Fall 2019

Challenges in Large-Scale Machine Learning Systems: Security and Correctness

Emad Alsuwat

CHALLENGES IN LARGE-SCALE MACHINE LEARNING SYSTEMS: SECURITY AND CORRECTNESS

by

Emad Alsuwat

Bachelor of Computer Science
Taif University, 2008

Master of Science
University of South Carolina, 2014

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

Linyuan Lu, Committee Member

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

DEDICATION

I am dedicating this dissertation to my beloved family. This work could not be done with you. I am very thankful for all of your support and encouragement along the way.

ACKNOWLEDGMENTS

First and foremost I express my sincere gratitude for my advisors Prof. Csilla Farkas and Prof. Marco Valtorta. It has been an honor to be their PhD student. I am very thankful for their motivation, immense knowledge, and continuous support. I appreciate all the contributions of time and ideas to make my PhD dissertation experience productive and stimulating.

Besides my advisors, I would like to thank the rest of my dissertation committee members, Prof. John Rose, Prof. Chin-Tser Huang, and Prof. Linyuan Lu, for their insightful comments and encouragement.

I would like to thank my family for all their unlimited encouragement and love. I would like to sincerely thank my parents who raised me with a love of science and for providing me the support I need. I also would like to thank my brothers, sisters, wife and son for supporting me spiritually throughout my life.

Last but not the least, I would like to thank my friends for all their love and true friendship. My time at the University of South Carolina was made enjoyable in large part due to the many friends that became a part of my life. I am grateful for time spent with them and grateful for unforgettable memories.

ABSTRACT

In this research, we address the impact of data integrity on machine learning algorithms. We study how an adversary could corrupt Bayesian network structure learning algorithms by inserting contaminated data items. We investigate the resilience of two commonly used Bayesian network structure learning algorithms, namely the PC and LCD algorithms, against data poisoning attacks that aim to corrupt the learned Bayesian network model.

Data poisoning attacks are one of the most important emerging security threats against machine learning systems. These attacks aim to corrupt machine learning models by contaminating datasets in the training phase. The lack of resilience of Bayesian network structure learning algorithms against such attacks leads to inaccuracies of the learned network structure.

In this dissertation, we propose two subclasses of data poisoning attacks against Bayesian networks structure learning algorithms: (1) Model invalidation attacks when an adversary poisons the training dataset such that the Bayesian model will be invalid, and (2) Targeted change attacks when an adversary poisons the training dataset to achieve a specific change in the structure. We also define a novel measure of the strengths of links between variables in discrete Bayesian networks. We use this measure to find vulnerable sub-structure of the Bayesian network model. We use our link strength measure to find the easiest links to break and the most believable links to add to the Bayesian network model. In addition to one-step attacks, we define long-duration (multi-step) data poisoning attacks when a malicious attacker attempts to send contaminated cases over a period of time. We propose to use the distance measure between Bayesian network models and the value of data conflict to detect data poisoning attacks. We propose a 2-layered

framework that detects both traditional one-step and sophisticated long-duration data poisoning attacks. Layer 1 enforces "reject on negative impacts" detection; i.e., input that changes the Bayesian network model is labeled potentially malicious. Layer 2 aims to detect long-duration attacks; i.e., observations in the incoming data that conflict with the original Bayesian model.

Our empirical results show that Bayesian networks are not robust against data poisoning attacks. However, our framework can be used to detect and mitigate such threats.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Machine learning algorithms, including Bayesian Network algorithms, are not secure against adversarial attacks. A machine learning algorithm is a *secure learning algorithm* if it functions well in adversarial environments [10]. Recently, several researchers addressed the problem of attacking machine learning algorithms [10, 16, 63, 50]. Data poisoning attacks are considered one of the most important emerging security threats against machine learning systems [43]. These attacks aim to corrupt the machine learning model by contaminating the data in the training phase.

Data poisoning attacks against Support Vector Machines (SVMs) [16, 66, 67, 47, 42, 19, 32] and Neural Networks (NNs) [69] have been studied extensively. However, we found no research on evaluating the vulnerabilities of Bayesian network learning algorithms against adversarial attacks.

In this dissertation, we investigate data poisoning attacks against Bayesian network algorithms. We study two classes of attacks against Bayesian network structure learning algorithms: model invalidation attacks and targeted change attacks. For model invalidation attacks, an adversary poisons the training dataset such that the learned Bayesian model will be invalid. For targeted change attacks, an adversary poisons the training dataset to achieve a particular goal, such as masking or adding a link in a Bayesian network model [6] [8] [9].

For example, assume that DB_1 is a learning dataset, and the model B_1 is the learning outcome when feeding DB_1 to a Bayesian network learning algorithm. Figure 1.1 shows

Table 1.1: Selected tuples from the original dataset DB_1

X	B	D	A	S	L	T	E
No	Yes	No	Yes	No	No	Yes	No
No	No	No	No	No	No	Yes	No
Yes	No	Yes	No	No	No	No	No
No	No	No	No	No	Yes	No	No
No	No	No	No	No	No	Yes	No
No	Yes	No	Yes	No	No	Yes	Yes
...							
...							
...							
No	No	Yes	No	No	Yes	No	No
No	No	No	Yes	No	No	Yes	No

the learning outcome when feeding DB_1 to the PC learning algorithm.

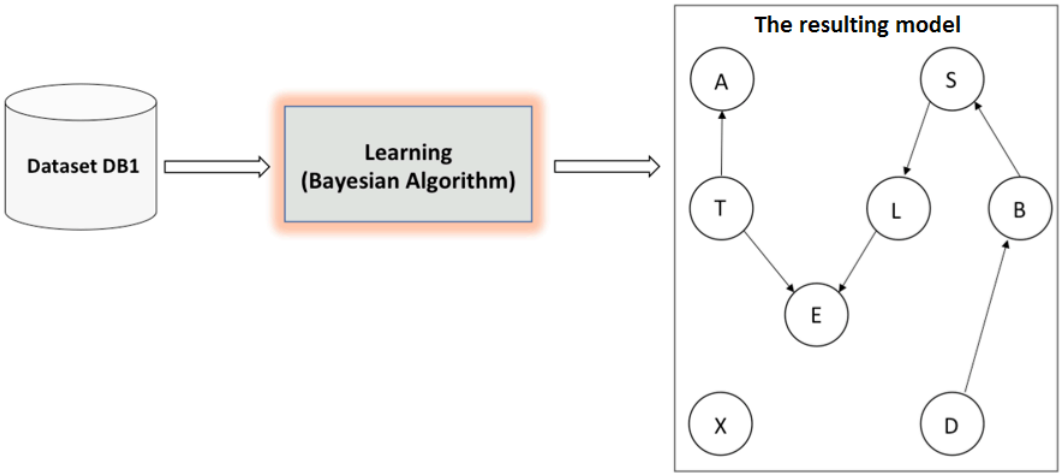


Figure 1.1: The Bayesian learning outcome when feeding DB_1 to the PC algorithm

Table 1.1 shows a sample of the original DB_1. Assume that the attacker has access to DB_1. If the attacker wants to corrupt the learned model, he/she may modify the data in DB_1. Table 1.2 shows the dataset DB_1' with changes of three data items.

Using the new corrupted dataset DB_1', the learned Bayesian model is as shown in Figure 1.2. In this model (Figure 1.2), the link from node T to node A is missing. Clearly, the attacker succeeded in corrupting the structure of the model.

Table 1.2: DB_1', which is equal to DB_1 except for three changes in bold font

X	B	D	A	S	L	T	E
No	Yes	No	**No**	No	No	Yes	No
No	No	No	No	No	No	Yes	No
Yes	No	Yes	No	No	No	No	No
No	No	No	No	No	Yes	No	No
No	No	No	No	No	No	Yes	No
No	Yes	No	Yes	No	No	**No**	Yes
...							
...							
...							
No	No	Yes	No	No	Yes	No	No
No	No	No	Yes	No	No	**No**	No

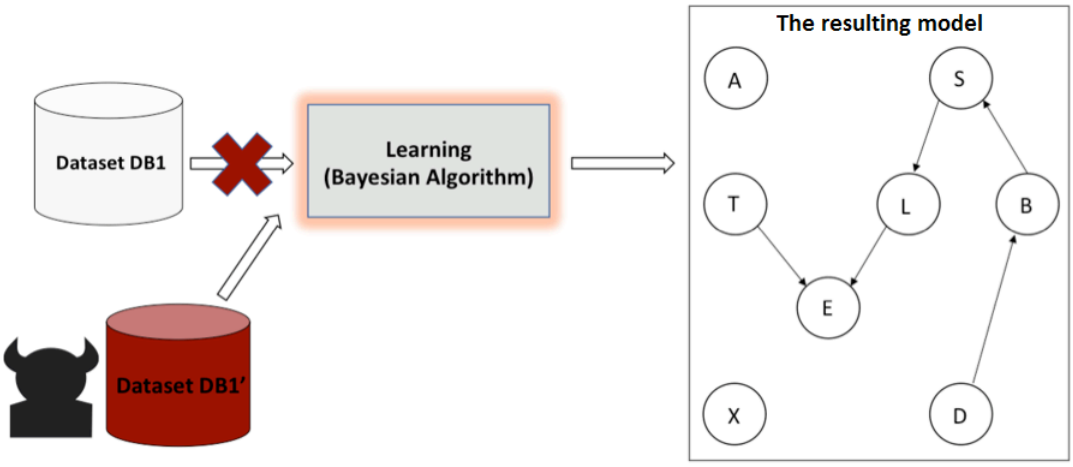


Figure 1.2: The Bayesian learning outcome when feeding DB_1 to the PC algorithm

In this dissertation, we also aim to define machine learning security best practices with the goal of detecting and preventing these types of attacks. Succeeding in building a good defensive measure against these attacks will advance the research field of adversarial machine learning and minimize the risk of data poisoning attacks, which is one of the most important emerging security threats.

The main contributions of this dissertation are as follows: we propose two subclasses of data poisoning attacks against Bayesian network structure learning algorithms: (1) Model invalidation attacks when an adversary poisons the training dataset such that the Bayesian

network model will be invalid, and (2) Targeted change attacks when an adversary poisons the training dataset to achieve a specific change in the learned structure. We define a novel measure of strengths of links between variables in discrete Bayesian networks. We show how to use this measure to evaluate the robustness of Bayesian network models. That is, we use our link strength measure to find the easiest links to break and the most believable links to add to a given Bayesian network model. In addition to traditional one-step data poisoning attacks, we define long-duration data poisoning attacks when an attacker may spread the malicious workload over a period of time. We propose a 2-layered framework to detect data poisoning attacks against Bayesian network structure learning algorithms. Our 2-layered framework detects both one-step and long-duration data poisoning attacks. We use the distance between Bayesian network models, B_1 and B_2, denoted as $\mathbf{ds}(B_1, B_2)$, to detect malicious data input (Equation 2.3) for one-step attacks. For long-duration attacks, we use the value of data conflict (Equation 2.5) to detect potentially poisoned data. Our framework relies on offline analysis to validate the potentially malicious datasets.

We implement our approaches and apply them to the Chest Clinic Network. Our empirical results show that Bayesian network structure learning algorithms are vulnerable to data poisoning attacks. Moreover, even a small number of adversarial data may be sufficient to corrupt the model. We show the effectiveness of our framework to detect both one-step and long-duration attacks. Our results indicate that the distance measure $\mathbf{ds}(B_1, B_2)$ (Equation 2.3) and the conflict measure $Conf(c, B_1)$ (Equation 2.5) are sensitive to poisoned data.

1.2 Running Example and Test Setup

In this dissertation, we demonstrate the robustness of Bayesian network structure learning algorithms against the proposed data poisoning attacks. We also develop detection methods against such adversarial attacks. The feasibility of such attacks and detection methods is investigated through empirical results on the Chest Clinic Network [34].

To set up the test, we first present a canonical Bayesian network, the Chest Clinic Network (also called Visit to Asia network). The Chest Clinic Network was created by Lauritzen and Spielgelhalter in 1988 [34]. As shown in Figure 1.3, Visit to Asia is a simple, fictitious network that could be used at a clinic to diagnose arriving patients. It consists of 8 nodes and 8 edges. The nodes are as follows:

1) *(node A)* shows whether the patient lately visited Asia;

2) *(node S)* shows if the patient is a smoker;

3) *(node T)* shows if the patient has Tuberculosis;

4) *(node L)* shows if the patient has lung cancer;

5) *(node B)* shows if the patient has Bronchitis;

6) *(node E)* shows if the patient has either Tuberculosis or lung cancer;

7) *(node X)* shows whether the patient X-ray is abnormal; and

8) *(node D)* shows if the patient has Dyspnea.

The edges indicate the causal relations between the nodes. A simple example for a causal relation is: Visiting Asia may cause Tuberculosis and so on. Lauritzen and Spielgelhalter's complete description of this simple network is as follows:

> 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 [34].

We implemented the Chest Clinic Network using *Hugin™ Research 8.1*. Then we simulated dataset of $10,000$ cases for our experiments by using *Hugin™ case generator* [38, 49]. We call this dataset DB_1. Using the PC algorithm on dataset DB_1 with 0.05 significance setting [38], the resulting structure is given in Figure 1.4. Also, Using the LCD algorithm on dataset DB_1 with 0.05 significance setting [38], the resulting structure is given in Figure 1.5. While the networks that were learned by the PC and LCD algorithms belong to different Markov equivalence classes than the original Chest Clinic Network, we will use these networks of Figure 1.4 and Figure 1.5 as the starting points of our experiments.

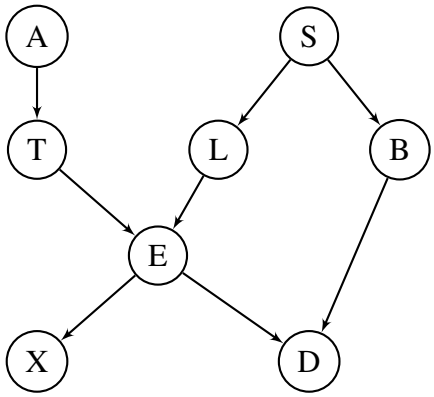


Figure 1.3: The original Chest Clinic Network.

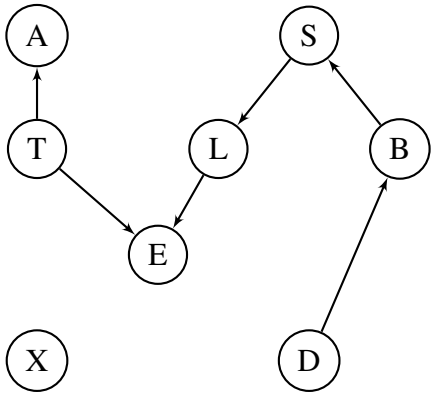


Figure 1.4: B_1, the result of feeding DB_1 to the PC algorithm with significance level at 0.05

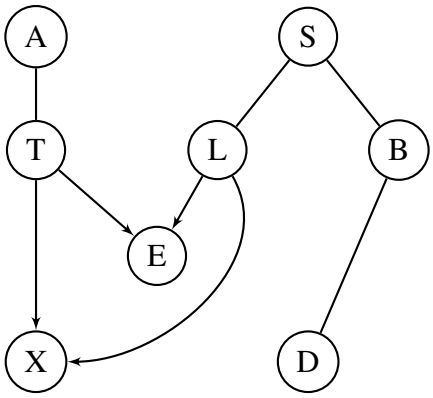


Figure 1.5: The Bayesian network model B_3, the result of feeding DB_1 to the LCD algorithm with significance level at 0.05

It is important to point out that proposed attacks require the existence of a triple in the attacked Bayesian network model and their ease depends on the link strength measure. Insertion or removal of edges in Bayesian networks is restricted by the topology of the model. For example, for shielding a collider, it is necessary to insert an edge between its parents. However, attacks must not violate the requirement that a Bayesian network is define as a directed acyclic graph. For example, we cannot insert a new edge from E to S in the model B_1 because it would create a cycle. We will use link strength measures as a security analysis tool for checking the feasibility of the proposed attacks. Another important note is that proposed data poisoning attacks may influence the decision making process that uses the poisoned model. For example, an attack on the Chest Clinic Network that aims to mask the edge from smoking, node S, to lung cancer, node L, may impact decision making as the decision maker will no longer believe that smoking is a cause of lung cancer. However, analysis of the impact on high-level (abstract) decision making needs further evaluation. It is not the purpose of this dissertation.

1.3 Research Tasks

The goals of this dissertation is to address the following major research tasks:

1. Adversarial Attacks against Bayesian Networks - the goal of this research task is to determine if adversarial attacks against Bayesian networks exist. The following subtasks have been completed:

   a) Research publications.

   b) Define two subclasses of data poisoning attacks against Bayesian network models.

   c) Develop the threat model.

   - Completed: 1a, 1b, 1c
   - Remaining: None
   - Emad Alsuwat, Marco Valtorta, and Csilla Farkas, *Bayesian structure learning attacks*, Tech. report, University of South Carolina, SC, USA, 2018.

2. Link Strength Measure in Discrete Bayesian Networks - the goal of this research task is to define a new link strength measure between random variables in discrete Bayesian networks. The following subtasks have been completed:

   a) Research and study existing link strength measures.

   b) Propose a new link strength measure definition.

   c) Test our proposed definition of link strength.

   d) Implement our link strength measure and establish the results.

   e) Compare our link strength measure with existing measures.

   - Completed: 2a, 2b, 2c, 2d, 2e
   - Remaining: None
   - Emad Alsuwat, Marco Valtorta, and Csilla Farkas, *How to generate the network you want with the pc learning algorithm*, Proceedings of the 11th Workshop on Uncertainty Processing (WUPES'18), 2018, pp. 1 – 12.

3. Adversarial Attacks against Bayesian Networks - the goal of this research task is to study model invalidation attacks based on the notion of d-separation. The following subtasks have been completed:

   a) Develop model invalidation attacks based on the notion of d-separation - creating a new converging connection (v-structure).

   b) Develop an algorithm for attacks based on creating a new converging connection (v-structure).

   c) Implement the algorithm and establish the results.

   d) Develop model invalidation attacks based on the notion of d-separation - breaking an existing converging connection (v-structure).

   e) Develop an algorithm for attacks based on breaking an existing converging connection (v-structure).

   f) Implement the algorithm and establish the results.

   - Completed: 3a, 3b, 3c, 3d, 3e, 3f
   - Remaining: None
   - Emad Alsuwat, Hatim Alsuwat, Marco Valtorta, and Csilla Farkas, *Cyber attacks against the pc learning algorithm*, 2nd International Workshop on A.I. in Security, 2018, pp. 19 – 35.
   - Emad Alsuwat, Marco Valtorta, and Csilla Farkas, *Bayesian structure learning attacks*, Tech. report, University of South Carolina, SC, USA, 2018.

4. Adversarial Attacks against Bayesian Networks - the goal of this research task is to study model invalidation attacks based on marginal independence tests. The following subtasks have been completed:

   a) Develop model invalidation attacks based on marginal independence tests - removing the weakest edge.

b) Develop an algorithm for attacks based on removing the weakest edge.

c) Implement the algorithm and establish the results.

d) Develop model invalidation attacks based on marginal independence tests - breaking an existing converging connection (v-structure).

e) Develop an algorithm for attacks based on adding the most believable yet incorrect edge.

f) Implement the algorithm and establish the results.

- Completed: 4a, 4b, 4c, 4d, 4e, 4f
- Remaining: None
- Emad Alsuwat, Hatim Alsuwat, Marco Valtorta, and Csilla Farkas, *Cyber attacks against the pc learning algorithm*, 2nd International Workshop on A.I. in Security, 2018, pp. 19 – 35.
- Emad Alsuwat, Marco Valtorta, and Csilla Farkas, *Bayesian structure learning attacks*, Tech. report, University of South Carolina, SC, USA, 2018.

5. Adversarial Attacks against Bayesian Networks - the goal of this research task is to study targeted change attacks. The following subtasks have been completed:

a) Develop targeted change attacks.

b) Develop an algorithm for attacks based on a specific goal.

c) Implement the algorithm and establish the results.

- Completed: 5a, 5b, 5c
- Remaining: None
- Emad Alsuwat, Hatim Alsuwat, Marco Valtorta, and Csilla Farkas, *Cyber attacks against the pc learning algorithm*, 2nd International Workshop on A.I. in Security, 2018, pp. 19 – 35.

- Emad Alsuwat, Hatim Alsuwat, Marco Valtorta, and Csilla Farkas, *Data poisoning attacks against Bayesian network structure learning algorithms*, International Journal of General Systems, 2019, pp. 1-29.

6. Adversarial attacks against the LCD algorithm- the goal of this research task is to use our link strength measure to evaluate the robustness of the LCD algorithm against model invalidation attacks. The following subtasks have been completed:

   a) Study the LCD algorithm thoroughly.

   b) Contact the author of the LCD algorithm to fix the R package for the LCD algorithm.

   c) Use our link strength measure to study the robustness of the LCD algorithm.

   d) Implement our experiments and establish the results.

   - Completed: 6a, 6b, 6c, 6d
   - Remaining: None

7. Adversarial Attacks against Bayesian Networks- the goal of this research task is to define long-duration data poisoning attacks against Bayesian network structure learning algorithms The following subtasks have been completed:

   a) Develop long-duration data poisoning attacks.

   b) Develop an algorithm for the defined attacks.

   c) Implement the algorithm and establish the results.

   - Completed: 7a, 7b, 7c
   - Remaining: None
   - Alsuwat, E., Alsuwat, H., Rose, J., Valtorta, M., Farkas, C.: *Long duration data poisoning attacks on Bayesian networks*. Tech. rep., University of South Carolina, SC, USA (2019)

8. Development of Detection framework for data poisoning attacks against Bayesian Networks Adversarial Attacks- the aim of this research task is to build a detective framework for detecting both one-step and long-duration data poisoning attacks against Bayesian network structure learning algorithms. The following subtasks have been completed:

   a) Research the existing defensive methods against data poisoning attacks.

   b) Identify a detective method.

   c) Build framework

   d) Develop algorithms for first and second layers of detection.

   e) Implement algorithms and establish the results.

   - Completed: 8a, 8b, 8c, 8d, 8e
   - Remaining: None
   - Alsuwat, E., Alsuwat, H., Rose, J., Valtorta, M., Farkas, C.: *Long duration data poisoning attacks on Bayesian networks*, The 33rd Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy, 2019, pp. 3-22.

1.4 Dissertation Outline

The rest of this dissertation is structured as follows:
In chapter 2, we present an overview of background information.
In chapter 3, we present an overview of the proposed system
In chapter 4, we propose a novel link strengths measure between random variables in discrete Bayesian network.
In chapter 5, we identify model invalidation attacks against the PC algorithm.
In chapter 6, we identify targeted change attacks against the PC learning algorithm.
In chapter 7, we use our proposed link strength measure to investigate the robustness of the

LCD algorithm against such attacks.

In chapter 8, we present long-duration data poisoning attacks against Bayesian network structure learning algorithms.

