EXPLORING OPTIMIZATIONS IN MULTI-TASK RECOMMENDATION VIA MANIPULATING AUXILIARY GRADIENTS

An Undergraduate Research Scholars Thesis

by

RONALD LEE

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: Dr. James Caverlee

May 2023

Major: Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Ronald Lee, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Faculty Research Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Exploring Optimizations in Multi-task Recommendation via Manipulating Auxiliary Gradients

Ronald Lee
Department of Computer Science & Engineering
Texas A&M University

Faculty Research Advisor: Dr. James Caverlee
Department of Computer Science & Engineering
Texas A&M University

Auxiliary Task Learning has proven to be an effective way to transfer knowledge between tasks. This is the case in many personalized recommendation scenarios, where many auxiliary tasks can boost the performance of the primary task via a shared layer in a multi-task network. However, we often encounter a gradient imbalance issue when updating the parameters of the neural network. For example, the gradient magnitudes for an auxiliary task could vary significantly compared to the target task which leads to imbalanced information transfer between tasks. This could cause a particular auxiliary task to over or under influence the output of the target task.

Multiple works have been done to resolve the gradient imbalance issues. Some methods include directly manipulating the gradients with respect to the shared parameters in order to balance gradient magnitudes. Other approaches identify conflicting gradient direction between tasks and dynamically scale auxiliary gradients depending on their direction agreement with the target task. This thesis seeks to understand the baseline benchmark techniques that exist and to combine multiple techniques to improve recommendation accuracy. Current methods of direct gradient manipulation are not combined as they have shown sub-optimal performance. In this work, we evaluate and compare these different methods on a large, public recommendation data set. We explore various ways to combine both methods with the aim of achieving boosted performance. In

our analysis, we learn that naively combining gradient balancing methods with gradient direction based methods yield sub-optimal results; however, they outperform the baseline methods which suggests that more refined techniques could improve upon state-of-the-art performance.

ACKNOWLEDGMENTS

Contributors

I would like to thank my faculty advisor, Dr. James Caverlee, and my friends and mentors from the Infolab for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, thanks to my family for their encouragement and to my friends for their patience and support.

The data analyzed/used for this thesis were provided by Alibaba, and can be found at https://tianchi.aliyun.com/dataset/649.

All other work conducted for the thesis was completed by the student independently.

NOMENCLATURE

NDCG	Normalized Discounted Cumulative Gain
TAMU	Texas A&M University
NN	Neural Network
PV	Page view
A2C	Add-to-Cart
MTL	Multi-task Learning
ATL	Auxiliary-task Learning
UB-2017	User Behvior 2017 Dataset

1. INTRODUCTION

Traditionally, personalized recommendations have been determined through methods of collaborative filtering, which rely on past user-item interactions and assumes that these interactions are explainable via a latent model. These models have traditionally ignored using neural networks since simpler methods, such as Matrix Factorization [1], have proven to be effective in personalized recommendations. However, advances in deep learning [2], a technique that is used to extract hidden features from data to make predictions, have accelerated progress in many fields across machine learning, including recommendation. This has led to the integration of Neural Networks (NNs) to encode user and item features and be used in recommendation via approaches like Neural Collaborative Filtering [3].

In these neural collaborative filtering models, the performance accuracy and training time can often be boosted through transfer learning from related auxiliary information. This is due to multiple reasons. First, NNs are supervised models that rely on quality labels in order to make accurate predictions. The acquisition to generate these labels can be expensive, but auxiliary task information extracts additional training labels to aid model performance [4]. Second, learning multiple tasks has been shown to create more robust representations of the data to generalize better on unseen data [5]. This has proven to be true in many sub-domains, including natural language processing [6] and computer vision [7] [8].

In personalized recommendation, Auxiliary Task Learning (ATL) has been used to improve prediction accuracy and data efficiency. For example, e-commerce and online-ad platforms such as Amazon and Facebook aim to boost the performance of purchase prediction, a target task, by transferring knowledge from user auxiliary tasks, such as predicting whether a user will "like" an item or predicting the likelihood of a user clicking on an item.

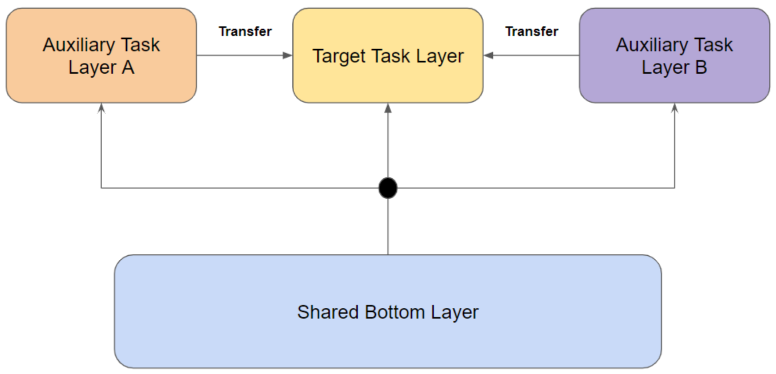

Figure 1: Transfer Learning from 2 auxiliary tasks improving the target task.

