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**.

ACM Reference Format:
Jin Chen, Defu Lian, Binbin Jin, Xu Huang, Kai Zheng, and Enhong Chen. 2022. Fast Variational AutoEncoder with Inverted Multi-Index for Collaborative Filtering. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3485447.3512068

*This work was done when the author Jin Chen was at University of Science and Technology of China for intern.
†Corresponding authors

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France.

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512068

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 representative 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 can not 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[1] 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 Mult-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 $\boldsymbol{y}_u$ of dimension M. Each entry y_{ui} indicates an interaction record to the item i, where $y_{ui} = 0$ indicates no interaction. Mult-VAE [21] is a representative nonlinear recommender method for modeling such implicit data. It particularly encodes $\boldsymbol{y}_u$ with a Gaussian-distributed latent factor $\boldsymbol{z}_u$ and then decodes it to $\hat{\boldsymbol{y}}_u$, a probability distribution over all items. The objective is to

[1] https://github.com/HERECJ/FastVae_Gpu

maximize the evidence lower bound (ELBO):

$$\mathcal{L}(u) = \mathbb{E}_{z_u \sim q_\phi(\cdot|\boldsymbol{y}_u)}[\log p_\theta(\boldsymbol{y}_u|z_u)] - KL\big(q_\phi(z_u|\boldsymbol{y}_u)||p(z_u)\big), \quad (1)$$

where $q_\phi(z_u|\boldsymbol{y}_u)$ is the variational posterior with parameters ϕ to approximate the true posterior $p(z_u|\boldsymbol{y}_u)$. $q_\phi(z_u|\boldsymbol{y}_u)$ is generally assumed to follow the Gaussian-distribution whose mean and variance are estimated by the encoder of Mult-VAE. That is, $z_u \sim \mathcal{N}\big(\mathrm{MLP}_\mu(\boldsymbol{y}_u;\phi), \mathrm{diag}(\mathrm{MLP}_{\sigma^2}(\boldsymbol{y}_u;\phi))\big)$, where MLP_μ and MLP_{σ^2} denote multilayer perceptrons (MLPs). $p(z_u)$ is the prior Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. $p_\theta(\boldsymbol{y}_u|z_u)$ is the generative distribution conditioned on z_u. The observed data $\boldsymbol{y}_u$ is assumed to be drawn from the multinomial distribution, which motivates the widely-used multinomial log-likelihood in Eq. (1):

$$\log p_\theta(\boldsymbol{y}_u|z_u) = \sum_{i\in\mathcal{I}} \log p_\theta(\hat{y}_{ui}|z_u) = \sum_{i\in\mathcal{I}} \log \frac{\exp(z_u^\top \boldsymbol{q}_i)}{\sum_{j\in\mathcal{I}} \exp(z_u^\top \boldsymbol{q}_j)},$$

where $\mathcal{I}$ is the set of all items, z_u and $\boldsymbol{q}_i$ 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 $P(i) = \frac{\exp(o_i)}{\sum_{j=1}^{|\mathcal{I}|}\exp(o_j)}$. Denoting model parameters by θ, the gradient of the log-likelihood loss is computed as $\nabla_\theta \log P(i) = \nabla_\theta o_i - \mathbb{E}_{j\sim P}\nabla_\theta o_j$. 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 sampling a small set of candidate samples Φ 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:

$$Q(y_i|z_u) = \frac{\exp(z_u^\top \boldsymbol{q}_i)}{\sum_{j\in\mathcal{I}} \exp(z_u^\top \boldsymbol{q}_j)}. \quad (2)$$

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 $\boldsymbol{q} \in \mathbb{R}^D$ is an item vector, we first evenly split it into m distinct subvectors (i.e., $\boldsymbol{q} = \boldsymbol{q}^1 \oplus \boldsymbol{q}^2 \oplus \cdots \oplus \boldsymbol{q}^m$ where $\oplus$ is the concatenation). Then, each subvector $\boldsymbol{q}^l$ is mapped to an element of a fixed-size vector set by a quantizer $f_l : f_l(\boldsymbol{q}^l) \in C^l = \{\boldsymbol{c}_k^l | k \in \{1, ..., K\}\}$, where C^l is the vector set (i.e. codebook) of size K in the l-th subspace and the element $\boldsymbol{c}_k^l$ is called a codeword. Therefore, $\boldsymbol{q}$ is mapped as follows:

$$\boldsymbol{q} \to f_1(\boldsymbol{q}^1) \oplus f_2(\boldsymbol{q}^2) \oplus \cdots \oplus f_m(\boldsymbol{q}^m) = \boldsymbol{c}_{k_1}^1 \oplus \boldsymbol{c}_{k_2}^2 \oplus \cdots \oplus \boldsymbol{c}_{k_m}^m.$$

where $k_l (1 \le l \le m)$ is the index of the mapped codeword from $\boldsymbol{q}^l$. 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 $\tilde{\boldsymbol{q}} = \boldsymbol{q} - \boldsymbol{c}_{k_1}^1 \oplus \boldsymbol{c}_{k_2}^2$ 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, $\boldsymbol{q}_i = \boldsymbol{c}_{k_1}^1 \oplus \boldsymbol{c}_{k_2}^2 + \tilde{\boldsymbol{q}}_i$ is a vector of an item i, Ω_{k_1,k_2} is the set of items which are assigned to $\boldsymbol{c}_{k_1}^1$ in the first subspace and $\boldsymbol{c}_{k_2}^2$ in the second subspace. The softmax probability $Q(y_i|z_u)$ can be decomposed as follows:*

$$\begin{aligned}
Q(y_i|z_u) &= P_u^1(k_1)\cdot P_u^2(k_2|k_1)\cdot P_u^3(y_i|k_1,k_2),\\
P_u^1(k_1) &= \frac{\psi_{k_1}\exp({z_u^1}^\top \boldsymbol{c}_{k_1}^1)}{\sum_{k=1}^K \psi_k \exp({z_u^1}^\top \boldsymbol{c}_k^1)},\\
P_u^2(k_2|k_1) &= \frac{\omega_{k_1,k_2}\exp({z_u^2}^\top \boldsymbol{c}_{k_2}^2)}{\underbrace{\sum_{k=1}^K \omega_{k_1,k}\exp({z_u^2}^\top \boldsymbol{c}_k^2)}_{\psi_{k_1}}},\\
P_u^3(y_i|k_1,k_2) &= \frac{\exp(z_u^\top \tilde{\boldsymbol{q}}_i)}{\underbrace{\sum_{j\in\Omega_{k_1,k_2}} \exp(z_u^\top \tilde{\boldsymbol{q}}_j)}_{\omega_{k_1,k_2}}}.
\end{aligned} \quad (3)$$

