Conjoint Modeling of Temporal Dependencies in Event Streams

Ankur P. Parikh[*†]
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
apparikh@cs.cmu.edu

Asela Gunawardana[*]
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
aselag@microsoft.com

Christopher Meek
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
meek@microsoft.com

Abstract

Many real world applications depend on modeling the temporal dynamics of streams of diverse events, many of which are rare. We introduce a novel model class, Conjoint Piecewise-Constant Conditional Intensity Models, and a learning algorithm that together yield a data-driven approach to parameter sharing with the aim of better modeling such event streams. We empirically demonstrate that our approach yields more accurate models of two real world data sets: search query logs and data center system logs.

1 Introduction

Event streams—temporal sequences of discrete events, are ubiquitous in many domains such as the firing patterns of neurons [2], gene expression data [7], system error logs [13], and web search engine query logs. Learning a model for the temporal dependencies between events can be useful for understanding and exploiting the dynamics in such domains. For example, learning that particular system events on a machine predict failures at a later time may allow a system administrator to prioritize preventive maintenance. Understanding how web search queries for commercially valuable terms are dependent on prior queries, possibly for other topics, can help in targeted advertising. In many cases the data exhibits complex temporal dependencies. For example, what a user will query for at a particular time can depend on queries issued in the last few minutes, the previous day, as well as in the extended past. In a data center, the likelihood of a machine failing may depend on cascades of various prior warnings, errors, and failures.

In many domains, it is valuable to model fine distinctions between event types. For example, in targeted advertising, it is valuable to distinguish whether a user will issue queries related to mental health or to back pain rather than simply predicting that a user will issue a healthcare query. In a data center, it is more useful to learn that particular disk errors predict failed reboots than to know that generic error messages predict generic failures. While useful to model, such fine grained event types are rarer than the coarser grained ones that include them, and are therefore more difficult to model. Many models of temporal dependencies in event streams, such as the *Piecewise-Constant Conditional Intensity Model* (PCIM) [8], learn the dependencies of each event type separately, using independent sub-models for each event type. Thus, they are not able to model rare events well.

In this paper, we address this problem using parameter sharing, by introducing *Conjoint Piecewise-Constant Conditional Intensity Models* (C-PCIMs) and a learning algorithm for C-PCIMs, which yield a novel data-driven approach to modeling fine grained event streams. C-PCIMs generalize PCIMs by allowing parameters to be shared across event types, and our learning algorithm uses the data to determine which parameters should be shared. In particular, we give a conjugate prior that allows parameter learning for the C-PCIM to be performed as efficiently as for the PCIM, and that leads to a closed-form marginal likelihood, allowing efficient structure learning. During structure learning, the C-PCIM learns what event types in what historical contexts can be modeled by shared parameters, thereby allowing more efficient use of data during parameter estimation. In cases where events are structured, with the different event types having known attributes, we show how structure learning can take advantage of these attributes to distinguish between different event types when their depen-

[*] These authors contributed equally to the work.

[†] This research was conducted while APP was a research intern at Microsoft Research, Redmond.

dencies differ, while sharing parameters when they do not. Finally, we give empirical evidence that demonstrates the value of C-PCIMs in two real applications which are not well addressed by existing approaches– modeling the temporal query dynamics of web search users and modeling the temporal dynamics of system events in a data center. In the second application, we demonstrate that the expressive power of C-PCIMs combined with the data driven learning approach allows us to relax the strong assumption of identical machines, yielding further accuracy improvements.

2 Related Work

Event streams can be modeled in either discrete or continuous time. Using discrete time approaches such as Hidden Markov Models (HMMs) [1, 14] and Dynamic Bayesian Networks (DBNs) [5] require event times to be discretized, which requires a choice of sampling rate, and with it, trade-offs involving fidelity of representation, time-span of dependencies, and computational cost. We avoid this choice, and model event streams in continuous time. There have been a number of recent approaches for modeling continuous-time processes. C-PCIMs, like PCIMs [8], model event streams as marked point processes, where events have both an arrival time and a label specifying the type of event, via conditional intensity functions. A number of other closely related approaches [15, 23, 24] use regression techniques such as generalized linear models, Cox regression and Aalen regression to model conditional intensity functions. Continuous Time Noisy-Or [21] and Poisson cascades [22] are also approaches for modeling event streams. These approaches do not address model selection, and require a parametric form for temporal dependencies to be specified. Predictive performance is strongly impacted by this modeling choice, which is domain dependent [21, 22]. There has also been some recent work on nonparametric Bayesian approaches for modeling unlabeled event streams [17]. Continuous Time Bayesian Networks (CTBNs) [12] and Markov Jump Processes [16] are Markov process models of the trajectories of discrete variables over continuous time. In contrast to PCIMs and C-PCIMs, they are Markov process models. A CTBN can be used to model an event stream by modeling each kind of event as a transition of a "toggle" variable [20], and using latent state variables to model their dynamics, to give a continuous time analog of an HMM.

