Graphical Models Project
Final Report
Using Kalman Filters to Learn Correlations

B. Aditya Prakash
Computer Science Department
Carnegie Mellon University
bodicher@andrew.cmu.edu

Abstract

Learning correlations between time-series is an important problem in itself. In this project, we focus on trying to learn whether specific time-series corresponding to BGP routers and specific ‘suspicious’ IPs are correlated or not. We try to learn these correlations from the Kalman Filter parameters. We have implemented the EM algorithm for Kalman Filter learning. After learning the parameters, we run the k-means clustering algorithm on a particular parameter to determine the clusters. Intuitively, time series in the same cluster are correlated. The experiments show promising domain interpretable results.

1 Introduction

The Kalman Filter [1] is an efficient recursive filter that estimates the state of a dynamical system from a series of noisy measurements. It defines a probabilistic model that captures the time evolution and measurement processes. The Kalman Filter is widely used in many real-time tracking applications. Hence, it gives us an ideal tool to learn correlations between time series. We analyze BGP (Border Gateway Protocol) router data corresponding to millions of BGP update messages and try to find out whether corresponding routers and some suspicious IP (Internet Protocol) addresses are dependent (correlated) to each other. BGP is a network protocol in which routers advertise paths to other routers with a view to find the globally shortest routing paths. Hence, BGP routers and IPs whose time series appear to be correlated will provide information on how update messages are being transmitted over the BGP network. This in turn would be valuable in fault discovery or anomalous behavior discovery. For example, spammers generally work together and hence may send updates to routers in a co-ordinated fashion so that more paths are routed through them. Hence, we would expect their corresponding time series to be correlated as well.

2 Data

We examine BGP Monitor data containing 18 million BGP update messages over a period of two years (09/2004 to 09/2006) from the Datapository project [2]. The primary source
of data is Abilene, an academic research network employing Juniper routers running a full-mesh of iBGP sessions. Abilene uses one Zebra monitoring router per point of presence (PoP) to collect BGP updates by establishing an iBGP session as a client. Hence, we have the BGP update data over 10 routers. For the purposes of this project, we simply aggregate the updates by a bin size and form a time series of the bulk number of updates per bin. We do this for each router choosing the bin size to be 600 seconds. We use the same binning technique for the time-series of IPs as well.

3 Traditional Methods

Traditionally, methods such as cross-correlation etc. are generally used to determine whether time-series are correlated or not. But unfortunately in our case, the noise in the measurements coupled with apparent burstiness (see Figure 1) and the length of the time-series make cross-correlations pretty much useless. This is because the cross-correlation metric has a tendency to get drowned by the huge spikes in the data. To see this note that cross-correlation is in essence just a dot-product on the time-series taken at each lag value. Hence, if in a particular lag, shifting one of the time-series results in some spikes getting aligned then that construction would result in a high score. This is undesirable because spikes in the data would most probably correspond to some transient anomalous events in the network world (such as a sudden increase in the number of updates due to a single misbehaving router). Hence, just aligning spikes would not imply a policy induced similarity which is what we are after. Thus, we want to use learning based techniques like Kalman Filters to solve this problem.

4 Kalman Filters

Kalman filters (see Figure 2) are based on linear dynamical systems discretised in the time domain. They are modelled on a Markov chain built on linear operators perturbed by Gaussian noise. The state of the system is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the visible outputs from the hidden state. The Kalman filter may be regarded as analogous to the hidden Markov model, with the key difference that the hidden state variables take
values in a continuous space (as opposed to a discrete state space as in the hidden Markov model). Additionally, the hidden Markov model can represent an arbitrary distribution for the next value of the state variables, in contrast to the Gaussian noise model that is used for the Kalman filter. There is a strong duality between the equations of the Kalman Filter and those of the hidden Markov model. The Kalman Filter assumes linear-Gaussian conditional distributions. The system can be written as [1]:

\[
\begin{align*}
    z_n &= Az_{n-1} + w_n \\
    x_n &= Cz_n + v_n \\
    z_1 &= \mu_0 + u
\end{align*}
\]

where \(x\)'s are the observed variables and \(z\)'s are the hidden variables and the noise terms have the distributions

\[
\begin{align*}
    w &\sim N(w|0, \Gamma) \\
    v &\sim N(v|0, \Sigma) \\
    u &\sim N(u|0, V_0)
\end{align*}
\]