1.1 Auxiliary-task Transfer Learning

Transfer learning is often done using a multi-task neural network. Typically, this is composed of 2 parts: a shared bottom layer and multiple, task-specific layers. In the shared bottom layer, inputs are fed into a neural network with shared parameters that return an output embedding of the inputs. Then, this output embedding, which was generated using the shared parameters in the bottom layer, is fed to multiple, task-specific neural networks. This way, information from the auxiliary task layers can be shared with the target task through the shared parameters of the bottom layer, which is illustrated in Figure 1.

1.2 Shortcomings of Auxiliary-Task Transfer Learning

1.2.1 Gradient Magnitude Imbalance Issue

The potential presence of the *imbalance of gradient magnitudes* poses a key challenge when attempting to transfer knowledge from auxiliary tasks to the main task. During the training process, each task has a corresponding loss and each loss has a corresponding gradient with respect to the shared parameters in the bottom layer of our neural network. These shared parameters are updated based on the sum of the gradients which could pose several issues. First, in the case where an auxiliary task has a much larger magnitude than the target task, the auxiliary task will have a much greater impact on how the shared parameters are updated than the target task, effectively *dominating* the impact of the target task loss. Second, converse to the first issue, an auxiliary task with a much smaller magnitude than the target task will have a very weak influence over how the shared parameters are updated, effectively *masking out* the impact of the auxiliary task. Either of these scenarios could lead to sub-optimal model performance at test time.

1.2.2 Conflicts in Gradient Direction with the Target Task

Another hypothesis in regards to the multi-task learning is that conflicting gradient directions between tasks is one of the key issues to the multi-task optimization problem, as conflicting gradient directions could hypothetically update the shared parameters in a way that is detrimental to making progress. The conflicting direction of gradients is often measured by comparing the cosine similarities between tasks, with a *negative cosine similarity* indicating conflicting gradients. Though much of this work has been done in the realm of multi-task learning, it is easily adaptable to the auxiliary task scenario, where we can simply choose a single, target task, and compare each auxiliary gradient to the target gradient.

1.2.3 Comparison with Learning Tasks Individually

When encountering a multi-task problem, it is natural to train a network that jointly predicts each task. This way, we can ideally discover some shared structure across some or all tasks such that we can realize higher performance and better efficiency. However, multi-task learning results in a complex optimization problem which can lead to *worse* overall performance and efficiency compared to learning each task individually [9] [10]. This provides the motivation for exploring potential solutions to tackle this multi-task optimization problem effectively, as doing so could lead to achieving the hypothesized performance gains at test time.

1.3 Overview

Though we've seen the benefits of ATL in the scenario where we auxiliary information is available, it's clear that the gradient imbalance problem hinders the overall performance of multi-task models. In this study, we seek to understand the current state-of-the-art methods that address this gradient imbalance problem. In particular, we look closely at MetaBalance [11], an algorithm that scales auxiliary gradient magnitudes towards the target task, and GradSurgery [12], which adjusts the auxiliary gradient values based on the direction agreement with the target task. We plan to run these methods on a comprehensive and public dataset, and compare the performances of these methods with a baseline model with no gradient adjustments. Finally, we further explore the gradient imbalance problem by combining these two methods and see if optimal performance is achievable.

2. RELATED WORK

2.1 Multi-Task Learning

Multi-task Learning (MTL) [13] has been used to boost training efficiency and robustness of multiple tasks by jointly training them. This means that, during training, the losses of each task are concatenated, and the total loss across all tasks are used to update the shared parameters of the model. In multi-task models, the Shared-bottom model [14] is most-widely used to capture these shared parameters. It's structure has task-specific NN layers that receive the common hidden embedding that comes from a shared bottom network. The difference between MTL and Auxiliary Task Learning is that, in Auxiliary Task Learning, we care only about the error in the target task. In MTL, we aim to jointly optimize to lower the loss across all tasks.

2.2 Auxiliary Task Learning in Recommendation

In many recommendation scenarios, we've seen the testing performance of a target task be enhanced via transfer of knowledge from auxiliary tasks. At Facebook [15] and Google [16], multiple tasks have been incorporated to improve the target task of post-click conversion rate (CVR) for their targeted ads. In other recommendation scenarios, such as song-recommendation, we've seen predictive and contrasting auxiliary tasks [17] be adapted to improve the performance of the target task.

2.3 Gradient Balancing Methods in Auxiliary-task Learning

While many methods exist for gradient-balancing in multi-task learning [18], these methods mostly down weight gradient magnitudes that dominate the target task. Further, since they are designed for multi-task learning, they care about the performance of all tasks equally. For Auxiliary Task Learning, MetaBalance [11] is a method that dynamically adjusts gradient weights either up or down depending on a specified relax-factor. It is also created for auxiliary task learning, where only the performance of the target task is considered during evaluation.