Conjoint PCIMs differ from PCIMs in the way that parameter are shared. Such approaches have been used in other problems such as for building hidden Markov models with large state spaces [9, 10], and for building n-gram language models [11]. Hierarchical Gamma-Exponential processes [18] are a hierarchical nonparametric Bayesian approach to conjoint modeling in Markov processes such as CTBNs. Regularization approaches may also be used for conjoint modeling [25], although we are unaware of applications to continuous time event modeling.

3 The Model

We represent an event sequence as $y = \{(t_i, l_i)\}_{i=1}^n$ with $0 < t_1 < \cdots < t_n$, where $t_i \in [0, \infty)$ is the time of the ith event and l_i is its label, drawn from a finite label set $\mathcal{L}$. The *history at time* t of event sequence y is the sub-sequence $h(t, y) = \{(t_i, l_i) \mid (t_i, l_i) \in y, t_i \leq t\}$. We write h_i for $h(t_{i-1}, y)$ when it is clear from context which y is meant. By convention $t_0 = 0$. We define the *ending time* $t(y)$ of an event sequence y as the time of the last event in y: $t(y) = \max(\{t : (t, l) \in y\})$ so that $t(h_i) = t_{i-1}$.

The data x, which is a particular event sequence, is modeled as a realization of a *regular marked point process* [4, 6] with likelihood

$$p(x|\theta) = \prod_{l \in \mathcal{L}} \prod_{i=1}^{n} \lambda_l(t_i|h_i, \theta)^{\mathbf{1}_l(l_i)} e^{-\Lambda_l(t_i|h_i;\theta)} \quad (1)$$

where $\lambda_l(t|h;\theta)$ is the *conditional intensity function* [4] for label l, and $\Lambda_l(t|h;\theta) = \int_{t(h)}^{t} \lambda_l(\tau|h;\theta)d\tau$. We write $\mathbf{1}_{\mathcal{Z}}(\cdot)$ for the indicator function of a set $\mathcal{Z}$ and $\mathbf{1}_z(\cdot)$ for the indicator of the singleton $\{z\}$. Intuitively, $\lambda_l(t|h;\theta)$ is the expected rate of events with label l at time t given the history h. Note that despite the similarity to the likelihood of a non-homogeneous Poisson process, this likelihood does not in general define a Poisson process as the conditioning on history can cause the independent increments property of Poisson processes to not hold. The conditioning on the entire history also means that such processes are non-Markovian. *Piecewise Constant Conditional Intensity Models* (PCIMs) [8] are a particular class of marked point process where the conditional intensity functions are restricted to be piecewise constant. In this paper, we introduce *Conjoint PCIMs* which are PCIMs that use a conjoint representation for the conditional intensity functions. These models are described below.

3.1 PCIMs

In this section, we review PCIMs [8]. PCIMs are a class of marked point process where the conditional intensity function for each label is a piecewise constant function of time, taking one of a finite number of values. That is $\lambda_l(t|y)$ is piecewise constant in t for all $t > t(y)$, and takes on values $\{\lambda_{ls}\}$ for $s \in \Sigma_l$, where Σ_l is a finite label-dependent *state set*. The value λ_{ls} taken on by $\lambda_l(t|y)$ at t for each y is specified

by a piecewise constant *state function* $\sigma_l(t, y)$, so that $\lambda_l(t|y) = \lambda_{l\sigma_l(t,y)}$.

Note that the state s summarizes all the information about t and y necessary for computing $\lambda_l(t|y)$ in that given s, $\lambda_l(t|y)$ can be computed without further information about t and y. However, unlike in Markov models such as CTBNs [12], the state does not contain all the information about t and y for predicting future states.

As described above, the conditional intensity function $\lambda_l(t|y)$ can be specified by a structure $S_l = (\Sigma_l, \sigma_l(\cdot, \cdot))$ consisting of the state set Σ_l and the state function $\sigma_l(\cdot, \cdot)$ and a parameter parameter vector θ_l composed of a non-negative intensity λ_{ls} for each $s \in \Sigma_l$. In turn, a PCIM is specified by a structure S and a parameter Θ that consist of the per-label structures S_l and per-label parameter vectors θ_l.

Gunawardana et al. show [8] that given the structure S, a product of Gamma distributions is a conjugate prior for Θ, and that under this prior, the marginal likelihood of the data can be given in closed form. Thus, parameter estimation can be done in closed form given a structure, and imposing a structural prior allows a closed form Bayesian score to be computed for a structure.

The structure of a PCIM can be represented by a set of decision trees [8]. In particular, each state function σ_l can be represented by a decision tree whose leaves represent the states $s \in \Sigma_l$, as shown in the example of Figure 1. The decision nodes in each tree contain functions that map a time t and a history y to one of its child nodes. These functions are piecewise constant in time, so that the state function $\sigma_l(t, y)$ represented by a decision tree is also piecewise constant. Structure learning for each label l is performed by starting with the trivial tree, and iteratively refining it greedily based on the closed form Bayesian score mentioned above. A detailed presentation of this learning procedure as generalized to C-PCIMs is given in section 4.1.

