

# Time-Discounting Convolution for Event Sequences with Ambiguous Timestamps

Takayuki Katsuki, Takayuki Osogami,  
Akira Koseki, Masaki Ono, Michiharu Kudo  
IBM Research - Tokyo  
Tokyo, Japan  
e-mail: {kats,osogami,akoseki,moono,kudo}@jp.ibm.com

Masaki Makino, Atsushi Suzuki  
Department of Endocrinology and Metabolism,  
Fujita Health University  
Aichi, Japan  
e-mail: {makinom,aslapin}@fujita-hu.ac.jp

**Abstract**—This paper proposes a method for modeling event sequences with ambiguous timestamps, a time-discounting convolution. Unlike in ordinary time series, time intervals are not constant, small time-shifts have no significant effect, and inputting timestamps or time durations into a model is not effective. The criteria that we require for the modeling are providing robustness against time-shifts or timestamps uncertainty as well as maintaining the essential capabilities of time-series models, i.e., forgetting meaningless past information and handling infinite sequences. The proposed method handles them with a convolutional mechanism across time with specific parameterizations, which efficiently represents the event dependencies in a time-shift invariant manner while discounting the effect of past events, and a dynamic pooling mechanism, which provides robustness against the uncertainty in timestamps and enhances the time-discounting capability by dynamically changing the pooling window size. In our learning algorithm, the decaying and dynamic pooling mechanisms play critical roles in handling infinite and variable length sequences. Numerical experiments on real-world event sequences with ambiguous timestamps and ordinary time series demonstrated the advantages of our method.

**Index Terms**—Electronic health records, Electronic healthcare, Health information management, Event sequence, Convolutional neural networks, Time series analysis

## I. INTRODUCTION

Temporal event sequences record timestamped events, which are ubiquitous in the real world and are addressed in various data mining problems. We address the scenario where the attached timestamps are ambiguous, as shown in Fig. 1. Such sequences are found in events that are recorded passively, i.e., without observers' instantaneous control. The timestamps do not represent when events have occurred but represent when they were recorded. In such cases, the time intervals are variable and the timestamps are not reliable. Thus, small time shifts in the observed sequences have no significant effect, and there are uncertainties in recorded timestamps. In medical informatics, for example, attention is being paid to analyzing such passively recorded data, as found in electronic health records (EHRs) for predicting risks to patients [1]–[3]. They are recorded when a patient is treated at a hospital.

Modeling event sequences is one of the fundamental problems in data mining, such as analyses on sensor time-series,

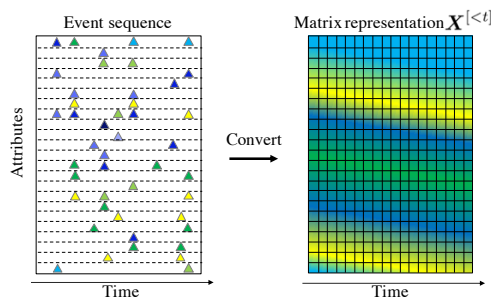


Fig. 1: Example of event sequence with ambiguous timestamps and its matrix representation. Horizontal and vertical axes respectively correspond to timestamps and attributes. Triangles indicate attribute values of events at corresponding times; yellow and blue mean high and low values, respectively.

economic data, and EHRs. While we can assume independent observations for non-sequential data, we must take into account the dependencies between successive events. In standard modeling approaches, one of two major assumptions is made: the observations occur at regular intervals (so the timestamps can be ignored) or at irregular intervals (so the timestamps must be considered). The first approach is typically used for time-series and language modeling. Once the order of the sequence is incorporated into the model, exact timestamps and durations are unimportant. There are many established models, including vector autoregressive (VAR) models [4], hidden Markov models [5], recurrent neural networks (RNN) [6], long short-term memory (LSTM) models [7], and Boltzmann machines for time series [8]–[16]. If the time intervals are not constant, the performance is degraded in return for simpler modeling for temporal dependencies.

The second approach is typically used for asynchronously observed event sequences, such as in log records and process-series data. Along with other features representing events, the timestamps or intervals between events are explicitly input to the model for encoding the dependencies between successive events. RNN models have been used for this purpose [17]–[19]. However, if the timestamps are not reliable, directly inputting them into the model might not be effective.

For modeling the event sequences with ambiguous timestamps, we addressed three major modeling requirements.

### Time-shift invariance

The model should be invariant against time shifts, since almost the same patterns would be recorded with a time shift if there is no instantaneous and on-demand control.

### Robustness against timestamp uncertainty

The model should be robust against uncertainty in timestamps, since the timestamps represent when they were recorded but not when events have occurred.

### Natural forgetting

The ability to forget meaningless past information should be inherited from time-series models, enabling the handling of infinite sequences and long-term dependency.

