Applying CTBNs on Host-level Network Anomaly Detection

Jing Xu and Christian R. Shelton

Department of Computer Science & Engineering
University of California, Riverside, CA

Introduction

In this work, we present an unsupervised learning method for detecting anomalies (or attacks) in network traffic at the individual host level. We use continuous time Bayesian networks (CTBNs) to build a model of normal behavior. We can then use this model to flag connection patterns that differ from this norm.

Continuous Time Bayesian Network

In (Nodelman et al., 2002), CTBN is introduced as a new model to describe stochastic processes over continuous time.
The dynamics of the CTBN system are modeled by a set of Conditional Intensity Matrices (CIM). In contrast to the homogeneous Markov process, which has a single large intensity matrix (IM) for all the variables, CTBN decomposes the overall IM into a set of CIMs attached to each variable. By specifying the instantiation of its parents, the evolving of a variable can be fully described by its CIM.
Network traffic is a temporal process. CTBN, which fits this problem well, is employed in this work.

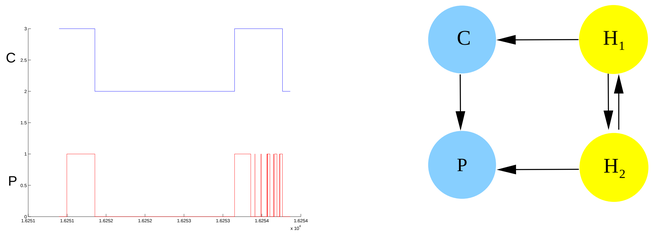


FIGURE 1: Left figure shows a sample trajectory of the network connections (top) and packets (bottom) connecting to a host. Right figure shows a sample CTBN model for network traffic. H_1 and H_2 are latent variables, C is the connection status, and P stands for the packet status.

Related Work

Anomaly detection has been widely studied. Most previous work focuses on content-based or connection-based features like number of data-bytes or number of connections within a window time.
In this work, we propose our method that uses only the connection and packet start/end time. We manage to learn the best fitted CTBN for the connection and packet time, and then use the learned CTBN model for testing.

Approach

Datasets

- DARPA: 35-day traces from the DARPA intrusion detection evaluation dataset. It contains labels and start/end time for all the connections.
  - Given two machines M_1 and M_2, we use two random variables C_{12} and C_{21} to model the connections from M_1 to M_2, and vice-versa. Each variable is binary valued, where 0 means the connection is off, and 1 means on.
- Intel: 31-day traces from the Intel Corp. There is start and end time for all the connections and packets.
  - Use the similar way as above, but we focus on a single host M. Collect all the connections and packets to that host, and use variable C for the connections, and P for the packets. They both take value 1 when on, and 0 when off.

The training and testing approach for both are the same. We show our approach on the DARPA dataset.

Learning Continuous Time Bayesian Networks

We attach 1 or 2 latent variables along with C_{12} and C_{21}, the observed nodes, into the CTBN. We then learn both structure and parameters of the network from the training data using EM.

Testing Using KL-Divergence

The testing procedure can be briefly described as following:

- Denote $q(t)$ to be the empirical distribution of testing data
- Denote $p(t)$ to be the learned CTBN distribution
- Calculate the KL-divergence

$$D(q\|p) = \int q(t) \log \frac{q(t)}{p(t)} dt = \int q(t) \log q(t) dt - \int q(t) \log p(t) dt$$

- Mark as abnormal if KL-divergence is above some threshold

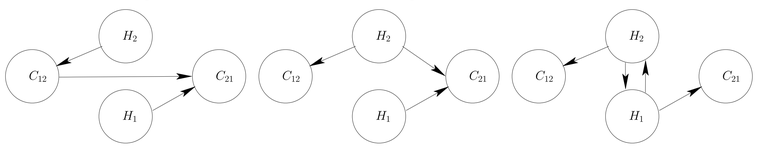
The first term of $D(q\|p)$ is the entropy of a real trajectory, and the second term is the log likelihood of the trajectory under the learned CTBN model. The second term can be calculated by following (Nodelman et al., 2003).
For the first term, we approximate by combining all the observable variables into a large variable C, and then compute the entropy of C's leaving state i and staying at state i:

