Detecting Contextual Network Anomaly in the Radio Network Controller from Bayesian Data Analysis

KIM SEONGHYUN
Detecting Contextual Network Anomaly in the Radio Network Controller from Bayesian Data Analysis

KIM SEONGHYUN

Master Thesis at CSC
Supervisor: Anders Holst
Examiner: Anders Lansner
Subject: Computer Science
Programme: Machine Learning
Abstract

This thesis presents Bayesian approach for a contextual network anomaly detection. Network anomaly detection is important in a computer system performance monitoring perspective. Detecting a contextual anomaly is much harder since we need to take the context into account in order to explain whether it is normal or abnormal. The main idea of this thesis is to find contextual attributes from a set of indicators, then to estimate the resource loads through the Bayesian model. The proposed algorithm offers three advantages. Firstly, the model can estimate resource loads with automatically selected indicators and its credible intervals. Secondly, both point and collective contextual anomalies can be captured by the posterior predictive distribution. Lastly, the structural interpretation of the model gives us a way to find similar nodes. This thesis employs real data from Radio Network Controller (RNC) to validate the effectiveness in detecting contextual anomalies.
Contents

List of Figures

List of Tables

1 Introduction 1
  1.1 Network anomaly detection 1
  1.2 Contextual anomaly 1
  1.3 Task description 2
  1.4 Structure of the thesis 3

2 Related Work 4
  2.1 Anomaly detection 4
  2.2 Variable selection 4
  2.3 State-space model 5
  2.4 Radio resource metric estimation 6
  2.5 Empirical Bayes method 8

3 Variational Bayesian methods 9
  3.1 Variational Bayesian inference 9
  3.2 Variational Expectation-Maximization algorithm 11
  3.3 Variational Bayesian linear regression 11

4 Bayesian Contextual Anomaly Detection 15
  4.1 Stochastic search variable selection 15
  4.2 The MCMC exploration 16
  4.3 Principal anomaly detection 16

5 Experiments 19
  5.1 Setup 19
  5.2 Estimation performance 21
  5.3 Inclusion probability 23
  5.4 Model robustness 24
  5.5 Finding similar nodes 25
  5.6 Detecting contextual anomaly 27

6 Discussion 29
  6.1 Major findings 29
  6.2 Limitations 29

7 Conclusion 30

8 References 31
List of Figures

1. Types of anomaly ................................................................. 2
2. Graphical model representation of state-space model .................. 5
3. Telecommunication network system ...................................... 7
4. Variational Bayesian linear regression in graphical model .......... 12
5. Overview of the SSVS algorithm with the MCMC algorithm ....... 17
6. Illustration of the principal anomaly of an unknown observation $z$ . 17
7. Knowledge Discovery System ............................................... 20
8. Time series plot comparing an estimated load against a measured load... 21
9. Histogram of model size ....................................................... 22
10. Estimation performance comparison trained by same node .......... 22
11. Inclusion probability ......................................................... 24
12. Model robustness ............................................................. 24
13. Example of the PDR device case ......................................... 25
14. Estimation performance trained by similar nodes .................... 26
15. Example of contextual point anomaly ................................... 27
16. Example of contextual collective anomaly ............................. 28
List of Tables

1  Comparison between State-space model and Box-Jenkins model approaches . 6
2  The hardware setting of the RNC. .................................................... 19
3  Results of average forecast errors from the devices. .......................... 23
4  Comparison between results from same nodes and similar nodes ......... 26
1 Introduction

This report is the result of a master’s thesis project at the KTH, performed at Ericsson AB, based in Stockholm, Sweden.

1.1 Network anomaly detection

Different subscriber behaviors, hardware setups, and software configurations in a telecommunication network node generate different processor load patterns. A company validates its product functionality by its strict evaluation process, such as key performance indicators, test pass rates, or trouble reports. However, absence of evidence is not evidence of absence. It is hard to find or to reproduce bugs when unknown cases deviated from normal behaviors. The root cause for these cases might be due to hardware malfunctions or software defects on a network node. We call this network anomaly.

Network anomaly detection refers to the problem of finding patterns in data that do not conform to the expected behavior occurring on a network [1]. A network anomaly refers to the faulty behavior in the computer system. For example, we can identify unusual peaks and drops of traffic which are different from normal pattern. The software that detects network anomalies is called Network Anomaly Detection System (NADS). Since it is hard to conclude possible abnormal behaviors by rule-based system, a company commonly applies machine learning techniques to detect network anomaly. This is one of the most common internal and external discovery methods in the industry because it can detect network anomalies quickly and automatically [2]. Network anomaly detection is important in computer system performance monitoring perspective. It is important to point out that NADS is different from Network Intrusion Detection System (NIDS). While both of them aim to detect significant changes in network behavior, NIDS aims to detect policy violations such as DDoS attacks rather than possible software defects. In this thesis, we aim to detect network anomalies, not network intrusions.

1.2 Contextual anomaly

Potential anomalies in continuous sequence can be divided into three different types [1]: a) a point anomaly, also known as an outlier, occurs when a data point is anomalous, b) a collective anomaly occurs when a group of data points is anomalous while each data point may not be anomaly by itself, and finally c) a contextual point or contextual collective anomaly occurs when a single data point is anomalous or collective data points are anomalous with regards to the context.

Figure 1 shows an illustration of those anomaly types where time is the contextual attribute which determines the position of the resource load on a network node. Note that each data point of these contextual anomalies may not be considered anomalies if they happened in other times. While it is easier to discover point anomaly, it is much harder to detect contextual anomaly since we need to take the context into account in order to explain whether it is normal or abnormal. For example, contextual collective anomaly in Figure 1 could be considered as normal if the data is collected on a Sunday since people tend to use their equipment less during the weekends. We, therefore, cannot simply judge the time-series plot by its shape if more than one contextual attributes are required to judge. A contextual anomaly detection algorithm is required to consider all the important variables that may
explain suspicious network node behaviors.

Detecting a contextual anomaly is hard in four ways. Firstly, it is hard to make a generic algorithm for anomaly detection. Since the definition of abnormal behavior is quite different from company to company, many anomaly detection algorithms have been developed for specific application to meet each company’s requirements. Secondly, the training data for an anomaly detection system are very asymmetric since few and diverse anomalies are observed against a much larger set of normal cases. This makes it hard to detect a new error when an unseen behavior presents itself. Thirdly, a metric that is too sensitive to differentiate will yield many false alarms. Similarly, a metric that is not sensitive enough will yield many missed detections. Lastly, a behavior that can be considered as normal in one situation can be abnormal in other situation. Therefore, there may be more than one normal behavior, and each behavior can be quite different depending on the context.

1.3 Task description

Main idea As we see in Figure 1, finding contextual attributes that may explain suspicious network behavior is the key to improve the contextual anomaly detection algorithm’s performance. When the number of possible indicators is large, it becomes difficult to know which indicators we need to look into. Unfortunately, this is a common condition for network devices. The main idea of this thesis is to find contextual attributes from a set of indicators, then to estimate the resource loads through the Bayesian model. The model will give us not only point estimates, but also credible intervals which are the keys to detect anomalies. The benefit of Bayesian methods is that they allow the model to work with a small amount of data while yielding a reliable decision. A more detailed description of the proposed algorithm is presented in Section 5.

Focus Bayesian anomaly detection has been researched in various institutes including the Swedish Institute of Computer Science (SICS), and has been performed well in real applications. In this thesis, we will introduce a new aspect of the Bayesian anomaly detection.

