Aalto University
School of Science and Technology
Faculty of Information and Natural Science
Department of Information and Computer Science

Li Yao

Anomaly Detection and Location with an Application to an Energy Management System

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, November 01, 2010

Supervisor: Professor Olli Simula
Instructor: Francesco Corona, D.Sc. (Tech.)

AALTO UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY

ABSTRACT OF THE
MASTER'S THESIS

Author:	Li Yao
Name of the Thesis:	Anomaly Detection and Location with an Application to an Energy Management System
Date:	November 01, 2010 **Number of pages:** xiii + 87
Department:	Department of Information and Computer Science
Professorship:	T-115 Computer and Information Science
Supervisor:	Professor Olli Simula
Instructor:	Docent Francesco Corona

This work is motivated by the requirement of misconfiguration detection in the energy management system. A configuration (instance) specifies the contract information between sellers, the energy companies providing electricity, gas and district heating, and buyers, the consumers purchasing the energy service. The correctness of an instance is critical because a misconfiguration (anomalous instance, or anomaly) may cause a dramatic loss for an energy company. Finding anomalies inside all instances is the focus of this work.

In order to report anomalies, data analysts normally have to face two constraints. First of all, there are no labels available showing whether an instance is anomalous or not. In addition, there is no domain knowledge of the relationships among the attributes of an instance. This work, under the two constraints, performs anomaly detection and location with the assumption that anomalies appear with low frequencies in the data set. Based on this assumption, the method reports for each instance its anomalous attributes which potentially cause the instance to be an anomaly. In order to locate the anomalous attributes, the method relies on the factorization of the joint probability of an instance. The method learns a Bayesian network from the categorical data set and factorizes the joint probability into a group of factors defined by conditional probabilities among attributes. Then checking each attribute based on the frequency-based assumption reveals whether it is normal or anomalous.

This work extends anomaly detection discussed in the current literature into anomaly location in which "masks" indicate the root of an anomaly. Furthermore, given little domain knowledge, the method statistically extends one's knowledge by constructing a Bayesian network that models the data set and indicates important business rules of the application domain.

Keywords: anomaly detection, Bayesian network, structural learning, Markov chain Monte Carlo, categorical data

Acknowledgments

This thesis was made in collaboration with Process Vision, a Finnish company specialized in information systems and applications for energy business. Dr. Pasi Lehtimäki and Mr. Sami Niemelä from Process Vision had been offering indispensable insights on their data set.

I would like to express my gratitude to my instructor Dr. Francesco Corona for providing me with invaluable guidance during the whole process. I would also like to thank Prof. Olli Simula for his supervision and for providing me with excellent working facilities in Department of Information and Computer Science at Aalto Univeristy.

Furthermore, I would like to thank Dr. Tapani Raiko, Dr. Alexander Ilin, Dr. Elia Liitiäinen, Mark van Heeswijk for the inspiring discussion during the work. All the members of Environmental and Industrial Machine Learning research group deserve special thanks for creating a positive working atmosphere.

I want to thank my family for their support during my study. Special thanks are reserved for my parents for encouraging me all these years.

Espoo, Nov. 1st, 2010

Me

Contents

Abbreviations and Notations

$\mathbf{X}$	$N \times K$ data matrix
N	Number of instances (observations)
K	Number of variables (attributes)
X_k	k-th variable, with $k = 1, \ldots, K$
$\mathbf{x}_n$	n-th observation $\mathbf{x}_n = \{x_{n1}, \ldots, x_{nK}\}$, with $n = 1, \ldots, N$
$\mathbf{r}_n$	Mapped n-th observation in the frequency space
$\mathbf{p}_n$	Mapped $\mathbf{r}_n$ in the log likelihood space
$\mathbf{T}$	Transition matrix of Markov chain
T	Threshold on frequency
P	Both probability and probability distribution
$\mathbf{m}_n$	K dimensional mask for $\mathbf{x}_n$
$\mathbf{B}$	A Bayesian network for $\mathbf{X}$
$\mathcal{D}$	Probability distribution encoded by $\mathbf{B}$
g	Network structure of $\mathbf{B}$
θ_g	Network parameters (conditional probability distributions) of $\mathbf{B}$

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and objectives

In the energy industry, the term "configuration" is conventionally adopted to describe a contract between a specific seller, the company selling energy, and a specific buyer, the consumer of the energy. Every contract is recorded in the energy management system which is used to configure the related hardware to serve the business. From the perspective of sellers, they would like to produce and deliver energy to buyers with purchased energy type and geographical location specified in the contract. Most importantly, sellers would like to correctly measure the energy consumption of their buyers in order to have a reference to charge them. Both the delivery and measurement are deployed and collected according to a configuration. The correctness of a configuration is critical since an error may cause a dramatic loss for the energy company.

Given a collection of configurations, one of the primary tasks for the data analyst is to detect those that are correct and those that are not. The task can be referred to as "misconfiguration detection" or, in a more general sense, "anomaly detection". Another task is to diagnose a misconfiguration by indicating the position from which the anomaly is potentially generated. The task can be denoted as "misconfiguration location" or "anomaly location". Often, little or no knowledge on the application domain is available, meaning that the analyst does not necessarily have information about the correctness of a configuration and the relationships existing between its attributes.

Motivated by a practical application to energy management systems, in this thesis we investigate an approach to address the anomaly detection and location problem. Concretely, we introduce a procedure for detecting anomalies in multivariate categorical data sets. In addition, the procedure suggests the location of the anomaly within a configuration, thus providing a useful tool for inspecting and correcting it.

1.2 Overview and organization of the thesis

In a categorical data set, each row corresponds to an instance (or configuration) and each column corresponds to an attribute (variable) of the instance; each instance takes a categorical value in each of its attributes. Anomalous instances can be considered as uncommon events, anomaly detection reports whether an instance is anomalous or not. For an anomalous instance, anomaly location reports which of its attributes can be considered as uncommon events. This suggests that normal instances are more frequent than anomalies and normal attributes are more frequent than anomalous ones. An intuitive approach for anomaly detection and location can, thus, be based on frequency analysis. For example, we can examine the number of appearances of an instance and compute its joint probability; if the instance has a low joint probability (not frequent), it is very likely to be an anomaly. However, detection is not able to indicate, inside an instance, which attributes are likely to be anomalous; this is because detection alone does not exploit the relationships among attributes. On the other hand, anomaly location explicitly exploits these relationships using conditional probabilities; an attribute with a low conditional probability is likely to be anomalous. Therefore, anomaly location is more general than anomaly detection because, after locating the anomalous attributes, one can draw the conclusion that an instance with any anomalous attribute is anomalous while an instance without any anomalous attribute is normal. For such reasons, this thesis focuses on anomaly location and discusses a novel probabilistic procedure to approach the task.

Clearly, locating anomalous attributes inside an instance requires information on conditional dependencies. This information is usually not accessible for analysis because an exhaustive list of all the "rules" existing between the attributes is not available. In this thesis, we assume that this information can be extracted by factorizing the joint probability of an instance in a number of factors equal to the number of attributes. Each factor is the conditional probability of that attribute given some other attributes. Such a factorization offers a possibility to examine the factors instead of considering the joint probability as a whole. For the task, it is critical to obtain a factorization that best describes the conditional dependencies among attributes; that is, a factor should explain how the attribute it corresponds to is dependent on the other attributes. Such a factorization can be realized from data by learning a Bayesian network where the conditional dependencies/independencies among attributes are represented as a directed acyclic graph. Given the factorization, locating an anomaly corresponds to assigning to each instance a "mask" indicating whether its attributes are anomalous or not. Masks are assigned by comparing instances with a number of prototypical configurations also defined from data.

The thesis consists of two parts: the first part overviews the problem of anomaly detection and introduces a procedure for anomaly location, the second part supports the presentation with an application on a real-world problem of analyzing configurations in an energy management system. In Part I (Theory), Chapter 2 briefly reviews the techniques and applications of anomaly detection, Chapter 3 discusses

in detail the generation of a "mask" from a factorization and Chapter 4 focuses on obtaining the factorization by constructing a Bayesian network using the Markov chain Monte Carlo (MCMC) structural learning technique. Toy examples are provided throughout Chapter 3 and 4 to demonstrate the ideas and principles. In Part II (Application), Chapter 5 introduces a data set from the energy industry and in Chapter 6 the procedure is applied to the data set followed by the analysis of the results.

Part I

Theory

Chapter 2

Review of Anomaly Detection

Anomaly detection is an important topic that has been studied in various research and application domains. This chapter gives a cursory review of this developing field and establishes a big picture of approaching a typical anomaly detection problem.

2.1 Different aspects of an anomaly detection problem

As discussed in [73], when dealing with an anomaly detection problem, data analysts need to take into account several aspects that determine the applicability of any technique. First issue to consider is the nature of the data set. Conventionally, a data set is a collection of instances (or observations, cases, entities) each of which is described by a set of attributes (or variables, features, dimensions). The attributes may be of different data types including binary, categorical or continuous. Each instance may have more than one variables. In the latter multivariate case, there can be a mixture of categorical and continuous attributes. Sometimes an instance may include one or more missing values. The tools available rely on the nature of the data. For instance, it is dubious to compute the Euclidian distance on binary attributes since binary attributes are not traditionally considered to be distributed in the real-valued and continuous Euclidian space. Thus tools available to continuous attributes need to be refined for categorical data sets. Furthermore, from the perspective of the relationships between instances, two neighboring instances may be correlated, which means the order of those instances contains important information of the underlying process generating the data. On the contrary, instances may be independently and identically distributed (i.i.d.). In all, different nature of a data set may lead to substantially different approaches to solve the anomaly detection problem.

Secondly, considering the type of anomalies, there are two categories:

- *Contextual anomalies* (or conditional anomalies as in [68]): An observation can be either normal or anomalous depending on the context or its neighborhood. A striking example is an observation from a time-series of stock price. This

observation is highly correlated with its neighbors and is meaningful only within its neighboring time period. It is the context that makes it normal or anomalous. This type of anomalies are most common in time-series data discuss in [64] and [77].

- *Point anomalies*: This is the most common case in which anomalies are recognized with respect to the rest of the data. Unlike the contextual anomalies, there is no situation in which an anomalous instance becomes normal in different contexts.

Thirdly, the availability of a *label* that indicates whether an instance is normal or anomalous has a major impact on the methods used to detect anomalies. *Supervised* methods make use of the labels to train a predictive model that covers normal instances as well as anomalous ones. An unseen instance is then compared against the trained model and it is associated with either label, thus being classified into either normal or anomalous cases. There exist several major issues in supervised approaches. One of them is the imbalanced availability of labels between normal and anomalous cases since anomalous cases, in most of the real world applications, are and should be, by definition of anomaly, rather rare. This problem has widely been investigated in [17] and [61]. In [71], the authors discussed another issue of obtaining the accurate and representative labels and propose to inject artificial anomalies into a normal data set to obtain a labeled training data set. However, if training data has only the normal cases labeled but not the anomalous ones, the *semisupervised* methods, dicussed in [22], train a predictive model only for normal data and try to differentiate anomalous instances by using the trained normal model. Besides supervised and semisupervised approaches, *unsupervised* techniques do not require any *label* and are widely applicable. Unsupervised methods are based on the assumption that normal instances are far more frequent than the anomalies in the data set. It suffers from high false alarm rates when this basic assumption does not hold. It should be noted that obtaining labels for a standard training set is always expensive and needs substantial effort. Moreover, the behavior of anomalies is dynamic and it becomes infeasible to label them whenever new types of anomalies appear.

Finally, after an anomaly is detected, it is important to decide how to report it. In some situations, it suffices to assign a *label* coded either normal or anomalous to a tested instance. However, in other situations, each instance gets a *score* assessing, for example, how much it behaves like a normal case. A threshold is needed to establish a boundary in the ranked scores, which inevitably includes a degree of arbitrariness and demands a certain amount of domain knowledge.

2.2 Techniques

Detecting anomalies in a data set has been studied in the statistics community since the 19^{th} century starting from [26] and has been the topic of a number of surveys and review articles as well as books. In [41], the authors provide an extensive sur-

vey of anomaly detection techniques developed in machine learning and statistical domains. A broad review of anomaly detection techniques for numeric as well as symbolic data is presented in [4]. Anomaly detection has become a multi-dicipline subject involving such research as machine learning, data mining, statistics, information theory motivated by application in several domains including Cyber-Intrusion detection, fraud detection, medical anomaly detection, industrial damage detection, image processing, textual anomaly detection and sensor networks. We illustrate some of the most important anomaly detection techniques in this section.

2.2.1 Classification

Classification is one of the most well researched field in machine learning and data mining, being the main topic of several classic books such as [6] and [11]. It learns a model from a set of labeled training data and then assigns labels to the unlabeled part of the data set. In anomaly detection, it is either possible to build models for both normal and anomalous instances or we model the boundary between those two. Whichever case it is, the universal assumption in building a classification based method is that there exist models that separate normal from anomalous instances.
.

- Neural network. Neural network, comprehensively discussed in [37], has been adapted to fit into the task of anomaly detection by following two steps. Firstly, a multilayer feedforward neural network is trained on normal instances with labels. During the training, the weights are adapted to minimize the classification error. Secondly, unlabeled instances are input into the network and then classified into two classes (normal and abnormal) according to its output from the network. There are several variants of multilayer feedforward neural network being used in anomaly detection such as Replicator Neural networks in [36] and Oscillatory Networks in [40].

- Support Vector Machine (SVM). Introduced in [76], SVMs have been applied to anomaly detection in the one-class setting in [63]. Kernels, such as radial basis function (RBF) kernel, can be used to learn much more complex regions than a plane (hyperplane if the dimensionality of the plane is more than two) to separate normal instances from anomalous ones. Robust Support Vector Machine (RSVM), which uses the distance between each data points and the center of a class to calculate the adaptive margin, introduced in [67], has been applied to the anomaly detection in computer security.

- Association rule. This set of techniques are of special interest when a data set is made of only categorical values. It tries to learn the rules that capture the normal behavior of a system. Given a set of learned rules, an instance that violates rules is considered as an anomaly. Association rule mining techniques, introduced by [3], have been used for one-class anomaly detection by generating rules from the data in an unsupervised fashion. Every discovered rule is

associated with a frequency indicating whether it is a strong rule or a weak rule. However, the set of rules generated by those algorithms are usually of large size and it is usually difficult to determine the threshold to differentiate strong rules from weak rules, as discussed in [73]. Most of the applications make use of the discovered rules and their frequencies in various fashion.

All classification-based methods, including the rule-based association rule mining, rely on the availability of the training labels and need specific domain knowledge to label the training data or set the threshold, which limits the application of this category.

2.2.2 Nearest neighbor

As a class of unsupervised techniques, the nearest neighbor-based methods differ themselves from others based on the assumption that normal data instances occur in dense neighborhoods while anomalies occur far from their closest neighbors. As one may expect, it is critical to define a metric to measure the distance between instances. For continuous attributes, Euclidean distance is a popular choice. However, as demonstrated in [73], other types of metric are also available if defined properly. For categorical attributes, a simple Hamming distance that measures between two instances the number of positions having different values in the same attributes is useful. However, even more complex distance metrics may be used depending on the specific data mining task, as discussed in [14].

In this set of methods, the anomaly score of a data instance is defined as the average distance to its φ nearest neighbors in a given data set. This basic technique has been applied to anomaly detection in satellite images in [16] with its variants focusing on improving the computational efficiency of the basic techniques by using a nested loop algorithm leading to a near linear time efficiency after some pruning, as discussed in [10]. Density-based anomaly detection estimates the density of the neighborhood of each data instance. An instance that lies in the neighborhood with low density is declared to be anomalous and otherwise normal. Since the average distance to the φ nearest neighbors for a given data instance can be viewed as an estimate of the inverse of the density of the instance in the data set, the nearest neighbor techniques can be considered as density-based methods. As an improvement to the density methods, the authors of [15] assign an anomaly score to a given instance, known as Local Outlier Factor (LOF). For any given data instance, the LOF score is equal to the ratio of the average local density of its φ nearest neighbors and the local density of data instance itself.

Even though improved under some circumstances, it remains a drawback that the nearest neighbor based techniques has the complexity of $\mathcal{O}(N^2)$ (N is the number of instances) since these techniques normally involve finding nearest neighbors for each instance. Another disadvantage is that if data has normal instances that do not have enough close neighbors, or if the data has anomalies that have enough close

neighbors, the techniques might fail to label them correctly.

2.2.3 Clustering

Cluster analysis or clustering is the assignment of a set of observations into subsets (clusters) so that observations in the same cluster are similar and observations from different clusters are different. Clustering techniques, extensively discussed in literature such as [11] and [6] on statistics and machine learning require a similarity metric between instances. Even though some semisupervised clustering techniques can be found in [9], clustering mainly focuses on an unsupervised fashion in which instances fall into different groups. Clusters are formed by using the similarity metric and the selection of the metric follows the same discussion as nearest neighbor techniques. The main difference between these two types of methods, however, is that clustering evaluates each instance with respect to the cluster it belongs to while nearest neighbor methods analyze each instance with respect to its local neighborhood. Based on different assumptions, clustering-based anomaly detection can be divided into 2 categories.

- Assumption 1: Normal data instances belong to a cluster, while anomalies do not belong to any cluster. Several clustering algorithms do not force every data instance into a cluster such as DBSCAN in [28], and ROCK in [35]. The disadvantage of such techniques is that they are not optimized to find anomalies, since the main aim of the underlying clustering algorithm is to find clusters.
- Assumption 2: Normal data instances lie close to their closest centroid, forming dense clusters, while anomalies are far away from their closest cluster centroid, forming sparse clusters. It follows two steps. Firstly, the data is clustered using a clustering algorithm. Secondly, for each data instance, its distance to its closest cluster centroid is considered as its anomaly score. Following the two steps, the authors in [66] study thoroughly Self-Organizing Maps (SOM), K-means Clustering, and Expectation Maximization (EM) to cluster the instances and then use the clustering memberships to label them. In particular, SOM has been widely used to detect anomalies in intrusion detection in [66] and [8].

The merit of clustering-based techniques is that they operate in an unsupervised mode. After establishing the clusters, the labeling part of the algorithm is fast since each given instance needs only to be checked against each of the clustering centroids. Since there are 2 different assumptions guiding the methods used to detect anomalies, it is usually not quite clear which one holds in the given data set. Another major disadvantage is its computational complexity when data set is large. Most of the clustering algorithms have $\mathcal{O}(N^2K)$ where N is the number of instances and K is the dimensionality.

2.2.4 Other techniques

There exist several other anomaly detection techniques that do not belong to any of the three categories discussed above. They are based on different assumptions as well. If the assumption holds that data is generated from a particular distribution, statistical techniques provide a statistically justifiable solution for anomaly detection. While building the underlying distribution generating the data set, parametric techniques estimate the distribution from the given data as in [27]. If such an assumption does not hold and there is no *a priori* knowledge of any underlying distribution, non-parametric techniques such as histogram assuming no distribution can be used as shown in [23]. If the assumption holds that anomalies in the data set induce irregularities inside the information content of the data set, in [7] the authors apply information theory to implement an optimization that minimizes the irregularities in the anomaly-contaminated data set. If it holds that data can be embedded into a lower dimensional subspace in which normal instances and anomalies appear significantly different, projection-based anomaly detection techniques can be used such as Principle Component Analysis (PCA) in [44] and Linear Discriminant Analysis (LDA) in [56].

2.3 Applications

Anomaly detection, due to the fact that anomalies lead to significant information, has been used in many real world applications. For instance, an anomaly in network traffic may suggest a malicious intrusion by computer hackers who provoke potential threats to the security of a network. In medical diagnosis, an anomaly may reveal a critical part of a human body having cancer. In finance, credit card transaction frauds apart from normal usage may indicate a theft or other criminal behaviors.

2.3.1 Intrusion detection

Intrusion detection, as introduced in [60], refers to detection of malicious activities (break-ins, penetrations) in a computer system. The main challenges in this field is the huge amount of data. So most intrusion detection techniques need to be computationally efficient in order to handle the online streaming of the network data. The existing techniques for intrusion detection include statistical profiling using histograms in [55] and [22], statistical modeling in [27] and [74], artificial neural networks in [33], SVM in [43] and rule-based system in [48].

2.3.2 Fraud detection

Fraud detection refers to detection of criminal activities occurring in commercial organizations such as banks, credit card companies, insurance agencies, cell phone companies, stock market. The anomaly is defined as the malicious and unauthorized behavior. This field covers extensively large real world domains under the term of *activity monitoring* introduced by the authors of [29]. The most prominent work has

been done in credit card fraud detection, mobile phone fraud detection, insurance claim fraud detection and insider trading detection. Recent contribution includes [13] focusing on clustering in credit fraud detection, [1] on parametric statistical modeling on mobile phone fraud detection, and [24] using statistical profiling on insider trading detection.

2.3.3 Medical and public health anomaly detection

Anomaly detection in medical and public health domains typically works with patient records. The anomalies can be abnormal patient condition, instrumentation errors, or even recording errors. The main challenge is that medical anomaly detection is a very critical domain requiring high degree of accuracy and data is typically consisted of a mixture of data types including age, blood type, weight and even temporal such as Electrocardiograms(ECG) and Electroencephalograms(EEG). The most recent efforts devoting on this specific field can be seen in [49] using nearest neighbor based techniques and [78] using Bayesian networks.

2.3.4 Industrial damage detection

Industrial units suffer damage due to continuous usage and normal wear and tear and such damage needs to be detected early to prevent further losses. Data mostly comes from sensors and this field can be further divided into 2 categories: Fault Detection in Mechanical Units that deals with defects in mechanical components such as motors, engines and so on and Structural Defect Detection that deals with defects in physical structures. For the first category, the authors of [30] and [45] use spectral-based methods and parametric statistical modeling. For the latter category, the authors of [42] focus on the mixture of statistical models.

2.3.5 Image processing

Anomaly detection techniques dealing with images are either interested in any changes in an image over time or in regions that appear abnormal on the static image. This domain includes satellite imagery in [75], digit recognition in [21] and video surveillance in [62]. The remaining challenge is the large throughout of videos that requires intensive computation.

2.3.6 Textual anomaly detection

The techniques used in textual anomaly detection mainly detect novel topics or events stories in a collection of documents or news articles. The anomalies are corresponding to new interesting events or an unusual topic. Data in this task is typically sparse and high dimensional and sometimes contains temporal information of the collecting time. This topic is closely related to text mining discussed in [54], [69] and [70].

2.3.7 Other domains

Anomaly detection, with its mainstream applications in the domains discussed above, also finds its way in many other domains and they are briefly summarized in Table 2.1.

