Fall 2019

Cybersecurity Issues in the Context of Cryptographic Shuffling Algorithms and Concept Drift: Challenges and Solutions

Hatim Alsuwat

CYBERSECURITY ISSUES IN THE CONTEXT OF CRYPTOGRAPHIC SHUFFLING ALGORITHMS AND CONCEPT DRIFT: CHALLENGES AND SOLUTIONS

by

Hatim Alsuwat

Bachelor of Computer Science
Taif University, 2008

Master of Engineering
University of South Carolina, 2015

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2019

Accepted by:

Csilla Farkas, Major Professor

Marco Valtorta, Major Professor

John Rose, Committee Member

Chin-Tser Huang, Committee Member

Eva Czabarka, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

DEDICATION

This dissertation is dedicated to my beloved family. I am very thankful for your unlimited love, encouragement, and support that inspired me to complete this long journey.

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere appreciation to my major advisors Prof. Csilla Farkas and Prof. Marco Valtorta for their patience, motivation, knowledge, and support. I am very thankful for their guidance that made my Ph.D. experience exciting and productive.

I also would like to express my sincere appreciation to the rest of my dissertation committee members, Prof. John Rose, Prof. Chin-Tser Huang, and Prof. Eva Czabarka, for their encouragement and wise comments.

Last but not the least, I would like to thank my beloved family: my parents, brothers, sisters, wife, and daughter, for supporting and inspiring me throughout my life.

ABSTRACT

In this dissertation, we investigate and address two kinds of data integrity threats. We first study the limitations of secure cryptographic shuffling algorithms regarding preservation of data dependencies. We then study the limitations of machine learning models regarding concept drift detection. We propose solutions to address these threats.

Shuffling Algorithms have been used to protect the confidentiality of sensitive data. However, these algorithms may not preserve data dependencies, such as functional dependencies and data-driven associations. We present two solutions for addressing these shortcomings: (1) Functional dependencies preserving shuffle, and (2) Data-driven associations preserving shuffle. For preserving functional dependencies, we propose a method using Boyce-Codd Normal Form (BCNF) decomposition. Instead of shuffling the original relation, we recommend to shuffle each BCNF decomposition. The final shuffled relation is constructed by joining the shuffled decompositions. We show that our approach is lossless and preserves functional dependencies if the BCNF decomposition is dependency preserving. For preserving data-driven associations, we generate the transitive closure of the sets of attributes that are associated. Attributes of each set are bundled together during shuffling.

Concept drift is a significant challenge that greatly influences the accuracy and reliability of machine learning models. There is, therefore, a need to detect concept drift in order to ensure the validity of learned models. We study the issue of concept drift in the context of discrete Bayesian networks. We propose a probabilistic graphical model framework to explicitly detect the presence of concept drift using latent variables. We employ latent variables to model real concept drift and uncertainty drift over time. For modeling

real concept drift, we propose to monitor the mean of the distribution of the latent variable over time. For modeling uncertainty drift, we suggest to monitor the change in belief of the latent variable over time, i.e., we monitor the maximum value that the probability density function of the distribution takes over time. We also propose a probabilistic graphical model framework that is based on using latent variables to provide an explanation of the detected posterior probability drift across time.

Our results show that neither cryptographic shuffling algorithms nor machine learning models are robust against data integrity threats. However, our proposed approaches are capable of detecting and mitigating such threats.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In this dissertation, we investigate data integrity threats. For this we first study the limitations of secure cryptographic shuffling algorithms regarding data dependency preservation in relational databases. We then study at the vulnerabilities of machine learning algorithms regarding concept drift detection.

1.1 DATA DEPENDENCIES PRESERVING SHUFFLE

Data breaches have become part of the modern landscape. The combined growth of the internet and data processing capabilities has resulted in a data explosion that has amplified the need for privacy protections [9]. Given the interconnected, technological nature of the world communications, data breaches triggered a growing need to protect peoples' sensitive information. No individual, organization, or government is immune from cyber attacks [9]. For instance, the 2005 MasterCard breach left over 40 million cardholders exposed to potential abuse, while 2006 led to the theft of critical personal data, including names, dates of birth and social security numbers for over 26 million U.S. veterans [71]. Access control, intrusion detection, and data use policies can be crafted to deter and prevent thefts and intrusion. However, current technologies are still vulnerable to intrusion and unauthorized access threats [71].

Database shuffling has long been used as a means of protecting sensitive data [1, 6, 18, 37, 59]. When a malicious user gains access to a database (such as attacks in [8, 20, 38, 44]), shuffling may serve to reduce the possibility of full tuple exposure. We seek a means of ensuring that shuffling algorithms are not easily detected by malicious users by ensuring

that data values appear plausible and consistent. In this work, we address the preservation of Functional Dependencies (FDs) and data-driven dependencies [16, 27].

We demonstrate our approach using the secure cryptographic shuffling algorithm [26]. The aim of the secure cryptographic shuffling algorithm is to diffuse the relationship between data entries in a database and the relationship between the fields of individual data entries. The objectives are to prevent the attacker from obtaining the entire database when the database storage server is compromised. Our solution is compatible with the current relational database design and could be used to provide an additional layer of protection over encryption.

The basic idea of the secure cryptographic shuffling algorithm is illustrated in Table 1.1 and 1.2.

Table 1.1: The original relation before using the secure cryptographic shuffling algorithm

SSN	EmpName	Rank	Salary	Gender
222-22-2222	James	Secretary	65,000	Male
333-33-3333	Alex	Chair	85,000	Male
444-44-4444	Sarah	Professor	75,000	Female
555-55-5555	Kevin	Secretary	65,000	Male
666-66-6666	Mark	Professor	75,000	Male

Table 1.2: The relation after using the secure cryptographic shuffling algorithm

SSN	EmpName	Rank	Salary	Gender
333-33-3333	Kevin	Professor	75,000	Female
555-55-5555	James	Chair	65,000	Male
666-66-6666	Sarah	Secretary	85,000	Male
222-22-2222	Mark	Secretary	75,000	Male
444-44-4444	Alex	Professor	65,000	Male

Consider the relation in Table 1.1 with the following FDs:

1. $SSN \rightarrow EmpName, Rank, Salary, Gender$

Table 1.3: The relation after using our proposed approach with secure cryptographic shuffling algorithm

SSN	EmpName	Rank	Salary	Gender
555-55-5555	Sarah	Chair	65,000	Female
666-66-6666	Kevin	Professor	85,000	Male
222-22-2222	Mark	Secretary	75,000	Male
333-33-3333	James	Professor	85,000	Male
444-44-4444	Alex	Secretary	75,000	Male

2. $Rank \rightarrow Salary$

There is also a commonly known association between attributes names and gender. Shuffling algorithms generally do not preserve data-driven dependencies or functional dependencies. Note that, in Table 1.2 Sarah's gender is "male" and Kevin's is "female"; thus, violating commonly known association between names and genders. Note also that Table 1.2 violates the set of FDs.

There is a need to preserve data dependencies while using shuffling algorithms to ensure that an attacker would consider the shuffled relation to be a valuable source of data. Using our proposed approach will preserve data dependencies are preserved as shown in Table 1.3.

To generate Table 1.3, we first bundle attributes $EmpName$ and $Gender$ together. This ensures that the shuffling will not result in obviously incorrect association such as the 1st and 3rd tuples of Table 1.2. Next, we decompose the relation into two relations:

1. R_1 ($Rank$, $Salary$)

2. R_2 (SSN, $Rank$, [$EmpName, Gender$])

Both R_1 and R_2 are in BCNF form. We individually shuffle R_1 and R_2, then use the natural join to create the shuffled relation in Table 1.3.

In this dissertation, we will provide solutions to address the problems of violations of FDs and data-driven inference attacks by shuffling algorithms.

Remarks. We assume that malicious attackers have full knowledge of the metadata of the database. That is, we assume that malicious attackers know the set of functional dependencies and the set of data-driven dependencies. In cybersecurity, it is a common practice to assume that malicious users know the metadata [55]. Functional dependencies represent real world restrictions. Their violation would result in database inconsistencies. Moreover, data-driven dependencies are frequently publicly known. Therefore, any violation of these dependencies will alert the attackers about the altered state of the database.

We consider database dependencies as fundamental part of databases, therefore crucial to preserve. We recommend that our algorithms are deployed in the following order:

(1) First, all functional dependencies must be preserved.

(2) Second, public knowledge data-driven dependencies must be preserved.

(3) Third, domain-specific data-driven dependencies must be preserved.

1.1.1 Our Approach

Common or publicly known associations among attributes may reveal that a relation has been altered. Therefore, we are interested in modeling associations among database attributes to improve the robustness of secure cryptographic shuffling algorithms.

In this dissertation, we study two types of associations:

(1) functional dependencies, and

(2) data-driven associations among attributes.

We show that shuffling algorithms do not preserve FDs and data-driven associations. Our approach can be used with secure cryptographic shuffling algorithms to preserve FDs and data-driven dependencies.

For our approach, we show how to use functional dependencies, to identify attributes that must be shuffled together. We use statistical methods, such χ^2, Pearson Correlation

Coefficient (*PCC*), and *ANOVA* tests, to model and extract data-driven association among attributes. We use these associations to group attributes to be shuffled together. Thus, limiting the risk of discovery of the shuffle.

The main contributions of our approaches in this section can be summarized as follows:

1. We define two approaches for addressing the shortcomings of shuffling algorithms: FD-preserving shuffle and data-driven association-preserving shuffle.

2. We develop a solution to preserve the set of FDs while applying the secure cryptographic shuffling algorithm.

3. We develop a method to preserve data-driven associations while applying shuffling algorithms.

4. We provide proof sketches of the properties of our algorithms. That is, we show that our algorithms preserve FDs and data-driven dependencies without requiring any change of the shuffling algorithms.

Our analysis and empirical results show that our new approaches are feasible and promising.

1.2 Concept Drift Detection

In recent years, machine learning models are increasingly used in many real-world applications. A common challenge for machine learning systems is to model environments in which data evolves over time, a phenomenon that is commonly known as concept drift [24].

Detecting concept drift is crucial and active research in machine learning systems. Concept drift influences the accuracy and reliability of machine learning models. Current approaches to detect concept drift use latent variables [12, 14]. Latent variables (a.k.a. unobserved variables) are variables that are not immediately observed but instead they are inferred from different variables that are observed and directly measured. An advantage of

concept drift detection techniques that are based on using latent variables is that they tend to estimate the desired effects on the machine learning models more reliably than traditional detection techniques. A large number of observable variables can be aggregated in a model to represent an underlying concept, making it easier to understand the data and detect concept drift over time. However, current efforts for detecting concept drift using latent variables either limited to contentious Bayesian networks [12] or not directly applicable to discrete Bayesian networks [14]. In addition, previous efforts for detecting concept drift using latent variables [12, 14] are limited to naive Bayes classifiers and therefore cannot be used to model concept drift that involves concepts span over multiple variables.

In this dissertation, we propose a technique for detecting concept drift in the context of discrete Bayesian networks using latent variables. Our technique extends the Borchani et al. [12] approach such that it is directly applicable to discrete Bayesian networks. Borchani et al. represent concept drift using unobserved variables in continuous domains, namely in conditional linear Gaussian models. In addition to modeling posterior probability distribution drift, we propose a new method for modeling uncertainty drift. Unlike previous research that is solely limited to handling the presence of concept drift (we refer the reader to Iwashita and Papa [30] for a recent survey), we propose a new framework that is based on using the novel idea of latent variables to find an explanation of the occurring real concept drift.

1.2.1 Our Approach

We propose a modeling framework for detecting the presence of concept drift in the context of discrete Bayesian networks using latent variables. Unlike previously proposed approaches [12, 14] which are limited to naive Bayes classifiers, our framework is applicable to general Bayesian network models. We use latent variables to model two types of drifts over time:

- *Posterior Distribution Drift*, and

- *Uncertainty Drift*.

We develop a modeling technique using latent variables that is able to detect posterior distribution drift. We provide a new method for modeling and detecting concept drift via modeling uncertainty over time, i.e., the amount of belief that changes over time. In addition, we propose an explanation framework that is able to use latent variables to not only detect the presence of real concept drift but also provide an explanation of the occurring drift.

We have implemented our approaches and presented our empirical results. Our results indicate that using latent variables to develop a modeling technique to detect the existence of concept drift and an explanation framework to find interpretations of the occurring drift is an efficient mechanism. We show that our approaches are not only sensitive to changes in both real concept drift and uncertainty drift but also can quickly detect and explain the presence of drifts.

1.3 Research Tasks

The objective of this dissertation is to present our findings, which aim to address the following main research tasks:

1. Shuffling Algorithms Limitations Analysis - this research task studies the preservation of data dependencies while applying secure cryptographic shuffling algorithms. That is, we investigate whether or not current secure cryptographic shuffling algorithms preserve data dependencies in the shuffled relations. For this task, we have completed the following subtasks:

   1.1 Research publications.

   1.2 Define the shortcomings of current secure cryptographic shuffling algorithms.

   1.3 Functional dependencies preserving shuffle - this research subtask develops a solution to preserve a given set of functional dependencies after applying se-

cure cryptographic shuffling algorithms. For this research subtask, we have preformed the following:

a) Investigation of up to 3NF relations.

b) BCNF-based FD Preserving Shuffle.

c) Generic functional dependencies preserving shuffle.

1.4 Restoring original relations - this research subtask investigates the process of restoring the original relation from the shuffled relation. For this research subtask, we have preformed the following:

a) Develop an algorithm to restore shuffled BCNF relations.

b) Develop an algorithm to restore shuffled relations using the generic shuffling algorithm.

1.5 Data-driven association preservation shuffle - this research subtask develops a solution to preserve data-driven association in a given relation after applying secure cryptographic shuffling algorithms. For this research subtask, we have preformed the following:

a) Define an association graph.

b) Develop an algorithm for modeling and extraction of data-driven association based shuffle.

- Completed: 1.1, 1.2, 1.3a, 1.3b, 1.3c, 1.4a, 1.4b, 1.5a, 1.5b.
- Remaining: None
- H. Alsuwat, E. Alsuwat, T. Geng, C. Huang, and C. Farkas, "Data dependencies preserving shuffle in relational database", 2019 2nd International Conference on Data Intelligence and Security (ICDIS), June 2019, pp. 180 - 187. (citation number [4] in the references)

- T. Geng, H. Alsuwat, C-T. Huang, and C. Farkas, "Securing relational structured database with attribute association-aware shuffling". Accepted in: The 2019 IEEE Conference on Dependable and Secure Computing.

2. Modeling Concept Drift: this task studies the issue of concept drift in the context of discrete Bayesian networks. That is, we propose a probabilistic graphical model framework to explicitly detect the presence of concept drift using latent variables. For this research task, we have preformed the following subtasks:

   2.1 Research publications.

   2.2 Define the limitations of concept drift detection methods in Bayesian networks.

   2.3 Define posterior probability drift.

   2.4 Define uncertainty drift.

   2.5 Propose a framework for detecting the presence of concept drift in the context of discrete Bayesian networks using latent variables.

   2.6 Use latent variables to model

   a) Posterior probability drift, and

   b) Uncertainty drift.

   2.7 Generalization of our framework into higher dimensions.

   2.8 Implement our approaches and present our empirical results.

   - Completed: 2.1, 2.2, 2.3, 2.4, 2.5, 2.6a, 2.6b, 2.7, 2.8.
   - Remaining: None.
   - H. Alsuwat, E. Alsuwat, M. Valtorta, J. Rose, and C. Farkas, "Modeling concept drift in the context of discrete bayesian networks", Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1: KDIR, INSTICC, SciTePress, 2019, pp. 214 - 224. (citation number [5] in the references)

3. Explaining Concept Drift: this task focuses on finding an explanation of concept drift in the context of discrete Bayesian networks. That is, we propose a framework to help find explanations of the occurring posterior probability drift using latent variables. For this research task, we have preformed the following subtasks:

   3.1 Research publications.

   3.2 Propose a framework for explaining real concept drift - this research subtask aims to build a framework that is capable to explain the detected posterior probability drift across time.

   3.3 Propose a Kullback-Leibler (KL) divergence based measure - this research subtask aims to find an explanation for the occurring posterior probability drift

   3.4 Develop an algorithm for finding an explanation of posterior probability drift across time.

   3.5 Implement our approaches and present our empirical results.

   - Completed: 3.1, 3.2, 3.3, 3.4, 3.5.
   - Remaining: None.
   - H. Alsuwat, E. Alsuwat, M. Valtorta, J. Rose, and C. Farkas, "Concept Drift in the Context of Discrete Bayesian Networks: A Modeling Technique and an Explanation Framework", **To be submitted to** the ACM Transactions on Management Information Systems (TMIS).

1.4 Dissertation Outline

The remaining of this dissertation is organized as follows:

– In chapter 2, we present an overview of background information.

– In chapter 3, we present our approach for addressing the problem of shuffling algorithms and preserving functional dependencies.

– In chapter 4, we demonstrate the process of restoring the original relations after applying our secure cryptographic shuffling algorithm.

– In chapter 5, we present our approach for addressing the problem of shuffling algorithms and data-driven association discovery.

– In chapter 6, we present our approaches for modeling concept drift in the context of discrete Bayesian networks using latent variables.

– In chapter 7, we present our explanation framework for finding interpretations of the occurring real concept drift using latent variables in discrete Bayesian networks.

– Finally, in chapter 8, we conclude and discuss the future directions of our work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we provide a brief overview of background information and related work for this dissertation.

2.1 ASSOCIATION VIA STATISTICAL TESTING

Knowledge related to measuring associations between attributes in database systems is essential for crafting solutions to challenges in information management. Various techniques allow one to measure the strength of an association between random variables. Each of these techniques is dependent upon data type.

When variables of interest are categorical, the chi-squared test (χ^2 test, also known as "Pearsons chi-squared test" or the "chi-square test of independence") [15, 21] is suitable for discovering statistically-significant associations between two categorical attributes. Simply, this is a goodness of fit test in which the data are compared to a model based on the null hypothesis of independence [13]. The χ^2 test allows one to decide whether to accept or reject the null hypothesis. If the difference between the observed and the expected values is less than or equal to 0.05, then the null hypothesis is rejected [10]. Conversely, if the difference between the observed and the expected values is larger than 0.05, the null hypothesis is accepted.

When variables of interest are quantitative, the correlation coefficient is suitable for measuring the association between quantitative variables [13]. The Pearson correlation coefficient (PCC), or "Pearson's r," [54, 62] was the first formal method for measuring statistical correlation and remains one of the most widely used methods. The PCC is a linear

test used to evaluate the association between two variables and returns a value between 1 and -1. When the returned value is close to 1 or -1, the correlation between the two variables is strong. Otherwise, if the value is close to zero, the correlation is weak [36].

When variables of interest are one categorical and one quantitative, the analysis of variance (ANOVA) test [45] is suitable for discovering the association between them [64]. One-way ANOVA test is one of the most powerful and widely used methods in statistical inference [42]. It is used to measure the association between an independent variable (the categorical variable) and a dependent variable (the quantitative variable), which is broken down by the states on the independent variable. ANOVA test then examines for statistically-significant differences between the means of two or more populations of the dependent variable [64]. ANOVA test allows one to either accept or reject the null hypothesis. The null hypothesis is accepted if all means of samples of the dependent variable are equal [42]. Conversely, the null hypothesis is rejected if there is at least one sample of the population of the dependent variable which has a different mean value.

