Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9290576, 11 pages
https://doi.org/10.1155/2020/9290576

Research Article

Nonindependent Session Recommendation Based on Ordinary Differential Equation

Zhenyu Yang, Mingge Zhang, Guojing Liu, and Mingyu Li

School of Computer Science and Technology, Qilu University of Technology (Shandong Academy of Sciences), Jinan 250353, China

Correspondence should be addressed to Zhenyu Yang; yang_zhenyu@163.com

Received 17 October 2019; Revised 26 November 2019; Accepted 18 December 2019; Published 10 February 2020

Academic Editor: Francisco R. Villatoro

Copyright © 2020 Zhenyu Yang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The recommendation method based on user sessions is mainly to model sessions as sequences in the assumption that user behaviors are independent and identically distributed, and then to use deep semantic information mining through Deep Neural Networks. Nevertheless, user behaviors may be a nonindependent intention at irregular points in time. For example, users may buy painkillers, books, or clothes for different reasons at different times. However, this has not been taken seriously in previous studies. Therefore, we propose a session recommendation method based on Neural Differential Equations in an attempt to predict user behavior forward or backward from any point in time. We used Ordinary Differential Equations to train the Graph Neural Network and could predict forward or backward at any point in time to model the user's nonindependent sessions. We tested for four real datasets and found that our model achieved the expected results and was superior to the existing session-based recommendations.

1. Introduction

The traditional recommendation algorithms are based on the fact that all items that interact with the user are independent of each other. These methods ignore the continuity of information about the user's behavioral sequence and increasingly fail to meet the individual needs of the user. For example, in the Item-based recommendation algorithms [1–3], more attention is paid to the similarity between items, and the similarity calculation is used to infer the user's preference. The User-based recommendation algorithms [4–6] are similar to the Item-based recommendation. The difference is that the similarity between the preferences for different users is calculated. Although there are many hybrid recommendation methods [7–9] to compensate for the incompleteness of the above two methods, personalized recommendations are still not implemented. However, in recent years, due to the increasing amount of data, the limitations of traditional algorithms have become more and more visible, and Neural Networks have once again been pushed highly to the research. Network structures such as Long Short-Term Memory (LSTM) [10–12], Gated Recurrent Unit (GRU) [13, 14], and Recurrent Neural Networks (RNNs) [15] have been widely used in user behavior serialization modeling problems and personalized recommendation [15, 16].

The environment of the recommendation system is more complex and varied than expected. The user's behavior sequences are often indefinite, and the internal correlation between each sequence is very close, so Graph Neural Networks (GNN) [17, 18] are introduced into the recommendation system. We use a graph-structured model to capture the transformation of items and generate accurate item embedding vectors accordingly. This is clearly different from traditional methods, because we are not building a single item embedding. With this in mind, more and more people are beginning to use the graph network to implement the method of session recommendation.

In recent years, more and more researchers have focused on recommendation methods based on user sessions. In most session-based recommendation studies, a unified definition of a session is a sequence of records of a temporal relationship that can be used to track a user's browsing, clicking, or purchasing behavior. Compared to other methods of directly

modeling the relationship between users and items, the session-based approach can bring more implicit feedback.

In recent years, more and more researchers have focused on recommendation methods based on user sessions. In most session-based recommendation studies, a unified definition of a session is a sequence of records of a temporal relationship that can be used to track a user's browsing, clicking, or purchasing behavior. Compared to other methods of directly modeling the relationship between users and items, the session-based approach can bring more implicit feedback. But these studies are based on the fact that user behaviors are independent of each other, but it is not the case in real life.

In our research, we still use the idea of GNNs to model sessions, and Gated Graph Neural Networks (GGNNs) [19, 20] capture the complex transitions between items within the session. However, since the discretization of such data has been often undefined, leading to the direct use of neural networks to learn such data, there may be problems of data loss or inaccurate inconsistencies in certain time intervals. At this time, we introduce Ordinary Differential Equations (ODE) [21–23] to make up for this shortcoming.

We will present our model in this article: Sess-ODEnet. It uses Neural Differential Equations to propose another novel idea for session recommendation. The original intent of this approach was to model the user's nonindependent intent. In other words, we can get the potential trajectory at any point in time by solving the Ordinary Differential Equations [24], which allows us to make forward or backward predictions at any point in time.

The contributions of this work are as follows:

(i) We built a new Neural Ordinary Differential Equation model and used it to model session data. For the first time, the Ordinary Differential Equation solver is used for session recommendation. We used the reverse-mode automatic differentiation to help us track any potential state of the user.

(ii) Utilizing the update mechanism of the Gated Graph Neural Network to make ODE's solution more accurate. We used ODE to solve the GGNNs. It preserves the spatial modeling capabilities of the GNN in the user's session and preserves the learning power of complex transformations.

(iii) The user's sessions are predicted at any point in time through the Neural Ordinary Differential Equations. It shows that the model has a better ability to predict irregularity of users.

(iv) The complexity of the model in obtaining the optimal solution is very small, compared with other neural networks.

2. Related Work

A reexamination of the user's behavioral sequence becomes very important. More and more researchers agree to use the form of conversation to represent the sequence of user behavior, so there are more and more researches based on user sessions.

For the user, the relationship between the items inside each session is more closely compared to other items, so session-based recommendations are very significant. For the first time, Hidasi et al. [25] applied RNNs to session recommendation, treating a series of click behaviors occurring in a session as a sequence, and completing item-based sequence modeling to predict user preferences. Jannach and Ludewig [26] combined RNNs with kNN in 2017 and applied it to session recommendations, recognizing user behavior by RNNs, taking into account the mixture of simultaneous signals and sequence patterns.