The parameters of the model, denoted by, \(\theta = \{A, \Gamma, C, \Sigma, \mu_0, V_0\}\), can be determined efficiently using maximum likelihood through the EM algorithm. Briefly, in the E step, we solve the inference problem of determining the local posterior marginals for the latent variables, which can be solved efficiently using the sum-product algorithm. For learning, we also need to solve the problem of finding the marginal for a node \(z_t\) given all the observations \(x_1\) to \(x_n\). For temporal data, this corresponds to the inclusion of future as well as past observations. This problem is solved by propagating messages from \(x_n\) to \(x_1\) and combining this information with that obtained during the forward message passing stage used to compute the local posterior marginals. Due to lack of space, we don’t give all the detailed equations, but they can be found in the excellent text by Bishop [1]. In our case, we construct the \(X\) matrix of all the time-series corresponding to the routers/IPs over the entire time-period. Hence \(X\) will be of dimension \(T \times N\) for \(T\) time-ticks and \(N\) sources.

Figure 2: Snapshot of Graphical Model for a Kalman Filter: Time \(t - 1\) and \(t\)

5 Using the model to detect correlations

Once the model has been learnt, we now turn to use that model to understand how the routers are correlated. One way could be using the \(C\) matrix (size \(N \times H\)) as a matrix where the rows correspond to \(H\)-dimensional points. Here \(H\) is the hidden dimension i.e. the number of hidden variables in the system. For purposes of our study, we assume that \(H = N\), that there are as many hidden dimensions as time-series’. This is a fair assumption as apriori we don’t know whether the time-series are generated from a similar control variable. Instead, if we have strong prior knowledge (domain based or otherwise) we can set it lesser as well. We then use a clustering algorithm such as K-means to cluster these points.
5.1 Intuition

To see why the above works, recall that the $C$ matrix is essentially just a projection matrix of the latent variables over the observed variables. Hence, assuming the latent variables are independent, correlated time series’ will show up as correlated rows of the $C$ matrix. K-means is just a simple algorithm to cluster the row-points.

Another justification becomes clear when we look at the various subspace methods to learn the model parameters. The most straightforward technique which relies on SVD (Singular Value Decomposition) is given in Overschee and Moor [3]. An estimate of $C$ satisfying certain canonical assumptions [4] would be $C = U$, where $X \approx U \Sigma V^T$, the SVD decomposition of the observation matrix. Hence, intuitively, $C$ is just the matrix of top-K left-singular vectors of $X$. And clustering $U$ is a well-known method in spectral clustering to cluster graphs after transforming their laplacians to the singular-vector space.

Another way to think about it comes from the fact that PCA can be used a relaxation to the K-means algorithm [5]. Ideally we would want to run the K-means algorithm (using some distance metric) on the original time-series itself. But this is prohibitive due to the length of the time-series (high-dimensionality curse). Note that PCA is essentially derived from the SVD formulation. Hence, instead on doing K-means on the original data matrix, we are transforming each time-series to a point in the PCA space, and then clustering them. Points (time-series) having similar components on the PCA will be grouped together by any clustering algorithm acting in that space. Hence, we run K-means on the $C$ matrix to recover clusters of time-series.

6 Experiments

![Figure 3: Result for Routers clustering $\lambda = 0.3$](image)

We used the Kalman Filter EM learning algorithm (Section 4) implemented in MATLAB. We use the MATLAB in-built function for performing K-means on the $C$ matrix. We performed several different experiments which are described next.