In chapter 9, we develop detection framework for the identified data poisoning attacks against Bayesian network structure learning algorithms.

Finally, in chapter 10, we conclude and briefly discuss future work.

CHAPTER 2

LITERATURE REVIEW

2.1 BAYESIAN NETWORKS

Bayesian Networks (BNs) are probabilistic graphical models in which vertices represent a set of random variables and arcs represent probabilistic dependencies between vertices. Formally (according to [45]), we say $BN = (G, P)$ is a Bayesian network, where $G = (V, E)$ is a direct acyclic graph (with $V = \{x_1, x_2, ..., x_n\}$ being the set of random variables or nodes, and E being the set of edges or arcs) and P is a joint probability distribution of the random variables, if it satisfies the following Markov condition: every node is conditionally independent of its non-descendants given its parents.

The following factorization of the joint probability distribution (also known as global probability distribution) of $V = \{x_1, x_2, ..., x_n\}$ into a product of local probability distributions is equivalent to the Markov property for both discrete and continuous variables, as shown in equation 2.1 and 2.2 respectively [45].

$$P(V) = \prod_{i=1}^{n} P(x_i \mid parent(x_i)) \tag{2.1}$$

$$f(V) = \prod_{i=1}^{n} f(x_i \mid parent(x_i)) \tag{2.2}$$

Example 2.1. [**Traveling Activity**]

Figure 2.1 presents a Bayesian network for a traveling activity. This example shows a discrete Bayesian network with a domain of five Boolean variables, which include and are represented as follows:

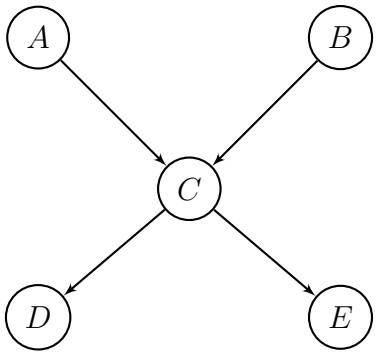


Figure 2.1: A simple BN for modeling a traveling activity

1) S – the event that it is summer time;

2) M – the event that the person has money;

3) T – the event that the person is going to travel;

4) H – the event that the person is happy; and

5) P – the even that the person is going to meet new people.

Instead of enumerating the probability distributions of the five domain variables used in figure 2.1 (25 possible combinations), We define the joint probability distribution of this Bayesian network as indicated:

$$P(S, M, T, H, P) = P(S) \times P(M) \times P(T \mid S, M) \times P(H \mid T) \times P(P \mid T)$$

2.2 The Notion of D-separation

In a Bayesian network, there are three basic connections among variables as follows [48]:

1. *Serial connections* (also called *pipelined influences*): in a serial connection (shown in figure 2.2), changes in the certainty of A will affect the certainty B, which in turn will affect the uncertainty of C. Therefore this shows information may flow from node A through B to C, unless there is evidence about B (B is known, or B is *instantiated*).

Table 2.1: Conditional probability tables for a simple BN for modeling a traveling activity

M	
True	0.9

S	
True	0.25

T				
M	False		True	
S	False	True	False	True
True	0.01	0.15	0.3	0.9

H		
T	False	True
True	0.7	0.95

P		
T	False	True
True	0.1	0.8

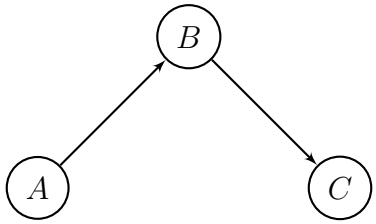


Figure 2.2: An example of a serial Connection

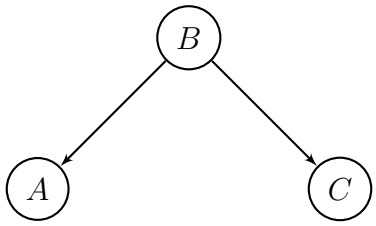


Figure 2.3: An example of a diverging connection

2. *Diverging connections*: in a diverging connection (shown in figure 2.3), changes in the certainty of A will affect the certainty B, which in turn will affect the uncertainty of C. Therefore this shows information may flow from node A through B to C, unless there is evidence about B.

3. *Converging connections* (a.k.a. *v-structure*): in a converging connection (shown in figure 2.4), changes in the certainty of A cannot affect the certainty C through B, and

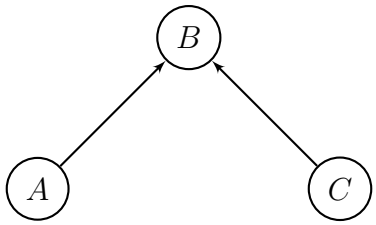


Figure 2.4: An example of a converging connection

vice versa. Therefore this shows information cannot flow between A and C through B, unless there is evidence about B.

The previously discussed three types of connections in a casual network are used in the definition of *d-separation* [48]:

Definition 2.2. (d-separation)

Two distinct variables A and B in a causal network are d-separated ("d" for "directed graph") if for all paths between A and B, there is an intermediate variable V (distinct from A and B) such that either

- the connection is serial or diverging and V is instantiated, or
- the connection is converging, and neither V nor any of V's descendants have received evidence.

2.3 Structure Learning in Bayesian Networks

There are three main approaches to learn the structure of Bayesian networks: *constraint-based*, *score-based*, or *hybrid* algorithms.

(I) Constraint-based algorithms count on conditional independence tests to determine the DAG of the learned Bayesian network. The *Inductive Causation* (IC) algorithm [64] was the first constraint-based algorithm, which introduced a framework for learning the structure of causal models. IC's framework consists of three steps as follows:

(i) Find the skeleton (all pairs of dependent variables)

(ii) Remove indirect dependencies (by defining colliders)

(iii) Complete orienting the remaining undirected edges if any (avoiding cycles).

All constraint-based algorithms, such as the PC algorithm [60, 61] and NPC algorithm [62], follow the theoretical framework introduced by the IC algorithm.

(II) Score-based algorithms, such as AIC [1], BDe [56], K2 [21], and BIC algorithm [27], assign a score for each Bayesian network structure (this score indicates how well the Bayesian network structure fits the data) and then perform a (usually greedy) search algorithm to select the structure with the highest score.

(III) Hybrid algorithms, such as CB [59] and EGS algorithm [22], rely on the idea of using both constraint-based algorithms and score-based algorithms. The use of constraint-based algorithms will reduce the search space (i.e., it will reduce the number of candidate DAGs). Thenceforth, score-based algorithms can be used to select the optimal DAG.

We will focus on the PC algorithm since it is an integral part of this paper. *The PC algorithm* (named after the authors, the first letter of their first names, **P**eter Spirtes and **C**lark Glymour) is a constraint-based algorithm for learning the structure of a Bayesian network from data. The PC algorithm follows the theoretical framework of the IC algorithm to determine the structure of causal models [57, 53]. According to [61], the process performed by the PC algorithm to learn the structure of Bayesian networks can be summarized as follows:

(i) For every pair of variables, perform statistical tests for conditional independence.

(ii) Determine the skeleton (undirected graph) of the learned structure by adding a link between every pair of statistically dependent variables.

(iii) Identify colliders (v-structures) of the learned structure (A $\rightarrow$ B $\leftarrow$ C).

(iv) Identify derived directions.

(v) Randomly, complete orienting the remaining undirected edges without creating a new collider or a cycle.

For the implementation of this paper, we used *the Hugin PC algorithm* (by $Hugin^{TM}$ *Decision Engine* [49, 38]), "which is a variant of the original PC algorithm due to [61]" [29].

2.4 Prior to Posterior Updating

Bayes' theorem is a simple mathematical formula that inverts conditional probabilities (i.e., given the conditional probability of event B given event A, how to calculate the conditional probability of event A given event B). The statement of Bayes' theorem is: For two events A and B,

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)},$$

where

(i) $P(A \mid B)$ is the conditional probability of event A given event B (called the posterior probability),

(ii) $P(B \mid A)$ is the conditional probability of event B given event A (called the likelihood),

(iii) $P(A)$ is the marginal probability of event A (called the prior probability), and

(iv) $P(B)$ is the marginal probability of event B ($P(B) > 0$) [45].

Unlike classical statistics, Bayesian statistics treats parameters as random variables whereas data is treated as fixed. For Example, let θ be a parameter, and D be a dataset, then Bayes' theorem can be expressed mathematically as follows:

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)} \tag{2.3}$$

In equation 2.3, $P(\theta \mid D)$ is the *posterior distribution*, which is the ultimate goal for Bayesian statistics since it measures the uncertainty about the parameters θ after seeing the dataset D. $P(D \mid \theta)$ is the *likelihood*, which describes how likely the dataset D is if the truth is parameter θ. $P(\theta)$ is the *prior distribution*, which is a marginal probability of our belief before seeing data. $P(D)$ is the *marginal probability* of D, which is a normalization constant to ensures that the sum of the posterior distribution sums to 1 over all values of parameter θ [36]. Thus, since $P(D)$ is constant, we can write Bayes' theorem in one of the most useful form in Bayesian update and inference as follows:

$$P(\theta \mid D) \propto P(D \mid \theta) \times P(\theta) \tag{2.4}$$

$$Posterior \propto Likelihood \times Prior \tag{2.5}$$

In Bayesian analysis, the results of the experiment could be used to update the belief about the parameter θ. In simple cases, we can compute the posterior distribution for the parameter θ by multiplying the prior distribution and the likelihood function as shown in equation 2.5. However, it is convenient mathematically for the prior and the likelihood to be conjugate. A prior distribution is a *conjugate prior* for the likelihood function if the posterior distribution belongs to the same distribution as the prior [54]. For example, the beta distribution is a conjugate prior for the binomial distribution (as a likelihood function) because the posterior distribution obtained by multiplying the prior and the likelihood belongs to the same distribution as the prior (thus, both the prior and the posterior have beta distributions).

Let's consider the effect of different priors on the posterior distribution. A completely uninformative prior is the beta distribution with parameters $\alpha = 1$ and $\beta = 1$. The posterior distribution in this case is equivalent to the likelihood function since we have a completely uninformative prior. More informative priors will have a greater influence on the posterior distribution for a given sample size. On the other hand, larger sample sizes will give the likelihood function more influence on the posterior distribution for a given prior distribution. In practice, this means that we can obtain a precise estimate of the posterior

distribution using smaller sample sizes when we use more informative priors. Similarly, we may need larger sample sizes when we use a weak or uninformative prior.

$$P(\theta \mid D) \propto Beta(\alpha, \beta) \times Binomial(n, \theta) \tag{2.6}$$

$$P(\theta \mid D) \propto Beta(y + \alpha, n - y + \beta) \tag{2.7}$$

Equation 2.7 is the formula that we are going to use in this paper for prior to posterior update. Starting with a prior distribution $Beta(\alpha, \beta)$, we add the count of successes,y, and the count of failures, $n - y$, from the dataset D (where n is total number of entries in D) to α and β, respectively. Thus, $Beta(y + \alpha, n - y + \beta)$ is the posterior distribution. For a theoretical justification of the use of the beta distribution to model parameter uncertainty, see [45].

2.5 Link Strengths in Bayesian Networks

The concept of link strength in Bayesian networks was introduced first by Boerlage in 1992 [18]. In his thesis, Boerlage introduced the concepts of both connection strength and link strength in a binary Bayesian network model. *Connection strength* for any two variables A and B in a Bayesian network model B_1 is defined as measuring the strength between these two variables by testing all possible paths between them in B_1, whereas *link strength* is defined as measuring the strength these two random variables taking into account only the direct edge $A - B$ [18]. Methods for link strengths measurements are not studied sufficiently. Imme Ebert-Uphoff in her 2009 paper [24] presented a tutorial on how to measure connection strengths and link strengths in discrete Bayesian networks. Ebert-Uphoff concluded that there is a limited literature on link strengths, and there is more need to apply and use link strengths measures in structure learning and other purposes [24]. However, to the authors' best knowledge, there are no more recent publications that address link strengths measurements in discrete Bayesian networks. In this paper, we define a novel and not computationally expensive link strengths measure in discrete Bayesian networks.

2.6 ADVERSARIAL MACHINE LEARNING

Adversarial machine learning is the research field that studies the design of efficient machine learning algorithms in adversarial environments [28]. Attacks against machine learning systems have been organized by [11, 10, 28] according to three features: Influence, Security Violation, and Specificity. First, influence of the attacks on machine learning models can be either causative or exploratory. Causative attacks aim to corrupt the training data whereas exploratory attacks aim to corrupt the classifier at test time. Second, security violation of machine learning models can be a violation of integrity, availability, or privacy. An integrity violation is an attack that aims to misclassify false positives with the goal of gaining unauthorized access to the system. An availability violation is an attack that aims to misclassify both false positives and false negatives and leads to denial of service. A privacy violation is an attack in which an adversary is able to reap confidential information from a machine learning model. Third, specificity of the attacks against machine learning models can be either targeted, or indiscriminate. Targeted attacks aim to corrupt machine learning models to misclassify a particular class of false positives whereas indiscriminate attacks have the goal of misclassifying all false positives.

Evasion attacks [63, 13, 26, 33, 31] and Data poisoning attacks [16, 41, 40, 2] are two of the most common attacks on machine learning systems [28]. Evasion attacks are exploratory attacks at the testing phase. In an evasion attack, an adversary attempts to pollute the data for testing the machine learning classifier; thus causing the classifier to misclassify adversarial examples as legitimate ones. Data poisoning attacks are causative attacks, in which an adversary attempts to corrupt the machine learning classifier itself by contaminating the data on training phase.

In this dissertation, we study the resilience of two commonly used Bayesian network algorithms, namely the PC algorithm and the LCD algorithm, against data poisoning attacks. Since no study has been performed on evaluating the vulnerabilities of these algorithms against poisoning attacks, we will just explore the line of data poisoning research on dif-

ferent machine learning fields.

There has been a long line of work on poisoning attacks of support vector machines (SVMs) [16, 66, 67, 47, 42, 19, 32]. In Neural Networks (NNs), there has been a recent study of data poisoning attacks in which the authors investigated the process of data generation poisoning and proposed two poisoning methods, including a direct gradient method and a generative method [69].

2.7 Defenses and Countermeasures for Data Poisoning attacks

In this section, we will give a brief overview of adversarial machine learning research; focusing on data poisoning. Recent surveys on adversarial machine learning can be found in [10, 25, 35].

2.7.1 Data Poisoning Attacks

As machine learning algorithms have been widely used in security-critical settings such as spam filtering and intrusion detection, adversarial machine learning has become an emerging field of study. Attacks against machine learning systems have been organized by [11, 10, 28] according to three features: Influence, Security Violation, and Specificity. Influence of the attacks on machine learning models can be either causative or exploratory. Causative attacks aim to corrupt the training data whereas exploratory attacks aim to corrupt the classifier at test time. Security violation of machine learning models can be a violation of integrity, availability, or privacy. Specificity of the attacks can be either targeted or indiscriminate. Targeted attacks aim to corrupt machine learning models to misclassify a particular class of false positives whereas indiscriminate attacks have the goal of misclassifying all false positives.

Evasion attacks and Data poisoning attacks are two of the most common attacks on machine learning systems [28]. Evasion attacks [26, 33, 31] are exploratory attacks at the testing phase. In an evasion attack, an adversary attempts to pollute the data for testing the

machine learning classifier; thus causing the classifier to misclassify adversarial examples as legitimate ones. Data poisoning attacks [40, 2, 16, 42, 32, 69] are causative attacks, in which adversaries attempt to corrupt the machine learning classifier itself by contaminating the data in the training phase.

Data poisoning attacks have been studied extensively during the last decade [43, 6, 16, 42, 32, 15, 14, 17, 12, 69]. However, attacks against Bayesian network algorithm have not been studied. In our previous work, we were addressed data poisoning attacks against Bayesian network algorithms [8, 9, 6]. We studied how an adversary could corrupt the Bayesian network structure learning algorithms by inserting contaminated data into the training phase. We showed how our novel measure of strengths of links for Bayesian networks [9] can be used to do a security analysis of attacks against Bayesian network structure learning algorithms. However, our approach did not consider long-duration attacks.

2.7.2 Defenses and Countermeasures

Data sanitization is a best practice for security optimization in the adversarial machine learning context [20]. It is often impossible to validate every data source. In the event of a poisoning attack, data sanitization adds a layer of protection for training data by removing contaminated samples from the targeted training data set prior to training a classifier. Reject on Negative Impact is one of the widely used method for data sanitization [10, 20, 35]. Reject on Negative Impact defense assesses the impact of new training sample additions, opting to remover or discard samples that yield significant, negative effects on the observed learning outcomes or classification accuracy [10, 20]. The base training set is used to train a classifier, after which, the new training instance is added and a second classifier is trained [10]. In this approach, classification performance is evaluated by comparing error rates (accuracy) between the original and the new, retrained classifier resulting from new sample integration [35]. As such, if new classification errors are substantially higher

compared to the original or baseline classifier, it is assumed that the newly added samples are malicious or contaminated and are therefore removed in order to maximize and protect classification accuracy [10].

CHAPTER 3

OVERVIEW OF THE PROPOSED SYSTEM

3.1 OVERVIEW OF ADVERSARIAL ATTACKS AGAINST BAYESIAN NETWORK MODELS

Data integrity is a key requirement for correct machine learning applications, such as Bayesian network structure learning algorithms. In this research, we study how an adversary could corrupt the PC structure learning algorithm. An attacker may attempt to corrupt the machine learning model by poisoning the input dataset with the ultimate goal of influencing the output model. In this research, we propose a threat model to investigate both attacks that aim to arbitrarily invalidate the learning outcome and attacks that aim to achieve a specific goal. We use this threat model to study the resilience of Bayesian network algorithms, namely the PC algorithm, against data poisoning attacks.

Like all security problems, the problem of adversarial attacks against Bayesian networks is to design a security prevention and detection model against these attacks. Our ongoing work is about developing prevention methods against these defined attacks.

3.2 THREAT MODEL FOR DATA POISONING ATTACKS AGAINST THE PC ALGORITHM

In this section, we present the general framework of how attackers can use exploratory attacks to corrupt the learned Bayesian model by the PC algorithm. The attacker first uses the PC algorithm to learn the structure of the Bayesian network model. If the learned structure is what the adversarial opponent wants, then the “poisoned" dataset DB_2 is pro-

duced. Otherwise, the user adds contaminated cases to the learning dataset and relearn the Bayesian model using the PC algorithm until the desired model is obtained. This process is illustrated in Figure 3.1.

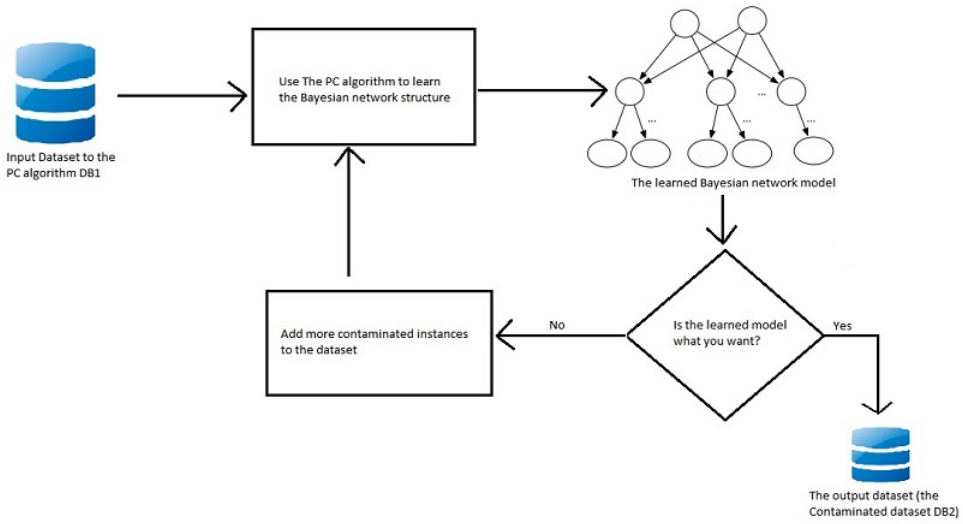


Figure 3.1: Overview of how data poisoning attacks against Bayesian network structure learning algorithms work.

In this dissertation, we study the resilience of two of the most commonly used Bayesian network algorithms, namely the PC algorithm and the LCD algorithm, against data poisoning attacks. To the authors' best knowledge, no study has been performed on evaluating the vulnerabilities of Bayesian network structure learning algorithms against poisoning attacks. We present the two subclasses of data poisoning attacks against the Bayesian network algorithms: 1) Model invalidation attacks and 2) Targeted change attacks.

CHAPTER 4

LINK STRENGTHS FROM DATA IN DISCRETE BAYESIAN NETWORKS

4.1 INTRODUCTION

We introduce a novel link strengths measure between two random variables in a discrete Bayesian network model (denoted as L_S). It is essential to not only study the existence of a link in a causal model but also define a reliable link strengths measure that is useful in Bayesian reasoning [18, 24]. The new defined link strengths measure assigns a number to every link in a Bayesian network model. This number represents the lowest confidence of all possible combinations of assignments of posterior distributions. The defined link strengths measure will be used to rank edges from the most to the least believable edge, rank edges from the weakest to the strongest edge, and justify a plausible process in any causal model.

4.2 DEFINITION OF THE PROPOSED LINK STRENGTHS MEASURE (L_S)

In this section, we present the definition of our new link strength measure (we named it L_S). Our novel approach is as follows:

Definition 4.1. The link strengths measure (L_S) is defined as

$$L\_S(Variable_1 \rightarrow Variable_2) = \min_{y \in Y}(pdf(\frac{y+\alpha}{\alpha+n+\beta}); \alpha, \beta, y, n) \tag{4.1}$$

where $Y = \{n_{11}, n_{12}, \ldots, n_{1j}, n_{21}, n_{22}, \ldots, n_{2j}, \ldots, n_{i1}, n_{i2}, \ldots, n_{ij}\}$, *pdf* is the probability density function, and $\frac{y+\alpha}{\alpha+n+\beta}$ is the mean of the posterior distribution.

4.3 EXPLANATION

Given a discrete dataset DB_1 and a Bayesian network structure B_1 learned by the PC algorithm using DB_1, for every link $Variable_1 \rightarrow Variable_2$ in B_1, build a contingency table [39] for the two discrete variables $Variable_1$ and $Variable_2$ with i and j states, respectively (as shown in table 4.1).

Table 4.1: A contingency table for two discrete variables $Variable_1$ and $Variable_2$ with i and j states, respectively.