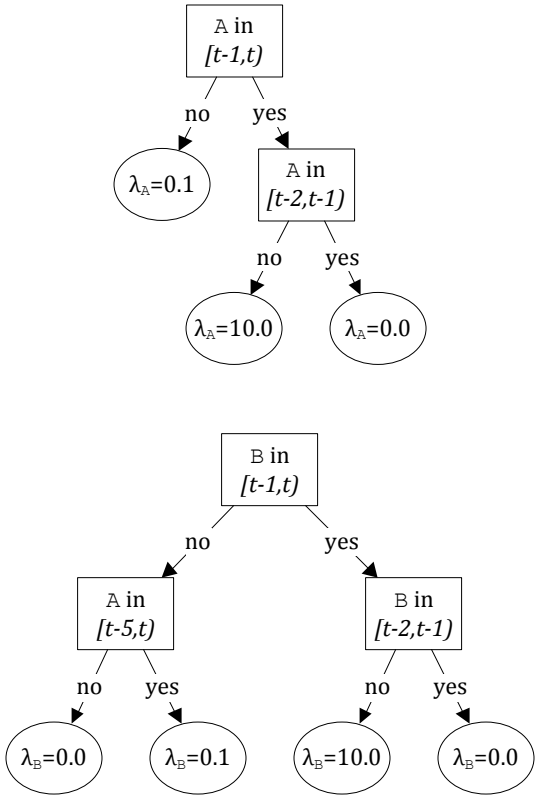

Figure 1: Example Decision trees representing a PCIM for a problem with $\mathcal{L} = \{\mathtt{A}, \mathtt{B}\}$.

3.2 Conjoint PCIMs

Conjoint PCIMs (C-PCIMs), like PCIMs, are marked point processes where the conditional intensity function $\lambda_l(t|y)$ are piecewise constant and take on a finite number of values, but unlike in PCIMs, the conditional intensity functions in C-PCIMs take on values from a single set of values shared across all labels $l \in \mathcal{L}$. Thus $\lambda_l(t|y)$ takes on values $\{\lambda_s\}$ for $s \in \Sigma$ which are shared across all l. Which of these values is taken on by $\lambda_l(t|y)$ is specified by a C-PCIM state function $\sigma(l, t, y)$ which unlike PCIM state functions, is also a function of the label l whose conditional intensity function is being evaluated. Thus, $\lambda_l(t|y) = \lambda_{\sigma(l,t,y)}$. C-PCIMs therefore allow an intensity value λ_s to be shared across conditional intensity functions for different labels, possibly at different times and for different histories. Thus, a C-PCIM is defined by a structure $S = (\Sigma, \sigma(\cdot, \cdot, \cdot))$ consisting of a state set Σ and a state function $\sigma(\cdot, \cdot, \cdot)$ as well as a parameter vector $\Theta = \{\lambda_s\}_{s\in\Sigma}$, all of which are shared across labels $l \in \mathcal{L}$.

We use a decision tree representation of the structure S of a C-PCIM. However, instead of using a different decision tree for each label l as in Gunawardana et al. [8], we use a single tree that is used across all labels, as shown in the example of Figure 2. The leaves of the tree represent states $s \in \Sigma$. However, the decision nodes of a C-PCIM tree contain functions that can depend on l as well as on t and y. In particular, each decision node contains a *basis state function* f chosen from a given set $\mathcal{B}$. Each basis state function $f(l, t, y)$ is piecewise constant in t for each l and y, and takes values from a finite basis state set Σ_f. Thus, a decision node with basis state function f has a child node corresponding to each $s' \in \Sigma_f$. Since the basis state functions are defined to be valid piecewise constant state functions, the mapping from (l, t, y) to

leaves given by the tree is also a valid piecewise constant state function $\sigma(l, t, y)$.

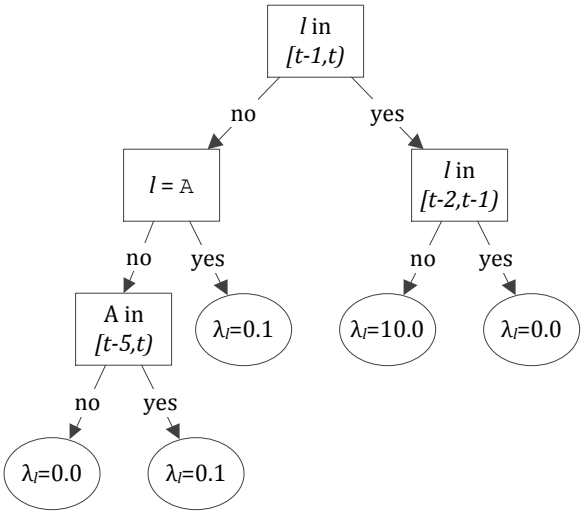

Figure 2: Decision tree representing a C-PCIM equivalent to the example PCIM of Figure 1.

