FOTIC: Fourier Transform on Continuous-time Convolutions Model of event sequences. Final status report

Maria Ivanova [1]

Abstract

Event sequences data, specifically transactional data, occur in various domains, including e-commerce, healthcare, and finance. This data is typically large in terms of amount of available data and the length of event sequences per client, and this data is often sparse and non-uniform, thus, the classic approaches for time series processing are inapplicable.

To allow continuous time, the COTIC (Zhuzhel et al., 2023) method uses specific parametric intensity functions defined at each moment on top of existing models. Due to the parametric nature, these intensities represent only a limited class of event sequences. The COTIC method is based on a continuous convolution neural network suitable for non-uniform occurrence of events in time. In COTIC, dilations and multi-layer architecture efficiently handle dependencies between events.

In this project, we go further and would like to speed up the convolution operations in the COTIC network via the Fourier transform.

1. Introduction

Processing of transactional data plays a key role in e-comm and banking industry because of the significant financial implications.

In practice existing machine learning methods are designed to make this process more effective. In most methods transactions are grouped by one type of participants (e.g. users, clients) and each of them is presented as a sequence of transactions. Next, sequence encoder (e.g. RNN (Babaev et al., 2022)) is applied to extract features for prediction.

Although such an approach enables learning similarity of transaction participants, it doesn't take into account the nature of event sequences as temporary point processes which closely related to time series with non-uniform measurements.

2. Related work

Constant advances in deep learning techniques make researchers look for efficient neural architectures to model event sequences. The work (Mei & Eisner, 2017) adapts standard LSTM network to work with non-uniform sequences as this architecture efficiently handles long-term dependencies and deals with the effects not taken into account by the Hawkes process. Despite the fact that LSTM network is superior to standard RNNs in capturing long-term time dependencies in data, it has some drawbacks as well, for example, gradient explosion or problem with the parallelization of the computational process due to the sequential nature of data supplied to the model. To overcome these issues, the Transformer Hawkes Process (THP) (Zuo et al., 2020) was proposed. It adopts the attention mechanism for event sequence modelling. While efficiently identifying long-term dependencies, it can also be optimized in a parallel manner since THP doesn't have a recurrent structure. However, training time of this model is still relatively long.

The COTIC(Zhuzhel et al., 2023) method is based on one-dimensional continuous convolutional neural network architecture and it constructs representations of a temporal point process and predicts its' intensity. The COTIC model takes into account the continuous-time structure of the point process while learning its parameters and predicting target variables from representations at any point in time. The COTIC model doesn't imply any parametric structure for the probability interpolation between events, thus it is possible to model large variety of possible dependecies between events.

The training time of the COTIC approach is comparable to that of existing baselines but has better performance than others. The only model that provides comparable results but much slower than COTIC method, is NeuralHawkes, because it is based on a continuous recurrent neural network. CKConv method (Romero et al., 2021) formulates kernels as continuous functions to process sequential data, which

[1]Skoltech, Moscow. Correspondence to: Maria Ivanova <maria.ivanova@skoltech.ru>.

Selected Topics in Data Science, 2023.

can construct arbitrarily large kernels. CKConv speed up the convolution operations via the convolution theorem with F the Fourier transform.

In our project we would like to apply the Fourier transform for the convolution operations of the COTIC method and compare the results in terms of efficiency and performance.

3. Preliminaries

3.1. Intensity function and likelihood

When one deals with event sequences, there is an open question: how to consider the non-uniform nature of time lags. One of the approaches is to use the intensity function.

Let's consider the event sequence $\boldsymbol{S} = \{(t_i, m_i)\}_{i=1}^{L}$. Each event is described as a tuple of time and event mark. One can consider return time $T_i = t_i - t_{i-1}$ as a random variable. Then, one can introduce its PDF, which shows the probability of the event to happen in an infinitely small interval, and the Survival Function, equal to the probability that the event has not happened till the moment of t:

$$f(t) = \lim_{\Delta t \to +0} \frac{\mathbb{P}\left(t < T_i \leq t + \Delta t\right)}{\Delta t}, \tag{1}$$

$$S(t) = \mathbb{P}\left(T_i \geq t\right) = \int_t^{\infty} f(z)dz. \tag{2}$$

Consequently, we denote the intensity function of the sequence:

$$\lambda(t) = \mathbb{E}\left[\frac{dN(t)}{dt}\right] = \frac{f(t)}{S(t)}, \tag{3}$$

where $N(t) = \#\{t_j \colon t_j < t\}$ – the number of events occurred before time t.

