distribution:

$$r_u^{*k} \sim \mathit{Mult}(\mathrm{N}_u,\ l_u^{*k}) \tag{10}$$

where $\mathrm{N}_u = \sum_{i=1}^{I} r_{ui}$ computes the number of items interacted by u and is used as the number of trials, and l_u^{*k} is used as the probability vector of the multinomial distribution. As a result, we have K inferred probability vectors of u's preference towards each item. We then perform the following element-wise operation on the probability vectors:

$$\mathrm{M}(r_u^*) = \frac{1}{K}\sum_{k=1}^{K} r_u^{*k} \tag{11}$$

where each element of the output vector $\mathrm{M}(r_u^*)$ indicates the mean of u's probabilities on an item i. Eventually, we fill in each chosen position (say, (u, i)) by using the value inside the computed vector $\mathrm{M}(r_u^*)$. We denote the final matrix where all the chosen positions are imputed with r imputation rate as $\bar{\mathcal{R}}^r$.

We claim that the imputed values in $\bar{\mathcal{R}}^r$ have strengths in the following aspects. (1) Unlike existing methods that naively determine where to impute, we calculate the element-wise variance values of the user-item probability rates and selectively fill the *reliable* missing cells with low uncertainty (*uncertainty-aware positioning*). (2) In addition, since existing methods impute deterministically, there is a risk that the values will be *biased*, whereas our method is a *multiple imputation* approach that fills in statistically robust element-wise mean values of the user-item interactions.

For the final recommendation, any CF model can be trained to predict the preferences on the unobserved entries in the original matrix $\mathcal{R}$ in its own way by using $\bar{\mathcal{R}}^r$ as training data, instead of using $\mathcal{R}$.

3.3 Implementation Details

We implemented our framework with Python 3.9.6 and Tensorflow 2.5.0. For the sampling-based and dropout-based methods, we tested a different number of size K with $\{30,\ 50,\ 70,\ 100\}$, and, for the ensemble-based method, we tested K with $\{1,\ 5,\ 10,\ 15,\ 20\}$. Additionally, the dropout rate p for the dropout-based method was varied with $\{0.05,\ 0.1,\ 0.15,\ 0.2,\ 0.25\}$. The imputation rate r was examined from 0.05 to 1, increasing by 0.05. For the training of our base model, MultVAE, we mainly refer to its settings presented in the original paper [26]. We set the number of epochs as 200, batch size as 500, learning rate as 0.001, and used the Adam optimizer. The model structure was defined as $\{Input - 600 - 200 - 600 - Output\}$, where the sizes of input and output are equal to the number of items. Moreover, the optimal β value is learned by updating the KL term 20,000 times through KL annealing [26], where the initial value β was set to 1. More details can be found in our code.

4 EXPERIMENTAL SETTINGS

This section describes how we set up our experiments to prove the effectiveness of our UA-MI framework, where we implement three types of uncertainty quantification methods based on sampling, dropout, and ensemble, respectively. Our experiments are designed to answer the following three research questions:

- **RQ1: Effectiveness**. How much more effective is our UA-MI framework compared with several baseline imputation strategies?
- **RQ2: Scalability**. How much additional time does our UA-MI framework require?
- **RQ3: Sensitivity**. How much do the accuracy and execution time of our UA-MI vary depending on the size K? And what is the optimal hyper-parameter setting?

4.1 Dataset

We performed experiments using the following four real-life datasets: Amazon-Book[2] (A-Book, in short), CiteULike-a[3], Epinions[4], and MovieLens (ML)-20M[5]. **A-Book** includes ratings for books by Amazon users. **Epinions** contains ratings on items given by users of a product review site. **ML-20M** contains users' movie ratings.

[2]http://jmcauley.ucsd.edu/data/amazon/links.html
[3]https://citeulike.org/faq/data.adp
[4]https://www.cse.msu.edu/ tangjili/datasetcode/truststudy.htm
[5]https://grouplens.org/datasets/movielens/20m

Table 1: Data statistics.

