A HAWKES' EYE VIEW OF NETWORK INFORMATION FLOW

Michael G. Moore and Mark A. Davenport

Georgia Institute of Technology
School of Electrical and Computer Engineering
{mmoore90,mdav}@gatech.edu

ABSTRACT

An important problem that arises in the analysis of many complex networks is to identify the common pathways that enable the flow of information (or other quantities) through the network. This is a particularly challenging problem when the only information observed consists of the timing of events in the network. We develop a framework based on multidimensional Hawkes processes that can be used to determine how events are related. This extends the capability of Hawkes process-based models to infer how network events relate. We then show how a simple dynamic program can exploit this data to recognize chains of events and provide a much deeper insight into the behavior of nodes within the network. Simulations are provided to demonstrate the capabilities and limitations of this framework.

Index Terms— point processes, Hawkes processes, dynamic programming, network information flow

1. INTRODUCTION

A fundamental problem in the analysis of complex networks is determining how information or other quantities flow through the network. For example, in a communication network we might be interested in who is communicating with whom; in a network of neurons we may wish to know how certain neurons affect other neurons in the network; or in a financial network we might want to identify the end parties of transactions that pass through (possibly many) intermediaries.

When we can observe interactions between nodes in the network in detail, there are a number of techniques that can help to identify this kind of structure. However, in many cases we may have only limited information about what is happening. For example, in a wireless network we might be able to identify the source of a transmission but not the destination. In many other communication/social networks, privacy concerns and other practical limitations might preclude knowledge of the intended recipient of an interaction. Such limitations also arise in neural networks, financial transaction networks, and many others. However, the timings of events (e.g., transmissions in a wireless network) in the network are often easier to obtain. Furthermore, this information can be sufficient to learn about the network if we assume that events at one node are likely to induce a response at some other node.

This can be formalized using the notion of multidimensional Hawkes processes (MHPs). MHPs have found applications modeling social networks [1, 2], communication networks [3], neural networks [4], financial transactions [5], and more. Fitting an MHP to a collection of recorded events can provide insight to the structure of the network. However, this model is largely limited to discovering the direct connections. If nodes act as intermediaries between others, the meaningful interactions in a network can become obfuscated. For example, packets in wireless ad hoc networks may be forwarded multiple times to reach their final destination. It may be of greater interest to know the source and destination of the packet rather than the topology of the network. An MHP can be used to determine which nodes are connected, but does not directly reveal this source/sink information.

In this paper, we explore novel techniques based on MHPs which aim to uncover the relationship between events more explicitly. Recognizing relationships between events (rather than nodes) can enable us to discover more complex traffic patterns within a network, something not provided by existing approaches based on MHPs.

2. HAWKES PROCESSES

A Hawkes process is a point process with an autoregressive dependence on past events, where rate at any instant is a function of recent events in the process (the statistical rate is a function of the previous empirical rate). Specifically, the *conditional intensity function* (CIF), or rate, of a Hawkes process given event times t_k is

$$\lambda(t) = \mu + A \sum_k \gamma(t - t_k). \tag{1}$$

The kernel $\gamma(t)$ is a known function that is strictly causal ($\gamma(t) = 0$ for $t \leq 0$), integrable, and is usually constrained to

This work was supported by grants NRL N00173-14-2-C001, AFOSR FA9550-14-1-0342, NSF CCF-1350616, CCF-1409406, and CMMI-1537261.

978-1-4673-7802-4/16/$31.00 ©2016 IEEE

be nonnegative. The innovation rate μ and excitability A are also commonly nonnegative (if it is possible to achieve a negative CIF then such cases must be handled specially as there is no meaningful interpretation). This results in a process that is self-exciting, where events tend to occur in clusters.

A *multidimensional Hawkes process* (MHP) extends this notion to a group of processes that can excite each other. In this case, the CIF of process i is described by

$$\lambda_i(t) = \mu_i + \sum_j A_{ij} \sum_{k \in K_j} \gamma(t - t_k) \tag{2}$$

where K_j is the set of events originating from node j. The matrix A now describes how likely nodes are to excite each other. Specifically, A_{ij} determines how likely process i is to react (by creating another event) to an action by process j. In this way, A forms a weighted and directed adjacency matrix for the network.

