An Adaptive Data-Driven Imputation Model for Incomplete Event Series

Jiadong Chen, Hengyu Ye, Xiaofeng Gao(✉), Fan Wu, Linghe Kong, and Guihai Chen

MoE Key Lab of Artificial Intelligence, Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
{chenjiadong998,cs_22_yhy,linghe.kong}@sjtu.edu.cn,
{gao-xf,fwu,gchen}@cs.sjtu.edu.cn

Abstract. Event sequences play as a general fine-grained representation for temporal asynchronous event streams. However, in practice, event sequences are often fragmentary and incomplete with censored intervals or missing data, making it hard for downstream prediction and decision-making tasks. In this work, we propose a fresh extension on the definition of the temporal point process, which conventionally characterizes chronological prediction based on historical events, and introduce *inverse point process* that characterizes counter-chronological attribution based on future events. These two point process models allow us to impute missing events for one partially observed sequence with conditional intensities in two symmetric directions. We further design a peer imitation learning algorithm that lets two models cooperatively learn from each other, leveraging imputed sequences given by the counterpart as the supervised signal. The training process consists of iterative learning of two models and facilitates them to achieve a consensus. We conduct extensive experiments on both synthetic and real-world datasets, which demonstrate that our model can recover incomplete event sequences very close to the ground-truth, with averagely 49.40% improvement compared with related competitors measured by normalized optimal transport distance.

Keywords: Event Sequence · Temporal Point Process · Missing Data · Sequence Imputation

1 Introduction

Event sequences record temporal events by a sequence of tuples (m_i, t_i), where m_i denotes discrete event markers (i.e., types) and t_i denotes continuous timestamps. Event sequences play as a general fine-grained representation for event streams that are ubiquitous in many real-life applications, e.g., 1) medical records for patients in hospital [5]; 2) user's activity and behaviors in social networks [16]; 3) individual's visiting points of interests in a large city [18], etc. One common way for modeling event sequences is to treat arrival of each event as a random variable and formulate an event sequence as a stochastic point process of

X. Yang et al. (Eds.): ADMA 2023, LNAI 14176, pp. 3–18, 2023.
https://doi.org/10.1007/978-3-031-46661-8_1

which a (conditional) intensity function is introduced to characterize the conditional distribution of next event given historical events. For point process, prior works [27] attempt to model intensity functions in statistical aspects, while some recent works [22] leverage deep neural networks to learn a more expressive neural intensity model. All of these methods assume that input event sequences are complete based on which one can learn a robust representation via the point process model.

However, in most practical situations, the collected event sequences are usually fragmentary and partially observed due to common missing-data mechanisms. One typical example could be medical records which chronologically mark down symptoms, diagnoses, and medications for patients. Data from one or few hospitals cannot guarantee completeness since patients usually come over to different hospitals for medical assistance. Therefore, it is greatly in demand to build a theoretically sound and practically effective approach to recover partially observed event sequences. To solve the above-mentioned problem, one may confront three non-trivial challenges. The first question is *how to build a model that possesses enough expressiveness to estimate missing data with any possible timestamp and marker for any given observed sequence (Q1)*. Second, there is no ground-truth information that can be used as supervision for learning, so a following question arises: *how to design a self-supervised approach for model learning on event sequence imputation (Q2)* ? Third, even if one manages to define or construct some artificial signals as supervision, it is hard to characterize the discrepancy between an estimated sequence and a target one, which induces another question: *how to properly define optimization objective (Q3)*?

In this paper, we propose a novel Peer Imitation Learning point process model for Event Sequence imputation (PILES) that efficiently solve the above three challenging questions. To answer Q1, we propose an acceptance-rejection strategy that can adaptively sample missing events based on a neural intensity function parametrized by bidirectional Long-Short Term Meomory (LSTM) network. The neural intensity model guarantees enough representation capacity for event sequences with any latent process, while the acceptance-rejection strategy can incrementally sample missing events between any two observed events in an input sequence. To answer Q2, we extend the definition of temporal point process, which conventionally characterizes chronological prediction for next events based on historical ones, and pioneeringly propose inverse point process model that characterizes counter-chronological attribution for previous events based on future ones. The two point process models allow us to tackle one partially observed sequence in a bidirectional view where we sample missing events via conditional intensities in two directions. Then the sampling results given by one model can naturally be used as supervision for another model, through which we can achieve self-supervised learning.

To answer Q3, i.e., measuring distance between imputed event sequences given by two models, we propose a novel peer imitation learning algorithm that enforces consistency between one model's output (as model policy) and the counterpart's (as expert policy). Similar to GAIL [9], we introduce discriminators

and adversarial training to learn better consistency measurement, which equivalently minimizes the Jensen-Shannon divergence between two model distributions. Extensive experiment results on four synthetic datasets and two real-world datasets demonstrate that PILES is capable of giving much closer reconstructed event sequences to ground-truth event sequences compared with unsupervised competitors, with an averaged improvement of 49.40% measured by normalized optimal transport distance. Also, ablation studies show the necessity of proposed units for superior performance.

Our contributions are summarized as follows.

i) **New Aspect:** We propose an inverse point process model that characterizes conditional intensities for previous events in a direction of attribution, which can cooperate with conventional (forward) point process model to deal with sequence imputation in unsupervised situation. To our knowledge, this is the first work on fully unsupervised imputation for asynchronous event sequences.
ii) **Methodology:** We propose a novel peer imitation learning approach to synergize two point process models via pushing them to achieve a consensus, which enables self-supervised learning in unsupervised case.
iii) **Experiments:** We conduct experiments on six datasets with different latent structures and demonstrate convincing superiority of new model and its generality in three missing-data situations. All source code and data will be made publicly available.

2 Related Works

In this section, we briefly discuss related works and highlight their differences from our work. For event sequence inference, the work [6] proposes an importance sampling-based algorithm to analyze marked event sequences under the condition of continuous time Bayesian Networks [15], and develops filtering and smoothing algorithms to deal with incomplete data. Similarly, [17] designs an inference framework based on Markov Chain Monte Carlo (MCMC), and particularly employs a jumping strategy for missing data in sequences. However, these two approaches heavily rely on specific parametric models with strong prior assumptions, suffering from limitations for generalization to event streams with different latent structures. Besides, [12] considers a more general neural intensity function, integrating Z-transform [1] for learning model parameters. Such a method could only handle uni-dimensional events and would fail when it comes to high-dimensional event markers.

