Network Hawkes Process Edge Detection with Shallow Learning

Kurtis Liu, University of Wisconsin—Madison , {kliu89@wisc.edu}

I. Abstract

The Network Hawkes process is a temporal model for a sequence of events on a set of nodes, where a event on some nodes would induce a probability on other nodes to trigger an event. A previous work, (Scott W. Linderman & Ryan P. Adams, 2014), attempted to recover the underlying network, given the sequence of events. The model consisted of a large set of complicated priors, and searched for the solution with Gibbs sampling.

I removed all the priors, except one only for experiment issues, restated the problem as a optimization problem. With a neural network with no hidden layer, I can give a model that predicts the Poisson lambda rate, and the edge weights between the nodes. With a neural network with 1 hidden layer, and a constitutional layer, I decrease the degree of freedom. I also prove that the true model lies in both hypothesis spaces, and recover the basis forming the impulse function with the network edge weights.

II. Terminology & Preliminaries

In this paper, K is the number of nodes, dt_max is the max time a event can induce another.

III. Changes from Linderman & Adams 2014

For the network hawkes process, the probability of an event happening at time t, node k is modeled as eq.0. Linderman decomposes the impulse responses to eq.1. Then a Log Gaussian Cox process to model the background rate eq.2. $A_{k,k'}$ is an adjacency matrix of the network. $W_{k,k'}$ is the weight of the edge. $g_{k,k'}$ is an positive impulse function with [0, dt_max] support. μ Is the background rate , and α is a weight. All variables follow a distribution prior.

$$\lambda_k(t)=\lambda_{0,k}(t)+\sum_{e,time(e)<t}\lambda'_{node(e),k}(time(e)-t) \qquad \text{(eq. 0)}$$

$$\lambda'_{k,k'}(\Delta t)=A_{k,k'}W_{k,k'}g_{k,k'}(\Delta t) \qquad \text{(eq. 1)}$$

$$\lambda_{0,k}(t)=\mu_k+\alpha_k\exp(y(t)),\ y(t)\sim GP(0,K(t,t')) \qquad \text{(eq. 2)}$$

For our model, we only keep eq.0. But model the others as follows:

$$\lambda'_{k,k'}(\Delta t)=f_{k,k'}(\Delta t) \qquad \text{(eq. 3)}$$

$$\lambda_{0,k}(t)=\mu_k \qquad \text{(eq. 4)}$$

When the model using discrete time bins, $f_{k,k'}(\Delta t)=A_{k,k',\Delta t}$ can be determined point by point.

IV. The Maximization Target

The maximization target is the log likelihood for the predicted rate, given the observed events from our model (eq.5). For a small dataset (wrt. dt_max), the best solution is $\lambda_k(t)=\delta_k(t)$, where $\delta_k(t)$ indicates an event.

$$\sum_{k,t,\,event\,at(k,t)}-\log(\lambda_k(t))+\sum_{k,t,\,no\,event\,at(k,t)}-\log(1-\lambda_k(t)) \qquad \text{(eq. 5)}$$

To solve the over fitting problem on a small data set, we smooth $\delta_k(t)$ with a kernel, and add a background rate (eq.6). Our optimization target is now $\lambda_k(t)=\mu_k(t)$, in (eq.7).

$$\mu(t)=\gamma\cdot\frac{events}{timebins}+(1-\gamma)kernel(t)*\delta(t) \qquad \text{(eq. 6)}$$

$$\sum_k\sum_t \lambda_k(t)-\mu_k(t)dt \qquad \text{(eq. 7)}$$

This smoothing step is the only prior we used in our model. γ is the prior on how many events come from the background rate, and how many others are induced. $\frac{events}{timebins}$ is the observed background rate.

V. A New Found Constraint

When a hawkes model runs to infinity, and the background rate is non-zero, we found a constraint (eq.8). a_{ij} here is the influence from node i to j. in our case, $a_{ij}=\int_t f_{i,j}(t)$.

$$\begin{pmatrix} a_{11} & a_{12}\dots & a_{1K} \\ a_{21} & a_{22}\dots & a_{2K} \\ \vdots & \vdots\dots & \vdots \\ a_{K1} & a_{K2}\dots & a_{KK} \end{pmatrix}\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_K \end{pmatrix}=\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_K \end{pmatrix}-\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \end{pmatrix} \qquad \text{(eq. 8)}$$

This is the steady state of the hawkes process. It provides a better view on how the background rate should be estimated. Instead of just using $\frac{\gamma\cdot events}{timebins}$, we mix can a gradient decent step towards this equation in our later optimization process.

