# Adap-$\tau$: Adaptively Modulating Embedding Magnitude for Recommendation

Jiawei Chen*,‡
leepyhunt@zju.edu.cn
Zhejiang University
China

Junkang Wu*
jkwu0909@gmail.com
University of Science and Technology of China
China

Jiancan Wu
wujcan@gmail.com
University of Science and Technology of China
China

Sheng Zhou
zhousheng_zju@zju.edu.cn
Zhejiang University
China

Xuezhi Cao
caoxuezhi@meituan.com
Meituan
China

Xiangnan He†
xiangnanhe@gmail.com
University of Science and Technology of China
China

## ABSTRACT

Recent years have witnessed the great successes of embedding-based methods in recommender systems. Despite their decent performance, we argue one potential limitation of these methods — the embedding magnitude has not been explicitly modulated, which may aggravate popularity bias and training instability, hindering the model from making a good recommendation. It motivates us to leverage the embedding normalization in recommendation. By normalizing user/item embeddings to a specific value, we empirically observe impressive performance gains (9% on average) on four real-world datasets. Although encouraging, we also reveal a serious limitation when applying normalization in recommendation — the performance is highly sensitive to the choice of the temperature $\tau$ which controls the scale of the normalized embeddings.

To fully foster the merits of the normalization while circumvent its limitation, this work studied on how to adaptively set the proper $\tau$. Towards this end, we first make a comprehensive analyses of $\tau$ to fully understand its role on recommendation. We then accordingly develop an adaptive fine-grained strategy Adap-$\tau$ for the temperature with satisfying four desirable properties including adaptivity, personalized, efficiency and model-agnostic. Extensive experiments have been conducted to validate the effectiveness of the proposal. The code is available at https://github.com/junkangwu/Adap_tau.

## CCS CONCEPTS

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

∗ Jiawei Chen and Junkang Wu contribute equally to the work;
† Xiangnan He is the corresponding author;
‡ Work mostly done at the University of Science and Technology of China (USTC).


*WWW '23, April 30-May 4, 2023, Austin, TX, USA*

ACM ISBN 978-1-4503-9416-1/23/04…$15.00
https://doi.org/10.1145/3543507.3583363

## KEYWORDS

recommendation system, temperature, adaptiveness

**ACM Reference Format:**
Jiawei Chen*,‡, Junkang Wu*, Jiancan Wu, Sheng Zhou, Xuezhi Cao, and Xiangnan He†. 2023. Adap-$\tau$: Adaptively Modulating Embedding Magnitude for Recommendation. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30-May 4, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3543507.3583363

## 1 INTRODUCTION

Being able to provide personalized suggestions, recommender system (RS) has been widely applied in numerous applications such as social media [4, 24], advertising [13, 16] and E-commerce [39]. Representation learning is a common paradigm in recommendation, ranging from early matrix factorization (MF) [23] to recent advanced graph-based models [10, 32, 35]. It learns user/item representation (i.e., embeddings) from the historical interactions and then makes a prediction based on the embedding similarity. Inner product inherited from MF has been widely applied for measuring embedding similarity, not only because it achieves competitive performance in practice but supports efficient retrieval.

Despite the success, we argue that existing embedding-based methods may not be sufficient for generating satisfactory recommendation — i.e., they do not explicitly modulate the embedding magnitude, which may incur two serious problems, as revealed in our both theoretical and empirical analyses: 1) The free-varying magnitude potentially aggravates the popularity bias. Specifically, we find that the embedding magnitude of popular items grows much more quickly than unpopular items. Those popular items usually exert excessive contribution to model training and finally obtain undesirable higher scores. 2) The highly diverse magnitude hurts model convergence. Through our visual analysis, it is found that even with a proper regularizer, the magnitude of item embeddings is still in a state of rising rather than converging even with numerous epochs.

Being aware of the weaknesses of uncontrolled embedding magnitude, it would be natural to leverage embedding normalization in recommendation. Although embedding normalization has been touched a lot in other fields [8, 31], it is less explored in RS. By

normalizing user/item embeddings into a specific value w.r.t. temperature $\tau$ [1] (i.e., $1/\sqrt{\tau}$), we observe impressive performance gains ranging from 5% to 20% on four benchmark real-world datasets. Although encouraging, we also reveal a limitation of applying normalization in recommendation — the performance is highly sensitive to the choice of the temperature $\tau$ that controls the scale of the normalized embeddings. Even a small fluctuation (e.g., 0.04) would cause a dramatic performance reduction (sometimes over 10%). Worse still, the proper $\tau$ may evolve with the data and model shift. Finding the optimal $\tau$ could be extremely hard, which involves a tedious and expensive grid search, heavily hindering the application of the normalization.

To fully foster the merits of the normalization and circumvent its limitation, this work studied an unexplored problem — *how to adaptively set the proper $\tau$ without requiring notorious hyperparameter tuning*. Towards this end, we first make comprehensive analyses of $\tau$ and reveal its two important roles in model learning:

- Leveraging temperature could adjust the magnitude of the gradient, while too small or too large $\tau$ would both increase the risk of gradient vanishment.
- The temperature $\tau$ balances the contributions from the hard negative instances and easy instances. A smaller $\tau$ would make the model pay more attention to hard negative items while a larger $\tau$ make the model treat them equally.

Being aware of the role of the temperature in recommendation, we deduce the following two principles to guide the design of the adaptive strategy:

**Principle 1.** *Adaption principle: temperature should be adaptive to avoid gradient vanishing.*

Principle 1 corresponds to the core role of $\tau$ — avoiding gradient vanishment. Considering the gradient could vary widely with the data distribution and the model changing, $\tau$ should be adapted accordingly.

**Principle 2.** *Fine-grained principle: it is beneficial to specify the temperature in a user-wise manner — i.e., the harder the samples of a user are distinguished, the larger the temperature should be employed for the user.*

Principle 2 is motivated by the hard-mining property. Note that in a typical RS, the data quality usually varies greatly from user to user [5]. For the users with much noisy feedback, the model should be more conservative with lifting $\tau$, as the hard samples are likely to be abnormal. Instead, for those users whose feedback is clear and sufficient, lowering $\tau$ to be more aggressive could bring more informative items and enhance model convergence and discrimination. As such, we believe that fine-grained $\tau$ that can adapt to the diverse hardness of different users would be better.

Based on the aforementioned principles, we propose an adaptive fine-grained strategy named Adap-$\tau$ for specifying the temperature. Towards Principle One, we delve into a global benchmark temperature that maximizes the cumulated magnitude of the gradients. The task is non-trivial, as conventional optimization would involve nested iteration and heavy traversal, incurring serious efficiency problems. Thus, here we develop a skillful approximation and derive a simple close-formed solution for acceleration. Towards principle two, we monitor the loss for each user and adapt the temperature accordingly — a larger loss suggests that samples of the user are hard to be differentiated, which would adaptively increase the value of $\tau$ to reduce the difficulty.

Finally, we emphasize that Adap-$\tau$ has the following desirable advantages: 1) Adaptability: it is fully adapted to different datasets without requiring any hyper-parameter searching about $\tau$. 2) Personalized: it gives a personalized $\tau$ that can adapt to the diverse hardness of different users. 3) Efficiency: it just involves simple computation without requiring any extra iteration. 4) Model-agnostic: it can be easily plugged in existing embedding-based methods (e.g., MF [23], LightGCN [10]) with few codes amended.

**Contributions.** We summarize the contributions as below:

- Revealing the essential of leveraging embedding normalization in recommendation, and identifying one potential limitation — performance is highly sensitive to the temperature $\tau$.
- Conducting thorough analyses of the temperature $\tau$, uncovering its two important roles in model training and delivering two principles for the temperature specification.
- Proposing an adaptive fine-grained strategy for the temperature with satisfying the four desirable properties including adaptivity, personalized, efficiency, and model-agnostic.
- Conducting extensive experiments on four datasets to demonstrate the superiority of our proposal in multiple aspects of recommendation accuracy, adaptivity, and efficiency.

## 2 PRELIMINARIES

In this section, we present some background of recommendation.

**Task Formulation.** Suppose we have a recommender system with a user set $\mathcal{U}$ and an item set $\mathcal{I}$. Let $n$ and $m$ denote the number of users and items in RS. The collected implicit feedback can be expressed by a matrix $Y \in \{0,1\}^{n\times m}$, whose element $y_{ui}$ represents whether a user $u$ has interacted (e.g., click) with an item. For convenience, we collect the whole positive instances as $\mathcal{D} \equiv \{(u,i)|y_{ui}=1\}$; and the positive items (users) for each user $u$ (item $i$) as $\mathcal{P}_u \equiv \{i|y_{ui}=1\}$ ($\mathcal{P}_i \equiv \{u|y_{ui}=1\}$). The task of RS is to recommend items for each user that he may be interested in.

