11-2023

Learning to Learn from Sparse User Interactions

Krishna Prasad Neupane
kpn3569@g.rit.edu

Learning to Learn from Sparse User Interactions

by

Krishna Prasad Neupane

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computing and Information Sciences

B. Thomas Golisano College of Computing and
Information Sciences

Rochester Institute of Technology
Rochester, New York
November, 2023

Learning to Learn from Sparse User Interactions

by

Krishna Prasad Neupane

Committee Approval:

We, the undersigned committee members, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computing and Information Sciences.

Dr. Qi Yu Dissertation Advisor	Date
Dr. Xumin Liu Dissertation Committee Member	Date
Dr. Rui Li Dissertation Committee Member	Date
Dr. Zhiqiang Tao Dissertation Committee Member	Date
Dr. Ricardo Figueroa Dissertation Defense Chairperson	Date

Certified by:

Dr. Pengcheng Shi Ph.D. Program Director, Computing and Information Sciences	Date

Learning to Learn from Sparse User Interactions

by

Krishna Prasad Neupane

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences Ph.D. Program in
Computing and Information Sciences
in partial fulfillment of the requirements for the
Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

The ability to learn from user interactions provides an effective means to understand user intent through their behaviors that is instrumental to improve user engagement, incorporate user feedback, and gauge user satisfaction. By leveraging important cues embedded in such interactions, a learning system can collect key evidence to uncover users' cognitive, affective, and behavioral factors, all of which are critical to maintain and/or increase its user base. However, in practical settings, interactive systems are still challenged by sparse user interactions that are also dynamic, noisy, and highly heterogeneous. As a result, both traditional statistical learning methods and contemporary deep neural networks (DNNs) may not be properly trained to learn meaningful patterns from sparse, noisy, and constantly changing learning signals.

In recent years, the *learning to learn (L2L)* (or meta-learning) paradigm has been increasingly leveraged in diverse application domains, such as computer vision, gaming, and healthcare to improve the learning capacity from sparse data. L2L tries to mimic the way how humans learn from many tasks that allows them to generalize often from extremely few examples. Inspired by such an attractive learning paradigm, this dissertation aims to contribute a novel L2L framework that is able to effectively learn and generalize well from sparse user interactions to collectively address the challenges of learning from sparse user interactions. The L2L framework is comprised of four interconnected components. The first component focuses on learning from sparse interactions that constantly change over time. For this, we introduce a Dynamic Meta Learning (DML) model that integrates a meta learning module with a sequential learning module, where data sparsity is tackled by the first module and the later module captures constantly changing user interaction behaviors. The second component focuses on dealing with noisy interactions to detect true user intent through uncertainty quantification. This component integrates evidential learning with meta learning, in which the former quantifies the uncertainty by leveraging evidence of each interaction and the later tackles sparse interactions. Furthermore, the concept of evidence is utilized to guide the predictive model to find more important and informative interactions in sparse data to enhance model training. The third component emphasizes dealing with dynamic and heterogeneous user behavior in sparse interactions. This component aims to ensure long-term user satisfaction by combining reinforcement learning, which handles constantly evolving user behaviors and evidential learning, which leverages evidence-based exploration to tackle data heterogeneity. The last component aims to advance the contemporary dynamic models which require to partition the time into arbitrary intervals to support model training and inference. We develop a novel Neural Stochastic Differential Equation (NSDE) model in the `L2L` setting that captures continuously changing user behavior and integrates with evidential theory to achieve evidence-guided learning. This method leverages the power of a NSDE solver to capture user's continuously evolving preferences over time which results in richer user representation than previous discrete dynamic methods. Furthermore, we derive a mathematical relationship between the interaction time gap and model uncertainty to provide effective recommendations.

Acknowledgments

Above all else, I wish to convey my deep appreciation to Prof. Qi Yu, my advisor, for the unwavering support, motivation, and passion that he has consistently shown throughout my doctoral journey. Every aspect of this thesis reflects the guidance and nurturing I received from him, whether it was the late nights leading up to paper submissions or the extended periods of mentorship. Under his guidance, I not only honed my research abilities but also experienced significant personal growth. Working with such an exceptional mentor and role model has been a genuine privilege.

I would also like to thank my doctoral committee consisting of Prof. Xumin Liu, Prof. Rui Li, Prof. Zhiqiang Tao, and the defense chair Prof. Ricardo Figueroa for their time, effort, and valuable suggestions to improve this dissertation. I would also like to thank the Ph.D. program director Prof. Phengcheng Shi, and the faculty members Charles Gruener, and MinHong Fu for continuous guidance, insights, and administrative support. I am grateful to Prof. Minseok Kwon, and Prof. Yi Wang who guided me in the early phase of my doctoral studies.

I express my gratitude to the RIT-MINING lab members (especially Ervine Zheng for long-term collaboration) for their valuable input and collaborative efforts that have significantly expanded my outlook and enriched my knowledge in the field. I would also like to express my appreciation to my friends: Aayush, Kishan, Manoj, Sushant, Sandesh, Prashnna, Kamal, Jwala, Kushal, Shusil, Utsav, Nibesh, Robik, Binyul, Hitesh, Tejan, Pradeep, Abhisekh, Shradha, Deep Shankar, Spandan, Rupak, Anjali, Sunita, Dilip, Bhawani, Bipana, Manish, and Saru for their friendship and support throughout my PhD journey.

Last but not least, I extend my heartfelt thanks to my spouse Laxmi Sharma Neupane, daughter Eliza Neupane, mother Balaki Neupane, sister Parbati Bashyal, brothers Bishnu Neupane, Dhurba Neupane, Shankar Neupane, and papa Om Prakash Sharma, for their unwavering support and motivation during challenging moments.

This page is dedicated to my late father, Pitambar Neupane...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Learning from user interactions involves understanding user intent and interaction processes, developing user models, and providing personalized services. Furthermore, the learning model leverages insights from user interaction data to enhance user satisfaction. However, learning from user interactions is still challenged by the sparse human behaviors that are also dynamic, noisy, and highly heterogeneous. As a result, existing methods are less effective when learning from large-scale historical interaction data that cover a massive user base over a long period of time whereas useful learning signals remains rather limited for each individual users that are further concealed by users' constantly changing behavior coupled with a complex interaction environment that is noisy and heterogeneous.

Recently, the Learning to Learn (`L2L`) paradigm is gaining popularity in the machine learning field due to its capability of learning and generalizing well as similar to human learning and generalization from extremely few examples (e.g. often just a single example suffices to teach a new thing) [79]. In this dissertation, we propose to leverage the `L2L` paradigm to effectively learn and generalize from sparse user interactions and further ease the dynamic, noisy, and heterogeneity problems. First, we aim to tackle the problem of constantly changing user interactions over time. Second, we further extend with uncertainty-aware quantification to deal with the problem of noisy sparse interactions. Third, we leverage exploitation and exploration concept to deal with dynamic and heterogeneity problems in sparse interaction and provides users long-term satisfaction. Finally, we tackle dynamic and noisy sparse interaction capturing continuous user intent and quantifying uncertainty-aware preferences.

In this chapter, we introduce user interactions and their importance, describe the problem statement and research challenges, summarize our contribution, mention research plans, and finally provide the outline of the dissertation.

1.1 User Interactions

With the enormous growth of Internet usage, many sectors like commerce, media, health, etc. have been shifted to digital systems. This result understating in-depth knowledge of user interactions has become a crucial research concern to provide user satisfaction and advancement of digital systems. For example, e-commerce provides personalized products for each consumer by learning the consumer's needs and interests from users' historical interactions. Further, many users' behavior is evolving over time and data are too noisy, which results most of the user interaction systems falling into the data sparsity problem. In the following paragraphs, we describe user interactions in an interactive system and their challenges in detail.

An interactive system allows a human to interact with systems in a more natural way. It has been used in many research fields like gaming, healthcare, robotics, etc to collect user interactions. For example, in the healthcare domain, virtual reality (VR) has been used as an interacting system to collect data from various participants (like painting maze games for ASD and typically developing (TD) patients). Data collection for those systems is challenging due to user dynamic behavior, and, noisy and heterogeneous data. Further, participants are very few and the training model in small data set adds extra difficulties. Similarly, in the field of e-commerce, recommending user-preferred products requires maintaining a user base. Many users are very less active and have sparse interactions. Further, interactions are noisy, changing over time, and heterogeneous which causes recommender systems very prone to maintain their user base and learn user intent.

1.2 Problem Statement and Research Challenges

Traditional model learning requires enormous user interaction data to effectively understand and learn the user intent. In many real-world scenarios, user interaction data are largely sparse, noisy, and dynamic. This results data hungry existing methods greatly lacking learning from sparse interactions, handling noisy and dynamic nature, and are ineffective in providing user intent. There requires an immediate method to deal with such problems. In this proposed dissertation, we aim to address that learning problem from challenging sparse interaction leveraging the learning to learn (`L2L`) framework. We point out the limitations of existing methods as some major challenges in learning from user interaction that motivate us to formulate the proposed `L2L` framework.

Data Sparsity. Data sparsity occurs due to an inadequate number of user interactions and causes learning difficulty to the model. This largely limits the quality of system performance. For example, in e-commerce systems, if a user has interacted with a small number of items, it is quite difficult to generate accurate preference due to limiting the learning capability of the model for that user. Existing methods heavily suffer from data sparsity problems and need to devise a novel method to handle this problem.

Dynamic Interactions. User behaviors are not static and likely to change over time. Changing behavior influences user interaction in the system. Capturing user behavior over time is the key challenge in many real-world systems to provide convincing performance. Existing systems are prone to dynamic interactions and do not adequately capture user intent to maintain their engagement in the system.

Noisy Interactions. Noisy interaction occurs when real data is corrupted or distorted by the data acquisition system and is inherent to the data. These noisy data could lead to meaningless information and cause the system to behave differently from the expected performance. Many existing methods largely suffer from data noise and are unable to suppress its effect.

Heterogeneous Interactions. Heterogeneous interactions are variable data due to quite diverse interactions from the user. The model falls into potential ambiguity while learning user preferences from those heterogeneous data. Most of the existing methods largely lack to handle such a problem and acquire poor performance.

1.3 Learning to Learn Framework for Sparse User Interactions

Our solutions to the aforementioned challenges are well organized under a novel learning to learn (`L2L`) framework to provide effective learning from sparse user interactions. The `L2L` framework and its related mechanisms are a powerful tool to tackle sparse interactions, user-shifting behaviors, noisy interactions, and heterogeneous interactions problems. Figure 1.1 depicts the overall diagram of the framework. First mechanism leverages meta-learning as the `L2L` framework, which is a class of few-shot learning (i.e. only a few samples are used while training) and combines with a sequential module to handle the time-changing sparse interaction. This integrated model captures time-specific and time-evolving user preferences and is represented as a Dynamic Meta-Learning Model (DML) as shown in the figure. Similarly, the second mechanism leverages evidence learning with the `L2L` framework (MetaEDL) to tackle noisy interaction and provide confidence prediction utilizing uncertainty-aware user preference. The model quantifies the uncertainty of prediction via utilizing epistemic uncertainty (i.e. model uncertainty) and then provides user preferences based on both epistemic and interaction values.

We further utilize `L2L` with evidence learning and reinforcement learning methods to form a model called meta-evidential reinforcement learning (MetaERL) to tackle dynamic and heterogeneous problems in user interactions. The model utilizes an evidence-based actor-critic network to provide effective exploration by

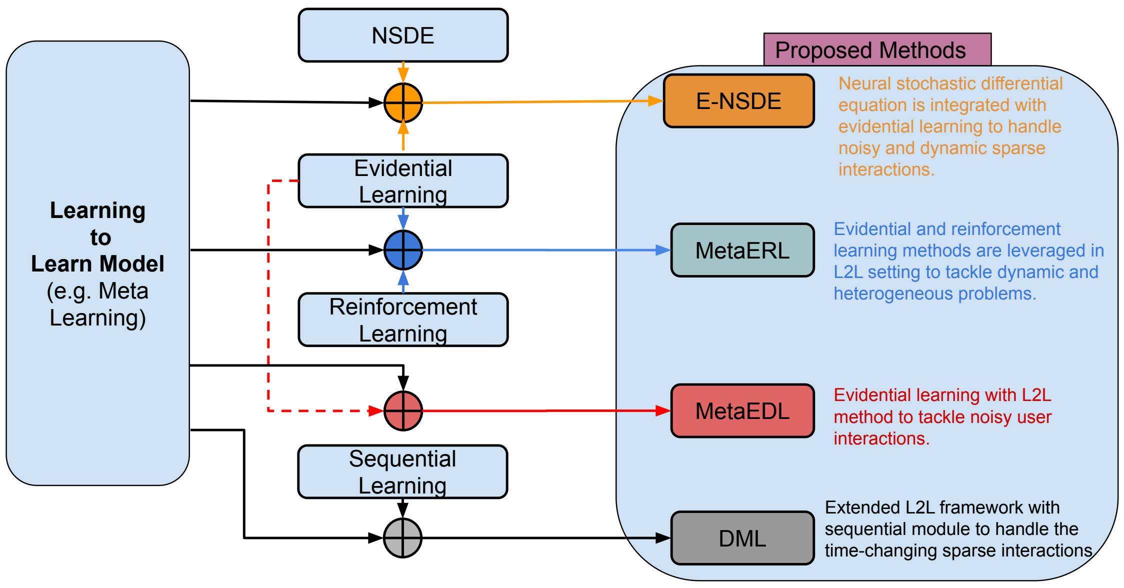

NSDE
Proposed Methods
Learning to Learn Model (e.g. Meta Learning)
Evidential Learning
Reinforcement Learning
Sequential Learning
E-NSDE
Neural stochastic differential equation is integrated with evidential learning to handle noisy and dynamic sparse interactions.
MetaERL
Evidential and reinforcement learning methods are leveraged in L2L setting to tackle dynamic and heterogeneous problems.
MetaEDL
Evidential learning with L2L method to tackle noisy user interactions.
DML
Extended L2L framework with sequential module to handle the time-changing sparse interactions

Figure 1.1: Learning to Learn (L2L) Framework for Sparse User Interactions

maximizing long-term reward. We also derive evidential policy to optimize the policy so that it provides a maximum return in long run. Besides, we also customized the RNN network to capture users' evolving nature over time. Fourth, We also provided a novel approach called E-NSDE in the L2L setting in which we integrate evidence learning with neural stochastic differential equation techniques to capture both uncertainty in prediction and the user's continuously evolving preference over time. This method essentially captures a richer representation of the users via SDE solver which is the lack of existing dynamic models and also considers predictive uncertainty to provide effective user preferences.

1.4 Summary of Contributions

We provide effective learning to learn (L2L) framework to learn from sparse user interactions. We first introduce a popular L2L approach called meta-learning with a recurrent neural network to capture user's evolving preferences over time and provide dynamic user preferences. We devise a dynamic meta learning model that contributes to effective learning from sparse user interaction. We further contribute evidence-guided meta-learning to consider evidence of the prediction that guides learning systems to provide effective user preference. Additionally, we utilize learning to learn with reinforcement learning for user dynamics and evidence learning for uncertainty-aware exploration for long-term engagement to contribute a meta evidential reinforcement learning model (MetaERL). Finally, we contribute the neural SDE method with evidential learning (E-NSDE) in learning to learn setting to capture users' continuous behavior to reflect users' real-life evolving scenarios.

1.5 Outline

This dissertation proposal is organized as follows, Chapter 2 reviews related work of learning to learn from user interactions, interactive systems, the traditional recommendation systems, and learning to learn methods. Chapter 3 proposes a dynamic meta-learning model for constantly changing interaction over time. Chapter 4 introduces meta-evidential learning to handle noisy interaction leveraging evidence for uncertainty-aware quantification. Similarly, Chapter 5 proposes two generic solutions for sparse user interactions. The first method leverages the reinforcement learning (RL) method to tackle data heterogeneity and dynamic interaction problems and provide long-term user engagement. Further, the RL method is combined with the evidential learning method to tackle noisy interactions. The second method is designed to discover users' unique behavior with the help of an evidential-reinforce attention mechanism in the RL setting for the sparse data regime. Chapter 6 includes the approach that captures the user's continuous time preference leveraging neural SDE architecture to provide more accurate user interest. In this chapter, we leverage the importance of the interaction gap and its role in modeling user uncertainty. In Chapter 7, we provide a conclusion of the dissertation and also the future work. Finally, Chapter 8 includes a list of publications and submitted papers.

Chapter 2

Literature Review

In this chapter, we provide a literature review of user interactions considering interaction systems and recommender systems. Further, as we leverage the L2L concept, we will describe related work from diverse sectors.

2.1 Learning from User Interactions

Learning from user interactions involves understanding user intent, incorporating user feedback, and improving user satisfaction [58]. We describe existing works in interactive systems and recommender systems to cover the related works of user interactions over the proposed dissertation.

2.1.1 Interactive Systems

The interactive system involves high-level user interaction with the systems. Liu et al. [54] leverage user interaction to understand the user-seeking intention in web searches. Ferro et al. [19] model user dynamics and explicit knowledge of user interactions to explore a promising area of search space. Similarly, [23] utilizes reinforcement learning from user interactions with keywords to direct exploratory searches. We further describe user interactions in a recommender system as an interactive system to provide prior works in the following sub-section.

2.1.2 Recommender Systems

In this section, we provide a literature review of existing methods from matrix factorization to recent deep learning methods for the recommendation.

Matrix Factorization Matrix factorization is a commonly used collaborative filtering approach that characterizes users and items through the latent factors inferred from the item rating patterns [46]. Using singular value decomposition (SVD) for recommendation is popularized by the *Simon Funk* [22] in the Netflix prize competition and a probabilistic version is introduced by [59]. SVD is extended to SVD++ [44] for processing the implicit feedback.

The above static models do not include the important temporal information, which should be considered for analyzing user-item interactions in a time-sensitive setting. Dynamic matrix factorization has been developed to address the issue, which allows latent features to change with time. Some works introduce time-specific factors, such as timeSVD++, which uses additive bias to model user-related temporal changes [45]. However, only using time-specific factors may over-fit the model to sparse input data, because the dependency of latent variables across time is not considered. Other works introduce time-evolving factors, where the Markov structure is assumed in which latent variables at one period are dependent on variables in the previous period. There have been works that employ Gaussian state-space models to introduce time-evolving factors with a one-way Kalman filter [27, 75]. To handle implicit data, Sahoo et al. propose an extension to the hidden Markov model [67], where clicks are drawn from a negative binomial distribution and Charlin et al. [11] introduces a Gaussian state-space model with the Poisson emission. However, the Markov structure may not work well with factors oscillating a lot instead of evolving from time to time, as it may not capture the time-specific factors that may significantly affect users' activity and items' popularity. Besides, matrix-factorization-based algorithms may suffer from limited expressive power and may not be able to capture the complex nature of user-item interactions.

Deep Learning Models Recent works in recommender systems [13, 28, 96] utilize deep learning to provide better recommendations. Cheng et al. [13] propose to jointly train wide linear models and deep neural networks to combine the benefits of memorization and generalization. Similarly, DeepFM [28] integrates the power of deep learning and factorization machines models to learn low- and high-order feature interactions simultaneously from the input. DIEN [96] formulates interest evolution network as a deep learning model to capture latent temporal interests and evolving interests for better recommendations. These models are very sensitive to the features and might need important features information and large datasets while training. Also, sequential recommendation systems with RNN [33] that utilizes whole session information and hierarchical attention networks [93] which consider long and short-term user preferences are used to recommend next items. However these methods are not specifically designed for the cold-start users and required large datasets. Moreover, differing from traditional sequential recommendation where next item is predicted based on previous interactions, our model considers limited current interactions to learn user's recent preference using meta-learning and uses remaining historical interactions to capture user's long-term interest.

Graph-Based Models Another popular line of recommendation systems is graph-based models. A graph captures high-order user-item interactions through an iterative process to provide effective recommendations [30]. Users and items are represented as a bipartite graph in [9] and links are predicted to provide recommendation. Similarly, a graph-based framework called Neural Graph Collaborative Filtering (NGCF) [84] explicitly encodes the collaborative signal in the form of high-order connectivities in a user-item bipartite graph via embedding propagation. However, these methods are unable to capture long-term user preferences or deal with cold-start problems.

Sequential Models Sequential models understand the sequential user behaviors via user-item interactions, and model the evolution of users' preferences and item popularity over time [17,83]. Tang et al. [78] utilizes convolutional sequence embedding to capture union level and point level contributions of historical items via horizontal and vertical filters and provides top-N sequential recommendations. Similarly, Kang et al. [39] introduce a self-attentive mechanism to handle both long and short-term user preferences in a sequential setting. Also, some graph-based sequential recommendation models are introduced to provide the temporal preference [21,86]. In each session, a sequence graph is created between consecutively clicked items and the model predicts the next item. These methods are constructed to predict the next item without specifying a specific user in each session and hence not suitable for cold-start user recommendation. Besides, Sequential models focus on users' evolving preferences (i.e., recent interactions) but largely neglect long-term users' preferences.

Meta-learning Models The user-item interaction data is usually sparse because a user may only interact with a few items within the large item pool. In such cases, making recommendations can be viewed as a few-shot learning problem. Meta-learning [6,68] is recently becoming a popular few-shot learning approach that learns from similar tasks and can generalize quickly and efficiently for the few-shot unseen new tasks. Finn et al. [20] propose a model-agnostic meta-learning model that learns global parameters from a large number of tasks and performs as a good generalization on a new task that has few samples utilizing the few steps of gradients. To address the cold-start problem in item recommendation, Vartak et al. [81] introduce a meta-learning strategy with two deep neural network architectures: a *linear classifier network* whose weights are learned from the item history, and a *neural network* whose biases are adjusted with the item history. It that focuses on the items-cold-start problem to recommend cold-start items considering that items arrive continuously in the Twitter Timeline. On the contrary, our model focuses on recommendations for the time-sensitive, cold-start users by capturing users' preferences over time. Similarly, recent work based on meta-learning is done to estimate user preferences in [47] and scenario-specific recommendation in [16]. Both works use the information of users and items to generate user and item embeddings. Meta-learning is specifically utilized to learn heterogeneous information networks to alleviate cold-start problems [57]. Also, meta-learning is applied in click-through rate (CTR) prediction [63] where desirable embeddings for the new ads are generated via meta-learner. These embedding methods are designed for a static setting, making them not applicable to learn user preferences evolving over time.

2.2 Learning to Learn Methods

Learning to learn is a new research direction in machine learning [10]. As a single example is sufficient for a human to correctly generalize, learning to learn method effectively learns and is generalized with extremely few examples [1] as similar to human learning. Meta-learning [68] is gaining popularity as a learning to learn approach that learns from similar tasks and can generalize quickly and efficiently for unseen new tasks with limited data samples. A popular way of meta-learning is to train meta-learner which efficiently updates model parameters in a few-shot setting [6,69]. The learning meta-learner concept is applied in the DNN and optimizes the DNN parameters in [35]. Finn et al. [20] propose a model-agnostic meta-learning model that learns global parameters from a large number of tasks and performs as a good generalization on a new task that has few samples utilizing the few steps of gradients.

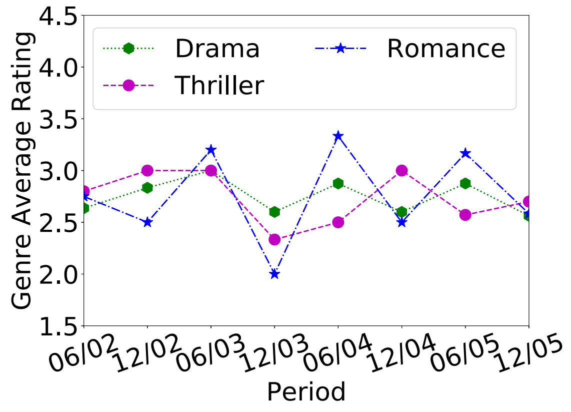
Chapter 3

Dynamic Meta-Learning for Sparse User Interactions

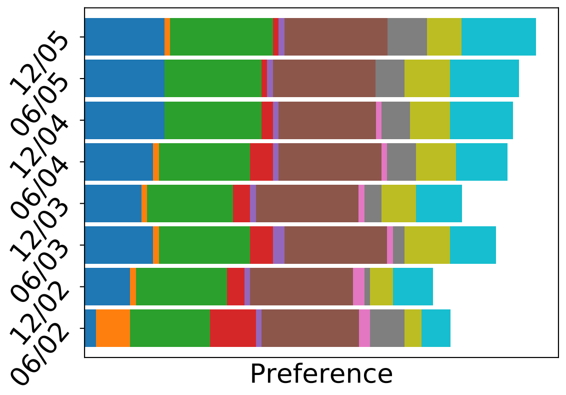
In this chapter, we discuss a dynamic meta learning model to tackle the problem of sparse user interactions changing over time. Learning and understanding from those sparse interactions to achieve accurate user intent require effective means of learning paradigms. Our proposed method aims to address that problem by leveraging the L2L framework which combines meta learning module, and a sequential module. The former module effectively learns and is generalized well from very limited interactions, and the latter module captures dynamic user interaction behavior over time. The dynamic meta learning model can be applied in various applications such as e-commerce, healthcare, gaming, robotics, etc which are suffered from sparse as well as dynamic user interactions. In this chapter, we apply the dynamic meta learning model in the recommeder systems as an application of the e-commerce sector to tackle sparse and dynamic user interaction problems.

3.1 Dynamic Meta-Learning for Recommendation

The recommender system has long been used as an effective means to improve user experience and to provide personalized recommendations in diverse fields such as media, entertainment, and e-commerce. [75, 87]. One effective way of recommendation is via Collaborative Filtering (CF) [25, 73], which recommends items based on similar users' preferences. CF assumes that users who had similar interactions with some items in the past are likely to have the same preference on other items, and leverages the observed user-item interactions to make predictions for the missing parts, which indicate the potential items of interest to users.



(a)



(b)

Figure 3.1: For User (ID:2181970): (a) shows dynamic changes of ratings for three genre types over time (b) demonstrates how user's latent preferences evolve over time.

Matrix Factorization (MF) is one commonly used CF technique that exploits user and item latent factors to capture their inherent attributes. Most existing MF methods model user preferences and item attributes as static factors [44, 46]. Some recent efforts consider the dynamic changes in the user-item interactions by modeling user preferences as shifting latent factors over time, allowing them to provide more timely recommendations [11, 27, 75].

When user-item interactions are very sparse, making accurate recommendations based on limited information is highly challenging, which is usually referred to as the cold-start problem. A viable solution to handle the cold-start problem is to utilize extra side information such as item descriptions, user profiles [80], and social relationships [91]. While various side information is widely available for items (i.e., movies, songs, and books), the user-related side information is typically very scarce as acquiring users' personalized information may be practically difficult due to privacy issues. Several recent works adopted meta-learning for few-shot learning in the recommender systems to alleviate the cold-start problem [16,47,81]. Meta-learning aims to learn global knowledge from the historical information of many similar tasks (users) and then provide quick adaptation for a new task (user) with limited interaction information. Although the meta-learning approaches show promising results [47, 81], they are primarily designed for static settings and hence not effective in providing *timely recommendations that best reflect users' current interests*.

Figure 3.1a shows that the average movie ratings for three genre types of an example user from the Netflix dataset, which oscillate significantly from 2002 to 2005. The average rating indicates an overall satisfaction on each type of genre items that the user interacts with for each period, and serves as an indicator for the change of users' preference. Those changes are usually caused by the interplay of two contributing factors.

First, a user's preference over different types of items (e.g., movie or music) may change over time, which we refer to as the *time-evolving factor*. Second, a user's behavior in a specific period may vary significantly from other periods due to the impact of money/time budget or other external causes, which we refer to as the *time-specific factor*. Given sufficient user interactions over time, both factors could be effectively learned to provide accurate and timely recommendations. However, in practice, many users may have interactions in prior periods but become largely inactive with few interactions in recent periods for various reasons. We define these users as the *time-sensitive cold-start users* due to scarce recent interactions. Solely relying on the historical interactions of these users may lead to outdated recommendations that do not match their recent interests. Furthermore, the limited recent interactions pose a user cold-start problem for the current period that makes existing CF-based techniques less effective.

The main challenge with these time-sensitive cold-start users is to simultaneously capture their most recent interests and evolving preferences, which are keys to achieve accurate and timely recommendations. In this work, we focus on this special *time-sensitive cold-start problem*, which is critical for a recommender system to maintain its user base. We propose to dynamically factorize a user's (latent) preference into time-specific and time-evolving representations in order to capture the time-specific and time-evolving factors from both the historical and current user-item interactions. For example, the Netflix dataset consists of a user's interactions with a large movie set in the form of user ratings. The variation of movie ratings from the same user may be affected by the change of user preference, rating criteria, or other (unknown) factors. In addition, a user's preference for different genres may evolve over multiple periods, as shown in Figure 3.1b, which corresponds to the proportion of different factors in the time-evolving representation discovered by our proposed model.

The proposed model consists of two distinct modules: a *meta-learning module* and a *recurrent module*. The former aims to capture time-specific latent factors through limited interaction data by leveraging the shared knowledge learned from other users. The latter aims to capture time-evolving latent factors by nesting a recurrent neural network, and it can be jointly optimized with the meta-learning module through the model-agnostic meta-learning approach [20]. Finally, we seamlessly integrate the two modules by merging the time-specific and time-evolving factors to form the user representation. This user representation further interacts with an item embedding (which is also optimized during model training) to provide the final recommendations. Our experimental results clearly show that the proposed model makes timely recommendations that closely resemble the dynamically changed user ratings as a result of effectively integrating the complementary factors capturing the user preferences.

The **main contributions** of this work are five-fold: (i) the **first work** to formulate the time-sensitive cold-start problem that is critical to maintain the user base of a recommender system; (ii) a **novel integrated**

recommendation framework to model sparse dynamic user-item interactions and extract time-evolving and time-specific factors of user preferences simultaneously; (iii) a **time-sensitive meta-learning module** to effectively handle user cold-start problems by leveraging knowledge shared across multiple users from the current recommendation period to adapt to any specific user's case using limited interaction information, (iv) a **time-evolving recurrent module** to effectively capture the gradual shift of users' preferences over time, and (v) an **integrated training process** that combines these two models to simultaneously learn time-specific and time-evolving factors and optimizes the item embedding.