A central task in applying MHPs involves estimating the parameters μ and A. The negative log-likelihood of the activity of process i over the interval $t \in [0, T]$ is

$$\mathcal{L}_i(\mu, A) = \int_0^T \lambda_i(t) - \sum_{k \in K_i} \log \lambda_i(t_k). \tag{3}$$

Maximum likelihood estimation requires us to minimize this convex function. Possible tools for performing this operation include SPIRAL [6] and composite self-concordant minimization [7]. The authors find that a projected quasi-Newton method that approximates the Hessian by its diagonal is a simple program that works extremely well in practice. The negative log-likelihood of all processes combined is simply $\mathcal{L}(\mu, A) = \sum_i \mathcal{L}_i(\mu, A)$ but, since each likelihood depends on only an isolated subset of the parameters, each term can be optimized individually.

3. INTER-EVENT INFLUENCE

The MHP model provides a way to estimate how much influence *nodes* have on each other in a network, but another important question concerns how much influence individual *events* exert. Specifically, we might wish to know which events are responsible for causing which other events in a sample. To address this question, we will assume that events have a single cause – events can be generated either spontaneously (for reasons internal to the node) or can be a result of a *single* previously-observed event. While this assumption can be relaxed in practice, the following probability-based statements rely upon it.

The CIF of an MHP dictates the rate of event occurrence through a compositional structure that adds the contributions of multiple terms. Let i_k be the node that created the event at time t_k (the i such that $k \in K_i$) and consider

$$\lambda_{ik}(t) = A_{ii_k} \gamma(t - t_k), \tag{4}$$

which is just one component of the overall CIF in (2). $\lambda_{ik}(t)$ tells us how much intensity event k contributes to process i at any time t. Because rate is the probability of occurrence within a differential time interval, we can make probabilistic statements about the cause of an event that occurs at a specific time. By evaluating $\lambda_{i_\ell k}(t_\ell)$ we can quantify the contribution of event k toward generating an event at time t_ℓ (on process i_ℓ). The probability that event k is responsible for event ℓ is

$$\mathbb{P}(k \to \ell) = \frac{\lambda_{i_\ell k}(t_\ell)}{\lambda_{i_\ell}(t_\ell)}. \tag{5}$$

We can organize these probabilities into an event causation probability matrix E where $E_{k\ell} = \mathbb{P}(k \to \ell)$. Note that, because $\gamma(t)$ is strictly causal, there is no possibility that events are caused by later or concurrent events (including themselves). This means our event causation probabilities form a (weighted) directed, acyclic graph. If $\gamma(t)$ is time-localized (virtually zero beyond some value of t) then this graph is also sparse and, if events are time-ordered, banded.

We can also use this scheme to consider chains of potentially connected events. For example, the probability that event a caused event b which in turn resulted in event c is $\mathbb{P}(a \to b) \cdot \mathbb{P}(b \to c)$, and the same can be done for arbitrarily long candidate sequences. It is also worth noting that we can compute the probability that an event is an innovation (not the result of a previous event, but rather the start of a new chain) as

$$\mathbb{P}(\varnothing \to \ell) = \frac{\mu_{i_\ell}}{\lambda_{i_\ell}(t_\ell)}, \tag{6}$$

or the probability that an event k is a terminus (does not result in another event) as

$$\mathbb{P}(k \to \varnothing) = \prod_\ell (1 - \mathbb{P}(k \to \ell)). \tag{7}$$

One can use these quantities to identify the endpoints of a chain of related events.

4. EVENT CHAINS

If we wish to find the most likely chain of events that includes a particular event then a simple dynamic program, such as the Dijkstra algorithm [8], can be used to do so. In some contexts, this may be all we care about. However, given the significant ambiguity that often arises in our model, the "most probable" chain can still be very unlikely. We will see later that it can be beneficial to consider additional candidates to gain an effective understanding of the network dynamics.

Even if E is banded and acyclic, the number of possible chains in the graph is still exponential in the total number of events. But if we only consider chains with at least some minimum probability, we get a dramatic reduction (to roughly linear in the number of events). This can still be a large number, however, so it will be worth examining an efficient way to explore them all.