The main focus is to use statistical model that finds the best indicators automatically
and to conclude potential network anomalies of device resource loads based on their context. The experiments on the real dataset validate the effectiveness in detecting network anomaly. Throughout the thesis, we assume the reader has basic knowledge of probability, statistics, and calculus.

**Goals and limitations** The project aims to develop a machine learning algorithm for selecting the best indicators in order to provide statistical evidence to conclude possible anomalous system performance, and to validate its practical utility by using real data. The priority will be on accomplishing significant results in the following areas:

- The algorithm provides accurate estimation.
- The algorithm concludes possible contextual anomalies.
- The algorithm needs to be learned and improved continuously based on data over time.
- The algorithm offers better insights to find similar network nodes.

1.4 **Structure of the thesis**

The rest of this paper is structured as follows.

- Section 2 presents a review of related work.
- Section 3 describes the variational Bayesian inference in detail.
- Section 4 describes the detail of the proposed algorithm.
- Section 5 describes how the method works in real data.
- Section 6 discusses its practical ability.
- Section 7 describes concluding remarks.
2 Related Work

In this section, the literature reviews of this research topic are introduced.

2.1 Anomaly detection

As an overview, we will briefly mention anomaly detection approaches such as density based method [16], graph stream [15], and distance-based method [17] [18] [19]. Because these approaches can handle imbalanced datasets and make use of unlabeled data which are cheaper to obtain than labeled data, they are the most commonly used methods in the industry. These methods are also known as clustering analysis, which makes them not only effective in detecting anomalies occurred collectively among a background of normal data but also in detecting point anomalies. Because of these advantages, clustering analysis are preferred methods for collective anomaly detection.

Clustering approaches to detect anomalies, however, have several drawbacks. Firstly, they are not easily generalized in multi-dimensional data because the neighborhood can be arbitrary. Secondly, a major drawback to clustering algorithm is that anomalies that does not belong to any cluster can be assigned to a larger cluster. Finally, it is a subjective choice to determine the number of clusters. Especially in the case for fast online anomaly detection, where the size of abnormal class is unknown, and the space and time conditions are varied, finding the number of clusters in these conditions is very difficult.

Bayesian approach In this thesis, we will design the statistical anomaly detection using Bayesian approach as an alternative. The general idea in statistical anomaly detection is to build a statistical model over normal cases and to compare new samples with trained models when they arrived. Samples that have small probabilities of being generated by the statistical model are considered anomalies, that is, they are very unlikely to belong to a set of normal cases.

The proposed algorithm uses variable selection to solve problems stated above. Firstly, it will utilize the Bayesian approach to automatically find its best indicators from the total among to reduce the dimensions of raw data. Secondly, it will learn a posterior predictive distribution from the indicators which shows the uncertainty of a point estimate for normality to find anomalies regarding to its context. Thirdly, the proposed algorithm uses the top indicators that have higher inclusion probability to find the best $k$ and to form a predictive posterior distribution as an approximate solution.

2.2 Variable selection

When it comes to building a multiple regression model, variable selection refers to finding a promising subsets of regressors to the formula $y = x_1\beta_1 + \cdots + x_p\beta_p + \eta$. If the number of potential regressors becomes large, it becomes hard to induce the best set of regressors. In the worst case, the number of regressors can be larger than the number of measurements due to data sparsity. For example, the data generated from a telecommunication network testing device can be smaller than the number of traffic events category due to the cost. Variable selection is important because a) it will explain the data more easily by removing redundant regressors, b) collinearity is caused if too many similar regressors are in the model, and c) noises from irrelevant regressors will be also reduced.
One of the methods to select the best indicators is a variable importance estimation. A random forest [24] is widely used to measure a variable importance. It is a machine learning method for classification and regression that calculates the importance of individual feature after each training. However, a variable importance estimation is limited because it cannot yield decision boundary based on selected regressors only, which is critical to contextual anomaly detection. This is due to the fact that it builds an entire model including all regressors. Another idea is the Lasso method [25], a shrinkage and selection method for linear regression. The Lasso is a sparse approximation method that roughly solves regression with its sparse vector. However, since it ignores a posterior uncertainty about their coefficients, we cannot use it to detect contextual anomaly because a credible interval is required to conclude anomaly.

Bayesian approach Much progress has been made in recent years in the Bayesian variable selection [11] [12] [13] with hierarchical priors to induce shrinkage to candidate subsets of parameters in multivariate analysis. Among them we followed George and McCulloch’s method [12], also known as Stochastic search variable selection where it offers advantages to solve the problems. First of all, it tries to find the best model from all possible ways of combining a model and to learn the linear relationship between independent variables and a dependent variable. Finally, the model generates posterior distributions on the regression coefficients with inclusion probability for each group of selected regressors. More details about Bayesian variable selection will be shown in Section 4.1.

2.3 State-space model

State-space model is a probabilistic model of observations $y_{1:t}$ and hidden states $\alpha_{1:t}$. Figure 2 shows a graphical model representation of state-space model.

![Graphical model representation of state-space model](image)

The goal of state-space models is to recursively estimate hidden states $p(\alpha_t \mid y_{1:t})$ where they are continuous. State-space models are expressed in two equations as follows. Equation (1), the transition equation, defines how the state vector changes over time. Equation (2), the observation equation, defines how the response variable depends on the state.

$$
\begin{align*}
\alpha_t &= g(\alpha_{t-1}, \eta_t), \\
y_t &= h(\alpha_t, \varepsilon_t),
\end{align*}
$$

$$
\begin{align*}
\eta_t &\sim N(0, Q_t), \\
\varepsilon_t &\sim N(0, H_t),
\end{align*}
$$
where \( y_t \) is an observation at time \( t \), \( \alpha_t \) is a hidden state, \( \varepsilon_t \) is a measurement error, \( \eta_t \) is a process error, \( g \) is a transition model, \( h \) is an observation model, \( \varepsilon_t \) is a system noise, \( \eta \) is an observation noise, \( Q_t \) and \( H_t \) are parameters \( \theta \) that are independent of time, and \( p \) is a desirable number of independent variables.

State-space models are well suited for time-series forecasting. Although the Box-Jenkins ARMA model [23] could be an alternative, the state-space model is more practical as compared to Box-Jenkins approach. Firstly, the state-space model could be set up in a recursive form. The calculations for the model in a recursive form enables large models to be calculated more effectively. Also, applying Bayesian method in state-space framework is easy and widely used in real applications. In the case of Box-Jenkins, however, recursive form is intractable due to the nature of the model, which makes it a less attractive option for real applications. Secondly, we can easily interpret the state-space model with components because it is possible to add components to the model in a straightforward way. Thirdly, the Box-Jenkins model is inefficient due to its reliance on sample autocorrelation. More detailed comparison can be found in James Durbin’s book [21].

<table>
<thead>
<tr>
<th></th>
<th>State-space models</th>
<th>Box-Jenkins models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursive form</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Structural analysis</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Auto correlation reliance</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1: Comparison between State-space model and Box-Jenkins model approaches

We can perform an online Bayesian inference for the parameters of a linear regression model with the state-space model. In this case, the hidden states represent regression parameters in this case. More details will be shown in Section 3.

2.4 Radio resource metric estimation

In the telecommunication industry, a radio resource metric estimation (RME) [26] is a technique for estimating the desired resource loads based on measured data which has a random component. In this paper, we used data from the Radio Network Controller among telecommunication network system devices.

The Radio Network Controller (RNC) is a Wideband Code Division Multiple Access (WCDMA) node which administrates and controls radio resources and mobile equipments within a telecommunications network. In order to support vast amounts of signals and user data associated with subscriber traffic events (e.g. call setups), the RNC requires a real-time operating system, multi-CPU, and multicore hardware devices in order to handle the traffic demands.