2.4 Adapting Auxiliary Tasks via Gradient-Direction based methods

Similar to some of the gradient balancing methods, many methods exist to adjust the weights of auxiliary tasks based on its direction with respect to the target task. In many of these scenarios, the auxiliary task gradients are completely ignored or masked out [19] if its direction conflicts with the direction of the target gradient. In other cases, such as GradientSurgery, the auxiliary tasks are adjusted or down weighted if their directions conflict [12]. This differs from gradient magnitude balancing methods, such as MetaBalance, because it solely focuses on direction rather than magnitude. Individually, gradient magnitude balancing seems to have better performance than direction-based methods, but in our experiments, we aim to apply both of these methods together.

3. METHODS

In this section, we clarify our design choices for experimental set-up, as well as explain our reasoning for choosing such methods. Many of our practices are standard and align with current, baseline research in this domain. However, active research in areas such as neural collaborative filtering, data sampling, and more can impact future design choices for similar experiments.

3.1 Dataset

UserBehavior-2017[*] is a public dataset of anonymized user activities and interactions from Alibaba. The data is sourced from Taobao, and it contains over 100M interactions across 4 different user behaviors: purchase, page view (PV), add to cart (A2C), and favorite. Many e-commerce and content recommendation platforms generate revenue via purchase conversions through their recommendations. Thus, we will consider the purchase behavior to be the primary task and consider PV, A2C, and favorite to be the auxiliary tasks assisting purchase performance. We represent the prediction of each behavior as a binary classification problem, where a user either does or does not interact with a particular item. Considering that this dataset only contains positive samples (e.g., only instances of interaction), negative interactions will be randomly sampled from user-item pairs with no interaction.

The size of this dataset is sufficient to capture the detailed interaction and transfer of information across tasks. It is also comparable with other benchmark datasets for similar recommendation studies. In future works, we can expand our methods across many datasets with different user behaviors.

3.2 Data Preprocessing

Since collaborative filtering in personalized recommendation relies on past user-item interactions to make predictions, we preprocess the data to ensure that each item and each user has sufficient individual information. For UserBehavior-2017, we filter out users that purchase fewer than 10 unique items. Similarly, we filter our items that are purchased by fewer than 10 unique users.

[*]https://tianchi.aliyun.com/dataset/649

We then split our interactions randomly into a standard trainnig set (70%), validation set (15%), and evaluation set (15%). Another consideration we make is the high relation between auxiliary tasks, such as add-to-cart, and our target task, purchase prediction. We avoid any possible information leakage in our experiment by filtering out user-item pairs from the auxiliary information if the user-item pair also exists in the validation or testing set for purchase interactions. In Table 1 below, we've presented the overall statistics of the raw and pre-processed datasets. An important consideration to note is that we are using a very small subset of the original data, as it ensures that each unique user and item has sufficient samples to train a model on. We will further discuss the tradeoffs and implications of our data pre-processing in later sections.

Table 1: Statistics of the Pre-processed dataset

Dataset	# Users	# Items	# Purchase	# A2C	# PV	# Favorite
Raw UB-17	987,994	4,162,024	2,015,839	5,530,446	89,716,264	2,888,258
Pre-processed UB-17	16,012	24,608	87,179	51,985	383,648	19,039

3.3 Model Architecture

While this experiment requires a multi-task network in order to test our hypothesis, designing an improved multi-task network is not the emphasis of this thesis. Thus, we will adopt the combination of MLP layer and Matrix Factorization layer as a shared bottom network, which has been widely adopted for many recommendation tasks [3]. This model is based on the Pytorch[†] implementation and be compatible with GPUs to accelerate the training process. In this model, we create a user and item embedding for both the MLP and Matrix Factorization layers respectively. Each embedding is fed through it's respective layer and concatenated in a concatenation layer in the shared bottom of the NN. Finally, this concatenated embedding is fed multiple to the task-specific towers, which are each identical in architecture. Since we treat each task as a binary classification problem, the task-specific architecture is simply a collection of linear layers outputting a single number, interpreted as a probability. The specific architecture is shown in Figure 2.

[†]https://github.com/guoyang9/NCF

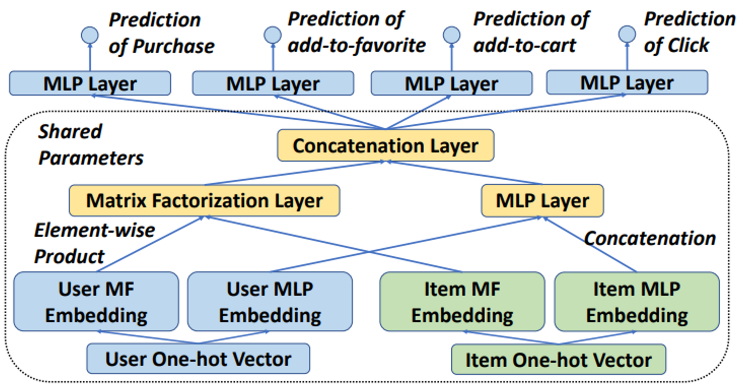

Figure 2: Implemented Multi-Task recommendation network used in training and evaluation.