We conduct extensive experiments over multiple real-world datasets and compare with representative state-of-art dynamic and meta-learning-based recommender systems to demonstrate the effectiveness of the proposed model.

3.2 Related Work

Related work for this work is described in the section 2.1.2 of Chapter 2.

3.3 Proposed Model

Problem Settings. We propose a dynamic recommendation model, where the input data is represented as $\{\mathcal{U}, \mathcal{I}, \mathcal{H}\}, \mathcal{U}$ is the user set, $\mathcal{I}$ is the item set, and $\mathcal{H}$ is the set of time periods. A time period $t \in \mathcal{H}$ defines a particular time interval where the interactions for user u with the item i are aggregated based on timestamps. We perform recommendation in each time period with a recommendation function as

$$f^t_{\theta^t_u,\omega}(i) = \hat{r}^t_{(u,i)} \qquad \forall u \in \mathcal{U}, i \in \mathcal{I}, t \in \mathcal{H} \tag{3.1}$$

where $\hat{r}^t_{(u,i)}$ is the recommended score for item i assigned by user u at period t, θ^t_u is user latent factor at time t, and ω is the parameter of the recurrent neural network module. The goal of a recommender system is to predict the recommendation scores that can accurately capture a user's true preference on items over time so that the recommended items are likely to be adopted by the users.

We formulate dynamic recommendations as a few-shot regression problem in the meta-learning setting. Users are *dynamically* partitioned into *meta-train* and *meta-test* sets based on their interactions in the current recommendation period t. In particular, the meta-train user set includes users with sufficient interactions, while the meta-test user set includes time-sensitive cold-start users who have only a few interactions in the current time period. Details for the train-test user splits are discussed in the experiment section. We consider

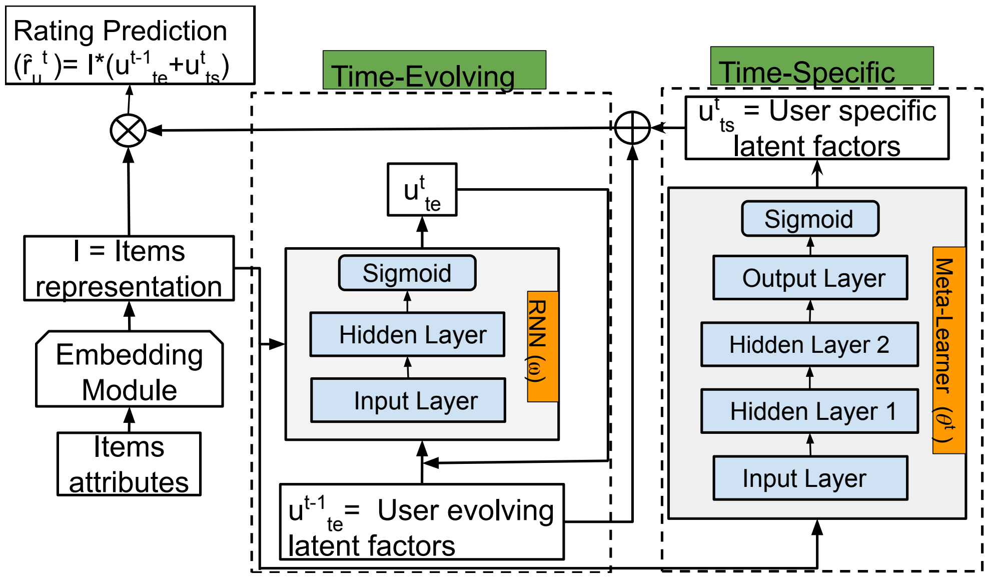


Figure 3.2: The proposed model captures time-evolving factors via a recurrent network module and time-specific factors through a meta-learning module.

a distribution over tasks $P(\mathcal{T})$, and each user is represented as a few-shot regression task $\mathcal{T}_u^t$ sampled from the given task distribution. In general, a task includes a *support set* $\mathcal{S}_u^t$ and a *query set* $\mathcal{Q}_u^t$. The support set includes a user's interactions at period t where k is interpreted as the number of shots (i.e., interactions). The query set includes the rest interactions of this user at period t.

$$\begin{aligned} \mathcal{T}_u^t \sim P(\mathcal{T}): \quad & \mathcal{S}_u^t = \{(u, i_j), r^t_{(u,i_j)}\}_{j=1:k}, \\ & \mathcal{Q}_u^t = \{(u, i_j), r^t_{(u,i_j)}\}_{j=k+1:N_t} \end{aligned} \tag{3.2}$$

where N_t is the number of items a user interacted with at period t and $r^t_{(u,i_j)}$ represents label (i.e. rating or count) from user u to item i_j. We adopt episodic training [82], where the training task mimics the test task for efficient meta-learning. The support set $\mathcal{S}$ in each episode works as the labeled training set on which the model is trained to minimize the loss over the query set $\mathcal{Q}$. The training process iterates episode by episode until convergence.

We summarize the major notations used throughout the work in Table 3.1.

3.3.1 Model Overview

To leverage item information such as text descriptions, our model generates an initial item representation (I) using an embedding matrix $E \in \mathbb{R}^{d \times m}$, where m is the dimension of input item attributes, and d is the dimension of embedding. The embedding is generated from item attributes following [13, 47]. An item i is

Table 3.1: Summary of Notations

Notation	**Description**
u, i	user and item
z_i, e_i	item i's original encoding and embedding
$\hat{r}^t_{(u,i)}$, $r^t_{(u,i)}$	predicted and ground truth scores for user u on item i at period t
u^t_{ts}, u^t_{te}	time-specific and time-evolving factors of user u at period t
θ^t, θ^t_u	meta and user-specific parameters of time-specific module at period t
ω	parameter of time-evolving module
$\mathcal{S}^t_u$, $\mathcal{Q}^t_u$	support and query sets in task corresponding to user u at period t
D^t_u	items interacted with user u at period t

first encoded as a binary vector $z_i \in \mathcal{R}^m$, where the corresponding index of item attributes is set to 1 and 0 otherwise. The binary vector is then transformed using the embedding matrix: $e_i = Ez_i$. The embedding of all items can be stacked as: $I = [e_1, e_2, ..., e_n]$ where n is the total number of items. The embedding matrix E will be optimized along with the model training process after the user latent factor is learned, and details are provided at the end of this section.

Figure 3.2 shows that the proposed model consists of a time-specific meta-learning module and a time-evolving recurrent neural network module to generate time-specific user latent factors u^t_{ts} and time-evolving latent factors u^t_{te}, both of which contribute to the final prediction $f^t(u, i)$. Details of them are described in following sections. After both modules are trained, the model learns time-specific user factors u^t_{ts} and time-evolving user factors u^t_{te}. These user factors are merged to interact with the item embedding to provide recommendations for the user. The recommendation for a user u at the current period t is denoted as a vector $\hat{r}^t_u$.

Making recommendations can be viewed as a regression problem. By using the mean square error (MSE) function, the loss for a specific user u is formulated as:

$$\begin{aligned} \mathcal{L}_{\mathcal{T}^t_u}[f^t_{\theta^t_u,\omega}] &= \sum_i ||f^t_{\theta^t_u,\omega}(i) - r^t_{(u,i)}||^2_2, \\ f^t_{\theta^t_u,\omega}(i) &= (u^t_{ts} + u^{t-1}_{te}) \cdot e_i \end{aligned} \tag{3.3}$$

where $r^t_{(u,i)}$ is user-item interaction (rating or count). The user representation is the vector sum of time-specific u^t_{ts} and time-evolving user factors u^{t-1}_{te} (as a compact single representation reducing the number of trainable parameters that helps to avoid overfitting), and the prediction is achieved by the dot product of u and item embedding i. Note that prediction for the current time includes time-specific user factors from current time period i.e. u^t_{ts} and time-evolving user factors from the previous time period i.e. u^{t-1}_{te}. In general, dynamic recommender systems utilize the information from the previous period to predict for the next period, and we follow this standard setting.

The total loss is formed by aggregating all users in the meta-train set, regularized by the L_2 norm of key model parameters. Let θ_u^t and θ^t denote the local (i.e., user-specific) and global parameters of the time-specific meta-learning module, ω denote the parameters of the time-evolving recurrent neural network module. Training of a dynamic recommendation model can be formulated as:

$$\begin{aligned}\arg\min_{\theta^t,\omega} \quad & \sum_{\mathcal{T}_u^t \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega}] + \frac{\lambda}{2}(||\theta^t||_2^2 + ||\omega||_2^2), \\ & \theta_u^t = \theta^t - \alpha_{\theta^t} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta^t,\omega}]\end{aligned} \tag{3.4}$$

where θ_u^t is one gradient step from global parameter θ^t of the meta-learned time-specific module with α being the step size and λ is the regularization parameter.

3.3.2 Time-Specific Meta-Learning Module

This module aims to capture time-specific user factors by only considering the information from the current recommendation period. The meta-learner takes input from the specific period, which is a different setting than the existing meta-learning-based recommender systems [16, 47]. In this way, the model can capture the latent factors associated with that specific period to provide more accurate and timely recommendations. We consider each user as a learning task. Our goal is to learn a meta parameter θ^t that represents a time-specific global user representation given the meta-training set. We follow the standard setting of few-shot learning [20], where the distribution over tasks is represented as $p(\mathcal{T})$. The model is trained iteratively by sampling tasks from $p(\mathcal{T})$. The meta-learning module generates time-specific user latent factors (u_{ts}^t) as:

$$u_{ts}^t = f_{meta}^t(\mathcal{T}_u^t; \theta^t) \tag{3.5}$$

where $\mathcal{T}_u^t$ represents task of a user u at period t. The task $\mathcal{T}_u^t$ includes $\mathcal{S}_u^t$ and $\mathcal{Q}_u^t$. We first pass $\mathcal{S}_u^t$ into f_{meta}^t to adapt user-specific model parameter θ_u^t from the global user model parameter θ^t and then we provide $\mathcal{Q}_u^t$ into the f_{meta}^t to generate time-specific user factors (u_{ts}^t).

We apply an optimization-based meta-learning approach [20] to learn time-specific user factors, as shown in Figure 3.2. The meta-learning network consists of one input layer, two fully connected hidden layers, and one output layer. The first and second hidden layers have 128 and 64 hidden units with ReLU activation, while the last layer estimates time-specific user factors with a linear function followed by sigmoid activation. The input to the meta-learning model is the item embedding for the users on a particular period. Algorithm 4 shows the training process that learns the model parameters. For the time-specific module, the local update (line 7) is done for the user specific parameter, which is achieved by one or more gradients from the global parameter:

$$\theta_u^t = \theta^t - \alpha \nabla_{\theta^t} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta^t,\omega}] \tag{3.6}$$

In this update, the loss function is computed with the support set. Similarly, the global update (line 11) is conducted with the new item interactions of each user from the query set for the meta update:

$$\theta^t = \theta^t - \beta \nabla_{\theta^t} \sum_{\mathcal{T}_u^t \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega}] \tag{3.7}$$

This process continues to find a good global parameter shared by all users in each period.

3.3.3 Time-Evolving Module

User preferences usually change dynamically over time. By capturing the time-evolving factors and integrating them with the time-specific factor, the proposed model can recover the user's true preference more accurately. To this end, we formulate time-evolving user factors (u_{te}^t) for each user using a nested recurrent neural network (RNN):

$$u_{te}^t = f_{rnn}^t(u_{te}^{t-1}, D_u^t; \omega) \tag{3.8}$$

where D_u^t is the set of items that user u interacted with at time t, u_{te}^{t-1} is the previous time period time-evolving user factors, and ω is the network parameter. Notice that the input and output of the RNN are both latent variables instead of observations. We use SGD to update the parameter of RNN:

$$\omega = \omega - \gamma \nabla_\omega (\mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega}] + \frac{\lambda}{2}||\omega||_2^2) \tag{3.9}$$

where γ is the step size. As shown in the time-evolving module of Figure 2, the vector representation of a hidden layer u_{te}^t is a time-evolving factor of user u at period t and helps to propagate influence from the previous period to the next period [94]. The updates of time-specific user factors through meta-learning and time-evolving user factors through nested RNN are summarized in Algorithm 4. The recommendation process is summarized in Algorithm 2.

Joint Item Embedding Optimization. Let $\mathcal{L}_{emb}$ denote a differentiable loss function used to train the embedding matrix E. And let $\mathcal{G}$ denote the decoding module, which is followed by attribute-wise sigmoid transformation:

$$\begin{aligned} d_i = \mathcal{G}(Ez_i), \quad [\hat{z}_i]_j &= \text{sigmoid}(\eta_j^T d_i) \\ &= \frac{1}{1+\exp(-\eta_j^T d_i)} \quad \forall j \in \{1, ..., K\} \end{aligned} \tag{3.10}$$

where z_i is the original item representation in a binary vector, K is the length of z_i, E is the embedding matrix, d_i is the decoded item representation, η is the parameter for attribute-wise Sigmoid transformation, and $\hat{z}_i$ is the recovered item representation. The loss function for learning the embedding matrix is a negative log-likelihood and is represented as:

$$\mathcal{L}_{emb} = -\sum_{i \in I} \sum_{j} [z_i]_j \log [\hat{z}_i]_j \tag{3.11}$$

Other designs for item embedding with a differentiable loss function can also be applied.

To jointly train the embedding matrix, time-specific and time-evolving modules, we combine those losses, and optimize the total loss with respect to θ^t, ω and E.

$$\arg\min_{\theta^t,\omega,E} \sum_{\mathcal{T}_u^t \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega,E}] + \xi \mathcal{L}_{emb} + \frac{\lambda}{2}(||\theta^t||_2^2 + ||\omega||_2^2)\theta_u^t = \theta^t - \alpha_{\theta^t} \mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta^t,\omega,E}] \tag{3.12}$$

where ξ is the weight to be tuned. If ξ is set to a very large value, matrix E is determined only by $\mathcal{L}_{emb}$.

Note that the encoded item embedding is used for both modules. By fixing θ_u^t and ω, it is possible to back-propagate and calculate the gradient with respect to E, which is updated in each task as

$$E = E - \gamma \nabla_E (\mathcal{L}_{\mathcal{T}_u^t}[f^t_{\theta_u^t,\omega,E}] + \mathcal{L}_{emb}) \tag{3.13}$$

Also notice that $\mathcal{L}_{emb}$ is not dependent on θ^t and ω. Therefore, when embedding matrix is fixed, the loss function reduces to Eq (4.22), and θ^t and ω are updated without considering $\mathcal{L}_{emb}$.

3.4 Experiments

We conduct experiments on two movie datasets: *Netflix* and *MovieLens-1M* that consist of users' ratings of movies as explicit feedback and one music dataset: *Last.fm* that consists of users' play counts of music tracks as implicit feedback. Besides reporting the overall recommendation performance and comparing with state-of-the-art baselines, we also investigate key properties of the model, including: (1) each module's performance when used in isolation, (2) impact of varying time period lengths, (3) recommendation performance with no interactions in the current period, and (4) impact of hyper-parameters in the model.

Settings. We initialize both meta-learning and recurrent models with random initial values. Model learning rates are set through a grid search, and the Adam optimizer is applied with L2-regularization. In a

Algorithm 1 Model training

```
Require: Set of time periods: $\mathcal{H}$
Require: Hyperparameters: $\alpha, \beta, \gamma$
 1: for $t \in \mathcal{H}$ do
 2:     Initialize meta learner, $\theta^t$
 3:     while not converge do
 4:         Sample tasks $\mathcal{T}_u^t \sim p(\mathcal{T})$
 5:         for all $\mathcal{T}_u^t$ do
 6:             Sample support set $\mathcal{S}_u^t$ for local update
 7:             Perform local update with $\mathcal{S}_u^t$ for time-specific module using Equation (4.23)
 8:             Sample query set $\mathcal{Q}_u^t$ for meta update
 9:             Update time-evolving module with $\mathcal{D}_u^{t-1}$ using Equation (3.9)
10:         end for
11:         Perform meta update with $\mathcal{Q}_u^t$ for time-specific module using Equation (4.24)
12:     end while
13: end for
```

Algorithm 2 Recommendation for time-specific cold-start users

```
Require: Trained meta parameter $\theta^t$, RNN parameter $\omega$, recommendation time period $t$
 1: Identify cold-start user set for $t$
 2: for each user $u$ in the set do
 3:     Form support set $\mathcal{S}_u^t$ from current interactions
 4:     Perform local update with $\mathcal{S}_u^t$ for time-specific module using Equation (4.23)
 5:     Compute user factors using Equations (3.5) and (3.8)
 6:     Make recommendation using Equation (4.21)
 7: end for
```

dynamic meta-learning setting, the recurrent module takes historical interactions up to $t-1$ time period as an input while predicting for the next time period t, but the meta-learning module takes few recent interactions from the current time as a support set (e.g., in our setting, we have three months of period, so it takes K-shot interactions from the first month of the t time period and remaining interactions of two months as query set). We split users into meta-train and meta-test sets. To make the problem more challenging, we consider time-sensitive users with few interactions in the current period as test users. We set the few-shot ($K = 5$) to select the limited interactions for the support set.

For non-meta learning methods, the meta-train and meta-test split does not apply. To make a fair comparison, we first collect user interactions from period 1 to $t-1$ and then take the first few interactions (K) from the current time period t to create the training set. The remaining interactions from time period t are used for the test set. This mimics the few-shot problem for the current time period, and then a few interactions are available for the model to learn the user factors.

Different from explicit ratings that are constrained in a range, some implicit counts could take very large values. As a result, most of the models are not specifically designed for implicit data. To address this issue, we take a logarithm transformation on the Last.fm dataset to make the observed entries more balanced. Since DPF is specifically designed for counts data (only compared on Last.fm), we do not take logarithm transformation during training, but take such transformation on prediction and observation when calculating RMSE to make a fair comparison.

Datasets. The Netflix dataset has around 100 million interactions, 480,000 users, and nearly 18,000 movies rated between 1998 to 2005. We preprocessed the dataset similar to [49], which consists of user-item interactions from 01/2002 to 12/2005. Movie attributes are taken from the IMDB website, in which we considered genres, directors, actors, movie descriptions, and overall movie ratings as the key movie attributes. The MovieLens-1M dataset includes 1M explicit feedback (i.e., ratings) made by 6,040 anonymous users on 3,900 distinct movies from 04/2000 to 02/2003. This dataset has very dense ratings in a particular time range. Thus we split datasets in a period of 6-months and got a total of 6 periods in contrast to the other two datasets, which have 16 periods considering a period of 3 months. We use the same preprocessing for this dataset as we did for the Netflix dataset. The Last.fm dataset is created by crawling the user interactions and track details from the Last.fm database. This dataset includes 12,902 unique tracks and 548 users from 01/2012 to 12/2015. Tracks are described with artists, tags, and summary information.

Methods for Comparison. For comparison, we include matrix factorization based static and dynamic models, deep learning based models, graph models, sequential models, and meta-learning models:

- *Matrix factorization (MF):* The standard MF model SVD++ [44] that also exploits both explicit and implicit feedback is used here as a static baseline.
- *Dynamic models:* We use timeSVD++ [45], collaborative Kalman Filter (CKF) [27], and dynamic Poisson factorization (DPF) [11] as the time-evolving models.
- *Deep learning models:* We use Wide and deep [13], DeepFM [28] as static, and DIEN [96] as a dynamic models for deep learning-based recommendation. However, most of them are developed for click-through rate prediction in their original forms.
- *Graph-based model:* Most graph-based models are designed for static settings. For comparison,

we use graph convolutional matrix completion (GC-MC) [9], which models recommendation as link prediction in the graph, and neural graph collaborative filtering (NGCF) [84] that utilizes embedding propagation over user-item graphs.

- *Sequential model:* We use Sequential Recommendation via Convolutional Sequence Embedding (Caser) [78], which models recommendation as a unified and flexible structure to capture both preferences and sequential patterns, and transformer-based sequential recommendation model (SASRec) [39] in our comparison.
- *Meta-learning models:* We follow the model-agnostic meta-learning model (MAML) to implement the meta-learning model similar to MeLU [47]. We also compared with the meta-learning model in [81] that focuses on item cold-start problem (referred to as ML-ICS). The model is also designed for the classification setting, so we have to make adjustments to fit into our context.

Evaluation Metrics. For evaluation, we analyze the experimental results in terms of both the deviation of predicted values from the ground truth and the errors of the ranking sequences. We use Root Mean Squared Error (RMSE) and Normalized Discounted Cumulative Gain (NDCG) averaged across all test users. RMSE is usually reported for explicit data, while NDCG is usually reported for implicit data:

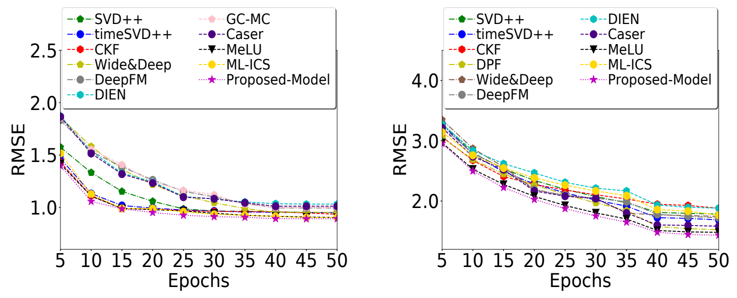
$$\begin{aligned} \text{RMSE} &= \sqrt{\sum_{r_{u,i}\in O} (\hat{r}_{u,i} - r_{u,i})^2/|O|}, \\ \text{NDCG}_u &= \sum_n \frac{rel_n^{pred}}{\log_2(1+n)} / \sum_n \frac{rel_n^{ideal}}{\log_2(1+n)} \end{aligned} \tag{3.14}$$

where O is the observation set for the test set and rel_n is the relevancy of n^{th} item in the ranking sequence for user u, which is binary for implicit data or the rating for explicit data. To penalize the negative feedback, we linearly mapped the ratings to a range of [-1,1]. The NDCG is the fraction of Discounted Cumulative Gain (DCG) of recommendation result over the ideal DCG.

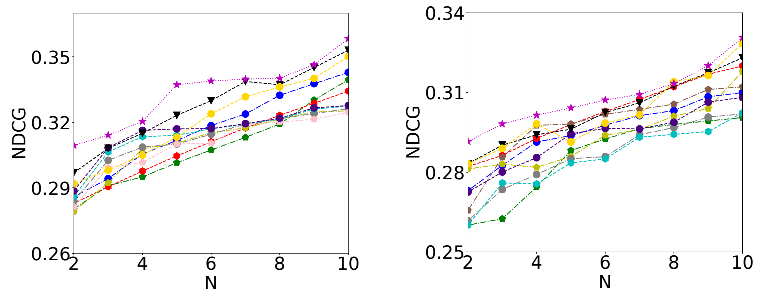
3.4.1 Recommendation Performance

The experimental results are shown in Figure 6.2. We evaluate NDCG based on the top N recommendation list and RMSE based on the training epochs. The RMSE is stable after 30-40 epochs in all datasets.

The average results of NDCG and RMSE considering all periods with the range of deviation are shown in Table 3.2, for the Netflix, Last.fm, and MovieLens datasets, respectively. The proposed model clearly demonstrates the advantage of combining time-specific and time-evolving user factors that lead to a superior recommendation accuracy as compared with other competitive models. Both explicit and implicit

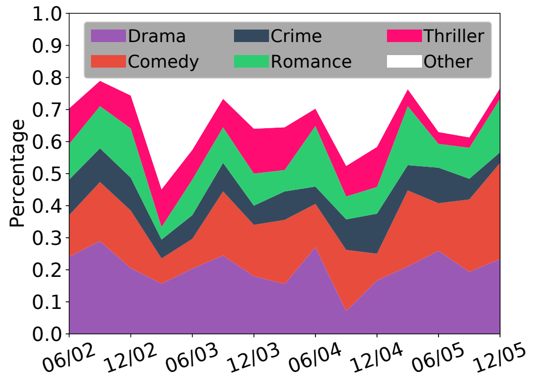


(a) RMSE

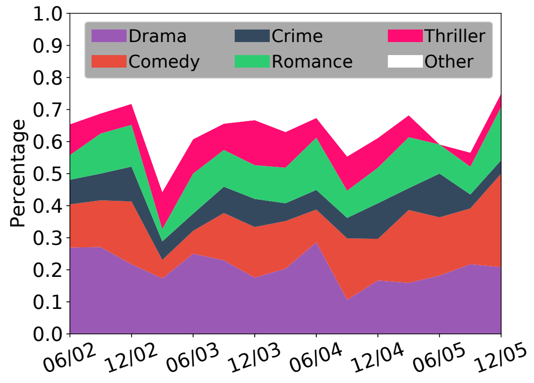


(b) NDCG

Figure 3.3: (a) RMSE based on the training epochs and (b) NDCG based on the top N recommendations for Netflix, Last.fm, and Movielens-1M datasets respectively.



(a) Ground Truth



(b) Recommended

Figure 3.4: Dynamic trend of movie genres in Netflix: ground truth (a) vs model recommended (b)

datasets are highly sparse, and MF models' performance is poor due to the sparse interactions. Also, MF models suffer from cold-start problems, and thus their performances are fairly limited, as shown in Table 3.2.

Category	**Model**	**Netflix**		**Last.fm**		**MovieLens-1M**	
		RMSE	**NDCG**	**RMSE**	**NDCG**	**RMSE**	**NDCG**
MF	SVD++	0.9797±0.03	0.2915	1.7829±0.08	0.2882	1.0825±0.04	0.3023
Dynamic	timeSVD++	0.9538±0.06	0.3115	1.6912±0.11	0.2962	1.0483±0.03	0.3224
	CKF	0.9337±0.04	0.3130	1.5316±0.32	0.3018	1.0652±0.04	0.3151
	DPF	N/A	N/A	1.5227±0.43	0.3085	N/A	N/A
Deep Learning	Wide and Deep	0.9904±0.04	0.2864	1.7253±0.22	0.2727	1.1364±0.06	0.2932
	DeepFM	0.9811±0.03	0.2930	1.6815±0.21	0.2971	1.1723±0.05	0.2882
	DIEN	1.0345±0.04	0.2832	1.9225±0.26	0.2714	1.1872±0.14	0.2843
Graph	GC-MC	1.0760±0.03	0.2901	N/A	N/A	1.1704±0.08	0.2913
	NGCF	1.0321±0.03	0.3026	1.5612±0.23	0.2896	1.1216±0.05	0.3103
Sequential	Caser	1.0124±0.03	0.3101	1.5824±0.31	0.2931	1.1339±0.08	0.3012
	SASRec	N/A	0.3246	N/A	0.3103	N/A	0.3238
Meta-Learning	MeLU	0.9213±0.05	0.3232	1.2580±0.28	0.3122	1.0685±0.08	0.3214
	ML-ICS	0.9332±0.04	0.3173	1.2408±0.24	0.3142	1.0845±0.06	0.3244
Proposed	**Ours**	**0.8925±0.03**	**0.3472**	**1.2203±0.16**	**0.3385**	**0.9945±0.08**	**0.3351**

Table 3.2: Recommendation Results (RMSE and NDCG)

Similarly, deep learning models require sufficient training data and hence largely suffer in the few-shot recommendation setting. Moreover, these models might need extra side information, like user profile and item details, for better recommendations. For example, DIEN needs cleverly chosen interest features like user behavior, and the absence of those features limits its performance, as shown in Table 3.2. Similarly, the poor performance of graph-based models in both movie datasets implies that these methods are insufficient to handle cold-start problems. Like other existing models, the performance of a sequential model is less effective for the cold-start users in all three datasets. The reason could be that the model is less effective in capturing long-term user preferences. In contrast, the meta-learning approaches show better results by leveraging shared knowledge across the users. However, in the time-specific cold-start setting, test users have very limited interactions. In particular, the meta-learning model doesn't benefit from time-evolving aspects of the user interests, and thus underperforms the proposed model.

We use an illustrative example to further demonstrate how the proposed model effectively captures the underlying user interest and its evolution in Figure 3.4. The recommended movie genres are compared to the user's favorite genres based on the provided true ratings. The result shows that the recommendation matches user's changing taste over time well. It is also interesting to see that the proposed model accurately detects some dramatic changes in user's ratings (e.g., from 12/02 to 06/03 and from 06/04 to 12/04), which were likely to be caused by some time-specific factors.

We further present an example to show how the meta-learning module effectively captures time-specific factors in the form of popular trends in a specific period from the global user space and transfers the (meta) knowledge to the cold-start users with very limited interactions. Table 3.3 demonstrates top-5 time-specific (period 4) popular movies learned by the meta-learning module, which shares that knowledge with the test users (i.e., two users are shown in Table 3.3 and some time-specific popular movies like ’Best in Show’ are recommended to them). This example demonstrates how our model provides effective recommendations to users by capturing time-specific factors.

Users	**Movies**
Training users	[’Best in Show’, ’Chicken Run’, ’Sommersby’, ’Bedrooms and Hallways’, ’The Mod Squad’]
Test user (ID:5636)	[’Mr. Mom’, ’Best in Show’, ’Shower’, ”We’re No Angels”, ’Groundhog Day’]
Test user (ID:5539)	[’Best in Show’, ’An Ideal Husband’, ’Life Is Beautiful’, ’Breaking Away’, ’Kramer vs. Kramer’]

Table 3.3: Time-specific popular movies learned and predicted by the meta-learning module

Performance of Individual Modules. The proposed model integrates two modules to capture time-specific and time-evolving latent factors. We study their contribution in detail to show how the user’s time-specific interest and time-evolving preferences affect the overall recommendation performance.

- **Time-specific Meta-Learning (TS-ML) Module.** This module is specifically designed to capture users’ time-specific interest in each period. Different from [47], it only relies on time-specific data so that the proposed model is more generally applicable.
- **Time-evolving RNN (TE-RNN) Module.** This module is designed to capture users’ gradual shift of interest by utilizing historical interactions.

