Personalized Multi-task Training for Recommender System

Liangwei Yang[1*], Zhiwei Liu[2], Jianguo Zhang[2], Rithesh Murthy[2], Shelby Heinecke[2],
Huan Wang[2], Caiming Xiong[2] and Philip S. Yu[1]

[1]*Department of Computer Science, University of Illinois Chicago*, USA

{lyang84, psyu}@uic.edu

[2]*Salesforce AI Research*, USA

{zhiweiliu, jianguozhang, rithesh.murthy, shelby.heinecke, huan.wang, cxiong}@salesforce.com

Abstract—In the vast landscape of internet information, recommender systems (RecSys) have become essential for guiding users through a sea of choices aligned with their preferences. These systems have applications in diverse domains. Personalization is a key technique in RecSys, where modern methods leverage representation learning to encode user/item interactions into embeddings, forming the foundation for personalized recommendations. However, integrating information from multiple sources to enhance recommendation performance remains challenging. This paper introduces a novel approach named **PMTRec**, the first personalized multi-task learning algorithm to obtain comprehensive user/item embeddings from various information sources. Addressing challenges specific to personalized RecSys, we develop modules to handle personalized task weights, diverse task orientations, and variations in gradient magnitudes across tasks. **PMTRec** dynamically adjusts task weights based on gradient norms for each user/item, employs a Task Focusing module to align gradient combinations with the main recommendation task, and uses a Gradient Magnitude Balancing module to ensure balanced training across tasks. Through extensive experiments on three real-world datasets with different scales, we demonstrate that **PMTRec** significantly outperforms existing multi-task learning methods, showcasing its effectiveness in achieving enhanced recommendation accuracy by leveraging multiple tasks simultaneously. Our contributions open new avenues for advancing personalized multi-task training in RecSys.

Index Terms—Recommender System, Multi-task Training

I. INTRODUCTION

Regarding the pervasive landscape of information available on the Internet [1], recommender systems (RecSys) [2] play a pivotal role in facilitating the exploration of items for users. RecSys improves the efficiency of information digestion and has permeated into every corner of our lives, such as news feeds [3], game suggestions [4], and shopping recommendations [5]. The great commercial value of RecSys also establishes its indispensability in web services and drives the success of many Internet companies [6]–[8].

Personalization is the key technique for an effective RecSys [9]. Personalized RecSys suggests specific candidate items for each user based on his/her own historical interactions. It offers each user a distinct experience with the web service tailored to their interests. Due to the success of representation learning [10], current state-of-the-art methods [11], [12] in

RecSys encode user/item interactions into dense vector representations, referred to as embeddings. The embedding of each user/item contains his/her/its unique information, which is the bedrock for providing personalized recommendations. Based on the encoded embedding, these methods recommend items with the highest similarities for each user. Besides user-item interactions, more information can be found in web services and applications such as users' social networks [13] and item features [14]. Then, a natural question arises: Can we obtain a more comprehensive embedding from multiple sources of information to improve the recommendation performance?

In this paper, we seek to answer this question with multi-task learning [15], which has been shown effective in various application domains such as computer vision [16] and natural language processing [17]. The core idea behind it is to obtain a comprehensive user/item embedding that can perform multiple tasks at the same time. As illustrated in Figure 1, we show three tasks associated with distinct information. The black edges denote the recommendation task to predict users' preference for items, which is the main task for optimization. The blue edge indicates social information, where the social link prediction task between users characterizes social information into embedding of users. The red line represents two items in the same "Electronics" category. Predicting whether two items share the same category encodes item category similarity into item embedding. Training user/item embedding with all these tasks endows incorporating information from multiple sources, leading to a more comprehensive vector representation.

Several key challenges are still under-explored when developing a multi-task learning framework for personalized RecSys, which are threefold as follows: (1) Personalized task
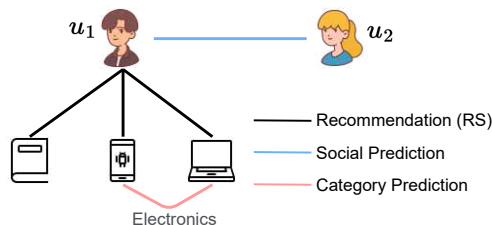

Fig. 1. A toy example of multi-task learning in RecSys.

*The work is finished during an internship at Salesforce AI Research.

weights. For each user, the importance of different tasks should be distinct. As illustrated in Figure 1, the suitable task weights for u_1 and u_2 are different. u_1 has sufficient item interactions to obtain an accurate gradient for updating embedding, which suggests a higher RecSys task weight for u_1. However, u_2 does not interact with any item before, in which case the RecSys task fails to provide informative information, and it is beneficial for u_2 to have a higher weight on the Social Prediction task. Supporting specific task weights for each user is the first challenge, and it is essential for personalized training with multiple tasks. (2) Varied task orientations. The purpose of different tasks is varied, leading to diverged embedding updates during training, which may conflict with each other or even be meaningless for improving the main RecSys task. Handling the varied updating orientations from multiple tasks to improve the RecSys task poses the second challenge. (3) Difference in gradient magnitude across tasks. The loss functions for multiple tasks are inconsistent, and thus the gradient magnitude could be magnificently different. Directly averaging all those gradients from multiple tasks may overlook tasks with smaller gradients. However, balancing the gradient magnitude difference and making the small gradient counts are an important yet under-explored challenge.

This paper proposes the first Personalized Multi-task Training algorithm for Recommender System (PMTRec) to obtain a more comprehensive user/item embedding from multiple information sources. Different modules are designed accordingly to deal with specific challenges. We dive into the gradient level during the backward propagation to enable personalized task weights. PMTRec first collects the gradient from each task during backpropagation and then combines the gradient on each user embedding separately. PMTRec conducts meticulous user/item-specific gradient level operations rather than the existing task-wise algorithms [16], [18]–[20]. For a specific gradient of one user from one task, PMTRec firstly calculates the norm to measure the importance. A gradient with a higher magnitude signifies a greater impact on optimizing the corresponding task. PMTRec then utilizes the gradient norm to calculate task weight for gradient combination. The gradient norm on each task for different users is distinct, leading to personalized gradient weights. We also designed a *Task Focusing* module to tackle the varied task orientation challenge. We enforce the learning process gradually, focusing on the main RecSys task during the training stage, achieved by increasing the corresponding gradient combination weight. After initial exploration of other tasks, the Task Focusing module leads to the combined gradient gradually aligning well with the RecSys task, leaving the varied task orientations imperceptible at the final stage. Different tasks generate gradients of different magnitudes. Multi-task training algorithms overlook gradients with smaller magnitudes without considering the difference. Thus, we further propose the Gradient Magnitude Balancing module to deal with this challenge and balance the training of all tasks. Code of PMTRec is also published [1]. Our

contributions can be summarised as follows:

- Conceptually, for personalized multi-task learning, we first illustrate the problem, identify the challenges, and propose feasible solutions for recommender system.
- Methodologically, we propose the novel PMTRec algorithm that operates on the gradient level during backpropagation to provide personalized gradient weights.
- Experimentally, we conduct extensive experiments on three real-world datasets with varied scales to justify the effectiveness of our proposed PMTRec algorithm.

