Shou, X., Gao, T., Subramanian, D., Bhattacharjya, D. & Bennett, K. P. Multi-Label Event Prediction in Continuous Time. Proceedings of the AAAI Conference on Artificial Intelligence. To appear 2023.

Concurrent Multi-Label Prediction in Event Streams

Xiao Shou [1] **Tian Gao** [2] **Dharmashankar Subramanian** [2]
Debarun Bhattacharjya [2] **Kristin Bennett** [1]

[1] Rensselaer Polytechnic Institute
[2] IBM AI Research

Abstract

Streams of irregularly occurring events are commonly modeled as a marked temporal point process. Many real-world datasets such as e-commerce transactions and electronic health records often involve events where multiple event types co-occur, e.g. multiple items purchased or multiple diseases diagnosed simultaneously. In this paper, we tackle multi-label prediction in such a problem setting, and propose a novel Transformer-based Conditional Mixture of Bernoulli Network (TCMBN) that leverages neural density estimation to capture complex temporal dependence as well as probabilistic dependence between concurrent event types. We also propose potentially incorporating domain knowledge in the objective by regularizing the predicted probability. To represent probabilistic dependence of concurrent event types graphically, we design a two-step approach that first learns the mixture of Bernoulli network and then solves a least-squares semi-definite constrained program to numerically approximate the sparse precision matrix from a learned covariance matrix. This approach proves to be effective for event prediction while also providing an interpretable and possibly non-stationary structure for insights into event co-occurrence. We demonstrate the superior performance of our approach compared to existing baselines on multiple synthetic and real benchmarks.

1 Introduction

Various types of human activities consist of irregularly occurring events over a period of time. For example, online customer transaction records involve purchases at a particular time for an account associated with an individual, and electronic health records (EHRs) keep track of a patient's health history including diagnoses and treatments throughout their life. Temporal point processes (TPPs) provide a suitable continuous-time mathematical tool for modeling event streams, where discrete events happen irregularly (Daley and Jones 2003). A classic approach to model event sequences as TPPs is through the Hawkes process, in which a simple parametric form is used to capture temporal dependence among events (Hawkes 1971). In the past few years, many researchers have developed neural TPP models that have achieved fruitful results on standard benchmarks for predictive tasks, because neural networks are capable of capturing more complex dependencies (Du et al. 2016; Mei and Eisner 2016; Xiao et al. 2017; Upadhyay, De, and Gomez-Rodriguez 2018; Omi, Ueda, and Aihara 2019; Shchur, Biloš, and Günnemann 2019; Zuo et al. 2020; Zhang et al. 2020a; Shchur et al. 2020; Boyd et al. 2020; Gao et al. 2020; Gu 2021).

Many real-world applications involve event streams with *concurrent labels*, i.e. multiple labels that occur simultaneously in an event (for the recorded temporal granularity). For example, in the aforementioned applications of e-commerce and healthcare, multiple items can be purchased at the time of a transaction, and multiple diseases can be diagnosed during a single provider visit. Importantly, concurrent event label occurrences can be highly correlated. For instance, Amazon's recommender system has the 'frequently bought together' option, and comorbidities such as type 2 diabetes and Alzheimer's disease for the elderly (Chatterjee and Mudher 2018) are very common in healthcare.

Although many proposed neural TPPs excel at solving prediction problems, most are not directly applicable for concurrent multi-label event streams. A notable exception is a recent approach for modeling EHR data using attention-based neural TPP models (Enguehard et al. 2020). This class of model captures long-term, nonsequential dependencies of contexts (Bahdanau, Cho, and Bengio 2014) and also jointly models dependence of time with the associated labels.

As far as we are aware, existing approaches however fail to provide meaningful structural dependence among coinciding event labels at a given timestamp, and are unable to incorporate such dependence for label prediction. We note that although there is some prior work on graphical representations of TPPs (Didelez 2008; Bhattacharjya, Subramanian, and Gao 2018), these are directed graphs that capture historical dependence when each event is associated with exactly one event label. In contrast, our approach involves an undirected graph for representing relations between concurrent labels; it is conceptually similar to prior work on time series graphs (Eichler 1999).

In this paper, we propose a new approach for modeling concurrent event labels using neural TPPs. Our main contributions are as follows:

- We formalize the multi-label prediction problem in event streams and propose a general framework for modeling concurrent labels in event streams. Crucially, our method

allows for both complex temporal dependence as well as probabilistic dependence between concurrent labels.
- We enable potentially incorporating domain knowledge for the prediction task through an approach that regularizes the predicted probability from our model. This can improve model reliability whenever applicable.
- To offer meaningful insights, we propose a two-step procedure for discovering an undirected graphical structure among concurrent event labels and illustrate its effectiveness through a case study on transaction data.
- We conduct an extensive empirical investigation including ablation studies and demonstrate superior performance of our proposed model as compared to state-of-the-art baselines for next event multi-label prediction.

2 Related Work and Background

Temporal Point Processes

A marked temporal point process is a stochastic process that generates not only a timestamp but also a label associated with it: for an generated event sequence $\mathcal{E}_l$, $\mathcal{E}_l = \{(t_i, y_i)\}_{i=1}^{n_l}$, where each event epoch is a tuple of a timestamp and its label. Each timestamp t_i is the time of occurrence and $t_i \in \mathbb{R}^+$, and each label y_i belongs to the label set $\mathcal{L}$, whose cardinality is M. One common approach to characterize a TPP is through the conditional intensity function $\lambda^*(t)$. A classic form of the conditional intensity function is the Hawkes process (also called the self-exciting point process) which has been applied to model many phenomena in social networks, financial systems and Internet Protocol television (IPTV) systems (Hawkes 1971; Zhou, Zha, and Song 2013; Bacry, Mastromatteo, and Muzy 2015; Luo et al. 2015).

Neural Temporal Point Processes

The expressive power of neural networks has enriched the TPP literature. Researchers have applied deep neural networks to model TPPs since the recurrent marked temporal point process (RMTPP) (Du et al. 2016). A review of neural TPP models appeared recently (Shchur et al. 2021). Neural TPPs are able to capture more complex dependencies among events than their parametric counterparts. The main idea in RMTPP is to use an RNN (or its modern variants) to capture the historical dependency of the events via the history embedding or context. The history embedding for i^{th} event $\mathbf{h}_i$ is modeled through a recurrent relation:

$$\mathbf{h}_i = \psi(\mathbf{W^t} t_i + \mathbf{W^y} y_i + \mathbf{W^h}\mathbf{h}_{i-1} + \mathbf{b^h}) \tag{1}$$

where $\mathbf{W^t}$, $\mathbf{W^y}$, $\mathbf{W^h}$, $\mathbf{b^h}$ denote weights for the time and mark at i^{th} event, weights and bias for the history embedding respectively; $\mathbf{h}_{i-1}$ is the history embedding at $i-1^{th}$ event, and ψ is an activation function. The conditional intensity function at the i^{th} event can be modeled as exponential intensity, $\lambda^*(t_i) = \exp(w(t_i - t_{i-1}) + \mathbf{v}^T\mathbf{h}_{i-1} + b)$ where w, $\mathbf{v}$ and b are weights and bias. The label probabilities are modeled independently from time and obtained via softmax:

$$p^*(y_i = m) = p(y_i = m|\mathbf{h}_{i-1}) = \frac{\exp(\mathbf{V}_m^y \mathbf{h}_{i-1} + b_m^y)}{\sum_{m=1}^{M} \exp(\mathbf{V}_m^y \mathbf{h}_{i-1} + b_m^y)} \tag{2}$$

where $\mathbf{V}^y$ and $\mathbf{b}^y$ are label weights and bias, and subscript m represents m^{th} row of $\mathbf{V}^y$ and m^{th} entry of $\mathbf{b}^y$. The neural Hawkes process extends the classical Hawkes process with neural networks so that it is self-modulating: past events can not only excite but also inhibit future events (Mei and Eisner 2016). Instead of modeling the instantaneous conditional intensity function, FullyNN directly models the cumulative intensity through a feed-forward network (Omi, Ueda, and Aihara 2019).