For event sequence imputation, [19] and [21] focus on specific application scenarios. The former targets intrapolated estimation for blood glucose concentration given specific time in an observed sequence of patient's medical records, while the latter aims to predict interactions between two users in social networks given a timestamp in a sequence of user activity. In these methods, notably, the arrival time of target events is assumed to be known in advance, which plays as informative features and substantially reduces the problem difficulty. Moreover, [20] puts forward an MCMC-based model to recover incomplete event sequences

driven by Hawkes process [8]. However, the method assumes time intervals that contain missing events to locate specific positions for event imputation. Also, the Hawkes process used in this model introduces strong inductive bias and suffers from limitations when it comes to more general point processes. Recently, [14] develops the Neural Hawkes Particle Smoothing method for imputing missing events, and however, it requires a large number of ground-truth complete sequences for training an intensity model that provides sufficient information for latent process in input event sequences as important prior information. Unfortunately, in most practical scenarios, one has no access to such complete event sequences.

3 Problem Formulation

An event sequence consists of a series of tuples $e_i = (m_i, t_i)$, where m_i stands for the type of an event (a.k.a. event marker) and t_i denotes continuous event timestamp. We use E to represent an event sequence composed of N marker-time tuples.

$$E = \{e_i\}_{i=1}^{N}, \quad \forall\ 1 \leqslant i \leqslant N: \qquad m_i \in \mathcal{M},\ t_i \in [0, T]\,.$$

where $\mathcal{M}$ is a marker set of size M.

Temporal Point Process. One common way to model event sequences is via point process [3] which treats the arrival of each event as a random variable given the history of previous events. To characterize the probability for when the next event would happen, a (conditional) intensity function is defined as $\lambda(t|\mathcal{H}_t) := \frac{\mathbb{P}(N(t+dt)-N(t)=1|\mathcal{H}_t)}{dt}$, where $\mathcal{H}_t$ and $N(t)$ denote the history of previous events and number of events before time t, respectively. The intensity function induces conditional density distribution for arrival time of the next event, $P(t|\mathcal{H}_t) = \lambda(t|\mathcal{H}_t)\exp(-\int_{t_n}^{t}\lambda(\tau|\mathcal{H}_t)d\tau)$. The marker of a new event usually obeys a certain categorical distribution $P(m|\mathcal{H}_t)$. There are various ways to specify the intensity function as parametric forms. For instance, the simplest case is Poisson process which has a constant intensity over time. Also, many previous works leverage various forms to parameterize the intensity function by assuming different inductive bias for event sequences, such as inhomogeneous Poisson process, Hawkes process, self-correcting process, etc. Recent works propose to use deep neural networks to model the intensities [25]. Beyond these expressive networks, [25] proposes to train a neural point process via generative adversarial networks, and [13,22] adopt reinforcement learning or imitation learning to further improve learning on asynchronous sequences.

Real-life event sequences are often partially discovered. Given an observed sequence $E = \{(m_i, t_i)\}$, we define U_i as a subsequence of missing events in time interval of two adjacent observed events $[t_i, t_{i+1}]$ and each missing event is denoted by $\varepsilon_{i,j} = (\mu_{i,j}, \tau_{i,j})$, where $\mu_{i,j}$ and $\tau_{i,j}$ denote the marker and time of the j-th unobserved event in U_i respectively. $U_i = \{\varepsilon_{i,j}\}_{j=1}^{n_i}$, $\forall\ 1 \leqslant j \leqslant n_i:\ \mu_{i,j} \in \mathcal{M},\ \tau_{i,j} \in [t_i, t_{i+1}]$. Note that U_i can be an empty set ($n_i = 0$) or

with arbitrary length n_i. Here we define $t_0 = 0$ and $t_{N+1} = T$. The ground-truth complete event sequence $\overline{E}$ is the union of observed sequence E and unobserved event sequences U_i, $\overline{E} = E \cup U_1 \cup U_2 \cup \cdots \cup U_N$.

Given a collection of partially observed event sequences $\{E_k\}_{k=1}^K$, our target is to build a model that can impute all the missing events (including timestamps of markers) in E_k, as an estimation for $U_{k,i}$, $i = 1, \cdots, N_k$, $k = 1, \cdots, K$. Note that the ground-truth complete sequences $\{\overline{E}_k\}$ are not available as input, and one can only leverage $\{E_k\}$ for model learning and inference. Besides, other prior information about missing data is unknown, such as total number of missing events or missing-data mechanisms (interval-censored or missing-at-random). We call this as extremely or fully unsupervised situation.

4 Proposed Method

We propose PILES, a Peer Imitation Learning point process model for Event Sequence imputation, to solve the problem in fully unsupervised setting. An illustration of our proposed framework is depicted in Fig. 1.

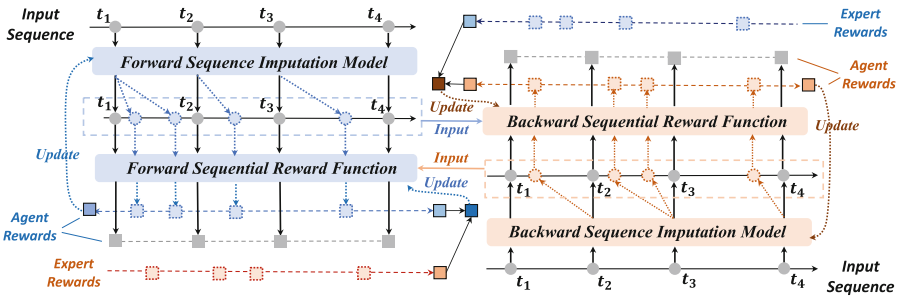


Fig. 1. An illustration of our peer imitation learning method for unsupervised event sequence imputation. Given input event sequence, missing events are imputed in both forward and backward directions.