	A-Book	CiteULike-a	Epinions	ML-20M
Users (#)	46,276	5,551	18,392	136,677
Items (#)	145,051	16,940	23,849	20,108
Interactions (#)	2,448,908	210,447	362,165	9,990,030
Sparsity (%)	0.03%	0.22%	0.08%	0.36%
Valid / Test Users (#)	5,000	500	2,000	10,000

CiteULike-a includes each user's implicit feedbacks on articles through taggings. For each explicit feedback dataset (A-Book, Epinions, and ML-20M), we converted it into an implicit feedback matrix by binarizing four or higher scores to one among the ratings and discarding the rest. We also applied the 20-core setting to A-Book and the 5-core setting to Epinions and ML-20M [12, 26, 42]. Since CiteULike-a is an implicit feedback dataset, we did not perform pre-processing for it. The summary statistics after the pre-processing are provided in Table 1.

4.2 Competing Methods

To highlight the effectiveness of our imputation, we carefully designed several baseline imputation strategies and employed several state-of-the-art CF models for implicit feedback. Then a combination of each imputation strategy and each model becomes our competitor.

4.2.1 Imputation Strategies. We designed the following three baseline imputation frameworks to compare with our imputation framework: *All Missing As Negative* (AMAN) [33], *Random Position Imputation* (RPI), and *Confident Value Imputation* (CVI). **AMAN** is a simple but very widely-used strategy for CF with implicit feedback to solve the class imbalance issue [33]. It imputes all missing values of the matrix as 0. **RPI** is a strategy that randomly selects where to impute at a pre-defined imputation rate. On the other hand, **CVI** first predicts the values of missing entries using any CF model. Then, assuming that higher or lower values are more confident, it imputes the values of missing entries as those confident values at a pre-defined imputation rate [14, 20]. For example, if the imputation rate is set to 10%, 5% of all predicted empty cells are selected from the lowest value and then the other 5% from the highest value.

4.2.2 CF Models. Among the three baseline imputation strategies introduced above, AMAN does not require a CF model for computing imputed values since it fills all missing values with zeros. The other two frameworks, RPI and CVI, require CF models, and for this reason, we employ three popular CF models: *Weighted Regularized Matrix Factorization* (**WRMF**)[6] [13], *Multinomial Variational Autoencoder* (**MultVAE**)[7] [26], and *Light Graph Convolution Network* (**LightGCN**)[8] [12]. These are the state-of-the-art models representing linear latent factor models, deep generative models, and graph neural network models for CF, respectively. We set their hyper-parameters to the default values suggested by the corresponding papers. We do not employ the neighborhood-based methods such as

[6]https://implicit.readthedocs.io/en/latest/als.html
[7]https://github.com/dawenl/vae_cf
[8]https://github.com/kuandeng/LightGCN

AutAI [34] and AdaM [35] since they were found to be less accurate and not scalable [2].

We believe that the *combinations* of each baseline imputation strategy and each employed model can replicate the existing imputation methods[9]. For example, CVI-WRMF can be considered as an implicit feedback version of *l*-Injection [20], which is an extended version of Zero-Injection [14]. In addition, CVI-MultVAE is able to cover the existing generative model based imputation methods such as RAGAN [2] and AR-CF [4], and also able to cover the Autoencoder-based imputation methods such as [22] and [23].

4.3 Evaluation Protocol

Given the training data (i.e., the original matrix $\mathcal{R}$), the compared imputation frameworks except AMAN perform imputation in their respective ways with imputation rate r (for example, CVI adopts one of the three models, WRMF, MultVAE, and LightGCN, and then predicts the values of empty cells. It finally creates $\bar{\mathcal{R}}^r$ by filling some highest and lowest values). Note that imputation is only done on the training data, not on the fold-in sets of validation and test data. After the imputation is completed, all the compared frameworks commonly use MultVAE as a final CF model to perform the top-N recommendation by learning $\bar{\mathcal{R}}^r$.

In the case of the AMAN framework, zeros, which indicate negative preference, are imputed to all empty cells. Therefore, in the AMAN framework, r is always 1.0 which means 100% imputation. For this framework, we not only used MultVAE, but also WRMF and LightGCN to perform the final top-N recommendations. In fact, this produces the same results as running the original MultVAE, WRMF, and LightGCN because they are methods that basically assume AMAN. Therefore, it can be seen that the top-N recommendation results from this provide baseline performances regardless of imputation.

4.4 Metrics

To produce a top-N list, we adopt the settings suggested by the authors of MultVAE [26]. To elaborate, we separated 10,000 users from Amazon-Book, 1,000 users from CiteULike-a, 4,000 users from Epinions, and 20,000 users from ML-20M. Then, we used each half of those users for validation and another half for testing. The interactions left by the remaining users are used as training data. Among the interactions from the separated users, 80% is randomly selected and employed as a fold-in set which is used as input data for the models during the testing phase, and the remaining 20% is used as a fold-out set, which is used as test data [26].

