Fast Variational AutoEncoder with Inverted Multi-Index for Collaborative Filtering

Jin Chen†
University of Electronic Science and Technology of China, University of Science and Technology of China
chenjin@std.uestc.edu.cn

Defu Lian†
University of Science and Technology of China
liandefu@ustc.edu.cn

Binbin Jin
Huawei Cloud Computing Technologies Co., Ltd.
jinbinbin1@huawei.com

Xu Huang
University of Science and Technology of China
gwjiang@mail.ustc.edu.cn

Kai Zheng
University of Electronic Science and Technology of China
zhengkai@uestc.edu.cn

Enhong Chen
University of Science and Technology of China
cheneh@ustc.edu.cn

ABSTRACT
Variational AutoEncoder (VAE) has been extended as a representative nonlinear method for collaborative filtering. However, the bottleneck of VAE lies in the softmax computation over all items, such that it takes linear costs in the number of items to compute the loss and gradient for optimization. This hinders the practical use due to millions of items in real-world scenarios. Importance sampling is an effective approximation method, based on which the sampled softmax has been derived. However, existing methods usually exploit the uniform or popularity sampler as proposal distributions, leading to a large bias of gradient estimation. To this end, we propose to decompose the inner-product-based softmax probability based on the inverted multi-index, leading to sublinear-time and highly accurate sampling. Based on the proposed proposals, we develop a fast Variational AutoEncoder (FastVAE) for collaborative filtering. FastVAE can outperform the state-of-the-art baselines in terms of both sampling quality and efficiency according to the experiments on three real-world datasets.

CCS CONCEPTS
• Information systems → Recommender systems.

1 INTRODUCTION
Recommendation techniques play a role in information filtering to address the information overload in the era of big data. After decades of development, recommendation techniques have shifted from the latent linear models to deep non-linear models for modeling side features and feature interactions among sparse features. Variational AutoEncoder [18] has been extended as a representational nonlinear method (Mult-VAE) for recommendation [21], and received much attention among the recommender system community in recent years [30, 34, 35, 40, 46]. Mult-VAE encodes each user’s observed data with a Gaussian-distributed latent factor and decodes it to a probability distribution over all items, which is assumed a softmax of the inner-product-based logits. Mult-VAE then exploits multinomial likelihood as the objective function for optimization, which has been proved to be more tailored for implicit feedback than the Gaussian and logistic likelihood. However, the bottleneck of Mult-VAE lies in the log-partition function over the logits of all items in the multinomial likelihood. The time to compute the loss and gradient in each training step grows linearly with the number of items. When there are an extremely large number of items, the training of Mult-VAE is time-consuming, making it impractical in real recommendation scenarios. To address this problem, self-normalized importance sampling is used for approximation [5, 14] since the exact gradient involves computing expectation with respect to the softmax distribution. The approximation of the exact gradient leads to the efficient sampled softmax, but it does not converge to the same loss as the softmax. The only way to eliminate the bias is to treat the softmax distribution as the proposal distribution, but it is not efficient.

In spite of the well-known importance of a good proposal, many existing methods still often use simple and static distributions, like uniform or popularity-based distribution [22]. The problem of these proposals lies in large divergence from the softmax distribution, so that they need a large number of samples to achieve a low bias of gradient. The recent important method is to use quadratic kernel-based distributions [6] as the proposal, which are not only closer to the softmax distribution, but also efficient to sample from. However, the quadratic kernel is not always a good approximation
of the softmax distribution, and it suffers from a large memory footprint due to the feature mapping of the quadratic kernel.

Recently maximum inner product search (MIPs) algorithms have been widely used for fast top-k recommendation with low accuracy degradation [26, 31, 36], but they always return the same results to the same query so that they cannot be directly applied for item sampling. On this account, the MIPs indexes have been constructed over the randomly perturbed database for probabilistic inference in log-linear models and become a feasible solution to sample from the softmax distribution [27]. However, this not only increases both data dimension and sample size, but also makes the samples correlated. Moreover, this may also require to rebuild the MIPs index from scratch once the model gets updated, which has a significant impact on the training efficiency. Therefore, it is necessary to design sampling algorithms tailored for the MIPs indexes.

To this end, based on the popular MIPs index – inverted multi-index [3], we propose a series of proposal distributions, from which items can be efficiently yet independently sampled, to approximate the softmax distribution. The basic idea is to decompose item sampling into multiple stages. In each except the last stage, only a cluster index is sampled given the previously sampled clusters. In the last stage, items being simultaneously assigned to these sampled clusters are sampled according to uniform, popularity, or residual softmax distribution. Since there are a few items left, items are sampled from these approximated distributions in sublinear or even constant time. In some cases, the decomposed sampling is as exact as sampling from softmax, such that the quality of sampled items can be guaranteed. These samplers are then adopted to efficiently train Variational AutoEncoder for collaborative filtering (FastVAE for short). FastVAE is evaluated extensively on three real-world datasets, demonstrating that FastVAE outperforms the state-of-the-art baselines in terms of sampling quality and efficiency.

The contributions can be summarized as follows:

- To the best of our knowledge, we discover high-quality approximated softmax distributions for the first time, by decomposing the softmax probability based on the inverted multi-index.
- We design an efficient sampling process for these approximate softmax distributions, from which items can be independently sampled in sublinear or even constant time. These samplers are applied for developing the fast Variational AutoEncoder.
- We evaluate extensively the proposed algorithms on four real-world datasets, demonstrating that FastVAE performs at least as well as VAE for recommendation. Moreover, the proposed samplers are highly accurate compared to existing sampling methods, and perform sampling with high efficiency.