4 Learning Conjoint PCIMs

In this section, we directly generalize the parameter and structure learning approaches for PCIMs [8] to apply to C-PCIMs. For C-PCIMs, the likelihood of equation (1) can be written as

$$p(x|S,\Theta) = \prod_{s\in\Sigma} \lambda_s^{c_s(x)} e^{-\lambda_s d_s(x)} \tag{2}$$

where $d_s(x)$ and $c_s(x)$ are sufficient statistics of the data. $d_s(x)$ is the total duration spent in state s, i.e., that $\sigma(l, t, h(t, x)) = s$ for some l. $c_s(x)$ is the number of times a label l occurs in x when the state function maps to state s for label l. Formally,

$$d_s(x) = \sum_{l\in\mathcal{L}} \int_0^{t(x)} \mathbf{1}_s\left(\sigma\left(l, \tau, h\left(\tau, x\right)\right)\right) d\tau$$
$$c_s(x) = \sum_i \mathbf{1}_l\left(l_i\right) \mathbf{1}_s\left(\sigma(l, t_i, h_i)\right).$$

Note that since $\sigma(l, \tau, h)$ in the integral above is piecewise constant in τ, the integral reduces to a sum over the constant pieces of $\sigma(l, \tau, h)$.

A product of Gamma priors on λ_s is conjugate for Θ. The corresponding prior and the posterior densities for λ_s are given by

$$p(\lambda_s|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda_s^{\alpha-1} e^{-\beta\lambda_s}$$
$$p(\lambda_s|\alpha,\beta,x) = p\left(\lambda_s|\alpha + c_s(x), \beta + d_s(x)\right).$$

In our experiments, we obtain the point estimate $\hat{\Theta} = \mathbf{E}[\Theta|S, x]$ from the training data, given by

$$\hat{\lambda}_s = \frac{\alpha + c_s(x)}{\beta + d_s(x)}.$$

4.1 Structure Learning

For structure learning, we can write the marginal likelihood of the data x given the structure S in closed form as

$$p(x|S) = \prod_{s\in\Sigma} \underbrace{\frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\Gamma(\alpha + c_s(x))}{(\beta + d_s(x))^{\alpha+c_s(x)}}}_{\gamma_s(x)}.$$

As with PCIMs, we use a Bayesian decision tree building procedure [3] in order to learn the structure S. We begin with the trivial structure $\Sigma = s_0$, $\sigma(l, t, y) = s_0$ with only the root node s_0, and refine the structure S by iteratively splitting leaves $s \in \Sigma$ based on basis state functions $f \in \mathcal{B}$. In particular Given a current structure $S = (\Sigma, \sigma)$, a new structure $S' = (\Sigma', \sigma')$ is produced by selecting a state $s \in \Sigma_l$ and a basis state function $f \in \mathcal{B}$ and refining s based on f as follows:

$$\Sigma'_l = \left(\bigcup_{s'\in\Sigma_f} \{s \odot s'\}\right) \cup \Sigma_l \backslash s$$
$$\sigma'(l, t, y) = \begin{cases} s \odot f(l, t, y) & \text{if } \sigma(l, t, y) = s \\ \sigma(l, t, y) & \text{otherwise} \end{cases}$$

where $\odot$ is the concatenation operator. Thus, S' can be represented as a tree where leaf s of the sub-tree S has been split according to the result of f.

In order to select the state s and basis state function f to use in producing a refined structure S' from S, we define a factored prior

$$p(S) \propto \kappa^{|\Sigma|}$$

on the structure S. The posterior probability of a structure S given the data x is then proportional to $p(S)p(x|S) = \prod_{s\in\Sigma} \kappa\gamma_s(x)$, which can be computed in closed form. Thus, the gain in $p(S|x)$ due to splitting state s using basis state function f is

$$\begin{aligned} \text{Gain}(S \to S') &= \frac{p(S'|x)}{p(S|x)} \\ &= \frac{\prod_{s'\in\Sigma'} \kappa\gamma_{s'}(x)}{\prod_{s\in\Sigma} \kappa\gamma_s(x)} \\ &= \frac{\prod_{s'\in\Sigma_f} \kappa\gamma_{s\odot s'}(x)}{\kappa\gamma_s(x)}. \end{aligned}$$

We refine the structure greedily, choosing the refinement with the highest gain, until no further gain results.