* I only have this equation in theory. It is not integrated into the code. I guess a direct gradient decent gives a positive feedback. When b grows, a lowers, then b grows even more. So I need another way to use this constraint in the optimization process.

** Linderman used the constraint max |eig(a_{ij}) |< 1 , meaning all event influences decay, and does not have infinite feedback. His implementation just terminates when the constraint is not met.

VI. Optimization via 1-Layer Neural Net

Eq.7 has a least square solution, but the least square solution does not guarantee positive edges on the graph. I modify the non-linear step of a neural net to make positive edges.

$$output=\frac{1}{1+e^{-wx+c}}+B$$

The input x is the K x dt_max vector, consisting of all the event indicators in the dt_max time window. C is set to 10, or any negative constant. The idea of C is to let the output be close to 0 by default, but x can contribute to a positive impulse. The output is a K length vector, the predicted rate at each node given the previous events.

After each stochastic gradient decent step, I do the following to keep positive edges.

$$w=max(0,w)$$

The network is then convoluted over time. Each training data is the [T-dt_max,T-1] x K event indicators, the truth output is the length K vector, $\lambda_k(T)$.

VII. Optimization via Neural Net with 1 Hidden Layer

In Appendix B. of (Scott W. Linderman & Ryan P. Adams, 2014), he showed that the impulse functions can be modeled by a set of basis impulse functions. I design a 2-layer neural net that models this.

Layer 1: $y_{out1} = sigmoid(W_1 \cdot x)$

Layer 2: $y_{out} = sigmoid(W_2 \cdot y_{out1}) + B$

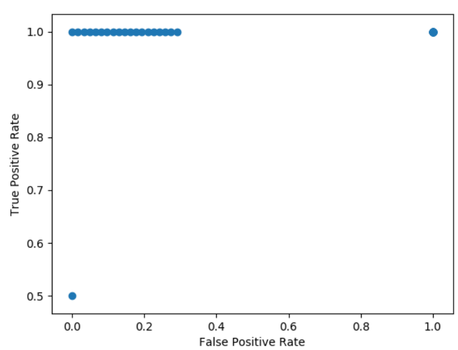
W1 is the matrix mapping the K*dt_max vector to the coefficient space of the N basis vectors.
W2 is the matrix mapping from coefficient space to the length dt_max impulse function.
Since the basis vectors are shared between nodes, W2 is convoluted over K expected impulses of length dt_max.

The 1 hidden layer neural net reduces the degree of freedom, by using a convolution. It also outputs basis vectors that might show characteristics of the impulses.

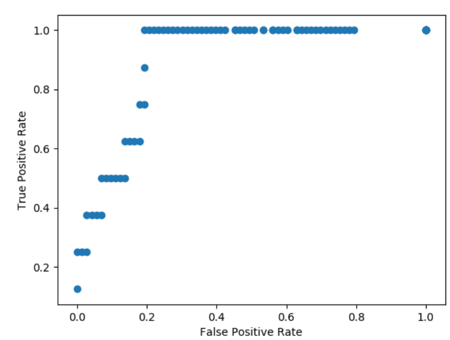
VIII. Experiment Results

The training data I use is generated by a synthetic Hawkes process. Since I used a sigmoid function, the edge weights are non-linear.

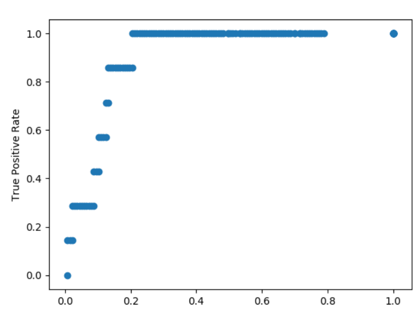
ROC curves for edge prediction



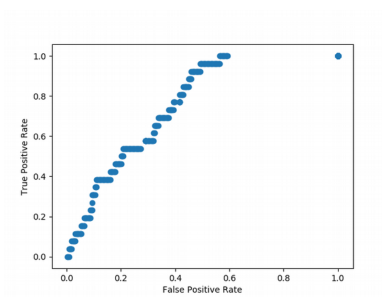
(K = 8)



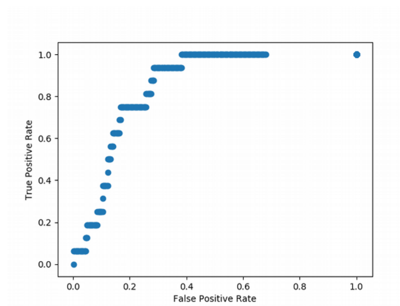
(K = 9)