The main benefit of using the intensity function instead of the PDF is that it has less restrictions. The intensity has to be non-negative and should lay in L_1. There is no need to check that it integrates to 1 as it would be if we were to work with the PDF.

In the same time, it is easy to connect the intensity function with the PDF. If one considers an event sequence $S_{1:k}$, the chain rule should be applied for the calculation of its likelihood:

$$f(s) = f(t_1)f(t_2|t_1) \cdot \ldots \cdot f(t_k|t_1, \ldots, t_{k-1}) =$$
$$= \exp\left(-\int_0^T \lambda(t)dt\right) \prod_{j=1}^{k} \lambda(t_j). \tag{4}$$

Once a $model(\boldsymbol{s}, \theta)$ which returns the intensity is defined, it is possible to optimize this model via the likelihood maximization. However, one should notice that, from the computational stability point of view, it is beneficial to optimize the logarithm of likelihood instead.

However, one can notice that it is not straightforward to compute the integral in the likelihood, as the neural net predictions can be complex and cannot be integrated easily. Regarding this obstacle, Monte-Carlo estimator of the integral may be considered.

3.2. COTIC model of event sequences

There are plenty of works which cover event sequence forecasting. In this section, we provide a brief review of the COTIC method we compare our model with.

3.2.1. Notation

Suppose we observe n sequences in a sample $D = \{S_i\}_{i=1}^{n}$. Each sequence consists of tuples $S_i = \{(t_{ij}, \mathbf{m}_{ij})\}_{j=1}^{T_i}$, where $t_{ij} \in [0, T]$ are event times, and $\mathbf{m}_{ij} \in \mathbb{R}^d$ denote event type marks for the i-th sequence in a dataset. If, instead of $\mathbb{R}^d$, we have a discrete set of possible marks $\{1, \ldots, K\}$ we further associate an embedding of each mark instead. We use the notation $S = \{(t_j, \mathbf{m}_j)\}_{j=1}^{k}$ when an arbitrary sequence from the dataset is meant. For the current event number k', we denote the observations history as $S_{1:k'} = \{(t_j, \mathbf{m}_j)\}_{j=1}^{k'}$.

3.2.2. Problem statements

For an event sequence, we consider a main problem for the reconstruction of the intensity of a point process and two downstream problems on top of it: return time and event type prediction. The model itself can be divided into backbone and three heads: intensity, return time and event type. Backbone is trained using the likelihood function only and, thus, downstream problems do not alter the embeddings.

Return time prediction. The goal is to predict the time difference $\Delta t_{k+1} = t_{k+1} - t_k$ from the current event k to the next one t_{k+1} using $S_{1:k}$.

Event type prediction. Events belong to K distinct types. Thus, one could aim to predict the type of the next event $\mathbf{m}_{k+1}$ using $S_{1:k}$.

3.2.3. Continuous convolution of event sequences

As the time intervals $[t_j, t_{j+1}]$ are not necessarily equal to each other, the considered time series are nonuniform. Standard 1-dimensional CNNs are designed for equally lagged data, however, we can extend this idea to a nonuniform case. The considered event sequence S can be rewritten in the functional form:

$$\mathbf{m}(t) = \sum_{j=1}^{k} \mathbf{m}_j \delta(t - t_j), \tag{5}$$

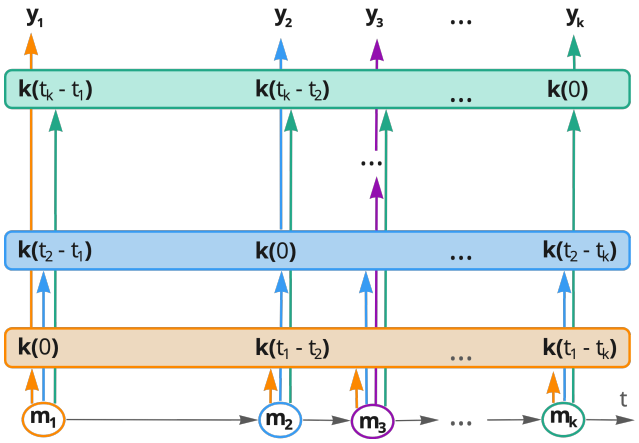

Figure 1. Scheme of the continuous-time convolutional layer used in COTIC model.

where $\delta(\cdot)$ is the Dirac delta function.