	Variable₂			
Variable₁	**State$_1$**	$\cdots$	**State$_j$**	**Observed Row Total**
State$_1$	$[n_{11}], (e_{11}), < ts_{11} >$	$\cdots$	$[n_{1j}], (e_{1j}), < ts_{1j} >$	$\sum_{t=1}^{j} n_{1t}$
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
State$_i$	$[n_{i1}], (e_{i1}), < ts_{i1} >$	$\cdots$	$[n_{ij}], (e_{ij}), < ts_{ij} >$	$\sum_{t=1}^{j} n_{it}$
Observed Column Total	$\sum_{t=1}^{i} n_{t1}$	$\cdots$	$\sum_{t=1}^{i} n_{tj}$	n (Observed Grand Total)

The above contingency table (Table 4.1) is structured as follows:

1. $[n_{ij}]$ is the cell's observed counts obtained from dataset DB_1,

2. (e_{ij}) is the cell's expected counts, calculated as follows:

$$\frac{\textit{Observed Row Total} \times \textit{Observed Column Total}}{\textit{Observed Grand Total (denoted as n)}}$$

3. $< ts_{ij} >$ is the cell's chi-square test statistic, calculated as follows:

$$\frac{(n_{ij} - e_{ij})^2}{e_{ij}}$$

To measure the strength of links of a causal model, we perform the following two steps:

(1) We compute the posterior distributions for each link $Variable_1 \rightarrow Variable_2$ as follows:

$$P(Variable_2 \mid Variable_1) = Beta(y + \alpha, n - y + \beta),$$

where $Variable_2 \mid Variable_1$ is all possible combinations of discrete states of $Variable_2$ and $Variable_1$, and then

(2) We use our link strengths measure as presented in equation 4.1.

Note that $\frac{y+\alpha}{\alpha+n+\beta}$ in equation 4.1 is obtained by simply substituting α with $y+\alpha$ and β with $n-y+\beta$ in $\frac{\alpha}{\alpha+\beta}$.

4.4 INTERPRETATION

For any two random variables in a causal model ($Variable_1$ with i states and $Variable_2$ with j states), there are $i \times j$ combinations of assignments of posterior distributions. For every posterior distribution, we have a prior distribution that is a conjugate prior for the likelihood function. For instance, a posterior distribution in the form $Beta(y+\alpha, n-y+\beta)$ has a Beta-distributed prior, $Beta(\alpha, \beta)$, which is a conjugate prior for the likelihood function, $Binomial(n, \theta)$. Considering all $i \times j$ posterior distributions for the two random $Variable_1$ and $Variable_2$, we can measure the uncertainty of that link by measuring how peaked the posterior distributions (Beta distributions in our experiments) are; thus, we can identify the link strength based on the uncertainty level. The more peaked the posterior distribution is, the more certainty we have about the posterior distribution probability. The peak of a beta distribution, $Beta(\alpha', \beta')$, is reached at its mean, $\frac{\alpha'}{\alpha'+\beta'}$. Thus, the peak of the posterior distribution is reached at $\frac{y-\alpha}{n-y+\beta}$. In the defined link strengths measure, we define the link strength for any link between two random variables in a causal model as the value of the smallest peak. This point is the point at which the model has seen the fewest number of cases; thus, it is the most critical point through which this link can be manipulated.

4.5 PRACTICAL USAGES

We use this measure to identify weak edges (i.e., low values of L_S). These edges are the easiest to remove from a given causal model. We also use the L_S value to identify location

Table 4.2: Posterior distributions for the Chest Clinic Network.

Link	Posterior Distributions (Beta Distributions)
P(T | A)	Beta(10,99) Beta(106,9789) Beta(99,10) Beta(9789,106)
P(L | S)	Beta(481,4510) Beta(47,4966) Beta(4510,481) Beta(4966,47)
P(B | S)	Beta(3019,1972) Beta(1514,3899) Beta(1972,3019) Beta(3899,1514)
P(E | T)	Beta(115,1) Beta(523,9365) Beta(1,115) Beta(9365,523)
P(E | L)	Beta(527,1) Beta(111,9365) Beta(1,527) Beta(9365,111)
P(D | B)	Beta(3638,895) Beta(725,4746) Beta(895,3638) Beta(4746,725)
P(D | E)	Beta(520,118) Beta(3843,5523) Beta(118,520) Beta(5523,3843)
P(X | E)	Beta(624,14) Beta(454,8912) Beta(14,624) Beta(8912,454)

for new edges to be added. We claim that the highest L_S value, the most believable the new edge is.

4.6 Experimental Results

In this section, we will evaluate the proposed link strength measure (L_S) on the original Chest Clinic Network. Given the Chest Clinic network model as shown in Figure 1.3 and the dataset DB_1, we followed the *two steps* presented in section 4.

Table 4.2 contains the posterior distributions (Beta Distributions) calculated in *step 1* as follows:

Figure 4.1 shows the final link strength evaluation (L_S) which is calculated in *step 2* as follows:

We observe that the edge $T \rightarrow A$ is the weakest edge in Chest Clinic network with the score 14.75256. Also, we can see that the edge $E \rightarrow D$ is the second weakest edge with the score 25.73502 and so on. The strongest edge in Chest Clinic network is the edge $L \rightarrow E$ with the score 129.2983.

4.7 Comparison with Previous Link Strength Measures

In this section, we will compare our link strength measure (L_S) with Mutual Information link strengths measure. Shannon in [58] introduced the concept of Mutual Information

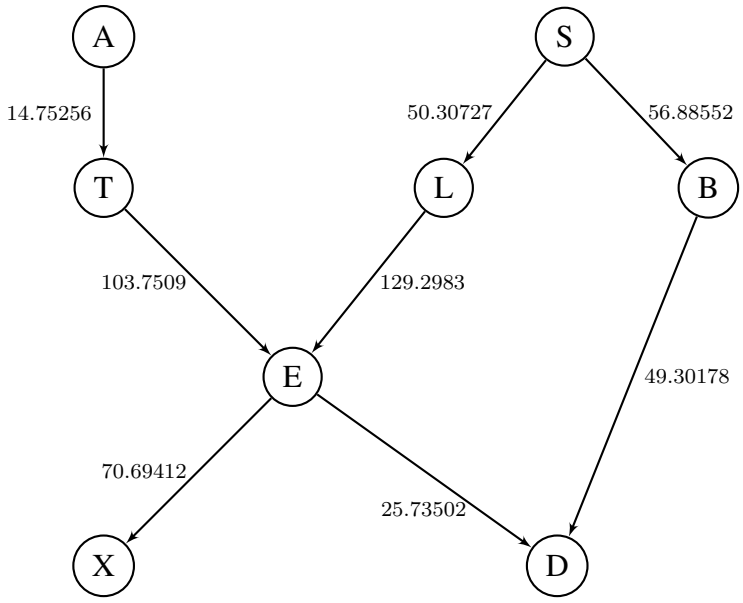


Figure 4.1: Results of L_S on the Chest Clinic Network.

(MI) in the context of communication theory and Pearl in [52] proposed its expanded use to measure connection strength in Bayesian Networks; it is defined as:

$$MI(X,Y) = \sum_{x,y} P(x,y) log_2(\frac{P(x,y)}{P(x)P(y)}) \tag{4.2}$$

MI measures the how edge in a causal model are related to each others by (1) detecting any sort of relationship and (2) employing straightforward interpretation of the amount of data shared between the datasets (3) while remaining insensitive to dataset size, as characteristic of p-value testing [55]. This simplified MI calculation reflects and measures connection strength between X and Y based on the degree or strength of influence the state of X affects the state of Y through the comparison of $U(Y)$ and $U(Y|X)$. Put another way, the MI formula seeks to determine the amount of uncertainty in Y that can be reduced by knowledge of state of X if nothing else is known [24]. Therefore, the MI between two datasets (X and Y) is typically estimated from statistical analysis of the (x, y) pairs between the two datasets [55].

The following table (Table 4.3) presents the results of using our link strength (L_S) and MI link strength to compute strengths of links of the Chest Clink Network. Note that, we

rank the edge in the column *rank* from the weakest to the strongest edge. For more technical details about how to use MI link strength measure, we refer the reader to Appendix C.

Table 4.3: Using L_S and MI to compute link strength of the original Chest Clinic Network

Link	Our Link strength Measure L_S		Mutual Information MI	
	Score	Rank	Score	Rank
$A \rightarrow T$	14.75256	1	0.0006	1
$S \rightarrow L$	50.30727	4	0.0303485	4
$S \rightarrow B$	56.88552	5	0.06665	5
$T \rightarrow E$	103.7509	7	0.0296	3
$L \rightarrow E$	129.2983	8	0.2675	7
$E \rightarrow D$	25.73502	2	0.02575	2
$B \rightarrow D$	49.30178	3	0.3508	8
$E \rightarrow X$	70.69412	6	0.2236	6

Both link strengths measures agree on the fact that the edge $A \rightarrow T$ is the weakest link in the Chest Clinic Network. However, our link strength measure functions better since it is able to identify the deterministic edges. That is, deterministic edges $T \rightarrow E$ and $L \rightarrow E$ are hard edges to break. In addition, MI measure computes the link strength measure using the conditional probability tables whereas our link strength measure uses a given dataset to compute the strengths of links, which makes our link strength efficient for security application as it is sensitive to changes in data.

CHAPTER 5

MODEL INVALIDATION ATTACKS

5.1 OVERVIEW OF MODEL INVALIDATION ATTACKS

A model invalidation attack against the PC algorithm is a malicious active attack in which adversarial opponents try to corrupt the original model in any way. We demonstrate adversarial attacks to decrease the validation status of the model using the least number of changes.

In such an event, adversaries create some formal disturbance in the model. For example, they will try to add imprecise or incorrect data to change the model validation status so that the model is rendered invalid. We distinguish between two ways to invalidate Bayesian network models:

1) Attacks based on the notion of d-separation and

2) Attacks based on marginal independence tests.

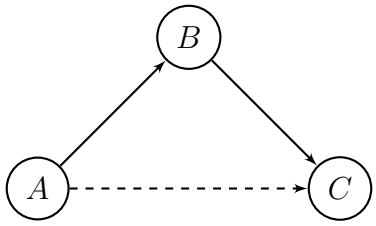
In what follows, we present an item list and short description for all the algorithms that are going to be presented in this chapter of the dissertation:

Algorithm	Description
Algorithm 1	Creating a New Converging Connection
Algorithm 2	Breaking an Existing Converging Connection
Algorithm 3	Edge Deleting
Algorithm 4	Removing a Weak Edge
Algorithm 5	Edge adding
Algorithm 6	Adding the Most Believable yet Incorrect Edge

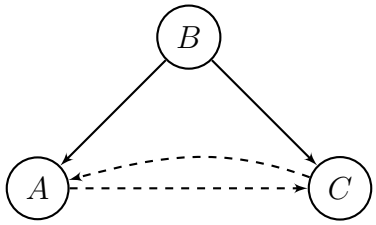
5.2 Model Invalidation Attacks Based on the Notion of D-separation

Based on the definition of d-separation, adversaries may attempt to introduce a new link in any triple $(A - B - C)$ in the BN model. This newly inserted link $(A - C)$ will introduce a v-structure in the Bayesian model, thus change the independence relations.

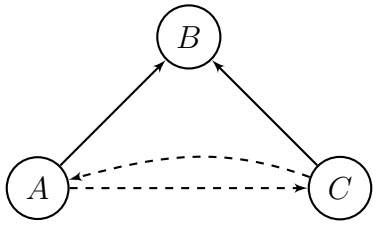
Theorem 5.1. *Let B_1 and B_2 be two Markov equivalent BNs, and let $< A, B, C >$ be a path in B_1. If a new link is added to B_1 creating B_1', then B_1' and B_2 are not Markov equivalent.*



(a) Adding the dashed link to the serial connection.



(b) Adding one of the dashed links to the diverging connection.



(c) Adding one of the dashed links and shielding collider B.

Figure 5.1: Three cases for the proof of Theorem 5.1.

Proof Sketch. Adding a new edge to the path $< A, B, C >$ in Bayesian network model B_1 affects the Markov equivalence class of B_1 (two Bayesian networks are *Markov equivalent* if and only if they have the same skeleton and the same v-structures (unshielded colliders) [3]). Any sound learning algorithm will try to avoid the occurrence of a cycle; thus, in

the triple $(A - B - C)$, either an existing collider is shielded, and a new link is introduced (as shown in Figure 5.1c) or a new link is added (as shown in Figures 5.1a and 5.1b). In either case, the Markov equivalence class of B_1 will be violated. □

Within model invalidation attacks based on the notion of d-separation, we can further identify two subclasses:

5.2.1 Creating a New Converging Connection (V-structure)

Adversarial attackers can corrupt Bayesian network models by introducing a new converging connection. Adversaries will attempt to poison the learning dataset with the goal of introducing a new v-structure by adding a new link to any serial or diverging connection in Bayesian network models. Adding such an edge will not only introduce a new collider but also change the equivalence class of the learned Bayesian network model.

Theorem 5.2. *Let B_1 be a Bayesian network model, and let $< A, B, C >$ be a path in B_1 with either a serial connection or diverging connection, then introducing a new edge on the path $< A, B, C >$* must *create a new converging connection in B_1.*

Proof Sketch. Trivially follows. [See figures 5.1a and 5.1b]. □

We have developed Algorithm 1 to test the resilience of the PC learning algorithm against this type of attacks. It checks the feasibility of poisoning a given dataset DB_1 with the ultimate goal of introducing a new converging connection (v-structure) in the learned Bayesian network model from DB_1.

Let n be the number of cases in DB_1, and let β be *data poisoning rate* at which we are allowed to add new "contaminated" cases to DB_1 (we default set $\beta \leq 0.05$); Algorithm 1 presents algorithmic details of data poisoning attacks that aim to introduce a new v-structure in a Bayesian model as follows:

Algorithm 1 starts by learning the structure of the Bayesian network model B_1 from dataset DB_1. The poisoned dataset DB_2 is initialized to DB_1. Then the algorithm tests

Algorithm 1: Creating a New Converging Connection Procedure

Input : Dataset DB_1 ▷ Original dataset with n cases
Output: Contaminated dataset DB_2 or a failure message

Procedure `New Converging Connection(`DB_1`)`
  Use the PC algorithm for learning the structure of Bayesian network model B_1 from dataset DB_1 (using the default significance level at 0.05 [38]);
  Let DB_2 = DB_1;
  for *Every serial or diverging connection triple ($A - B - C$) in B_1 where A and C are d-connected by B* **do**
    Construct the $i \times j$ the contingency table for variables A and C;
    for $K = 1, 2, ..., \beta \times n$ ▷ *where $\beta \times n$ is the maximum number of contaminated tuples to be added to DB_1* **do**
      In the contingency table for the link $A - C$, determine the cell that has the highest test statistics value ($cell_{ij}$);
      DB_2 for adding the link $A - C$ = DB_2 + $\boldsymbol{d}$ ▷ where $\boldsymbol{d}$ is an instance in which $A = state\ i$, $C = state\ j$, and all other variables in $\boldsymbol{d}$ are No;
      Run the PC algorithm on DB_2 to learn the structure of the Bayesian network;
      if *There is a new edge between vertex A and C* **then**
        Return BD_2 ;
        Let DB_2 = DB_1;
        Continue to test the next triple;
      end
      if $K = \beta \times n$ **then**
        Return msg "Failed to introduce the link $A - C$ within a feasible number of cases" ;
        Let $DB_2 = DB_1$;
        Continue to test the next triple;
      end
    end
  end
end

every serial and diverging triple $(A - B - C)$ in B_1 for the feasibility of adding a new link to that triple (this is accomplished by adding case-by-case to the attack dataset DB_2, and then applying the PC algorithm every time a new case is added to DB_2).

Algorithm 1 tests the feasibility of adding new links to every serial and diverging connections in B_1. For each serial and diverging triple in B_1, the algorithm terminates if it succeeds in adding a new link or just prints a failure message if the number of added cases is more than $\beta \times n$. Our empirical results are given in section 5.4.

5.2.2 Breaking an Existing Converging Connection (V-structure)

Adversaries can exploit Bayesian network models by breaking an existing converging connection. The PC algorithm starts by identifying *unshielded colliders* (v-structure with unmarried parents) when learning the Bayesian network structure from data [61]; therefore, attacking v-structures will make a significant corruption to the learned BN structures since the learned model will have a different equivalence class than the expected one. Such an adversarial attack can be done by marrying the parents of an unshielded collider. Note that, if vertex B is an *unshielded collider* on the path $< A, B, C >$, then A and C are independent unconditionally, but are dependent conditionally on B in most cases (faithfulness assumption [61]).

Theorem 5.3. *Let B_1 be a Bayesian network model, and let B be an unshielded collider on the path $< A, B, C >$, then introducing a new edge on the path $< A, B, C >$* must *break the existing converging unshielded connection at vertex B.*

Proof Sketch. Trivially follows. [See figure 5.1c]. □

We have developed Algorithm 2 to check the robustness of the PC algorithm against the feasibility of shielding an existing converging connection. Given a dataset DB_1, Algorithm 2 tests the feasibility of shielding an existing converging connection.

Let n be the number of cases in DB_1, and let β be *data poisoning rate* at which we are allowed to add new "poisoned" cases to DB_1 (we default set $\beta \leq 0.05$); Algorithm 2 presents the following algorithmic details of data poisoning attacks that aim to check the feasibility of breaking an existing unshielded collider:

Algorithm 2: Breaking an Existing Converging Connection Procedure

Input : Dataset DB_1 ▷ Original dataset with n cases
Output: Contaminated dataset DB_2 or a failure message

Procedure `Breaking a Converging Connection(`DB_1`)`
  Use the PC algorithm for learning the structure of Bayesian network model B_1 from dataset DB_1 (using the default significance level at 0.05 [38]);
  Let DB_2 = DB_1;
  for *Every converging connection triple* $(A - B - C)$ *in* B_1 *where* A *and* C *are d-separated by* B **do**
    Construct the $i \times j$ the contingency table for variables A and C;
    for $K = 1, 2, ..., \beta \times n$ ▷ *where* $\beta \times n$ *is the maximum number of contaminated tuples to be added to* DB_1 **do**
      In the contingency table for the link $A - C$, determine the cell that has the highest test statistics value ($cell_{ij}$);
      DB_2 for adding the link $A - C$ = DB_2 + ***d*** ▷ where ***d*** is an instance in which $A = state\ i, C = state\ j$, and all other variables in ***d*** are No;
      Run the PC algorithm on DB_2 to learn the structure of the Bayesian network;
      if *There is a new edge between vertex* A *and* C **then**
        Return BD_2 ;
        Let DB_2 = DB_1;
        Continue to test the next triple;
      end
      if $K = \beta \times n$ **then**
        Return msg "Failed to introduce the link $A - C$ within a feasible number of cases" ;
        Let $DB_2 = DB_1$;
        Continue to test the next triple;
      end
    end
  end
end

Algorithm 2 starts by learning the structure of the Bayesian network model B_1 from dataset DB_1. The poisoned dataset DB_2 is initialized to DB_1. Then the algorithm tests

every converging triple $(A - B - C)$ in B_1 for the feasibility of adding a new link to shield that triple (the poisoning attack is conducted by adding case by case to the attack dataset DB_2, and applying the PC algorithm every time a new case is added to DB_2).

Algorithm 2 tests the feasibility of adding a new link to every converging connection in B_1. For each converging triple in B_1, the algorithm terminates if it succeeds in shielding a collider or just prints a failure message when the number of added cases is more than $\beta \times n$. Our empirical results are presented in section 5.4.

5.3 Model Invalidation Attacks Based on Marginal Independence Tests

When learning the structure of a Bayesian network model from data, the PC algorithm starts by analyzing the conditional independence statements between variables. It performs χ^2 statistical test on the given dataset to establish the set of statistical independence statements for the learned causal model [48]. Using this information of how the PC algorithm works, adversarial attackers may contaminate the input dataset with the goal of removing weak edges or adding the most believable yet incorrect links. Based on the direct impact of marginal independence tests on the PC algorithm, model invalidation attacks can be divided into two main types:

1) removing weak edges, and

2) adding the most believable yet incorrect edge.

5.3.1 Feasibility of deleting an Edge from a Causal Model

Before we define attacks based on removing weak edges, we need to define a new algorithm (Algorithm 3) for checking the feasibility for deleting an edge from a given causal model.

Given a dataset, DB_1, a Bayesian network model, B_1, where B_1 is the result of the learned causal model when given DB_1 as an input to the PC learning algorithm, and a contingency table for two random variables in B_1 ($variable_1$ with i states and $variable_2$

with j states), we introduce Algorithm 3 which checks the feasibility of deleting an existing edge in a causal model.

Let n be the number of cases in DB_1, and let β be *data poisoning rate* at which we are allowed to edit or "contaminate" DB_1 (we default set $\beta \leq 0.05$); algorithm 3 presents algorithmic details of data poisoning attacks that aim to delete an existing edge in a Bayesian network model as follows:

Algorithm 3: Edge Deleting Procedure

Input : Dataset DB_1 ▷ Original dataset with n cases
Output: Contaminated dataset DB_2 or a failure message

Procedure `Edge Deleting(`DB_1, B_1, $A - C$`)`
  Construct the $i \times j$ the contingency table for $variable\ A$ and $variable\ C$ from DB_1;
  Let $DB_2 = DB_1$;
  for $K = 1, 2, ..., \beta \times n$ ▷ *where $\beta \times n$ is the maximum number of contaminated tuples to be added to DB_1* **do**
    In the contingency table for the link $A - C$, determine the cell that has the highest test statistics value ($cell_{ij}$) and the cell that has the smallest test statistics value($cell_{i'j'}$);
    DB_2 for deleting the link $A - C$ = DB_2' ▷ where DB_2' is the old DB_2 expect an instance is transferred from ij state to $i'j'$ state;
    Run the PC algorithm on DB_2 to learn the structure of the Bayesian network model;
    if *There is no edge between vertex A and vertex C* **then**
      Return BD_2;
    end
    if $K = \beta \times n$ **then**
      Return msg "Failed to introduce the link $A - C$ within a feasible number of cases" ;
    end
    Update the $i \times j$ the contingency table for $variable\ A$ and $variable\ C$ from DB_2;
  end
end

Algorithm 3 starts by constructing the contingency table for $variables\ A$ and C. The contingency table in algorithm 3 is needed to accelerate the process of removing an edge from a Bayesian model. That is, moving cases from the cell with the highest test statistics

value to the cell with the lowest test statistics value will significantly accelerate the edge deletion process. The poisoning attack in this algorithm is performed by modifying case by case in the poisoned dataset DB_2, and then applying the PC algorithm every time a case is modified in DB_2.

Algorithm 3 returns a dataset DB_2 if deleting the edge $A - C$ is feasible; otherwise, a failure message will be printed since the number of added cases will be more than $\beta \times n$.

5.3.2 Removing a Weak Edge

We show that it is feasible to use link strengths measure to identify and rank the edges on a causal model from the weakest to the strongest. Thus, adversarial opponents may attempt to poison the learning dataset with the goal of removing weak edges.

We have developed Algorithm 4 to check the resilience of the PC algorithm against attacks that target weak edges.