2 RELATED WORK
We mainly survey related work about efficient softmax, negative sampling and maximum inner product search. Please refer to the survey [44, 47] for deep learning-based recommender systems, and the survey [1] for classical recommendation algorithms.

2.1 Efficient Softmax Training
Sampled softmax improves training based on self-normalized importance sampling [5] with a mixture proposal of unigram, bigram

and trigrams. Hierarchical softmax [25] uses the tree structure and lightRNN [19] uses the table to decompose the softmax probability such that the probability can be quickly computed. Noise-Contrastive Estimation [12] uses nonlinear logistic regression to distinguish the observed data from some artificially generated noise, and has been successfully used for language modeling by treating the unigram distribution as the noise distribution [24]. Sphere softmax [8, 41] replaces the exponential function with a quadratic function, allowing exact yet efficient gradient computation.

2.2 Negative Sampling in RS
Dynamic negative sampling (DNS) [49] draws a set of negative samples from the uniform distribution and then picks the item with the largest prediction score. Similar to DNS, the self-adversarial negative sampling [39] draws negative samples from the uniform distribution but treats the sampling probability as their weights. Kernel-based sampling [6] picks samples proportionally to a quadratic kernel, making it fast to compute the partition function in the kernel space and to sample entries in a divide and conquer way. Locality Sensitive Hashing (LSH) over randomly perturbed databases enable sublinear time sampling [27] and LSH itself can generate correlated and unnormalized samples [38], which allows efficient estimation of the partition function. Self-Contrast Estimator [9] copied the model and used it as the noise distribution after every step of learning. Generative Adversarial Networks [16, 43] directly learn the noise distribution via the generator networks.

2.3 Maximum Inner Product Search
The MIPS problem is challenging since the inner product violates the basic axioms of a metric, such as a triangle inequality and non-negative. Some methods try to transform MIPS to nearest neighbor search (NNS) approximately [36] or exactly [4, 28]. The key idea of the transformation lies in augmenting database vectors to ensure them an (nearly) identical norm, since MIPS is equivalent to NNS when the database vectors are of the same norm. After the transformation, a bulk of algorithms can be applied for ANN search, such as Euclidean Locality-Sensitive Hashing [7], Signed Random Projection [37] and PCA-Tree [4]. Several existing work also studies quantization-based MIPS by exploiting additive nature of inner product, such as additive quantization [6], composite quantization [48] and even extends PQ from the Euclidean distance to the inner product [10]. Similarly, the graph-based index has been extended to MIPS [26], achieving remarkable performance.

3 PRELIMINARIES
3.1 Multi-VAE
Assuming recommender models operate on N users’ implicit behavior (e.g. click or view) over M items, where each user u is represented by the observed data y_u of dimension M. Each entry y_{ui} indicates an interaction record to the item i, where y_{ui} = 0 indicates no interaction. Multi-VAE [21] is a representative nonlinear recommender method for modeling such implicit data. It particularly encodes y_u with a Gaussian-distributed latent factor z_u and then decodes it to \hat{y}_u, a probability distribution over all items. The objective is to
maximize the evidence lower bound (ELBO):

\[
\mathcal{L}(u) = \mathbb{E}_{z_u} - q_\phi(y_u)[\log p_\theta(y_u|z_u)] - KL(q_\phi(z_u|y_u)||p(z_u)),
\]

where \( q_\phi(z_u|y_u) \) is the variational posterior with parameters \( \phi \) to approximate the true posterior \( p(z_u|y_u) \). \( q_\phi(z_u|y_u) \) is generally assumed to follow the Gaussian-distribution whose mean and variance are estimated by the encoder of Mult-VAE. That is, \( z_u \sim N(\text{MLP}_\mu(y_u; \phi), \text{diag}(\text{MLP}_\sigma(y_u; \phi))) \), where MLP\(_\mu\) and MLP\(_\sigma\) denote multilayer perceptrons (MLPs). \( p(z_u) \) is the prior Gaussian distribution \( N(0, I) \). \( p_\theta(y_u|z_u) \) is the generative distribution conditioned on \( z_u \). The observed data \( y_u \) is assumed to be drawn from the multinomial distribution, which motivates the widely-used multinomial log-likelihood in Eq. (1):

\[
\log p_\theta(y_u|z_u) = \sum_{i \in I} \log p_\theta(y_{ui}|z_u) = \sum_{i \in I} \log \frac{\exp(z_{ui}^T q_i)}{\sum_{j \in I} \exp(z_{uj}^T q_j)},
\]

where \( I \) is the set of all items, \( z_u \) and \( q_i \) is the latent representation of user \( u \) and item \( i \), respectively.

### 3.2 Sampled Softmax

Optimizing the multinomial log-likelihood of Mult-VAE is time-consuming due to the log-partition function over the logits of all items. Given one user’s inner-product logit \( o_i \) for item \( i \), the preference probability for the item \( i \) is calculated by \( P(i) = \frac{\exp(o_i)}{\sum_{j \in I} \exp(o_j)} \).

Denoting model parameters by \( \theta \), the gradient of the log-likelihood loss is computed as \( \nabla_\theta \log P(i) = \nabla_\theta \log q_i - \nabla_\theta \log p_\theta \). Therefore, it takes linear costs in the number of items to compute the loss and gradient. This hinders the multinomial likelihood from the practical use in the real-world scenario with millions of items.