II. PRELIMINARIES

This section illustrates the preliminaries for PMTRec, including a problem statement and multi-task learning.

A. Problem Statement

Given a set of users $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and a set of items $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$, the personalized RecSys task aims to generate a list of items for a given user u. The most important information for training a recommendation model is the currently observed historical interactions, which can be represented as a user-item bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ and there is an edge $(u, i) \in \mathcal{E}$ between u and i if u has interacted with i historically. The adjacency matrix of $\mathcal{G}$ is represented by $\mathbf{R} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$. For implicit feedback [21] such as clicks, views, and purchases, $R_{u,i} = 1$ if user u has interacted with item i, and $R_{u,i} = 0$ otherwise. For explicit feedback, the user provides, such as rating score, $R_{u,i}$ is the explicit rating score from u to i. Besides the direct historical interactions, auxiliary information can also be observed from both the user and item side, which is termed by profile $\mathcal{P} = (\mathcal{P}^u, \mathcal{P}^i)$. $\mathcal{P}^u$ contains all the auxiliary information from the user side, such as features and social friends. $\mathcal{P}^i$ represents the item side auxiliary information such as category and co-view information [22]. Based on all the above information, personalized RecSys aims to predict the preference score in the adjacency matrix of $\mathcal{G}$ and recommends a list of k items with the top-k highest score.

B. Multi-task Learning

Multi-task learning (MTL) aims to learn a comprehensive model to accomplish multiple tasks by training them simultaneously. This learning paradigm assumes different tasks can enhance each other and perform better on all tasks. Without loss of generality, we integrate and illustrate the current multi-task learning algorithms [16], [18]–[20] from the gradient level. Each task is trained and can be categorized by a loss function $\mathcal{L}$, which directly leads to a gradient $\nabla \mathcal{L}$ after backpropagation [23]. Assuming we have T tasks, then a list of the gradient for different tasks is obtained $(\nabla \mathcal{L}_1, \nabla \mathcal{L}_2, ..., \nabla \mathcal{L}_T)$. An MTL algorithm can be seen as a pooling function over the gradient list to obtain the combined gradient for descent $\nabla = \text{Pool}_{\text{MTL}}(\nabla \mathcal{L}_1, \nabla \mathcal{L}_2, ..., \nabla \mathcal{L}_T)$. Different MTL algorithms differ in the design of the pooling function, which can be parameterized or heuristic.

III. METHOD

PMTRec is shown in Figure 2, which includes a forward propagation to gather gradients from multiple tasks and a backward stage to aggregate gradients from different tasks. Our contribution lies in the backward stage, which contains the task focusing and gradient magnitude balancing modules.

A. Forward Propagation

In the forward propagation stage, PMTRec computes the losses for different tasks. It consists of an encoder to obtain representation and several task modules for loss computation.

1) Encoder: Current state-of-the-art personalized Rec-Sys [11], [12] learns an encoder to embed historical interactions $\mathcal{G}$ into dense embedding representations as:

$$\mathbf{E} = \text{Encoder}(\mathcal{G}), \tag{1}$$

where $\mathbf{E} = (\mathbf{E}^u, \mathbf{E}^i)$. $\mathbf{E}^u \in \mathbb{R}^{|\mathcal{U}| \times d}$ and $\mathbf{E}^i \in \mathbb{R}^{|\mathcal{I}| \times d}$ represent the user/item embedding separately. $\mathbf{e}_u \in \mathbf{E}^u$ is user u's embedding, which encodes the specific information related to u's historical interaction. $\mathbf{e}_u$ is the bedrock to provide personalized recommendation for u, and d is the dimension size. In PMTRec, we use matrix factorization [21] as the encoder to directly embed interactions into dense user/item vectors. PMTRec aims to encode more information into those vector representation with auxiliary tasks.

2) Multi-task Loss computation: For RecSys, tasks $\mathcal{T}$ are classified into the recommendation task t_{RecSys} and auxiliary tasks $\mathcal{T}_{Aux} = \{t_1, t_2, ..., t_{|\mathcal{T}_{Aux}|}\}$. Auxiliary tasks can be built in different ways based on the available information. As illustrated in Figure 2, Task 1 is built purely from item embedding $\mathbf{E}^i$ and item profile $\mathcal{P}^i$. It aims to embed the item's profile information to the corresponding item embedding. Task 2 is built in a similar way from the user side. Tasks should be designed based on the available information for each dataset. For example, we demonstrate the task design in detail for the three experimental datasets in Section IV. Feeding embedding table $\mathbf{E}$ encoded from Section III-A1 into the loss computation module of different tasks, we can obtain the losses:

$$(\mathcal{L}_{RecSys}, \mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_{|\mathcal{L}_{Aux}|}) \tag{2}$$

B. Backward Propagation

During the backward propagation stage, PMTRec combines the gradients in a personalized fashion, i.e., computes specific gradient weights for each user. It consists of gradient collection, task focusing and gradient magnitude balancing modules.

1) Gradient Collection: For each user/item, each loss $\mathcal{L}$ in Equation 2 leads to one gradient on the corresponding embedding vector, denoted as $\bigtriangledown_{\mathbf{e}}\mathcal{L}$. Then a collection of gradients from different tasks is represented by:

$$(\bigtriangledown_{\mathbf{e}}\mathcal{L}_{RecSys}, \bigtriangledown_{\mathbf{e}}\mathcal{L}_1, \bigtriangledown_{\mathbf{e}}\mathcal{L}_2, ..., \bigtriangledown_{\mathbf{e}}\mathcal{L}_{|\mathcal{L}_{Aux}|}), \tag{3}$$

where $\mathbf{e}$ is the embedding for one user/item. Then PMTRec measures the impactfulness of the gradient by l_2 norm as $n_t = |\bigtriangledown_{\mathbf{e}}\mathcal{L}_t|_2$. A larger n_t indicates the current user/item is more impactful to optimize task t. Then a list of norms is computed from Equation 3 as $(n_{RecSys}^{Init}, n_1, n_2, ..., n_{|\mathcal{L}_{Aux}|})$.