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 $P_u^1(\cdot)$, indicating to select the codeword from the first codebook

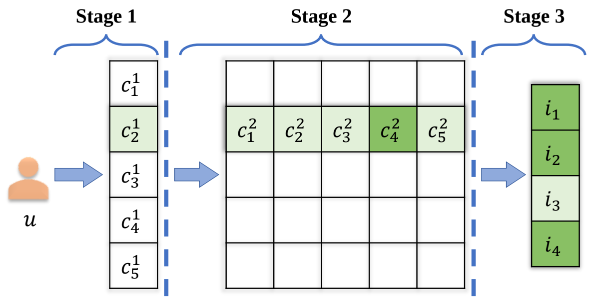


Figure 1: An illustration of the sampling. Firstly, draw a codeword index ($k_1 = 2$) in the first codebook, and then draw another codword index ($k_2 = 4$). Finally, item i_3 is sampled from $\Omega_{2,4}$, which is the set of items assigned to k_1 and k_2.

C^1. Third, we sample another index k_2 from the conditional probability $P_u^2(\cdot|k_1)$, indicating to select the codeword from the second codebook C^2 given the first index. Finally, a candidate item can be sampled from the residual softmax in $P_u^3(\cdot|k_1, k_2)$. An important observation is that ω_{k_1,k_2} is absolutely not empty, such that each time an item can be sampled out in the last step. Figure 1 illustrates the procedure and Algorithm 1 details the workflow.

Time complexity analysis. From Algorithm 1, we see that the overall procedure can be split into two parts. Lines 1-3 describe the initialization part to obtain codebooks and lines 4-13 describe the sampling part with the computation of the probability. Being independent to users, the initialization part is only executed once in $O(KMDt)$, where M is the number of items and t is the number of iterations in K-means. Thanks to the Vose-Alias method sampling techniques [42], the sampling part only takes $O(1)$ time to sample an item. Unfortunately, it is necessary to compute the inner-product logits over all items, which takes $O(MD)$ time. This indicates that it is no more efficient than sampling an item from the softmax distribution directly.

Algorithm 1: MIDX Sampling

Input: Items' vectors $\{\boldsymbol{q}_i | i \in \mathcal{I}\}$, user vector $\boldsymbol{z}_u$, sampling size T, codebook size K

Output: Candidate samples with sampling probability (Φ)

1 $C^1, C^2 \leftarrow$ ProductQuantization($\{\boldsymbol{q}_i | i \in \mathcal{I}\}, K$) ;
2 Compute residual vectors for all items $\{\tilde{\boldsymbol{q}}_i | i \in \mathcal{I}\}$;
3 Compute $\Omega_{k_1,k_2}, \forall 1 \le k_1, k_2 \le K$;
`// Sampling part in` $O(MD + T)$
4 **for** $k_1 = 1$ **to** K **do**
5   **for** $k_2 = 1$ **to** K **do**
6   Compute ω_{k_1,k_2} and construct $P_u^3(\cdot|k_1, k_2)$ in Eq. (3);
7   Compute ψ_{k_1} and construct $P_u^2(\cdot|k_1)$ in Eq. (3);
8 Construct $P_u^1(\cdot)$ in Eq. (3);
9 Initialize $\Phi = \emptyset$;
10 **for** $i = 1$ **to** T **do**
11   Respectively sample k_1, k_2, i from $P_u^1(\cdot), P_u^2(\cdot|k_1)$ and $P_u^3(\cdot|k_1, k_2)$ in order;
12     $Q(y_i|z_u) \leftarrow P_u^1(k_1) P_u^2(k_2|k_1) P_u^3(y_i|k_1, k_2)$;
13     $\Phi \leftarrow \Phi \cup (i, Q(y_i|z_u))$;
14 Return Φ;

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 $P_1(\cdot)$, $P_2(\cdot|k_1)$ and $P_3(\cdot|k_1, k_2)$ in Eq. (3). To address this issue, we design three variants of MIDX sampling by reducing the time for computing $P_1(\cdot)$, $P_2(\cdot|k_1)$ and $P_3(\cdot|k_1, k_2)$. 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 $P_3(\cdot|k_1, k_2)$ with a non-personalized and static distribution, it will be efficient to prepare $P_1(\cdot)$ and $P_2(\cdot|k_1)$, 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 $P_1(\cdot)$ and $P_2(\cdot|k_1)$ remain the same as that in Theorem 4.1, $P_3(\cdot|k_1, k_2)$ is replaced with a uniform distribution, i.e. $P_3(y_i|k_1, k_2) = \frac{1}{|\Omega_{k_1,k_2}|}$, where $|\Omega_{k_1,k_2}|$ denotes the number of items in the set. Then, the proposal distribution is equivalent to:*

$$
\begin{aligned}
Q_{uni}(y_i|z_u) &= \frac{\exp({z_u^1}^\top \boldsymbol{c}_{k_1}^1)\exp({z_u^2}^\top \boldsymbol{c}_{k_2}^2)}{\sum_{k,k'} |\Omega_{k,k'}| \exp({z_u^1}^\top \boldsymbol{c}_k^1)\exp({z_u^2}^\top \boldsymbol{c}_{k'}^2)} \\
&= \frac{\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_i - \tilde{\boldsymbol{q}}_i))}{\sum_{j\in\mathcal{I}} \exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_j - \tilde{\boldsymbol{q}}_j))}.
\end{aligned}
\tag{4}
$$

The proof is attached in the Appendix. Theorem 5.1 shows that each time the codeword with the large inner product and with more items are more likely to be sampled.