(K = 12)

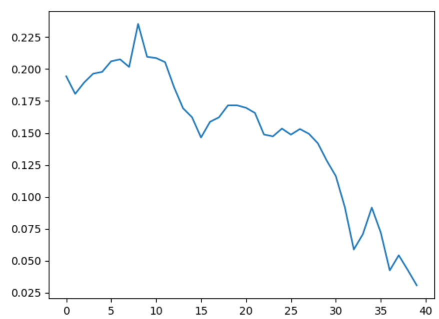


(K = 20, T=10000)

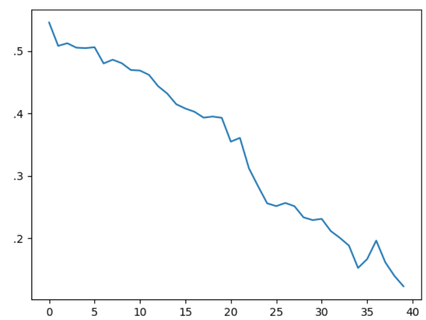


(K = 20, T=20000)

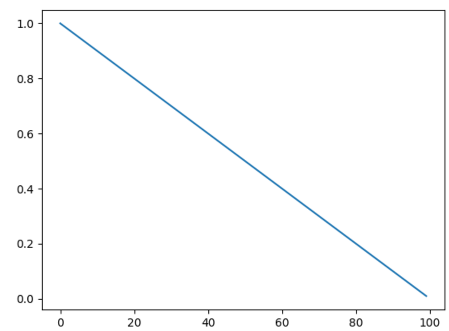
Predicted impulse functions



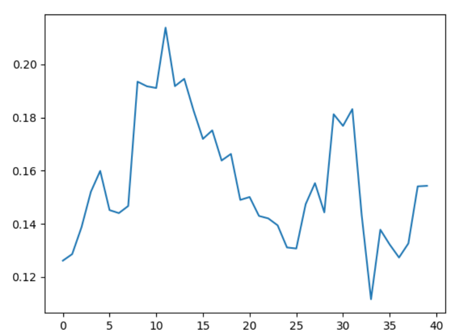
(predicted impulse 1)



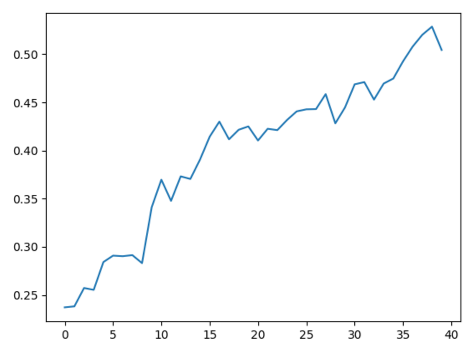
(predicted impulse 2)



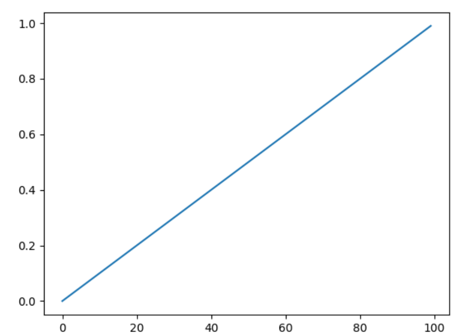
(true impulse)



(predicted impulse 1)



(predicted impulse 2)



(true impulse)

IIX. Analysis

1. The accuracy of edge prediction depends highly on the size of the training data.
2. The γ prior of the background rate has little influence on the accuracy of the edge prediction.
3. The γ prior of the background rate has little influence on the predicted impulse function.
4. The γ prior of the background rate almost directly decides our predicted background rate.

IX. Conclusion

The project started as removing the priors from the previous work. Stated a optimization problem. Then used a convolutional neural network to solve the optimization. The original model could be linearized in the discrete time bin case, which made us able to design a CNN with the same Network Hawkes process.

I designed a 1 layered CNN, which I could extract the impulse and weight information. I also propose a 2 layered CNN with a smaller degree of freedom, which is capable of extracting edge weight, impulse, and impulse basis information.

*** the first convolution is across time, the second is across nodes.

Appendix

Smoothing events $\mu(t)=\gamma\cdot\frac{events}{timebins}+(1-\gamma)kernel(t)*\delta(t)$