Algorithm 1 The training process of PMTRec

Require: Training set $\mathcal{G}$, user/item embedding table $\mathbf{E}$, main Rec-Sys task $\mathcal{L}_{RecSys}$, auxiliary tasks $\mathcal{L}_{Aux} = \{\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_{|\mathcal{L}_{Aux}|}\}$, a hyper-parameter τ to control the gradient magnitude, total training epoch S, learning rate lr, task-focusing base α.
1: **for** $s = 1, 2, 3, ...,$ to S **do**
2:    **for** Each mini-batch of user-item interactions $\mathcal{B}$ **do**
3:       $Grad = [\bigtriangledown_{\mathbf{E}}\mathcal{L}_{RecSys}(\mathcal{B})]$
       Gradient Collection
4:       **for** $\mathcal{L} \in \mathcal{L}_{Aux}$ **do**
5:          $Grad_{\mathcal{L}} = \bigtriangledown_{\mathbf{E}}\mathcal{L}(\mathcal{B})$
6:          $Grad$ append $Grad_{\mathcal{L}}$
7:       **end for**
8:       $Grads = \text{Stack}(Grad, \text{dim} = 0)$
9:       $Norm = \text{Norm}(Grads, \text{dim} = -1)$
       Task Focusing
10:      $Norm[0] = Norm[0] * \alpha^s$
       Gradient Magnitude Balancing
11:      $Attn_{weight} = \text{Softmax}(Norm/\tau, \text{dim} = 0)$
12:      $\bigtriangledown_{\mathbf{E}} = \text{Sum}(Attn_{weight}.\text{unsqueeze}(-1) * Grads, \text{dim} = 0).$
13:      $\mathbf{E} = \mathbf{E} - lr * \bigtriangledown_{\mathbf{E}}.$
14:    **end for**
15: **end for**

---

2) Task Focusing: PMTRec treats the RecSys task as the main task and all the other tasks as auxiliary ones for further improvement. Auxiliary tasks are built differently, leading to varied task orientations. Thus we design the Task Focusing module to focus the training on the main RecSys task gradually. To this end, a hyper-parameter $\alpha > 1$ is introduced as the increasing base. Before transforming the gradient norm from Section III-B1 into gradient weights, we multiply the weight on the main RecSys task as:

$$n_{RecSys} = n_{RecSys}^{Init} * \alpha^s, \tag{4}$$

where s is the current epoch number. In this design, PMTRec will explore all tasks first and gradually focus on the main RecSys task. A larger α will fasten the focusing speed. Though in exponential design, the following Softmax computation with built-in log-sum-exp trick ensures the number stability.

3) Gradient Magnitude Balancing: Loss functions within Equation 2 are different due to the varied task designs, making the gradient magnitude essentially different. The gradient with a small magnitude will be overlooked if we directly transform the gradient norm into gradient weight. Thus, we propose the gradient magnitude balancing module to balance the impact of different gradient magnitudes by a Softmax with temperature:

$$w_t = \frac{\exp^{n_t/\tau}}{\sum_{t^* \in \mathcal{T}} \exp^{n_{t^*}/\tau}}, \tag{5}$$

where τ is the hyper-parameter to balance gradient magnitude. All task weights will be the same without being impacted by different gradient magnitudes when $\tau \to +\infty$. Conversely, PMTRec will only count the gradient with the largest norm when $\tau \to -\infty$. A suitable τ balance should be selected for the best gradient magnitude. With Equation 6, we can obtain the gradient weights as $(w_{RecSys}, w_1, w_2, w_{|\mathcal{L}_{Aux}|})$. Then
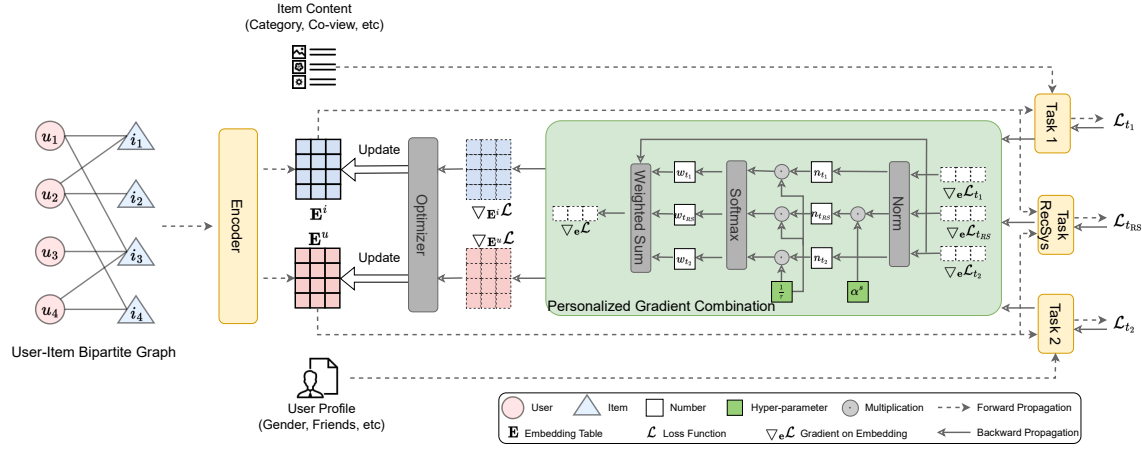
Fig. 2. Model Framework. Our proposed PMTRec is illustrated in detail with two propagation stages. (1) Forward propagation computes the user/item embedding and the losses for different tasks. (2) Backward propagation pools the gradient obtained from each task with a personalized gradient combination for each user/item. The combined gradient is then used to update the embedding with an optimizer. (Best viewed in color)

PMTRec obtains the gradient on the corresponding user/item by weighted sum on the gradients:

$$\bigtriangledown_{\mathbf{e}}\mathcal{L} = \sum_{t\in\mathcal{T}} w_t \bigtriangledown_{\mathbf{e}} \mathcal{L}_t, \tag{6}$$

where $\bigtriangledown_{\mathbf{e}}$ is the combined gradient for a specific use/item and is used to update corresponding embedding $\mathbf{e}$. The variance in gradient norms observed across distinct tasks for different users/items culminates in the derivation of personalized gradient combination weights. It positions PMTRec as a pioneering instance of personalized multi-task training algorithms.

C. Discussion

RecSys naturally faces the data sparsity issue [24], [25] as the candidate item pool is too large for each user to explore. Information from external knowledge such as social networks [13] and knowledge graph [26]–[28] has proven to provide extra information to enrich user/item representation. However, previous methods model external knowledge as part of the encoder, such as adding auxiliary edges besides the user-item bipartite graph, which needs sophisticated design within the encoder. PMTRec proposes another solution, i.e., embed the external knowledge into user/item representation with multi-task training. It can keep the model intact and embed that information with more tasks.

We further analyze the overhead time complexity caused by multi-task training. For each user/item, PMTRec firstly computes the norm of an embedding gradient in Section III-B1, which takes $(O(d))$. The Task Focusing module in Section III-B2 will only cost $O(1)$ as it only computes on the RecSys task. Then the Gradient Magnitude Balancing module in Section III-B3 takes $O(|\mathcal{T}|)$. To sum up, the overall time complexity of PMTRec is $O((|\mathcal{U}| + |\mathcal{I}|) \times (d + |\mathcal{T}|))$, where $\mathcal{U}, \mathcal{I}, \mathcal{T}$ is the user/item/task set respectively and d is the embedding dimension. It is noted that the $(|\mathcal{U}|+|\mathcal{I}|)$ dimension can be easily batched up for parallel computation, which makes the wall clock time even much faster. The batched

version of PMTRec is shown in Algorithm 1 with all the modules marked correspondingly, and $\bigtriangledown_{\mathbf{E}}$ is the aggregated gradient on embedding $\mathbf{E}$.

