Continual Learning for Time-to-Event Modeling

Manisha Dubey CS17RESCH11003@IITH.AC.IN
Indian Institute of Technology Hyderabad, India

P.K. Srijith SRIJITH@CSE.IITH.AC.IN
Indian Institute of Technology Hyderabad, India

Maunendra Sankar Desarkar MAUNENDRA@CSE.IITH.AC.IN
Indian Institute of Technology Hyderabad, India

Abstract

Temporal point process serves as an essential tool for modeling time-to-event data in continuous time space. Despite having massive amounts of event sequence data from various domains like social media, healthcare etc., real world application of temporal point process are not capable of thriving in continually evolving environment with minimal supervision while retaining previously learnt knowledge. To tackle this, we propose *HyperHawkes*, a hypernetwork based continually learning temporal point process for continuous modeling of time-to-event sequences with minimal forgetting. We demonstrate the application of the proposed framework through our experiments on two real-world datasets.

Keywords: Time-to-Event Modeling, Temporal Point Process, Continual Learning

1. Introduction

Various applications like earthquake occurrences, financial transactions etc. are associated with the collection of discrete asynchronous events where event occurrences are represented with timestamps. Each event sequence, consisting of a series of timestamps, can be associated with a separate entity. For example, in social media, each user can be associated with the time of posting a tweet, and each tweet can be viewed as an event. A principled mathematical framework to model such sequences in continuous time space is the temporal point process (14). Hawkes process (24), a self-exciting point process has been widely used in a wide array of practical applications like epidemic modeling(6), earthquake prediction(8) etc. Recent works improve the performance of the classic Hawkes process by considering neural networks for modeling such event sequences (5; 2; 4; 1). Despite having improved performance, the neural network-based Hawkes process is challenged by a limitation on the practical side. Real-world event occurrences happen sequentially in continuous streams. Different techniques (19; 23; 20; 12) have been proposed in this regard by consolidating knowledge either in various spaces like data, weight or meta space for the domain of vision and NLP. However, there is no existing literature for time-to-event modeling. Therefore, a realistic and challenging problem is to continually learn time-to-event models in an ever-changing environment while retaining previously learned knowledge.

Towards this, we consider a practical and under-explored setting where we consider a continual learning setup where time-to-event prediction tasks (sequences) arrive sequentially in an online manner. To the best of our knowledge, there is no prior work on continual learning for time-to-event modeling. We aim to develop neural network based Hawkes process models which can continually

learn while retaining previous knowledge. To achieve these, we propose *HyperHawkes*, a hypernetwork based Hawkes process which empowers the neural network based Hawkes process models to perform continual learning of time-to-event data. HyperHawkes uses a hypernetwork or a meta-network which can generate sequence-specific parameters for the neural network based Hawkes process using a meta-information about the sequence called sequence descriptors. By incorporating descriptor-conditioned hypernetwork, we are able to learn and predict sequence specific parameters and consequently use these parameters to predict event occurrences on unseen sequences. We recast the descriptor-conditioned hypernetwork to include a regularizer over the hypernetwork output which encourages the hyper-network to correctly generate the parameters for the previous tasks while learning to generate the parameters for new task. Our contributions can be summarized as:

- We propose a novel problems setup for continual learning of time-to-event prediction for applications involving event sequence data.
- We propose *HyperHawkes*, a sequence descriptor-conditioned hypernetwork based neural Hawkes process which can generate sequence specific parameters to address continual learning of event sequences. We augment HyperHawkes with continual learning abilities by employing a regularization technique, hence avoiding catastrophic forgetting over time-to-event sequences.
- We demonstrate the effectiveness of the proposed model on this setup for two real-world datasets and show that the proposed model is able to predict sequences continually while retaining information from previous event sequences.

2. Proposed Model

Problem Definition: Assume we are given a collection of N sequences $\mathcal{D} = \{(\mathcal{T}^1, \mathbf{d}^1), (\mathcal{T}^2, \mathbf{d}^2), ..., (\mathcal{T}^N, \mathbf{d}^N\}$ where $\mathbf{d}^i$ represents the sequence descriptor and $\mathcal{T}^i$ represents the times of occurrence of n^i events in the i^{th} sequence, i.e. $\mathcal{T}^i = \{t^i_j\}_{j=1}^{n^i}$ and we assume these sequences arrive one after the other in the order of their index. Our goal is to continually learn the sequences while avoiding catastrophic forgetting from the previous sequences.

