Anomaly Detection Report

Daniel Hsu

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250

November 3, 2017

Abstract

In this paper, we use variational recurrent neural network to investigate the anomaly detection problem on graph time series. The temporal correlation is modeled by the combination of recurrent neural network (RNN) and variational inference (VI), while the spatial information is captured by the graph convolutional network. In order to incorporate external factors, we use feature extractor to augment the transition of latent variables, which can learn the influence of external factors. With the target function as accumulative ELBO, it is easy to extend this model to on-line method. The experimental study on traffic flow data shows the detection capability of the proposed method.

1   Introduction

When datasets become large and complex, it is often more important and interesting to know what stands out in the data than to learn about its general behavior. The anomaly detection, an important branch of data mining, is to discover rare occurrences in datasets. This field has wide applications in finance, security, health care, law enforcement, and many others.

During past decades, there has been a large body of research work on anomaly detection, such as detecting network failure or network intrusion [1], inspecting card and telecommunications fraud [2], classifying Web and email spam [3], monitoring data-center [4], detecting malware/spyware [5], and image/video surveillance [6]. Anomaly detection has been studied in a variety of data domains including high-dimensional data [7], uncertain data [8], streaming data [9], network data [10], and time series data [11].

Different types of data always require techniques from different domains. Inspecting anomalies in time series needs to examine the behavior of the data across time. Thus, several models were proposed in the statistics literature, which includes autoregressive integrated moving average (ARIMA), autoregressive moving average (ARMA), CUmulative SUM Statistics (CUSUM), vector autoregression (VARMA), exponentially weighted moving average, etc [14] [15] [16]. In this work, we focus on multi-dimensional time series in streams, where temporal dependencies are highly non-linear and data is analyzed in an online way. Yamanishi et al. [17] present SmartSifter algorithm for time series in streams, which employs an online discounting learning algorithm to incrementally learn the probabilistic mixture model, with a decay factor to account for anomalies. Other than the online discounting methods, large number of methods use dynamically maintained cluster models for computing outliers from data streams such as [18].

Because of the rising applications of social network, wearable devices and sensor networks, the analysis of structured (graph) time series becomes more and more important. In this work, we also incorporate graph structures of data into anomaly detection. Ide et al. [19] proposed an eigenvector-based method for anomaly detection over graph streams. A similar method is also proposed by Akoglu et al. [20] to spot anomalous points in time at which many agents in an agent network change their behavior in a way such that it deviates from the norm. However, these methods only learn graph behavior in the time interval h, and cannot work well on time series with long and complex repeating patterns.

In this paper, we combine variation inference (VI) and recurrent neural network (RNN) to model the graph time series, and detect anomalies based on log likelihood. RNNs can be employed for a wide range of tasks as they inherit their flexibility from plain neural networks. This includes universal approximation capabilities, since RNNs are capable of approximating any measurable sequence to sequence mapping and have been shown to be Turing complete [21]. However, the internal transition structure of the standard RNN is entirely deterministic. There is recent evidence that when complex sequences are modeled, the performances of RNNs can be dramatically improved when uncertainty is included in their hidden states [22] [23] [24]. In [23], authors propose to extend the variational autoencoder (VAE) into a recurrent framework for modeling complex sequences which is called variational RNN (VRNN). The proposed time series model is based on VRNN.

In order to incorporate the structural information, the graphical convolution neural network (GCNN) is applied to extract features from graphical data. We use K-localized filter to incorporate and utilize the local information on the graph. In practice, time series are influenced by external features, which should be considered in outliers detection. Our model is conditioned on the external features as extra input. The experiments on human wearable senor data and traffic flow data show the detection capability of the proposed model.