To measure the top-N recommendation performance, we used two widely-used metrics: Recall@N and Normalized Discounted Cumulative Gain (NDCG)@N. They compare the list of the recommended items for each user u in the fold-out set and the fold-out items that have interactions with u, denoted as $\mathcal{I}_{fo}(u)$, which are the ground truth items. Formally, Recall@N and NDCG@N are calculated as follows [26]:

$$\begin{aligned} \text{Recall@}N &= \frac{\sum_{n=1}^{N} \mathbb{I}[\mathcal{I}_n(u) \in \mathcal{I}_{fo}(u)]}{min(N, \mid \mathcal{I}_{fo}(u) \mid)} \\ \text{NDCG@}N &= \sum_{n=1}^{N} \frac{2^{\mathbb{I}[\mathcal{I}_n(u) \in \mathcal{I}_{fo}(u)]} - 1}{\log(n+1)} \end{aligned} \tag{12}$$

where $\mathcal{I}_n(u)$ indicates the item in n^{th} rank of u's recommendation list. Recall@N uses an indicator function $\mathbb{I}[\cdot]$ to calculate the ratio of how many items up to the N^{th} rank are included in the ground truth items. Unlike Recall@N, NDCG@N gives larger weights for items that match the higher rank items up to the N^{th} rank.

5 RESULTS AND ANALYSES

This section reports and analyzes the experimental results.

5.1 Performance Comparisons (RQ1)

We set the number of recommended items N to various range, such as {1, 5, 10, 50}, to evaluate the performance in diverse scenarios, from recommending a small number of items to providing a large list of recommended items. Tables 2 and 3 show the top-N recommendation accuracies of each imputation framework for each dataset. Note that the result of NDCG@1 is not shown because it is the same as Recall@1. For fair comparison, we tested various imputation rates for each method, increasing from 5% to 100% (incrementing by 5%), and the best accuracy and the corresponding imputation rate (denoted as I.R. in the tables) are reported for each method (except for the AMAN strategy).

First, we focused on how the accuracy with AMAN-MultVAE improves with each method, because MultVAE is commonly used as a final recommender and thus it well-highlights the effectiveness of imputation. From this perspective, the proposed UA-MI framework consistently and significantly improved the performance of AMAN-MultVAE in all cases (12.8% on average and 43.5% at most), while the RPI and CVI methods marginally improved the performance and even showed worse performances in some cases. Considering that the imputation strategy is the only difference and the rest remains the same, we notice that our uncertainty-aware imputation strategy is much more effective than the alternatives.

Next, we compared our framework with the best competitor for each case, whose result is underlined in the tables. We observed meaningful improvements as well: our UA-MI exhibited 8.8% higher accuracy than the best competing methods on average. The best improvement was achieved on CiteULike-a with respect to Recall@1 (15.8%). In order to confirm that the best results achieved by our framework are statistically significant, we performed a paired t-test on the results and confirmed their statistical significance. In the tables, the sign ** indicates statistical significance at $\alpha = 0.01$ level ($p < 0.01$). As shown in the tables, UA-MI with ensemble showed especially notable performance among our three methods. However, the training cost of the ensemble-based method is relatively more expensive than the other methods, which will be discussed in the next two subsections.

We further analyzed the performance of our UA-MI with ensemble depending on the different imputation rate. Figure (3) shows the change of its accuracy as the imputation rate increases on each

[9] Most imputation methods are tailored to deal with explicit rating matrix, so converting them to work in an implicit matrix and tuning their hyper-parameters from the scratch was infeasible.

Table 2: Comparison results on Amazon-Book and CiteULike-a. The best accuracy for each case is written in bold. The sign ** ($p < 0.01$) is used to remark the statistical significance of the best accuracy. Relative improvement over the best competitor for each case is shown at the bottom of the table.