We propose *HyperHawkes*, a hypernetwork based neural Hawkes process for time-to-event modeling. For time-to-event modeling, we consider the Fully Neural Hawkes Process (FNHP) (1) as the base model. Integrating hypernetwork with fully neural Hawkes process, we introduce descriptor-conditioned hypernetwork to generate weights for each sequence which can perform time-to-event modeling. The descriptor-conditioned hypernetwork learns separate weights of FNHP for each sequence. Inspired by (12), we use this framework for continual learning of tasks by using a hypernetwork based regularizer. Now we discuss each of these pieces in detail.

2.1. Fully Neural Hawkes Process

We employ fully neural Hawkes process (1) as base model for time-to-event modeling. It uses a combination of recurrent neural network and feedforward neural network to model the intensity function. We represent history by using hidden representations generated by recurrent neural networks (RNNs) at each time step. The hidden representation $\boldsymbol{h}^i_j$ at time t^i_j is obtained as $\boldsymbol{h}^i_j = RNN(\tau^i_j, \boldsymbol{h}^i_{j-1}; W^i_r)$ where $\tau^i_j = t - t^i_j$ and W^i_r represents the parameters associated with RNN. This is used as input to a feedforward neural network to compute the intensity function (hazard function) and consequently the cumulative hazard function for computing the likelihood of event

occurrences. We model conditional intensity as a function of the elapsed time from the most recent event $\lambda(t|H_t^i) = \lambda(t - t_j^i|\boldsymbol{h}_j^i)$ where $\lambda(\cdot)$ is a non-negative function referred to as a hazard function. Therefore, we define cumulative hazard function in terms of inter-event interval (τ_j^i)

$$\Phi(\tau_j^i|\boldsymbol{h}_j^i) = \int_0^{\tau_j^i} \lambda(s|\boldsymbol{h}_j^i)ds \tag{1}$$

Cumulative hazard function is modeled using a feed-forward neural network (FNN) $\Phi(\tau_j^i|\boldsymbol{h}_j^i) = FNN(\tau_j^i, h_j^i; W_t^i)$. However, cumulative hazard function has to be a monotonically increasing function of τ_j^i and positive valued. We achieve these by maintaining positive weights and positive activation functions in the neural network (17; 1). The hazard function itself can be then obtained by differentiating the cumulative hazard function with respect to τ as $\lambda(\tau_j^i|\boldsymbol{h}_j^i) = \frac{\partial}{\partial \tau_j^i}\Phi(\tau_j^i|\boldsymbol{h}_j^i)$ The log-likelihood of observing event times is defined as follows using the cumulative hazard function-

$$\sum_{j=1}^{n^i} \log p(t_j^i|\mathcal{H}_j^i; W^i) = \sum_{j=1}^{n^i} \big(\log(\frac{\partial}{\partial \tau_j^i}\Phi(\tau_j^i|\boldsymbol{h}_{j-1}^i; W^i)) - \Phi(\tau_j^i|\boldsymbol{h}_{j-1}^i; W^i)\big) \tag{2}$$

where $\tau_j^i = t_j^i - t_{j-1}^i$ and $W^i = \{W_r^i, W_t^i\}$ represents the combined weights associated with RNN and FNN. In NHP, the weights of the networks are learnt by maximizing the likelihood given by (2).