2 Preliminary

2.1 Recurrent Neural Network

RNNs are discrete-time statespace models trainable by specialized weight adaptation algorithms. The input to RNN is a variable-length sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ which can be recursively processed. And when processing each symbol, RNN maintains its internal hidden state $\boldsymbol{h}$. The operation of RNN at each timestep t can be formulated as

$$\boldsymbol{h}_t = f_\theta(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

where f is the deterministic state transition function and θ is the parameter of f. RNNs model sequences based on the parameterization a factorization of the joint sequence probability distribution as a product of conditional probabilities such that:

$$
\begin{aligned}
p(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T) &= \prod_{t=1}^{T} p(\boldsymbol{x}_t | \boldsymbol{x}_{<t}) \\
p(\boldsymbol{x}_t | \boldsymbol{x}_{<t}) &= g_\tau(\boldsymbol{h}_{t-1})
\end{aligned}
\tag{1}
$$

where g is the function mapping the hidden state to the output distribution conditioned on previous observations, which is parameterized by τ.

The representational power of an RNN is limited by the output function g in (1). With the deterministic transition function f, the joint probabilities $p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ which can be represented by the RNN are determined by the function g.

In this work, the function f can be realized by long short-term memory [25]. Since the state transition is performed in a deterministic way, when modeling high-dimensional and structured time series, the variability embedded in the hidden states cannot be learned and expressed. That's because the conditional output probability density is the only source of variability. So, it is necessary to introduce stochasticity into the hidden state transition. And in analyzing some highly structured time series, the hidden state $\boldsymbol{h}_t$ of RNNs is not expressive enough to capture the state transition dynamics. Some previous work have introduced extra latent variables to model the output functions [24] [23] [27]. Different from other work, following [23], this work makes the prior distribution of the latent random variable $\boldsymbol{z}_t$ at timestep t dependent on all the preceding inputs via the RNN hidden state $\boldsymbol{h}_{t-1}$ which can improve the representational power of the model.

2.2 Variational Autoencoder

Different from previous work, we detect anomalies based on the conditional probability of time series conditioned on previous data. However, it is difficult to compute the probability of complex time series. Recently variational autoencoder (VAE) [28] has been shown to be a powerful tool to approximate the intractable complex posterior in the data space. The VAE introduces a set of latent random variables $\boldsymbol{z}$, designed to capture the variations underlie the observed variables $\boldsymbol{x}$. Typically VAE models the conditional probability $p(\boldsymbol{x}|\boldsymbol{z})$ as a highly flexible function approximator such as a neural network, which makes the inference of the posterior $p(\boldsymbol{z}|\boldsymbol{x})$ intractable. Thus VAE uses a variational approximation $q(\boldsymbol{z}|\boldsymbol{x})$ of the posterior, which introduces the evidence lower bound (ELBO):

$$\log p(\boldsymbol{x}) \geq -\mathrm{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) + \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] \qquad (2)$$

where $\mathrm{KL}(P|Q)$ is the Kullback-Leibler divergence between two distributions P and Q.

In [28], the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ is a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$ and the mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}^2$ are modeled as the output of neural networks of $\boldsymbol{x}$. The prior $p(\boldsymbol{z})$ is assumed to simple standard Gaussian distribution. The training process is to maximize the ELBO, which can yield optimal selection of parameters for generative model $p(\boldsymbol{x}|\boldsymbol{z})$ and inference model $q(\boldsymbol{z}|\boldsymbol{x})$. Based on re-parameterizing trick, we formulate $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$ and rewrite:

$$\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[\log p(\boldsymbol{x}|\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon})]$$

where $\boldsymbol{\epsilon}$ is drawn from standard Gaussian distribution. Then, both generative and inference model can be trained by standard backpropagation technique based on gradient descent.

3 Anomaly Detection Model

In this work, our model is based on variational recurrent neural network (VRNN) [23], which is a recurrent extension of VAE. And we extend it to anomaly detection on graph time series by graph Laplacian transform. Moreover, in order to better model the practical situation, we incorporate external features as extra input.

3.1 Graph Laplacian Transform

The graph time series always demonstrate strong spatial dependency. Each node is strongly influenced by its neighbors. For example, on highways, the sensors are installed every 1-2 miles, and the traffic flow of adjacent sensors are highly correlated. Moreover, the wearable sensors on

human body are always dependent on the neighboring sensors, and this dependency is different in different activities. However, VRNN cannot explicitly model such spatial dependencies. In this work, VRNN is augmented by modeling spatial dependency in an efficient way.