We propose a *time-discounting convolution* method that uses a specific convolutional structure and a *dynamic pooling* mechanism. Our convolutional structure has a uni-directional convolution mechanism across time with two kinds of parameter sharing for efficiently representing the dependency between events in a time-shift invariant manner. It also has a mechanism of naturally forgetting by discounting the effects of past observations. The structure is based on the eligibility trace in dynamic Boltzmann machines (DyBMs) [12]–[16], whose learning rules have a biologically important characteristic, i.e., spike-timing-dependent synaptic plasticity (STDP). This is our first contribution. The dynamic pooling mechanism provides robustness against the uncertainty in timestamps and enhances the time-discounting capability by dynamically changing the window size in its pooling operations. This is our second contribution.

Several time-convolutional models have been proposed to capture shift-invariant patterns while representing temporal dependencies. The time-delay neural network [20] is a pioneering model. Convolutional neural networks (CNNs) have recently been applied to sequential data, such as stock price movements [21], sensor outputs [22], radio communication signals [23], videos [24], and EHRs [1], [3], [25]. However, these models do not have a temporal nature, i.e., the natural forgetting capability, which is one of our modeling requirements. To represent the both convolutional and temporal natures, stacking combinations of convolutional models and time-series models have also been studied recently [26], [27]. Our model inherently has both a time-series and convolutional aspects. This reduces the number of model parameters, which is quite useful when the amount of training data is limited.

We empirically evaluated the effectiveness of the proposed method in numerical experiments to examine its utility of the method for the real event sequences with ambiguous timestamps and general workability of the method for the real time-series data. We found that the proposed method improves predictive accuracy.

## II. PREDICTION FROM TEMPORAL EVENT SEQUENCES

Our goal is to construct a model for predicting objective variables,  $\mathbf{y}^{[t]}$  at time  $t$ , from an event sequence before time  $t$ ,  $\mathbf{X}^{[<t]}$ , where vector  $\mathbf{y}^{[t]}$  can be a future event itself (autoregression) or any other unobservable variable (regres-

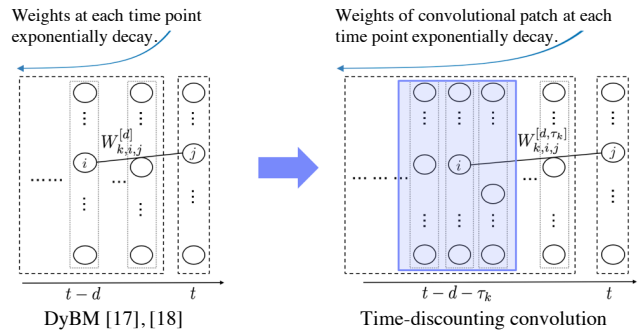


Fig. 2: Proposed time-discounting convolution method.

sion or classification), and  $\mathbf{X}^{[<t]}$  represents all the observed sequences before time  $t$ .

An event sequence is a set of records of events with timestamps. A record has the values of  $D$  attributes, and each of the attributes may or may not be observed with other attributes at the same time, as shown in Fig. 1. For ease of analysis, the sequence  $\mathbf{X}^{[<t]}$  is usually represented as a  $D \times T$  matrix [3], [25], [28]. The horizontal dimension represents the timestamp at regular intervals with the highest temporal resolution in the sequence, and the vertical dimension represents the attribute values, as shown in Fig. 1, i.e.,  $\mathbf{X}^{[<t]} \equiv \{\mathbf{x}^{[t-d]}\}_{d=t-T}^{t-1}$ , where  $\mathbf{x}^{[t]}$  is the vector of attribute values at  $t$ . If the event records are originally observed at regular intervals and all the attributes are always observed, the temporal event sequence is reduced to ordinary time-series data. If the original observation intervals vary over time and all the attributes are not observed simultaneously, several elements of the matrix will be missing. We replace missing values with large negative values (sufficiently lower than the minimum value of each attribute), which works well with our dynamic pooling described in the following section.

We learn the parameters  $\theta$  of our prediction model,  $\mathbf{f}(\bullet, \theta)$ , by minimizing the objective function:

$$\mathcal{L}(\theta) \equiv \sum_{t=1}^N \mathcal{L}^{[t]}(\theta), \quad \text{where} \quad (1)$$

$$\mathcal{L}^{[t]}(\theta) \equiv L(\mathbf{y}^{[t]}, \mathbf{f}(\mathbf{X}^{[<t]}, \theta)),$$

where  $N$  is the number of training samples ( $N > T$ ) and  $L(\bullet)$  is a loss function, which is selected for each task and the corresponding objective variables  $\mathbf{y}^{[t]}$ , from mean squared error, cross entropy, log likelihood, and other functions.

By using the learned parameters,  $\hat{\theta}$ , we can predict  $\mathbf{y}$ ;

$$\hat{\mathbf{y}}^{[t]} \equiv \mathbf{f}(\mathbf{X}^{[<t]}, \hat{\theta}). \quad (2)$$

We define our prediction model in the following section.