2.2 Privacy Preserving in Traditional Database

The problem of preserving privacy within the traditional database context has been extensively studied in recent times. Numerous techniques have been proposed to support data mining without compromising privacy, such as perturbation methods [66]. Perturbation methods are data masking techniques that are widely used for privacy preservation [46]. Perturbation-based methods perturb or alter individual data values or query results by swapping, condensation, adding noise, or shuffling [47, 66, 26]. Tradeoffs between information loss and privacy preservation must be weighed in when choosing and implementing methods for securing databases.

Data swapping is a transformation technique that modifies the dataset by altering dataset attribute values from selected records to reduce risk of identification or disclosure while still preserving data integrity and utility [18]. Swapping is a pre-tabular method and thus

appropriately applied to microdata; in this technique, the exchange of data values across data records with specified proximity undermines intruder confidence that the identified information is associated with the data target [70]. All original values are maintained in the dataset as only value positions are swapped [31].

Condensation or aggregation approaches work by condensing data into various groups of predetermined size to maintain a specified level of statistical information; the clusters condensed statistics can be used to generate pseudo-data from each group, thereby resulting in a synthetic data set is the same as the original from which it was created [31, 57]. Privacy preservation is achieved by using pseudo-data, which adds an additional security layer without necessitating redesign of existing data mining algorithms. However, information loss occurs in condensation as large number of records is streamlined into a single statistical cluster, thus also affecting data mining outcomes due to this observed information loss [57].

Similar to data swapping, adding noise is another value distortion approach wherein the original data values are modified by adding some random number (noise) to create artificial data values, essentially masking the original data, thus reconstructing data distribution without compromising the original values of the data [31, 57]. However, new algorithms are required to effectively mine the data as needed.

In our previous work, we introduced a novel shuffling approach to enhance the security of relational database [26]. We discussed the cases where an attack that is capable of recovering the entire database with only the individual data files in the MySQL environment. It is essentially a feature of the database management system, but malicious users can employ it to get access to the data if they can break into the servers operating system. Then, we proposed a cryptographic algorithm to shuffle the data values in each field of the database table in order to assure privacy and reduce the risk of data compromise.

As to the shuffling algorithm, there are two steps: shuffling inside a block and shuffling of the blocks. The behaviors depend on the following variables: l, w, $k1$, $k2$ where l is the total number of tuples in the relational database, and w, $k1$, and $k2$ are three random num-

bers generated according the specific algorithms. Indeed, w decides how many blocks the entire column will be divided into, and w is essentially the number of tuples in each block except for the last block. $k1$ determines the shuffling inside each block and $k2$ determines the shuffling of blocks. Both shuffling of elements inside one block and shuffling of blocks follow the rule that element at position $[i]$ would be moved to a position based on $mod\ w$.

One significant issue about shuffling is that some attributes in the tuple are highly related to each other, and shuffling them separately will break their relationship. If such highly associated attributes are shuffled individually and independently, the resulting mismatching attribute values in the same tuple can be easily detected by the attacker and expose the existence of shuffling. This arises the need for "bundle shuffling" in which a set of highly associated attributes must be bound together by assigning the same key set to them with the goal of preserving data association when using our cryptographic shuffling algorithm. If we bundle the highly associated attributes and shuffle them together, then the shuffled table becomes more deceptive for the attacker to consider it as a truly valuable source of data.

One of the objective of this dissertation is to bridge the gap between shuffling algorithms and data preservation dependencies, as current perturbation techniques fail in this regard. Therefore, necessary privacy preservation performance can be optimized by enhancing shuffling algorithms so that they successfully preserve the data dependencies after shuffling.

2.3 CONCEPT DRIFT

In this section, we will give a brief overview of concept drift, a definition of concept drift, a concept drift classification, and concept drift detection methods.

2.3.1 CONCEPT DRIFT OVERVIEW

Applications are increasingly critically dependent on concept schemes for the semantic interoperability of their data [65]. As data evolves over time, real-time data analytics are undermined as the models built to foster this learning becomes obsolete [73]. In machine learning, concept drift is a nonstationary learning problem that develops over time, often because the training data and application data mismatch in real life scenarios [43, 24]. Therefore, concept drift is associated with a greater probability for prediction inaccuracies due to misalignment driven by changes in the statistical properties of the target variable. Most real-world applications confront some form and degree of shift, which renders this topic highly relevant to the existing and emerging machine learning community [43]. Concept drift thus plays a key role in machine learning and predictive analytics optimization, as adequately accounting for this phenomenon strengthens the overall integrity, utility, and functionality of the machine learning model. Recent surveys on concept drift can be found in [30, 24]. In this dissertation, we address two types of concept drift as follows:

1. Posterior Probability Drift, and
2. Uncertainty Drift.

POSTERIOR PROBABILITY DRIFT

We take the Bayesian view that we model each parameter in a Bayesian network as a random variable, beta distributed, and we use the mean of that random variable for inference. *Posterior Probability drift* refers to changes in the mean of the posterior distribution from time point t_i to time point t_j, where $i, j \geq 0$ and $j > i$. Posterior distribution drift is caused by either change in the prior or the likelihood function or both as the posterior distribution is proportional to the product of the prior and the likelihood function (Posterior $\propto$ Prior $\times$ Likelihood function [39]).

$$P_{t_i}(A \mid B) \neq P_{t_j}(A \mid B) \tag{2.1}$$

where $P(A \mid B)$ is the conditional probability of A given B.

In the case of discrete Bayesian networks, Equation 2.1 can be written as follows:

$$(\frac{y+\alpha}{n+\alpha+\beta})_{t_i} \neq (\frac{y+\alpha}{n+\alpha+\beta})_{t_j} \tag{2.2}$$

which means that the mean of the beta distribution at time point t_i is not equal to the mean of the beta distribution at time point t_j.

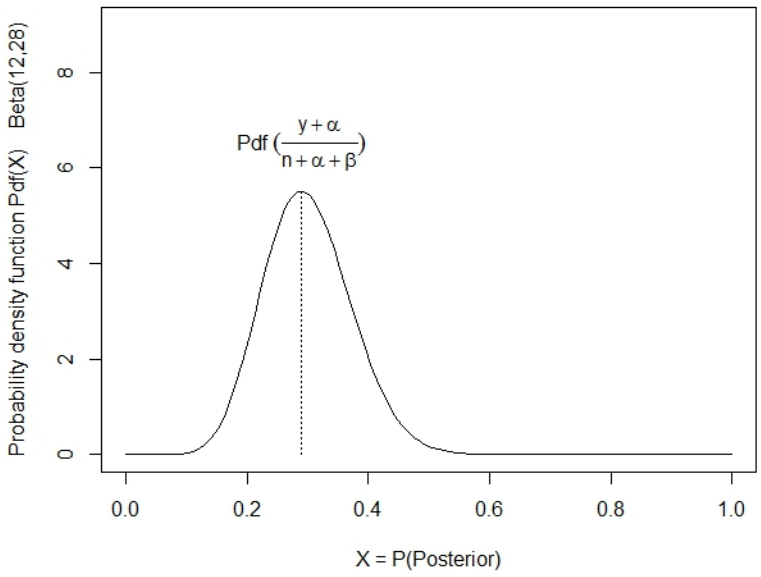
Example 2.1. In the chest clinic network, assume that there are two datasets DB_1 and DB_2. We use DB_1 and DB_2 to learn the structure of the Bayesian networks B_1 and B_2 respectively. Figure 2.1 illustrates a graphical representation of the posterior distribution drift for the posterior assignment $P(T = yes \mid A = yes)$.

Figure 2.1a shows a posterior probability distribution value ($\frac{y+\alpha}{n+\alpha+\beta}$) of the assignment $P(T = yes \mid A = yes)$ in model B_1 whereas Figure 2.1b shows the drifted posterior probability distribution value of the same assignment in model B_2. Figure 2.1b shows a case of posterior distribution drift since there is a P_{B_2} in the model, B_2, that has a different value: $P_{B_2}(T = yes \mid A = yes)$ than the P_{B_1} value of the original validated model B_1: $P_{B_1}(T = yes \mid A = yes)$.

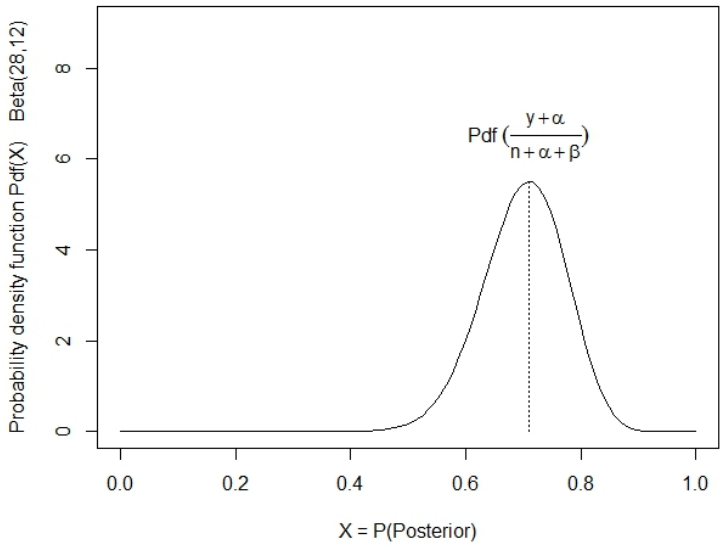
Uncertainty Drift

Entropy measures the amount of uncertainty in an input dataset [61]. In machine learning systems, when learning the structure of models, B_1 and B_2, from datasets DB_1 and DB_2 respectively, there is an expected drift in the beliefs between the two models B_{valid} and B_{new}. However, this drift should not be substantial. There is, therefore, a need to define uncertainty drift in this context.

Uncertainty drift is a variable that reflects the change in beliefs over time. This change happens when the probability density function changes from the model B_1 at time point $t = i$ to the model B_2 at time point $t = i + 1$. This kind of change is mainly caused by the change in the total number of observed cases between datasets DB_1 and the new dataset



(a) The posterior probability in B_1 is the intersection of the vertical dotted line with the X axis.



(b) The posterior probability in B_2 is the intersection of the vertical dotted line with the X axis.

Figure 2.1: A graphical representation to illustrate Posterior Distribution Drift

DB_2. Note that, in the case of nonstationary environments the dataset DB_2 is the union of the dataset DB_1 and a new incoming dataset, which we denote as DB_{new}

$$Pdf_{B_1}(P(A \mid B)) \neq Pdf_{B_1}(P(A \mid B)) \tag{2.3}$$

where $P(A \mid B)$ is the posterior probability.

In the case of discrete Bayesian networks, having a prior that is conjugate for the likelihood function will make it mathematically convenient to calculate the posterior distribution since the posterior distribution will be from the same family of distribution as the prior [56].

For instance, multiplying a beta distributed prior, $Beta(\alpha, \beta)$, with a binomial distributed likelihood function, $Binomial(n, \theta)$, yields a beta distributed posterior distribution, $Beta(y + \alpha, n - y + \beta)$, where n is the total number of cases, and y is the count of successes [3]. Thus, we can write Equation 2.3 as follows:

$$Pdf_{B_1}(\frac{y+\alpha}{n+\alpha+\beta}) \neq Pdf_{B_2}(\frac{y+\alpha}{n+\alpha+\beta}). \tag{2.4}$$

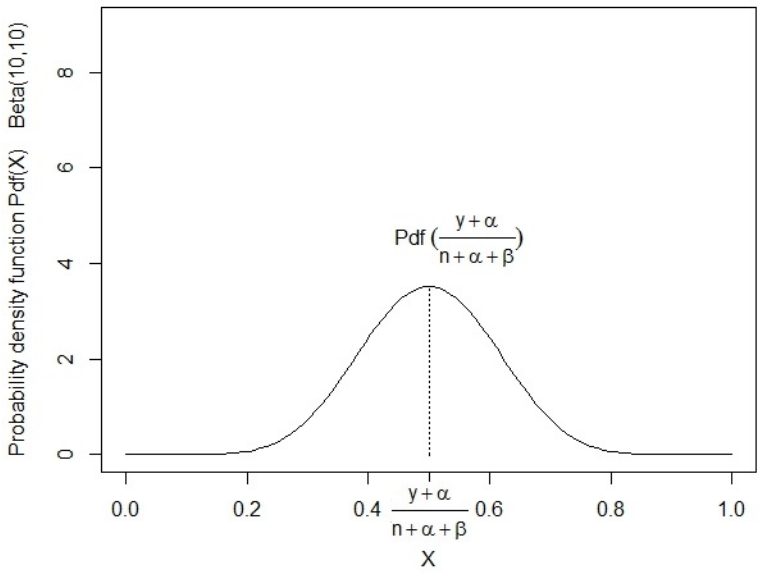
where $\frac{y+\alpha}{n+\alpha+\beta}$ is the mean of the beta distribution.

Example 2.2. In the chest clinic network[35], a graphical representation of the uncertainty drift for the posterior assignment $P(T = yes \mid A = yes)$ is illustrated in Figure 2.2. Assume that there are two datasets DB_1 and DB_2 that are used to learn the structure of the belief networks B_1 and B_2 respectively.

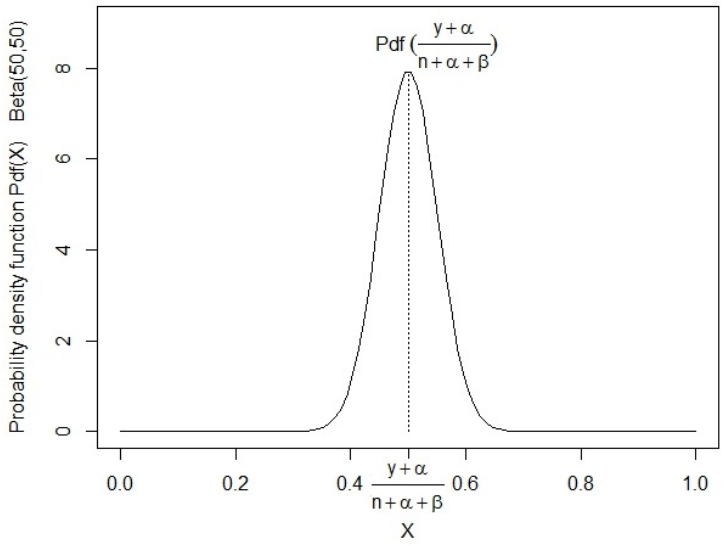
Figure 2.2a shows a Pdf of the assignment $P(T = yes \mid A = yes)$ in model B_1 whereas Figure 2.2b shows the drifted Pdf of the same assignment in model B_2. Figure 2.2b shows a case of uncertainty drift since there is a Pdf in the model, B_{new}, that has a different value: $pdf_{B_2}(P(T = yes \mid A = yes))$ than the Pdf value of the original model B_1: $pdf_{B_1}(P(T = yes \mid A = yes))$.

2.3.2 Definition of Concept Drift

In the real-world, change is constant. The assumption that both training and testing data follow the same distributions is typically violated in real-world applications because testing



(a) The vertical dotted line represents the amount of uncertainty in B_{valid}.



(b) The vertical dotted line represents the amount of uncertainty in B_{new}.

Figure 2.2: A graphical representation to illustrate Uncertainty Drift

(unseen) data commonly experience a phenomenon that results in a change to the distribution of a single feature, a combination of features, or class boundaries [43]. Traditional machine learning algorithms make predictions of future data by leveraging trained statistical models constructed using previously collected labeled/unlabeled training data [52]. Machine learning strategies generally employ static models that use historical data [68]. In practical application, data, context, concepts, and relationships can change, leading to concept drift, which can erode predictive performance as these models assume a static relationship between input and output variables [43, 65].

Concept drift “is the situation in which the statistical properties of the target concept change over time" [65]. In machine learning, concept drift specifically refers to unexpected changes to underlying data distribution over time within the unknown and hidden relationship between input and output variables [73, 25]. There are two primary kinds of concept drift: (1) sudden (abrupt, instantaneous), and (2) gradual [63]. Recurring or cyclical change can also occur [72].

To understand concept drift, it is also necessary to define a concept. According to [65], “a concept refers to different objects at different points in time". Concept intention, extension, and labeling can all change or shift over time, which influences the meaning of the target concept (concept drift). Concept shift is the hardest challenge among different types of dataset shifts, as there are multiple factors that can contribute to this, including (1) a changing context that triggers changes in target concepts, (2) changes in user behaviors and/or tasks, (3) changes to class definitions, (4) changes between training and test phases [43, 52]. Furthermore, the target concept may depend on some hidden context, which makes it difficult to explicitly integrated in the form of predictive features (i.e., weather prediction and seasonal changes; customer buying preferences shift according to time of day/season, income/inflation levels; etc.) [69]. Changes occurring within this hidden context can lead to more or less radical shifts in the target concept. It can be difficult to differentiate between true concept drift and noise, as algorithm robustness affects noise

responsivity [69].

Simply stated, concept drift occurs when a concept has different meanings at different times; the greater the gap between these two meanings, the more unstable the concept becomes [65]. Concept drift occurs when the target concept is so unstable that part or all of its meaning is better represented by a different concept.

In contemporary scientific literature, concept drift may also be referred to as concept shift, dataset shift, data fracture, covariate shift, and other terms common throughout the literature [43]. The lack of a standardized term complicates scientific inquiry and synthesis, as there are important differences in these related terminologies. In differentiating between dataset shift types and definitions, concept drift occurs when different data distributions are associated with changes in class definitions or the target "concept" to be learned [43].

2.3.3 Concept Drift Classification

In contemporary scientific literature, several researchers have proposed to characterize types of concept drift [67, 24, 30]. Webb et al. [67] categorized types of concept drift based on (i) Drift subject, which indicates what aspects of the joint probability drifts over a period of time, (ii) Drift frequency, which shows how often concept drift happens during a particular time, (iii) Drift transition, which indicates the means wherein the process of changing from one concept to another occurs, (iv) Drift reoccurrence, which shows whether or not the occurring concept drift has previously appeared, and (v) Drift magnitude, which points out the degree of drift between two time points.

Drift subject is mathematically defined as a change in the joint probability between two time points t_0 and t_1 as follows: $P_{t_0}(X, y) \neq P_{t_1}(X, y)$, where X is the input variables and y is the target variable [24]. Drift subject is divided into two types [24]: (1) real concept drift, and (2) virtual concept drift. Real concept drift occurs when the conditional probability changes on the target variable y whereas the input variables X remain unchanged, i.e., the posterior probability changes between two time points t_0 and t_1 as

follows: $P_{t_0}(y \mid X) \neq P_{t_1}(y \mid X)$. Virtual concept drift occurs when the prior distribution changes between two time points t_0 and t_1 while the posterior probability remains unchanged [63, 69], i.e., $P_{t_0}(X) \neq P_{t_1}(X)$. Real concept drift is the most important aspect in the category of drift subject since changes in real concept drift will degrade the accuracy of the machine learning model and thus require an update of the model [32]. Therefore, the discussion of this dissertation is mainly related to the notion of real concept drift which we refer to as concept drift.

2.3.4 Concept Drift Detection

One of the challenging tasks in the context of concept drift is to rapidly detect concept drift and provide a practical measure of drift magnitude. A variety of concept drift detection methods have been recently developed. Gama et al. [24] categorized such methods into four general groups as follows: (1) methods based on sequential analysis (members of this group include the Cumulative Sum (CUSUM) and the Page-Hinkley (PH) [51]), (2) methods based on statistical process control (members of this group include the Drift Detection Method (DDM) [23], the Early Drift Detection Method (EDDM) [7], and the Exponentially Weighted Moving Average (EWMA) [58]), (3) methods based on contextual approaches (a member of this group includes the Splice system [28]), and (4) methods based on Monitoring distributions on two different time-windows (members of this group include the Adaptive sliding Window (ADWIN) [11], the Adaptive Cumulative Windows Model (ACWM) [60], and SEED Drift Detector (SEED) [29]).