Algorithm 1 Low Cost Paths for Acyclic Graphs

inputs: $E' \in [0,\infty)^{n\times n}$, $k_{\text{start}} \in \{1\dots n\}$, $k_{\text{stop}} \in \{1\dots n\}$, $\alpha' \in [0,\infty)$
initialize: $R^{\text{cost}}_{k_{\text{stop}},1} = 0$, $R^{\text{node}}_{k_{\text{stop}},1} = k_{\text{stop}}$, $R^{\text{next}}_{k_{\text{stop}},1} = \varnothing$, $u_{\{1\dots n\}\setminus k_{\text{stop}}} = \text{true}$, $u_{k_{\text{stop}}} = \text{false}$, $n^{\text{path}}_{k_{\text{stop}}} = 1$
call: TRAVERSE(k_{start})
for $i \in \{1\dots n^{\text{path}}_{k_{\text{start}}}\}$ **do**
  $r = R_{k_{\text{start}}i}$, $C_i = r^{\text{cost}}$, $\ell = 0$
  while $r \neq \varnothing$ **do**
    $\ell = \ell + 1$, $P_{i\ell} = r^{\text{node}}$, $r = r^{\text{next}}$
return P, C

SUBROUTINE: TRAVERSE(k)
initialize: $\ell = 0$
for i s.t. $E'_{ki} \leq \alpha'$ **do**
  if u_i **then**
    recurse: TRAVERSE(i)
  for $j \in \{1\dots n^{\text{path}}_i\}$ **do**
    if $E'_{ki} + R^{\text{cost}}_{ij} \leq \alpha'$ **then**
      $\ell = \ell + 1$, $R^{\text{cost}}_{k\ell} = E'_{ki} + R^{\text{cost}}_{ij}$, $R^{\text{node}}_{k\ell} = k$, $R^{\text{next}}_{k\ell} = R_{ij}$
$n^{\text{path}}_k = \ell$, $u_k = \text{false}$
return

We want to find all sequences of events (paths) $S = \{s_1 \dots s_\sigma\}$ such that $\mathbb{P}(S) \geq \alpha$ where

$$\mathbb{P}(S) = \prod_{i=1}^{\sigma-1} E_{s_i,s_{i+1}}. \tag{8}$$

To do this, we will use a graph traversal algorithm we will call *Low Cost Paths for Acyclic Graphs* (LCPAG). This dynamic program addresses the *all simple paths* problem [9] with modifications such that it prunes paths with excessive cost and does not work on graphs with cycles. To maintain consistency with other graph search algorithms, which traditionally use additive rather than multiplicative cost, we will instead present an algorithm to find all paths S such that $C \leq \alpha'$ where

$$C = \sum_{i=1}^{\sigma-1} E'_{s_i,s_{i+1}}. \tag{9}$$

This is equivalent to our problem if we use $E'_{ij} = -\log E_{ij}$ and $\alpha' = -\log \alpha$.

The algorithm is presented in Algorithm 1. The idea is that if we know all the paths (and their costs) from a k_i to k_{stop} then finding k_i is just as good as finding k_{stop}. We begin our search from k_{start}, where we ask each of its neighbors for every path (and associated cost) they know that goes through them and reaches k_{stop}. If a neighbor's path's cost plus its cost from k_{start} is less than α', we extend that path by connecting k_{start} (and record the new cost). Now we know every way to reach k_{stop} from k_{start}.

Of course, k_{start}'s neighbors may not know how to reach k_{stop} and, if they don't, must ask their neighbors before they can answer. We recurse the function for every node (or at least those that could potentially be along viable routes) that doesn't know the paths it can take to reach k_{stop}. Once a node knows all the paths to the end, we mark it (set u_k to false in the algorithm) and use its saved result instead of recomputing. The constraint that the graph is acyclic ensures that we can finish evaluating a node completely before we evaluate its parent. We can follow R like a linked list (though it is actually a reversed-tree or funnel), inspecting the cost from the head, to see what events are in a possible chain. The final stage of the algorithm (the part following the subroutine call) simply does this to provide a listing of every path.

We can incorporate the innovation and termination probabilities by adding two dummy vertices. One dummy vertex connects to every event with the corresponding innovation probability and the other is connected from every event using the termination probability. By finding all plausible paths from the innovation dummy to the termination dummy, we have found every possible chain of events with probability α or greater.