		Amazon-Book							CiteULike-a						
		I.R.	Recall@N				NDCG@N			I.R.	Recall@N				NDCG@N		
			$N=1$	$N=5$	$N=10$	$N=50$	$N=5$	$N=10$	$N=50$		$N=1$	$N=5$	$N=10$	$N=50$	$N=5$	$N=10$	$N=50$
AMAN	WRMF	100%	.0447	.0382	.0466	.1225	.0440	.0442	.0716	100%	.1430	.1501	.1671	.2988	.1437	.1419	.1850
	MultVAE	100%	.0624	.0509	.0608	.1391	.0528	.0570	.0874	100%	.1720	.1781	.1981	.3435	.1717	.1789	.2289
	LightGCN	100%	.0617	.0510	.0609	.1390	.0531	.0570	.0873	100%	.1680	.1792	.1974	.3454	.1721	.1776	.2290
RPI	WRMF	50%	.0699	.0515	.0607	.1389	.0540	.0581	.0885	70%	.1830	.1777	.1962	.3443	.1728	.1799	.2331
	MultVAE	70%	.0709	.0496	.0620	.1375	.0537	.0587	.0879	10%	.1800	.1796	.1992	.3419	.1726	.1825	.2293
	LightGCN	50%	.0691	.0504	.0608	.1384	.0524	.0576	.0874	60%	.1810	.1777	.1981	.3428	.1742	.1794	.2290
CVI	WRMF	40%	.0710	.0500	.0607	.1392	.0525	.0578	.0885	30%	.1790	.1787	.1975	.3433	.1745	.1808	.2295
	MultVAE	10%	.0716	.0505	.0619	.1397	.0524	.0587	.0879	10%	.1750	.1797	.1985	.3415	.1754	.1834	.2313
	LightGCN	20%	.0716	.0501	.0616	.1392	.0527	.0582	.0880	30%	.1820	.1796	.1926	.3423	.1739	.1805	.2300
UA-MI	Sampling	30%	.0729	.0528	.0632	.1408	.0548	.0591	.0891	10%	.1980	.1851	.2094	.3451	.1779	.1871	.2362
	Dropout	10%	**.0756****	.0515	.0631	.1489	.0552	.0596	.0906	10%	.2020	.1905	.2147	.3525	.1847	.1937	.2403
	Ensemble	15%	.0746	**.0574****	**.0664****	**.1574****	**.0607****	**.0649****	**.0965****	10%	**.2120****	**.1962****	**.2167****	**.3675****	**.1911****	**.2018****	**.2497****
Improvement			5.6%	11.5%	7.1%	12.7%	12.4%	10.6%	9.0%		15.8%	9.2%	8.8%	6.7%	9.0%	10.0%	7.1%

Table 3: Comparison results on Epinions and ML-20M. The best accuracy for each case is written in bold. The sign ** ($p < 0.01$) is used to remark the statistical significance of the best accuracy. Relative improvement over the best competitor for each case is shown at the bottom of the table.

		Epinions							ML-20M						
		I.R.	Recall@N				NDCG@N			I.R.	Recall@N				NDCG@N		
			$N=1$	$N=5$	$N=10$	$N=50$	$N=5$	$N=10$	$N=50$		$N=1$	$N=5$	$N=10$	$N=50$	$N=5$	$N=10$	$N=50$
AMAN	WRMF	100%	.0183	.0287	.0416	.1069	.0216	.0297	.0507	100%	.3262	.2746	.2987	.4974	.2800	.2855	.3425
	MultVAE	100%	.0230	.0343	.0507	.1306	.0286	.0349	.0578	100%	.3644	.3101	.3283	.5356	.3163	.3143	.3808
	LightGCN	100%	.0210	.0342	.0512	.1297	.0281	.0351	.0560	100%	.3662	.3081	.3288	.5354	.3171	.3151	.3811
RPI	WRMF	40%	.0256	.0342	.0535	.1337	.0283	.0357	.0579	60%	.3683	.3117	.3282	.5354	.3182	.3162	.3819
	MultVAE	30%	.0231	.0344	.0517	.1342	.0292	.0365	.0583	10%	.3681	.3118	.3278	.5354	.3185	.3159	.3822
	LightGCN	70%	.0237	.0354	.0524	.1317	.0285	.0361	.0584	10%	.3654	.3116	.3280	.5347	.3185	.3160	.3819
CVI	WRMF	20%	.0305	.0349	.0516	.1354	.0297	.0360	.0579	10%	.3679	.3113	.3281	.5358	.3183	.3155	.3817
	MultVAE	40%	.0315	.0350	.0539	.1290	.0295	.0364	.0583	10%	.3674	.3117	.3284	.5351	.3180	.3159	.3820
	LightGCN	10%	.0297	.0349	.0526	.1359	.0291	.0365	.0578	40%	.3682	.3117	.3276	.5353	.3181	.3160	.3821
UA-MI	Sampling	10%	.0281	.0362	.0542	.1395	.0310	.0371	.0598	30%	.3695	.3126	.3301	.5378	.3190	.3172	.3832
	Dropout	15%	.0302	.0374	.0559	.1463	.0322	.0385	.0619	15%	.3786	.3216	.3424	.5425	.3307	.3265	.3989
	Ensemble	15%	**.0330****	**.0386****	**.0579****	**.1525****	**.0333****	**.0394****	**.0633****	20%	**.3877****	**.3352****	**.3523****	**.5653****	**.3436****	**.3404****	**.4143****
Improvement			4.8%	9.0%	7.4%	12.2%	12.1%	7.9%	8.4%		5.3%	7.5%	7.3%	5.5%	7.9%	7.7%	8.4%