IV. EXPERIMENT

This section empirically evaluates the proposed PMTRec on three real-world datasets. The goal is to answer the five following research questions (RQs).

- **RQ1:** Is PMTRec effective compared with other multi-task learning algorithms?
- **RQ2:** Does each designed task help in improving the recommendation performance?
- **RQ3:** Does the designed task focusing and gradient magnitude balancing module plays a role in PMTRec?
- **RQ4:** What is the impact of hyper-parameter α and τ?
- **RQ5:** How does the training curve of the newly proposed multi-task training algorithm look like?

A. Experimental Setup

1) Datasets: We compare PMTRec against baselines on three real-world datasets with different sizes. The detailed data statistics of each dataset are shown in Table I. Epinion [2] collects user's reviews on items with user's social friends on e-commerce website. Based on the available information, the Social Prediction task (predict whether two users are social friends) and the Category Prediction task (predict whether two items belong to the same category) are constructed as auxiliary tasks. Video Game and Office [14] are both from the Amazon platform. For the Video Game dataset, we build a co-view and co-buy prediction task to predict whether two items are co-viewed/co-bought together frequently. As for the Office dataset, we construct the Alignment/Uniformity task [12] to directly regularize on the embedding table as well as a rating prediction task to predict the user's explicit rating score towards items. Following previous researches [11], [14], a 5-core setting is

[2]https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.htm

TABLE I
STATISTICS OF THE DATASETS.

Dataset	Epinion	Video Game	Office
#Users	22,167	55,223	4,905
#Items	296,278	17,408	2,420
#Interactions	922,267	497,577	51,441
Density	0.014%	0.051%	0.433%
Task 1	Social Prediction	Co-View Prediction	Alignment Uniformity
Task 2	Category Prediction	Co-Buy Prediction	Rating Regression

applied to all datasets, i.e., we remove users/items with less than 5 interactions to keep the dataset quality.

2) Baselines: We compare PMTRec with 7 baselines in multi-task learning to test the effectiveness of PMTRec.

- EW: It is the most basic method by directly adding all losses together to optimize user/item representation.
- GradDrop [29]: It detects the gradient conflict on each parameter and drops the conflict signal with a scale-measured probability. A smaller scale is easier to be dropped.
- PCGrad [16]: Project Conflicting Gradients (PCGrad) changes one of the optimization directions of conflict gradients to make the gradients conflict-free.
- GradVac [18]: Gradient Vaccine (GradVac) is the updated version of PCGrad to further align the unconflict gradients.
- CAGrad [30]: Conflict-averse gradient descent (CAGrad) minimizes the average loss function by leveraging the worst local improvement of individual tasks.
- Aligned MTL [31]: It proposes to use a condition number of a linear system of gradients as a stability criterion to guide the multi-task learning optimization.
- Nash MTL [32]: Nash MTL suggests conceptualizing the process of combining gradients as akin to a bargaining game, wherein individual tasks engage in negotiation to mutually agree upon a collective direction for parameter updates.
- RLW [20]: Random Loss Weight (RLW) assigns loss weight randomly in each epoch to explore the different tasks.

3) Evaluation Method: We randomly split the dataset into a training set (60%), validation set (20%), and test set (20%). Models are trained on the training set, hyper-parameters are tuned based on the performance on the validation set, and the reported results are from the test set. We test the effectiveness of PMTRec on Top-K personalized recommendation task with three widely adopted evaluation metrics: Recall@K (R@K), Hit Ratio@K (HR@K), and Normalized Discounted Cumulative Gain@K (N@K). To comprehensively assess our model, we present evaluation results for Top-20 and Top-40 recommendations in our experiments. Our evaluation approach calculates average metrics across all users within the test set. These metrics are computed based on the rankings of items that users have not interacted with. To align with established research practices [11], [12], we adopt the complete full ranking for evaluation. This method entails ranking all items that a user has not yet engaged with, thereby providing a comprehensive evaluation.

4) Experiment Settings: To make a fair comparison, we test PMTRec and all the other baselines under the same setting. User/Item embedding size d is fixed to 32 with Xavier [33] initialization. To keep simplicity, we directly use the embedding table as the Encoder. It is to be noted that PMTRec also supports other encoders. We fix the batch size as 2048 and utilize Adam optimizer [34] to optimize the parameters. For the remaining hyper-parameters, we used the grid search method to find the optimal settings for PMTRec: the learning rate is searched in $\{0.1, 0.05, 0.01, 0.005, 0.001\}$, coefficient of weight decay is tuned in $\{1e^{-2}, 1e^{-4}, 1e^{-6}, 1e^{-8}\}$. We search the task focusing base α from 1.0 to 1.2 with a step size of 0.02, and the temperature τ in gradient magnitude balancing module in $\{10, 1, 1e^{-1}, 1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}, 1e^{-6}\}$. We stopped the training early without improvement in successive 10 epochs on the main recommendation task for all the experiments and reported the results on the test set.

B. RQ1: Performance Comparison

We compare PMTRec on 3 real-world datasets with 8 baselines in multi-task learning. The experiment results are shown in Table II. We can have the following observations:

- PMTRec consistently outperforms the second-best method on all three datasets. On the Epinion dataset, PMTRec even surpasses the second best with 9.8% improvement in N@5. It justifies the effectiveness of PMTRec compared with all previous multi-task training algorithms for RecSys.
- The improvement of PMTRec for different datasets varies. PMTRec achieves remarkable improvement on Epinion and Office datasets while only achieving about 2% improvement on the Video Game dataset. We assume it is because the designed tasks do not provide much extra information. The co-view, co-buy item prediction task can be easily inferred from user-item interactions.
- RLW nearly always ranks the second best across all three datasets. RLW randomly assigns task weights in each epoch during training. It is a quite simple method to add more randomness for gradient exploration. It surpasses the sophisticated designed gradient manipulation methods such as PCGrad and GradVac by a large margin. The previous gradient manipulation methods directly consider all the parameter sets without personalized consideration, which hinders their performance.

To show the compatibility of PMTRec with the current widely used RecSys, we further conduct experiments on different encoders, including MF [21], NGCF [35] and LightGCN [11]. Experiment results are shown in Figure 3. We can observe that PMTRec always achieves the best performance regardless of the encoder. It shows that PMTRec is compatible with current RecSys algorithms. Besides, we can also observe the performance improvement with a more advanced encoder. It shows the improvement brought by multi-task training is orthogonal to the encoder, and we can integrate it within current RecSys algorithms to achieve better performance.