4.1 Forward Imputation Model

Given a partially observed event sequence $E = \{e_1, e_2, \cdots, e_N\}$, our target is to estimate missing events between each pair of adjacent observed events e_i and e_{i+1}. Note that missing events may exist at any position in $[0, T]$, so we need an imputation model with enough expressiveness to accommodate all possible results. To this end, our forward imputation model chronologically samples new event markers and timestamps in an iterative manner from $t = 0$ to T, using newly imputed events to update historical sequence which gives conditional intensities for subsequent imputation.

Suppose we finish imputing events before observed event e_i and currently focus on imputing missing events between e_i and e_{i+1} and denote the latest imputed event as $\hat{\varepsilon}_{i,j}$, and then the current status of reconstructed sequence can be represented as $\hat{E}[i,j] = E \cup \hat{U}_1 \cup \hat{U}_2 \cup \cdots \cup \hat{U}_{i-1} \cup \{\hat{\varepsilon}_{i,1}, \cdots, \hat{\varepsilon}_{i,j}\}$, where $\hat{U}_p = \{\hat{\varepsilon}_{p,q}\}$ denotes set of imputed events in $[t_p, t_{p+1}]$ for $p = 1, \cdots, i-1$, and $\hat{\varepsilon}_{p,q} = (\hat{\mu}_{p,q}, \hat{\tau}_{p,q}) \in \hat{U}_p$ denotes the q-th imputed event between observed events e_p and e_{p+1}. Then the problem boils down to how to sample the next imputed event $\hat{\varepsilon} = (\hat{\mu}, \hat{\tau})$. We next introduce embeddings for events and neural intensity model that play as building blocks for our forward imputation model.

Event Embeddings. We embed each marker-time pair (m, t) into low-dimensional vectors via learnable embedding matrices. For event marker $m \in \mathcal{M}$, we first generate a one-hot representation $\mathbf{v} \in \{0,1\}^M$, and then multiply it with an embedding matrix $\mathbf{W}_M \in \mathbb{R}^{d \times M}$ to obtain an embedding vector $\mathbf{m} = \mathbf{W}_M \mathbf{v}$. To encode continuous timestamp $t \in \mathbb{R}$, we adopt a linear transformation $\mathbf{t} = \mathbf{w}_T t + \mathbf{b}_T$ to get its representation. Here $\mathbf{W}_M \in \mathbb{R}^{d \times M}$, $\mathbf{b}_T \in \mathbb{R}^d$, and $\mathbf{w}_T \in \mathbb{R}^d$ are all trainable parameters. The final representation vector $Emb(e)$ for event $e = (m, t)$ is the combination of marker and time representations with a integrating parameter β:

$$Emb(e) = \beta \cdot \mathbf{W}_M \mathbf{v} + (1 - \beta) \cdot (\mathbf{w}_T t + \mathbf{b}_T). \tag{1}$$

Neural Intensity Model. We further model intensity function over an event sequence using bidirectional LSTMs to concurrently utilize both history and future information. Such Bi-LSTM based encoding technique is widely used in event sequence modeling, such as recent work [14]. Given the current status of imputed sequence in Sect. 4.1 we define history subsequence $H[i,j]$ and future subsequence $F[i,j]$:

$$\begin{aligned} H[i,j] &= \{e_1, e_2, \cdots, e_i\} \cup \hat{U}_1 \cup \cdots \cup \hat{U}_{i-1} \cup \{\hat{\varepsilon}_{i,1}, \cdots, \hat{\varepsilon}_{i,j}\} \\ F[i,j] &= \{e_N, e_{N-1}, \cdots, e_{i+1}\} \end{aligned} \tag{2}$$

we encode $H[i,j]$ and $F[i,j]$ using two LSTMs $g_h(\cdot)$ and $g_f(\cdot)$ respectively to get the hidden states $\mathbf{h}_{i,j}, \mathbf{z}_{i,j} \in \mathbb{R}^H$,

$$\mathbf{h}_{i,j} = g_h\left(Emb(H[i,j])\right), \quad \mathbf{z}_{i,j} = g_f\left(Emb(F[i,j])\right), \tag{3}$$

where $Emb(H[i,j])$ and $Emb(F[i,j])$ denote the matrices generated by mapping each event in the sequence to an embedding vector using Eq. (1). From Eq. (2), we can see that the historical events, including both observed events and imputed events, are encoded to get the hidden state $\mathbf{h}_{i,j}$. In contrast, the future events, which only include observed events, are encoded to get $\mathbf{z}_{i,j}$. Then final hidden state $\mathbf{x}_{i,j} \in \mathbb{R}^{2H}$ can be set as the concatenation of $\mathbf{h}_{i,j}$ and $\mathbf{z}_{i,j}$: $\mathbf{x}_{i,j} = [\mathbf{h}_{i,j}, \mathbf{z}_{i,j}]$. Similar to [4], we define latent intensities $\boldsymbol{\lambda}_{i,j} \in \mathbb{R}^M$ over M marker classes $\boldsymbol{\lambda}_{i,j} = \mathbf{W}_h \mathbf{x}_{i,j} + \mathbf{b}_h$. where $\mathbf{W}_h \in \mathbb{R}^{(2H) \times M}$ and $\mathbf{b}_h \in \mathbb{R}^M$ are trainable parameters. Here each scalar element in $\boldsymbol{\lambda}_{i,j}$ represents conditional intensity for a certain class of marker.

Sampling Events for Imputation. We proceed to present how to sample next imputed events $\hat{\varepsilon} = (\hat{\mu}, \hat{\tau})$ given the conditional intensities. In general, we first sample marker $\hat{\mu}$ and obtain the corresponding intensity value from $\boldsymbol{\lambda}_{i,j}$ to estimate timestamp $\hat{\tau}$. Given the hidden state vector $\mathbf{x}_{i,j}$, the next marker $\hat{\mu} \in \mathcal{M}$ can be sampled from a softmax distribution

$$\hat{\mu} \sim P\left(m \mid H[i,j], F[i,j]\right) = \frac{\exp\left((\mathbf{W}_p \mathbf{x}_{i,j} + \mathbf{b}_p)_{[m]}\right)}{\sum_{m'=1}^{M} \exp\left((\mathbf{W}_p \mathbf{x}_{i,j} + \mathbf{b}_p)_{[m']}\right)}. \tag{4}$$

where $\mathbf{a}_{[k]}$ denotes the k-th element of vector $\mathbf{a}$. Here $\mathbf{W}_p \in \mathbb{R}^{(2H)\times M}$ and $\mathbf{b}_h \in \mathbb{R}^M$. Then the corresponding intensity λ can be indexed from $\boldsymbol{\lambda}_{i,j}$. Given the latest imputed timestamp $\hat{\tau}_{i,j}$, we can sample the time interval of next imputed event $\delta\hat{t}$ from a continuous distribution $\mathcal{T}(\delta t; \lambda)$ parametrized by $\lambda = \boldsymbol{\lambda}_{i,j[\hat{\mu}]}$. Similar to [22], $\mathcal{T}$ can be chosen as Exponential distribution and the time interval can be sampled from a conditional density:

$$\delta\hat{t} \sim P(\delta t \mid H[i,j], F[i,j]) = \mathcal{T}(\delta t; \lambda). \tag{5}$$