3.4 Proposed Method

3.4.1 Problem Statement

Our goal of this experiment is to improve the performance of the target task (i.e., purchase prediction) via predicting auxiliary tasks (i.e., add-to-cart) alongside the target task on this neural network. Ideally, we can transfer useful knowledge from these auxiliary tasks to update the shared parameters of the model to converge towards more robust features for predicting our target task. To illustrate this objective, let θ denote a subset shared parameters of our model, outlined in Figure 2. Thus, θ could be the bias vector or weights matrix in any component of the shared bottom layer of the network, including the One-hot Embeddings or MLP layer. θ is learned by jointly minimizing the target loss $\mathcal{L}_{tar}$ with the auxiliary losses $\mathcal{L}_{aux,i}$ .. $\mathcal{L}_{aux,K}$ $i = 1, .., K$ where i represents the i-th auxiliary task for K total auxiliary tasks such that:

$$\mathcal{L}_{total} = \mathcal{L}_{tar} + \sum_{i=1}^{K} \mathcal{L}_{aux,i} \tag{1}$$

Assuming that we update the shared parameter θ via gradient descent with step size of α, then at step t:

$$\theta^{t+1} = \theta^{t} - \alpha * G^{t}_{total} \tag{2}$$

where G^t_{total} is the gradient of $\mathcal{L}^t_{total}$ with respect to the shared parameter θ:

$$G^t_{total} = \nabla_\theta \mathcal{L}^t_{total} = \nabla_\theta \mathcal{L}^t_{tar} + \sum_{i=1}^{k} \nabla_\theta \mathcal{L}^t_{aux,i} \tag{3}$$

and G_{total} is determined by adding up the gradients of the target task and auxiliary task. For simplicity, we can abbreviate the notations to mean:

- G_{tar} (e.g., $\nabla_\theta \mathcal{L}_{tar}$): the gradient of the target task loss $\mathcal{L}_{tar}$ with respect to the shared parameters θ
- $G_{aux,i}$ (e.g., $\nabla_\theta \mathcal{L}_{aux,i}$): the gradient of the i-th auxiliary task loss $\mathcal{L}_{aux,i}$ with respect to the shared parameters θ, where $i = 1, .., K$
- $||G||$: The L2 norm of the corresponding gradient

However, two key issues arise in the optimization of the shared parameters using auxiliary tasks. First comes the scenario where there is a significant magnitude imbalance in the auxiliary task with respect to the target task. As seen from Equation 3 above, the total gradient of step t, G^t_{total} is determined by the cumulative sum of the target gradient and auxiliary gradient. Thus, if $||G_{aux,i}||$ is significantly greater than the target gradient $||G_{tar}||$, then the auxiliary gradients will have more influence in how the shared parameters θ are updated, thus leading to lower overall performance. In the converse scenario, if $||G_{aux,i}||$ is much smaller than $||G_{tar}||$, the the auxiliary gradient will have very little influence on updating the shared parameters θ, thus becoming too weak to assist the target task.

Another key issue is the scenario where there is a direction dispute in how to update the shared parameter θ. In this case, gradients from different tasks will point away from each other, indicated by a negative inner product between the gradient vectors. This disagreement is visualized in Figure 3. While the existence of direction disagreements can cause optimization problems, they are not detrimental on their own. A simple average of the gradients could, theoretically, provide the correct direction and step to descend the multi-task objective. It is the co-occurence of direction disagreements with other scenarios, such as a gradient magnitude imbalance or a high curvature in the multi-task landscape that can lead to significant degradations in optimization performance. For example, if a gradient magnitude imbalance co-occurs with direction disagreements, the task gradient with higher magnitude will dominate the optimization direction. In the case where high positive curvature co-occurs with a direction disagreement, then the improvement in performance from the dominating auxiliary task may be significantly overestimated, while the degradation in performance from the dominated target task may be significantly underestimated. We look to solve these 2 key optimization issues using existing, well-tested algorithms introduced in the next section.

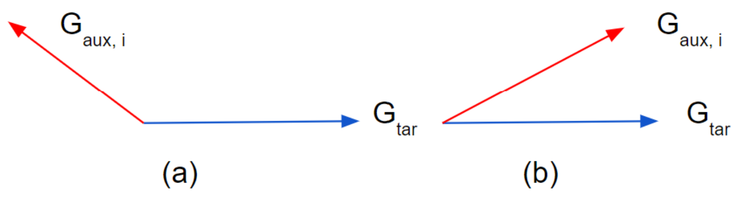

Figure 3: In (a), the target task and auxiliary task i have conflicting gradient directions, which can lead to destructive interference. In (b), the target task and auxiliary task have direction agreement, leading to constructive interaction in updating θ

3.4.2 MetaBalance for Gradient-Magnitude Balancing

While there are many methods for directly adjusting the gradient magnitudes during training, MetaBalance [11] proves to be one of the most effect methods that dynamically adjusts gradient magnitudes to yield better performance results. Thus, in this experiment, we utilize MetaBalance as our method to adjust the gradient weights. MetaBalance calculates the gradient magnitudes within each training iteration before applying a strategy (either reducing and/or enlarging) to manipulate the magnitude proximity of auxiliary gradients.