## III. TIME-DISCOUNTING CONVOLUTION

### A. Prediction model with time-discounting convolution

We propose a time-discounting convolution method for the prediction model. It has a convolutional structure across time,

where the weight of a convolutional patch decays exponentially at each time point, as shown in Fig. 2. The proposed model has parameters  $\theta \equiv [\mathbf{W}, \mathbf{b}]$  and predicts the  $j$ -th element of  $\mathbf{y}^{[t]}$  by the use of a non-linear function  $h(\bullet)$  that maps a  $K \times T$ -dimensional input to a 1-dimensional output:

$$\begin{aligned} & [f(\mathbf{X}^{[<t]}, \theta)]_j \\ &= h\left(\left\{\sum_i \sum_{\tau_k=0}^{T_k} x_i^{[t-d-\tau_k]} W_{k,i,j}^{[d,\tau_k]} - b_k\right\}_{1 \leq k \leq K, 1 \leq d \leq T}\right), \end{aligned} \quad (3)$$

where  $W_{k,i,j}^{[d,\tau_k]}$  is a convolutional parameter across time with  $\tau_k$  for the  $i$ -th attribute value  $x_i^{[t-d]}$  at time  $t-d$ , and  $T_k$  is the time length of the  $k$ -th convolutional patch. We use bias parameter  $b_k$  individually for each of the  $K$  feature maps. The non-linearity in  $h(\bullet)$  makes this apparently redundant formulation meaningful, analogous to CNNs. We define specific functional forms of  $h(\bullet)$  for each task in Section IV along with the details of the implementation.

We use two different parametric forms for  $W_{k,i,j}^{[d,\tau_k]}$ :

$$W_{k,i,j}^{[d,\tau_k]} = \lambda^{d+\tau_k} U_{k,i,j}, \quad \text{and} \quad (4)$$

$$W_{k,i,j}^{[d,\tau_k]} = \mu^d V_{k,\tau_k,i,j}, \quad (5)$$

where  $\lambda, \mu \in [0, 1]$  is the decay rate. Note that  $W_{k,i,j}^{[d,\tau_k]}$  forms a tensor and corresponds to a patch in CNNs. Eq. (4) uses  $U_{k,i,j}$  consisting of a single parameter across time for each  $k, i$ , and  $j$ . In the convolutional patch based on Eq. (4),  $U_{k,i,j}$  is replicated and used for multiple temporal positions of the time convolution with the decay rate of  $\lambda^{d+\tau_k}$ . The parameterization with shared parameters in Eq. (4) works similarly to the eligibility trace of DyBM, as shown on the left side of Fig. 2; that is, the convolutional patch extracts the feature that represents the frequency of each observation and its distance from the current prediction. Eq. (5) uses  $V_{k,\tau_k,i,j}$  consisting of individual parameters for each time  $\tau_k$  and other indexes  $k, i$ , and  $j$ . The  $V_{k,\tau_k,i,j}$  forms the convolutional patch by itself and all the parameters in this patch decay together by  $\mu^d$  in accordance with the timestamp  $d$ . We use Eqs. (4) and (5) in the same proportion in our  $K$  feature maps. The proposed method can capture discriminative temporal information in a shift-invariant manner because of its convolutional operation. Also, it can naturally forget past information, and prediction and gradients in learning do not diverge in the limit of  $T \rightarrow \infty$  thanks to the decay rates  $\lambda$  and  $\mu$  in Eqs. (4) and (5).

We can use our model as a layer in a neural network, can incorporate a neural network into our model via  $h(\bullet)$ , or as a preprocessor of the input  $\mathbf{X}^{[<t]}$  in Eq. (3). In our experiments, we actually used our model along with a fully-connected layer and activation functions for prediction.

### B. Dynamic pooling

We introduce dynamic pooling as a powerful mechanism to avoid overfitting to vague timestamps and past meaningless information. Our pooling window increases in accordance

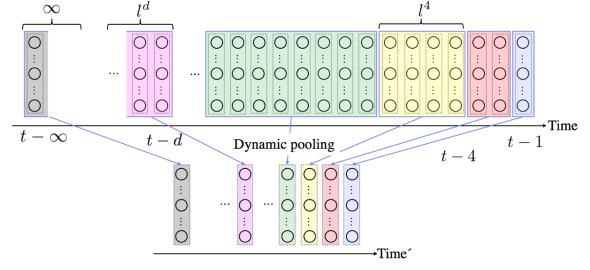


Fig. 3: Dynamic pooling

with the time from prediction point  $t$  as shown in Fig. 3. Specifically, we let

$$z_i^{[t]} = \max_{\tau_k \in [0, l_0 \times l^t]} (x_i^{[t-\tau_k]}), \quad (6)$$

where  $l_0$  is the initial window size and  $l$  is the growth rate of the window. We can dynamically downsample the observed event sequences or latent representations by taking the maximum value over sub-temporal regions along with increasing the window size exponentially.