Sampled softmax is one popular approximation approach for log-softmax based on the self-normalized importance sampling. Since the second term of \( \nabla_\theta \log P(i) \) involves an expectation, it can be approximated by a small set of samples \( \Phi \) from a proposal \( Q \). This can be equivalently achieved by adjusting \( o'_j = o_j - \log Q(j), \forall j \in \{ i \} \cup \Phi \) and computing the softmax over \( \{ i \} \cup \Phi \) (i.e. sampled softmax). Obviously, the computational cost for loss and gradient is significantly reduced. However, to guarantee the gradient of the sampled softmax unbiased, Bengio and Senécal [5] showed that the proposal \( Q \) should be equivalent to the softmax distribution \( P \). Since it is computationally expensive to sample from the softmax distribution, many existing methods simply use the uniform or popularity-based proposal. One recent important method [6] proposed to adopt quadratic kernel-based distributions as the proposal. However, it is not always a good approximation of the softmax distribution and suffers from a large memory footprint. Thus, it is necessary to seek a more accurate and flexible sampler.

### 4 Exact Sampling with Inverted Multi-Index

As demonstrated, to guarantee the gradient of the sampled softmax unbiased, it is necessary to draw candidate items from the softmax probability with the inner-product logits:

\[
Q(y_u|z_u) = \frac{\exp(z_{ui}^T q_i)}{\sum_{j \in I} \exp(z_{uj}^T q_j)}.
\]

To achieve this goal, inspired by the popular inverted multi-index [3, 11, 17] for the approximate maximum inner product search (MIPS) and nearest neighbor search (ANNs), we provide a new way for sampling items from multiple multinomial distributions in order. Technical details will be elaborated below.

The inverted multi-index [3] generalizes the inverted index with multiple codebook quantization, such as product quantization [15] and additive quantization [2]. Below we demonstrate with product quantization, whose basic idea is to independently quantize multiple subvectors of indexed vectors. Formally, suppose \( q \in \mathbb{R}^D \) is an item vector, we first evenly split it into \( m \) distinct subvectors (i.e., \( q = q^1 \oplus q^2 \oplus \cdots \oplus q^m \) where \( \oplus \) is the concatenation). Then, each subvector \( q^l \) is mapped to an element of a fixed-size vector set by a quantizer \( f_l : f_l(q^l) \in C^l = \{ c^l_k | k \in \{ 1, \ldots, K \} \} \), where \( C^l \) is the vector set (i.e. codebook) of size \( K \) in the \( l \)-th subspace and the element \( c^l_k \) is called a codeword. Therefore, \( q \) is mapped as follows:

\[
q \rightarrow f_1(q^1) \oplus f_2(q^2) \oplus \cdots \oplus f_m(q^m) = c_1^k \oplus c_2^k \oplus \cdots \oplus c_m^k.
\]

where \( k_l (1 \leq l \leq m) \) is the index of the mapped codeword from \( q^l \). The codewords of each codebook can be simply determined by the K-means clustering [15], where the \( l \)-th subvectors of all items’ vectors are grouped into \( K \) clusters. In the following, we demonstrate sampling with 2 codebooks for simplicity (i.e. \( m = 2 \)), which is the default option of inverted multi-index.

With the quantization, each item vector is only approximated by the concatenation of codewords. To eliminate the difference between item vector and its approximation, we add a residual vector \( \hat{q} = q - c_1^k \oplus c_2^k \) to the approximation. It is well-known that the inverted multi-index only assigns each item to a unique codeword in each subspace, making it possible to develop sublinear-time sampling methods from the softmax distribution. The following theorem lays the foundation.

**Theorem 4.1.** Assume \( z_u = z_u^1 \oplus z_u^2 \) is a vector of a user \( u \), \( q_i = c_1^k \oplus c_2^k + \hat{q}_i \) is a vector of an item \( i \), \( \Omega_{k_1, k_2} \) is the set of items which are assigned to \( c^1_{k_1} \) in the first subspace and \( c^2_{k_2} \) in the second subspace. The softmax probability \( Q(y_u|z_u) \) can be decomposed as follows:

\[
Q(y_u|z_u) = P^1_{\Omega_{k_1}}(k_1) \cdot P^2_{\Omega_{k_2}}(k_2) | P^1_{\Omega_{k_1}}(y_u|k_1, k_2), P^2_{\Omega_{k_2}}(y_u|k_1, k_2),
\]

\[
P^1_{\Omega_{k_1}}(k_1) = \frac{\psi_{k_1}}{\Sigma_{k_2=1}^K \psi_{k_2} \exp(z_{u_1}^T c^1_{k_2})},
\]

\[
P^2_{\Omega_{k_2}}(k_2) = \frac{\exp(z_{u_2}^T \hat{q}_i)}{\Sigma_{k_1=1}^K \exp(z_{u_1}^T c^1_{k_1}) \exp(z_{u_2}^T \hat{q}_i)}.
\]

The proof is attached in the Appendix. Theorem 4.1 can be straightforwardly extended to the case where \( m > 2 \). Surprisingly, this theorem provides a new perspective to exactly sample a candidate item from the softmax probability in Eq. (2), which is called MIDX sampler. First of all, we should construct three multinomial distributions in Eq. (3). Second, we sample an index \( k_1 \) from \( P^1_{\Omega_{k_1}}(\cdot) \), indicating to select the codeword from the first codebook.
5 APPROXIMATE SAMPLING WITH INVERTED MULTI-INDEX

The reason why MIDX spends much time on sampling part is that it involves computing inner-product logits over all items when preparing P1(·), P2(·|k1) and P3(·|k1, k2) in Eq. (3). To address this issue, we design three variants of MIDX by reducing the time for computing P1(·), P2(·|k1) and P3(·|k1, k2). Although these samplers only approximate the softmax distribution, we theoretically show that the divergence between them is small.