Shchur, Biloš, and Günnemann (2019) propose intensity-free modeling of TPPs. This approach allows characterizing TPPs with inter-event times $\tau_i = t_i - t_{i-1} \in \mathbb{R}^+$. A mixture of log-normal distributions is used to capture the conditional density of τ_i. The history dependence of τ can be modeled through a neural density network (Bishop 1994; Rezende and Mohamed 2015), in which the parameters depend on the history embedding.

Transformers for Event Streams

Attention and transformer models have been used to model event data in recent years (Xiao et al. 2019; Zhang et al. 2020a; Zuo et al. 2020; Gu 2021). The self-attention mechanism, in this context, relates different event instances of a single sequence in order to compute a representation of the sequence. The architecture of transformers for TPPs consists of an embedding layer and a self-attention layer. In Transformer Hawkes Processes (THP) (Zuo et al. 2020), for example, temporal embedding for t_i is through

$$[z(t_i)]_c = \begin{cases} \cos(t_i/10000^{\frac{c-1}{d}}) & \text{if } c \text{ is odd} \\ \sin(t_i/10000^{\frac{c}{d}}) & \text{if } c \text{ is even} \end{cases} \tag{3}$$

where d is the dimension of encoding, and subscript c denotes c^{th} dimension. Time embedding and one-hot encoded types are combined to form the embedded input $\mathbf{X}$ to be fed into the attention module. The dot-product attention is computed as:

$$\mathbf{S} = \mathbf{A}_s\mathbf{V} = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{M_k}})\mathbf{V} \tag{4}$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ are query, key and value matrix; they are linear transformations of $\mathbf{X}$. $\mathbf{A}_s$ is the attention score matrix. The output $\mathbf{S}$ is then fed into a pointwise feed forward neural network (FFN) to learn a high level representation of the sequence for modeling the conditional intensity function.

Multi-Label Prediction

Multi-label classification is an extensively studied machine learning task (Tsoumakas and Katakis 2007). Many successful approaches have been proposed in the setting of tabular data (Li et al. 2016; Yang et al. 2019) and also for set prediction (Zhang, Hare, and Prugel-Bennett 2019; Xie et al. 2017) particularly in image classification. A recent study considers the use of LSTM for multi-label classification in time-series for fault detection (Zhang et al. 2020b). Enguehard et al. (2020) apply encoder-decoder models for multi-label prediction in TPPs without considering potential correlation among concurrently occurring event labels – we address this important aspect here in our proposed approach.

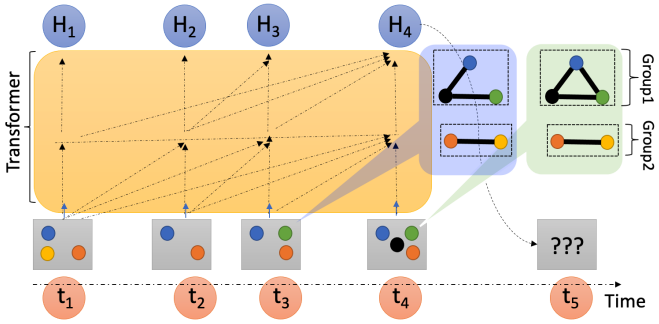


Figure 1: Concurrent multi-label prediction in event streams with a transformer architecture. The history embedding $\mathbf{H}_i$'s are used as features for next event label prediction. Two groups of labels for each epoch are observed – an example is crytocurrency transaction where group1 consists different actions, and group2 coin types. The structure of the 2 groups may change over time.

3 Task and Model Description

Concurrent Multi-Label Prediction

We bridge point process models with multi-label classification and formalize concurrent multi-label prediction in event streams. Consider a concurrent multi-label event sequence dataset where each sequence $\mathcal{E}_l$ consists of event epochs $(t_i, \mathbf{y}_i)_{i=1}^{n_l}$ where t_i is a timestamp and $\mathbf{y}_i \in \{0,1\}^M$ is a binary M-dimensional vector. We formally define the problem as:

Definition 1 *Given a set of label candidates Y = {1, 2, ..., M} and a time-stamped event dataset with events of the form $(t_i, \mathbf{y}_i)$, the multi-label prediction task in event streams aims to map any prior history $h_j = (t_i, \mathbf{y}_i)_{i=1}^{j}$ to a subset of the label set as its next event labels $\mathbf{y}_{j+1}$ for $j = \{1, 2, 3, ..., n_l - 1\}$.*

An illustrative example is shown in Figure 1. While state-of-the-art TPP models exist for multi-class prediction and can be as modeled via Eq. 2, a simple augmentation of labels will result in an exponential number of classes. Furthermore, limited label subset instances will impair model learnability. To capture both *inter*-epoch label interaction and *intra*-epoch label dependence, we propose a novel **T**ransformer-based **C**onditional **M**ixture **B**ernoulli **N**etwork (TCMBN) model for concurrent multi-label prediction in event streams. Our approach embeds event epochs as temporal encodings (Zuo et al. 2020) through Eq. 3. However, we allow *multi-hot* encoded labels combined with its temporal encoding to form an embedded input $\mathbf{X} = \mathbf{UY} + \mathbf{Z}$ where $\mathbf{Z} \in \mathbb{R}^{d \times n_l}$, $\mathbf{U} \in \mathbb{R}^{d \times M}$ is a trainable weight matrix and $\mathbf{Y} \in \{0,1\}^{n_l \times M}$ is a multi-hot encoding with max sequence length n_l. The inter label-label attention among epochs are captured by the dot-product attention via Eq. 4. The output after B blocks of attention layers and FFN with residual connection for event epoch i which we denote as $\mathbf{H}_i$ can be considered as a high level representation of the past up to t_i. The expressiveness of transformers for sequence modeling with position encoding is thoroughly examined by prior work (Yun et al. 2019). With temporal encoding, the following holds:

Theorem 1 *Transformers with temporal encodings are universal approximators for any continuous sequence-to-sequence function with compact domain, i.e. they approximate any continuous functions f: $\mathbf{X} \to \mathbf{H}$ with ϵ error w.r.t. p-norm where $1 \leq p < \infty$ and $\mathbf{X}, \mathbf{H} \in \mathbb{R}^{d \times n_l}$.*

The universal approximation property of temporally encoded transformer for event sequence-to-sequence is crucial in continuous time, which differentiates significantly from its time series (or sequences in natural language process) counterpart. Learned history features can be used for any downstream tasks. Given $\mathbf{H}_i$ for each event epoch, our problem evolves into a static multi-label prediction of next epoch.

Remark. Given a set of label candidates Y = {1, 2, ..., M} and data points with history-embedded features $(\mathbf{H}_i, \mathbf{y}_{i+1})$, TCMBN aims to learn a classifier that maps each historical feature $\mathbf{H}_i$ to a subset of the label set for prediction.

In order to distinguish the history between different input epochs at t_i and t_j with their respective embedding $\mathbf{X}_i$ and $\mathbf{X}_j$, we show there exists a transformer that separates the two, which gives feasibility guarantee for our multi-label classification problem; otherwise, any classifier will fail to distinguish two labels with identical features. The following is a consequence of Theorem 1.

Theorem 2 *There exists a transformer g with temporal encoding that separates two points $(\mathbf{X}_i, \mathbf{H}_i)$ and $(\mathbf{X}_j, \mathbf{H}_j)$, i.e. $g(\mathbf{X})_i = \mathbf{H}_i \neq g(\mathbf{X})_j = \mathbf{H}_j$ for some $\mathbf{X}_i \neq \mathbf{X}_j$ where $\mathbf{X}_i, \mathbf{X}_j, \mathbf{H}_i, \mathbf{H}_j \in \mathbb{R}^d$.*

While many multi-label neural classifiers are available (Read et al. 2021; Liu et al. 2021), we propose the marriage of neural density estimation (Bishop 1994; Rezende and Mohamed 2015) and a conditional Bernoulli mixture model (Li et al. 2016) for multi-label classification given a history embedding, due to the flexibility of neural mixture models and non-diagonal covariance of a multivariate Bernoulli mixture with K components ($K > 1$) which naturally encodes label correlation. The responsibility network π and mean network μ which make up the conditional mixture of Bernoulli network (CMBN) have the following structure:

$$\pi = \text{softmax}(MLP_\pi(\mathbf{H}_i)) \tag{5}$$

$$\mu_{m,k} = \text{sigmoid} \odot (MLP_\mu(\mathbf{H}_i)) \tag{6}$$

where MLP specifies multi-layer perceptron and $\odot$ signifies component wise operation for all K components, each with dimension M. In particular, for 2-layer MLP, we use ELU activation (Clevert, Unterthiner, and Hochreiter 2015):

$$MLP := \mathbf{W}_2(ELU(\mathbf{W}_1\mathbf{H}_i) + \mathbf{b_1}) + \mathbf{b_2}) \tag{7}$$