Let $\mathbf{k}(t)\colon \mathbb{R} \to \mathbb{R}^{d_{out} \times d}$ be a kernel function. In order to make the model causal and prevent peeking into future, let's state $\mathbf{k}(t) = 0,\ t < 0$. We use the following definition of the convolution:

$$\mathbf{y}(t) = (\mathbf{k} * \mathbf{m})(t) = \int_0^\infty \mathbf{k}(t-\tau)\mathbf{m}(\tau)d\tau = \sum_{j=1}^{k} \mathbf{k}(t-t_j)\mathbf{m}_j \tag{6}$$

The last equality holds due to the specific form of $\mathbf{m}(t)$ given in (5).

In our neural network layer, we can use the vector function of time $\mathbf{y}(t)$ in two ways. The first way corresponds to our need to get $\mathbf{y}(t)$ at some specific point. In this case, we query $\mathbf{y}(t)$ at this moment of time to have a representation. The second way corresponds to the usage of $\mathbf{y}(t)$ as an input to the next convolutional layer. In this case, we limit $\mathbf{y}(t)$ to the set $\{t_1, \ldots, t_k\}$ obtaining a sequence of vectors $\{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ with $\mathbf{y}_i = \mathbf{y}(t_i)$.

Now we use the functional form (5) to have an input to the next layer. The scheme of this continuous convolutional layer is presented in Figure 1. This approach allows us not to lose the temporal structure for all the layers and use all the benefits of the continuous convolution instead of the classic one. Also, in contrast to (Shi et al., 2021) our model can return the intensity in any point and do not imply any interpolations and there is no question on how one should choose points.

3.2.4. COTIC MODEL ARCHITECTURE

Continuous 1-dimensional convolution introduced by (6) allows us to get a new continuous representation of an input event sequence. We propose to parameterize convolution kernel $\mathbf{k}$ by a fully-connected neural network and stack several such convolutions to obtain embeddings of the considered sequences.

To receive embeddings at the final layer we apply different non-uniform convolution kernels to previous layers: in more detail, the proposed model applies L convolution layers to the input $\mathbf{m}(t)$ with non-linearities σ being the LeakyReLU activation:

$$\mathbf{h}_{i,1:k} = \sigma\left(\mathbf{k}_{\theta_L} * \sigma\left(\ldots * \sigma\left(\mathbf{k}_{\theta_2} * \sigma\left(\mathbf{k}_{\theta_1} * \mathbf{m}(t)\right)\right)\ldots\right)\right).$$

In order to reduce the computational complexity on each layer we use limited kernel size, that restrict the number of previous points. Also, we use dilated convolution to increase the receptive field. So, the result of the application of L convolutions layers to the sequence of events $S_{i,1:k}$ is the matrix $H_{i,1:k}$ that consists of embeddings for each event $\mathbf{h}_{ij} \in \mathbb{R}^d$. The embeddings induce the function $\mathbf{h}(t)$ of the form (5).

Now we are ready to produce intensity function, expected event type and expected return time via the following formulas:

$$\Delta\widehat{t}_{i,k+1} = \text{MLP}_1\left(\mathbf{h}_{i,k}\right), \tag{7}$$

$$\widehat{m}_{i,k+1} = \text{MLP}_2\left(\mathbf{h}_{i,k}\right), \tag{8}$$

$$\boldsymbol{\lambda}(t) = \text{softplus}\left(\text{Linear}\left(\sigma\left(\mathbf{k}_{\theta_\lambda} * \mathbf{h}(t)\right)\right)\right). \tag{9}$$

Here MLP_i are multi-layer perceptron prediction head networks, $\Delta\widehat{t}_{i,k+1}$ is the predicted return time for the $(k+1)$-th event, $\widehat{m}_{i,k+1}$ is the predicted event type, $\boldsymbol{\lambda}(t)$ is the estimated vector process intensity function that has the output dimensional equal to the number of possible event types.

To sum up, once the embeddings $\mathbf{h}_i$ of a sequence are obtained by a L-layer CCNN, they are fed into MLPs to produce the final predictions. We use one additional continuous convolution followed by a linear layer to estimate the intensity function of the process at an arbitrary time step.