In order to model spatial dependency, in vertex domain, the hidden state transition of certain node should incorporate the hidden states of neighboring nodes weighted by correlation coefficients. However, it is quite time-consuming to consider the pairwise dependencies of every vertex in a large graph. So, it is difficult to express a meaningful translation operator in the vertex domain [31]. Therefore, we transform the graph time series from vertex domain into spectral domain by graph Laplacian transform.

Assume that the graph time series are defined on undirected and connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$, where $\mathcal{V}$ is a finite set of $|\mathcal{V}| = n$ vertices, $\mathcal{E}$ is a set of edges and $W \in \mathbb{R}^{n \times n}$ is a weighted adjacency representing the weight between two vertices. Define the snapshot of graph time series at time t, $\boldsymbol{x}_t \in \mathbb{R}^n$, as a signal defined on the nodes of the graph, where the i-th element of $\boldsymbol{x}_t$ is its value at the i-th node. Define the diagonal matrix $D \in \mathbb{R}^n$ as $D_{ii} = \sum_j W_{ij}$. The graph Laplacian operator $\boldsymbol{y} = \boldsymbol{L}\boldsymbol{x}$ [32] is the one-step diffusion of the signal defined on the graph, where the Laplacian matrix $\boldsymbol{L}$ is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \in \mathbb{R}^n$ and normalized version is $\boldsymbol{L} = \boldsymbol{I}_n - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$.

Diagonalizing the Laplacian matrix $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, where $\boldsymbol{U} \in \mathbb{R}^n$ and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ are eigenvectors and eigenvalues of the Laplacian matrix $\boldsymbol{L}$, the graph Fourier transform is defined as $\hat{\boldsymbol{x}} = U^T\boldsymbol{x}$ and its inverse as $\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$. The spectral filtering of graph signals [33] is defined as:

$$\boldsymbol{y} = g_\theta(\boldsymbol{L})\boldsymbol{x} = g_\theta(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T)\boldsymbol{x} = \boldsymbol{U}g_\theta(\boldsymbol{\Lambda})\boldsymbol{U}^T\boldsymbol{x}$$

where $g_\theta(\boldsymbol{\Lambda}) = \mathrm{diag}(\theta_1\lambda_1, \ldots, \theta_n\lambda_n)$ and $\theta \in \mathbb{R}^n$ is a vector of Fourier coefficients. However, this filter is enough, since the Laplacian operator is local and working on 1-hop neighborhoods. The kth power of Laplacian operator is supported by exactly k-hop neighbors [34], representing the signals on the graph at different scales. Since the kth power of Laplacian matrix is expensive to compute, We apply the Chebyshev polynomial expansion [33] in the following:

$$\boldsymbol{y} = g_\omega(\boldsymbol{L})\boldsymbol{x} = \sum_{k=0}^{K-1} \omega_k \boldsymbol{L}^k \boldsymbol{x} = \boldsymbol{U}\sum_{k=0}^{K-1} \omega_k \boldsymbol{\Lambda}^k \boldsymbol{U}^T\boldsymbol{x} \approx \sum_{k=0}^{K-1} \tilde{\omega}_k T_k(\tilde{\boldsymbol{\Lambda}})\boldsymbol{x} \tag{3}$$

where $g_\omega(\boldsymbol{L})$ is the learned filter based on Laplacian matrix parameterized by $\omega \in \mathbb{R}^K$, and $T_k(\tilde{\boldsymbol{\Lambda}}) \in \mathbb{R}^{n \times n}$ is the Chebyshev polynomial of order k with $\tilde{\boldsymbol{\Lambda}} = 2\boldsymbol{\Lambda}/\lambda_{\max} - \boldsymbol{I}$. The computational complexity is reduced from $O(|\mathcal{V}|^2)$ to $O(K|\mathcal{E}|)$.