dataset. We observed a common behavior that the accuracy increases until the imputation ratio reaches to 10% ∼ 20%, and then consistently drops afterwards. The observation confirms that carefully selecting a small portion of (but highly confident) empty cells to impute is much helpful in alleviating the sparsity issue rather than just increasing imputed empty cells which has high uncertainty causing performance degradation.

We believe that UA-MI's outperformance can be attributed to the following factors. First, our UA-MI tried to make the imputation more *reliable* by carefully selecting the imputation position. CVI, one of our baselines, also considered the extreme values to be confident positions and filled those predicted positions first, but it failed to show consistent outperformance, and in some cases, the performance rather worsened. Such values derived from a deterministic model may be insufficient information to determine the positions. On the other hand, UA-MI filled the values with confidence in positions where there was low variance, i.e. low epistemic uncertainty. As a result, our strategy provides more reliable values with the given matrix, which leads to consistent performance improvement in all cases.

Moreover, unlike existing imputation techniques, UA-MI pursued multiple imputation to fill in *robust* values. It has been shown through our experimental results that this way is more effective than RPI and CVI methods following single imputation, which is

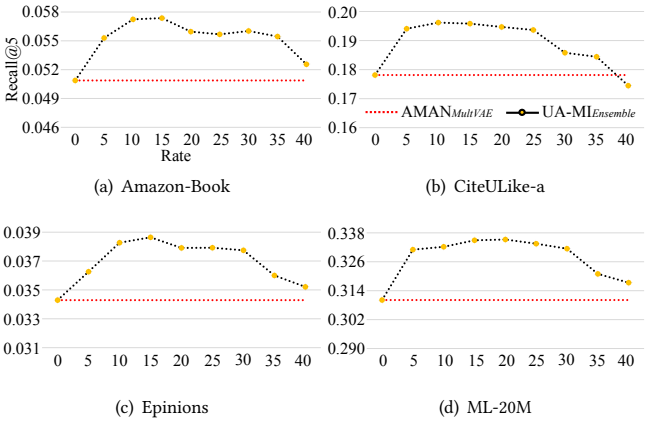


Figure 3: Accuracy of UA-MI with ensemble depending on imputation rate.

based on a deterministic model that may lead to bias where there is a risk that imputed values deviate from the true values [27]. On the other hand, our UA-MI tries to obtain the mean of multiple interaction probabilities derived by the multinomial distribution parameterized by the stochastic outputs of the uncertainty quantification methods.

As a result, our suggested work solves the data sparsity issue more effectively by performing reliable and robust imputation. This has resulted in better recommendation performance compared to several existing imputation strategies.

5.2 Scalability Analysis (RQ2)

We analyzed overall training and inference time consumed by our UA-MI framework, because it involves additional computations such as generation of K predictive rates for each user and computation of the element-wise variance and mean for the positioning and imputation, which are not present in the existing imputation methods. We ran all our experiments on a single machine equipped with an i9 11900K Intel CPU, 128GB RAM, and NVIDIA GeForce RTX3090 GPU. Table 4 reports the training or running time for each step. The size of the given matrix $\mathcal{R}$ for each data is also provided to help readers. The training times of the three baseline models (WRMF, MultVAE, and LightGCN) are also reported for comparison.

Among the three methods, the sampling-based and dropout-based methods took about 3 times longer than just training the MultVAE model on the larger datasets (e.g., Amazon-Book and ML-20M), and took about 1.3 times more time than MultVAE on smaller datasets (e.g., CiteULike-a and Epinions). However, the ensemble-based method consumed about 11 times longer than MultVAE because it requires K times of training to construct an explicit ensemble of MultVAE models (we set K as 10 here), while the other two methods need only one trained model to generate K sample matrices on the inference phase. We believe that parallel training of the ensemble members by using multiple servers will be able to relieve such long training time significantly.