Table 3.4 reports the recommendation performances of each module and compares them with the proposed integrated model. First, each module performs reasonably well, and the recommendation results are comparable with some of the state-of-the-art baselines. It is also interesting to see that the meta-learning model outperforms the time-evolving module in all cases. This demonstrates the stronger impact of the time-specific factors when making recommendations to users. It also justifies the proposed meta-learning module’s effectiveness that successfully captures these latent factors by learning the global trend from other users during a

Table 3.4: Comparison of recommendation performance (Average RMSE and NDCG) using each module alone and the proposed integrated model for all three datasets

Dataset	**Model**	**RMSE**	**NDCG**
Netflix	TS-ML	0.9380±0.02	0.3103
	TE-RNN	0.9478±0.03	0.3011
	Proposed	**0.8925±0.03**	**0.3472**
Last.fm	TS-ML	1.2791±0.12	0.3073
	TE-RNN	1.9938±0.34	0.2910
	Proposed	**1.2203±0.16**	**0.3385**
MovieLens-1M	TS-ML	1.0935±0.07	0.3144
	TE-RNN	1.2505±0.13	0.3162
	Proposed	**0.9945±0.08**	**0.3351**

specific period. Finally, the proposed model that integrates both modules achieves the best recommendation performance, because it can capture both the time-evolving and time-specific factors.

We further provide an illustrative example from the MovieLens-1M dataset for a user with ID:3462 to show each module's contribution to the final prediction. Figure 3.5 shows that final predicted ratings are the combination of both modules, and they effectively complement each other to provide better final performance.

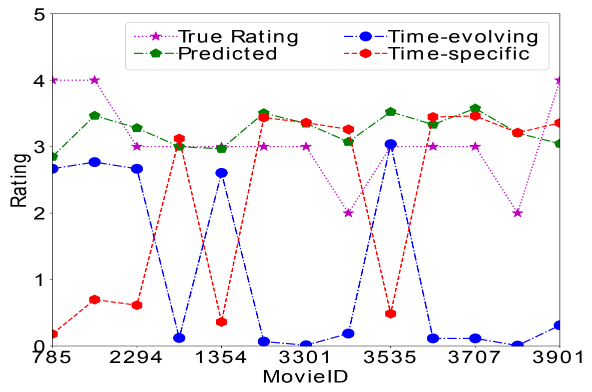


Figure 3.5: Contribution of time-evolving and time-specific modules to the recommendation of different movies, where they jointly contribute to the predicted ratings.

Varied Recommendation Period Lengths. In this paragraph, we show a more fine-grained temporal analysis by varying the period length. We use three different period lengths: 1 month, 3 months, and 6 months, in the Netflix dataset and report the model performance. We also select one representative from each

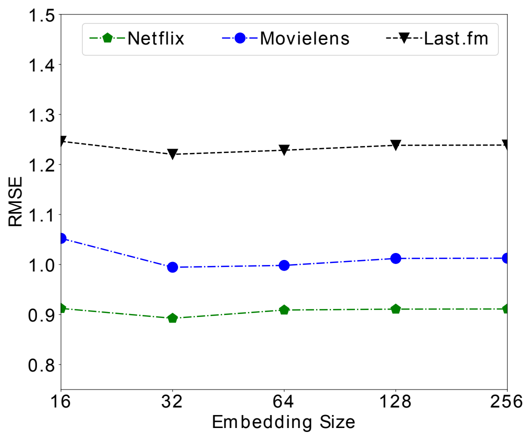
Table 3.5: RMSE and NDCG in varying period lengths in months

Model	**1**		**3**		**6**	
	RMSE	**NDCG**	**RMSE**	**NDCG**	**RMSE**	**NDCG**
SVD++	1.0923±0.04	0.2892	0.9797±0.03	0.2915	0.9730±0.05	0.2901
timeSVD++	1.0742±0.04	0.3086	0.9538±0.06	0.3125	0.9641±0.04	0.31142
deepFM	1.1260±0.09	0.2817	0.9811±0.04	0.2930	0.9585±0.06	0.2891
MeLU	1.0037±0.05	0.3138	0.9213±0.05	0.3232	0.9414±0.04	0.3204
Proposed	**0.9864±0.06**	**0.32070**	**0.8925±0.03**	**0.3472**	**0.9226±0.03**	**0.3317**

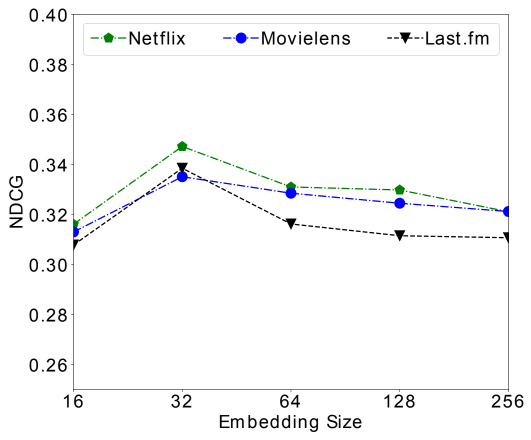
category of baseline models and report the RMSE and NDCG in Table 3.5. The user base and the number of interactions per user are limited when we set the length to 1 month. Due to this, matrix factorization methods and deep learning methods suffer largely and have poor performance compared to the proposed model. In the case of 6 months, due to more interactions, deep learning models are performing better than other smaller period lengths. Also, we perform $K = 5$-shots for meta-learning, and due to large interactions present in a 6-month period length, it is quite possible not to get those shots from the most recent interactions. This could hurt the performance of meta-learning, which achieves slightly lower performance than the 3-month period. Overall, in all period lengths, our model outperforms others with a noticeable margin.

No Interactions in the Current Period. We further study the model performance where a user doesn't have any interaction in the current period. In this case, our time-specific module utilizes global user factors instead of user-specific factors due to lack of interaction, and hence no adaptation is made. Similarly, the time-evolving module just forwards its previous evolving user factors to the next time period to provide an integrated recommendation.

We provide an illustrative example, where we randomly choose five test users with user ids: $\{$870391, 1197396, 757756, 920368, 1918714$\}$ and remove their interactions in a given period (i.e., period = 4). We compare the prediction performance with the regular model where these interactions are not removed to show the impact of completely missing interactions in the current period. Table 3.6 shows that the proposed model provides quite robust results. In Period 4, due to missing interactions, the meta-learning module only utilizes global user factors. Similarly, for Period 5, the time-evolving module doesn't have inputs in Period 4 and hence forwards the user's time-evolving factors from Period 3. The performance with no interactions is slightly worse than having interactions.



(a) RMSE



(b) NDCG

Figure 3.6: Impact of item embedding size

Table 3.6: Results with/without interactions in current period

Period	**Interactions**		**No Interactions**	
	RMSE	**NDCG**	**RMSE**	**NDCG**
4	0.8720±0.04	0.3382	0.8953±0.03	0.3226
5	0.8466±0.04	0.3448	0.8569±0.04	0.3415

Hyperparameter Tuning. We perform fine-tuning of model learning rates through a grid search from {1e-5, 1e-4, 1e-3, 1e-2, 1e-1} and select best value as $\alpha = \beta = \gamma = 1e^{-4}$. In our implementation, we learn user embedding from the model but item embedding is generated from the corresponding attributes. We embed each attribute (or feature) of an item in a vector of size 32. Each item in movies datasets has 5 attributes (genre, director, actors, plot, and overall rating), which are concatenated to generate the final item embedding of size 160. We applied different vector sizes [16,32,64,128,256] for each attribute and at 32 we got the optimal performance. Increasing the size didn't improve the results as shown in Figure 4.5 for three datasets. So, we choose 32 for each attribute and the final embedding size is set as 160.

Time Complexity. Denote the computational complexity of embedding, time-specific and time-evolving module for one round of backpropagation as $O(m_1)$, $O(m_2)$, $O(m_3)$. In one epoch, the model iterates through T tasks, and in each task, the model performs a local update for the time-specific module on the support set of size S and also updates the time-evolving module on both support S and query set Q. After iterating through the tasks, a meta update is performed utilizing query set Q on the time-specific module. Overall, the total time complexity is $O(m_1 + T(m_2 + m_3)(S + Q))$ for each iteration. We also provide the actual time taken by the proposed model to complete one training iteration utilizing GeForce RTX 2070 GPU. The proposed model considers each user as a task where the time-specific module and time-evolving module take 0.0065s and 0.0023s, respectively, to train on one task. Considering Movielens-1M dataset,

Table 3.7: Netflix Periodic RMSE Results

Period	SVD++	timeSVD++	CKF	MeLU	DeepFM	Wide & Deep	GC-MC	Caser	DIEN	ML-ICS	Proposed
1	0.9813	0.9840	0.9552	0.9201	1.2706	1.3178	1.2912	1.2845	1.3102	0.9335	0.9205
2	0.9862	0.9895	0.9683	0.9481	1.1117	1.1321	1.2903	1.2882	1.3033	0.9498	0.9258
3	0.9795	0.9795	0.9693	0.9440	1.0489	1.0921	1.2732	1.2634	1.2724	0.9445	0.9479
4	0.9742	0.9708	0.9524	0.9374	1.0896	1.0777	1.2613	1.2404	1.2543	0.9401	0.8771
5	0.9800	0.9710	0.9432	0.9415	1.0630	1.0115	1.2311	1.2127	1.2167	0.9426	0.9279
6	0.9757	0.9740	0.9564	0.9495	0.9911	0.9904	1.1374	1.1206	1.1515	0.9487	0.9533
7	0.9744	0.9725	0.9612	0.9506	1.0323	0.9895	1.1068	1.1745	1.0984	0.9622	0.8723
8	0.9766	0.9661	0.9526	0.9418	0.9706	1.0122	1.049	1.1132	1.0401	0.9517	0.8789
9	0.9737	0.9603	0.9412	0.9424	0.9902	0.9897	1.0442	1.0724	1.0311	0.9503	0.8786
10	0.9726	0.9597	0.9228	0.9391	0.9961	0.9655	1.0918	1.0843	1.0511	0.9430	0.9209
11	0.9722	0.9595	0.9332	0.9360	0.9497	1.0570	1.03403	1.0563	1.0615	0.9315	0.8997
12	0.9731	0.9511	0.9541	0.9438	0.9761	0.9648	1.0885	1.0245	1.0528	0.9396	0.9427
13	0.9361	0.9556	0.9243	0.9351	0.9710	0.9540	1.0451	1.0373	1.0417	0.9306	0.8746
14	0.9658	0.9431	0.9416	0.9457	0.9720	0.9635	1.031	1.0143	1.0531	0.9487	0.8824
15	0.9673	0.9457	0.9358	0.9378	1.0050	0.9598	1.0162	1.0142	1.0237	0.9441	0.8812
16	0.9670	0.9340	0.9337	0.9428	0.9866	0.9550	1.0254	1.0078	1.0145	0.9468	0.8340

we used 80% training users and hence total training time for both modules in each iteration are 31.40s and 11.11s. Similarly, the embedding module takes 3.14s. This gives the total time of 41.65s/iteration.

3.5 Periodical Recommendation Results

We report the detailed result for each prediction period for all three datasets to demonstrate how the predictions evolve over time.

Netflix. As can be seen from Table A.1, in most periods, the proposed model performs better than others except for a few periods like 3 and 6, where the proposed model slightly under-performs the meta-learning model. A possible explanation is that in these periods, time-specific user interest might largely deviate from the time-evolving user factors, and hence their combined recommendation is less accurate.

Last.fm Datasets. The period-wise results for the Last.fm dataset on one run is shown below in Table A.2. The proposed model achieves good results in this implicit feedback (i.e., counts) dataset. The high variation of the counts indicates users' music listening habits are fluctuating significantly. The results in Table A.2 show that matrix factorization and deep learning baseline models are less effective in capturing those variations. In contrast, the proposed model simultaneously captures those variations in the form of users' specific biases and the gradual shift of preferences effectively.

Table 3.8: Last.fm periodic RMSE results

Period	SVD++	timeSVD++	CKF	DPF	MeLU	DeepFM	Wide & Deep	Caser	DIEN	ML-ICS	Proposed
1	1.6507	1.6920	1.8353	1.6953	1.3153	2.1434	2.1993	1.8132	2.4720	1.3233	1.3173
2	2.3052	2.1647	1.3964	1.3238	1.5567	2.1336	2.1675	1.3854	2.1291	1.5407	1.5320
3	2.1972	1.9160	1.7007	2.0814	1.3408	2.0674	2.1194	1.6772	2.1065	1.3531	1.3465
4	2.1574	1.7385	1.5462	1.4654	1.1346	2.0757	2.0464	1.5232	1.9520	1.1287	1.1047
5	1.5858	1.6520	1.6602	1.5214	1.1687	2.0202	2.0033	1.5843	1.9876	1.1732	1.1578
6	1.7879	1.8460	1.5555	1.3815	1.2602	2.0763	2.1651	1.5278	1.8816	1.2821	1.0809
7	1.5348	1.6498	1.6527	1.3448	1.6359	1.9372	1.9595	1.6227	1.8392	1.6324	1.6226
8	1.7527	1.7350	1.5798	1.3617	1.5987	1.8883	1.8839	1.5482	1.8806	1.6037	1.4270
9	2.2419	1.9977	1.4311	1.5085	1.4493	1.9546	2.0015	1.4326	1.9011	1.4488	1.3338
10	1.9101	1.7901	1.4572	2.3051	1.1753	1.8346	1.8766	1.4104	1.8743	1.1714	1.0368
11	1.5709	1.4704	1.3067	1.4337	1.1528	1.7557	1.8178	1.2974	1.8515	1.1553	1.0864
12	1.5864	1.5760	1.3353	1.2409	1.2226	1.7666	1.7346	1.3136	1.8483	1.2105	1.0933
13	1.5228	1.3637	1.4613	1.0546	1.0546	1.7518	1.7192	1.4463	1.8617	1.0720	1.0123
14	1.7330	1.5966	1.6130	1.2670	1.0809	1.6417	1.6800	1.5812	1.8445	1.0912	1.0603
15	1.9891	1.6644	1.4429	1.3264	1.1094	1.6815	1.7253	1.4293	1.7623	1.0996	1.0824
16	1.4638	1.4106	1.4334	1.2547	1.3206	1.6900	1.9963	1.4017	1.7456	1.3322	1.0688

Table 3.9: Movielens periodic RMSE results

Period	SVD++	timeSVD++	CKF	MeLU	DeepFM	Wide & Deep	GC-MC	Caser	DIEN	ML-ICS	Proposed
1	1.0615	1.0496	1.1155	1.2234	1.5341	1.3719	2.3345	2.2941	2.3438	1.2412	1.2253
2	0.9954	0.9932	0.9847	1.1613	1.3659	1.2686	1.7912	1.7247	1.7065	1.1803	1.0444
3	1.0445	0.9982	1.0305	0.9341	1.2267	1.2585	1.2576	1.2289	1.2019	0.9423	0.8487
4	1.1499	1.1189	1.0508	0.9776	1.2492	1.2346	1.1332	1.1223	1.1623	0.9711	0.9084
5	1.0918	1.0611	1.1468	1.0308	1.1615	1.1052	1.1346	1.1286	1.1587	1.0514	0.9053
6	1.0773	1.0688	1.0699	1.1377	1.0994	1.0672	1.1112	1.1021	1.1421	1.1127	1.0377

Movielens Datasets: Periodic results for the Movielens dataset are shown in Table A.3. In the first period, we notice that meta-learning models are not performing well, whereas SVD models are performing well. This is because the dataset has very dense interactions in the first period. Meta-learning models only use k-shot for learning, but SVD models benefit from maximum interactions. For the proposed model, time-evolving user factors don't contribute in the first period. Also, time-specific factors are based on meta-learning. Hence, its performance is not better than the baselines, but the proposed model achieves a better performance in the subsequent periods

3.6 Conclusion

In this chapter, we formulate a novel time-sensitive cold-start problem and present a dynamic recommendation framework to address its unique challenges. The framework integrates a time-sensitive meta-learning module with a time-evolving recurrent module. The former handles the user cold-start problem by learning global knowledge among users from their interaction information in the current recommendation period. This module is jointly optimized with the time-evolving recurrent module that captures a user's gradually shifted preferences. A merged user representation is generated using the two modules' outputs and interacts with the item embedding to provide the final recommendations.

Chapter 4

Meta Evidential Learning for Sparse User Interactions

In this chapter, we focus to deal with the problem of noisy sparse interaction and provide a systematic learning method extending L2L framework with evidential learning that detects true user intent by quantifying the uncertainty. The proposed method integrates meta learning module with an evidential learning module in which the former module deals with sparse interactions and the latter module leverages evidence to quantify the uncertainty that is instrumental to address the noisy interactions problem. To show the model's effectiveness, we leverage meta evidential learning method into recommender systems as one application throughout the chapter to tackle the issues of sparse and noisy interactions.

4.1 Meta Evidential Learning for Cold-start Recommendation

Recommender systems exploit data mining techniques and prediction algorithms to predict users' interest in products, services, and information among a large number of available items [2]. Commonly used approaches can be generally categorized as collaborative filtering-based, content-based, and hybrid systems. Collaborative filtering methods recommend items to the target users based on the similar taste of existing users [25]. These methods mostly suffer from data sparsity that leads to the cold-start problems (i.e., not able to handle new users and/or items with limited interactions). Content-based methods [56] address this issue by utilizing users' demographic information (e.g., age, gender, and nationality) and item content (e.g., genres, directors, and actors). While various extra-information is available for the items, acquiring users' personalized information is usually difficult due to privacy issues. Hybrid models combine the benefits of

Table 4.1: Epistemic uncertainty and RMSE loss for two distinct users from Movielens 1M dataset based on the number of interactions.

UserID	**Interactions**	**Epistemic**	**RMSE**
4515	24	0.5133	1.0345
4575	164	0.6812	1.7038

both collaborative and content-based systems but remain less effective to the cold-start users/items.

Several recent works have attempted to address the cold-start problem in recommender system through meta-learning [47, 81]. In particular, meta-learning models the cold-start recommendation as a few-short learning problem. By arranging existing users' item-consumption history as the training tasks, it learns a global meta-model that can adapt to users/items with limited interactions with improved recommendation accuracy. Most existing methods, including meta-learning models, use the number of interactions as the primary factor to identify the cold-start users. However, they ignore the nature of the interactions as not all the interactions are equally important for a recommender system to construct an accurate (latent) profile for users to provide effective recommendations. As certain interactions can bring much higher value to the system than others, it is essential to consider both the *number* and the *value* of the interactions to most properly handle cold-start recommendations.

Table 4.1 shows two examples users from the Movielens dataset with significantly different numbers of interactions. As can be seen, the second user is much more active than the first user who has much fewer interactions and may be regarded as cold-start. However, more interactions may not necessarily lead to a more accurate recommendation result, which is evidenced by a higher root mean squared error (RMSE) for the first user. In fact, the larger recommendation error is also reflected by a higher model (or epistemic) uncertainty. This example, along with more illustrative examples provided in our experiment section, helps to further confirm the distinct values of different interactions. It also implies the important role of using *uncertainty* to quantify the model confidence when making a recommendation that could indicate the cold-start level of a user (i.e., how well the system knows the user).

In general, a recommender system's prediction is very sensitive to observed user-item interactions, especially when they are limited. Hence, a precise and calibrated uncertainty estimation is useful for interpreting the model confidence in cold-start recommendations. There are two common types of uncertainty: *aleatoric* that captures the uncertainty introduced by the noises in the data and *epistemic* that captures the model uncertainty due to lack of understanding of the data [41]. Aleatoric uncertainty is usually irreducible and can be directly estimated from data. Bayesian models offer a natural way to capture model uncertainty, and hence Bayesian neural networks have been commonly used to estimate the epistemic uncertainty of deep learning (DL) models.

However, Bayesian DL models usually conduct posterior inference through Monte Carlo (MC) sampling, which poses a very high computational cost due to a large number of parameters in a DL model and their complex dependencies. Consequently, directly extending the current deep learning-based recommend systems through Bayesian modeling will prevent them from scaling to a large user-item space.

To address the above key challenges, we propose a meta evidential learning model, referred to as *MetaEDL*, to provide uncertainty-aware cold-start recommendations. By integrating a meta-learning module with evidential learning, MetaEDL is able to leverage all existing users' historical interactions to learn a global model that can easily and accurately adapt to cold-start users with limited interactions. Furthermore, we construct a hierarchical model that provides a generative process to model the likelihood of the user-item interactions. Instead of performing an expensive posterior inference, evidential learning is adopted to directly predict the hyper-parameters of the prior distributions of the parameters in the likelihood function, using a non-Bayesian deep neural network. These predicted hyperparameters have a natural interpretation as *pseudo counts*, which can serve as evidence to quantify the model confidence for its recommendations.

The main contributions of this work are four-fold:

- A novel recommendation model that integrates meta-learning and evidential learning to provide uncertainty-aware cold-start recommendations.
- Posterior inference through evidential learning to ensure good efficiency that allows a recommender system to scale to a large user-item space.
- Using pseudo count based evidence that provides a deeper insight to understand the value of different interactions that is instrumental to identify truly cold-start users, going beyond just using the interaction count.
- An integrated end-to-end training process that optimizes the embeddings and meta evidential learning modules.

We conduct extensive experiments over four real-world datasets and compare with state-of-the-art models to demonstrate the effectiveness of the proposed *MetaEDL* model.

4.2 Related Work

Section 2.1.2 of chapter 2 has included most of the related work for this work and here we specifically mention related work for only uncertainty-aware recommendation.

Uncertainty in Recommender Systems. All the above methods primarily focus on personalized recommendation but lack in handling uncertainty. Recently, Gaussian embedding-based recommendation [36] attempts to capture user and item uncertainty but does not measure model uncertainty. Our proposed method not only provides an effective recommendation but also measures both data and model uncertainty utilizing the evidential learning approach [3].

4.3 Problem Formulation

For a recommendation model, input data is represented as $\{U,I\}$, where U is the user set and I is the item set. Table C.1 summarizes the major notations used throughout the sections. We perform recommendation and uncertainty quantification for each user with a recommendation function as:

$$f_{\theta_u,E_u,E_i}(u,i) = \{\gamma_{(u,i)}, \nu_{(u,i)}, \alpha_{(u,i)}, \beta_{(u,i)}\} \; \forall u \in \mathcal{U}, i \in \mathcal{I} \tag{4.1}$$

where $\gamma_{(u,i)}$ is the recommended score for item i assigned by user u, $\nu_{(u,i)}, \alpha_{(u,i)}$,and $\beta_{(u,i)}$ are the model evidence (which will be detailed along with the meta evidential module) for user u on item i, θ_u is user specific model parameter; E_u, and E_i are user and item embedding module parameters. The goal of a recommender system is to predict the scores with confidence so that it can accurately capture a user's true preference on items in belief that the recommended items are likely to be adopted by the users.

We formulate recommendations as a few-shot regression problem in the meta-learning setting. Users are *dynamically* partitioned into *meta-train* and *meta-test* sets. The meta-train user set includes users with sufficient interactions, while the meta-test user set includes cold-start users who have only a few interactions. We consider a distribution over tasks $P(\mathcal{T})$, and each user is represented as a few-shot regression task $\mathcal{T}_u$ sampled from the given task distribution. In general, a task includes a *support set* $\mathcal{S}_u$ and a *query set* $\mathcal{Q}_u$. The support set includes a user's interactions where k is interpreted as the number of shots (i.e., interactions). The query set includes the rest interactions of this user.

$$\mathcal{T}_u \sim P(\mathcal{T}): \; S_u = \{(u,i_j), r_{(u,i_j)}\}_{j=1}^{N}, \quad Q_u = \{(u,i_j), r_{(u,i_j)}\}_{j=N+1}^{N+M} \tag{4.2}$$

where N is the number of items a user interacted, and $r_{(u,i_j)}$ represents label (i.e. rating or count) from user u to item i_j.

Table 4.2: Summary of Notations

Notation	Description
u, i	user and item
e_i, z_i	item i's one-hot encoding and embedding
e_u, z_u	user u's one-hot encoding and embedding
$r_{(u,i)}$	ground truth score for user u on item i
θ, θ_u	meta and user-specific parameters of meta-learning module
E_u, E_i	users and items embedding matrix
$\mathcal{S}_u$, $\mathcal{Q}_u$	support and query sets in task corresponding to user u
$\gamma_{(u,i)}$	model prediction for user u on item i
$\nu_{(u,i)}$,$\alpha_{(u,i)}$, $\beta_{(u,i)}$	model evidence for user u on item i as an hyperparmaters of an evidential distribution

We adopt episodic training [82], where the training task mimics the test task for the efficient meta-learning. The support set $\mathcal{S}$ in each episode works as the labeled training set on which the model is trained to minimize the loss over the query set $\mathcal{Q}$. This training process is iteratively carried out episode by episode until convergence.

4.4 Proposed Model

The proposed model consists of two major components: an embedding module and a meta evidential learning module, as shown in Figure 4.1. The embedding module generates user and item embeddings and is forwarded to the evidential meta-learning module, where prediction and model evidence are produced as a final output.

4.4.1 Embedding Module

We represent user u, and item i in one hot encoding considering unique user and item IDs: $e_u \in \mathcal{R}^n$ where n is the total number of users and $e_i \in \mathcal{R}^m$ where m is the total number of items respectively. This one hot vector is then transformed using the embedding matrix E_u for user and E_i for item in d-dimension: $z_u = E_u e_u$ and $z_i = E_i e_i$. The embedding matrix is optimized along with the model training process. We use gradient descent to update the both users and item embedding matrix:

$$\begin{aligned} E_u &= E_u - \xi \nabla_{E_u} (\mathcal{L}_{\mathcal{T}_u}[f_{\theta_u, E_u, E_i}]) \\ E_i &= E_i - \xi \nabla_{E_i} (\mathcal{L}_{\mathcal{T}_u}[f_{\theta_u, E_u, E_i}]) \end{aligned} \tag{4.3}$$

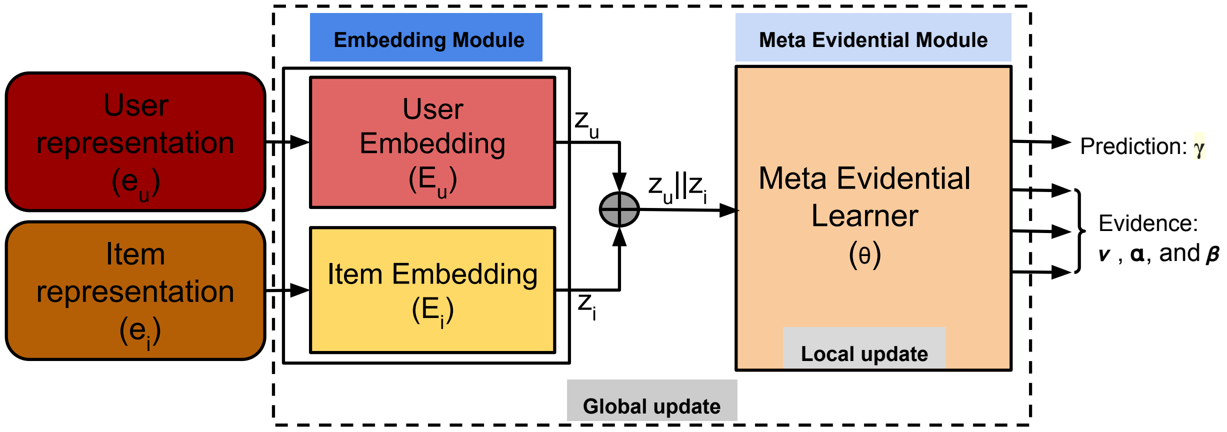


Figure 4.1: Overview of the proposed model.

where ξ is the step size, and $\mathcal{L}_{\mathcal{T}_u}[f_{\theta_u,E_u,E_i}]$ is an end-to-end meta evidential user-specific loss and detail is given in Equation (4.21).

4.4.2 Meta Evidential Learning Module

We formulate the meta-learning module as a non-Bayesian neural network to estimate a target interaction score and its associated evidence to learn both aleatoric and epistemic uncertainty. We accomplish this by placing evidential priors over the original Gaussian likelihood function and training the neural network to infer the hyperparameters of the evidential distribution as similar to [3]. The key intuition of employing evidential learning in recommender systems is that it allows us to assign evidence to the predicted interaction, where the evidence can be used to formulate the prediction score while capturing the model confidence.

A hierarchical model. The recommendation problem is set up in such a way that the target (rating and count), y_n, is drawn i.i.d. from a Gaussian distribution with unknown mean and variance (μ, σ^2). Model evidence can be introduced by further placing a prior distribution on (μ, σ^2), leading to a hierarchical model. To ensure conjugacy, we choose a Gaussian prior on the unknown mean and an Inverse-Gamma prior on the unknown variance:

$$p(y_n|\mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2) \tag{4.4}$$

$$p(\mu|\gamma, \sigma^2\nu^{-1}) = \mathcal{N}(\gamma, \sigma^2\nu^{-1}) \tag{4.5}$$

$$p(\sigma^2|\alpha, \beta) = \text{Inv-Gamma}(\alpha, \beta) \tag{4.6}$$

where $\text{Inv-Gamma}(z|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{z}\right)^{\alpha+1}\exp(-\frac{\beta}{z})$ with $\Gamma(.)$ being a gamma function; γ, ν, α, and β are parameters of the corresponding prior distributions.