The contribution of this work belongs to the last one of the four groups. Methods based on monitoring distributions on two different time-windows are techniques that use statistical tests to compare the distributions of a fixed reference window on the previous data and a sliding window on the most recent data [24]. Kifer et al. were first to propose comparing two detection window distributions in relation to data streams [33]. The team's presented algorithms assessed samples taken from two probability distributions to identify

key differences in the distributions. Another example of such methods, proposed in [22], is the VFDTc system, which is an algorithm for mining in nonstationary environments with the ability to detect and adapt to concept drift. The VFDTc system is used in concept drift resolution through ongoing monitoring of observed differences between two class-distributions, including evaluation of: 1) class-distribution when a node was a leaf, and 2) weighted sum of class-distributions in the nodes leaf-descendants [22].

Other more recent concept drift detection methods based on monitoring distributions on two different time-windows were proposed in [12] and [14].

In this dissertation, we study concept drift detection via comparing distributions on two different time-windows. We aim to use latent variables to model and detect concept drift in the context of discrete Bayesian networks. Borchani et al. proposed a modeling technique with conditional linear Gaussian (CLG) that used latent variables to detect concept drift [12]. Their model is applicable to continuous Bayesian networks and was applied to continuous domains. Cabanas et al. proposed a method for detecting concept drift in discrete streaming data [14]. Their proposed preprocessing algorithm transferred discrete data into continuous data before applying Borchani at el. model to detect concept drift. However, Cabanas et al.'s technique is susceptible to data loss and results in increased processing overhead when used in incremental learning domains.

CHAPTER 3

SHUFFLING ALGORITHMS AND PRESERVATION OF FUNCTIONAL DEPENDENCIES

In this chapter, we investigate the problem of preserving the set of functional dependencies of normalized relations after applying our secure cryptographic shuffling algorithm. Our goal is to guarantee that if a relation instance, r, satisfies set $\mathcal{F}$ of functional dependencies, then the shuffled relation instance $r^{'}$ also satisfies $\mathcal{F}$.

Definition 3.1. Let $R(A_1, ..., A_n)$ represent a relation schema, where R is the name of the relation and A_1, ..., A_n is the set of attribute names. An instance relation of R is denoted as r.

3.1 INVESTIGATION OF UP TO 3NF RELATIONS

Let r be a relation in 3NF with respect to a set $\mathcal{F}$ of functional dependencies. Shuffling r using our secure cryptographic shuffling algorithm results in a new relation $r^{'}$, such that $r^{'}$ may not satisfy $\mathcal{F}$.

As illustrated in section 3.2.3, we observe that relations in the third normal form may not satisfy the set of functional dependencies after applying secure cryptographic shuffling algorithms. Note that the results of using our secure cryptographic shuffling algorithm on 1NF and 2NF relations would follow with similar results. Thus, applying our secure cryptographic shuffling algorithm on relations up to 3NF may pose a security risk as attackers may be able to discover that the relations are shuffled.

To preserve the set of functional dependencies of a 3NF relation, we need to define the set of attributes for “bundle shuffling." That is, we need to define the set of attributes that must be bound together when using our shuffling algorithm and thereby assign the same key set to these attributes. As a result of bundle shuffling, we guarantee that the resulted shuffled relation $r^{'}$ does not violate the set of functional dependencies.

Theorem 3.2. *Given a relation r with schema $R(A_1, \ldots, A_n)$ in 3NF and set $\mathcal{F} = \{F_1, \ldots, F_k\}$ of functional dependencies satisfied by r. We say that the set $\mathcal{F}$ will be satisfied by $r^{'}$, where $r^{'}$ is the relation after shuffling, if we shuffle together all attributes $\{A_i\}^+$, where $\{A_i\}^+$ is the closure of attributes A_i under the set $\mathcal{F}$.*

Proof Sketch. The proof is trivial because for any two tuples t_1 and t_2 in r, if $t_1[A_i] = t_2[A_i]$, then $F = \{A_i \rightarrow A_j\}$ requires that $t_1[A_j] = t_2[A_j]$. But because we bundle A_i and A_j together while applying our secure cryptographic shuffling algorithm, then it must always be satisfied for the shuffled relation if it was satisfied for the original relation. □

Bundling attributes together will reduce the confusion we can introduce in our system. In the next section, we investigate options that will reduce the need for attribute bundling.

3.2 Experimental Results of Investigation of up to 3NF Relations

In this section, we present our experimental results of applying our secure cryptographic shuffling algorithm to 1NF, 2NF, and 3NF relations. We illustrate how the experiments carried out by our secure cryptographic shuffling algorithm preserve or violate the set of functional dependencies. Our experimental results is shown as follows:

3.2.1 Investigation of 1NF Relations

Let R be a Student relation in the 1NF as shown in Table 3.1 with the following set of functional dependencies:

1. {StdID, CourseNumber } → StdName,

2. {StdID, CourseNumber } $\rightarrow$ CourseName,

3. CourseNumber $\rightarrow$ CourseName, and

4. StdID $\rightarrow$ StdName

We test whether or not shuffling R using our secure cryptographic shuffling algorithm will preserve the set of functional dependencies.

Table 3.1: Student Relation.

StdID	CourseNumber	StdName	CourseName
1111	CSCE551	James	Math
2222	CSCE522	David	Database
3333	CSCE123	Thomas	Physics
1111	CSCE522	James	Database

Table 3.2: Student Relation after using secure cryptographic shuffling algorithm.

StdID	CourseNumber	StdName	CourseName
1111	CSCE551	David	Database
2222	CSCE522	James	Math
3333	CSCE123	James	Database
1111	CSCE522	Thomas	Physics

Table 3.2 is the result of using our secure cryptographic shuffling algorithm on the student relation in Table 3.1. We observe that the set of functional dependencies are not preserved after applying our secure cryptographic shuffling algorithm as shown in Table 3.2. Therefore, we have presented a counter example to show that relations in the 1NF does not preserve the set of functional dependencies after using our secure cryptographic shuffling algorithm.

3.2.2 Investigation of 2NF Relations

Let R be an Employee relation in the second normal form as show in Table 3.3, with the following set of functional dependencies:

1. SSN $\rightarrow$ EmpName,

2. SSN $\rightarrow$ Rank,

3. SSN $\rightarrow$ Gender,

4. SSN $\rightarrow$ Salary, and

5. Rank $\rightarrow$ Salary.

We test whether or not shuffling R using our secure cryptographic shuffling algorithm will preserve the set of functional dependencies.

Table 3.3: The original relation before secure cryptographic shuffling.

SSN	EmpName	Rank	Salary	Gender
222-22-2222	James	Secretary	35,000	Male
333-33-3333	Alex	Chair	150,000	Male
444-44-4444	Sarah	Professor	130,000	Female
555-55-5555	Kevin	Secretary	35,000	Male
666-66-6666	Mark	Professor	130,000	Male

Table 3.4: The relation after secure cryptographic shuffling.

SSN	EmpName	Rank	Salary	Gender
333-33-3333	Kevin	Professor	130,000	Female
555-55-5555	James	Chair	35,000	Male
666-66-6666	Sarah	Secretary	150,000	Male
222-22-2222	Mark	Secretary	130,000	Male
444-44-4444	Alex	Professor	35,000	Male

Table 3.4 is the result of using our secure cryptographic shuffling algorithm on the student relation in Table 3.3. We observe that the functional dependencies are not preserved after applying our secure cryptographic shuffling algorithm as shown in Table 3.4. Therefore, we have presented a counter example to show that relations in the 2NF does not preserve the set of functional dependencies after using our secure cryptographic shuffling algorithm.

3.2.3 INVESTIGATION OF 3NF RELATIONS

The Student Enrollment relation, r_1, is shown in Table 3.5. The relation has three attributes: Student ID (StdID), Subject (course info), and Professor, teaching the course. The relation r_1 is in 3NF with respect to the following set of functional dependencies:

F_1**:** {StdID, Subject} $\rightarrow$ Professor

F_2**:** Professor $\rightarrow$ Subject

Table 3.5: Original Student Enrollment Relation

StdID	Subject	Professor
1111-11-1111	DBMS	Prof1
1111-11-1111	Math	Prof2
2222-22-2222	Physics	prof3
4444-44-4444	DBMS	Prof4
5555-55-5555	DBMS	Prof1

Table 3.6: Shuffled Student Enrollment Relation

StdID	Subject	Professor
1111-11-1111	Math	Prof1
1111-11-1111	DBMS	Prof2
2222-22-2222	DBMS	prof3
4444-44-4444	Physics	Prof4
5555-55-5555	DBMS	Prof1

The result of the shuffling r_1 using our secure cryptographic shuffling algorithm is shown in Table 3.6. We observe that the set of functional dependencies, $\mathcal{F}$, of the original relation r_1 do not hold for the shuffled relation $r_1^{'}$. That is, the functional dependency F_2 is violated.

3.3 BCNF-BASED FUNCTIONAL DEPENDENCIES PRESERVING SHUFFLE

Let a relation schema R and its instance r, where R in BCNF with respect to a set $\mathcal{F}$ of FDs be given. We introduce Algorithm 1, which shuffles an instance r of R using the secure cryptographic shuffling algorithm such that the functional dependencies are preserved.

Algorithm 1: BCNF-based FD preserving shuffle

Input : Relation instance r with schema $R(A_1, \ldots, A_n)$ such that R in BCNF with respect to a set of functional dependencies $\mathcal{F} = \{F_1, \ldots, F_k\}$, where $\mathcal{F}$ is a canonical cover.

Output: Relation instance $r^{'}$, which is the relation instance r after shuffling the data values of each attribute $A_1, \ldots, A_n$

```
Procedure BCNF-based solution(r)
    Given F, let K = {K_1, ..., K_l} be the set of all candidate keys;
    For each K_i bundle K_i's attributes together (Bundle composite keys together);
    r' = Shuffle r using the secure cryptographic shuffling algorithm;
    Return r';
end
```

In Algorithm 1, we apply the secure cryptographic shuffling algorithm on a given relation r which is in BCNF with $\mathcal{F}$ set of FDs, resulting in a shuffled relation $r^{'}$ that satisfies $\mathcal{F}$.

Theorem 3.3. *Given a relation r (with schema R) in BCNF and $\mathcal{F}$ set of FDs satisfied by r, then Algorithm 1 will generate $r^{'}$ such that $r^{'}$ is shuffled and $r^{'}$ preserves $\mathcal{F}$.*

Proof Sketch. Consider relation instance r over schema $R(A_1, \ldots, A_n)$, the set of functional dependencies $\mathcal{F} = \{F_1, \ldots, F_k\}$ that hold on r, and R is in BCNF. Assume by contradiction that Algorithm 1 did not preserve the functional dependencies in $\mathcal{F}$. But then, there must exist an $F_i : X_i \rightarrow Y_i$ ($Y_i = R - X_i$) such that $r^{'}$ (the output of Algorithm 1) contains two tuples t_1 and t_2 such that $t_1[X_i] = t_2[X_i]$ and $t_1[Y_i] \neq t_2[Y_i]$. Since $F_i : X_i \rightarrow Y_i$ and X_i is a key for r, there can only be one tuple in the original relation r with values $t_1[X_i] = t_2[X_i]$. But then the shuffling algorithm could not create two different tuples t_1 & t_2 such that $t_1[X_i] = t_2[X_i]$. This contradicts our initial assumption that there

are two tuples in $r^{'}$ with same values for attributes X_i but different values for attributes Y_i. □

We showed that it is safe to apply the secure cryptographic shuffling algorithm in this case with no need to bundle shuffle a LHS attributes with the corresponding RHS attributes since the LHS of the functional dependencies in BCNF relations is a key, for which there is a unique RHS value. As such, the risk of violating the set of functional dependencies is eliminated.

3.4 Experimental Results of BCNF-based Functional Dependencies Preserving Shuffle

Given a relation instance r with schema R such that R in BCNF with respect to a set $\mathcal{F}$ of FDs, we used Algorithm 1 to evaluate the relationship between use of the secure cryptographic shuffling algorithm on r and its impact on $\mathcal{F}$ (as described in section 3.3). Our aim is to use Algorithm 1 to apply the cryptographic shuffling algorithm to a given relation instance r and observe its performance.

Table 3.7: The original Employee relation, r_2

SSN	EmpName	Salary
222-22-2222	James	65,000
333-33-3333	Alex	85,000
444-44-4444	Sarah	75,000
555-55-5555	Kevin	65,000
666-66-6666	Mark	75,000

Let the Employee relation, r_2, in BCNF as shown in Table 3.7 be given, with the following set of FDs that hold for r_2:

1. SSN $\rightarrow$ EmpName, and
2. SSN $\rightarrow$ Salary.

Table 3.8: The shuffled Employee relation, $r_2^{'}$

SSN	EmpName	Salary
333-33-3333	Kevin	75,000
222-22-2222	Mark	65,000
666-66-6666	Sarah	85,000
555-55-5555	James	75,000
444-44-4444	Alex	65,000

We implemented Algorithm 1. The performance of our implementation was comparable to the performance of shuffling the original relation using the secure cryptographic shuffling algorithm. Tables 3.7 and 3.8 show the original relation, r_2, and the shuffled relation, $r_2^{'}$, respectively. Note that both tables satisfy FDs 1 and 2. We observe that the resulting relation from Algorithm 1, $r_2^{'}$, appears useful because there is no way for attackers to detect that this relation is shuffled and not original. Therefore, we can conclude from this experiment that it is safe to use Algorithm 1 to apply the secure cryptographic shuffling algorithm on BCNF relations.

3.5 Generic Functional Dependency Preserving Shuffle

In this section, we present a generalization of the case of section 3.3 where we can shuffle a given relation that is not in BCNF such that set of FDs is preserved. We introduce Algorithm 2, a generic FD preserving shuffle.

Definition 3.4. Given a relation instance r with schema $R(A_1, \ldots, A_n)$ and a set of FDs $\mathcal{F} = \{F_1, \ldots, F_l\}$, we say that the BCNF decomposition of R into $R_1, \ldots, R_k$ is dependency preserving if $\{\mathcal{F}_1 \cup \cdots \cup \mathcal{F}_k\} \equiv \mathcal{F}$, where $\mathcal{F}_i (i = 1, \ldots, k)$ is the projection of $\mathcal{F}$ on R_i.

Definition 3.5. Let F be a set of functional dependencies on a relation R, a decomposition of R into $\{R_1, \ldots, R_n\}$ has the property of a lossless join decomposition with respect to F if and only if the natural join of the decomposed relations $R_1, \ldots, R_n$ produces the relation

Algorithm 2: Generic FD Preserving Shuffle

Input : Relation instance r with schema $R(A_1, \ldots, A_n)$, a canonical cover of a set of FDs $\mathcal{F} = \{F_1, \ldots, F_l\}$ that hold for r

Output: Relation instance $r^{'}$, which is the relation instance r after shuffling the data values of each attribute $A_1, \ldots, A_n$

```
Procedure Generic FD Preserving solution(r, F)
    Decomposition of r into r_1, ..., r_k such that r_i (i = 1, ..., k) is in BCNF form;
    Let F' be the union of the FDs projected on r_is ;
    if F' ≡ F i.e., F' and F are logically equivalent then
        for 1 ≤ i ≤ k do
            r'_i = Algorithm 1 (r_i);
        end
        r' = r'_1 ⋈ ··· ⋈ r'_k;
        Return r' ;
    else
        Return "FDs may be violated when shuffled" ;
        Exit ;
    end
end
```

R with no spurious tuples [19]. For instance, a decomposition of R into R_1 and R_2 is lossless join if and only if at least one of the following dependencies is in F^+:

1. $R_1 \cap R_2 \rightarrow R_1$
2. $R_1 \cap R_2 \rightarrow R_2$

where F^+ the closure of the set F of given functional dependencies.

Theorem 3.6. *Given a relation r and $\mathcal{F}$ set of FDs satisfied by r, Algorithm 2 preserves FDs if the BCNF decomposition is FD preserving.*

Proof Sketch. Consider relation instance r over schema $R(A_1, \ldots, A_n)$, the set of functional dependencies $\mathcal{F} = \{F_1, \ldots, F_l\}$ that hold on r. Let $R_1, \ldots, R_k$ be a dependency-preserving BCNF decomposition of r and $\mathcal{F}^{'}$ the union of the projection of $\mathcal{F}$ on R_is. Assume by contradiction that Algorithm 2 did not preserve a functional dependency F. Because the decomposition was dependency preserving ($\mathcal{F}^{'} \equiv \mathcal{F}$), F must have been re-

moved as the result of the shuffling. We know that from Theorem 3.3 that all $\mathcal{F}_i$'s are preserved in $r^{'}$. But then F must have been lost due to the natural join. When doing the natural join, ${r_1}^{'} \bowtie \cdots \bowtie {r_i}^{'}$, the only functional dependency that may not be preserved are the ones that bridge two relations say from a subset of attributes of $r_i^{'}$ (denoted as Attr($r_i^{'}$)) to a subset of attributes of $r_j^{'}$, (denoted as Attr($r_j^{'}$)). That is, F: Attr($r_i^{'}$) $\rightarrow$ Attr($r_j^{'}$). Since the decomposition was dependency preserving, this can only happen if Attr($r_i^{'}$) is a key for r. But then, there must be only one tuple in r_i (and also in r) for each unique value combination for Attr($r_i^{'}$). This contradicts our original assumption. □

The computational complexity is dominated by the complexity of the shuffling algorithm (presented in [26]) and the association-rule finding algorithm (presented in [13] and [36]). Our preprocessing, i.e., bundling the associated attributes and BCNF decomposition, imposes little performance cost. The bundling is linear on the number of attributes. The worst-case complexity of the BCNF decomposition algorithms is exponential on the number of attributes [41, 50]. Koehler [34] presented a method that, while it has exponential complexity as the worst-case, in most cases is efficient in practice.

3.6 Experimental Results of Generic functional dependency Preserving Shuffle

We evaluated the effectiveness of our generic functional dependency preserving shuffle (as described in section 3.5). Our plan is to use Algorithm 2 to apply the secure cryptographic shuffling algorithm to a given relation instance r that is not in BCNF with set $\mathcal{F}$ of functional dependencies such that the resulting shuffled relation instance $r^{'}$ satisfies $\mathcal{F}$.

Given the Student relation instance, r, as shown in Table 3.9 with the following set $\mathcal{F}$ of functional dependencies that holds for r:

1. SId$\rightarrow$ SName,

2. SId$\rightarrow$ SGPA, and

3. DId $\rightarrow$ Dname.

We tested whether or not shuffling r using Algorithm 2 – resulting in $r^{'}$ – would preserve the set of functional dependencies for $r^{'}$.