Then we have timestamp for next event as $\hat{\tau} = \hat{\tau}_{i,j} + \delta\hat{t}$. Here one can also consider estimation for $\delta\hat{t}$ using approximated expectation [4] or deterministic mapping [24]. Once we obtain a sampled event $\hat{\varepsilon} = (\hat{\mu}, \hat{\tau})$ as candidate, we need to check its validity by taking the following acceptance-rejection method to judge if it can be properly added to the current status of reconstructed sequence. If $\hat{\tau} < t_{i+1}$, i.e., the timestamp of the sampled event is before the next observed event, we keep it in the sequence, set $\hat{\varepsilon}_{i,j+1} = \hat{\varepsilon}$, and update the history and future subsequences for next imputation step:

$$\begin{aligned} H[i, j+1] &= H[i,j] \cup \{\hat{\varepsilon}_{i,j+1}\}, \\ F[i, j+1] &= F[i,j]. \end{aligned} \tag{6}$$

If the timestamp $\hat{\tau}$ exceeds the time of next observed event t_{i+1}, we discard $\hat{\varepsilon} = (\hat{\mu}, \hat{\tau})$ and continue to impute missing events between observed events e_{i+1} and e_{i+2} by updating the history and future subsequences:

$$\begin{aligned} H[i+1, j] &= H[i,j] \cup \{e_{i+1}\}, \\ F[i+1, j] &= F[i,j] \setminus \{e_{i+1}\}. \end{aligned} \tag{7}$$

Sequence Reconstruction. When we go to the next step with updated history and future subsequences given by Eq. (6) or Eq. (7), we can encode $H[i, j+1]$, $F[i, j+1]$ (resp. $H[i+1, j]$, $F[i+1, j]$) by Eq. (3) and continue to impute the next missing event. Note that here we do not need to encode the whole sequence again, since for historical events, we can simply update the hidden state $\mathbf{h}_{i,j+1}$ (resp. $\mathbf{h}_{i+1,j}$) using the newly included event $\hat{\varepsilon}_{i,j+1}$ (resp. e_{i+1}). For future events, we can inversely encode the whole sequence $\{e_N, \cdots, e_2, e_1\}$ before the imputation process, and store and reuse the results during the imputation process. Therefore,

our imputation model still maintains a linear time complexity w.r.t. the length of sequence. The whole process is executed iteratively until sampled timestamp reaches T, and we use $\hat{E}^F$ to denote the final imputed event sequence in forward direction: $\hat{E}^F = E \cup \hat{U}_1^F \cup \hat{U}_2^F \cup \cdots \cup \hat{U}_N^F$, where $\hat{U}_i^F$ denotes a subsequence of imputed events between e_i and e_{i+1}, i.e., $\hat{U}_i^F = \{(\hat{\mu}_{i,j}, \hat{\tau}_{i,j})\}, \ \forall\ 1 \leqslant j \leqslant \hat{n}_i$. Here we use a superscript F to highlight its forward direction for imputation to distinguish from the backward version in Sect. 4.2.

4.2 Inverse Point Process and Backward Model

While we obtain reconstructed event sequences, it is hard for model optimization since we have no ground-truth data as supervision. To solve the obstacle, we take a different view from traditional perspective. In temporal point process model, it conventionally characterize conditional intensities for future events given a history of events. In essence, it considers *prediction* in a chronological order. Why cannot we take an inverse perspective and tackle conditional intensities for historical events given future ones, which focuses on *attribution* in a counter-chronological order.

To this end, we introduce the concept of *Inverse Point Process*, which could be identified as a complementary latent model for traditional point process. We define conditional intensity function for inverse temporal point process as follows.

Definition 1. (Inverse Temporal Point Process)*: Given future events $F(t)$ after time t, conditional intensity $\lambda(t|F(t))$ is defined as the probability density of a possible hidden event existing in $[t - \Delta t, t]$, where $N'(t)$ represents the number of future events after time t.*

$$\lambda\left(t|F(t)\right) := \lim_{\Delta t \to 0} \frac{\mathbb{P}(N'(t) - N'(t - \Delta t) = 1 \, F(t))}{\Delta t}. \tag{8}$$

One can realize that the definitions of inverse point process and traditional point process are symmetric in nature. Both of them are based on countings and probability density of event occurrence within a unit period of time. The only difference lies in that the original definition uses countings of historical events, while inverse point process model considers future ones. Therefore, it is natural to consider a (symmetric) backward imputation model that can sample imputed events in input sequences from the future to the beginning.