5 Basis State Functions for C-PCIMs

The modeling power of a family of C-PCIM is determined by the basis $\mathcal{B}$ of state functions selected. The basis needs to capture the aspects of the history that determine the intensities of events, and need to distinguish labels with different intensities. In addition, the basis needs to allow the sharing of parameters between labels to allow for generalization of event behavior between labels. This is particularly important in problems where some labels occur rarely. We will give basis state functions that take advantage of known structure in the label sets in order to do this. We first describe how we capture label space structure through label attributes, and then give an ontology of basis state functions that make use of this structure.

5.1 Structured Labels

When the labels have a known structure, we will take advantage of it in order to define models that can learn dependencies between events with labels that may be rare, or even not occur in the training data. For example, in data center system event logs, events may be labeled with the machine on which the event took place and the type of the event. An event of type `disk-sector-error` may occur on machine `9,045`. While we may have never observed a disk sector error on a different machine `6,732`, we may wish to allow the structure learning procedure to determine whether the behavior of `disk-sector-error` events generalizes across machines. Thus, we would like the basis state functions to be able to query for the message represented by a label independently of the machine.

In general, we assume that the label set $\mathcal{L}$ has a set of attributes $\mathcal{A}$, where each attribute $a \in \mathcal{A}$ can take values in a set $\mathcal{V}_a$. Label l takes on value $v_a(l)$ of attribute a. In the example above, $\mathcal{A} = \{\texttt{machine-id}, \texttt{event-type}\}$, and $\mathcal{V}_{\texttt{machine-id}}$ ranges over all the machines in the data center, and $\mathcal{V}_{\texttt{event-type}}$ ranges over all possible events. If prior information about the labels is available, it may be encoded through label attributes. For example, if we know a priori that machines in the data center are grouped into database servers and web servers, we could introduce an attribute `server-type` that takes on values $\mathcal{V}_{\texttt{server-type}}\{\texttt{database}, \texttt{web}\}$. On the other hand, we can access label identity as an attribute by using the attribute `identity` with $\mathcal{V}_{\texttt{identity}} = \mathcal{L}$, $v_{\texttt{identity}}(l) = l$. In the descriptions below, we will therefore always assume that there are label attributes defined. In cases where no structural information is available we will simply use $\mathcal{A} = \{\texttt{identity}\}$. In particular, the basis state functions for PCIMs [8] do not explicitly use label attributes, but can be described as using this trivial identity attribute.

5.2 Types of Basis State Functions

Allowing large classes of basis state functions that depend arbitrarily on both the history and time as well as the label is difficult computationally. We therefore restrict attention to three specific classes of basis state functions in this paper.

History Basis State Functions: A history basis state function $f(l, t, y)$ depends only on the history y and the time t and not the label l. In this paper, we concentrate on a particular class of history basis state functions $f_{a,v,d_1,d_2}(l, t, y)$ indexed by an attribute $a \in \mathcal{A}$, a value $v \in \mathcal{V}_a$ and time offsets $d_2 > d_1 \geq 0$, and given by

$$f_{a,v,d_1,d_2}(l,t,y) = \begin{cases} 1 & \text{if } \exists (t', l') \in y : \\ & \qquad t' \in [t - d_2, t - d_1) \\ & \qquad \wedge\, v_a(l') = v \\ 0 & \text{otherwise.} \end{cases}$$

That is $f_{a,v,d_1,d_2}(l, t, y)$ tests if the history y contains an event in the time window between d_1 and d_2 before t, whose label has the value v of attribute a.

Example. In modeling web search query logs, the history basis state function $f_{\texttt{query-category},\texttt{Health},1\text{ hr},1\text{ day}}(l, t, y)$ tests whether y contains a query whose `query-category` attribute is `Health` between 1 hour and 1 day before t.

Label Basis State Functions: A label basis state function $f(l, t, y)$ depends only on the label l and not the time t nor the history y. A label basis state function is $f_{a,v}(l, t, y)$ is indexed by an attribute $a \in \mathcal{A}$, a value $v \in \mathcal{V}_a$ and is given by

$$f_{a,v}(l,t,y) = \begin{cases} 1 & \text{if } v_a(l) = v \\ 0 & \text{otherwise.} \end{cases}$$

That is $f_{a,v}(l, t, y)$ simply tests whether the attribute a of label l has value v.

Example. The label basis state function $f_{\texttt{query-category},\texttt{Health}}(l, t, y)$ tests whether l has `query-category` attribute `Health`.

Match Basis State Functions: A match basis function $f_{a,d_1,d_2}(l, t, y)$ is given by

$$f_{a,d_1,d_2}(l,t,y) = \begin{cases} 1 & \text{if } \exists (t', l') \in y : \\ & \qquad t' \in [t - d_2, t - d_1) \\ & \qquad \wedge\, v_a(l') = v_a(l) \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the match basis state function tests whether the history y contains an event in the time window between d_1 and d_2 before t, whose label matches l in attribute a. This kind of basis state function is useful in modeling repetitions of a given attribute in an event stream.

Example. The basis state function $f_{\texttt{query-category},1\text{ hr},1\text{ day}}(l,t,y)$ tests whether y contains a query with the same `query-category` attribute as l between 1 hour and 1 day before t.