TABLE II
OVERALL COMPARISON. THE BEST IS BOLDED, AND THE RUNNER-UP IS UNDERLINED.

| Dataset	Metric	EW	GradDrop	PCGrad	GradVac	CAGrad	Aligned MTL	Nash MTL	RLW	PMTRec	Imp
Epinion	R@5	0.0130	0.0131	0.0142	0.0142	0.0142	0.0123	0.0156	0.0151	**0.0170**	8.97%
	R@10	0.0219	0.0224	0.0233	0.0233	0.0240	0.0203	0.0251	0.0238	**0.0266**	5.97%
	R@20	0.0356	0.0355	0.0364	0.0364	0.0379	0.0314	0.0408	0.0381	**0.0413**	1.22%
	R@40	0.0562	0.0570	0.0573	0.0573	0.0596	0.0494	0.0633	0.0598	**0.0639**	0.94%
	N@5	0.0128	0.0130	0.0137	0.0137	0.0139	0.0121	0.0153	0.0146	**0.0168**	9.80%
	N@10	0.0161	0.0164	0.0170	0.0170	0.0175	0.0150	0.0188	0.0178	**0.0202**	7.44%
	N@20	0.0205	0.0207	0.0213	0.0213	0.0221	0.0186	0.0239	0.0225	**0.0249**	4.18%
	N@40	0.0263	0.0267	0.0272	0.0272	0.0282	0.0236	0.0303	0.0286	**0.0312**	2.97%
Video Game	R@5	0.0435	0.0435	0.0408	0.0511	0.0397	0.0321	0.0454	0.0461	**0.0467**	1.30%
	R@10	0.0716	0.0716	0.0654	0.0657	0.0643	0.0528	0.0732	0.0730	**0.0750**	2.45%
	R@20	0.1109	0.0974	0.1044	0.1046	0.1006	0.0843	0.1128	0.1129	**0.1154**	2.21%
	R@40	0.1677	0.1480	0.1591	0.1599	0.1533	0.1296	0.1682	0.1668	**0.1713**	1.84%
	N@5	0.0347	0.0347	0.0326	0.0329	0.0313	0.0256	0.0360	0.0365	**0.0373**	2.19%
	N@10	0.0451	0.0451	0.0416	0.0419	0.0404	0.0333	0.0463	0.0464	**0.0478**	3.01%
	N@20	0.0571	0.0497	0.0535	0.0538	0.0515	0.0429	0.0584	0.0585	**0.0601**	2.73%
	N@40	0.0712	0.0624	0.0670	0.0676	0.0646	0.0541	0.0722	0.0719	**0.0740**	2.92%
Office	R@5	0.0243	0.0281	0.0213	0.0213	0.0325	0.0144	0.0224	0.0336	**0.0347**	3.27%
	R@10	0.0401	0.0458	0.0350	0.0349	0.0513	0.0248	0.0403	0.0553	**0.0594**	7.41%
	R@20	0.0655	0.0752	0.0576	0.0576	0.0819	0.0417	0.0685	0.0849	**0.0905**	6.59%
	R@40	0.1074	0.1139	0.0893	0.0891	0.1247	0.0676	0.1100	0.1321	**0.1474**	11.58%
	N@5	0.0192	0.0229	0.0165	0.0165	0.0260	0.0124	0.0198	0.0259	**0.0281**	8.49%
	N@10	0.0253	0.0295	0.0217	0.0217	0.0332	0.0163	0.0267	0.0344	**0.0372**	8.13%
	N@20	0.0334	0.0387	0.0289	0.0289	0.0428	0.0217	0.0362	0.0439	**0.0473**	7.74%
	N@40	0.0443	0.0489	0.0372	0.0372	0.0539	0.0287	0.0474	0.0562	**0.0619**	10.14%
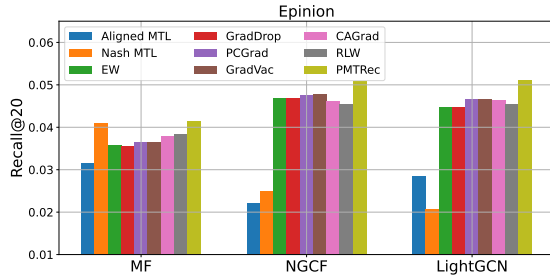
Fig. 3. Experiments on different encoders

C. RQ2: Task Investigation

This section aims to investigate whether each designed task helps improve the recommendation performance. Towards this end, we experiment with individual tasks, and Table III shows the influence of tasks on recommendation performance across different datasets. To have a thorough investigation on the effectiveness of multi-task learning in RecSys, we conduct these experiments with different encoders, including MF [21], NGCF [36] and LightGCN [11]. We can have the following observations in this experiments:

- We observe the improvements across all three datasets comparing pure BPR performance and the one combining BPR, T1, and T2. On the Epinion dataset, PMTRec improves the BPR performance from 0.0117 to 0.0170 on R@5 with two extra auxiliary tasks, a 45.2% improvement. The huge improvement indicates the necessity of combining knowledge from extra tasks to alleviate the data sparsity problem in RecSys. The experiment results also validate PMTRec can effectively fuse the knowledge from auxiliary tasks to improve the RecSys performance.

- There is a general trend of performance improvements when additional tasks are integrated. For example, the pure RecSys task on the Epinion dataset achieves an N@5 of 0.0118. Adding T1 increases to 0.0158, and adding T2 increases to 0.0166. Jointly combining the two tasks achieves the best performance of 0.0168.
- The impact of individual tasks varies among datasets. On Epinion and Video Game datasets, RecSys performance gradually improves with more tasks. However, the addition of T2 on the Office dataset results in a performance drop, which indicates the auxiliary tasks should be designed carefully to align with the main RecSys task.

D. RQ3: Ablation Study

Ablation study experiment on task focusing and gradient magnitude balancing modules is shown in Figure 4. We can have the following observations:

- We observe performance deterioration across all three datasets upon omitting any module. It shows each module contributes to the effectiveness of PMTRec.
- Excluding the task-focusing module leads to a slight decline in the Epinion and Video Game datasets but substantially decreases the Office dataset. Notably, Recall@40 and NDCG@40 drop by 5.7% and 4.7% on Office, respectively. It reveals that gradually focusing the training on the main RecSys task can improve the main RecSys task, which is overlooked by all previous multi-task training algorithms.
- The omission of the gradient magnitude balancing module has a marked impact on the Epinion and Video Game datasets. Specifically, the Recall@40 and NDCG@40 metrics are decreased by 4.4% and 7.6% on Epinion, and by 14.7% and 15.1% on Video Game, highlighting the module's critical role in model performance.