Table 2.1: Anomaly detection in other domains

speech recognition	[5]
robot behavior control	[20]
traffic monitoring	[65]
detecting faults in Web application	[72]
detecting anomalies in biological data	[52]
detecting associations among criminal activities	[50]
detecting anomalies in Customer Relationships	[38]
detecting anomalies in astronomical data	[25]
detecting ecosystem disturbances	[47]
detecting anomalies in census data	[51]

Chapter 3

From Anomaly Detection to Anomaly Location

After a brief review of the available methods in research and applications of anomaly detection, in this chapter we introduce our method of anomaly detection and location to handle a type of data set having the following characteristics:

- All the attributes of an instance are categorical.
- No training labels are available.
- The relationships among attributes are unknown.

In order to facilitate the discussion in this work, we define here the following mathematical notations.

In the categorical space, denote $\mathbf{X}$ as a categorical data set with N rows (instances) and K columns (attributes, variables). Denote $X_k(k = 1, \ldots, K)$ as k-th variable representing the k-th column in $\mathbf{X}$. Denote $\mathbf{x}_n = \{x_{n1}, \ldots, x_{nK}\}(n = 1, \ldots, N)$ as the n-th observation in $\mathbf{X}$. Given $\mathbf{x}_n$, we denote the joint probability of the instance $\mathbf{x}_n$ as $P(\mathbf{x}_n) = P(x_{n1}, x_{n2}, \ldots, x_{nK})$.

We denote factorization of $P(\mathbf{x}_n)$ as an operation $\mathcal{F}(P(\mathbf{x}_n)) = \prod_{k=1}^{K} f_k(x_{nk})$. The factorization consists of the multiplication of K factors $f_k(x_{nk})$ each of which denotes one factor as a function of x_{nk}.

In the continuous space, denote $\mathbf{r}_n = \{r_{n1}, \ldots, r_{nK}\}(n = 1, \ldots, N)$ as an observation mapped from categorical $\mathbf{x}_n$. After the mapping, $\mathbf{X}$ becomes $\mathbf{R}$ and X_k becomes $R_k(k = 1, \ldots, K)$.

The nature of the data set discussed above plays a strong constraint on the anomaly detection task. The lack of understanding of the application domain makes it difficult to conceptually define normal and anomalous instances. Classification-based methods are not applicable since the data set contains no labels for training.

Nearest neighbor and clustering-based methods rely strongly on the selection of the distance metric. For categorical data, there are two types of operations that may lead to defining a metric. One is to check whether two values are the same or not, which is used in computing Hamming distance. The other is to count the appearances of a certain value, namely, frequency-based counting. The latter operation opens the door to the real value space in which the Euclidean distance and many techniques are steadily available. Thus, one intuitive step is to rewrite each categorical instance by replacing the original categorical value with its frequency, which turns $\mathbf{x}_n$ into $\mathbf{r}_n$.

Our solution follows the idea of replacing categorical data with real values. Then, the nearest neighbor approach discussed in Chapter 2 is applied to the transformed data with the Euclidean distance as the distance metric. In order to decide which frequency to use to replace the categorical attribute, our solution in this chapter contains 2 directions each of which manages to perform $\mathcal{F}(P(\mathbf{x}_n)) = \prod_{i=1}^{K} f_i(x_{ni})$ based on dependent and independent assumptions between attributes. Furthermore, given an anomaly detection problem with a data set discussed above, our solution is able to locate the most likely anomalous attributes inside each given instance.

In order to clarify the theoretical discussion of this chapter, our theory will be applied to a toy example with $N = 8$ instances shown in Table 3.1 where $\mathbf{x}_n = (x_{n,1}, x_{n,2}, x_{n,3})$ representing n-th instance with 3 attributes $X_1 = x_{n,1}, X_2 = x_{n,2}$ and $X_3 = x_{n,3}$.

Table 3.1: The first toy example.

configuration	X_1	X_2	X_3	$P(\mathbf{x}_n)$
$\mathbf{x}_1$	1	2	2	0.125
$\mathbf{x}_2$	1	1	2	0.125
$\mathbf{x}_3$	1	2	3	0.125
$\mathbf{x}_4$	1	1	3	0.125
$\mathbf{x}_5$	2	2	3	0.125
$\mathbf{x}_6$	2	3	3	0.125
$\mathbf{x}_7$	3	2	2	0.125
$\mathbf{x}_8$	4	3	1	0.125

3.1 Anomaly detection with joint and marginal and conditional probability

In this section, we perform anomaly detection by focusing on individual instances as a whole and discuss the method to decide an anomalous score to be assigned to the examined instance. We also demonstrate the limitation of this method.

3.1.1 A fundamental assumption

According to Chapter 2, there is one critical assumption that should be considered when dealing with anomaly detection in categorical data – normal instances take the attributes that dominate the data set and anomalous instances take uncommon attributes with low frequencies of appearances. This assumption is fundamental to an unsupervised method and it is the backbone of all discussions of this work. If this assumption does not hold, the definition of an anomaly is violated and an anomaly detection algorithm will produce the unreliable result.

3.1.2 Conditional and marginal and joint probability estimation

Anomaly detection relies on the frequency of an instance. Considering an instance as a whole, given $\mathbf{X}$, we would like to evaluate the joint probability $P(\mathbf{x}_n)$ for each instance. Thus, each instance will get a score based on their joint probabilities by which the instances are ranked. A low joint probability suggests an anomalous instance while a high joint probability suggests a normal instance. Then the problem boils down to determine a threshold T in order to label all the instances larger than T as normal and the ones smaller than T as anomalous.

However, thresholding the ranking of joint probabilities is not always reliable. This measurement is only applicable when two types of instances have clear difference between their joint probabilities $P(\mathbf{x}_n)$. It is likely that they take very similar joint probabilities, which makes the decision of the threshold a nontrivial task. Because $P(\mathbf{x}_n)$ only considers an instance as a whole, it ignores the relationships between attributes.

The limitation of using only joint probability as a standard for anomaly detection can be easily demonstrated by using the toy example in Table 3.1. If we compute the joint probability of all 8 instances from $\mathbf{x}_1$ to $\mathbf{x}_8$ by counting how many times combination of the attributes $\{x_{n1}, x_{n2}, x_{n3}\}$ appears, we have their joint probabilities:

$$P(\mathbf{x}_1) = P(\mathbf{x}_2) = P(\mathbf{x}_3) = \cdots = P(\mathbf{x}_8) = \frac{1}{8}$$

They share the same joint probability that gives no information about whether an instance is anomaly or not. However, after a quick observation of the example, it can be easily seen that $x_{8,1} = 4$ and $x_{8,3} = 1$ is most likely to be the signs of $\mathbf{x}_8$ being an anomalous instance, which can be discovered by considering the marginal probability of $P(x_{8,1} = 4) = P_{min}(x_{n1}) = \frac{1}{8}$ and $P(x_{8,3} = 1) = P_{min}(x_{n3}) = \frac{1}{8}$.

Given an instance, if the joint probability $P(\mathbf{x}_n)$ can not be used to differentiate normal instances from anomalous ones, we then focus on a small subset of variables. For example, two instances having the same $P(\mathbf{x}_n)$ may have very different $P(x_{ni}|x_{nj})$ where $i, j \in 1, \ldots, K$ and $i \neq j$. Finding conditional probabilities is easy

given joint and marginal probabilities. Suppose we have joint probabilities $P(x_1, x_2)$, marginal probabilities of $P(x_{n1})$ and $P(x_{n2})$, finding the conditional probabilities of $P(x_{n1}|x_{n2})$ and $P(x_{n2}|x_{n1})$ is straightforward by using Bayes' theorem:

$$P(x_{n1}|x_{n2}) = \frac{P(x_{n1}, x_{n2})}{P(x_{n2})} \quad \text{and} \quad P(x_{n2}|x_{n1}) = \frac{P(x_{n1}, x_{n2})}{P(x_{n1})} \tag{3.1}$$

Essentially, estimation of conditional and joint probability leads to the rule-based anomaly detection techniques discussed in Chapter 2. A strong rule is a joint or conditional probability having a large value, in the language of rules, a strong support with high frequency while a weak rule is a joint or conditional probability having a small value. However, rule-based anomaly detection suffers from the problem that there exist huge amount of rules in a given data set and some domain knowledge is needed to filter out the most important rules and then to set up a threshold on frequency to differentiate normal and anomalous instances, which limits its use.

In order to overcome the limitation, we propose our method. Firstly, we manage to use only the joint probability $P(\mathbf{x}_n)$ as a score on $\mathbf{x}_n$ indicating the likelihood of a given instance being anomalous. This step is not always reliable for the reasons discussed before in this section. Secondly, we ignore the score and for each $\mathbf{x}_n$ is used to locate the anomalous attributes $X_k(k = 1, \ldots, K)$. This is achieved by the factorizing its joint probability as $\mathcal{F}(P(\mathbf{x}_n)) = \prod_{k=1}^{K} f_k(x_{nk})$. The factorization generates exactly the same number of factors as the number of attributes. The closer the factorization to the joint probability, the better quality it is and the more accurately the anomalous attributes can be located. Those joint probabilities from step one are used in step two to measure the quality of a given factorization $\mathcal{F}(P(\mathbf{x}_n))$ and also to compute any conditional probabilities by (3.1) when necessary. We will explain how the factors in $\mathcal{F}(P(\mathbf{x}_n)) = \prod_{k=1}^{K} f_k(x_{nk})$ can be replaced by conditional and marginal probabilities to locate the exact positions of anomalous attributes.

One thing worth mentioning is the computation of joint and marginal probabilities. Association rules mining with an efficient algorithm "Apriori" proposed in [2] represents a research field that has been widely studied to discover patterns inside categorical data. The association rule mining has a history of finding product combinations bought by customers in supermarket data. In [12], the author proposes one of the most efficient implementation by using a specialized data structure called "Trie" to speed up the computation. This algorithm is used in this work to compute the complete set of joint and marginal probabilities given a categorical data of an overwhelming size.

3.2 Anomaly location with factorizations and masks

The last section explains the use of conditional, marginal and joint probabilities found by Apriori algorithm and Bayes' Theorem to perform anomaly detection.

This section moves from anomaly detection to anomaly location by using those probabilities to derive an approach focusing on finding the location of anomalous attributes in a given instance. First of all, we examine two types of factorization based on assumptions regarding the independencies among variables. This is followed by introducing the concept of masks for each factorization to locate the anomalous attributes.

3.2.1 Independent factorization

The factorization can be achieved by using the assumption that $X_k(k = 1, \ldots, K)$ are statistically independent of each other. So the joint probability $P(\mathbf{x}_n)$ can be factorized as (3.2).

$$\mathcal{F}(P(\mathbf{x}_n)) = \prod_{k=1}^{K} f_k(x_{nk}) = \prod_{k=1}^{K} P(x_{nk}) \tag{3.2}$$

Given factorization (3.2) with the same number of factors as the number of attributes for an instance $\mathbf{x}_n$, we not only obtain another way to compute the joint probability $P(\mathbf{x}_n)$ by using only K marginals $P(x_{nk})$, but also transform the instance from the original categorical space to the continuous space by replacing its x_{nk} by r_{nk} with $r_{nk} = p(x_{nk})$ followed by the computations of marginals $p(x_{nk})$ and recomputation of joint probability $P(\mathbf{x}_n) = \prod_{k=1}^{K} P(x_{nk})$.

For the toy example, the marginal probabilities can be computed out very easily as:

- $P(X_1 = 1) = 0.5, P(X_1 = 2) = 0.25, P(X_1 = 3) = 0.125, P(X_1 = 4) = 0.125$
- $P(X_2 = 1) = 0.25, P(X_2 = 2) = 0.5, P(X_2 = 3) = 0.25$
- $P(X_3 = 1) = 0.125, P(X_3 = 2) = 0.375, P(X_3 = 3) = 0.5$

Following the assumption of independence between X_k, the joint probability of each instance can be factorized as:

$$\begin{aligned} P(\mathbf{x}_1) &= P(x_{1,1} = 1, x_{1,2} = 2, x_{1,3} = 2) = P(x_{1,1} = 1)P(x_{1,2} = 2)P(x_{1,3} = 2) \\ &= 0.5 \times 0.5 \times 0.375 = 0.094 \end{aligned}$$

$$\begin{aligned} P(\mathbf{x}_2) &= P(x_{2,1} = 1, x_{2,2} = 1, x_{2,3} = 2) = P(x_{2,1} = 1)P(x_{2,2} = 1)P(x_{2,3} = 2) \\ &= 0.5 \times 0.25 \times 0.375 = 0.047 \end{aligned}$$

$$\begin{aligned} P(\mathbf{x}_3) &= P(x_{3,1} = 1, x_{3,2} = 2, x_{3,3} = 3) = P(x_{3,1} = 1)P(x_{3,2} = 2)P(x_{3,3} = 3) \\ &= 0.5 \times 0.5 \times 0.5 = 0.125 \end{aligned}$$

Table 3.2: Independent factorization

configuration	R_1	R_2	R_3	$P(\mathbf{x}_n) = \prod_{k=1}^{K} P(x_{nk}) = \prod_{k=1}^{K} r_{nk}$	ranking
$\mathbf{r}_1$	0.5	0.5	0.375	0.094	6
$\mathbf{r}_2$	0.5	0.25	0.375	0.047	4
$\mathbf{r}_3$	0.5	0.5	0.5	0.125	7
$\mathbf{r}_4$	0.5	0.25	0.5	0.0625	5
$\mathbf{r}_5$	0.25	0.5	0.5	0.0625	5
$\mathbf{r}_6$	0.25	0.25	0.5	0.031	3
$\mathbf{r}_7$	0.125	0.5	0.375	0.023	2
$\mathbf{r}_8$	0.125	0.25	0.125	0.004	1

$$P(\mathbf{x}_4) = P(x_{4,1} = 1, x_{4,2} = 1, x_{4,3} = 3) = P(x_{4,1} = 1)P(x_{4,2} = 1)P(x_{4,3} = 3)$$
$$= 0.5 \times 0.25 \times 0.5 = 0.0625$$

$$P(\mathbf{x}_5) = P(x_{5,1} = 2, x_{5,2} = 2, x_{5,3} = 3) = P(x_{5,1} = 2)P(x_{5,2} = 2)P(x_{5,3} = 3)$$
$$= 0.25 \times 0.5 \times 0.5 = 0.0625$$

$$P(\mathbf{x}_6) = P(x_{6,1} = 2, x_{6,2} = 3, x_{6,3} = 3) = P(x_{6,1} = 2)P(x_{6,2} = 3)P(x_{6,3} = 3)$$
$$= 0.25 \times 0.25 \times 0.5 = 0.031$$

$$P(\mathbf{x}_7) = P(x_{7,1} = 3, x_{7,2} = 2, x_{7,3} = 2) = P(x_{7,1} = 3)P(x_{7,2} = 2)P(x_{7,3} = 2)$$
$$= 0.125 \times 0.5 \times 0.375 = 0.023$$

$$P(\mathbf{x}_8) = P(x_{8,1} = 4, x_{8,2} = 3, x_{8,3} = 1) = P(x_{8,1} = 4)P(x_{8,2} = 3)P(x_{8,3} = 1)$$
$$= 0.125 \times 0.25 \times 0.125 = 0.004$$

By the mapping from categorical values into real values, the toy example in Table 3.1 is transformed into the one in Table 3.2 in which column 6 shows the minimum-to-maximum ranking based on the joint probabilities in column 5 in which $\mathbf{r}_8$ representing originally $\mathbf{x}_8$ is most likely to be an anomaly while $\mathbf{r}_3$ representing $\mathbf{x}_3$ is most likely to be a normal instance. By the discussion of anomaly detection in Section 3.1, the higher the ranking, the more likely the associated instance to be an anomaly.

3.2.2 Fully dependent factorization

Even though the independent assumption introduces a coarse approximation of the joint probability, it still is useful in some cases in which factors representing at-

tributes with lower frequencies need to be taken into account. Next, we place an even stronger 2-part assumption not only on the relationships between attributes but also on the unnatural way of factorizing the joint probability:

1. complete dependence between attributes of $\mathbf{x}_n$.
2. the joint probability $P(\mathbf{x}_n)$ can be factorized by conditional probability of each attribute given all the other attributes as (3.3).

$$\mathcal{F}(P(\mathbf{x}_n)) = \prod_{k=1}^{K} f_k(x_{nk}) = \prod_{k=1}^{K} P(x_{nk}|x_{n1}, x_{n2}, \ldots, x_{n[k-1]}, x_{n[k+1]}, \ldots, x_{nK}) \tag{3.3}$$

Consider the toy example again, the joint probability becomes $P(x_{n1}, x_{n2}, x_{n3}) = P(x_{n1}|x_{n2}, x_{n3})P(x_{n2}|x_{n1}, x_{n3})P(x_{n3}|x_{n1}, x_{n2})$.

$$\begin{aligned} P(\mathbf{x}_1) &= P(x_{1,1} = 1, x_{1,2} = 2, x_{1,3} = 2) \\ &= P(x_{1,1} = 1|x_{1,2} = 2, x_{1,3} = 2) \\ &\times P(x_{1,2} = 2|x_{1,1} = 1, x_{1,3} = 2) \\ &\times P(x_{1,3} = 2|x_{1,1} = 1, x_{1,2} = 2) \\ &= 1 \times 0.5 \times 0.5 = 0.25 \end{aligned}$$

Similarly,

$$P(\mathbf{x}_2) = 1 \times 0.5 \times 0.5 = 0.25 \quad P(\mathbf{x}_3) = 0.5 \times 0.5 \times 0.5 = 0.25$$

$$P(\mathbf{x}_4) = 1 \times 0.5 \times 0.5 = 0.25 \quad P(\mathbf{x}_5) = 0.5 \times 0.5 \times 1 = 0.25$$

$$P(\mathbf{x}_6) = 1 \times 0.5 \times 1 = 0.5 \quad P(\mathbf{x}_7) = 0.5 \times 1 \times 1 = 0.5$$

$$P(\mathbf{x}_8) = 1 \times 1 \times 1 = 1$$

By the replacement, the toy example in Table 3.1 is transformed into the continuous-valued data set in Table 3.3. The column 6 shows the minimum-to-maximum ranking based on the joint probabilities in column 5. In this case, $\mathbf{r}_3$ is most likely to be an anomalous instance while $\mathbf{r}_8$ becomes perfectly normal. So far, one can see that different assumptions dramatically influence the result of anomaly detection.

3.2.3 Locating anomalous attributes by masks

We define a *mask* for an instance $\mathbf{x}_n$ as a binary vector $\mathbf{m}_n = \{m_{n,1}, \ldots, m_{n,K}\}$ having the same dimension K as an instance $\mathbf{x}_n$. Among $\mathbf{x}_n$, each individual attributes $x_{n,k}$ receives a mask $m_{n,k}$ such that $x_{n,k}$ is covered by $m_{n,k}$. As a binary vector, $m_{n,k}$ may recieve either 0 or 1. $m_{n,k} = 0$ means the according attribute $x_{n,k}$ it covers in the instance is anomalous. When $m_{n,k} = 1$, $x_{n,k}$ is normal. It is likely that a given instance is associated with a mask with $m_{n,k} = 1$ for all $k = \{1, \ldots, K\}$, which

Table 3.3: Fully dependent factorization

configuration	R_1	R_2	R_3	$P(\mathbf{x}_n)$	ranking
$\mathbf{r}_1$	1	0.5	0.5	0.25	2
$\mathbf{r}_2$	1	0.5	0.5	0.25	2
$\mathbf{r}_3$	0.5	0.5	0.5	0.125	1
$\mathbf{r}_4$	1	0.5	0.5	0.25	2
$\mathbf{r}_5$	0.5	0.5	1	0.25	2
$\mathbf{r}_6$	1	0.5	1	0.5	3
$\mathbf{r}_7$	0.5	1	1	0.5	3
$\mathbf{r}_8$	1	1	1	1	4

means the instance is perfectly normal. Otherwise, all the positions in the instance taking the mask 0 indicate the potentially anomalous attributes. The generation of a mask follows three steps.

Step 1: constructing a hypercube in the continuous Euclidean space

The process is summarized by $\mathbf{x}_n \rightarrow \mathbf{r}_n \rightarrow \mathbf{p}_n$. Given $\mathbf{x}_n$, it is mapped into $\mathbf{r}_n$ by using two types of factorizations each of which is based on one assumption discussed before. So for each $\mathbf{r}_n$, $P(\mathbf{r}_n) = \prod_{k=1}^{K} f_k(x_{n,k}) = \prod_{k=1}^{K}(r_{n,k})$. In order to construct a Euclidean space containing all instances, we use (3.4) to transform the product into summation so that $\mathbf{r}_n$ is mapped into a point $\mathbf{p}_n$ in the log-likelihood space of $\mathbf{x}_n$. $\mathbf{p}_n$ has the coordinates of $\{log_e(r_{n,1}), \ldots, log_e(r_{n,K})\}$.

$$log_e[P(\mathbf{x}_n)] = log_e[\prod_{k=1}^{K} f_k(x_{n,k})] = \sum_{k=1}^{K} log_e[f_k(x_{n,k})] = \sum_{k=1}^{K} log_e(r_{n,k}) \tag{3.4}$$

After the coordinates of point $\mathbf{p}_n$ are decided, we construct a hypercube in K dimensional Euclidean space so that there are no points falling outside the hypercube. To fully specify the hypercube, one needs only two such points that point $\mathbf{p}_{max}$ having the maximal $log_e(r_{n,k})$ across all n in each direction k and $\mathbf{p}_{min}$ having the minimal $log_e(r_{n,k})$ across all n in each direction k. Given $\mathbf{p}_{max}$ and $\mathbf{p}_{min}$, the coordinates of all 2^K vertexes of the hypercube can be derived from those two points and the boundary hypercube can be fully constructed. Note that we use a relatively loose definition of "hypercube" since its edges in our case are not necessarily equal.

Step 2: assign masks to vertexes of the hypercube

After the hypercube is fully constructed, we assign a unique mask for each of the vertexes of the hypercube. The point $\mathbf{p}_{max}$, the vertex closest to the origin in the space, is assigned a perfect mask $\mathbf{m}_n = \{m_{n,1} = 1, \ldots, m_{n,K} = 1\}$. On the

contrary, the point $\mathbf{p}_{min}$, the farthest to the origin, is assigned the worst mask $\mathbf{m}_n = \{m_{n,1} = 0, \ldots, m_{n,K} = 0\}$. Following the same pattern of generating the coordinates of the vertexes, each of the other vertexes receives a unique mask.

Step 3: assign masks to instances

Given a point $\mathbf{p}_n$ representing an instance $\mathbf{x}_n$, $\mathbf{p}_n$ finds its nearest vertex by comparing the distances between each of the vertexes of the hypercube and itself. Finally, the point gets the mask exactly the same as its nearest vertex. There are situations in which a point gets more than one nearest neighbors among vertexes, thus leading to multiple masks for one instance. Intuitively, this method can be considered as a clustering technique in Chapter 2 using Euclidean distance as the metric. The speciality is that the clustering centroids are the vertexes of the hypercube.