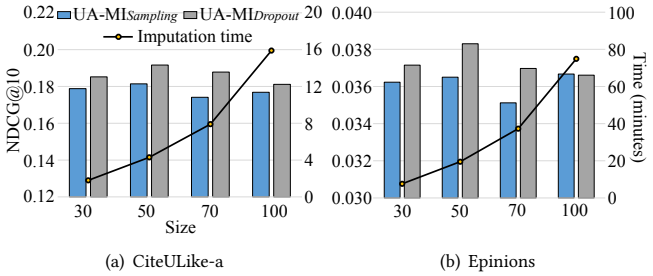


Figure 4: Accuracy and imputation time with regard to sample size K.

5.3 Hyper-parameter Analysis (RQ3)

In this section, we analyzed the speed-accuracy trade-off in our framework according to the changes of two key hyper-parameters: the sample size K and the dropout rate p.

5.3.1 Sample Size K. In our UA-MI equipped with sampling and dropout, K influences on their recommendation accuracy and the imputation time; higher K generates a larger number of sample matrices that help in quantifying better uncertainty estimates, but makes the imputation time longer. In terms of Central Limit Theorem (CLT) [16], the sample size should be at least 30 for a reasonable approximation. Therefore, we tested the following values: {30, 50, 70, 100} for the sample size K.

Figure 4 reports the recommendation accuracy achieved and the total imputation time spent by our UA-MI with sampling and dropout on CiteULike-a and Epinions with regard to NDCG@10 (this is due to the space limitation; other cases showed very similar trend). We observed that the recommendation performance increases as the sample size reaches 50, and after then, it shows no meaningful change and even degrades. From the viewpoint of time consumption, the total imputation time linearly increases as the sample size increases. Through these results, we suggest setting K around 30 to 50 that provides both satisfactory recommendation accuracy and reasonable imputation time.

We performed similar experiments on our UA-MI equipped with ensemble. Table 5 shows the changes of recommendation performance and imputation time according to the ensemble size K. Since this method is not related with Central Limit Theorem, we tested different set of values as follows: {1, 5, 10, 15, 20} for the ensemble size K following prior work [19]. Similarly, we observed the performance improvement until the ensemble size reaches 10, and then it shows no meaningful change. However, the training time increases in proportion to the number of ensemble members. Hence, we suggest setting K around 10 for a fine balance between the model training time and accuracy.

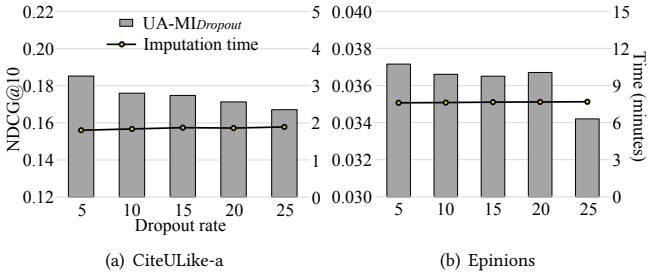
5.3.2 Dropout Rate p. Finally, we discuss the impact of the dropout rate p on the accuracy and overall running time of our UA-MI with dropout. According to Yarin Gal's study [8], in the case of a standard neural network model with 4 hidden-layers and 1024 units, a dropout rate of 25% provides reasonable predictions. However, compared to the network used in the study, MultVAE used in our

Table 4: Execution time analysis (minutes).

		Amazon-Book	CiteULike-a	Epinions	ML-20M
Size of $\mathcal{R}$		6,712M	94M	438M	2,748M
Baselines	WRMF	4.2	0.1	0.5	1.2
	MultVAE	118.6	4.2	14.7	44.4
	LightGCN	135.9	4.8	16.9	50.8
Sample generation	Sampling (size $K = 30$)	$118.6 \times 1 + 75.5$	$4.2 \times 1 + 0.6$	$14.7 \times 1 + 2.9$	$44.4 \times 1 + 22.9$
	Dropout (size $K = 30$)	$118.6 \times 1 + 82.6$	$4.2 \times 1 + 0.7$	$14.7 \times 1 + 3.0$	$44.4 \times 1 + 23.7$
	Ensemble (size $K = 10$)	$118.6 \times 10 + 33.6$	$4.2 \times 10 + 0.3$	$14.7 \times 10 + 1.4$	$44.4 \times 10 + 11.4$
Positioning & imputation (rate = 20%)	Sampling	134.2	1.2	4.8	40.6
	Dropout	135.7	1.1	5.2	41.4
	Ensemble	134.0	1.1	4.7	39.7
Total	Sampling	328.3	6.0	22.4	107.9
	Dropout	336.9	6.0	22.9	109.5
	Ensemble	1,353.6	43.4	153.1	495.1