**Embedding-based Model.** Embedding-based methods are widely utilized in RS. They would first transform user/item features (e.g., IDs) into vectorized representations (i.e., $\mathbf{e}_u, \mathbf{e}_i$), and then make predictions based on the embedding similarity. The widely-used similarity functions include inner product [14] and neural network [11]. As suggested by recent work [19, 33, 36], the inner product supports highly efficient retrieval and usually exhibits stronger performance. Thus, for convenience, this work simply takes the representative inner product for analyses, i.e., model prediction can be expressed as $\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i$.

**Loss function.** There are multiple choices of loss functions for training a recommendation model including pointwise loss (e.g., BCE [12, 26], MSE [9, 17]), pairwise loss(e.g., BPR [25]) and Softmax loss [34]. Recent work [34] finds Softmax loss could mitigate popularity bias, achieves great training stability, and aligns well with the ranking metric. It usually achieves better performance than others and thus attracts a surge of interest in recommendation. In addition, Softmax loss can be considered as an extension of commonly-used BPR loss [25]. As such, we cast Softmax as the representative loss

[1] Instead of directly introducing a parameter for controlling the scale of the normalization, here we refer to recent work that usually utilizes a temperature.

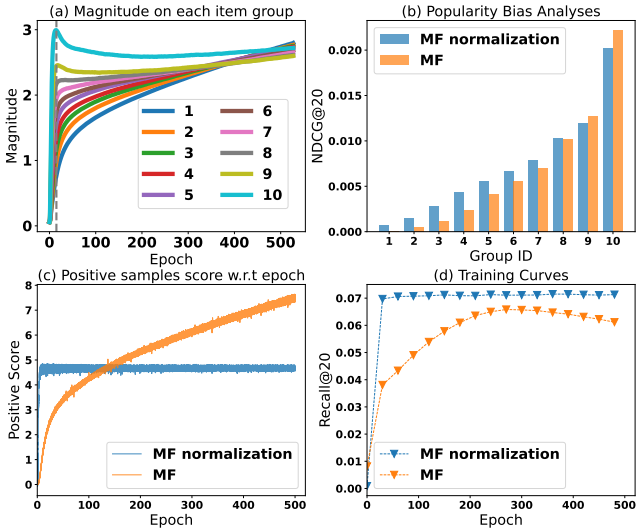


**Figure 1: Empirical studies on Yelp2018: Fig. (a) and Fig. (b) represent item embedding magnitude of the different groups across the training procedure and respective performance. The larger GroupID is, the more popular items the group contains. Fig. (c) and Fig. (d) depict the positive samples score and corresponding performance in the training procedure.**

for analyses, which can be formulated as:

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|} \sum_{(u,i)\in\mathcal{D}} \log \frac{\exp(\hat{y}_{ui})}{\sum_{j\in\mathcal{I}} \exp(\hat{y}_{uj})} \tag{1}$$

In practice, we usually conduct negative sampling or mini-batch strategy [22] for acceleration. But they are not our focus and here we simply refer to the original loss for theoretical analyses.

**Embedding Normalization.** This work studies the nature of embedding normalization in recommendation. On the basis of inner product, we leverage embedding normalization in prediction as:

$$\hat{y}_{ui} = \frac{\mathbf{e}_u^\top \mathbf{e}_i}{\|\mathbf{e}_u\| \, \|\mathbf{e}_i\|} \cdot \frac{1}{\tau} \tag{2}$$

where the magnitude of user/item embeddings has been rescaled. The first factor:

$$f(u,i) = \frac{\mathbf{e}_u^\top \mathbf{e}_i}{\|\mathbf{e}_u\| \, \|\mathbf{e}_i\|} \tag{3}$$

can be understood as cosine similarity, where the magnitude has been isolated; and the second factor $1/\tau$ rescales the normalized embeddings. We remark that instead of directly introducing a parameter controlling the scale, we borrow a similar idea in contrastive learning [3, 8] and utilize the conventional temperature. The alignment could make our findings better generalized to other domains.

## 3 ANALYSES OVER EMBEDDING NORMALIZATION

In this section, we first validate the essential of leveraging embedding normalization in RS (Sec. 3.1), and then identify one potential limitation (Sec. 3.2). Finally, we conduct thorough analyses of the temperature and uncover its two important roles (Sec. 3.3).

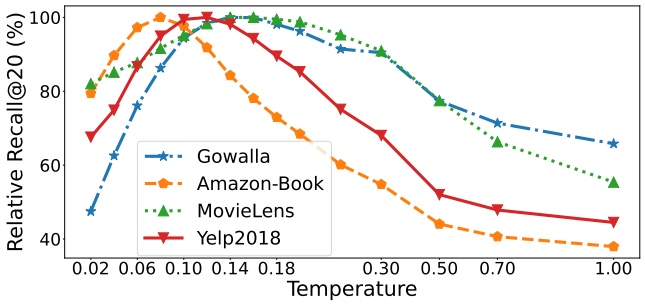


**Figure 2: Relative recall@20 over four datasets with $\tau$ .**

**Table 1: Performance comparisons of MF with/without embedding normalization. The column of "norm" represents whether to conduct normalization for the user or item representation. For example, Y-N stands for adopting normalization on the user side but not on the item side. Here we simply report the results on Yelp and Amazon-book. But similar results can be obtained on other datasets, as presented in Table 2.**

| norm? | Yelp2018 | | Amazon-book | |
|---|---|---|---|---|
| | Recall | NDCG | Recall | NDCG |
| N-N | 0.0677 | 0.0554 | 0.0457 | 0.0352 |
| Y-N | 0.0709 | 0.0585 | 0.0529 | 0.0419 |
| N-Y | 0.0703 | 0.0577 | 0.0513 | 0.0399 |
| Y-Y | **0.0714** | **0.0586** | **0.0542** | **0.0422** |

### 3.1 Necessity of Normalization

*3.1.1* ***Theoretical Analysis.*** We start with theoretical analysis to show that without normalization the magnitude of popular items grows much more quickly than unpopular items. In fact, we have:

Lemma 1. *By choosing inner product without controling magnitude, we have change of item embedding magnitude $\delta_i$ in each iteration:*

$$\delta_i = \sum_{u\in\mathcal{P}_i} 2\eta \, \frac{|\mathcal{P}_u| + 1}{m \cdot \mathsf{E}_{j\in\mathcal{I}} \exp(\tilde{f}(u,j) - \tilde{f}(u,i))} - 1 \; \tilde{f}(u,i) \tag{4}$$

*At the early stage of the training procedure, $\delta_i$ obeys:*

$$\delta_i \propto |\mathcal{P}_i| \tag{5}$$

*where $\tilde{f}(u,i) = (e_u^T \cdot e_i)$ denotes the inner product of embedding without normalizaiton, $|\mathcal{P}_u|$ and $|\mathcal{P}_i|$ represents the frequency of user $u$ and item $i$, and $\mathcal{P}_i$ denotes the set of users observed in $\mathcal{D}$ which are interactived with $i$.*

The proof of the lemma is presented in Appendix B.1. We can draw an observation from Lamma 1: Note that at the early stage of the training procedure, users and items are distributed uniformly. In other words, $\exp(\tilde{f}(u,j) - \tilde{f}(u,i))$ and $\tilde{f}(u,i)$ cannot tell remarkable difference, while the magnitude of popular items will obtain explosive rising in term of $|\mathcal{P}_i|$.

*3.1.2* ***Empirical Analysis.*** From Lemma 1, we know that the magnitude is correlated with item popularity. In this subsection,

we explore its negative impact on recommendation through rich experiments.

**Experiments design.** To show the impact of free-varying magnitude, here we conduct four experiments: **(1)** We first visualize the magnitude of item embedding with different item popularity during the training (Fig. 1 a). Here we follow [34] and split items into ten groups in terms of item popularity. The larger group ID indicates the group contains more popular items. **(2)** We also report the performance in terms of different item groups (Fig. 1 b). **(3)** The predictive scores of positive instances with training epochs is presented in Fig. 1 c. **(4)** We visualize the performance of MF with or without normalization (Fig. 1 d). All experiments are conducted on the MF backbone and Yelp2018 [10] dataset. Similar results can be observed on other models (like LightGCN) and datasets. The details of experimental settings can refer to section 5.1.

**Free-varying magnitude aggravates popularity bias.** If we focus on the early stage of the training (cf. Fig. 1 (a)), the magnitude of popular items rises rapidly which is consistent with theoretical proofs. Therefore, popular items are prone to obtain higher scores as the magnitude directly contributes to model prediction. Besides, the diverse magnitude also hurt the training of the user embedding. The gradient of user embedding can be written as: $\frac{\partial L}{\partial \mathbf{e}_u} = \sum_{u,i} \frac{\partial L}{\partial f(u,i)} \mathbf{e}_i$, where popular items with larger magnitude would exert excessive contribution and potentially overwhelm the signals from others. The model would sink into biased results. Fig. 1 (b) provides the evidence. The model would sink into biased results. As can be seen, the model with normalization yields much fairer results than the model without normalization.