Dynamic pooling is used in the proposed method both as a preprocessor of input  $\mathbf{X}^{[<t]}$  and function  $h(\bullet)$  in Eq. (3). As a preprocessor of  $\mathbf{X}^{[<t]}$ , we first apply dynamic pooling to a raw sequence then use the preprocessed sequence as  $\mathbf{X}^{[<t]}$ . As  $h(\bullet)$ , we apply dynamic pooling to the latent representations that are the inputs to  $h(\bullet)$ .

Dynamic pooling leads to tractable analysis of the missing values by ignoring them in its max operation as the first pooling layer. It also enables us to easily handle infinite sequences when we make the final window size infinite. Also, we can handle the varying (horizontal) dimensions of  $\mathbf{X}^{[<t]}$  across different  $t$  in the same manner as handling infinite sequences. The pooling layer after convolution works as an ordinary pooling method, i.e., the patterns having the largest effect are extracted. Because the rate of selecting each time point in the max operation decreases due to the growth of the window width in accordance with the time length from the prediction point  $t$ , the expected effect of each time point decays exponentially. This is also similar to the eligibility trace in DyBM. We can define other pooling mechanisms, such as *dynamic mean-pooling*, by replacing the  $\max(\bullet)$  operation with another operation.

### C. Learning Model Parameters

In our experiments, we tackled autoregression and classification problems. We define the objective function  $\mathcal{L}^{[t]}(\theta)$  for each problem setting and derive the learning rules. For the autoregression problems, we use the L2-norm in Eq. (1):

$$\mathcal{L}^{[t]}(\theta) \equiv \|\mathbf{X}^{[t]} - f(\mathbf{X}^{[<t]}, \theta)\|^2, \quad (7)$$

For the classification problems, we use the following form in Eq. (1):

$$\mathcal{L}^{[t]}(\theta) \equiv L_c(\mathbf{y}^{[t]}, \sigma(f(\mathbf{X}^{[<t]}, \theta))), \quad (8)$$

where the function  $L_c(\bullet)$  is the cross entropy, and function  $\sigma(\bullet)$  is the softmax function. For the classification problems, we use  $\sigma(f(\mathbf{X}^{[<t]}, \theta))$  as the prediction model.

The model parameters are learned by mini-batch gradient descent. The update rule with  $t$  is defined as

$$\theta \leftarrow \theta - \eta \frac{1}{M} \sum_{m=t-M+1}^t \frac{\partial \mathcal{L}^{[m]}(\theta)}{\partial \theta}, \quad (9)$$

where  $\eta$  is the learning rate and  $M$  is the mini-batch size. The specific gradients of the parameters are omitted due to space limitations (see the Appendix). In the training phase, the dynamic pooling is simply passed through the gradient to the unit selected as maximum, analogous to ordinary max-pooling. In the mini-batch gradient descent, the learning rate  $\eta$  is controlled using the Adam optimizer with the hyperparameters recommended in [29], and the mini-batches are set as  $M = 16$  examples. We used the same procedure for all the models we compared in our experiments. The detailed settings of the hyperparameters  $K$ ,  $T_k$ ,  $\lambda$ ,  $\mu$ ,  $l_0$ , and  $l$  are described for each experimental task in Section IV.

#### D. Relationships to Other Models

Our model can be seen as a generalization of DyBM. The corresponding prediction model by the DyBM is defined as

$$[f_{\text{DyBM}}(\mathbf{X}^{[<t]}, \theta)]_j = h \left( \left\{ \sum_i x_i^{[t-d]} W_{k,i,j}^{[d]} - \sum_i b_{k,i} \right\}_{1 \leq k \leq K, 1 \leq d \leq T} \right). \quad (10)$$

This is a special case of our model (Eq. (3)) when  $T_k = 0$  for any  $k$ . In Eq. (10), the conduction delay of DyBM is assumed to be zero. In other words, we extend the summation and definition of the eligibility trace in DyBM to the 1D-convolutional operation in our model. We also extended DyBMs to be applicable to classification problems and neural-network layers.

Our model reduces to a VAR model with lag  $T$  if we use only Eq. (5), let  $h(\bullet)$  be the summation over  $d$ , and set  $T_k = T$ ,  $\mu = 1$ , and  $k = 1$ . Eq. (3) then reduces to

$$[f_{\text{VAR}}(\mathbf{X}^{[<t]}, \theta)]_j = \sum_i \sum_{\tau=1}^T x_i^{[t-\tau]} V_{\tau,i,j} - b, \quad (11)$$

where we omit  $k$  and  $d$  because of the above assumptions.

From the above relationships, our model can be seen as an ensemble of convolutional terms (generalization of eligibility trace of DyBM) and VAR-like terms. The convolution with  $T_k \neq \infty$  or  $T$  particularly differentiates our model from them.