Table 3.9: The original Student relation, r

SId	SName	SGPA	DId	DName
111	Alex	3.5	D1	Computers
111	Alex	3.5	D3	Math
222	Sarah	4.00	D4	Science
333	Kevin	3.1	D6	Stat
444	Mark	2.3	D1	Computer

Table 3.10: Decomposing Table 3.9 into sub-relations r_1, r_2, and r_3

(a) r_1

SId	SName	SGPA
111	Alex	3.5
222	Sarah	4.00
333	Kevin	3.1
444	Mark	2.3

(b) r_2

DId	DName
D1	Computers
D3	Math
D4	Science
D6	Stat

(c) r_3

SId	DId
111	D1
111	D3
222	D4
333	D6
444	D1

Table 3.11: Shuffling r_1, r_2, and r_3 in Table3.10 into $r_1^{'}$, $r_2^{'}$, and $r_3^{'}$, respectively

(a) $r_1{}'$

SId	SName	SGPA
111	Mark	4.00
222	Kevin	3.5
444	Alex	3.1
333	Sarah	2.3

(b) $r_2{}'$

DId	DName
D4	Math
D6	Computers
D1	Science
D3	Stat

(c) $r_3{}'$

SId	DId
222	D4
111	D3
444	D1
111	D1
333	D6

Table 3.12: The shuffled Student relation, r'

SId	SName	SGPA	DId	DName
222	Kevin	3.5	D4	Math
111	Mark	4.00	D3	Stat
444	Alex	3.1	D1	Science
111	Mark	4.00	D1	Science
333	Sarah	2.3	D6	Computer

Assume that r shown in Table 3.9 is decomposed into sub-relations r_1, r_2, and r_3 shown in Tables 3.10a, 3.10b, and 3.10c, respectively. Our generic functional dependency preserving shuffle algorithm, Algorithm 2, used Algorithm 1 to shuffle sub-relations r_1, r_2,

and r_3 resulting in shuffled sub-relations $r_1^{'}$ (Tables 3.11a), $r_2^{'}$ (Tables 3.11b), and $r_3^{'}$ (Tables 3.11c), respectively. The outcome of Algorithm 2, $r^{'}$ (Table 3.12), was obtained by natural joining of the shuffled sub-relations $r_1^{'}$, $r_2^{'}$, and $r_3^{'}$.

We observe that the set $\mathcal{F}$ of functional dependencies of the original relation r (Table 3.9) is preserved in the shuffled relation $r^{'}$ (Table 3.12) after using Algorithm 2 in order to apply the secure cryptographic shuffling algorithm. We also observe the shuffled relation $r^{'}$ becomes more useful in deceiving attackers as it looks to be a true and valuable source of data.

CHAPTER 4

RESTORING ORIGINAL RELATIONS

In this chapter, we demonstrate the process of restoring the original relations after applying our secure cryptographic shuffling algorithm.

4.1 RESTORING SHUFFLED BCNF RELATIONS

Given the outcome of Algorithm 1, a shuffled relation instance $r^{'}$ with the schema $R(A_1, \ldots, A_n)$ such that R in BCNF with respect to a set of functional dependencies $\mathcal{F}$, we introduce Algorithm 3, which unshuffle $r^{'}$ to restore the original relation instance r.

Algorithm 3: Restoring Shuffled BCNF Relations

Input : Relation instance $r^{'}$ with schema $R(A_1, \ldots, A_n)$ such that R in BCNF with respect to a set of functional dependencies $\mathcal{F} = \{F_1, \ldots, F_k\}$, where $\mathcal{F}$ is a canonical cover.

Output: Relation instance $\bar{r}$, which is the relation instance $r^{'}$ after unshuffling the data values of each attribute $A_1, \ldots, A_n$ such that $\bar{r} = r$ and $\bar{r}$ satisfies $\mathcal{F}$.

1 **Procedure** `Restoring BCNF Relation(`$r^{'}$`)`
2 Given $\mathcal{F}$, let $K = \{K_1, \ldots, K_l\}$ be the set of all candidate keys;
3 Bundle K_i $(i = 1, \ldots, l)$ together (Bundle composite keys together to prevent key violation);
4 $\bar{r}$ = Unshuffle $r^{'}$ using our restore algorithm (presented in [26]);
5 Return $\bar{r}$;
6 **end**

In Algorithm 3, we apply our restore algorithm on a given relation $r^{'}$ which is in BCNF with $\mathcal{F}$ set of functional dependencies, resulting in a relation $\bar{r}$ that is equivalent to r and satisfies $\mathcal{F}$.

Theorem 4.1. *Given a relation r' (with schema R) in BCNF and F set of functional dependencies satisfied by r', then Algorithm 3 will generate $\bar{r}$ such that $\bar{r}$ is equivalent to the original relation r and $\bar{r}$ satisfies $\mathcal{F}$.*

Sketch. Trivially follows [see Theorem 3.3]. □

4.2 Experimental Results of Restoring Shuffled BCNF Relations

In this section, we evaluated the effectiveness of our restoring shuffled BCNF relations (as described in section 4.1). Our goal is use Algorithm 3 to restore the original relation r. Given the output of Algorithm 1, a relation instance r' with schema R in BCNF, as shown in table 4.2, with the following set $\mathcal{F}$ of functional dependencies that holds for r':

1. SSN $\rightarrow$ EmpName, and
2. SSN $\rightarrow$ Salary.

Table 4.1: The original Employee relation, r

SSN	EmpName	Salary
222-22-2222	James	35,000
333-33-3333	Alex	150,000
444-44-4444	Sarah	130,000
555-55-5555	Kevin	35,000
666-66-6666	Mark	130,000

Table 4.2: The output of Algorithm 1, r'

SSN	EmpName	Salary
333-33-3333	Kevin	130,000
555-55-5555	James	35,000
666-66-6666	Sarah	150,000
222-22-2222	Mark	130,000
444-44-4444	Alex	35,000

We tested whether or not restoring $r^{'}$ using Algorithm 3 – resulting in $\bar{r}$ such that $\bar{r}$ is equivalent to the original relation r, as shown in table 4.1, and $\bar{r}$ satisfies $\mathcal{F}$.

Table 4.3: The output of Algorithm 3, $\bar{r}$

SSN	EmpName	Salary
222-22-2222	James	35,000
333-33-3333	Alex	150,000
444-44-4444	Sarah	130,000
555-55-5555	Kevin	35,000
666-66-6666	Mark	130,000

The result of restoring $r^{'}$ using Algorithm 3 is shown in Table 4.3. We observe that Algorithm 3 succeeded in restoring the original relation r such that $\bar{r}$ is equivalent to the original relation r and the set $\mathcal{F}$ of functional dependencies $\bar{r}$ hold for r.

4.3 Restoring Shuffled Relations Using Generic Functional Dependencies Preserving Shuffle

Given the outcome of Algorithm 2, a shuffled relation instance $r^{'}$ with the schema $R(A_1, \ldots, A_n)$ such that R is not in BCNF with respect to a set of functional dependencies $\mathcal{F}$, we introduce Algorithm 4, which unshuffle $r^{'}$ to restore the original relation instance r.

In Algorithm 4, we apply our restore algorithm on a given relation instance $r^{'}$ which is not in BCNF with $\mathcal{F}^{'}$ set of functional dependencies, resulting in a relation $\bar{r}$ with $\bar{\mathcal{F}}$ set of functional dependencies such that $\bar{r}$ is equivalent to the original relation instance r and $\bar{\mathcal{F}}$ $\equiv \mathcal{F}^{'}$ and thereby $\bar{\mathcal{F}} \equiv \mathcal{F}$.

Theorem 4.2. *Given a shuffled relation $r^{'}$, $\mathcal{F}^{'}$ set of functional dependencies satisfied by $r^{'}$, and the history of decomposing the original relation r into $r^{'}$, Algorithm 4 reconstruct the relation $\bar{r}$ such that $\bar{r}$ is equivalent to the original relation r and $\bar{r}$ satisfies $\mathcal{F}$.*

Proof Sketch. Consider relation instance $r^{'}$ over schema $R(A_1, \ldots, A_n)$, and the set of functional dependencies $\mathcal{F}^{'} = \{F_1^{'}, \ldots, F_j^{'}\}$ that hold for $r^{'}$, and the history of decompos-

Algorithm 4: Restoring Shuffled Generic Relations

Input :

1. Relation instance $r^{'}$ with schema $R(A_1, \ldots, A_n)$,
2. Set of functional dependencies $\mathcal{F}^{'}$ = $\{F_1^{'}, \ldots, F_j^{'}\}$ that hold for $r^{'}$,
3. History of BCNF decomposition of R into $R_1, \ldots, R_j$ that was used for the shuffling with projection of $\mathcal{F}^{'}$ on R_i $(i = 1, \ldots, j)$ s.t. $\{\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_j\} \equiv \mathcal{F}^{'}$.

Output: Relation instance $\bar{r}$, which is the relation instance $r^{'}$ after unshuffling the data values of each attribute $A_1, \ldots, A_n$ and set of functional dependencies $\bar{\mathcal{F}}$ that hold for $\bar{r}$, where $\bar{\mathcal{F}} \equiv \mathcal{F}^{'}$.

```
Procedure Restoring Generic Relations(r, F)
    for 1 ≤ i ≤ j do
        r̄_i = Algorithm 3 (r'_i);
        F̄_i = F'_i (Proved in Theorem 3.3) ;
    end
    r̄ = r̄_1 ⋈ ··· ⋈ r̄_j;
    F̄ = F̄_1 ∪ ··· ∪ F̄_j;
    Return r̄, F̄;
end
```

ing the original relation r into $r^{'}$, where $r_1^{'}, \ldots, r_j^{'}$ is the dependency-preserving BCNF decomposition of $r^{'}$.

Assume by contradiction that Algorithm 4 did not reconstruct the relation $\bar{r}$ such that $\bar{r}$ is equivalent to the original relation r and $\bar{r}$ satisfies $\mathcal{F}$. But then, there must exist be spurious tuples in $\bar{r}$ that are not in $r^{'}$. Since the decomposition is in BCNF and dependency preserving, it has the property of a lossless join decomposition. But then Algorithm 4 will not generate spurious tuples in $\bar{r}$.

This contradicts our initial assumption that there are spurious tuples in $\bar{r}$ that are not in $r^{'}$. □

4.4 Experimental Results of Restoring Shuffled Relations Using Generic Functional Dependencies Preserving Shuffle

In this section, we evaluated the effectiveness of our restoring generic relations(as described in section 4.3). Our goal is use Algorithm 4 to restore the original relation r.

Table 4.4: The original Student relation, r

SId	SName	SGPA	DId	DName
111	Alex	3.5	D1	Computers
111	Alex	3.5	D3	Math
222	Sarah	4.00	D4	Science
333	Kevin	3.1	D6	Stat
444	Mark	2.3	D1	Computer

Table 4.5: The output of Algorithm 2, r'

SId	SName	SGPA	DId	DName
222	Kevin	3.5	D4	Math
111	Mark	4.00	D3	Stat
444	Alex	3.1	D1	Science
111	Mark	4.00	D1	Science
333	Sarah	2.3	D6	Computer

Given the outcome of Algorithm 2,$r^{'}$, a shuffled relation instance $r^{'}$ with the schema $R(A_1, \ldots, A_n)$ such that R is not in BCNF, as shown in table 4.5, with the following set $\mathcal{F}^{'}$ of functional dependencies that holds for $r^{'}$:

1. SId$\rightarrow$ SName,
2. SId$\rightarrow$ SGPA, and
3. DId $\rightarrow$ Dname.

Also, given the history of BCNF decomposition of R into $R_1, \ldots, R_j$ that was used for the shuffling with projection of $\mathcal{F}^{'}$ on R_i $(i = 1, \ldots, j)$ such that $\{\bar{\mathcal{F}}_1, \ldots, \bar{\mathcal{F}}_j\} \equiv \mathcal{F}^{'}$. We

tested whether or not restoring $r^{'}$ using Algorithm 4 – resulting in $\bar{r}$ such that $\bar{r}$ is equivalent to the original relation r, as shown in table 4.4, and $\bar{\mathcal{F}} \equiv \mathcal{F}^{'}$.

Table 4.6: Decomposing $r^{'}$ (Table 4.5) (the output of Algorithm 2) into sub-relations $r_1^{'}$, $r_2^{'}$, and $r_3^{'}$, respectively.

(a) $r_1{}'$

SId	SName	SGPA
111	Mark	4.00
222	Kevin	3.5
444	Alex	3.1
333	Sarah	2.3

(b) $r_2{}'$

DId	DName
D4	Math
D6	Computers
D1	Science
D3	Stat

(c) $r_3{}'$

SId	DId
222	D4
111	D3
444	D1
111	D1
333	D6

Table 4.7: Restoring sub-relations $r_1^{'}$, $r_2^{'}$, and $r_3^{'}$ in Table 4.6 into $\bar{r_1}$, $\bar{r_2}$, and $\bar{r_3}$ respectively.

(a) $\bar{r_1}$

SId	SName	SGPA
111	Alex	3.5
222	Sarah	4.00
333	Kevin	3.1
444	Mark	2.3

(b) $\bar{r_2}$

DId	DName
D1	Computers
D3	Math
D4	Science
D6	Stat

(c) $\bar{r_3}$

SId	DId
111	D1
111	D3
222	D4
333	D6
444	D1

Table 4.8: The restored relation, $\bar{r}$.

SId	SName	SGPA	DId	DName
111	Alex	3.5	D1	Computers
111	Alex	3.5	D3	Math
222	Sarah	4.00	D4	Science
333	Kevin	3.1	D6	Stat
444	Mark	2.3	D1	Computer

Given the history of BCNF decomposition of the original relation r with projection of $\mathcal{F}$ on R_i ($i = 1, \ldots, j$) such that $\{\mathcal{F}_1^{'}, \ldots, \mathcal{F}_j^{'}\} \equiv \mathcal{F}$, we decompose $r^{'}$ (the output of Algorithm 2) shown in Table 4.5 into sub-relations $r_1^{'}$, $r_2^{'}$, and $r_3^{'}$ as shown in Tables 4.6a, 4.6b, and 4.6c respectively. Our restoring shuffled generic relations algorithm, Algorithm 4, used

Algorithm 3 to restore the shuffled sub-relations $r_1^{'}$, $r_2^{'}$, and $r_3^{'}$ resulting in unshuffled sub-relations $\bar{r_1}$, $\bar{r_2}$, and $\bar{r_3}$ as shown in Tables 4.7a, 4.7b and 4.7c, respectively. The outcome of Algorithm 4, $\bar{r}$, was obtained by natural joining of the unshuffled sub-relations $\bar{r_1}$, $\bar{r_2}$, and $\bar{r_3}$.

We observe that Algorithm 4 succeeded in reconstruct the relation $\bar{r}$ such that $\bar{r}$ is equivalent to the original relation r and $\bar{r}$ satisfies $\mathcal{F}$

We also observe that the set $\mathcal{F}$ of functional dependencies of the original relation r shown in Table 4.4 is preserved in the unshuffled relation $\bar{r}$ (Table 4.8) after using Algorithm 4 to restore the shuffled generic relations. Therefore, we can conclude that the set $\bar{\mathcal{F}}$ of functional dependencies of (the output of Algorithm 4) is equivalent to the set $\mathcal{F}^{'}$ of functional dependencies (the output of Algorithm 2) and thereby $\bar{\mathcal{F}} \equiv \mathcal{F}$.

CHAPTER 5

SHUFFLING ALGORITHMS AND DATA-DRIVEN ASSOCIATION DISCOVERY

Data-driven association discovery refers to the process of searching for and identifying relationships among attributes based on empirical data. It allows one to determine whether data values corresponding to one attribute are significantly related to data values corresponding to another attribute.

In this chapter, we aim to define methods for determining the existence of a relationship between attributes in a given relation instance r.

To find an association between attributes, one must measure the statistical strength of the relationship between them. This can be found by using the χ^2 test (for categorical attributes) [13], Pearsons Correlation Coefficient PCC (also known as Pearsons r) test (for numerical Attributes) [36], or the ANOVA test when one attribute is categorical and the other is numerical [64]. In the following sections, we demonstrate how to use statistical methods to extract associations between attributes in a relation r. We also investigate risks relating to the use of data-driven association discovery methods by malicious users.

5.1 ASSOCIATION GRAPH

We can utilize statistical tests to build an *association graph*, G, which is defined as an undirected graph $G = (V, E)$ where V is the set of vertices, which represent the set of attributes, and E is the set of links, which indicate whether any two attributes in the graph are associated or not. The association graph G allows us to visualize the association among

attributes in a given relation instance r.

Definition 5.1. An association graph G is defined as $G = (V, E)$, where in V is the set of vertices, which represent the set of attributes, and E is the set of links, which indicate whether any two attributes in the graph are associated or not.

When analyzing the association between two attributes A_i and A_j in a given relation instance r, we measure the degree of dependency between them based their data values. If A_i and A_j are associated in r, then there must be a link between V_i and V_j in G.

Within relational database, there are three types of association discovery between attributes:

(1) Categorical Attributes Association: two categorical attributes A_i and A_j in relation instance r are associated if there is a link between V_i and V_j in G such that the strength of the link E_{ij} is significant when using χ^2 statistical test.

(2) Numerical Attributes Association: two numerical attributes A_i and A_j in relation instance r are associated if there is a link between V_i and V_j in G such that the strength of the link E_{ij} is significant when using Pearson Correlation Coefficient PCC statistical test.

(3) Mixed Attributes Association: two mixed attributes A_i and A_j, where A_i is categorical and A_j is numerical, in relation instance r are associated if there is a link between V_i and V_j in G such that the strength of the link E_{ij} is significant when using the *ANOVA* statistical test.

5.2 Modeling and Extraction of Data-driven Association-based Shuffle

In this section, we introduce Algorithm 5 which helps us to identify relationships among attributes of a given relation instance r. It also allows us to detect particular subsets of

attributes in r that must be shuffled together. The association between two attributes A_i and A_j can be notated as follows:

Notation 5.2. Let A_i and A_j be two attributes. We denote an association between A_i and A_j as $A_i \; \overline{a} \; A_j$.

Algorithm 5: Modeling and Extraction of Association between Attributes

Input : Relation instance r with schema $R(A_1, \ldots, A_n)$
Output: Relation instance $r^{'}$, which is the relation instance r after shuffling the data values of each attribute $A_1, \ldots, A_n$ such that data-driven associated attributes are shuffled together.

```
1  Procedure Testing for Attributes Association(r)
2      Start by drawing an undirected graph G = (V, E), where V is {A1, ...., An}
         and E is empty;
3      for every two vertices Ai and Aj in G do
4          if Ai and Aj are both categorical then
5              Use χ² to measure the strength of association;
6              if the association is significant then
7                  Draw a link between Ai and Aj;
8              end
9          else if Ai and Aj are both numerical then
10             Use PCC to measure the strength of association;
11             if the association is significant then
12                 Draw a link between Ai and Aj;
13             end
14         else
15             Use ANOVA test to measure the strength of association;
16             if the association is significant then
17                 Draw a link between Ai and Aj;
18             end
19         end
20     end
21     Let Si (i = 1, ..., k) be a strongly connected components in G;
22     r' = Shuffle r using the secure cryptographic shuffling algorithm such that
         every subset of attributes Si is bundle shuffled together;
23     Return r';
24 end
```

In Algorithm 5, we used the χ^2 test, Pearson correlation coefficient test, and ANOVA test to find associations among the database attributes. If we detect an association among

attributes, it follows that a potential attacker may also detect an association among values of these attributes. Our goal is to identify sets of attributes that should be shuffled together to reduce or completely avoid the security risks as attackers may be able to discover that the database is a corrupted database (shuffled database). Our empirical results are given in section 5.3.