We use a separate set of weights and biases $\mathbf{W}_1, \mathbf{W}_2, \mathbf{b_1}, \mathbf{b_2}$ for π and μ network respectively for flexible learning. We emphasize that π, μ are functions of history embedded features and can be used to compute covariance matrix in the sections to follow. Thus, with a focus on next event label(s) prediction (time prediction can be tackled with either mixture of log-normals (Shchur, Biloš, and Günnemann 2019)

or RMTPP (Du et al. 2016)), our generative approach for concurrent multi-label prediction aims to optimize:

$$p(\mathbf{y}_{i+1}|\mathbf{H}_i) = \sum_{k=1}^{K} \pi_k BER(\mathbf{y}_{i+1}; \mu_k) \tag{8}$$

where $BER(\mathbf{y}; \mu_{\mathbf{k}}) = \prod_{l=1}^{M} Ber(y_m; \mu_{m,k})$. BER signifies the multivariate Bernoulli distribution and Ber the univariate Bernoulli parametrized by the $\mu_{m,k}$ network. Alternatively, our approach can be viewed as elegantly solving a neural network parameterized linear systems of equations:

$$\mu(\mathbf{H}_i)\pi(\mathbf{H}_i) = \mathbf{y}_{i+1} \tag{9}$$

Background Knowledge Injection

We enable potentially incorporating background knowledge into the concurrent multi-label prediction problem. Our approach differs from prior use of a semantic loss function for multi-class prediction where only 1 class is predicted (Xu et al. 2018). In the multi-label classification setting where each class of events is not mutually exclusive, we consider a practical case involving the existence of exhaustive groups which comprise all class labels such that within each group, only q events can happen. For example, $q = 1$ indicates mutually exclusive classes within each group. Such a partition of groups is not uncommon in disease management and trip itineraries where diseases are categorized and flights have certain hubs and routes. In crypto-transactions, exactly one action in the transaction group will take place and simultaneously one type of coin will be traded at each epoch. Let $\hat{p}_{c_j}$ denote the predicted probability for c^{th} class in j^{th} group (with a total of o_j classes and G groups). We propose the following background knowledge loss for an instance of prediction:

$$L_{BK} = \sum_{g=j}^{G} (\sum_{c_j=1}^{o_j} \hat{p}_{c_j} - q)^2 \tag{10}$$

The least square loss can be considered a soft regularization of allocation of probabilities. Such loss will promote our model to find the mode of the joint distribution $p(\mathbf{y}_{i+1}|\mathbf{H}_i)$. TCMBN seeks to minimize the following objective given N sequences of $\mathcal{E}_l = \{(t_i, \mathbf{y}_i)\}_{i=1}^{n_l}$ for $l = \{1, 2, ..., N\}$:

$$\begin{aligned} \mathcal{L}_{LL} + \lambda \mathcal{L}_{BK} = &- \sum_{l}^{N} \sum_{i}^{n_l} (\log p^*(\mathbf{y}_{i+1})) \\ &+ \lambda \sum_{l}^{N} \sum_{i}^{n_l} \sum_{g=j}^{G} (\sum_{c_j=1}^{o_j} \hat{p}_{c_j, i+1, l} - q)^2 \end{aligned} \tag{11}$$

where λ trades off between negative log-likelihood loss and background knowledge loss.

Capturing Structure among Concurrent Labels

We propose a novel approach to learn the label structure of an undirected graph $G(i) = (V, E_i)$ at each epoch t_i, where the vertices V are the labels, and the edges E_i encodes label dependencies which may change over time by numerically approximating the precision matrix of the multivariate Bernoulli distribution (Banerjee, El Ghaoui, and d'Aspremont 2008; Bai et al. 2019; Ravikumar, Wainwright, and Lafferty 2010). Since we learn the covariance in closed-form at each epoch from TCMBN, we can compute its inverse numerically. A more attractive feature is to learn a sparse precision matrix as in the multivariate Gaussian case (Friedman, Hastie, and Tibshirani 2008). In our setting, jointly approximating the precision matrix with a sparsity constraint while performing concurrent multi-label classification is challenging because of the symmetric positive definiteness of the matrix. Hence we propose a two-step procedure analogous to learning Gaussian graphical models (Giraud 2015):

Step 1: Solve minimization problem: minimize log-likelihood term for event streams and obtain learned covariance matrices as in the following from TCMBN.

$$\text{Cov}(\mathbf{y}_{i+1}) = \sum_{k=1}^{K} \pi_k [\Sigma_k + \mu_{:,k}(\mu_{:,k})^T] - \mathbb{E}(\mathbf{y}_{i+1})\mathbb{E}(\mathbf{y}_{i+1})^T \tag{12}$$

where $\mathbb{E}(\mathbf{y}) = \sum_{k=1}^{K} \pi_k \mu_k$ is the weighted mean of all the component means. $\Sigma_k = \text{diag}(\mu_{m,k}(1 - \mu_{m,k}))$ for $k \in K$ is the variance of a univariate Bernoulli with mean $\mu_{m,k}$. Note that all terms above depend on $\mathbf{H_i}$; we dropped such a dependency to avoid notational cluttering.

Step 2: For each learned covariance matrix $\hat{\mathbf{C}}$: solve the following least-squares sparse approximation (LSSA) to obtain the estimated precision matrix:

$$\begin{aligned} \min_{\mathbf{P}} \quad & \frac{1}{2} ||\hat{\mathbf{C}} * \mathbf{P} - \mathbf{I}||_F^2 + \gamma ||\mathbf{F} \circ \mathbf{P}||_1 \\ & \text{s.t.} \quad \mathbf{P} \in \mathbf{S}_{++}^n. \end{aligned} \tag{13}$$

where $\mathbf{P}$ is the precision matrix to optimize, $\mathbf{I}$ the identity matrix, $\mathbf{1}$ matrix of ones, $\mathbf{F} = \mathbf{1} - \mathbf{I}$, γ the shrinkage parameter, F the Frobenius norm, $\mathbf{S}_{++}^n$ the set of symmetric positive definite matrices, and $\circ$ Hadamard product of two matrices. Note that LSSA is a convex optimization problem and can be solved by efficient solvers such as cvxopt (Andersen et al. 2013). We emphasize that our approach TCMBN-LSSA is completely data-driven and capable of capturing the nonstationarity of labels (Trivedi et al. 2019).

Table 1: Properties of four benchmark event datasets.

Dataset	# Classes	# Seqs.	Avg. Len.	Data type
Synthea	232	2500	43	Simulated EHR
Dunnhumby	24	2500	93	Real Transaction
MIMIC III	169	6644	3	Real EHR
Defi	39	5000	27	Real Finance

4 Experiments

Model Setting, Baselines & Evaluation Metrics

We implement and train our model with Pytorch and report results using 64 Bernoulli mixture components for all experiments. Hyper-parameter λ is chosen from

$\{0.1, 0.01, 0.001\}$ and is only used if domain knowledge is injected; otherwise it is set to 0. Further details and codes are included in Appendix A in supplementary material. We compare our model with encoder-decoder based TPP models (Enguehard et al. 2020) including RMTPP (Du et al. 2016), intensity-free (Shchur, Biloš, and Günnemann 2019) and attention-based models for comparison (Xiao et al. 2019). For the baselines, best performing results are selected from running a few default settings according to the package provided by the authors[1]. Other potential baselines are not suitable for the concurrent multi-label setting (Zhang et al. 2020a; Gu 2021). We run a transformer model with a logistic regression for each class (dubbed Transformer-LG) by adapting from transformer hawkes process (Zuo et al. 2020) and an RNN model with CMBN (dubbed RNN-CMBN) for further comparison. We evaluate the performance of different models on the task of multi-label prediction on the test data using several evaluation metrics. Specifically, we consider the weighted ROC-AUC score (Biloš, Charpentier, and Günnemann 2019; Enguehard et al. 2020) as well as the weighted F1 score and the Hamming score (Sorower 2010).