The RNC is one of the most important nodes of a telecommunication network. If it is not operating optimally due to the lack of available resources, this can restrict the capacity for handling customer demand and could costs the companies to lose potential customers. Among the CPU and memory devices, also known as network node resources, a Common Channel (CC) device, a Dedicated Control (DC) device, a Module Controller (MC), and a Packet Data Router (PDR) device are the most critical in maintaining the customer demand. In the event of failure in any of these devices, they could block services to the customers.
However, finding a system performance issue in the RNC can be difficult. The RNC has a highly complex structure with a large number of diverse and interconnected hardwares to route a vast amount of signals and user data on a multitude of different transmission paths via the Radio Base Stations (RBS). For example, two RNC nodes with identical hardware settings and software configurations, along with the same number of subscribers, can behave differently if the subscriber’s usage patterns are different. On top of that, there are situations where abnormal behavior is not discovered when resources have become too large. Moreover, operators used to set software configurations differently to fulfill their own requirements, which are hidden to analysts. Therefore, identifying problems in a complex node is not an easy task, and may cost huge amount of man-hours.

Existing methods for formulating a model of the normal resource usages involve a large number of calculations related to subscriber traffic models and how resources are consumed by the RNC node. Such numerical models are typically very complex, and often depend on the assumptions made of the system’s physical properties, the user’s behavior on the network, and, particularly in the case of radio based communications networks, the geographical surroundings of the network.

The above examples can be found in Chapter 8 of A. Toskala’s book [27] and M. Hunukumbure’s method [28]. A desired result in these approaches are very difficult due to the complexity of the system. Wenbiao Wu [29] proposed a linear regression method for simulating a network node, but the method required a set of carefully selected indicators to be used, and only point estimate can be considered. In anomaly detection systems, however, the key question to discovering a anomaly is how much it deviates from the normal behaviors when it happens. Equation 3 represents the linear regression model of the RNC resources.

\[
y_t = \sum_{i=0}^{p} x_{t,i} \beta_i \tag{3}\]

where \(y_t\) is a resource load at time \(t\), \(p\) is a desirable number of traffic events, \(x_{t,i}\) is \(i^{th}\) traffic event at time \(t\), and \(\beta_i\) is a cost of \(i^{th}\) traffic event.
A good way to tackle this problem is through the machine learning approach. During network operations, the RNC continuously generates a large amount of data and logs. These data are rich in many information; ranging from subscriber traffic events and operational state to internal and external nodes events. Having access to data from different nodes and regions makes it possible to monitor operations of nodes and to predict their behavior as well as possible coming issues. Machine learning provides effective tools for this type of analysis.

2.5 Empirical Bayes method

The empirical Bayes method is a procedure that approximates the marginal likelihood, as known as the evidence in Bayesian statistics, from the data instead of integrating data out. The process of calculating parameters from a set of observations, which is called an inversion problem, can be formalized by Bayesian inference as follows:

\[
p(\theta \mid y) = \frac{p(y \mid \theta)p(\theta)}{p(y)}
\]  

In practice, however, it is hard to calculate the marginal likelihood \( p(y) = \int p(y, \theta) d\theta \) since it is impossible or too expensive. There are several ways to approximately estimate \( p(y) \). First, the Laplace approximation provides a way of estimate a density when its normalization constant is not available. However, the Laplace approximation assumes Gaussian distribution and it becomes ineffective when the true posterior is a multimodal distribution. Second, another approach is the stochastic method such as sampling. It draws samples \( \theta^{(1)}, \ldots, \theta^{(m)} \sim p(\theta \mid y) \). The pure MCMC algorithm can be one example, it is, however, computationally expensive. Finally, the maximum likelihood estimation (MLE) and the maximum a posterior (MAP) estimation examine probability density, not mass, therefore that they cannot properly calculate the integral.

In this thesis, variational Bayesian method is used to approximate the integral. Variational Bayesian inference is a Bayesian method for approximating density that is similar to the true posterior distribution. The variational Bayesian method constructs a lower bound on marginal likelihood \( p(y) \) by training, and it attempts to optimise a lower bound iteratively over samples. Since it will reduce the amount of computation time in high-dimensional space, it is one of the most popular methods to approximate the integral. More details will be shown in Section 3.
3 Variational Bayesian methods

In this section, we will introduce the theory for variational Bayesian methods. We will describe how the variational Bayesian inference can be applied to implement multiple linear regression models. This section is a theoretical core of the thesis.

3.1 Variational Bayesian inference

Variational Bayesian inference is a Bayesian method for approximating density $p(y)$ that is similar to the true posterior distribution. For background information, please refer to Section 2.5.

**KL divergence** The KL divergence is a measurement of the differences between probability distributions $q$ and $p$. For $q$ and $p$, the Kullback-Leibler (KL) divergence of $p$ from $q$ is defined as follows:

$$\text{KL}[q \parallel p] = \int q(\theta) \ln \frac{q(\theta)}{p(\theta)} d\theta$$  \hspace{1cm} (5)

**Analytical proxy** Equation 6 represents $p(y)$ in variational Bayesian inference with analytical proxy $q(\theta)$.

$$p(y) = \frac{p(y, \theta)}{p(\theta | y)}$$

$$\ln p(y) = \ln \frac{p(y, \theta)}{p(\theta | y)}$$

$$= \int q(\theta) \ln \frac{p(y, \theta)}{p(\theta | y)} d\theta$$

$$= \int q(\theta) \ln \frac{p(y, \theta)}{p(\theta | y)} q(\theta) d\theta$$

$$= \int q(\theta)(\ln \frac{q(\theta)}{p(\theta | y)} + \ln \frac{p(y, \theta)}{q(\theta)})d\theta$$

$$= \int q(\theta)(\ln \frac{q(\theta)}{p(\theta | y)})d\theta + \int q(\theta)(\ln \frac{p(y, \theta)}{q(\theta)})d\theta$$

$$= \text{KL}[q \parallel p] + F(q, y)$$  \hspace{1cm} (6)

where $\text{KL}[q \parallel p]$ is the KL divergence between approximate density $q(\theta)$ and true posterior $p(\theta | y)$. $q(\theta)$ refers approximate density in hypothesis space also known as an analytical proxy in variational calculus. $F(q, y)$ is the lower bound which is also known as the variational free energy in variational calculus. $F(q, y)$ can be represented as follows:

$$F(q, y) = \int q(\theta) \ln \frac{p(y, \theta)}{q(\theta)} d\theta$$

$$= \int q(\theta) \ln p(y, \theta) d\theta - \int q(\theta) \ln q(\theta) d\theta$$  \hspace{1cm} (7)
Note that the KL divergence is greater or equal to 0, and maximizing $F(q, y)$ is equivalent to minimizing $\text{KL}[q \parallel p]$ as in Equation 9.

$$\ln p(y) - F(q, y) = \text{KL}[q \parallel p] \geq 0$$ (9)

The true distribution $p(y)$ can also be reformulated as follows:

$$\ln p(y) = \ln \int p(y, \theta)d\theta = \ln \int q(\theta)\frac{p(y, \theta)}{q(\theta)}d\theta$$ (10)

Then, we can use the Jensen’s inequality [9] that based on a log function, which also is a concave function.

$$\ln \int q(\theta)\frac{p(y, \theta)}{q(\theta)}d\theta \geq \int q(\theta)\ln \frac{p(y, \theta)}{q(\theta)}d\theta$$ (11)