5.1 MIDX with Uniform

If replacing the multinomial distribution P3(·|k1, k2) with a non-personalized and static distribution, it will be efficient to prepare P1(·) and P2(·|k1), since they only involve computing the inner product between user vector and codewords instead of the whole item vectors. A straightforward choice is the uniform distribution. The resultant variant is called MIDX-Uni, whose distribution is derived based on the following theorem.

**Theorem 5.1.** Suppose P1(·) and P2(·|k1) remain the same as that in Theorem 4.1, P3(·|k1, k2) is replaced with a uniform distribution,
Table 1: Space and Time complexity of sampling T items from different proposals. Denote by M the number of items, D the representation dimension, and K the codebook size, B the sample size of DNS’s uniform sampling. (K, D ≪ M)

<table>
<thead>
<tr>
<th>Proposals Q</th>
<th>Space</th>
<th>Sample Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>1</td>
<td>T</td>
</tr>
<tr>
<td>Popularity</td>
<td>M</td>
<td>T</td>
</tr>
<tr>
<td>DNS [49]</td>
<td>MD</td>
<td>BDT</td>
</tr>
<tr>
<td>Kernel [6]</td>
<td>MD²</td>
<td>$D^2T\log M$</td>
</tr>
<tr>
<td>MIDX in Eq. (3)</td>
<td>MD</td>
<td>MD + T</td>
</tr>
<tr>
<td>MIDX_Uni in Eq. (4)</td>
<td>$KD + K^2 + M$</td>
<td>$KD + K^2 + T$</td>
</tr>
<tr>
<td>MIDX_Pop in Eq. (5)</td>
<td>$KD + K^2 + M$</td>
<td>$KD + K^2 + T$</td>
</tr>
</tbody>
</table>

The proof is attached in the Appendix. Generally, let $c_i$ be occurring frequency of item $i$, $\text{pop}(i)$ can be set to $c_i \log(1 + c_i)$ or $c_i^{3/4}$ [23]. We empirically find that $c_i$ achieves comparatively better performance. Theorem 5.2 shows that the sampling probability of an item is additionally affected by the popularity, such that the more popular items are more likely to be sampled. Regarding the time complexity, it takes $O(KD + K^2)$ time in the preprocessing part, which is the same as MIDX_Uni.

Table 1 summarizes the time and space complexity for item sampling from different proposals, which demonstrates the superiority of MIDX_Uni and MIDX_Pop in space and time cost. Thanks to the independence of the users, the MIDX_Uni and MIDX_Pop can be implemented on the GPUs, which accelerates the sampling procedure. Note that the initialization time refers to constructing indexes, such as alias tables, inverted multi-index or tree.

5.3 Theoretical Analysis

In this section, we further theoretically explain the bias of the proposed distribution from the softmax distribution.

Theorem 5.3. Assuming that the residual embedding $|| \tilde{q}_i || \leq C$, the Kullback–Leibler divergence from the softmax distribution $Q(y|z_u)$ to the proposed distribution $Q_{\text{uni}}(y|z_u)$ can be bounded from above:

$$0 < D_{KL}[Q_{\text{uni}}(y|z_u)||Q(y|z_u)] \leq 2C||z_u||$$

The proof is attached in the Appendix. The divergence of the proposal from Eq. (2) depends on $\exp(z_u^T \tilde{q}_i)$. Therefore, when $|| \tilde{q}_i ||, \forall i \leq M$ (i.e., distortion of product quantization) is small, the divergence between them is small. With the increasing granularity of space partition (the number of clusters in K-means), the residual vectors are of small magnitude such that the upper bound becomes smaller. This indicates that the approximate distribution is less deviated from the softmax distribution.

Theorem 5.4. Assuming that the residual embedding $|| \tilde{q}_i || \leq C$, the Kullback–Leibler divergence from the softmax distribution $Q(y|z_u)$ to the proposed distribution $Q_{\text{pop}}(y|z_u)$ can be bounded from above:

$$0 < D_{KL}[Q_{\text{pop}}(y|z_u)||Q(y|z_u)] \leq 2C||z_u|| + \log \frac{\max \text{pop}(\cdot)}{\min \text{pop}(\cdot)}$$

The proof is attached in the Appendix.

6 FASTVAE

We train Mult-VAE with sampled softmax, where we use the proposed proposals for item sampling (FastVAE for short). As shown in Section 3.1, the objective function in Eq. (1) consists of two terms. Regarding the first term, the expectation can be efficiently approximated by drawing a set of user vectors $\{z_u^{(1)}, z_u^{(2)}, \cdots, z_u^{(S)} \}$ from the variational posterior $q_\phi(y|z_u)$. By incorporating the sampled softmax, we draw a small set of candidate items $\Phi_u$ from one of our proposed samplers (i.e., MIDX_Uni and MIDX_Pop) and then the first term becomes:

$$\mathbb{E}_{z_u \sim q_\phi}(y|z_u)[\log p(y|z_u)]$$

$$= \frac{1}{S} \sum_{s=1}^{S} \sum_{i \in U} y_{ui} \log \frac{\exp(z_u^{(s)T} q_{i1} - \log Q(y|z_u^{(s)})]}{\sum_{j \in \Phi_u} \exp(z_u^{(s)T} q_{ij} - \log Q(y|z_u^{(s)})]}$$

For the second term, both the variational posterior $q_\phi(z_u|y_u)$ and the prior distribution $p(z_u) = \mathcal{N}(\mu, \sigma^2)$ and $p(z_u) = \mathcal{N}(0, 1)$, the KL divergence is computed as:

$$-KL(q_\phi(z_u|y_u)||p(z_u)) = \frac{1}{2} \left( \log \sigma^2 - \mu^2 - \sigma^2 + 1 \right)^T 1.$$

By maximizing the objective function, all parameters can be joint-optimized.