TABLE III

| Model	Tasks	Epinion				Video Game				Office			
		R@5	R@20	N@5	N@20	R@5	R@20	N@5	N@20	R@5	R@20	N@5	N@20
MF	BPR	0.0117	0.0349	0.0118	0.0198	0.0427	0.1085	0.0336	0.0553	0.0281	0.0729	0.0243	0.0402
	+T1	0.0158	0.0395	0.0158	0.0238	**0.0469**	0.1151	0.0373	0.0600	0.0372	0.0860	0.0304	0.0470
	+T2	0.0167	**0.0414**	0.0166	0.0249	0.0465	**0.1161**	0.0368	0.0596	0.0232	0.0684	0.0190	0.0345
	+T1,T2	**0.0170**	0.0413	**0.0168**	**0.0250**	0.0467	0.1154	**0.0374**	**0.0601**	0.0347	**0.0905**	0.0281	**0.0473**
NGCF	BPR	0.0118	0.0294	0.0121	0.0180	0.0433	0.1105	0.0344	0.0565	0.0232	0.0689	0.0197	0.0355
	+T1	0.0181	0.0461	0.0185	0.0278	0.0463	0.1139	0.0369	0.0593	0.0281	0.0812	0.0201	0.0382
	+T2	0.0184	0.0481	0.0187	0.0287	0.0450	0.1117	0.0359	0.0579	0.0202	0.0688	0.0145	0.0308
	+T1,T2	**0.0195**	**0.0507**	**0.0199**	**0.0302**	0.0463	0.1138	0.0370	0.0594	**0.0354**	0.0823	**0.0295**	**0.0454**
LightGCN	BPR	0.0122	0.0320	0.0129	0.0195	0.0418	0.1041	0.0331	0.0538	0.0260	0.0723	0.0216	0.0380
	+T1	0.0173	0.0459	0.0178	0.0273	0.0442	0.1133	0.0355	0.0585	0.0338	0.0846	0.0258	0.0450
	+T2	**0.0201**	0.0499	0.0196	0.0296	0.0445	0.1138	0.0356	0.0585	0.0221	0.0713	0.0165	0.0332
	+T1,T2	0.0198	**0.0511**	**0.0197**	**0.0301**	**0.0448**	**0.1144**	0.0355	**0.0586**	0.0338	**0.0931**	**0.0280**	**0.0466**
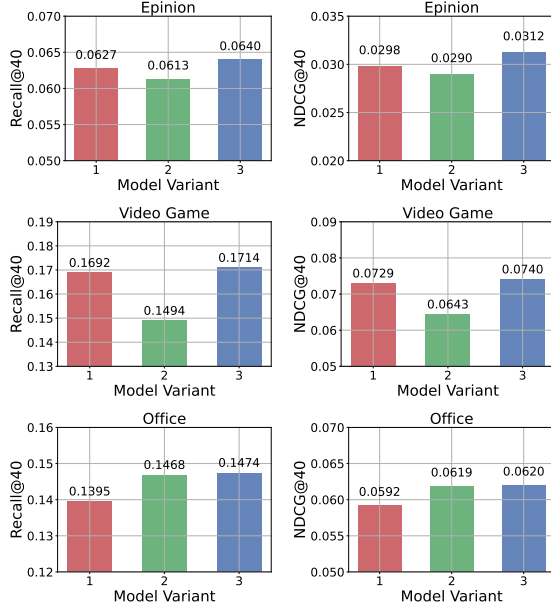
Fig. 4. Ablation study of PMTRec. Variant 1 omits the task focusing module, Variant 2 excludes the gradient magnitude balancing module, and Variant 3 represents the full PMTRec.
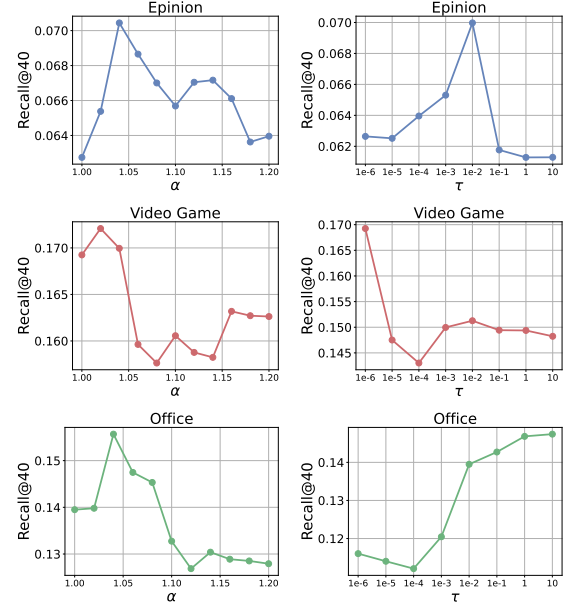

Fig. 5. Sensitivity analysis of PMTRec's hyper-parameters: α is the increment base in the task focusing module, while τ balances the gradient magnitude in the balancing module.

E. RQ4: Hyper-Parameter Sensitivity

This section aims to study the Hyper-parameter sensitivity of PMTRec. It introduces two hyper-parameters in different modules. α is the multiplying base in the task-focusing module. With a larger α, the training of PMTRec is faster focused on the main RecSys task. τ is the temperature in the gradient magnitude balancing module. All task weights will be the same without being impacted by different gradient magnitudes when $\tau \rightarrow +\infty$. Conversely, PMTRec will only count the gradient with the most prominent norm when $\tau \rightarrow -\infty$. We show the experiment results on all three datasets in Figure 5. From the results, we can observe that:

- With the increase of α, the performance of PMTRec firstly increases to the peak point and then quickly decreases. At the initial stage, increasing α will focus the training of

PMTRec on the RecSys task at the final stage, showing the advantages of task focusing. When α is large, PMTRec focuses the training on RecSys quickly without enough exploration of auxiliary tasks. It reveals the importance of learning procedures on auxiliary tasks.
- As τ increases, distinct curve patterns emerge across various datasets. For the Epinion dataset, performance initially reaches a peak before declining. Conversely, optimal performance in the Video Game dataset is achieved with the smallest τ, while the Office dataset attains its highest performance with the largest τ. Auxiliary tasks are constructed in different ways for different datasets. It shows uniqueness of datasets and task construction.
- In most cases, PMTRec does not perform best when $\alpha = 1$ or $\tau = 1$. When the hyper-parameter is equal to 1, the
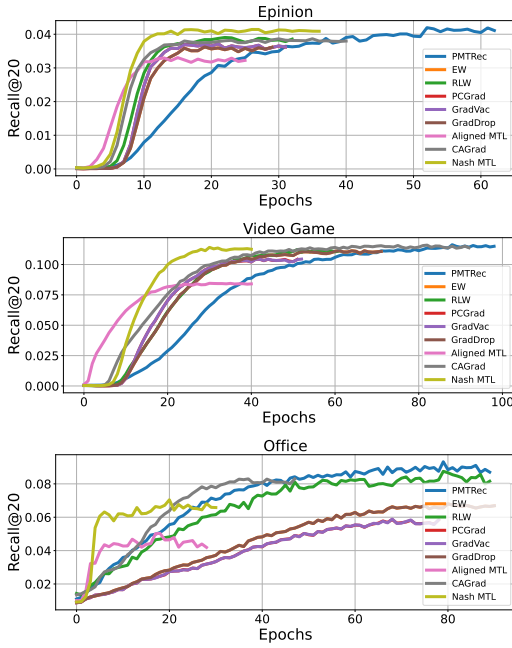
Fig. 6. The training curve on three datasets.

corresponding module does not play a role, leading to inferior performance. It re-validates the effectiveness of each designed module in PMTRec.