Interpreting hyper-parameters. Besides serving as the parameters of the corresponding prior distributions in the hierarchical model, the hyper-parameters $(\gamma, \nu, \alpha, \beta)$ offer very intuitive meanings, which set the stage to use them in the proposed evidential learning. The best way to show this is to couple these prior distributions with a set of actual observations, $\mathbf{y} = (y_1, ..., y_N)^\top$. Given the Gaussian likelihood in (4.4), we compute joint posterior distribution $p(\mu, \sigma^2|\mathbf{y})$ factorized as $p(\mu|\mathbf{y}, \sigma^2)p(\sigma^2|\mathbf{y})$. We first derive the conditional posterior of μ:

$$p(\mu|\mathbf{y}, \sigma^2) = \mathcal{N}(\gamma_N, \sigma_N^2) \tag{4.7}$$

$$\gamma_N = \frac{\nu}{\nu + N}\gamma + \frac{1}{N+\nu}\sum_{n=1}^{N} y_n \tag{4.8}$$

$$\sigma_N^2 = \frac{\sigma^2}{\nu + N} = \frac{\sigma^2}{\nu_N} \tag{4.9}$$

where $\nu_N = \nu + N$. From (4.8), we can see that the posterior mean is the convex combination of the prior mean γ and the maximum likelihood estimation of the mean, given by $\frac{1}{N}\sum_{n=1}^{N} y_n$. Similarly, the variance in the posterior distribution is ν_N times smaller than the prior variance. As a result, ν can be interpreted as the 'effective' prior observations for the prior mean γ. We continue to derive the posterior of σ^2:

$$p(\sigma^2|\mathbf{y}) = \text{Inv-Gamma}(\alpha_N, \beta_N) \tag{4.10}$$

$$\alpha_N = \alpha + \frac{N}{2} \tag{4.11}$$

$$\beta_N = \beta + \frac{1}{2}\sum_{n=1}^{N}(y_n - \mu)^2 \tag{4.12}$$

First, (4.11) shows that after observing N data samples, the prior parameter α is increased by $\frac{N}{2}$ to reach the posterior parameter α_N. This has the effect of treating the prior hyper-parameter as 2α 'effective' prior observations of 'pseudo' data samples. Similarly, by multiplying both sides of (4.12) by 2, we have

$$2\beta_N = 2\beta + \sum_{n=1}^{N}(y_n - \mu)^2 = 2\beta + N\sigma_{ML}^2 \tag{4.13}$$

where σ_{ML}^2 denotes the maximum likelihood estimator of the variance arising the from data samples $(y_1, ..., y_N)$. From this, hyper-parameter β can be interpreted as the 2β total 'prior' variance arising from the corresponding 2α 'effective' prior observations' of 'pseudo' data samples.

Mapping hyper-parameters to evidence-based uncertainty. The above analysis provides an intuitive interpretation of key hyper-parameters introduced along with the prior distributions in the hierarchical model.

This will help to understand their key roles in defining different types of uncertainties introduced next. In particular, since both ν and α are essentially the 'effective' prior observations, it is natural to treat their posterior counterpart ν_N and α_N as the *evidence* to support (or suspect) a prediction given training samples $(y_1, ..., y_N)$. Furthermore, β_N can be treated as the total uncertainty that combines two sources of uncertainty: the prior variance β from the pseudo samples and the variance σ^2_{ML} of the actually observed data samples.

We start defining the model prediction and uncertainty from the data (referred to as aleatoric uncertainty) as

$$\text{Prediction: } \mathbb{E}[\mu] = \gamma_N \tag{4.14}$$

$$\text{Aleatoric: } \mathbb{E}[\sigma^2] = \frac{\beta_N}{\alpha_N - 1} \tag{4.15}$$

where both can be directly obtained as the mean from the corresponding Gaussian and Inv-Gamma posteriors defined in (4.7) and (4.10), respectively. It is interesting to see that the uncertainty from the data is proportion to the total uncertainty β_N and decreases with (both pseudo and actual) observations. Next, we quantify the uncertainty of the model prediction (referred to as epistemic uncertainty) by showing an important relationship with the aleatoric uncertainty through the following theorem.

Theorem 1 *Given a hierarchical model as specified by* (4.4)-(4.6) *and a set of observed (training) data samples* $(y_1, ..., y_N)$*, the epistemic uncertainty that quantifies the variance of the posterior mean (as the model prediction), given by Var*$[\mu]$*, is* $\frac{1}{\nu_N}$ *times of the aleatoric uncertainty:*

$$Var[\mu] = \frac{E[\sigma^2]}{\nu_N} = \frac{\beta_N}{\nu_N(\alpha_N - 1)} \tag{4.16}$$

where ν_N *is defined in* (4.9).

First, note that we cannot directly use the variance given by the posterior distribution in (4.7) as it is still conditioned on σ^2. Since Var$[\mu]$ is defined on the marginal posterior $p(\mu|\mathbf{y})$, we need to further marginalize σ^2, which gives

$$\begin{aligned}\text{Var}[\mu] &= \int\int \left[\mu^2 p(\mu|\sigma^2) - (\mathbb{E}[\mu])^2\right] p(\sigma^2) d\mu d\sigma^2 \\ &= \gamma_N^2 - (\mathbb{E}[\mu])^2 + \int \frac{\sigma^2}{\nu} p(\sigma^2) d\sigma^2 \\ &= \frac{\beta_N}{\nu_N(\alpha_N - 1)}\end{aligned} \tag{4.17}$$

where we omit the dependency on $\mathbf{y}$ to keep the notation uncluttered.

Interpretation. The relationship between epistemic and aleatoric uncertainty given in Theorem 1 has a very intuitive interpretation when we treat both ν_N and α_N as evidence. For the aleatoric uncertainty, it starts with a total uncertainty of β_N and continues to decrease with the collection of more data samples that add to the evidence term α_N. The epistemic uncertainty further considers the belief of a given model (μ). For a model with a total evidence of ν_N, the remaining uncertainty arising from data will further scale down by ν_N, leading to the model uncertainty given in (4.17). By fixing the prior pair (α, ν), we have Var$[\mu] \sim \frac{1}{N^2}$, which implies that the epistemic uncertainty will decrease in a speed of N^2.

Learning Meta-Evidential Distribution. The evidence-based uncertainty analysis reveals that conducting regression based recommendation for cold-start users using a traditional Bayesian model in the few-shot setting faces a number of key challenges. First, in a typical few-shot setting, N is small, which leads to a large epistemic uncertainty that makes the prediction much less confident. While this could be addressed by choosing proper evidence pair (α, ν), setting hyper-parameters is inherently challenging without sufficient prior knowledge. Second, such an approach does not benefit from the meta-learning framework, which aims to leverage useful information from other warm-start users to improve cold-start recommendations.

To tackle these challenges, we propose to directly learn an evidential distribution from data through meta-learning to estimate the potential interaction between a user-item pair (u, i). Instead of solely providing a prediction by using predefined priors and a set of training samples, we directly predict the posterior mean $\gamma_{(u,i)}$, the evidence pair $(\nu_{(u,i)}, \alpha_{(u,i)})$, and the total uncertainty $\beta_{(u,i)}$. Intuitively, for a potential interaction, if it shares important properties with existing interactions, the meta evidential learning model should predict a high evidence along with a low total uncertainty. As evidenced by our experiments, such highly confident predictions supported by the predicted evidence usually lead to a higher accuracy. In contrast, for interactions with a high uncertainty, recommending such items to a user could help the model more effectively capture the user's true preference that leads to more accurate recommendations in future interactions. Our experiments show that by focusing on interactions with a high epistemic uncertainty, our model can use much less interactions to accurately capture a cold-start users' preference.

We start by defining a loss function that is formed through the evidence and total uncertainty parameters. Given an observed score $r_{(u,i)}$ resulted from an interaction between user u and item i, we marginalize the likelihood parameters (μ, σ^2), which gives the marginal likelihood function

$$\begin{aligned} p(r_{(u,i)}|\gamma_{(u,i)}, \nu_{(u,i)}, \alpha_{(u,i)}, \beta_{(u,i)}) =& \int\int p(r_{(u,i)}|\mu, \sigma^2) p(\mu, \sigma^2|\gamma_{(u,i)}, \nu_{(u,i)}, \alpha_{(u,i)}, \beta_{(u,i)}) d\mu d\sigma^2 \\ =& \int\int \mathcal{N}(\mu, \sigma^2)\mathcal{N}(\gamma_{(u,i)}, \sigma^2\nu^{-1})\text{IG}(\alpha_{(u,i)}, \beta_{(u,i)}) d\mu d\sigma^2 \\ =& \text{St}\left(r_{(u,i)}; \gamma_{(u,i)}, \frac{\beta_{(u,i)}(1+\nu_{(u,i)})}{\nu_{(u,i)}\alpha_{(u,i)}}, 2\alpha_{(u,i)}\right) \end{aligned} \tag{4.18}$$

where IG is short for Inv-Gamma and St(.) is a student-t distribution on target variable $r_{(u,i)}$ with respective location and scale parameters.

We adopt an evidential loss, which utilizes the above marginal likelihood while computing predicted loss. This includes negative log-likelihood ($\mathcal{L}^{NLL}[f_{\theta_u,E_u,E_i}]$) to maximize the marginal likelihood and an evidential regularizer ($\mathcal{L}^{R}[f_{\theta_u,E_u,E_i}]$) to impose a high penalty on the predicted error with a low uncertainty (or a large confidence). We first formulate the negative log-likelihood, given by

$$\begin{aligned}\mathcal{L}^{NLL}[f_{\theta_u,E_u,E_i}] &= -\log(p(r_{(u,i)}|\gamma_{(u,i)},\nu_{(u,i)},\alpha_{(u,i)},\beta_{(u,i)}) \\ &= -\log\left(St(r_{(u,i)};\gamma_{(u,i)},\frac{\beta_{(u,i)}(1+\nu_{(u,i)})}{\nu_{(u,i)}\alpha_{(u,i)}},2\alpha_{(u,i)})\right) \\ &= \frac{1}{2}\log\left(\frac{\pi}{\nu_{(u,i)}}\right)-\alpha_{(u,i)}\log(2\beta_{(u,i)}(1+\nu_{(u,i)})) \\ &+(\alpha_{(u,i)}+\frac{1}{2})\log((r_{(u,i)}-\gamma_{(u,i)})^2\nu_{(u,i)} \\ &+2\beta_{(u,i)}(1+\nu_{(u,i)}))+\log\left(\frac{\Gamma(\alpha_{(u,i)})}{\Gamma(\alpha_{(u,i)}+\frac{1}{2})}\right)\end{aligned} \tag{4.19}$$

We formalize our own evidence regularizer, which considers epistemic uncertainty to penalize confident predicted errors. We multiply the predicted error with the inverse epistemic uncertainty that scales up the error when the predicted evidence is high causing high inverse epistemic uncertainty and vice-versa. Conversely, it will be less penalized if the prediction is close to the target score:

$$\mathcal{L}^{R}[f_{\theta_u,E_u,E_i}] = |r_{(u,i)}-\gamma_{(u,i)}|\cdot\left(\frac{\nu_{(u,i)}(\alpha_{(u,i)}-1)}{\beta_{(u,i)}}\right) \tag{4.20}$$

In meta evidential setting, we compute the loss for a specific user u, which can be formulated with user evidential loss as:

$$\begin{aligned}\mathcal{L}_{\mathcal{T}_u}[f_{\theta_u,E_u,E_i}] &= \sum_{u,i\sim\mathcal{T}_u}\mathcal{L}[f_{\theta_u,E_u,E_i}(u,i)], \\ \mathcal{L}[f_{\theta_u,E_u,E_i}(u,i)] &= \mathcal{L}^{NLL}[f_{\theta_u,E_u,E_i}(u,i)]+\lambda_1\mathcal{L}^{R}[f_{\theta_u,E_u,E_i}(u,i)]\end{aligned} \tag{4.21}$$

where λ_1 is a regularization parameter.

The total loss is formed by aggregating all users in the meta-train set, regularized by the L_2 norm of key model parameters. Let θ_u and θ denote the local (user-specific) and global parameters of the meta evidential learner. Training the meta evidential learning as a recommendation model can be formulated as the following optimization problem:

$$\begin{aligned}\min_{\theta} \quad & \sum_{\mathcal{T}_u\sim p(\mathcal{T})}\mathcal{L}_{\mathcal{T}_u}[f_{\theta_u,E_u,E_i}]+\frac{\lambda_2}{2}||\theta||_2^2, \\ & \theta_u = \theta-\eta\nabla_\theta\mathcal{L}_{\mathcal{T}_u}(f_{\theta,E_u,E_i})\end{aligned} \tag{4.22}$$

Algorithm 3 Meta Evidential Training

```
Require: Hyperparameters: $\alpha, \beta, \gamma, \nu$
Require: Stepsize hyperparameters: $\eta, \xi$
 1: Initialize user embedding, $E_u$
 2: Initialize item embedding, $E_i$
 3: Initialize meta learner, $\theta$
 4: while not converge do
 5:     Sample tasks $\mathcal{T}_u \sim p(\mathcal{T})$
 6:     for all $\mathcal{T}_u$ do
 7:         Sample support set, $\mathcal{S}_u \in \mathcal{T}_u$ for the local update
 8:         Perform local update with $\mathcal{S}_u$ for meta evidential learning module using Equation (4.23)
 9:         Sample query set $\mathcal{Q}_u \in \mathcal{T}_u$ for the global update
10:     end for
11:     Perform global update with $\mathcal{Q}_u$ for the meta evidential learning module using Equation (4.24), and
    for user and item embedding using Equation (4.3)
12: end while
```

where θ_u is one gradient step update from global parameter θ of the meta evidential learner with η being the step size, and λ_2 is the regularization parameter.

We apply an optimization-based meta-learning approach [20] to learn user specific factors, as shown in Figure 4.1. The meta evidential learning consists of three fully connected linear layers with ReLU activation in the first two, while the last layer predicts ratings or count and its evidence. The input to the meta evidential learning model is the concatenation of user embedding (z_u) and item embedding (z_i) for each user, i.e., $(z_u||z_i)$. Algorithm 4 shows the training process that learns the model parameters. For the meta evidential learning module, the local update (line 8) is done for the user-specific parameter, which is achieved by one or more gradients from the global parameter:

$$\theta_u = \theta - \eta \nabla_\theta \mathcal{L}_{\mathcal{T}_u}[f_{\theta, E_u, E_i}] \tag{4.23}$$

In this update, the loss function is computed with the support set. Similarly, global update (line 11) is done with the new item interactions of each user from the query set for the meta update:

$$\theta = \theta - \xi \nabla_\theta \sum_{\mathcal{T}_u \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_u}[f_{\theta_u, E_u, E_i}] \tag{4.24}$$

This process continues to find a good global parameter shared by all users. Note that in both local and global meta updates, evidence and uncertainty parameters are incorporated via the loss function as given in Equation (4.21).

4.5 Experiments

In this section, we provide the details of datasets, baselines, experimental settings, results and discussion, and hyperparameter tuning.

4.5.1 Dataset Description

We evaluated our model on four public benchmark datasets. Three are explicit datasets where users provide explicit ratings: MovieLens 1M, Netflix, and Book Crossing datasets; and one implicit dataset where users have implicit interactions, (e.g., count): Last.fm-1K dataset.

- *MovieLens 1M:* This dataset includes 1M explicit feedback (ratings) made by 6,040 anonymous users on 3,900 distinct movies from 04/2000 to 02/2003.
- *Netflix:* This dataset has around 100 million interactions, 480,000 users, and nearly 18,000 movies rated between 1998 to 2005. We pre-processed the dataset and selected 6,042 users that consist of user-item interactions from 01/2002 to 12/2005.
- *Book Crossing:* This dataset contains 278,858 users providing 1,149,780 ratings (explicit/implicit) about 271,379 books in a 4-week crawl (08-09 of 2004). We further pre-processed the dataset and selected 751 users with their corresponding interactions.
- *Last.fm:* This dataset contains the whole listening history (till 05/2009) for nearly 1,000 users.

4.5.2 Baselines

For comparison, we include matrix factorization based deep learning models and meta-learning based recommendation models:

- *DeepFM:* Deep learning based factorization machine [28]. It integrates the power of deep learning and factorization machines models to learn low- and high-order feature interactions.
- *Wide & Deep:* Both deep and wide networks are used to exploit the benefits of generalization and memorization [13].
- *Graph based model:* We use graph convolutional matrix completion (GC-MC) [9], which models recommendation as link prediction in the graph.
- *MeLU:* It utilizes both user and item embeddings to learn meta knowledge which is used to adapt for a new user to perform prediction [47].

- *VAMPIRE:* A Bayesian meta-learning method which estimates the uncertainty in the model-agnostic meta-learning using variational inference [61]. We applied this method in recommender system; however, it is not adopted in any recommendation literature.

4.5.3 Evaluation Metrics

For evaluation, we analyze the experimental results in terms of the deviation of predicted values from the ground truth and use Root Mean Squared Error (RMSE) and Normalized Discounted Cumulative Gain (NDCG) averaged across all test users. RMSE is usually reported for explicit data, while NDCG is usually reported for implicit data:

$$\begin{aligned} \text{RMSE} &= \sqrt{\sum_{r_{u,i} \in O} (\hat{r}_{u,i} - r_{u,i})^2 / |O|}, \\ \text{NDCG}_u &= \sum_n \frac{rel_n^{pred}}{\log_2(1+n)} \Big/ \sum_n \frac{rel_n^{ideal}}{\log_2(1+n)} \end{aligned} \tag{4.25}$$

where O is the observation set for the test set, rel_n is the relevancy of n^{th} item in the ranking sequence for user u, which is binary for implicit data or the rating for explicit data. The NDCG is the fraction of Discounted Cumulative Gain (DCG) of recommendation result over the ideal DCG.

We measure uncertainty based on model evidence for each user. We compute *aleatoric* and *epistemic* uncertainty as given in Equation (4.15) and (4.17). The aleatoric uncertainty measures uncertainty in the data, and epistemic uncertainty measures uncertainty in the model.

4.5.4 Settings

We initialize the meta-evidential learning model with random initial values. Model learning rates are set through a grid search, and the Adam optimizer is applied with L2-regularization. In the evidential meta-learning setting, we split users into meta-train users (who have enough interactions > 30) and meta-test users (who have few interactions ≤ 30). For example, in Movielens-1M dataset, we split 5,231 users for meta-train and 809 users for meta-test . In general, the meta-learning model does K-shot learning in which K examples are taken from each task as a support set and the rest for a query set. In our case, the model takes historical interactions of the users and performs any-shot meta-evidential learning. In other words, we vary the number of interacted items into support set by making a fixed number of items in the query set (e.g., for Movielens-1M dataset, we fixed interacted items in the query set equal to 10 and the rest of interactions for support set). We performed this any-shot learning to capture users' uncertainty that helps to show model

Table 4.3: Performance of Recommendation (average RMSE and NDCG)

Model	MovieLens-1M		Book Crossing		Netflix		Last.fm	
	RMSE	**NDCG**	**RMSE**	**NDCG**	**RMSE**	**NDCG**	**RMSE**	**NDCG**
deepFM	1.0254±0.03	0.2913	4.0889±0.06	0.2733	0.9699±0.02	0.2915	1.1939±0.05	0.2807
Wide & Deep	1.0218±0.03	0.2932	4.1341±0.08	0.2745	0.9686±0.02	0.2944	1.1847±0.05	0.2812
GC-MC	1.0313±0.03	0.2872	4.1405±0.10	0.2712	0.9816±0.03	0.2814	N/A	N/A
MeLU	1.0195±0.02	0.3308	3.7388±0.05	0.2811	0.9613±0.02	0.3265	1.0711±0.03	0.3102
MetaEDL	**1.0114±0.02**	**0.3493**	**3.7026±0.04**	**0.3046**	**0.9525 ±0.02**	**0.3488**	**1.0183±0.03**	**0.3233**

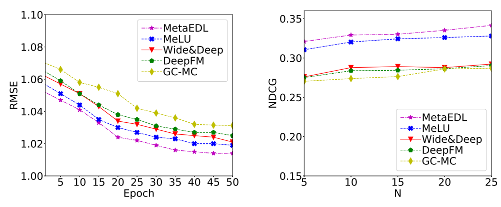
confidence for each user with varying length of their interactions. This setting is designed explicitly for cold-start recommendations since we consider test users with few interactions.

4.5.5 Results and Discussion

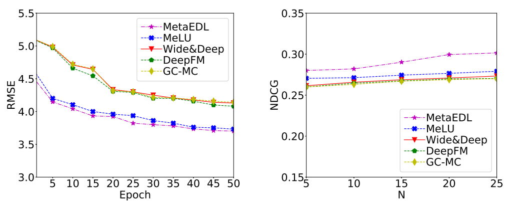
The experimental results for the proposed model and baselines are summarized in Table 6.2. We compute the average RMSE considering all users with the range of deviation for all datasets: MovieLens 1M, Book Crossing, Netflix, and Last.fm, respectively. The proposed model benefits from the meta-learning module, and hence it can effectively handle cold-start users who have few interactions like those in Book Crossing datasets. We also observe from Table 6.2 that deep learning and graph based models have poor performance on the Book Crossing datasets than meta-learning models like MeLU and the proposed model achieves significant improvements. For the last.fm dataset, the meta-learning models have shown a clear indication of improvement again over deep learning and graph based models is not applicable due to implicit datasets. For the Movielens 1M and Netflix datasets, most users have enough interactions, and hence all models achieve comparable performances. We further provide top N NDCG performance ranging from top 5 to top 25 and their respective values for each model. For this, we chose those test users who have 30 interactions so that we can use 25 interactions for query set to compute NDCG. Similarly, we report the average RMSE values of test users considering the number of training epochs. Both NDCG and RMSE are shown in Figure 6.2.

We further analyze the relationship between predictive confidence and model performance as shown in Figure 4.3.

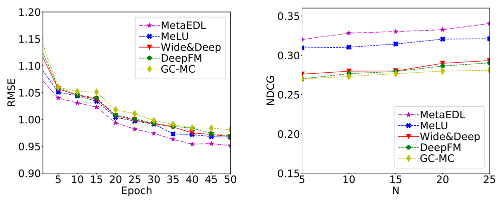
The predictive confidence is estimated based on an epistemic uncertainty threshold and model performance with each user's RMSE loss. The figure shows that RMSE loss and epistemic uncertainty positively correlate: higher uncertainty leads to a higher RMSE loss. This relationship applies to all datasets. The Book Crossing dataset has a larger RMSE loss due to its range of target labels are from 0-10. The last.fm has count data, and its target labels vary largely. To address this, we did a logarithmic transformation of the



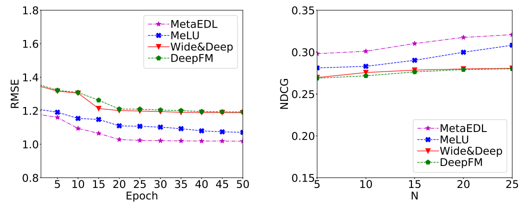
(a) MovieLens-1M



(b) Bookcrossing

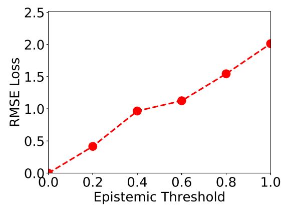


(c) Netflix

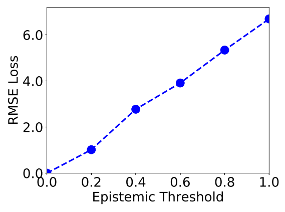


(d) Last.fm

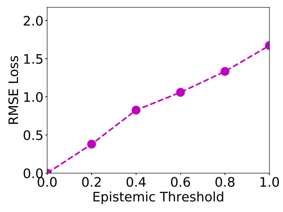
Figure 4.2: NDCG based on the top N recommendations and RMSE based on the number of training epochs



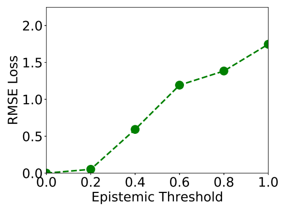
(a) Movielens 1M



(b) Book Crossing



(c) Netflix



(d) Last.fm

Figure 4.3: Relationship between epistemic uncertainty threshold and observed error for four datasets.

count data to make the network's inputs have a reasonable range.

Our main objective is to show the type of interactions matters more than the number of interactions that users have. To demonstrate this, we estimate uncertainty considering the diversity in genres of each user in the Movielens 1M dataset. We initially find the top 10 highest epistemic and lowest epistemic users. Based on their preference of genres, we group users in two groups: $group$ 1 with high drama and low romance and else $group$ 2. Then, we compute the corresponding RMSE loss and both aleatoric and epistemic uncertainty as shown in Table 4.4.

Table 4.4: Uncertainty for group of users with different genres

User Group	**Users**	**RMSE**	**Epistemic**	**Aleatoric**
Group 1	1734	1.0139	0.4447	0.1711
Group 2	265	1.0465	0.4651	0.1713

We observed that users with low epistemic uncertainty are most likely to watch movies that belong to the drama genre, and less likely to watch romance movies. Likewise, users with higher epistemic uncertainty like both genres. To extend our analysis, we further analyze the Book Crossing dataset in which we group users based on their interactions with books published by popular authors. We follow a similar approach applied in Movielens dataset and group users based on low epistemic with popular authors as a $group$ 1 and rest $group$ 2. Table 4.5 shows that popularity bias exists in the recommender systems and can be captured by this proposed model.

Table 4.5: Uncertainty for group of users with popular authors.

User Group	**Users**	**RMSE**	**Epistemic**	**Aleatoric**
Group 1	6	3.1255	0.8377	0.4066
Group 2	255	3.7406	0.9111	0.4092

Intuitively, the proposed model utilizes epistemic uncertainty to show the popularity bias. Our model not only provides model uncertainty via epistemic uncertainty but also provides data uncertainty through aleatoric uncertainty. From Tables 4.4 and 4.5, for each dataset their user representation doesn't varies much. This empirically supports the way we represent our user and item embeddings. We represent each user and item embedding just doing one hot encoding utilizing unique IDs of user and item, and hence they have the same level of noise in representation.

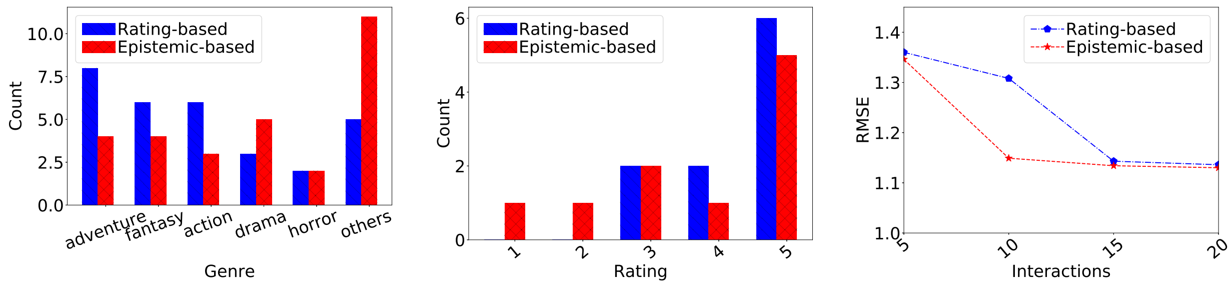


Figure 4.4: Genre count and rating count for the items selected in the support set with size 10 and RMSE for the query set

4.5.6 Uncertainty-Aware Recommendations

In this set of experiments, we show how the model effectively leverages predicted uncertainty to recommend the *most informative items* rather than solely based on the predicted ratings. For this, we randomly chose a test user (*ID: 41*) from the Movielens-1M dataset. This user has a total of 25 interactions, and we randomly choose 20 interactions that serve as the candidate pool to form the support set. The remaining 5 interactions are used for a query set. We perform uncertainty-based recommendation to tackle cold-start problem where we recommend a few items from the pool according to their epistemic uncertainty (instead of predicted ratings). By collecting only limited interaction results, we expect the model to learn the most from the cold-start user (by reducing the epistemic uncertainty) to provide more accurate recommendation in the future. To demonstrate that the uncertainty-based recommendation can lead to better future predictions, we also employ the classical rating based recommendation to select same number of highest rated items. After the adaptation using the selected support set, both methods will be evaluated on the same query set for comparison.

We first show total counts of genres and ratings by the left and middle plots of Figure 4.4. From those plots, we can clearly see that the epistemic method selects more diverse genres with more count in *others* genres. It also selects items with relatively lower ratings than the rating-based method. Table 4.6 shows important items selected by both methods: most of the rating-based movies belong to *adventure* and *action* genres, whereas the epistemic method selects more diverse genres. This suggests that rating based recommendation seems more specific to the adventure movies, whereas epistemic method selects more diverse genres, including *drama*, *adventure*, and a higher number of *others* genres. Here, the support set has 15 different genres and each movie can have multiple genres. We showed 5 major genres in Figure 4.4 and remaining genres as an accumulated count in the others genre. This clearly shows that uncertainty based model focuses on more informative items so that model becomes more confident after observing those diverse but important interactions.

We further investigate how interactions selected based on epistemic uncertainty help to provide a better future recommendation. For this, we make fast adaptation of our meta-train model with those few interactions resulted from the recommended items and then perform testing on the query set. We start by adding 5 interactions and continue to add 5 in each round until all the items in the candidate pools are used.

As we can see from the right plot of Figure 4.4, after adding 10 interactions based on the recommended items, the epistemic method achieves almost optimal performance on the query set. In contrast, the rating based method requires more than 15 interactions to achieve similar performance on the query set. In Table 4.7, we report the movie names of the top-3 recommended items for both methods on the query set along with the ground-truth. It clearly shows that the epistemic uncertainty-based method recommends more accurate items, which matches movies from highly rated ground truth ones.

The above experiments clearly show that using limited but highly informative items for recommendation, we could more accurately capture the latent preference users that is instrumental for cold-start recommendations.

Table 4.6: Important Movies for user 41.

Method	**Important Items (Movies)**
Rating	['Star Wars: Episode I - The Phantom Menace', 'Star Wars: Episode IV - A New Hope', 'Anastasia', 'Star Wars: Episode V - The Empire Strikes Back', 'Arachnophobia']
Epistemic	['I Still Know What You Did Last Summer', 'Star Wars: Episode IV - A New Hope', 'Arachnophobia','Gladiator', 'Mystery Science Theater 3000: The Movie genre']

Table 4.7: Top 3 recommended items from random and epistemic methods

Method	**Recommended Movies**
Rating	['The Deep End of the Ocean', 'A Life Less Ordinary', 'Braveheart']
Epistemic	['Titanic', 'A Life Less Ordinary', 'The Deep End of the Ocean']
Ground-Truth	['Titanic', 'Star Wars: Episode VI - Return of the Jedi', 'A Life Less Ordinary']

4.5.7 Hyperparameter Tuning

MetaEDL exploits evidential regularizer loss ($\mathcal{L}^R$) with regularization constant (λ_1=1e-3) which is cross validated from $\{$0, 1e-4, 1e-3, 1e-2, 1e-1, 1$\}$. Similarly, we cross validated coefficient of the L_2 regularizer λ_2=1e-4, learning rates ξ=1e-4, and η=1e-3 and embedding dimension d=64.