**Free-varying magnitude hurts convergence.** If we turn our attention to the end of training in Fig. 1 (c), we observe that even with numerous epochs (e.g., 500), the predicted scores and embedding magnitude of vanilla MF are still in a state of rising rather than convergence while the performance drop consistently (Fig. 1 (d)). But when we leverage normalization in MF, we observe impressive improvement — the model arrives at convergence quickly with much fewer epochs (i.e., 20) and performs stable with more epochs.

**Normalization boosts performance.**. To further validate the merit of the normalization, here we directly test the recommendation performance with or without conducting normalization on the user or item embeddings (Table 1). As can be seen, the model with both-side normalization (i.e., Y-Y) remarkably outperforms the model with one-side normalization (i.e., Y-N or N-Y); and they both surpass the model without normalization (N-N).

## 3.2 Limitation of Normalization

Although we have proved the superiority of normalization in recommendation tasks, here reveal one potential limitation of the normalization — the performance is highly sensitive to the temperature $\tau$.

To validate this point, we test the recommendation performance w.r.t. $\tau$ ranging from 0.02 to 1 with a rather fine-grained step-size 0.02. The result is shown in Fig. 2, where we report the relative performance with the best for better visualization. We make the following observations: 1) the performance is highly sensitive to $\tau$. Even a small fluctuation (e.g., changing from 0.08 to 0.12 on Amazon-Book) would cause a dramatic performance reduction (e.g., 10%); 2) Different datasets require rather different $\tau$. For example, Amazon-Book dataset reaches the best performance when $\tau = 0.08$, but MoiveLens reaches with $\tau = 0.16$. If we simply transfer the optimum $\tau$ in one dataset (e.g., Moivelens) to another (e.g., Amazon-Book), we would get rather poor performance (e.g., over 30% reduction).

As a result, finding optimal $\tau$ is highly difficult, which heavily hinders the application of embedding normalization. Methods like grid search or automated hyperparameter search [6] are potential to find the optimum, but they are highly time-cost expensive. As such, we believe it is essential to pursue a automatic mechanism that could specify the proper $\tau$ adaptively.

## 3.3 Roles of Temperature

In this subsection, we make a comprehensive analysis of $\tau$ and reveal its two important roles in model learning.

*3.3.1* ***Avoiding gradient vanishment.*** The temporature mainly affect the gradient of the loss function $L$ w.r.t. $f(u,i)$. For convenient, let notation $p_{ui}(\tau)$ be the logit of the instance $(u,i)$ governed by the parameter $\tau$, i.e.,

$$p_{ui}(\tau) = \frac{exp(\frac{f(u,i)}{\tau})}{\sum_{j \in \mathcal{I}} \exp(\frac{f(u,j)}{\tau})} \tag{6}$$

The gradient $\frac{\partial L}{\partial f(u,i)}$ can be written as:

$$\frac{\partial L}{\partial f(u,i)} = \begin{cases} \frac{1}{\tau} p_{ui}(\tau)(1 - \sum_{k \in \mathcal{P}_u} p_{uk}(\tau)), & \text{for} \quad y_{ui} = 1 \\ -\frac{1}{\tau} p_{ui}(\tau)(\sum_{k \in \mathcal{P}_u} p_{uk}(\tau)), & \text{for} \quad y_{ui} = 0 \end{cases} \tag{7}$$

The expected magnitude of the graidents can be written as:

$$\mathbb{E}_i[|\frac{\partial L}{\partial f(u,i)}|] = \frac{2}{m\tau} \sum_{i \in \mathcal{P}_u} p_{ui}(\tau)(1 - \sum_{k \in \mathcal{P}_u} p_{uk}(\tau)) \tag{8}$$

which can be understood as the product of the sum of positive logits ($\sum_{i \in \mathcal{P}_u} p_{ui}(\tau)$) and the sum of the negative logits (1- $\sum_{k \in \mathcal{P}_u} p_{uk}(\tau)$). When $\tau$ is too small, due to the explosion nature of exponential function, the disparity on $f(u,i)$ would be amplified, and positive instances usually obtain extremely larger logits than negative (e.g., $\sum_{i \in \mathcal{P}_u} p_{ui}(\tau) \to 1$). The gradient would vanish. On the contrary, when $\tau$ is too large, the logits $p_{ui}$ do not exhibit much difference. But due to the long tail nature of RS — i.e., the number of negative instances is much larger than positive, the sum of positive logits would be quite small and the gradient vanishes again.

Appendix C.2 provides an example of how the gradient magnitude varies with the temperature $\tau$. As can be seen, too large or too small $\tau$ would cause gradient vanishment. As such, $\tau$ should be specified carefully and adapted for obviating gradient vanishment.

*3.3.2* ***Hard-mining.*** Hard-mining of $\tau$ has been uncovered by some recent work in contrastive learning [30]. Here we borrow their ideas but provide more insightful analyses in terms of RS scenarios. As discussed before, the exponential function with small $\tau$ would amplify the disparity. Hence those hard negative samples with larger $f(u,i)$ would have extremely larger logits $p_{ui}$, contributing more on model training. On the contrary, larger $\tau$ tends to make the model treat the negative samples equally.

This property highly motivates us to give a user-wise $\tau$. Note that in a typical RS, the data quality usually vary greatly from user to user. For the users with much noisy feedback, it would be unwise to concentrate much on the hard negative samples, as they are likely to be noisy samples. But for those users with clear and sufficient feedback, lowering $\tau$ would be a better choice as it could bring more informative samples and thus enhances model convergence and discrimination. As such, continuing on the habit of fixed $\tau$ is no longer a wise choice. It would be better to give fine-grained $\tau$ that can adapt to the diverse hardness of different users.

More interestingly, this treatment could bring another advantage. From eq.(7), we find the gradient is discounted by $1/\tau$. Giving personalized $\tau$ could also adjust the confidence of users – i.e., the users with higher-quality data would make more contributions on training.

## 4 PROPOSED METHOD

To address this problem, in this section, we propose *Adap-$\tau$* that is able to adaptively and automatically modulate the embedding magnitude in recommender system. Ada-$\tau$ is developed based on the following principles:

- *(P1) Adaption principle: temperature should be adaptive to avoid gradient vanishing.*
- *(P2) Fine-grained principle: it is beneficial to specify the temperature in a user-wise manner — i.e., the harder the samples of a user are distinguished, the larger temperature should be employed for the user.*

### 4.1 Adap-$\tau_0$: Towards Adaptive Temperature

Towards principle (P1), we delve into an automatic temperature that maximizes the magnitude of the gradients:

$$\tau_0 = \arg\max_{\tau} \mathsf{E}_{u\in U, i\in I}[|\frac{\partial L}{\partial f(u,i)}|] \tag{9}$$

Directly optimizing the equation (9) is computationally infeasible, as it would involve nested optimization and heavy traversal over each user-item pair. Thus, we turn to pursue an efficient approximated solution. Here we first derive the tight upper bound of the objective, which can be easily optimized. In fact, we have:

Lemma 2. *Let $p_{ui}(\tau)$ be the logit of the instance $(u,i)$ governed by the parameter $\tau$, i.e., $p_{ui}(\tau) = exp(f(u,i)/\tau)/\sum_i \exp(f(u,i)/\tau)$ and $\tau$ be lower bounded[2] by T. The objective is bounded with:*

$$\mathsf{E}_{u,i}[|\frac{\partial L}{\partial f(u,i)}|] \le \frac{2}{mT}(\mathsf{E}_u[\tilde{\mathrm{O}}_{i\in N_u} p_{ui}(\tau)] - \mathsf{E}_u^2[\tilde{\mathrm{O}}_{i\in N_u} p_{ui}(\tau)]) \tag{10}$$

*The optimum of the upper bound is achieved iff the following condition holds:*

$$E_u[\tilde{\mathrm{O}}_{i\in N_u} p_{ui}(\tau)] = \frac{1}{2} \tag{11}$$

The proof of the lemma is presented in Appendix B.2. Now the question lies on solving the Equation (11), which is still intractable. Here we explore a reasonable approximate and have:

[2]In practical, we usually control the lower bound of $\tau$ to guarantee numerical stability.