3.2.4 Masks for independent factorization

Table 3.4: Masks of vertexes for both independent and fully dependent assumptions

vertex	mask	$m_{n,1}$	$m_{n,2}$	$m_{n,3}$
$\mathbf{v}_1$	$\mathbf{m}_1$	0	0	0
$\mathbf{v}_2$	$\mathbf{m}_2$	0	0	1
$\mathbf{v}_3$	$\mathbf{m}_3$	0	1	0
$\mathbf{v}_4$	$\mathbf{m}_4$	0	1	1
$\mathbf{v}_5$	$\mathbf{m}_5$	1	0	0
$\mathbf{v}_6$	$\mathbf{m}_6$	1	0	1
$\mathbf{v}_7$	$\mathbf{m}_7$	1	1	0
$\mathbf{v}_8$	$\mathbf{m}_8$	1	1	1

Table 3.5: 3-D cube under the assumption of independence

row	$log_e(R_1)$	$log_e(R_2)$	$log_e(R_3)$	vertex	$v_{n,1}$	$v_{n,2}$	$v_{n,3}$
$\mathbf{p}_1$	-0.6931	-0.6931	-0.9808	$\mathbf{v}_1$	-2.0794	-1.3862	-2.0794
$\mathbf{p}_2$	-0.6931	-1.3863	-0.9808	$\mathbf{v}_2$	-2.0794	-1.3863	-0.6931
$\mathbf{p}_3$	-0.6931	-0.6931	-0.6931	$\mathbf{v}_3$	-2.0794	-0.6931	-2.0794
$\mathbf{p}_4$	-0.6931	-1.3863	-0.6931	$\mathbf{v}_4$	-2.0794	-0.6931	-0.6931
$\mathbf{p}_5$	-1.3863	-0.6931	-0.6931	$\mathbf{v}_5$	-0.6931	-1.3863	-2.0794
$\mathbf{p}_6$	-1.3863	-1.3863	-0.6931	$\mathbf{v}_6$	-0.6931	-1.3863	-0.6931
$\mathbf{p}_7$	-2.0794	-0.6931	-0.9808	$\mathbf{v}_7$	-0.6931	-0.6931	-2.0794
$\mathbf{p}_8$	-2.0794	-1.3863	-2.0794	$\mathbf{v}_8$	-0.6931	-0.6931	-0.6931

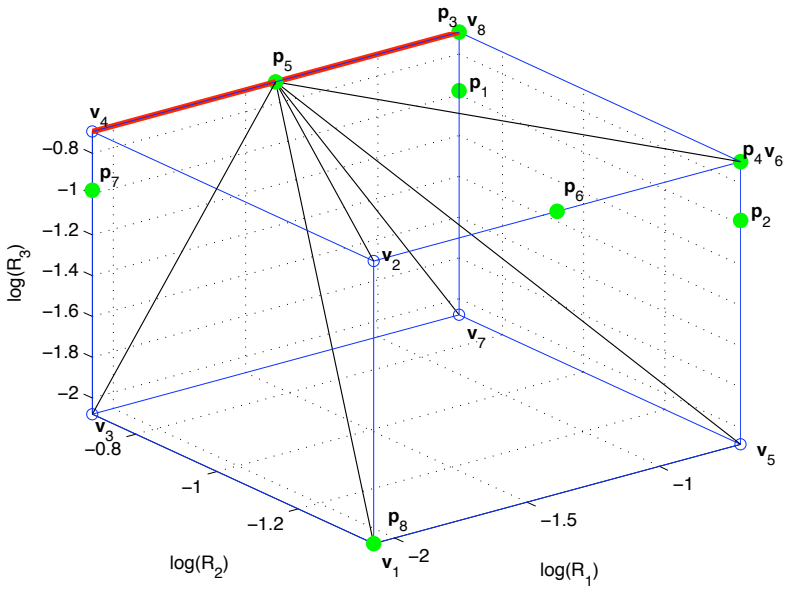


Figure 3.1: Masks for independent case. Black lines show the distances between $\mathbf{p}_5$ and all vertexes of the cube. Red line shows the shortest distance among them.

Table 3.6: Masks of instances in the independent case.

instance	mask	instance	mask
$\mathbf{x}_1 \rightarrow \mathbf{r}_1 \rightarrow \mathbf{p}_1$	$\mathbf{m}_8$	$\mathbf{x}_5 \rightarrow \mathbf{r}_5 \rightarrow \mathbf{p}_5$	$\mathbf{m}_4, \mathbf{m}_8$
$\mathbf{x}_2 \rightarrow \mathbf{r}_2 \rightarrow \mathbf{p}_2$	$\mathbf{m}_6$	$\mathbf{x}_6 \rightarrow \mathbf{r}_6 \rightarrow \mathbf{p}_6$	$\mathbf{m}_2, \mathbf{m}_6$
$\mathbf{x}_3 \rightarrow \mathbf{r}_3 \rightarrow \mathbf{p}_3$	$\mathbf{m}_8$	$\mathbf{x}_7 \rightarrow \mathbf{r}_7 \rightarrow \mathbf{p}_7$	$\mathbf{m}_4$
$\mathbf{x}_4 \rightarrow \mathbf{r}_4 \rightarrow \mathbf{p}_4$	$\mathbf{m}_6$	$\mathbf{x}_8 \rightarrow \mathbf{r}_8 \rightarrow \mathbf{p}_8$	$\mathbf{m}_1$

Following the discussion of the last section, we extend (3.4) to (3.5) by using the assumption of independence between attributes of $\mathbf{x}_n$ discussed in Section 3.2.1.

$$log_e[P(\mathbf{x}_n)] = log_e[\prod_{k=1}^{K} f_k(x_{n,k})] = \sum_{k=1}^{K} log_e[f_k(x_{n,k})] = \sum_{k=1}^{K} log_e(r_{n,k}) = \sum_{k=1}^{K} log_e(P(x_{n,k})) \tag{3.5}$$

Take the toy example again. By using the results in Table 3.2 and instructions of building the hypercube in Section 3.2.3, we have Table 3.5 showing the coordinates of all 8 vertexes of the 3-D cube and their masks in Table 3.4. The 3-D cube is shown in Figure 3.1 indicating the process of associating each $\mathbf{p}_n$ to its nearest vertexes.

To find the positions of the actual attributes that may cause the anomaly, we

use the mask of the nearest vertexes of $\mathbf{p}_n$ as the mask of $\mathbf{r}_n$ and thus $\mathbf{x}_n$ in which 0 shows the possible position that contains relatively low conditional or marginal probability and thus an anomalous location. The mask for all $\mathbf{p}_n$ is shown in Table 3.6.

3.2.5 Masks for fully dependent factorization

We extend (3.4) to (3.6) by using the assumption made in Section 3.2.2.

$$log_e[P(\mathbf{x}_n)] = log_e[\prod_{k=1}^{K} f_k(x_{n,k})] = \sum_{k=1}^{K} log_e[f_k(x_{n,k})] = \sum_{k=1}^{K} log_e(r_{n,k}) = \sum_{k=1}^{K} log_e(P(x_{n,k}))$$
$$= \sum_{k=1}^{K} log_e[P(x_{nk}|x_{n1}, x_{n2}, \ldots, x_{n[k-1]}, x_{n[k+1]}, \ldots, x_{nK})] \quad (3.6)$$

For the toy example, by using the results in Table 3.3 and instructions of building the hypercube in Section 3.2.3, we have Table 3.7 showing the coordinates of all 8 vertexes of the 3-D cube. The 3-D cube is shown in Figure 3.2.

Table 3.7: 3-D cube under the assumption of full dependence

row	$log_e(R_1)$	$log_e(R_2)$	$log_e(R_3)$	vertex	$v_{n,1}$	$v_{n,2}$	$v_{n,3}$
$\mathbf{p}_1$	0	-0.6931	-0.6931	$\mathbf{v}_1$	-0.6931	-0.6931	-0.6931
$\mathbf{p}_2$	0	-0.6931	-0.6931	$\mathbf{v}_2$	-0.6931	-0.6931	0
$\mathbf{p}_3$	-0.6931	-0.6931	-0.6931	$\mathbf{v}_3$	-0.6931	0	-0.6931
$\mathbf{p}_4$	0	-0.6931	-0.6931	$\mathbf{v}_4$	-0.6931	0	0
$\mathbf{p}_5$	-0.6931	-0.6931	0	$\mathbf{v}_5$	0	-0.6931	-0.6931
$\mathbf{p}_6$	0	-0.6931	0	$\mathbf{v}_6$	0	-0.6931	0
$\mathbf{p}_7$	-0.6931	0	0	$\mathbf{v}_7$	0	0	-0.6931
$\mathbf{p}_8$	0	0	0	$\mathbf{v}_8$	0	0	0

Table 3.8: Masks of instances in the fully dependent case.

instance	mask	instance	mask
$\mathbf{x}_1 \rightarrow \mathbf{r}_1 \rightarrow \mathbf{p}_1$	$\mathbf{m}_5$	$\mathbf{x}_5 \rightarrow \mathbf{r}_5 \rightarrow \mathbf{p}_5$	$\mathbf{m}_2$
$\mathbf{x}_2 \rightarrow \mathbf{r}_2 \rightarrow \mathbf{p}_2$	$\mathbf{m}_5$	$\mathbf{x}_6 \rightarrow \mathbf{r}_6 \rightarrow \mathbf{p}_6$	$\mathbf{m}_6$
$\mathbf{x}_3 \rightarrow \mathbf{r}_3 \rightarrow \mathbf{p}_3$	$\mathbf{m}_1$	$\mathbf{x}_7 \rightarrow \mathbf{r}_7 \rightarrow \mathbf{p}_7$	$\mathbf{m}_4$
$\mathbf{x}_4 \rightarrow \mathbf{r}_4 \rightarrow \mathbf{p}_4$	$\mathbf{m}_5$	$\mathbf{x}_8 \rightarrow \mathbf{r}_8 \rightarrow \mathbf{p}_8$	$\mathbf{m}_8$

As the independent case, we map the mask of the nearest vertex of $\mathbf{p}_n$ as its navigation code in which 0 shows the possible position that contains relatively low

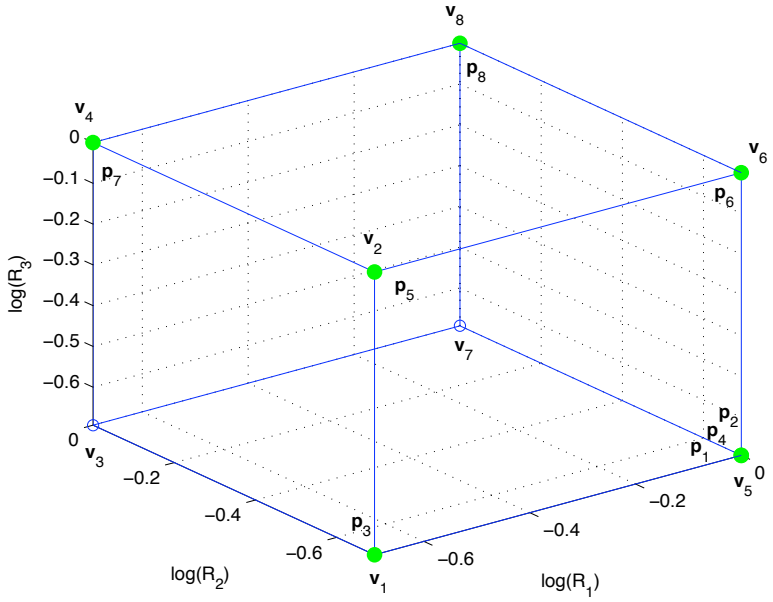


Figure 3.2: All instances are distributed on the vertexes of the cube in fully dependent case.

conditional probability and thus suspicious anomalous location. According to Figure 3.2, all instances overlap with one of the vertexes. The navigation code for all $\mathbf{p}_n$ is shown in Table 3.8.

A simple investigation of the masks generated by both dependent and independent cases shows the advantages and disadvantages of using masks to locate the anomalous attributes. Based on the independent assumption, only $\mathbf{x}_5$ and $\mathbf{x}_6$ individually get two masks while others get only one mask. In the fully dependent case, everyone gets one mask. The multiple-mask phenomenon results from the small size of the toy example. In practice, the more instances and categorical values of variables we have in the data set, the more diversities of frequencies we will get for different factors, the less likely an instance falls exactly in the middle along one direction in the hypercube, thus more and more instances recieve only one mask. Another observation of both results is that most of the points lie on the boundary of the cube, especially in the second case in which all points fall on the vertexes. This is not always the case since for most of the real world problems, there are more variations – attributes getting hundreds or even thousands categorical values with different frequencies, which will most likely bring most the points inside the hypercube.

We also suggest that even though both assumptions are statistically ambiguous and arbitrary in the way that the joint probability computed by the factorization is

far away from real, they both turn out to be useful. By taking a look at instance $\mathbf{x}_8 = \{x_{8,1} = 4, x_{8,2} = 3, x_{8,3} = 1\}$ in both cases. It respectively gets the worst mask $\mathbf{m}_1$ and perfect mask $\mathbf{m}_8$ in two cases. In fact, considering all 8 instances, $x_{n,1} = 4$ and $x_{n,3} = 1$ only appears in $\mathbf{x}_8$. It is fair to say that $\mathbf{x}_8$ is extremely anomalous since both $x_{8,1}$ and $x_{8,3}$ take the lowest frequencies, which supports the independent assumption. On the other hand, it is fair to trust the second case to say $\mathbf{x}_8$ is perfect normal since two variables getting lowest frequency may suggest a new instance following new rules between those two variables that has never appeared before. In conclusion, those two assumptions pay attention to different aspects of the given instances. The most common way to utilize the result is to test them with domain knowledge.

Chapter 4

Factorization by Bayesian Network

The assumptions of independence and complete dependence used in the last chapter suggest ways to factorize the joint probability. Both assumptions, however, give poor approximations of the joint probability even though they may be useful to locate the anomalous attributes. This chapter focuses on a factorization without any assumption of independence that directly leads to a much more accurate approximation to the joint probability.

In order to accurately factorize the joint probability $P(\mathbf{x}_n) = P(x_{n1}, x_{n2}, \ldots, x_{nK})$, given no knowledge of dependencies and independencies between variables of $X_k (k = 1, \ldots, K)$, the only way to achieve this is to use the chain rule of probability, for example:

$$P(x_{n1}, \ldots, x_{nK}) = P(x_{n1})P(x_{n2}|x_{n1})P(x_{n3}|x_{n1}, x_{n2}) \ldots P(x_{nK}|x_{n1}, \ldots, x_{K-1})$$

The merit, compared with the independent and fully dependent assumptions made before, is that it is an exact factorization having exactly the same joint probability as the real one by counting. The problem is that one may end up with huge amount of different factorizations by using the chain rule. For example, another factorization changing the first two factors can be:

$$P(x_{n1}, \ldots, x_{nK}) = P(x_{n2})P(x_{n1}|x_{n2})P(x_{n3}|x_{n1}, x_{n2}) \ldots P(x_{nK}|x_{n1}, \ldots, x_{K-1}) \tag{4.1}$$

One may notice that the number of different factorizations by the chain rule depends on the arrangement of variables, meaning that for each permutation of the variable set $\{X_1, \ldots, X_K\}$, there is a unique factorization of the joint, which gives $K!$ factorizations in total. Even though these are exact factorizations, the enormous number of factors poses another serious question: Which factorization should be used to generate the mask, as we did in Chapter 3, to locate the anomalous positions inside each given instance when no *a priori* knowledge is in practice available?

The answer becomes slightly simpler if the true dependencies and independencies between groups of $\{x_{n1}, \ldots, x_{nK}\}$ are taken into consideration. For example, if x_{n3} is only dependent on x_{n1}, in (4.1) the factor $P(x_3|x_1, x_2)$ can be simplified into $P(x_3|x_1)$. Out of those factorizations, we are looking for the factorization that fully specifies the conditional independencies between variables. This target can be achieved by several approaches. The most intuitive way is to check each of the factors and try to simplify them by eliminating variables while keeping the conditional probability unchanged, as is shown by the previous example. However, this approach suffers from the factorial number of different factorizations and is unable to differentiate the factorizations since all of them, whether simplified or not, are exact factorization of the joint probability. In order to overcome this limitation, the better approach is to construct a Bayesian network that best expresses the conditional independencies in the given data set. In particular, this chapter focuses on the problem of structural learning of a Bayesian network given a categorical data set.

4.1 Bayesian network in general

Bayesian networks exploit the conditional independence within a joint distribution and the use of directed acyclic graph (DAG) allows a compact and natural representation of those independencies. Given a data set $\mathbf{X}$ with N rows and K columns, a Bayesian network $\mathbf{B}$ for a set of variables $X_k (k = 1, \ldots, K)$ consists of a network structure $\mathbf{g}$ that encodes a set of conditional independence assertions about variables X_k and a set of local probability distributions θ_g associated with each variable.

X_k denotes both the variable and its corresponding node in DAG. The network structure g is a DAG in which each node X_k corresponds to one and only one variable X_k in $\mathbf{X}$ and Pa_k denotes the parents of node X_k in g. θ_g is a set of conditional probability table (e.g. Multinomial) for discrete variables, or continuous probability distributions (e.g. Multivariate Gaussian) for continuous variables, or sometimes even a mixture of both. The Bayesian network can be interpreted in two ways:

- $\mathbf{B}$ can be semantically translated into a set of local independencies such that each node X_k is conditionally independent of its nondescendants, given its parents.
- $\mathbf{B}$ indicates a factorization of joint probability as:

$$P(x_1, \ldots, x_k) = \prod_{k=1}^{K} P(x_k|Pa_k)$$

Normally, given a Bayesian network defining a structure $\mathbf{g}$ and local probability distributions θ_g on g and a probability distribution $\mathcal{D}$, the connection between these two different concepts replies on the semantics of the Bayesian network. It is the conditional independencies between variables that connects a Bayesian network and

a probability distribution together. The two interpretations of **B** are insured by the following two theorems in [46]:

- Let **B** be a Bayesian network over a set of random variables X_k and $\mathcal{D}$ be a joint distribution over the same space. If all the conditional independencies specified in **B** hold in $\mathcal{D}$, then $\mathcal{D}$ factorizes according to **B**.
- Let **B** be a Bayesian network over a set of random variables X_k and $\mathcal{D}$ be a joint distribution over the same space. If $\mathcal{D}$ factorizes according to **B**, then all the conditional independencies specified in **B** hold in $\mathcal{D}$.

These two complementing theorems shows the fundamental connection between conditional independencies encoded by Bayesian network and the factorization of a probability distribution into local probability models: Conditional independencies imply factorization and factorization implies conditional independencies.

There are three important things one may notice from the two theorems above. Firstly, a Bayesian network shows nothing but conditional independencies in a given distribution. The factorization is nothing but a compressed representation of the given distribution based on the Bayesian network. Secondly, the factors only indicate the local dependencies in the graph. For example, if there exists a factor such as $P(x_k|Pa_k)$, it does not mean x_k is only dependent on its parents among all the variables. Thirdly, it is necessary that **B** does not mislead us regarding independencies in $\mathcal{D}$: any independence that **B** asserts must also hold in $\mathcal{D}$. Conversely, $\mathcal{D}$ may have additional independencies that are not reflected in **B**.

Finally, one has to notice that structurally different Bayesian networks may be equivalent in the way they specify the same set of conditional independencies. It is therefore natural to believe those equivalent networks can all be used to factorize a distribution. There is no intrinsic property of $\mathcal{D}$ that would allow us to associate it with one graph rather than another equivalent one. This phenomenon appears in structural learning of a Bayesian network in which there are multiple optimal graphs associated with the given data set.

4.2 Structural learning in general

Learning a Bayesian network generally consists of two different parts. One part is to learn the parameters θ_g of the local probabilities given the structure of a network and a data set. Another is to learn the structure g of the network given only a data set. This section gives an introduction to the structural learning.

4.2.1 Constraint-based structural learning

The earliest approach of constraint-based structural learning can be found in [59] followed by [57] and [18]. This type of approach considers a Bayesian network as a representation of independencies and tests for conditional dependence and independence from data and find a Bayesian network to explain those relationships. Constraint-based methods are intuitive: They decouple the problem of finding the structure from the notion of independency search and follow closely the definition of Bayesian network. Unfortunately, according to [46], this type of approach can be sensitive to failures in individual independence tests and proved not applicable in most of the situations.

4.2.2 Score-based structural learning

The second and most popular approach is score-based structure learning. This type of approach views a Bayesian network as a statistical model and addresses the discovery of a Bayesian network as a model selection process. The approach follows three steps:

- Defining a hypothesis space of potential models which are all the possible network structures to be considered.
- Defining a score function which measures how well a certain network structure fits the data.
- Searching through the whole model space to find the network structure with the highest score.

It is straightforward to define a hypothesis space of potential models. Usually, the dimensionality K of $\{X_1, X_2, \ldots, X_K\}$ decides the number of nodes in the potential networks. The appearance of an edge and its direction between two nodes defines the variations in the model space. The number of different models grows superexponentially with the number of nodes — $2^{\mathcal{O}(K^2)}$. Therefore, the exhaustive search is only feasible when $K \leq 5$ due to the computational workload. Some searching heuristics are needed to guide the search to explore the whole model space and find the global optimal graph in a reasonable time.

The remaining concern of score-based structure learning is the selection of the score function. Intuitively, there are two options. One natural choice is the likelihood function that measures the probability of the data given a model. Assume we want to maximize the likelihood of the model. We denote a model $\mathbf{B}$ as a pair $[g, \theta_g]$. The task is to find both a graph g and parameters θ_g that maximize the likelihood function L:

$$max_{g,\theta_g} L([g, \theta_g] : \mathbf{X}) = max_g [max_{\theta_g} L([g, \theta_g] : \mathbf{X})]$$

When searching through the graph space, given any encountered graph g, we use the give data set to estimate its parameter $\hat{\theta}_g$. Then the score function becomes:

$$score_L(g : \mathbf{X}) = log[L([g, \hat{\theta}_g] : \mathbf{X})]$$

Another option of the score function is the Bayesian score. Since we have uncertainty on the network structure and its parameters, we define a structure prior $P(g)$ as a prior probability on different graph structures and a parameter prior $P(\theta_g|g)$ as the prior of different choice of parameters for a given graph structure. By Bayes' rule, we have:

$$P(g|\mathbf{X}) = \frac{P(\mathbf{X}|g)P(g)}{P(\mathbf{X})} \tag{4.2}$$

Ignoring the normalizing denominator, we have the Bayesian score as:

$$score_B(g : \mathbf{X}) = log[P(\mathbf{X}|g)] + log[P(g)] \tag{4.3}$$

The first term on the right of (4.3) takes into consideration the uncertainty of all possible parameters given a graph g and can be marginalized over all possible θ_g as:

$$L(g : \mathbf{X}) = P(\mathbf{X}|g) = \int_{\theta_g} P(\mathbf{X}|\theta_g, g)P(\theta_g|g)d\theta_g \tag{4.4}$$

where $P(\mathbf{X}|\theta_g, g)$ is the likelihood of the data given the network and $P(\mathbf{X}|g)$ is called the marginal likelihood of data given structure. Note that we use $L(g : \mathbf{X})$ in Bayesian approach as the marginal likelihood function compared with $L([g, \theta_g] : \mathbf{X})$ in maximum likelihood estimation as the likelihood function.