Synthetic Concurrent Multi-Label Events

We generate 2 synthetic datasets: Poisson-MBN (PM) and Hawkes-MBN (HM) with $M = 5$ labels, where timestamps are generated using a Poisson and exponential kernel Hawkes process respectively. For each timestamp, we partition the 5 labels into 2 groups – one with 3 classes and the other with 2. For each class in each group, we count the number of historical occurrences of the class and normalize by their total sum: this is the probability for each class to generate labels for the next event epoch. Thus, our approach for generation naturally induces a long history dependence and interaction among labels in every epoch (both pair-wise and 3^{rd} order interaction). We generate 5 simulations, each of which consists of a total of 1000 sequences and randomly split 60-20-20 training-dev-test subsets. Further details regarding data generation are supplied in Appendix B.

Results. Table 2 compares 7 different models on PM and HM. Our model achieves the best performance over all baselines using all three metrics. Note that the encoder-decoder models do not incorporate potential correlation among labels at each timestamp. A close competitor is RNN-CMBN – while it is able to capture the correlation, it may have missed the complex history dependence. Table 3 summarizes the performance of the models for each class. TCMBN consistently achieves much better ROC-AUC scores for all classes in both experiments compared to baselines. Interestingly, all models predict better when the label interaction is pairwise in Group 1 (classes 1 and 2) than that of the third order interaction in Group 2 (classes 3, 4 and 5).

Real-World Concurrent Multi-Label Events

We consider the following 4 event datasets for the task of multi-label prediction. Their properties are summarized in Table 1. We incorporate domain knowledge only on the Defi dataset where each event label consists of only 1 coin type and 1 transaction type.

Synthea. This is a simulated EHR dataset that *closely* mimics real EHR data (Walonoski et al. 2018). Each module generates patient populations with events that can occur in a medical history of a synthetic patient.[2]

Dunnhumby. We extract this dataset from Kaggle's Dunnhumby - The Complete Journey dataset.[3] We use the $transaction$ file with household level transactions over two years from a group of 2,500 households frequently shopping at a retailer. To roll up the item types, each item is mapped to its department category based on the $product$ file.

MIMIC III. The MIMIC III database provides patient-level de-identified health-related data associated with the Israel Deaconess Medical Center between 2001 and 2012 (Johnson, Pollard, and Mark 2016; Johnson et al. 2016; Goldberger et al. 2000). We extract hospital admission records for each patient to include admission time and ICD-9 codes which were mapped to CCS codes as labels.

Defi. This dataset provides user-level cryptocurrency trading history under a specific protocol called Aave.[4] The data includes timestamp, transaction type and coin type for each transaction. Coupled marks that concatenate the transaction and coin type are used as labels for each event.

Results. Table 4 summarizes the performances of 7 different models on the benchmarks. TCMBN achieves overall superior performance compared to all baselines. In particular, it achieves the best results on 6 out of 12 evaluations, and close to best performance on the other 6 evaluations; none of the other models compare in terms of consistent robust performance on this task. A suggested strong baseline – GRU-CP (Enguehard et al. 2020) – achieves noticeably inferior results than our model.

Ablation I: Mixture Components

Figure 2 shows the effects of varying mixture components. Overall, we did not observe dramatic changes in predictive performance with varying the number of mixture components. The most changes comes with respect to Hamming loss, while all models perform roughly the same with respect to weighted ROC-AUC and F1 scores. TCMBN with 32 mixture components appears to perform best, while our choice of 64 components comes after. The lack of noticeable variation in performance for neural mixtures of Bernoulli models is different from the classical E-M type of mixture of experts where performance is usually boosted with more components, as observed in tabular multi-label classification task (Li et al. 2016). The optimization mechanism underlies this difference: the likelihood in the classical E-M approach is bound to increase with more mixture components, while stochastic gradient descent in our neural models does not guarantee this increase. Nevertheless, even with a relatively small number of mixture components, our approach shows competitive predictive strength.

[1] https://github.com/babylonhealth/neuralTPPs

[2] https://github.com/synthetichealth/synthea

[3] https://www.kaggle.com/frtgnn/dunnhumby-the-complete-journey

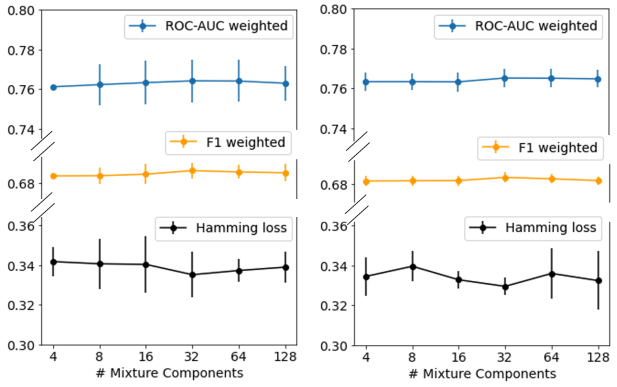
[4] aave.com

Table 2: Overall predictive performances on PM & HM datasets as measured by weighted ROC-AUC, weighted F1 and Hamming loss. Mean values are shown along with standard deviation in parentheses. Best results are in bold.

Data	Metric/Model	GRU-CP	GRU-RMTPP	GRU-LNM	GRU-ATTN	Transformer-LG	RNN-CMBN	TCMBN (ours)
PM	ROC-AUC	0.751(0.012)	0.733(0.026)	0.724(0.028)	0.669(0.038)	0.568(0.013)	0.752(0.009)	**0.764(0.011)**
	F1	0.676(0.005)	0.651(0.015)	0.662(0.012)	0.639(0.009)	0.620(0.005)	0.677(0.007)	**0.686(0.004)**
	Hamming	0.345(0.008)	0.416(0.039)	0.409(0.055)	0.467(0.058)	0.561(0.006)	0.342(0.008)	**0.337(0.006)**
HM	ROC-AUC	0.754(0.006)	0.751(0.011)	0.755(0.005)	0.650(0.019)	0.585(0.003)	0.755(0.011)	**0.765(0.005)**
	F1	0.675(0.004)	0.665(0.016)	0.675(0.004)	0.640(0.009)	0.619(0.001)	0.679(0.006)	**0.683(0.002)**
	Hamming	0.345(0.013)	0.378(0.041)	0.338(0.006)	0.506(0.011)	0.563(0.002)	0.342(0.006)	**0.336(0.013)**

Table 3: ROC-AUC for each class on PM & HM. Best results are in bold. G:Group C:Class

Data	Metric/Model	GRU-CP	GRU-RMTPP	GRU-LNM	GRU-ATTN	Transformer-LG	RNN-CMBN	TCMBN (ours)
PM	G1C1	0.771(0.016)	0.774(0.018)	0.780(0.016)	0.769(0.019)	0.580(0.024)	0.775(0.016)	**0.783(0.018)**
	G1C2	0.778(0.023)	0.780(0.022)	0.786(0.020)	0.777(0.017)	0.581(0.020)	0.778(0.013)	**0.791(0.019)**
	G2C1	0.728(0.019)	0.691(0.058)	0.636(0.086)	0.584(0.082)	0.560(0.009)	0.733(0.011)	**0.738(0.015)**
	G2C2	0.732(0.012)	0.678(0.054)	0.682(0.064)	0.574(0.064)	0.554(0.016)	0.728(0.013)	**0.747(0.020)**
	G2C3	0.724(0.012)	0.704(0.017)	0.685(0.057)	0.552(0.081)	0.551(0.016)	0.722(0.010)	**0.740(0.009)**
HM	G1C1	0.786(0.014)	0.790(0.010)	0.791(0.011)	0.793(0.015)	0.602(0.011)	0.783(0.017)	**0.799(0.011)**
	G1C2	0.781(0.014)	0.784(0.018)	0.786(0.016)	0.781(0.017)	0.598(0.010)	0.780(0.017)	**0.790(0.016)**
	G2C1	0.722(0.014)	0.714(0.015)	0.720(0.016)	0.520(0.016)	0.571(0.013)	0.730(0.008)	**0.735(0.010)**
	G2C2	0.729(0.013)	0.713(0.023)	0.729(0.012)	0.531(0.031)	0.570(0.015)	0.730(0.013)	**0.741(0.013)**
	G2C3	0.725(0.009)	0.719(0.011)	0.720(0.012)	0.524(0.021)	0.569(0.009)	0.729(0.011)	**0.734(0.010)**



(a) PM (b) HM

Figure 2: TCMBN results with varying mixture components.