Lemma 3. *Let $\mathsf{F}$ (or $\mathsf{F}^+$) be the distribution of $f(u,i)$ over all instances (or positive instances). Let $\mathbf{f}$ (or $\mathbf{f}^+$) be a random variable that sampled from $\mathsf{F}$ (or $\mathsf{F}^+$). Suppose the distribution $\mathsf{F}$ and $\mathsf{F}^+$ have a sub-exponential tail such that the following conditions hold for some $\lambda, \lambda_+ > 0$:*

$$\begin{aligned} p((\mathbf{f} - \mathsf{E}_{\mathsf{F}}[\mathbf{f}]) > b) &\le 2e^{-2b/\lambda} \\ p((\mathbf{f}_+ - \mathsf{E}_{\mathsf{F}_+}[\mathbf{f}_+]) > b) &\le 2e^{-2b/\lambda_+} \end{aligned} \tag{12}$$

*When $\tau_0 \geqslant \max(2\lambda, 2\lambda_+, T)$, it can be approximated as:*

$$\tau_0 \approx \frac{\sigma_+^2 - \sigma^2}{-(\mu^+ - \mu) + \sqrt{(\mu^+ - \mu)^2 + 2(\sigma_+^2 - \sigma^2)\log(\frac{nm}{2|D|})}} \tag{13}$$

*where $|D|$ denotes the number of positive instances in the datasets, $\mu$ (or $\mu_+$) and $\sigma^2$ (or $\sigma_+^2$) denotes the mean and variance of $\mathbf{f}$ (or $\mathbf{f}_+$). when $\sigma_+^2$ is close to $\sigma^2$ (cf. Appendix C.1), the expression can be simplified as:*

$$\tau_0 \approx \frac{\mu_+ - \mu}{\log(\frac{nm}{2|D|})} \tag{14}$$

The proof of the lemma is presented in Appendix B.3. Here we make a hypothesis on the distribution — i.e., $\mathsf{F}$ and $\mathsf{F}_+$ are convergent and the tails of the distribution decay at least as fast as exponential one (that decay as $e^{-2t/\lambda}$). The hypothesis is practical as the sub-exponential random variables is actually common. It contains Guassian, exponential, Gamma, Pareto, Cauchy, etc.. Besides, Hoeffding [1] proofed that all bounded random variables are sub-exponential.

In fact, in our experiments, we always observe that $f$ and $f_+$ are convergence into a specific region, with a pretty small $\lambda$ and $\lambda_+$. Also, we observe that the two distribution usually has a quite close variance (cf. Appendix C.1). These observations validate the Equation (14) can be safely applied. Our empirical study presented in Section 5 also validate the superiority of the proposed strategy.

### 4.2 Adap-$\tau$: Towards Adaptive Fine-grained Temperature

Towards principle (2), we introduce personalized temperatures $\tau_u$ for each user and leverage a *Superloss* [2] to supervise their learning. Specifically, the role of Superloss is to monitor the loss of the samples for each user, and to adaptively adjust the temperature values accordingly. It is composed of a loss-aware term and a regularization term:

$$J = \frac{L(u) - m_u}{\tau_u} + \beta(\log\tau_u - \log\tau_0)^2 \tag{15}$$

where $L(u)$ denotes the cumulated loss of the samples of a specific $u$, reflecting how difficult the samples to be differentiated by the model. $m_u$ is a threshold that ideally separates easy samples from hard samples based on their respective loss, which can be set as the mean of the $L(u)$ in practical. The larger $L(u)$ would bring the more penalty, which would reduce the $1/\tau_u$ to a larger extent and adaptively pushes the temperature towards larger value. Remarkably, to make a fair comparison of $L(u)$ over users, here we choose the common temperature to calculate $L(u)$.

Meanwhile, to prevent the temperature from sinking into extreme values that incurs gradient vanishing, a regularizer has also been introduced. This regularizer aims at pulling the learned $\tau_u$ to

**Table 2: Performance comparison between Adap-$\tau$ and other similar strategy. 'No-Norm' denoted the method without normalization. 'Grid Search $\tau$' denoted the method leveraging grid search to find optimal $\tau$. 'C-$\tau$' and 'Cu-$\tau$' utilize the neural network to learn $\tau$ following the work [31].**

| Backbone | strategy | Yelp2018 | | Amazon-book | | Movielens | | Gowalla | |
|---|---|---|---|---|---|---|---|---|---|
| | | Recall | NDCG | Recall | NDCG | Recall | NDCG | Recall | NDCG |
| MF | No Norm | 0.0677 | 0.0554 | 0.0457 | 0.0352 | 0.2721 | 0.2525 | 0.1616 | 0.1366 |
| | Grid Search $\tau$ | 0.0714 | 0.0586 | 0.0542 | 0.0422 | 0.2789 | 0.2624 | 0.1761 | 0.1399 |
| | C-$\tau$ | 0.0647 | 0.0528 | 0.0538 | 0.0418 | 0.2472 | 0.2260 | 0.1723 | 0.1362 |
| | Cu-$\tau$ | 0.0691 | 0.0566 | 0.0541 | 0.0421 | 0.2600 | 0.2398 | 0.1751 | 0.1383 |
| | Adap-$\tau_0$ | 0.0714 | 0.0585 | 0.0549 | 0.0427 | 0.2792 | 0.2638 | 0.1754 | 0.1386 |
| | Adap-$\tau$ | **0.0721** | **0.0594** | **0.0553** | **0.0430** | **0.2815** | **0.2673** | **0.1838** | **0.1506** |
| LightGCN | No Norm | 0.0649 | 0.0530 | 0.0411 | 0.0315 | 0.2576 | 0.2427 | 0.1830 | 0.1554 |
| | Grid Search $\tau$ | 0.0730 | 0.0605 | 0.0596 | 0.0477 | 0.2767 | 0.2575 | 0.1878 | 0.1577 |
| | C-$\tau$ | 0.0653 | 0.0537 | 0.0571 | 0.0453 | 0.2529 | 0.2282 | 0.1731 | 0.1431 |
| | Cu-$\tau$ | 0.0690 | 0.0571 | 0.0586 | 0.0468 | 0.2582 | 0.2357 | 0.1797 | 0.1488 |
| | Adap-$\tau_0$ | 0.0724 | 0.0603 | 0.0601 | 0.0480 | 0.2744 | 0.2571 | 0.1841 | 0.1526 |
| | Adap-$\tau$ | **0.0733** | **0.0612** | **0.0612** | **0.0490** | **0.2787** | **0.2615** | **0.1901** | **0.1590** |

be close to the $\tau_0$ that has approximately largest gradient magnitude. A parameter $\beta$ is introduced to balance both effects. It can be simply set as 1.0 without requiring grid search. Instead of learning the optimal $\tau_u$ via back-propagation, we prefer to find the close-formed solution, which does not involve extra iteration causing efficiency issue or notorious fine-tuning on the extra parameters of the learning rate or decay. In fact, we have a closed-form solution from Eq. (15):

$$\tau_u^* = \tau_0 \cdot \exp(\mathsf{W}(\max(-\frac{1}{e}, \frac{L(u) - m_u}{2\beta}))) \tag{16}$$

where $\mathsf{W}(.)$ stands for the Lambert-W function, which is the inverse function of $xexp(x)$. As intended, $\tau_u^*$ is monotonically increasing with the user loss $L(u)$ — the user with a larger loss would acquire a larger temperature $\tau_u$ to down weight the confidence of the user. Meanwhile, the $\tau_0$ acts as a baseline to scale the temperature into a proper region.

## 4.3 Discussion

We show that the proposed Adap-$\tau$ satisfies the following three desirable properties:

**Personalization.** As for user-wise adaption, owing to the capability of *Superloss*, our model could calculate the specific $\tau$ in terms of their cumulated loss. The larger train loss suggests the data may contain more noises, and thus drives the model to be more conservative. $\tau$ would become larger accordingly to slow the pace of hard-mining and down-weigh the contribution of this user.

**Adaption.** As for data-wise adaption, our Equation (9) automatically computes proper $\tau$ without extra manual intervention, thus avoiding the notorious hyper-parameter search for $\tau$.

**Model-agnostic:** Actually, our Adap-$\tau$ can be easily plugged in many embedding-based methods. We do not deeply intervene on the model, but simply introduce embedding normalization and adaptive temperatures calculated from Eq. (16).

**Efficiency:** Our calculation about adaptive $\tau$ is a straightforward close-formed without requiring extra iteration. Also, the temperature can be calculated efficiently. As for $\mu^+$, we compute it with element-wise multiplication of positive instances which costs $O(2|\mathcal{D}|d)$, where $|\mathcal{D}||$ denotes the number of positive instances and $d$ represents dimension. And $\mu$ is calculated by the cosine similarity between users and mean value of all items representation for simplicity, which cost $O(nd)$, where $n$ denotes the number of users. Its total complexity is $O(2|E|d + nd)$ without backward cost. More detailed analyses could refer to Appendix D.

**Table 3: Statistics of the datasets**

| Dataset | #Users | #Items | #Interactions | Density |
|---|---|---|---|---|
| Yelp2018 | 31,831 | 40,841 | 1,666,869 | 0.0128% |
| Amazon-Book | 52,643 | 91,599 | 2,984,108 | 0.062% |
| Movielens | 6,022 | 3,043 | 995,154, | 5.431% |
| Gowalla | 29,858 | 40,981 | 1,027,370 | 0.084% |

# 5 EXPERIMENTS

In this section, we present comprehensive experiments to demonstrate the effectiveness of our model. Our experiments are intended to address the following research questions::

- **RQ1:** How does Adap-$\tau$ perform compared with other strategies?
- **RQ2:** Does our Adap-$\tau$ adapt to different datasets and users?
- **RQ3:** How does the model equipped with embedding normalization and adaptive $\tau$ perform compared with state-of-the-art in terms of both accuracy and efficiency?