The main reason of choosing Bayesian score over the likelihood score is to avoid overfitting. Maximum likelihood is overly "optimistic" in its estimation of the score: As a point estimation, the parameter estimated from the training set may not always provide the best fit of the unseen data. On the contrary, Bayesian approach integrates the likelihood $P(\mathbf{X}|\theta_g, g)$ over different choices of parameters θ_g, thus, the expected likelihood is measured, which is more conservative in the estimation of the "goodness" of the model.

4.2.3 Understanding score functions

Essentially, using score functions to perform the model selection converts the Bayesian network structural learning to an optimization problem in which we search in the entire model space to select the model with the maximal score. The result may seem unrelated to the Bayesian network defined on the conditional dependencies since there is no guarantee that the optimal network achieved by maximizing the score is exactly the optimal DAG covering all the conditional dependencies satisfied by the actual probability distribution $\mathcal{D}$. We explain the connection in this part.

According to [46], the likelihood score in the structural learning can be decom-

posed as follows:

$$score_L(g : \mathbf{X}) = N \sum_{k=1}^{K} I_{\hat{\mathcal{D}}}(X_k; Pa_k^g) - N \sum_{k=1}^{K} H_{\hat{\mathcal{D}}}(X_k) \tag{4.5}$$

where N is still the number of instances and $\hat{\mathcal{D}}$ is the empirical distribution observed in the data set $\mathbf{X}$ and $I(X_k; Pa_k^g)$ is the mutual information measuring the amount of shared knowledge between X_k and its parents in graph g and $H(X_i)$ is the entropy measuring the uncertainty of X_k.

Note that the second term on the right-hand side of (4.5) does not depend on the network structure and thus can be ignored when we compare two structures with the same data set. The first term I can be considered as the degree of dependence between a variable and its parents. Thus, the likelihood score measures the strength of the dependencies between variables and their parents. The likelihood score prefers a network where the parents of each variable are most informative on their children

To consider the likelihood score from another angle, we have from [46] the following corollary:

$$\frac{1}{N} score_L(g : \mathbf{X}) = H_{\hat{\mathcal{D}}}(X_1, \ldots, X_K) - \sum_{k=1}^{K} I_{\hat{\mathcal{D}}}(X_k; \{X_1, \ldots, X_{k-1}\} - Pa_k | Pa_k) \tag{4.6}$$

Similarly, the first term on the right-hand side of (4.6) does not depend on the graph. The second term defines the conditional mutual information of X_i and its preceding variables given its parents. Maximizing the likelihood score is equivalent to minimizing the second term such that the shared information between X_i and its precedents given its parents is as small as possible.This second term receives the minimal value of zero when the conditional independence holds in the given data set.

By using (4.5) and (4.6), we understand that the likelihood score-based optimization is intrinsically connected with the process of looking for an optimal DAG. Now consider the term of marginal likelihood in the Bayesian score and we have the following conclusion from [46]:

$$logP(\mathbf{X}|g) = log(L([g, \hat{\theta}_g])) - \frac{logN}{2} Dim[g] + \mathcal{O}(1) \tag{4.7}$$

where the first term is $score_L(g : \mathbf{X})$ — the likelihood score, and $Dim[g]$ is the number of independent parameters in g, or model dimensionality. If we omit the third term on the right-hand side of (4.7), we have the Bayesian Information Criterion (BIC) score . Substituting (4.5) into (4.7), we have the following BIC score which

is an approximation of $score_B$.

$$score_{BIC}(g:\mathbf{X}) = N\sum_{k=1}^{K} I_{\hat{\mathcal{D}}}(X_k; Pa_k) - N\sum_{k=1}^{K} H_{\hat{\mathcal{D}}}(X_k) - \frac{logN}{2}Dim[g] \tag{4.8}$$

As an approximated version of the Bayesian score, the BIC score shown in (4.8) indicates the behavior of the structural learning with $score_B$. First of all, on the right-hand side of (4.8), with fixed N, the second term is the same for all g. The third term penalizes the likelihood score by considering the model complexity and number of instances. That is, the stronger the dependence of X_k and Pa_k, the higher the BIC score and Bayesian score; the more complex the network, the lower the BIC score and Bayesian score. Another important thing to be noticed is that as the number of instances N increases, the first two terms increase by N and the third term increases by $logN$, thus, according to [46], the larger N is, the more emphasis will be given to the fit to data and the more the score exhibits the preference of more complex structures. However, for a small data set, the penalty term outweighs the likelihood term.

4.3 Markov chain Monte Carlo in general

Markov chain Monte Carlo (MCMC) is a widely used sampling method to approximate the posterior probability distribution of $P(g|\mathbf{X})$ and the complex integration of marginal likelihood $P(\mathbf{X}|g)$ in the score-based structural learning of Bayesian networks. This section reviews the basic ideas of MCMC. More detailed discussion can be found in [31].

4.3.1 Markov chain

Considering one dimensional random variable z, and z_t denotes the value of the random variable at time t. The random variable follows a Markov process if the transition probabilities between different values in the state space depend only on the random variable's current state, that is:

$$P(z_{t+1}|z_t, z_{t-1}, \ldots, z_1) = P(z_{t+1}|z_t)$$

Thus, to predict the value of the random variable, the only information needed is the transition probability $P(z_{t+1}|z_t)$ denoted by $\mathbf{T}(i,j) = P(z_{t+1} = j|z_t = i)$.

Let $\pi_j(t) = P(z_t = j)$ denote the chain in state j at time t and let $\boldsymbol{\pi}(t)$ denote the row vector of the state space probability at step t and $\boldsymbol{\pi}(0)$ denote a starting vector corresponding to the probability of the chain starting at all possible states. The starting probability $\boldsymbol{\pi}(0)$ and the transition matrix $\mathbf{T}$ fully specify a Markov chain. According to Chapman-Kolomogrov equation, the Markov process can be described as:

$$\boldsymbol{\pi}(t+1) = \boldsymbol{\pi}(t)\mathbf{T}$$

The most interesting feature of a Markov chain is its stationary distribution denoted by $\boldsymbol{\pi}^*$, where the vector of probabilities of being in any particular given state is independent of the initial condition. Thus, the stationary distribution satisfies

$$\boldsymbol{\pi}^* = \boldsymbol{\pi}^* \mathbf{T}$$

In order to make sure the Markov chain converges to a stationary distribution, two conditions must be satisfied.

1. The chain is irreducibile: one can always go from any state to any other state (it may take several steps).
2. The chain is aperiodic: the chain is not forced into some cycle of fixed length between certain states.

4.3.2 Monte Carlo

The original Monte Carlo approach was a method developed by physicists to use random number generation to compute integrals. Suppose we would like to compute a complex integral

$$\int_a^b h(z)dz$$

If $h(z)$ can be decomposed into the production of a function $f(z)$ and a probability density function $P(z)$ defined over the interval (a, b), then note that

$$\int_a^b h(z)dz = \int_a^b f(z)P(z)dz = E_{P(z)}[f(z)]$$

so that the integral can be expressed as an expectation of $f(z)$ over the density function $P(z)$. Thus, if we draw a large number of $z_1, z_2, \ldots, z_Q$ of z with the probability function $P(z)$, then, we can use the average of $f(z_1), f(z_2), \ldots, f(z_Q)$ to approximate the integration as

$$\int_a^b h(z)dz = E_{P(z)}[f(z)] \approx \frac{1}{Q}\sum_{q=1}^{Q} f(z_q)$$

where $P(z)$ is designed to be a stationary distribution of a Markov chain.

4.4 Bayesian structural learning using MCMC

According to [19], learning a Bayesian network from a given data set is NP-hard. The main reason is due to the extremely large size of the candidate graph space. The number of DAGs as a function of the number of nodes K, is super-exponential

Table 4.1: The number of DAGs given the number of nodes K

K	$f(K)$
1	1
2	3
3	25
4	543
5	29281
6	3,781,503
7	1.1×10^9
8	7.8×10^{11}
9	1.2×10^{15}
10	4.2×10^{18}

in K, following the recurrence

$$f(K) = \sum_{k=1}^{K} (-1)^{k+1} \binom{K}{k} 2^{k(K-k)} f(K-k)$$

and examples are given in Table 4.1. Since the number of DAGs is super-exponential in the number of nodes, we cannot exhaustively search the graph space. So we either use a deterministic search algorithm (e.g. greedy search, hill climbing) or a stochastic search algorithm (e.g. simulated annealing, tabu search). Among the family of stochastic search algorithms, MCMC is in practice the most widely used one in structure learning.

Based on the general discussion of MCMC and Bayesian network structural leaning, we discuss in the next section the design and implementation of structural learning algorithm using MCMC following the icebreaking work by [53] which forms the foundation of most state-of-art algorithms today. One of the most recent development can be found in [34].

4.4.1 MCMC structural learning algorithm

The MCMC structural learning algorithm uses MCMC to search the whole graph space with the stationary distribution of the Markov chain converging to the posterior distribution of the graph space given the data set, denoted by $P(g|\mathbf{X})$. This is a stepwise algorithm and in each step l, a graph $Samples[l]$ is drawn from the graph space. The jumping between steps is guided by the ratio R measuring the goodness of a jump. The whole algorithm is stochastic in the way that if a jumping is good, we accept the new graph, if not, we accept it with the probability R. We run the algorithm for sufficiently long so that the Markov chain converges to $P(g|\mathbf{X})$. On

the other hand, MCMC can be considered as an optimization method in which the sampled graphs will ultimately cover the distribution of graph space including the global optimum having the largest $P(g|\mathbf{X})$ – the highest model score.

The overall stages of the algorithm are summarized as following:

- Initialize a graph g_{old}
- $l = 1$ to L until the Markov chain converges to $P(g|\mathbf{X})$
  1. Find all neighbors $nbgs(g_{old})$ of g_{old}.
  2. Pick randomly a graph g_{new} from $nbgs(g_{old})$.
  3. Compute acceptant ratio $R = \frac{score(g_{new})}{score(g_{old})}$.
  4. Move from graph to graph:
     - If $R \geq 1$,
       (a) $g_{old} = g_{new}$.
       (b) $Samples[l] = g_{old}$
     - Else,
       (a) $g_{old} = g_{new}$ with probability R
       (b) $Samples[i] = g_{old}$

4.4.2 Construction of Markov chain in search space

Due to the intractable size of the graph space, the authors in [34] construct a Markov chain by specifying certain rules of how to move from one graph to another step by step, namely, how to construct the transition matrix $\mathbf{T}$ of a chain in that space to make sure the stationary distribution $P(g|\mathbf{X})$ is achieved given sufficient runs of the iteration of the algorithm described in Section 4.4.1. We summarize this in brief.

Consider a graph g_{old} and how to generate g_{new}. Define $nbgs(g_{old})$ as the one-step neighbors of g_{old} including g_{old} itself. All the neighbors of g_{old} are generated by adding or deleting only one edge from g_{old}. The cell $\mathbf{T}_{old,new}$ in transition matrix is one if $g_{new} \in nbgs(g_{old})$ and if g_{new} is also a legal DAG which satisfies the condition that there is no cycles inside g_{new}. As argued by [53], by using this rule to generate the g_{new} and thus the transition matrix $\mathbf{T}$ during each iteration, the Markov chain satisfies the irreducibility, meaning that there is a positive probability of going from one point (graph) of the search space to another and aperiodicity, meaning that returning to any g_{old} takes not fixed but random steps. By the convergence theorem introduced in Section 4.3.1, the chain is assured to converge to its stationary distribution.

Even though the constructed Markov chain is assured to converge to its stationary distribution $\boldsymbol{\pi}^*$, there is so far no guarantee that the stationary distribution is

exactly the posterior distribution $p(g|\mathbf{X})$. In order to achieve this, they define the acceptant ratio R as:

$$R = \frac{\#(nbgs(g_{new}))P(g_{new}|\mathbf{X})}{\#(nbgs(g_{old}))P(g_{old}|\mathbf{X})}$$

where $\#nbgs(g_{new})$ and $\#nbgs(g_{old})$ represent the number of graphs in $nbgs(g_{old})$ and $nbgs(g_{new})$, $P(g_{new}|\mathbf{X})$ and $P(g_{old}|\mathbf{X})$ denote the posterior of the model given data set as defined in (4.2). Since g_{new} and g_{old} differ each other only by one edge, the ratio of $\frac{\#nbgs(g_{new})}{\#nbgs(g_{old})}$ approximates one.

In order to compute R in each step of MCMC, we observe that $P(g_{new}|\mathbf{X})$ and $P(g_{old}|\mathbf{X})$ can both be factorized using Bayesian rule following (4.2) and the denominators $P(\mathbf{X})$, known as normalization constant, cancel out each other in R and the priors $P(g_{new})$ and $P(g_{old})$ cancel each other as well assuming g_{new} and g_{old} share the same prior distribution. After transformation and simplification, R becomes

$$R = \frac{P(\mathbf{X}|g_{new})}{P(\mathbf{X}|g_{old})}$$

which is also called *Bayes Factor*. According to [39], the marginal likelihood which is the integration over the parameter θ_g shown in (4.4) in Bayes factor, can be analytically computed by (4.9) assuming the Dirichlet distribution for each cell in the conditional probability table of each discrete node in g.

$$L(g) = \prod_{k=1}^{K}\prod_{i=1}^{q_k}\frac{\Gamma(M'_{ki})}{\Gamma(M_{ki}+M'_{ki})}\prod_{j=1}^{w_k}\frac{\Gamma(M_{kij}+M'_{kij})}{\Gamma(M'_{kij})} \tag{4.9}$$

When there are K nodes in g, each node k has q_k configurations of its parents and gets w_k discrete values itself. M_{kij} and M'_{kij} count the number of appearances for node k getting value $j = w_k$ with its parents configured as $i = q_k$ in the data set and in the prior respectively. M_{ki} and M'_{ki} count the number of appearances for node k having the configuration of its parents $i = q_k$ in the data set and in the prior respectively. $\Gamma()$ is the gamma function used in Dirichlet distribution.

Further simplification of R is possible from [39] that Bayes factor can be computed using local computations. Specifically,

$$R = \frac{L(g_{new}, k)}{L(g_{old}, k)}$$

where k refers to the random variable whose parents are different in g_{new} and g_{old} and $L(g, k)$ denoted by

$$L(g,k) = \prod_{i=1}^{q_k}\frac{\Gamma(M'_{ki})}{\Gamma(M_{ki}+M'_{ki})}\prod_{j=1}^{w_k}\frac{\Gamma(M_{kij}+M'_{kij})}{\Gamma(M'_{kij})}$$

4.4.3 Convergence of MCMC

As an iterative simulation, MCMC needs to be run long enough so that the posterior probability distribution can be resembled by the stationary distribution of the designed Markov chain. The major concern of using MCMC is to decide appropriately the number of iterations, which is, on the other hand, to develop measurement to monitor the convergence to the Markov chain. If MCMC is stopped too early to achieve the convergence, the simulated posterior distribution may still be far away from real. Otherwise, if MCMC is not stopped when it converges, it would be a waste of computational power. Furthermore, one can not be easily sure if MCMC converges even though it has already be run for a long time.

There are several techniques to measure the convergence of MCMC. The first one is to discard samples from the beginning part of the whole iteration, which is traditionally referred to as the removal of burn-ins because the convergence should be independent of the starting point of the iteration and the effect of different starting point should be diminished as much as possible. The general practice usually disregards the first half of the whole iteration and this proportion is flexible depending on the different tasks.

Another practical technique is to measure the convergence of MCMC by using multiple chains, in particular by simulating multiple sequences with starting points dispersed throughout the parameter space. The quantities of interest such as the mean and the variance are then examined both within and between multiple chains. Besides, only when the distribution of each simulated chain is close to the distribution of all the chains mixed together can they approximate the same target distribution – the posterior probability.

The authors in [32] introduces the concept of *potential scale reduction factor* (PSRF) to monitor the convergence via multiple chains. Suppose we have simulated m sequences each of which has the length n after removing the burn-in samples. Then for the quantity of interest ψ, the simulated draws are denoted as $\psi_{i,j}(i = 1, \ldots, n; j = 1, \ldots, m)$. The authors suggest to compute $\mathcal{B}$ – the between-sequence variance and $\mathcal{W}$ – the within-sequence variance as:

$$\mathcal{B} = \frac{n}{m-1}\sum_{j=1}^{m}(\bar{\psi}_{\cdot,j} - \bar{\psi}_{\cdot,\cdot})^2, \qquad \mathcal{W} = \frac{1}{m}\sum_{j=1}^{m} s_j^2$$

where

$$\bar{\psi}_{\cdot,j} = \frac{1}{n}\sum_{i=1}^{n}\psi_{i,j} \qquad \bar{\psi}_{\cdot,\cdot} = \frac{1}{m}\sum_{j=1}^{m}\bar{\psi}_{\cdot,j} \qquad s_j^2 = \frac{1}{n-1}\sum_{i=1}^{n}(\psi_{i,j} - \bar{\psi}_{\cdot,j})^2$$

The marginal posterior variance of the quantity given by $var(\psi)$ can be estimated

as a weighted average of $\mathcal{W}$ and $\mathcal{B}$ as:

$$\widehat{var}(\psi) = \frac{n-1}{n}\mathcal{W} + \frac{1}{n}\mathcal{B}$$

Remember that the scale of a probability distribution measures its statistical dispersion of the distribution. We monitor the convergence of the iterative simulation by estimating the factor by which the scale of the current distribution for ψ might be reduced if the simulations were continued. The PSRF is estimated by

$$\widehat{\mathcal{R}} = \sqrt{\frac{\widehat{var}(\psi)}{\mathcal{W}}}$$

which declines to one as $n \to \infty$. According to [32], if the PSRF is high, then we have reason to believe that proceeding for more simulations may improve the inference about the target distribution, thus the stationary distribution of MCMC and the quantity of interest ψ. The authors also recommend that normally $\widehat{\mathcal{R}}$ under 1.1 indicates the relatively good convergence of MCMC.

In the structure learning of a Bayesian network, the convergence of MCMC can be monitored by any quantity of interest. In this work, we select the quantity of interest as the appearance of the directed edge between two nodes X_i and X_j. If there is a directed edge from X_i to X_j, $\psi_{ij} = 1$ and zero otherwise. If there is a directed edge from X_j to X_i, $\psi_{ji} = 1$ and zero otherwise. Since the Bayesian network is an directed graph, we need K^2 quantities to fully cover all the situations. The quantities also includes the edge connecting the node to itself when $i = j$. Obviously, this edge does not exist in any candidate graph, the quantities regarding this type of edge is always zeros, leading to $\mathcal{W} = 0$ and therefore infinite $\widehat{\mathcal{R}}$ denoted by "–". When $i \neq j$, ψ_{ij} follows the computation of $\mathcal{B}$ and $\mathcal{W}$ and thus having its own $\widehat{R}$ indicating the goodness of the convergence. However, it is also likely that $\mathcal{W} = 0$ when $i \neq j$ when ψ_{ij} receives only one value throughout all iterations.

4.4.4 Toy example one

To demonstrate the power of structural learning of a Bayesian network using MCMC, we introduce an example firstly proposed in [58] – "Burglary or Earthquake". Mr. Holmes is working in his office when he receives a phone call from his neighbor Dr. Watson, who tells Mr. Holmes that his alarm has gone off. Convinced that a burglar has broken into his house, Holmes rushes to his car and heads for home. On his way home, he listens to the radio, and in the news it is reported that there has been a small earthquake in the area. Knowing that earthquakes have a tendency to make burglar alarms go off, he returns to his work. The Bayesian graph regarding to this example is shown in Figure 4.1. The parameters (local marginal and conditional probabilities) of the bayesian network are as following:

- $P(E) : P(E = 0) = 0.9, P(E = 1) = 0.1$

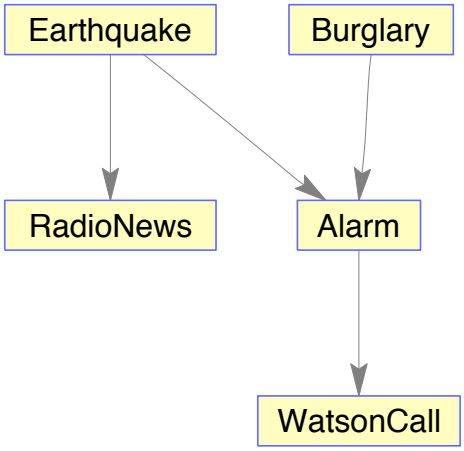


Figure 4.1: Bayesian network for the Burglary or Earthquake example.

- $P(B) : P(B = 0) = 0.01, P(B = 1) = 0.99$
- $P(A|E, B) : P(A = 0|E = 0, B = 0) = 0.95, P(A = 0|E = 1, B = 0) = 0.8, P(A = 0|E = 0, B = 1) = 0.3, P(A = 0|E = 1, B = 1) = 0.01, P(A = 1|E = 0, B = 0) = 0.05, P(A = 1|E = 1, B = 0) = 0.2, P(A = 1|E = 0, B = 1) = 0.7, P(A = 1|E = 1, B = 1) = 0.999$
- $P(W|A) : P(W = 0|A = 0) = 0.7, P(W = 0|A = 1) = 0.05, P(W = 1|A = 0) = 0.3, P(W = 1|A = 1) = 0.95$
- $P(R|E) : P(R = 0|E = 0) = 0.65, P(R = 0|E = 0) = 0.01, P(R = 0|E = 0) = 0.35, P(R = 0|E = 0) = 0.99$

By using the graph structure and the local probabilities, we are able to generate samples sequentially from parents to children in Figure 4.1 by following the sequence $P(E)$, $P(B)$, $P(R|E)$, $P(A|E, B)$ and $P(W|A)$. We would like to use those samples to relearn the Bayesian network by MCMC. The number of samples should be large enough to cover as many the possible configurations of those 5 nodes as possible. The more samples we have, the better the coverage of all the possible combinations of the nodes, the higher quality the structure of a Bayesian network that can be learned from the samples.