DyBM can be considered a temporal expansion of the restricted Boltzmann machine (RBM). By replacing the temporal sequences with hidden variables, the RBM's prediction model for the  $j$ -th hidden variable is

$$[f_{\text{RBM}}(\mathbf{x}, \theta)]_j = h \left( \left\{ \sum_i x_i W_{k,i,j} - \sum_i b_{k,i,j} \right\}_{k=1}^K \right). \quad (12)$$

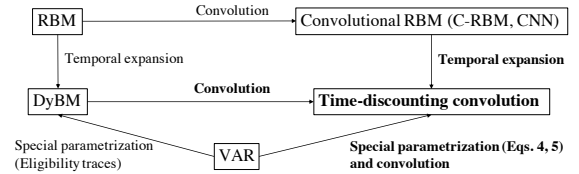


Fig. 4: Relationships among RBM, DyBM, C-RBM, VAR, and our model (time-discounting convolution).

Its convolutional extension is a convolutional restricted Boltzmann machine (C-RBM). For two-dimensional data, the prediction model for the  $(i, j)$ -th hidden variable is

$$[f_{\text{CRBM}}(\mathbf{x}, \theta)]_{i,j} = h \left( \left\{ \sum_{r,s} W_{k,r,s} x_{i+r,j+s} - b_k \right\}_{k=1}^K \right), \quad (13)$$

From Eq. (13), our model can be seen as a temporal expansion of a C-RBM with 1D-convolution and special parameterizations in Eqs. (4) and (5) having exponential decay inspired by DyBM. We summarize these relationships in Fig. 4.

## IV. EXPERIMENTAL RESULTS

We assessed the effectiveness of our method in numerical experiments. First, we applied our method to a real-world event sequence with ambiguous timestamps extracted from an EHR. Since our method was designed for general event sequences including ordinary time-series data, we then evaluated its effectiveness for real-world time-series data.

#### A. Prediction from Real-world Event Sequence with Ambiguous Timestamps

We evaluated the proposed method using two real-world event sequence datasets, EHRs for patients at a Japanese hospital [30]. The first dataset included data on 30,117 patients treated for diabetic nephropathy (DN). We constructed a model for predicting progression of DN from stage 1 to stage 2 after 180 days from the latest record in the input EHR (binary classification task). The progression label of the  $i$ -th input EHR was defined as  $y_i \in \{0, 1\}$  such that  $y_i = 0$  means that the patient remained in stage 1 and  $y_i = 1$  means that the patient had progressed to stage 2. The  $i$ -th input EHR was represented as a 180-day sequence of real-valued results of the lab tests, where we represented the sequence as a matrix  $\mathbf{X}_i \in \mathbb{R}^{D \times T}$  for which the horizontal dimension corresponds to the timestamp (time length  $T = 180$ ) and the vertical dimension corresponds to the lab tests having  $D = 25$  attributes, i.e., Albumin, Albuminuria, ALT(GPT), Amylase, AST(GOT), Blood Glucose, Blood Platelet Count, BMI, BUN, CPK, CRP, eGFR, HbA1c, Ht, Hgb, K, Na, RBC, Total Bilirubin, Total Cholesterol, Total Protein, Troponin, Uric acid, WBC count, and  $\gamma$ -GTP. The second dataset included data on 36,502 patients treated for cardiovascular disease (CVD). We constructed a model for predicting the occurrence of major cardiovascular events after 180 days from the latest record in the input EHR (binary classification task). The label of the  $i$ -th input EHR was defined as  $y_i \in \{0, 1\}$  such that  $y_i = 0$

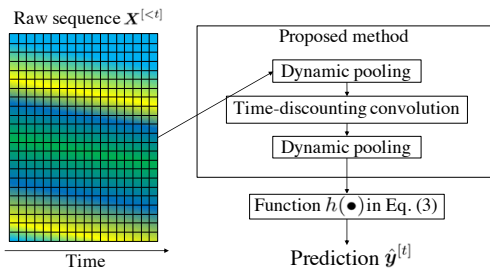


Fig. 5: Overall structure of proposed method for experiments.

TABLE I: Comparison of proposed and baseline methods in terms of AUC (higher is better).

Task	DN	CVD
DyBM	0.607	0.620
CNN	0.647	0.635
CNN w/ dynamic pooling	0.664	0.636
Proposed w/o dynamic pooling	0.660	0.635
Proposed w/ dynamic pooling	<b>0.674</b>	<b>0.647</b>

means that the patient did not experience any of the events and  $y_i = 1$  means that the patient had experienced the event. The definition of  $X_i$  was the same as for the DN case. For both tasks, following [3], the first 67% of each dataset was used for training, and the remaining 33% was used for testing. We standardized the attribute values by subtracting its mean and dividing by its standard deviation in the training data.