- $H_q(\tau) = H_{leave}(\tau) + H_{stay}(\tau)$
- Derive the jump matrix S from learned CTBN model, and approximate $H_{leave}(\tau)$ from S and τ
- $H_{stay}(\tau)$ can be approximated by M-Spacing entropy estimators in one dimension (Miller, 2003)

Experimental Results

Results on DARPA Dataset

For the DARPA dataset, we assume two hidden nodes affecting the connections. Using different initializations, we get different structures after learning.



Although initialized differently, most of the learned models have similar performances.

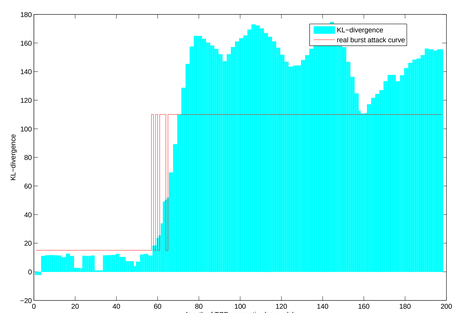


FIGURE 2: KL-Divergence of fixed time sliding windows on a testing trajectory.

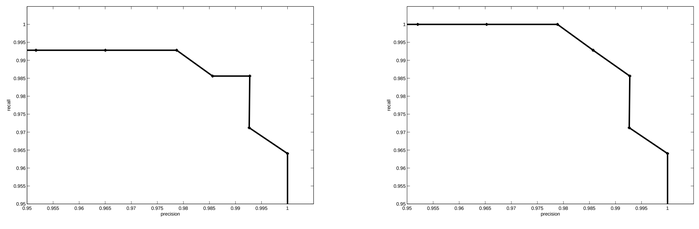


FIGURE 3: Precision recall curves of sliding windows with fixed time (left) and fixed number of transitions (right).

Results on Intel Dataset

For the Intel dataset, we use similar techniques. However, because there are no labels, we cannot derive the precision recall curves. Instead, we use samples from learned CTBN model to measure the ability of model fitting.

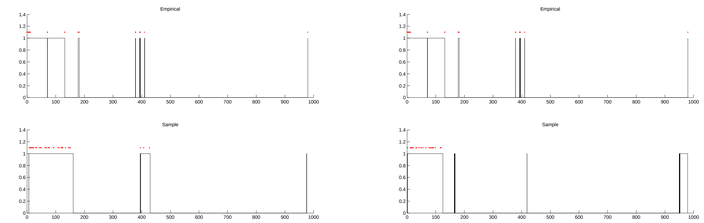


FIGURE 4: A comparison between real and sampled trajectories.

The top figures show two real trajectories from the Intel dataset. The bottom figures are the corresponding sample trajectories from learned CTBN models.

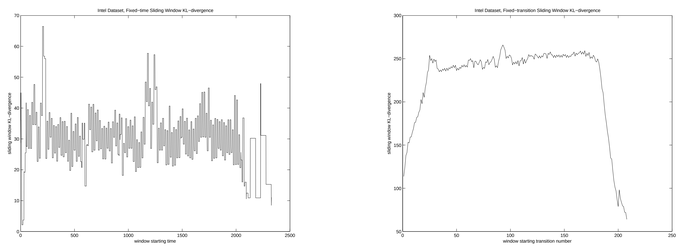


FIGURE 5: KL-Divergence of sliding windows with fixed time (left) and fixed number of transitions (right)

Conclusions

We use CTBN to model a network of 2 observable variables and some hidden variables. We learn both structure and parameters for the model using connection and packet timings on network traffic traces. We use KL-divergence to evaluate the deviation from real model to learned model and do experiments on two datasets.

Acknowledgments

This work is funded by Intel Research.