100 burn-ins and 400 iterations are used for each of the chain. We use the adjacency matrix to represent the DAG in which element in row i and column j gets one if there is a directed edge from node X_i to node X_j and gets zero if there is no edge between node X_i to node X_j. We use PSRF as the monitor of the convergence

of 5 chains of MCMC starting from the following 5 DAGs.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The chains converge fairly well with

$$PSRF = \begin{bmatrix} - & 1.14 & 1.06 & 1.04 & 1.06 \\ 1.07 & - & - & 1.18 & 1.03 \\ 1.06 & 1.01 & - & - & - \\ 1.16 & 1.18 & 1.00 & - & 1.06 \\ 1.04 & 1.15 & - & 1.06 & - \end{bmatrix}$$

in which all elements marked by "−" have variance zero and all the other elements take value around 1.10 that indicates the good convergence. After the convergence, by combining 400 iterations for each of the 5 chains, we have a chain of length 2,000 from which the DAG with highest score are extracted. All the Bayesian scores associated with iterations are shown in Figure 4.2. Clearly from Figure 4.2, there are a large amount of DAGs taking the highest score. After the examination of those DAGs, they all correspond to the same DAG:

$$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

which is exactly the optimal DAG corresponding to the original Bayesian network described in Figure 4.1. Therefore, by ensuring the convergence of MCMC, we successfully find the optimal DAG.

4.4.5 Toy example two

Now we move back to the first toy example introduced in Chapter 3 shown in Table 3.1 and apply the MCMC structural learning discussed in this chapter to locate

anomalies in that toy example. All 8 instances are used in MCMC scoring process to get the optimal graph g^*.

Before applying the optimization, one may notice that the exhaustive search in the whole DAG space is feasible in this toy example since the dimension of the data set is only $K = 3$. By checking Table 4.1 that shows the number of DAGs as a function of K, only 25 DAGs need to be considered. Thus, as a comparison to the guided search by MCMC, we firstly perform the exhaustive search among all 25 DAGs and get $score_B$ for each of them. The resulting DAGs are shown in the 5 by 5 graph matrix in Figure 4.3(a) with each cell corresponding to a candidate DAG having its $score_B$ denoted by the gray scale in Figure 4.3(b) at the same index position.

From Figure 4.3(b), we can easily find two graphs both of which have maximal $score_B = -33.7751$:

$$score_B(g_1^*, \mathbf{X}) = score_B(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \mathbf{X}) = score_B(g_2^*, \mathbf{X}) = score_B(\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix})$$

On the other hand, it is possible to achieve the g_1^* and g_2^* by MCMC. By using 5 parallel chains with different starting points and 200 iterations of which 100 are removed as burn-ins, the PSRF monitoring the convergence of the edge for each pair

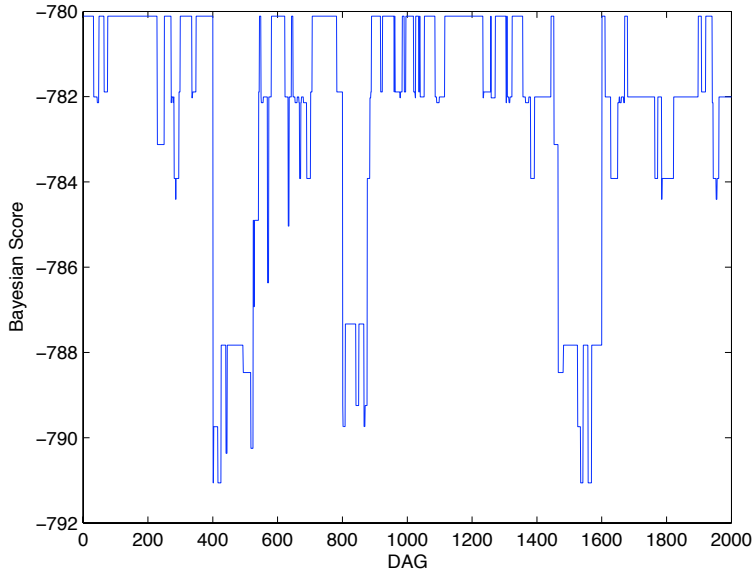
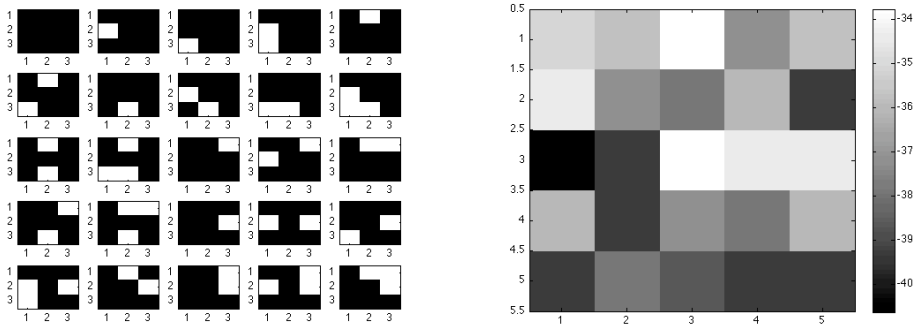


Figure 4.2: The $score_B$ of all 2,000 DAGs generated by 5 chains after MCMC converges.



(a) All DAGs for the toy examples.

(b) Bayesian score for all DAGs for the toy examples.

Figure 4.3: All DAGs and their scores.

of nodes is:

$$PSRF = \begin{bmatrix} - & 1.023 & 1.008 \\ 1.045 & - & 1.032 \\ 1.006 & 1.023 & - \end{bmatrix}$$

that justifies the convergence of MCMC.

By combining all 5 chains together, a chain of length 500 contains the two global optimums g_1^* and g_2^* with exactly the same $score_B$ as those discovered in the exhaustive search.

Figure 4.5(a) and Figure 4.5(b) are equivalently optimal with the same $score_B$. There is no more information in the data set that can differentiate between them. However, they provide two different factorizations of the joint probability $P(\mathbf{x}_n)$.

The resulting two oversimplified DAGs may be surprising because clearly there exists dependencies between X_1 and X_2 and between X_2 and X_3 if we estimate the mutual information between those two pairs by, for example:

$$I(X_1; X_2) = \sum_{x_{n1} \in X_1} \sum_{x_{n2} \in X_2} P(x_{n1}, x_{n2}) log(\frac{P(x_{n1}, x_{n2})}{P(x_{n1})P(x_{n2})})$$

The mutual information for each pair of variables are $I(X_1; X_2) = 0.75$, $I(X_1; X_3) = 0.91$ and $I(X_2; X_3) = 0.405$. As long as $I(X_i; X_j) \neq 0$, X_i and X_j are not independent. The optimal DAG learned from the data set fails to address the dependencies between X_1 and X_2 and between X_2 and X_3. This is because of the effect of penalization of $score_B$ compared with $score_L$. In Figure 4.4, we compare three types of score regarding the 25 candidate DAGs. By using the maximum likelihood (ML)

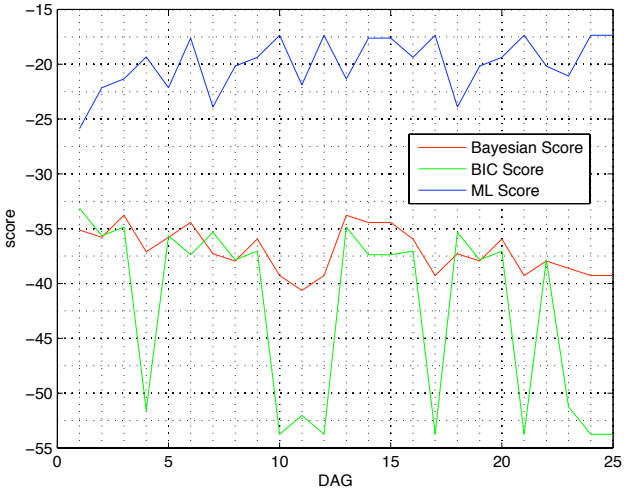
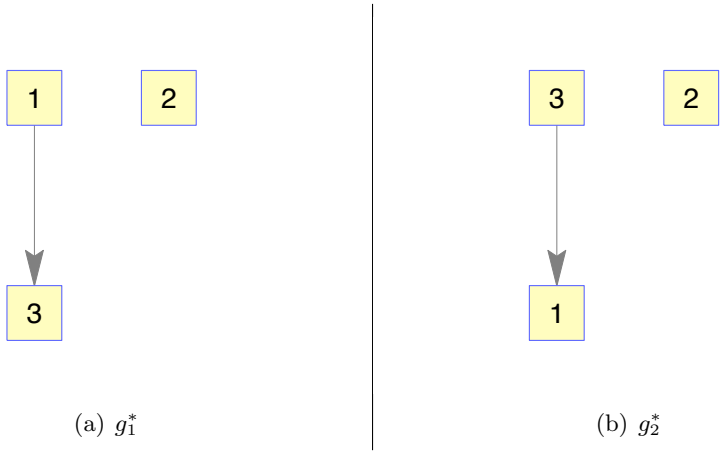


Figure 4.4: Three types of score for 25 DAGs in the toy example.



(a) g_1^* (b) g_2^*

Figure 4.5: Two optimal DAGs learned from MCMC with $score_B$

score, the highest score goes to DAGs $\{g_6, g_{10}, g_{12}, g_{14}, g_{15}, g_{17}, g_{21}, g_{24}, g_{25}\}$ that include all the 3-edge DAGs and three 2-edge DAGs. None of the 1-edge DAGs is preferred. However, by using Bayesian score, two 1-edge DAGs are preferable. Remember from (4.7) in Section 4.2.3 that the $score_B$ can be considered as the likelihood score penalized by the number of samples and the number of independent

(free) parameters. The number of free parameters for each of the 25 DAGs is:

$$\begin{bmatrix} 7 & 13 & 13 & 31 & 13 \\ 19 & 11 & 17 & 17 & 35 \\ 29 & 35 & 13 & 19 & 19 \\ 17 & 35 & 11 & 17 & 17 \\ 35 & 17 & 29 & 35 & 35 \end{bmatrix} \tag{4.10}$$

The behavior of $score_B$ is approximated by $score_{BIC}$ which is computed out by penalizing $score_L$ with the term $\frac{logN}{2} Dim[g]$ where $N = 8$ and $Dim[g]$ from (4.10). As has been discussed before, the asymptotic behavior of $score_B$ observed from $score_{BIC}$ assures that as N increases, more complicated models will eventually get higher score. However, when $N = 8$, the penalizing term outweighs the likelihood term, resulting the preference of the oversimplified DAGs g_1^* and g_2^*.

Factorization one

By following the same computation for the toy example as in Chapter 3 with the two assumptions, we compute the joint probability of each instance by using the factorization $P(\mathbf{x}_n) = P(x_{n1})P(x_{n2})P(x_{n3}|x_{n1})$ from g_1^*:

$$\begin{aligned} P(\mathbf{x}_1) &= P(x_{1,1} = 1, x_{1,2} = 2, x_{1,3} = 2) \\ &= P(x_{1,1} = 1) \times P(x_{1,2} = 2) \times P(x_{1,3} = 2|x_{1,1} = 1) \\ &= 1 \times 0.375 \times 0.5 = 0.94 \end{aligned}$$

Similarly,

$$P(\mathbf{x}_2) = 0.5 \times 0.25 \times 0.5 = 0.0625 \quad P(\mathbf{x}_3) = 0.5 \times 0.375 \times 0.5 = 0.094$$

$$P(\mathbf{x}_4) = 0.5 \times 0.25 \times 0.5 = 0.0625 \quad P(\mathbf{x}_5) = 0.25 \times 0.5 \times 1 = 0.125$$

$$P(\mathbf{x}_6) = 0.25 \times 0.25 \times 1 = 0.0625 \quad P(\mathbf{x}_7) = 0.125 \times 0.5 \times 1 = 0.0625$$

$$P(\mathbf{x}_8) = 0.125 \times 0.25 \times 1 = 0.0313$$

The ranking for all instances is shown in Table 4.2. The distribution of the instances in the log-likelihood space is shown in Figure 4.6. The masks used to locate the anomalous positions for each instance are in Table 4.3.

Table 4.2: Factorization one: $P(\mathbf{x}_n) = P(x_{n1})P(x_{n2})P(x_{n3}|x_{n1})$

configuration	R_1	R_2	R_3	$P(\mathbf{x}_n)$	ranking
$\mathbf{r}_1$	0.5	0.375	0.5	0.094	4
$\mathbf{r}_2$	0.5	0.25	0.5	0.0625	2
$\mathbf{r}_3$	0.5	0.375	0.5	0.094	3
$\mathbf{r}_4$	0.5	0.25	0.5	0.0625	2
$\mathbf{r}_5$	0.25	0.5	1	0.125	5
$\mathbf{r}_6$	0.25	0.25	1	0.0625	2
$\mathbf{r}_7$	0.125	0.5	1	0.0625	2
$\mathbf{r}_8$	0.125	0.25	1	0.0313	1

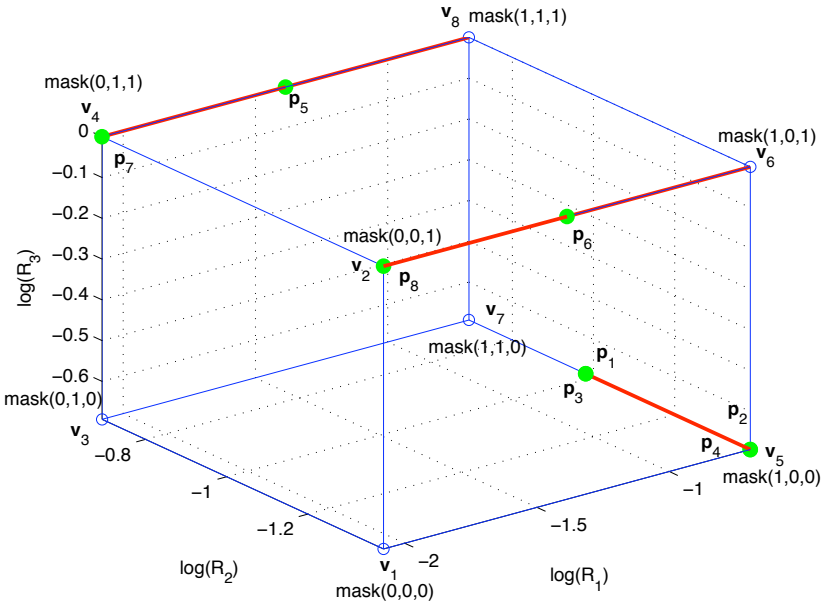


Figure 4.6: Instance distribution for factorization generated by the first optimal Bayesian network g_1^* and $P(\mathbf{x}_n) = P(x_{n1})P(x_{n2})P(x_{n3}|x_{n1})$.

Factorization two

We compute the joint probability of each instance by using the factorization $P(\mathbf{x}_n) = P(x_{n1}|x_{n3})P(x_{n2})P(x_{n3})$ from g_2^*:

$$\begin{aligned} P(\mathbf{x}_1) &= P(x_{1,1} = 1, x_{1,2} = 2, x_{1,3} = 2) \\ &= P(x_{1,1} = 1|x_{1,3} = 2) \times P(x_{1,2} = 2) \times P(x_{1,3} = 2) \\ &= 0.666 \times 0.5 \times 0.375 = 0.125 \end{aligned}$$

Table 4.3: Masks of instances by factorization from g_1^*.

instance	mask	instance	mask
$\mathbf{x}_1 \rightarrow \mathbf{r}_1 \rightarrow \mathbf{p}_1$	$\mathbf{m}_5$	$\mathbf{x}_5 \rightarrow \mathbf{r}_5 \rightarrow \mathbf{p}_5$	$\mathbf{m}_4, \mathbf{m}_8$
$\mathbf{x}_2 \rightarrow \mathbf{r}_2 \rightarrow \mathbf{p}_2$	$\mathbf{m}_5$	$\mathbf{x}_6 \rightarrow \mathbf{r}_6 \rightarrow \mathbf{p}_6$	$\mathbf{m}_2, \mathbf{m}_6$
$\mathbf{x}_3 \rightarrow \mathbf{r}_3 \rightarrow \mathbf{p}_3$	$\mathbf{m}_5$	$\mathbf{x}_7 \rightarrow \mathbf{r}_7 \rightarrow \mathbf{p}_7$	$\mathbf{m}_4$
$\mathbf{x}_4 \rightarrow \mathbf{r}_4 \rightarrow \mathbf{p}_4$	$\mathbf{m}_5$	$\mathbf{x}_8 \rightarrow \mathbf{r}_8 \rightarrow \mathbf{p}_8$	$\mathbf{m}_2$

Similarly,

$$P(\mathbf{x}_2) = 0.666 \times 0.25 \times 0.375 = 0.0624 \quad P(\mathbf{x}_3) = 0.5 \times 0.5 \times 0.5 = 0.125$$

$$P(\mathbf{x}_4) = 0.5 \times 0.25 \times 0.5 = 0.0625 \quad P(\mathbf{x}_5) = 0.5 \times 0.5 \times 0.5 = 0.125$$

$$P(\mathbf{x}_6) = 0.5 \times 0.25 \times 0.5 = 0.0625 \quad P(\mathbf{x}_7) = 0.333 \times 0.5 \times 0.375 = 0.0624$$

$$P(\mathbf{x}_8) = 1 \times 0.25 \times 0.125 = 0.03125$$

The ranking of all the instances is shown in Table 4.4. The distribution of the

Table 4.4: Factorization two: $P(\mathbf{x}_n) = P(x_{n1}|x_{n3})P(x_{n2})P(x_{n3})$

configuration	R_1	R_2	R_3	$P(\mathbf{x}_n)$	ranking
$\mathbf{r}_1$	0.666	0.5	0.375	0.125	4
$\mathbf{r}_2$	0.666	0.25	0.375	0.0624	2
$\mathbf{r}_3$	0.5	0.5	0.5	0.125	4
$\mathbf{r}_4$	0.5	0.25	0.5	0.0625	3
$\mathbf{r}_5$	0.5	0.5	0.5	0.125	4
$\mathbf{r}_6$	0.5	0.25	0.5	0.0625	3
$\mathbf{r}_7$	0.333	0.5	0.375	0.0624	2
$\mathbf{r}_8$	1	0.25	0.125	0.0313	1

Table 4.5: Masks of instances by factorization from g_2^*.

instance	mask	instance	mask
$\mathbf{x}_1 \rightarrow \mathbf{r}_1 \rightarrow \mathbf{p}_1$	$\mathbf{m}_8$	$\mathbf{x}_5 \rightarrow \mathbf{r}_5 \rightarrow \mathbf{p}_5$	$\mathbf{m}_4$
$\mathbf{x}_2 \rightarrow \mathbf{r}_2 \rightarrow \mathbf{p}_2$	$\mathbf{m}_6$	$\mathbf{x}_6 \rightarrow \mathbf{r}_6 \rightarrow \mathbf{p}_6$	$\mathbf{m}_2$
$\mathbf{x}_3 \rightarrow \mathbf{r}_3 \rightarrow \mathbf{p}_3$	$\mathbf{m}_4$	$\mathbf{x}_7 \rightarrow \mathbf{r}_7 \rightarrow \mathbf{p}_7$	$\mathbf{m}_4$
$\mathbf{x}_4 \rightarrow \mathbf{r}_4 \rightarrow \mathbf{p}_4$	$\mathbf{m}_2$	$\mathbf{x}_8 \rightarrow \mathbf{r}_8 \rightarrow \mathbf{p}_8$	$\mathbf{m}_5$

instances in the log-likelihood space is shown in Figure 4.7. The masks used to

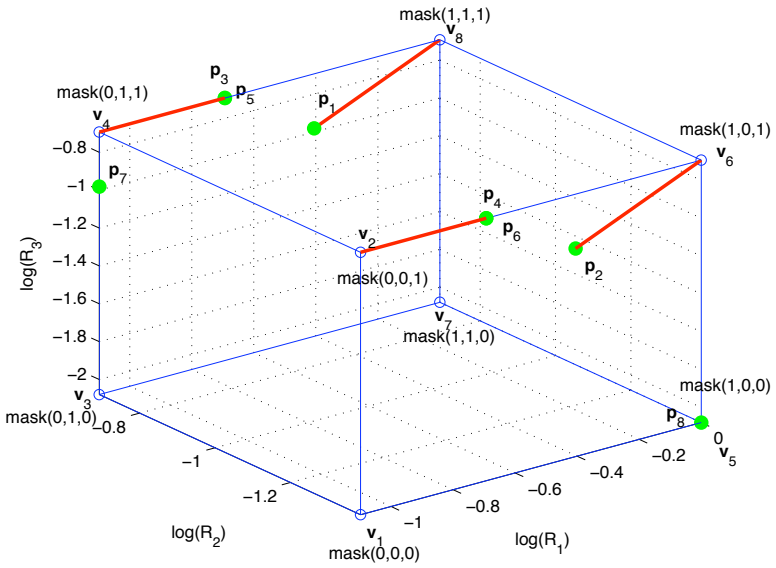


Figure 4.7: Instance distribution for factorization generated by the second optimal Bayesian network g_2^* and $P(\mathbf{x}_n) = P(x_{n1}|x_{n3})P(x_{n2})P(x_{n3})$.

locate the anomalous positions for each instance are shown in Table 4.5.

Understanding the factorizations

So far, we have used 2 factorizations from the last chapter and 2 from this chapter to factorize joint probability given 8 instances and 3 variables:

- Independent factorization:
  $\mathcal{F}_1(P(\mathbf{x}_n)) = \prod_{k=1}^{3} f_k(x_{nk}) = \prod_{k=1}^{3} P(x_{nk})$ with the assumption of statistical independence between variables.

- Fully dependent factorization:
  $\mathcal{F}_2(P(\mathbf{x}_n)) = \prod_{k=1}^{3} f_k(x_{nk}) = \prod_{k=1}^{3} P(x_{nk}|x_{n1}, x_{n2}, \ldots, x_{n[k-1]}, x_{n[k+1]}, \ldots, x_{n3})$ with the assumption that all variables are conditional dependent of all the other variables.

- Bayesian factorization one:
  $\mathcal{F}_3(P(\mathbf{x}_n)) = P(x_{n1})P(x_{n2})P(x_{n3}|x_{n1})$ with no assumption of dependence between variables.

- Bayesian factorization two:
  $\mathcal{F}_4(P(\mathbf{x}_n)) = P(x_{n1}|x_{n3})P(x_{n2})P(x_{n3})$ with no assumption of dependence between variables.

At the end of the last chapter, we explain how $F_1(P(\mathbf{x}_n))$ and $F_2(P(\mathbf{x}_n))$ can be useful to detect anomalies with different understanding of the given data set. In

Table 4.6: Compare the joint probability.