Let DB_1 be an input dataset, and let n be the number of cases in DB_1, Algorithm 4 provides algorithmic details of data poisoning attacks that aim to delete the weakest edge in a Bayesian model as follows:

Algorithm 4: Removing a Weak Edge Procedure

Input : Dataset DB_1 ▷ Original dataset with n cases
Output: Contaminated dataset DB_2 or a failure message

Procedure `Removing a Weak Edge(`DB_1`)`
  Use the PC algorithm for learning the structure of Bayesian network model B_1 from dataset DB_1 (using the default significance level at 0.05 [38]);
  Use L_S to rank the edges of B_1 from the weakest to the strongest;
  Let $A - C$ be the weakest edge to be deleted from B_1;
  Test the feasibility of deleting the edge $A - C$ from B_1 using *Algorithm 3*;
  if *Algorithm 3 returns* DB_2 **then**
    Return DB_2;
  else
    Return msg "*Algorithm 3* failed to delete the link $A - C$ within a feasible number of cases";
  end
end

Algorithm 4 starts by learning the structure of the Bayesian network model B_1 from dataset DB_1. It then calculates the strength of each link in the model B_1 and rank them from the weakest to the strongest edge. After that, Algorithm 4 checks the robustness of the PC algorithm against the feasibility of deleting the weakest edge in B_1 by calling Algorithm 3.

Algorithm 4 returns a contaminated dataset DB_2 if deleting the weakest edge is feasible; otherwise, a failure message is printed since the number of added cases is more than $\beta \times n$. Our empirical results are presented in section 5.5.

5.3.3 Feasibility of Adding an Edge to a Causal Model

Before we define attacks based on adding the most believable yet incorrect edge, we need to define a new algorithm (Algorithm 5) for checking the feasibility for deleting an edge from a given causal model.

We have presented Algorithms 1 and 2 in sections 5.2.1 and 5.2.2, receptively. Algorithms 1 and 2 check the feasibility of introducing a new link in a given Bayesian network triple $(A - B - C)$. In this section, given a dataset DB_1, a model B_1, which is the result of feeding DB_1 to the PC algorithm, we introduce Algorithm 5 which checks the feasibility of adding a link between two nodes that do not lie in a triple in a Bayesian network model.

Let n be the number of cases in DB_1, and let β be *data poisoning rate* at which we are allowed to "poison" DB_1 (we default set $\beta \leq 0.05$); algorithm 5 presents algorithmic details of data poisoning attacks that aim to introduce a link between two vertices that do not lie in a triple in a Bayesian network model as follows:

Algorithm 5 starts by constructing the contingency table for $variables$ A and C. The use of the contingency table in this algorithm will accelerate the process of adding a new edge between two nodes that do not lie in a triple in a Bayesian network model. That is, adding more observed cases to the cell with the highest test statistics value will dramatically accelerate the process of adding a link to a causal model. The data poisoning attack in this

Algorithm 5: Edge adding Procedure

Input : 1) Dataset DB_1 ▷ Original dataset with n cases.
2) A Bayesian network model B_1 ▷ B_1 is the of the resulted model when given DB_1 as an input to the PC algorithm
3) The link $A - C$ ▷ The link we intend to add to B_1

Output: Contaminated dataset DB_2 or a failure message

Procedure `Edge Adding(`DB_1, B_1, $A - C$`)`
  Construct the $i \times j$ the contingency table for $variable\ A$ and $variable\ C$;
  Let $DB_2 = DB_1$;
  for $< K = 1, 2, ..., \beta \times n >$ ▷ *$\beta \times n$ is the maximum number of poisoned tuples to be added to DB_1* **do**
    In the contingency table for the link $A - C$, determine the cell that has the highest test statistics value ($cell_{ij}$);
    DB_2 for the link AC = DB_2 + ***d*** ▷ where ***d*** is an instance in which $A = state\ i, C = state\ j$, and all other variables in ***d*** are No;
    Run the PC algorithm on DB_2 to learn the structure of the Bayesian network model;
    if *There is a new edge between vertex A and vertex C* **then**
      Return BD_2.
    end
    if $K = \beta \times n$ **then**
      Return msg "Failed to introduce the link $A - C$ within a feasible number of cases"
    end
  end
end

algorithm is performed by adding case by case to the poisoned dataset DB_2, and then applying the PC algorithm every time a new case is added to DB_2.

Algorithm 5 terminates by either returning the poisoned dataset DB_2 if deleting the edge $A - C$ is feasible or by returning a failure message since the number of added cases will be more than $\beta \times n$.

5.3.4 Adding the Most Believable yet Incorrect Edge

We show that it is feasible to use link strengths measure to identify and rank the edges on a causal model from the most to the least believable edge. Thus, adversaries can cleverly use data poisoning attacks craft the input dataset to the Bayesian network model so that adding

those incorrect yet plausible edges is viable.

We have developed Algorithm 6 to check the robustness of the PC algorithm against this attack.

Let DB_1 be an input dataset, and let n be the number of cases in DB_1, Algorithm 6 provides algorithmic details of data poisoning attacks that aim to add the most believable link to a Bayesian model as follows:

Algorithm 6: Adding the Most Believable yet Incorrect Edge Procedure

Input : Dataset DB_1 ▷ Original dataset with n cases
Output: Contaminated dataset DB_2 or a failure message

Procedure `Adding the Most Believable yet Incorrect Edge(`DB_1`)`
  Use the PC algorithm for learning the structure of Bayesian network model B_1 from dataset DB_1 (using the default significance level at 0.05 [38]);
  Choose a set of edge Q that could be added to B_1;
  Use L_S to rank the set of edges Q from the most to the least believable edge;
  Let $A - C$ be the most believable edge to be added to B_1;
  if *$A - C$ lies in a a serial or diverging triple $A - B - C$* **then**
    Use *Algorithm 1* to check the feasibility of adding the link $A - C$;
    if *Algorithm 1 returns DB_2* **then**
      Return DB_2;
    else
      Return msg "*Algorithm 1* failed to introduce the link $A - C$" ;
    end
  else if *$A - C$ lies in a converging triple $A \rightarrow B \leftarrow C$* **then**
    Use *Algorithm 2* to check the feasibility of adding the link $A - C$;
    if *Algorithm 2 returns DB_2* **then**
      Return DB_2;
    else
      Return msg "*Algorithm 2* failed to introduce the link $A - C$"
    end
  else
    Use *Algorithm 5* to check the feasibility of adding the link $A - C$;
    if *Algorithm 5 returns DB_2* **then**
      Return DB_2
    else
      Return msg "*Algorithm 5* failed to introduce the link $A - C$"
    end
  end
end

Algorithm 6 starts by learning the structure of the Bayesian network model B_1 from dataset DB_1. It then asks the user to choose the set of edges that could be added to B_1. Algorithm 6 then uses link strengths measures to rank the set of edges Q from the most to the least believable edge. Let the most believable edge to be added to B_1 be $A - C$. If the link $A - C$ introduces a new v-structure in a triple $A - B - C$, then Algorithm 1 is called. On the hand, if the link $A - C$ shield a collider B in a triple $A - B - C$, then Algorithm 2 is called. Otherwise, Algorithm 5 is called to add a link between two vertices that do not lie in a triple in a Bayesian network model.

In all different scenarios, Algorithm 6 returns a contaminated dataset DB_2 if adding the most believable edge is feasible; otherwise, a failure message is printed since the number of added cases will be more than $\beta \times n$. Our empirical results are presented in section 5.5.

5.4 Empirical Results for Model Invalidation Attacks Based on the Notion of D-separation

In this experiment, we evaluated the effectiveness of model invalidation attacks based on the notion of d-separation (section 5.2) to poison the Chest Clinic Network dataset DB_1. Our aim is to introduce a new v-structure. That is,

1) add the links $D - S$, $B - L$ and $S - E$ to the serial connections $D \rightarrow B \rightarrow S$, $B \rightarrow S \rightarrow L$ and $S \rightarrow L \rightarrow E$, respectively, and

2) add the link $A - E$ to the diverging connection $A \leftarrow T \rightarrow E$.

We also study the robustness of the PC learning algorithm against the attacks aiming to break an existing v-structure, i.e., to shield the collider $T \rightarrow E \leftarrow L$.

We present our results in Figures 5.2, 5.3, 5.4, 5.5, and 5.6. We succeeded to invalidate (change the Markov equivalence class) the model learned by the PC algorithm. We had to introduce 74 corrupt cases (data items) to introduce the link $D - S$. To introduce links $B - L$, $S - E$, and $A - E$ required 13, 40, and 3 corrupt cases, respectively. To shield

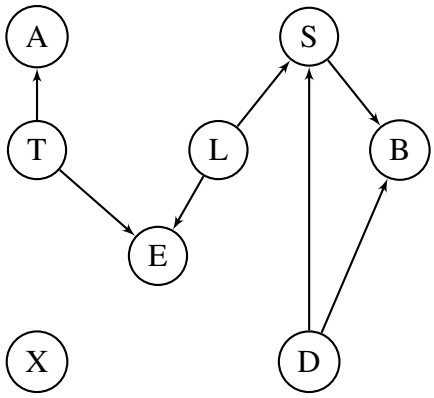


Figure 5.2: Introducing a new converging connection in the triple $D - B - S$.

the collider E, we only needed 8 poisoning data items. In addition, when we increased the number of corrupted data items, the PC learning algorithm was acting unstably. Our results after adding 17 poising cases to introduce the malicious link $T - L$ is in Figure 5.7.

We also observed that the choice of corrupt data items affects the efficiency of the attack. That is, when introducing a malicious link between two random variables, a cell with a higher test statistics value $< ts_{ij} >$ in the contingency table of these two random variables requires fewer corrupt data items than a cell with a lower test statistics value. For example, when poisoning dataset DB_1 to add the link $D - S$, we needed more corrupt data items as the value of test statistics got lower. The results are as follows: the cell with $D = yes$ and $S = yes$ required 74 cases, the cell with $D = yes$ and $S = no$ required 272 cases, the cell with $D = no$ and $S = yes$ required 1120 cases, and the cell with $D = no$ and $S = no$ required 1701 cases. Overall, we showed that the PC algorithm is vulnerable to model invalidation attacks based on the notion of d-separation.

5.5 Empirical Results for Model Invalidation Attacks Based on Marginal Independence Tests

Link strength measure (L_S) is needed for the second experiment. Given B_1 model as shown in Figure 1.4 and the dataset DB_1, we followed the *two steps* presented in section 4. Table 5.1 contains the posterior distributions calculated in *step 1*. Figure 5.8 shows the

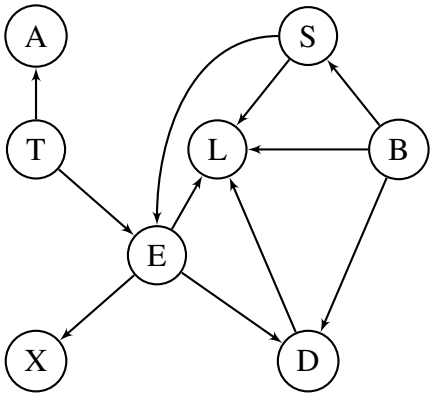


Figure 5.3: Introducing a new converging connection in the triple $B - S - L$.

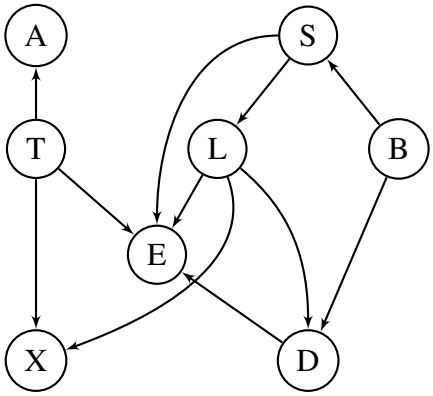


Figure 5.4: Introducing a new converging connection in the triple $S - L - E$.

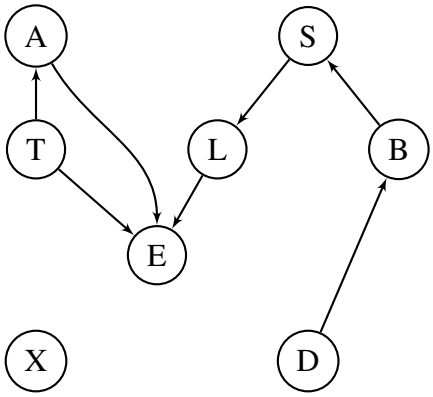


Figure 5.5: Introducing a new converging connection in the triple $A - T - E$.

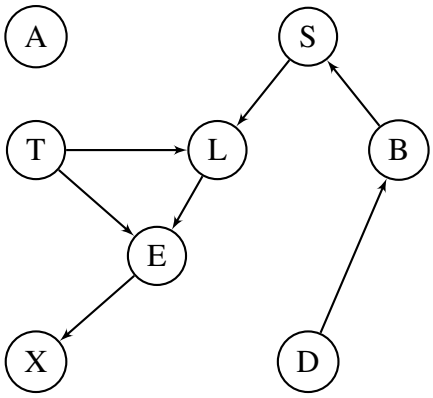


Figure 5.6: Breaking an existing converging connection in the triple $T - E - L$.

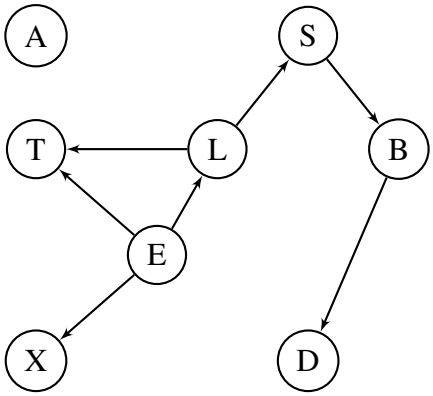


Figure 5.7: The result of using 17 cases to break the v-structure $T \rightarrow E \leftarrow L$.

Table 5.1: Posterior distributions for the Chest Clinic Network.

Link	**Posterior Distributions (Beta Distributions)**
P(T | A)	Beta(10,99) Beta(106,9789) Beta(99,10) Beta(9789,106)
P(L | S)	Beta(481,4510) Beta(47,4966) Beta(4510,481) Beta(4966,47)
P(B | S)	Beta(3019,1972) Beta(1514,3899) Beta(1972,3019) Beta(3899,1514)
P(E | T)	Beta(115,1) Beta(523,9365) Beta(1,115) Beta(9365,523)
P(E | L)	Beta(527,1) Beta(111,9365) Beta(1,527) Beta(9365,111)
P(D | B)	Beta(3638,895) Beta(725,4746) Beta(895,3638) Beta(4746,725)

final link strength evaluation (L_S), calculated in *step 2*.

We will use these strength measures in this section and in section 6.2 to illustrate the ease of removing existing links and adding links to a causal model.

In this experiment, we evaluated the effectiveness of model invalidation attacks based on marginal independence tests (section 5.3) to poison the Chest Clinic Network dataset

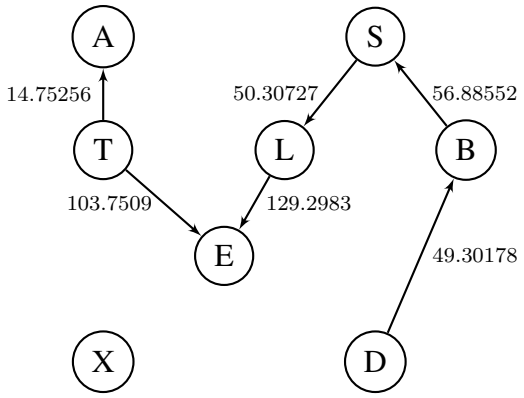


Figure 5.8: Results of L_S on the learned model by the PC algorithm B_1.

DB_1. In this experiment, we check the resilience of the PC algorithm against the feasibility of deleting the weakest edge in the Bayesian model B_1. To determine the weakest edge in B_1, we do the following:

1) use the defined link strength measure L_S to rank the edges of B_1 from the weakest to the strongest edge, and

2) check the feasibility of poisoning dataset DB_1 to remove the weakest edge.

We also study the robustness of the PC algorithm against attacks aiming to add the most believable yet incorrect edge to B_1. To determine the most believable edge to be added to B_1, we do the following:

1) determine the set of edges Q that could be added to the model B_1 (in this experiment, we let $Q = \{A - S, T - S, D - S, L - B, L - T\}$),

2) use the defined link strength measure to rank the set of edges Q from the most to the least believable edge, and

3) check the feasibility of poisoning dataset DB_1 to add the most believable edge.

We present our results of deleting the weakest edge from B_1 in Table 5.2 and Figure 5.9. We succeeded to invalidate the model learned by the PC algorithm. We had to modify only 3 cases to break the weakest link $A - T$. Our results of adding the most believable edge to

Table 5.2: The result of using L_S to rank B_1 edges from the weakest to the strongest.

Link	**Link Strength** L_S	**Rank**
$A \rightarrow T$	14.75256	1
$S \rightarrow L$	50.30727	3
$S \rightarrow B$	56.88552	4
$T \rightarrow E$	103.7509	5
$L \rightarrow E$	129.2983	6
$B \rightarrow D$	49.30178	2

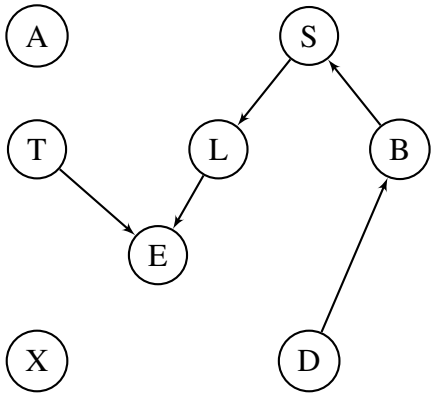


Figure 5.9: The result of removing the weakest link in B_1, $A \rightarrow T$

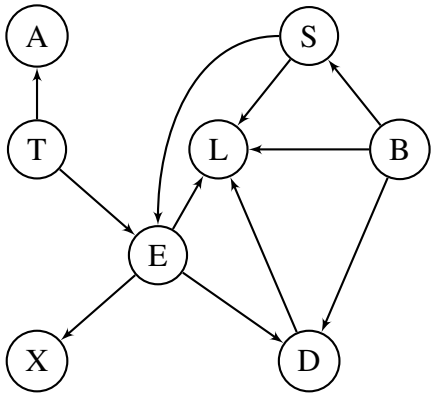


Figure 5.10: The result of adding the most believable link to B_1, $B \rightarrow L$.

B_1 are presented in Tables 5.3, 5.4, and Figure 5.10. We succeeded to fool the PC algorithm and invalidate the learned model. We had to introduce only 13 corrupt data items to add the most believable link $B - L$.

We observed that when removing an edge from a causal model, the choice of corrupt data items has an impact on the efficiency of the attack. That is, transferring data items from

Table 5.3: Posterior distributions for the set of edges Q.

Link	**Posterior Distributions (Beta Distributions)**
P(S | A)	Beta(57, 52) Beta(4934, 4961) Beta(57, 52) Beta(4934, 4961)
P(T | S)	Beta(49, 4942) Beta(67, 4946) Beta(49, 4942) Beta(67, 4946)
P(D | S)	Beta(2728, 2263) Beta(1635, 3378) Beta(2728, 2263) Beta(1635, 3378)
P(L | B)	Beta(312, 4221) Beta(216, 5255) Beta(312, 4221) Beta(216, 5255)
P(L | T)	Beta(5, 111) Beta(523, 9365) Beta(5, 111) Beta(523, 9365)

Table 5.4: L_S results.

Link	**{Link strength L_S}**	**Rank**
$A \rightarrow S$	8.313748	5
$S \rightarrow T$	28.66903	3
$S \rightarrow D$	54.90557	2
$B \rightarrow L$	91.51039	1
$T \rightarrow L$	21.92398	4

the cell with the highest test statistics value to the cell with the lowest test statistics value in a contingency table of two random variables will accelerate the process of removing the link between them. Overall, we showed that the PC algorithm is vulnerable to model invalidation attacks based on marginal independence tests.

CHAPTER 6

TARGETED CHANGE ATTACKS

6.1 OVERVIEW OF TARGETED CHANGE ATTACKS

A targeted change attack against the PC algorithm is an active malicious attack in which malicious agents try to move from the state of "what I have" to the state of "what I want" by poisoning the learning dataset. Adversaries attempt to plan attacks against Bayesian network models using the least number of changes. That is, they will attempt to move from the existing model to the desired model using the least and inconspicuous number of changes. As such, adversaries assess the difficulty of entering or modifying data that promises to intentionally change the current model into the desired model. By doing so, the adversary is able to make the changed model behave exactly as they want.

A targeted change attack is more harmful and sophisticated than model invalidation attack. For this, adversaries attempt to poison the input dataset aiming for a specific result of the BN model; therefore, it misclassifies a certain class of false positives and false negatives.

Let DB_1 be an input dataset to the PC learning algorithm, and let n be the number of cases in DB_1, Algorithm 7 provides algorithmic details of targeted data poisoning attacks that aim to implement a complete attacking scenario against a given Bayesian model as follows:

Algorithm 7 starts by learning the structure of the Bayesian network model B_1 from dataset DB_1. It then uses the defined link strengths measure to rank the edges of B_1 from the weakest to the strongest edge. A malicious user can enter the set of edges Q that the

Algorithm 7: Targeted Change Attacks Procedure

Input : Dataset DB_1 ▷ Original dataset with n cases
Output: Contaminated dataset DB_2 or a failure message

```
Procedure Targeted Change Attacks(DB1)
    Use the PC algorithm for learning the structure of Bayesian network model B1
     from dataset DB1 (setting the significance of the Hugin PC to the default
     level, which is 0.05 [38]);
    Use L_S to rank the edges of B1 from the weakest to the strongest edge;
    Choose a set of edge Q that could be added to B1;
    Use L_S to rank the set of edges Q from the most to the least believable edge;
    Plan a targeted attack (the set of edges to be added or deleted from B1);
    repeat
        if there is a need to introduce a new link in B1 then
            Use Algorithm 1 to introduce a new v-structure, Algorithm 2 to break
             an existing collider, or Algorithm 5 to add a link between two vertices
             that do not lie in a triple;
        end
        if there is a need to delete an existing link then
            Use Algorithm 3;
        end
        if there is a need to remove the weakest edge then
            Use Algorithm 4;
        end
        if there is a need to add the most believable edge then
            Use Algorithm 6;
        end
    until the targeted attack is achieved;
end
```

user wants add to the model B_1. The defined link strength measure is used to rank the set of edge Q from the most to the least believable edge.

The malicious user then plans a targeted change attack. The adversary in this case chooses the set of edges that could be added to or deleted from the causal model B_1. For example, an attacker may think it is feasible to achieve his goal by adding a new plausible link and deleting an existing one.

If the attacker wants to add a new link $A - C$ and this new link introduces a new v-structure in a triple $A - B - C$, then *Algorithm 1* is called. On the hand, if the link $A - C$ shield a collider B in a triple $A - B - C$, then *Algorithm 2* is called. Otherwise, *Algorithm 5*

is called to add a link between two vertices that do not lie in a triple in a Bayesian network model.