Theorem 5.3. *Given a relation instance r with schema $R(A_1, \ldots, A_n)$, Algorithm 5 will generate $r^{'}$ such that $r^{'}$ satisfies data driven associations if we bundle shuffle the associated subset of attributes, $S_i \in \{A_1, \ldots, A_n\}$, together.*

Proof Sketch. Trivially follows. □

Property 5.4. Let A_i, A_j and A_k be attributes in a relation R. We say that an attribute association is:

i **Transitive** ,i.e., if $A_i \ \overline{a} \ A_j$ and $A_j \ \overline{a} \ A_k$ then $A_i \ \overline{a} \ A_k$.

ii **Reflexive** i.e., If $A_i \ \overline{a} \ A_i$

iii **Symmetric** i.e if $A_i \ \overline{a} \ A_j$ then $A_j \ \overline{a} \ A_i$.

5.3 Experimental Results of Modeling and Extraction of Data-driven Association-based Shuffle

In our last experiment, we validate the effectiveness of modeling and extraction of data-driven association-based shuffling discussed in (section 5.2) in order to enhance the security of the cryptographic shuffling algorithm. Our aim is to use Algorithm 5 to discover sets of attributes that want to be shuffled together to increase the resulting diffusion.

Table 5.1: Sample records of a bank dataset

ID	Sex	Income	Car	SavingAcc
ID12101	FEMALE	17546	NO	NO
ID12102	MALE	30085.1	YES	NO
ID12103	FEMALE	16575.4	YES	YES
ID12104	FEMALE	20375.4	NO	NO
ID12105	FEMALE	50576.3	NO	YES
⋮	⋮	⋮	⋮	⋮
ID12699	MALE	14711.8	NO	YES
ID12700	MALE	26671.6	YES	NO

We evaluate Algorithm 5 on a bank dataset, which consists of 600 records [sample records are shown in Table 5.1]. We aim to determine viable sets of semantically-associated attributes for bundle shuffling in a way that minimizes risk of inference.

The results of using Algorithm 5 to measure the statistical strength of data-driven association between attributes in the bank dataset are shown in Table 5.2. The outcome of Algorithm 5 is the association graph G shown in Figure 5.1. The association graph indicates which attributes must be shuffled together to prevent the hazard of inference problems. That is, each set of strongly connected components is a set of attributes that require bundle shuffling by assigning the same key set to them. S ={{Car, Income, SavingAcc}, {ID}, {Sex}} is the set of sets of attributes to be bundle-shuffled.

Table 5.2: The results of measuring statistical strength of data-driven association between bank dataset attributes

Var 1	Var 2	Test	P-value	Sig
Sex	Car	χ^2	0.870	No
Sex	Income	Anova	0.560	No
Sex	SavingAcc	χ^2	0.86	No
SavingAcc	Income	Anova	0.000	Yes
SavingAcc	Car	χ^2	0.401	No
Car	Income	Anova	0.046	Yes

We observe that attributes income, SavingAcc, and car are associated in G (Car $\overline{a}$ Income and Income $\overline{a}$ SavingAcc) and thus require bundle shuffling when using the secure cryptographic shuffling algorithm. We also observe that attributes ID and Sex are independent and not semantically associated to any other attributes. Thus, these attributes have no special requirements related to bundle shuffling. We showed that Algorithm 5 succeeded in modeling and extracting data-driven association among attributes of the given bank dataset.

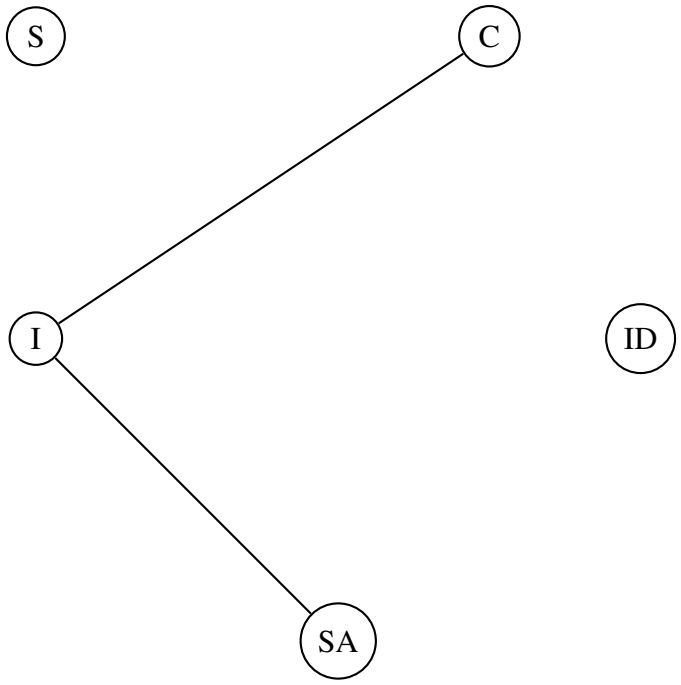


Figure 5.1: The resulting Association Graph G of the bank database.

CHAPTER 6

MODELING CONCEPT DRIFT IN THE CONTEXT OF DISCRETE BAYESIAN NETWORKS

6.1 INTRODUCTION

Concept drift is a significant challenge that greatly influences the accuracy and reliability of machine learning models. Therefore, there is a need to detect concept drift in order to ensure the validity of learned models. In this dissertation, we study the issue of concept drift in the context of discrete Bayesian networks. We propose a probabilistic graphical model framework to explicitly detect the presence of concept drift using latent variables. We employ latent variables to model real concept drift and uncertainty drift over time. For modeling real concept drift, we propose to monitor the mean of the distribution of the latent variable over time. For modeling uncertainty drift, we suggest to monitor the change in beliefs of the latent variable over time, i.e., we monitor the maximum value that the probability density function of the distribution takes over time.

The main focus of this chapter of the dissertation is to propose a probabilistic graphical model framework for detecting concept drift in the context of discrete Bayesian networks. Our proposed technique for detecting concept drift is based on using latent variables. The proposed modeling framework is an extension of Borchani et al. [12] framework such that it is directly applicable to discrete Bayesian networks. Borchani et al. represent concept drift using unobserved variables in continuous domains, namely in conditional linear Gaussian models. Our modeling framework for detecting the presence of concept drift using latent variables is applicable to general Bayesian network models and not limited to naive Bayes

classifier (previous modeling frameworks [12, 14] are limited to naive Bayes classifiers). In addition to modeling the posterior probability drift, we propose a new technique for modeling uncertainty, i.e., the amount of belief, across time.

We have implemented our proposed frameworks and present our empirical results using two of the most commonly used Bayesian networks in Bayesian experiments, namely the Burglary-Earthquake Network and the Chest Clinic network.

This chapter of the dissertation is structured as follows. In section 6.2, we present the problem setting. In section 6.3, we present our framework for detecting concept drift using latent variables in discrete Bayesian networks. In section 6.4, we extend our modeling framework into higher dimensions. In section 6.5 we present our empirical results.

6.2 Problem Setting

We focus on modeling concept drift in the context of discrete Bayesian networks. In a nonstationary environment, we assume that at each time point t (for $t = 1, 2, \dots$) data arrives in a batch (a.k.a. *a window*), which is a collection of cases. Let $Batch\ [A_1, \dots, A_m]$ be the schema of the incoming batch with attributes $A_1, \dots, A_m$. We assume without loss of generality that the incoming batches have equal sizes, i.e., each batch contains n cases. Let $Batch\ t = \{case_1^t, \dots, case_n^t\}$ be a collection of cases (a.k.a. *observations* or *findings*) that arrives at time t.

$$\underbrace{case_1^1, case_2^1, \dots, case_n^1}_{\text{Batch 1, t = 1}}, \underbrace{case_1^2, case_2^2, \dots, case_n^2}_{\text{Batch 2, t = 2}}, \dots$$

Each finding, denoted as $case$, is over attributes $A_1, \dots, A_m$ and of the form $case = < A_1 = v_1, \dots, A_m = v_m >$ (or simply can be written as $case = < v_1, \dots, v_m >$), such that v_k is the value of attribute A_k ($1 \leq k \leq m$). When a new batch $Batch\ t + 1$ arrives at time point $t+1$, the Bayesian network model can simply be updated using Bayes' theorem.

To detect the presence of concept drift between two time points $t = i$ and $t = i + 1$, we consider two types of drifts as follows: (1) *Posterior Distribution Drift*, and (2) *Uncertainty*

Drift.

6.2.1 POSTERIOR DISTRIBUTION DRIFT; A.K.A. REAL CONCEPT DRIFT

Posterior distribution drift occurs when the conditional probability changes on the target variable whereas the input variables remain unchanged [24]. That is, the value of the posterior probability at time $t = i$, $P_{t_i}(y \mid A)$, is not equal to the value of the posterior probability at time $t = i + 1$, $P_{t_{i+1}}(y \mid A)$.

In Bayesian statistics, Bayes' theorem can be written in a useful form for Bayesian network update and inference as follows: The posterior probability is proportional to the product of the prior probability and the likelihood (Posterior probability $\propto$ Prior probability $\times$ Likelihood [39]). Having a prior that is conjugate for the likelihood function will make it mathematically convenient to calculate the posterior distribution since the posterior distribution will be from the same family of distribution as the prior [56]. For instance, multiplying a beta-distributed prior, $Beta(\alpha, \beta)$, with a binomial-distributed likelihood function, $Binomial(n, \theta)$, yields a beta-distributed posterior distribution, $Beta(q + \alpha, n - q + \beta)$, where n is the total number of cases, and q is the count of successes [3].

In what follows, we consider detecting the presence of posterior distribution drift in the context of discrete Bayesian networks with respect to a random variable $\boldsymbol{X}$ that is beta-distributed, which we denote as $\boldsymbol{X} \sim Beta(\alpha, \beta)$. We capture the existence of posterior distribution drift by monitoring the mean of the beta distribution at every time point $t = i$, denoted as μ_i, i.e., the expected value of $\boldsymbol{X}$ at every time point $t = i$, $\boldsymbol{E}(\boldsymbol{X})$, as follows:

$$\mu_i = \boldsymbol{E}(\boldsymbol{X}) = \frac{q_i + \alpha}{\alpha + n_i + \beta} \tag{6.1}$$

where n_i and q_i are the total number of cases and the count of successes at time $t = i$, respectively, and hyperparameters α, β are greater than or equal to 1.

6.2.2 UNCERTAINTY DRIFT

Measuring the amount of uncertainty in input data is defined as entropy [61]. *Uncertainty drift* is a variable that reflects the change in beliefs over time. That is, for a random variable $\boldsymbol{X}$, the maximum value that a probability density function $f_i(x; \alpha, \beta)$ takes at time $t = i$ is not equal to the maximum value that a probability density function $f_{i+1}(x; \alpha, \beta)$ takes at time $t = i + 1$. This kind of drift is mainly caused by the change in the total number of observed cases. It is important to point out that modeling uncertainty drift in the context of Bayesian networks is powerful as it is a sensitive diagnostic for detecting real concept drift.

Herein, we consider detecting the presence of uncertainty drift in the context of discrete Bayesian networks with respect to a random variable $\boldsymbol{X}$ is beta-distributed, $\boldsymbol{X} \sim Beta(\alpha, \beta)$. We capture the existence of uncertainty drift by monitoring the maximum value that the probability density function of the beta distribution takes at every time point $t = i$, which we denote as ψ_i, as follows:

$$\begin{aligned} \psi_i &= \max_{\boldsymbol{X}=\boldsymbol{x}} f_i(x; \alpha, \beta, n_i, q_i) \\ &= f_i(\frac{q_i + \alpha - 1}{\alpha + n_i + \beta - 2}; \alpha, \beta, n_i, q_i) \end{aligned} \tag{6.2}$$

where n_i and q_i are the total number of cases and the count of successes at time $t = i$, respectively, x is the mode of the beta distribution ($0 \leq x \leq 1$), and hyperparameters α, β are greater than or equal to 1.

In our setting, we iterate over time steps ($t = 1, 2, \dots$). At each time point $t = i$, we use the incoming batch, $Batch\ i$, to update the current Bayesian network model. We then use our approaches to detect the existence of model drift. we assume that the distribution of the data does not change inside the batch, i.e., we capture the presence of model drift across time steps ($t = 1, 2, \dots$) and not within the set of observations arrives at a particular time point. If the variations in the values of μ_i and ψ_i are important, we conclude that our Bayesian network model has drifted.

We summarize the notations we use in this chapter in Table 6.1.

Table 6.1: Notations.

Notation	Description
$\boldsymbol{Batch\ [A_1, ..., A_m]}$	The schema of incoming batch with attributes $A_1, ..., A_m$
$\boldsymbol{Batch\ i}$	A collection of cases that arrives at time i
$\boldsymbol{case_j^i}$	The j^{th} observation of $Batch\ i$
$\boldsymbol{\mu_i}$	The mean of the posterior probability at time i
$\boldsymbol{\psi_i}$	The maximum value that the PDF takes at time i
$\boldsymbol{X}$	A random variable
$\boldsymbol{X} \sim Beta(\alpha, \beta)$	A random variable that is beta-distributed
$\boldsymbol{X} \sim Dir(\alpha_1, \ldots, \alpha_r)$	A random variable that is Dirichlet-distributed

6.3 Modeling Concept Drift Using Latent Variables

In this section, we present a modeling technique for detecting concept drift in discrete Bayesian networks. We explicitly model concept drift using latent variables. To avoid unnecessary complication, we assume that only posterior distribution and uncertainty drift over time, i.e., for each edge $A \rightarrow B$ in a Bayesian network model BN_1, we detect the existence of concept drift by monitoring the posterior distribution drift and uncertainty drift of $A \rightarrow B$ over time.

Our modeling technique for detecting the presence of concept drift in discrete Bayesian networks is described using plate notation as shown in Figure 6.1. The fundamental idea of our modeling approach is to add a latent node for each edge $A \rightarrow B$ in a given Bayesian network model BN_1. We call this latent node U_{AB}^t. It is important to point out that for each collection of observation j of time t, the unobserved node U_{AB}^t is added as the child of the observed nodes A_j^t and B_j^t.

The latent variable U_{AB}^t captures the posterior drift and the uncertainty drift for each collection of observations j of time t. It is essential to point to the fact that both values of observed variables A_j^t and B_j^t contribute to the drift of the latent variable U_{AB}^t as described in the following section.

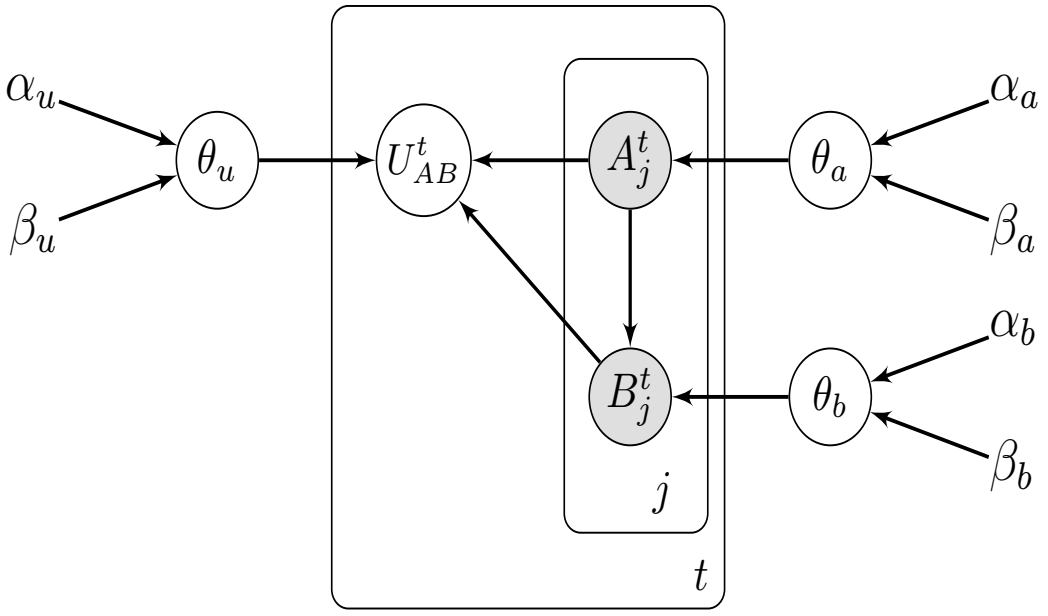


Figure 6.1: Modeling concept drift with latent variables in discrete Bayesian networks. A^t_j and B^t_j are observed nodes. U^t_{AB} is a latent (unobserved) node. θ_a, θ_b, and θ_u are model parameters. α_a, β_a, α_b, β_b, α_u, and β_u are model hyperparameters.

6.3.1 Posterior Distribution Drift

In our modeling technique presented in Figure 6.1, the posterior distribution drift of the latent variable U^t_{AB} that is monitored at each time point $t = i$ is as follows:

$$\begin{aligned}\mu_i &= P_{t_i}(U^t_{AB} \mid A^t, B^t) \\ &= \frac{q_i + \alpha_u}{\alpha_u + n_i + \beta_u}\end{aligned}$$

where n_i and q_i are the total number of cases and the count of successes at time $t = i$, respectively, and hyperparameters α_u, β_u are greater than or equal to 1.

6.3.2 Uncertainty Drift

In our modeling technique shown in Figure 6.1, to capture the uncertainty drift of the latent variable U^t_{AB} over time, we monitor the maximum value that a probability density function

$f_i(x; \alpha_u, \beta_u)$ of the latent variable takes at each time point $t = i$ as follows:

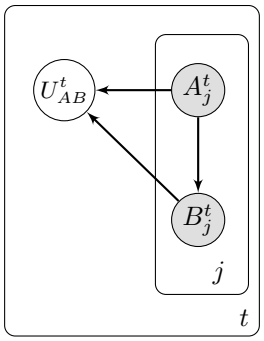
$$\begin{aligned}\psi_i &= \max_{\boldsymbol{X}=\boldsymbol{x}} f_i(x; \alpha_u, \beta_u, n_i, q_i) \\ &= f_i(\frac{q_i + \alpha_u - 1}{\alpha_i + n_i + \beta_u - 2}; \alpha_u, \beta_u, n_i, q_i)\end{aligned}$$

where n_i and q_i are the total number of cases and the count of successes at time $t = i$, respectively, x is the mode of the beta distribution ($0 \leq x \leq 1$), and hyperparameters α_u, β_u are greater than or equal to 1.

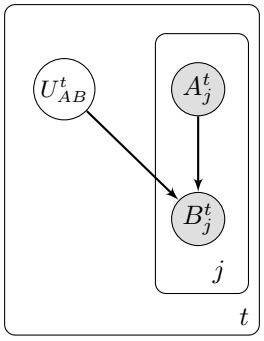
It is important to emphasize that our modeling technique, at each time point $t = i$, receives j observations where $j = 1$ to n. These observations are used to update the Bayesian network model. The latent variable U^t_{AB} is then used to capture the presence of posterior drift (i.e., drift in the value of μ_i) and uncertainty drift (i.e., drift in the value of ψ_i). If the values of μ_i and ψ_i vary significantly, we conclude that our Bayesian network model has drifted.

The a priori expected values of concept and uncertainty drifts can be expressed via the prior distribution for the latent node U^t_{AB}. We use hyperparameters α_u and β_u to express the prior knowledge that we may have about concept and uncertainty drifts at a particular time point.

An important point to be made concerning the development of our modeling technique for detecting concept drift (presented in Figure 6.1) is that it contains no causal interpretation. We do not place any causal assumption on the interaction between the observed variables and the latent variable. Despite the fact that it is mathematically feasible to build causal and non-causal modeling techniques (as shown in Figure 6.2) to detect the presence of concept drift, it is not necessary to consider causal effects between variables as these effects are not the main focus of our modeling approach. For this reason, we tolerate that the interpretation of our modeling approach of concept drift is merely statistical, i.e., associational.