Table 5: Ensemble size test: accuracy (N = 10) V.S. total training & imputation time (minutes).

	CiteULike-a			Epinions		
Size	Recall	NDCG	Time	Recall	NDCG	Time
1	.1890	.1794	4.2	.0511	.0352	14.9
5	.1963	.1838	21.1	.0537	.0370	74.4
10	**.2093**	**.1923**	42.2	**.0557**	**.0387**	148.8
15	.2036	.1917	63.4	.0554	.0381	223.1
20	.2084	.1883	84.5	.0542	.0385	297.5



(a) CiteULike-a (b) Epinions

Figure 5: Accuracy and imputation time depending on dropout rate.

work is shallower and has an extremely high input and output dimensions, so it is necessary to have a lower dropout rate. As a result, we tested a dropout rate ranging from 5% to 25% and increasing it every 5%. We fix K as 30 and the imputation rate as 20% for this experiment. Figure 5 shows the top-N recommendation accuracy and imputation time according to the different dropout rate. We observed that the recommendation accuracy was highest when the dropout rate was 5%, and as the dropout rate increases the performance tends to get worse. However, the imputation time remains consistent regardless of the dropout rate. Based on this result, we fixed the dropout rate to 5% for all the experiments.

6 CONCLUSION

Throughout the paper, we discussed how to effectively solve the data sparsity issue. We proposed a novel imputation framework, named *Uncertainty-Aware Multiple Imputation* (UA-MI), which exploits the element-wise mean and variance values obtained by estimating uncertainty in missing user-item interactions. We explore three ways to quantify uncertainty based on the given user-item matrix: one performs sampling in the latent space through the reparameterization, another applies dropout to the trained network, and the other constructs an explicit ensemble of networks. Then we took a strategy to selectively impute positions with low variance by using the element-wise variance values as information indicating uncertainty. Moreover, by filling in the mean of multiple interaction probabilities, the imputation is more reliable and robust. Our proposed framework is successfully implemented on top of the MultVAE [26] model. From the experiments, our framework outperformed other imputation methods, and in particular, the ensemble-based method showed remarkable accuracy.

ACKNOWLEDGMENTS

This work was partly supported by (1) Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-01373,Artificial Intelligence Graduate School Program(Hanyang University)), (2) the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2021R1A2C1094863), and (3) the Bio & Medical Technology Development Program of the National Research Foundation (NRF) funded by the Korean government (MSIT) (No. NRF-2021M3E5D2A01021156).

REFERENCES

[1] Paul D Allison. 2001. *Missing data*. Sage publications.

[2] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jaeho Choi. 2019. Rating augmentation with generative adversarial networks towards accurate collaborative filtering. In *The World Wide Web Conference*. 2616–2622.

[3] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. CFGAN: A generic collaborative filtering framework based on generative adversarial networks. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 137–146.

[4] Dong-Kyu Chae, Jihoo Kim, Duen Horng Chau, and Sang-Wook Kim. 2020. AR-CF: Augmenting virtual users and items in collaborative filtering for addressing cold-start problems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1251–1260.

[5] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. 39–46.

[6] Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908* (2016).

[7] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *Advances in Neural Information Processing Systems* (2019).

[8] Yarin Gal. 2016. Uncertainty in deep learning. (2016).

[9] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. PMLR, 1050–1059.

[10] MA Ganaie, Minghui Hu, et al. 2021. Ensemble deep learning: A review. *arXiv preprint arXiv:2104.02395* (2021).

[11] Alex Graves. 2011. Practical variational inference for neural networks. *Advances in neural information processing systems* 24 (2011).

[12] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[14] Won-Seok Hwang, Juan Parc, Sang-Wook Kim, Jongwuk Lee, and Dongwon Lee. 2016. "Told you I didn't like it": Exploiting uninteresting items for effective collaborative filtering. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 349–360.