3.2 Generative Model

In this model, VAE is applied in a recurrent setting. At each timestep, different from previous work [30], the prior of latent variables is dependent on the hidden state $\boldsymbol{h}_{t-1}$ of RNN, which makes the output function incorporate the temporal structure of the time series. Then we have:

$$\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{0,t}, \mathrm{diag}(\boldsymbol{\sigma}_{0,t}^2)), \text{ where } [\boldsymbol{\mu}_{0,t}, \boldsymbol{\sigma}_{0,t}] = \varphi_\tau^{\mathrm{prior}}(\boldsymbol{h}_{t-1}) \tag{4}$$

where $\boldsymbol{\mu}_{0,t}$ and $\boldsymbol{\sigma}_{0,t}$ denote the mean and variance of the prior distribution. Then, the generating distribution will be conditioned on both latent variable $\boldsymbol{z}_t$ and hidden state $\boldsymbol{h}_{t-1}$ such that:

$$\boldsymbol{x}_t|\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{\mu}_{x,t}, \mathrm{diag}(\boldsymbol{\sigma}_{x,t}^2)), \text{ where } [\boldsymbol{\mu}_{x,t}, \boldsymbol{\sigma}_{x,t}] = \varphi_\tau^{\mathrm{dec}}(\varphi_\tau^{\boldsymbol{z}}(\boldsymbol{z}), \boldsymbol{h}_{t-1}) \tag{5}$$

where $\boldsymbol{\mu}_{x,t}$ and $\boldsymbol{\sigma}_{x,t}$ denote the mean and variance of the generating distribution, $\varphi_\tau^{\mathrm{prior}}$ and $\varphi_\tau^{\mathrm{dec}}$ can be realized by neural networks. Since the latent variables $\boldsymbol{z}$ are assumed to be independent

of each other at the same time, the feature extractor $\varphi_\tau^{\boldsymbol{z}}(\cdot)$ can introduce dependencies to better model the sequential variation. Inside the RNN, the hidden state is updated as follows:

$$\boldsymbol{h}_t = f_\theta(\varphi_\tau^{\boldsymbol{x}}(\boldsymbol{x}_t), \varphi_\tau^{\boldsymbol{z}}(\boldsymbol{z}_t), \boldsymbol{h}_{t-1}) \tag{6}$$

where transition function f parameterized by θ is realized by LSTM cell [25], which can capture both long and short-term temporal dependencies. In order to capture the spatial dependencies, the function $\varphi_\tau^{\boldsymbol{x}}(\cdot)$ is modeled by graph spectral filter (3) with Chebyshev polynomial expansion, i.e. $\varphi_\tau^{\boldsymbol{x}}(\boldsymbol{x}) = \sum_{k=0}^{K-1} \tilde{\omega}_k T_k(\tilde{\boldsymbol{\Lambda}})\boldsymbol{x}$. From (6), we know that hidden state $\boldsymbol{h}_t$ is dependent on $\boldsymbol{x}_{\leq t}$ and $\boldsymbol{z}_{\leq t}$. The distributions (4) and (5) define $p(\boldsymbol{z}_t|\boldsymbol{x}_{<t}, \boldsymbol{z}_{<t})$ and $p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t})$ respectively. We can see that the parameterization of this generative model is motivated by the factorization of joint probability as below:

$$p(\boldsymbol{x}_{\leq T}, \boldsymbol{z}_{\leq T}) = \prod_{t=1}^{T} p(\boldsymbol{x}_t|\boldsymbol{z}_{\leq t}, \boldsymbol{x}_{<t})p(\boldsymbol{z}_t|\boldsymbol{x}_{<t}, \boldsymbol{z}_{<t}) \tag{7}$$

3.3 Conditional Inference Model

Denote the external feature at time t, such as holiday and meteorological data, as a extra input vector $\boldsymbol{e}_t$. It is first processed by a two-layer neural networks denoted by $\varphi_\tau^{\text{ext}}(\boldsymbol{e}_t)$. Then it is directly added to the mean of latent variables $\boldsymbol{\mu}_{z,t}$, which has no influences on the variance $\boldsymbol{\sigma}_{z,t}$. Based on Gaussian assumption, the approximate posterior is a function of input $\boldsymbol{x}_t, \boldsymbol{e}_t$ and hidden state $\boldsymbol{h}_{t-1}$ as below:

$$\boldsymbol{z}_t|\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_{z,t} + \varphi_\tau^{\text{ext}}(\boldsymbol{e}_t), \text{diag}(\boldsymbol{\sigma}_{z,t}^2)), \text{ where } [\boldsymbol{\mu}_{z,t}, \boldsymbol{\sigma}_{z,t}] = \varphi_\tau^{\text{enc}}(\varphi_\tau^{\boldsymbol{x}}(\boldsymbol{x}), \boldsymbol{h}_{t-1}) \tag{8}$$