If the attacker wants to delete an existing edge. There are two algorithms that can check the feasibility of achieving this goal. *Algorithm 3* checks the feasibility of deleting any edge in a Bayesian network model, and *Algorithm 4* checks the feasibility of deleting the weakest edge in a Bayesian network model.

In all different scenarios, *Algorithm 7* returns a contaminated dataset DB_2 if achieving the targeted attack is feasible; otherwise, a failure message will be printed if the number of added cases will be more than $\beta \times n$, where β is *data poisoning rate* at which we are allowed to add new "poisoned" cases to DB_1 (we default set $\beta \leq 0.05$)

6.2 Empirical Results for Targeted Change Attacks

A further goal of this research is to study the influence of targeted change attacks on our dataset DB_1. We validate the effectiveness of targeted change attacks described in *Algorithm 7* (section 6) to poison the Chest Clinic network dataset DB_1 with the goal of achieving a particular change to the model. *Algorithm 7* checks the robustness of the PC algorithm against the feasibility of implementing a targeted change attack.

Given the link strength measure L_S for ranking the edges of the model B_1 from the weakest to the strongest edge (Table 5.2) and given L_S for ranking the set of edges Q that could be added to the model B_1 from the most to the least believable edge (Table 5.4), we aim to achieve the following targeted attack against the model B_1:
Change the model B_1 such that it concludes that smoking (S) causes dyspnoea (D) but not lunge cancer(L).
Our attack had the following two steps (see Figure 6.1):

1) use *Algorithm 7* to delete the link $S \rightarrow L$, and then

2) use *Algorithm 7* again to add the link $S \rightarrow D$.

We present our results in Figures 6.2, and 6.3. We observed that *Algorithm 7* succeeded to delete the link $S \rightarrow L$ by modifying only 114 data items in our dataset DB_1, resulting in a dataset DB_2 (Figure 6.2). Then we fed DB_2 to *Algorithm 7* succeeded to add the link $D \rightarrow S$. We needed only 74 cases to introduce the link $D \rightarrow S$ in dataset DB_2 (Figure 6.3). Overall, we showed that the PC algorithm is vulnerable to targeted change attacks.

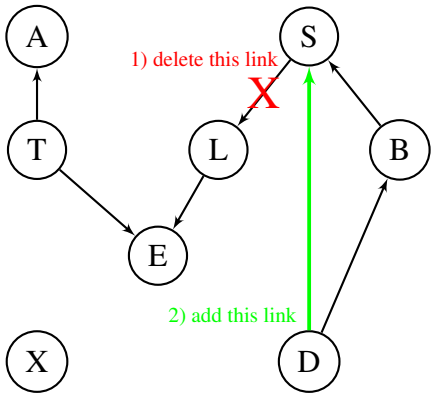


Figure 6.1: A targeted attack against model B_1

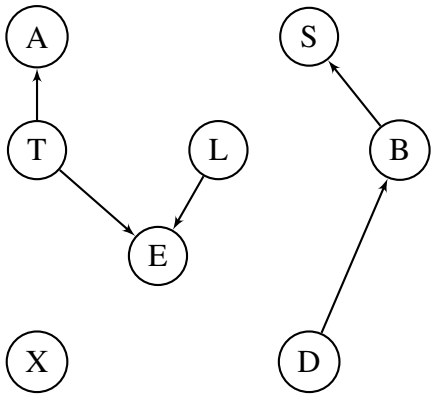


Figure 6.2: The model B_1 after achieving *step 1* (deleting $S \rightarrow L$)

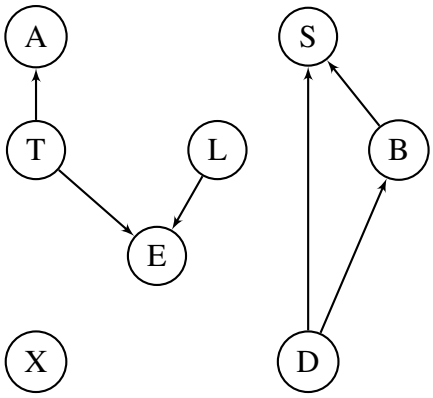


Figure 6.3: The model B_1 after achieving the *two steps* of the targeted attack

CHAPTER 7

ADVERSARIAL ATTACKS AGAINST THE LCD ALGORITHM

7.1 INTRODUCTION

Probabilistic Graphical Models (PGMs) have become a powerful framework for not only representing but reasoning with probabilistic knowledge. Learning the structure of probabilistic graphical models, namely Bayesian networks, can be performed either subjectively from knowledge of experts or objectively from observed data [30, 44]. The former method is used when the structure of the probabilistic model is simple. Whereas, the latter method is used when the structure of the graphical model is intricate for human brains to process.

Learning the structure of Bayesian networks from data is very important in machine learning and artificial intelligence applications. There are three main approaches to learn the structure of Bayesian networks from data. The constraint-based approach, such as the IC algorithm [64], the PC algorithm [60, 61] and the NPC algorithm [62], counts on conditional independence tests to determine the DAG of the learned Bayesian network. The score-based approach, such as AIC [1], BDe [27, 45], K2 [21], and BIC algorithm [56], assigns a score for each Bayesian network structure (this score indicates how well the Bayesian network structure fits the data) and then perform a (usually greedy) search algorithm to select the structure with the highest score. The hybrid approach, such as CB [59] and EGS algorithm [22], relies on the idea of using both constraint-based algorithms and score-based algorithms. The use of constraint-based algorithms will reduce the search space (i.e., it will reduce the number of candidate DAGs). Thenceforth, score-based algorithms can be used to select the optimal DAG.

Within the constraint-based approach, there are two main methods for learning the structure of Bayesian networks: 1) The traditional constraint-based method in which the structure of the whole Bayesian network is constructed over all variables, such as the IC [64], PC [60, 61], NPC [62] algorithms. 2) The decomposable constraint-based method in which the structure of the large Bayesian network is decomposed into many small Bayesian networks, such as LCD (Learn Chain graphs via Decomposition) algorithm [68, 37]. In both methods, Bayesian structure learning algorithms are prone to model inaccuracies resulting from corrupt data in the training phase (a.k.a Data Poisoning Attacks [6]).

Data poisoning attacks are one of the most important emerging security threat against machine learning systems. These attacks aim to corrupt the machine learning model by contaminating the data in the training phase [6, 9, 43]. The lack of resilience in Bayesian network algorithms against such attacks leads to model inaccuracies when learning the structure from data. Therefore, it is crucial to address the robustness of Bayesian network algorithms against data poisoning attacks especially if these models are going to be used in machine learning applications to reliably automate jobs.

In this section, we present an empirical analysis of the robustness of the LCD algorithm against adversarial attacks that aim to invalidate the new to-be-learned Bayesian model as defined in Chapter 5. We investigate two potential model invalidation attacks against the LCD algorithm: (1) Model invalidation attacks based on the notion of d-separation, and (2) Model invalidation attacks based on marginal independence tests.

7.2 Empirical Results

To conduct the experiments of this section, we used the same dataset that was generated to evaluate the robustness of the PC learning algorithm (dataset DB_1 as shown in Chapter 1). Using the LCD algorithm on dataset DB_1 with 0.05 significance setting, the resulting structure is given in Figure 1.5. While the new learned network using the LCD algorithm (network B_3 as shown in Figure 1.5) belongs to a different Markov equivalence

class than the original Chest Clinic network (Figure 1.3), we will use the network B_3 as the starting point of our experiments.

7.3 Empirical Results of Model Invalidation Attacks Based on the Notion of D-separation

In this experiment, we evaluated the effectiveness of model invalidation attacks based on the notion of d-separation (as introduced in section 5.2) to poison the Chest Clinic Network dataset DB_1 and thereby impact the new to-be-learned model by the LCD algorithm. We study the robustness of the LCD structure learning algorithm against the data poisoning attacks that aim to introduce a new v-structure to the Bayesian model B_3. That is,

1) add the links $D - S$, $B - L$ and $S - E$ to the serial connections $D \rightarrow B \rightarrow S$, $B \rightarrow S \rightarrow L$ and $S \rightarrow L \rightarrow E$, respectively, and

2) add the link $A - E$ to the diverging connection $A \leftarrow T \rightarrow E$.

We also study the robustness of the LCD structure learning algorithm against the attacks aiming to break an existing v-structure, i.e., to shield the collider $T \rightarrow E \leftarrow L$.

We present our results in Figures 7.1, 7.2, 7.3, 7.4, and 7.5. We succeeded to invalidate the model learned by the LCD algorithm. We had to introduce 90 contaminated cases in order to introduce the link $D - S$. To introduce links $B - L$, $S - E$, and $A - E$ required 13, 46, and 4 corrupt cases, respectively. To shield the collider E, we only needed 8 poisoning data items.

7.4 Empirical Results of Model Invalidation Attacks Based on Marginal Independence Tests

In this section, we need our link strength measure (L_S) to measure the strengths of links of the Bayesian network model B_3. Our goal is to determine (1) the weakest edge to be deleted from B_3, and (2) the most beleivable edge to be added to B_3. The results of our

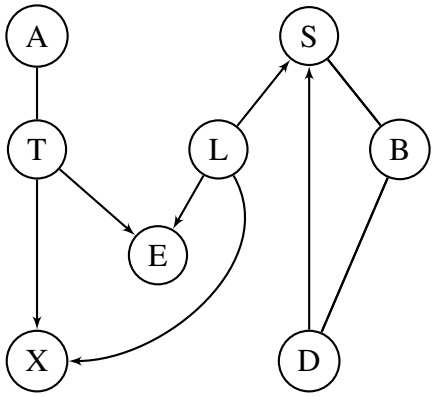


Figure 7.1: The result of adding the edge $D - S$ to the Bayesian model B_3

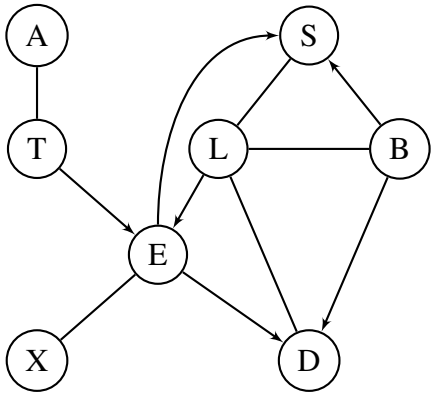


Figure 7.2: The result of adding the edge $B - L$ to the Bayesian model B_3

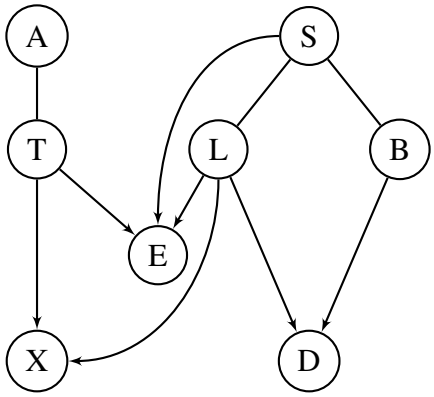


Figure 7.3: The result of adding the edge $S - E$ to the Bayesian model B_3

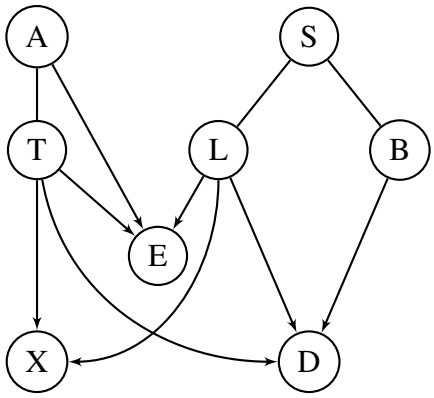


Figure 7.4: The result of adding the edge $A - E$ to the Bayesian model B_3

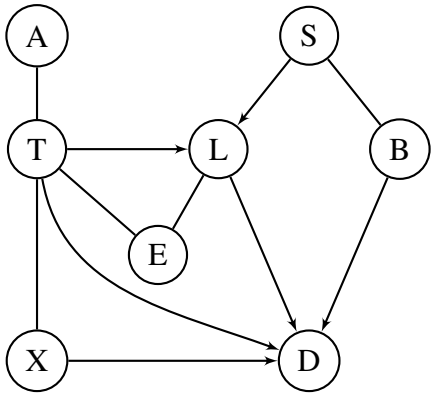


Figure 7.5: The result of adding the edge $T - L$ to the Bayesian model B_3

link strength measure to measure the strengths of the Chest Clinic Network are presented in Figure 5.8. We note that the edge $A - T$ is the weakest edge in the Bayesian network model B_3. Also, the edge $D - S$ is the most believable edge that could be added to the model B_3.

We present our results of deleting the weakest edge from B_3 in Figure 7.6. We succeeded to invalidate the model learned by the LCD algorithm. We had to modify only 3 cases to break the weakest link $A - T$.

We present our results of adding the most believable yet incorrect edge to the model B_3 in Figure 7.7. We succeeded to invalidate the model learned by the LCD algorithm. We had to modify only 90 cases to add the most believable link $D - S$.

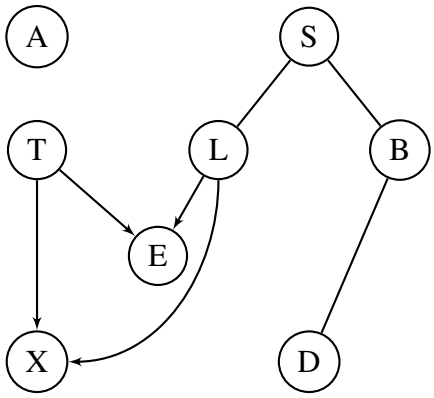


Figure 7.6: The result of deleting the weakest edge $A - T$ from the Bayesian model B_3

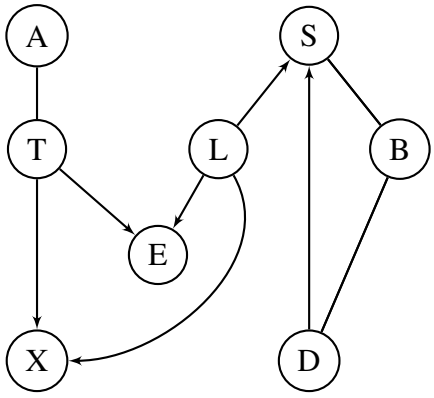


Figure 7.7: The result of adding the most believable yet incorrect edge $D - S$ to the Bayesian model B_3

Table 7.1: Summary of the required number of corrupt cases to contaminated the dataset DB_1.

Link	**# of required cases to corrupt the PC Algorithm**	**# of required cases to corrupt the LCD Algorithm**
Add $A \rightarrow E$	3	5
Add $B \rightarrow L$	13	13
Add $D \rightarrow S$	74	90
Add $S \rightarrow E$	40	46
Add $T \rightarrow L$	8	8
Remove $A \rightarrow T$	3	3

7.5 WHICH ALGORITHM IS MORE ROBUST TO DATA POISONING ATTACKS: THE PC ALGORITHM OR THE LCD ALGORITHM?

From the previous experiments that were presented in sections 7.3 and 7.4, we can summarize the required number of corrupt cases to contaminate the dataset DB_1 as follows:

We observe that the number of required cases to contaminate the LCD algorithm is always larger than or equal to the number of required cases to contaminate the PC algorithm. We observe that the LCD algorithm is more robust to data poisoning attacks than the PC algorithm. We conjecture that this is due to the fact that the LCD algorithm learns the structure of the Bayesian network via the decomposition approach.

CHAPTER 8

LONG-DURATION DATA POISONING ATTACKS

8.1 INTRODUCTION

In our the previous chapters, we studied data poisoning attacks against Bayesian structure learning algorithms. For a Bayesian structure learning algorithms, given the dataset, $\mathcal{DS}^{\text{v}}$, and the corresponding model, B_1 (Equation 9.1), a malicious attacker attempts to craft an input dataset, $\mathcal{DS}^{\text{p}}$, such that this contaminated dataset will have an immediate impact on $\mathcal{DS}^{\text{v}}$ and thereby on B_1. The defender periodically retrains the machine learning system to recover the structure of the new model, B_2, using $\mathcal{DS}^{\text{u}}$, the combination of the original dataset $\mathcal{DS}^{\text{v}}$ and the attacker supplied $\mathcal{DS}^{\text{p}}$. We call such an attack a "one-step" data poisoning attack as malicious attackers send all contaminated cases at once.

In this chapter, we introduce long-duration data poisoning attacks against structure learning algorithms. *Long-duration poisoning attacks* are adversarial multi-step attacks in which a malicious attacker attempts to send contaminated cases over a period of time, $t = \{1, 2, \ldots, w\}$. That is, at every time point i, a malicious attacker sends in a new dataset, $\mathcal{DS}_i^{\text{c}}$, which contains N_i cases, $\lambda_i \text{N}_\text{i}$ of which are corrupted cases for some $0 < \lambda_i < 1$ (λ_i is the data poisoning rate at which we allowed to add contaminated cases to $\mathcal{DS}_\text{i}^{\text{c}}$ at iteration i). Even though the defender periodically retrains the model, $B_2^{'}$, at time i using the dataset $\mathcal{DS}_i^{\text{l\_d}}$, which is equal to $\mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{i} \mathcal{DS}_t^{\text{c}}$, it is not easy to detect the long-duration attack since such an attack is not instantaneous.

By the end of the long-duration poisoning attack, i.e., at time point w, the attacker would have injected $\bigcup_{t=1}^{w} \mathcal{DS}_t^{\text{c}}$ to $\mathcal{DS}^{\text{v}}$, resulting in a new dataset, $\mathcal{DS}_w^{\text{l\_d}}$. We assume that

attackers cannot add more than βN cases to $\mathcal{DS}^{\text{v}}$ (i.e., $0 < \bigcup_{t=1}^{w} \lambda_t N_{\text{t}} < \beta N$). When the defender retrains the model, $B_2^{'}$, using the dataset $\mathcal{DS}_w^{\text{l\_d}}$, the attack will dramatically affect the resulting model. Note that this attack is sophisticated since the attacker may not need to send contaminated cases with the last contaminated dataset (the w^{th} dataset) in the long-duration attack, i.e., $\mathcal{DS}_w^{\text{c}}$ may trigger the attack with no poisoned cases, as our experiments show.

We propose causative, long-duration model invalidation attacks against Bayesian network structure learning algorithms. Such attacks are defined as malicious active attacks in which adversarial opponents attempt to arbitrarily corrupt the structure of the original Bayesian network model in any way. The goal of adversaries in these attacks is to poison the validated training dataset, $\mathcal{DS}^{\text{v}}$, over a period of time $t = \{1, \ldots, w\}$ using the contaminated dataset $\bigcup_{i=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$ such that $\mathcal{DS}^{\text{v}}$ will be no longer valid. We categorize causative long-duration model invalidation attacks against Bayesian network structure learning algorithms into two types: (1) Model invalidation attacks based on the notion of d-separation and (2) Model invalidation attacks based on marginal independence tests.

Causative, long-duration model invalidation attacks which are based on the notion of d-separation are adversarial attacks in which adversaries attempt to introduce a new link in any triple $(A - B - C)$ in the original Bayesian network model, B_1. The goal of the introduced malicious link, $(A - C)$, is to change the independence relations and the Markov equivalence class of B_1. Within such attacks, we can identify two subtypes: (i) Creating a New Converging Connection (V-structure), and (ii) Breaking an Existing Converging Connection (V-structure). See chapter 5 for more algorithmic details.

Causative, long-duration model invalidation attacks which are based on marginal independence tests are adversarial attacks in which adversaries attempt to use marginal independence tests in order to change the conditional independence statements between variables in the original model, B_1. Such attacks can be divided into two main subtypes: (i) Removing the Weakest Edge, and (ii) Adding the Most Believable Edge yet incorrect Edge. See

chapter 5 for more algorithmic details.

Algorithm 8 provides algorithmic details of long duration data poisoning attacks on machine learning algorithms that aim to achieve a certain attack by sending in contaminated cases over a period of time t.

Algorithm 8: Long Duration Poisoning Attacks

Input : $\mathcal{DS}^{\text{v}}[x^1; \ldots; x^T]$ ▷ Validated dataset with attributes $x^1; \ldots; x^T$.

Output: $\mathcal{DS}_w^{\text{l\_d}}[x^1; \ldots; x^T]$, or a failure message if the required number of cases to poison $\mathcal{DS}^{\text{v}}[x^1; \ldots; x^T]$, $\sum_{t=1}^{w} \lambda_t N_t$, is greater than βN.

Procedure `Long Duration Attacks(`$\mathcal{DS}^{\text{v}}$`)`
- $\tau = \beta N$ ▷ we default set $\beta = 0.05$;
- $B_1 = BN\_Algo(\mathcal{DS}^{\text{v}}[x^1; \ldots; x^T])$;
- Choose w ▷ Number of times over which smoothly contaminated datasets will be sent;
- **for** *t = 1; t* $\leq$ *w; t++* **do**
  - Generate $\mathcal{DS}_{\text{t}}^{\text{Clean}}$;
- **end**
- Plan an attack against the dataset ($\mathcal{DS}^{\text{v}} \cup \sum_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{Clean}}$) ▷ Note that there are several types of attacks defined in Appendix **??**;
- **for** *t = 1; t* $\leq$ *w; t++* **do**
  - Craft a new dataset, $\mathcal{DS}_{\text{t}}^{\text{p}}$ ▷ where $0 \leq \lambda_t N_{\text{t}} \leq \frac{\tau}{w}$;
  - $\mathcal{DS}_{\text{t}}^{\text{s\_c}} = (\mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{p}})$;
  - $\mathcal{DS}_t^{\text{l\_d}}[x^1; \ldots; x^T] = \mathcal{DS}^{\text{v}}[x^1; \ldots; x^T] \cup \mathcal{DS}_{\text{t}}^{\text{s\_c}}[x^1; \ldots; x^T]$;
  - $B_2^{'} = BN\_Algo(\mathcal{DS}_t^{\text{l\_d}}[x^1; \ldots; x^T])$;
- **end**
- **if** $\sum_{t=1}^{w} \lambda_t N_t \leq \beta N$ **then**
  - Return $\mathcal{DS}_t^{\text{l\_d}}[x^1; \ldots; x^T]$;
- **else**
  - Return msg "Algorithm **??** failed to achieve the long duration attack";
- **end**

end

8.2 Empirical Results

8.2.1 One-step Data Poisoning Attacks

To set up the experiment, we implemented the Chest Clinic Network using *Hugin™ Research 8.1.* We then used *Hugin™ case generator* [38, 49] to generate a simulated dataset

of $20,000$ cases. We call this dataset $\mathcal{DS}^{\text{v}}$. Using the PC algorithm on dataset $\mathcal{DS}^{\text{v}}$ with 0.05 significance setting [38], the resulting validated structure, $B_1 = PC\_Algo(\mathcal{DS}^{\text{v}})$, is given in Figure 1.4. While the two networks in Figures 1.3 and 1.4 belong to different Markov equivalence classes, we will use the validated network B_1 as the starting point of our experiment.