## 5.1 Experimental Settings

*5.1.1* ***Datasets and Metrics.*** We adopt four real-world datasets, Yelp2018[10], Amazon-book [10], Movielens [37] and Gowalla [12], to evaluate our model. For pair comparison, the Amazon-book, Yelp2018, and Gowalla are exactly the same as [10] used. The MovieLens is from [37] which is collected from the website *movielens.umn.edu* and we use the version of 1M. Following [10, 32], we leverage the routine strategy — 10-core setting to preprocess the dataset. After standardization, we report the statistics of the above dataset in Tab. 3. As for evaluation metrics, we adopt all-ranking protocol to compute recall@20 and ndcg@20.

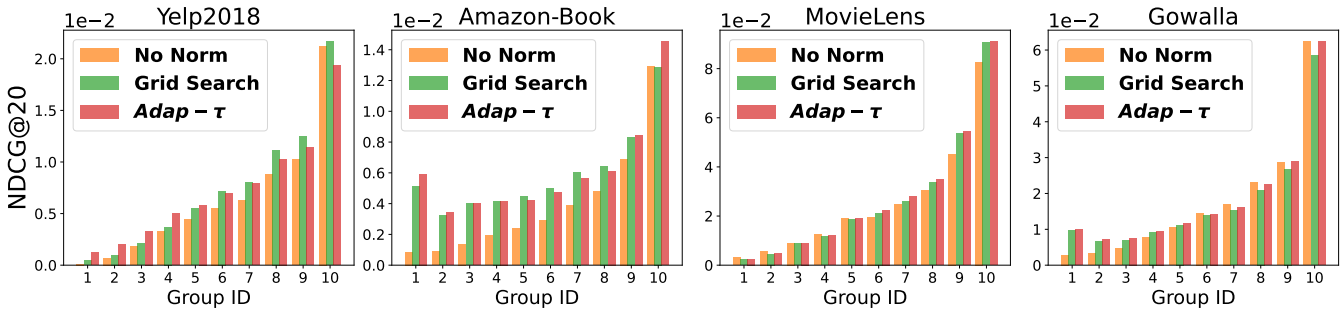


**Figure 3: Performance comparison over different item groups among different strategies.**

*5.1.2* ***Baselines.*** We validate the effectiveness of our method on two representative backbones: MF and LightGCN. Six strategies are tested in our experiments. Compare to our Adap-$\tau$, we also adopt the following strategies:

- No norm: Adopting inner product where the user/item embeddings have not been normalized.
- Grid Search $\tau$: Normalizing the embeddings into a specific value (i.e., $1/\overline{p}(\tau)$), where $\tau$ is specified via fine-grained grid search (i.e., step-size=0.02).
- C-$\tau$: Following a similar topic in computer vision [31], and leveraging a neural network to directly learn $\tau$ from the objective function.
- Cu-$\tau$: A stronger baseline, where we improve the above C-$\tau$ and leverage the neural network to model personalized $\tau$.
- Adap-$\tau_0$: Leveraging an automatic $\tau_0$ to avoid grid search.
- Adap-$\tau$: Enhancing Adap-$\tau_0$ with personalized temperatures $\tau_u$.

Meanwhile, we compare the model with various types of SOTA models, range from SGL[33], SimpleX[21], SimSGL[36], NCL[19].

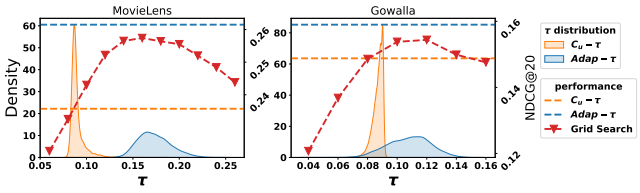


**Figure 4: The dashed red curves denote the performance of the Grid Search $\tau$ with the different $\tau$. The dashed orange and blue lines indicate the performance of Adap-$\tau$ and Cu-$\tau$. We also report the distribution of the learned personalized $\tau$ for Adap-$\tau$ and Cu-$\tau$ as marked by the orange and blue regions.**

*5.1.3* ***Parameter Settings.*** For a fair comparison, the embedding size is fixed to 64 and the initialization is unified with Xavier [7]. A grid search is conducted to confirm the optimal parameter setting for each model. Detailed implementation refers to A.

## 5.2 Performance Comparsion (RQ1)

In this subsection, we conduct multiple experiments to validate the effectiveness of our strategy.

*5.2.1* ***Effectiveness of our strategy.*** As can be seen from Tab. 3, with few exceptions, Adap-$\tau_0$ and Adap-$\tau$ that do not utilize any hyperparameter tuning on $\tau$, consistently outperforms the grid search baseline in all datasets and backbones. This result is highly encouraging, suggesting the limitation of embedding normalization can be obviated.

*5.2.2* ***Impact of Adaptive Temperature.*** From Table 2, we observe that, Adap-$\tau$ obtains a superior performance against Cu-$\tau$ and C-$\tau$. We attribute this phenomenon to that C-$\tau$ is highly sensitive to initialized value and lack of benchmarked $\tau$ to decide proper distribution of $\tau$ (cf. Fig. 4). Cu-$\tau$ and C-$\tau$ lack of critical supervisory signal to control the problem of gradient vanishment. Hence, it still exhibits inferior performance than our model.

*5.2.3* ***Impact of Fine-grained Temperature.*** By comparing with Adap-$\tau$ and Adap-$\tau_0$, we observe that Adap-$\tau$ consistently outperform Adap-$\tau_0$. This result indicates the superiority of leveraging personalized $\tau$ in RS. Once various users exhibit distinct rules or individual property, it is intuitively hard to control them by a fixed $\tau$. Similar results can be observed by Cu-$\tau$ outperforming C-$\tau$.

*5.2.4* ***Fair Recommendation.*** As we mentioned in the introduction, we argue that the model without normalization will aggravate popularity bias. In this subsection, We give more detailed analyses from a fairness perspective. To prove that, we follow [33], splitting items into ten groups *w.r.t.* their interaction frequency. adding normalization with Grid Search $\tau$ or adopting our Adap-$\tau$ can relieve the tensely contradictory relationship between long-tail and task of the normal recommendation. As we observe in Fig 3, the bar of "Grid Search $\tau$" and "Adap-$\tau$" surpass the counterpart by a considerable margin in smaller GoupID. It could also verify that our model has the capability of popularity debias from the side.

## 5.3 Adaptiveness Exploration (RQ2)

Through previous experiments, we have realized our Adap-$\tau$ could adapt to the different datasets, backbone models, and users. In this section, we take a step further and explore how Adap-$\tau$ adapt to the data and users with different ratio of noise. Adding noises to the data would significantly change the data distribution, gradient magnitude, as well as the level of user hardness. We believe exploring the performance of Adap-$\tau$ on such a challenging task would help us further understand the adaptivity of Adap-$\tau$.

In this section, we exploit the Adaptiveness of our model over different "noisy data". Two strategies are adopted to add noise to the datasets. 1) In terms of the interaction frequency per user, we added false-positive items at the same proportion. 2) Splitting the