Time complexity analysis. When computing the sampling probability, the computation only involves the inner product between the user vector and all codewords, which takes $O(KD)$ to compute. In addition, it takes $O(K^2)$ since it should calculate the normalization constant in $P_2(\cdot|k_1)$ for each k_1. Overall, the time complexity of the preprocessing part is $O(KD + K^2)$. Since the codebook size K is much smaller than the number of items M, MIDX_Uni sampling is much more efficient than the MIDX sampling.

5.2 MIDX with Popularity

Besides the uniform distribution, another widely-used static distribution is derived from popularity. If a user does not interact with a popular item, she may be truly uninterested in it since the item is highly likely to be exposed to the user. Therefore, by introducing the popularity, we design the second variant, **MIDX_Pop**, whose distribution is derived by the following theorem.

Theorem 5.2. *Suppose $P_1(\cdot)$ and $P_2(\cdot|k_1)$ remain the same as that in Theorem 4.1, $P_3(\cdot|k_1, k_2)$ is replaced with a distribution derived from the popularity, i.e. $P_3(y_i|k_1, k_2) = \frac{pop(i)}{\sum_{j\in\Omega_{k_1,k_2}} pop(j)}$, where $pop(i)$ can be any metric of the popularity. Then, the proposal distribution is*

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. ($K, D \ll M$)

Proposals Q	Space	Sample Time
Uniform	1	T
Popularity	M	T
DNS [49]	MD	BDT
Kernel [6]	MD^2	$D^2T\log M$
MIDX in Eq. (3)	MD	$MD+T$
MIDX_Uni in Eq. (4)	$KD+K^2+M$	$KD+K^2+T$
MIDX_Pop in Eq. (5)	$KD+K^2+M$	$KD+K^2+T$

equivalent to:

$$Q_{pop}(y_i|z_u) = \frac{\exp(z_u^\top(\boldsymbol{q}_i - \tilde{\boldsymbol{q}}_i) + \log pop(i))}{\sum_{j\in\mathcal{I}}\exp(z_u^\top(\boldsymbol{q}_j - \tilde{\boldsymbol{q}}_j) + \log pop(j))}. \tag{5}$$

The proof is attached in the Appendix. Generally, let c_i be occurring frequency of item i, $pop(i)$ 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 $\|\tilde{\boldsymbol{q}}_i\| \le C$, the Kullback–Leibler divergence from the softmax distribution $Q(\boldsymbol{y}_\cdot|z_u)$ to the proposed distribution $Q_{uni}(\boldsymbol{y}_\cdot|z_u)$ can be bounded from above:*

$$0 < \mathcal{D}_{KL}\left[Q_{uni}(\boldsymbol{y}_\cdot|z_u)||Q(\boldsymbol{y}_\cdot|z_u)\right] \le 2C\|z_u\|.$$

The proof is attached in the Appendix. The divergence of the proposal from Eq. (2) depends on $\exp(z_u^\top\tilde{\boldsymbol{q}}_i)$. Therefore, when $\|\tilde{\boldsymbol{q}}_i\|, \forall 1 \le i \le M$ (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 $\|\tilde{\boldsymbol{q}}_i\| \le C$, the Kullback–Leibler divergence from the softmax distribution $Q(\boldsymbol{y}_\cdot|z_u)$ to the proposed distribution $Q_{pop}(\boldsymbol{y}_\cdot|z_u)$ can be bounded from above:*

$$0 < \mathcal{D}_{KL}\left[Q_{pop}(\boldsymbol{y}_\cdot|z_u)||Q(\boldsymbol{y}_\cdot|z_u)\right] \le 2C||z_u|| + \log\frac{\max pop(\cdot)}{\min pop(\cdot)}.$$

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 $q_\phi(\cdot|\boldsymbol{y}_u)$. By incorporating the sampled softmax, we draw a small set of candidate items Φ_u from one of our proposed samplers (i.e., MIDX_Uni and MIDX_Pop) and then the first term becomes:

$$\mathbb{E}_{z_u\sim q_\phi(\cdot|\boldsymbol{y}_u)}[\log p_\theta(\boldsymbol{y}_u|z_u)]$$
$$= \frac{1}{S}\sum_{s=1}^{S}\sum_{i\in\mathcal{I}} y_{ui}\log\frac{\exp\left({z_u^{(s)}}^\top\boldsymbol{q}_i - \log Q(y_i|z_u^{(s)})\right)}{\sum_{j\in\{i\}\cup\Phi_u}\exp\left({z_u^{(s)}}^\top\boldsymbol{q}_j - \log Q(y_j|z_u^{(s)})\right)}.$$

For the second term, both the variational posterior $q_\phi(z_u|\boldsymbol{y}_u)$ and the prior distribution $p(z_u)$ follow Gaussian distributions, so that the KL divergence has a closed-form solution. Suppose $q_\phi(z_u|\boldsymbol{y}_u) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ and $p(z_u) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$, the KL divergence is computed as:

$$-KL\big(q_\phi(z_u|\boldsymbol{y}_u)||p(z_u)\big) = \frac{1}{2}(\log\boldsymbol{\sigma}^2 - \boldsymbol{\mu}^2 - \boldsymbol{\sigma}^2 + \boldsymbol{1})^\top\boldsymbol{1}.$$

By maximizing the objective function, all parameters can be jointly 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 0.5 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 summerized 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

Table 2: Dataset Statistics

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

representations from the well-trained model and then compute metrics by looking at how well the model ranks the unseen history.

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.
- **Mult-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.
- **Popularity** 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 forall 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 codewodes for each code book. 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 behaves 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

[2]https://github.com/lyst/lightfm
[3]https://github.com/guoguibing/librec
[4]https://github.com/DefuLian/PRIS

Table 3: Comparisons with baselines w.r.t NDCG@50 and Recall@50 ($\Delta = 10^{-4}$).