Ablation II: Background Knowledge Constraint

We train TCMBN without background knowledge loss on the synthetic datasets and Defi. A comparison of two cases in Table 5 shows that our prediction without the additional loss consistently gets worse in terms of F1-weighted score, while fluctuating with respect to Hamming loss. Such discrepancy is likely due to class imbalance in our setting.

Structure Discovery for Synthetic Binary I.I.D. Data (without Timestamps)

We further test our two-step CMBN-LSSA approach for learning label dependency from i.i.d. data by generating binary data with a dimension of 5 and 8 variables based on the Ising model: $Prob(x_1, x_2, ..., x_M) = \exp(\Theta_0 + \sum \Theta_i x_i + \sum_{i<j} \Theta_{ij} x_i x_j)$ according to a procedure described elsewhere (Ravikumar, Wainwright, and Lafferty 2010). The second order interaction coefficient Θ_{ij} indicates the presence of an edge between node i and j if $\Theta_{ij} \neq 0$. We compare our approach with a few representative models: Logistic-Neighborhood-Max (L-N-M) model (Ravikumar, Wainwright, and Lafferty 2010), current state-of-the-art NeurISE (Lokhov et al. 2020) and Mobius Inversion (M-I) (De Canditiis 2019). A detailed description on training these models is in Appendix C. As shown in Table 6, our approach is most effective at recovering structures and outperforms all baselines for both dimensions with small sample sizes. It is worth noting that our model is more efficient to run compared to NeurISE because the latter requires looping through all dimensions (i.e. coordinate-wise training) during training while ours does not. Thus, our proposed two-step procedure is also a suitable tool for discovering undirected structures even for applications involving i.i.d. data.

Dunnhumby: Case Study of Structure Discovery for Concurrent Multi-Label Events

Figure 3 shows two discovered structures, one at an earlier time ($i = 1^{st}$) and the other at a later time ($i = 101^{st}$), from Dunnhumby. Each nonzero entry in the learned precision matrix is shown as an edge on the graph. A partial correlation score for two nodes V_i and V_j given the other nodes is shown as an edge weight and calculated as below: $\rho_{V_i,V_j|\mathbf{V}\setminus\{V_i,V_j\}} = -\frac{\mathbf{P}_{ij}}{\sqrt{\mathbf{P}_{ii}\mathbf{P}_{jj}}}$ where $\mathbf{P}_{ij}$ denotes the

Table 4: Overall predictive performance on 4 real applications. Best results are in bold.

Data	Metric/Model	GRU-CP	GRU-RMTPP	GRU-LNM	GRU-ATTN	Transformer-LG	RNN-CMBN	TCMBN (ours)
Synthea	ROC-AUC	0.841	0.805	0.834	0.707	0.619	0.679	**0.852**
	F1	0.419	0.367	0.408	0.293	0.227	0.212	**0.456**
	Hamming	0.014	**0.012**	0.015	0.022	**0.012**	0.042	0.014
Dunnhumby	ROC-AUC	0.654	0.620	0.636	0.620	0.552	**0.697**	0.691
	F1	0.592	0.558	0.568	0.563	0.555	**0.599**	0.598
	Hamming	0.127	0.619	0.155	0.162	0.881	0.123	**0.122**
MIMIC III	ROC-AUC	0.679	0.677	0.678	0.577	0.566	0.635	**0.752**
	F1	0.288	0.273	0.279	0.241	0.240	0.270	**0.354**
	Hamming	0.078	**0.051**	0.090	0.135	0.384	0.080	0.058
Defi	ROC-AUC	**0.776**	0.767	0.734	0.655	0.585	0.502	0.771
	F1	0.462	0.449	0.417	0.363	0.324	0.276	**0.469**
	Hamming	0.071	**0.069**	0.072	0.105	0.411	0.469	0.073

Table 5: Results on synthetic datasets without background knowledge constraint.

Data/Metric	ROC-AUC	F1	Hamming
PM	0.764(0.010)	0.684(0.003) $\downarrow$	0.344(0.009) $\uparrow$
HM	0.765(0.004)	0.682(0.003) $\downarrow$	0.332(0.010) $\downarrow$
Defi	0.767 $\downarrow$	0.464 $\downarrow$	0.071 $\downarrow$

Table 6: F1 score evaluation on structure recovery on synthetic binary i.i.d. data with 5 and 8 dimensions. Mean F1 score and standard deviation (in parentheses) are shown. Best results for each dataset are in bold.

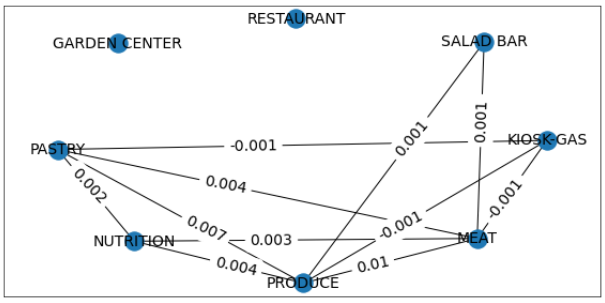
Nodes	Samples	M-I	L-N-M	NeurISE	CMBN-LSSA
5	150	0.72(0.10)	0.29(0.24)	0.73 (0.07)	**0.80(0.00)**
	300	0.72(0.16)	0.67(0.14)	0.75 (0.07)	**0.82(0.05)**
	500	0.80(0.06)	0.82(0.14)	0.83(0.06)	**0.86(0.00)**
8	150	0.65(0.12)	0.39(0.26)	0.71 (0.03)	**0.73(0.02)**
	300	0.66(0.04)	0.65(0.17)	0.75 (0.05)	**0.76(0.02)**
	500	0.67(0.06)	0.79(0.07)	**0.79 (0.03)**	0.76(0.01)

(i, j) entry of the matrix. A distinctive pattern from both graphs is restaurant and garden center visits are independent of the other shopping behavior. Another identified pattern is that produce and meat are most strongly linked together since both are associated with grocery shopping. The two structures show slightly differing shopping behaviors of the households, possibly due to seasonality. For example, at the earlier epoch, getting gas tends to be negatively partially correlated with purchasing produce, meat and pastries; later, it negatively links to nutritional products and salad bar as well. In practice, learning dynamic graph structures of items could provide valuable insights for better goods allocation, strategic planning of promotions, and etc.

5 Conclusion

We have proposed an effective interpretable approach for concurrent multi-label learning in event streams. To the best of our knowledge, we are the first to systematically model

Learned Graph at $i = 1^{st}$ event.



Learned Graph at $i = 101^{st}$ event.

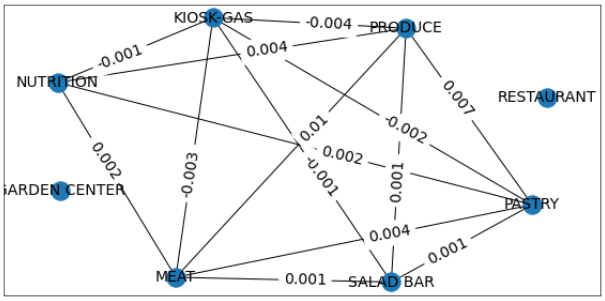


Figure 3: Learned undirected structure on Dunnhumby. For clarity, we only show 8 of the 24 departments as labels.

multi-label prediction and undirected structure learning in continuous time. We learn a neural mixture density that can potentially also incorporate any meaningful domain knowledge ; our approach shows promising results on experiments across domains. We note that our model is flexible enough to embed additional features for each event epoch, if and when available. Our work can be extended in a few directions. For example, while (T)CMBN-LSSA is a powerful two-step procedure, one could potentially develop end-to-end joint learning of event sequences and structure among labels. We have shown that our framework can provide beneficial and actionable insights for many data-driven applications involving event streams.

References

Andersen, M. S.; Dahl, J.; Vandenberghe, L.; et al. 2013. CVXOPT: A Python package for convex optimization. *Available at cvxopt. org*, 54.