F. RQ5: Training Curve Comparison

As a new multi-task training algorithm, we further show the training curve of PMTRec in Figure 6. All the baselines are also included. We have the following observations:

- On the Epinion and Video Game dataset, other baselines improve quickly at the first few epochs and reach stable on inferior performance. The improvement of PMTRec is more stable and achieves the highest performance at the final stage. At the initial stage, PMTRec tends to explore auxiliary tasks first without fast improvement on the RecSys task. With more epochs, the task-focusing module gradually plays a role in enforcing model train toward the RecSys task.
- On Office dataset, PMTRec also improves stability and achieves the highest performance compared to other multi-task training methods. The training curve on the Office dataset justifies the effectiveness of PMTRec.
- Another interesting finding is PMTRec seems more stable across datasets. For all three datasets, PMTRec improves at similar paces. They all improve fastest on 25 epochs and reach the best performance on about 60 epochs. However, other baselines do not seem to have a similar property. All the baselines on the Epinion dataset improve fast at the initial stage, while no similar trend is observed on the Office dataset. We assume personalized updates in PMTRec lead to this observation as it captures the personalization character of the RecSys task.

V. RELATED WORK

A. Multi-task Learning

Multi-Task Learning (MTL) [37]–[39] is an approach that focuses on collectively training multiple interconnected tasks. The goal is to enhance their generalization capabilities by capitalizing on shared insights across these tasks. There has been a fast development of MTL in recent years. Current methods can be classified into two categories: conflict resolution-based methods and randomness-based methods. Conflict resolution-based methods aim to identify and solve the gradient conflict/imbalance from different tasks. MGDA-UB [40] utilizes multiple gradient descent algorithms to find a common descending direction among all gradients to avoid the gradient conflict by solving a quadratic programming problem. GradNorm [19] aims to solve the gradient imbalance problem by constraining the gradient magnitudes to be similar. PCGrad [16] modifies the optimization direction of conflicting gradients, redirecting them to achieve gradient conflict resolution and promote coherence among the gradients. GradVac [18] represents an enhanced iteration of PCGrad, designed to align non-conflicting gradients in similar directions more effectively. CAGrad [30] employs the concept of worst local improvement from individual tasks, and it minimizes the average loss function, thereby guiding the algorithm's trajectory through regularization. Aligned MTL [31] suggests utilizing a conditioned number of linear gradient systems as a stability criterion, guiding optimizing multi-task learning to avoid conflict. TAWT [41] proposes target aware cross task weighted training algorithm. IMTL-G [42] strives to identify an aggregated gradient characterized by uniform-length projections onto the gradients of each individual task. RotoGrad [43] addresses conflict issues by aligning gradient magnitudes and directions concurrently. It achieves this by introducing a trainable rotation matrix to modify the gradient direction for each task, followed by the computation of weights that enforce uniform gradient magnitudes across all tasks. On the other hand, randomness-based methods aim to add more gradient exploration based on multiple tasks. UW [44] employs consistent uncertainties as loss weights for individual tasks, which are adaptively adjusted through backpropagation. GradDrop [29] posits that conflict arises from discrepancies in gradient signs across various tasks. To mitigate this conflict, GradDrop drops gradient values based on magnitude-measured probability. RLW [20] introduces variability by assigning loss weights randomly in each epoch, facilitating the exploration of task optimization from diverse perspectives. Nash MTL [32] attains top-notch performance across various Multi-Task Learning (MTL) benchmarks. The approach advocates for interpreting the gradient combination process as a form of bargaining game, where individual tasks negotiate to reach a consensus on a shared direction for parameter updates.

Previous methods do not consider personalized gradient combination, which stands PMTRec out from previous methods. It is the first multi-task training algorithm that supports assigning different task weights for each user/item.

B. Representation Learning for RecSys

The core of existing recommender systems is learning user/item representations. BPR-MF [21] is one of the most popular methods to characterize user-item interactions and train user/item representations. FM [45] further incorporates real-value attributes of users and items to learn representations. Additionally, if given a user-item rating matrix, NMF [46] is widely adopted to learn user/item embeddings. As deep learning technology develops, NCF [35] is proposed to encode user/item IDs with neural networks as embeddings. Recently, the successes of graph neural network GNN [47]–[49] prompt the graph-based collaborative filtering, such as NGCF [36] and LightGCN [11]. Those graph-based collaborative filtering methods learn user/item representation via aggregating neighbor information, which characterizes the high-order signals in the user-item graph. Learning user/item representation based on graph embedding also opens more potential to harness additional information. GraphRec [50] and ConsisRec [13] devise additional graph propagation on social graphs, which enables the joint learning of social and recommendation information. Besides social information between users, those multiple relations between items [51]–[53], such as co-view and co-buy, also benefit the representation learning for the recommendation. The success of contrastive learning in representation learning [54] motivates the novel DirectAU [12] loss for the recommendation. DirectAU optimizes the user/item representation by minimizing uniformity and alignment losses, eliminating the inefficient negative sampling process in learning representation. GraphAU [24] further considers the graph-related high-order alignment signal during optimizing user/item representation. With the emerging ability of large language models (LLM), most recent researches also investigate improving user/item representation with pre-trained large language models for the RecSys task. P5 [55] and OpenP5 [56] train different tasks within large language models by next-token-prediction task. LlmRec [57] augments user-item bipartite graph with LLMs to learn a more informative user/item representation. RLMRec [58] proposes contrastive alignment and generative alignment to align user/item embedding to the large language model encoded space. CARec [59] directly aligns the user/item's ID embedding with semantic embedding in the iterative alignment phase. Previous research shows that a more discriminative, comprehensive, and informative user/item representation is the foundation for effective RecSys.

This paper proposes to directly learn user/item representation under a multi-task learning paradigm and proposes the first personalized multi-task training method PMTRec. It can encode information from multiple tasks to obtain a more comprehensive and informative user/item representation.

VI. CONCLUSION

This paper introduces PMTRec, a novel multi-task training algorithm specifically designed for recommender systems. This algorithm distinctively supports personalized gradient combinations for individual user/item. It intricately navigates the backpropagation process, manipulating gradients directly at their level. Within PMTRec, two principal modules are constructed: a task-focusing and a gradient magnitude balancing module. The former progressively concentrates training efforts on the primary recommender system (RecSys) task, whereas the latter equilibrates gradients of varying magnitudes. Empirical evaluations conducted on real-world datasets substantiate PMTRec's efficacy. As the first multi-task training algorithm designed explicitly for the recommender system, we also open-sourced PMTRec for further research.

REFERENCES

[1] V. Mayer-Schönberger and K. Cukier, *Big data: A revolution that will transform how we live, work, and think.* Houghton Mifflin Harcourt, 2013.

[2] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou, "Recommender systems," *Physics reports*, vol. 519, no. 1, pp. 1–49, 2012.

[3] C. Wu, F. Wu, T. Qi, Q. Liu, X. Tian, J. Li, W. He, Y. Huang, and X. Xie, "Feedrec: News feed recommendation with various user feedbacks," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 2088–2097.