We note that history basis state functions are equivalent to the history basis functions of PCIMs [8] when the attribute a is restricted to be the `identity` attribute described above. Thus, a PCIM can be represented as a C-PCIM that uses only the `identity` attribute, and whose state function uses a tree that first splits based on every possible label basis state function, and then uses only history basis state functions. We use the term *Attribute PCIM* (A-PCIM) to refer to the slight generalization of PCIMs that allows the history basis state functions to use arbitrary attributes.

6 Experimental Results

In this section we evaluate the value of C-PCIMs on two real world tasks with large structured label spaces. The first is to model the query behavior of web search users, and the second is to model the behavior of a cluster of machines in a commercial data center.

In order to evaluate the value of C-PCIMs in modeling event streams with large label spaces, we compare C-PCIMs to PCIMs. To explore the gains due to conjoint modeling as opposed to the use of richer label attributes in modeling, we also compare to A-PCIMs. It has been shown that PCIMs have better computational and predictive performance in comparison to Poisson networks [15], which model conditional intensity functions using generalize linear models. We therefore do not compare to Poisson networks and other closely related approaches that use regression techniques to model the conditional intensity functions [23, 24]. CTBNs with "toggle" variables modeling events and latent variables modeling dynamics are a natural baseline for continuous time event modeling. We explored building such models using the CTBN-RLE toolkit [20], but the current version does not scale to these data sets [19].

6.1 Web Search Query Behavior

Queries issued by the users of a major commercial search engine were used to generate a training set consisting of approximately 100,000 queries collected from approximately 6,000 users over a two month period and a test set consisting of approximately 170,000 queries from approximately 6,000 users over the next month. There was no user or time overlap between the training and test sets. The test set was restricted to contain only users whose query history was available for at least two weeks. The queries were automatically mapped to a two level hierarchy of categories such as `Health & Wellness/Mental Health`. Thus, our label set consisted of the 476 categories in the hierarchy, while the 37 coarse level categories were used as a second (i.e. non-`identity`) attribute, which we name `coarse`. All labels occurred in the training set.

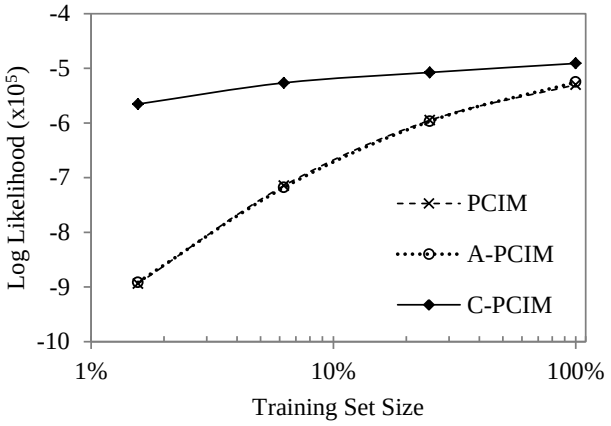

Figure 3: Test set log likelihood of PCIM, A-PCIM, and C-PCIM for the search query data, as a function of training set size.

We use a product of terms of the form of equation (1), one per user, to model the data, choosing not to model inter-user dependencies. We investigated three model classes for modeling users' query behavior. First, we built PCIMs which used only history basis state functions using the `identity` attribute (i.e. the history basis functions only tested for the occurrence of specific labels in the history). Second, we built A-PCIMs that were also allowed to use the `coarse` attributes in the history basis functions. Third, we built C-PCIMs that also used label basis state functions that used the `identity` and `coarse` attributes and match state functions that used the `identity` attribute. Match state functions using the `coarse` attribute were not allowed due to reduce the computation time. All models used a prior with $\alpha = 1/365$ and $\beta = 1$ day, and $\kappa = 0.001$. The history and match basis state functions used the time windows $[t-1\text{ hr}, t)$, $[t-1\text{ day}, t-1\text{ hr})$, $[t-7\text{ days}, t-1\text{ day})$, and $(\infty, t-7\text{ days})$. All models took less than 12 hours to train on a single 3 GHz workstation.

Figure 3 shows the test set log likelihood of all three

models as the amount of training data is varied. The log likelihoods of the PCIM and A-PCIM are nearly indistinguishable, while the C-PCIM performs much better, especially with smaller amounts of training data. This is the expected behavior, since C-PCIMs are better able to share parameters. Note that since C-PCIMs, A-PCIMs, and PCIMs define the same family of marked point processes, we expect them to approach the same predictive performance as the training set grows. While the gap between C-PCIMs and A-PCIMs/PCIMs does shrink as the amount of training data grows, C-PCIMs have higher training set likelihood even when all the training data is used. We examined the likelihoods assigned to each test event by the C-PCIM and the A-PCIM trained with the full training set, in order to determine the statistical significance of this gap. We grouped the per-event likelihoods by label, and found that the C-PCIM significantly outperforms the A-PCIM ($p = 0.01$) on 391 out of the 436 labels observed in the test set, while under-performing the A-PCIM on none, according to a paired sign test.