1) *Implementation*: Since we were solving the classification tasks, we used Eq. (8) for the objective function. We show the overall structure of the proposed method for the experiments in Fig. 5. We first applied dynamic pooling to the raw matrix  $X$ . Then, the outputs of the first dynamic pooling were inputted to time-discounting convolution. After that, we applied dynamic pooling again. Finally, we used a fully connected neural network as  $h(\bullet)$  in Eq. (3),  $\sum_{k,d} W_{k,d}^{[f]} \delta(g_{k,d}) + b_j^{[f]}$ , where  $W^{[f]}$  and  $b^{[f]}$  are parameters,  $g$  is the inputs to  $h(\bullet)$  in Eq. (3), and the function  $\delta(\bullet)$  is a rectified linear unit (ReLU) [31]. We also used L1-regularization for the hidden units, which are outputs of the second dynamic pooling, in the optimization of Eq. (1). We tuned the regularization parameter  $c$  of L1-regularization and hyperparameters of the proposed method using the last 20% of the training data as validation data. We then trained the model using all of the training data and the tuned parameters. The candidates for  $c$  were  $\{10^{-2}, 10^{-1}, 10^0\}$ . The hyperparameters candidates were  $K \in \{4, 8, 16, 24\}$ ,  $\lambda, \mu \in \{0.8, 0.85, 0.9, 0.95\}$ ,  $l_0 \in \{1, 2, 3, 4, 5, 10\}$ , and  $l \in \{1.0, 1.05, 1.1, 1.2\}$ . We used four different  $T_k$ : 1, 2, 4, and the sequence length. We used them in the same proportion in our  $K$  feature maps.

2) *Results*: We used the area under the curve (AUC) as the evaluation metric since the tasks were binary classification. We compared the results against those of two baseline methods, DyBM, a state-of-the-art model for the time-series data, and CNN, a state-of-the-art model for EHR analysis. For fair comparison, we tuned their hyperparameters in the

TABLE II: Comparison of proposed and baseline methods in terms of RMSE (lower is better).

Task	Sunspot		Price	
	Average	Best	Average	Best
VAR	0.213	0.213	0.329	0.165
DyBM	0.0734	0.0717	<b>0.0466</b>	0.0302
CNN	0.0751	0.0697	0.0845	0.0306
Proposed	<b>0.0719</b>	<b>0.0690</b>	0.0489	<b>0.0291</b>

same manner as with the proposed method for each task. For prediction with DyBM, we used the prediction model of DyBM defined in Eq. (10) (DyBM). For the prediction with CNN, we used the prediction model by replacing the time-discounting convolution in Fig. 5 with ordinary convolutional layer in Eq. (13) (CNN) and that with the two dynamic pooling in Fig. 5 (CNN w/ dynamic pooling). We also present the results of the proposed method without dynamic pooling. As shown in Table I, the AUC for the proposed method was better than those for the baselines. This shows that the convolutional structure and our temporal parameterization work well for event sequences with ambiguous timestamps. Moreover, the values for the DN dataset were higher than the 0.64 reported for a stacked convolutional autoencoder model using the same dataset [3]. The selected hyperparameters for the proposed method were  $c = 0.01$ ,  $K = 8$ ,  $\lambda = 0.95$ ,  $\mu = 0.95$ ,  $l_0 = 4$ , and  $l = 1.05$  for DN, and  $c = 1.0$ ,  $K = 8$ ,  $\lambda = 0.80$ ,  $\mu = 0.80$ ,  $l_0 = 1.0$ , and  $l = 1.05$  for CVD.

## B. Prediction from Real-world Time-series Data

We also evaluated the proposed method using two real-world time-series datasets. The first dataset was a publicly available dataset containing the monthly sunspot number (Sunspot). We constructed a model for predicting the sunspot number for the next month from the obtained sunspot time-series data (autoregression task). The time-series data  $X_i \in [0, 1]^{D \times T}$  had  $D = 1$  dimension and  $T = 2,820$  time steps (corresponding to January 1749 to December 1983). The second dataset was a publicly available dataset containing the weekly retail gasoline and diesel prices (Price). We constructed a model for predicting the prices for the following week from the obtained price time-series data (autoregression task). The  $X_i \in [0, 1]^{D \times T}$  had  $D = 8$  dimensions (corresponding to eight locations in the U.S.) and  $T = 1,223$  time steps (corresponding to April 5th, 1993, to September 5th, 2016). For both tasks, following [14], the first 67% of each time series was used for training, and the remaining 33% was used for testing. We normalized the values of each dataset in such a way that the values in the training data were in  $[0, 1]$  for each dimension, as in [16].

1) *Implementation*: Since we were solving the autoregression tasks, we used Eq. (7) for the objective function. In these tasks, the overall structure of the proposed method and the hyperparameters candidates were the same as in the event-sequence experiment in Section IV-A.