**Mean-field theory** A common way of restricting the class of approximate posteriors $q(\theta)$ is to assume those posteriors are independent to each other. Equation (12) represents the mean-field assumption.

$$q(\theta) = \prod_i q_i(\theta_i)$$ (12)

Equation (13) represents variational Bayesian inference with mean-field theory as follows:

$$F(q, y) = \int \prod_i q(\theta_i) \times (\ln p(y, \theta) - \sum_i \ln q(\theta_i))d\theta$$

$$= \int q(\theta_j)(\prod_i q(\theta_i)(\ln p(y, \theta) - \ln q(\theta_j)))d\theta - \int q(\theta_j)\prod_i q(\theta_i) \sum_i \ln q(\theta_i)d\theta$$

$$= \int q(\theta_j)(\int \prod_i q(\theta_i) \ln p(y, \theta) - \ln q(\theta_j))d\theta - \int q(\theta_j)\int \prod_i q(\theta_i) \ln \prod_i q(\theta_i) d\theta_j d\theta_j$$

$$= \int q(\theta_j)(\int \prod_i q(\theta_i) \ln p(y, \theta) - \ln q(\theta_j))d\theta + c$$

$$= \text{KL}[q(\theta_j) \parallel (\prod_i q(\theta_i) \ln p(y, \theta)d\theta)_{\setminus j}] + c$$

$$= \text{KL}[q(\theta_j) \parallel p(\theta | y)] + c$$ (13)

Since our goal is to maximize $F(q, y)$, we can define analytical proxy $q(\theta_j)$ as follows:

$$q(\theta_j) = \arg \max_{q(\theta_j)} F(q, y)$$ (14)

$$= (\prod_i q_i \ln p(y, \theta)d\theta)_{\setminus j}$$ (15)

$$= \text{KL}[q(\theta_j) \parallel p(\theta | y)] + c$$ (16)
3.2 Variational Expectation-Maximization algorithm

This section represents Expectation-Maximization (EM) algorithm for probabilistic models using variational approach. Variational Bayesian method can also be working in an iterative manner, where it is common to use the algorithm similar to the EM algorithm [10]. The major difference is that the variational Bayesian tries to minimize the distance $KL[q(\theta) \| p(\theta \mid y)]$ over time. The algorithm cycles over the parameters which recalculates the expected value of the log likelihood given the current estimates. The loop continues until the distance between new log likelihood $\ln p(y \mid \theta_{t+1})$ and new lower bound $F(q, y)_{t+1}$ given $\theta$ converges.

Equation (17) shows the E step where it finds $q(\theta)_{t+1}$ to maximize a lower bound $F(q, y)$. After the E step, the distance $KL[q(\theta_j) \| p(\theta \mid y)]$ between the lower bound and the log likelihood $\ln p(y \mid \theta)$ becomes close to each other.

Equation (18) shows the M step. It updates $\theta_{t+1}$ while holding $q(\theta)_{t+1}$ fixed. After the M step, the new likelihood $\ln p(y \mid \theta_{t+1})$ is updated, it becomes closer to the true posterior $\ln p(y)$.

$$q(\theta)_{t+1} = \arg \max_{q(\theta_t)} F(q, y) \quad (17)$$

$$\theta_{t+1} = \arg \max_{\theta_t} F(q(\theta)_{t+1}, y) \quad (18)$$

3.3 Variational Bayesian linear regression

In this section, variational Bayesian method for linear regression is introduced. It is based on the concepts of analytical proxy and variational EM algorithm that we have explained above. Linear regression is one of the most important techniques in statistical analysis because it is easier to analyse and can effectively approximate many systems. The Bayesian methods have been proposed in this subject [4] with normal-inverse-gamma prior. They allow us to specify regression coefficients in both prior and posterior uncertainties.

In this regression model, $p(y_t \mid \beta)$ can be estimated by

$$p(y_t \mid \beta) = N(y_t \mid \beta^T x_t, \lambda^{-1}) \quad (19)$$

The goal is to infer parameters by using normal-inverse-gamma prior given observed values $x$s and $y$s. Equation (20) shows normal-inverse-gamma prior with three sub models such as coefficients $\beta$, coefficients precision $\alpha$, and noise precision $\lambda$.

$$p(\alpha) = Ga(\alpha \mid a, b)$$

$$p(\beta \mid \alpha) = N(\beta \mid 0, \alpha^{-1}) \quad (20)$$

$$p(\lambda) = Ga(\lambda \mid c, d)$$

The method is to find the posterior $p(\beta, \alpha, \lambda \mid Y)$, by using variational approximation with

$$q(\beta, \alpha, \lambda) = q_\beta(\beta)q_\alpha(\alpha)q_\lambda(\lambda) \quad (21)$$
**Figure 4**: Here is an example of a variational Bayesian linear regression in graphical model where square indicates the regression model, circles indicate random variables, and filled-in circle indicates measured values. Since it uses normal-inverse-gamma prior, $\beta$ has a normal distribution while $\alpha$ and $\lambda$ have an inverse-gamma distribution.

**Expressions**  Normal distribution and gamma distribution are used in a normal-inverse-gamma prior.

Equation (22) represents univariate normal distribution.

$$\ln N(x \mid \mu, \lambda^{-1}) = -\frac{1}{2} \lambda x^2 + \lambda \mu x + c$$

Equation (23) represents multivariate normal distribution.

$$\ln N_d(x \mid \mu, \Lambda^{-1}) = -\frac{1}{2} x^T \Lambda x + x^T \Lambda \mu + c$$

Equation (24) represents gamma distribution.

$$\ln Ga(x \mid a, b) = a \ln b - \ln \Gamma(a) + (a - 1) \ln x - bx$$

$$= (a - 1) \ln x - bx + c$$

Zellner’s g-prior in equation (25) is widely used to estimate the prior precision with $g = 1$ and $w = 0.5$.

$$\Sigma^{-1} = \frac{g}{n} X^T X$$

Since g-prior becomes improper when $X^T X$ is not positive definite, instead we use an improved g-prior. Equation (26) shows an improved g-prior where $\text{diag}$ is the diagonal matrix.

$$\Sigma^{-1} = \frac{g}{n} \left\{ w(X^T X) + (1 - w) \text{diag}(X^T X) \right\}$$
Coefficients precision  Prior $\alpha$ is the prior precision over $\beta$. In $q^*(\alpha)$ can be represented as follows:

\[
\ln q^*(\alpha) = (\ln p(y, \beta, \alpha, \lambda))_{q(\beta, \lambda)} + c
\]
\[
= (\ln \prod N(y_i \mid \beta^T x_i, \lambda^{-1}))_{q(\beta, \lambda)} + \ln N_d(\beta \mid 0, \alpha^{-1} I))_{q(\beta, \lambda)} + \ln Ga(\alpha \mid a_0, b_0))_{q(\beta, \lambda)} + c
\]
\[
= (-\frac{1}{2} \ln |\alpha^{-1} I| - \frac{d}{2} \ln 2\pi - \frac{1}{2} (\beta - 0)^T \alpha I (\beta - 0))_{q(\beta)} + (a_0 \ln b_0 - \ln \Gamma(a_0) + (a_0 - 1) \ln \alpha - b_0 \alpha)_{q(\beta)} + c
\]
\[
= \frac{d}{2} \alpha - \frac{1}{2} (\beta^T \beta)_{q(\beta)} + (a_0 - 1) \ln \alpha - b_0 \alpha + c
\]
\[
= (\frac{d}{2} + a_0 - 1) \ln \alpha - (\frac{1}{2} (\beta^T \beta)_{q(\beta)} + b_0) \alpha + c
\]

therefore,

\[
q^*(\alpha) = Ga(\alpha \mid a_n, b_n)
\]