Li et al. [27] studied a hybrid encoder (NARM), which has an attention mechanism to model the user's sequential behavior, capture the user's main purpose of the current session, and then combine it into a unified session representation. NARM's consideration of the primary purpose of the user's current action distinguishes it from previous sequence modeling methods.

However, the user's behavior sequences do not always have the same lengths. For these variable-length sequences, there are also many studies on their processing. According to the idea of word2vec, Barkan and Koenigstein [28] proposed a method of Item2vec. In item2vec, the item in the session is equivalent to the word in word2vec. If it appears in the same collection (sequence purchased or clicked by the same user), the item is considered a positive example. Although this method processes the indefinite length sequence by embedding and then calculates the probability of the prediction by calculating the similarity of the embedded vector. Grbovic and Cheng [29] proposed a "list embedding" method that provides a new perspective for item embedding. The internal relationship between these listings that the user clicks is mined by the context of the user's click sequence.

These proposed embedded-based methods focus on sequences of variable length but inevitably make the elements inside the session more sequential [30–33]. We need to consider the intimacy of all the items inside the session, not the items that are next to each other. Ideally, we can keep the intimacy of the items inside the session and prevent them from relying too much on timing and spatial location. In fact, the Graph Neural Network can do this.

Graph Neural Networks are further used to recommend modeling system scenarios [34,35]. Wu et al. [36] proposed SR-GNN in 2018 to model the session sequence of graph structure data to extract the embedded vector of the item using GNNs. The advantage of this is that the model does not rely on the user's relevant representation at this time, using the embedded layer of the session for the recommendation. It is different from the classic embedding method. It not only solves the problem of embedding unequal sequences but also gives different representations for the embedding of each item.

3. Description of Nonindependent Identically Distributed Intention Problems

In this section, we will primarily describe the intelligibility of the problem we are trying to solve, which will not contain

more algorithmic details. We also give descriptive meanings to the symbols that appear in our work.

3.1. Problem Description. The data in the recommendation system appears as multi-source heterogeneous data. In this case, the analysis and prediction of user behavior through traditional independent and identical distribution assumptions will bring specific problems. Therefore, the processing of such multi-source heterogeneous data is a fundamental problem that the recommendation system cannot avoid. It requires us to design the model, and it needs to be more specific to the actual problem and the characteristics and complexity of the actual data.

Let us consider the state of user behavior in real life. For example, a user has purchased K identical items in a year. In the previous recommendation algorithms, they considered the user's K-time behavior as independent and identical distribution. However, it is easy to imagine that he might have purchased the item for different reasons at different points in time. We refer to this behavior as "nonindependent identical distribution intent." Compared to Independent and identically distributed situations, the above user behavior is more common in real life.

This kind of thinking is straightforward to be recognized. We cannot get specific factors that affect user behavior, but we can mine such a layer of information from user behavior data. One idea is that we can use the context information of the user's click action to determine if the same action is due to the same situation.

From the perspective of continuous-time and space, we looked for another analysis method for the user's nonindependent behavior. The problem we are trying to solve is how to model such intentions. We try to achieve this by using Neural Ordinary Differential Equations [37]. In its forward propagation process, the original discrete states inside the network are connected by time t, presenting a continuous spatial state. However, how do we get the potential preferences of segmented users? Reverse-mode automatic differentiation [38] can solve this problem. It helps us solve the adjoint states at different times. That is, we assume that user behavior is different between different points in time. This means that we can "split" the user state in the continuous latent space.

3.2. Definition of Symbols. We describe the key representations used by the model and show them in Table 1 for quick access.

4. Sess-ODEnet: Use ODE Solver to Make Session Recommendation

In this section, we will introduce our model in detail. Our work can be divided into two parts. First, we can get a neural network that can be solved and then use ODE to solve the neural network to get the predicted results.

4.1. Recognition Network and Ordinary Differential Equation Solver. According to the description in Neural Ordinary Differential Equations [21], we need to define a recognition network. It can be arbitrary. We first considered the importance of user sessions. In the actual application scenario, long-term, orderly history may not be relevant to the user. User behavior may occur when an action occurs to an irregular point in time. Therefore, the internal continuity of the conversation and the irregular time points are the focus of our research.

4.1.1. Session Graph. In our model, we still use the definition of the session graph in [36]. Each session graph appears as a graph structure derived from the user's click sequence. The session graph is defined as $\mathbb{G} = (\mathscr{V}, \mathscr{E})$. In the session graph, each node represents an item $v_{s,i} \in \mathscr{V}$, and the user clicks on the item in the session as the edge of the graph, denoted as $(v_{s,i-1}, v_{s,i}) \in \varepsilon_s$. We assign a normalized weight to each edge, which is calculated as the number of occurrences of that edge divided by the degree of the starting node of that edge. This is to avoid duplicate items in the user's session. We embed each item in a unified embedding space. The node vector $v \in \mathbb{R}^d$ represents the potential vector of the item learned through the Graph Neural Networks, where d is the dimension. Thus, each session s can be represented as an embedded vector $\mathbb{S}$ consisting of the node vectors used in the session graph.

4.1.2. GGNNs as Recognition Network. Because Ordinary Differential Equation solver is suitable for any form of a neural network, we use Gated Graph Neural Networks (GGNNs) to learn session graphs. It is to allow information to propagate in space to model the complex relationships between items.