To understand the practical impact of the likelihood gains, we used importance sampling [8] to forecast whether each test user would issue `Health & Wellness/Mental Health` queries on the eighth day in the test set given their behavior in the first week. Precision-recall curves for the three models are given in Figure 4. Although there are only eight test users who issued these queries on that day, C-PCIMs make much better predictions than A-PCIMs and PCIMs. Such predictions are useful in applications such as targeted display (banner) advertising, where an advertiser may only wish their advertisements to be shown to web users who are interested in a topic related to their advertisement. The assumption is that users who will issue a query in a particular category during the day may be more receptive to advertisements related to that category during that day.

6.2 Data Center Machine Behavior

System logs from a cluster of machines in a commercial data center were used to generate a data set of about 300,000 logged system events from 71 machines, over the period of a month. There were 221 possible messages. This gave 15,691 possible labels, each of which was a machine-message combination. Each machine belonged to one of four machine types that was known a priori. Thus, each label had four attributes $\{\texttt{machine}, \texttt{message}, \texttt{machine-type}, \texttt{identity}\}$. The first two weeks of data was used for training, and the rest for testing.

We use a product of terms of the form of equation (1), one per machine, to model the data, choosing to not allow inter-machine dependencies. We experimented with using the power of C-PCIMs to allow machine specific dependencies. In particular, we compare a PCIM and a C-PCIM that treat machines identically with a PCIM and a C-PCIM that allow machine specific dependencies. Non-identical modeling of machines may be useful if, for example, a certain machine is old is more prone to failures. Models that treat machines identically are allowed to use only the `message` attributes of labels. The PCIM that treats machines identically is forced to use the same conditional intensity function for all labels that agree on their `message` attributes, pooling data from these labels during training. Note that in this setting, PCIMs and A-PCIMs are equivalent in both the identical machine and non-identical machine cases.

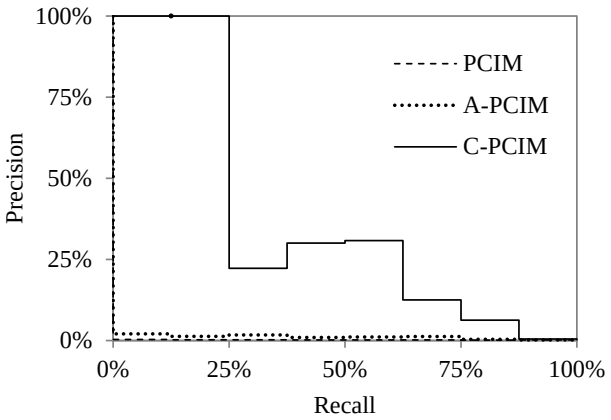

Figure 4: Precision-Recall curves of PCIM, A-PCIM, and C-PCIM for predicting `Health & Wellness/Mental Health` queries.

All models used the prior $\alpha = 0.01$, $\beta = 1.0$ day, and $\kappa = 0.01$. The C-PCIM trees were truncated at a depth of 30 to save computation. The history and match basis state functions used the time windows $[t - 20 \text{ min}, t)$ and $[t - 1 \text{ hr}, t - 20 \text{ min})$. The models took less than 2 hours to train, except the non-identical machine PCIM, which learned a separate tree for each of the 5,307 labels present in the training data. This took less than 12 hours.

Figure 5 shows the log likelihood of events after the first two weeks given the events of the first two weeks for all four models. Note the log likelihoods can be positive since the likelihoods are density values and not probabilities. It can be seen that building separate PCIMs for each label, without pooling data across machines for each message results in a large loss in test set likelihood, due to data sparsity. Both C-PCIMs outperform the PCIMs. Note that the identical machine C-PCIM only has access to message information, and therefore does not have access to any label struc-

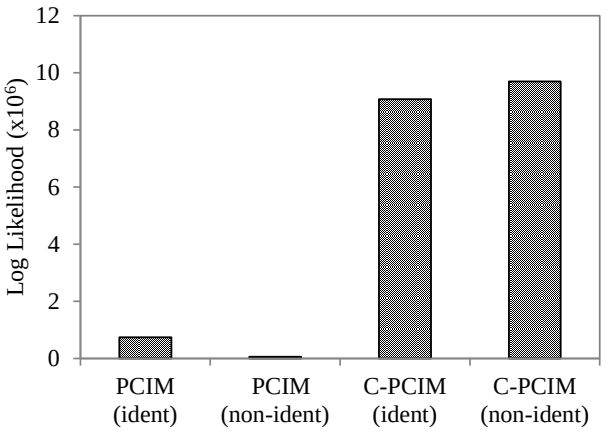

Figure 5: Test set log likelihood of PCIMs and C-PCIMs that treat machines as identical, and PCIMs and C-PCIMs that do not, for the data center event log data.

ture. However, it gives a large gain over PCIMs due to the use of match basis state functions, which model repeated events. The ability of the non-identical machine C-PCIM to leverage label structure to learn dependencies specific to machines or machine types leads to a further gain in likelihood. Unfortunately, events of interest such as machine failures are very rare in this data set, so that there are not enough test cases to obtain statistically significant comparisons between C-PCIMs and PCIMs.