Analysis of Evedential Regularization Parameter

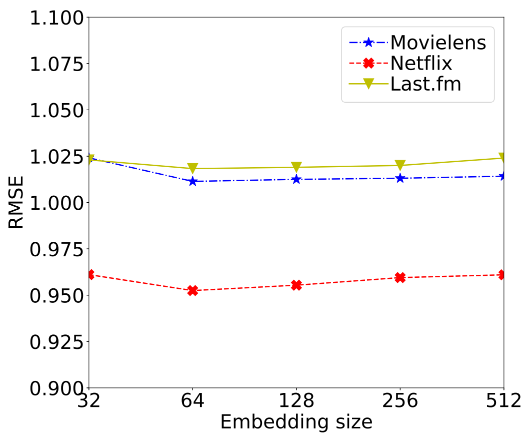
One of the key hyperparameters of the *MetaEDL* model is the regularizer coefficient (λ_1) for evidential learning. We cross-validated this parameter with empirical results of model RMSE loss for different λ_1 values on two datasets (other datasets are omitted due to limited space) as shown in Table C.2.

Analysis of Embedding Dimension

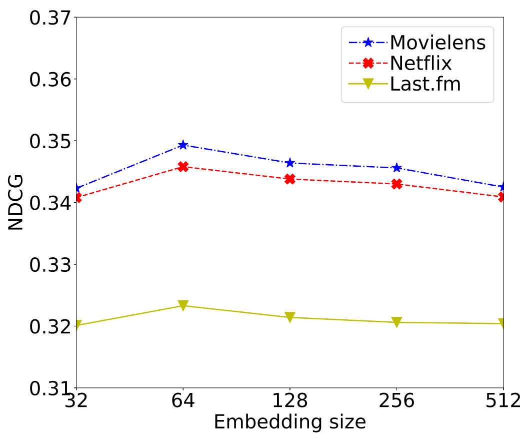
We generate user and item embeddings using the embedding module. We perform grid search for embedding dimension (d) of the user and item in *MetaEDL* for three datasets as shown in Figure 4.5. The Book Crossing dataset follows a similar trend, and we didn't incorporate it into the figure due to its higher RMSE range.

Table 4.8: Average RMSE loss with different regularizer values

Regularizer	**MovieLens 1M**	**Book Crossing**
(λ_1)	**RMSE**	**RMSE**
0	1.0138±0.02	3.7132±0.04
1e-4	1.0131±0.02	3.7082±0.04
1e-3	**1.0114±0.02**	**3.7026±0.04**
1e-2	1.0137±0.02	3.7103±0.04
1e-1	1.0242±0.02	3.7210±0.04
1	1.0248±0.02	3.7321±0.04



(a) RMSE



(b) NDCG

Figure 4.5: Impact of item embedding size

4.6 Conclusions

In summary, we present a novel meta evidential learning recommendation framework that integrates evidential learning with a meta-learning approach to provide uncertainty-aware cold-start recommendations. The proposed framework handles the user cold-start problem by adopting global knowledge of similar users from their interaction information and leveraging evidential learning for efficient posterior inference to further quantify the model confidence. Experimental results on four real-world datasets and comparison with the state-of-the-art competitive models clearly demonstrate the effectiveness of the proposed model.

Chapter 5

Meta Evidential Reinforcement Learning for Sparse User Interactions

In this chapter, we propose two evidential reinforcement learning approaches for sparse user interactions. The goal of both techniques is to address the learning limitations present in the existing techniques. Our first approach tries to address the challenging problem of balancing exploitation (with high predicted ratings) and exploration (with evidence-based uncertainty) strategies for effective recommendations. We formulate an evidential RL framework that augments the maximum reward RL objective with evidence-based uncertainty maximization. More importantly, the evidence-based uncertainty formulation substantially improves exploration and robustness by acquiring diverse behaviors that are indicative of a user's long-term interest. Our second approach leverages evidential reinforced attention to uniquely discover the user behavior patterns from sparse datasets collected via interactive systems. Further, evidential reinforced attentions (ERA) perform evidence-aware exploration and attentively selects and builds sub-spatiotemporal sets. In particular, we introduce a uniquely designed reward function to encourage attention to the unknown but potentially important behavioral sub-sequences. The reward simultaneously considers both the prediction accuracy for exploitation and evidence-based uncertainty estimation for unknown behavior exploration. Inspired by the learning to learn setting we formulate a task in few-shot learning, and design the training objective as a sub-trajectory classification problem.

5.1 Evidential Reinforcement Learning for Dynamic Recommendation

Recommender systems (RS) have been widely used for providing personalized recommendations in diverse fields such as media, entertainment, and e-commerce by effectively improving user experience [73, 75, 87]. Most existing RS methods consider recommendation as a static process, and therefore they cannot consider users' evolving preferences. Some efforts have been devoted to capture users' evolving preferences by shifting the user latent preference over time [11, 27, 45]. Similarly, sequential recommendation methods [39, 78] attempt to incorporate users' dynamic behavior by leveraging previously interacted items. However, both the abovementioned static and dynamic recommendation methods primarily focus on maximizing the immediate (short-term) reward when making recommendations. As a result, they fail to take into account whether these recommended items will lead to long-term returns in the future, which is essential to maintain a stable user base for the system in the long run. further, they primarily rely on standard exploration strategies (ϵ-greedy), which are less effective in a large item space with sparse reward signals given the limited interactions for most users. Therefore, they may not be able to learn the optimal policy that captures effective user preferences and achieves the maximum expected reward in the long run.

To address the above key challenges, we conduct novel deep evidential reinforcement learning (DERL) that utilizes a balanced exploitation and exploration strategy for effective recommendations. We formulate an evidential RL framework that augments the maximum reward RL objective with evidence-based uncertainty maximization. More importantly, the evidence-based uncertainty formulation substantially improves exploration and robustness by acquiring diverse behaviors that are indicative of a user's long-term interest. The proposed DERL seamlessly integrates two major components: a customized *RNN* and an *Evidential Actor-Critic (EAC)* module. The former primarily focuses on generating the current state of the environment by aggregating the previous state, current items captured by a sliding window, and future recommended items from the RL agent. This provides effective means of dynamic state representation for better future recommendations. Meanwhile, the EAC module leverages evidence-based uncertainty to effectively explore the item space to identify items that potentially align with the user's long-term interest. It encourages learning the optimal policy by maximizing a novel conservative evidential Q-value to achieve a maximum long-term cumulative reward. The main contribution are:

- A novel recommendation model that integrates reinforcement learning with evidential learning to provide uncertainty-aware recommendations.
- Evidence-based uncertainty maximization to enable stability and effective exploration.
- An off-policy formulation to effectively promote the reuse of previously collected data while stabilizing model training, which is important to address data scarcity in recommender systems.
- Seamless integration of a customized RNN, an actor-critic network, and an evidential network to provide

an end-to-end integrated training process.

We conduct extensive experiments over four real-world datasets and compare with state-of-the-art baselines to demonstrate the effectiveness of the proposed model.

5.1.1 Related Work

Static models. Matrix Factorization (MF) leverages user and item latent factors to infer user preferences [22,44,46]. MF is further extended with Bayesian Personalized Ranking (BPR) [66] and Factorization Machine (FM) [65]. Recently, deep learning-based recommender systems [13, 28] have achieved impressive performance. DeepFM [28] integrates traditional FM and deep learning to learn low- and high-order feature interactions. Both wide and deep networks are jointly trained in [13] for better memorization and generalization. In graph-based methods [9], users and items are represented as a bipartite graph and links are predicted to provide recommendations. Similarly, Neural Graph Collaborative Filtering [84] explicitly encodes the collaborative signal via high-order connectivities in the user-item bipartite graph via embedding propagation.

Dynamic and sequential models. Dynamic model shifts latent user preference over time to incorporate temporal information. TimeSVD++ [45] considers time-specific factors, which uses additive bias to model user and item related temporal changes. Gaussian state-space models have been used to introduce time-evolving factors with a one-way Kalman filter [27]. To process implicit data, Sahoo et al. extended the hidden Markov model [67], and Charlin et al. [11] further augmented it with the Poisson emission. However, these models capture user evolving preference, and they are less aware of future interactions and provide recommendations based on fixed strategies. Similarly, sequential models utilize users' historical interactions to capture users' preferences over time. Tang et al. utilized a CNN architecture to capture union level and point level contributions [78]. Also, Kang et al. leveraged transformer-based user representation to better capture their interest [39] and Sun et al. utilized bidirectional encoder for sequential recommendation [74]. Sequential models neglect long-term users' preferences. The proposed DERL model aims to fill this critical gap by performing evidence guided exploration and maximizing total expected reward.

RL-based models. RL-based RS models aim to learn an effective policy to maximize the total expected reward in the long run. The on-policy learning with contextual bandit [50] and Markov Decision Process (MDP) [95] exploits by interacting with real customers in an online environment. A collaborative contextual bandit algorithm called CoLin [85] utilizes graph structure in a collaborative manner. On the other hand, off-policy utilizes Monte Carlo (MC) and temporal-difference (TD) methods to achieve stable and efficient learning with users' history [18]. Similarly, model-based RL models user-agent interaction via a generative adversarial network [5]. Pseudo Dyna-Q [98] further integrates both direct and indirect RL approaches in a single unified framework without requiring real customer interactions. However, the above methods utilize random exploration strategies, which are less effective at capturing users' long-term preferences. In contrast, our DERL utilizes evidence-based uncertainty to systematically explore the item space to maximize the long-term reward.

5.1.2 Preliminaries

We first introduce the standard RS setup in RL and provide an overview of the evidential theory.

Recommendation Formulation with RL. We formulate recommendation tasks in an RL setting, where an RL agent interacts with the environment (users and items) to recommend the next items to a user over time in a sequential order to maximize the cumulative reward. We design this problem as the MDP, which includes a sequence of states, actions, and rewards. More formally, a tuple $(\mathcal{S}, \mathcal{A}, p, r)$ is defined as:

- **State space ($\mathcal{S}$):** A state $\mathbf{s}_t = \texttt{RNN}(\cdot|\mathbf{s}_{t-1}, \mathbf{u}_t) \in \mathcal{S}$ is generated by a customized RNN that utilizes previous state $\mathbf{s}_{t-1}$ and current user $\mathbf{u}_t$ embedding which is generated from the concatenation of M recently interacted items provided by a sliding window (see details later) and an RL-agent.
- **Action space ($\mathcal{A}$):** An action $\mathbf{a}_t \in \mathcal{A}$ is represented as a continuous parameter vector that recommends top-N items for a user based on the current state $\mathbf{s}_t$ at time t.
- **Transition probability (p):** The transition probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ quantifies the probability from state $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ with an action $\mathbf{a}_t$.
- **Reward (r):** The environment provides an immediate reward as a feedback based on items recommended (actions $\mathbf{a}_t$) to the user in state $\mathbf{s}_t$.

Uncertainty and the Evidential Theory. Theory of evidence is a generalization of Bayesian theory to subjective probabilities [15]. We briefly introduce subjective logic (SL) [38] and discuss uncertainty estimation based on SL. SL is a probabilistic logic that is built upon probability theory and belief theory. It represents uncertainty by introducing vacuity of evidence in its opinion, which is a multinomial random variable y in domain $\mathbb{Y} = \{1, ..., K\}$. This opinion can be equivalently represented by a K-dimensional Dirichlet distribution Dir$(p|\alpha)$ where $\boldsymbol{\alpha}$ is a strength over K classes and $p = (p_1, ..., p_K)^\top$ governs a categorical distribution over $\mathbb{Y}$. The term *evidence* is the measure of the number of supportive observations from data for each class. It has a fixed relationship with the concentration parameter $\boldsymbol{\alpha}$ given a non-informative prior. Let e_k be the evidence for a class k. SL measures different types of second-order uncertainty through evidences, including vacuity, dissonance, and a few others [37]. In particular, vacuity corresponds to the uncertainty mass of a subjective opinion ω:

$$vac(\omega) = \mathcal{U} = \frac{K}{S}, \quad S = \sum_{k=1}^{K}(e_k + 1) \tag{5.1}$$

Since vacuity is defined by a lack of evidence in the data sample, it provides a natural way to facilitate the exploration of an RL agent, which will be detailed next.

5.1.3 Proposed Model

The proposed model is shown in Figure 5.2. The model includes a recurrent neural network (RNN) to maintain dynamic state space, and an evidential-actor-critic (EAC) module to explore the item space by introducing the evidence-based uncertainty (vacuity) into a new evidential RL setting.

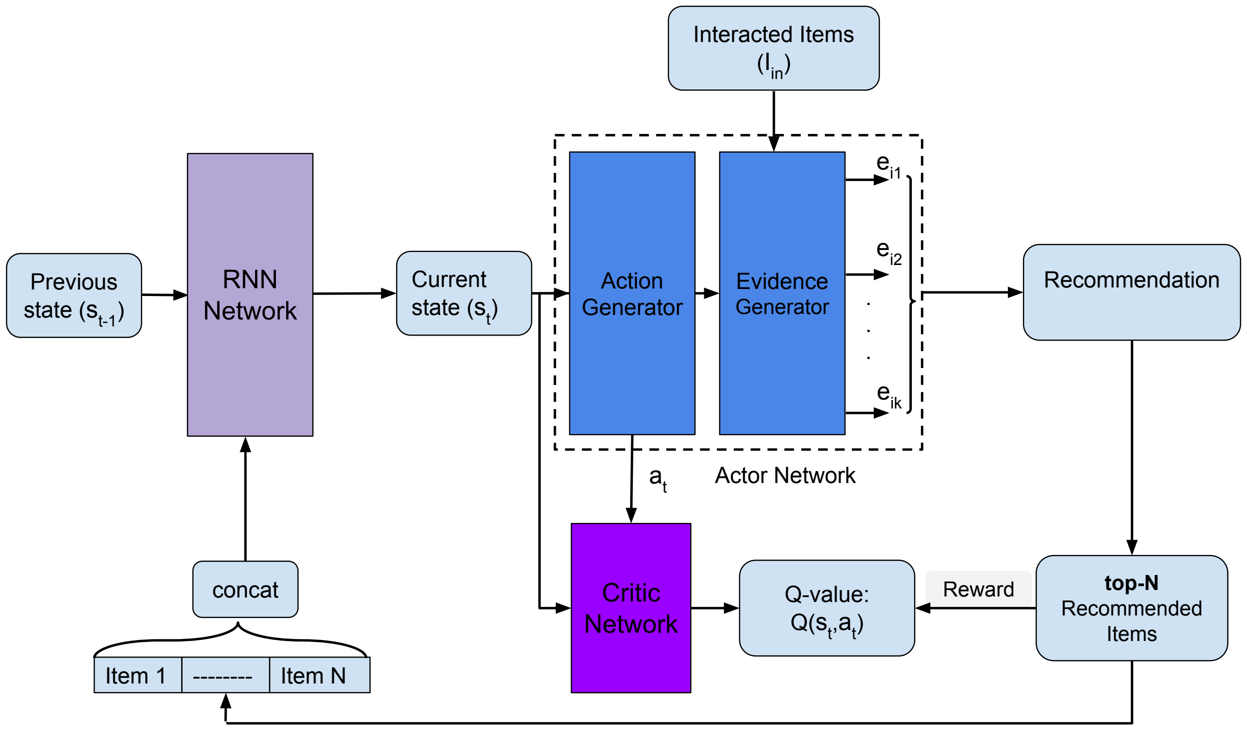


Figure 5.1: Overview of the proposed framework

5.1.4 Environment Setup

The environment consists of user-interacted items (an item pool $\mathcal{I}$) from this user's interaction history H_u , embedding matrix E to generate the user embedding, the RNN for dynamic state generation, and an evidential reward process (ERP) that specifies an incentive mechanism to each action of the agent. Our reward process encourages a balance between exploitation (based on predicted ratings) and exploration (based on evidence-based uncertainty) when making recommendations to users. In particular, a recommended list consists of a limited number of items. The proposed ERP ranks candidate items according to an evidential score that integrates the predicted rating and evidence-based uncertainty:

$$\text{score}_{u,i} = \widehat{\text{rating}}_{u,i} + \lambda \mathcal{U}_\pi(i|\mathbf{s}_t, \mathbf{a}_t) \tag{5.2}$$

where λ balances the rating and the uncertainty, and $\widehat{\text{rating}}_{u,i}$ is the predicted rating. Given K possible rating classes, the evidence network (introduced later in this section) outputs an evidence vector $\mathbf{e}_i = (e_{i1}, ..., e_{iK})^\top$ for each item i. This will allow us to evaluate $\widehat{\text{rating}}_{u,i}$ as $\sum_{k=1}^{K} p_{ik} \times k$ where p_{ik} is rating probability given by (5.9). Meanwhile, uncertainty $\mathcal{U}_\pi(i|\mathbf{s}_t, \mathbf{a}_t)$ for item i can be evaluated through (5.1). Based on the evidential score, an RL agent will choose the top-N items to form a list $\mathcal{N}_u$ and recommend them to the user. As feedback to the agent, the user provides the actual rating for each recommended item. Consequently, the evidential reward is

$$r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) = \frac{1}{N}\left(\sum_{i \in \mathcal{N}_u} (\text{rating}_{u,i} - \tau) + \lambda \mathcal{U}_\pi(i|\mathbf{s}_t, \mathbf{a}_t)\right) \tag{5.3}$$

where $\text{rating}_{u,i}$ is the user assigned ground truth rating and τ is a threshold chosen based on the rating mechanism ($\tau = 3$ for a $1 - 5$ rating system). Given the evidential reward, we introduce an evidential Q-value, which can be computed by repeatedly applying the Bellman operator (B^π):

$$B^\pi Q^e(\mathbf{s}_t, \mathbf{a}_t) r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim \pi}[V(\mathbf{s}_{t+1})] \tag{5.4}$$

where $V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi}[Q^e(\mathbf{s}_t, \mathbf{a}_t)]$.

The evidential Q-value will be used for the update of the EAC module, which is introduced later in this section.

5.1.5 The Customized RNN for Latent State Generation

A specially designed RNN is used to maintain the state space of a dynamic RS environment. In particular, a state $\mathbf{s}_t$ is generated by aggregating three pieces of information: the previous state $\mathbf{s}_{t-1}$, items interacted by the user in the the current step, and newly recommended items. Here, *item* is also an embedding vector which encodes item entity information. By aggregating all these information, the current state can evolve from the previous state by effectively capturing the past preference and future predicted preference of the user. In particular, newly interacted items are extracted from the user's interaction history H_u using a sliding window and the currently recommended items are obtained by invoking the $\mathbf{a}_{t-1}$. Assume that a total M items are obtained with a half from the sliding window and the rest from the action. These M items then go through an embedding matrix to produce a user embedding u_t for time step t. Then, $\mathbf{s}_t$ is formed by

$$\mathbf{s}_t = \texttt{RNN}(\mathbf{s}_{t-1}, \mathbf{u}_t) \tag{5.5}$$

To train the customized RNN, we collect additional data tuples $[\mathbf{s}_{t-1}, \mathbf{u}_t]$ into the reply buffer. We then sample batches from the buffer and send $\mathbf{u}_t$ and $\mathbf{s}_{t-1}$ to the RNN module that generates the current state $\mathbf{s}_t$. After that, we send $\mathbf{s}_t$ to action network that samples $\mathbf{a}_t$ from the action distribution. Action $\mathbf{a}_t$ will then go through the evidence network to predict the evidence vector for each candidate item. Finally, we compute evidential loss J_{Evi} as defined in (5.10) and conduct backpropagation with respect to RNN parameter ω:

$$\nabla_\omega J_{\texttt{RNN}}(\omega) = \nabla_\omega J_{Evi}(\psi) \tag{5.6}$$

In this way, the computing graph is maintained even in the offline setting and the RNN can be trained as in the standard supervised setting.

5.1.6 Evidential Actor Critic (EAC)

Training goal. A standard RL model maximizes the expected sum of reward. We consider a generalized evidential reward function r^e_π defined in (5.3), which augments the standard RL objective with the average evidence-based uncertainty of the recommended items to encourage exploration of the item space. We achieve our training goal by updating the evidential actor network that finds the optimal policy to maximize the expected cumulative evidential reward as:

$$J_\pi = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t)\sim D}(r^e_\pi(\mathbf{s}_t, \mathbf{a}_t)) \tag{5.7}$$

where D is the distribution of $(\mathbf{s}_t, \mathbf{a}_t)$ from the data or the replay buffer and T is the total time steps in the episode. A novel benefit of the new objective is to allow the agent to interact with more informative items for more effective exploration of a large item space. EAC consists of three key networks: *action network*, *evidence network*, and *critic network*, which will be detailed next.

Action network. The action network (or policy network) utilizes the current state $\mathbf{s}_t$ from the offline replay buffer and outputs a policy distribution $\pi(.|\mathbf{s}_t)$, which is modeled as a Gaussian. From this distribution, we sample an action $\mathbf{a}_t$ that is used in the evidence and the critic networks to provide recommendations and direct the policy update, respectively. For action network update, we use backward update signals from the critic network:

$$\nabla_\phi J_\pi(\phi) = (-\nabla_{\mathbf{a}_t} Q^e(\mathbf{s}_t, \mathbf{a}_t)) \times \nabla_\phi \pi(\cdot|\mathbf{s}_t, \phi) \tag{5.8}$$

This gradient extends DDPG style policy update [53] by utilizing the chain rule to the Q-network that updates the action network.

Evidence network. The evidence network predicts a Dirichlet distribution of class probabilities, which can be considered as an evidence collection process. The learned evidence is informative to quantify the predictive uncertainty of recommended items. The evidence network takes action $\mathbf{a}_t$ from replay buffer and item pool $\mathcal{I}$ to provide class level evidence. Then the probability assigning rating k is

$$p_{ik} = \frac{(e_{ik} + 1)}{S_i} \tag{5.9}$$

where e_{ik} is the evidence collected for rating k for item i. To train the evidence network, we define a standard evidential loss by utilizing the MSE loss between rating class probability p_{ik} and the one-hot ground truth label y_i, in which $y_{ik} = 1$ if k is the correct rating, otherwise $y_{ik} = 0$:

$$J_{Evi}(\psi) = \sum_{k=1}^{K} (y_{ik} - p_{ik})^2 + \frac{p_{ik}(1 - p_{ik})}{S_i + 1} \tag{5.10}$$

We update the network by backpropagating the evidential loss $J_{Evi}(\psi)$ with its network parameters ψ.

Critic network. The critic network is designed to approximate evidential Q value utilizing the current state $\mathbf{s}_t$ and action $\mathbf{a}_t$ in a fully connected neural network $Q_\theta(\mathbf{s}_t, \mathbf{a}_t)$. This Q-value judges whether the agent generated actions matches the current state $\mathbf{s}_t$ requirements. We derived an update formulation for the critic network following the double DQN [31] method that utilizes two critic networks to stabilize training process, achieve faster convergence, and provide better Q-value as:

$$\tilde{Q}^e(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{\mathbf{s}_{t+1} \sim D, \mathbf{a}_{t+1} \sim \pi}[r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) + \gamma \times \min\{Q^e(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}), \hat{Q}^e(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})\}] \tag{5.11}$$

where $\hat{Q}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})$ is a target network, which is updated slowly to stabilize the training process.

Now the *evidential Q-function* parameters can be trained by minimizing the temporal difference (TD) error:

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, r^e_\pi(\mathbf{s}_t, \mathbf{a}_t)) \sim D}[\frac{1}{2}(Q^e(\mathbf{s}_t, \mathbf{a}_t) - \tilde{Q}^e(\mathbf{s}_t, \mathbf{a}_t))^2] \tag{5.12}$$

where D is the distribution of $(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, \mathbf{a}_{t+1}, r^e_\pi(\mathbf{s}_t, \mathbf{a}_t))$ in offline buffer.

Further, Q-network is optimized with stochastic gradient decent.

5.1.7 Derivation of Evidential Policy Iteration

We derive evidential policy iteration as a general method for learning optimal uncertainty policies by alternating between evidential policy evaluation and evidential policy improvement in the maximum uncertainty framework. We compute the value of a policy π according to the maximum uncertainty objective of Eq. (5.32). DERL expresses policy as a Gaussian policy with mean and covariance of an action neural network. With the above settings, we show the evidential policy iteration can achieve the optimal policy at convergence.

Lemma 2 (Evidential Policy Evaluation) *Given the Bellman operator B^π in Eq.* (5.4) *and $Q^{n+1} = B^\pi Q^n$, the Q-value will converge to the evidential Q-value of policy π as $n \to \infty$.*

Proof Given the evidential reward defined as $r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) = \frac{1}{N}\left(\sum_{i\in\mathcal{N}_u}(\text{rating}_{u,i} - \tau) + \lambda\mathcal{U}_\pi(i|\mathbf{s}_t, \mathbf{a}_t)\right)$ the update rule for evidential Q-value can be written as:

$$Q^e(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_\pi \sum_{t'=t}^{\infty} \gamma^{t'} r^e_\pi(\mathbf{s}_{t'}, \mathbf{a}_{t'}) = r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1}}[Q^e(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})] \tag{5.13}$$

Then based on the evaluation convergence rule [76] with finite action space, it is guaranteed that the Q-value will converge to the evidential Q-value of policy π.

Lemma 3 *(Evidential Policy Improvement) Given a new policy π_{new} that is updated via Eq* (5.8)*, then $Q^e_{\pi_{new}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t)$.*

Proof The policy can be updated towards the new Q-value function. Consider the updated policy π_{new} as the optimizer of the maximization problem.

$$\pi_{new} = \underset{\pi'}{\text{argmax}}\, J_\pi(\phi) = \underset{\pi'}{\text{argmax}}\, \mathbb{E}_{\mathbf{s}_t \sim D, \mathbf{a}_t \sim \pi'}[Q^e_{\pi'}(\mathbf{s}_t, \mathbf{a}_t)] \tag{5.14}$$

Denote the old policy as π_{old}. Using the update rule specified in Eq 5.8 with a sufficiently small step size, we get an updated policy π_{new} that satisfies

$$\mathbb{E}_{\mathbf{a}_t \sim \pi_{new}}[Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t)] \geq \mathbb{E}_{\mathbf{a}_t \sim \pi_{old}}[Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t)] \tag{5.15}$$

Given Equation B.3, we have the following inequality

$$\begin{aligned}
Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t) \leq & r^e(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \sim \pi_{new}}[Q^e_{\pi_{old}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})] \\
\leq & r^e(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \sim \pi_{new}}[r^e(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})] \\
& + \mathbb{E}_{\mathbf{s}_{t+2}, \mathbf{a}_{t+2} \sim \pi_{new}}[Q^e_{\pi_{old}}(\mathbf{s}_{t+2}, \mathbf{a}_{t+2})] \\
& \ldots \\
= & Q^e_{\pi_{new}}(\mathbf{s}_t, \mathbf{a}_t)
\end{aligned} \tag{5.16}$$

where $r^e(\mathbf{s}_t, \mathbf{a}_t)$ is a evidential reward in step t. Therefore, we show that the new policy π_{new} ensures $Q^e_{\pi_{new}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t)$.

Theorem 4 *(Evidential Policy Iteration) Alternating between evidential policy evaluation and evidential policy improvement for any policy* $\pi \in \Pi$ *converges to an optimum evidential policy* π^* *such that* $Q^{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^e_{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ *for all* $(\mathbf{s}_t, \mathbf{a}_t)$.

Proof Let π_i denote the policy at iteration i. We already show that the sequence $Q^e_{\pi_i}(\mathbf{s}_t, \mathbf{a}_t)$ is monotonically increasing. Since $Q^e_{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ is bounded above, the sequence converges to some π^*. At convergence, it must be the case that $J_{\pi^*}(\pi^*(.|\mathbf{s}_t)) \leq J_{\pi^*}(\pi(.|\mathbf{s}_t))$ for $\pi \neq \pi^*$. Based on Lamma 11, we have $Q^e_{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) > Q^e_{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t)$. In other words, the evidence value of any other policy π is lower than that of the converged policy π^*. Therefore, it guarantees convergency to an optimal policy π^* such that:

$$Q^e_{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^e_{\pi}(\mathbf{s}_t, \mathbf{a}_t) \tag{5.17}$$

5.1.8 Experiments

We conduct extensive experiments on four real-world datasets that contain explicit ratings: *Movielens-1M*, *Movielens-100K*, *Netflix*, and *Yahoo! Music*. For baseline comparisons, we use dynamic models, sequential models, and reinforcement learning-based models. We consider each user an episode (to maintain the learning to learn framework) for the RL setting and split users into 70% as training users and 30% as test users. For each user, we select the first $M = 10$ interacted items to represent an initial state $\mathbf{s}_0$. In the next state, we utilize the previous state representation and concatenate five items embedding from the sliding window and other five items embedding from RL agent to generate current state $\mathbf{s}_t$ by passing through the RNN module.

Evaluation metrics. We use two standard metrics to measure the recommendation performance. We also use test rewards for the RL-based methods.

- **Precision@N**: It is the fraction of the top-N items recommended in each step of the episode that are positive (rating $> \tau$) to the user. We compute the average of all test users as the final precision.
- **nDCG@N**: Normalized Discounted Cumulative Gain (nDCG) measures ranking quality, considering the relevant items within the top-N of the ranking list in each step of the RL episode.
- **Test Reward**: It measures average reward considering rewards of top-N recommended items in each step for the RL episode.

5.1.9 Recommendation Performance Comparison

Table 6.2 summarizes the recommendation performance from all models. The proposed model benefits from both the RNN module and EAC module so that it provides better results in all four datasets on both standard metrics. The dynamic and sequential models achieve less ideal performance due to their focus on short-term user interest and inability to provide long-run or future preference. RL methods have shown a clear advantage due to their focus on maximizing expected long-term rewards. Thanks to the evidence-based uncertainty exploration, DERL achieves the best performance among all DL based models.