	MovieLens-10M		Gowalla		Netflix		Amazon	
	NDCG@50	Recall@50	NDCG@50	Recall@50	NDCG@50	Recall@50	NDCG@50	Recall@50
WRMF	0.3194±0.3Δ	0.4967±0.6Δ	0.1316±0.1Δ	0.2223±0.1Δ	0.3020±0.1Δ	0.3653±0.6Δ	0.0919±1.4Δ	0.1802±3.0Δ
BPR	0.2915±2.4Δ	0.4642±3.2Δ	0.1216±1.1Δ	0.1978±1.9Δ	0.2742±1.7Δ	0.3283±1.2Δ	0.0740±2.2Δ	0.1441±4.2Δ
WARP-MF	0.2968±2.3Δ	0.4785±3.3Δ	0.1273±0.7Δ	0.2073±1.7Δ	0.2953±1.2Δ	0.3539±1.2Δ	0.0798±1.4Δ	0.1615±3.7Δ
AOBPR	0.2934±0.5Δ	0.4753±0.3Δ	0.1385±0.4Δ	0.2369±0.8Δ	0.2952±0.4Δ	0.3560±0.8Δ	0.0906±1.7Δ	0.1763±2.5Δ
DNS	0.3153±2.7Δ	0.4988±3.4Δ	0.1622±1.5Δ	0.2761±2.9Δ	0.2974±2.4Δ	0.3594±2.5Δ	0.1119±1.7Δ	0.2186±3.4Δ
PRIS	0.3162±1.6Δ	0.4937±3.3Δ	0.1657±2.7Δ	0.2736±3.1Δ	0.2975±2.5Δ	0.3608±2.9Δ	0.1189±2.9Δ	0.2244±4.1Δ
SA	0.3237±1.4Δ	0.5066±2.0Δ	0.1704±1.2Δ	0.2866±1.8Δ	0.3177±3.5Δ	0.3784±2.8Δ	0.1378±1.8Δ	0.2401±2.9Δ
Mult-VAE	0.3206±2.5Δ	0.5037±2.7Δ	0.1751±3.2Δ	0.2911±5.7Δ	0.3227±2.8Δ	0.3841±3.1Δ	0.1441±0.9Δ	0.2483±2.9Δ
FastVAE	**0.3275**±2.5Δ	**0.5078**±2.4Δ	**0.1797**±2.0Δ	**0.2971**±2.1Δ	**0.3238**±3.0Δ	**0.3845**±2.7Δ	**0.1404**±2.1Δ	**0.2434**±3.5Δ

MIDX-like proposals could accurately approximate the softmax distribution and sample informative items. The improvements may lie in the oversampling of less popular items.

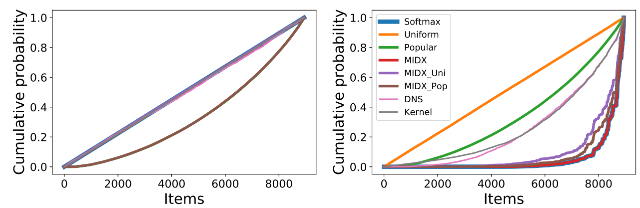
Finding 2: FastVAE outperforms all state-of-the-art baselines on two datasets. The averaged relative improvements over the best baseline are up to 2.61% and 1.72% in terms of NDCG@50 and Recall@50, respectively. This indirectly implies the effectiveness of the proposed samplers at sampling high-informative items. Note that WRMF usually works better than static-sampling-based baselines, as WRMF treats all unobserved data as negative. However, the lack of differentiation among them leads to sub-optimal solutions compared to Mult-VAE, whose objective function (i.e. full softmax) also takes all items into account.

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 the Kernel-based sampler oversamples items with negative logits [6] since long-tailed items are more likely to yield negative logits.



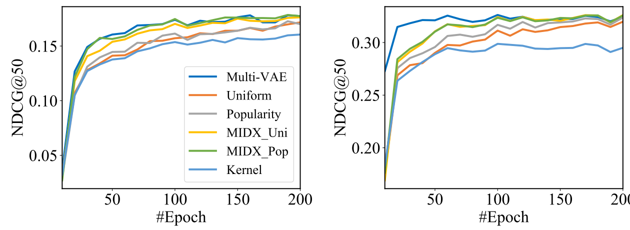
(a) Randomly-initialized model (b) Well-trained model

Figure 2: Cumulative probability distribution of different samplers. Items are sorted by popularity.

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.

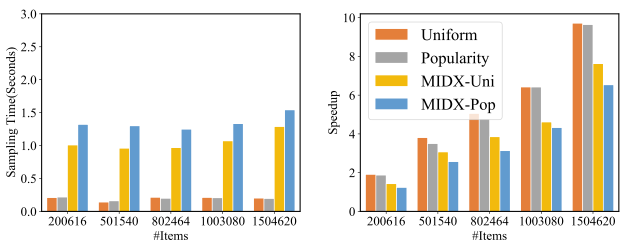
Finding 4: The MIDX-variant samplers can capture the dynamic update of the model. In particular, when the model is well-trained, the MIDX-variant samplers well approximate the softmax; when the model is only randomly initialized, most dynamic samplers are similar to the static samplers. The latter observation 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.



(a) Gowalla (b) Netflix

Figure 3: Effectiveness of different samplers in terms of recommendation performance.



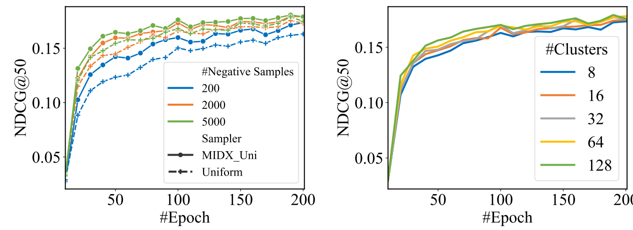
(a) Sampling Time (b) Training Speedup

Figure 4: Average Running Time v.s. Item numbers.