2) *Results*: We evaluated the methods by using the average test root mean squared error (RMSE) after 1000 iterations

and that of the best case. We compared the results against those of three baseline methods, VAR, DyBM, and CNN. For DyBM and CNN, we used as the same implementation of them in the event-sequence experiment. For VAR, we simply used Eq. (11) for the prediction function. As shown in Table II, the RMSE for the proposed method was comparable to or better than those of the baselines. Moreover, the RMSE values for the proposed method were lower than the 0.0734 [14] and 0.0698 [16] for the Sunspot data and the 0.0564 [14] and 0.0399 [16] for the Price data, which were reported as the results from experiments including other DyBM variants and LSTM models. These results indicate that the convolutional structure and our temporal parameterization work well even for ordinary time-series data. The selected hyperparameters for the proposed method were  $c = 0.01$ ,  $K = 4$ ,  $\lambda = 0.85$ ,  $\mu = 0.85$ ,  $l_0 = 1.0$ , and  $l = 1.0$  for Sunspot, and  $c = 0.01$ ,  $K = 8$ ,  $\lambda = 0.85$ ,  $\mu = 0.85$ ,  $l_0 = 1.0$ , and  $l = 1.0$  for Price.

## V. CONCLUSION

We proposed a time-discounting convolution method that can handle time-shift invariance in event sequences and has robustness against the uncertainty in timestamps while maintaining the important capabilities of time-series models. Experimental evaluation demonstrated that the proposed method was comparable to or even better than state-of-the-art methods in several prediction tasks using event sequences with ambiguous timestamps and ordinary time-series data. The next step in our work is to develop a learning algorithm in an online manner for the proposed method. Actually, we can approximately update the model parameters in an online manner without back propagation through infinite sequences or storing infinite sequences by leveraging dynamic pooling. Increasing the interpretability of our method is another interesting next step.

## ACKNOWLEDGMENTS

Takayuki Katsuki and Takayuki Osogami were supported in part by JST CREST Grant Number JPMJCR1304, Japan.

## APPENDIX

### A. Specific gradients for model parameters

Here, we show the gradients used in our learning of the model parameters by mini-batch gradient descent in Section III-C. The gradient of  $\mathcal{L}^{[m]}(\boldsymbol{\theta})$  with respect to the parameter  $U_{k,i,j}$  is

$$\frac{\partial \mathcal{L}^{[m]}(\boldsymbol{\theta})}{\partial U_{k,i,j}} = \frac{\partial \mathcal{L}^{[m]}(\boldsymbol{\theta})}{\partial \mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta})} \left[ \frac{\partial \mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta})}{\partial \boldsymbol{\eta}_{j,k}} \right]^\top \frac{\partial \boldsymbol{\eta}_{j,k}}{\partial U_{k,i,j}}, \quad (14)$$

where we shorten the partial set of the inputs of the function  $h(\bullet)$  on the right-hand side of Eq. (3) related to the  $j$ -th element of  $\mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta})$  and the index  $k$  in the  $K \times T$ -dimensional inputs as  $\boldsymbol{\eta}_{j,k}$  and

$$\frac{\partial \boldsymbol{\eta}_{j,k}}{\partial U_{k,i,j}} = \left\{ \sum_{\tau_k} \lambda^{d+\tau_k} x_i^{[m-d-\tau_k]} \right\}_{d=1}^T. \quad (15)$$

The gradient of  $\mathcal{L}^{[m]}(\boldsymbol{\theta})$  with respect to the parameters  $V_{k,\tau_k,i,j}$  and  $b_k$ ,  $\frac{\partial \mathcal{L}^{[m]}(\boldsymbol{\theta})}{\partial V_{k,\tau_k,i,j}}$  and  $\frac{\partial \mathcal{L}^{[m]}(\boldsymbol{\theta})}{\partial b_k}$ , are almost the same to Eq (14) — they differ only for the gradient of  $\boldsymbol{\eta}_{k,j}$  with regard to  $V_{k,\tau_k,i,j}$  and  $b_k$ . We show them simply as

$$\frac{\partial \boldsymbol{\eta}_{j,k}}{\partial V_{k,\tau_k,i,j}} = \left\{ \mu^d x_i^{[m-d-\tau_k]} \right\}_{d=1}^T, \quad (16)$$

$$\frac{\partial \boldsymbol{\eta}_{j,k}}{\partial b_k} = \{-1\}_{d=1}^T. \quad (17)$$

Here,  $\frac{\partial \mathcal{L}^{[m]}(\boldsymbol{\theta})}{\partial \boldsymbol{\eta}_{j,k}}$  for autoregression problems with Eq. (7) is

$$\frac{\partial \mathcal{L}^{[m]}(\boldsymbol{\theta})}{\partial \mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta})} = 2\mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta}) - 2\mathbf{X}^{[m]}, \quad (18)$$

and that for classification problems with Eq. (8) is

$$\frac{\partial \mathcal{L}^{[m]}(\boldsymbol{\theta})}{\partial \mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta})} = \frac{1 - \mathbf{y}^{[m]}}{1 - \boldsymbol{\sigma}(\mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta}))} - \frac{\mathbf{y}^{[m]}}{\boldsymbol{\sigma}(\mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta}))}. \quad (19)$$

Through the function  $h(\bullet)$  and the gradient  $\frac{\partial \mathbf{f}(\mathbf{X}^{[<m]}, \boldsymbol{\theta})}{\partial \boldsymbol{\eta}_{j,k}}$ , other functions, such as the activation function, and other layers can be applied to our model and the learning algorithm.