$P_{real}(\mathbf{x}_n)$	$\mathcal{F}_1(P(\mathbf{x}_n))$	$\mathcal{F}_2(P(\mathbf{x}_n))$	$\mathcal{F}_3(P(\mathbf{x}_n))$	$\mathcal{F}_4(P(\mathbf{x}_n))$
0.1250	0.09400	0.2500	0.09400	0.1250
0.1250	0.04700	0.2500	0.06250	0.06240
0.1250	0.1250	0.1250	0.09400	0.1250
0.1250	0.06250	0.2500	0.06250	0.06250
0.1250	0.06250	0.2500	0.1250	0.1250
0.1250	0.03100	0.500	0.06250	0.06250
0.1250	0.02300	0.500	0.06250	0.06240
0.1250	0.00400	1	0.03130	0.03130
distance	0.2208	1.0533	0.1623	0.1563

this chapter, after another two factorizations, we propose an intuitive standard to measure how well those 4 factorizations fit the data set.

In Table 4.6, we compare the joint probability $\mathcal{F}_i(P(\mathbf{x}_n))$ with $i = \{1, 2, 3, 4\}$ computed out from those factorization and the real joint probability $P_{real}(\mathbf{x}_n)$ from Table 3.1. Thus the Euclidean distance between $\mathcal{F}_i(P(\mathbf{x}_n))$ and $P_{real}(\mathbf{x}_n)$ is shown in the last row of Table 4.6 from which we draw the conclusion that with respect to the approximation to $P_{real}(\mathbf{x}_n)$:

- $\mathcal{F}_3(P(\mathbf{x}_n))$ and $\mathcal{F}_4(P(\mathbf{x}_n))$ perform better than $\mathcal{F}_1(P(\mathbf{x}_n))$ and $\mathcal{F}_2(P(\mathbf{x}_n))$.
- $\mathcal{F}_1(P(\mathbf{x}_n))$ is much better than $\mathcal{F}_2(P(\mathbf{x}_n))$.
- $\mathcal{F}_4(P(\mathbf{x}_n))$ is slightly better than $\mathcal{F}_3(P(\mathbf{x}_n))$.

So the masks generated by $\mathcal{F}_4(P(\mathbf{x}_n))$ is the one we trust most to locate the anomalous positions.

Part II

Application

Chapter 5

A Real World Problem

5.1 Problem description

Energy companies normally sell different types of energy utilities such as electricity, gas and district heating. They provide their products to a multitude of customers ranging from families to business, from government to public facilities. Due to the diversity of their customers, energy companies usually offer a variety of products to satisfy various requirements of customers. The products they sell may differ in several ways.

First of all, products are divided into three types of utilities – electricity, gas and district heating – depending on the actual type and business setting of the given energy company. Regarding the products they sell under each of the three utilities, those products additionally differ each other by using different configurations on their product features such as how to charge the customer buying this product, in which time period the product is valid and whether or not the customer receives some discount on the price they pay for the product, all of which are explicitly specified in the configuration of a product and thus the contract between the buyer – the customer and the seller – the energy company.

Secondly, besides the configuration of a product, the details of monitoring the energy flow and consumption are also specified in the contract. For instance, in order to measure the amount of energy being transferred from a seller to a buyer, a metering point needs to be properly set that consists of the information of its geological location, working calendar and the type of energy it measures.

Therefore, a contract between a seller and a buyer normally makes agreement on those following issues:

- The identification of the buyer and the seller.
- The utility the seller provides.
- The configuration of the product the seller offers for the utility.

- The configuration of the corresponding metering point located at the buyer end.

From the perspective of the energy companies, they are facing the contaminations in their contract database because of either hardware malfunction or human carelessness. Those contaminations, or misconfigurations, need to be detected to ensure the health of the whole business. As complicated as any real world business, the data from energy industry is of enormous size and reflects numerous business rules. It may seem easy to detect the contamination if one could find human experts who understand all the business rules of the energy industry. However, finding such a group of experts is impractical and manually checking the contracts stored in the database entry by entry is infeasible even though such experts exist. Therefore, anomaly location techniques introduced in Chapter 3 and 4 fit perfectly to this task. The given data set fits into the characteristics of the special data set introduced at the beginning of Chapter 3 and the problem can be properly solved by using the method from Chapter 4. In this chapter, we firstly introduce the data set from a real world energy company that contains misconfigurations. In the next chapter, we implement the anomaly location techniques discussed in Chapter 4 on this data set and analyze the results.

5.2 Data set description

The whole data set is collected as a table consisting of 211,764 rows and 18 columns. All the table cells are categorical. The columns (attributes) contain all information of the business process. Their names and description are shown in Table 5.1. The second column shows the name of the attributes and the third column shows the unique categorical values the attribute may take and the fourth column shows the meaning of the attributes in the business world.

Multiple rows in the data set may correspond to one single contract inside which one delivery is described by multiple rows each of which specifies a single product component in the contract.

Conceptually, the whole data set contains 6 high-level entities involved in the business process – buyer, seller, utility, delivery, product and metering point. The 6 entities are unified together with other supportive information to form a contract having the following structure with the attribute IDs matched to Table 5.1:

- Contract: X_1
  - Buyer: X_3
  - Seller: X_2
  - Contract Type: X_{14}
  - Utility: X_{15}

Table 5.1: Column attributes description.

X_i	name	size	description
X_1	ContractID	9729	serial number (SN) of the contract
X_2	SellerID	3	only three sellers are involved in the data set.
X_3	BuyerID	6580	SN of the energy consumer
X_4	DeliveryID	9749	SN of delivery inside a contract
X_5	DeliveryType	2	how often the reading of the consumption is checked
X_6	Del.Start	168	the starting time of the delivery
X_7	Del.Stop	495	the ending time of the delivery
X_8	MPID	6675	SN of the metering point
X_9	MEAS.Type	15	various measurements of gas, electricity and heating
X_{10}	MEAS.DataType	5	how measurements are reported
X_{11}	MEAS.CAL	8 with missing	the measuring period
X_{12}	MEAS.Start	2488	the starting time of the metering point
X_{13}	MEAS.Stop	490	the ending time of the metering point
X_{14}	ProductID	4	general feature of the product
X_{15}	UtilityID	3	electricity, gas or district heating
X_{16}	PComponent	4	product component related to charging
X_{17}	PC.CalID	7 with missing	the working period of a product component
X_{18}	Pricelist	2	the price of a product component

  - Deliveries: X_4
    - Delivery 1: X_4
      - Delivery Type: X_5
      - Validity: X_6 and X_7
      - Product: X_{14}
      - Price List: X_{18}
      - Metering Point: X_8
    - Delivery 2: X_4
    - ...

  One contract may contain several deliveries each of which includes the configurations of a product and its metering point. The diversity of products directly reflects the complexity of the energy industry, which can be realized by product components as following:

- Product: X_{14}
  - Product Type: X_{14}
  - Utility: attribute X_{15}
  - Product Components: X_{16}

        * Component 1: X_{16}
            · Calendar: X_{17}
        * Component 2: X_{16}
            · ...
        * ...

    Another set of important configurations are related to the metering points with the structure revealed as following:

- Metering Point: X_8
    - Utility: X_{15}
    - Measurements: no direct mapping to the data set
        * Measurement 1: no direct mapping to the data set
            · Validity: X_{12} and X_{13}
            · Data Type: X_{10}
            · Measurement Type: X_9
            · Calendar: X_{11}
        * Measurement 2
        * Measurement 3
        * Measurement 4

In order to further analyze the data set, the whole data set is then divided into 4 sections based on three utilities and their sellers.

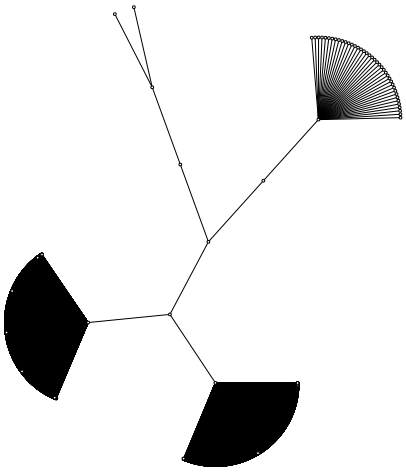
1. Electricity: 211,419 rows
    (a) $Seller_A$: covers 62,707 rows
    (b) $Seller_B$: covers 148,712 rows
2. Gas: 21 rows
    (a) $Seller_C$: covers 21 rows
3. District Hearing
    (a) $Seller_C$: covers 324 rows

A visualization of the whole data set is shown in Figure 5.1.

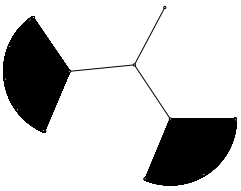
5.2.1 Utility 1: electricity with seller A and seller B

Electricity is sold by both $Seller_A$ and $Seller_B$ and covers most of the data set. The contracts related to electricity are divided by two sellers for the reason that two sellers represent two energy companies having their own business models and rules.

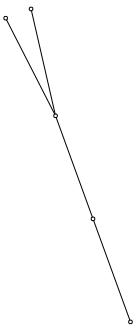
Figure 5.1: The whole data set with 3 columns: root→utility→seller→buyer. 3 utilities are divided from root node to expand to 3 directions each of which follows the order of utility→seller→buyer. Electricity dominates the whole data set with over-condensed (due to the fact that $Seller_A$ and $Seller_B$ each includes more than 6,000 buyers) connections with buyers as the leaf nodes. Gas only involves 2 buyers as the leaf nodes. District heating involves 40 buyers as the leaf nodes.



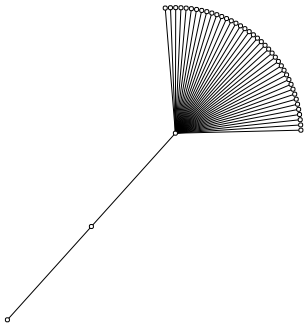
(a) root → all



(b) root → electricity



(c) root → gas



(d) root → district heating

Table 5.2: Summary of contracts related to $Seller_A$.

utility	electricity		
contracts	3138	buyers	3077
deliveries	3144	delivery types	2
metering points	3088	metering type	6
metering data types	4	metering calendars	8
productIDs	1	product components	2
product calendars	6	price list	2
total rows	62707	rows with good Cal	8324
rows with only X_{11} missing	2185(3.5%)	rows with only X_{17} missing	28506(45.5%)
rows with both X_{11}, X_{17} missing	2118(3.4%)	rows with no missing	29898(47.7%)
rows with either missing	32809(52.3%)		

Table 5.3: Summary of contracts related to $Seller_B$.

utility	electricity		
contracts	6547	buyers	6405
deliveries	6558	delivery types	2
metering points	6488	metering type	6
metering data types	4	metering calendars	8
productIDs	2	product components	4
product calendars	5	price list	1
total rows	148712	rows with good Cal	11132
rows with only X_{11} missing	5714(3.8%)	rows with only X_{17} missing	67028(45.1%)
rows with both X_{11}, X_{17} missing	5506(3.7%)	rows with no missing	70464(47.4%)
rows with either missing	78248(52.6%)		

Seller A

Summary of $Seller_A$ shown in Table 5.2 demonstrates the basic characteristics of those contracts with $Seller_A$. The integers denote the number of all possible unique values of the according statistics in the table.

Seller B

Similarly, the summary of $Seller_B$ is shown in Table 5.3.

5.2.2 Utility 2: gas with seller C

The summary of $Seller_C$ selling gas is shown in Table 5.4. Note that due to the speciality of district heating which is provided by $Seller_C$ to its 2 buyers all the year round, column 11 (MEAS.CAL) and column 17 (PC.CALENDARID) are always missing value indicating the gas is offered and transmitted all the time.

Table 5.4: Summary of contracts related to $Seller_C$ with gas.

utility	gas		
contracts	4	buyers	2
deliveries	5	delivery types	1
metering points	4	metering type	2
metering data types	2	metering calendars	0 (Missing)
productIDs	2	product components	2
product calendars	0 (Missing)	price list	1
total rows	21	rows with good Cal	21
rows with only X_{11} missing	21(100%)	rows with only X_{17} missing	21(100%)
rows with both X_{11}, X_{17} missing	21(100%)	rows with no missing	21(100%)
rows with either missing	0(0%)		

Table 5.5: Summary of contracts related to $Seller_C$ with district heating.

utility	district heating		
contracts	40	buyers	40
deliveries	42	delivery types	1
metering points	40	metering type	8
metering data types	3	metering calendars	0 (Missing)
productIDs	1	product components	2
product calendars	0 (Missing)	price list	1
total rows	324	rows with good Cal	312
rows with only X_{11} missing	324(100%)	rows with only X_{17} missing	324(100%)
rows with both X_{11}, X_{17} missing	324(100%)	rows with no missing	324(100%)
rows with either missing	0(0%)		

5.2.3 Utility 3: district heating with seller C

The summary of $Seller_C$ selling district heating is shown in Table 5.5. Similarly as gas, due to the speciality of heating which is provided by $Seller_C$ to its 40 buyers at the same time period (winter) of a year, column 11 (MEAS.CAL) and column 17 (PC.CALENDARID) always have missing values indicating the heating is offered at the same time of the whole year.

5.3 Preprocessing and visualization

The first step of data processing focuses on the nature of those missing values in the data set. We demonstrate the way to correctly understand those missing values. Then in the second step, we manually define a rule with respect of the correctness of calendars in the data set. This intuitive rule serves as an example to show

that misconfiguration location can be performed by just specifying rules and check those contract records that break those rules. However, due to the little domain knowledge, we cannot rely on this rule-based anomaly detection since we barely know any business rules. In the last step, we formally define 4 subsets after attributes reduction and they are used in the next chapter to locate the anomalous locations.

5.3.1 Missing values

Table 5.6: $Seller_A$ and $Seller_B$: some examples showing the rules of missing values in X_{11} and X_{17}.

seller	appearances (rows)	X_9	X_{10}	X_{16}	X_{11} is missing	X_{17} is missing
$Seller_A$	2117	1	3	1	Y	Y
	2184	1	3	2	Y	N
	5725	3	1	1	N	Y
	5724	3	2	1	N	Y
	8402	3	6	1	N	Y
	217	1080	1	1	N	Y
$Seller_B$	5363	1	3	1	Y	Y
	5431	1	1	1	Y	Y
	5499	1	3	2	N	Y
	12013	3	1	1	N	Y
	12018	3	2	1	N	Y
	5363	1	3	2	Y	N

As has been stated before, when it comes to $Seller_C$, the missing value is understandable. However, there are still a large portion of data containing missing values regarding $Seller_A$ and $Seller_B$. This section we uncover the rules leading to those missing values.

We investigate individually all the contracts related to $Seller_A$ and $Seller_B$ and find that attributes X_9, X_{10} and X_{16} decide as a whole the appearances of missing values in X_{11} and X_{17}. Note that we use the actual value directly from the data set. One does not need to understand the real meaning of those values since this needs some expertise from the energy industry. The missing values are generated by rules with some examples shown in Table 5.6.

Notice that for both sellers, when the values of X_9, X_{10} and X_{16} are fixed, the appearance of missing values in either X_{11} or X_{17} is decided. This can be explained as the rule for missing values. Note that for $Seller_A$ there exists only two out of 32,809 rows that violate the pattern shown in Table 5.6 which are very likely to be

misconfigurations. In $Seller_B$ there exists 1,118 (1.43% of 78,248) rows that violate this pattern. However, by a close observation, all the violations from $Seller_B$ result from the alternation between value "141440" and the missing value. For instance, there are in all 5,431 rows in $Seller_B$ with $X_9 = 1, X_{10} = 1, X_{16} = 1$. We expect $X_{11} = Missing$ and $X_{17} = Missing$ according to the rule in Table 5.6. Instead, all the 68 violations out of 5,431 rows take the value $X_{11} = Missing$ which is correct and $X_{17} = 141440$ which violates the rule. Similarly, by checking all the other rules causing missing values in $Seller_B$, we find that all the violations are caused by replacing the attribute that is meant to be missing by the value of "141440". We suspect that the value alternation may be caused by a large scale machine malfunction. The actual reason is beyond our knowledge to induce, which would be left for the experts who have a better understanding of this application domain.

Briefly put, there is no need to fill the missing values, they are missing because they obey some business rules. Hence, missing values are missing because they should be missing.

5.3.2 Filtering by rules

We propose here a rule as an example to discover a large proportion of misconfigurations from the data set before going to any further discussion. Consider the 4 columns $\{X_6, X_7, X_{12}, X_{13}\}$ in Table 5.1. We define the rule $rule_0 = (X_6 \geq X_{12}) \bigcap (X_7 \leq X_{13}) \bigcap (X_6 < X_7) \bigcap (X_{12} < X_{13})$. By applying this rule, we detect 19,789 out of 211,764 rows that have the configuration violating $rule_0$, which means 90.6% of the whole set has the inconsistent calendar information for the products and their metering points. We believe that there are likely to be more extra inconsistencies hidden inside those configurations with wrong calendars. Therefore, we continue to use those configuration with both good and bad calendars for future analysis.

5.3.3 Attributes reduction

As explained before, in order to facilitate the preceding anomaly location in the next chapter, we divide the whole data set into subsets by utilities and their sellers. Since the purpose of this work is to perform the anomaly location within the categorical data set, there are several attributes that are not related to this task. In particular, for all the 18 attributes describing a contract (row), we exclude 4 columns $\{X_6, X_7, X_{12}, X_{13}\}$ by using the rule on calendars introduced in Section 5.3.2. Furthermore, another 5 columns $\{X_1, X_2, X_3, X_4, X_8\}$ are also excluded since those are merely the IDs that have nothing to do with any potential misconfigurations. That finally leaves us 9 columns $\{X_5, X_9, X_{10}, X_{11}, X_{14}, X_{15}, X_{16}, X_{17}, X_{18}\}$.

Since X_{15} specifies the type of utility, the data set with those 9 columns are further divided into 3 subsets by X_{15}. Each of the 3 subsets are divided into even smaller subsets by sellers. Therefore, we have the following 4 data sets $\mathbf{X}$s with $K = 8$

dimensions prepared for the anomaly location in the next chapter:

- $\mathbf{X}_{Electricity,Seller_A}$: $N = 62,707$ rows
- $\mathbf{X}_{Electricity,Seller_B}$: $N = 148,712$ rows
- $\mathbf{X}_{Gas,Seller_C}$: $N = 21$ rows
- $\mathbf{X}_{Heating,Seller_C}$: $N = 324$ rows

Before getting into the next chapter, we would like to show another visualization of one of those 4 data sets $\mathbf{X}_{Electricity,Seller_A}$ in Figure 5.2.

At the higher level, the figure shows the tree structure of the configurations related to $Seller_A$ starting from the root node in the middle, branching by the sequence of: $root \rightarrow X_5 \rightarrow X_9 \rightarrow X_{10} \rightarrow X_{11} \rightarrow X_{14} \rightarrow X_{16} \rightarrow X_{17} \rightarrow X_{18}$. Those 8 attributes form 8 levels of the tree after the root node. In the figure, root node expands into firstly 2 branches since there are 2 possible values X_5 can take, thus divide the whole tree into upper and lower part. Continue with the lower part, $X_5 = 1$ is expanded into 6 parts since X_9 can have in all 6 different values when $X_5 = 1$. Following the this pattern, the whole tree is constructed.

When branching, the frequency of a specific node is also taken into consideration. Again, starting from the root, $X_5 = 1$ (the parent of the lower part) and $X_5 = 2$ (the parent of the upper part) have different appearances meaning that there are different number of rows in the data set having $X_5 = 1$ and $X_9 = 2$. The number of rows taking $X_5 = 1$ is 122 and $X_5 = 2$ is 62,585. Different frequency range is marked in the tree by different color. Thus the connection between root node and the node of $X_5 = 1$ is yellow. Similarly the node of $X_5 = 2$ is colored red. Take the lower part again. When $X_5 = 1$, X_9 may take 6 values $\{1, 2, 3, 4, 8, 1080\}$ with different frequencies, thus with different colors. Particularly, $Freq(X_5 = 1, X_9 = 1) = 10$, $Freq(X_5 = 1, X_9 = 2) = 10$, $Freq(X_5 = 1, X_9 = 3) = 74$, $Freq(X_5 = 1, X_9 = 4) = 4$, $Freq(X_5 = 1, X_9 = 8) = 4$ and $Freq(X_5 = 1, X_9 = 1080) = 20$, the sum of which is exactly the frequency of $X_5 = 1$, which is the frequency of their parent node.

Low frequencies are of our special interests because it follows the fundamental assumption in Chapter 3 that misconfigurations appear much less than correct ones. By investigating the figure, not only can we understand general structure of the configurations, but also we spot some special values with extremely low appearances colored by blue, which might be considered as misconfigurations.

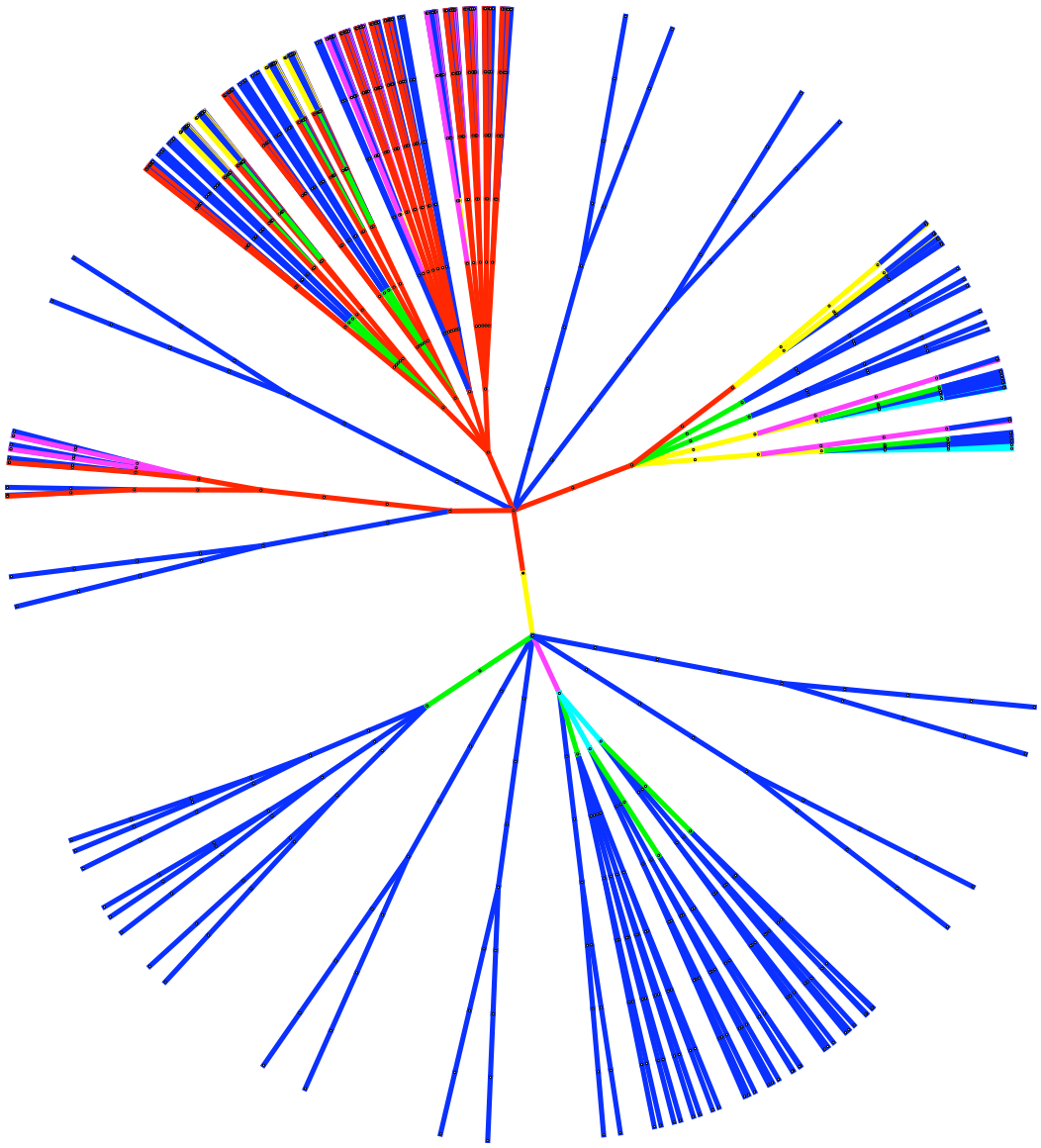


Figure 5.2: Visualization of $\mathbf{X}_{Electricity,Seller_A}$. Colors denote the frequency: blue = [0,10], green = (10,20], aqua = (20,40], purple=(40,80], yellow = (80,160], red=otherwise.