(a) Non-causal.



(b) Causal.

Figure 6.2: Options for building a modeling approach for detecting concept drift.

6.4 Generalization of our Framework into Higher Dimensions

To expand our modeling framework for variables with more than two states, we can use the Dirichlet distribution, which is a continuous multivariate probability distribution. In Bayesian statistics, Dirichlet distribution, which is denoted as $Dir(\alpha_1, \ldots, \alpha_r)$, is parameterized by r hyperparameters $\alpha_1, \ldots, \alpha_r$ such that α_i ($1 \leq i \leq r$) is integer and $\alpha_i \geq 1$ [48]. This distribution is the generalization of the beta distribution for $r > 2$, i.e., beta is a special case when $r = 2$.

A Dirichlet distributed prior is conjugate for the likelihood function that is multinomial distributed. That is, multiplying a Dirichlet-distributed prior, $Dir(\alpha_1, \ldots, \alpha_r)$, with a multinomial-distributed likelihood function, $Multi(w_1, \ldots, w_r; c_1, \ldots, c_r)$, yields a Dirichlet-distributed posterior distribution, $Dir(\alpha_1 + c_1, \ldots, \alpha_r + c_r)$, where $\alpha_1, \ldots, \alpha_r$ are Dirichlet distribution hyperparameters, $w_1, \ldots, w_r$ are Dirichlet distributed random vari-

ables, and $c_1, \ldots, c_r$ are the number of occurrences of each category.

We focus on detecting the presence of posterior distribution drift in the context of discrete Bayesian networks with respect to a random variable $\boldsymbol{X} = [\boldsymbol{X_1}, ..., \boldsymbol{X}_r]$ that is Dirichlet-distributed, which we denote as $\boldsymbol{X} \sim Dir(\alpha_1, \ldots, \alpha_r)$. We capture the existence of posterior distribution drift by monitoring the mean of the Dirichlet distribution at every time point $t = i$, denoted as μ_i, i.e., the expected value of $\boldsymbol{X_j}$ at every time point $t = i$, $\boldsymbol{E}(\boldsymbol{X_j})$, as follows:

$$\begin{aligned} \mu_i &= \boldsymbol{E}(\boldsymbol{X_j}) \\ &= \frac{\alpha_j + c_j}{\alpha_{all}} \end{aligned}$$

where $\alpha_{all} = \sum_{s=1}^{r} \alpha_s + c_s$ and c_j is the number of occurrences of $\boldsymbol{X_j}$.

In addition to detecting the posterior drift, we consider detecting the presence of uncertainty drift in the context of discrete Bayesian networks with respect to a random variable $\boldsymbol{X}$ is Dirichlet-distributed as described above. We capture the existence of uncertainty drift by monitoring the maximum value of $\boldsymbol{X_j}$ that the probability density function of the Dirichlet distribution takes at every time point $t = i$, which we denote as ψ_i, as follows:

$$\begin{aligned} \psi_i &= \max_{\boldsymbol{X_j}=\boldsymbol{x}} f_i(x; \alpha, \beta, \alpha_{all}, c_j) \\ &= f_i(\frac{\alpha_j + c_j - 1}{\alpha_{all} - r}; \alpha, \beta, \alpha_{all}, c_j) \end{aligned}$$

where $\frac{\alpha_j + c_j - 1}{\alpha_{all} - r}$ is the mode of the Dirichlet distribution.

6.5 Empirical Results

We have implemented our modeling framework and tested our approach using two of the most commonly used example networks in Bayesian experiments, Burglary-Earthquake Network and Chest Clinic network.

6.5.1 BURGLARY-EARTHQUAKE NETWORK

The Burglary-Earthquake Network was created by Pearl [53] and is a commonly used example in Bayesian networks. As shown in Figure 6.3, the Burglary-Earthquake Network is a fictitious network that could be used to model an alarm system in a house. The network consists of five nodes and four edges. The nodes are as follows:

(1) Node **B** shows if there is a burglary,

(2) Node **E** shows whether there is an earthquake,

(3) Node **A** shows if the alarm goes off,

(4) Node **M** shows if Mary calls, and

(5) Node **J** shows if John calls.

The causal relations between the nodes in this network is expressed by directed edges. For instance, the edge $B \rightarrow A$ means that burglary may cause the alarm to be activated and so on. We refer the readers to [53] for a full description of this network.

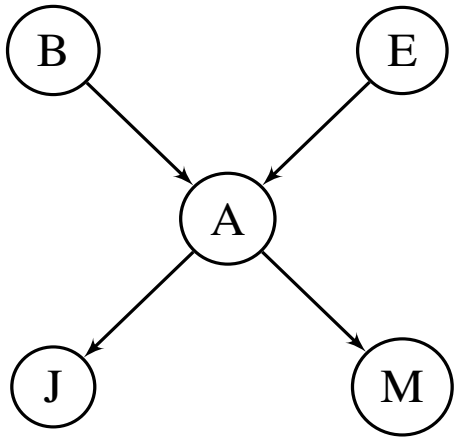


Figure 6.3: The original Burglary-Earthquake Network.

We apply our approach for detecting the presence of concept drift in discrete domains over time to the Burglary-Earthquake Network. To set up our experiment, we have implemented this network using *Hugin™ Research 8.4*. *Hugin™ case generator* [40, 49] is

then used to generate 15 simulated datasets of $1,000$ cases each. These datasets are named $Batch\ 1$ through $Batch\ 15$. During the simulation process of some datasets, the posterior probabilities are changed in order to simulate the existence of concept drift as follows:

(1) The edge $B \rightarrow A$:

   (i) the posterior probabilities, $P(A = F \mid B = F)$ and $P(A = T \mid B = F)$, are changed during the simulation process of datasets $Batch\ 3$, $Batch\ 7$, and $Batch\ 12$.

   (ii) the posterior probabilities, $P(A = T \mid B = F)$ and $P(A = T \mid B = T)$, are changed during the simulation process of the dataset $Batch\ 3$.

(2) The edge $E \rightarrow A$: the posterior probabilities, $P(A = F \mid E = F)$ and $P(A = T \mid E = F)$, are changed during the simulation process of the dataset $Batch\ 4$.

(3) The edge $A \rightarrow J$: the posterior probabilities, $P(J = F \mid A = T)$, is changed during the simulation process of the dataset $Batch\ 7$.

(4) The edge $A \rightarrow M$: the posterior probabilities, $P(M = F \mid A = F)$ and $P(M = T \mid A = T)$, are changed during the simulation process of the dataset $Batch\ 7$.

In our experiment, we assume that at each time point t ($t = 1, \ldots, 15$), we receive $Batch\ t$ which has j instances (we set $j = 1,000$ cases).

To implement our framework, we added a latent node for each edge in the Burglary-Earthquake Network. That is, we added latent nodes U^t_{BA}, U^t_{EA}, U^t_{AJ}, and U^t_{AM} to detect the presence of real concept drift and uncertainty drift for the edges $B \rightarrow A$, $E \rightarrow A$, $A \rightarrow J$, and $A \rightarrow M$, respectively, as shown in Figure 6.4. We assume that we have no prior knowledge about concept drift. That is, we assume that all hyperparameters of the latent variables, $\alpha(.)$ and $\beta(.)$, are equal to 1.

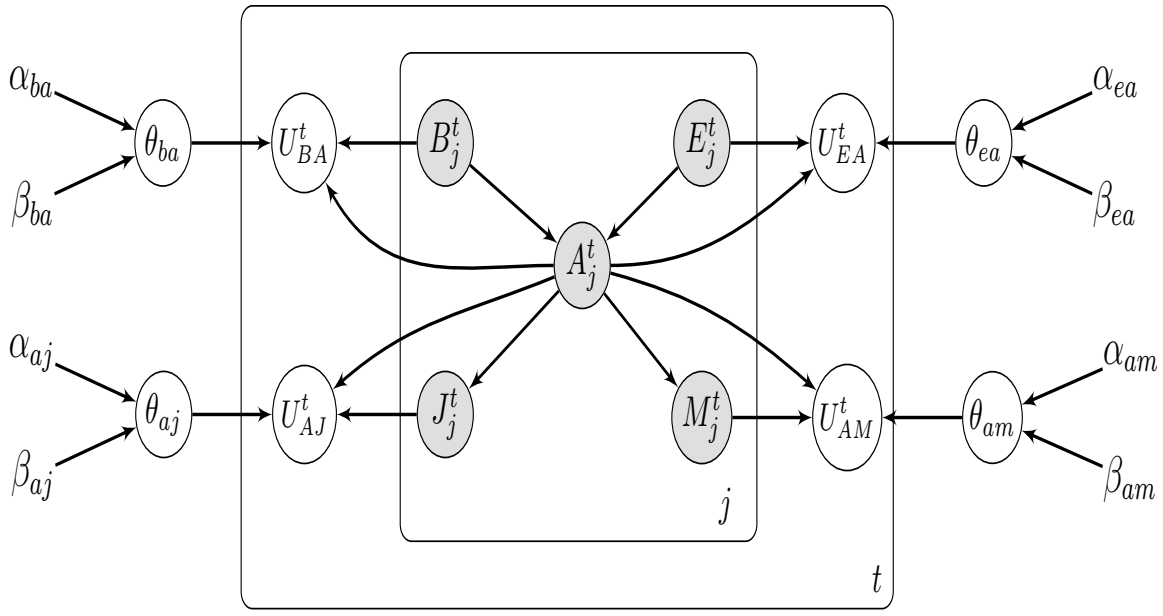


Figure 6.4: Our proposed framework for modeling concept drift with latent variables in the Burglary-Earthquake Network.

The results of using our framework to detect the presence of real concept drift and uncertainty drift are summarized in Table 6.2 and Table 6.3, respectively. Note that values shown in bold in Table 6.2 and Table 6.3 indicate the presence of drift.

Our framework succeeded in detecting the existence of real concept drift and uncertainty drift. We observe that a change in the posterior probability and the uncertainty is reflected by a variation in the evolution of the corresponding latent variable. For instance, we observe drifts in the posterior probabilities and the uncertainties of the latent variable U^t_{BA}, namely when $U = u \mid B = F,\ A = F$ and $U = u \mid B = F,\ A = T$, at time points 3, 7, and 12. We also observe that the posterior and the uncertainty of the latent variable U^t_{BA} drift at time point 3 namely when $U = u \mid B = T,\ A = F$ and $U = u \mid B = T,\ A = T$.

We observe that our framework is sensitive to changes in the underlying distribution of data that newly incoming batches may cause. That is, if the number of observations in the newly incoming $Batch\ t$ at time t is less than the expected number of observations, then the framework shall report a drop in the posterior and the uncertainty at time t and vice versa. For instance, for the edge $B \rightarrow A$, namely when $U = u \mid B = F,\ A = F$, our

Table 6.2: Results of using our framework to detect the presence of real concept drift in the Burglary-Earthquake Network.

(a) The result of using the latent variable U^t_{BA} to detect the presence of real concept drift for the edge $B \rightarrow A$.

Posterior of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{B=F,A=F}$	0.98	0.98	**0.94**	0.95	0.96	0.96	**0.94**	0.95	0.96	0.96	0.96	**0.94**	0.95	0.95	0.96
$U^t_{B=F,A=T}$	0.006	0.006	**0.04**	0.032	0.027	0.024	**0.04**	0.032	0.029	0.027	0.026	**0.04**	0.032	0.031	0.029
$U^t_{B=T,A=F}$	0.0005	0.0005	**0.003**	0.002	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
$U^t_{B=T,A=T}$	0.01	0.01	**0.02**	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

(b) The result of using the latent variable U^t_{EA} to detect the presence of real concept drift for the edge $E \rightarrow A$.

Posterior of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{E=F,A=F}$	0.96	0.96	0.96	**0.92**	0.93	0.93	0.94	0.94	0.94	0.95	0.95	0.95	0.95	0.95	0.95
$U^t_{E=F,A=T}$	0.01	0.01	0.01	**0.06**	0.05	0.04	0.04	0.03	0.03	0.03	0.03	0.03	0.02	0.02	0.02
$U^t_{E=T,A=F}$	0.016	0.015	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013	0.013
$U^t_{E=T,A=T}$	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

(c) The result of using the latent variable U^t_{AJ} to detect the presence of real concept drift for the edge $A \rightarrow J$.

Posterior of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{A=F,J=F}$	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93	0.93
$U^t_{A=F,J=T}$	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
$U^t_{A=T,J=F}$	0.002	0.002	0.002	0.002	0.002	0.002	**0.005**	0.005	0.005	0.004	0.004	0.004	0.004	0.004	0.003
$U^t_{A=T,J=T}$	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

(d) The result of using the latent variable U^t_{AM} to detect the presence of real concept drift for the edge $A \rightarrow M$.

Posterior of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{A=F,M=F}$	0.97	0.97	0.97	0.97	0.97	0.97	**0.95**	0.95	0.96	0.96	0.96	0.96	0.96	0.96	0.96
$U^t_{A=F,M=T}$	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
$U^t_{A=T,M=F}$	0.0059	0.0055	0.0053	0.0052	0.0053	0.0054	0.0055	0.0054	0.0054	0.0054	0.0056	0.0058	0.0059	0.0062	0.0063
$U^t_{A=T,M=T}$	0.012	0.011	0.011	0.011	0.011	0.010	**0.031**	0.029	0.027	0.025	0.024	0.023	0.022	0.021	0.020

framework captured a drop in the posterior and uncertainty drifts in the incoming batch at time point $t = 3$, $Batch\ 3$. This drop is due to that fact that the number of observed cases in $Batch\ 3$ was less than the expected number of cases.

It should be noted that after each drift, the values of the posterior and uncertainty will be smoothly re-increasing/re-decreasing attempting to recover from the drift. It is also important to point out that if the number of cases in the newly incoming batch is as expected, our framework concludes that there is no drift to anticipate, and thus no action needs to be taken. Explanations of the other experiments for other edges trivially follow the explanation of the edge $B \rightarrow A$.

Table 6.3: Results of using our framework to detect the presence of uncertainty drift in the Burglary-Earthquake Network.

(a) The result of using the latent variable U^t_{BA} to detect the presence of uncertainty drift for the edge $B \rightarrow A$.

Uncertainty of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{B=F,A=F}$	100.12	141.89	**92.23**	117.43	139.34	160.91	**149.16**	165.48	181.69	196.20	210.32	**200.34**	213.09	225.98	238.56
$U^t_{B=F,A=T}$	176.08	239.53	**110.58**	143.84	173.97	203.90	**179.13**	200.81	223.02	243.46	262.87	**238.04**	255.17	271.71	288.88
$U^t_{B=T,A=F}$	368.79	736.67	**419.45**	559.04	659.50	791.25	876.52	1001.63	1075.17	1100.22	1210.16	1272.71	1332.53	1434.97	1489.15
$U^t_{B=T,A=T}$	120.16	174.42	**172.15**	209.05	239.86	269.18	270.64	293.38	315.84	335.86	354.83	371.80	389.11	408.88	425.84

(b) The result of using the latent variable U^t_{EA} to detect the presence of uncertainty drift for the edge $E \rightarrow A$.

Uncertainty of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{E=F,A=F}$	65.90	93.85	118.30	**93.13**	110.12	125.49	139.30	152.34	165.06	177.75	189.38	200.50	211.33	222.22	232.59
$U^t_{E=F,A=T}$	110.77	160.13	203.07	**109.76**	133.27	156.12	177.12	197.56	217.20	236.51	256.01	274.14	291.73	310.17	327.28
$U^t_{E=T,A=F}$	103.32	148.89	192.56	224.58	252.62	272.56	291.30	310.42	327.11	348.07	361.79	379.68	396.79	413.21	429.03
$U^t_{E=T,A=T}$	125.87	174.43	212.23	241.51	270.10	295.95	319.71	339.92	362.59	385.71	407.63	425.18	443.61	461.32	479.88

(c) The result of using the latent variable U^t_{AJ} to detect the presence of uncertainty drift for the edge $A \rightarrow J$.

Uncertainty of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{A=F,J=F}$	50.10	70.86	86.79	99.88	111.89	123.16	132.71	142.09	150.68	158.79	166.51	173.99	181.16	188.05	195.24
$U^t_{A=F,J=T}$	57.31	81.07	98.99	113.62	127.27	139.17	153.76	164.29	174.01	183.21	191.80	200.35	208.54	215.99	223.75
$U^t_{A=T,J=F}$	271.21	391.32	482.51	559.05	626.30	687.01	**459.35**	511.28	560.94	608.57	647.27	684.19	726.51	774.78	822.20
$U^t_{A=T,J=T}$	103.32	146.44	181.48	210.82	236.56	267.54	294.44	313.32	329.78	345.52	364.22	379.68	393.45	408.88	425.84

(d) The result of using the latent variable U^t_{AM} to detect the presence of uncertainty drift for the edge $A \rightarrow M$.

Uncertainty of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{A=F,M=F}$	77.67	111.99	136.36	157.73	176.53	194.16	**156.79**	173.49	187.71	200.94	213.50	225.03	236.74	246.86	257.27
$U^t_{A=F,M=T}$	120.16	178.65	215.57	247.11	280.35	305.22	330.48	360.53	380.07	394.85	412.89	428.49	445.26	459.79	473.95
$U^t_{A=T,M=F}$	176.08	250.97	308.18	356.32	391.04	423.24	453.32	487.13	518.79	543.58	562.72	577.34	592.19	603.58	615.41
$U^t_{A=T,M=T}$	120.16	170.49	209.04	244.26	272.56	305.22	**190.01**	212.59	232.98	253.50	272.82	291.47	310.56	327.51	345.56

All in all, we have shown that our framework that is based on using latent variables to detect the presence of concept drift is effective and sensitive to changes in the underlying distribution of data in nonstationary environments over time. Our framework was successfully able to detect the existence of both real concept drift and uncertainty drift. Our new proposed approach for capturing uncertainty drift is sensitive and useful as it can ensure the occurrence of real concept drift.

6.5.2 CHEST CLINIC NETWORK

The Chest Clinic network, a.k.a. the Visit to Asia network, was created by Lauritzen and Spielgelhalter [35] and is widely used in Bayesian network experiments. This network is a simple, fictitious medical network which could be employed in a medical facility to

diagnose patients as shown in Figure 6.5. The Chest Clinic network consists of eight nodes, which represent random variables, and eight edges, which indicate the causal relations between the nodes.

The nodes are as follows:

(1) Node **A** indicates if the patient has been to Asia recently,

(2) Node **S** indicates whether the patient smokes,

(3) Node **B** indicates whether the patient has Bronchitis or not,

(4) Node **L** indicates whether the patient has been diagnosed with lung cancer,

(5) Node **T** indicates whether the patient has been diagnosed with Tuberculosis,

(6) Node **E** indicates whether the patient has been diagnosed with either lung cancer or Tuberculosis,

(7) Node **X** indicates whether the X-ray results are positive, and

(8) Node **D** indicates whether the patient has been diagnosed with Dyspnea.

A complete description of this medical Bayesian network model is as follows [35]:

> Shortness-of-breath (dyspnoea) may be due to tuberculosis, lung cancer, or bronchitis, or none of them, or more than one of them. A recent visit to Asia increases the chances of tuberculosis, while smoking is known to be a risk factor for both lung cancer and bronchitis. The results of a single chest X-ray do not discriminate between lung cancer and tuberculosis, as neither does the presence or absence of dyspnoea.