3.2.5. LOSS FUNCTIONS AND TRAINING PIPELINE

The proposed model is trained with a combined loss function consisting of several terms.

In order to train the convolutional part of the model, we use *the negative log-likelihood* derived from (4) as the loss function:

$$L_{ll} = \int_0^{t_k} \lambda(t)dt - \sum_{j=1}^{k} \log \lambda_{m_j}(t_j), \tag{10}$$

where the integral of the intensity function is calculated by the Monte-Carlo estimator.

The prediction heads (7), (8) are trained with the standard *LogCosh* and *CrossEntropy* losses respectively:

$$L_{time}\left(\hat{t}_{k+1}, t_{k+1}\right) = LogCosh\left(\hat{t}_{k+1} - t_{k+1}\right) = \\ = \left(\hat{t}_{k+1} - t_{k+1}\right) + \\ + \log\left(1 + e^{-2\left(\hat{t}_{k+1} - t_{k+1}\right)}\right) \quad (11)$$

$$L_{type}\left(\widehat{m}_{k+1}, m_{k+1}\right) = CrossEntropy\left(\widehat{m}_{k+1}, m_{k+1}\right) = \\ = -\sum_{l=1}^{K} m_{k+1}^{(l)} \log \frac{\exp\left\{\left(\widehat{m}_{k+1}^{(l)}\right)\right\}}{\sum_{c=1}^{K} \exp\left\{\left(\widehat{m}_{k+1}^{(c)}\right)\right\}}, \quad (12)$$

where $m_{k+1}^{(l)}$ are the scores for the true event type and $\widehat{m}_{k+1}^{(l)}$ are the predicted event type scores for the $(k+1)$-th event in a sequence. The loss functions above are calculated for a single sequence. Given a batch of sequences, the losses are averaged.

The resulting loss function for the prediction head training is a weighted sum: $L_{heads} = \alpha L_{time} + \beta L_{type}$.

The training pipeline for the model is the following. We first train its convolutional part with the log-likelihood loss (10) for N_0 epochs while the prediction heads are frozen. Afterward, we continue training the whole model jointly with the combined loss $L = L_{ll} + L_{heads}$.

4. Experiments

4.1. Datasets

We used Age dataset of event sequences to compare our method with the COTIC. Table 1 summarizes statistics for the dataset, and below we provide additional details.

Dataset	# of event types	Avg seq. len.	Median seq. len.	# of seq.
Age	60	862.4	845	30000

Table 1. Statistics of the Age dataset used for the evaluation of the models.

Age dataset contains sequences of transaction records stemming from clients at financial institutions. For each client, we have a sequence of transactions. We describe each transaction with a discrete code, identifying the type of transaction, and the Merchant Category Code, such as a "bookstore", "ATM", "drugstore", and more. According to transactions' history, we categorize clients into age categories. The data provides each client's age bin with four different bins in total.

4.2. Training Details

For all the experiments, we split the datasets into 3 parts (train set, test set and validation set) in the ratio $8:1:1$.

We train the models via Adam optimizer for the maximum number of 100 epochs. Training details for all the baseline methods are provided below. We performed experiments in two scenarios.

The code of the COTIC method and experiments of the proposed COTIC FFT method can be found in the GitHub repository [1].

4.2.1. Scenario 1

Scenario 1. We limit all sequences by the percent (100%, 80%, 60%, 40% and 20%) of the median number of events. We used batch size 240 and one GPU A100 to provide experiments.

The results of the Scenario 1 experiments are presented in Table 2.

4.2.2. Scenario 2

We limit all sequences by the number of event types which occurs most of all (60, 50, 40, etc.) We used batch size 160 and one GPU A100 to provide experiments.

The results of the Scenario 2 experiments are presented in Table 3.

5. Conclusions

As per experiments the training time of the COTIC FFT method is higher than vanila COTIC method. However, the accuracy of the Return Time and Event Type metrics sometimes higher for COTIC FFT method than vanila COTIC method.