## REFERENCES

- [1] Y. Cheng, F. Wang, P. Zhang, and J. Hu, "Risk prediction with electronic health records: A deep learning approach," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 432–440.
- [2] Z. Che, Y. Cheng, S. Zhai, Z. Sun, and Y. Liu, "Boosting deep learning risk prediction with generative adversarial networks for electronic health records," in *Data Mining (ICDM), 2017 IEEE International Conference on*. IEEE, 2017, pp. 787–792.
- [3] T. Katsuki, M. Ono, A. Koseki, M. Kudo, K. Haida, J. Kuroda, M. Makino, R. Yanagiya, and A. Suzuki, "Risk prediction of diabetic nephropathy via interpretable feature extraction from ehr using convolutional autoencoder," *Studies in health technology and informatics*, vol. 247, pp. 106–110, 2018.
- [4] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Berlin Heidelberg, 2005, vol. Part I.
- [5] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *The annals of mathematical statistics*, vol. 37, no. 6, pp. 1554–1563, 1966.
- [6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [8] G. W. Taylor, G. E. Hinton, and S. T. Roweis, "Modeling human motion using binary latent variables," in *Advances in neural information processing systems*, 2007, pp. 1345–1352.
- [9] G. E. Hinton and A. D. Brown, "Spiking boltzmann machines," in *Advances in neural information processing systems*, 2000, pp. 122–128.
- [10] I. Sutskever and G. Hinton, "Learning multilevel distributed representations for high-dimensional sequences," in *Artificial Intelligence and Statistics*, 2007, pp. 548–555.
- [11] I. Sutskever, G. E. Hinton, and G. W. Taylor, "The recurrent temporal restricted boltzmann machine," in *Advances in Neural Information Processing Systems*, 2009, pp. 1601–1608.
- [12] T. Osogami and M. Otsuka, "Seven neurons memorizing sequences of alphabetical images via spike-timing dependent plasticity," *Scientific Reports*, vol. 5, p. 14149, 2015.
- [13] —, "Learning dynamic boltzmann machines with spike-timing dependent plasticity," *arXiv preprint arXiv:1509.08634*, 2015.
- [14] S. Dasgupta and T. Osogami, "Nonlinear dynamic Boltzmann machines for time-series prediction," in *The 31st AAAI Conference on Artificial Intelligence (AAAI-17)*, January 2017.

- [15] H. Kajino, "A functional dynamic Boltzmann machine," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017, pp. 1987–1993.
- [16] T. Osogami, H. Kajino, and T. Sekiyama, "Bidirectional learning for time-series models with hidden units," in *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, August 2017, pp. 2711–2720.
- [17] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Machine Learning for Healthcare Conference*, 2016, pp. 301–318.
- [18] S. Xiao, J. Yan, X. Yang, H. Zha, and S. M. Chu, "Modeling the intensity function of point process via recurrent neural networks," in *AAAI*, 2017, pp. 1597–1603.
- [19] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific reports*, vol. 8, no. 1, p. 6085, 2018.
- [20] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.
- [21] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Ijcai*, 2015, pp. 2327–2333.
- [22] D. Singh, E. Merdivan, S. Hanke, J. Kropf, M. Geist, and A. Holzinger, "Convolutional and recurrent neural networks for activity recognition in smart environment," in *Towards Integrative Machine Learning and Knowledge Extraction*. Springer, 2017, pp. 194–205.
- [23] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Unsupervised representation learning of structured radio communication signals," in *Sensing, Processing and Learning for Intelligent Machines (SPLINE), 2016 First International Workshop on*. IEEE, 2016, pp. 1–5.
- [24] K. Bascol, R. Emonet, E. Fromont, and J.-M. Odobez, "Unsupervised interpretable pattern discovery in time series using autoencoders," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. Springer, 2016, pp. 427–438.
- [25] T. Katsuki, M. Ono, A. Koseki, M. Kudo, K. Haida, J. Kuroda, M. Makino, R. Yanagiya, and A. Suzuki, "Feature extraction from electronic health records of diabetic nephropathy patients with convolutional autoencoder," in *AAAI 2018 Joint Workshop on Health Intelligence*, 2018.
- [26] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 1715–1725.
- [27] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1529–1537.
- [28] F. Wang, N. Lee, J. Hu, J. Sun, and S. Ebadollahi, "Towards heterogeneous temporal clinical event pattern discovery: a convolutional approach," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 453–461.
- [29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference for Learning Representations (ICLR2015)*, 2015.
- [30] M. Makino, M. Ono, T. Itoko, T. Katsuki, A. Koseki, M. Kudo, K. Haida, J. Kuroda, R. Yanagiya, and A. Suzuki, "Artificial intelligence predicts progress of diabetic kidney disease-novel prediction model construction with big data machine learning," 2018.
- [31] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.