Table 5.1: Performance of Recommendation (average P@N and nDCG@N)

Category	**Model**	**MovieLens-1M**		**MovieLens-100K**		**Netflix**		**Yahoo! Music**	
		P@5	**nDCG@5**	**P@5**	**nDCG@5**	**P@5**	**nDCG@5**	**P@5**	**nDCG@5**
Dynamic MF	timeSVD++	0.5341	0.4328	0.5034	0.4145	0.5234	0.4220	0.5267	0.4190
Sequential	CASER	0.5762	0.4613	0.5434	0.4428	0.5633	0.4542	0.5745	0.4365
	ϵ-greedy	0.5977	0.4834	0.5580	0.4556	0.5850	0.4765	0.5909	0.4812
Proposed	**DERL**	**0.6313**	**0.5365**	**0.6379**	**0.5386**	**0.6336**	**0.5372**	**0.6232**	**0.5330**

5.1.10 Conclusion

In this section, we propose a novel deep evidential reinforcement learning framework for dynamic recommendations. The proposed DERL framework learns a more effective recommendation policy by integrating both the expected reward and evidence-based uncertainty. DERL integrates a customized RNN to generate the current state that accurately captures user interest and an evidential-actor-critic module to perform evidence-based exploration to optimize policy by improving an evidential Q-value. We theoretically prove the convergence behavior of the proposed evidential policy integration strategy. Experimental results on real-world data and comparison with the state-of-the-art competitive models demonstrate the effectiveness of the proposed model.

5.2 Evidential Reinforced Attentions for Users Unique Behavioral Pattern Discovery

Machine learning-driven human behavior analysis is gaining attention in behavioral/mental healthcare, due to its potential to identify behavioral patterns that cannot be recognized by traditional assessments. Further, collections of data may be incomplete and noisy, and thus lead to a very sparse dataset. In addition, health research requires a higher level of interpretability on the analytical results as well as reasonable generalization with few examples. This makes current existing models (deep neural networks) inappropriate since they lack interpretability due to complex architecture and also required extremely large datasets to train the model.

To address these challenges, we propose a novel evidential reinforced attention that attentively selects and build sub-spatiotemporal sets. In particular, we introduce a uniquely designed reward function to encourage attention to the unknown but potentially important behavioral sub-sequences we called them as behavioral patterns. The reward simultaneously considers both the prediction accuracy for exploitation and evidence-based uncertainty estimation for unknown behavior exploration. Inspired by the task formulation in few-shot learning works, we design the training objective as a sub-trajectory classification problem. Evidence reinforce attention takes sub-trajectory data for training rather than the entire trajectory and thus is capable of processing incomplete sequential data, which is common for behavioral studies of children. As a result, informative behavioral patterns can be effectively identified (with noisy ones excluded) to ensure good interpretability along with improved predictive accuracy. We summarize our main contributions below:

- a novel end-to-end ERA model to analyze sparse, dynamic, and noisy behavioral data.
- evidential reinforced attentions to identify behavioral patterns that are both discriminative and highly interpretable.
- use of learning to learn task formulation to achieve accurate predictions with limited sparse information from incomplete sequences of behavioral data, making it more realistic and effective to support real-world behavioral studies.

5.2.1 Related Work

Machine learning driven digital behavioral biomarker discovery. In recent years, there has been a growing interest in identifying data-driven biomarkers leveraging machine learning techniques [4, 8, 48]. These biomarkers have unique advantages over traditional biomarkers such as analysis at both the individual and population level, continuous measures, and passive monitoring [4]. Lee et al. [48] leverage various machine learning approaches with putative biomarkers as an imbalance between sympathetic and parasympathetic nervous activity to predict cognitive fatigue. Further, establishing robust neuroimaging biomarkers using structural magnetic resonance imaging (MRI) with traditional machine learning mechanisms are used to diagnose and tailor treatment for ASD patients [62]. In contrast to these approaches, our model is uniquely designed to capture users' signature behavioral patterns to better differentiate different user groups.

Reinforcement learning. RL has been increasingly used to solve computer vision and natural language processing problems. For example, [60] applies reinforced visual attention to recognize important image patches for digit classification. [64] introduces a neural network model with novel intra-attention and a new training method that combines standard supervised word prediction and RL. In medical assessment, [92] proposes an RL-based synthetic sample selection method that learns to choose synthetic images containing reliable and informative features.

Our work designs a new reward function that balances classification accuracy and evidence-based exploration. Instead of performing relatively simple synthetic sample selection, we provide novel ways to achieve evidential reinforced attentions to handle complex and sparse sequential data.

5.2.2 Preliminaries

Data collection. The data used for this article were collected using multiple virtual reality (VR) games [42]. While all gaming data were collected using a similar setup, where participants sat in front of a screen-based VR, the contents were quite different, including 2D and 3D `Maze Painting`, `Word Scanning`, and `Coloring` following a template.

Evidential learning and uncertainty. Evidential learning is an evidence acquisition process where every training sample adds support to learn higher order evidential distribution [3, 70]. Given the target y_n, is drawn i.i.d. from a Gaussian distribution with unknown mean and variance (μ, σ^2) the model evidence can be introduced by further placing a prior distribution on (μ, σ^2). Leveraging Gaussian prior on the unknown mean and the Inverse-Gamma prior on the unknown variance, the posterior of (μ, σ^2) is the Normal-Inverse-Gamma (NIG) distribution.

5.2.3 Proposed Model

We aim to develop a model $\mathcal{F}$ that can accurately predict, and identify the behavioral patterns from multi-modal sequential data. In this work, we focus on behavioral patterns that can be used to effectively distinguish ASD and TD children given the recorded multimodal behavioral observations (gaze and touch) during the game-play:

$$\mathcal{F} : \{\mathbf{g}_n, \mathbf{t}_n\}_{n=1}^{N_e} \to y;\ \mathbf{g}_n \in \mathbb{R}^{M_g}, \mathbf{t}_n \in \mathbb{R}^{M_t} \tag{5.18}$$

where $y \in [0, 1], \mathcal{T} = \{\mathbf{g}_n, \mathbf{t}_n\}_{n=1}^{N_e}$ represents the entire data within an episode (trajectory). In this trajectory, $(\mathbf{g}_n, \mathbf{t}_n)$ represents n^{th} instance and N_e is the number of gaze and touch data points in the episode. Each gaze feature is M_g dimensional, each touch feature is M_t dimensional, $y = 1$ represents an ASD user, and $y = 0$ is a TD user. The length of trajectory N_e varies across the users and episodes.

Inspired by the task formulation in few-shot learning [24, 26], we design the training objective as a sub-trajectory classification problem. Specifically, we randomly sample a sub-trajectory $\mathcal{T}^s = \{\mathbf{g}_n, \mathbf{t}_n\}_{n=k}^{k+N_s}$ of length N_s ($\forall k \in [1, N_e - N_s]$) from the trajectory $\mathcal{T}$ (ignoring padding $3p$ for simplicity) and train the model to accurately identify the user group based on the limited sub-trajectory information ($\mathcal{F} : \mathcal{T}^s \to y$). This addresses the limited data problem, enables the model to train on a large number of training tasks, and encourages the model to capture multiple identifying patterns of users. Moreover, this sub-trajectory-based classification is likely to be more realistic and representative in real-world settings especially involving children.

Evidential Reinforced Attentions (ERA). As shown in Figure 5.2, we first construct the current state embedding ($\mathbf{e}_t$) by concatenating the embedding network generated embedding $\mathbf{d}$ and the RL-agent selected attentive subset embedding $\mathbf{d}_{attn}^t$:

$$\mathbf{e}_t = \texttt{concat}(\mathbf{d}, \mathbf{d}_{attn}^t) \tag{5.19}$$

We leverage a state encoder (SE), which takes the current state embedding ($\mathbf{e}_t$) and previous state ($\mathbf{s}_{t-1}$) to generate current state-space ($\mathbf{s}_t$) as:

$$\mathbf{s}_t = \texttt{SE}(\mathbf{e}_t, \mathbf{s}_{t-1}; \theta_{se}) \tag{5.20}$$

We develop an evidential policy network (π_{θ_e}) parameterized by θ_e, which takes the $\mathbf{s}_t$ as an input and output evidential distribution parameters ($\gamma, \nu, \alpha, \beta$). Then, the likelihood of choosing an action, $\mathbf{a}_t$ is obtained by marginalizing over the prior parameters (μ, σ^2):

$$\begin{aligned} p(\mathbf{a}_t|\gamma, \nu, \alpha, \beta) &= \int_{\sigma^2} \int_{\mu} p(\mathbf{a}_t|\mu, \sigma^2) p(\mu, \sigma^2|\gamma, \nu, \alpha, \beta) \mathrm{d}\mu \mathrm{d}\sigma^2 \\ &= \mathrm{St}\left(\mathbf{a}_t; \gamma, \beta(1+\nu)/(\nu\alpha), 2\alpha\right) \end{aligned} \tag{5.21}$$

where St($\mathbf{a}_t; \gamma, \beta(1 + \nu)/(\nu\alpha), 2\alpha$) is the Student-t distribution with location, scale, and degrees of freedom respectively, which is achieved by placing a NIG evidential prior on a Gaussian likelihood.

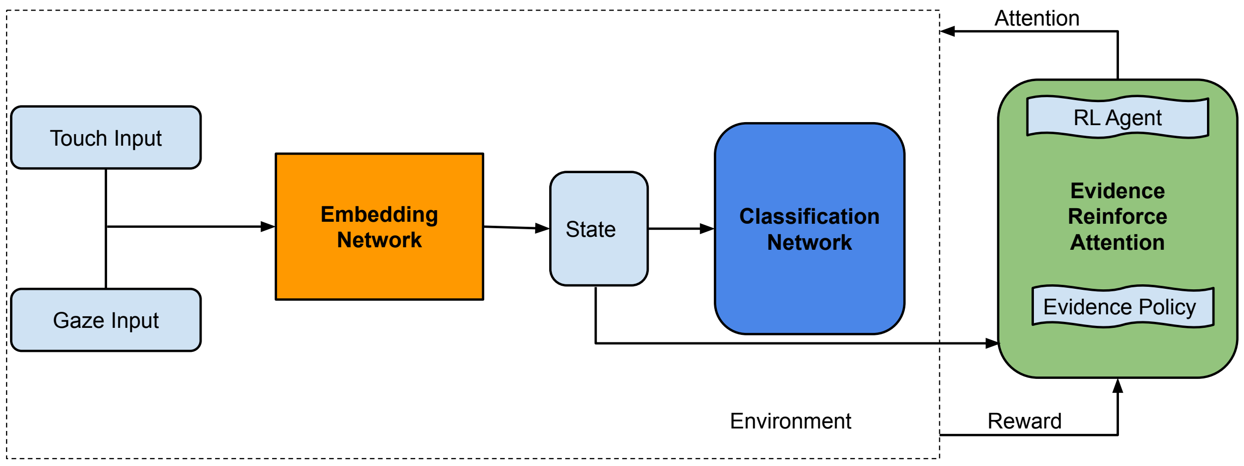


Figure 5.2: Overview of the proposed framework

From this Student-t distribution, we sample an action $\mathbf{a}_t$ that provides the attentive location in temporal-set and directs the policy update, respectively. It should be noted that our action is a continuous vector of length N_a representing all the starting position of an attention window.

Specifically, for each attention $a_t^k (k \in [1, N_a])$, we apply a sigmoid function (σ) and multiply it with length of $(N_s - W)$, where W represents the size of attention window. We then generate the attention starting index using a floor function:

$$idx_t^k = \lfloor \sigma(a_t^k) \cdot (N_s - W) \rfloor \tag{5.22}$$

We construct an all-zero mask M_t of length N_s and then flip the entries indexing in $[idx_t^k, idx_t^k + W], \forall k \in [1, N_a]$ to 1. The RL selected attentive subset embedding in time step t $\mathbf{d}_{attn}^t$ is then obtained by $M_t \cdot \texttt{concat}(\Phi_t(\mathbf{h}^i), \Phi_s(\mathbf{h}^i))$, where $\cdot$ symbolizes dot product function.

We design a novel evidential reward function that incorporates standard RL reward computed with predicted classification accuracy and epistemic uncertainty that captures policy network's uncertainty while providing an action:

$$\begin{aligned} r^e(\mathbf{s}_t, \mathbf{a}_t) &= r(\mathbf{s}_t, \mathbf{a}_t) + \lambda \texttt{epistemic}(\pi_{\theta_e}(\cdot|\mathbf{s}_t)) \\ r(\mathbf{s}_t, \mathbf{a}_t) &= 1\{p_T = y_s\} \end{aligned} \tag{5.23}$$

where $r(\mathbf{s}_t, \mathbf{a}_t)$ is a predictive reward representing the classification accuracy at last time step T, p_T is the last time step's predicted result while y_s is the user category label corresponding to sub trajectory $\mathcal{T}^s$, and $\texttt{epistemic}(\pi_{\theta_e}(\cdot|\mathbf{s}_t)) = \text{Var}[\mu] = \frac{\beta}{\nu(\alpha-1)}$ is the epistemic uncertainty.

Given the evidential reward, we introduce an epistemic value function, $V^e(\mathbf{s}_t)$, which can be computed by repeatedly applying the Bellman operator (B^π):

$$B^\pi V^e(\mathbf{s}_t) r^e(\mathbf{s}_t, \mathbf{a}_t) + \gamma_{RL} \mathbb{E}_{\mathbf{s}_{t+1} \sim \pi}[V(\mathbf{s}_{t+1})] \tag{5.24}$$

The detailed workflow of the ERA module is presented in Figure 5.2. The module takes $\mathbf{s}_t$ as input to the evidential policy network that generates evidential distribution parameters. We further marginalize those parameters and achieve a predictive student-t distribution and from which we sample an action $\mathbf{a}_t$. We generate attention masks based on the provided action and then select attentive gaze and touch embeddings and compute evidential reward simultaneously.

5.2.4 Derivation of Epistemic Policy Iteration

We derive epistemic policy iteration to achieve optimal policy by alternating between epistemic policy evaluation and epistemic policy improvement.

Lemma 5 (Epistemic Policy Evaluation) *Given the Bellman operator B^π in* (5.24) *and $V^{n+1} = B^\pi V^n$, the value will converge to the epistemic value of policy π as $n \to \infty$.*

Proof Given the evidential reward defined as $r^e(\mathbf{s}_t) = r_\pi(\mathbf{s}_t) + \lambda \texttt{epistemic}_\pi(.|\mathbf{s}_t))$ the update rule for epistemic value can be written as:

$$V^e(\mathbf{s}_t) = \mathbb{E}_\pi \sum_{t'=t}^{\infty} \gamma^{t'} r^e(\mathbf{s}_{t'}) = r^e(\mathbf{s}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}}[V^e(\mathbf{s}_{t+1})] \tag{5.25}$$

Following the convergence rule [76] with finite action space, it is guaranteed that the value will converge to the epistemic value of policy π.

Lemma 6 (Epistemic Policy Improvement) *Given a new policy π_{new} that is updated via* (5.33)*, then $V^e_{\pi_{new}}(\mathbf{s}_t) \geq V^e_{\pi_{old}} \mathbf{s}_t$ for all $\mathbf{s}_t$.*

Proof The policy can be updated towards the new value function. Consider the updated policy π_{new} as the optimizer of the maximization problem.

$$\pi_{new} = \operatorname*{argmax}_{\pi\prime} J_\pi(\phi) = \operatorname*{argmax}_{\pi'} \mathbb{E}_{\mathbf{s}_t}[V^e_{\pi'}(\mathbf{s}_t)] \tag{5.26}$$

Denote the old policy as π_{old}. Using the update rule specified in Eq (5.33) with a sufficiently small step size, we get an updated policy π_{new} that satisfies

$$\mathbb{E}_{\mathbf{a}_t \sim \pi_{new}}[V^e_{\pi_{old}}(\mathbf{s}_t)] \geq \mathbb{E}_{\mathbf{a}_t \sim \pi_{old}}[V^e_{\pi_{old}}(\mathbf{s}_t) \tag{5.27}$$

Given Eq (B.3), we have the following inequality

$$\begin{aligned} V^e_{\pi_{old}}(\mathbf{s}_t) \leq & r^e(\mathbf{s}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \sim \pi_{new}}[V^e_{\pi_{old}}(\mathbf{s}_{t+1})] \\ \leq & r^e(\mathbf{s}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \sim \pi_{new}}[r^e(\mathbf{s}_{t+1})] \\ & + \mathbb{E}_{\mathbf{s}_{t+2}, \mathbf{a}_{t+2} \sim \pi_{new}}[V^e_{\pi_{old}}(\mathbf{s}_{t+2})] \\ & \ldots \\ = & V^e_{\pi_{new}}(\mathbf{s}_t) \end{aligned} \tag{5.28}$$

where $r^e(\mathbf{s}_t)$ is a evidential reward in step t. Therefore, we show that the new policy π_{new} ensures $V^e_{\pi_{new}}(\mathbf{s}_t) \geq V^e_{\pi_{old}}(\mathbf{s}_t)$ for all $\mathbf{s}_t$.

Theorem 7 (Epistemic Policy Iteration) *Alternating between epistemic policy evaluation and epistemic policy improvement for any policy $\pi \in \Pi$ converges to an optimum epistemic policy π^* such that $V^{\pi^*}(\mathbf{s}_t) \geq V^e_\pi(\mathbf{s}_t)$ for all $\mathbf{s}_t$.*

Proof Let π_i denote the policy at iteration i. We already show that the sequence $V^e_{\pi_i}(\mathbf{s}_t)$ is monotonically increasing. Since $V^e_\pi(\mathbf{s}_t)$ is bounded above, the sequence converges to some π^*. At convergence, it must be the case that $J_{\pi^*}(\pi^*(.|\mathbf{s}_t)) \leq J_{\pi^*}(\pi(.|\mathbf{s}_t))$ for $\pi \neq \pi^*$. Based on Lamma 11, we have $V^e_{\pi^*}(\mathbf{s}_t) > V^e_\pi(\mathbf{s}_t)$ for all $\mathbf{s}$. In other words, the evidence value of any other policy π is lower than that of the converged policy π^*. Therefore, it guarantees convergency to an optimal policy π^* such that:

$$V^e_{\pi^*}(\mathbf{s}_t) \geq V^e_\pi(\mathbf{s}_t) \tag{5.29}$$

5.2.5 Training and Inference

The training procedure involves the parameter update associated with both classification and ERA modules. The classification module is trained with supervised learning utilizing binary cross-entropy loss as:

$$\mathcal{L}(\theta_{NN}) = -\frac{1}{T}\sum_{t=1}^{T} y_s \cdot \log(\mathtt{NN}(\mathbf{s}_t)) + (1 - y_s) \cdot \log(1 - \mathtt{NN}(\mathbf{s}_t)) \tag{5.30}$$

where NN is a binary classifier neural network, and y_s is a ground truth label. The classification network is updated as:

$$\theta_{NN} \longleftarrow \theta_{NN} - \eta_{NN} \nabla_{\theta_{NN}} \mathcal{L}(\theta_{NN}) \tag{5.31}$$

Similarly, the ERA module finds the optimal policy to maximize long-term expected cumulative evidential reward as:

$$J_\pi(\theta_e) = \sum_{t=1}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \pi} (r^e_\pi(\mathbf{s}_t, \mathbf{a}_t)) \tag{5.32}$$

where T is the total number of time steps in the episode.

For ERA module update, we update the evidential policy network with policy gradient method:

$$\theta_e \longleftarrow \theta_e + \eta_e \nabla_{\theta_e} J_\pi(\theta_e) \tag{5.33}$$

where $\nabla_{\theta_e} J_\pi(\theta_e)$ is proportion to

$$\mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \pi} [r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) \nabla_{\theta_e} \ln \mathrm{St}(\mathbf{a}_t; \pi_{\theta_e}(\cdot|\mathbf{s}_t))]$$

During inference, we input the model with variable length sub-trajectories or full episode trajectories to test the model's capability in classifying the provided trajectory belonging to ASD or TD users. We provide details of the inference process in the experimental section.

5.2.6 Experiments

In this section, we conduct quantitative experiments to evaluate the proposed ERA model on three different games regarding ASD detection, `Maze Painting`, `Word Scanning`, and `Coloring`. First, we compare our model with multiple state-of-the-art baselines. The performance comparison shows that our model surpasses all the competitors by a significant margin.

Comparison baselines. We compare with multiple state-of-the-art baselines, all of which are trained using the same sub-trajectory-based task formulation:

- **Recurrent Classifier** (LSTM) [34]: We consider a LSTM based classifier as a simple recurrent based baseline that leverages the multimodal sequential information for classification. The model uses two sequence encoders to generate the task representation. Specifically, the output from the final time step is concatenated to obtain the task representation.

Table 5.2: Classification Performance Comparison

Dataset	Maze-2D	Maze-3D	Maze-Mixed	Coloring	Word Scanning
LSTM [34]	65.0±6.7	63.0±5.5	68.8±4.8	62.0±3.8	62.5±4.1
LSTM-FCN [40]	91.5±3.6	93.4±3.5	90.4±3.8	86.0±4.2	89.0±4.1
ERA	**94.0±3.8**	**95.3±5.3**	**95.0±5.2**	**91.0±3.9**	**93.5±3.8**

- **LSTM-FCN** [40]: LSTM-FCN explores the augmentation of fully convolutional networks with LSTM RNN sub-modules for time series classification.

Classification performance. We compare ERA's predictive accuracy with the baselines on the 2D, 3D and mixed (including both 2D and 3D) game data for `Maze Painting` as well as the other two games in Table 5.2. As can be seen, ERA outperforms all baselines by a significant margin. Working across all these real-world datasets further demonstrates our model's generalization capability in human behavior analysis.

5.2.7 Conclusion

In this section, we prosposed evidential reinforced attention to discover subtle and complex multimodal human behavioral patterns. The proposed ERA model leverages RL agent to perform evidential reinforced attention that learns an effective policy to select representative embeddings as attention signatures and further boosts the performance. Experimental results on multiple real-world datasets demonstrate the effectiveness of the proposed model.

Chapter 6

Evidential Stochastic Differential Equation for Sparse User Interactions

In this chapter, we focus on dealing with the problem of noisy sparse interaction and provide a systematic learning method extending the L2L framework with evidential learning and stochastic differential equation that captures users' preference change over time leveraging continuous architecture i.e. neural stochastic differential equations (NSDE), and then integrate with evidential learning framework to recommend both important and diverse items for effective recommendation.

6.1 Evidential Stochastic Differential Equation for Sequential Recommendation

Various sequential recommendation models [74, 83, 88, 90] have been proposed to capture dynamics in user behaviors. They mainly leverage the user's sequential historical interactions and aim to predict the next item that a user like to interact with. Although these methods attempt to infer high-quality user representation with sequential interaction information, they assume a uniform time interval among consecutive user interactions and are insufficient to capture users' continuously evolving behavior. In reality, the actual time intervals vary dramatically. This may significantly impact the recommendation performance in practice.

To demonstrate the challenge as outlined above, we provide an example in Table 6.1, where we follow the standard sequential recommendation similar to [39, 74] by predicting the next item considering other 100 negative items from the item pool. We select a user (ID:13) from the Movielens-100K dataset and provide

Table 6.1: Ranking of the ground-truth item of the next interaction in a sequence for a user (ID:13) considering the gap between two consecutive interactions

Interaction	**Gap (Seconds)**	**Ranking**	
		BERT4Rec	**E-NSDE**
6 to 7	44	4	4
13 to 14	623,591	24	16
116 to 117	62	6	3
150 to 151	896,291	56	18

recommended ranking results together with the interaction time gap in the sequence. The recommendation performance for a sequential model BERT4Rec deteriorates significantly when the gap between the consecutive interactions becomes large. For example, when the next interaction occurs soon after the first one, BERT4Rec consistently ranks the ground-truth item in a top-10 list. However, for a much larger gap, the ground-truth item drops out of the top-20 or even top-50 recommendation list.

To address the issues above, few recent efforts [14, 29, 77] have adopted the concept of Neural Ordinary Differential Equations (NODE) that maps the existing discrete neural network to a continuous model, which provides the ability to capture users' continuously evolving preferences. Although these methods leverage the power of NODE and provide richer representation, they lack the capability to learn model uncertainty, which could be essential to understand the user behavior when recommending the next item.

There is a variant of NODE called Neural Stochastic Differential Equations (NSDE), which adds Brownian motion terms to incorporate stochasticity via a diffusion function. NSDE has been successfully applied in the field of computer vision [43, 89] to model uncertainty. Inspired by the idea of NSDE, we propose to use NSDE as the sequential recommendation model that not only helps to capture users' continuously changing behavior but also model stochasticity in the user and item's evolving representation. While NSDE can model the noise due to stochasticity in user and item representations, it does not capture the uncertainty in user-item interactions. To fill this critical gap, we further incorporate evidential deep learning (EDL) [3, 70], aiming to capture model uncertainty by gathering evidence from user-item interactions. The proposed E-NSDE seamlessly integrates two major components: an NSDE module and an EDL module to capture continuous user preferences over time and provide important and diverse items for an effective recommendation. The NSDE module is responsible for learning user and item representations over time. Furthermore, the EDL module utilizes this richer representation of users and items to identify important and diverse items that the model needs to learn to capture users' actual behavior with the help of uncertainty-aware exploration.

The main contribution of this work is fourfold:

- a novel recommendation model that integrates neural stochastic differential equations with evidential learning to provide uncertainty-aware sequential recommendations,
- leveraging interaction and time-guided evidential uncertainty to maximize information gain through exploration of a large item pool,
- using a monotonic network to ensure a mathematical relationship between interaction time gap and predicted uncertainty,
- an end-to-end integrated training process with seamless integration of NSDE and EDL modules.

6.2 Related Work

For traditional baselines like sequential models and graph models please refer to Chapter 2.

Neural ODE in Recommender Systems. Neural ODE solver has also been used in recommender systems [14, 77]. The learnable time ODE-based collaborative filtering [14] redesigns linear graph convolution networks on top of the NODE that learns the optimal architecture and smooth ODE solutions for effective collaborative filtering. Similarly, [77] utilizes meta-learning enhanced neural ODE for citywide next POI Recommendation. Basically, it models city-invariant information and city-specified information separately to achieve accurate citywide next POI recommendation.

6.3 Preliminaries

Problem Formulation. The user set (U) and item set (I) are the two sets of input data for a recommendation model, respectively, where $|U|$ and $|I|$ denote the total number of users and items. We represent a user $u_t \in U$ and item $i_t \in I$ at time t. In a sequential recommendation setting, users' behavior sequences are chronological order and hence represent interaction sequences for user u as [i_0,i_1,...,i_{t-1},i_t] at time t. We perform recommendation and uncertainty quantification for each user with a recommendation function as:

$$f_\Theta(u_t, i_t) = \{\gamma_{(u_t,i_t)}, \nu_{(u_t,i_t)}, \alpha_{(u_t,i_t)}, \beta_{(u_t,i_t)}\} \\ \forall u_t \in U, i_t \in I \tag{6.1}$$

where $\gamma_{(u_t,i_t)}$ is the recommended score for item i assigned by user u, $\nu_{(u_t,i_t)}$, and $\alpha_{(u_t,i_t)}$ are the model evidence, and $\beta_{(u_t,i_t)}$ is a total uncertainty coming from both pseudo and actual data samples at time t.

Evidential Learning. Evidential learning is an evidence acquisition process where every training sample adds support to learn higher order evidential distribution [3, 70]. Given the target y_n, is drawn i.i.d. from a Gaussian distribution with unknown mean and variance (μ, σ^2) the model evidence can be introduced by further placing a prior distribution on (μ, σ^2). Leveraging Gaussian prior on the unknown mean and the Inverse-Gamma prior on the unknown variance, the posterior of (μ, σ^2) is the Normal-Inverse-Gamma (NIG) distribution. We choose a Gaussian prior on the unknown mean and an Inverse-Gamma prior on the unknown variance to ensure conjugacy as:

$$\begin{aligned} &p(y_n|\mu, \sigma^2) = \mathcal{N}(\mu, \sigma^2),\ p(\mu|\gamma, \sigma^2\nu^{-1}) = \mathcal{N}(\gamma, \sigma^2\nu^{-1}) \\ &p(\sigma^2|\alpha, \beta) = \text{Inv-Gamma}(\alpha, \beta) \end{aligned} \tag{6.2}$$

where Inv-Gamma$(z|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)}\left(\frac{1}{z}\right)^{\alpha+1}\exp(-\frac{\beta}{z})$ with $\Gamma(.)$ being a gamma function; $\mathbf{m} = (\gamma, \nu, \alpha, \beta)$ are parameters of the corresponding prior distributions. The posterior of (μ, σ^2) follows a Normal Inverse-Gamma (NIG):

$$p(\mu, \sigma^2|\mathbf{m}) = \frac{\beta^\alpha\sqrt{\nu}}{\Gamma(\alpha)\sqrt{2\pi\sigma^2}}\left(\frac{1}{\sigma^2}\right)^{\alpha+1}\exp\left\{-\frac{2\beta + \nu(\gamma-\nu)^2}{2\sigma^2}\right\} \tag{6.3}$$

Given a NIG posterior, one can derive the mean ($\mathbb{E}[\mu]$), aleatoric ($\mathbb{E}[\sigma^2]$) and epistemic (Var$[\mu]$) uncertainty as:

$$\mathbb{E}[\mu] = \gamma,\ \mathbb{E}[\sigma^2] = \frac{\beta}{\alpha-1},\ \text{Var}[\mu] = \frac{\beta}{\nu(\alpha-1)} \tag{6.4}$$

Neural Ordinary Differential Equations (NODE) Ordinary Differential Equations (ODEs) are used to model continuous-time hidden dynamics in neural networks [12] and can be defined as:

$$dh_t = f_\psi(h_t, t)dt, \quad h_0 \in \mathcal{R}^d \tag{6.5}$$

where $f(.)$ is a neural network with parameter ψ and h_0 is an initial value.

Further, leveraging Eq (6.5) and integrating these dynamics forward, one can compute $\mathbf{h}(t_{i+1})$ from $\mathbf{h}(t_i)$ by solving the following Riemann integral problem:

$$\mathbf{h}(t_{i+1}) = \mathbf{h}(t_i) + \int_{t_i}^{t_{i+1}} f(\mathbf{h}(t_i), t; \psi)dt \tag{6.6}$$

Neural Stochastic Differential Equations (NSDE) We could view Stochastic Differential Equations (SDE) as an ODE with infinitesimal noise added throughout time:

$$dh_t = f_\psi(h_t, t)dt + g_\omega(h_t, t)dB_t \tag{6.7}$$

where $f(.)$ and $g(.)$ are drift and diffusion functions respectively, B_t is a Brownian motion.