(28)

where $\alpha_n = a_0 + \frac{d}{2}$ and $b_n = b_0 + \frac{1}{2} (\beta^T \beta)_{q(\beta)}$

Coefficients  $\ln q^*(\beta)$ can be represented as follows:

\[
\ln q^*(\beta) = (\ln p(y, \beta, \alpha, \lambda))_{q(\alpha, \lambda)} + c
\]
\[
= (\ln \prod N(y_i \mid \beta^T x_i, \lambda^{-1}))_{q(\alpha, \lambda)} + \ln N_d(\beta \mid 0, \alpha^{-1} I))_{q(\alpha, \lambda)} + \ln Ga(\alpha \mid a_0, b_0))_{q(\alpha, \lambda)} + c
\]
\[
= \sum_{i}^{n} (\frac{1}{2} \ln \lambda - \frac{1}{2} \ln 2\pi - \frac{\lambda}{2} (y_i - \beta^T x_i)^2)_{q(\alpha, \lambda)} + (-\frac{1}{2} \ln |\alpha^{-1} I| - \frac{d}{2} \ln 2\pi - \frac{1}{2} \beta^T \alpha I \beta)_{q(\alpha)} + c
\]
\[
= -\frac{\lambda q(\lambda)}{2} \sum_{i}^{n} (y_i - \beta^T x_i)^2 - \frac{1}{2} \alpha q(\alpha) \beta^T + c
\]
\[
= -\frac{\lambda q(\lambda)}{2} y^T y + \lambda q(\lambda) \beta^T X^T y - \frac{1}{2} \beta^T \alpha q(\alpha) I \beta + c
\]
\[
= -\frac{1}{2} \beta^T (\lambda q(\lambda) X^T X + \alpha q(\alpha)) \beta + \beta^T \lambda q(\lambda) X^T y + c
\]

therefore,

\[
\ln q^*(\beta) = N_d(\beta \mid \mu_n, \Lambda_n^{-1} \lambda q(\lambda))
\]

(30)

where $\Lambda_n = \alpha q(\alpha) I + X^T X$ and $\mu_n = \Sigma_n^{-1} \lambda q(\lambda) X^T y$, 

13
Noise precision  \( \ln q^*(\lambda) \) can be represented as follows:

\[
\ln q^*(\lambda) = (\ln p(y, \beta, \alpha, \lambda))_{q(\beta, \alpha)} + c
\]

\[
= \left( \sum_{i} \frac{1}{2} \ln \lambda - \frac{1}{2} \ln 2\pi - \frac{\lambda}{2} (y_i - \beta^T x_i)^2 \right)_{q(\beta, \alpha)} +
\]

\[
(c_0 \ln d_0 - \ln \Gamma(c_0) + (c_0 - 1) \ln \lambda - d_0 \lambda)_{q(\beta, \alpha)} + c
\]

\[
= \frac{n}{2} \ln \lambda - \frac{\lambda}{2} y^T y + \lambda \beta^T_{q(\beta)} X^T y - \frac{\lambda}{2} \beta^T_{q(\beta)} X^T X \beta_{q(\beta)} + (c_0 - 1) \ln \lambda - d_0 \lambda + c
\]

\[
= (c_0 + \frac{n}{2} + 1) \ln \lambda - (d_0 + \frac{1}{2} y^T y - \beta^T_{q(\beta)} X^T y + \frac{1}{2} \beta^T_{q(\beta)} X^T X \beta_{q(\beta)}) \lambda + c
\]

therefore,

\[
q^*(\lambda) = G\alpha(\lambda \mid c_n, d_n)
\]  

where \( c_n = c_0 + \frac{n}{2} \) and \( d_n = d_0 + \frac{1}{2} y^T y - \beta^T_{q(\beta)} X^T y + \frac{1}{2} \beta^T_{q(\beta)} X^T X \beta_{q(\beta)} \)
4 Bayesian Contextual Anomaly Detection

In this section, Bayesian contextual anomaly detection is presented as the proposed algorithm where it makes use of Bayesian variable selection to find possible contextual attributes from a set of indicators. It also gives us posterior predictive distribution of an unknown observation. The principal anomaly detection will be used to conclude whether an unknown observation is anomalous or not by using the posterior predictive distribution. We will describe Bayesian variable selection in Section 4.1 and 4.2, and in Section 4.3, we will present the principal anomaly detection.

Proposed algorithm The main problem is to construct a resource load estimator and to detect contextual anomalies in resource loads within an RNC node. We treated it as a Bayesian variable selection problem using the SSVS algorithm in Section 4.1, and considering each group of variables $\gamma$ as a set of potential regressors $X_1, \cdots, X_p$ given a resource load $Y$ as a dependent variable. Then, we make use of those selected regressors to conclude anomaly from Bayesian analysis by using the principal anomaly detection in Section 4.3. As a result, the proposed algorithm provides the possibility to estimate resource loads and concludes possible anomalous resource load behaviors with regards to the context.

A posterior predictive distribution cannot be calculated without the MCMC algorithm because calculating the sum over the model space is intractable. We ran the MCMC algorithm for 10,000 iterations and discarded the first 1,000 because we need to take the burn-in process to reach its equilibrium distribution in consideration. The threshold for inclusion probability is 0.1 which is a widely used value in many research papers.

4.1 Stochastic search variable selection

We followed the procedure called Stochastic search variable selection (SSVS) [12]. It is based on the idea of variational Bayesian linear regression as in Section 3.3. We can infer parameters of the linear model by using variational Bayesian linear regression. However, it assumes we selected a proper set of indicators. The SSVS algorithm provides us a way to select the best indicators by using a spike-and-slab prior.

Spike-and-slab prior A spike-and-slab prior is a hierarchical prior and can be divided into two parts: a) a spike prior, for a subset of all the predictors, and b) a slab prior, for a weakly informative distribution on coefficients of the selected predictors. Equation (33) describes a spike-and-slab prior.

Equation (34) is called a spike prior. The number of sets of indicators would be $2^p - 1$, where $p$ is the number of potential indicators. $p(\gamma_j)$ is an inclusion probability of set $j$ that determines whether or not set $j$ should be included to the marginal probability calculation, also known as Bayesian averaging. $\pi_k$ is a boolean value for determining if $k$th traffic event should be included. If $\pi_k = 0$ then $\beta_k = 0$, and $\pi_k = 1$ then $\beta_k \neq 0$ where $\pi \in (0, 1)$.

For example, if there are four traffic event candidates that may influence device loads, and that the experts believed the first and the last traffic should be included, then, the sets of events would be $2^4 - 1 = 15$ and $\gamma = (1, 0, 0, 1)$. $\gamma = (1, 0, 0, 1)$ refers the first and last traffic events will be included.