The gated graph neural network embeds each session graph into the graph, at which point each node contains context information. The GGNNs can be represented by the following equation:

$$\begin{aligned}
S_v^{(1)} &= \left[x_v^T, 0\right]^T, \\
a_v^{(t)} &= A_{v:}^T \left[S_1^{(t-1)T}, \ldots, S_{|v|}^{(t-1)T}\right]^T + b, \\
Z_v^t &= \sigma\left(W^z a_v^{(t)} + U^z S_v^{(t-1)}\right), \\
r_v^t &= \sigma\left(W^r a_v^{(t)} + U^r S_v^{(t-1)}\right), \\
\widetilde{S_v^{(t)}} &= \tanh\left(Wav(t) + U\left(r_v^t \odot S_v^{(t-1)}\right)\right), \\
S_v^{(t)} &= \left(1 - Z_v^t\right) \odot S_v^{(t-1)} + Z_v^t \odot \widetilde{h_v^{(t)}}.
\end{aligned} \tag{1}$$

Among them, $S_v^{(1)}$ represents the initial state of the $\mathscr{D}$ dimension of the node v. x_v^T represents node feature. A is an adjoint matrix that includes in- and out-degrees, and $a_v^{(t)}$ represents a 2-dimensional vector of the result of the interaction between nodes and adjacent nodes. Since the adjoint matrix A contains both in-degree and out-degree, the result of the calculation is similar to that of a cyclic neural network; that is, it contains bidirectional information

Table 1: Global symbols definition.

Symbol	Meaning
$\mathbb{G} = (\mathscr{V}, \mathscr{E})$	Session graph. $\mathscr{V}$ denotes a node in the graph, and $\mathscr{E}$ denotes an edge between the nodes
$\mathbb{S}$	The embedding vector of the session graph
g	A single layer representation of a gated graph neural networks
t, θ	They represent each moment in the graph and the current gradient
$a(t)$	Adjoint. It represents the dependence of the descending gradient on the hidden state at each time point
$S(t)$	Potential state of time t
$S(t_{\text{ini}})$	Initial state of time t
$S(t_{\text{ter}})$	Terminal state of time t
ODEsessSolver	Ordinary differential equation solver
I_t	It represents the sampled output at any time t obtained by the ODE solver
$q(\cdot)$	It represents the posterior probability of the sampled output. Specifically, it represents the probability of the item to be recommended

transfer, where $(1 - Z_v^t)$ chooses which information to forget, $Z_v^t\odot$ indicates which new information to remember, and $r_v^t\odot$ determines which new information is generated from which past information. $\widetilde{h_v^{(t)}}$indicates the newly generated information, and $S_v^{(t)}$ indicates the node status of the final update.

It shows that we take advantage of the gated graph neural network, which is its "forgetting" and "update" mechanisms. It essentially acts as an attention mechanism, which allows us to consider the user's long-term and short-term preferences better because information with too small weight will be filtered out. Finally, we can get the final representation of the individual node vectors in the graph.

4.1.3. ODEsessSolver. We will detail the ordinary differential equation solver, and its forward and backward propagation processes, which are the core of Neural Ordinary Differential Equations.

4.2. Forward Propagation on the Graph. In GGNNs, the activation values of all layers need to be preserved after forwarding propagation, because these activation values are used to make backpropagation gradients on the computational path. However, it takes up an ample memory space, which makes the training process of the network limited. Therefore, we use GGNNs to parameterize the derivative of the hidden state, rather than directly parameterizing the hidden state as usual. It brings two benefits: (1) the level and parameters of continuity are implemented within the graph network; (2) the continuous graph network space eliminates the need for hierarchical propagation gradients and parameter updates.

In order to make the network hierarchy continuous, it is required that the error between hidden layers should be close to infinity within a network. When our GGNNs are added to a hidden layer that approaches infinity, the network can be considered to be contiguous. We represent this continuous transformation as an Ordinary Differential Equation:

$$\frac{\mathrm{d}S(t)}{\mathrm{d}t} = g(S(t), t, \theta), \tag{2}$$

where g denotes the Gated Neural Network layer, t changes from the initial to the end. The change in $S(t)$ represents the forward propagation result. At this point, we can see that we only need to find the solution of the equation, which is equivalent to completing the forward propagation. Formally, we need to transform the above formula to find the solution we need. Given the initial state $S(t_{\text{ini}})$ and the Gated Graph Neural Networks, the hidden state $S(t_{\text{ter}})$ of the end time is solved:

$$\begin{aligned} \frac{\mathrm{d}S(t)}{\mathrm{d}t} &= g(S(t), t, \theta), \\ \int_{t_{\text{ini}}}^{t_{\text{ter}}} \mathrm{d}S(t) &= \int_{t_{\text{ini}}}^{t_{\text{ter}}} g(S(t), t, \theta)\mathrm{d}t, \\ S(t_{\text{ter}}) &= S(t_{\text{ini}}) + \int_{t_{\text{ini}}}^{t_{\text{ter}}} g(S(t), t, \theta)\mathrm{d}t. \end{aligned} \tag{3}$$

Note that both $S(t_{\text{ini}})$and $\int_{t_{\text{ini}}}^{t_{\text{ter}}} g(S(t), t, \theta)\mathrm{d}t$ can be solved by ODEsessSolver. At this point, we solved the termination state $S(t_{\text{ter}})$, which is equivalent to the completion of the forward propagation. This method has a high degree of maturity and recognition in the field of mathematics, we only need to regard it as a "black box solver."

We gave pseudocode for forwarding propagation shown in Table 2. As you can see, we can define a neural" network and an ordinary differential equation solver. The solution results can be obtained by "feeding" the initial state, the final state, and the neural network to the solver.

Among them, gatedNet is defined as a Gated Graph Neural Network, S and t, respectively, represent state and time, and West Tower is a learnable parameter of the model.