Bacry, E.; Mastromatteo, I.; and Muzy, J.-F. 2015. Hawkes Processes in Finance. *Market Microstructure and Liquidity*, 1(01): 1550005.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*.

Bai, T.; Egleston, B. L.; Bleicher, R.; and Vucetic, S. 2019. Medical concept representation learning from multi-source data. In *Proc. of the Twenty-Eighth Int. Joint Conference on Artificial Intell.*, 4897–4903.

Banerjee, O.; El Ghaoui, L.; and d'Aspremont, A. 2008. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *The Journal of Machine Learning Research*, 9: 485–516.

Bhattacharjya, D.; Subramanian, D.; and Gao, T. 2018. Proximal graphical event models. *Advances in Neural Information Processing Systems*, 31.

Biloš, M.; Charpentier, B.; and Günnemann, S. 2019. Uncertainty on Asynchronous Time Event Prediction. *arXiv preprint arXiv:1911.05503*.

Bishop, C. M. 1994. Mixture Density Networks. Technical report. Aston University.

Boyd, A.; Bamler, R.; Mandt, S.; and Smyth, P. 2020. User-Dependent Neural Sequence Models for Continuous-Time Event Data. *arXiv preprint arXiv:2011.03231*.

Chatterjee, S.; and Mudher, A. 2018. Alzheimer's disease and type 2 diabetes: a critical assessment of the shared pathological traits. *Frontiers in neuroscience*, 12: 383.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Daley, D. J.; and Jones, D. V. 2003. *An Introduction to the Theory of Point Processes: Elementary Theory of Point Processes*. Springer.

De Canditiis, D. 2019. Learning Binary Undirected Graph in Low Dimensional Regime. *arXiv preprint arXiv:1907.11033*.

Didelez, V. 2008. Graphical models for marked point processes based on local independence. *Journal of Royal Statistical Society, Ser. B*, 70(1): 245–264.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. In *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 1555–1564.

Eichler, M. 1999. *Graphical Models in Time Series Analysis*. Ph.D. thesis, University of Heidelberg, Germany.

Enguehard, J.; Busbridge, D.; Bozson, A.; Woodcock, C.; and Hammerla, N. 2020. Neural Temporal Point Processes For Modelling Electronic Health Records. In *Machine Learning for Health*, 85–113. PMLR.

Friedman, J.; Hastie, T.; and Tibshirani, R. 2008. Sparse Inverse Covariance Estimation with the Graphical Lasso. *Biostatistics*, 9(3): 432–441.

Gao, T.; Subramanian, D.; Shanmugam, K.; Bhattacharjya, D.; and Mattei, N. 2020. A Multi-Channel Neural Graphical Event Model with Negative Evidence. In *Proc. of AAAI Conference on Artificial Intelligence*, volume 34, 3946–3953.

Giraud, C. 2015. Introduction to High-dimensional Statistics. *Monographs on Statistics and Applied Probability*, 139: 139.

Goldberger, A. L.; Amaral, L. A.; Glass, L.; Hausdorff, J. M.; Ivanov, P. C.; Mark, R. G.; Mietus, J. E.; Moody, G. B.; Peng, C.-K.; and Stanley, H. E. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *circulation*, 101(23): e215–e220.

Gu, Y. 2021. Attentive Neural Point Processes for Event Forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 7592–7600.

Hawkes, A. G. 1971. Point Spectra of some Mutually Exciting Point Processes. *Journal of the Royal Statistical Society: Series B (Methodological)*, 33(3): 438–443.

Johnson, A.; Pollard, T.; and Mark, R. 2016. MIMIC-III Clinical Database (version 1.4). PhysioNet.

Johnson, A. E.; Pollard, T. J.; Shen, L.; Li-Wei, H. L.; Feng, M.; Ghassemi, M.; Moody, B.; Szolovits, P.; Celi, L. A.; and Mark, R. G. 2016. MIMIC-III, a Freely Accessible Critical Care Database. *Scientific Data*, 3(1): 1–9.

Li, C.; Wang, B.; Pavlu, V.; and Aslam, J. 2016. Conditional Bernoulli Mixtures for Multi-label Classification. In *Int. Conf. on Machine Learning*, 2482–2491. PMLR.

Liu, S.; Zhang, L.; Yang, X.; Su, H.; and Zhu, J. 2021. Query2label: A simple transformer way to multi-label classification. *arXiv preprint arXiv:2107.10834*.

Lokhov, A. Y.; Misra, S.; Vuffray, M.; et al. 2020. Learning of Discrete Graphical Models with Neural Networks. *arXiv preprint arXiv:2006.11937*.

Luo, D.; Xu, H.; Zhen, Y.; Ning, X.; Zha, H.; Yang, X.; and Zhang, W. 2015. Multi-task Multi-dimensional Hawkes Processes for Modeling Event Sequences. In *Proc. of the Twenty-Fourth Int. Joint Conf. on Artificial Intelligence*.

Mei, H.; and Eisner, J. 2016. The Neural Hawkes Process: A Neurally Self-modulating Multivariate Point Process. *arXiv preprint arXiv:1612.09328*.

Omi, T.; Ueda, N.; and Aihara, K. 2019. Fully Neural Network Based Model for General Temporal Point Processes. *arXiv preprint arXiv:1905.09690*.

Ravikumar, P.; Wainwright, M. J.; and Lafferty, J. D. 2010. High-dimensional Ising Model Selection using L1-regularized Logistic Regression. *The Annals of Statistics*, 38(3): 1287–1319.

Read, J.; Pfahringer, B.; Holmes, G.; and Frank, E. 2021. Classifier chains: a review and perspectives. *Journal of Artificial Intelligence Research*, 70: 683–718.

Rezende, D.; and Mohamed, S. 2015. Variational inference with normalizing flows. In *Int. Conf. on Machine Learning*, 1530–1538. PMLR.

Shchur, O.; Biloš, M.; and Günnemann, S. 2019. Intensity-free Learning of Temporal Point Processes. *arXiv preprint arXiv:1909.12127*.

Shchur, O.; Gao, N.; Biloš, M.; and Günnemann, S. 2020. Fast and Flexible Temporal Point Processes with Triangular Maps. *arXiv preprint arXiv:2006.12631*.

Shchur, O.; Türkmen, A. C.; Januschowski, T.; and Günnemann, S. 2021. Neural Temporal Point Processes: A Review. *arXiv preprint arXiv:2104.03528*.

Sorower, M. S. 2010. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis*, 18: 1–25.

Trivedi, R.; Farajtabar, M.; Biswal, P.; and Zha, H. 2019. Dyrep: Learning representations over dynamic graphs. In *International conference on learning representations*.

Tsoumakas, G.; and Katakis, I. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3): 1–13.

Upadhyay, U.; De, A.; and Gomez-Rodriguez, M. 2018. Deep Reinforcement Learning of Marked Temporal Point Processes. *arXiv preprint arXiv:1805.09360*.

Walonoski, J.; Kramer, M.; Nichols, J.; Quina, A.; Moesel, C.; Hall, D.; Duffett, C.; Dube, K.; Gallagher, T.; and McLachlan, S. 2018. Synthea: An Approach, Method, and Software Mechanism for Generating Synthetic Patients and the Synthetic Electronic Health Care Record. *Journal of the American Medical Informatics Association*, 25(3): 230–238.

Xiao, S.; Yan, J.; Farajtabar, M.; Song, L.; Yang, X.; and Zha, H. 2019. Learning time series associated event sequences with recurrent point process networks. *IEEE transactions on neural networks and learning systems*, 30(10): 3124–3136.

Xiao, S.; Yan, J.; Yang, X.; Zha, H.; and Chu, S. M. 2017. Modeling the Intensity Function of Point Process Via Recurrent Neural Networks. In *Proc. of Conf. on Artificial Intelligence (AAAI)*, 1597–1603.

Xie, P.; Salakhutdinov, R.; Mou, L.; and Xing, E. P. 2017. Deep determinantal point process for large-scale multi-label classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 473–482.

Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and Broeck, G. 2018. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, 5502–5511. PMLR.

Yang, Y.-Y.; Lin, Y.-A.; Chu, H.-M.; and Lin, H.-T. 2019. Deep Learning with a Rethinking Structure for Multi-label Classification. In *Asian Conf. on Machine Learning*, 125–140. PMLR.