2.2. HyperHawkes: Hypernetwork based Neural Hawkes Process for Continual Learning

Hypernetwork is a meta-network which produces parameters used by other networks (18). As discussed in the above section, the neural Hawkes process comprises of two building blocks - RNN and FNN. So, we use hypernetwork to produce weights for these two components. We use a feed-forward neural network (FNN) to produce parameters $W^i = \{W_r^i, W_t^i\}$ associated with the FNHP. We use two different types of hypernetworks, $f_r(\cdot)$ producing W_r^i and $f_t(\cdot)$ producing W_t^i. Given a sequence description $\mathbf{d}^i$, parameters for the RNN are generated as $W_r^i = f_r(\mathbf{d}^i; \theta_{fr})$ where θ_{fr} denotes the parameters of the hypernetwork (weight vectors of a neural network). Note that the hypernetwork parameters are the same across the sequences. The descriptor $\mathbf{d}^i$ is used to generate the sequence specific parameters. As discussed above, the cumulative hazard function is a monotonically increasing function of τ_j^i and is positive-valued. The hypernetwork which will generate parameters for cumulative hazard function has to fulfill these properties. This can be achieved when hypernetwork generates only positive weights for which we use a positive activation function. Hypernetwork $f_t(\cdot)$ for FNN can be written as $W_t^i = f_t(\mathbf{d}^i; \theta_{ft})$ where θ_{ft} denotes parameters of hypernetwork $f_t(\cdot)$. We propose 2 variants of HyperHawkes **1) HyperHawkes-FNN:** Hypernetwork is considered only for the FNN modeling the cumulative hazard function. **2) HyperHawkes-FNN-RNN:** This variant uses two separate hypernetworks, one to model the RNN modeling the history and the second to model the FNN.

Ideally, we want our model to remember the parameters of the neural Hawkes process for each sequence. A naive approach to achieve this is through storing and replaying over previous data, which is obviously memory expensive and unrealistic. However, *HyperHawkes*, being conditioned on the sequence descriptor, can be modified to handle this problem. The direct use of the *HyperHawkes* training would result in hypernetworks forgetting the generation of the FNHP parameters corresponding to past event sequences. We overcome this by incorporating a regularization on the hypernetwork parameters such that it penalizes any change to the FNHP parameters produced from old sequences.

Given a sequence description $\mathbf{d}^s$ for the descriptor $\mathcal{T}^s$, our descriptor conditioned hypernetwork $f_r(\cdot)$ can generate parameters W_r^s and $f_t(\cdot)$ can generate parameters W_t^s. To perform continual learning, we use regularization to penalize changes in $\{W_r^c, W_t^c\}$ generated for past sequences in order to retain information from those sequences and to learn continually. The regularization is applied to the hypernetwork parameters while learning a new event sequence, and this prevents adaptation of the hypernetworks parameters completely to the new event sequence. For a new event sequence $\mathcal{T}^s$ and its corresponding descriptor $\mathbf{d}^s$, the hypernetwork parameters are learnt by minimizing the following continual learning loss over events in the sequence:

$$= \sum_{j=1}^{n^s} -\log p(t_j^s | \mathcal{H}_j^s; (f_r(\mathbf{d}^s; \theta_{fr}), f_t(\mathbf{d}^s; \theta_{ft}))) + \frac{\beta}{s-1} \sum_{c=1}^{s-1} \Bigg(\| f_r(\mathbf{d}^c; \theta_{fr}) - f_r(\mathbf{d}^c; \bar{\theta}_{fr}) \|^2 + \| f_t(\mathbf{d}^c; \theta_{ft}) - f_t(\mathbf{d}^c; \bar{\theta}_{ft}) \|^2 \Bigg) \tag{3}$$

where $\{\bar{\theta}_{fr}, \bar{\theta}_{ft}\}$ represents the stored hypernetwork parameters after learning until sequence $s-1$ and $\{\theta_{fr}, \theta_{ft}\}$ represent the hypernetwork parameters learnt considering the event sequence s and regularization to avoid forgetting. The regularization term ensures that the newly learnt hypernetwork parameters will be able to produce the required main network parameters from the past event sequences given the sequence descriptor without forgetting and the regularization constant β captures the importance associated with it. So, in this way, we try to retain the information from previous sequences at a meta-level by the use of sequence-conditioned hypernetwork on the top of neural Hawkes process using a simple regularization term within the framework of *HyperHawkes*.

3. Experiments and Results

Datasets Due to paucity of standard datasets for event modeling tasks which contain sequence descriptor as well, we use: **1)Yelp:**[1]: This is a dataset comprising of business information and their check-in information. Each business is associated with 82 attributes like *By Appointment Only, Business Category, Business Timings* etc. Using these attributes, we create a vector of length 1229 representing a business. This vector acts as a sequence descriptor of the business. **2) Meme:**[3]: This dataset(22) tracks the popular phrases and quotes which appear appear most frequently over time in news media and blogs. Each meme is associated with the content and timestamps when they were quoted in the media. We select top 200 english phrases and find Doc2Vec representation of the meme content of length 100. This Doc2Vec representation is considered as sequence descriptor of each meme.