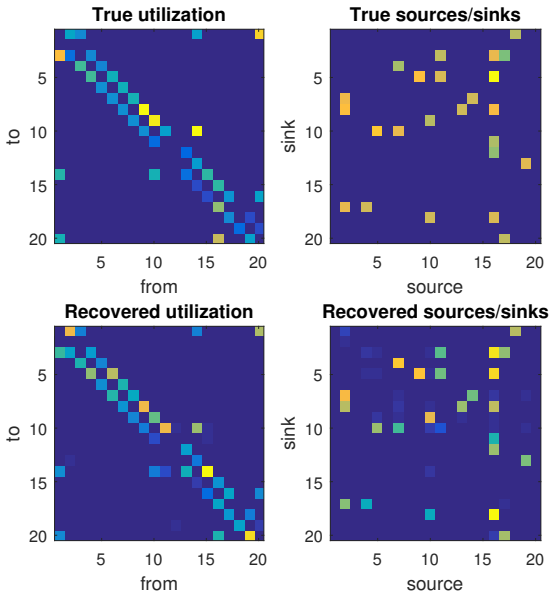
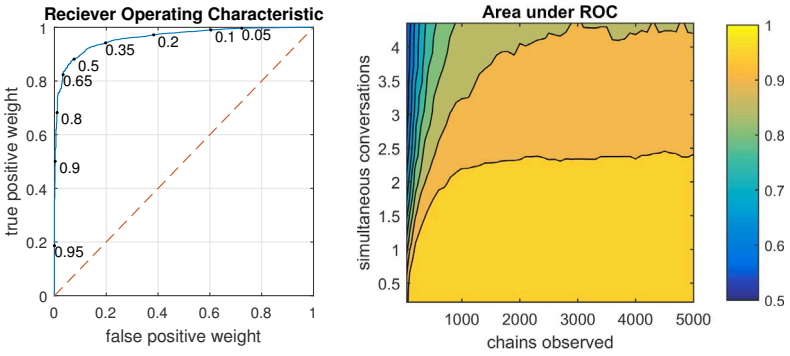


Fig. 1. Ground truth (top) and recovered (bottom) link utilization (left) and source/sink pairs (right, measured as in (10)) for example simulation using $\alpha = 50\%$. Yellow (bright) values indicate stronger connections. Though the MHP accurately infers the utilization parameters, it may not reflect the actual behavior of the network.

5. TRACKING THE FLOW OF INFORMATION

The LCPAG algorithm allows us to group events in an MHP into probable chains of events. This can allow us to track the flow of information in a network. Identifying common information pathways (for example, identifying pairs of nodes which are communicating with each other) in a complex network is a different problem than identifying the local structure of a network. While MHPs can be effective in learning local network structure, they do not directly provide much insight



(a) (b)

Fig. 2. (a) Example comparison of hit and false-alarm rate of sum-probabilities ("What fraction of the total weight of correct and incorrect chains do we find?") included when we limit our consideration to paths of some minimum probability. The probability thresholds α that result in certain trade-offs are labeled. (b) Average area under ROC curve when thresholding the recovered source/sink pairs. Contours are drawn every 0.05 area.

into the common information pathways. However, we will see below that the techniques described above can effectively leverage MHPs to learn this deeper structure.

We will consider a simulated network where messages are relayed from one node to another in order to reach their destination. These messages are passed between sets of sources and sinks (destinations).[1] It is important to note that in order to simulate such a network, we do *not* generate data using an MHP as an MHP contains no explicit way to enforce causal structure between events.

Nevertheless, we can choose to fit an MHP model to the data as described in Section 2. One such realization is shown in Fig. 1. We observe that although the MHP parameters in the matrix A provide an accurate estimate of the local connectivity (and utilization) of the different links in the network, they do not resemble the underlying source/sink structure.

To perform source/sink recognition, we use the LCPAG algorithm to identify likely chains of events. Let $\mathcal{P}_{ij}^{\alpha}$ be the set of all event chains that begin with an event corresponding to node j and end with an event corresponding to node i and have probability at least α. We can estimate the number of event chains beginning with node j and ending with node i as

$$B_{ij}(\alpha) = \sum_{\rho \in \mathcal{P}_{ij}^{\alpha}} \mathbb{P}(\rho). \tag{10}$$

Increasing α means fewer candidate paths are considered. Since we expect the true event chains to be relatively plausible, increasing α can disproportionately diminish the impact of incorrect chains (at least to a point). This results in a "receiver operating characteristic" (ROC) trade-off, such as in Fig. 2(a).