From the figure, we have the following finding: *The MIDX-based samplers contributes to faster convergence compared to the baseline samplers.* Compared with the static samplers, i.e. Uniform and Popularity, the MIDX-based samplers tends 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 perform 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. Though MIDX-variant samplers could produce a good approximation to the softmax, it is still unclear how efficiently items are sampled. Therefore, we increase the number of items in the Amazon dataset while keeping the sample size at 200. Experiments are run for 5 times and we report the average running time of each epoch w.r.t the sampling and training time in Figure 4. The *training Time* contains the sampling time and inference time since the sampling procedure is implemented after the user encoding. We compare the running time with the Multi-VAE and report the speedup. The running time of the Kernel sampler is not reported here because it is difficult to implement in GPUs and is substantially longer than the other samplers.

From this figure, we have following findings: *The MIDX_Uni is efficient than MIDX_Pop sampler, but less efficient than static samplers.* The static samplers require less than 0.5 seconds to sample items during each epoch, while the MIDX-based samplers take about 1.5 seconds. Indeed, as the number of items increases, the training time of Multi-VAE increases from 2.3(s) to 10.2(s), which is substantially longer than the sampling time. With the increasing of items, the training process is substantially more accelerated. The MIDX-based sampler also accelerates the training time more than six times when the number of items reaches about 1.5 million, confirming the suggested samplers' great efficiency.



(a) Number of Negative Samples (b) Number of Clusters

Figure 5: Sensitive analysis on the Gowalla dataset.

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 performances 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).

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.

[2] Artem Babenko and Victor Lempitsky. 2014. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 931–938.

[3] Artem Babenko and Victor Lempitsky. 2014. The inverted multi-index. *IEEE transactions on pattern analysis and machine intelligence* 37, 6 (2014), 1247–1260.

[4] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of RecSys'14*. ACM, 257–264.

[5] Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks* 19, 4 (2008), 713–722.

[6] Guy Blanc and Steffen Rendle. 2018. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning*. PMLR, 590–599.

[7] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. ACM, 253–262.

[8] Alexandre de Brébisson and Pascal Vincent. 2015. An exploration of softmax alternatives belonging to the spherical loss family. *arXiv preprint arXiv:1511.05042* (2015).

[9] Ian J Goodfellow. 2014. On distinguishability criteria for estimating generative models. *arXiv preprint arXiv:1412.6515* (2014).

[10] Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In *Artificial Intelligence and Statistics*. 482–490.

[11] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*. PMLR, 3887–3896.

[12] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. 297–304.

[13] Y. Hu, Y. Koren, and C. Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of ICDM'08*. IEEE, 263–272.

[14] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. 1–10.

[15] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[16] Binbin Jin, Defu Lian, Zheng Liu, Qi Liu, Jianhui Ma, Xing Xie, and Enhong Chen. 2020. Sampling-decomposable generative adversarial recommender. *Advances in Neural Information Processing Systems* 33 (2020), 22629–22639.

[17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[18] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

[19] Xiang Li, Tao Qin, Jian Yang, and Tie-Yan Liu. 2016. LightRNN: Memory and computation-efficient recurrent neural networks. In *Advances in Neural Information Processing Systems*. 4385–4393.

[20] Defu Lian, Qi Liu, and Enhong Chen. 2020. Personalized ranking with importance sampling. In *Proceedings of The Web Conference 2020*. 1093–1103.

[21] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational Autoencoders for Collaborative Filtering. In *Proceedings of WWW'18*. International World Wide Web Conferences Steering Committee, 689–698.

[22] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. 3111–3119.

[24] Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*. 2265–2273.

[25] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model.. In *Aistats*, Vol. 5. Citeseer, 246–252.

[26] Stanislav Morozov and Artem Babenko. 2018. Non-metric similarity graphs for maximum inner product search. *Advances in Neural Information Processing Systems* 31 (2018), 4721–4730.

[27] Stephen Mussmann and Stefano Ermon. 2016. Learning and inference via maximum inner product search. In *International Conference on Machine Learning*. PMLR, 2587–2596.

[28] Behnam Neyshabur and Nathan Srebro. 2015. On Symmetric and Asymmetric LSHs for Inner Product Search. In *Proceedings of ICML'15*. 1926–1934.

[29] R. Pan, Y. Zhou, B. Cao, N.N. Liu, R. Lukose, M. Scholz, and Q. Yang. 2008. One-class collaborative filtering. In *Proceedings of ICDM'08*. IEEE, 502–511.

[30] Vineeth Rakesh, Suhang Wang, Kai Shu, and Huan Liu. 2019. Linked variational autoencoders for inferring substitutable and supplementary items. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 438–446.

[31] Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using cone trees. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 931–939.

[32] Steffen Rendle and Christoph Freudenthaler. 2014. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 273–282.

[33] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of UAI'09*. AUAI Press, 452–461.

[34] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 600–608.

[35] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I Nikolenko. 2020. RecVAE: A new variational autoencoder for Top-N recommendations with implicit feedback. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 528–536.

[36] Anshumali Shrivastava and Ping Li. 2014. Asymmetric LSH (ALSH) for sublinear time Maximum Inner Product Search (MIPS). In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*. 2321–2329.

[37] Anshumali Shrivastava and Ping Li. 2014. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). *arXiv preprint arXiv:1410.5410* (2014).

[38] Ryan Spring and Anshumali Shrivastava. 2017. A new unbiased and efficient class of lsh-based samplers and estimators for partition function computation in log-linear models. *arXiv preprint arXiv:1703.05160* (2017).

[39] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[40] Da Tang, Dawen Liang, Tony Jebara, and Nicholas Ruozzi. 2019. Correlated variational auto-encoders. In *International Conference on Machine Learning*. PMLR, 6135–6144.

[41] Pascal Vincent, Alexandre de Brébisson, and Xavier Bouthillier. 2015. Efficient Exact Gradient Update for training Deep Networks with Very Large Sparse Targets. *Advances in Neural Information Processing Systems* 28 (2015), 1108–1116.

[42] Alastair J Walker. 1977. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)* 3, 3 (1977), 253–256.