[4] L. Yang, Z. Liu, Y. Wang, C. Wang, Z. Fan, and P. S. Yu, "Large-scale personalized video game recommendation via social-aware contextualized graph neural network," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 3376–3386.

[5] Y. Gu, Z. Ding, S. Wang, and D. Yin, "Hierarchical user profiling for e-commerce recommender systems," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 223–231.

[6] C. A. Gomez-Uribe and N. Hunt, "The netflix recommender system: Algorithms, business value, and innovation," *ACM Transactions on Management Information Systems (TMIS)*, vol. 6, no. 4, pp. 1–19, 2015.

[7] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[8] Z. Liu, L. Zou, X. Zou, C. Wang, B. Zhang, D. Tang, B. Zhu, Y. Zhu, P. Wu, K. Wang *et al.*, "Monolith: real time recommendation system with collisionless embedding table," *arXiv preprint arXiv:2209.07663*, 2022.

[9] X. Qian, H. Feng, G. Zhao, and T. Mei, "Personalized recommendation combining user interest and social circle," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 7, pp. 1763–1777, 2013.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "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*, 2020, pp. 639–648.

[12] C. Wang, Y. Yu, W. Ma, M. Zhang, C. Chen, Y. Liu, and S. Ma, "Towards representation alignment and uniformity in collaborative filtering," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 1816–1825.

[13] L. Yang, Z. Liu, Y. Dou, J. Ma, and P. S. Yu, "Consisrec: Enhancing gnn for social recommendation via consistent neighbor aggregation," in *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*, 2021, pp. 2141–2145.

[14] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *proceedings of the 25th international conference on world wide web*, 2016, pp. 507–517.

[15] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, 2018.

[16] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[17] D. Dong, H. Wu, W. He, D. Yu, and H. Wang, "Multi-task learning for multiple language translation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1723–1732.

[18] Z. Wang, Y. Tsvetkov, O. Firat, and Y. Cao, "Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models," *arXiv preprint arXiv:2010.05874*, 2020.

[19] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International conference on machine learning*. PMLR, 2018, pp. 794–803.

[20] B. Lin, F. Ye, Y. Zhang, and I. W. Tsang, "Reasonable effectiveness of random weighting: A litmus test for multi-task learning," *arXiv preprint arXiv:2111.10603*, 2021.

[21] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[22] Z. Fan, Z. Liu, C. Wang, P. Huang, H. Peng, and S. Y. Philip, "Sequential recommendation with auxiliary item relationships via multi-relational transformer," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 525–534.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[24] L. Yang, Z. Liu, C. Wang, M. Yang, X. Liu, J. Ma, and P. S. Yu, "Graph-based alignment and uniformity for recommendation," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 4395–4399.

[25] H. Yin, Q. Wang, K. Zheng, Z. Li, and X. Zhou, "Overcoming data sparsity in group recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 7, pp. 3447–3460, 2020.

[26] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 950–958.

[27] S. Wang, L. Yang, J. Gong, S. Zheng, S. Du, Z. Liu, and S. Y. Philip, "Metakrec: collaborative meta-knowledge enhanced recommender system," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 665–674.

[28] S. Wang, Z. Fan, J. Gong, X. Wei, and S. Y. Philip, "Transgnn: Towards knowledge enhanced top-k recommendation via transformed heterogeneous graph neural network," in *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 2023, pp. 304–314.

[29] Z. Chen, J. Ngiam, Y. Huang, T. Luong, H. Kretzschmar, Y. Chai, and D. Anguelov, "Just pick a sign: Optimizing deep multitask models with gradient sign dropout," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2039–2050, 2020.

[30] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 878–18 890, 2021.

[31] D. Senushkin, N. Patakin, A. Kuznetsov, and A. Konushin, "Independent component alignment for multi-task learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 20 083–20 093.

[32] A. Navon, A. Shamsian, I. Achituve, H. Maron, K. Kawaguchi, G. Chechik, and E. Fetaya, "Multi-task learning as a bargaining game," *arXiv preprint arXiv:2202.01017*, 2022.

[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] X. He and T.-S. Chua, "Neural factorization machines for sparse predictive analytics," in *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2017, pp. 355–364.

[36] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 2019, pp. 165–174.

[37] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, pp. 41–75, 1997.

[38] L. Li, H. Zhu, S. Zhao, G. Ding, and W. Lin, "Personality-assisted multi-task learning for generic and personalized image aesthetics assessment," *IEEE Transactions on Image Processing*, vol. 29, pp. 3898–3910, 2020.

[39] X. Tang, Y. Qiao, F. Lyu, D. Liu, and X. He, "Touch the core: Exploring task dependence among hybrid targets for recommendation," in *Proceedings of the 18th ACM Conference on Recommender Systems*, 2024, pp. 329–339.

[40] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[41] S. Chen, K. Crammer, H. He, D. Roth, and W. J. Su, "Weighted training for cross-task learning," *arXiv preprint arXiv:2105.14095*, 2021.

[42] L. Liu, Y. Li, Z. Kuang, J. Xue, Y. Chen, W. Yang, Q. Liao, and W. Zhang, "Towards impartial multi-task learning." iclr, 2021.

[43] A. Javaloy and I. Valera, "Rotograd: Gradient homogenization in multitask learning," *arXiv preprint arXiv:2103.02631*, 2021.

[44] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[45] S. Rendle, "Factorization machines," in *2010 IEEE International conference on data mining*. IEEE, 2010, pp. 995–1000.

[46] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in neural information processing systems*, vol. 13, 2000.

[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[48] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[49] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[50] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019, pp. 417–426.

[51] D. Xu, C. Ruan, J. Cho, E. Korpeoglu, S. Kumar, and K. Achan, "Knowledge-aware complementary product representation learning," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 681–689.

[52] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Product knowledge graph embedding for e-commerce," in *Proceedings of the 13th international conference on web search and data mining*, 2020, pp. 672–680.

[53] Z. Fan, Z. Liu, S. Heinecke, J. Zhang, H. Wang, C. Xiong, and P. S. Yu, "Zero-shot item-based recommendation via multi-task product knowledge graph pre-training," *arXiv preprint arXiv:2305.07633*, 2023.

[54] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9929–9939.

[55] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5)," in *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022, pp. 299–315.

[56] S. Xu, W. Hua, and Y. Zhang, "Openp5: An open-source platform for developing, training, and evaluating llm-based recommender systems," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 386–394.

[57] W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Llmrec: Large language models with graph augmentation for recommendation," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024, pp. 806–815.

[58] X. Ren, W. Wei, L. Xia, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Representation learning with large language models for recommendation," *arXiv preprint arXiv:2310.15950*, 2023.

[59] C. Wang, L. Yang, Z. Liu, X. Liu, M. Yang, Y. Liang, and S. Y. Philip, "Collaborative semantic alignment in recommendation systems," *arXiv preprint arXiv:2310.09400v3*, 2024.