The key issue with auxiliary task optimization comes from the gradient magnitude imbalance of the gradients G_{tar} and $G_{aux,i}$ ... $G_{aux,K}$. This magnitude imbalance may negatively effect the optimization of the target task performance in many ways. MetaBalance proposes an algorithm to adapt auxiliary gradients from the perspective of gradient magnitude. The basic formula for MetaBalance is as follows:

1. **Calculate the gradients**. In each training iteration, we calculate the losses of G^t_{tar} and $G^t_{aux,i}$ with respect to the shared parameter θ (e.g., $\nabla_\theta \mathcal{L}_{tar}$ and $\nabla_\theta \mathcal{L}_{aux,i}$).

2. **Applying the optimization strategy**. After calculating the target and auxiliary gradients, we can choose which optimization strategy to apply. We can scale up the auxiliary gradients with smaller magnitude compared to the target task, reduce the auxiliary gradients with larger magnitude compared to the target task, or apply both strategies in conjunction with one another. Typically, this strategy can be selected based on the performance of each respective strategy on the validation set.

3. **Balancing the gradients**. Each auxiliary gradient $G^t_{aux,i}$ is balanced by first dividing by the norm $||G^t_{aux,i}||$ to create a unit vector, then scaled to have the same magnitude as the target gradient by multiplying by $||G^t_{tar}||$. We can control the magnitude proximity to the target task via the relax factor [11], r, that is a hyper-parameter. Thus, we balance $G^t_{aux,i}$ with the following equation:

$$G^t_{aux,i} = (G_{aux,i} * \frac{||G^t_{tar}||}{||G^t_{aux,i}||}) * r + G^t_{aux,i} * (1 - r) \tag{4}$$

4. **Calculate Gradient Moving Averages.** Rather than using the $||G^t_{tar}||$ and $||G^t_{aux,i}||$, we apply the moving averages of the gradients, m^t_{tar} and $m^t_{aux,i}$ respectively, to account for the variance in magnitudes across training iterations. Overall, this results in better performance and stability in the training as seen in [20].

5. **Updating Shared Parameters**. After G^t_{total} is calculated by summing G^t_{tar} together with the balanced $G^t_{aux,1}..G^t_{aux,K}$, G^t_{total} is used to update the shared parameters θ via gradient descent.

3.4.3 Gradient Surgery for direction based manipulation

In order to address the potential issue of direction disagreements between task gradients, there are many potential solutions. Cosine similarity between the gradient vectors of multiple tasks is a common measurement used to determine whether there is a direction disagreement [19]. A common solution is to mask out gradients with a direction disagreement with the target task. However, we still aim to use this conflicting information between the gradients to achieve better convergence for our model and masking out these gradients entirely would, in theory, remove all this information from our parameter updates during back-propagation. Thus, for this study, we will address conflicting gradient direction using Gradient Surgery [12]. Gradient Surgery (GradSurgery) is a multi-task algorithm that manipulates the gradients of each task based on their direction disagreement with all other tasks by projecting the gradients of a disagreeing task onto the normal plane of the current task. The basic implementation of GradSurgery is as follows:

1. **Compute Cosine Similarity**. In each training iteration, for each pair of tasks, $\mathcal{T}_i^t$ and $\mathcal{T}_j^t$, the cosine similarity between gradients G_i^t and G_j^t is computed to determine whether or not there is a direction disagreement, with negative cosine similarities indicating disagreement.

2. **Applying Projection**. If the cosine similarity is negative, we adjust the gradient G_i^t via its projection onto the normal plane of the complimentary gradient G_j^t such that:

$$G_i^t = G_i^t - (G_j^t * \frac{G_i^t * G_j^t}{||G_j^t||^2}) \tag{5}$$

   If the cosine similarity is positive, then there is no direction disagreement and the inital gradient G_i^t remains the same.

3. **Repeating Projection on all Tasks**. This process is then repeated across all task pairs $\mathcal{T}_i^t$ and $\mathcal{T}_j^t$ where $i \neq j$, where the tasks are sampled in random order in the current batch iteration.

The original implementation of GradSurgery was created for the multi-task scenario where the evaluation performance of all tasks is equally important. In our case, we only care about the performance of the target task (e.g., purchase prediction), so we can easily adjust GradSurgery to update the auxiliary task gradients with respect to the target task gradient, rather than updating all task gradients. We can see a visual representation of our GradSurgery in Figure 4.

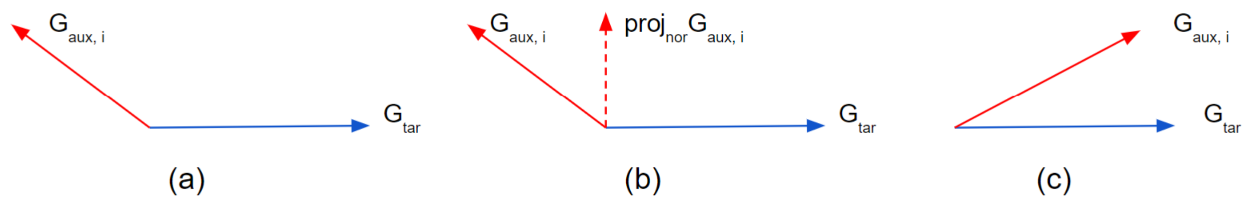

Figure 4: In (a), we see there is a negative cosine similarity between auxiliary and target gradients. In (b), we adjust the auxiliary gradient via projection onto the norm of the target gradient. In (c), since the cosine similarity is positive, no change is made to auxiliary gradients.