where $\boldsymbol{\mu}_{z,t}$ and $\boldsymbol{\sigma}_{z,t}$ denote the mean and variance of the approximate posterior. Conditioning on $\boldsymbol{h}_{t-1}$, the posterior follows the factorization:

$$q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T}) = \prod_{t=1}^{T} q(\boldsymbol{z}_t|\boldsymbol{x}_{\leq t}, \boldsymbol{z}_{<t}) \tag{9}$$

3.4 Detection Objective Function

Applying ELBO at each timestep, based on factorizations (7) and (9), we have the accumulative ELBO as below:

$$\mathbb{E}_{q(\boldsymbol{z}_{\leq T}|\boldsymbol{x}_{\leq T})}\left[\sum_{t=1}^{T}(-\text{KL}(q(\boldsymbol{z}_t|\boldsymbol{x}_{\leq T}, \boldsymbol{z}_{<T})\|p(\boldsymbol{z}_t|\boldsymbol{x}_{<T}, \boldsymbol{z}_{<T})) + \log p(\boldsymbol{z}_t|\boldsymbol{x}_{<T}, \boldsymbol{z}_{\leq T}))\right] \tag{10}$$

which is learning objective function during the training. By maximizing (10), we first train this model on time series data without anomalies. The optimization problem is solved by ADAM [29] algorithm. The optimal generative and inference model can be learned simultaneously. During the testing phase, at time t, we detect the anomaly based on the ELBO bound as below:

$$b(\boldsymbol{x}_t) := \mathbb{E}_{q(\boldsymbol{z}_t|\boldsymbol{x}_t)}\left[-\text{KL}(q(\boldsymbol{z}_t|\boldsymbol{x}_t)\|p(\boldsymbol{z}_t|\boldsymbol{x}_t)) + \log p(\boldsymbol{z}_t|\boldsymbol{x}_t)\right] \tag{11}$$

which is an approximate to the log likelihood.

3.5 Likelihood Ratio Test

Different from previous work [30], we can locate anomalies on the graph via likelihood ratio test (LRT). In statistics, people use the likelihood ratio test to compare the fit of two models, one of which (the null model) is a special case of (or "nested within") the other (the alternative model). This often occurs when testing whether a simplifying assumption for a model is valid, as when two or more model parameters are assumed to be related. Each of the two compared models, the null model and the alternative model, is separately fitted to the data with the log-likelihood recorded [35]. The test statistics (often denoted by Λ) is twice negative the difference in these log-likelihoods:

$$\Lambda = -2\log \frac{\text{likelihood for null model}}{\text{likelihood for alternative model}}$$

Whether the alternative model fits the data significantly better than the null model can be determined by deriving the probability or p-value of the obtained difference Λ.

When applying LRT to a single node $r_i i$ on the graph, based on the generative model, we assume its value follows Gaussian distribution P with mean $\mu_{x,t}^{(i)}$ and variance $\sigma_{x,t}^{(i)}$ calculated in (5). Suppose the observed value at node r_i is x_i, the likelihood ratio is:

$$\Lambda(r_i) = -2\log \frac{P(x_i|\Theta)}{\sup\{P(x_i|\Theta')\}}$$

where Θ' is the new parameter changing over Θ that fits the observed data best; sup denotes the supreme function that finds the maximizer of $P(x_i|\Theta')$ over Θ'. For simplicity, we assume the mean and variance in the Θ' has same proportion as Θ. The anomalous degree od of this test is calculated as below:

$$\text{od} = \chi^2_{\text{cdf}}(\Lambda, \text{df})$$

where χ^2_{cdf} denotes the cumulative density function of Chi-Square distribution; df is the degree of freedom, which means the the number of free parameters of the null model and the alternative model, respectively. The nodes with od larger than a given threshold are likely to be anomalous.