Equations (35) and (36) describe a slab prior, where $v$ is a prior sample size and $ss$ is a prior sum of squares. We can calculate a slab prior by using variational Bayesian inference.
that we have discussed in Section 3.

\[
p(\beta, \gamma, \lambda) = p(\beta, \gamma | \gamma, \lambda)p(\lambda | \gamma)p(\gamma) \tag{33}
\]

\[
(\gamma) \sim \prod_{k=1}^{K} \pi_k^{\gamma_k}(1 - \pi_k)^{1-\gamma_k} \tag{34}
\]

\[
(\beta, \gamma | \gamma, \lambda) \sim N(\beta, \alpha^{-1}\lambda) \tag{35}
\]

\[
(\alpha | \gamma) \sim Ga\left(\frac{1}{2}, \frac{ss}{2}\right) \tag{36}
\]

**Spike prior** A spike prior uses the independent Bernoulli prior distributions on the inclusion indicators, where \(\gamma_j\) is an inclusion probability at set \(j\). If there is any set which \(\gamma_j\) is more than the threshold after the iterations, then variables in those sets will be included in the linear model. For example, if \(\gamma_j = 1.0\), then indicators in set \(j\) will very likely to be included. We initialize \(\pi = 0.5\) and use an inclusion probability threshold as 0.1. The inclusion probability for each \(k\) is evolved over time. If its estimation is closer to the measurement during the iteration, then its inclusion probability is going to be closer to 1.

**Slab prior** A slab prior is a conjugate normal-inverse-gamma distribution. The method to calculate the slab is the same as variational Bayesian linear regression. Section 3.3 shows this in details, especially, in Equation (28), (30), and (32). Note that a slab prior does not depend on \(\gamma\). We initialize it with \(\beta = 0\), \(a = 1\), and \(ss = (1 - R^2_{\text{expected}})s_y^2\) where \(s_y\) is the sample standard deviation of \(y\) and \(R^2_{\text{expected}} = 0.8\).

### 4.2 The MCMC exploration

We need to calculate the marginal posterior of \(\gamma\). Equation (37) shows the marginal posterior inclusion probability.

\[
\gamma | y = C(y)\frac{\Sigma_{\gamma}^{-1} \frac{1}{2} p(\gamma)}{|\Lambda_{\gamma}^{-1}|^{\frac{1}{2}} d_{\gamma}^{-1}} \tag{37}
\]

where \(C(y)\) is a normalizing constant, \(\Sigma_{\gamma}\) is an improved g-prior in Equation (26), \(\Lambda\) is described in Equation (29), and \(d\) is described in Equation (32).

Since the posterior distribution of the parameters does not have a closed form, \(C(y)\) is intractable to compute in high dimensional spaces. Instead, we calculated an approximate inference using the MCMC algorithm for each set of \(\gamma\).

### 4.3 Principal anomaly detection

In this section, we describe the principal anomaly detection. The principal anomaly detection [20] is a Bayesian way for statistical anomaly detection. The general idea in the principal anomaly detection is to build Bayesian model over all possible parameters and to compare new samples with posterior predictive distributions of the model when they arrive. Equation (38) shows the posterior predictive distribution of variational Bayesian inference.
Step 1: Loop over $j$, update $\gamma | y$.

Step 1.1: Loop over $k$, drawing each $\gamma_{(j,k)} | y, \gamma_{(j,-k)}$ using the MCMC algorithm, where $k$ is an index of all variables.

Step 1.2: Update $\gamma_j | y$ after the loop.

Step 2: Loop over $j$, update $\theta_j$ using variational Bayesian linear regression given $\gamma_j$, where $\theta_j$ is parameters of the model of set $j$.

Step 3: Repeat for many iterations.

Figure 5: Overview of the SSVS algorithm with the MCMC algorithm

$$
p(z | X, Y) = \int_{\theta} p(z | X, \theta)p(\theta | X, Y)d\theta
= \int_{\beta} \int_{\lambda} p(z | X, \beta, \lambda)p(\beta, \lambda | X, Y)d\beta d\lambda
= \int_{\beta} \int_{\lambda} p(z | X, \beta, \lambda)q(\beta)q(\lambda)d\beta d\lambda
$$

(38)

where $z$ is an unknown observation, and $\theta$ is parameters of the model,

Figure 6: Illustration of the principal anomaly of the posterior predictive distribution. The shaded area refers to the principal anomaly $A(z | \theta)$, and the blue line refers to $p(z | \theta)$.

**Principal anomaly**  The principal anomaly of an unknown observation $z$ is the probability of generating more common samples than $z$ from a posterior predictive distribution. An unknown observation will be considered as anomalous if it has a very small probability of being generated by posterior predictive distribution. Figure 6 shows the illustration of the principal anomaly. Equation (39) describes a principal anomaly $A(z | \theta)$. 

17
\[ A(z \mid \theta) = \int_{x \in \omega} p(x \mid \theta) d\omega \]  
\[(39)\]

where \( z \) is an unknown observation, \( \theta \) is parameters of the model, and \( \omega = \{x : p(x \mid \theta) > p(z \mid \theta) \} \).

**Detecting anomaly**  Note that we can measure the degree of abnormality by using the principal anomaly. The threshold is selected to get a fixed proportion of false alarms, and we use 95% credible interval as threshold in this project. In the example of Figure 6, new observation \( z \)’s probability \( p(z \mid \theta) \) is low, therefore \( A(z \mid \theta) \) would be considered as potential anomaly if a threshold for \( A(z \mid \theta) \) is low enough.
5 Experiments

In this section, we analyse the performance of the proposed algorithm. As we stated in Section 1.3, the result of the thesis project is a proof of concept and a performance measurement of the algorithm on real data. The performance of the method is evaluated on the real data generated from four different RNC devices — CC, DC, MC, and PDR — which are the most important devices of the RNC node. We will discuss the experiment setup in Section 5.1. In Section 5.2, 5.3, and 5.4, we illustrate and compare the accuracy of estimation performance. We trained and tested the model on the same node for the experiments, but it is only in Section 5.5 where we used separate nodes to train and test the model. In Section 5.6, we evaluated the proposed algorithm to detect contextual point and collective anomalies.

5.1 Setup

**Dataset**  The data obtained from the EIS Insight database at Ericsson contains the RNC traffic data from the live and the test nodes. The live nodes generate traffic data on the RNC nodes from various telecommunication operators, and the test nodes generate data from the overload protection, robustness, stability or other characteristic testing scenarios.

We analyzed time series by giving the resource load of the four devices using data from ten live nodes from three different operators within the period from 2nd March 2015 to 29th May 2015. Each data contains the traffic events as well as resource loads from the four devices. The sampling interval is 15 minutes. The measured resource load is a mean value within the period from the devices. For example, in the case of the DC device’s resource load, a measured resource load is the mean value from nine DC devices. Within a 24-hour period, we are able to fetch 96 data samples corresponding to the number of rows.

Since there are too many candidates of contextual attributes, we established 23 traffic events as potential regressors. As a result, we are able to refine the traffic indicator’s range between 1138 and 2180 in a given node. Table 2 represent the number of devices per board in the RNC node. In our setting, the measured loads of each device is a mean value of the same devices, and therefore the variance of residuals values will be higher if the number of device is larger. For example, if the RNC consists of only one board, a measured load of the DC device is a mean value of nine different DC devices at the same time.

<table>
<thead>
<tr>
<th># device per board</th>
<th>CC</th>
<th>DC</th>
<th>MC</th>
<th>PDR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>9</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

*Table 2:* The hardware setting of the RNC.

For the purpose of preprocessing the data, we removed all the data that includes any missing regressors or device resource loads, and converted all of the values by standard score as in Equation (40).

\[
\bar{x} = \frac{x - \mu}{\sigma} \tag{40}
\]

where \(\bar{x}\) is standardized \(x\), \(\mu\) is the mean of the sample population, and \(\sigma\) is the standard deviation of the sample population.
Architecture  We construct the system similar to the knowledge discovery system [30]. Figure 7 shows a block diagram that represents the method. The proposed method is implemented in R and C++11 language.