6.1 Routers

As mentioned in Section 2, we construct a time series corresponding to each router and then learn a Kalman Filter for the consolidated matrix. Due to the absence of a ground-truth clustering, we approached the problem in two ways.
6.1.1 Recursive bi-clustering

In this method, at each stage, we cluster the points into 2 clusters and then cluster the 2 clusters recursively separately. We do this till we are left with clusters of size 2. These then correspond to pairs of routers which are correlated. The results after recursive bi-clustering are shown below (routers in the same cluster):

- washington and nyc
- atlanta and houston
- denver and kentucky
- seattle and los angeles
- chichago and indianapolis

6.1.2 Regularizing Score

Here, we use a regularizer to choose the simplest model (the fewest clusters) that can describe the data well. In particular, we used $\lambda \ast (\#clusters)$ as the regularizer, where $\lambda$ is a given constant. So in effect we try to minimize the regularized score $\delta$,

$$\delta = \Sigma_d(x, centers) + \lambda \ast (\#clusters)$$ (7)

We ran K-means using different number of clusters and plotted the different values of $\delta$. A valid question is that the behavior of such a plot can change depending on $\lambda$. Thus, we try to identify regions of $\lambda$ which show stable (consistent) behavior w.r.t. the number of clusters. We then choose the value which minimizes the score in the region.

The result is shown in Figure 3 $\lambda$ value .3. In fact, we found that the score was pretty stable for values ranging from .2 to .6. Hence, we take the number of clusters to be equal to 4. The clusters themselves are:

- washington and nyc
- chichago, indianapolis, denver and kentucky
- seattle and los angeles
- atlanta and houston
6.1.3 Discussion

Note that the results in both the bi-clustering and regularizer case match very well to the notion that geographically closer routers tend to be more correlated than others. Specifically, for example, in the regularized score case, the router clusters correspond to different network regions in the country (East, Central, West, South).

This is because the BGP routing protocol itself tries to find shorter routes which results in packets being sent locally to nearby routers rather than routers far away. Other reasons could be the service provider who owns these routers and hence routers with same service providers will tend to run similar protocol details.

6.2 Suspicious IPs

We had previously developed an algorithm to find particular IPs which ‘persistently’ send a regular (near-constant) number of updates for a long period of time (2-3 months) [6]. Clearly, these IPs are suspicious and reasons for such behavior can range from spamming, hijacking to bugs in the routers. This is because, in equilibrium, BGP routers should stabilize and the exchange of messages should be minimum.

Hence, we look into these IPs specifically by constructing time-series (similar to ones for routers). We then cluster them as above (we use the regularized score formulation here). The result is in Figure 4 for λ value .3. The number of clusters as suggested from these plots is 3. The clusters we found were:

- 207.157.115.0, 192.211.42.0, 216.109.38.0
- 192.94.104.0
- 222.200.236.0, 222.203.64.0, 222.202.96.0

6.2.1 Discussion

Interestingly, each cluster of these IPs corresponded to a single entity! Specifically, the first group of IPs all belong to the Alabama Supercomputing Network, the second to the UNE Medical Center and the third group to CERNET, China. Moreover, the Alabama network was contacted and they attributed the anomaly to changes while transitioning address space causing BGP to route flap [6]. This demonstrates that the clusters generated are meaningful and do cluster time-series’ which have policy induced similarities. This can be due to problems/bugs in the routers or even a co-ordinated attack in case of spammers.

7 Conclusions

We give a way of clustering time-series after learning a consolidated Kalman Filter for them. This involves clustering the \( C \)-matrix by considering the rows as data points. The efficacy of this method was demonstrated by applying it to real-world BPG data where the clusters corresponded to domain interpretable concepts.

Note that these clusters were found by using nothing more than just the aggregated number of updates per bin size - the particular details of the updates themselves was not considered at all - which routes were advertised, which were withdrawn etc. This is encouraging as information is being learnt without looking at the details of the protocol itself. Here, BGP is a very complex protocol and hence, such methods will help towards automated discovery of anomalies in BGP systems.
8 Future Work

There are lots of avenues for future work. Note that at present we have looked at only one aggregation level - 600 seconds. Maybe more correlations will be visible at other granularity levels. Hence, one idea would be to try this out at different bin sizes. Also, we can look into applying this method to other datasets and settings as well. Specifically, we can use it in analyzing motion capture datasets. These datasets record motions such as walking, running, jumping etc. as time-series. Can we just by looking at the time-series, cluster all walking time-series together, all running motions together? This can be an exciting application.

Other than this, there can be other ways as well in which learnt Kalman Filters be used for detection of correlations. One way could be to use the model learnt from say $k$ routers to predict the next time tick of a remaining router and then measure the success.

References