Yun, C.; Bhojanapalli, S.; Rawat, A. S.; Reddi, S. J.; and Kumar, S. 2019. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*.

Zhang, Q.; Lipani, A.; Kirnap, O.; and Yilmaz, E. 2020a. Self-attentive Hawkes Process. In *International Conference on Machine Learning*, 11183–11193. PMLR.

Zhang, W.; Jha, D. K.; Laftchiev, E.; and Nikovski, D. 2020b. Multi-label prediction in time series data using deep neural networks. *arXiv preprint arXiv:2001.10098*.

Zhang, Y.; Hare, J.; and Prugel-Bennett, A. 2019. Deep set prediction networks. *Advances in Neural Information Processing Systems*, 32.

Zhou, K.; Zha, H.; and Song, L. 2013. Learning Social Infectivity in Sparse Low-rank Networks using Multi-dimensional Hawkes Processes. In *Artificial Intelligence and Statistics*, 641–649. PMLR.

Zuo, S.; Jiang, H.; Li, Z.; Zhao, T.; and Zha, H. 2020. Transformer Hawkes Process. In *International Conference on Machine Learning*, 11692–11702. PMLR.

Appendix for Concurrent Multi-Label Prediction in Event Streams

A Model Implementation and Training Details

We implement the transformer module for TCMBN with partial adaption from the Transformer Hawkes Process Zuo et al. [2020] [1]. Training of TCMBN is performed by mini-batch stochastic gradient descent given training event sequences, validated by dev subset and test on the test subset. The ratio of the three sets is 60-20-20. Adam [Kingma and Ba, 2014] optimizer is used to train our model. The experiment setting for hyperparameters in TCMBN for multi-label prediction is given in Table 1. Baseline model Transformer-LG is performed with the same set of hyperparameters. Baseline model RNN-MBN is implemented partially from Intensity-free Point Process Shchur et al. [2019] yet with CMBN module of 64 components and is trained with default setting [2]. (While other configurations for the above baselines are possible, their empirical performance is inferior.) The best threshold $\lambda = \{0.1, 0.01, 0.001\}$ for each model is chosen based on dev set when domain knowledge loss is injected. We inject such loss in the modeling of synthetic data (PM and HM) and Defi. We set each q to be 1 for the above datasets. In addition, we set $\lambda = 0$ for the rest of datasets. All our experiments are performed on a private server with TITAN RTX GPU. We attach our code in the supplementary material package.

B Synthetic Sequence Generation

For synthetic data generation, we generate 1000 sequences according to the following procedure. The dynamics for a generated event sequence is driven by either a Poisson or a Hawkes Process with exponential kernel. We use a constant rate of 0.2 and time horizon of (0,100] to generate Poisson timestamps. We use baseline rate of 0.2, decay constant of 0.5, and adjacency matrix of [0.1] as described in the tick library of Hawkes model with exponential kernel.[3]

We partition 5 labels into two groups: one group of 2 and the group of 3. The un-normalized probability of co-occurence of labels for each event epoch depends only on the number of historical occurrences. Specifically, for each label, we check the counts of each specific label in each group in the history and obtain an un-normalized probability.

[1] https://github.com/SimiaoZuo/Transformer-Hawkes-Process

[2] https://github.com/shchur/ifl-tpp

[3] https://x-datainitiative.github.io/tick/modules/hawkes.html

Table 1: Hyperparameters for TCMBN. n_layers: the number of blocks for multi-headed self-attention module; d_model: the dimension of the value vector after attention has been applied; n_head: the number attention heads; d_inner: the dimension of the hidden layer of the feed forward neural network; d_v: the dimension of the value vector; d_k : the dimension of the key vector;num_comps : number of mixture of Bernoulli components.

Parameter	PM/HM	Synthea	Dunnhumby	Mimic3	Defi
batch_size	8	8	8	16	8
n_head	4	4	6	1	1
n_layers	1	1	1	1	1
d_model	64	64	32	64	32
d_inner	32	32	16	32	16
d_v	32	32	16	32	16
d_k	32	32	16	32	16
dropout	0.1	0.1	0.1	0.1	0.1
epoch	100	100	100	100	100
num_comps	64	64	64	64	64
learning_rate	0.0002	0.002	0.01	0.0015	0.006

We generate each event label according to the normalized probability distribution which serves as mean of multivariate Bernoulli distribution with diagonal covariance. Together we combine timestamps and labels to form our PM and HM datasets by performing 5 simulation, each of which consists of 1000 event sequences. We show more details in Algorithm 1 with pseudo code in Python for PM; HM can be obtained similarly.

C Models for Structural Learning

C.1 Synthetic Data

We describe the procedure to simulate the binary data. From the Ising model below, when the node dimension M = 5, 6 out of 10 parameters Θ_{ij} are randomly chosen with mixed coupling, i.e. $\Theta_{ij} = \pm 0.5$ with equal probability [Ravikumar et al., 2010].

$$P(x_1, x_2, ..., x_M) = \exp(\Theta_0 + \sum \Theta_i x_i + \sum_{i<j} \Theta_{ij} x_i x_j) \tag{1}$$

Then the probability mass function of categorical distribution (of 2^M configurations) were calculated and samples were generated accordingly [De Canditiis, 2019]. For M = 8, 17 out of 28 parameters are randomly chosen with mixed coupling. For large M, Gibbs sampling can be performed. The Matlab implementation is available online[4].

[4]https://www.iac.cnr.it/ danielad/software.html

Algorithm 1: Poisson Concurrent Event Generator

Input: Given label set: $\mathcal{L}$, $|\mathcal{L}| = 5$, A generated univariate Possion Process data $D_t = \{t_i\}_{i=1}^{n}$

Output: A generated multi-label sequence D_y

μ = [0.5,0.5,0.5,0.5,0.5]
μ_{new} = [0,0,0,0,0]
D_y = []
for $i \leftarrow 1$ **to** n **do**
    sample = diag_bernoulli_generator(μ)
    while *sum(sample) == 0* **do**
        sample = diag_bernoulli_generator(μ)
    end
    D_y.append(sample)
    counts = np.sum(np.array(D_y),axis=0)
    if *(counts[0]+counts[1]) != 0* **then**
        μ_{new}[0] = counts[0]/ (counts[0]+counts[1])
        μ_{new}[1] = counts[1]/ (counts[0]+counts[1])
    end
    if *(counts[2]+counts[3]+counts[4]) != 0* **then**
        μ_{new}[2] = counts[2]/ (counts[2]+counts[3]+counts[4])
        μ_{new}[3] = counts[3]/ (counts[2]+counts[3]+counts[4])
        μ_{new}[4] = counts[4]/ (counts[2]+counts[3]+counts[4])
    end
    for $k \leftarrow 1$ **to** $\mathcal{L}$ **do**
        if *μ_{new}[k] != 0* **then**
            μ[k] = μ_{new}[k]
        end
    end
end
Return: D_y

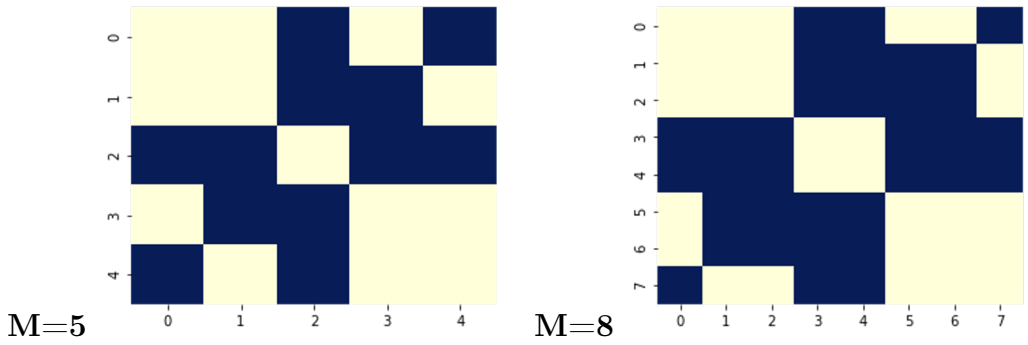


Figure 1: Ground truth structures of simulated data for 5 nodes and 8 nodes. Each black block indicates $\Theta_{ij} \neq 0$, while each yellow indicates $\Theta_{ij} = 0$.