Similar to NODE, we are also able to compute the forward dynamics of NSDE i.e. $\mathbf{h}(T)$ from initial value $\mathbf{h}(t_0)$ integrating Eq (6.7) as:

$$h(T) = h(t_0) + \int_{t_0}^{T} f(h(t), t; \psi)dt + \int_{t_0}^{T} g(h(t), t; \omega)dB_t \tag{6.8}$$

6.4 E-NSDE: Evidential Stochastic Differential Equation for Recommendation

Overview. We propose a novel time-aware sequential recommendation model as shown in Figure 6.1. The model includes a Neural Stochastic Differential Equations (NSDE) to capture continuous time-evolving user dynamics and an evidential module to capture uncertainty in user-item interactions and also to provide uncertainty-aware exploration leveraging interaction time gap. The NSDE module takes the initial representations of users and items, and uses the interaction time gap to generate refined user and item representations. Subsequently, these improved user and item representations are fed into the EDL module. Within the EDL module, the rating network generates a rating score, while the monotonic network produces evidential parameters that incorporate the interaction time gap, establishing a direct link to the model's predicted uncertainty. Our approach adheres to the conventional sequential training strategy and incorporates supervised signals derived from evidential learning. We delve into detailed discussions in the subsequent sections.

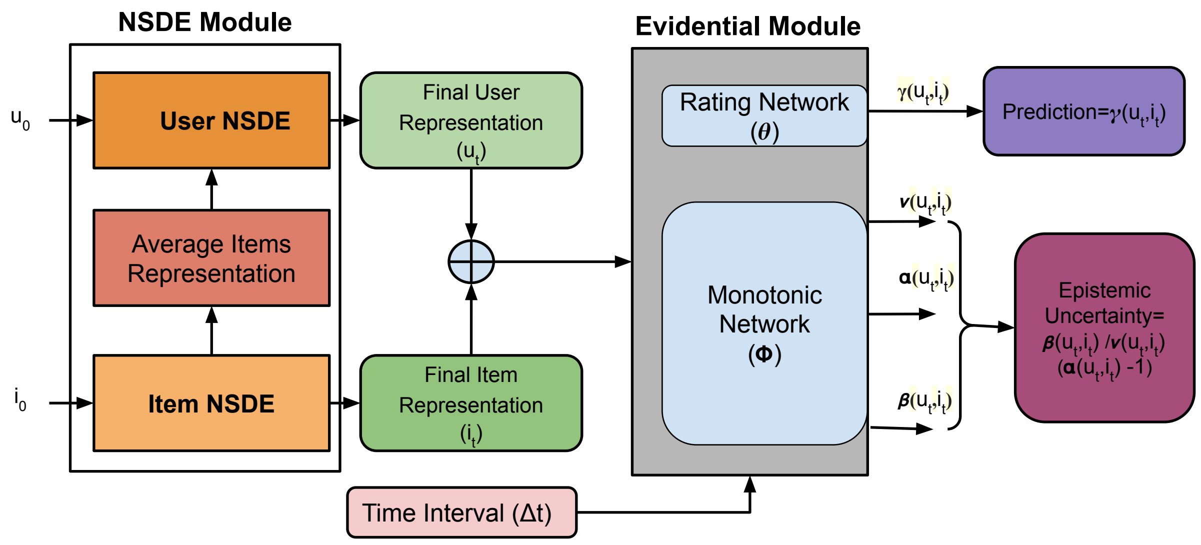


Figure 6.1: Overview of E-NSDE framework. The framework includes user and item NSDE modules to generate the final user and item representation and an EDL module to provide an uncertainty-aware prediction.

6.4.1 NSDE Based User and Item Representations

The Neural Stochastic Differential Equations (NSDE) include two key components: drift and diffusion functions. The drift component captures the evolving nature in the system and the diffusion component captures stochasticity in the system. The proposed NSDE for recommender system advances contemporary sequential models from the following aspects: 1) existing methods require partitioning the time into uniform intervals to support model training and inference, while NSDE removes this requirement, providing additional flexibility; 2) the existing methods largely negate stochasticity in the system, but the NSDE incorporates it into the form of inherent noise. This suggests a better fit of the NSDE into sequential recommendation and essentially supports generating richer user and item representations leveraging the power of the SDE solver to capture users' continuously evolving preferences over time than previous discrete sequential and deterministic ODE methods.

The user fine-grained representation processes based on NSDE formulation can be written following the Eq (6.8):

$$u(T) = u(t_0) + \int_{t_0}^{T} f(u(t), t; \psi)dt + \int_{t_0}^{T} g(u(t), t; \omega)dB_t \tag{6.9}$$

where $u(t_0)$ represents the user's initial representation which is computed with aggregation of a few initial interacted items (i.e., $u(t_0) = agg(i_o, i_1, ..i_k)$), and T represents the final time.

For the NSDE formulation in recommender systems, we intuitively show the role of both drift and diffusion components. We thought of the drift component as a user's internal evolution and diffusion component as an inherent noise that occurs when interacting with environments.

We leverage the diffusion function with Brownian motion to capture the user's inherent noise over each interaction. This term is crucial in SDE to capture stochasticity in the system. By capturing stochasticity, we incorporate the impact of noisy user-item interactions. As it includes the Brownian motion for stochasticity, we relate this in recommender systems to incorporate noise considering the time interval of interaction. Here we first define the basic definition and properties of the Brownian motion and then mention the detail of its applicability in the recommender systems.

Lemma 8 (Standard Brownian Motion) *A standard Brownian motion B_t is a stochastic process that satisfies the following properties: a) B_t-B_s is normally distributed with zero mean and (t-s) variance, i.e., $\mathcal{N}$(0,t-s) for all t $\geq s \geq 0$; b) For every pair of disjoint time intervals $[t_1, t_2]$ and $[t_3, t_4]$, with t_1<$t_2 \leq t_3 \leq t_4$, the increments i.e., B_{t_2}-B_{t_1} and B_{t_4}-B_{t_3} are independent random variables.*

Theorem 9 *A larger the interaction time gap (t_2-t_1) incurs a higher uncertainty in user interest.*

Given the Lemma 8, we can show that the increase in interaction time gap (t_2-t_1) increases the chance the user may deviate from the current interest. The last term of Eq (6.9) shows that if there is a long time gap in the interaction, there is usually a larger deviation in the user interest:

$$\int_{t_1}^{t_2} g(u(t), t; \omega) dB_t \propto B_{t_2} - B_{t_1} \tag{6.10}$$

Due to Lemma 8, the Brownian interval B_{t_2}-B_{t_1} is going to increase, which also increases the noise accumulation. It indicates that if there is a longer interaction gap, then recommending the next item is relatively uncertain due to the user's less involvement in the current time.

Given this Theorem 9, the final term in Eq. (6.9) holds the user's time deviation and its impact in increasing large variance or noise in the system. Further, the first term captures the user's initial representation with some interactions, and the second drift component provides the user's evolving interest. Considering all of these three components, the SDE solver captures the user's richer representation over time.

Similarly, NSDE-based item representation processes can be formulated as:

$$i(T) = i(t_0) + \int_{t_0}^{T} f(i(t), t; \psi) dt + \int_{t_0}^{T} g(i(t), t; \omega) dB_t \tag{6.11}$$

where, $i(t_0)$ represents item's initial representation

6.4.2 Evidential Module

We leverage an evidential learning technique to provide an uncertainty-aware model prediction for effective recommendations. The evidential module consists of two key networks: `Rating Network`, and `Monotonic Network`.

Rating Network. The rating network utilizes the fine-grained user u_t and item i_t representations from the NSDE and outputs the predicted score $\gamma_{(u_t,i_t)}$ of the corresponding user item interactions. We adopt an evidential loss as the marginal likelihood while computing the predicted loss. This includes the negative log-likelihood ($\mathcal{L}^{NLL}[f_\Theta]$) to maximize the marginal likelihood and an evidential regularizer ($\mathcal{L}^{R}[f_\Theta]$) to impose a high penalty on the predicted error with low uncertainty (i.e., high confidence). We first formulate the negative log-likelihood, given by

$$\mathcal{L}^{NLL}[f_\Theta(u_t,i_t)] = -\log(p(r_{(u_t,i_t)}|\mathbf{m}(u_t,i_t)) \tag{6.12}$$

where, $\mathbf{m}(u_t,i_t) = (\gamma_{(u_t,i_t)}, \nu_{(u_t,i_t)}, \alpha_{(u_t,i_t)}, \beta_{(u_t,i_t)})$ are model parameters at time t, and $p(r_{(u_t,i_t)}|\mathbf{m}(u_t,i_t))$= $St(r_{(u_t,i_t)}; \gamma_{(u_t,i_t)}, \frac{\beta_{(u_t,i_t)}(1+\nu_{(u_t,i_t)})}{\nu_{(u_t,i_t)}\alpha_{(u_t,i_t)}}, 2\alpha_{(u_t,i_t)})$ is a student t-distribution acquired after placing a NIG evidential prior on Gaussian likelihood function. We formalize our own evidence regularizer, which considers epistemic uncertainty to penalize confidently predicted errors. We multiply the predicted error with the inverse epistemic uncertainty that scales up the error, which encourages high inverse epistemic uncertainty when the predicted evidence is high (and vice-versa). Conversely, it will be less penalized if the prediction is close to the target score:

$$\mathcal{L}^{R}[f_\Theta(u_t,i_t)] = |r_{(u_t,i_t)} - \gamma_{(u_t,i_t)}| \cdot \left(\frac{\nu_{(u_t,i_t)}(\alpha_{(u_t,i_t)}-1)}{\beta_{(u_t,i_t)}}\right) \tag{6.13}$$

The regularized EDL loss for each sequential update is given as:

$$\mathcal{L}_{EDL}(u_t,i_t) = \mathcal{L}^{NLL}[f_\Theta(u_t,i_t)] + \lambda\mathcal{L}^{R}[f_\Theta(u_t,i_t)] \tag{6.14}$$

where λ is a regularization coefficient.

Monotonic Network. We adopt the concept of a monotonic network [72] in the context of building the relationship between the interaction time gap and model uncertainty. Intuitively, the monotonic network is designed in such a way that the increase in input, i.e., time interval (Δt), increases the output, i.e., the variance of the predicted rating. The variance or epistemic uncertainty is computed as given by Eq (6.4). For this, the nominator term, i.e., total uncertainty $\beta_{(u_t,i_t)}$ should need to be increased with the increase in Δt, and the denominator terms, i.e., pseudo-observations ν and α should need to be decreased with the increase in Δt. We theoretically show this intuition in the following theoretical section and showed a mathematical relation. We maintain this by performing exponential transformation of the network weights as:

$$\phi = \exp\left(\phi_{init}\right) \tag{6.15}$$

where ϕ_{init} is the network's initial weight. To ensure increasing monotonicity, we assign all positive weights to the network layers. Similarly, to decrease the monotonicity, we assign negative weights to the last layer and positive weights to other layers of the network. We update the network utilizing the total loss similar to the rating network.

Lemma 10 (Monotonic increase of total uncertainty β) *Given an increased time interval Δt and positive network weights W, the output i.e. total uncertainty $\beta_{(u_t,i_t)}$ of the evidential monotonic network increases monotonically.*

Please refer to Appendix C.2.1.

Lemma 11 (Monotonic decrease of pseudo-observations α and ν) *Given an increased time interval Δt and negative last layer network weight (W_L), and other layer weights are positive $W_{0,\ldots,L-1}$ the output i.e. pseudo-observations $\alpha_{(u_t,i_t)}$, and $\nu_{(u_t,i_t)}$ of the evidential monotonic network decreases monotonically.*

Please refer to Appendix C.2.2.

Theorem 12 *(An increase in time interval (Δt) results in an increase in epistemic uncertainty $Var[\mu]$). Given the Lemma 10 and 11, an increase in the input time interval (Δt) of the evidential monotonic network increases the epistemic uncertainty $Var[\mu]$.*

The epistemic uncertainty equation from Eq (6.4):

$$\mathcal{U}_{(u_t,i_t)} = \text{Var}[\mu] = \frac{\beta_{(u_t,i_t)}}{\nu_{(u_t,i_t)}(\alpha_{(u_t,i_t)} - 1)}$$

Given the increase $\beta_{(u_t,i_t)}$ from Lemma 10 and decrease in $\alpha_{(u_t,i_t)}$ and $\nu_{(u_t,i_t)}$ from Lemma 11 the nominator of the epistemic uncertainty increases, and the denominator decreases. This proves that the increase in the time interval (Δt) increases the epistemic uncertainty var$[\mu]$ of the evidential monotonic network. We enforced the constraints on ($\beta_{(u_t,i_t)}, \alpha_{(u_t,i_t)}, \nu_{(u_t,i_t)}$) with a `soflplus` activation (and additional +1 added to $\alpha_{(u_t,i_t)}$ since $\alpha_{(u_t,i_t)} > 1$).

Interpreting Hyper-parameters in the context of Recommender Systems. Besides serving as the parameters of the evidential prior distributions, the hyper-parameters $(\nu_{(u_t,i_t)}, \alpha_{(u_t,i_t)}, \beta_{(u_t,i_t)})$ offer very intuitive meanings. First, both $\nu_{(u_t,i_t)}$ and $\alpha_{(u_t,i_t)}$ are essentially the 'pseudo' prior observations, and their posterior can be treated as the *evidence* to support a prediction. In the context of the recommendation, their relation with time interval is inverse, because the large time gap causes a decrease in the number of pseudo items, as mentioned in Lemma 3. Second, the $\beta_{(u_t,i_t)}$ hyperparameter combines total uncertainty from pseudo samples and observed data. Lemma 2 shows that an increase in time interval will result in an increase in the uncertainty (due to a smaller number of pseudo and interacted items to the user), and therefore the model is less confident to provide an accurate prediction.

Weighted Bayesian Personalized Ranking (WBPR) Loss. To leverage the effective exploration for the long-term, we formulate weighted BPR loss which is computed from non-interacted items i.e. negative items $(j_t \in \mathcal{N}_t)$ which are similar to the user's future interacted positive items. We first select similar negative items from the user non-interacted item pool and then leverage cosine similarity with future positive item embeddings. Further, the model provides uncertainty-aware predicted rating score for those negative items leveraging both rating and monotonic network output as:

$$\hat{r}_{(u_t,j_t)} = \gamma_{(u_t,j_t)} + \eta \mathcal{U}_{(u_t,j_t)} \tag{6.16}$$

where j_t represents non-interacted items at time t and η is scalar to control the influence of epistemic uncertainty.

We then compute weight coefficients based on uncertainty-aware predicted scores with cosine similarity $CosineSim(.)$ as:

$$w_{(i_t,j_t)} = \begin{cases} \texttt{max(CosineSim(f\_emb, j\_emb))}, \text{ if } \hat{r}_{(u_t,j_t)} > \tau \\ \texttt{min(CosineSim(f\_emb, j\_emb))}, \text{ otherwise} \end{cases}$$

where f_emb, j_emb are future and negative item embedding respectively. We then formulate weighted BPR loss utilizing a negative log-likelihood function as:

$$\mathcal{L}_{WBPR}(u_t, i_t) = \sum_{(u_t, i_t, j_t \in \mathcal{N}_t)} w_{(i_t,j_t)} \times (-\ln(\sigma(\hat{r}_{(i_t,j_t)}))) \tag{6.17}$$

where $\hat{r}_{(i_t,j_t)} = \hat{r}_{(u_t,i_t)} - \hat{r}_{(u_t,j_t)}$, $\sigma(.)$ is the sigmoid.

Remark. The intuition behind this weighted BPR formulation is to learn effective exploration by providing higher weight to the non-interacted items which are quite similar to user future positive items so that model can learn quickly a diverse and wider range of user-evolving behavior to benefit in the future.

Table 6.2: Performance of Recommendation (average P@N, and nDCG@N)

Category	**Model**	**MovieLens-100K**		**Movielens-1M**		**Netflix**	
		P@5	**nDCG@5**	**P@5**	**nDCG@5**	**P@5**	**nDCG@5**
Dynamic MF	timeSVD++	0.4010±0.015	0.3420±0.013	0.3917±0.016	0.3508±0.013	0.3756±0.016	0.2951±0.013
	CKF	0.4053±0.017	0.3620±0.013	0.3928±0.016	0.3552±0.015	0.3600 ±0.017	0.2986±0.014
Graph	NGCF	0.4059±0.014	0.3662±0.012	0.3978±0.016	0.3587 ±0.018	0.3574±0.015	0.3167±0.017
	LightGCN	0.4103±0.014	0.3702±0.013	0.4028±0.017	0.3632±0.015	0.3617±0.013	0.3204±0.016
	CASER	0.4096 ±0.012	0.3663±0.015	0.4021±0.014	0.3626±0.016	0.3658 ±0.013	0.3189±0.012
Sequential	SASRec	0.4105±0.013	0.3740 ±0.011	0.4112±0.015	0.3708±0.017	0.3746±0.012	0.3257±0.014
	BERT4Rec	0.4149±0.014	0.3781±0.011	0.4163±0.012	0.3754 ±0.013	0.3793±0.011	0.3295±0.013
	S^3-Rec	0.4124 ±0.012	0.3755±0.014	0.4134±0.013	0.3715 ±0.014	0.3786±0.016	0.3274±0.013
	CL4SRec	0.4210±0.016	0.3821±0.017	0.4205±0.013	0.3781 ±0.015	0.3814 ±0.016	0.3318±0.012
	LT-OCF	0.4267±0.013	0.3785±0.015	0.4141±0.016	0.3673 ±0.014	0.3848±0.012	0.3313±0.013
ODE	GRU-ODE	0.4398±0.014	0.3902 ±0.017	0.4275±0.013	0.3792±0.012	0.3994±0.013	0.3417±0.015
Proposed	**E-NSDE**	**0.4711±0.015**	**0.4112±0.013**	**0.4551±0.011**	**0.3982±0.016**	**0.4194±0.013**	**0.3637±0.015**

The overall loss of the end-to-end model training is acquired by combining the EDL and WBPR loss:

$$\mathcal{L}(u_t, i_t) = \mathcal{L}_{EDL}(u_t, i_t) + \zeta \mathcal{L}_{BPR}(u_t, i_t) \tag{6.18}$$

where ζ represents the balancing factor between EDL and WBPR loss.

Training and inference details are provided in Appendix section C.3.

6.5 Experiments

We conduct extensive experiments on three real-world datasets that contain explicit ratings: *Movielens-100K*, *Movielens-1M*, and *Netflix*. For baseline comparisons, we use dynamic, sequential, graph-based, and ODE-based recommendation methods. For more details about datasets, baselines, experimental setup, and implementation please check Appendix section C.4, C.5, and **??** respectively.

Evaluation metrics. To evaluate the proposed and baseline recommendation models, we follow the sequential recommendation setup similar to [74]. We consider one ground truth item in each sequential recommendation. We use two standard metrics to measure the recommendation performance.

- **Precision@N (P@N)**: It is the fraction of the top-N items recommended in each sequence to the user . We reported the average overall sequence precision value as the final precision. Further, due to only one ground truth in the target, the P@N is equivalent to Recall@N.
- **nDCG@N** : Normalized Discounted Cumulative Gain (nDCG) measures ranking quality, considering the relevant items within the top-N of the ranking list in each recommendation.

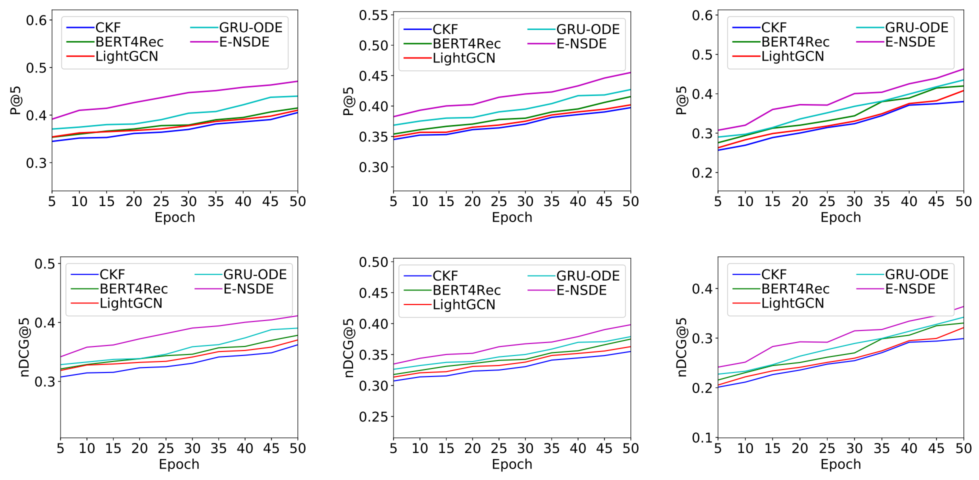


Figure 6.2: Precision (top) and nDCG (bottom) Performance @5: MovieLens-100K , MovieLens-1M , and Netflix datasets.

6.5.1 Recommendation Performance Comparison

Table 6.2 summarizes the recommendation performance from all models for three real-world datasets. The proposed model benefits from both the SDE module, which continuously captures user evolving preferences, and the evidential module, which estimates prediction confidence, and thus achieves better results in all three datasets. The dynamic models achieve less ideal performance due to their focus on discrete-term user interest and inability to provide continuous user preference. Graph-based methods take advantage of recently interacted items and have shown better performance than traditional dynamic methods. However, they may not be good enough to capture user sequential interest. Further, sequential methods benefit from sequential learning and have promising results. However, they lack to consider the time component in the recommendation and hence are less effective than the proposed method. ODE-based methods have shown a clear advantage due to their focus on capturing users' continuous behavior over time, but they cannot estimate model confidence on predictions and hence have lower performance value than the proposed method. We provided a detailed plot of precision and nDCG @5 in Figure 6.2 considering test users in each training epoch.

6.5.2 Qualitative Study

We performed a qualitative study on the impact of interaction time gap and its relation to uncertainty to provide important and diverse items. Table 6.3 shows the proposed E-NSDE model provides diverse tastes (i.e.

'Thriller' and 'Sci_Fi') movies for the larger interaction time gap (Δt)=896,291 seconds that are important for the future to the user (ID:13). In the other hand, the existing GRU-ODE model suggest only popular genres like 'Drama' and fails to explore.

Table 6.3: Important movies recommended to the user (ID:13) at sequence 150 to 151 with gap (Δt): 896,291 seconds.

Model	Important Movies (Genre)	Future Movie's Genre
GRU-ODE	Dead Man Walking ('Drama')	'Thriller'
	Richard III ('Drama')	'Drama'
	Mad Love ('Romance')	'Mystery'
E-NSDE	GoldenEye ('Thriller')	'Crime'
	Taxi Driver ('Drama')	'Sci_Fi'
	Twelve Monkeys ('Sci_Fi')	

We also did an ablation study, first, we provide the impact of each key component in the proposed E-NSDE method as shown in Table 6.4. From the table, it is clear that each component is helping to improve recommendation performance. Second, we provide the impact of hyperparameters (λ, ζ, η) and embedding size (d) in the Appendix section C.6.

Table 6.4: Ablation study results on Movielnes-100K

NSDE	EDL	WBPR	Performance	
			P@5	nDCG@5
✓			0.4065	0.3677
✓	✓		0.4523	0.3962
✓		✓	0.4120	0.3715
✓	✓	✓	0.4711	0.4112

6.6 Conclusion

We propose a novel evidential stochastic differential equation (E-NSDE) model for the time-aware sequential recommendations. The proposed model seamlessly integrates an NSDE module and an EDL module to capture users' continuously evolving behavior and model predictive uncertainty at the same time. Our proposed model effectively leverages the interaction time gap and provides uncertainty-aware recommendations with diverse items to the user. Further, we theoretically derive mathematical relationships between the interaction time gap and model uncertainty to enhance the learning process. Experimental results on real-world data and comparison with the state-of-the-art competitive models demonstrate the effectiveness of the proposed model.

Chapter 7

Conclusion and Future Work

In this chapter, we conclude the dissertation and also provide some future directions.

7.1 Conclusion

Learning from user interactions remains a complex task due to sparse human behavior and that is also dynamic, noisy, and highly heterogeneous. Consequently, current approaches are limited in their effectiveness. To address this challenge, we offer an efficient framework for learning to learn (`L2L`) framework to learn from sparse user interactions. Initially, we introduce a prevalent `L2L` strategy known as meta-learning, employing a recurrent neural network to capture the evolving user preferences over time and provide dynamic user preferences in Chapter 3. We have developed a dynamic meta-learning model that enhances the effectiveness of learning from scarce user interactions. We extend our contributions to include evidence-informed meta-learning in Chapter 4, which incorporates predictive evidence to guide learning systems in delivering effective user preferences. Furthermore, we leverage reinforcement learning for learning-to-learn with respect to user dynamics and employ evidence-based learning for uncertainty-aware exploration, aiming to enhance long-term user engagement in Chapter 5. This culmination of efforts resulted in the development of a meta-evidential reinforcement learning model, known as MetaERL. Finally, we introduce the Neural Stochastic Differential Equation (SDE) method with integrated evidential learning (E-NSDE) within the learning-to-learn framework in Chapter 6, enabling the capture of users' continuous behaviors that accurately reflect their evolving real-life scenarios. This framework is applicable to many domains including e-commerce, media, health, and many time-evolving user scenarios that specifically have very sparse datasets.

In conclusion, our framework provides an effective means of quick adaptation from global user knowledge leveraging the L2L approach and dynamic user intent via sequential and dynamic models for users with sparse interactions.

7.2 Future Work

Meta Evidential Graph Contrastive Learning for Sequential Recommendations In this future work, we will focus on how to effectively handle the users' evolving preferences over time in sparse data domains like recommender systems. We plan to use meta learning as a learning-to-learn approach to tackle the sparse interactions and the evidential graph contrastive learning method to provide effective user representation so that it recommends important items that are more useful for model training and long-term prediction.

There are several contrastive learning-based methods in recommender systems but they largely generate the contrastive view of the user in a static setting [52, 55, 71] and fail to deal with evolving user preferences for users who have very limited interactions. Also, they neglect the importance of interaction time while generating graph augmentation. Further, they do random sampling to augment the graph and don't have a specific mechanism to capture the uncertainty in the augmented graph. Similarly, meta-learning-based models are used to tackle the sparse problem in recommender systems but they fail to provide important and diverse recommendations for an effective recommendation.

To address the above problems of those methods, we plan to do future research on a novel model by integrating the meta-evidential learning method with a graph contrastive learning method that handles sparse interaction problems in the user's evolving scenario by leveraging the benefits of exploitation and exploration graph augmentation views to generate users' richer dynamic representation.

To accomplish this, in the future, we plan to conduct the research in the short term and long term:

1. **Short Term Plan:** In the short term, we plan to construct datasets and devise the model. For **dataset construction**, we need to collect the datasets in a task-wise manner to train the model. We will process and construct the dataset for each user as a single task. This fits the task level setting of the problem and we will mainly focus on those sparse user interactions which are very difficult in testing time. Further, **design model** will involve a complex combination of meta, evidential, and graph contrastive learning methods. We design the model to handle sparse user interactions from the perspective of few-shot learning utilizing meta-learning and capturing effective users' preferences via contrastive graph learning and also providing important and diverse items via evidential learning.

2. **Long Term Plan:** In the long term, we plan to design the loss function for the end-to-end training, perform experiments, and better fine-tune the proposed model. For **loss function design**, we leverage the power of contrastive learning and evidential learning approaches, thus, we will design a loss function that incorporates the impact of both approaches for better model training and generalization. Similarly, **experiment and evaluation**, we perform experiments on the above-collected datasets considering important evaluation metrics for a recommendation. Finally, we will do **fine-tuning and ablation study** of the proposed model for better explainability and scalability of the model.

Chapter 8

List of Publications

8.1 Published

1. Wang, Dingrong, Deep Shankar Pandey, Krishna Prasad Neupane, Zhiwei Yu, Ervine Zheng, Zhi Zheng, and Qi Yu. ”Deep Temporal Sets with Evidential Reinforced Attentions for Unique Behavioral Pattern Discovery.” in ICML 2023.
2. Neupane, Krishna Prasad, Ervine Zheng, Yu Kong, and Qi Yu. ”A Dynamic Meta-Learning Model for Time-Sensitive Cold-Start Recommendations.” in AAAI, 2022.
3. Neupane, Krishna Prasad, Ervine Zheng, and Qi Yu. ”MetaEDL: Meta Evidential Learning For Uncertainty-Aware Cold-Start Recommendations.” in ICDM, 2021.
4. Kwon, Minseok, Krishna Prasad Neupane, John Marshall, and M. Mustafa Rafique. ”CuVPP: Filter-based Longest Prefix Matching in Software Data Planes.” in CLUSTER, 2020.
5. Neupane, Krishna Prasad, Kabo Cheung, and Yi Wang. ”EmoD: An end-to-end approach for investigating emotion dynamics in software development.” in ICSME, 2019.
6. Neupane, Krishna. ”An approach for investigating emotion dynamics in software development.” in ASE, 2019.

8.2 Submitted

1. Neupane, Krishna Prasad, Ervine Zheng, and Qi Yu. ”Evidential Stochastic Differential Equations for Sequential Recommendations.” In Submission, 2024.

2. Wang, Dingrong, Krishna Prasad Neupane, Ervine Zheng, and Qi Yu. ”Evidential Conservative Q-Learning for Dynamic Recommendations ” In Submission 2024.

3. Sapkota, Hitesh, Krishna Prasad Neupane, and Qi Yu. ”Meta Evidential Transformer for Few-Shot Open-Set Recognition.” In Submission, 2024.

4. Lui, Yang, Domenic Mangano, Krishna Prasad Neupane, Samuel Malachowsky, and Daniel Krutz. ”Empathy-Building Interventions: A Comparative Analysis of Experiential and Expression-Based Interventions” In Submission, 2024.