Chapter 6

Anomaly Location with Bayesian Networks

In this chapter, theory from Chapter 4 is applied to the data set introduced in Chapter 5. In particular, structural learning of Bayesian networks finds the optimal network structure with the highest Bayesian score. Then the optimal structure is used to factorize the joint probability, detecting the factors with relatively low frequencies. For each of the configurations, anomalous attributes are located by constructing the hypercube and investigating the distribution of the data points (instances), as discussed in Chapter 3. The final result is a collection of masks corresponding to all the data points. The mask, as discussed in Chapter 3 and 4, can be effectively used to indicate the potential attributes causing a misconfiguration.

We retain 9 columns$\{X_5, X_9, X_{10}, X_{11}, X_{14}, X_{15}, X_{16}, X_{17}, X_{18}\}$ from the original 18-column data set for the reason explained in Chapter 5 and thus divide the 9-column data set into 4 subsets, which generates 4 data sets each of which has 8 columns. The optimal Bayesian network is learned given each of the 4 data sets. However, not every column is necessarily involved in the learning process. The column is not used if it only takes one value for all the instances, which already clearly indicates the statistical independence of this column from the others. Even though those columns are not used in the structural learning, they are manually inserted back into the learnt Bayesian network by simply adding the nodes representing those columns into the learned DAG. No edge is needed for those independent nodes.

For the convenience of the following discussion in this chapter and consistency with the notations used in the two toy examples in Chapter 4, we represent the 8 columns as $\{X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8\}$ and a configuration (instance) as $\mathbf{x}_n$ and its joint probability as $P(\mathbf{x}_n)$. n denotes any given instance.

6.1 Electricity

Over 98% of the data set is covered by contracts related to electricity.

6.1.1 Seller A

This section uses data set $\mathbf{X}_{Electricity,Seller_A}$ with 62,707 rows of 214 unique configurations which contain an uncertain number of misconfigurations. By a close examination, the column X_{14} gets only one value through all the configurations. So it is discarded temporarily and added back to the learned Bayesian network. According to Table 4.1, 7 nodes construct 1.1×10^9 graphs that cover the complete graph space.

By following the discussion in Chapter 4, PSRF checks the convergence of the MCMC searching process. MCMC is run 5 times each of which starts from a different initial graph. In order to achieve the convergence, each of the 5 runs contains 2,000 iterations, generating 2,000 graphs. After the convergence is guaranteed, 5 chains are combined together, forming a collection of 10,000 graphs. Then each of the graphs is scored by $score_B$. The graph with the highest score is considered as the optimal structure for the factorization of the joint probability.

$$PSRF = \begin{bmatrix} - & 1.02 & 1.07 & 1.12 & 1.07 & 1.18 & 1.02 \\ 1.07 & - & 1.02 & 1.10 & 1.00 & 1.02 & 1.02 \\ 1.14 & 1.01 & - & 1.01 & 1.01 & 1.00 & 1.02 \\ 1.27 & 1.11 & 1.01 & - & 1.01 & 1.00 & 1.01 \\ 1.09 & 1.00 & 1.02 & 1.03 & - & 1.07 & 1.02 \\ 1.21 & 1.02 & 1.02 & 1.00 & 1.06 & - & 1.01 \\ 1.07 & 1.00 & 1.02 & 1.05 & 1.01 & 1.04 & - \end{bmatrix}$$

The optimal graph is shown in Figure 6.1 and the factorization of joint probability according to Figure 6.1 is:

$$\begin{aligned} P(\mathbf{x}_n) = {} & P(x_{n,1})P(x_{n,2}|x_{n,1}, x_{n,3}, x_{n,4}, x_{n,7})P(x_{n,3}|x_{n,1}, x_{n,4}, x_{n,7}) \\ & P(x_{n,4}|x_{n,1})P(x_{n,5})P(x_{n,6}|x_{n,2}, x_{n,3}, x_{n,4}, x_{n,7})P(x_{n,7}|x_{n,1}, x_{n,4}) \\ & P(x_{n,8}|x_{n,1}, x_{n,2}, x_{n,3}, x_{n,4}, x_{n,6}, x_{n,7}) \end{aligned}$$

In order to intuitively understand the goodness of this factorization, we compare two joint probabilities, real joint probability and the one approximated by the factorization above for each of the 214 unique instances. The average difference on all 214 pairs is 2.95×10^{-15}, which convincingly proves a reliable factorization.

Then, the masks for locating the anomalous attributes in $\mathbf{X}_{Electricity,Seller_A}$ are generated based on the discussion from Chapter 3. For each of the 214 unique configurations, a close examination reveals that there are 4 masks mapping to 8 columns. Those 4 masks follow the same pattern on attributes $x_{n,5}$ and $x_{n,6}$. For instance, the configuration $\{1, 1, 1, 141440, 1, 1, 0, 1\}$ are assigned the following four masks:

1. 0 1 1 1 0 0 1 1

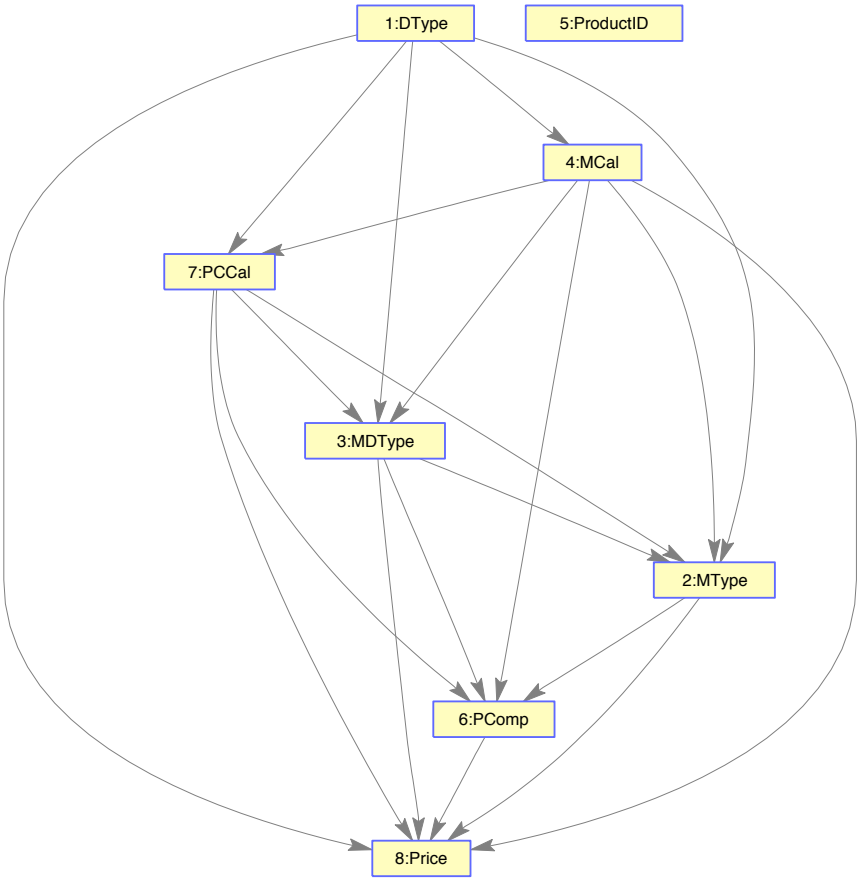


Figure 6.1: Optimal Bayesian network for seller A.

2. 0 1 1 1 0 1 1 1
3. 0 1 1 1 1 0 1 1
4. 0 1 1 1 1 1 1 1

Attributes $x_{n,5}$ and $x_{n,6}$ get $\{m_{n,5}, m_{n,6}\}$ as $\{0,0\}$, $\{0,1\}$, $\{1,0\}$, $\{1,1\}$ which include all the 0-1 combinations. The reason for this phenomenon can be explained as following:

- Attribute X_5 gets only one value for all configurations, resulting in $\mathbf{p}_{max}$ and $\mathbf{p}_{min}$ taking the same value on the 5th direction. So any point in the Euclidean space overlaps $\mathbf{p}_{max}$ and $\mathbf{p}_{min}$ on its 5th direction, getting both masks 0 from $\mathbf{p}_{min}$ and 1 from $\mathbf{p}_{max}$.
- Attribute X_6 receives half 0 and half 1 on $m_{n,6}$. In fact, it is the factorization $P(x_{n,6}|x_{n,2}, x_{n,3}, x_{n,4}, x_{n,7})$ that makes it getting only 1 value in-

Table 6.1: Examples of configurations and masks for seller A.

configuration	X_1	X_2	X_3	X_4	X_7	X_8	frequency
$\mathbf{x}_1$	1	3	3	610869	0	1	2
mask	0	1	1	1	1	1	-
$\mathbf{x}_2$	2	3	1	141440	0	1	5427
mask	1	1	1	1	1	1	-
$\mathbf{x}_3$	2	3	1	141447	0	1	4
mask	1	1	0	1	1	1	-
$\mathbf{x}_4$	2	1080	1	141440	0	1	93
mask	1	1	1	1	1	1	-
$\mathbf{x}_5$	2	4	3	610870	0	1	1
mask	1	1	1	0	1	1	-
$\mathbf{x}_6$	2	1080	1	141452	141452	1	37
mask	1	1	1	1	1	1	-

stead of 2 because there exists another clear pattern in the data set that if $x_{n,2}, x_{n,3}, x_{n,4}, x_{n,7}$ are fixed, there is only one value $x_{n,6}$ can get.

So both $m_{n,5}$ and $m_{n,6}$ are useless to explain the anomalous location and are excluded from the masks. Not surprisingly, after this operation, every single unique configuration is assigned only one mask. Out of 214 different configurations, 172 of them get one anomalous attribute and 42 of them are perfectly normal. The average frequency of all the configurations getting 1 anomalous attribute is 8.94 out of 62,707 while the average frequency of all the normal configurations is 1,456.40 out of 62,707, which is consistent with the fundamental assumption made at the beginning of Chapter 3 that normal configurations dominate the whole data set.

Table 6.1 shows some of the examples of configurations and their masks. Generally, the higher the frequency, the more likely the configuration is normal. Configuration $\mathbf{x}_2$, as one of the most frequent, is perfectly normal, as well as $\mathbf{x}_4$ and $\mathbf{x}_6$. Compared with the total number of rows 62,707, the frequencies of $\mathbf{x}_4$ and $\mathbf{x}_6$ are both small, however, they are considered as normal due to the use of effective factorization.

6.1.2 Seller B

Similarly as the experiment on seller A, this section uses data set $\mathbf{X}_{Electricity,Seller_B}$ with 148,712 rows containing 153 unique configurations which include an uncertain number of misconfigurations. The column X_8 gets only one value through all the configurations. So it is discarded temporarily and added back to the learned Bayesian network. The graph space has the same size as that of seller A.

PSRF is still employed to monitor the convergence of the MCMC searching process. MCMC is run 5 times each of which starts from different initial graph. Each of the 5 runs contains 2,000 iterations, thus generating 2,000 graphs. Even though the size of the data set doubles compared with that of seller A, the number of unique configurations declines. In fact 2,000 iterations suffice to achieve the convergence. After the convergence has been justified by PSRF, 5 chains are combined together, forming a collection of 10,000 graphs each of which is scored by $score_B$. The graph with the highest score is considered as the optimal structure for the factorization of the joint probability.

$$PSRF = \begin{bmatrix} - & 1.01 & 1.00 & 1.01 & 1.08 & 1.01 & 1.01 \\ 1.03 & - & 1.00 & 1.06 & 1.15 & 1.01 & 1.11 \\ 1.00 & 1.09 & - & 1.10 & 1.11 & 1.00 & 1.00 \\ 1.03 & 1.11 & 1.10 & - & 1.20 & 1.06 & 1.18 \\ 1.02 & 1.01 & 1.05 & 1.19 & - & 1.06 & 1.35 \\ 1.00 & 1.01 & 1.00 & 1.10 & 1.26 & - & 1.06 \\ 1.01 & 1.03 & 1.00 & 1.19 & 1.21 & 1.06 & - \end{bmatrix}$$

The optimal graph is shown in Figure 6.2 and the factorization of the joint probability according to Figure 6.2 is:

$$\begin{aligned} P(\mathbf{x}_n) = {} & P(x_{n,1}|x_{n,4}, x_{n,5}, x_{n,6}, x_{n,7})P(x_{n,2}|x_{n,3}, x_{n,4}, x_{n,5}, x_{n,6}, x_{n,7}) \\ & P(x_{n,3}|x_{n,1}, x_{n,4}, x_{n,5}, x_{n,6}, x_{n,7})P(x_{n,4}|x_{n,5})P(x_{n,5})P(x_{n,6}|x_{n,4}, x_{n,5}, x_{n,7}) \\ & P(x_{n,7}|x_{n,4}, x_{n,5})P(x_{n,8}) \end{aligned}$$

Both joint probabilities — real joint probability and estimated one by the factorization above, are compared on each of the 153 unique instances. The average difference on all 153 pairs is 3.80×10^{-14}, which is small enough to convincingly prove an accpetable approximation.

For seller B, the attribute X_8 causes the phenomenon of multiple masks, for the same reason explained in the experiment of seller A. So $m_{n,8}$ is useless to explain the anomalous location and is excluded from the masks. After this operation, every single unique configuration gets only one mask. Out of 153 different configurations in all 148,712 rows, 107 of them get 1 anomalous attribute, 6 of them get 2 anomalous attributes and 40 of them are perfectly normal. The average frequency of all the configurations getting 1 anomalous attribute is 16.13 while getting 2 anomalous attributes is 6 and perfectly normal is 3,673.45, which again is consistent with the fundamental assumption made before. Some examples are shown in Table 6.2.

Table 6.2: Examples of configurations and masks for seller B.

configuration	X_1	X_2	X_3	X_4	X_5	X_6	X_7	frequency
$\mathbf{x}_1$	1	1	1	141440	2	1	0	12
mask	0	0	1	1	1	1	1	-
$\mathbf{x}_2$	1	1	1	141440	4	1	141440	3
mask	1	1	1	1	0	1	1	-
$\mathbf{x}_3$	1	3	6	141440	2	1	0	3
mask	0	1	1	1	1	1	1	-
$\mathbf{x}_4$	2	3	1	141440	2	1	0	12,000
mask	1	1	1	1	1	1	1	-
$\mathbf{x}_5$	1	8	1	141440	2	1	0	4
mask	0	0	1	1	1	1	1	-
$\mathbf{x}_6$	2	3	2	141451	2	1	141440	180
mask	1	1	1	1	1	1	1	-
$\mathbf{x}_7$	2	3	3	141447	2	1	0	5364
mask	1	1	1	1	1	1	1	-
$\mathbf{x}_8$	2	3	6	141440	2	1	0	1
mask	1	1	0	1	1	1	1	-
$\mathbf{x}_9$	2	3	6	141440	2	2	141451	68
mask	1	1	1	1	1	1	0	-
$\mathbf{x}_{10}$	2	8	1	0	2	1	0	39
mask	1	1	1	1	1	1	1	-
$\mathbf{x}_{11}$	2	3	6	141452	2	4	141440	1
mask	1	1	1	1	1	1	0	-
$\mathbf{x}_{12}$	2	3	2	141440	4	1	0	5
mask	1	1	1	1	0	1	1	-
$\mathbf{x}_{13}$	1	3	3	610869	2	1	0	4
mask	1	1	1	0	1	1	1	-

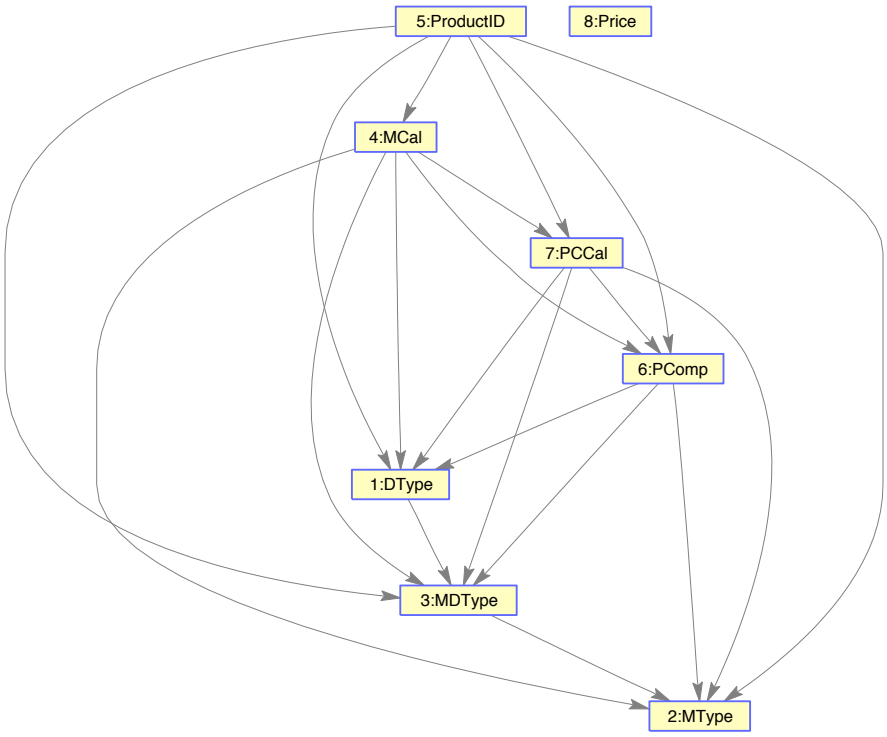


Figure 6.2: Optimal Bayesian network for seller B.

6.2 Gas

$\mathbf{X}_{Gas,Seller_C}$ only covers 21 out of 211,764 rows in the whole data set. There are only 9 unique configurations in all 21 rows. Even though it is fairly easy for any human expert to evaluate those configurations one by one to detect anomalous attributes, for the completeness of this work, the same methodology for seller A and B is repeated in this section. Note that 21 rows with 9 unique configurations are not statistically reliable to draw any significant conclusion of the general population and its internal conditional dependencies.

Another speciality of $\mathbf{X}_{Gas,Seller_C}$ is that 4 attributes X_1, X_4, X_7, X_8 get single value across all the data set, leaving 4 attributes X_2, X_3, X_5, X_6 to be taken into consideration. From the perspective of MCMC, 5 different chains with 200 iterations

for each of them suffice to make the satisfactory convergence with

$$PSRF = \begin{bmatrix} - & - & - & 1.01 \\ 1.12 & - & 1.01 & 1.00 \\ 1.11 & 1.01 & - & 1.01 \\ - & 1.00 & 1.00 & - \end{bmatrix}$$

with "−" denoting zero variance, as has been discussed in Chapter 4.

As the case of the second toy example in Chapter 4, multiple optimal networks are expected since the data set is too small to fully differentiate between those optimal graphs by $score_B$. In fact, from 543 possible graphs, 6 equivalently optimal Bayesian networks are returned from the MCMC process, as shown in Figure 6.3.

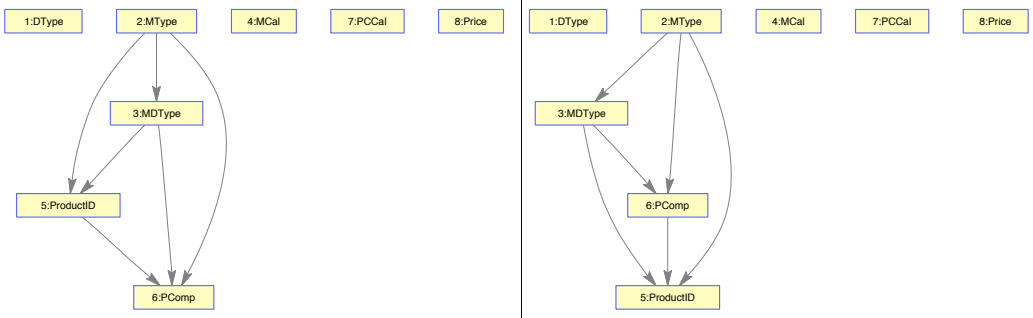
By following the 6 optimal Bayesian networks, joint probability can be factorized into 6 forms, leading to the dilemma of deciding the factorization to be used. Unfortunately, there is nothing one can do without any further knowledge of the problem domain, i.e., *a priori* knowledge is needed at this stage. Simply comparing the approximation accuracy between the factorized joint probability and real joint probability offers no help since this computation shows all 6 factorizations are exact factorizations with 100% accuracy.