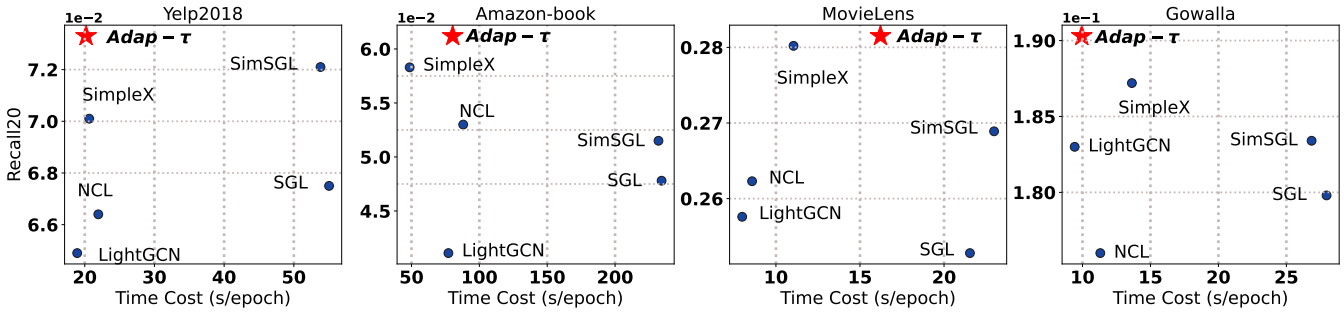


**Figure 5: Performance comparisons in terms of both recommendation accuracy and efficiency.**

**Table 4: Results of MF under different ratio "noisy data"**

| ratio | model | Yelp2018 | | Amazon-book | |
|---|---|---|---|---|---|
| | | Recall | NDCG | Recall | NDCG |
| 0.1 | Grid Search | 0.0722 | 0.0601 | 0.0564 | 0.0455 |
| | Adap-$\tau$ | **0.0735** | **0.0613** | **0.0577** | **0.0467** |
| 0.2 | Grid Search | 0.0703 | 0.0584 | 0.0534 | 0.0432 |
| | Adap-$\tau$ | **0.0717** | **0.0593** | **0.0546** | **0.0443** |
| 0.3 | Grid Search | 0.0696 | 0.0577 | 0.0509 | 0.0409 |
| | Adap-$\tau$ | **0.0702** | **0.0584** | **0.0520** | **0.0422** |
| 0.4 | Grid Search | 0.0678 | 0.0563 | 0.0493 | 0.0400 |
| | Adap-$\tau$ | **0.0685** | **0.0569** | **0.0507** | **0.0412** |
| 0.5 | Grid Search | 0.0667 | 0.0554 | 0.0481 | 0.0388 |
| | Adap-$\tau$ | **0.0672** | **0.0560** | **0.0487** | **0.0394** |

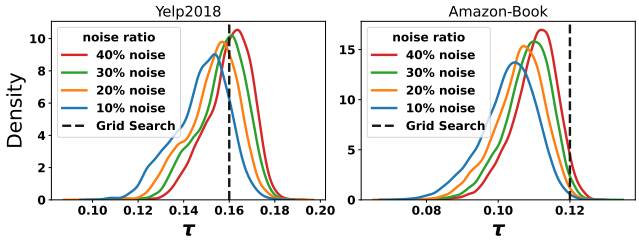


**Figure 6: The distribution of $\tau$ with different ratio of noise data. Here noisy data is added via personalized manner, which different users are affected by different noise ratio.**

users into four groups randomly, and we add fake items at a specific proportion according to the group ID. Strategy 1 concentrates on the **global adaptiveness** confronted with the same ratio of noisy data, while strategy 2 focuses on the **local adaptiveness** with respect to the "noisy ratio" individually.

*5.3.1* ***Global Adaptiveness.*** When we focus on the result over strategy one, we observe that in Table 4, the more noisy data added into training dataset, the larger $\tau$ will be chosen by grid searching. Meanwhile, our Adap-$\tau$ utilizes the feedback of each user and adaptively adjusts the $\tau$ to balance the hard-mining, which verifies its robustness against noise data and flexibility. Experiments also show its superior performance.

*5.3.2* ***Local Adaptiveness.*** As we randomly split items into four groups and add 10%,20%,30%,40% ratio of noise data, the distribution of $\tau$ in each group shows the respective order in Fig 6. Furthermore, with the rising amount of noise data, the adaptive $\tau$ also goes steadily up. And regardless of the ratio of noise data and the difference in adding strategy, our model achieves competitive results against hyperparameter searching without any tuning.

## 5.4 Comparison with SOTA (RQ3)

In this subsection, we are curious about how our model compares to those state-of-the-art models. Here, we choose two representative and powerful baseline: SGL[33], SimpleX[21], SimSGL[36], NCL[19], where SimpleX claims it surpass over 11 benchmark datasets and compared with 29 existing CF models in total.

From Figure 5, we can see our proposed model Adap-$\tau$ obtains competitive results compared with state-of-the-art consistently. Meanwhile, we compare the time cost of each model, verifying that our method including the calculation of $\tau_0$ brings low time cost.

# 6 RELATED WORK

## 6.1 Recommendation System

The basic task of recommendation system is to predict potential interaction, which is called collaborative filtering. Existing methods could be roughly divided into three categories: MF-based methods [15, 29], VAE-based methods [18, 20, 27] and GNN-based methods [10, 32, 35]. GNN-based methods are inclined to achieve state-of-the-art performance with the development of Graph Neural Networks. For example, PinSage [35] borrows the idea from GraphSage while NGCF [32] devised NGCF. As the particularity of CF, LightGCN [10] throws away heavy and burdensome operations to show critical factors in the aggregation mechanism. However, the inner product represents the traditional measurement in the above models, while limited research tends to analyze its impact combine CF task.

## 6.2 Temperature and Normalization

Temperature has exhibited its capability in numerous fields such as CV and NLP in particular with contrastive learning [3, 8]. Motivated by the success in other areas, recommendation combined with contrastive learning has received scant attention in recent research literature [21, 28, 33, 34, 38]. Although normalizaiton and $\tau$ are heuristically used by a small amount of work, it still lacks comprehensive exploration in recommendation. [21] is mostly related work which analyzes the existing loss function and proposes a cosine contrastive loss to achieve hard-mining. However, its loss function filters easy items and needs detailed hyper parameter (margin and weight) searching, which restricted its flexibility in application

and different datasets. To our best of knowledge, [34] is the first work which utilize softmax loss directly into recommendation task. Despite the success of softmax has verified in recommendation system, how to understand it deeply and comprehensively with RS still remains challenge. As [34] can not combine MF and softmax loss perfectly, here, our work focus on the softmax loss along with basic backbone from the pespective of theory and experiments.

## 7 CONCLUSION AND FUTURE WORK

In this work, we focus on the embedding magnitude in recommendation system. With theoretical and empirical analysis, we emphasize the importance of embedding normalization. And we point out of the issue of straightforward normalization. Hence, we propose two principles to guide the adaptive learning of $\tau$. On these basis, we develop an adaptive and personalized $\tau$ without repeated searching over $\tau$ among different datasets. Experiments verify that our simple method is effective with different backbone in numerous dataset.

Embedding magnitude is overlooked in many areas. We believe this study could draw researchers' attention on this issue and inspire more work on this line. It would be interesting to explore fine-grained temperature in other fields.

## ACKNOWLEDGMENTS

This work is supported by the National Key Research and Development Program of China (2021YFF0901603), the National Natural Science Foundation of China (62102382, U19A2079, 62121002), the CCCD Key Lab of Ministry of Culture and Tourism and the Starry Night Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-001).

## REFERENCES

[1] Henry W Block and Zhaoben Fang. 1988. A multivariate extension of Hoeffding's lemma. The Annals of Probability (1988), 1803–1820.
[2] Thibault Castells, Philippe Weinzaepfel, and Jerome Revaud. 2020. SuperLoss: A Generic Loss for Robust Curriculum Learning. In Advances in Neural Information Processing Systems, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 4308–4319. https://proceedings.neurips.cc/paper/2020/file/2cfa8f9e50e0f510ede9d12338a5f564-Paper.pdf
[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In International conference on machine learning. PMLR, 1597–1607.
[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems. 191–198.
[5] Magdalini Eirinaki, Malamati D Louta, and Iraklis Varlamis. 2013. A trust-aware system for personalized user recommendations in social networks. IEEE transactions on systems, man, and cybernetics: systems 44, 4 (2013), 409–421.
[6] Matthias Feurer and Frank Hutter. 2019. Hyperparameter optimization. In Automated machine learning. Springer, Cham, 3–33.
[7] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 249–256.
[8] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 9729–9738.
[9] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. 355–364.
[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 639–648.
[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web. 173–182.
[12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web. 173–182.
[13] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In Proceedings of the eighth international workshop on data mining for online advertising. 1–9.
[14] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. 549–558.
[15] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE international conference on data mining. Ieee, 263–272.
[16] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2553–2561.
[17] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 426–434.
[18] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference. 689–698.
[19] Zihan Lin, Changxin Tian, Yupeng Hou, and Wayne Xin Zhao. 2022. Improving Graph Collaborative Filtering with Neighborhood-enriched Contrastive Learning. In Proceedings of the ACM Web Conference 2022. 2320–2329.
[20] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning disentangled representations for recommendation. Advances in neural information processing systems 32 (2019).
[21] Kelong Mao, Jieming Zhu, Jinpeng Wang, Quanyu Dai, Zhenhua Dong, Xi Xiao, and Xiuqiang He. 2021. SimpleX: A Simple and Strong Baseline for Collaborative Filtering. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 1243–1252.
[22] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018).
[23] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In 2008 Eighth IEEE International Conference on Data Mining. IEEE, 502–511.
[24] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2110–2119.
[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012).
[26] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 255–262.
[27] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In The World Wide Web Conference. 3251–3257.
[28] Hao Tang, Guoshuai Zhao, Yuxia Wu, and Xueming Qian. 2021. Multisample-based Contrastive Loss for Top-k Recommendation. IEEE Transactions on Multimedia (2021).
[29] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. arXiv e-prints (2018), arXiv–1807.
[30] Feng Wang and Huaping Liu. 2021. Understanding the behaviour of contrastive loss. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2495–2504.
[31] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. 2017. Normface: L2 hypersphere embedding for face verification. In Proceedings of the 25th ACM international conference on Multimedia. 1041–1049.
[32] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval. 165–174.
[33] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 726–735.
[34] Jiancan Wu, Xiang Wang, Xingyu Gao, Jiawei Chen, Hongcheng Fu, Tianyu Qiu, and Xiangnan He. 2022. On the Effectiveness of Sampled Softmax Loss for Item Recommendation. arXiv preprint arXiv:2201.02327 (2022).