7 EXPERIMENTS

In the evaluation, the following three research questions are addressed. First, does FastVAE outperform the state-of-the-art baselines in terms of recommendation quality? Second, how accurately do the proposal distributions approximate the softmax distribution? Third, how efficiently are items sampled from the proposals? More details about the experimental settings are referred to in the Appendix.

7.1 Experimental Settings

7.1.1 Datasets. Experiments are conducted on the four public datasets for evaluation. The MovieLens10M (shorted as ML10M) dataset is a classic movie rating dataset, whose ratings range from 0 to 5. We convert them into 0/1 indicating whether the user has rated the movie. The Gowalla dataset includes users’ check-ins at locations in a location-based social network and is much sparser than the MovieLens dataset. The Netflix dataset is another famous movie rating dataset but with much more users. The Amazon dataset is a subset of customers’ ratings for Amazon books, where the rating scores are integers from 1 to 5, and books with scores higher than 4 are considered positive. For all the datasets, We filter out users and items with less than 10 interactions. The details are summarized in the Table 2.

For each user, we randomly sample 80% of interacted items to construct the history vector and fit the models to the training items. For evaluation, we take the user history to learn the necessary
7.1.2 Baselines. We compare our FastVAE with the following competing collaborative filtering models. The dimension of latent factors for users and items is set to 32 by default. Unless specified, we adopt the matrix factorization as the basic models.

- **WRMF** [13, 29], weighted regularized matrix factorization, is a famous collaborative filtering method for implicit feedback. It sets a prior on uninteracted items associated with the confidence level of being negative. It learns parameters by alternating least square method in the case of square loss. We tune the parameter of the regularizer of uninteracted items within [1,5,10,20,50,100,200,500]. The coefficient of L2 regularization is fixed to 0.01.

- **BPR** [33], Bayesian personalized ranking for implicit feedback, utilizes the pair-wise logit ranking loss between positive and negative samples. For each pair of interacted user and item, BPR randomly samples several uninteracted items of the user for training and applies stochastic gradient descent for optimization. We set the number of sampled negative items as 5 and tune the coefficient of regularization with [2,1,0.5].

- **WARP-MF** [45] uses the weighted approximate-rank pair-wise loss function for collaborative filtering. Given a positive item, it uniformly samples negative items until the rating of the sampled item is higher. The rank is estimated based on the sampling trials. We use the implementation in the lightFM\(^2\). The maximal number of trails is set to 50. The coefficient of the regularization is tuned within [0.05, 0.01, 0.005, 0.001] and the learning rate is tuned within \([10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}]\).

- **AOBPR** [32] improves the BPR with adaptive sampling method. We use the version implemented in LibRec\(^3\). The parameter for the geometric distribution is set to 500 and the learning rate is set to 0.05. We tune the coefficient of the regularization within [0.005, 0.01, 0.02].

- **DNS** [49] dynamically chooses items according to the predicted ranking list for the topk recommendation. Specifically, the dynamic sampler first draws samples uniformly from the item set and the item with the maximum rating is selected.

- **PRIS** [20] utilizes the importance sampling to the pairwise ranking loss for personalized ranking and assigns the sampling weight to the sampled items. We adopt the joint model implemented in the open resource code\(^4\). The number of clusters is set to 16.

- **Self-Adversarial (SA)** [39], a self-supervised method for negative sampling, is recently proposed for the recommendation. It utilizes uniform sampling and assigns the sampling weight for the negative item depending on the current model.

- **Multi-VAE** [21], variational autoencoders for collaborative filtering, is the work of learning user representations with variational autoencoders in recommendation systems. It learns the user representation by aiming at maximizing the likelihood of user click history. We mainly focus on the comparison with VAE and we will introduce the parameter setting in the following part.

In addition to these recommendation algorithms, we conduct experiments with the following samplers.

- **Uniform** sampler is a common sampling strategy that randomly draws negatives from the set of items for optimization, widely used for sampled softmax.

- **Population** sampler is correlated with the popularity of items, where the items with higher popularity have a greater probability of being sampled. The popularity is computed as \(\log(f_i + 1)\) where \(f_i\) is the occurring frequency of item \(i\). We normalize the popularity of all items for sampling.

- **Kernel** based sampler [6] is a recent method for adaptively sampled softmax, which lowers the bias by the non-negative quadratic kernel. Furthermore, the kernel-based sampler is implemented with divide and conquer depending on the tree structure.

7.1.3 Evaluation Metrics. Two standard metrics are utilized for evaluating the quality of recommendation, Normalized Discounted Cumulative Gain (NDCG) and Recall. A higher NDCG@k represents the positive items in the test data are ranked higher in the ranking list. Recall@k measures the fraction of the positive items in the test data. All algorithms are fine-tuned based on NDCG@50. After that, we run 5 times cross-validation.