6.3 District heating

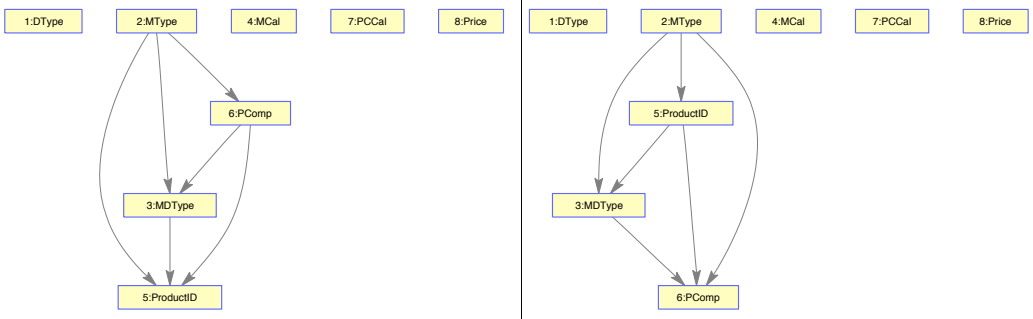
$\mathbf{X}_{Heating,Seller_C}$ covers 324 rows with 18 unique configurations. It is more statistically reliable than $\mathbf{X}_{Gas,Seller_C}$ yet still not informative enough compared with $\mathbf{X}_{Electricity,Seller_A}$ and $\mathbf{X}_{Electricity,Seller_B}$. The attributes $\{X_1, X_4, X_5, X_7, X_8\}$ get only one possible value and need to be excluded from the structural learning process, which leaves only 3 attributes $\{X_2, X_3, X_6\}$. This makes the exhaustive search feasible since the graph space consists only 25 candidates that have already been shown in Figure 4.3(a) in Chapter 4.

The $score_B$ for all the DAGs are:

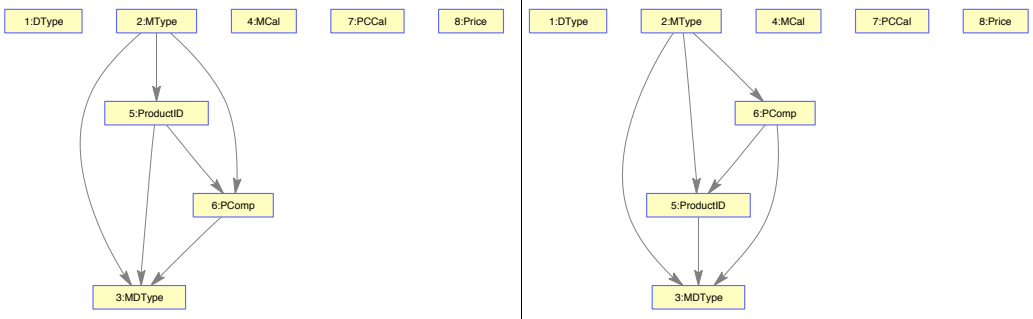
$$\begin{bmatrix} -464.77 & -461.55 & -462.39 & -460.68 & -239.34 \\ -236.96 & -455.91 & -452.69 & -453.53 & -451.82 \\ -241.21 & -238.82 & -238.16 & -234.93 & \underline{-12.73} \\ -229.30 & \underline{-14.59} & -432.99 & -429.76 & -430.60 \\ -428.89 & -207.56 & -371.66 & -368.44 & -146.23 \end{bmatrix}$$



(a) g_1^* (b) g_2^*



(c) g_3^* (d) g_4^*



(e) g_5^* (f) g_6^*

Figure 6.3: 6 optimal DAGs learned from MCMC with $\mathbf{X}_{Gas,Seller_C}$

Thus the best DAG g_1^* and second best DAG g_2^* are:

$$g_1^* = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad g_2^* = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Meanwhile, the optimal DAG found by 5 MCMC chains with 200 iterations is

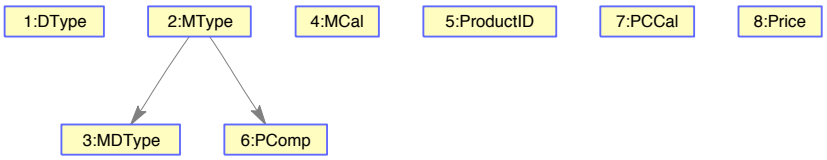


Figure 6.4: Optimal Bayesian network for $\mathbf{X}_{Heating,Seller_C}$.

shown in Figure 6.4, which is exactly the best DAG g_1^* by exhaustive search. The convergence monitor

$$PSRF = \begin{bmatrix} - & - & 1.01 \\ 1.00 & - & - \\ - & 1.02 & - \end{bmatrix}$$

Thus the joint probability can be factorized as:

$$P(\mathbf{x}_n) = P(x_{n,1})P(x_{n,2})P(x_{n,3}|x_{n,2})P(x_{n,4})P(x_{n,5})P(x_{n,6}|x_{n,2})P(x_{n,7})P(x_{n,8})$$

which is an exact factorization of the real joint probability with zero difference between them.

The 6th factor $P(x_{n,6}|x_{n,2})$ exhibits an unusual pattern. Even though $x_{n,6}$ may get two possible values when $x_{n,2}$ is fixed, the possibilities of getting two values are equal. Therefore, for all the 18 unique configurations, $P(x_{n,6}|x_{n,2}) \equiv 0.5$, which directly makes the 6th element of the mask $m_{n,6}$ being equally likely to take either 0 or 1. So $m_{n,6}$ should be excluded.

Finally, some examples are shown in Table 6.3. There are only 9 unique configurations left after all the unrelated columns are eliminated. In the table, only $\{x_{n,2} = 22, x_{n,3} = 1\}$ and $\{x_{n,2} = 23, x_{n,3} = 1\}$ receive the perfect mask $\{m_{n,2} = 1, m_{n,3} = 1\}$ indicating normal configurations. All the other 7 configurations are anomalous with possible misconfigured attributes on either $x_{n,2}$ or $x_{n,3}$. One may notice the strangeness when considering, for example, $\mathbf{x}_4$ in which $x_{4,3}$ can only take

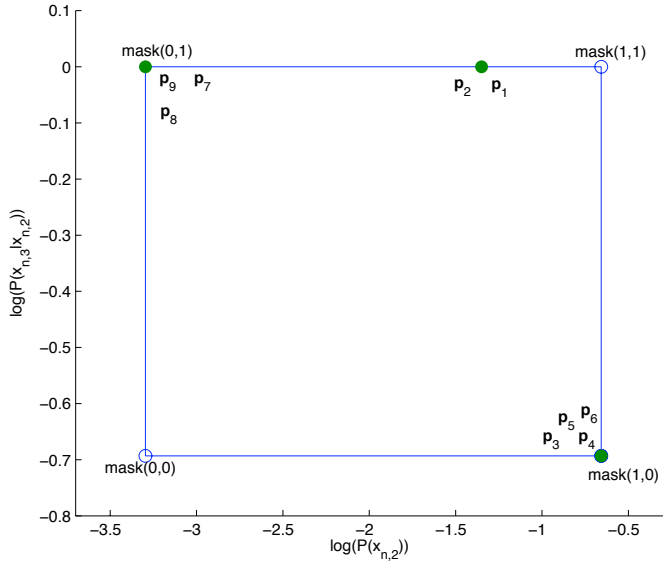


Figure 6.5: The hypercube reduces to a rectangle with $\mathbf{X}_{Heating,Seller_C}$.

Table 6.3: Examples of configurations for $\mathbf{X}_{Heating,Seller_C}$.

configuration	X_2	X_3	frequency	mask	$m_{n,2}$	$m_{n,3}$
$\mathbf{x}_1 \rightarrow \mathbf{r}_1 \rightarrow \mathbf{p}_1$	22	1	42	$\mathbf{m}_1$	1	1
$\mathbf{x}_2 \rightarrow \mathbf{r}_2 \rightarrow \mathbf{p}_2$	23	1	42	$\mathbf{m}_2$	1	1
$\mathbf{x}_3 \rightarrow \mathbf{r}_3 \rightarrow \mathbf{p}_3$	23	5	42	$\mathbf{m}_3$	1	0
$\mathbf{x}_4 \rightarrow \mathbf{r}_4 \rightarrow \mathbf{p}_4$	25	3	6	$\mathbf{m}_4$	1	0
$\mathbf{x}_5 \rightarrow \mathbf{r}_5 \rightarrow \mathbf{p}_5$	26	3	6	$\mathbf{m}_5$	1	0
$\mathbf{x}_6 \rightarrow \mathbf{r}_6 \rightarrow \mathbf{p}_6$	27	3	6	$\mathbf{m}_6$	1	0
$\mathbf{x}_7 \rightarrow \mathbf{r}_7 \rightarrow \mathbf{p}_7$	28	3	6	$\mathbf{m}_7$	0	1
$\mathbf{x}_8 \rightarrow \mathbf{r}_8 \rightarrow \mathbf{p}_8$	31	3	6	$\mathbf{m}_8$	0	1
$\mathbf{x}_9 \rightarrow \mathbf{r}_9 \rightarrow \mathbf{p}_9$	65	3	6	$\mathbf{m}_9$	0	1

the value 3 when $x_{4,2} = 25$. However, according to the $\mathbf{m}_4$, $x_{4,3} = 3$ is incorrect. If one needs to correct the value of $x_{4,3}$, there are basically no suitable values to be selected to replace those incorrect values. The reason can be explained in Figure 6.5. The blue rectangle constraints all the data points inside its realm in which $\mathbf{p}_4$ overlaps the bottom right point of the boundary. Even though in the direction of $x_{n,2}$, $\mathbf{p}_4$ behaves perfectly as it reaches the vertex on the bottom right corner, it meanwhile has the longest distance to the top boundary and therefore it is considered as incorrect in the vertical direction. There is no replacement of $x_{4,3}$ because no instance lies on the boundary line between mask(1,0) and mask(1,1).

Chapter 7

Conclusion

Motivated by the task of misconfiguration detection for the energy industry, this work introduces a procedure of anomaly (misconfiguration) detection and location that can be used not only in the energy industry but also in more general scenarios. Given a multivariate categorical data set generated from an application domain, without any training labels, it is fair to believe that normal instances are more common than anomalous ones. Based on this, we establish a fundamental assumption that an anomalous instance takes the attributes appearing with low frequencies in the data set while a normal instance takes attributes appearing with high frequencies. This assumption directs the approach of anomaly detection and location. Anomaly detection considers an instance as a whole and reports whether an instance is anomalous by analyzing its joint probability. However, anomaly detection overlooks the relationships among attributes. Compared with anomaly detection, anomaly location utilizes those relationships and reports, for each instance, its potentially anomalous attributes, thus providing a useful tool for inspecting and correcting it. This work focuses mainly on anomaly location.

Anomaly location needs the information of relationships among attributes to investigate each attribute inside an instance instead of considering an instance as a whole. However, due to the lack of *a priori* knowledge of the application domain generating the data set, this information can only be learnt from data. In this work we assume that this information can be extracted from factorizing the joint probability of an instance into a number of factors equal to the number of attributes. Each factor is modeled by a conditional probability of one attribute given some others. The factorization can be obtained in different ways. One may assume that the attributes are independent of each other or one attribute is dependent on all the others, which are both arbitrary. We look for such a factorization that, for each of its factors, the according attribute and those it conditions on are as dependent as possible. Such a factorization can be realized by learning a Bayesian network from data. The DAG with the maximal Bayesian score by MCMC-based structural learning indicates the optimal factorization. Then a low-frequency factor suggests an anomalous attribute and a high-frequency one suggests a normal attribute. There are several methods to

differentiate low-frenquency factors from high-frequency ones. In this work, we set up a number of prototypical instances (PI) whose factors define a space enclosing all the instances in the data set. Each instance is associated with one of PI using a nearest neighbor approach. As a result, each instance receives a mask from its nearest PI indicating the anomalous attributes.

Our method of anomaly location has several merits. Firstly, it provides masks from which the reasons causing the anomalies are deduced from anomalous attributes. Secondly, it does not rely on any specific domain knowledge. One can simply perform anomaly detection even though he/she understands little about the categorical data set. Thirdly, Bayesian networks learnt from the data set not only provides a factorization but also gives some insights of the data set. The conditional dependencies/independencies discovered by the Bayesian network may offer valuable information to understand the application domain.

With regard to future work, there are three possible improvements. Firstly, if one does have some domain knowledge of the data set, for instance, the conditional dependencies among a group of variables, this knowledge should be embedded into the construction of the Bayesian network as a fixed connection between parent nodes and their child node. When structural search moves from one DAG to another, the new DAG should always retain the fixed connection. Secondly, the score-based structural learning of Bayesian networks suffers from the enormousness of the searching space. MCMC can take extremely long time before the theoretical convergence is reached. When it comes to data sets from some real world applications with hundreds of thousands of configurations across hundreds of dimensions, the convergence may become too slow to accept. It remains an open question of how to speed up the convergence of the Markov chain. Thirdly, the method currently works only on the complete observed categorical data set. When there are missing values, computing the Bayesian score is computationally challenging since it involves marginalization (a complex integration) over all the possible missing values, which usually makes exact computation intractable. Thus methods such as variational Bayesian approximation, Laplace approximation and Cheeseman-Stutz approximation can be used to approximate the posterior. An alternative approach is to perform structural EM algorithm in which a heuristic is constructed to help identify helpful moves during the graph search.

Bibliography

[1] Deepak Agarwal. An empirical bayes approach to detect anomalies in dynamic multidimensional arrays. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 26–33, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. pages 487–499, 1994.

[3] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[4] Malik Agyemang, Ken Barker, and Rada Alhajj. A comprehensive survey of numeric and symbolic outlier mining techniques. *Intelligent Data Analysis*, 10:521–538, December 2006.

[5] S. Albrecht, J. Busch, M. Kloppenburg, F. Metze, and P. Tavan. Generalized radial basis function networks for classification and novelty detetion: self-organization of optional bayesian decision. *Neural Netw.*, 13:1075–1093, December 2000.

[6] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2nd edition, 2010.

[7] Shin Ando. Clustering needles in a haystack: An information theoretic analysis of minority and outlier detection. In *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 13–22, Washington, DC, USA, 2007. IEEE Computer Society.

[8] Daniel Barbará, Ningning Wu, and Sushil Jajodia. Detecting novel network intrusions using bayes estimators. In *Proceedings of the First SIAM Conference on Data Mining*, April 2001.

[9] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 59–68, New York, NY, USA, 2004. ACM.

[10] Stephen D. Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 29–38, New York, NY, USA, 2003. ACM.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1st edition, 2006.

[12] Ferenc Bodon. A fast APRIORI implementation. In Bart Goethals and Mohammed J. Zaki, editors, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, volume 90 of *CEUR Workshop Proceedings*, Melbourne, Florida, USA, 19. November 2003.

[13] R. Bolton and D. Hand. Unsupervised profiling methods for fraud detection. In *Proceedings of the Conference on Credit Scoring and Credit Control VII*, 1999.

[14] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation abstract. Technical report, 2008.

[15] Markus M. Breunig, Hans-Peter Kriegel, and Raymond T. Ng. Optics-of: Identifying local outliers. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '99, pages 262–270, London, UK, 1999. Springer-Verlag.

[16] Simon Byers and Adrian E. Raftery. Nearest neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93:577–584, 1996.

[17] Nitesh V. Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6:1–6, June 2004.

[18] J. Cheng. Learning bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, May 2002.

[19] David Maxwell Chickering. Learning Bayesian networks is NP-Complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

[20] Paul A. Crook, Stephen Marsland, Gillian Hayes, and Ulrich Nehmzow. A tale of two filters - on-line novelty detection. Technical report, 2002.

[21] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Advances in neural information processing systems. pages 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[22] D. Dasgupta and N. S. Majumdar. Anomaly detection in multidimensional data using negative selection algorithm. In *Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume*

02, CEC '02, pages 1039–1044, Washington, DC, USA, 2002. IEEE Computer Society.

[23] Dorothy E. Denning. An intrusion-detection model. *IEEE Transaction on Software Engineering*, 13:222–232, February 1987.

[24] Steve Donoho. Early detection of insider trading in option markets. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 420–429, New York, NY, USA, 2004. ACM.

[25] Haimonti Dutta, Chris Giannella, Kirk D. Borne, and Hillol Kargupta. Distributed top-k outlier detection from astronomy catalogs using the demac system. In *SDM*. SIAM, 2007.

[26] F.Y. Edgeworth. On discordant observations. *Philosoph.Mag.*, pages 364–375, 1887.

[27] Eleazar Eskin. Anomaly detection over noisy data using learned probability distributions. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 255–262, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[28] Martin Ester, Hans peter Kriegel, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[29] Tom Fawcett and Foster Provost. Activity monitoring: noticing interesting changes in behavior. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, pages 53–62, New York, NY, USA, 1999. ACM.

[30] Ryohei Fujimaki, Takehisa Yairi, and Kazuo Machida. An approach to spacecraft anomaly detection problem using kernel feature space. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 401–410, New York, NY, USA, 2005. ACM.

[31] Dani Gamerman and Hedibert F. Lopes. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference, Second Edition (Chapman & Hall/CRC Texts in Statistical Science)*. Chapman and Hall/CRC, 2006.

[32] Stephen P. Brooks Andrew Gelman, Stephen P. Brooksand, and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, 7:434–455, 1998.

[33] Anup K. Ghosh, Aaron Schwartzbard, and Michael Schatz. Learning program behavior profiles for intrusion detection. In *Proceedings of the Workshop on*

Intrusion Detection and Network Monitoring, pages 51–62, Berkeley, CA, USA, 1999. USENIX Association.

[34] Paolo Giudici and Robert Castelo. Improving markov chain monte carlo model search for data mining. *Machine Learning*, 50:127–158, 2003. 10.1023/A:1020202028934.

[35] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: a robust clustering algorithm for categorical attributes. *Inf. Syst.*, 25:345–366, July 2000.

[36] Simon Hawkins, Hongxing He, Graham J. Williams, and Rohan A. Baxter. Outlier detection using replicator neural networks. In *Proceedings of the 4th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK 2000, pages 170–180, London, UK, 2002. Springer-Verlag.

[37] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2 edition, July 1998.

[38] Zengyou He, Joshua Zhexue Huang, Xiaofei Xu, and Shengchun Deng. Mining class outliers: Concepts, algorithms and applications. In *WAIM*, pages 589–599, 2004.

[39] David Heckerman, Dan Geiger, and David M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995. 10.1007/BF00994016.

[40] Tuong Vinh Ho and Jean Rouat. A novelty detector using a network of integrate and fire neurons. In *Proceedings of the 7th International Conference on Artificial Neural Networks*, ICANN '97, pages 103–108, London, UK, 1997. Springer-Verlag.

[41] Victoria Hodge and Jim Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126, October 2004.

[42] Garry Hollier and Jim Austin. Novelty detection for strain-gauge degradation using maximally correlated components. In *ESANN*, pages 257–262, 2002.

[43] Liao Y. Hu, W. and V. R Vemuri. Robust anomaly detection using support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 282–289, 2003.

[44] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[45] Eamonn Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle. Finding the most unusual time series subsequence: algorithms and applications. *Knowl. Inf. Syst.*, 11:1–27, December 2006.

[46] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[47] Yufeng Kou and Chang-Tien Lu. Spatial weighted outlier detection. In *Proceedings of SIAM Conference on Data Mining*, 2006.

[48] Wenke Lee, Salvatore J. Stolfo, and Kui W. Mok. Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review*, 14:533–567, December 2000.

[49] Jessica Lin, Eamonn Keogh, Ada Fu, and Helga Van Herle. Approximations to magic: Finding unusual medical time series. In *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems*, CBMS '05, pages 329–334, Washington, DC, USA, 2005. IEEE Computer Society.

[50] Song Lin, Song Lin, Donald E. Brown, and Donald E. Brown. An outlier-based data association method for linking criminal incidents. In *In Proceedings of 3rd SIAM Data Mining Conference*, 2003.

[51] Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Algorithms for spatial outlier detection. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 597–, Washington, DC, USA, 2003. IEEE Computer Society.

[52] James W. Macdonald and Debashis Ghosh. COPA—cancer outlier profile analysis. *Bioinformatics*, 22:2950–2951, November 2006.

[53] David Madigan, Jeremy York, and Denis Allard. Bayesian graphical models for discrete data. *International Statistical Review / Revue Internationale de Statistique*, 63(2):215–232, 1995.

[54] Larry M. Manevitz and Malik Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, March 2002.

[55] Carla Marceau. Characterizing the behavior of a program using multiple-length n-grams. In *Proceedings of the 2000 workshop on New security paradigms*, NSPW '00, pages 101–110, New York, NY, USA, 2000. ACM.

[56] Aleix M. Martinez, Aleix M. Mart'inez, and Avinash C. Kak. PCA versus LDA. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:228–233, 2001.

[57] Christopher Meek. Causal inference and causal explanation with background knowledge. pages 403–441, August 1995.

[58] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[59] Judea Pearl and T.S. Verma. A theory of inferred causation, 1991.

[60] Vir V. Phoha. *Internet security dictionary*. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[61] Clifton Phua, Damminda Alahakoon, and Vincent Lee. Minority report in fraud detection: classification of skewed data. *SIGKDD Explor. Newsl.*, 6:50–59, June 2004.

[62] Dragoljub Pokrajac. Incremental local outlier detection for data streams. In *In Proceedings of IEEE Symposium on Computational Intelligence and Data Mining*, pages 504–515, 2007.

[63] Gunnar Rätsch, Sebastian Mika, Bernhard Schölkopf, and Klaus-Robert Müller. Constructing boosting algorithms from SVMs: An application to one-class classification. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24:1184–1199, September 2002.

[64] S. Salvador and Chan. Learning states and rules for time-series anomaly detection. Technical report, Department of Computer Science, Florida Institute of Technology Melbourne, 2003.

[65] Shashi Shekhar, Chang Tien Lu, and Pusheng Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 371–376, New York, NY, USA, 2001. ACM.

[66] Bivens A. Embrechts M. Palagiri C. Smith, R. and B Szymanski. Clustering approaches for anomaly-based intrusion detection. In *Intelligent Engineering Systems through Artificial Neural Networks*, pages 579–584, 2002.

[67] Qing Song, Wenjie Hu, and Wenfang Xie. Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,*, 32(4):440 –448, nov. 2002.

[68] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Conditional anomaly detection. *IEEE Trans. on Knowl. and Data Eng.*, 19:631–645, May 2007.

[69] A.N. Srivastava. Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques. 2006.

[70] A.N. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. pages 3853 –3862, mar. 2005.

[71] Ingo Steinwart, Don Hush, and Clint Scovel. A classification framework for anomaly detection. *J. Mach. Learn. Res.*, 6:211–232, December 2005.

[72] Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 418–425, Washington, DC, USA, 2005. IEEE Computer Society.

[73] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[74] Gaurav Tandon and Philip K. Chan. Weighting versus pruning in rule validation for detecting network and host anomalies. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 697–706, New York, NY, USA, 2007. ACM.

[75] James Theiler and D. Michael Cai. Resampling approach for anomaly detection in multispectral images. In *in Proc. SPIE*, pages 230–240, 2003.

[76] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[77] Mangeas M. Weigend, A. S. and A. N. Srivastava. Nonlinear gated experts for time-series: Discovering regimes and avoiding overfitting. *J. Neural Syst.*, 1995.

[78] Gregory Cooper Michael Wagner Weng-Keen Wong, Andrew Moore. Bayesian network anomaly pattern detection for disease outbreaks. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 808–815, Menlo Park, California, August 2003. AAAI Press.