[35] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 974–983.
[36] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 1294–1303.
[37] Wenhui Yu and Zheng Qin. 2020. Graph convolutional network for recommendation with low-pass collaborative filters. In International Conference on Machine Learning. PMLR, 10936–10945.
[38] Chang Zhou, Jianxin Ma, Jianwei Zhang, Jingren Zhou, and Hongxia Yang. 2021. Contrastive learning for debiased candidate generation in large-scale recommender systems. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 3985–3995.
[39] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 1059–1068.

# A EXPERIMENTAL SETTINGS

We implement our model in PyTorch and will release our implementation (codes, parameter setting, and training log) to enhance reproducibility. For a fair comparison, the embedding size is fixed to 64 for all methods and the initialization is unified with Xavier [7]. A grid search is conducted to confirm the optimal parameter setting for each model. To be more specific, learning rate is tuned among 1e-3, 5e-3, 1e-4 and the coefficient of $L_2$ regularization term is searched in $\{1e^{-9}, 1e^{-8}, ..., 1e^{-1}\}$. As for the backbone of LightGCN, the number of layers is tuned among 1,2,3, where dropout is adopted or not to prevent over-fitting. Focus on the traditional contrastive loss, temperature $\tau$ is a fine-grained search with an interval of 0.02. And the number of negative sampling is varying in 200, 400, 800, 1500. Note that we perform SimpleX baseline in a detailed grid search accumulated more than 3500 experiments per dataset, where margin searched among $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and weight tuning among $\{50, 100, 150, 200, 400, 800\}$.

# B PROOFS

## B.1 Proof of the lemma 1

Proof. Let $\mathcal{L}(u)$ be the softmax loss on user $u$, we have

$$\mathcal{L}(u) = -\sum_{i\in\mathcal{P}_u} \log \frac{\exp(f(u,i))}{\sum_{j\in\mathcal{I}} \exp(f(u,j))} \tag{17}$$

Owing to items $i$ are combined with two parts: $y_{ui} = 0$ and $y_{ui} = 1$. According to the [34], we are told that the gradient of $\mathcal{L}(u)$ *w.r.t.* $f(u,i)$ is composed of two parts:

$$\nabla_1 = -1 + \frac{1 + mp_n(i)}{\exp(f(u,i)) + m\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,j))} \exp(f(u,i)) \tag{18}$$

$$\nabla_2 = \sum_{k\in\mathcal{P}(u)\setminus\{i\}} \frac{1 + mp_n(i)}{\exp(f(u,k)) + m\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,j))} \exp(f(u,i)) \tag{19}$$

Then, we use the fact that $\exp(f(u,k)) \ll N\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,j)), k \in \mathcal{P}_u$ when $m \to \infty$, the sum of Eq (18, 19) develop into:

$$\begin{aligned}
\nabla_{all} &= -1 + \sum_{k\in\mathcal{P}(u)} \frac{mp_n(i)\exp(f(u,i))}{m\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,j))} + \frac{\exp(f((u,i))}{m\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,j))} \\
&= -1 + \frac{mp_n(i)|\mathcal{P}_u| + 1}{m\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,i))} \exp(f(u,j)) \\
&= -1 + \frac{p_n(i)|\mathcal{P}_u| + \frac{1}{m}}{\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,i) - f(u,j))}
\end{aligned} \tag{20}$$

where $|\mathcal{P}_u|$ represents the frequency of user $u$, $p_n(i)$ denotes the probability of item sampling which default to $1/m$. In terms of the normal gradient descent, we have

$$\begin{aligned}
\delta_i &= ||e_i + \eta \frac{\partial L(u)}{\partial i}||^2 - ||e_i||^2 \\
&= ||e_i + \eta \frac{\partial L(u)}{\partial f(u,i)} \frac{\partial f(u,i)}{\partial i}||^2 - ||e_i||^2 \\
&= 2\eta \ \frac{p_n(i)|\mathcal{P}_u| + \frac{1}{m}}{\mathsf{E}_{j\in\mathcal{I}} \exp(f(u,j) - f(u,i))} - 1 \ f(u,i) + o(\eta^2 \cdot \frac{\partial L(u)}{\partial i}) \\
&\approx 2\eta \ \frac{|\mathcal{P}_u| + 1}{m \cdot \mathsf{E}_{j\in\mathcal{I}} \exp(f(u,j) - f(u,i))} - 1 \ f(u,i)
\end{aligned} \tag{21}$$

Omitting the term $o(\eta^2 \cdot \frac{\partial L(u)}{\partial i})$ and considering gradient descent *w.r.t* item $i$ towards various users, the change of item $i$'s magnitude over Eq (21) will evolve into:

$$\delta_i = \sum_{u} 2\eta \ \frac{|\mathcal{P}_u| + 1}{m \cdot \mathsf{E}_{j\in p_n} \exp(f(u,j) - f(u,i))} - 1 \ f(u,i) \tag{22}$$

We can draw an observation from Lamma 1: Note that at the early stage of the training procedure, users and items are distributed uniformly. In other words, $\exp(f(u,j) - f(u,i))$ and $\mathsf{E}_u f(u,i)$ cannot tell remarkable difference, while the magnitude of popular items will obtain explosive rising in term of $|\mathcal{P}_i|$. That is:

$$\delta_i \propto |\mathcal{P}_i| \tag{23}$$

Combine with above observation, we can prove the LEMMA 1.

## B.2 Proof of Lemma 2

Proof. Note that the gradient $\frac{\partial L}{\partial f(u,i)}$ can be written as:

$$\frac{\partial L}{\partial f(u,i)} = \begin{cases} \frac{1}{\tau} p_{ui}(\tau)(1 - \sum_{k\in N_u} p_{uk}(\tau)), & \text{for} \quad y_{ui} = 1 \\ -\frac{1}{\tau} p_{ui}(\tau)(\sum_{k\in N_u} p_{uk}(\tau)), & \text{for} \quad y_{ui} = 0 \end{cases} \tag{24}$$

Thus, the expression (9) can be transformed into:

$$\begin{aligned}
\mathsf{E}_{u,i}[|\frac{\partial L}{\partial f(u,i)}|] &= \frac{1}{m\tau}\mathsf{E}_u[2\sum_{i\in N_u} p_{ui}(\tau)(1 - \sum_{k\in N_u} p_{uk}(\tau))] \\
&\le \frac{2}{m\tau}(\mathsf{E}_u[\sum_{i\in N_u} p_{ui}(\tau)] - \mathsf{E}_u^2[\sum_{i\in N_u} p_{ui}(\tau)]) \\
&\le \frac{2}{mt}(\mathsf{E}_u[\sum_{i\in N_u} p_{ui}(\tau)] - \mathsf{E}_u^2[\sum_{i\in N_u} p_{ui}(\tau)])
\end{aligned} \tag{25}$$

where Cauchy Inequality is employed. The upper bound has a quadratic form. The optimal condition can be easily obtained by transforming the expression into $-(\mathsf{E}_u[\sum_{i\in N_u} p_{ui}(\tau)]-\frac{1}{2})^2+1$.

## B.3 Proof of the lemma 3

Proof. For convenient, for each user $u$, let $a_u$ be the sum of rescaled prediction over his positive instances $\sum_{i\in N_u} exp(f(u,i)/\tau)$, and $b_u$ be the sum over all instances $\sum_i exp(f(u,i)/\tau)$. The expression of equation (11) can be transformed into:

$$\begin{aligned}\mathsf{E}_u[\sum_{i\in N_u} p_{ui}(\tau)] &= \frac{\mathsf{E}_u[a_u]}{\mathsf{E}_u[b_u]} + \frac{\mathsf{E}_u[\frac{a_u}{b_u}]\mathsf{E}_u p[b_u] - E[a_u]}{\mathsf{E}_u[b_u]} \\ &= \frac{\mathsf{E}_u[a_u]}{\mathsf{E}_u[b_u]} - \frac{Cov(\frac{a_u}{b_u}, b_u)}{E_u[b_u]} \\ &\approx \frac{\mathsf{E}_u[a_u]}{\mathsf{E}_u[b_u]}\end{aligned} \tag{26}$$

where $Cov(\frac{a_u}{b_u}, b_u)$ denotes the covariance between the variables, which is bounded by the variance of $b_u$, i.e., $Cov(\frac{a_u}{b_u}, b_u) \le \mathrm{Var}_u[b_u]$. Considering in practice the value of $\mathrm{Var}_u[b_u]$ is usually quite small, here we simply drop out the covariance term for derivation.