[43] Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 515–524.

[44] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 6332–6338.

[45] Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning* 81, 1 (2010), 21–35.

[46] Xianwen Yu, Xiaoning Zhang, Yang Cao, and Min Xia. 2019. VAEGAN: A Collaborative Filtering Framework based on Adversarial Variational Autoencoders.. In *IJCAI*. 4206–4212.

[47] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 1–38.

[48] Ting Zhang, Chao Du, and Jingdong Wang. 2014. Composite Quantization for Approximate Nearest Neighbor Search. In *Proceedings of ICML'14*. 838–846.

[49] Weinan Zhang, Tianqi Chen, Jun Wang, and Yong Yu. 2013. Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. 785–788.

A APPENDIX

In the appendix, we provide the proofs of theorem 5.1, theorem 5.2,theorem 5.3 and theorem 5.4.

For better illustration, we review some important notations here. In the following, denote by $\mathcal{I}$ the set of items, z_u a vector of the user u, $\boldsymbol{q}_i$ a vector of the item i. The softmax probability with the inner-product logits can be compudated by:

$$Q(y_i|z_u) = \frac{\exp(z_u^\top \boldsymbol{q}_i)}{\sum_{j\in\mathcal{I}} \exp(z_u^\top \boldsymbol{q}_j)}.$$

Particularly, $\boldsymbol{q}_i$ can be decomposed based on the codebooks. It is formulated as $\boldsymbol{q}_i = \boldsymbol{c}^1_{k_1} \oplus \boldsymbol{c}^2_{k_2} + \tilde{\boldsymbol{q}}_i$ where $\boldsymbol{c}^i_{k_i}$ is the k_i-th codeword index of i-th codebook and $\tilde{\boldsymbol{q}}_i$ is the residual vector.

Theorem A.1 (**Theorem 4.1**). *Assume $z_u = z^1_u \oplus z^2_u$ is a vector of a user u, $\boldsymbol{q}_i = \boldsymbol{c}^1_{k_1} \oplus \boldsymbol{c}^2_{k_2} + \tilde{\boldsymbol{q}}_i$ is a vector of an item i, Ω_{k_1,k_2} is the set of items which are assigned to $\boldsymbol{c}^1_{k_1}$ in the first subspace and $\boldsymbol{c}^2_{k_2}$ in the second subspace. The softmax probability $Q(y_i|z_u)$ can be decomposed as follows:*

$$\begin{aligned}
Q(y_i|\boldsymbol{z}_u) &= P^1_u(k_1)\cdot P^2_u(k_2|k_1)\cdot P^3_u(y_i|k_1,k_2),\\
P^1_u(k_1) &= \frac{\psi_{k_1}\exp({z^1_u}^\top \boldsymbol{c}^1_{k_1})}{\sum_{k=1}^K \psi_k \exp({z^1_u}^\top \boldsymbol{c}^1_k)},\\
P^2_u(k_2|k_1) &= \frac{\omega_{k_1,k_2}\exp({z^2_u}^\top \boldsymbol{c}^2_{k_2})}{\underbrace{\sum_{k=1}^K \omega_{k_1,k}\exp({z^2_u}^\top \boldsymbol{c}^2_k)}_{\psi_{k_1}}},\\
P^3_u(y_i|k_1,k_2) &= \frac{\exp(z_u^\top \tilde{\boldsymbol{q}}_i)}{\underbrace{\sum_{j\in\Omega_{k_1,k_2}} \exp(z_u^\top \tilde{\boldsymbol{q}}_j)}_{\omega_{k_1,k_2}}}.
\end{aligned} \tag{6}$$

Proof.

$$\begin{aligned}
Q(y_i|\boldsymbol{z}_u) =& \frac{\exp(\boldsymbol{z}_u^\top \boldsymbol{q}_i)}{\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top \boldsymbol{q}_j)}\\
=& \frac{\exp({\boldsymbol{z}^1_u}^\top \boldsymbol{c}^1_{k_1})\exp({\boldsymbol{z}^2_u}^\top \boldsymbol{c}^2_{k_2})\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i)}{\sum_{k=1}^K \exp({\boldsymbol{z}^1_u}^\top \boldsymbol{c}^1_k)\underbrace{\sum_{k'=1}^K \exp({\boldsymbol{z}^2_u}^\top \boldsymbol{c}^2_{k'})\sum_{j\in\Omega_{k,k'}}\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_j)}_{\Psi_k}}\\
=& \frac{\Psi_{k_1}\exp({\boldsymbol{z}^1_u}^\top \boldsymbol{c}^1_{k_1})}{\sum_{k=1}^K \Psi_k\exp({\boldsymbol{z}^1_u}^\top \boldsymbol{c}^1_k)}\cdot\frac{\exp({\boldsymbol{z}^2_u}^\top \boldsymbol{c}^2_{k_2})\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i)}{\Psi_{k_1}}\\
=& P^1_u(k_1)\cdot\frac{\exp({\boldsymbol{z}^2_u}^\top \boldsymbol{c}^2_{k_2})\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i)}{\sum_{k=1}^K\exp({\boldsymbol{z}^2_u}^\top \boldsymbol{c}^2_k)\underbrace{\sum_{j\in\Omega_{k_1,k}}\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_j)}_{\omega_{k_1,k}}}\\
=& P^1_u(k_1)\cdot\frac{\omega_{k_1,k_2}\exp({\boldsymbol{z}^2_u}^\top \boldsymbol{c}^2_{k_2})}{\sum_{k=1}^K\omega_{k_1,k}\exp({\boldsymbol{z}^2_u}^\top \boldsymbol{c}^2_k)}\cdot\frac{\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i)}{\omega_{k_1,k_2}}\\
=& P^1_u(k_1)\cdot P^2_u(k_2|k_1)\cdot P^3_u(y_i|k_1,k_2).
\end{aligned}$$