For example, suppose a certain node r_i at time t follows Gaussian distribution $P \sim \mathcal{N}(100, 200)$. Assume the its observed value at time t is 30. Then we calculate the degree of anomaly at r_i as below. The likelihood of the null model is $L_{\text{null}} = P(30|\mathcal{N}(100, 200))$. In order to achieve the supreme of likelihood, the alternative model should have mean 30 and variance 60. The likelihood of alternative model is $L_{\text{alter}} = P(30|\mathcal{N}(30, 60))$. According to above equations, the anomalous degree is calculated as below:

$$\begin{aligned} \Lambda(r_i) &= -2\log \frac{L_{\text{null}}}{L_{\text{alter}}} = 25.70 \\ \text{od} &= \chi^2_{\text{cdf}}(25.70, \text{df} = 1) = 0.999 \end{aligned}$$

Normally, we set the threshold of anomalous degree to be 0.95.

4 Experimental Study: Beijing Taxi Flows

In this experiment, the city of Beijing is partitioned into a 32×32 grid map based on the longitude and latitude where a grid denotes a region. The dataset contains Beijing's taxicab's trajectories and meteorological data [36]. At t-th time interval, the inflow and outflow in all 32×32 regions can be denoted as a tensor $\boldsymbol{x}_t \in \mathbb{R}^{2 \times 32 \times 32}$ where $(\boldsymbol{x}_t)_{0,i,j}, (\boldsymbol{x}_t)_{1,i,j}$ represent the amount of taxi inflow and outflow in region (i, j). The inflow matrix is shown as below.
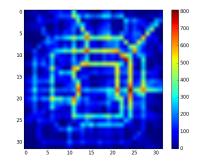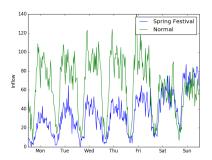
Figure 1: Beijing City Map.

Figure 2: Inflow in every region of Beijing.

The traffic flows are always influenced by external factors, such as weather and holidays. There are 16 types of weather conditions, such as sunny and rainy. The holidays include all public holidays in China, with total number of 41. The figures below show the influence of Chinese Spring Festival and thunderstorm on the CBD of Beijing, i.e., grid (18,22) on the map.
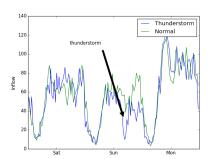

Figure 3: Spring Festival: Feb 8-14 2016, Nor-mal: Feb 15-21 2016.

Figure 4: Thunderstorm: Aug 10-12 2013, Nor-mal: 17-19 2013.

The other external factors are weekday, temperature and windspeed. The weekdays are ranging from Monday to Sunday. The temperature is from $-24.1°C$ to $41.0°C$. And windspeed is within 0 and 48.6mph. For examples, traffic flows in weekends are usually lower than working days. And strong wind can make people stay at home and reduce the traffic. In implementation, we use one-hot binary vectors to represent weekday, holiday and weather, and use float numbers to represent temperature and windspeed.

The dataset contains traffic flows ranging in two time intervals, i.e., 1st Mar. 2015 to 30th Jun. 2015, and from 1st Nov. 2015 to 10th Apr. 2016. The traffic flow in every region is scaled into $[0, 1]$. Each data point shows the traffic flow in 30 minutes. The first 80% data points are used for training, and the rest is for testing. Since the external factors are already given, we assume that the training set doesn't have any anomalies.

Anomalies are manually added into testing set. All anomalies can be categorized into global anomaly and local anomaly. The global anomaly is always outstanding from global perspective, which has significant difference from normal values and covers a relatively wide range. The local anomalies are bound between the seasonal minimum and maximum and may not appear to be

outliers from a global perspective. We evaluate the detection performances on the following types of anomalies ($k = 0, 1$, anomaly center $(p, q) \in [32, 32]^2$):

- (a) global mean shift (GMS): $(\boldsymbol{x}_t)'_{k,i,j} = (\boldsymbol{x}_t)_{k,i,j} + \mu$ with $\mu \in [0.8, 1.3], (i, j) \in [p \pm 3, q \pm 3]^2$ and $t \in [30, 60]$

- (b) local mean shift (LMS): $(\boldsymbol{x}_t)'_{k,i,j} = (\boldsymbol{x}_t)_{k,i,j} + \mu$ with $\mu \in [0.4, 0.6], (i, j) \in [p \pm 1, q \pm 1]^2$ and $t \in [5, 10]$