When evaluating our results, we are also curious to see if the original implementation of GradSurgery, which updates the gradients of every task via projection onto the normal of every other task gradient, can perform well. We will include these results in the next section, where we compare the effectiveness of our adjusted GradSurgery (i.e., all auxiliary tasks are projected onto the normal of the target task) vs the original GradSurgery (all gradients, inluding the target task, are adjusted with respect to all other gradients).

3.5 Evaluation Criteria

To evaluate our model, we first consider and rank all items for each user according to the purchase probability, with the top-K items being compared against the ground-truth. Then, we adopt three metrics: Normalized Discounted Cumulative Gain (NDCG@K), Precision (P@K), and Recall@K (R@K)

3.5.1 Normalized Discounted Cumulative Gain

Normalized Discounted Cumulative Gain (NDCG) is a well-known metric that measures the ranking quality of web searches and recommendation. We first consider the Discounted Cumulative Gain (DCG), which penalizes relevant items being ranked lower in the query by reducing the relevance logarithmically proportional to the ranking position of the item itself. In this scenario, since we treat each task as a binary classification problem, the relevance score for item i, rel_i, will be 0 or 1:

$$DCG@K = \sum_{i=1}^{K} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{6}$$

$$NDCG@K = \frac{\sum_{i=1}^{K} \frac{2^{rel_i}-1}{log_2(i+1)}}{\sum_{i=1}^{REL_K} \frac{2^{rel_i}-1}{log_2(i+1)}} \tag{7}$$

3.5.2 Precision

Precision at K is a common metric in information retrieval and recommendation. Precision at k is the proportion of recommended items in the top-k set that are relevant. Mathmatically, it can be represented as the number of relevant items in top k divided by k:

$$P@K = \frac{\sum_{i=1}^{K} rel_i}{K} \tag{8}$$

3.5.3 Recall

Recall is often used in conjunction with Precision to evaluate recommendation algorithms. Recall at k is the proportion of relevant items found in the top-k recommendations. Thus, as our K grows towards the total number of items in our item set, recall is monotonically increasing. Assuming we have M items that are relevant to our user, then:

$$R@K = \frac{\sum_{i=1}^{K} rel_i}{M} \tag{9}$$

3.5.4 Final Metrics

In the final experiments, we will use NDCG@10, NDCG@20, P@10, P@20, R@10, and R@20 to measure the robustness and performance of our model. This will give us a better sense of model performance, as the top 10-20 entries are likely to be seen by an end-user. Furthermore, the presence of position bias in recommender systems make it likely that a higher ranked item (regardless of true relevance) is more likely to be selected. We will show our metric outcomes in the next section.

4. RESULTS

To accurately measure the potential improvements of our experimental methods, we must have a sound baseline from which we can compare our results. In this section, we will specify our baseline model, a vanilla multi-task model with no gradient adjustments, and present how each of our experimental methods compare to the performance of the baseline method. We also analyze which method has the highest performance as well as interpret our findings and how they differ from our initial hypothesis.

4.1 Hyper-parameters

Here we outline the specifics of the model architecture as well as the hyper-parameters defined in training. In the multi-task network, the size of the user and item embeddings is 64. The shared MLP layer has 3 linear layers of size $\{32, 16, 8\}$. The size of the task-specific MLP layers are $\{64, 32\}$ and then predicted via binary classification. We also specify a 50% dropout rate for each layer to prevent overfitting [21] and apply a decay rate of e^{-7}. Furthermore, we use a Rectified Linear Unit (ReLU) as our activation function for each layer. We trained the model with batch size 256. For MetaBalance, we selected our relax factor by choosing the value that had the highest performance on the validation set. The results for this hyperparameter tuning can be seen in Table 1. Interestingly, the vanilla multi-task baseline, $r = 0.0$, seemed to have high performance for NDCG metrics. We also see that the lower values for relax factor tended to have higher performance, so for the experiments, we selected the relax factor to be 0.5, as it had the best overall performance on the validation set for the metrics of the top 10 items, while remaining a top 2 performer in all other categories. Further, we set the value β, which controls the exponential decay rate of the moving average of MetaBalance, to be empirically set to 0.9 based on the original study.

4.2 Experimental Trials

For this experiment, we are curious as to the performance of both MetaBalance and GradSurgery on the dataset. All experimental trials trained the model on the training set, and evaluation metrics were computed using the holdout test set. For comparison, we use a baseline method, Vanilla-Multi, that is a simple, multi-task network with no adjustment to it's gradient values. We then run a version of MetaBalance with the optimal relax factor selected over the validation set, which can be seen in Table 1 below. Our third experiment uses GradSurgery as the gradient optimization method, adjusted from the original implementation to fit the auxiliary task scenario where we only care about the performance of the target task. Finally, we are curious as to whether these two methods can be used together in a collaborative way to achieve higher performance that when these methods are used individually. For these combined methods, we try using MetaBalance in conjunction with adjusted version of GradSimilariy, as well as the original implementation of GradSimilarity. The results of these experiments are shown in Table 2. The original implementation of GradSurgery was implemented for the multi-task scenario with the aim to improve performance across all task predictions. Thus, in the original version of GradSurgery, we also project the target task's gradients onto the gradients of auxiliary tasks.