4.3. Reverse-Mode Automatic Differentiation on the Graph. To make the GGNNs' network hierarchy continuous, the challenge is how to make the gradient pass through ODEsessSolver. If the gradient is reversed back along the calculation path of forwarding propagation, it is very intuitive. However, the memory usage will be extensive, and the numerical error cannot be controlled. As stated, we treated the forward-propagating ODEsessSolver as a "black box operation," and the gradient does not need to be passed in at all, just bypassing it. We learned in the previous section that it uses GGNNs to parameterize the derivative of the hidden state, where the derivative of the parameterized hidden state is similar to

Table 2: Forward propagation procedure of model.

Pseudo-code for the forward propagation procedure on the graph
def $g(S, t, \theta)$
Return gatedNet $([S, t], \theta)$
def sess-ODEnet (S, θ)
Return ODEsessSolver (gatedNet, $S, t_{\text{ini}}, t_{\text{ter}}, \theta$)

constructing the hierarchy and parameters of continuity, rather than discrete levels. So the parameter is also a continuous space; we do not need to layer the gradient and update the parameters. It should be noted that ODE does not store any intermediate results during forwarding propagation, so it only needs to approximate the memory cost of the constant level.

In this way, taking the initial state $S(t_{\text{ini}})$ and the terminating state $S(t_{\text{ter}})$ as an example, we can give the loss function of the backpropagation:

$$\begin{aligned} L\left(S\left(t_{\text{ter}}\right)\right) &= L\left(S\left(t_{\text{ini}}\right)\right) + \int_{t_{\text{ini}}}^{t_{\text{ter}}} g\left(S(t), t, \theta\right)\mathrm{d}t \\ &= L\left(\text{ODEsessSolver}\left(S\left(t_{\text{ini}}\right), g, t_{\text{ini}}, t_{\text{ter}}, \theta\right)\right). \end{aligned} \tag{4}$$

It should be noted that the input to the loss function is the result of ODEsessSolver. It can be concluded from the above equation that the optimization problem is a gradient optimization problem that is converted to θ.

We use the adjoint sensitivity method [39] to calculate the inverse gradient. This method calculates the gradient by solving the second augmented Ordinary Differential Equation backward. This method is linear with the size of the problem, has low memory costs, and can explicitly control numerical errors. In this method, the dependence of the descending gradient on the hidden state $S(t)$ at each time point is defined as an adjoint $a(t)$ and has $a(t) = \partial L/\partial S(t)$. On every instant, there are

$$\frac{\mathrm{d}a(t)}{\mathrm{d}t} = -a(t)^T \frac{\partial g\left(S(t), t, \theta\right)}{\partial S}. \tag{5}$$

Among them, the accompanying amount of the initial time point t_{ini} can be directly solved by the ordinary differential equation. For $[t_1, \ldots, t_n]$, it can be calculated backward from its final value. For parameter θ, its gradient depends on the current hidden state $S(t)$ and the accompanying amount $a(t)$:

$$\frac{\partial L}{\partial \theta} = -\int_{t_{\text{ter}}}^{t_{\text{ini}}} a(t)^T \frac{\partial g\left(S(t), t, \theta\right)}{\partial \theta}\,\mathrm{d}t. \tag{6}$$

Among them, $a(t)^T (\partial g/\partial S)$ and $a(t)^T (\partial g/\partial \theta)$ are vector-Jacobian products [40]. They can all be evaluated by the automatic differentiation method [41–43]. The integrals forms of S, a, and $\partial L/\partial \theta$ are solved by ODE. In the solution process, the original state, the accompanying state, and other partial derivatives of the nodes in the layer at each moment are connected into a single vector. Note that when Loss relies on the intermediate state, we use the reverse-mode automatic differentiation [37,38,42] to decompose the derivative into a series of solutions according to the time interval between each successive output pair.

The method of reverse automatic differentiation is different from the backward propagation method of the neural networks. It means that we can get his termination state at any time through the user's initial state [44,45].

4.4. Session Recommendation Using ODEsessSolver. In the previous section, we have already mentioned that our goal is to find a way to model conversations differently from traditional sequential learning ideas. Since the user's behavior exhibits a continuous state or time-series data with different frequencies, durations, and starting points, we may have the same behavior at different points in time for different reasons. For example, we went to the pharmacy twice to buy painkillers, but one was because of a headache, and the other was because of a toothache. It makes it easy to tell from the context of the user's behavior that these are two different situations. At present, the application of modeling of this idea is not enough. Such irregular time-series data can be divided into discrete steps of weeks, days, or even hours by parallel sessions.

In Section 4.1, we have included some details of the gated graph neural network and the ordinary differential equation solver. To achieve nonindependent and identically distributed session prediction, they are critical tools. In this section, we use the above ideas and tools to implement recommendation-based dependent sequence prediction. We presented model diagrams (Figure 1) to understand the recommendation process better.

Give the observed time $T = \{t_{\text{ini}}, t_1, \ldots, t_n\}$, an initial state $S(t_{\text{ini}})$. Again, ODEsessSolver is used to calculate the potential state $S = \left\{S_{t_{\text{ini}}}, S_{t_1}, \ldots, S_{t_n}\right\}$ representing each time point, while generating the sampled output $I_t = \left\{I_{t_{\text{ini}}}, I_{t_1}, \ldots, I_{t_n}\right\}$ for each potential state in any time. Our model can be defined as follows:

$$\begin{aligned} S_{t_{\text{ini}}} &\sim P\left(S_{t_{\text{ini}}}\right), \\ S_{t_1}, S_{t_2}, \ldots, S_{t_n} &= \text{ODEsessSolver}\left(S_{t_1}, g, \theta_g, t_{\text{ini}}, t_1, \ldots, t_n\right), \\ I_{t_i} &\sim P\left(I \mid S_{t_i}, \theta_i\right), \end{aligned} \tag{7}$$

where each layer of the Gated Graph Neural Network takes the corresponding S at the current time point and outputs the gradient $\partial S(t)/\partial t = g\left(S(t), \theta_g\right)$. After the GGNNs consume the data in an orderly manner, the posterior probability of each sample is output, that is, the probability of the item we use for prediction:

$$q\left(S_{t_{\text{ini}}} \mid I_{t_{\text{ini}}}, I_{t_1}, \ldots, I_{t_n}\right). \tag{8}$$

For GGNNs, each of its graph network layers g is time-invariant and given any potential state, S_t; its anti-pattern derivative trajectory should be unique. At any time, we can make any session prediction forward or backward. For example, if the initial session state $S_{t_{\text{ini}}}$ is the current input session, its potential trajectory to the termination state $S_{t_{\text{ter}}}$ should be unique. Extrapolation from this termination state allows for the prediction of potential states in future time.