Baselines To the best of our knowledge, the proposed problem statement is the first work along this direction. So, we compare against HyperHawkes without any regularization as baseline. For another baseline, we consider FNHP trained continually. Mean negative log-likelihood (MNLL) and mean absolute error (MAE) are considered as evaluation metrics.

Results and Analysis Table 1 presents the averaged results over all tasks by enabling HyperHawkes for continual learning. We can observe that averaged performance for the proposed model is better than the case when no regularization is incorporated. Also, we can observe that both the proposed variants perform better than the model without using regularization (corresponds to the case

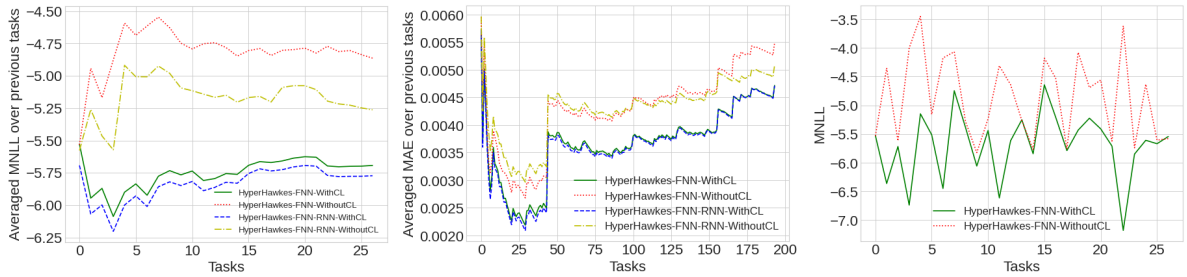
1. https://www.kaggle.com/datasets/yelp-dataset/
3. https://snap.stanford.edu/data/memetracker9.html

Table 1: Results for Continual Learning by comparing the performance of the proposed Hyper-Hawkes with regularization against under HyperHawkes without regularization (Lower MNLL and MAE indicates better performance)

Dataset	FNHP		HyperHawkes-FNN				HyperHawkes-FNN-RNN			
	MNLL	MAE	MNLL		MAE		MNLL		MAE	
			WithoutCL	WithCL	WithoutCL	WithCL	WithoutCL	WithCL	WithoutCL	WithCL
Yelp	-3.48000	0.00163	-4.8627	**-5.6928**	0.00159	**0.00149**	-5.2629	**-5.7727**	0.00152	**0.00150**
Meme	-3.93120	0.00521	-2.8550	**-5.1462**	0.00548	**0.00471**	-3.8254	**-5.1192**	0.00508	**0.00471**

Figure 1: Plots displaying the performance of HyperHawkes for Continual Learning for the variant HyperHawkes-FNN and HyperHawkes-FNN-RNN 1a) Average MNLL over previous sequences for Yelp 1b) Average MAE over previous sequences for Meme 1c) MNLL for each sequence for Yelp HyperHawkes-FNN

(a) (b) (c)



when β from Equation 3 is set to 0). We can also notice that HyperHawkes-FNN and HyperHawkes-FNN-RNN with memorization enabling continual learning performs better than FNHP. Hence the use of regularization within the framework of HyperHawkes supports that proposed method can avoid catastrophic forgetting. Fig 1) displays sequence-wise performance for both the datasets for the proposed variants HyperHawkes-FNN and HyperHawkes-FNN-RNN. Fig 1a) displays average MNLL over previous sequences for both the models for Yelp. This shows that while training the model without regularization over new sequences, the network is unable to retain information learnt from previous sequences, hence MNLL increases as we train new sequences. However, with the use of regularization, we can avoid catastrophic forgetting, hence having lower MNLL for successive tasks. Similar behavior is observed by Fig 1b) as well which displays average MAE over previous sequences for both the variants, with and without CL for Meme. So, this corroborates that use of regularization with HyperHawkes can help in backward transfer. Fig 1c) shows MNLL for each sequence using the model HyperHawkes-FNN-RNN with regularization. MNLL for the model with regularization is having lower MNLL as compared to the model without any regularization. This essentially reflects that the proposed model is able to forward transfer the knowledge learnt from previous sequences as well. So, the model is able to perform forward and backward transfer, which are important continual learning desiderata. To conclude, presented results suggest that proposed framework can aid in avoiding catastrophic forgetting while learning continually.