Backward Imputation Model. Given a partially observed event sequence $E = \{e_1, e_2, \cdots, e_N\}$, similar to Sect. 4.1, we suppose the current imputation status represented by $\hat{E}[i, j] = E \cup \hat{U}_N \cup \hat{U}_{N-1} \cup \cdots \cup \hat{U}_{i+1} \cup \{\hat{\varepsilon}_{i,-j}, \cdots, \hat{\varepsilon}_{i,-1}\}$ and the latest imputed event is $\hat{\varepsilon}_{i,-j}$. We employ another two LSTMs $g'_f(\cdot)$ and $g'_h(\cdot)$ to encode history and future subsequences defined as

$$H'[i, -j] = \{e_1, e_2, \cdots, e_i\}$$
$$F'[i, -j] = \{e_N, \cdots, e_{i+1}\} \cup \hat{U}_{N-1} \cup \cdots \cup \hat{U}_{i+1} \cup \{\hat{\varepsilon}_{i,-j}, \cdots, \hat{\varepsilon}_{i,-1}\},$$

based on which we can obtain hidden state vector $\mathbf{x'}_{i,j}$ and intensity vector $\boldsymbol{\lambda}'_{i,j}$. After that, we sample a newly imputed event marker $\hat{\mu}$ and time interval $\delta\hat{t}$ using softmax distribution and exponential distribution as in Eq. (4) and Eq. (5) respectively, and the timestamp of next imputed event is calculated by $\hat{\tau} = \hat{\tau}_{i,-j} - \delta\hat{t}$. Then we conduct a similar acceptance-rejection strategy for the newly sampled $\hat{\varepsilon} = (\hat{\mu}, \hat{\tau})$. If $\hat{\tau} > t_i$, we set $\hat{\varepsilon}_{i,-j-1} = \hat{\varepsilon}$ and update history and future subsequences by

$$\begin{aligned} H'[i,-j-1] &= H'[i,-j], \\ F'[i,-j-1] &= F'[i,-j] \cup \{\hat{\varepsilon}_{i,-j-1}\}. \end{aligned} \tag{9}$$

Otherwise, we have

$$\begin{aligned} H'[i-1,-j] &= H'[i,-j] \setminus \{e_i\}, \\ F'[i-1,-j] &= F'[i,-j]. \end{aligned} \tag{10}$$

Finally, the model outputs a reconstructed sequence $\hat{E}^B$ where we use the superscript B to highlight its backward direction. We denote trainable parameters in backward imputation model as θ_B.

Given imputed sequences $\hat{E}^F$ and $\hat{E}^B$ by forward and backward models respectively, we can build a self-supervised learning approach by enforcing consistency between the results of two models. One straightforward way is to consider Maximum Likelihood Estimation (MLE) that uses output of the counterpart as observed 'ground-truth' labels, which equivalently minimizes the Kullback-Leibler (KL) divergence [11] between two model distributions. The objective for forward imputation model can be $\max_{\theta_F} \left(\log P_{\theta_F}\left(\hat{E}^B\right)\right)$, where

$$P_{\theta_F}(\hat{E}^B) = \prod_{\varepsilon=(\mu,\tau)\in\hat{E}^B} P_{\theta_F}(\mu) \cdot P_{\theta_F}(\delta t). \tag{11}$$

Here $P_{\theta_F}(\mu)$ is given by Eq. (4) and $P_{\theta_F}(\delta t)$ (where δt denoting time interval between event ε and the next imputed event) is given by Eq. (5). Similarly, the objective for backward imputation model can be $\max_{\theta_B} \left(\log P_{\theta_B}\left(\hat{E}^F\right)\right)$, where

$$P_{\theta_B}(\hat{E}^F) = \prod_{\varepsilon=(\mu,\tau)\in\hat{E}^F} P_{\theta_B}(\mu) + P_{\theta_B}(\delta t) \tag{12}$$

and $P_{\theta_B}(\mu)$ as well as $P_{\theta_B}(\delta t)$ are given by the backward model.

4.3 Peer Imitation Learning

The MLE-based self-supervised learning enables two models to learn from each other in unsupervised situation. However, one limitation is that MLE loss enforces hard match of two models and assigns each imputed event in the sequence with equal importance. Some prior works focusing on learning point process models for complete event sequences propose to use reinforcement learning [13] and, especially, imitation learning [22,24] for optimization by 1) treating

the prediction as model policy and ground-truth data as expert policy and 2) guiding the model to generate policy close to the expert. The imitation learning approach brings about some tolerance on local mismatch of two sequences and would adaptively allocate different importance to event pairs via a learnable discriminator that is jointly optimized to provide an adequate measurement.

In our case, we have no ground-truth data as expert policy. Like the MLE method, we can use the output of one model as 'ground-truth' data for target of the other model's training. The intuition is from real-world scenarios when students take classes: 1) students can directly learn from what teachers taught in class (ground-truth expert policy), which corresponds to the case in [22,24] with complete sequences in training set, and 2) students can also learn from the peers through discussion and knowledge sharing when teachers are absent. Inspired by this, we propose a novel *peer imitation learning* approach to improve the MLE-based optimization. We let one model's output (as model policy) be close to the counterpart's (as expert policy) through a measurement given by a neural reward function, which is with the same spirit of the discriminator in GAIL [9]. In this way, we free the model optimization from distance-based objectives via joint learning of sequence imputation model and reward function in an adversarial manner.

The objective for forward model can be written as

$$\min_{\theta_F} \max_{w_F} \mathbb{E}_{\hat{E}^B \sim P_{\theta_B}} \left(\log D_{w_F}(\hat{E}^B) \right) + \mathbb{E}_{\hat{E}^F \sim P_{\theta_F}} \left(1 - \log D_{w_F}(\hat{E}^F) \right), \tag{13}$$

where P_{θ_F} and P_{θ_B} are given in Eq. (11) and (12) and we introduce a discriminator $D_{w_F}(\cdot)$ (with parameters w_F) that aims to maximize reward for backward model's imputation and minimize reward for forward model's imputation. As shown in [7], Eq. (13) is equivalent to minimizing Jensen-Shannon divergence between two model distributions P_{θ_F} and P_{θ_B}. Similarly, for backward model, the objective is

$$\min_{\theta_B} \max_{w_B} \mathbb{E}_{\hat{E}^F \sim P_{\theta_F}} \left(\log D_{w_B}(\hat{E}^F) \right) + \mathbb{E}_{\hat{E}^B \sim P_{\theta_B}} \left(1 - \log D_{w_B}(\hat{E}^B) \right), \tag{14}$$

where $D_{w_B}(\cdot)$ is a discriminator with parameters w_B.

Neural Reward Functions. We further present the specification of two discriminators. We take $D_{w_F}(\cdot)$ for illustration and $D_{w_B}(\cdot)$ can be specified in a similar way. Assume $\hat{E}$ as an imputed sequence with $\hat{n}$ imputed events in total. $D_{w_F}(\hat{E})$ can be a sequence-to-sequence model, mapping $\hat{E}$ to a sequence of reward values for each imputed event: $[r_1, r_2, \cdots, r_{\hat{n}}] = d_F(\hat{E})$, where $d_F(\cdot)$ is a Bi-LSTM.