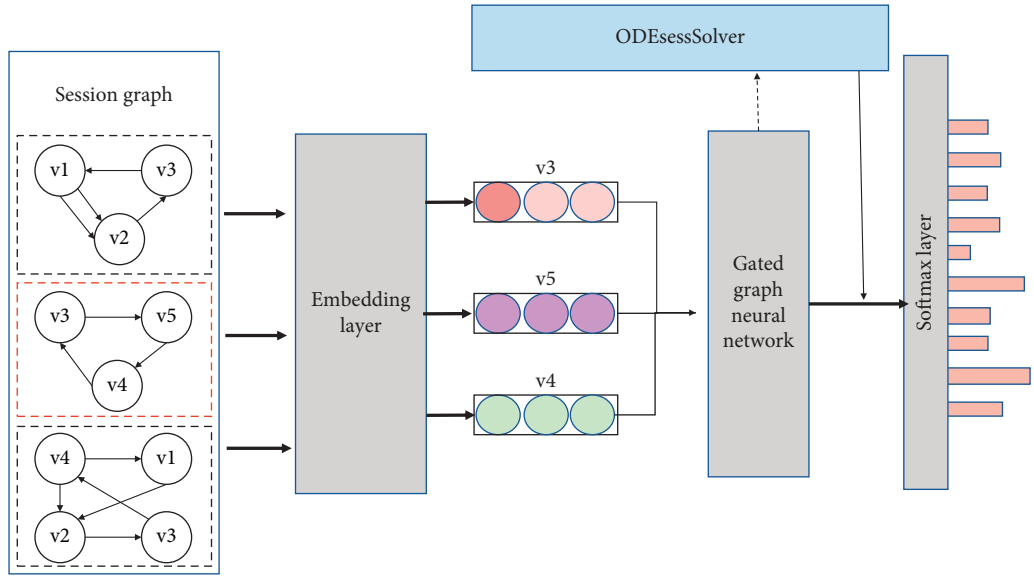

Figure 1: A macro view of the SessODEnet model. On the whole, the neural ordinary differential equations model consists of four parts, namely, the construction of the session graph, the learning of the gated graph neural network, the ordinary differential equation solver, and the final output.

That is, by solving the termination state $S_{t_{ter}}$, we can predict the potential behavior of the user for a certain period time from this point in the future.

According to the results of the ordinary differential equation solver, we used softmax to get the results of multi-classification. In fact, we only need to get the first few items of the forecast.

We gave a diagram (Figure 2) containing this example to feel the process visually:

5. Experimental Design

5.1. Experimental Settings. Our experiment was to verify the correctness of the idea of applying Neural Ordinary Differential Equations to conversational recommendations. In the context of session-based sequence learning, we chose Gated Graph Neural Networks as the recognition network and explored the user's preferred sessions using the Latent Neural Ordinary Differential Equations Model proposed in [21]. We used a learning rate of 0.005 in training and used the Adam optimizer for optimization learning. To alleviate the overfitting, we use the L2 loss function and take advantage of the early-stop training method. Ten epochs without improvement mean that training saturation stops training. Our experiments were implemented on Windows 10, Python 3.6, and pytorch 1.0 frameworks and accelerated using TITAN XP GPUs.

5.2. Experimental Datasets. We tested for four real datasets, including two large datasets in the form of conversations (Yoochoose (http://2015.recsyschallenge.com/challege.html) 1/64 and Diginetica (http://cikm2016.cs.iupui.edu/cikm-cup)), a dataset containing the user's music playback history (separated by timestamp) (Last.fm (https://grouplens.org/datasets/hetrec-2011/) And a user behavior data set Retailrocket (https://www.kaggle.com/retailrocket/ecommerce-dataset) (Users' clickstream data).

The Yoochoose dataset is from RecSys Challenge 2015, which contains user clickstreams on e-commerce sites within 6 months. The Diginetica dataset is from the CIKM Cup 2016, which uses only its transaction data. After removing sessions of length 1 and too few occurrences, we can get the data shown in Table 3. These two datasets are classic datasets based on session recommendations. These two datasets are commonly used for session-based recommendations. We used a data preprocessing method similar to [36] because this processing method allows us to get a clearer input.

Last.fm is a dataset about music recommendations. The data includes a list of the most popular artists, as well as the number of plays and tags for songs. Retailrocket dataset is the behavioral data of a real e-commerce website user. It includes, a slightly shorter period of time than Yoochoose, only 4.5 months of website visitor behavior data. In the dataset, there are three types of user behaviors: browsing, adding to shopping cart, and transaction. We used Table 4 to show the specific data of these two data sets. These two datasets are commonly used in recommendation systems based on user behavior analysis. They are characterized by easy access to user behavior in the form of sequences.

5.3. Baseline and Metrics

5.3.1. Baseline. Based on our mission objectives, we chose the baseline algorithms below to compare our experimental results.

POP: it is a popular model that recommends the most popular items in the training set for the user.

S-POP: it recommends the most popular items in the current session.