[15] Niels Bruun Ipsen, Pierre-Alexandre Mattei, and Jes Frellsen. 2021. Not-MIWAE: Deep Generative Modelling with Missing not at Random Data. In *International Conference on Learning Representations*.

[16] Mohammad Rafiqul Islam. 2018. Sample size and its role in Central Limit Theorem (CLT). *Computational and Applied Mathematics Journal* 4, 1 (2018), 1–7.

[17] Phimmarin Keerin, Werasak Kurutach, and Tossapon Boongoen. 2012. Cluster-based KNN missing value imputation for DNA microarray data. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 445–450.

[18] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[19] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems* 30 (2017).

[20] Jongwuk Lee, Won-Seok Hwang, Juan Parc, Youngnam Lee, Sang-Wook Kim, and Dongwon Lee. 2017. l-Injection: toward effective collaborative filtering using uninteresting items. *IEEE transactions on knowledge and data engineering* 31, 1 (2017), 3–16.

[21] Jongwuk Lee, Dongwon Lee, Yeon-Chang Lee, Won-Seok Hwang, and Sang-Wook Kim. 2016. Improving the accuracy of top-n recommendation using a preference model. *Information Sciences* 348 (2016), 290–304.

[22] Jae-woong Lee and Jongwuk Lee. 2017. IDEA: Imputation-boosted denoising autoencoder for collaborative filtering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2143–2146.

[23] Youngnam Lee, Sang-Wook Kim, Sunju Park, and Xing Xie. 2018. How to impute missing ratings? Claims, solution, and its application to collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*. 783–792.

[24] Yeon-Chang Lee, Sang-Wook Kim, and Dongwon Lee. 2018. gOCCF: Graph-theoretic one-class collaborative filtering based on uninteresting items. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[25] Steven Cheng-Xian Li, Bo Jiang, and Benjamin Marlin. 2018. MisGAN: Learning from Incomplete Data with Generative Adversarial Networks. In *International Conference on Learning Representations*.

[26] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*. 689–698.

[27] Roderick JA Little and Donald B Rubin. 2019. *Statistical analysis with missing data*. Vol. 793. John Wiley & Sons.

[28] Chao Ma and Cheng Zhang. 2021. Identifiable Generative Models for Missing Not at Random Data Imputation. *Advances in Neural Information Processing Systems* 34 (2021).

[29] Hao Ma, Irwin King, and Michael R Lyu. 2007. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 39–46.

[30] Pierre-Alexandre Mattei and Jes Frellsen. 2019. MIWAE: Deep generative modelling and imputation of incomplete data sets. In *International Conference on Machine Learning*. PMLR, 4413–4423.

[31] Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. 2020. Handling incomplete heterogeneous data using vaes. *Pattern Recognition* 107 (2020), 107501.

[32] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems* 32 (2019).

[33] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 502–511.

[34] Yongli Ren, Gang Li, Jun Zhang, and Wanlei Zhou. 2012. The efficient imputation method for neighborhood-based collaborative filtering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 684–693.

[35] Yongli Ren, Gang Li, Jun Zhang, and Wanlei Zhou. 2013. AdaM: Adaptive-maximum imputation for neighborhood-based collaborative filtering. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 628–635.

[36] Aude Sportisse, Claire Boyer, and Julie Josse. 2020. Estimation and imputation in probabilistic principal component analysis with missing not at random data. *Advances in Neural Information Processing Systems* 33 (2020), 7067–7077.

[37] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[38] Harald Steck. 2010. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 713–722.

[39] Daniel J Stekhoven and Peter Bühlmann. 2012. MissForest—non-parametric missing value imputation for mixed-type data. *Bioinformatics* 28, 1 (2012), 112–118.

[40] Steven K Thompson. 2012. *Sampling*. Vol. 755. John Wiley & Sons.

[41] Menghan Wang, Mingming Gong, Xiaolin Zheng, and Kun Zhang. 2018. Modeling dynamic missingness of implicit feedback for recommendation. *Advances in neural information processing systems* 31 (2018), 6669.

[42] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[43] Jinsung Yoon, James Jordon, and Mihaela Schaar. 2018. GAIN: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*. PMLR, 5689–5698.

[44] Seongwook Yoon and Sanghoon Sull. 2020. GAMIN: generative adversarial multiple imputation network for highly missing data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8456–8464.