Policy Gradients. To optimize Eq. (13), we REINFORCE algorithm and compute policy gradient for θ_F,

$$\nabla_{\theta_F} \mathbb{E}_{\hat{E}^F \sim P_{\theta_F}} \left(\log D_{w_F}(\hat{E}^F) \right) \approx \frac{1}{C} \sum_{c=1}^{C} \sum_{\varepsilon_i \in \hat{E}_c^F} \gamma^i r_i \cdot \nabla_{\theta_F} \log P_{\theta^F}(\varepsilon_i), \tag{15}$$

where we sample C imputed sequences $\{\hat{E}_c^F\}$ from the forward imputation model P_{θ_F}, $\gamma \in (0, 1]$ is a discount factor, $\varepsilon_i = (\mu_i, \tau_i)$ denotes the i-th imputed event in $\hat{E}_c^F$, and $P_{\theta_F}(\varepsilon_i) = P_{\theta_F}(\mu_i) \cdot P_{\theta_F}(\tau_i)$ (given by Eq. (4) and (5)).

Also, the policy gradient for the discriminator w_F can be computed by

$$\begin{aligned}
&\nabla_{w_F} \mathbb{E}_{\hat{E}^B \sim P_{\theta_B}} \left(\log D_{w_F}(\hat{E}^B) \right) + \mathbb{E}_{\hat{E}^F \sim P_{\theta_F}} \left(1 - \log D_{w_F}(\hat{E}^F) \right) \\
\approx &\frac{1}{C} \sum_{c=1}^{C} \sum_{\varepsilon_i \in \hat{E}_c^B} \gamma_i r_i \cdot \nabla_{w_F} \log P_{\theta_B}(\varepsilon_i) + \frac{1}{C} \sum_{c'=1}^{C} \sum_{\varepsilon_{i'} \in \hat{E}_{c'}^B} \gamma_{i'} r_{i'} \cdot \nabla_{w_F} \left(1 - \log P_{\theta_F}(\varepsilon_{i'}) \right).
\end{aligned} \tag{16}$$

The policy gradients for θ_B and w_B in Eq. (14) can be computed in similar ways. The training process is conducted iteratively. In each iteration of the algorithm, the parameters of forward and backward model are updated alternately, during which both of the modules (generator and discriminator) are adversarially optimized. In specific, we optimize over θ_F and w_F in N_s steps and then we turn to θ_B and w_B using N_s steps of updates.

5 Experiments

5.1 Experimental Setup

Datasets. For synthetic data, we adopt typical point processes for data generation: *Hawkes Process* [8], *Self-Correcting Process* [10], *Inhomogeneous Poisson Process*, and neural point process *RMTPP* [4]. For real-world datasets, we adopt *NYC-Taxi* [23] and *Elevator* [2] as widely used in previous works. We adopt the following hyper-parameter settings: learning rate $lr = 10^{-4}$, embedding dimension $d = 10$, batch size 128, integrating parameter in the event embeddings $\beta = 0.3$, discount factor $\gamma = 0.95$, update steps $N_s = 3$. All these hyper-parameters are optimized by grid search.

Missing-Data Mechanisms. We adopt three common missing-data mechanisms to generate incomplete event sequences.

1. Random Missing (RM). We consider a fixed missing probability p and each event in sequences has the same probability to be masked out. We adopt $p = 0.2$ in our experiments.
2. Time Intervals (TI). Another typical real-life missing-data mechanism could be the unavailability of data between a certain time interval. Correspondingly, given time upper bound T of event sequences, we cut off all events in $[\alpha_1 T, \alpha_2 T]$. We consider $\alpha_1 = 0.4$ and $\alpha_2 = 0.6$ in our experiments.
3. Marker Types (MT). In some cases, one may exactly know the arrival time of events but event markers (i.e., types) are unobserved. In our experiments, we randomly choose one marker class in each dataset and drop all the events with such class of markers to generated incomplete sequences.

Baselines. It is hard to provide a fair comparison between our unsupervised model and other weakly supervised or semi supervised models, such as [14,19–21], as we discussed in Sect. 2. Therefore, for a convincing comparison, we turn to migrate state-of-the-art neural intensity model and consider four different variants namely *Uni-RMTPP(F)*, *Uni-RMTPP(B)*, *Bi-RMTPP(F)*, and *Bi-RMTPP(B)* from Recurrent Marked Temporal Point Process (RMTPP) [4] as competitors for unsupervised sequence imputation. In addition, we also simplify our imitation learning as directly using MLE for optimization, which constructs two variants of our models **PILES-MLE(F)** and **PILES-MLE(B)** for ablation studies. Moreover, to investigate the gap between our imputation results in unsupervised learning to those of supervised learning, we apply ground-truth complete event sequences for training Bi-RMTPP(F), called **Oracle-SUP**.