GRU-Rec: it is a user sequence model built for session-based recommendations using cyclic neural networks

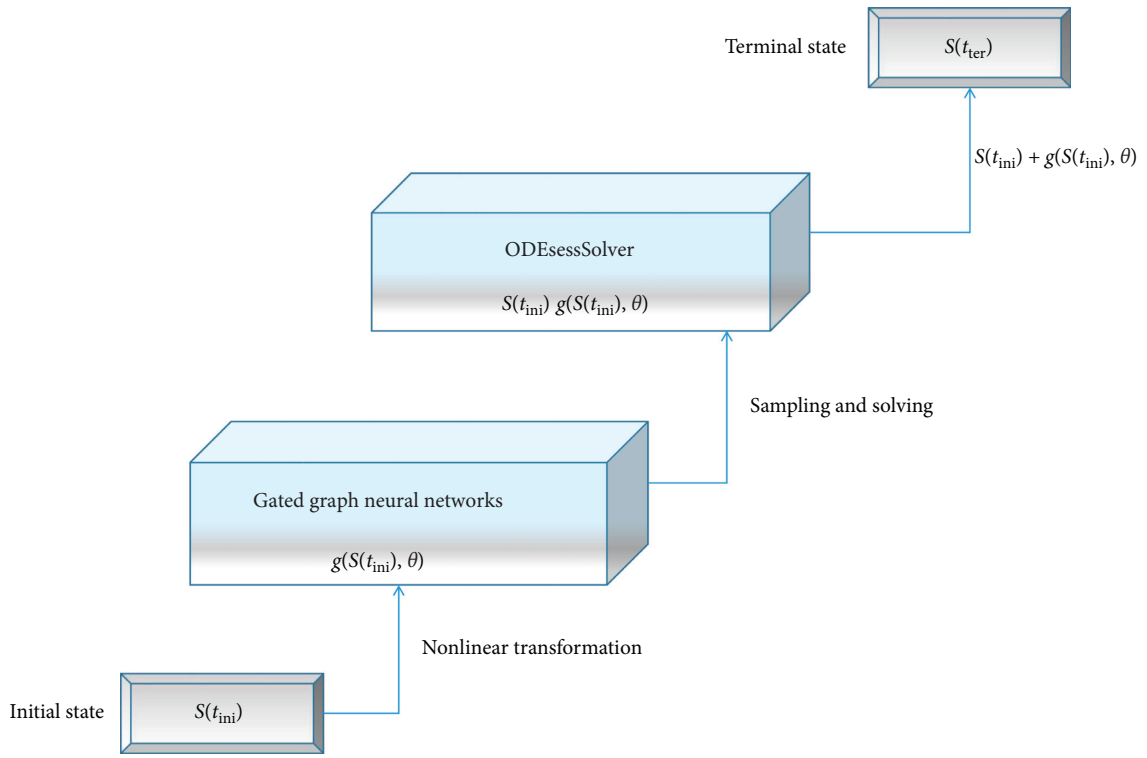

Figure 2: The process of processing the data by the model. It depicts the flow from the initial state to the termination state. For the input, we nonlinearly transform the input through gated graph neural networks (GGNNs) to get (g). ODEsessSolver integrates the neural network g and adds the initial state value to get the final prediction.

Table 3: Detailed statistics of the yoochoose 1/64 and diginetica datasets.

Datasets	Clicks	Train_sessions	Test_sessions	Items	Average length
Yoochoose 1/64	557248	369859	55898	16766	6.16
DigineticaI	982961	719470	60858	43097	5.12

Table 4: Detailed statistics of the Last.fm and retailrocket datasets.

Datasets	User	Items	Events
Last.fm	1892	17632	186479
Retailrocket	1407580	417053	2756101

(RNNs). It models the session by using a deep-loop neural network consisting of GRU cells.

NARM: the Neural Attentive Recommendation Machine. Based on the cyclic neural network, the attention mechanism is added. Based on the analysis of the sequential behavior of the cyclic neural network, the main behavior of the user is more closely concerned.

SR-GNN: this model was proposed by ShuWu et al. in January 2019 to aggregate separated session sequences into graph structure data. Through the Graph Neural Networks, the global session preference and local preference are comprehensively considered.

5.3.2. Metrics. We use the two evaluation indicators commonly used in the recommendation system, namely, the recall rate and the average reciprocal ranking, to qualitatively evaluate the experiment.

Recall@S: it is a very important one of the recommended system evaluation indicators and is used to measure the recall rate of the first S items in all test instances in the recommendation list. Recall@20: It represents the proportion of correctly recommended items in the top 20 items. Recall@50: It represents the proportion of correctly recommended items in the top 50 items.

Mean Reciprocal Rank: it measures the ranking of the predicted positions of the real target items in all test cases and counts them down and averages them as to accuracy.

MRR@20: it represents the average of the peer-level levels of the correctly recommended items in the top 20 items.

5.4. Results. We tested on four data sets. To train the model's ability to predict the data at irregular time points, we randomly selected the time points for extraction in each trajectory. At the same time, each time of new input is connected to the next predicted time difference to improve further the ability of the Gated Graph Neural Network to observe irregularities. We showed the experimental results in Table 5.

Table 5: Performance of our model compared to the baseline algorithms.