7.1.4 Experiment Settings. We develop the proposed algorithms FastVAE with Pytorch in a Linux system (2.10 GHz Intel Xeon Gold 6230 CPUs and a Tesla V100 GPU). We utilize the Adam algorithm with a weight decay of 0.01 for optimization. We implement the variational autoencoder with one hidden layer and the generative module would be \([M \rightarrow 200 \rightarrow 32]\). The active function between layers is ReLu by default. The input of user history is dropout with a probability of 0.5 before the linear layers. The batch size is set to 256 for all the datasets. The learning rate is tuned over \([0.1,0.01,0.001,0.0001]\). We train the models within 200 epochs.

For the FastVAE, the number of samples is set to 200 for the MovieLens10M and Netflix dataset, 1000 for the Gowalla dataset, 2000 for the Amazon dataset. There are 16 codewords for each codebook. Regarding the popularity based strategy, we follow the same popularity function \(\log(f_i + 1)\).

7.2 Comparisons with Baselines

The comparisons of recommendation quality (i.e., Recall@50 and NDCG@50) with baselines is reported in Table 3, which are based on 5-time independent trials. We report the results of FastVAE with MIDX_Pop here. We have the following findings.

Finding 1: By using the MIDX-like proposals, FastVAE with sampled softmax could behave almost as well as Multi-VAE with full softmax and even perform slightly better. Surprisingly, the averaged relative improvements are even up to 0.64% and 0.25% on four datasets in terms of NDCG@50 and Recall@50, respectively. This implies the

---

**Table 2: Dataset Statistics**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#User</th>
<th>#Item</th>
<th>#Interactions</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML10M</td>
<td>47,292</td>
<td>5,942</td>
<td>2,001,164</td>
<td>99.2879%</td>
</tr>
<tr>
<td>Gowalla</td>
<td>29,858</td>
<td>40,988</td>
<td>1,027,464</td>
<td>99.9160%</td>
</tr>
<tr>
<td>Amazon</td>
<td>56,257</td>
<td>50,154</td>
<td>1,418,076</td>
<td>99.9497%</td>
</tr>
<tr>
<td>Netflix</td>
<td>422,624</td>
<td>17,618</td>
<td>53,417,358</td>
<td>99.2826%</td>
</tr>
</tbody>
</table>

---

\(^2\)https://github.com/lyst/lightfm
\(^3\)https://github.com/guoguangming/librec
\(^4\)https://github.com/DefuLian/PRIS
### 7.3 Comparisons with Different Samplers

#### 7.3.1 Divergence between Proposals and the Softmax Distribution

In order to understand how accurately the proposal distributions approximate the softmax distribution, we investigate the divergence between the proposals and the softmax distribution on the MovieLens10M dataset. In particular, we randomly select a user, and compute her/his softmax distribution with a randomly-initialized model and well-trained model, respectively. Regarding the proposals, we sample 100,000 items from each of them and then plot the cumulative probability distribution. Regarding MIDX-like proposals in both cases, 64 clusters in each quantization will be used. The results of these two cases are reported in Figure 2, where items are sorted by popularity for better comparison. Note that since we have observed similar results from multiple users, only one user’s result is reported for illustration. We have the following findings.

**Finding 1:** The MIDX sampler is as accurate as softmax, based on full coincidence between softmax and MIDX in both cases. This is because the decomposition of the softmax distribution is fully exact, as shown in Section 4. However, since the item should be sampled from the residual softmax distribution, the time cost is so high that the MIDX sampler is not directly used in practice.

**Finding 2:** When the model is well-trained, the MIDX-variant samplers are much closer to the softmax distribution than Kernel and DNS. This implies that MIDX-variant samplers reduce the bias of sampled softmax. Though Kernel-based sampling also directly approximates the softmax distribution, it is almost as close as DNS to the softmax. Moreover, the Kernel-based sampling approximates the probability of long-tailed items less accurately. This is consistent with the fact that the Kernel-based sampler oversamples items with negative logits [6] since long-tailed items are more likely to yield negative logits.

**Finding 3:** The MIDX-Uni sampler approximates the softmax distribution a little less accurately than MIDX-Pop. These samplers mainly vary in item sampling in the last stage, but all depend on item vector quantization. This implies the effect of inverted multi-index at approximate sampling. Moreover, compared to the softmax distribution, these samplers are more likely to sample less popular items, evidenced by that their curves are slightly above the softmax.

**Finding 4:** The MIDX-variant samplers can capture the dynamic update of the model. In particular, when the model is well-trained, the MIDX-variant samplers well approximate the softmax; when the model is only randomly initialized, most dynamic samplers are similar to the static samplers. The latter observation is reasonable since randomized representations do not have cluster structures. This indicates that along with the training course of the model, the MIDX-variant samplers can be more and more informative.

#### 7.3.2 Effectiveness Study of Samplers

To validate the effectiveness of the proposed MIDX-based samplers, we investigate the recommendation performance during training epochs with different samplers aforementioned in the section 7.1.2. We report the changing curve of NDCG@50 and Recall@50 on the Gowalla and Netflix datasets in Figure 3. We sample 1,000 items for the Gowalla dataset and 200 items for the Netflix dataset with all the tested samplers. The number of the sampled items are greatly smaller than the number of total items.

![Figure 2: Cumulative probability distribution of different samplers. Items are sorted by popularity.](image-url)
our proposed MIDX-based samplers show superior efficiency compared to the baseline samplers. Compared with the static samplers, i.e., Uniform and Popularity, the MIDX-based samplers tend to sample more informative items so that they defeat the samplers during the whole training process. Although the Kernel sampler has a better estimation of the softmax distribution and can capture the dynamics of the softmax distribution, the sampling probability is not well attached for calculating the sampled softmax loss, so that it performs bad in terms of the recommendation quality. On the more sparse dataset, Gowalla, the MIDX_Uni and MIDX_Pop perform as well as the Multi-VAE and even has slightly better performance. This may implies the oversampling of the full softmax.