Now we turn to deal with the expression of $\mathsf{E}_u[a_u]$ and $\mathsf{E}_u[b_u]$. Based on Taylor's expansion of an exponential function, we have:

$$\begin{aligned}\mathsf{E}_u[b_u] &= m\mathsf{E}_f[\exp(\frac{\mathbf{f}}{\tau})] \\ &= m\exp(\mathsf{E}_f[\frac{\mathbf{f}}{\tau}])(1 + \frac{\mathsf{E}_f[(\mathbf{f}-\mu)^2]}{\tau^2 2!} + \sum_{k=3} \frac{\mathsf{E}_f[(\mathbf{f}-\mu)^k]}{\tau^k k!})\end{aligned} \tag{27}$$

when $\tau > 2\lambda$, it can be approximated as:

$$E_u[b_u] \approx m\exp(\frac{\mu}{\tau})(1 + \frac{E_f[(\mathbf{f}-\mu)^2]}{\tau^2 2!}) \tag{28}$$

since the higher-order term is bounded with:

$$|\sum_{k=3} \frac{E_f[(\mathbf{f}-\mu)^k]}{\tau^k k!}| \le \sum_{k=3} \frac{2(\lambda/2)^k k!}{\tau^k k!} = \frac{2(\frac{\lambda}{2\tau})^3}{1-\frac{\lambda}{2\tau}} \le \frac{1}{24} \tag{29}$$

where we use the fact that the central moment of a sub-exponential variable is bounded:

$$\begin{aligned}E_f[|(\mathbf{f}-\mu)|^k] &= \int_0^\infty p(|(\mathbf{f}-\mu)|^k > t)dt \\ &= \int_0^\infty p\ \ |(\mathbf{f}-\mu)|^k > t^{1/k}\ \ \mathrm{d}t \\ &\le \int_0^\infty 2e^{-\frac{2t^{1/k}}{\lambda}}\mathrm{d}t \\ &= 2(\lambda/2)^k k \int_0^\infty e^{-u}u^{k-1}\ \mathrm{d}u = 2(\lambda/2)^k k!\end{aligned} \tag{30}$$

Equation (28) can be further transformed into:

$$E_u[b_u] \approx m\exp(\frac{\mu}{\tau})\exp(\frac{E_f[(\mathbf{f}-\mu)^2]}{2\tau^2}) \tag{31}$$

as $\exp(\frac{E_f[(\mathbf{f}-\mu)^2]}{2\tau^2}) = 1+\frac{\sigma^2}{2\tau^2}+o((\frac{\sigma^2}{2\tau^2})^2)$ and $\frac{\sigma^2}{2\tau^2} \le \frac{\lambda^2}{2\tau^2} \le \frac{1}{8}$. Similar treatment can be conducted for $E_u[a_u]$ and we can get:

$$E_u[a_u] \approx \frac{|D|}{n}\exp(\frac{\mu_+}{\tau})\exp(\frac{\sigma_+^2}{2\tau^2}) \tag{32}$$

Thus, the original conditional equation can be transformed into:

$$\frac{E_u[a_u]}{E_u[b_u]} \approx \frac{\frac{|D|}{n}\exp(\frac{\mu_+}{\tau})\exp(\frac{\sigma_+^2}{2\tau^2})}{m\exp(\frac{\mu}{\tau})\exp(\frac{\sigma^2}{2\tau^2})} = \frac{1}{2} \tag{33}$$

With the reorganization, we can find the equation has a quadratic form *w.r.t.* $1/\tau$ and thus can write the root of the equation as:

$$\tau \approx \frac{\sigma_+^2 - \sigma^2}{-(\mu^+ - \mu) + \sqrt{(\mu^+ - \mu)^2 + 2(\sigma_+^2 - \sigma^2)\log(\frac{nm}{2|D|})}} \tag{34}$$

When $\sigma_+^2$ is quite close to $\sigma^2$, $\tau_0$ can be approximated with:

$$\tau_0 \approx \frac{\mu_+ - \mu}{\log(\frac{nm}{2|\{D||\}})} \tag{35}$$

The lemma gets proofed.

# C VALIDATION ON APPROXIMATION

## C.1 Approximation w.r.t. $\tau_0$

In our implementation, we pursue efficiency so as to simplify the expression of $\tau_0$. To verify its justifiability, we record the real value of $\sigma_+^2 - \sigma^2$ and corresponding $\tau_0$. Actually, we could obtain similar performance under Eq. 13 and Eq. 14.

**Table 5: Comparison between approximation on $\tau_0$.**

| Datasets | $\sigma_+^2 - \sigma^2$ | $\tau_0$ by Eq. 13 | $\tau_0$ by Eq. 14 |
|---|---|---|---|
| Yelp2018 | -0.004362 | 0.095602 | 0.099263 |
| Amazon-Book | -0.001356 | 0.078836 | 0.079994 |
| MovieLens | -0.017910 | 0.148159 | 0.156905 |
| Gowalla | -0.001148 | 0.094281 | 0.095187 |

## C.2 Gradient Vanishment

To verify the impact of $\tau$ on the gradient vanishment, we adopt different temperatures to train the model and record the respective gradient according to Eq. (8). Figure 7 verifies its effectiveness on gradient. In particular, when $\tau$ is larger than 0.3, the overall gradient is nearly zero. And the peak of Figure 7 is equivalent to the best $\tau$ via grid search.

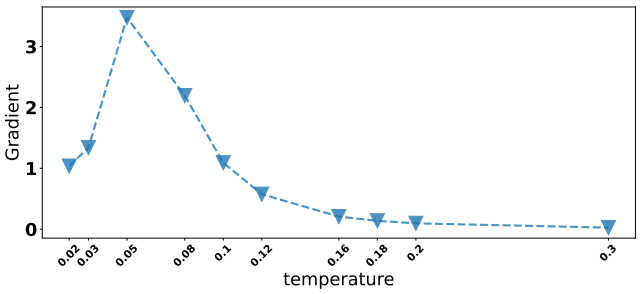


**Figure 7: The gradient w.r.t. $\tau$ on Gowalla.**

## D PSEUDO-CODE FOR ADAP-$\tau$

**Algorithm 1** Adap-$\tau$'s main learning algorithm.

**input:** Number of users and items:n,m; Users and Items in training set $U', I' \in \mathcal{D}$; Batch size: B; Number of neagative sampling: M;Dimension: d.
**for** epoch $\in \{1, \ldots, N\}$ **do**
  # 1. Compute $\tau_0$
  Get user and item embeddings $e(U'), e(I')$
  $e(U'), e(I') = normalize(e(U')), normalize(e(I'))$
  # Dimension $e(U')$: [n, d], $e(I')$: [m, d]
  $\mu_+ = mean(\boldsymbol{e}(U')^\top \boldsymbol{e}(I'))$ ▷ $O(|\mathcal{D}|d)$
  $\mu = mean(e(U')^\top mean(e(I', dim = 1)))$ ▷ $O(nd)$
  $\tau_0 = \frac{\mu_+ - \mu}{\log(\frac{nm}{2|\{D||\}})}$ ▷ $O(1)$
  Calculate acumulataed loss of each user $\hat{\mathcal{L}}(u)$
  # 2. Start Training.
  **for** sampled minibatch $\{\boldsymbol{u}, \boldsymbol{i}, \boldsymbol{j}\}$ in DataLoader **do**
    Get user and item embeddins $e(u), e(i), e(j)$
    $s(pos) = \boldsymbol{e}_u^\top \boldsymbol{e}_i / (\|\boldsymbol{e}_u\| \|\boldsymbol{e}_i\|)$ ▷ postive score $O(Bd)$
    $s(neg) = \boldsymbol{e}_u^\top \boldsymbol{e}_j / (\|\boldsymbol{e}_u\| \|\boldsymbol{e}_j\|)$ ▷ negative score $O(BdM)$
    Calculate $\tau_u$ according to $\hat{\mathcal{L}}(u), \tau_0$ as Eq. (16)
    **define** $\mathcal{L}(u)$ **as**
$$\mathcal{L}(u) = -\log \frac{\exp(s_{u,i}(pos)/\tau_u)}{\exp(s_{u,i}(pos)/\tau_u) + \sum_{j=1}^{M} \exp(s_{u,j}(neg)/\tau_u)}$$
    $\mathcal{L} = \frac{1}{B} \sum_{u=1}^{B} [\mathcal{L}(u)]$
    update networks $f$ to minimize $\mathcal{L}$
  **end for**
**end for**
**return** encoder network $f(\cdot)$.

Junkang Wu*, Jiancan Wu, Sheng Zhou, Xuezhi Cao, Xiangnan He„