![Figure 7: Knowledge Discovery System](image)

To check its estimation performance, we measured an average forecast error, which is a median value of the absolute differences between the measured loads of the device and estimated loads of the model. Since we followed a probabilistic approach, the forecast errors would be different for each time. We use a boxplot to show the distributions of average forecast errors. More details on all elements of the plot will be explained in each section.
5.2 Estimation performance

![Figure 8](image)

**Figure 8**: Above figure shows a time series plot comparing an estimated load against a measured load. Vertical line refers to the border between a training set and a test set. We used 96 data points to train the model, and tested it with another 384 data points. The black line indicates the measured loads, and the red dot line indicates the estimated loads from the model. The shaded area refers to the posterior predictive distribution with 95% credible intervals. Below figures shows a resource load of a CC device and two traffic events selected by the proposed algorithm.

Figure 8 shows a CC device of a live RNC node as an example. This time series plot shows the CC device resource load from the 2\textsuperscript{nd} of March to the 6\textsuperscript{th} of March 2015. It compares measured resource loads and estimated loads from the trained model with the selected regressors. The estimation is from the regression model based on the best indicators from the groups where inclusion probability is more than 0.1. We used a set of traffic events $D_{\text{train}} = (d_1, \cdots, d_{96})$ to train the model, and to test the model with upcoming four days $D_{\text{test}} = (d_1, \cdots, d_{384})$. In this case, the median value of residual errors is 0.38 in this case.

Figure 9 shows a histogram of model size for each device in ten live nodes. We used a set of traffic events $D_{\text{train}} = (d_{i, 1}, \cdots, d_{i, 96})$ to train models where $i \in (1, \cdots, 10)$. 300 models are used for each device. Since the inclusion probability threshold is 0.1, the expected value of the number of selected indicators are approximately $2 \approx 23 \times 0.1$ indicators in average. A median value of the CC and the PDR device is similar to the expected number of indicators, however, the DC and the MC devices are close to 3 or even higher. It shows the true number of the best indicators.

Figure 10 shows the distributions of forecast errors. We used a set of traffic events
**Figure 9:** Histogram of model size

**Figure 10:** Estimation performance comparison for four devices of ten live nodes. All elements of the plot are in Table 3
$D_{i,\text{train}} = (d_{i,1}, \cdots, d_{i,96})$ to model, and to test the model in the upcoming four days $D_{i,\text{test}} = (d_{i,1}, \cdots, d_{i,384})$ where $i \in (1, \cdots, 10)$.

Median values and variances from the results are described in Table 3. Clearly, trained models of the PDR devices generate closer estimation than that of the CC devices and the MC devices which appear to be roughly equivalent to each other in terms of their medians. However variance of the MC devices is slightly higher than that of the CC devices, and the estimation of the DC devices is less accurate and effective than the others. That is due to the difference between the expected number of indicators and the true number as we can see in Figure 9. Also, the number of the DC device is 3 or 9 times more than that of other devices so variance of residuals are much higher than others.

<table>
<thead>
<tr>
<th>Device</th>
<th>-1.5 IQR</th>
<th>25% quartile</th>
<th>Median</th>
<th>75% quartile</th>
<th>+1.5 IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>0.133</td>
<td>0.158</td>
<td>0.313</td>
<td>0.385</td>
<td>0.556</td>
</tr>
<tr>
<td>DC</td>
<td>0.192</td>
<td>0.310</td>
<td>0.548</td>
<td>0.729</td>
<td>1.187</td>
</tr>
<tr>
<td>MC</td>
<td>0.071</td>
<td>0.139</td>
<td>0.267</td>
<td>0.379</td>
<td>0.614</td>
</tr>
<tr>
<td>PDR</td>
<td>0.046</td>
<td>0.068</td>
<td>0.085</td>
<td>0.103</td>
<td>0.103</td>
</tr>
</tbody>
</table>

Table 3: Results of average forecast errors from the devices.

The PDR device is interesting since it shows good estimation performance even though its model size mean and variance is roughly similar to that of a DC device. It shows how much the PDR devices behave differently than other devices depending on their conditions. The PDR case will be discussed later in Figure 13.

The results from the analysis shows that: the proposed algorithm is able to select variables that has the most predictive powers. Furthermore, if the expected number of indicators by the algorithm is different from true number of indicators, then the estimation performance becomes less accurate and less effective. Finally, even though the setting is the same, the selections can be different depending on various conditions of network node.

### 5.3 Inclusion probability

To have a better understanding of the variational Bayesian inference, we fit the linear models over 10 live nodes $i$ with different inclusion probability threshold $p$ from 0.1 to 0.5. We used a set of traffic events from 24 hours $D_{i,\text{train}} = (d_{i,1}, \cdots, d_{i,96})$, with different $p$, and tested the model for the four subsequent days $D_{i,\text{test}} = (d_{i,1}, \cdots, d_{i,384})$ where $p \in (0.01, \cdots, 0.5)$ and $i \in (1, \cdots, 10)$.

Figure 11 shows the average forecast errors — a median value of absolute differences between the measured and estimated load of the model — over different thresholds on inclusion probability. From the result, we can see that the DC device is the most complex one, and the estimation performance becomes accurate as threshold increases. There are two reasons why the device is complex. Firstly, it is a complex model that has many factors which influence the DC device. Secondly, the number of the DC device is 3 or 9 times higher than that of other devices, therefore residuals are higher.
5.4 Model robustness

This case is identical to Section 5.3, but it contains with different data points $p$ of training sets from 16 to 96. Inclusion probability threshold is 0.1. We used a set of traffic events $D_{i,\text{train}} = (d_{i,1}, \cdots, d_{i,p})$ to train the model, and tested it in the four subsequent days $D_{i,\text{test}} = (d_{i,1}, \cdots, d_{i,384})$ where $p \in (16, \cdots, 96)$ and $i \in (1, \cdots, 10)$. Figure 12 shows the average forecast error over different number of training sets for each device. CC, MC, and PDR device shows less than 1.0 median forecast error at 26 which is less than the number of traffic events. However, in the case of the DC device, which is the most complex device, it requires more data points than other devices. Also, the CC device, with a smaller number of model size than others, converges first. As a result, the amount of required training data is going to be increased if the number of ground truth regressors are higher than the expected number of the model.

![Figure 11: Average forecast error for each device.](image1)

![Figure 12: Average estimation error for four devices from SSVS model for ten live nodes.](image2)
5.5 Finding similar nodes

In this investigation, we have presented experiments on finding similar nodes. The RNC nodes work differently than each other even though they have the same software version. Apart from the software version, the node works differently depending on subscribers’ traffic pattern, the hardware setting, and software configurations as we discussed in Section 2.4.

Figure 13 shows the PDR case as an example. The PDR device is designed to handle data packet connections and transmitting events. The proposed algorithm is supposed to select data packet related traffic events more than others. However, the device also shares its resource with OS, and therefore if the PDR device is underloaded, the unrelated traffic events will be selected.

![Figure 13: Example of PDR device](image)

The proposed algorithm selects the best indicators for each node, so we can make use of those selected indicators to find network nodes that are similar to each other. The reference node, which is a node used to train the model, and the target node to test the model, are different in this experiment. We assume that the two nodes are similar if they share the same best indicators, the same software version, and the same number of devices. In this experiment, we tested whether or not the proposed algorithm can estimate the resource loads of the target node by the model trained through the reference node. In case of Figure 13, we cannot estimate resource loads of the target node from the reference node since their
Figure 14: Estimation performance comparison, among the model trained by similar nodes, for four devices of ten live nodes.