□

Theorem A.2 (**Theorem 5.1**). *Suppose $P_1(\cdot)$ and $P_2(\cdot|k_1)$ remain the same as that in Theorem 4.1, $P_3(\cdot|k_1,k_2)$ is replaced with a uniform distribution, i.e. $P_3(y_i|k_1,k_2) = \frac{1}{|\Omega_{k_1,k_2}|}$ where $|\Omega_{k_1,k_2}|$ denotes the number of items in the set. Then, the proposal distribution is equivalent to:*

$$\begin{aligned}
Q_{uni}(y_i|z_u) &= \frac{\exp({z^1_u}^\top \boldsymbol{c}^1_{k_1})\exp({z^2_u}^\top \boldsymbol{c}^2_{k_2})}{\sum_{k,k'}|\Omega_{k,k'}|\exp({z^1_u}^\top \boldsymbol{c}^1_k)\exp({z^2_u}^\top \boldsymbol{c}^2_{k'})}\\
&= \frac{\exp(z_u^\top(\boldsymbol{q}_i-\tilde{\boldsymbol{q}}_i))}{\sum_{j\in\mathcal{I}}\exp(z_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))}.
\end{aligned}$$

Proof.

$$\begin{aligned}
&Q_{uni}(y_i|z_u) = P_1(k_1)\cdot P_2(k_2|k_2)\cdot P_3(y_i|k_1,k_2)\\
=&\frac{\Psi'_{k_1}\exp({z^1_u}^\top \boldsymbol{c}^1_{k_1})}{\sum_{k=1}^K\Psi'_k\exp({z^1_u}^\top \boldsymbol{c}^1_k)}\cdot\frac{\omega'_{k_1,k_2}\exp({z^2_u}^\top \boldsymbol{c}^2_{k_2})}{\sum_{k=1}^K\omega'_{k_1,k}\exp({z^2_u}^\top \boldsymbol{c}^2_k)}\cdot\frac{1}{|\Omega_{k_1,k_2}|}\\
=&\frac{\exp({z^1_u}^\top \boldsymbol{c}^1_{k_1})\exp({z^2_u}^\top \boldsymbol{c}^2_{k_2})}{\sum_{k=1}^K\sum_{k'=1}^K|\Omega_{k,k'}|\exp({z^1_u}^\top \boldsymbol{c}^1_k)\exp({z^2_u}^\top \boldsymbol{c}^2_{k'})}\\
=&\frac{\exp(z_u^\top(\boldsymbol{q}_i-\tilde{\boldsymbol{q}}_i))}{\sum_{j\in\mathcal{I}}\exp(z_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))}.
\end{aligned}$$

□

Theorem A.3 (**Theorem 5.2**). *Suppose $P_1(\cdot)$ and $P_2(\cdot|k_1)$ remain the same as that in Theorem 4.1, $P_3(\cdot|k_1,k_2)$ is replaced with a distribution derived from the popularity, i.e. $P_3(y_i|k_1,k_2) = \frac{pop(i)}{\sum_{j\in\Omega_{k_1,k_2}}pop(j)}$ where $pop(i)$ can be any metric of the popularity. Then, the proposal distribution is equivalent to:*

$$Q_{pop}(y_i|z_u) = \frac{\exp(z_u^\top(\boldsymbol{q}_i-\tilde{\boldsymbol{q}}_i)+\log pop(i))}{\sum_{j\in\mathcal{I}}\exp(z_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j)+\log pop(j))}.$$

Proof.

$$\begin{aligned}
&Q_{pop}(y_i|z_u) = P_1(k_1)\cdot P_2(k_2|k_2)\cdot P_3(y_i|k_1,k_2)\\
=&\frac{\Psi'_{k_1}\exp({z^1_u}^\top \boldsymbol{c}^1_{k_1})}{\sum_{k=1}^K\Psi'_k\exp({z^1_u}^\top \boldsymbol{c}^1_k)}\cdot\frac{\omega'_{k_1,k_2}\exp({z^2_u}^\top \boldsymbol{c}^2_{k_2})}{\sum_{k=1}^K\omega'_{k_1,k}\exp({z^2_u}^\top \boldsymbol{c}^2_k)}\cdot\frac{pop(i)}{\sum_{j\in\Omega_{k_1,k_2}}pop(j)}\\
=&\frac{pop(i)\exp({z^1_u}^\top \boldsymbol{c}^1_{k_1})\exp({z^2_u}^\top \boldsymbol{c}^2_{k_2})}{\sum_{k=1}^K\sum_{k'=1}^K\sum_{j\in\Omega_{k_1,k_2}}pop(j)\exp({z^1_u}^\top \boldsymbol{c}^1_k)\exp({z^2_u}^\top \boldsymbol{c}^2_{k'})}\\
=&\frac{\exp(z_u^\top(\boldsymbol{q}_i-\tilde{\boldsymbol{q}}_i)+\log pop(i))}{\sum_{j\in\mathcal{I}}\exp(z_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j)+\log pop(j))}.
\end{aligned}$$

□

Theorem A.4 (**Theorem 5.3**). *Assuming that the residual embedding $\|\tilde{\boldsymbol{q}}_i\| \le C$, the Kullback–Leibler divergence from the softmax distribution $Q(\boldsymbol{y}_\cdot|z_u)$ to the proposed distribution $Q_{uni}(\boldsymbol{y}_\cdot|z_u)$ can be bounded from above:*

$$0 < \mathcal{D}_{KL}\left[Q_{uni}(\boldsymbol{y}_\cdot|z_u)||Q(\boldsymbol{y}_\cdot|z_u)\right] \le 2C\|z_u\|.$$

Proof.