This source/sink recognition procedure can be very effective, as seen in the right half of Fig. 1, although it is naturally subject to some degree of inaccuracy. The primary driver of this inaccuracy is ambiguity caused by heavy event traffic: increasing event density makes it more challenging to understand the intent and effect of any individual event. This density can be increased by either increasing traffic volume or having nodes that act as bottlenecks in the network.

To visualize the impact of increasing traffic, we simulate many instances of the graph described above. We observe the area under the ROC curve that represents how many correct and incorrect pairs we discover (weighted by their probability) when excluding any source/sink pairs below a given threshold. The averaged results over many trials are shown in Fig. 2(b). When the number of observed chains is small, we may be missing many source/sink pairs because they are barely expressed. However, for moderate traffic volumes, we quickly reach a point where we can efficiently identify source/sink pairs. We also note that for any fixed number of observed chains, we see that increasing the traffic density (measured here by the average number of simultaneous conversations) increases ambiguity and limits the performance we can expect to achieve with this method.

In summary, we believe that the simulations described above are a convincing demonstration of the potential for MHPs combined with the LCPAG algorithm to identify and track the flow of information in settings where the main source of data is event timings. A limitation of this approach is the drop in performance in high-traffic networks, but we believe that this could be improved in future work by developing ways to "sparsify" the output of the LCPAG algorithm to eliminate chains of events through unusual sequences of nodes.

[1] The network begins with 20 nodes that connect in a ring and then adds additional bidirectional connections between pairs of nodes with a 3% probability. We then consider all pairs of nodes and assign them to be a source/sink pair with 5% probability. A source in a pair will send messages to the matching sink by using intermediate nodes (if necessary). When a node receives a message with a destination other than itself, it will pass it to a neighbor that is closer to the sink. Additionally, when a message arrives at the sink, we assume that the sink responds with a single message (to which no one will respond). In many scenarios the sink might reply by sending a message back to the original source, but since single (isolated) transactions between source/sink pairs likely represents the most challenging case, we restrict our attention to this setting. We observe 2000 source/sink transactions (event chains) in our simulation, approximately 105 per source/sink pair.

6. REFERENCES

[1] K. Zhou, H. Zha, and L. Song, "Learning social infectivity in sparse low-rank networks using multi-dimensional Hawkes processes," in *Proc. Int. Conf. Art. Intell. Stat. (AISTATS)*, Scottsdale, AZ, May 2013.

[2] M. Farajtabar, N. Du, M. Gomez-Rodriguez, I. Valera, H. Zha, and L. Song, "Shaping social activity by incentivizing users," in *Proc. Adv. in Neural Processing Systems (NIPS)*. Montréal, Québec, Dec. 2014.

[3] M. Moore and M. Davenport, "Learning network structure via Hawkes processes," in *Proc. Work. Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, Cambridge, United Kingdom, July 2015.

[4] P. Reynaud-Bouret, V. Rivoirard, and C. Tuleau-Malot, "Inference of functional connectivity in neurosciences via Hawkes processes," in *Proc. IEEE Global Conf. Signal and Information Processing (GlobalSIP)*, Austin, TX, Dec. 2013.

[5] E. Bacry, K. Dayri, and J. Muzy, "Non-parametric kernel estimation for symmetric Hawkes processes: Application to high frequency financial data," *Eur. Phys. J. B*, vol. 85, no. 157, 2012.

[6] Z. Harmany, R. Marcia, and R. Willett, "This is SPIRAL-TAP: Sparse Poisson Intensity Reconstruction ALgorithms – Theory and Practice," *IEEE Trans. Image Processing*, vol. 21, no. 3, pp. 1084–1096, 2012.

[7] Q. Tran-Dinh, A. Kyrillidis, and V. Cevher, "Composite self-concordant minimization," *J. Machine Learning Research*, vol. 16, pp. 371–416, 2015.

[8] E. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[9] G. Danielson, "On finding the simple paths and circuits in a graph," *IEEE Trans. Circuit Theory*, vol. 15, no. 3, pp. 294–295, 1968.