### 7.3.3 Efficiency Study of Samplers

From the figure, we have the following finding: The MIDX-based samplers contribute to faster convergence compared to the baseline samplers. Compared with the static samplers, i.e., Uniform and Popularity, the MIDX-based samplers tend to sample more informative items so that they defeat the samplers during the whole training process. Although the Kernel sampler has a better estimation of the softmax distribution and can capture the dynamics of the softmax distribution, the sampling probability is not well attached for calculating the sampled softmax loss, so that it performs bad in terms of the recommendation quality. On the more sparse dataset, Gowalla, the MIDX_Uni and MIDX_Pop perform as well as the Multi-VAE and even has slightly better performance. This may implies the oversampling of the full softmax.

### 7.4 Sensitivity Analysis

#### 7.4.1 Number of negative samples

We conduct experiments on the Gowalla datasets with the Uniform and MIDX_Uni sampler, as shown in Figure 5(a). The numbers of negative items are varied in [200, 1000, 5000]. Our proposed MIDX-based samplers show superior performance even if the number of sampled items is modest. With the increasing of the sample numbers, the two samplers perform better in terms of NDCG@50. When the number of negative items is greatly small, the MIDX_Uni also improves dramatically, indicating the good estimation of the softmax.

#### 7.4.2 Number of clusters

The number of clusters can greatly influence the performance of the approximation, as analysed in the Theorem 5.3. We further validate the influence of the cluster numbers in terms of the recommendation quality. The numbers of clusters are varied in {8, 16, 32, 64, 128}. We report the running curve in Figure 5(b). With the increasing of the cluster number, the MIDX_Uni sampler performs better in the initial training epochs, indicating the better estimation with more clusters. Meanwhile, the MIDX_Uni also behave well with less clusters, demonstrating the robustness of the MIDX_Uni sampler with respect to the cluster number.

### 8 CONCLUSION

In this paper, we discover the high-quality approximation of the softmax distribution by decomposing the softmax probability with the inverted multi-index, and design efficient sampling procedures, from which items can be independently sampled in sublinear or even constant time. These approximate samplers are exploited for fast training the variational autoencoder for collaborative filtering. The experiments on the three public real-world datasets demonstrate that the FastVAE outperforms the state-of-the-art baselines in terms of sampling quality and efficiency.

### ACKNOWLEDGMENTS

The work is supported by the National Natural Science Foundation of China (No. 62022077, 61972069, 61836007 and 61832017), and Shenzhen Municipal Science and Technology R&D Funding Basic Research Program (JCYJ20210324133607021).
For better illustration, we review some important notations here. In the following, denote by \( I \) the set of items, \( z_u \) a vector of the user \( u \), \( q_i \) a vector of the item \( i \). The softmax probability with the inner-product logits can be computed by:

\[
Q(y_i | z_u) = \frac{\exp(z_u^T q_i)}{\sum_{j \in I} \exp(z_u^T q_j)}
\]

Particularly, \( q_i \) can be decomposed based on the codebooks. It is formulated as \( q_i = e_{k_1}^i \odot e_{k_2}^i + \bar{q}_i \) where \( e_{k_1}^i \) is the \( k_1 \)-th codeword and \( \bar{q}_i \) is the residual vector.

**Theorem A.1 (Theorem 4.1).** Assume \( z_u = z_u^1 \odot z_u^2 \) is a vector of a user \( u \), \( q_i = e_{k_1}^i \odot e_{k_2}^i + \bar{q}_i \) is a vector of an item \( i \), \( \Omega_{k_1,k_2} \) is the set of items which are assigned to \( e_{k_1}^i \) in the first subspace and \( e_{k_2}^i \) in the second subspace. The softmax probability \( Q(y_i | z_u) \) can be decomposed as follows:

\[
\begin{align*}
Q(y_i | z_u) &= P_u^1(k_1) \cdot P_u^2(k_2 | k_1) \cdot P_y^0(y_i | k_1, k_2), \\
P_u^1(k_1) &= \sum_{k=1}^{K} \psi_k \exp(z_u^T e_{k_1}^i) \\
P_u^2(k_2 | k_1) &= \omega_{k_1,k_2} \exp(z_u^T e_{k_2}^i) \\
P_y^0(y_i | k_1, k_2) &= \sum_{j \in \Omega_{k_1,k_2}} \sum_{j \in I} \exp(z_u^T \bar{q}_j), \\
\end{align*}
\]

**Proof.**

\[
\begin{align*}
Q(y_i | z_u) &= \frac{\exp(z_u^T q_i)}{\sum_{j \in I} \exp(z_u^T q_j)} \\
&= \frac{\exp(z_u^T e_{k_1}^i \odot e_{k_2}^i + \bar{q}_i)}{\sum_{j \in I} \exp(z_u^T e_{k_2}^i \odot e_{k_2}^i + \bar{q}_j)} \\
&= \sum_{k=1}^{K} \frac{\exp(z_u^T e_{k_1}^i) \exp(z_u^T e_{k_2}^i) \exp(z_u^T \bar{q}_i)}{\sum_{j \in \Omega_{k_1,k_2}} \exp(z_u^T \bar{q}_j)} \\
&= P_u^1(k_1) \cdot \sum_{k=1}^{K} \frac{\exp(z_u^T e_{k_1}^i) \exp(z_u^T e_{k_2}^i) \exp(z_u^T \bar{q}_i)}{\sum_{j \in \Omega_{k_1,k_2}} \exp(z_u^T \bar{q}_j)} \\
&= P_u^1(k_1) \cdot P_u^2(k_2 | k_1) \cdot P_y^0(y_i | k_1, k_2).
\end{align*}
\]