- (c) global amplitude change (GAC): multiplying multiple dimensions of the data with $1 + (\boldsymbol{g}_t)_{k,i,j}$ where function $(\boldsymbol{g}_t)_{k,i,j}$ is a Gaussian random variable with 0 mean and $10(\boldsymbol{\sigma}_{x,t})_{k,i,j}$, and $(i, j) \in [p \pm 3, q \pm 3]^2, t \in [30, 60]$

- (d) local amplitude change (LAC): multiplying single dimensions of the data with $1 + (\boldsymbol{g}_t)_{k,i,j}$ where function $(\boldsymbol{g}_t)_{k,i,j}$ is a Gaussian random variable with 0 mean and $6(\boldsymbol{\sigma}_{x,t})_{k,i,j}$, and $(i, j) = (p, q), t \in [10, 20]$.

where $\boldsymbol{\sigma}_{x,t}$ is the generated variance in (5). The detection examples are shown as below. The y-axis label shows the index of the corresponding dimension. The red-dotted points are detected outliers.
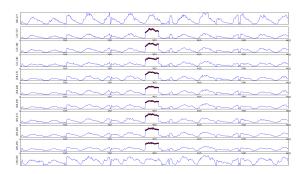

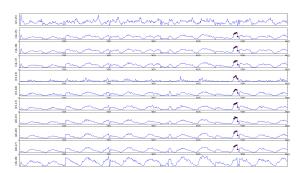Figure 5: Global Mean Shift: $\mu = 0.9, k = 0, (p, q) = (18, 18)$, duration is 30 data points.

Figure 6: Local Mean Shift: $\mu = 0.5, k = 0, (p, q) = (15, 16)$, duration is 10 data points.
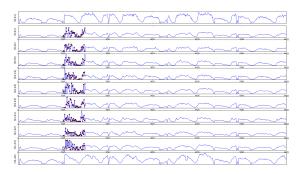
8

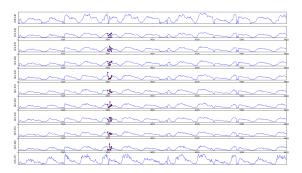Figure 7: Global Amplitude Change: $\sigma = 0.8, k = 0, (p, q) = (10, 22)$, duration is 50 data points.

Figure 8: Local Amplitude Change: $\sigma = 0.4, k = 0, (p, q) = (22, 15)$, duration is 10 data points.

The following table shows the detection performance on four types of anomalies, where each type of anomaly is tested 200 times in different locations.

Table 1: Results of the proposed detection algorithm on different anomalies. We use average precision (AP) and area under ROC curve (AUC) as performance measures.

	GMS	LMS	GAC	LAC
AP	1.00	0.91	0.95	0.88
AUC	1.00	0.93	0.89	0.92

5 Conclusion

This paper shows the the application of variational inference and recurrent neural networks on time series anomaly detection. With comprehensive experiments, we prove that the proposed model can have competitive performance on anomaly detection in graph time series. The spectral filter can better learn the spatial information, and the variational recurrent neural networks can model the temporal correlation. The multi-variable Gaussian model is sensitive enough to

outliers. Different from previous work, based on likelihood ratio test, our model can not only detect anomalous time points, but also localize the anomalous nodes (positions) on the graph. In the future, we plan to apply this model to detect video outliers.

References

[1] Q. Ding, N. Katenka, P. Barford, E. D. Kolaczyk, and M. Crovella. Intrusion as (anti)social communication: characterization and detection. In Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Beijing, China, pages 886-894. ACM, 2012.

[2] C. Cortes, D. Pregibon, and C. Volinsky. Communities of interest. Intelligent Data Analysis, 6(3):211-219, 2002.

[3] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: web spam detection using the web topology. In Proceedings of the 30th International Conference on Research and Development in Information Retrieval (SIGIR), Amsterdam, pages 423-430. ACM, 2007.

[4] L. Li, C. Liang, J. Liu, S. Nath, A. Terzis, and C. Faloutsos. Thermocast: A cyber-physical forecasting model for data centers. In Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA. ACM, 2011.