4. Conclusion

In this work, we address the problem of continual learning for time-to-event modeling. We propose HyperHawkes, a descriptor conditioned hypernetwork based neural Hawkes process which can generate event sequence specific parameters. We augment *HyperHawkes* with regularization which can aid in learning time-to-event sequences continually by avoiding catastrophic forgetting. Our experiments on two real-world datasets demonstrate the effectiveness of the proposed approach.

References

[1] Omi, Takahiro, and Kazuyuki Aihara. "Fully neural network based model for general temporal point processes." Advances in neural information processing systems 32 (2019).

[2] Du, Nan, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. "Recurrent marked temporal point processes: Embedding event history to vector." In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1555-1564. 2016.

[3] Graves, Alex. "Practical variational inference for neural networks." Advances in neural information processing systems 24 (2011).

[4] Xiao, Shuai, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen Chu. "Modeling the intensity function of point process via recurrent neural networks." In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, no. 1. 2017.

[5] Mei, Hongyuan, and Jason M. Eisner. "The neural hawkes process: A neurally self-modulating multivariate point process." Advances in neural information processing systems 30 (2017).

[6] Diggle, Peter, Barry Rowlingson, and Ting-li Su. "Point process methodology for on-line spatio-temporal disease surveillance." Environmetrics: The official journal of the International Environmetrics Society 16, no. 5 (2005): 423-434.

[7] Mohler, George O., Martin B. Short, P. Jeffrey Brantingham, Frederic Paik Schoenberg, and George E. Tita. "Self-exciting point process modeling of crime." Journal of the American Statistical Association 106, no. 493 (2011): 100-108.

[8] Hainzl, Sebastian, D. Steacy, and S. Marsan. "Seismicity models based on Coulomb stress calculations." Community Online Resource for Statistical Seismicity Analysis (2010).

[9] Zuo, Simiao, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. "Transformer hawkes process." In International conference on machine learning, pp. 11692-11702. PMLR, 2020.

[10] Zhang, Qiang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. "Self-attentive Hawkes process." In International conference on machine learning, pp. 11183-11193. PMLR, 2020.

[11] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[12] Von Oswald, Johannes, Christian Henning, João Sacramento, and Benjamin F. Grewe. "Continual learning with hypernetworks." arXiv preprint arXiv:1906.00695 (2019).

[13] Rizoiu, Marian-Andrei, Young Lee, Swapnil Mishra, and Lexing Xie. "A tutorial on hawkes processes for events in social media." arXiv preprint arXiv:1708.06401 (2017).

[14] Daley, Daryl J., and David Vere-Jones. An Introduction to the Theory of Point Processes. Volume II: General Theory and Structure. Springer., 2008.

[15] Bacry, Emmanuel, Iacopo Mastromatteo, and Jean-François Muzy. "Hawkes processes in finance." Market Microstructure and Liquidity 1, no. 01 (2015): 1550005.

[16] Sill, Joseph. "Monotonic networks." Advances in neural information processing systems 10 (1997).

[17] Chilinski, Pawel, and Ricardo Silva. "Neural likelihoods via cumulative distribution functions." In Conference on Uncertainty in Artificial Intelligence, pp. 420-429. PMLR, 2020.

[18] Ha, David, Andrew Dai, and Quoc V. Le. "Hypernetworks." arXiv preprint arXiv:1609.09106 (2016).

[19] Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan et al. "Overcoming catastrophic forgetting in neural networks." Proceedings of the national academy of sciences 114, no. 13 (2017): 3521-3526.

[20] Lopez-Paz, David, and Marc'Aurelio Ranzato. "Gradient episodic memory for continual learning." Advances in neural information processing systems 30 (2017).

[21] Xie, Yujia, Haoming Jiang, Feng Liu, Tuo Zhao, and Hongyuan Zha. "Meta learning with relational information for short sequences." Advances in neural information processing systems 32 (2019).

[22] Leskovec, Jure, Lars Backstrom, and Jon Kleinberg. "Meme-tracking and the dynamics of the news cycle." In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 497-506. 2009.

[23] Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." IEEE transactions on pattern analysis and machine intelligence 40, no. 12 (2017): 2935-2947.

[24] Hawkes, Alan G. "Spectra of some self-exciting and mutually exciting point processes." Biometrika 58, no. 1 (1971): 83-90.