We apply our approach for detecting the presence of concept drift in discrete domains to the Chest Clinic network. To avoid unnecessary computations, we use our framework to detect the presence of concept drift of the weakest edge in the Chest Clinic network. Using

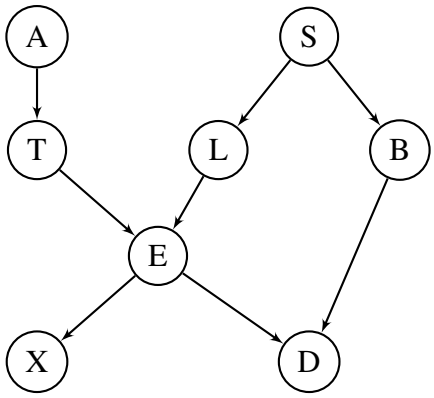


Figure 6.5: The original Chest Clinic network.

Alsuwat et al.'s link strength measure, the edge from $A \rightarrow T$ is the weakest edge in this network [2]. Therefore, we employ our framework to detect the existence of concept and uncertainty drifts of the edge $A \rightarrow T$.

To set up our experiment, we have implemented this network using *Hugin™ Research 8.4*. *Hugin™ case generator* [40, 49] is then used to generate 15 simulated datasets of $2,000$ cases each. These datasets are named $Batch\ 1$ through $Batch\ 15$. To simulate the presence of concept drift, we change the posterior probabilities during the simulation process as follows:

(1) the posterior probability $P(T = no \mid A = no)$ is changed during the simulation process of datasets $Batch\ 4$ and $Batch\ 11$.

(2) the posterior probability $P(T = yes \mid A = no)$ is changed during the simulation process of dataset $Batch\ 4$.

(3) the posterior probability $P(T = no \mid A = yes)$ is changed during the simulation process of dataset $Batch\ 11$.

(4) the posterior probability $P(T = yes \mid A = yes)$ is changed during the simulation process of datasets $Batch\ 2$ and $Batch\ 10$.

In this experiment, we assume that at each time point t ($t = 1, \ldots, 15$), our framework receives $Batch\ t$ which has j observations (j is set at $2,000$ cases).

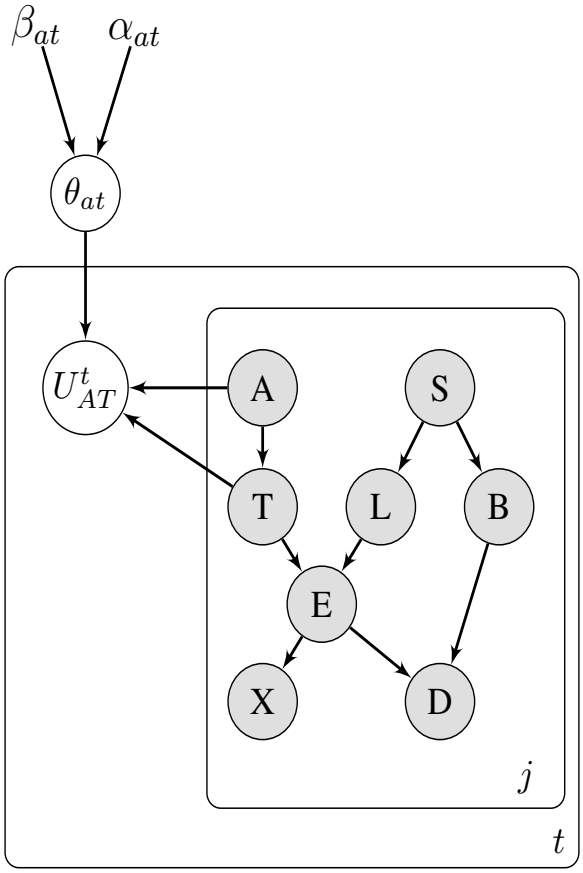


Figure 6.6: Our proposed framework for modeling concept drift of the weakest edge in the Chest Clinic network using a latent variable.

To implement our framework, we added a latent node for the weakest edge in the Chest Clinic network. That is, we added the latent node U^t_{AT} to detect the presence of real concept drift and uncertainty drift for the edges $A \rightarrow T$ as shown in Figure 6.6. We assume that we have no prior knowledge about concept drift, i.e., we assume that the hyperparameters of the latent variable U^t_{AT}, α_{at} and β_{at}, are equal to 1.

The results of applying our framework to detect the existence of real concept drift and uncertainty drift of the weakest edge in the Chest Clinic network are summarized in Table 6.4 and Table 6.5, respectively. Note that values shown in bold in Tables 6.4 and 6.5 indicate the presence of drift. Our framework was successfully able to detect the presence of real concept drift and uncertainty drift. We observe that a change in the posterior probability and the uncertainty is reflected by a variation in the evolution of the

latent variable U^t_{AT}. For example, we observe drifts in the posterior probabilities and the uncertainties of the latent variable U^t_{AT} as follows:

(1) when $U = u \mid A = no,\ T = no$, the posterior and the uncertainty drift at time points 4 and 11,

(2) when $U = u \mid A = yes,\ T = no$, the posterior and the uncertainty drift at time point 4,

(3) when $U = u \mid A = no,\ T = yes$, the posterior and the uncertainty drift at time point 11, and

(4) when $U = u \mid A = yes,\ T = yes$, the posterior and the uncertainty drift at time points 2 and 10.

Table 6.4: Results of using our framework to detect the presence of real concept drift of the weakest edge in the Chest Clinic network.

Posterior of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{A=no,T=no}$	0.98	0.98	0.98	**0.96**	0.96	0.97	0.97	0.97	0.97	0.97	**0.95**	0.96	0.96	0.96	0.96
$U^t_{A=yes,T=no}$	0.008	0.008	0.008	**0.02**	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
$U^t_{A=no,T=yes}$	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	0.009	**0.02**	0.02	0.02	0.02	0.02
$U^t_{A=yes,T=yes}$	0.0009	**0.002**	0.001	0.001	0.001	0.001	0.0009	0.0009	0.0009	**0.001**	0.001	0.001	0.001	0.001	0.001

Table 6.5: Results of using our framework to detect the presence of uncertainty drift of the weakest edge in the Chest Clinic network.

Uncertainty of	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$U^t_{A=no,T=no}$	137.74	184.65	228.18	**197.55**	231.22	262.59	290.81	317.06	340.94	361.32	**298.71**	318.90	338.57	357.81	376.66
$U^t_{A=yes,T=no}$	205.74	287.02	346.34	**236.70**	282.12	324.22	363.55	400.53	434.71	468.68	501.12	530.61	558.87	587.62	615.37
$U^t_{A=no,T=yes}$	183.30	263.10	326.76	374.87	419.77	462.42	499.52	534.05	564.76	595.52	**374.68**	402.07	429.08	455.05	480.74
$U^t_{A=yes,T=yes}$	736.31	**527.74**	751.41	955.80	1144.48	1373.21	1539.98	1696.67	1844.71	**1591.53**	1716.84	1838.06	1955.52	2069.48	2180.19

We observe that our framework is sensitive to changes in the underlying distribution of incoming data. Moreover, our framework is able to quickly detect the existence of drifts. Another important observation is that receiving more observations that belong to the cell with the highest test statistics value will reflect a higher variation of the evolution of the corresponding latent variable and thus will reflect a drift in the posterior and the uncertainty.

Overall, we have shown that our framework that is based on using latent variables to model concept drift in nonstationary environments is efficient to detect posterior and uncertainty drifts of the weakest edge in a given Bayesian network model.

CHAPTER 7

EXPLAINING CONCEPT DRIFT

7.1 INTRODUCTION

A common problem of using machine learning models in nonstationary environments is that data evolves over time. This phenomenon is known as concept drift wherein the distribution of the underlying data changes across time. Concept drift is a significant challenge that greatly influences the accuracy and reliability of machine learning models. Therefore, it is essential to build effective modeling techniques for detecting concept drift. Thus, such effective concept drift detection techniques are able to not only detect the existence of concept drift but also ensure the validity of learned models.

Detecting concept drift is crucial and active research in machine learning systems. The vast majority of concept drift literature has been devoted to the investigation and study of methods and techniques for detecting concept drift. However, fewer studies have researched the causes of the occurring concept drift.

This chapter of the dissertation primarily focuses on finding an explanation of concept drift in the context of discrete Bayesian networks. We use latent variables to help explain the occurring posterior probability drift. To be able to find an explanations of the occurring real concept drift, we need to extend our previously proposed modeling technique for detecting concept drift using latent variables in discrete Bayesian networks (presented in Chapter 6) to model concept drift across time. The extended version of our original modeling technique is as presented in Figure 7.1.

The model presented in Figure 7.1 shows the relations between latent variables across

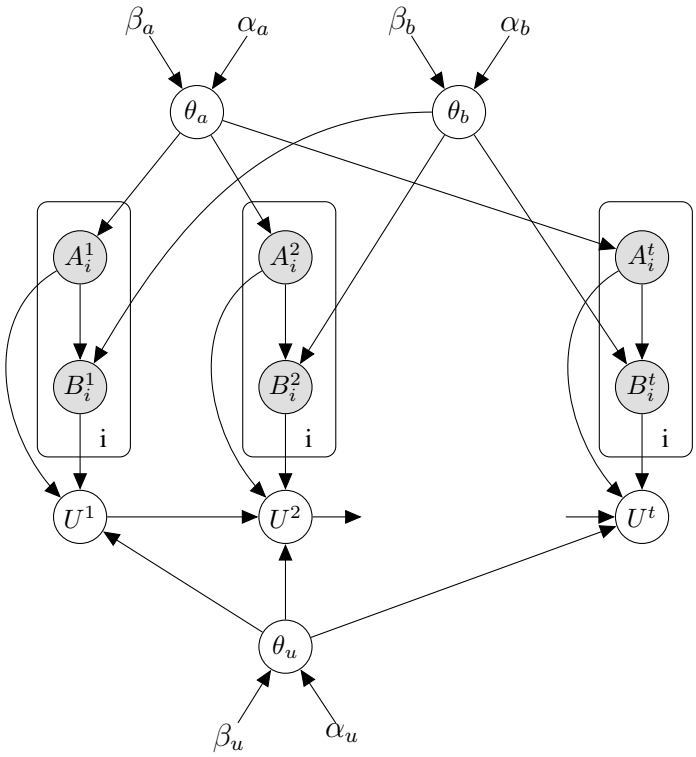


Figure 7.1: Modeling concept drift across time using latent variables.

time. We assume that the posterior probability drift is being captured and stored using our proposed modeling technique at specific time points. In what follows, we propose a framework to help find explanations of the occurring posterior probability drift.

7.2 Framework for Explaining Concept Drift

In this section, we present our framework for finding an explanation of the occurring concept drift. It is important to point out that our original modeling technique (presented in Chapter 6) is used to capture the existence of concept drift while the proposed framework in this section is used to find an explanation of the detected concept drift across time. That is, the new proposed probabilistic graphical model framework, which is based on using latent variables, provides an explanation of the detected posterior probability drift across time.

The main component of our proposed framework are as follows:

- **CDDM**: Concept Drift Detection Method, and
- **PPDE**: Posterior Probability Drift Explanation.

Concept Drift Detection Method

In the CDDM, our framework uses a concept drift detection technique to examine the incoming batches with the goal of detecting posterior probability drift.

In the CDDM, we use our proposed modeling technique that is based on using latent variables to detect the existence of concept drift. Our modeling technique is as described in Chapter 6. If our modeling technique detects a posterior probability drift, then we proceed with the Posterior Probability Drift Explanation in order to check for explanations of the occurring concept drift with the ultimate goal of distinguishing between natural concept drifts and malicious attacks. Note that, the outcome of CDDM is a sequence of posterior probability values denoted as Seq, where $Seq = \{v_1, v_2, \ldots, v_n\}$ such that $0 \leq v_i \leq 1$. Otherwise, there is no detected concept drift and thus our framework continues to receive new incoming batches.

Posterior Probability Drift Explanation

In the PPDE, our framework uses a "Kullback-Leibler (KL) Divergence Based Measure" defined below to detect whether the incoming sequence of posterior probability values has an explanation similar to one stored in "Previously Stored Sequences" database.

We develop a measure for finding an explanation for the occurring posterior probability drift. Our proposed measure is based on a commonly used distance measure, which is Kullback-Leibler (KL) Divergence [17], as follows:

Kullback-Leibler (KL) Divergence Based Measure

This measure quantifies the distance between each instance in current sequence of posterior probabilities $Seq = \{v_1, v_2, \ldots, v_n\}$, wherein a posterior probability drift has been detected, and the corresponding instance in a previously stored sequence, $Seq'_i = \{v_1^{i'}, v_2^{i'}, \ldots, v_n^{i'}\}$. Our KL divergence based measure is denoted by $KL(v_t \| v_t^{i'})$ and defined as follows:

Definition 7.1. Let $Seq = (v_1, \ldots, v_n)$ be a sequence of posterior probabilities wherein a real concept drift has been detected and $Seq'_i = (v_1^{i'}, \ldots, v_n^{i'})$ be a sequence of previously stored drifts. Then the KullbackLeibler divergence between two instances v_t and $v_t^{i'}$ is defined as follows:

$$KL(v_t \| v_t^{i'}) = |v_t \log \frac{v_t}{v_t^{i'}}| \tag{7.1}$$

Our Kullback-Leibler divergence based measure, $KL(v_t \| v_t^{i'})$, for providing an explanation of the detected real concept drift satisfies the following two properties:

- non-negativity: $\forall\, v_t \in Seq$, $v_t^{i'} \in Seq'_i$, $KL(v_t \| v_t^{i'}) \geq 0$ (even though we take only an instance of the probability distribution, the absolute value is enough to make our measure non-negative), and
- non-degeneracy: $\forall\, v_t \in Seq$, $v_t^{i'} \in Seq'_i$, $KL(v_t, v_t^{i'}) = 0$, only if $v_t = v_t^{i'}$.

In this section, we introduce Algorithm 6, which presents algorithmic details of how the PPDE checks for explanations of the occurring posterior probability drift.

In PPDE, we use Algorithm 6 as a method to detect if there is a similar explanation of the occurring real concept drift in the previously occurred and captured drifts. If the value of ω is greater than λ (where λ is a small number $0 \leq \lambda \leq 1$ that represents the required similarities between the corresponding instances (values) in sequence Seq and sequence $Seq_i^{'}$), then we conclude that sequence $Seq_i^{'}$ is not a proper explanation of the newly detected drift in sequence Seq. Otherwise, if we have successfully checked all instances of sequence Seq with the corresponding instances of sequence $Seq_i^{'}$, then we conclude that sequence $Seq_i^{'}$ is an explanation of the occurring real concept drift in sequence Seq.

We summarize the process of applying our framework in Figure 7.2. We describe the workflow of our framework as follows:

- Our framework receives a set of incoming batches, namely $Batch\ 1, \ldots, Batch\ n$ from an untrusted source. Such batches need to checked since they may have real concept drift or contaminated batches.

Algorithm 6: KL Divergence Based Measure for finding an Explanations of Posterior Probability Drift

Input :

- A sequence of posterior probability values, $Seq = \{v_1, v_2, \ldots, v_n\}$, wherein a posterior probability drift has been detected,
- A database of previously stored sequences DB_1, where $DB_1 = \{Seq_1^{'}, \ldots, Seq_k^{'}\}$ such that $Seq_i^{'} = \{v_1^{i'}, v_2^{i'}, \ldots, v_n^{i'}\}$, and
- λ where $0 \leq \lambda \leq 1$ is the rate at which the value v_t in sequence Seq and $v_t^{i'}$ in sequence $Seq_i^{'}$ are required to be similar.

Output: $Seq_i^{'}$, where $1 \leq i \leq k$, if there exists an explanation of the occurring real concept drift; otherwise, a message is printed as there is no explanation.

Procedure `KL Based Measure(`Seq, DB_1`)`
  for *(*$i = 1$*;* $i \leq k$*;* i^{++}*)* **do**
    for *(*$t = 1$*;* $t \leq n$*;* t^{++}*)* **do**
      $\omega = 0.0$ ▷ a variable that has the differences between the posterior probabilities v_t and $v_t^{i'}$;
      $\omega = |v_t \log \frac{v_t}{v_t^{i'}}|$;
      if $\omega > \lambda$ **then**
        Break;
      end
      if $t == n$ **then**
        Return $Seq_i^{'}$;
      end
    end
  end
  Return Msg "There exists no explanation for the occurring real concept drift";
end

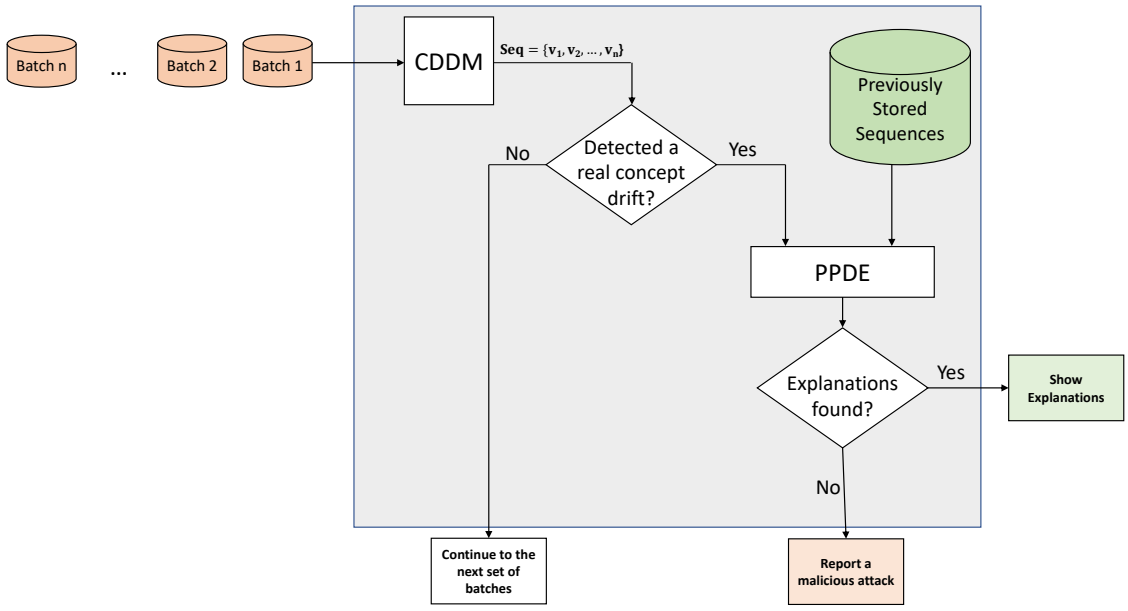


Figure 7.2: Framework for explaining the occurring posterior probability drift.

- the CDDM part of our framework checks for the existence of real concept drift. If a posterior probability drift has been detected, then our framework proceeds with the PPDE part. Otherwise, our framework continue to receive a new set of batches.

- the PPDE part of our framework checks for an explanation of the occurring posterior probability drift. If such an explanations exists, then our framework shows the explanation. Otherwise, a message is printed as there is no similar case has occurred before.

7.3 Empirical Results

We have implemented our approaches for finding explanations for posterior probability drift and applied them to the Burglary-Earthquake Network. The main goal of this section is to validate the effectiveness of our proposed framework for finding an explanation of a detected real concept drift. That is, we validate the effectiveness of the CDDM to detect an existing real concept drift and then validate the effectiveness of the PPDE, namely Al-

gorithm 6, to find whether there exists an explanation of the detected real concept drift or not.