Dataset	Diginetica			Yoochoose 1/64			Last.fm			Retailrocket		
	Recall@20	Recall@50	MRR@20	Recall@20	Recall@50	MRR@20	Recall@20	Recall@50	MRR@20	Recall@20	Recall@50	MRR@20
POP	10.91	—	0.23	6.71	—	1.65	3.40	—	3.40	—	—	—
S-POP	21.07	—	14.69	30.44	—	18.35	19.36	—	11.47	—	—	—
GRU-rec	43.82	—	15.46	60.64	—	22.89	31.46	—	1.57	51.23	—	19.96
SR-GNN	—	—	17.59	—	—	30.94	41.20	—	21.89	66.60	—	30.18
NARM	62.58	—	27.35	68.32	—	28.76	37.73	—	13.12	68.95	—	28.33
Sess-ODEnet	69.98 + 7.4%	74.11	30.07 + 2.72%	69.19 + 0.87%	70.13	29.38 − 1.56%	49.66 + 8.46%	51.09	18.13 − 3.76%	71.15 + 2.2%	71.83	29.31 − 0.87%

5.5. Result Analysis. We proved our conjecture through experiments on four real datasets. We analyze the results in two ways:

Our model gives a new definition of the specific potential state of a user session over a continuous period of time. In our baseline algorithm, NARM and SR-GNN are the models based on the session recommendation proposed in the past two years. When the model predicts user behavior, the results they get should be very similar, because these models are all proposed on independent and identically distributed assumptions. Nevertheless, our model is more focused on the different companion states that users present in different periods. These adjoint states indicate that although the user has generated the same behavior at different points in time (e.g., clicks, purchases), it may be for different reasons.

The ability to model user sessions in complex spaces is enhanced. Most of the recommended algorithms use RNNs or GRU to model user sequences. However, in a complex recommendation system, although the user's action behavior exhibits a sequence state, the inside is still closely related in the form of a graph network. In our model, the Gated Graph Neural Network combined with the Ordinary Differential Equation preserves the ability of the network to model complex data, enabling it to propagate in a contiguous space.

The model has a lower complexity in the solution process. When we use ordinary differential equations to solve the problem, we usually do not need to solve the complete form of the solution of the equation, as long as the obtained solution is gradually close to the optimal value. The Ordinary Differential Equation Solver eliminates the need for hierarchical propagation gradients and parameter updates by parameterizing the derivative of the network's hidden state [39]. We can get the desired result by solving the network once, without the need for a gradient-like approach. Therefore, our model usually has a constant storage cost for ordinary differential equations.

6. Conclusion

We must realize that session recommendation plays a vital role in the user's implicit preference mining, but it cannot be considered as an independent and identical distribution. We propose a recommendation model based on Neural ODE: Sess-ODEnet. The model combines differential equations with gated graph neural networks to model complex sessions. The model derives the representation vector of each embedded item by representing the session as the structure of the conversation graph and then through the Graph Neural Network. On this basis, the ODE solver is used to predict and recommend nonindependent intentions at any point in time. Not only that, Neural ODE is different from traditional neural network training methods, which makes our models have not only low memory usage but complicated search time as well.

However, the focus on this work is still to regard the user session as a continuous state in time. In the future, we hope to apply this model to the sequence of behaviors lacking time steps and to model the nonindependent intentions of users at higher levels, with recommendations [14].

Data Availability

The illustrative example data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by Shandong Natural Science Fund Project (No. ZR2017LF021).

References

[1] A. Jabakji and H. Dag, "Improving item-based recommendation accuracy with user's preferences on Apache Mahout," in *Proceedings of the IEEE International Conference on Big Data*, Washington, DC, USA, December 2016.

[2] S. Kant and T. Mahara, "Merging user and item based collaborative filtering to alleviate data sparsity," *International Journal of System Assurance Engineering & Management*, vol. 9, no. 1, pp. 173–179, 2018.

[3] P. Thakkar, K. Varma, and V. Ukani, "Outcome fusion-based approaches for user-based and item-based collaborative filtering," in *Information and Communication Technology for Intelligent Systems (ICTIS 2017)*, pp. 127–135, Springer, Cham, Switzerland, 2017.

[4] F. Font, J. Serrà, and X. Serra, "Class-based tag recommendation and user-based evaluation in online audio clip sharing," *Knowledge-Based Systems*, vol. 67, no. 3, pp. 131–142, 2014.

[5] C. Wang, Z. G. Zhu, Y. X. Zhang et al., "Improvement in recommendation efficiency and personalized of user-based collaborative filtering algorithm," *Journal of Chinese Computer Systems*, vol. 37, no. 3, pp. 428–432, 2016.

[6] P. Wang, Q. Qian, Z. Shang, and J. Li, "An recommendation algorithm based on weighted Slope one algorithm and user-based collaborative filtering," in *Proceedings of the 2016 Chinese Control and Decision Conference (CCDC)*, Yinchuan, China, May 2016.

[7] F. Yang, Y. Zheng, and C. Zhang, "Hybrid recommendation algorithm based on probability matrix factorization," *Journal of Computer Applications*, vol. 38, no. 3, pp. 644–649, 2018.

[8] V. S. Dixit, S. Gupta, and P. Jain, "A propound hybrid approach for personalized online product recommendations," *Applied Artificial Intelligence*, vol. 32, no. 9-10, pp. 1–17, 2018.

[9] G. Sottocornola, F. Stella, M. Zanker, and F. Canonaco, "Towards a deep learning model for hybrid recommendation," in *Proceedings of the International Conference on Web Intelligence*, Leipzig, Germany, August 2017.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] L. Jia, X. Hua, X. He, J. Deng, and X. Sun, "Tweet modeling with LSTM recurrent neural networks for hashtag

recommendation," in *Proceedings of the International Joint Conference on Neural Networks*, Vancouver, Canada, July 2016.

[12] S. T. Jishan and Y. Wang, "Audience activity recommendation using stacked-LSTM based sequence learning," in *Proceedings of the International Conference on Machine Learning & Computing*, Valencia, Spain, April 2017.

[13] T. Bansal, D. Belanger, and A. Mccallum, "Ask the GRU: multi-task learning for deep text recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems—RecSys '16*, Boston, MA, USA, September 2016.