$$\begin{aligned}
Q(y_i|z_u) &= \frac{\exp(z_u^\top \boldsymbol{q}_i)}{\sum_{j\in\mathcal{I}}\exp(z_u^\top \boldsymbol{q}_j)},\\
Q_{\text{uni}}(y_i|z_u) &= \frac{\exp(z_u^\top(\boldsymbol{q}_i-\tilde{\boldsymbol{q}}_i))}{\sum_{j\in\mathcal{I}}\exp(z_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))},
\end{aligned}$$

$$\frac{Q_{\text{uni}}(y_i|\boldsymbol{z}_u)}{Q(y_i|\boldsymbol{z}_u)} = \frac{\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top \boldsymbol{q}_j)}{\sum_{k\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_k-\tilde{\boldsymbol{q}}_k))}\cdot\exp(-\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i)$$
$$=\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))\cdot\frac{\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_j)}{\sum_{k\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_k-\tilde{\boldsymbol{q}}_k))}\cdot\exp(-\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i)$$
$$=\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))\cdot\frac{\exp(\boldsymbol{z}_u^\top(\tilde{\boldsymbol{q}}_j-\tilde{\boldsymbol{q}}_i))}{\sum_{k\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_k-\tilde{\boldsymbol{q}}_k))}$$
$$\le\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))\cdot\frac{\exp(|\boldsymbol{z}_u^\top(\tilde{\boldsymbol{q}}_j-\tilde{\boldsymbol{q}}_i)|)}{\sum_{k\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_k-\tilde{\boldsymbol{q}}_k))}$$
$$\le\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))\cdot\frac{\exp(|\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_j|+|\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i|)}{\sum_{k\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_k-\tilde{\boldsymbol{q}}_k))}$$
$$=\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_j-\tilde{\boldsymbol{q}}_j))\cdot\frac{\exp(2C\|\boldsymbol{z}_u\|)}{\sum_{k\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_k-\tilde{\boldsymbol{q}}_k))}$$
$$=\exp(2C\|\boldsymbol{z}_u\|),$$

$$\mathcal{D}_{KL}\left[Q_{\text{uni}}(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)||Q(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)\right]=\sum_{i\in\mathcal{I}}Q_{\text{uni}}(y_i|\boldsymbol{z}_u)\log\frac{Q_{\text{uni}}(y_i|\boldsymbol{z}_u)}{Q(y_i|\boldsymbol{z}_u)}$$
$$\le\sum_{i\in\mathcal{I}}Q_{\text{uni}}(y_i|\boldsymbol{z}_u)\log\exp(2C\|\boldsymbol{z}_u\|)$$
$$=\sum_{i\in\mathcal{I}}Q_{\text{uni}}(y_i|\boldsymbol{z}_u)2C\|\boldsymbol{z}_u\|$$
$$=2C\|\boldsymbol{z}_u\|\sum_{i\in\mathcal{I}}Q_{\text{uni}}(y_i|\boldsymbol{z}_u)$$
$$=2C\|\boldsymbol{z}_u\|.$$

$\mathcal{D}_{KL}\left[Q_{\text{uni}}(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)||Q(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)\right]>0$ holds due to the non-negativity of the Kullback–Leibler divergence. □

Theorem A.5 (**Theorem 5.4**). *Assuming that the residual embedding $||\tilde{\boldsymbol{q}}_i||\le C$, the Kullback–Leibler divergence from the softmax distribution $Q(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)$ to the proposed distribution $Q_{pop}(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)$ can be bounded from above:*

$$0<\mathcal{D}_{KL}\left[Q_{pop}(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)||Q(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)\right]\le 2C||\boldsymbol{z}_u||+\log\frac{\max pop(\cdot)}{\min pop(\cdot)}.$$

Proof.

Denote $\mathcal{E}(i)=\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_i-\tilde{\boldsymbol{q}}_i)+\log pop(i))$,

$$\frac{Q_{\text{pop}}(y_i|\boldsymbol{z}_u)}{Q(y_i|\boldsymbol{z}_u)}=\frac{\sum_{j\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top\boldsymbol{q}_j)\cdot\exp(-\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i+\log pop(i))}{\sum_{k\in\mathcal{I}}\exp(\boldsymbol{z}_u^\top(\boldsymbol{q}_k-\tilde{\boldsymbol{q}}_k)+\log pop(k))}$$
$$=\frac{\sum_{j\in\mathcal{I}}\mathcal{E}(j)}{\sum_{k\in\mathcal{I}}\mathcal{E}(k)}\cdot\frac{\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_j-\log pop(j))}{\exp(\boldsymbol{z}_u^\top\tilde{\boldsymbol{q}}_i-\log pop(i))}$$
$$=\frac{\sum_{j\in\mathcal{I}}\mathcal{E}(j)}{\sum_{k\in\mathcal{I}}\mathcal{E}(k)}\cdot\exp(\boldsymbol{z}_u^\top(\tilde{\boldsymbol{q}}_j-\tilde{\boldsymbol{q}}_i))\cdot\frac{pop(i)}{pop(j)}$$
$$\le\frac{\sum_{j\in\mathcal{I}}\mathcal{E}(j)}{\sum_{k\in\mathcal{I}}\mathcal{E}(k)}\cdot\exp(2C\|\boldsymbol{z}_u\|)\cdot\frac{\max pop(\cdot)}{\min pop(\cdot)}$$
$$=\frac{\max pop(\cdot)}{\min pop(\cdot)}\cdot\exp(2C\|\boldsymbol{z}_u\|),$$

$$\mathcal{D}_{KL}\left[Q_{\text{pop}}(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)||Q(\boldsymbol{y}_\cdot|\boldsymbol{z}_u)\right]=\sum_{i\in\mathcal{I}}Q_{\text{pop}}(y_i|\boldsymbol{z}_u)\log\frac{Q_{\text{pop}}(y_i|\boldsymbol{z}_u)}{Q(y_i|\boldsymbol{z}_u)}$$
$$\le\sum_{i\in\mathcal{I}}Q_{\text{pop}}(y_i|\boldsymbol{z}_u)\log\exp(2C\|\boldsymbol{z}_u\|+\log\frac{\max pop(\cdot)}{\min pop(\cdot)})$$
$$=\log\exp(2C\|\boldsymbol{z}_u\|+\log\frac{\max pop(\cdot)}{\min pop(\cdot)})\sum_{i\in\mathcal{I}}Q_{\text{pop}}(y_i|\boldsymbol{z}_u)$$
$$=2C\|\boldsymbol{z}_u\|+\log\frac{\max pop(\cdot)}{\min pop(\cdot)}.$$

□