We evaluated the effectiveness of one-step data poisoning attacks against the validated dataset $\mathcal{DS}^{\text{v}}$ (i.e., against the validated model B_1). An attacker aims to use one-step data poisoning attacks to inject in a contaminated dataset $\mathcal{DS}^{\text{p}}$ into $\mathcal{DS}^{\text{v}}$, resulting in the dataset $\mathcal{DS}^{\text{u}}$. The defender retrains the machine learning model by feeding the new dataset $\mathcal{DS}^{\text{u}}$ to the PC learning algorithm ($B_2 = PC\_Algo(\mathcal{DS}^{\text{u}})$), resulting in the model B_2.

We aim to study the attacker's goals, i.e., study the feasibility of one-step data poisoning attacks, which might be as follows: (i) introduce new v-structures: that is, (1) add the links $D - S$ and $S - E$ to the serial connections $D \rightarrow B \rightarrow S$ and $S \rightarrow L \rightarrow E$, respectively, and (2) add the link $A - E$ to the diverging connection $A \leftarrow T \rightarrow E$; (ii) break an existing v-structure $T \rightarrow E \leftarrow L$, i.e., shield the collider E; (iii) remove the weakest edge, i.e., remove the edge $T \rightarrow A$; and (iv) add the most believable edge, i.e., add the edge $B \rightarrow L$. (Note that, for finding the weakest link in a given causal model or the most believable link to be added to a causal model, we refer the readers to our previous works [9, 6] for technical details on how to measure link strength of causal models).

In all of the scenarios, the attacker succeeded in corrupting the new model that was going to be learned by the defender, the model B_2. The attacker had to introduce a dataset $\mathcal{DS}^{\text{p}}$ with 67 corrupt cases (data items) to introduce the link $D - S$ in the newly learned model B_2. To introduce links $S-E$ and $A-E$ required 21 and 7 corrupt cases, respectively. To shield the collider E, the attacker only needed 4 poisoning data items. The attacker had to modify only 3 cases to break the weakest link $A - T$. To add the most believable link $B - L$ required to only 7 corrupt data items.

8.2.2 LONG-DURATION DATA POISONING ATTACKS

To set up the implementation of long-duration attacks, let $\mathcal{DS}^{\text{v}}$ be a validated training dataset with attributes $x_1, \ldots, x_n$ and N cases, and β be *data poisoning rate* at which attackers are allowed to add new "contaminated" cases to $\mathcal{DS}^{\text{v}}$. Let $\mathcal{DS}^{\text{c}}_{\text{i}}$ be a newly crafted dataset also with attributes $x_1, \ldots, x_n$ and N_i cases, and λ_i be data poisoning rate at which attackers allowed to add new crafted cases to $\mathcal{DS}^{\text{c}}_{\text{i}}$ (we default set $0 \leq \bigcup_{t=1}^{w} \lambda_i N_{\text{i}} \leq \beta N$).

We start by calculating τ, which is the maximum number of poisoned cases that could be added to $\mathcal{DS}^{\text{v}}$ over a period of time $t = \{1, \ldots, w\}$. We then learn the structure of the validated model B_1 from $\mathcal{DS}^{\text{v}}$ using the PC algorithm.

We then iterate w times. In each iteration t, we generate a clean dataset $\mathcal{DS}^{\text{clean}}_{\text{t}}$ and a poisoned dataset $\mathcal{DS}^{\text{p}}_{\text{t}}$. We let $\mathcal{DS}^{\text{c}}_{\text{t}} = \mathcal{DS}^{\text{clean}}_{\text{t}} \cup \mathcal{DS}^{\text{p}}_{\text{t}}$ (note that, $\mathcal{DS}^{\text{c}}_{\text{t}}$ has N_t cases, $\lambda_t N_t$ of which are poisoned). After that, we create the union of $\mathcal{DS}^{\text{c}}_{\text{t}}$ and $\mathcal{DS}^{\text{v}}$, resulting in $\mathcal{DS}^{\text{l\_d}}_{t}$, which is used to learn the structure of model $B^{'}_2$. Note that, in each iteration the number of cases in $\mathcal{DS}^{\text{p}}_{\text{t}}$ should be between 0 (i.e., no poisoned cases) and $\frac{\tau}{w}$, which is the maximum number of poisoned cases that could be added to $\mathcal{DS}^{\text{c}}_{\text{t}}$ in the t^{th} iteration.

We terminate after iteration w. If $\bigcup_{t=1}^{w} \lambda_t N_t \leq \beta N$, we return $\mathcal{DS}^{\text{l\_d}}_{t}$; otherwise, we print a failure message since implementing the long-duration attack on $\mathcal{DS}^{\text{v}}$ is not feasible.

We assumed that $w = 4$, which means that the attacker is allowed to send in four contaminated datasets to achieve the long-duration data poisoning attack. We divided the $20,000$ case dataset that was generated for one-step data poisoning attacks in section 8.2.1 into five datasets as follows: $12,000$ cases are used as $\mathcal{DS}^{\text{v}}$; and the rest is divided into four datasets of $2,000$ cases each. We call these four datasets $\mathcal{DS}^{\text{Clean}}_{1}$, $\mathcal{DS}^{\text{Clean}}_{2}$, $\mathcal{DS}^{\text{Clean}}_{3}$, and $\mathcal{DS}^{\text{Clean}}_{4}$. Using the PC algorithm on dataset $\mathcal{DS}^{\text{v}}$ with 0.05 significance setting [38], the resulting validated structure, $B_1 = PC\_Algo(\mathcal{DS}^{\text{v}})$, is given in Figure 1.4, which is the starting point of this experiment.

We evaluated the effectiveness of long-duration data poisoning attacks against the validated dataset $\mathcal{DS}^{\text{v}}$ (i.e., against the validated model B_1). At every time point $t = \{1, \ldots, w\}$,

the attacker injects a contaminated dataset $\mathcal{DS}_{\text{t}}^{\text{Crafted}}$ into $\mathcal{DS}_{\text{t}}^{\text{Clean}}$, resulting in the dataset $\mathcal{DS}_{\text{t}}^{\text{c}}$. This resulting dataset is then sent in as a new source of information. The defender receives $\mathcal{DS}_{\text{t}}^{\text{c}}$ and retrains the validated model, B_1, by creating the union of $\mathcal{DS}^{\text{v}}$ and the new incoming dataset $\mathcal{DS}_{\text{t}}^{\text{c}}$ and feeding them to the PC algorithm, resulting in the model $B_2^{'}$ (i.e., $B_2^{'} = PC\_Algo(\mathcal{DS}^{\text{v}} \cup \mathcal{DS}_{\text{t}}^{\text{c}})$).

The results of our experiments are presented in Table 8.1. In all of the scenarios, the attacker succeeded in achieving the desired modification. In our experiments, we assumed that $t = \{1, \ldots, 4\}$. For every one of the studied long-duration attacks on the dataset $\mathcal{DS}^{\text{v}}$ (Tables 8.1a, 8.1b, 8.1c, 8.1d, 8.1e, and 8.1f), the adversary had to send in the attack over 4 datasets. That is, at every time point t (for $t = 1, \ldots, 4$), the attacker had to create the union of $\mathcal{DS}_{\text{t}}^{\text{Clean}}$ and $\mathcal{DS}_{\text{t}}^{\text{Crafted}}$ resulting in $\mathcal{DS}_{\text{t}}^{\text{c}}$, which was going to be sent to the targeted machine learning system as a new source of information. The defender, on the other hand, retrained the machine learning model every time a new incoming dataset $\mathcal{DS}_{\text{t}}^{\text{c}}$ arrived.

Note that, in our experiments, long-duration attacks require the same number of contaminated cases as the one-step data poisoning attacks. An important observation is that the malicious attacker does not always have to send poisoned cases in the last dataset that will trigger the attack. For instance, in our experiments, when introducing the link $A \rightarrow E$ (Table 8.1a), shielding collider E (Table 8.1b), and removing the weakest edge (Table 8.1f), the last contaminated dataset, $\mathcal{DS}_4^{\text{c}}$, had no contaminated cases, which makes it impossible for the defender to find what caused a change in the newly learned model.

Table 8.1: Results of long-duration data poisoning attacks against $\mathcal{DS}^{\text{v}}$.

(a) Introducing the link $A \rightarrow E$ in the diverging connection $A \leftarrow T \rightarrow E$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	3	1	3	0
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,003	2,004	2,007	2,007
$\mathcal{DS}_{t}^{\text{I\_d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,003	16,004	18,007	20,007
Model Change	No	No	No	Yes

(b) Breaking the v-structure $T \rightarrow E \leftarrow L$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	2	2	0	0
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,002	2,002	2,000	2,000
$\mathcal{DS}_{t}^{\text{I\_d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,002	16,004	18,004	20,004
Model Change	No	No	No	Yes

(c) Add the most believable edge, $B \rightarrow L$, to the causal model B_1.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	2	2	1	2
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,002	2,002	2,001	2,002
$\mathcal{DS}_{t}^{\text{I\_d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,002	16,004	18,005	20,007
Model Change	No	No	No	Yes

(d) Adding the link $D \rightarrow S$ to the serial connection $D \rightarrow B \rightarrow S$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	20	20	23	4
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,020	2,020	2,023	2,004
$\mathcal{DS}_{t}^{\text{I\_d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,020	16,040	18,063	20,067
Model Change	No	No	No	Yes

(e) Adding the link $S \rightarrow E$ to the serial connection $S \rightarrow L \rightarrow E$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	7	8	5	1
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,007	2,008	2,005	2,001
$\mathcal{DS}_{t}^{\text{I\_d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,007	16,015	18,020	20,021
Model Change	No	No	No	Yes

(f) Removing the weakest link, $T \rightarrow A$, from the causal model B_1.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	1	1	1	0
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,001	2,001	2,001	2,000
$\mathcal{DS}_{t}^{\text{I\_d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,001	16,002	18,003	20,003
Model Change	No	No	No	Yes

CHAPTER 9

DETECTING ADVERSARIAL ATTACKS IN THE CONTEXT OF BAYESIAN NETWORKS

9.1 INTRODUCTION

During the last decade, several researchers addressed the problem of cyber attacks against machine learning systems (see [35] for an overview). Machine learning techniques are widely used; however, machine learning methods were not designed to function correctly in adversarial settings [25, 28]. Data poisoning attacks are considered one of the most important emerging security threats against machine learning systems [51, 65]. Data poisoning attacks aim to corrupt the machine learning model by contaminating the data in the training phase [16]. Data poisoning was studied in different machine learning algorithms, such as Support Vector Machines (SVMs) [16, 42, 32], Principal Component Analysis (PCA) [15, 14], Clustering [17, 12], and Neural Networks (NNs) [69]. However, these efforts are not directly applicable to Bayesian structure learning algorithms.

There are two main methods used in defending against a poisoning attack: (1) robust learning and (2) data sanitization [20]. Robust learning aims to increase learning algorithm robustness, thereby reducing the overall influence that contaminated data samples have on the algorithm. Data sanitization eliminates contaminated data samples from the training data set prior to training a classifier. While data sanitization shows promise to defend against data poisoning, it is often impossible to validate every data source [20].

In this chapter, we use the causative model proposed by Barreno et al. [10] to contextualize Bayesian network vulnerabilities. We propose a 2-layered framework to detect

poisoning attacks from untrusted data sources. Layer 1 enforces “reject on negative impacts" detection [46]; i.e., input that changes the model is labeled malicious. Layer 2 aims to detect long-duration attacks; i.e., it looks for cases in the incoming data that conflict with the original Bayesian model.

The main contributions of this chapter are the following: We propose a 2-layered framework for detecting data poisoning attacks. Our 2-layered framework detects both one-step and long-duration data poisoning attacks. We use the distance between Bayesian network models, B_1 and B_2, denoted as $\mathbf{ds}(B_1, B_2)$, to detect malicious data input (Equation 9.3) for one-step attacks. For long-duration attacks, we use the value of data conflict (Equation 9.4) to detect potentially poisoned data. Our framework relies on offline analysis to validate the potentially malicious datasets. We present our empirical results, showing the effectiveness of our framework to detect both one-step and long-duration attacks. Our results indicate that the distance measure $\mathbf{ds}(B_1, B_2)$ (Equation 9.3) and the conflict measure $Conf(c, B_1)$ (Equation 9.4) are sensitive to poisoned data.

9.2 Problem Setting

We focus on structure learning algorithms in Bayesian networks. Let $\mathcal{DS}^{\mathrm{v}} = \{c_1, \ldots, c_N\}$ be a validated dataset with N case. Each case c is over attributes $x_1, \ldots, x_n$ and of the form $c =< x_1 = v_1, \ldots, x_n = v_n >$, where v_i is the value of attribute x_i. A Bayesian network model B_1 is learned by feeding a validated dataset $\mathcal{DS}^{\mathrm{v}}$ into a Bayesian structure learning algorithm, BN_Algo, such as the PC algorithm, which is the most widely used algorithm for structure learning in Bayesian networks [61], as shown in Equation 9.1.

$$B_1 = BN\_Algo(\mathcal{DS}^{\mathrm{v}}) \tag{9.1}$$

The defender attempts to divide an incoming dataset, $\mathcal{DS}^{\mathrm{p}}$, coming from an untrusted source, into clean and poisoned cases. The attacker aims to inject a contaminated dataset, $\mathcal{DS}^{\mathrm{p}}$ with the same attributes as $\mathcal{DS}^{\mathrm{v}}$ and N_1 cases, into the validated training dataset,

$\mathcal{DS}^{\mathrm{v}}$. A learning error occurs if $\mathcal{DS}^{\mathrm{u}}$, obtained by the union of $\mathcal{DS}^{\mathrm{v}}$ and $\mathcal{DS}^{\mathrm{p}}$, results in a Bayesian network learning model B_2 (shown in Equation 9.2), such that there is a missing link, a reversed link, or an additional link in B_2 that is not in B_1.

$$B_2 = BN\_Algo(\mathcal{DS}^{\mathrm{u}}) \tag{9.2}$$

To estimate the impact of the poisoned dataset on the validated dataset, we define a distance function between two Bayesian network models B_1 and B_2, denoted as $\mathbf{ds}(B_1, B_2)$. Intuitively, B_1 is the validated model and B_2 is the potentially corrupted model.

Let $B_1 = (V, E_1)$ and $B_2 = (V, E_2)$ be two Bayesian network models where $V = \{x_1, x_2, \ldots, x_n\}$ and $E = \{(x_u, x_v) : x_u, x_v \in V\}$. Let B_1 be the validated model resulting from feeding $\mathcal{DS}^{\mathrm{v}}$ to a Bayesian network structure learning algorithm, and B_2 be the newly learned model resulting from feeding $\mathcal{DS}^{\mathrm{u}}$ to a Bayesian network structure learning algorithm. Let $e_1 = (x_u, x_v)$ be a directed edge from vertex x_u to vertex x_v, and $e_2 = (x_v, x_u)$ be a directed edge from vertex x_v to vertex x_u (e_2 is the reverse of e_1). The distance function, $\mathbf{ds}(B_1, B_2)$, is a non-negative function that measures the changes in the newly learned model B_2 with respect to the original model B_1. The distance function, $\mathbf{ds}(B_1, B_2)$, is defined as follows:

(Distance measure) Let Bayesian network models $B_1 = (V, E_1)$ and $B_2 = (V, E_2)$ be the results of feeding $\mathcal{DS}^{\mathrm{v}}$ and $\mathcal{DS}^{\mathrm{u}}$, respectively, to a Bayesian network structure learning algorithm. $\mathbf{ds}(B_1, B_2)$ is defined as the sum of distances over pairs of vertices $(x_u, x_v) \in V \times V$ as follows:

$$\mathbf{ds}(B_1, B_2) = \sum_{(x_u, x_v) \in V \times V} \mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) \tag{9.3}$$

where $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2)$ is the distance between every pair of vertices $(x_u, x_v) \in V \times V$.

We define $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2)$ as the cost of making a change to B_1 that results in the newly learned model B_2. The function $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2)$ between the two Bayesian network models B_1 and B_2 is defined as follows [23]:

Status 1 (True Negative Edges): if $((e_1 \notin E_1 \;\&\&\; e_2 \notin E_1) \;\&\&\; (e_1 \notin E_2 \;\&\&\; e_2 \notin E_2))$, then there is no edge (neither e_1 nor e_2) between vertex x_u and vertex x_v in either models B_1 and B_2. Hence, $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 0$.

Status 2 (True Positive Edges): if $((e_1 \in E_1 \;\&\&\; e_1 \in E_2) \;||\; (e_2 \in E_1 \;\&\&\; e_2 \in E_2))$, then the same edge (either e_1 or e_2) appears from vertex x_u to vertex x_v in both models B_1 and B_2. Hence, $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 0$.

Status 3 (False Negative Edges): if $((e_1 \;||\; e_2 \in E_1) \;\&\&\; (e_1 \;\&\&\; e_2 \notin E_2))$, then there is an edge (either e_1 or e_2) from vertex x_u to vertex x_v in B_1 that does not exist in B_2. Without loss of generality, assume that the deleted edge from B_1 is e_1, then if the indegree of vertex x_v, denoted as $indegree(x_v)$, which is the number if edge incoming to vertex x_v, is greater than 1, then $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 8$; otherwise, $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 4$.

Status 4 (False Positive Edges): if $((e_1 \;\&\&\; e_2 \notin E_1) \;\&\&\; (e_1 \;||\; e_2 \in E_2))$, then there is an edge (either e_1 or e_2) from vertex x_u to vertex x_v in B_2 but not the in B_1. Without loss of generality, assume that the added edge to B_2 is e_1, then if the indegree of vertex x_v, is greater than 1, then $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 8$; otherwise, $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 4$.

Status 5 (False Positive and True Negative Edges): if $((e_1 \in E_1 \;\&\&\; e_2 \in E_2) \;\&\&\; (e_1 \in E_2 \;\&\&\; e_2 \in E_1))$, then the edge from vertex x_u to vertex x_v in B_1 is the reverse of the edge from vertex x_u to vertex x_v in B_2. Without loss of generality, assume that there is an edge, e_1, from x_u to x_v in B_1, then e_2 is the reverse of e_1 in B_2. If the indegree of vertex x_u, is greater than 1, then $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 8$; otherwise, $\mathbf{ds_{x_u x_v}}(B_1, B_2) = 2$.

To investigate the coherence of an instance case, $c =< x_1 = v_1, \ldots, x_n = v_n >$ (or simply $< v_1, \ldots, v_n >$), in $\mathcal{DS}^{\mathrm{p}}$ with the validated model B_1, we use *conflict measure*,

denoted as $Conf(c, B_1)$. Conflict measure, $Conf(c, B_1)$, is defined as follows:

(Conflict measure) Let B_1 be a Bayesian network model and let $\mathcal{DS}^{\mathrm{p}}$ be an incoming dataset, $Conf(c, B_1)$ is defined as the process of detecting how well a given case $< v_1, \ldots, v_n >$ fits the model B_1 according to the following equation:

$$Conf(c, B_1) = log_2 \frac{P(v_1) \ldots P(v_n)}{P(v)} \tag{9.4}$$

where $c =< v_1, \ldots, v_n >$, and $P(v)$ is the prior probability of the evidence v [48].

If $P(v) = 0$, then we conclude that there is inconsistency among the observations $< v_1, \ldots, v_n >$. If the value of $Conf(c, B_1)$ is positive, then we can conclude that $< v_1, \ldots, v_n >$ are negatively correlated (i.e., unlikely to be correlated as the model requires) and thus are conflicting with the model B_1. The higher the value of $Conf(c, B_1)$ is, the more incompatibility we have between B_1 and $< v_1, \ldots, v_n >$.

In this paper, we adopt the causative model proposed by Barreno et al. [10]. Attacks on machine learning systems are modeled as a game between malicious attackers and defenders. In our setting, defenders aim to learn a validated Bayesian network model B_1 using the dataset $\mathcal{DS}^{\mathrm{v}}$ with the fewest number of errors (minimum $\mathbf{ds}$ function). Malicious attackers aim to mislead the defender into learning a contaminated model B_2 using the dataset $\mathcal{DS}^{\mathrm{u}}$, obtained by polluting $\mathcal{DS}^{\mathrm{v}}$ with $\mathcal{DS}^{\mathrm{p}}$. We assume that malicious attackers have full knowledge of how Bayesian network structure learning algorithms work. Also, we assume that attackers have knowledge of the dataset $\mathcal{DS}^{\mathrm{v}}$. In addition, we assume that the poisoning percentage at which attackers are allowed to add new "contaminated" cases to $\mathcal{DS}^{\mathrm{v}}$, β, is less than or equal to 0.05. The game between malicious attackers and defenders can be modeled as follows:

1. **The defender:** The defender uses a validated dataset $\mathcal{DS}^{\mathrm{v}}$, to produce a validated Bayesian network model B_1.

2. **The malicious attacker:** The attacker injects a contaminated dataset, $\mathcal{DS}^{\mathrm{p}}$, to be

Table 9.1: Notations

Notation	Description
$\mathcal{DS}[x_1, \ldots, x_n]$	Schema for datasets with attributes $x_1, \ldots, x_n$
$\mathcal{DS}^{\mathrm{v}} = \{c_1, \ldots, c_N\}$	Validated dataset instance with attributes $x_1, \ldots, x_n$
$\mathcal{DS}^{\mathrm{p}} = \{c_1, \ldots, c_{N_1}\}$	Crafted dataset instance with attributes $x_1, \ldots, x_n$
β	Data poisoning percentage for $\mathcal{DS}^{\mathrm{v}}$
B_1	The result of feeding $\mathcal{DS}^{\mathrm{v}}$ to a learning algorithm
B_2	The result of feeding $\mathcal{DS}^{\mathrm{u}}$ to a learning algorithm
$\mathcal{DS}^{\mathrm{c}}_{\mathrm{i}} = \{c_1, \ldots, c_{N_i}\}$	Contaminated dataset instance at time point i
λ_i	Data poisoning rate for $\mathcal{DS}^{\mathrm{c}}_{\mathrm{i}}$
$\mathbf{ds}(B_1, B_2)$	Distance function between models B_1 and B_2
$Conf(c, B_1)$	Conflict measure of how well the case c fits B_1

unioned with the original dataset, $\mathcal{DS}^{\mathrm{v}}$, with the goal of changing the Markov equivalence class of the original validated model, B_1.

3. **Evaluation by the defender:**

   - The defender feeds the new dataset $\mathcal{DS}^{\mathrm{u}}$ (Note that, $\mathcal{DS}^{\mathrm{u}} = \mathcal{DS}^{\mathrm{v}} \cup \mathcal{DS}^{\mathrm{p}}$) to a Bayesian network structure learning algorithm, resulting in B_2.
   - The defender calculates the distance function $\mathbf{ds}(B_1, B_2)$.
   - If $\mathbf{ds}(B_1, B_2) = 0$, then Bayesian models B_1 and B_2 are identical. Otherwise, i.e., $\mathbf{ds}(B_1, B_2) > 0$, the newly learned Bayesian model B_2 is different from the original validated model B_1.
   - For each case c, the defender calculates the value of conflict measure $Conf(c, B_1)$.
   - If $Conf(c, B_1)$ is positive, then the case c conflict with the Bayesian model B_1. Otherwise, the newly incoming case is validated and added to $\mathcal{DS}^{\mathrm{v}}$.