[14] Q. Cui, S. Wu, Y. Huang, and L. Wang, "A hierarchical contextual attention-based GRU network for sequential recommendation," *Neurocomputing*, vol. 358, pp. 141–149, 2019.

[15] E. Smirnova and F. Vasile, "Contextual sequence modeling for recommendation with recurrent neural networks," in *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems—DLRS 2017*, Como, Italy, August 2017.

[16] S. Wu, W. Ren, C. Yu, G. Chen, D. Zhang, and J. Zhu, "Personal recommendation using deep recurrent neural networks in NetEase," in *Proceedings of theIEEE International Conference on Data Engineering*, Helsinki, Finland, May 2016.

[17] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining—KDD '19*, Anchorage, AK, USA, August 2019.

[18] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM Sigkdd International Conference on Knowledge Discovery & Data Mining*, Halifax, Canada, August 2017.

[19] D. Sorokin and I. Gurevych, "Modeling semantics with gated graph neural networks for knowledge base question answering," in *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, NM, USA, August 2018.

[20] D. Beck, G. Haffari, and T. Cohn, "Graph-to-sequence learning using gated graph neural networks," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Melbourne, Australia, July 2018.

[21] R. T. Q. Chen, Y. Rubanova, J. Bettencourt et al., "Neural ordinary differential equations," 2018, http://arxiv.org/abs/1806.07366.

[22] C. R. B. Bonin, G. C. Fernandes, R. W. Dos Santos, and M. Lobosco, "Mathematical modeling based on ordinary differential equations: a promising approach to vaccinology," *Human Vaccines Immunotherapeutics*, vol. 13, no. 2, pp. 484–489, 2017.

[23] D. A. Lyakhov, V. P. Gerdt, and D. L. Michels, "Algorithmic verification of linearizability for ordinary differential equations," in *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*, pp. 285–292, ISSAC'17, Kaiserslautern, Germany, July 2017.

[24] X. Song and X. Lv, "The relationships between some types of partial differential equations and ordinary differential equations as well as their applications," *Mathematical Methods in the Applied Sciences*, vol. 41, no. 5, pp. 2152–2161, 2018.

[25] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2015, http://arxiv.org/abs/1511.06939.

[26] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems—RecSys '17*, p. 344, ACM, Como, Italy, August 2017.

[27] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management-CIKM'17*, pp. 1419–1428, New York, NY, USA, November 2017.

[28] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *Proceedings of the 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, IEEE, Vietri sul Mare, Salerno, Italy, September 2016.

[29] M. Grbovic and H. Cheng, "Real-time personalization using embeddings for search ranking at airbnb," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 311–320, ACM, London, UK, August 2018.

[30] J. Yu, S. Yi, and Z. Yang, "Topic-STG: extending the session-based temporal graph approach for personalized tweet recommendation," in *Proceedings of the 23rd 2014 International Conference on World Wide Web Companion*, Seoul, Korea, April 2014.

[31] M. Ruocco, O. S. L. Skrede, and H. Langseth, "Inter-session modeling for session-based recommendation," in *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems-DLRS 2017*, Como, Italy, August 2017.

[32] F. Mi and B. Faltings, "Context tree for adaptive session-based recommendation," 2018, http://arxiv.org/abs/1806.03733.

[33] P. Christodoulou, S. P. Chatzis, and A. S. Andreou, "A variational latent variable model with recurrent temporal dependencies for session-based recommendation (VLaReT)," *Lecture Notes in Information Systems and Organisation*, Springer, Berlin, Germany, pp. 51–64, 2018.

[34] F. Fouss, K. Francoisse, L. Yen, A. Pirotte, and M. Saerens, "An experimental investigation of kernels on graphs for collaborative recommendation and semisupervised classification," *Neural Networks*, vol. 31, no. 7, pp. 53–72, 2012.

[35] W. Fan, Y. Ma, Q. Li et al., "Graph neural networks for social recommendation," in *Proceedings of the World Wide Web Conference*, San Francisco, CA, USA, May 2019.

[36] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 346–353, 2019.

[37] E. Dupont, A. Doucet, and Y. W. Teh, "Augmented neural odes," 2019, http://arxiv.org/abs/1904.01681.

[38] W. Xu and T. F. Coleman, "Solving nonlinear equations with the Newton-Krylov method based on automatic differentiation," *Optimization Methods and Software*, vol. 29, no. 1, pp. 88–101, 2014.

[39] M. L. Chambers, L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, E. F. Mishchenko, and D. E. Brown, "The mathematical theory of optimal processes," *OR*, vol. 16, no. 4, pp. 493-494, 1965.

[40] A. Walther, A. Griewank, and A. Best, "Multiple vector–Jacobian products are cheap," *Applied Numerical Mathematics*, vol. 30, no. 2, pp. 367–377, 2012.

[41] C. Mitev, "Verifying Jacobian Sparsity," in *Automatic Differentiation of Algorithms*, Springer, Berlin, Germany, 2002.

[42] A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei, "Automatic differentiation variational inference," *The Journal of Machine Learning Research*, vol. 18, no. 1, 2016.

[43] K. Kristensen, A. Nielsen, C. W. Berg, H. Skaug, and B. M. Bell, "TMB: automatic differentiation and laplace

approximation," *Journal of Statistical Software*, vol. 70, no. 5, 2016.

[44] A. Griewank, "Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation," *Optimization Methods and Software*, vol. 1, no. 1, pp. 35–54, 1992.

[45] W. Xu, X. Chen, and T. F. Coleman, "The efficient application of automatic differentiation for computing gradients in financial applications," *The Journal of Computational Finance*, vol. 19, no. 3, pp. 71–96, 2016.