**Theorem A.2 (Theorem 5.1).** Suppose \( P_y(.) \) and \( P_z(.|k_2) \) remain the same as that in Theorem 4.1, \( P_z(.|k_1, k_2) \) is replaced with a uniform distribution, i.e. \( P_y(y_i | k_1, k_2) = \frac{1}{|\Omega_{k_1,k_2}|} \) where \( |\Omega_{k_1,k_2}| \) denotes the number of items in the set. Then, the proposal distribution is equivalent to:

\[
Q_{un}(y_i | z_u) = \frac{\exp(z_u^T e_{k_1}^i) \exp(z_u^T e_{k_2}^i) \exp(z_u^T \bar{q}_i)}{\sum_{k,k'} |\Omega_{k,k'}| \exp(z_u^T e_{k_1}^i) \exp(z_u^T e_{k_2}^i) \exp(z_u^T \bar{q}_i)}
\]

**Proof.**

\[
Q_{un}(y_i | z_u) = P_u^1(k_1) \cdot P_u^2(k_2 | k_1) \cdot P_y^0(y_i | k_1, k_2)
\]

**Theorem A.3 (Theorem 5.2).** Suppose \( P_y(.) \) and \( P_z(.|k_2) \) remain the same as that in Theorem 4.1, \( P_z(.|k_1, k_2) \) is replaced with a distribution derived from the popularity, i.e. \( P_y(y_i | k_1, k_2) = \frac{\text{pop}(i)}{\sum_{j \in \Omega_{k_1,k_2}} \text{pop}(j)} \) where \( \text{pop}(i) \) can be any metric of the popularity. Then, the proposal distribution is equivalent to:

\[
Q_{pop}(y_i | z_u) = \frac{\exp(z_u^T \bar{q}_i) + \log \text{pop}(i)}{\sum_{j \in I} \exp(z_u^T \bar{q}_j) + \log \text{pop}(j)}
\]

**Proof.**

\[
Q_{pop}(y_i | z_u) = P_u^1(k_1) \cdot P_u^2(k_2 | k_1) \cdot P_y^0(y_i | k_1, k_2)
\]

**Theorem A.4 (Theorem 5.3).** Assuming that the residual embedding \( ||\bar{q}_i|| \leq C \), the Kullback–Leibler divergence from the softmax distribution \( Q(y_i | z_u) \) to the proposed distribution \( Q_{un}(y_i | z_u) \) can be bounded from above:

\[
0 < D_{KL}[Q_{un}(y_i | z_u) || Q(y_i | z_u)] \leq 2C || z_u ||
\]

**Proof.**

\[
Q(y_i | z_u) = \frac{\exp(z_u^T q_i)}{\sum_{j \in I} \exp(z_u^T q_j)}
\]

\[
Q_{un}(y_i | z_u) = \frac{\exp(z_u^T (q_i - \bar{q}))}{\sum_{j \in I} \exp(z_u^T (q_j - \bar{q}))}
\]
Theorem A.5 (Theorem 5.4). Assuming that the residual embedding $||\hat{q}_i|| \leq C$, the Kullback–Leibler divergence from the softmax distribution $Q(y_i|z_u)$ to the proposed distribution $Q_{\text{pop}}(y_i|z_u)$ can be bounded from above:

$$0 < D_KL[Q_{\text{muw}}(y_i|z_u)||Q(y_i|z_u)] \leq 2C||z_u|| + \log \frac{\max \text{pop}(-)}{\min \text{pop}(-)}.$$ 

Proof.

Denote $E(i) = \exp(\hat{q}_i) + \log \text{pop}(i)$,

$$Q_{\text{pop}}(y_i|z_u) = \frac{\sum_{j \in I} \exp(\hat{q}_j) \cdot \exp(-\hat{q}_i)}{\sum_{k \in I} \exp(\hat{q}_k - \hat{q}_i)}$$

$$= \sum_{j \in I} \exp(\hat{q}_j) \cdot \frac{\exp(\hat{q}_j)}{\sum_{k \in I} \exp(\hat{q}_k - \hat{q}_i)}$$

$$\leq \sum_{j \in I} \exp(\hat{q}_j) \cdot \frac{\exp(\hat{q}_j + ||\hat{q}_j||)}{\sum_{k \in I} \exp(\hat{q}_k - \hat{q}_i)}$$

$$= \sum_{j \in I} \exp(\hat{q}_j) \cdot \frac{\exp(2C||z_u||)}{\sum_{k \in I} \exp(\hat{q}_k - \hat{q}_i)}$$

$$= \exp(2C||z_u||).$$

$$D_KL[Q_{\text{muw}}(y_i|z_u)||Q(y_i|z_u)] = \sum_{i \in I} Q_{\text{muw}}(y_i|z_u) \log \frac{Q_{\text{muw}}(y_i|z_u)}{Q(y_i|z_u)}$$

$$\leq \sum_{i \in I} Q_{\text{muw}}(y_i|z_u) \log \exp(2C||z_u||)$$

$$= \sum_{i \in I} Q_{\text{muw}}(y_i|z_u) \cdot 2C ||z_u||$$

$$= 2C ||z_u|| \sum_{i \in I} Q_{\text{muw}}(y_i|z_u)$$

$$= 2C ||z_u|| \log \frac{\max \text{pop}(-)}{\min \text{pop}(-)}.$$