Note, that the goal of malicious attackers is to maximize the quantity $\mathbf{ds}(B_1, B_2)$. The notations used in this chapter are summarized in Table 9.1.

9.3 FRAMEWORK FOR DETECTING DATA POISONING ATTACKS

In this section, we present our detective framework for data poisoning attacks. Our techniques build on the data sanitization approach that was proposed by Nelson et al. [46]. We extend Nelson et al. approach such that it is applicable to detect both one-step and long-duration causative attacks.

The main components of our framework are: (1) Structure learning Algorithms: the PC learning algorithm, (2) FLoD: first layer of detection, and (3) SLoD: second layer of detection.

First Layer of Detection: In the FLoD, our framework uses "Reject On Negative Impact" defense [46] to examine the full dataset ($\mathcal{DS}^{\text{v}} \cup \mathcal{DS}^{\text{p}}$) to detect the impact of $\mathcal{DS}^{\text{p}}$ on $\mathcal{DS}^{\text{v}}$. The attacker aims to use $\mathcal{DS}^{\text{p}}$ to change the Markov equivalence class of the validated model, B_1. The first layer of detection detects the impact of adversarial attacks that aim to corrupt the model B_1 using one-step data poisoning attacks.

In the FLoD, we use the distance function $\mathbf{ds}$ described in section 9.2 as a method for detecting the negative impact of $\mathcal{DS}^{\text{p}}$ on the validated model B_1. If $\mathbf{ds}(B_1, B_2)$ is greater than zero, then the new incoming dataset, $\mathcal{DS}^{\text{p}}$, is potentiality malicious. In this case, we sent $\mathcal{DS}^{\text{p}}$ to be checked offline. Otherwise, we proceed with the second layer of detection, SLoD, looking for long-duration data poisoning attacks.

Algorithm 9 provides algorithmic details of FLoD detect one-step data poisoning attacks.

Second Layer of Detection: In the SLoD, our framework uses "Data Conflict Analysis" [48] to examine the newly incoming dataset $\mathcal{DS}^{\text{p}}$ to detect if $\mathcal{DS}^{\text{p}}$ has conflicting cases with the original model B_1. The Second layer of detection detects sophisticated adversarial attacks that aim to corrupt the model B_1, such as long-duration data poisoning attacks.

In the SLoD, we use the value of the conflict measure $Conf(c, B_1)$ described in section 9.2 as a method for detecting whether or not a case, c, in the newly incoming dataset, $\mathcal{DS}^{\text{p}}$, is conflicting with the original model B_1. If the $P(v)$ is equal to zero, then the case

Algorithm 9: First Layer of Detection

Input : $\mathcal{DS}^{\text{v}} = \{c_1, \ldots, c_N\}$ and $\mathcal{DS}^{\text{p}} = \{\bar{c_1}, \ldots, \bar{c}_{N_1}\}$
Output: $\mathbf{ds}(B_1, B_2)$

Generate B_1 from $\mathcal{DS}^{\text{v}}$;
Generate B_2 from $\mathcal{DS}^{\text{v}} \cup \mathcal{DS}^{\text{p}}$;
Calculate $\mathbf{ds}(B_1, B_2)$ ▷ as described in section 9.2;
if $\mathbf{ds}(B_1, B_2) > 0$ **then**
    Return $\mathbf{ds}(B_1, B_2)$;
    Send $\mathcal{DS}^{\text{p}}$ to be checked offline;
else
    Go to Algorithm 10;
end

c is inconsistent with the validated model B_1. If $Conf(c, B_1)$ is positive, then the case c is incompatible with the validated model B_1. In these two situations, we add inconsistent and incompatible cases to $\mathcal{DS}^{\text{conf}}$. $\mathcal{DS}^{\text{conf}}$ is then sent to be checked offline. Thereby, the model B_1 will be retrained according to the following equation: $B_1 = BN\_Algo(\mathcal{DS}^{\text{v}})$ where $\mathcal{DS}^{\text{v}} = \mathcal{DS}^{\text{v}} \cup (\mathcal{DS}^{\text{p}} \backslash \mathcal{DS}^{\text{conf}})$.

Algorithm 10 provides algorithmic details of the SLoD detect long-duration data poisoning attacks.

The process of applying our framework is summarized in Figure 9.1. The workflow of our framework is described as follows: (1) A validated dataset, $\mathcal{DS}^{\text{v}}$, which is a clean training dataset that is used to recover a validated machine learning model B_1. (2) A new incoming dataset, $\mathcal{DS}^{\text{p}}$, which is coming from an untrusted source and a potentially malicious dataset, is used along with $\mathcal{DS}^{\text{v}}$ to learn B_2. (3) FLoD checks for one-step data poisoning attacks. If model change occurs (i.e., $\mathbf{ds}(B_1, B_2) > 0$), send $\mathcal{DS}^{\text{p}}$ for offline evaluation. Else, (4) SLoD checks for long-duration data poisoning attacks. If the value of conflict measure is positive (i.e., $Conf(c, B_1) > 0$), send conflicting data to offline evaluation. Else, update the validated dataset.

Algorithm 10: Second Layer of Detection

Input : $\mathcal{DS}^{v} = \{c_1, \ldots, c_N\}$ and $\mathcal{DS}^{p} = \{\bar{c}_1, \ldots, \bar{c}_{N_1}\}$
Output: $\mathcal{DS}^{v}$, $\mathcal{DS}^{conf}$.

Generate B_1 from $\mathcal{DS}^{v}$;
$\mathcal{DS}^{conf} = \phi$;
for *every case* c *in* $\mathcal{DS}^{p}$ **do**
    Calculate $P(v)$ ▷ i.e., the probability of the evidence for c;
    if $P(v) = 0$ **then**
        $\mathcal{DS}^{conf} = \mathcal{DS}^{conf} \cup \{c\}$ ▷ i.e., c is inconsistent with B_1;
        $\mathcal{DS}^{p} = \mathcal{DS}^{p} \setminus \{c\}$ ▷ remove c from $\mathcal{DS}^{p}$;
    else
        $Conf(c, B_1) = log_2 \frac{P(v_1)...P(v_n)}{P(v)}$ ▷ calculate conflict measure for the case c;
        if $Conf(c, B_1) > 0$ **then**
            $\mathcal{DS}^{conf} = \mathcal{DS}^{conf} \cup \{c\}$ ▷ i.e., c is incompatible with B_1;
            $\mathcal{DS}^{p} = \mathcal{DS}^{p} \setminus \{c\}$;
        end
    end
    if $\mathcal{DS}^{conf} \neq \phi$ **then**
        Send $\mathcal{DS}^{conf}$ to be checked offline;
    end
    $\mathcal{DS}^{v} = \mathcal{DS}^{v} \cup (\mathcal{DS}^{p} \setminus \mathcal{DS}^{conf})$;
    Return $\mathcal{DS}^{v}$, $\mathcal{DS}^{conf}$;
end

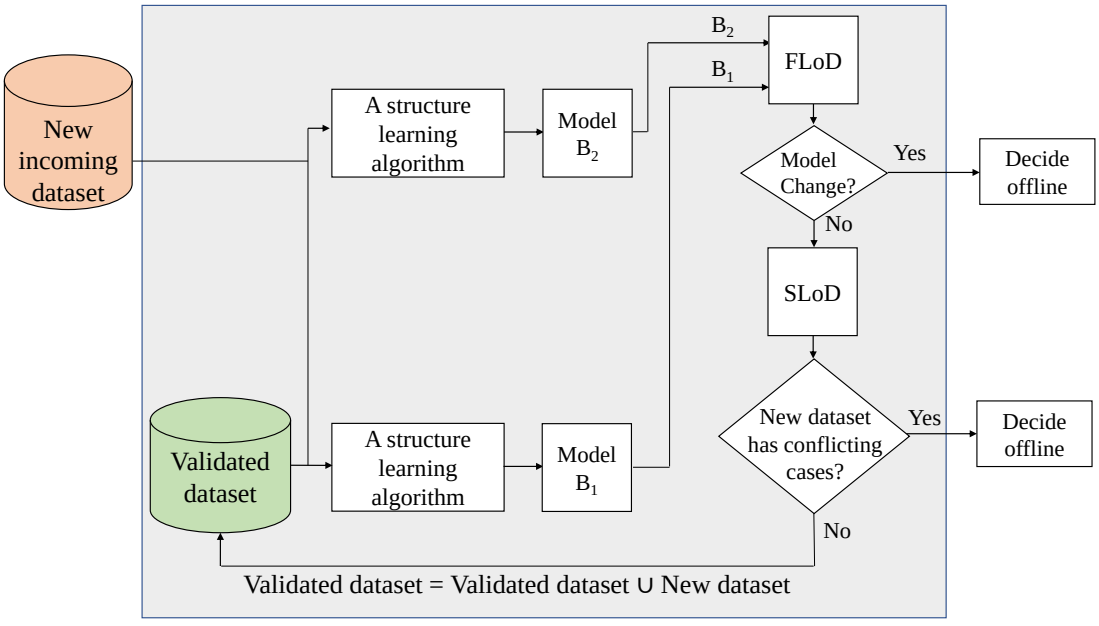


Figure 9.1: Framework for detecting data poisoning attacks.

9.4 Empirical Results

We implemented our prototype system using the Chest Clinic Network [34]. We used the Chest Clinic Network to demonstrate the data poisoning attacks and our detection capabilities. In each experiment, we manually generated poisoned datasets. Given the contingency table of two random variables A and B in a Bayesian network model with i and j states, respectively. To introduce a malicious link between A and B, we add corrupt cases to the cell with the highest test statistic value in the contingency table. To remove the link between A and B, we transfer cases from the cell with the highest test statistics value to the one with the lowest value.

9.4.1 Discussion: Detecting Data Poisoning Attacks

The results of using our framework to detect one-step data poisoning attacks are presented in Table 9.2. Algorithm 9 succeeded to detect the negative impact (i.e., the change in the Markov equivalence class) of the new incoming dataset $\mathcal{DS}^{\mathrm{p}}$ on the validated model B_1.

Table 9.2: Results of using FLoD to detect one-step data poisoning attacks.

Attack	Attack's class	$\mathbf{ds}(B_1, B_2)$ score
Introduce the link $A \rightarrow E$	New v-structure	12
Introduce the link $D \rightarrow S$	New v-structure	24
Introduce the link $S \rightarrow E$	New v-structure	54
Introduce the link $T \rightarrow L$	Shield an existing collider	16
Remove the link $A \rightarrow T$	Delete the weakest link	4
Introduce the link $B \rightarrow L$	Add the most believable link	32

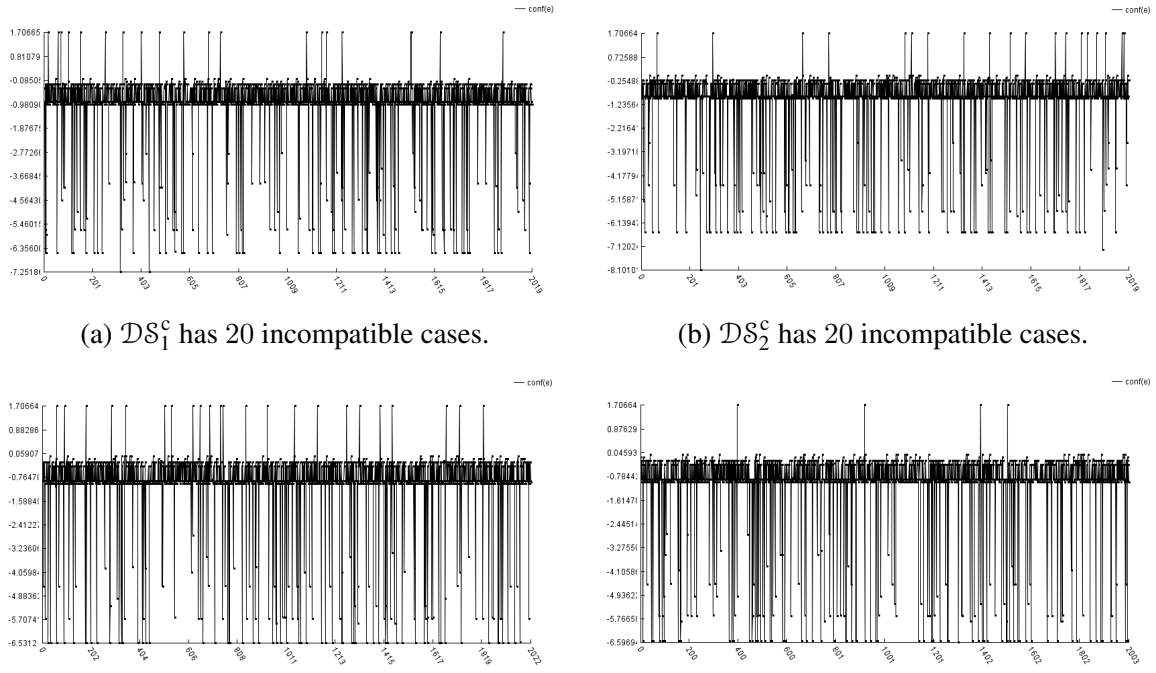
The results using our framework to detect long-duration data poisoning attacks are summarized in Table 9.3. Algorithm 10 succeeded to detect the long-duration impact of $\mathcal{DS}^{\mathrm{c}}$ on the validated dataset $\mathcal{DS}^{\mathrm{v}}$. Note, that FLoD using traditional reject on negative impact was not able to detect long-duration attacks. However, when using the SLoD, we were able to detect the conflicting cases, which are either inconsistent or incompatible with the original validated model B_1 (A detailed experiment is presented in Figure 9.2). Such

cases might be exploited by a malicious adversary to trigger the long-duration attack at a later time. Also, in some attacks no poisoned cases are even required to be sent with $\mathcal{DS}^{c}$ to trigger the long-duration attack, which is very hard to detect.

In summary, our 2-layered approach was able to detect both one-step and long-duration attacks. Moreover, our solution did not lose all the incoming datasets; we only send conflicting cases to be checked offline. We have carried out over 200 experiments for long-duration attacks. A comprehensive description of these experiments is given in [5].

Table 9.3: Results of using SLoD to detect long-duration data poisoning attacks.

Attack	Attack's class	Algorithm 10 decision
Introduce $A \rightarrow E$	New v-structure	Inconsistent observations
Introduce $D \rightarrow S$	New v-structure	Incompatible observations
Introduce $S \rightarrow E$	New v-structure	Inconsistent observations
Introduce $T \rightarrow L$	Shield an existing collider	Inconsistent observations
Remove $A \rightarrow T$	Delete weakest link	Inconsistent\Incompatible observations
Introduce $B \rightarrow L$	Add most believable link	Inconsistent observations



(a) $\mathcal{DS}_1^{\mathrm{c}}$ has 20 incompatible cases.

(b) $\mathcal{DS}_2^{\mathrm{c}}$ has 20 incompatible cases.

(c) $\mathcal{DS}_3^{\mathrm{c}}$ has 23 incompatible cases.

(d) $\mathcal{DS}_4^{\mathrm{c}}$ has 4 incompatible cases.

Figure 9.2: The result of using **SLoD** to detect a long-duration attack that aims to introduce the link $D \rightarrow S$ in the Chest Clinic dataset, $\mathcal{DS}^{\mathrm{v}}$. We present the case number in $\mathcal{DS}_{\mathrm{t}}^{\mathrm{c}}$ as the variable on the X-axis and the value of our conflict measure $Conf(c, B_1)$ as the variable on the Y-axis. A case is incompatible (conflicting) with the validated model B_1 if $Conf(c, B_1) > 0$.

CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 CONCLUSION

As machine learning techniques become more pervasive, it is important to be aware of the danger of malicious attackers based on introducing corrupted data items. In this dissertation, we demonstrated the vulnerability of structural learning algorithms for Bayesian networks to adversarial attacks [7]. We have developed a theoretical framework to classify data poisoning attacks against the Bayesian network structure learning algorithms [4]. We proposed a novel measure of link strength that is useful for security analysis in the context of Bayesian networks. We demonstrated the vulnerability of the PC algorithm against one-step and long-duration data poisoning attacks. We proposed a 2-layered framework for detecting data poisoning attacks.

We implemented our approaches using the Chest Clinic Network which is a widely used network in Bayesian networks. Our findings indicate that Bayesian network structure learning algorithms are highly sensitive to data poisoning attacks. We also demonstrated that attackers could corrupt the learning outcome in a way that structural learning algorithms will learn the desired structure. Our novel link strength measure plays a crucial role in identifying vulnerable network structure and the ease of corrupting the Bayesian model. Our results also indicate that Bayesian network structure learning algorithms are vulnerable to both one-step and long-duration data poisoning attacks. Our framework is effective in detecting both one-step and long-duration data poisoning attacks, as it thoroughly validates and verifies training data before such data is being incorporated into the model.

10.2 FUTURE WORK

Future work of this dissertation are as follows:

1. We will investigate the robustness of the LCD algorithm against long-duration data poisoning attacks. We will follow the same schema like the PC algorithm. That is, we will investigate long-duration model invalidation attacks and targeted change attack that aim to corrupt the validated Bayesian network model over time.

2. We aim to focus on offline validation of potentially malicious datasets. Currently, our approach detects datasets that either change the Bayesian network structure (distance measure) or in conflict with the validated model (conflict measure). We are investigating methods for (1) distinguishing actual model shift from model enrichment, i.e., our initial model was based on data that was not fully representative of the "true" distribution, and (2) determining if cases are truly conflicting or again if the initial model poorly approximates the "true" distribution.

3. We will also investigate the applicability of Wisdom of the Crowd (WoC) [70]. Rather than human experts, we plan to use an ensemble of classifiers, i.e., take the votes of competing algorithms instead of the votes of humans. In the case of an ensemble of classifiers, one could investigate the likelihood of unexpected cases and adjust the sensitivity to anomalies by how much perturbation it causes in the model.

4. We will investigate the possibility of using our link strength measure for defending against data poisoning attacks. In particular, we aim to monitor change in link-strength and find pattern of malicious (misuse) activities.

BIBLIOGRAPHY

[1] Hirotugu Akaike, *A new look at the statistical model identification*, IEEE transactions on automatic control **19** (1974), no. 6, 716–723.

[2] Scott Alfeld, Xiaojin Zhu, and Paul Barford, *Data poisoning attacks against autoregressive models.*, AAAI, 2016, pp. 1452–1458.

[3] Ayesha R Ali, Thomas S Richardson, Peter L Spirtes, and Jiji Zhang, *Towards characterizing markov equivalence classes for directed acyclic graphs with latent variables*, arXiv preprint arXiv:1207.1365 (2012).

[4] Emad Alsuwat, Hatim Alsuwat, John Rose, Marco Valtorta, and Csilla Farkas, *Detecting adversarial attacks in the context of bayesian networks*, IFIP Annual Conference on Data and Applications Security and Privacy, Springer, 2019, pp. 3–22.

[5] _____, *Long duration data poisoning attacks on Bayesian networks*, Tech. report, University of South Carolina, SC, USA, 2019.

[6] Emad Alsuwat, Hatim Alsuwat, Marco Valtorta, and Csilla Farkas, *Cyber attacks against the pc learning algorithm*, 2nd International Workshop on A.I. in Security, 2018, pp. 19 – 35.

[7] 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.

[8] Emad Alsuwat, Marco Valtorta, and Csilla Farkas, *Bayesian structure learning attacks*, Tech. report, University of South Carolina, SC, USA, 2018.

[9] _____, *How to generate the network you want with the pc learning algorithm*, Proceedings of the 11th Workshop on Uncertainty Processing (WUPES'18), 2018, pp. 1 – 12.

[10] Marco Barreno, Blaine Nelson, Anthony D. Joseph, and J. D. Tygar, *The security of machine learning*, Machine Learning **81** (2010), no. 2, 121–148.

[11] Marco Barreno, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar, *Can machine learning be secure?*, Proceedings of the 2006 ACM Symposium on Information, computer and communications security, ACM, 2006, pp. 16–25.

[12] Battista Biggio, Samuel Rota Bulò, Ignazio Pillai, Michele Mura, Eyasu Zemene Mequanint, Marcello Pelillo, and Fabio Roli, *Poisoning complete-linkage hierarchical clustering*, Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2014, pp. 42–52.

[13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli, *Evasion attacks against machine learning at test time*, Joint European conference on machine learning and knowledge discovery in databases, Springer, 2013, pp. 387–402.

[14] Battista Biggio, Luca Didaci, Giorgio Fumera, and Fabio Roli, *Poisoning attacks to compromise face templates*, Biometrics (ICB), 2013 International Conference on, IEEE, 2013, pp. 1–7.

[15] Battista Biggio, Giorgio Fumera, Fabio Roli, and Luca Didaci, *Poisoning adaptive biometric systems*, Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Springer, 2012, pp. 417–425.

[16] Battista Biggio, Blaine Nelson, and Pavel Laskov, *Poisoning attacks against support vector machines*, arXiv preprint arXiv:1206.6389 (2012).

[17] Battista Biggio, Ignazio Pillai, Samuel Rota Bulò, Davide Ariu, Marcello Pelillo, and Fabio Roli, *Is data clustering in adversarial settings secure?*, Proceedings of the 2013 ACM workshop on Artificial intelligence and security, ACM, 2013, pp. 87–98.

[18] Brent Boerlage, *Link strength in bayesian networks*, Ph.D. thesis, University of British Columbia, 1992.

[19] Cody Burkard and Brent Lagesse, *Analysis of causative attacks against svms learning from data streams*, Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics, ACM, 2017, pp. 31–36.

[20] Patrick PK Chan, Zhi-Min He, Hongjiang Li, and Chien-Chang Hsu, *Data sanitization against adversarial label contamination based on data complexity*, International Journal of Machine Learning and Cybernetics **9** (2018), no. 6, 1039–1052.

[21] Gregory F Cooper and Edward Herskovits, *A bayesian method for the induction of probabilistic networks from data*, Machine learning **9** (1992), no. 4, 309–347.

[22] Denver Dash and Marek J Druzdzel, *A hybrid anytime algorithm for the construction of causal models from sparse data*, Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., 1999, pp. 142–149.

[23] Martijn de Jongh and Marek J Druzdzel, *A comparison of structural distance measures for causal bayesian network models*, Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science series (2009), 443–456.

[24] Imme Ebert-Uphoff, *Tutorial on how to measure link strengths in discrete bayesian networks*, Tech. report, Georgia Institute of Technology, 2009.

[25] Joseph Gardiner and Shishir Nagaraja, *On the security of machine learning in malware c&c detection: A survey*, ACM Computing Surveys (CSUR) **49** (2016), no. 3, 59.

[26] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, *Explaining and harnessing adversarial examples*, arXiv preprint arXiv:1412.6572 (2014).