<table>
<thead>
<tr>
<th>Device</th>
<th>-1.5 IQR</th>
<th>25% quartile</th>
<th>Median</th>
<th>75% quartile</th>
<th>+1.5 IQR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>0.125 (0.93)</td>
<td>0.198 (1.25)</td>
<td>0.396 (1.26)</td>
<td>0.880 (2.28)</td>
<td>1.291 (2.32)</td>
</tr>
<tr>
<td>DC</td>
<td>0.335 (1.74)</td>
<td>0.410 (1.32)</td>
<td>0.590 (1.07)</td>
<td>1.131 (1.55)</td>
<td>1.566 (1.32)</td>
</tr>
<tr>
<td>MC</td>
<td>0.109 (1.53)</td>
<td>0.202 (1.45)</td>
<td>0.332 (1.24)</td>
<td>0.618 (1.63)</td>
<td>1.180 (1.92)</td>
</tr>
<tr>
<td>PDR</td>
<td>0.079 (1.71)</td>
<td>0.095 (1.39)</td>
<td>0.110 (1.29)</td>
<td>0.126 (1.22)</td>
<td>0.165 (1.60)</td>
</tr>
</tbody>
</table>

Table 4: Comparison between results from same nodes and similar nodes. Numbers in parentheses are a ratio between similar node case and same node case.
selections are different. In our setting, 4 out of 10 nodes have similar nodes.

Figure 14 shows the distributions of the forecast errors. 96 data points of a reference node are used to train the model, and 96 data points of a target node are used to test the model. The inclusion probability threshold is 0.1. Table 4 shows the average forecasting errors of devices from similar nodes.

The average forecast errors of the model trained by the similar node have, in average, increased by 21% than that of the model trained by the same node. We illustrated this in Table 3, and its quartiles and IQR values have also increased. Furthermore, the errors are higher than that of the same nodes. However, 75% quartile and 1.5 IQR forecast errors of the CC device have increased 2.28 and 2.32 times.

We suspect the reason is because the expected number of indicators in average is only 1.88 while the others are 2.92, 3.35, or 2.72. The lack of proper indicators will lead the model to have more noises.

5.6 Detecting contextual anomaly

This experiments illustrates how Bayesian contextual anomaly detection performs in the real world. The settings are the same as in Section 5.2, and a principal anomaly threshold of 0.95 is used. Since the proposed model can estimate resource loads of devices with its credible intervals, the model can also conclude whether newly arrived sample is unlikely to occur or not.

Figure 15 presents as an example of contextual point anomaly detection of a PDR device generated from the April 21-23, 2015. A resource load at the 110th time index shows an abnormal behavior, and the model concludes it as an anomaly given the evidence that the posterior predictive distribution with 95% credible intervals.

Figure 15: Example of contextual point anomaly

Figure 16 shows a contextual collective anomaly detection example of a MC device from the test node under an overload scenario test. The scenario is critical to the RNC node because the node cannot handle all the connection attempts instantly if it is overloaded. Overload protection process will run to prevent the node from being frozen by overloaded device. Left figure shows the measured and estimated loads from the model. The node is supposed to handle all the attempts, but right after it is overloaded, most of the attempts were not successfully handled by the node to protect the resource of the device. We can see that
the proposed model can detect the impact of the process when measured loads become higher than 95% credible interval. Right figure shows the cumulative errors between measured and estimated loads. We can see three different stages are in the test period — before it happens, while the process is running, and after it stopped. By using this figure, we can measure how much collective anomalies have impacted the device, and how the impact have decayed over time.

**Figure 16:** Example of contextual collective anomaly. Left model shows detection of contextual collective anomaly on overload scenario test. Right model shows cumulative absolute errors.
6 Discussion

The thesis work's major findings and limitations are summarized in the chapter.

6.1 Major findings

Firstly, the experiments show that the model can estimate the load and detect anomalous patterns with regards to its context. Since the approach for learning the linear relationship, the model can estimate resource loads irrespective of the shape of time series, such as daytime, midnight, weekend or holiday. If the measurement is out of the 95% posterior predictive distribution, then we can conclude result as a network anomaly, such as defects of the updated software or hardware setting maintenance. The selected traffic events can be quite different from each node even though they share the same software version.

Secondly, cumulative errors from the proposed model shows when a collective anomaly happens and how it decays. We found that the slope of forecast errors is constant. If an anomalous pattern occurs, however, then the slope of accumulating error changes and becomes constant until the impact ended. Figure 16, for example, shows that it is clear to see three different stages are in the forecast errors.

Thirdly, we found that the act of selecting traffic events is the key to find similar nodes. Two RNC nodes will be similar if the traffic events selections are the same. Table 4 shows that the estimation from similar nodes can estimate the resource loads of an RNC node. However, if the number of selected indicators for the device is too small, the results will have more noises than that of the model from the same node.

6.2 Limitations

Firstly, it may be hard to find the RNC that has consistent setting. For example, in Section 5.5, only 4 out of 10 network nodes have similar nodes. If we do not have a good reference node, then the proposed algorithm does not provide a way to estimate the resource load of an unknown network node unless we fetch a training set for the node.

Secondly, the proposed algorithm assumes that there is no time-varying regression coefficients. In this setting, as in Section 5.2, the cost of traffic events is not time-varying, therefore we used static regression coefficients in this thesis work. However, if one is trying to handle variables that require time-varying regression coefficients, also known as dynamic regression coefficients, then the proposed algorithm would be less effective.

Finally, although a spike-and-slab prior are designed to prevent us to select the wrong indicators, the model may still contain some spurious indicators among the best indicators. Furthermore, the model may also contain some predictors that have better predictive power, but has no causal relationship at all. We may need to judge the selection by domain experts to restrict potential candidate indicators. To solve this problem, we can use the prior probability distributions initialized by the information that we know as a remedy, which is commonly used in the Bayesian methods. Also, we need to pay attention to inclusion probability threshold which may lead us either underfitting or overfitting.
7 Conclusion

**Summary**  Bayesian approach for contextual network anomaly detection is presented. The proposed algorithm selects the best indicators that have the most predictive powers by variational Bayesian inference. The algorithm also enables us to estimate the network node activity and conclude possible anomalous patterns. Contextual anomalies, both point and collective, can be captured by the posterior predictive distribution constructed by the proposed algorithm. Selected indicators can help us to understand the node and to find similar nodes. Furthermore, the proposed method is fully Bayesian.

**Future work**  There are a few possible ways for future work. It will give a huge contribution if we can estimate network node activity by dissimilar network nodes. More specifically, we may research automatic ways to estimate scaling factor between network node activities. Also, it would be interesting to design an algorithm with more robust estimator, which has an influence on the practical utility. The interesting problem that we would like to research is to impute missing covariates. This will contribute to the project since the database is often affected by the condition of the node. For example, network nodes fail to log into the database if the node is overloaded. In addition, an extension of variational Bayesian method study of anomaly detection is also worth considering.
8 References


Acknowledgements

I wish to thank Anders Holst for giving me an opportunity to understand this subject not only for this project but others before it. I am grateful to Ian Marsh who gave me helpful feedbacks and advices on directions for experiments. For the comments on drafts of this thesis, I thank Anders Holst, Ian Marsh, Adam Hjerpe, and other readers especially Desmond Wong for correcting grammatical errors.

I would like to thank Ericsson AB for supporting the research. This work in this thesis was carried out at the company, with kind helps from the RNC experts also known as the Guardians. I would like to thank Saeid Cedighi, who worked extensively with me on the work, for letting me understand the RNC and get on the right track by his expertise in the RNC. Also, I wish thank to Pär Karlsson for providing test node data and the ADAT team especially Paolo Elena for reviewing experiment results.