Table 1. The comparison results evaluated by **Normalized OTD**

Methods	Hawkes			Self-Correcting			Inhomogeneous			Recurrent			Elevator			NYC Taxi		
	RM	TI	MT	RM	TI	MT	RM	TI	MT	RM	TI	MT	RM	TI	MT	RM	TI	MT
Oracle-SUP	10.58	11.59	18.66	3.239	4.214	7.980	0.011	0.085	0.017	0.172	0.193	0.491	0.368	0.623	0.056	2.172	2.107	5.989
PILES(F)	**15.24**	**13.30**	33.01	**4.508**	**6.276**	16.30	0.039	**0.098**	**0.068**	**0.177**	**0.204**	**0.586**	**0.505**	0.908	0.197	**2.229**	3.016	**9.091**
PILES(B)	24.67	19.92	42.29	9.132	13.21	**14.55**	0.047	0.111	0.101	0.526	0.285	0.701	0.648	0.912	0.118	2.917	**3.011**	13.47
PILES-MLE(F)	23.26	21.12	**32.60**	12.31	10.33	16.15	**0.036**	0.102	0.074	0.487	0.324	1.116	0.582	0.816	0.322	3.453	4.126	10.41
PILES-MLE(B)	28.43	26.57	49.37	15.43	17.29	25.37	0.055	0.124	0.148	0.619	0.721	1.134	0.665	0.694	0.136	3.314	3.164	16.32
Bi-RMTPP(F)	34.37*	34.52*	50.09	19.23	20.63	37.34	0.105*	0.101*	0.111	1.147*	1.141*	3.864 *	0.863	**0.797***	0.110	11.32*	10.37	35.65*
Bi-RMTPP(B)	39.76	38.55	50.23	20.58	22.57	39.86	0.147	0.176	0.196	1.837	1.546	4.932	0.419	0.817	0.086	11.41	10.16*	38.36
Uni-RMTPP(F)	36.02	35.20	48.08*	18.31	18.67	33.04	0.108	0.110	0.106*	1.911	1.443	5.532	0.765	0.816	0.133	12.64	15.23	58.38
Uni-RMTPP(B)	36.52	40.11	50.28	17.33*	14.07*	30.00*	0.187	0.193	0.178	2.204	1.703	5.921	0.674*	0.811	**0.091***	13.21	16.32	60.43
Improvement	55.6%	61.5%	31.3%	74.0%	55.4%	51.5%	62.9%	3.0%	35.8%	84.6%	82.1%	84.8%	25.1%	0.0%	0.0%	80.3%	70.4%	74.5%

5.2 Evaluation Metrics

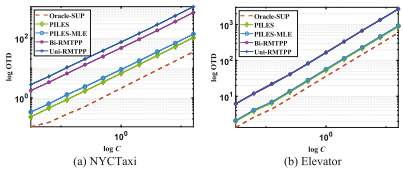
Evaluating event sequence imputation involves the difficulty for a well-defined distance metric between two asynchronous event sequences. It is intractable to align events in two sequences with asynchronous timestamps and different numbers of events. Recently, [14] proposes to use Optimal Transport Distance (OTD) for describing the event sequence similarity, which is computed by dynamic programming with pre-defined insertion and deletion costs. The fundamental intuition for OTD is to describe the distance between sequences by how much it cost to "modify" one sequence to become identical to another (similar to minimum edit distance in NLP [26]), where smaller OTD indicates better imputation performance. In specific, given a complete event sequence $E = \{e_1, e_2, \cdots, e_P\}$, and an imputed event sequence $\hat{E} = \{\hat{e}_1, \hat{e}_2, \cdots, \hat{e}_Q\}$, where $e_i = (m_i, t_i)$ and $\hat{e}_j = (\hat{\mu}_j, \hat{\tau}_j)$, we define an alignment set $\mathcal{S}$ for pairing events from two sequences: $\mathcal{S} = \{(i, j) \mid m_i = \hat{\mu}_j,\ i \in [1, P],\ j \in [1, Q]\}$, where we can discover that in our settings, only events with the same types of markers are allowed to be aligned.

We can also discover that given two event sequences, the alignment set $\mathcal{S}$ is not unique, since different events (say e_1, e_2 here) in one sequence could have the same type of markers so that given an event $\hat{e}$ in another sequence with identical marker, we can actually align $\hat{e}$ with either e_1 or e_2. Given two event sequences

E and $\hat{E}$, the distance D is defined as $OTD(E, \hat{E}) = \min_{\mathcal{S}} \sum_{(i,j)\in\mathcal{S}} |t_i - \hat{\tau}_j| + \sum_{i\notin\mathcal{S}} C_{insert} + \sum_{j\notin\mathcal{S}} C_{delete}$ Besides, we propose a normalized optimal transport distance (normalized OTD) as follows: $nOTD(E, \hat{E}) = \min_{\mathcal{S}} \sum_{(i,j)\in\mathcal{S}} \frac{|t_i - \hat{\tau}_j|}{T} + \sum_{j\notin\mathcal{S}} \frac{|T - \hat{\tau}_j|}{T} + \sum_{i\notin\mathcal{S}} \frac{|t_i|}{T}$.

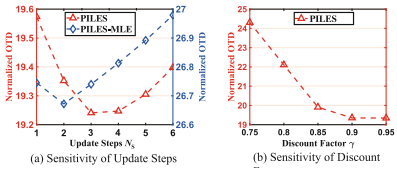
5.3 Results and Analysis

Comparison on OTD. Figure 2(a) and Fig. 2(b) show OTD results on NYC Taxi and Elevator datasets respectively. Here we set the insertion and deletion cost C from 2^{-5} to 2^4 and compare the performances of our model and several baselines. On both datasets, the OTD values increase approximately linearly as C grows in logarithmic space, and notably, PILES constantly outperforms others with a minimal gap to supervised method Oracle-SUP. Among comparative methods, bi-directional model generally performs better than the uni-directional counterpart due to better utility of event data in sequences. There are significant improvements achieved by PILES over two RMTPP models: 76.2% on NYC Taxi dataset and 42.4% on Elevator dataset.



(a) NYCTaxi (b) Elevator

Fig. 2. Optimal Transport Distance (OTD) with delete and insert cost $C \in [-5, 2^4]$ in logarithmic axes. Each data point is average of forward and backward reconstructions.



(a) Sensitivity of Update Steps (b) Sensitivity of Discount

Fig. 3. Sensitivity of update steps N_s and discount factor γ evaluated on NYC-Taxi in an RM setting. We fix $\gamma = 0.95$, and set $N_s = 3$ to evaluate the sensitivity of γ, where the learning rate is fixed at 10^{-4}.

Comparison on Normalized OTD. Table 2 presents results of normalized OTD on four synthetic datasets and two real-world datasets. As we can see, PILES generally provides the best performances in the majority of cases with different missing-data mechanisms on six datasets. The results demonstrate the superiority of proposed model. Notably, compared with RMTPP models, PILES manages to achieve 80.3% improvement on NYC-Taxi dataset with the setting of random missing (RM), which shows that the peer imitation learning approach contributes to better self-supervised learning by enforcing consistency between forward and backward imputation rather than training the model directly over imcomplete sequences. Comparing different missing-data mechanisms, we can see

that missing marker type (MT) appears to be much more difficult than other two missing-data mechanisms and there exist obvious gaps between unsupervised methods and Oracle-SUP in MP settings. Moreover, in terms of different data generating processes, we can see that when using neural intensity model to generate data, our model PILES gives very close results to Oracle-SUP with much smaller gaps compared to other data generating processes. This phenomenon indicates that our method has capability to capture complex latent process in input sequences and also, the inductive bias shared by both the model and data generation affects the performance a lot.