7 Conclusions

We have introduced Conjoint PCIMs, and shown how they improve upon PCIMs in modeling the dynamics of two real world event streams with large structured label sets. We have shown how conjoint modeling across labels allows better models to be built from sparse data, and how C-PCIMs can leverage known structure in the label space. We have also shown that the predictive gains of C-PCIMs are not achieved by Attribute PCIMs that leverage label structure but do not use conjoint modeling.

While it would be of interest to compare the performance of C-PCIMs with other approaches such as CTBNs, limitations in the currently available CTBN implementation prevented us from doing so. It would also be interesting to investigate approaches that extend CTBNs to allow them to take advantage of label structure in the manner of C-PCIMs. Another future direction of interest is to investigate other extensions of PCIMs that allow parameter sharing across labels, such as hierarchical Bayesian approaches or regularization approaches.

References

[1] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Ann. Math. Stat.*, 37(6):1554–1563, December 1966.

[2] Emery N. Brown, Robert E. Kass, and Parth P. Mitra. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuro.*, 7(5), 2004.

[3] David Maxwell Chickering, David Heckerman, and Christopher Meek. A Bayesian approach to learning Bayesian networks with local structure. In *UAI*, 1997.

[4] D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Elementary Theory and Methods*, volume I. Springer, 2nd edition, 2003.

[5] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *AAAI*, 1988.

[6] Vanessa Didelez. Graphical models for marked point processes based on local independence. *J. Roy. Stat. Soc., Ser. B*, 70(1):245–264, 2008.

[7] N. Friedman, I. Nachman, and D. Peér. Using Bayesian networks to analyze expression data. *J. Comp. Bio.*, 7:601–620, 2000.

[8] Asela Gunawardana, Christopher Meek, and Puyang Xu. A model for temporal dependencies in event streams. In *NIPS*, 2011.

[9] Mei-Yuh Hwang and Xuedong Huang. Shared-distribution hidden Markov models for speech recognition. *IEEE Trans. Spch. & Aud. Proc.*, 1(4):414–420, October 1993.

[10] Mei-Yuh Hwang, Xuedong Huang, and Fileno A. Alleva. Predicting unseen triphones with senones. *IEEE Trans. Spch. & Aud. Proc.*, 4(6):412–419, November 1996.

[11] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoust., Spch., & Sig.. Proc.*, ASSP-35(3):400–401, March 1987.

[12] Uri Nodelman, Christian R. Shelton, and Daphne Koller. Learning continuous time Bayesian networks. In *UAI*, 2003.

[13] Adam Oliner and Jon Stearley. What supercomputers say - an analysis of five system logs. In *IEEE/IFIP Conf. Dep. Sys. Net.*, 2007.

[14] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, February 1989.

[15] Shyamsundar Rajaram, Thore Graepel, and Ralf Herbrich. Poisson-networks: A model for structured point processes. In *AIStats*, 2005.

[16] Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *UAI*, 2011.

[17] Vinayak Rao and Yee Whye Teh. Gaussian process modulated renewal processes. In *NIPS*, 2011.

[18] Ardavan Saeedi and Alexandre Bouchard-Côté. Priors over recurrent continuous time processes. In *NIPS*, 2011.

[19] Christian R. Shelton. Personal communication, 2012.

[20] Christian R. Shelton, Yu Fan, William Lam, Joon Lee, and Jing Xu. Continuous time Bayesian network reasoning and learning engine. *JMLR*, 11, 2010.

[21] Aleksandr Simma, Moises Goldszmidt, John MacCormick, Paul Barham, Richard Brock, Rebecca Isaacs, and Reichard Mortier. CT-NOR: Representing and reasoning about events in continuous time. In *UAI*, 2008.

[22] Aleksandr Simma and Michael I. Jordan. Modeling events with cascades of Poisson processes. In *UAI*, 2010.

[23] Wilson Truccolo, Uri T. Eden, Matthew R. Gellows, John P. Donoghue, and Emery N. Brown. A point process framework relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *J. Neurophysiol.*, 93:1074–1089, 2005.

[24] Duy Q. Vu, Arthur U. Asuncion, David R. Hunter, and Padhraic Smyth. Continuous-time regression models for longitudinal networks. In *NIPS*, 2011.

[25] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. Roy. Stat. Soc., Ser. B*, 68(1):49–67, 2006.