Table 2: MetaBalance Relax Factor on Validation Set

Metric(%)	NDCG@10	R@10	P@10	NDCG@20	R@20	P@20
$r = 0.0$	**1.0486**	1.8175	0.2785	**1.4142**	3.1027	0.2445
$r = 0.3$	0.9479	1.8091	0.2801	1.3641	**3.3174**	**0.2564**
$r = 0.5$	1.0431	**1.8783**	**0.3007**	1.3950	3.1397	0.2485
$r = 0.7$	1.0049	1.7701	0.2722	1.3278	2.9320	0.2271
$r = 1.0$	0.9955	1.8427	0.2817	1.3458	3.0729	0.2445

4.3 Analysis

Naturally, the training over time has a significant impact on the performance of the model. We train the model using all of our training data, updating the parameters on a per-batch basis. After 20 epochs of training our model, we select the epoch with the best overall performance.

Table 3: Comparison of Methods

Metric(%)	NDCG@10	R@10	P@10	NDCG@20	R@20	P@20
Vanilla Multi	1.0407	1.7377	0.3732	1.4109	2.9271	0.3258
MetaBalance	**1.2844**	**2.1099**	**0.4408**	**1.6899**	**3.4426**	**0.3657**
GradSurgery	1.0662	1.7585	0.3884	1.4355	2.9739	0.3243
MB + GS (adj)	1.0331	1.6737	0.3642	1.3568	2.7219	0.3001
MB + GS (orig)	1.0825	1.8438	0.3843	1.3862	2.8362	0.3031

From the results, we observe that MetaBalance significantly outperformed all other methods tried. Further, the adjusted implementation of GradSurgery was able to outperform the baseline vanilla multi-task network by a significant amount, though only marginally outperformed the baseline compared to MetaBalance. Interestingly, when combining these methods together, we achieved **worse** overall performance compared to the baseline, indicating that combining these strategies in tandem is an ineffective way to achieve performance gains. This strategy of naively combining these methods can lead to either method interfering with each-other, leading to sub-optimal gains. We also decided to run an iteration of MetaBalance and GradSurgery together where we use the original implementation of GradSurgery that treats the problem as a multi-task recommendation rather than an auxiliary-task recommendation. We suspected that the multi-task implementation of GradSurgery can create a more robust model that is less likely to overfit in favor of the target task and, when combined with MetaBalance, could yield optimal results. Ultimately, we found that this method yielded similar sub-optimal results compared to running MetaBalance alone, but outperformed the use of MetaBalance with the adjusted implementation of gradient-surgery, suggesting that Gradient Surgery may not need adjustment in the auxiliary-task scenario. The original, multi-task implemetation of GradSurgery can be found here[*].

[*]https://github.com/WeiChengTseng/Pytorch-PCGrad

5. CONCLUSION

5.1 Perspectives

Recently, large investments have been made in the space of improving recommendation performance using neural network architectures. Multi-task learning has been a focal point of improving recommendation accuracy, as it helps alleviate some of the limitations of collecting quality labeled data by allowing the model to train on auxiliary data that was previously unused. From our experimental results, a relevant question would be to mention the low value ranges of NDCG, Recall, and Precision. After all, recommendation is naturally a complex problem and it is difficult to interpret implicit signals. For example, in our training, we sample unobserved user-item interactions and interpret them to be negative examples; however, it is possible that a user has a positive interaction with this item and simply has not seen it. Nonetheless, improvements in bottom line metrics such as purchase prediction, which we treated as a target task, can yield compounding returns at the scale of a large enough user base.

Advances in the field neural recommendation with multi-task learning would stand to share the breakthroughs of gradient optimization in multi-task networks, which would immensely catalyze and support research efforts outside of recommendation, such as object detection and localization, machine translation, etc. The gain to be made in this field is a deeper understanding of gradient interpretation and how to optimize the influence that each task has on the global optimization of a multi-task network.

While multi-task learning is clearly an exciting field with more discoveries to be made, it is also key to mention the importance of consistency across different studies, such as proper dataset benchmarking, consistent train-test split requirements, and standard data pre-processing techniques. Multi-task data tends to be difficult to acquire, and a non-standard benchmark dataset makes it difficult to compare the performance of different methods on a singular dataset. Similarly, some studies split their train, validation, and test datasets using different requirements, such as requiring a test user-item sample to have at minimum number of corresponding training samples for that particular user and/or item. This can skew the results of some studies, and it also avoids the cold start issue that is a common challenge for recommending new items and users. Finally, data preprocessing differences across studies make it difficult to compare methods, since some studies will do minimal pre-processing and others may filter out a subset of users or items according to different thresholds. After all, the performance of a particular method or model is based on the data that it is trained on. While smaller datasets may have advantages such as faster training times and manageability, they come at the risk of being an unrepresentative distribution of user-item samples. Thus, a common benchmark could prove useful in accelerating progress in the field.