Ablation Study. One more straight-forward approach for mutual learning is to directly use outputs from another model as supervised training data, based on which one can use MLE for optimization. Here we compare our PILES with peer imitation learning with the model using MLE as ablation study. As shown in Fig. 2 where we report results for PILES and PILES-MLE on six datasets, we can find that although slightly, PILES outperforms PILES-MLE on both datasets. Also in Table 1, the results show that PILES turn out to be superior than PILES-MLE in the majority of cases, which indicates that our proposed peer imitation learning is effective for event sequence imputation.

Parameter Sensitivity. We also analyze the sensitivity of our model on two hyper-parameters, the number of update steps N_s and discount factor γ, in Fig. 3. As shown in Fig. 3(a), PILES performs the best when N_s equals to 3. In fact, N_s controls how sufficiently two models' results are matched in each step of iterative training. When N_s is too small, two models are insufficiently learned from each other; when N_s is too large, they would 'over-fit' the estimation from the counterpart and get stuck by some local optima. In Fig. 3(b), we plot normalized OTD of PILES v.s. different discount factor γ's. We can see that the model performance declines faster when γ decreases. One possible reason is that when using relatively small γ, the rewards from previously imputed events would be weakened and the model focuses more on a few imputed events in a sequence, ignoring global information.

6 Conclusion

In this paper, we focus on reconstructing partially observed event sequences in a fully unsupervised setting. We propose a novel peer imitation learning point process framework which entails two-fold contributions on methodological level. First, we extend the definition of conventional temporal point process and propose inverse point process model. Second, we design a new peer imitation learning approach that lets two point process models in different directions cooperatively learn from each other to achieve a consensus on imputation results. Comprehensive experimental results on six datasets with multifarious missing-data mechanisms show the superiority of our method.

Acknowledgement. This work was supported by the National Key R&D Program of China [2020YFB1707900]; the National Natural Science Foundation of China [62272302, 62172276], and Shanghai Municipal Science and Technology Major Project [2021SHZDZX0102].

References

1. Antoniou, A.: Digital Signal Processing. McGraw-Hill, New York (2016)
2. Crites, R.H., Barto, A.G.: Improving elevator performance using reinforcement learning. In: NeurIPS, pp. 1017–1023 (1995)
3. Daley, D.J., Vere-Jones, D.: An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure. Springer, Heidelberg (2007). https://doi.org/10.1007/978-0-387-49835-5
4. Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., Song, L.: Recurrent marked temporal point processes: embedding event history to vector. In: SIGKDD, pp. 1555–1564 (2016)
5. Enguehard, J., Busbridge, D., Bozson, A., Woodcock, C., Hammerla, N.Y.: Neural temporal point processes for modelling electronic health records (2020)
6. Fan, Y., Xu, J., Shelton, C.R.: Importance sampling for continuous time bayesian networks. J. Mach. Learn. Res. **11**(Aug), 2115–2140 (2010)
7. Goodfellow, I.J., et al.: Generative adversarial nets. In: NeurIPS, pp. 2672–2680 (2014)
8. Hawkes, A.G.: Spectra of some self-exciting and mutually exciting point processes. Biometrika **58**(1), 83–90 (1971)
9. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: NeurIPS, pp. 4565–4573 (2016)
10. Isham, V., Westcott, M.: A self-correcting point process. Stochastic Process. Appl. **8**(3), 335–347 (1979)
11. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
12. Lee, Y., Vo, T.V., Lim, K.W., Soh, H.: Z-transforms and its inference on partially observable point processes. In: IJCAI, pp. 2369–2375 (2018)
13. Li, S., Xiao, S., Zhu, S., Du, N., Xie, Y., Song, L.: Learning temporal point processes via reinforcement learning. In: NeurIPS, pp. 10781–10791 (2018)
14. Mei, H., Qin, G., Eisner, J.: Imputing missing events in continuous-time event streams. In: ICML, pp. 4475–4485 (2019)
15. Nodelman, U., Shelton, C.R., Koller, D.: Continuous time bayesian networks. arXiv preprint arXiv:1301.0591 (2012)
16. Pan, Z., Huang, Z., Lian, D., Chen, E.: A variational point process model for social event sequences. In: AAAI, pp. 173–180 (2020)
17. Rao, V., Teh, Y.W.: MCMC for continuous-time discrete-state systems. In: NeurIPS, pp. 701–709 (2012)
18. Reinhart, A.: A review of self-exciting spatio-temporal point processes and their applications. Stat. Sci. **33**(3), 299–318 (2018)
19. Schaubel, D.E., Cai, J.: Multiple imputation methods for recurrent event data with missing event category. Can. J. Stat. **34**(4), 677–692 (2006)
20. Shelton, C.R., Qin, Z., Shetty, C.: Hawkes process inference with missing data. In: AAAI, pp. 6425–6432 (2015)

21. Stomakhin, A., Short, M.B., Bertozzi, A.L.: Reconstruction of missing data in social networks based on temporal patterns of interactions. Inverse Prob. **27**(11), 115013 (2011)
22. Upadhyay, U., De, A., Rodriguez, M.G.: Deep reinforcement learning of marked temporal point processes. In: NeurIPS, pp. 3172–3182 (2018)
23. Whong, C.: Foiling nyc's taxi trip data (2014)
24. Wu, Q., Zhang, Z., Gao, X., Yan, J., Chen, G.: Learning latent process from high-dimensional event sequences via efficient sampling. In: NeurIPS, pp. 3842–3851 (2019)
25. Xiao, S., Yan, J., Yang, X., Zha, H., Chu, S.M.: Modeling the intensity function of point process via recurrent neural networks. In: AAAI, pp. 1597–1603 (2017)
26. Zhao, Y., Jiang, H., Wang, X.: Minimum edit distance-based text matching algorithm. In: NLPKE, pp. 1–4 (2010)
27. Zhou, K., Zha, H., Song, L.: Learning triggering kernels for multi-dimensional hawkes processes. In: ICML, pp. 1301–1309 (2013)