C.2 (T)CMBN-LSSA Setting

We used CMBN component from TCMBN for binary structural discovery and fed the IID data into the model to learn a covariance matrix on Pytorch. The learned matrix is fed into LSSA, a program on Matlab to compute the estimated precision matrix. Table 2 shows the parameters we set for the two-step procedure.

We obtained the covariance matrices on Dunnhumby dataset at the two timestamps $i = 1$ and $i = 101$, solved the LSSA with penalty parameter $\gamma = 0.000005$ on Matlab.

Table 2: Parameters of (T)CMBN-LSSA on simulation experiments and Illinois Votes dataset

nodes	samples	mixtures	epochs	batchsize	penalty γ
5	150	2	5000	150	0.001
5	300	2	5000	300	0.001
5	500	2	10	500	0.001
8	150	4	5000	150	0.0005
8	300	4	5000	150	0.001
8	500	4	100	500	0.001
18(votes)	1246	8	200	150	0.001-0.015

C.3 L-N-M, M-I and NeurISE

We ran L-N-M and M-I on Matlab according to the work by De Canditiis [2019] and the Matlab program by the author. [4] Specifically, we used L-N-M to indicate Wainwright-max procedure [Höfling and Tibshirani, 2009] to symmetricize the logistic neighborhood approach :the estimated quantities are not symmetric, i.e. $\widehat{\Theta}_{ij} \neq \widehat{\Theta}_{ji}$. In addition, a 10-fold CV procedure was applied to select the best penalty parameter for each node.

The M-I model evaluates the empirical covariance matrix, numerically invert it and perform a threshold on elements of the precision matrix. The thresholds we used in the experiments are 0.2 and 0.4 as two quantiles to zero out the entries of $\widehat{\Theta}$.

We trained NeurISE[Lokhov et al., 2020] on Google Colab with the default implementation and setting by the authors.[5] The produced $\widehat{\Theta}$ were symmetricized by the Wainwright-max procedure. We also used the two quantiles 0.2 and 0.4 to zero out the entries of $\widehat{\Theta}$.

C.4 Illinois Votes

We analyze the different votes of the Illinois House of Representatives during the period of the 114-th and 115-th US Congresses (2015-2019), which is available at voteview.com Lewis et al. [2019]. The Illinois House of Representatives has 18 seats (one per district), each one corresponds to a US Representative belonging to the Democratic or the Republican parties. An imputed version is available online[6] Le Bars et al. [2020]. As shown in Figure 2, the recovered graphs on the shrinkage path where we increase the penalty parameter from 0.001 to 0.015, are getting closer and closer to the real situation where representatives of Democratic party (blue) and representatives of Republican party (red) are more strongly "connected" within the party than outside. The two parties are connected by 10-th seat, a supercollaborator, a role that some representatives get by acting more independently and position themselves in the middle of the parties. This result agrees with early work Le Bars et al. [2020].

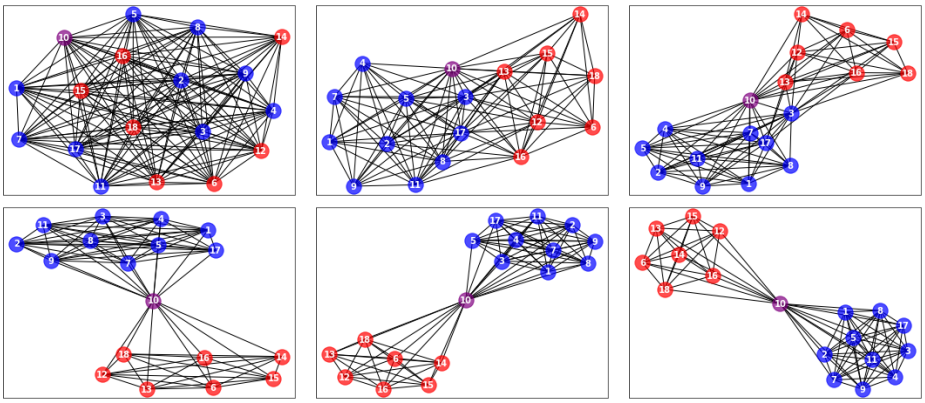


Figure 2: Recovered graphs for Illinois Votes data. Upper panel: From left to right the penalty parameter γ is 0.003, 0.006, 0.008; Lower panel: From left to right the penalty parameter γ is 0.011, 0.013, 0.015. When $\gamma = 0.011$, the Republican cluster (red) and the Democratic cluster (red) are formed, connected by a collaborator seat (purple).

[5]https://github.com/lanl-ansi/NeurISE
[6]https://github.com/BatisteLB/TVI-FL

D Proof

D.1 Theorem 1: Transformer with Temporal Encoding

Consider the type of transformer block described by Yun et al. [2019]. Our proof for a transformer with temporal encoding as a universal approximating function for any continuous function for sequence-to-sequence with compact support follows similarly as Proof of transformer with position encoding (Theorem 3 in Yun et al. [2019]). Without loss of generality, consider a sequence with timestamps $\{t_1, t_2, ..., t_n\}$ and let t_i being integer values and $t_i < t_j$ for all $i < j$. We choose a d-dimensional Temporal encoding for the n event epochs to be the following:

$$\mathbf{T} = \begin{bmatrix} 0 & t_2 - t_1 & \dots & t_n - t_1 \\ 0 & t_2 - t_1 & \dots & t_n - t_1 \\ \vdots & \vdots & & \vdots \\ 0 & t_2 - t_1 & \dots & t_n - t_1 \end{bmatrix}$$

This guarantees for all rows the coordinates are monotonically increasing and input values can be partition into cubes. The rest of the proof follows directly from proof of Theorem 3 by replacing n with t_n by performing quantization by feed-forward layers, contextual mapping by attention layers and function value mapping by feed-forward layers.

D.2 Theorem 2: Existence of a Transformer Network that Separates Points

We prove Theorem 2 by contradiction. Given any function f in the class of continuous sequence-to-sequence functions $\mathcal{F}$ that maps $\mathbf{X} \in R^{d \times L}$ to $\mathbf{H} \in R^{d \times L}$ with compact support. With loss of generality, consider $d = 1$. Suppose $\mathbf{X}_i \neq \mathbf{X}_j$ for some $i \neq j$ and $i, j \in \{1, 2, 3, 4, ..., L\}$. Given some $f \in \mathcal{F}$ that maps $\mathbf{X}$ to certain $\mathbf{H}^f$, where the i, j entries differ by $|\mathbf{H}_i^f - \mathbf{H}_j^f|$. For any transformer that cannot separate $(\mathbf{X}_i, \mathbf{H}_i)$ and $(\mathbf{X}_j, \mathbf{H}_j)$, that is $\mathbf{H}_i = \mathbf{H}_j$. The transformer cannot approximate the function f with ϵ distance since there is a always gap of $|\mathbf{H}_i^f - \mathbf{H}_j^f|$ with respect some entry-wise l^p norm. Thus universal approximating property fails.

References

Daniela De Canditiis. Learning binary undirected graph in low dimensional regime. *arXiv preprint arXiv:1907.11033*, 2019.

Holger Höfling and Robert Tibshirani. Estimation of sparse binary pairwise markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 10(4), 2009.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Batiste Le Bars, Pierre Humbert, Argyris Kalogeratos, and Nicolas Vayatis. Learning the piece-wise constant graph structure of a varying ising model. In *Int. Conf. on Machine Learning*, pages 675–684. PMLR, 2020.

Jeffrey B Lewis, Keith Poole, Howard Rosenthal, Adam Boche, Aaron Rudkin, and Luke Sonnet. Voteview: Congressional roll-call votes database. *See https://voteview.com/(accessed 27 July 2018)*, 2019.

Andrey Y Lokhov, Sidhant Misra, Marc Vuffray, et al. Learning of discrete graphical models with neural networks. *arXiv preprint arXiv:2006.11937*, 2020.

Pradeep Ravikumar, Martin J Wainwright, and John D Lafferty. High-dimensional ising model selection using l1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319, 2010.

Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-free learning of temporal point processes. *arXiv preprint arXiv:1909.12127*, 2019.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? *arXiv preprint arXiv:1912.10077*, 2019.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International Conference on Machine Learning*, pages 11692–11702. PMLR, 2020.