[1] https://github.com/ipmaria/COTIC

Median seq. len	Model	Training time (in seconds) per epoch	LogLikelihood per event ↑	Return Time MAE↓	Event Type Accuracy↑
100%	COTIC	114.29	0.899	0.756	0.320
	FFT COTIC (ours)	219.05	0.513	0.752	0.323
90%	COTIC	107.49	0.674	0.686	0.322
	FFT COTIC (ours)	219.60	**2.157**	**0.671**	**0.343**
80%	COTIC	91.20	-0.834	0.766	0.330
	FFT COTIC (ours)	169.20	**1.954**	**0.665**	**0.335**
70%	COTIC	86.15	-3.239	0.720	**0.368**
	FFT COTIC (ours)	139.00	-0.365	**0.690**	0.356
60%	COTIC	70.12	-0.358	0.792	0.323
	FFT COTIC (ours)	148.25	0.806	0.775	0.330
50%	COTIC	58.77	-0.868	0.811	0.325
	FFT COTIC (ours)	118.49	1.123	0.711	0.333
40%	COTIC	48.37	1.899	0.783	0.335
	FFT COTIC (ours)	115.32	-0.403	0.821	0.327
30%	COTIC	45.13	-0.562	0.846	0.328
	FFT COTIC (ours)	126.08	1.932	0.787	0.342
20%	COTIC	28.09	1.226	0.876	0.332
	FFT COTIC (ours)	89.52	0.407	0.874	0.333
10%	COTIC	16.79	1.788	0.895	0.338
	FFT COTIC (ours)	50.91	1.190	0.908	0.339

Table 2. Evaluation of COTIC and our COTIC FFT method presented in 3.2 on the Age dataset described in 4.1 per Scenario 1. We use three quality metrics: MAE of Return Time estimation, Accuracy of Event Type prediction and Log-Likelihood per event – for intensity function restoration. The return time value is normalized. See 3.2 and 4.2 for the details.

# Number of events types	seq. len	Model	Training time (in seconds) per epoch	Log Likelihood per event ↑	Return Time MAE↓	Event Type Accuracy↑
60	1148	COTIC	154.79	-1.347	0.741	0.335
		FFT COTIC (ours)	295.70	-0.034	0.748	0.319
50	1147	COTIC	153.16	-3.157	0.664	**0.373**
		FFT COTIC (ours)	259.12	0.341	**0.657**	0.342
40	1146	COTIC	153.24	1.892	0.759	0.346
		FFT COTIC (ours)	255.87	-3.091	**0.685**	**0.380**
30	1141	COTIC	154.13	1.169	0.688	0.344
		FFT COTIC (ours)	284.29	0.848	**0.658**	**0.363**
20	1129	COTIC	150.05	-1.791	0.680	0.403
		FFT COTIC (ours)	340.10	1.731	0.680	0.382
10	1098	COTIC	142.96	-0.517	0.774	0.380
		FFT COTIC (ours)	275.74	-1.597	0.770	0.386
5	1051	COTIC	133.94	2.246	0.790	0.445
		FFT COTIC (ours)	251.31	-0.941	0.790	0.431
1	791	COTIC	108.66	0.975	0.882	0.539
		FFT COTIC (ours)	245.96	-0.071	0.896	0.537

Table 3. Evaluation of COTIC and our COTIC FFT method presented in 3.2 on the Age dataset described in 4.1 per Scenario 2. We use three quality metrics: MAE of Return Time estimation, Accuracy of Event Type prediction and Log-Likelihood per event – for intensity function restoration. The return time value is normalized. See 3.2 and 4.2 for the details.

References

Babaev, D., Ovsov, N., Kireev, I., Ivanova, M., Gusev, G., Nazarov, I., and Tuzhilin, A. Coles: Contrastive learning for event sequences with self-supervision. In *Proceedings of the 2022 International Conference on Management of Data*, pp. 1190–1199, 2022.

Mei, H. and Eisner, J. M. The neural hawkes process: A neurally self-modulating multivariate point process. *Advances in neural information processing systems*, 30, 2017.

Romero, D. W., Kuzina, A., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. Ckconv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021.

Shi, H., Zhang, Y., Wu, H., Chang, S., Qian, K., Hasegawa-Johnson, M., and Zhao, J. Continuous cnn for nonuniform time series. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3550–3554. IEEE, 2021.

Zhuzhel, V., Grabar, V., Boeva, G., Zabolotnyi, A., Stepikin, A., Zholobov, V., Ivanova, M., Orlov, M., Kireev, I., Burnaev, E., et al. Continuous-time convolutions model of event sequences. *arXiv preprint arXiv:2302.06247*, 2023.

Zuo, S., Jiang, H., Li, Z., Zhao, T., and Zha, H. Transformer hawkes process. In *International conference on machine learning*, pp. 11692–11702. PMLR, 2020.