5.2 Thesis Outcome

Both MetaBalance and GradSimilarity outperformed the baseline multi-task model on the UserBehavior dataset, indicating that careful and intelligent manipulation of auxiliary gradients can help neural recommendation systems converge to optimal solutions. In fact, both of these methods outperformed the baseline model in nearly all metrics that were considered. However, we also showed that naively joining these two methods together will achieve sub-optimal results that can perform even worse than the baseline model. This leads us to suspect that performing direction-based adjustments with magnitude based adjustments interferes with each other and offsets the initial gains compared to using methods independently. We also discovered that using the original implementation of GradSurgery in conjunction with MetaBalance leads to improved performance over the baseline and the sole implementation of GradSurgery, which suggests potential for MetaBalance to help boost the performance of GradSurgery.

Overall, this study makes it clear that a more targeted and complex approach must be considered if we seek to achieve significant out performance of current, industry-leading methods. These methods should also be applied to larger, more representative datasets in the future.

5.3 Directions for Future Work

Recommendation performance has been boosted by the innovative ideas in multi-task learning, specifically in gradient balancing and manipulation techniques to achieve better convergence when training neural-based recommendation engines. While it is widely used in industry today, there is still much work to be done. As this thesis has shown, the collaborations and interactions of gradients for different tasks, each with their own loss functions, is incredibly complex. Future works that aim to gain a deeper interoperability of these interactions could lead to more intelligent experiments being run, and a better optimization strategy could be applied.

Another key challenge that can be addressed in future works is a more standardized and widely used dataset for multi-task recommendation. A lack of consistency in datasets is one of the greatest disparity in terms of neural-recommendation research and industry needs. This is likely because these two groups are optimizing for different outcomes; in industry, the top-line metrics are the ultimate priority, leading them to keep their datasets proprietary and only exposing limited data, typically within a short time window and using very few interaction categories, for the purposes of academic research. In other fields such as computer vision, widely accepted public datasets, such as CIFAR-10 or ImageNet, has provided the community with reliable benchmarks that accelerate the progress in the field. Similar datasets exist in the recommendation domain; however, they are not suitable for the multi-task scenario. In the future, a public benchmark with adequate representation of a user-item population as well as significant amounts of data could similarly accelerate the research in auxiliary-task and multi-task problems. Furthermore, data pre-processing is another area that lacks consensus and where the strategies and practices vary depending on the study. For example, as noted in this paper, we used a very small subset of the original data that was provided to us to ensure that there was sufficient data for our model to learn about each unique user and item. While this avoids the cold-start problem in recommendation (i.e., the issue of recommending a new item or recommending to a new user with no prior history), it is not truly representative of the data at our disposal. Future works could look to improve upon this shortcoming, ideally through a proper benchmarking of recommendation datasets.

REFERENCES

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

[4] B. Shi, J. Hoffman, K. Saenko, T. Darrell, and H. Xu, "Auxiliary task reweighting for minimum-data learning," 2020.

[5] L. Liebel and M. Körner, "Auxiliary tasks in multi-task learning," 2018.

[6] T. Trinh, A. Dai, T. Luong, and Q. Le, "Learning longer-term dependencies in rnns with auxiliary losses," in *International Conference on Machine Learning*, pp. 4965–4974, PMLR, 2018.

[7] L. Liebel and M. Körner, "Auxiliary tasks in multi-task learning," *arXiv preprint arXiv:1805.06334*, 2018.

[8] T. Bansal, D. Belanger, and A. McCallum, "Ask the gru: Multi-task learning for deep text recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, (New York, NY, USA), p. 107–114, Association for Computing Machinery, 2016.

[9] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[10] E. Parisotto, J. L. Ba, and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342*, 2015.

[11] Y. He, X. Feng, C. Cheng, G. Ji, Y. Guo, and J. Caverlee, "Metabalance: Improving multi-task recommendations via adapting gradient magnitudes of auxiliary tasks," in *Proceedings of the*

ACM Web Conference 2022, WWW '22, (New York, NY, USA), p. 2205–2215, Association for Computing Machinery, 2022.

[12] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, (Red Hook, NY, USA), Curran Associates Inc., 2020.

[13] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[14] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool, "Multi-task learning for dense prediction tasks: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[15] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini, *et al.*, "Deep learning recommendation model for personalization and recommendation systems," *arXiv preprint arXiv:1906.00091*, 2019.

[16] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

[17] Y. Tao, M. Gao, J. Yu, Z. Wang, Q. Xiong, and X. Wang, "Predictive and contrastive: Dual-auxiliary learning for recommendation," 2022.

[18] 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*, pp. 794–803, PMLR, 2018.

[19] Y. Du, W. M. Czarnecki, S. M. Jayakumar, M. Farajtabar, R. Pascanu, and B. Lakshminarayanan, "Adapting auxiliary losses using gradient similarity," *arXiv preprint arXiv:1812.02224*, 2018.

[20] I. Malkiel and L. Wolf, "Mtadam: Automatic balancing of multiple training loss terms," 2020.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.