Bibliography

[1] Woo-Kyoung Ahn and William F Brewer. Psychological studies of explanation—based learning. *Investigating explanation-based learning*, pages 295–316, 1993.

[2] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou. A survey paper on recommender systems. *arXiv preprint arXiv:1006.5278*, 2010.

[3] Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. *Advances in Neural Information Processing Systems*, 33, 2020.

[4] Lmar M Babrak, Joseph Menetski, Michael Rebhan, Giovanni Nisato, Marc Zinggeler, Noé Brasier, Katja Baerenfaller, Thomas Brenzikofer, Laurenz Baltzer, Christian Vogler, et al. Traditional and digital biomarkers: two worlds apart? *Digital biomarkers*, 3(2):92–102, 2019.

[5] Xueying Bai, Jian Guan, and Hongning Wang. A model-based reinforcement learning with adversarial training for online recommendation. *Advances in Neural Information Processing Systems*, 32, 2019.

[6] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, volume 2. Univ. of Texas, 1992.

[7] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.

[8] Brinnae Bent, Ke Wang, Emilia Grzesiak, Chentian Jiang, Yuankai Qi, Yihang Jiang, Peter Cho, Kyle Zingler, Felix Ikponmwosa Ogbeide, Arthur Zhao, et al. The digital biomarker discovery pipeline: An open-source software platform for the development of digital biomarkers using mhealth and wearables data. *Journal of clinical and translational science*, 5(1), 2021.

[9] Rianne van den Berg, Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[10] Rich Caruana. *Multitask learning*. Springer, 1998.

[11] Laurent Charlin, Rajesh Ranganath, James McInerney, and David M Blei. Dynamic poisson factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 155–162, 2015.

[12] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[13] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[14] Jeongwhan Choi, Jinsung Jeon, and Noseong Park. Lt-ocf: learnable-time ode-based collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 251–260, 2021.

[15] Arthur P Dempster. A generalization of bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1968.

[16] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. Sequential scenario-specific meta learner for online recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2895–2904, 2019.

[17] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–42, 2020.

[18] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pages 1447–1456. PMLR, 2018.

[19] Nicola Ferro, Claudio Lucchese, Maria Maistro, and Raffaele Perego. Boosting learning to rank with user dynamics and continuation methods. *Information Retrieval Journal*, 23:528–554, 2020.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[21] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.

[22] Simon Funk. Incremental svd method, 2006.

[23] Yang Gao, Christian M Meyer, and Iryna Gurevych. Preference-based interactive multi-document summarisation. *Information Retrieval Journal*, 23:555–585, 2020.

[24] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018.

[25] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[26] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921*, 2018.

[27] San Gultekin and John Paisley. A collaborative kalman filter for time-evolving dyadic processes. In *2014 IEEE International Conference on Data Mining*, pages 140–149. IEEE, 2014.

[28] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[29] Jiayan Guo, Peiyan Zhang, Chaozhuo Li, Xing Xie, Yan Zhang, and Sunghun Kim. Evolutionary preference learning via graph nested gru ode for session-based recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 624–634, 2022.

[30] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[31] Hado Hasselt. Double q-learning. In *Proc. of NeurIPS*, 2010.

[32] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng 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*, pages 639–648, 2020.

[33] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[35] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *Artificial Neural Networks—ICANN 2001: International Conference Vienna, Austria, August 21–25, 2001 Proceedings 11*, pages 87–94. Springer, 2001.

[36] Junyang Jiang, Deqing Yang, Yanghua Xiao, and Chenlu Shen. Convolutional gaussian embeddings for personalized recommendation with uncertainty. *arXiv preprint arXiv:2006.10932*, 2020.

[37] Audun Josang, Jin-Hee Cho, and Feng Chen. Uncertainty characteristics of subjective opinions. In *2018 21st International Conference on Information Fusion (FUSION)*, 2018.

[38] Audun Jøsang. Subjective logic. 2016.

[39] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

[40] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. *IEEE access*, 6:1662–1669, 2017.

[41] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[42] Ankit Koirala, Zhiwei Yu, Hillary Schiltz, Amy Van Hecke, Brian Armstrong, and Zhi Zheng. A preliminary exploration of virtual reality-based visual and touch sensory processing assessment for adolescents with autism spectrum disorder. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 29:619–628, 2021.

[43] Lingkai Kong, Jimeng Sun, and Chao Zhang. Sde-net: Equipping deep neural networks with uncertainty estimates. *arXiv preprint arXiv:2008.10546*, 2020.

[44] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[45] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, 2009.

[46] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[47] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. Melu: Meta-learned user preference estimator for cold-start recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1073–1082, 2019.

[48] Kar Fye Alvin Lee, Woon-Seng Gan, and Georgios Christopoulos. Biomarker-informed machine learning model of cognitive fatigue from a heart rate response perspective. *Sensors*, 21(11):3843, 2021.

[49] Bin Li, Xingquan Zhu, Ruijiang Li, Chengqi Zhang, Xiangyang Xue, and Xindong Wu. Cross-domain collaborative filtering over time. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[50] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, 2010.

[51] Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020.

[52] Xuewei Li, Aitong Sun, Mankun Zhao, Jian Yu, Kun Zhu, Di Jin, Mei Yu, and Ruiguo Yu. Multi-intention oriented contrastive learning for sequential recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 411–419, 2023.

[53] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[54] Jiqun Liu, Matthew Mitsui, Nicholas J Belkin, and Chirag Shah. Task, information seeking intentions, and user behavior: Toward a multi-level understanding of web search. In *Proceedings of the 2019 conference on human information interaction and retrieval*, pages 123–132, 2019.

[55] Zhuang Liu, Yunpu Ma, Yuanxin Ouyang, and Zhang Xiong. Contrastive learning for recommender system. *arXiv preprint arXiv:2101.01317*, 2021.

[56] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[57] Yuanfu Lu, Yuan Fang, and Chuan Shi. Meta-learning on heterogeneous information networks for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1563–1573, 2020.

[58] Rishabh Mehrotra, Prasanta Bhattacharya, and Emine Yilmaz. Uncovering task based behavioral heterogeneities in online search behavior. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 1049–1052, 2016.

[59] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[60] Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[61] Cuong Nguyen, Thanh-Toan Do, and Gustavo Carneiro. Uncertainty in model-agnostic meta-learning using variational inference. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 3090–3100, 2020.

[62] Alex M Pagnozzi, Eugenia Conti, Sara Calderoni, Jurgen Fripp, and Stephen E Rose. A systematic review of structural mri biomarkers in autism spectrum disorder: A machine learning perspective. *International Journal of Developmental Neuroscience*, 71:68–82, 2018.

[63] Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 695–704, 2019.

[64] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *CoRR*, abs/1705.04304, 2017.

[65] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.

[66] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[67] Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. A hidden markov model for collaborative filtering. *Mis Quarterly*, pages 1329–1356, 2012.

[68] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[69] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[70] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Advances in Neural Information Processing Systems*, 31:3179–3189, 2018.

[71] Jie Shuai, Kun Zhang, Le Wu, Peijie Sun, Richang Hong, Meng Wang, and Yong Li. A review-aware graph contrastive learning framework for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1283–1293, 2022.

[72] Joseph Sill. Monotonic networks. *Advances in neural information processing systems*, 10, 1997.

[73] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[74] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[75] John Z Sun, Dhruv Parthasarathy, and Kush R Varshney. Collaborative kalman filtering for dynamic matrix factorization. *IEEE Transactions on Signal Processing*, 62(14):3499–3509, 2014.

[76] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 1999.

[77] Haining Tan, Di Yao, Tao Huang, Baoli Wang, Quanliang Jing, and Jingping Bi. Meta-learning enhanced neural ode for citywide next poi recommendation. In *2021 22nd IEEE International Conference on Mobile Data Management (MDM)*, pages 89–98. IEEE, 2021.

[78] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 565–573, 2018.

[79] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[80] Lasitha Uyangoda, Supunmali Ahangama, and Tharindu Ranasinghe. User profile feature-based approach to address the cold start problem in collaborative filtering for personalized movie recommendation. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pages 24–28. IEEE, 2018.

[81] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. A meta-learning perspective on cold-start recommendations for items. In *Advances in neural information processing systems*, pages 6904–6914, 2017.

[82] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[83] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*, 2019.

[84] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[85] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. Contextual bandits in a collaborative environment. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 529–538, 2016.

[86] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353, 2019.

[87] Fenfang Xie, Liang Chen, Yongjian Ye, Zibin Zheng, and Xiaola Lin. Factorization machine based service recommendation on heterogeneous information networks. In *2018 IEEE International Conference on Web Services (ICWS)*, pages 115–122. IEEE, 2018.

[88] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*, pages 1259–1273. IEEE, 2022.

[89] Winnie Xu, Ricky TQ Chen, Xuechen Li, and David Duvenaud. Infinitely deep bayesian neural networks with stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pages 721–738. PMLR, 2022.

[90] Yuhao Yang, Chao Huang, Lianghao Xia, Chunzhen Huang, Da Luo, and Kangyi Lin. Debiased contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2023*, pages 1063–1073, 2023.

[91] Yuan Yao, Hanghang Tong, Guo Yan, Feng Xu, Xiang Zhang, Boleslaw K Szymanski, and Jian Lu. Dual-regularized one-class collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 759–768, 2014.

[92] Jiarong Ye, Yuan Xue, L. Rodney Long, Sameer K. Antani, Zhiyun Xue, Keith C. Cheng, and Xiaolei Huang. Synthetic sample selection via reinforcement learning. *CoRR*, abs/2008.11331, 2020.

[93] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention network. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

[94] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[95] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proc. of WWW*, 2018.

[96] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5941–5948, 2019.

[97] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020.

[98] Lixin Zou, Long Xia, Pan Du, Zhuo Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. Pseudo dyna-q: A reinforcement learning framework for interactive recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020.

Appendices

Appendix A

In this appendix, we mainly provide the additional results to better strengthen the experiment of Chapter 3.

A.1 Periodical Recommendation Results

We report the detailed result for each prediction period for all three datasets to demonstrate how the predictions evolve over time.

Netflix. As can be seen from Table A.1, in most periods, the proposed model performs better than others except for a few periods like 3 and 6, where the proposed model slightly under-performs the meta-learning model. A possible explanation is that in these periods, time-specific user interest might largely deviate from the time-evolving user factors, and hence their combined recommendation is less accurate.

Last.fm Datasets. The period-wise results for the Last.fm dataset on one run is shown below in Table A.2. The proposed model achieves good results in this implicit feedback (i.e., counts) dataset. The high variation of the counts indicates users' music listening habits are fluctuating significantly. The results in Table A.2 show that matrix factorization and deep learning baseline models are less effective in capturing those variations. In contrast, the proposed model simultaneously captures those variations in the form of users' specific biases and the gradual shift of preferences effectively.

Movielens Datasets: Periodic results for the Movielens dataset are shown in Table A.3. In the first period, we notice that meta-learning models are not performing well, whereas SVD models are performing well. This is because the dataset has very dense interactions in the first period. Meta-learning models only use

Table A.1: Netflix Periodic RMSE Results

Period	SVD++	timeSVD++	CKF	MeLU	DeepFM	Wide & Deep	GC-MC	Caser	DIEN	ML-ICS	Proposed
1	0.9813	0.9840	0.9552	0.9201	1.2706	1.3178	1.2912	1.2845	1.3102	0.9335	0.9205
2	0.9862	0.9895	0.9683	0.9481	1.1117	1.1321	1.2903	1.2882	1.3033	0.9498	0.9258
3	0.9795	0.9795	0.9693	0.9440	1.0489	1.0921	1.2732	1.2634	1.2724	0.9445	0.9479
4	0.9742	0.9708	0.9524	0.9374	1.0896	1.0777	1.2613	1.2404	1.2543	0.9401	0.8771
5	0.9800	0.9710	0.9432	0.9415	1.0630	1.0115	1.2311	1.2127	1.2167	0.9426	0.9279
6	0.9757	0.9740	0.9564	0.9495	0.9911	0.9904	1.1374	1.1206	1.1515	0.9487	0.9533
7	0.9744	0.9725	0.9612	0.9506	1.0323	0.9895	1.1068	1.1745	1.0984	0.9622	0.8723
8	0.9766	0.9661	0.9526	0.9418	0.9706	1.0122	1.049	1.1132	1.0401	0.9517	0.8789
9	0.9737	0.9603	0.9412	0.9424	0.9902	0.9897	1.0442	1.0724	1.0311	0.9503	0.8786
10	0.9726	0.9597	0.9228	0.9391	0.9961	0.9655	1.0918	1.0843	1.0511	0.9430	0.9209
11	0.9722	0.9595	0.9332	0.9360	0.9497	1.0570	1.03403	1.0563	1.0615	0.9315	0.8997
12	0.9731	0.9511	0.9541	0.9438	0.9761	0.9648	1.0885	1.0245	1.0528	0.9396	0.9427
13	0.9361	0.9556	0.9243	0.9351	0.9710	0.9540	1.0451	1.0373	1.0417	0.9306	0.8746
14	0.9658	0.9431	0.9416	0.9457	0.9720	0.9635	1.031	1.0143	1.0531	0.9487	0.8824
15	0.9673	0.9457	0.9358	0.9378	1.0050	0.9598	1.0162	1.0142	1.0237	0.9441	0.8812
16	0.9670	0.9340	0.9337	0.9428	0.9866	0.9550	1.0254	1.0078	1.0145	0.9468	0.8340

Table A.2: Last.fm periodic RMSE results

Period	SVD++	timeSVD++	CKF	DPF	MeLU	DeepFM	Wide & Deep	Caser	DIEN	ML-ICS	Proposed
1	1.6507	1.6920	1.8353	1.6953	1.3153	2.1434	2.1993	1.8132	2.4720	1.3233	1.3173
2	2.3052	2.1647	1.3964	1.3238	1.5567	2.1336	2.1675	1.3854	2.1291	1.5407	1.5320
3	2.1972	1.9160	1.7007	2.0814	1.3408	2.0674	2.1194	1.6772	2.1065	1.3531	1.3465
4	2.1574	1.7385	1.5462	1.4654	1.1346	2.0757	2.0464	1.5232	1.9520	1.1287	1.1047
5	1.5858	1.6520	1.6602	1.5214	1.1687	2.0202	2.0033	1.5843	1.9876	1.1732	1.1578
6	1.7879	1.8460	1.5555	1.3815	1.2602	2.0763	2.1651	1.5278	1.8816	1.2821	1.0809
7	1.5348	1.6498	1.6527	1.3448	1.6359	1.9372	1.9595	1.6227	1.8392	1.6324	1.6226
8	1.7527	1.7350	1.5798	1.3617	1.5987	1.8883	1.8839	1.5482	1.8806	1.6037	1.4270
9	2.2419	1.9977	1.4311	1.5085	1.4493	1.9546	2.0015	1.4326	1.9011	1.4488	1.3338
10	1.9101	1.7901	1.4572	2.3051	1.1753	1.8346	1.8766	1.4104	1.8743	1.1714	1.0368
11	1.5709	1.4704	1.3067	1.4337	1.1528	1.7557	1.8178	1.2974	1.8515	1.1553	1.0864
12	1.5864	1.5760	1.3353	1.2409	1.2226	1.7666	1.7346	1.3136	1.8483	1.2105	1.0933
13	1.5228	1.3637	1.4613	1.0546	1.0546	1.7518	1.7192	1.4463	1.8617	1.0720	1.0123
14	1.7330	1.5966	1.6130	1.2670	1.0809	1.6417	1.6800	1.5812	1.8445	1.0912	1.0603
15	1.9891	1.6644	1.4429	1.3264	1.1094	1.6815	1.7253	1.4293	1.7623	1.0996	1.0824
16	1.4638	1.4106	1.4334	1.2547	1.3206	1.6900	1.9963	1.4017	1.7456	1.3322	1.0688

Table A.3: Movielens periodic RMSE results

Period	SVD++	timeSVD++	CKF	MeLU	DeepFM	Wide & Deep	GC-MC	Caser	DIEN	ML-ICS	Proposed
1	1.0615	1.0496	1.1155	1.2234	1.5341	1.3719	2.3345	2.2941	2.3438	1.2412	1.2253
2	0.9954	0.9932	0.9847	1.1613	1.3659	1.2686	1.7912	1.7247	1.7065	1.1803	1.0444
3	1.0445	0.9982	1.0305	0.9341	1.2267	1.2585	1.2576	1.2289	1.2019	0.9423	0.8487
4	1.1499	1.1189	1.0508	0.9776	1.2492	1.2346	1.1332	1.1223	1.1623	0.9711	0.9084
5	1.0918	1.0611	1.1468	1.0308	1.1615	1.1052	1.1346	1.1286	1.1587	1.0514	0.9053
6	1.0773	1.0688	1.0699	1.1377	1.0994	1.0672	1.1112	1.1021	1.1421	1.1127	1.0377

k-shot for learning, but SVD models benefit from maximum interactions. For the proposed model, time-evolving user factors don't contribute in the first period. Also, time-specific factors are based on meta-learning. Hence, its performance is not better than the baselines, but the proposed model achieves a better performance in the subsequent periods

Appendix B

In this appendix, we mainly provide the theoretical proof of Chapter 5.

B.1 Proofs of Theoretical Results

In this section, we provide proofs of all lemmas and the theorem.

B.1.1 Proof of Lemma 1

Given the evidential reward defined as $r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) = \frac{1}{N}\left(\sum_{i \in \mathcal{N}_u}(\text{rating}_{u,i} - \tau) + \lambda \mathcal{U}_\pi(i|\mathbf{s}_t, \mathbf{a}_t)\right)$ the update rule for evidential Q-value can be written as:

$$Q^e(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_\pi \sum_{t'=t}^{\infty} \gamma^{t'} r^e_\pi(\mathbf{s}_{t'}, \mathbf{a}_{t'}) = r^e_\pi(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1}}[Q^e(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})] \tag{B.1}$$

Then based on the evaluation convergence rule [76] with finite action space, it is guaranteed that the Q-value will converge to the evidential Q-value of policy π.

B.1.2 Proof of Lemma 2

The policy can be updated towards the new Q-value function. Consider the updated policy π_{new} as the optimizer of the maximization problem.

$$\pi_{new} = \operatorname*{argmax}_{\pi'} J_\pi(\phi) = \operatorname*{argmax}_{\pi'} \mathbb{E}_{\mathbf{s}_t \sim D, \mathbf{a}_t \sim \pi'}[Q^e_{\pi'}(\mathbf{s}_t, \mathbf{a}_t)] \tag{B.2}$$

Denote the old policy as π_{old}. Using the update rule specified in Eq 5.8 with a sufficiently small step size, we get an updated policy π_{new} that satisfies

$$\mathbb{E}_{\mathbf{a}_t \sim \pi_{new}}[Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t)] \geq \mathbb{E}_{\mathbf{a}_t \sim \pi_{old}}[Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t)] \tag{B.3}$$

Given Equation B.3, we have the following inequality

$$\begin{aligned} Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t) \leq & r^e(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \sim \pi_{new}}[Q^e_{\pi_{old}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})] \\ \leq & r^e(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1}, \mathbf{a}_{t+1} \sim \pi_{new}}[r^e(\mathbf{s}_{t+1}, \mathbf{a}_{t+1})] \\ & + \mathbb{E}_{\mathbf{s}_{t+2}, \mathbf{a}_{t+2} \sim \pi_{new}}[Q^e_{\pi_{old}}(\mathbf{s}_{t+2}, \mathbf{a}_{t+2})] \\ & ... \\ = & Q^e_{\pi_{new}}(\mathbf{s}_t, \mathbf{a}_t) \end{aligned} \tag{B.4}$$

where $r^e(\mathbf{s}_t, \mathbf{a}_t)$ is a evidential reward in step t. Therefore, we show that the new policy π_{new} ensures $Q^e_{\pi_{new}}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^e_{\pi_{old}}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t)$.

B.1.3 Proof of Theorem 1

Let π_i denote the policy at iteration i. We already show that the sequence $Q^e_{\pi_i}(\mathbf{s}_t, \mathbf{a}_t)$ is monotonically increasing. Since $Q^e_{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ is bounded above, the sequence converges to some π^*. At convergence, it must be the case that $J_{\pi^*}(\pi^*(.|\mathbf{s}_t)) \leq J_{\pi^*}(\pi(.|\mathbf{s}_t))$ for $\pi \neq \pi^*$. Based on Lamma 11, we have $Q^e_{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) > Q^e_{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ for all $(\mathbf{s}_t, \mathbf{a}_t)$. In other words, the evidence value of any other policy π is lower than that of the converged policy π^*. Therefore, it guarantees convergency to an optimal policy π^* such that:

$$Q^e_{\pi^*}(\mathbf{s}_t, \mathbf{a}_t) \geq Q^e_{\pi}(\mathbf{s}_t, \mathbf{a}_t) \tag{B.5}$$

Appendix C

In this Appendix, we first summarize all notations used in Chapter 6. Next, we provide the theoretical proof for Lemma 2 and Lemma 3. Following that, we provide algorithms for the training and inference process, and then we provide datasets and baselines. Further, we provide an experimental setup along with additional results for the ablation study.

C.1 Summary of Notations

We summarize the major notations used throughout the paper in Table C.1.

C.2 Proof of Theoretical Results

In this section, we provide proofs of Lemma 2 and Lemma 3.

C.2.1 Proof of Lemma 2

Given the user final representation u_t, item final representation i_t, and interaction time gap Δt. The monotonic network produces $\beta_{(u_t,i_t)}$ as:

$$\begin{aligned} \beta_{(u_t,i_t)} &= W \times \text{concat}[u_t, i_t, \Delta t] \\ \frac{d\beta}{d(\Delta t)} &= W \end{aligned} \tag{C.1}$$

Given the corresponding weight W is positive then $\frac{d\beta}{d(\Delta t)} \geq 0$.

Table C.1: Summary of Notations

Notation	**Description**
U, I	user set and item set
u_t, i_t	user u and item i continuous representations at time t
$\beta_{(u_t,i_t)}$	total model uncertainty for user u on item i at time t
$\alpha_{(u_t,i_t)}, \nu_{(u_t,i_t)}$	model evidence for for user u on item i at time t
$\gamma_{(u_t,i_t)}$	predicted score for user u on item i at time t
$\hat{r}_{(u_t,i_t)}, r_{(u_t,i_t)}$	Uncertainty-aware predicted rating and ground truth for user u on item i at time t
$\hat{r}_{(i_t,j_t)}$	Predicted rating score difference between ground truth item i and negative item j for user u at time t
Θ	Overall model parameters i.e. Θ=(ψ, ω, θ, ϕ)
ψ, ω, θ, ϕ	SDE drift, SDE diffusion, rating, and monotonic networks parameters
Δt	interaction time gap between two consecutive items
$w_{(i_t,j_t)}$	Weight coefficients for negative item j based on uncertainty-aware predicted scores
τ	Threshold for uncertainty-aware predicted rating score
$\mathcal{N}_t$	Negative items at time t
$\mathcal{U}_{(u_t,i_t)}$	Epistemic uncertainty for user u and item i interaction at time t
λ, η, ζ	Balancing coefficient for EDL regularizer, uncertainty-aware rating, and WBPR loss respectively

C.2.2 Proof of Lemma 3

Given the same setup as Lemma 2, the monotonic network outputs $\alpha_{(u_t,i_t)}$, and $\nu_{(u_t,i_t)}$. We first consider $\alpha_{(u_t,i_t)}$:

$$\begin{aligned}
\alpha_{(u_t,i_t)} =& W_L \times h_{L-1}[W_{L-1,\dots} \times h_0[W_0 \times [\text{concat}[u_t, i_t, \Delta t]]] \\
=& W_L \times h_{L-1}[W_{L-1,\dots} \times h_0[W_0 \times \text{concat}[u_t, i_t]] \\
&+ W_L \times h_{L-1}[W_{L-1,\dots} \times h_0[W_0 \times \Delta t]] \\
\frac{d\alpha_{(u_t,i_t)}}{d(\Delta t)} =& W_L \times h'_{L-1} \times W_{L-1} \dots h'_0 \times W_0
\end{aligned}$$

when W_L is negative and $W_{L-1,\dots 0}$ are positives then $\frac{d\alpha}{d(\Delta t)} \leq 0$.

Similarly, we can show that $\frac{d\nu}{d(\Delta t)} \leq 0$. We leverage the monotonic non-linear activation function i.e. $ELU(.)$ to satisfy this condition.

C.3 Training and Inference

The training procedure involves the end-to-end parameter updates associated with the NSDE module and evidential module. Both modules utilize the overall loss mentioned in Eq (6.18), which includes a supervised signal from evidential loss and ranking loss from WBPR loss. The rating network (θ) and monotonic network (ϕ) are updated with the Adam optimizer, and the SDE module with model parameters (ψ, ω) is updated with SDE adjoint method [51]. Algorithm 4 shows the training process that learns the model parameters.

Algorithm 4 E-NSDE Training

```
Require: Hyperparameters: $\lambda, \eta, \zeta, \alpha, \beta, \gamma, \nu$
 1: Initialize both NSDE and EDL modules:$\Theta = (\psi, \omega, \theta, \phi)$
 2: while not converge do
 3:     Sample train user $\mathcal{T}_u$ from user pool $U$
 4:     for all $u \in \mathcal{T}_u$ do
 5:         Compute user and item final representations using Eq (6.9) and Eq (6.11) respectively from SDE
    module.
 6:         Compute interaction time gap $\Delta t$ with target item
 7:         Compute EDL loss for each sequence using Eq (6.14).
 8:         Compute weighted BPR loss for each sequence using Eq (6.17).
 9:         Perform end-to-end update using overall loss Eq 6.18
10:     end for
11: end while
```

During inference, we consider test users (i.e., distinct users from the training set) and perform standard sequential recommendations respecting the time interval of interactions.

C.4 Datasets and Baselines

We evaluated the E-NSDE model on three public benchmark datasets that contain explicit ratings:

- **Movielens-100K**[1]: This dataset contains 100,000 explicit ratings on a scale of (1-5) from 943 users on 1,682 movies. Each user at least rated 20 movies from September 19, 1997 through April 22, 1998.

[1] `https://grouplens.org/datasets/movielens/100k/`

- **Movielens-1M**[2]: This dataset includes 1M explicit feedback (i.e. ratings) made by 6,040 anonymous users on 3,900 distinct movies from 04/2000 to 02/2003.
- **Netflix** [7]: This dataset has around 100 million interactions, 480,000 users, and nearly 18,000 movies rated between 1998 to 2005. We pre-processed the dataset and selected 6,042 users with user-item interactions from 01/2002 to 12/2005.

Further, we used competitive baselines:

- **Dynamic models:** timeSVD++ [45] , and CKF [27]
- **Sequential models:** CASER [78], SASRec [39], BERT4Rec [74], S^3-Rec [97], and CL4SRec [88]
- **Graph-based models:** NGCF [84] and LightGCN [32]
- **ODE-based models:** LT-OCF [14], and GRU-ODE [29]

C.5 Experimental Setup

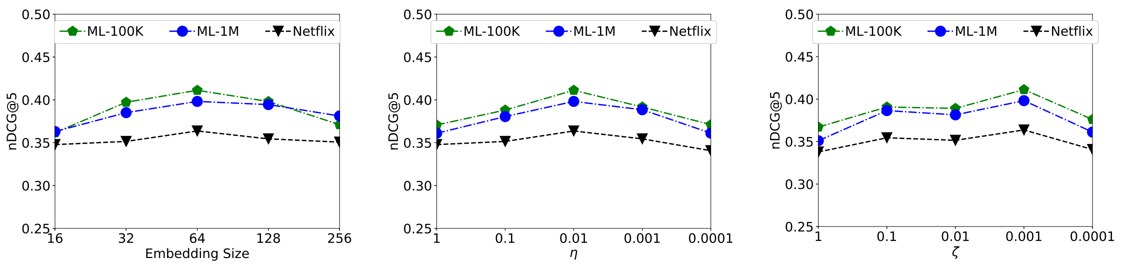
We set up the sequential recommendation models adopting next-item recommendation tasks, which were used in [74]. We first split users 70% into train and 30% in test. For each user, we leverage the fixed sequence length and hold out the next item of the behavior sequence as the target item. We follow the standard strategy in [74] for easy and fair evaluation. We leverage the actual time of interactions (in UNIX timestamp) to provide fine-grained user evolution.

C.6 Ablation Study

We perform an ablation study on the impact of hyperparameters (λ, η, ζ) and embedding size (d).

Evidential Regularization Parameter. One of the key hyperparameters of the *E-NSDE* model is the regularizer constant (λ) for the evidential learning. We cross-validated this parameter with empirical results of the model for the different λ values in two datasets as shown in Table C.2. From the table, our model achieves the best performance in both datasets with $\lambda = 0.001$.

[2]`https://grouplens.org/datasets/movielens/1M/`



(a) Embedding size (b) Uncertainty-aware rating (c) WBPR loss

Figure C.1: Average nDCG@5 plot for different embedding sizes, and balancing factors.

Table C.2: Average P@5 and nDCG@5 for *E-NSDE* with different regularizer values

Regularizer	**MovieLens 1M**		**Netflix**	
(λ)	**P@5**	**nDCG@5**	**P@5**	**nDCG@5**
0	0.4236	0.3786	0.4022	0.3574
0.0001	0.4412	0.3914	0.4134	0.3615
0.001	**0.4551**	**0.3982**	**0.4194**	**0.3637**
0.01	0.4518	0.3942	0.4096	0.3605
0.1	0.4224	0.3756	0.4064	0.3586
1	0.4116	0.3711	0.3923	0.3502

Embedding Dimension. We generate user and item embeddings using the embedding network. We perform a grid search for the embedding dimension (d) of the user and item representation in *E-NSDE* model as shown in Figure C.1a. From the plot, it shows that E-NSDE has the best performance with d=64.

Balancing Factors. We leverage grid search on uncertainty-aware ranking factor η, and WBPR loss balancing factor ζ on three datasets as shown in Figure C.1b and Figure C.1c respectively. From the plot, it shows a clear advantage with η=0.01, which indicates that the uncertainty-aware exploration component takes an effective role in providing the best performance for our proposed E-NSED model. Similarly, for ζ balancing factor integrated overall loss has the best performance when it is equal to 0.001.