[5] L. Invernizzi and P. M. Comparetti. Evilseed: A guided approach to finding malicious web pages. In IEEE Symposium on Security and Privacy, pages 428-442, 2012.

[6] B. Krausz and R. Herpers. MetroSurv: detecting events in subway stations. Multimedia Tools and Applications, 50(1):123-147, 2010.

[7] C. C. Aggarwal and P. S. Yu, Outlier detection for high dimensional data, SIGMOD Rec., vol. 30, pp. 37-46, May 2001.

[8] C. C. Aggarwal and P. S. Yu. Outlier detection with uncertain data, in Proc. 2008 SIAM Int. Conf. SDM, pp. 483-493.

[9] C. Aggarwal and K. Subbian. Event detection in social streams, in Proc. 12th SIAM Int. Conf. SDM, 2012, pp. 624-635.

[10] C. C. Aggarwal, Y. Zhao, and P. S. Yu, Outlier detection in graph streams, in Proc. 27th ICDE, Hannover, Germany, 2011, pp. 399-409.

[11] J. P. Burman and M. C. Otto, Census bureau research project: Outliers in time series, Bureau of the Census, SRD Res. Rep. CENSUS/SRD/RR-88114, May 1988.

[12] G. Deco and B. Schurmann, Neural learning of chaotic dynamics. Neural Process Lett., vol. 2, no. 2, pp. 23-26, 1995.

[13] Archer, Evan, et al. Black box variational inference for state space models. arXiv preprint arXiv:1511.07367 (2015).

[14] V. Barnett and T. Lewis, Outliers in Statistical Data. New York, NY, USA: Wiley, 1978.

[15] D. M. Hawkins, Identification of Outliers. London, U.K.: Chapman and Hall, 1980.

[16] P. J. Rousseeuw and A. M. Leroy, Robust Regression and Outlier Detection. New York, NY, USA: Wiley, 1987.

[17] K. Yamanishi and J.-I. Takeuchi. A unifying framework for detecting outliers and change points from non-stationary time series data, in Proc. 8th ACM Int. Conf. KDD, Edmonton, AB, Canada, 2002, pp. 676-681.

[18] K. Sequeira and M. Zaki. ADMIT: Anomaly-based data mining for intrusions, in Proc. 8th ACM Int. Conf. KDD, New York, NY, USA, 2002, pp. 386-395.

[19] T. Id and H. Kashima. Eigenspace-based anomaly detection in computer systems, in Proc. 10th ACM Int. Conf. KDD, Seattle, WA, USA, 2004, pp. 440-449.

[20] L. Akoglu and C. Faloutsos, Event detection in time series of mobile communication graphs, in Proc. Army Science Conf., 2010.

[21] B. Hammer. On the approximation capability of recurrent neural networks. Neurocomputing, 31(1):107-123, 2000.

[22] J. Bayer and C. Osendorfer. Learning stochastic recurrent networks. arXiv:1411.7610, 2014.

[23] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In NIPS, pages 2962-2970, 2015.

[24] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. arXiv:1206.6392, 2012.

[25] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735-1780, 1997.

[26] A. Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[27] J. Bayer and C. Osendorfer. Learning stochastic recurrent networks. arXiv preprint arXiv:1411.7610, 2014.

[28] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations (ICLR), 2014.

[29] D. Kingma, and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).

[30] M. Slch, J. Bayer, M. Ludersdorfer and P. van der Smagt. Variational inference for on-line anomaly detection in high-dimensional time series. arXiv preprint arXiv:1602.07109 (2016).

[31] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral Networks and Locally Connected Networks on Graphs. In International Conference on Learning Representations (ICML), 2014.

[32] F. R. K. Chung. Spectral Graph Theory, volume 92. American Mathematical Society, 1997.

[33] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems (NIPS), 2016.

[34] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine, 30(3):83-98, 2013.

[35] M. Wu, X. Song, C. Jermaine, et al, A LRT framework for fast spatial anomaly detection, In Proc. of KDD 09, pp. 887-896.

[36] Junbo Zhang, Yu Zheng, Dekang Qi. Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. In AAAI 2017.