To set up our experiment, we have implemented the Burglary-Earthquake Network using *Hugin™ Research 8.4*. To avoid unnecessary computations, we use our framework to find explanations of the detected concept drift of the edge $B \rightarrow A$ in the Burglary-Earthquake Network.

In our implementation, we have i ($i = 1, \ldots, k$) independent experiments. We call the outcome of each experiment a sequence of posterior probabilities that have real concept drift. We assume that each time point t ($t = 1, \ldots, n$) in each of the k experiments, we receive $Batch_t^{i'}$ which has j observations (we set $j = 1,000$ cases). At the end of each experiment, we store the resulting sequence, Seq_i', in a database of previously stored explanations DB_1.

Hugin™ case generator [40, 49] is then used to generate the database DB_1. We assume that $k = 6$ and $n = 5$. This means that we have 6 sequences, $Seq_1', \ldots, Seq_6'$, stored in DB_1. Each sequence Seq_i' for $1 \leq i \leq 6$ requires simulating 5 datasets of $1,000$ cases each. These datasets are named $Batch_1^{i'}$ through $Batch_5^{i'}$. During the simulation process, the posterior probabilities of the edge $B \rightarrow A$ are changed to simulate the existence of real concept drift. That is, the posterior probability $P(A = F \mid B = F)$ is changed during the simulation process of the following datasets: (1) $Batch_3^{1'}$, (2) $Batch_4^{2'}$, (3) $Batch_5^{3'}$, (4) $Batch_2^{4'}$, (5) $Batch_5^{4'}$, (6) $Batch_2^{5'}$, and (7) $Batch_2^{6'}$.

We present the results of generating the database of previously stored explanations, DB_1, in Table 7.1. We observe that our modeling approach, CDDM, presented in Chapter 6, succeeded in detecting the existence of all the simulated posterior probability drifts.

We next use *Hugin™ case generator* to generate 5 datasets of $1,000$ cases each. We name these datasets $Batch\ 1$ through $Batch\ 5$. During the simulation process of these datasets, we changed the posterior probabilities of the edge $B \rightarrow A$ at time point $t = 2$ in order to simulate the presence of real concept drift.

Table 7.1: A database of stored explanations denoted as DB_1. DB_1 has sequences of posterior probabilities, named Seq'_1 through Seq'_6, and each sequence has at least one real concept drift. Note that the real concept drifts are shown in bold font.

Sequence Number	Time point 1	Time point 2	Time point 3	Time point 4	Time point 5
Seq'_1	0.98	0.98	**0.94**	0.95	0.96
Seq'_2	0.96	0.96	0.96	**0.92**	0.93
Seq'_3	0.97	0.97	0.97	0.97	**0.93**
Seq'_4	0.85	**0.89**	0.88	**0.79**	0.81
Seq'_5	0.97	**0.93**	0.95	0.95	0.96
Seq'_6	0.85	**0.90**	0.88	0.87	0.87

We present the results of using the CDDM modeling part of our framework to detect the presence of real concept drift in the sequence $Seq = \{v_1, v_2, \ldots, v_n\}$ as shown in Table 7.2. We observe that our framework succeeded in detecting the presence of the simulated posterior probability drift at time point $t = 2$.

Table 7.2: The sequence Seq of posterior probabilities, which has one real concept drift at time point $t = 2$ shown in bold font.

Sequence Number	Time point 1	Time point 2	Time point 3	Time point 4	Time point 5
Seq	0.86	**0.91**	0.89	0.88	0.87

Our framework then sends Seq as an input to the PPDE part of our framework, namely Algorithm 6, in order to check whether there exists an explanation for the detected real concept drift in sequence Seq. We assume that $\lambda = 0.025$. This means that the KL divergences value between the corresponding values in the sequence Seq and each of stored sequences Seq'_i should be no larger than 0.025.

We present the result of using Algorithm 6 to check for explanations of the detected posterior probability drift in sequence Seq in Table 7.3. We observe that Algorithm 6 succeeded in finding an explanations for the occurring drift, Seq at time $t = 2$, by comparing sequence Seq with the stored sequences in DB_1. It is important to point out that cells filled with an *x* indicate that Algorithm 6 was not required to calculate those cells as there was a KL divergences value larger than 0.025, which causes Algorithm 6 to terminate and continues to find another explanation.

Table 7.3: The result of using Algorithm 6 to check for explanations of the detected real concept drift in sequence Seq.

Compare between	Time point 1	Time point 2	Time point 3	Time point 4	Time point 5
Seq'_1 and Seq	0.19	x	x	x	x
Seq'_2 and Seq	0.16	x	x	x	x
Seq'_3 and Seq	0.17	x	x	x	x
Seq'_4 and Seq	0.01	0.02	0.02	0.11	x
Seq'_5 and Seq	0.17	x	x	x	x
Seq'_6 and Seq	0.01	0.01	0.01	0.01	0.00

We observe that our proposed framework for explaining real concept drift is capable to reliably provide an explanation of the occurring drift. We observe that using latent variables to build frameworks is an efficient mechanism to both detect and explain concept drifts. We have shown that our frameworks based on using latent variables are not only efficient to model and detect the presence of concept drift but also able efficient to provide an explanation of the occurring real concept.

CHAPTER 8

CONCLUSION AND FUTURE WORK

In recent years, data integrity has become a major security challenge. In this dissertation, we have first studied the problem that cryptographic shuffling algorithms do not preserve data dependencies in relational databases. For this, we have developed methods for preserving functional dependencies and data-driven dependencies while using secure cryptographic shuffling algorithms. We have presented formal proofs of our proposed methods.

For addressing the issue of violating data dependencies while using shuffling algorithms, our results indicate that our new proposed techniques are able to preserve functional dependencies and data-driven dependencies when used with cryptographic shuffling algorithms. This, in turn, makes it more difficult for an adversary to detect that the database was shuffled.

We have then studied the issue of concept drift in the context of discrete Bayesian networks in nonstationary environments. We have proposed a framework for modeling concept drift using latent variables in discrete Bayesian networks. Our modeling technique using latent variables is sensitive to changes in the underlying distribution of data. We have also proposed a probabilistic graphical model framework for finding an explanation of the detected real concept drift. Our modeling framework is able to provide an explanation of the occurring posterior probability drift across time.

For addressing the issue of concept drift in the context of discrete Bayesian networks, we have implemented our proposed frameworks and applied them to the Burglary-Earthquake Network and the Chest Clinic Network, which are widely used networks in Bayesian experiments. Our results indicate that our frameworks are not only sensitive to changes in the

underlying distribution of incoming data but also can easily detect and provide an explanation of the occurring concept drift across time.

In our future work, we aim to extend our current work as follows:

We aim to extend our solutions for preserving data dependencies while using secure cryptographic shuffling algorithms to the cases where the BCNF decomposition does not preserve the functional dependencies. For this, we will investigate the possibility of using application specific enforcement to build code templates that enforce functional dependencies.

We will investigate the possibility of using an information entropy measure to increase the security of our proposed approaches for preserving data dependencies while using secure cryptographic shuffling algorithms. That is, the information entropy measure should be able to assure that the proposed approaches do not give more information to malicious users.

We aim to extend our proposed framework for modeling the presence of concept drift such that it is capable of distinguishing between malicious data modification/insertion and natural model drift.

We will investigate the possibility of using the expectation-maximization (EM) algorithm to estimate the values of the latent nodes. We aim to compare the sensitivity and the accuracy of this proposed method for modeling concept drift with our original modeling technique that was proposed in Chapter 6.

BIBLIOGRAPHY

[1] Nabil R Adam and John C Worthmann, *Security-control methods for statistical databases: a comparative study*, ACM Computing Surveys (CSUR) **21** (1989), no. 4, 515–556.

[2] Emad Alsuwat, Hatim Alsuwat, Marco Valtorta, and Csilla Farkas, *Adversarial data poisoning attacks against the pc learning algorithm*, International Journal of General Systems (2019), 1–29.

[3] Emad Alsuwat, Marco Valtorta, and Csilla Farkas, *How to generate the network you want with the pc learning algorithm*, Proceedings of WUPES **18** (2018), 1–12.

[4] H. Alsuwat, E. Alsuwat, T. Geng, C. Huang, and C. Farkas, *Data dependencies preserving shuffle in relational database*, 2019 2nd International Conference on Data Intelligence and Security (ICDIS), June 2019, pp. 180–187.

[5] Hatim Alsuwat., Emad Alsuwat., Marco Valtorta., John Rose., and Csilla Farkas., *Modeling concept drift in the context of discrete bayesian networks*, Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 1: KDIR, INSTICC, SciTePress, 2019, pp. 214–224.

[6] Bill Arbaugh, *Improving the tcpa specification*, Computer **35** (2002), no. 8, 77–79.

[7] Manuel Baena-Garcıa, José del Campo-Ávila, Raúl Fidalgo, Albert Bifet, R Gavalda, and R Morales-Bueno, *Early drift detection method*, Fourth international workshop on knowledge discovery from data streams, vol. 6, 2006, pp. 77–86.

[8] Liana B. Baker and Jim Finkle, *Sony playstation suffers massive data breach*, apr 2011.

[9] R Vidya Banu and N Nagaveni, *Evaluation of a perturbation-based technique for privacy preservation in a multi-party clustering scenario*, Information Sciences **232** (2013), 437–448.

[10] Paul Barrett, *Structural equation modelling: Adjudging model fit*, Personality and Individual differences **42** (2007), no. 5, 815–824.

[11] Albert Bifet and Ricard Gavalda, *Learning from time-changing data with adaptive windowing*, Proceedings of the 2007 SIAM international conference on data mining, SIAM, 2007, pp. 443–448.

[12] Hanen Borchani, Ana M Martínez, Andrés R Masegosa, Helge Langseth, Thomas D Nielsen, Antonio Salmerón, Antonio Fernández, Anders L Madsen, and Ramón Sáez, *Modeling concept drift: A probabilistic graphical model based approach*, International Symposium on Intelligent Data Analysis, Springer, 2015, pp. 72–83.

[13] Max Bramer, *Principles of data mining*, vol. 180, Springer, 2007.

[14] Rafael Cabañas, Andrés Cano, Manuel Gómez-Olmedo, Andrés R Masegosa, and Serafín Moral, *Virtual subconcept drift detection in discrete data using probabilistic graphical models*, International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Springer, 2018, pp. 616–628.

[15] William G Cochran, *The $\chi 2$ test of goodness of fit*, The Annals of Mathematical Statistics (1952), 315–345.

[16] Edgar F Codd, *A relational model of data for large shared data banks*, Communications of the ACM **13** (1970), no. 6, 377–387.

[17] Thomas M Cover and Joy A Thomas, *Elements of information theory*, John Wiley & Sons, 2012.

[18] Tore Dalenius and Steven P Reiss, *Data-swapping: A technique for disclosure control*, Journal of statistical planning and inference **6** (1982), no. 1, 73–85.

[19] Ramez Elmasri and Shamkant Navathe, *Fundamentals of database systems*, Addison-Wesley Publishing Company, 2010.

[20] Jim Finkle and Dhanya Skariachan, *Target cyber breach hits 40 million payment cards at holiday peak*, dec 2013.

[21] R. A. Fisher, *The conditions under which $\chi 2$ measures the discrepancey between observation and hypothesis*, Journal of the Royal Statistical Society **87** (1924), no. 3, 442–450.

[22] João Gama, Ricardo Fernandes, and Ricardo Rocha, *Decision trees for mining data streams*, Intelligent Data Analysis **10** (2006), no. 1, 23–45.

[23] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues, *Learning with drift detection*, Brazilian symposium on artificial intelligence, Springer, 2004, pp. 286–295.

[24] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia, *A survey on concept drift adaptation*, ACM computing surveys (CSUR) **46** (2014), no. 4, 44.

[25] Jing Gao, Wei Fan, Jiawei Han, and Philip S Yu, *A general framework for mining concept-drifting data streams with skewed distributions*, Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, 2007, pp. 3–14.

[26] Tieming Geng, Hatim Alsuwat, Chin-Tser Huang, and Csilla Farkas, *Securing relational structured database with attribute association-aware shuffling*, Tech. report, University of South Carolina, SC, USA, 2018.

[27] Terry Halpin and Tony Morgan, *Information modeling and relational databases*, Morgan Kaufmann, 2010.

[28] Michael Bonnell Harries, Claude Sammut, and Kim Horn, *Extracting hidden context*, Machine learning **32** (1998), no. 2, 101–126.

[29] David Tse Jung Huang, Yun Sing Koh, Gillian Dobbie, and Russel Pears, *Detecting volatility shift in data streams*, 2014 IEEE International Conference on Data Mining, IEEE, 2014, pp. 863–868.

[30] Adriana Sayuri Iwashita and João Paulo Papa, *An overview on concept drift learning*, IEEE Access **7** (2019), 1532–1547.

[31] Mohammad Ali Kadampur et al., *A noise addition scheme in decision tree for privacy preserving data mining*, arXiv preprint arXiv:1001.3504 (2010).

[32] Mark G Kelly, David J Hand, and Niall M Adams, *The impact of changing populations on classifier performance*, Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, Citeseer, 1999, pp. 367–371.

[33] Daniel Kifer, Shai Ben-David, and Johannes Gehrke, *Detecting change in data streams*, Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 180–191.

[34] Henning Koehler, *Finding faithful boyce-codd normal form decompositions*, International Conference on Algorithmic Applications in Management, Springer, 2006.

[35] Steffen L Lauritzen and David J Spiegelhalter, *Local computations with probabilities on graphical structures and their application to expert systems*, Journal of the Royal Statistical Society. Series B (Methodological) (1988), 157–224.

[36] Joseph Lee Rodgers and W Alan Nicewander, *Thirteen ways to look at the correlation coefficient*, The American Statistician **42** (1988), no. 1, 59–66.

[37] Chong K Liew, Uinam J Choi, and Chung J Liew, *A data distortion by probability distribution*, ACM Transactions on Database Systems (TODS) **10** (1985), no. 3, 395–411.

[38] Andrew Liptak, *Hackers accessed more personal data from equifax than previously disclosed*, feb 2018.

[39] Scott M Lynch, *Introduction to applied bayesian statistics and estimation for social scientists*, Springer Science & Business Media, 2007.

[40] Anders L Madsen, Frank Jensen, Uffe B Kjaerulff, and Michael Lang, *The hugin tool for probabilistic graphical models*, International Journal on Artificial Intelligence Tools **14** (2005), no. 03, 507–543.

[41] Mila Majster-Cederbaum, *Ensuring the existence of a bcnf-decomposition that preserves functional dependencies in o (n2) time*, (1990).

[42] Douglas C Montgomery, *Design and analysis of experiments*, John wiley & sons, 2017.

[43] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera, *A unifying view on dataset shift in classification*, Pattern Recognition **45** (2012), no. 1, 521–530.

[44] Steve Morgan, *Did uber throw its cso under the bus?*, nov 2017.

[45] Donald F Morrison, *Multivariate analysis of variance*, Encyclopedia of biostatistics **5** (2005).

[46] Krishnamurty Muralidhar and Rathindra Sarathy, *A theoretical basis for perturbation methods*, Statistics and Computing **13** (2003), no. 4, 329–335.

[47] _____, *Data shuffling - a new masking approach for numerical data*, Management Science **52** (2006), no. 5, 658–670.

[48] Richard E Neapolitan et al., *Learning bayesian networks*, vol. 38, Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[49] Kristian G Olesen, Steffen L Lauritzen, and Finn V Jensen, *ahugin: A system creating adaptive causal probabilistic networks*, Uncertainty in Artificial Intelligence, 1992, Elsevier, 1992, pp. 223–229.

[50] Sylvia L Osborn, *Testing for existence of a covering boyce-codd normal form*, Information Processing Letters **8** (1979), no. 1, 11–14.

[51] Ewan S Page, *Continuous inspection schemes*, Biometrika **41** (1954), no. 1/2, 100–115.

[52] Sinno Jialin Pan, Qiang Yang, et al., *A survey on transfer learning*, IEEE Transactions on knowledge and data engineering **22** (2010), no. 10, 1345–1359.

[53] Judea Pearl, *Probabilistic reasoning in intelligent systems: Networks of plausible inference*, Morgan Kaufmann, 1988.

[54] Karl Pearson, *Note on regression and inheritance in the case of two parents*, Proceedings of the Royal Society of London **58** (1895), 240–242.

[55] Charles P Pfleeger and Shari Lawrence Pfleeger, *Security in computing*, Prentice Hall Professional Technical Reference, 2002.

[56] Howard Raiffa and Robert Schlaifer, *Applied statistical decision theory*, Div. of Research, Graduate School of Business Administration, Harvard Univ., 1961.

[57] Ronica Raj and Veena Kulkarni, *A study on privacy preserving data mining: Techniques, challenges and future prospects*, International Journal of Innovative Research in Computer and Communication Engineering **3** (2015), no. 11.

[58] Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand, *Exponentially weighted moving average charts for detecting concept drift*, Pattern recognition letters **33** (2012), no. 2, 191–198.

[59] Rathindra Sarathy and Krishnamurty Muralidhar, *The security of confidential numerical data in databases*, information systems research **13** (2002), no. 4, 389–403.

[60] Raquel Sebastião, João Gama, and Teresa Mendonça, *Fading histograms in detecting distribution and concept changes*, International Journal of Data Science and Analytics **3** (2017), no. 3, 183–212.

[61] Claude Elwood Shannon, *A mathematical theory of communication*, ACM SIGMOBILE mobile computing and communications review **5** (2001), no. 1, 3–55.

[62] Stephen M Stigler, *Francis galton's account of the invention of correlation*, Statistical Science (1989), 73–79.

[63] A TSYMBAL, *The problem of concept drift: definitions and related work*, Technical Report, Department of Computer Science, Trinity College Dublin (2004).

[64] Stéphane Tufféry, *Data mining and statistics for decision making*, vol. 2, Wiley Chichester, 2011.

[65] Shenghui Wang, Stefan Schlobach, and Michel Klein, *What is concept drift and how to measure it?*, International Conference on Knowledge Engineering and Knowledge Management, Springer, 2010, pp. 241–256.

[66] Weiping Wang, Jianzhong Li, Chunyu Ai, and Yingshu Li, *Privacy protection on sliding window of data streams*, Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007. International Conference on, IEEE, 2007, pp. 213–221.

[67] Geoffrey I. Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean, *Characterizing concept drift*, Data Mining and Knowledge Discovery **30** (2016), no. 4, 964–994.

[68] Geoffrey I Webb, Loong Kuan Lee, François Petitjean, and Bart Goethals, *Understanding concept drift*, arXiv preprint arXiv:1704.00362 (2017).

[69] Gerhard Widmer and Miroslav Kubat, *Learning in the presence of concept drift and hidden contexts*, Machine learning **23** (1996), no. 1, 69–101.

[70] Leon Willenborg and Ton De Waal, *Elements of statistical disclosure control*, vol. 155, Springer Science & Business Media, 2012.

[71] Zhiqiang Yang, Sheng Zhong, and Rebecca N Wright, *Privacy-preserving queries on encrypted data*, European Symposium on Research in Computer Security, Springer, 2006, pp. 479–495.

[72] Indrė Žliobaitė, *Learning under concept drift: an overview*, arXiv preprint arXiv:1010.4784 (2010).

[73] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama, *An overview of concept drift applications*, Big data analysis: new algorithms for a new society, Springer, 2016, pp. 91–114.