[27] David Heckerman, Dan Geiger, and David M Chickering, *Learning bayesian networks: The combination of knowledge and statistical data*, Machine learning **20** (1995), no. 3, 197–243.

[28] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and JD Tygar, *Adversarial machine learning*, Proceedings of the 4th ACM workshop on Security and artificial intelligence, ACM, 2011, pp. 43–58.

[29] A Hugin Expert, *S, 2008*, Hugin Researcher API 7.0 (www. hugin. com).

[30] Michael Irwin Jordan, *Learning in graphical models*, vol. 89, Springer Science & Business Media, 1998.

[31] Alex Kantchelian, JD Tygar, and Anthony Joseph, *Evasion and hardening of tree ensemble classifiers*, International Conference on Machine Learning, 2016, pp. 2387–2396.

[32] Pang Wei Koh and Percy Liang, *Understanding black-box predictions via influence functions*, arXiv preprint arXiv:1703.04730 (2017).

[33] Pavel Laskov et al., *Practical evasion of a learning-based classifier: A case study*, Security and Privacy (SP), 2014 IEEE Symposium on, IEEE, 2014, pp. 197–211.

[34] 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.

[35] Qiang Liu, Pan Li, Wentao Zhao, Wei Cai, Shui Yu, and Victor CM Leung, *A survey on security threats and defensive techniques of machine learning: a data driven view*, IEEE access **6** (2018), 12103–12117.

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

[37] Zongming Ma, Xianchao Xie, and Zhi Geng, *Structural learning of chain graphs via decomposition*, Journal of Machine Learning Research **9** (2008), no. Dec, 2847–2880.

[38] 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.

[39] Mary L McHugh, *The chi-square test of independence*, Biochemia medica: Biochemia medica **23** (2013), no. 2, 143–149.

[40] Shike Mei and Xiaojin Zhu, *The security of latent dirichlet allocation*, Artificial Intelligence and Statistics, 2015, pp. 681–689.

[41] ———, *Some submodular data-poisoning attacks on machine learners*, (2015).

[42] ———, *Using machine teaching to identify optimal training-set attacks on machine learners.*, AAAI, 2015, pp. 2871–2877.

[43] Luis Muñoz-González, Battista Biggio, Ambra Demontis, Andrea Paudice, Vasin Wongrassamee, Emil C Lupu, and Fabio Roli, *Towards poisoning of deep learning algorithms with back-gradient optimization*, Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, ACM, 2017, pp. 27–38.

[44] Kevin P Murphy, *Machine learning: A probabilistic perspective*, MIT Press, 2012.

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

[46] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D Joseph, Benjamin IP Rubinstein, Udam Saini, Charles Sutton, JD Tygar, and Kai Xia, *Misleading learners: Co-opting your spam filter*, Machine learning in cyber trust, Springer, 2009, pp. 17–51.

[47] Andrew Newell, Rahul Potharaju, Luojie Xiang, and Cristina Nita-Rotaru, *On the practicality of integrity attacks on document-level sentiment analysis*, Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop, ACM, 2014, pp. 83–93.

[48] Thomas Dyhre Nielsen and Finn Verner Jensen, *Bayesian networks and decision graphs*, Springer Science & Business Media, 2009.

[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] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow, *Transferability in machine learning: from phenomena to black-box attacks using adversarial samples*, arXiv preprint arXiv:1605.07277 (2016).

[51] Andrea Paudice, Luis Muñoz-González, Andras Gyorgy, and Emil C Lupu, *Detection of adversarial training examples in poisoning attacks through anomaly detection*, arXiv preprint arXiv:1802.03041 (2018).

[52] Judea Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference. 1988*, 1988.

[53] ———, *Causality*, Cambridge university press, 2009.

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

[55] Brian C Ross, *Mutual information between discrete and continuous data sets*, PloS one **9** (2014), no. 2, e87357.

[56] Gideon Schwarz et al., *Estimating the dimension of a model*, The annals of statistics **6** (1978), no. 2, 461–464.

[57] Marco Scutari, *Learning bayesian networks with the bnlearn r package*, arXiv preprint arXiv:0908.3817 (2009).

[58] C Shannan and W Weaver, *The mathematical theory of communication. university of illinois press*, Urbana, USA (1963), 117.

[59] Moninder Singh and Marco Valtorta, *Construction of bayesian network structures from data: a brief survey and an efficient algorithm*, International journal of approximate reasoning **12** (1995), no. 2, 111–131.

[60] Peter Spirtes and Clark Glymour, *An algorithm for fast recovery of sparse causal graphs*, Social science computer review **9** (1991), no. 1, 62–72.

[61] Peter Spirtes, Clark N Glymour, and Richard Scheines, *Causation, prediction, and search*, MIT press, 2000.

[62] Harald Steck, *Constraint-based structural learning in bayesian networks using finite data sets*, Ph.D. thesis, Technische Universität München, Universitätsbibliothek, 2001.

[63] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus, *Intriguing properties of neural networks*, arXiv preprint arXiv:1312.6199 (2013).

[64] Thomas Verma and Judea Pearl, *Equivalence and synthesis of causal models*, UCLA, Computer Science Department, 1991.

[65] Yizhen Wang and Kamalika Chaudhuri, *Data poisoning attacks against online learning*, arXiv preprint arXiv:1808.08994 (2018).

[66] Han Xiao, Huang Xiao, and Claudia Eckert, *Adversarial label flips attack on support vector machines.*, ECAI, 2012, pp. 870–875.

[67] Huang Xiao, Battista Biggio, Blaine Nelson, Han Xiao, Claudia Eckert, and Fabio Roli, *Support vector machines under adversarial label contamination*, Neurocomputing **160** (2015), 53–62.

[68] Xianchao Xie, Zhi Geng, and Qiang Zhao, *Decomposition of structural learning about directed acyclic graphs*, Artificial Intelligence **170** (2006), no. 4-5, 422–439.

[69] Chaofei Yang, Qing Wu, Hai Li, and Yiran Chen, *Generative poisoning attack method against neural networks*, arXiv preprint arXiv:1703.01340 (2017).

[70] Sheng Kung Michael Yi, Mark Steyvers, Michael D Lee, and Matthew J Dry, *The wisdom of the crowd in combinatorial problems*, Cognitive science **36** (2012), no. 3, 452–470.

APPENDIX A

COMPUTATIONS OF POSTERIOR DISTRIBUTIONS

In order to run our experiments, we need to compute the posterior distribution (Beta distribution) for each link in the Bayesian network model. Posterior distribution is proportional to the prior distribution (Beta distribution) times the likelihood function (Binomial distribution).

In our working example, which is the Chest Clinic network, we will calculate the posterior distribution for each link in the network assuming a completely uninformative prior, i.e., a uniform distribution, Beta(1, 1), and using the dataset DB_1 to calculate the likelihood function. For each edge, given a statistical table from our data set DB_1 and uninformative prior Beta(1,1), we will calculate the posterior distribution using these equations:

$$Posterior = Prior \times Likelihood$$

$$P(\theta \mid y) = P(\theta) \times P(y \mid \theta)$$

$$P(\theta \mid y) = Beta(\alpha, \beta) \times Binomial(n, \theta)$$

$$P(\theta \mid y) = Beta(y + \alpha, n - y + \beta)$$

A.1 EDGES OF THE CHEST CLINIC NETWORK

In this appendix, we compute the posterior probability for each link in the original Chest Clinic Network as follows:

Table A.1: The contingency table of the observed counts of P($B \mid S$)

	S		
B	no	yes	missing
no	3898	1971	0
yes	1513	3018	0
missing	0	0	0

A.1.1 CALCULATING P($B \mid S$)

$$\begin{aligned} P(B = yes \mid S = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(4989, 3018) \\ &= Beta(3018 + 1, 4989 - 3018 + 1) \\ &= Beta(3019, 1972) \end{aligned}$$

$$\begin{aligned} P(B = yes \mid S = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(5411, 1513) \\ &= Beta(1513 + 1, 5411 - 1513 + 1) \\ &= Beta(1514, 3899) \end{aligned}$$

$$\begin{aligned} P(B = no \mid S = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(4989, 1971) \\ &= Beta(1971 + 1, 4989 - 1971 + 1) \\ &= Beta(1972, 3019) \end{aligned}$$

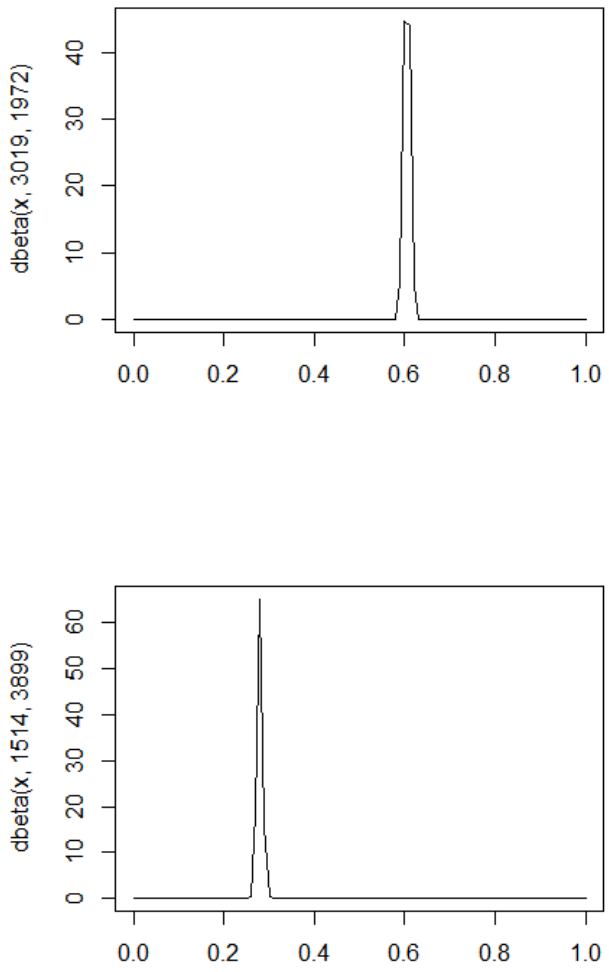


Figure A.1: Beta Distribution for P($B \mid S$)

$$
\begin{aligned}
P(B = no \mid S = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(5411, 3898) \\
&= Beta(3898 + 1, 5411 - 3898 + 1) \\
&= Beta(3899, 1514)
\end{aligned}
$$

Table A.2: The contingency table of the observed counts of P($L \mid S$)

	S		
L	no	yes	missing
no	4965	4509	0
yes	46	480	0
missing	0	0	0

A.1.2 CALCULATING P($L \mid S$)

$$
\begin{aligned}
P(L = yes \mid S = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(4989, 480) \\
&= Beta(480 + 1, 4989 - 480 + 1) \\
&= Beta(481, 4510)
\end{aligned}
$$

$$
\begin{aligned}
P(L = yes \mid S = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(5011, 46) \\
&= Beta(46 + 1, 5011 - -46 + 1) \\
&= Beta(47, 4966)
\end{aligned}
$$

$$
\begin{aligned}
P(L = no \mid S = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(4989, 4509) \\
&= Beta(4509 + 1, 4989 - -4509 + 1) \\
&= Beta(4510, 481)
\end{aligned}
$$

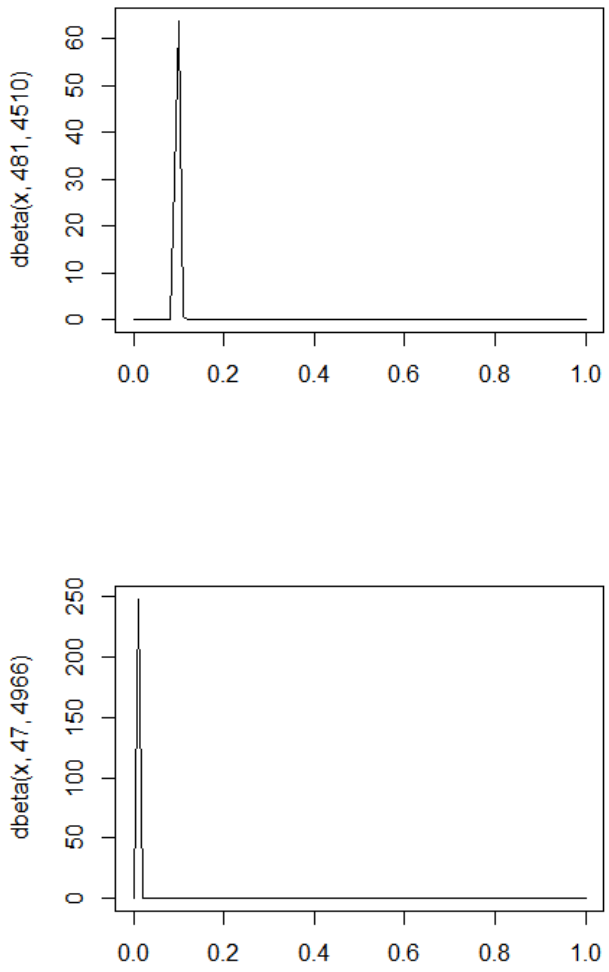


Figure A.2: Beta Distribution for P($L \mid S$)

$$
\begin{aligned}
P(L = no \mid S = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(5011, 4965) \\
&= Beta(4965 + 1, 5011 - -4965 + 1) \\
&= Beta(4966, 47)
\end{aligned}
$$

Table A.3: The contingency table of the observed counts of P($T \mid A$)

	A		
T	no	yes	missing
no	9788	98	0
yes	105	9	0
missing	0	0	0

A.1.3 CALCULATING P($T \mid A$)

$$\begin{aligned} P(T = yes \mid A = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(107, 9) \\ &= Beta(9 + 1, 107 - 9 + 1) \\ &= Beta(10, 99) \end{aligned}$$

$$\begin{aligned} P(T = yes \mid A = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(9893, 105) \\ &= Beta(105 + 1, 9893 - 105 + 1) \\ &= Beta(106, 9789) \end{aligned}$$

$$\begin{aligned} P(T = no \mid A = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(107, 98) \\ &= Beta(98 + 1, 107 - 98 + 1) \\ &= Beta(99, 10) \end{aligned}$$

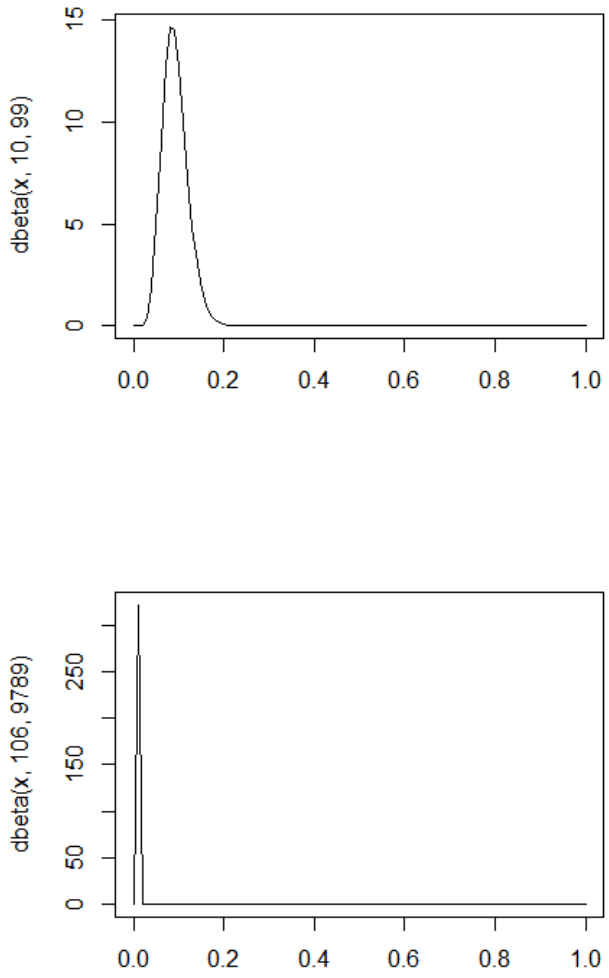


Figure A.3: Beta Distribution for P($T \mid A$)

$$
\begin{aligned}
P(T = no \mid A = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(9893, 9788) \\
&= Beta(9788 + 1, 9893 - 9788 + 1) \\
&= Beta(9789, 106)
\end{aligned}
$$

Table A.4: The contingency table of the observed counts of P($E \mid T$)

	T		
E	no	yes	missing
no	9364	0	0
yes	522	114	0
missing	0	0	0

A.1.4 CALCULATING P($E \mid T$)

$$
\begin{aligned}
P(E = yes \mid T = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(114, 114) \\
&= Beta(114 + 1, 114 - 114 + 1) \\
&= Beta(115, 1)
\end{aligned}
$$

$$
\begin{aligned}
P(E = yes \mid T = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(9866, 522) \\
&= Beta(522 + 1, 9893 - 522 + 1) \\
&= Beta(523, 9365)
\end{aligned}
$$

$$
\begin{aligned}
P(E = no \mid T = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(114, 0) \\
&= Beta(0 + 1, 114 - 0 + 1) \\
&= Beta(1, 115)
\end{aligned}
$$

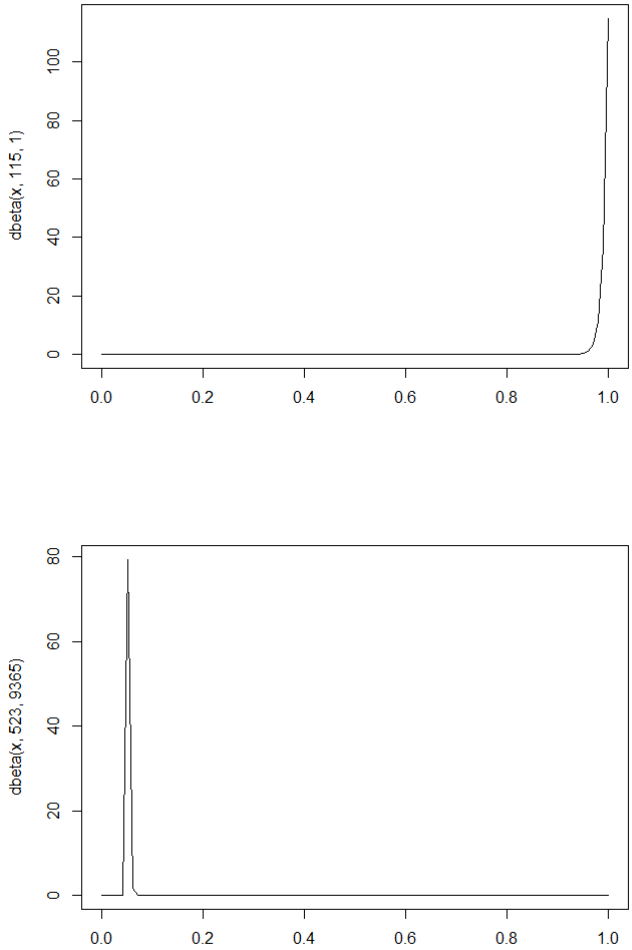


Figure A.4: Beta Distribution for P($E \mid T$)

$$
\begin{aligned}
P(E = no \mid T = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(9886, 9364) \\
&= Beta(9364 + 1, 9886 - 9364 + 1) \\
&= Beta(9365, 523)
\end{aligned}
$$

Table A.5: The contingency table of the observed counts of P($E \mid L$)

	L		
E	no	yes	missing
no	9364	0	0
yes	110	526	0
missing	0	0	0

A.1.5 CALCULATING P($E \mid L$)

$$
\begin{aligned}
P(E = yes \mid L = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(526, 526) \\
&= Beta(526 + 1, 526 - 526 + 1) \\
&= Beta(527, 1)
\end{aligned}
$$

$$
\begin{aligned}
P(E = yes \mid L = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(9474, 110) \\
&= Beta(110 + 1, 9474 - 110 + 1) \\
&= Beta(111, 9365)
\end{aligned}
$$

$$
\begin{aligned}
P(E = no \mid L = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(526, 0) \\
&= Beta(0 + 1, 526 - 0 + 1) \\
&= Beta(1, 527)
\end{aligned}
$$

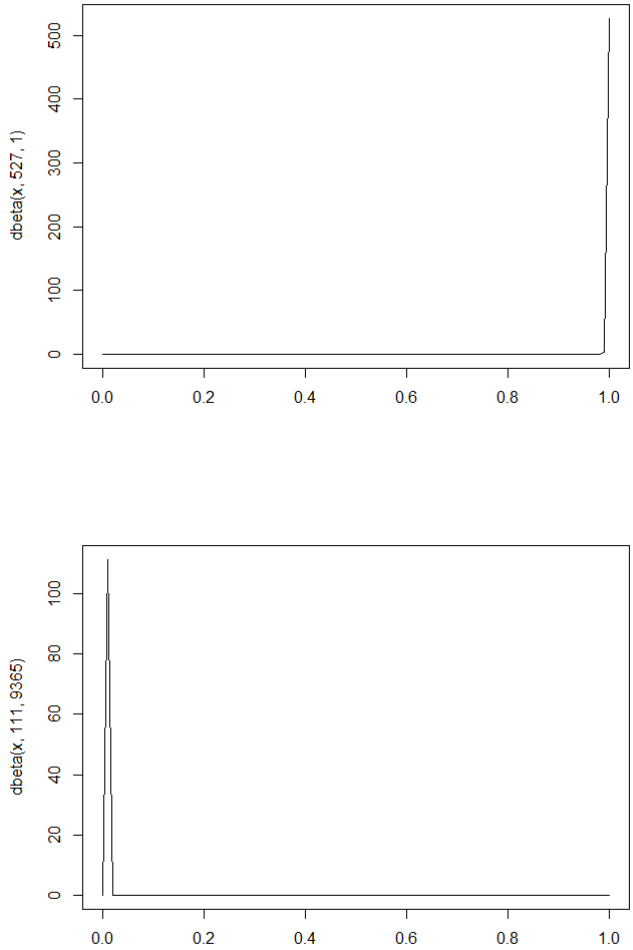


Figure A.5: Beta Distribution for P($E \mid L$)

$$
\begin{aligned}
P(E = no \mid L = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(9474, 9364) \\
&= Beta(9364 + 1, 9474 - 9364 + 1) \\
&= Beta(9365, 111)
\end{aligned}
$$

Table A.6: The contingency table of the observed counts of P($X \mid E$)

	E		
X	no	yes	missing
no	8911	13	0
yes	453	623	0
missing	0	0	0

A.1.6 CALCULATING P($X \mid E$)

$$\begin{aligned} P(X = yes \mid E = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(636, 623) \\ &= Beta(623 + 1, 636 - -623 + 1) \\ &= Beta(624, 14) \end{aligned}$$

$$\begin{aligned} P(X = yes \mid E = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(9364, 453) \\ &= Beta(453 + 1, 9364 - 453 + 1) \\ &= Beta(454, 8912) \end{aligned}$$

$$\begin{aligned} P(X = no \mid E = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(636, 13) \\ &= Beta(13 + 636 - 13 + 1) \\ &= Beta(14, 624) \end{aligned}$$

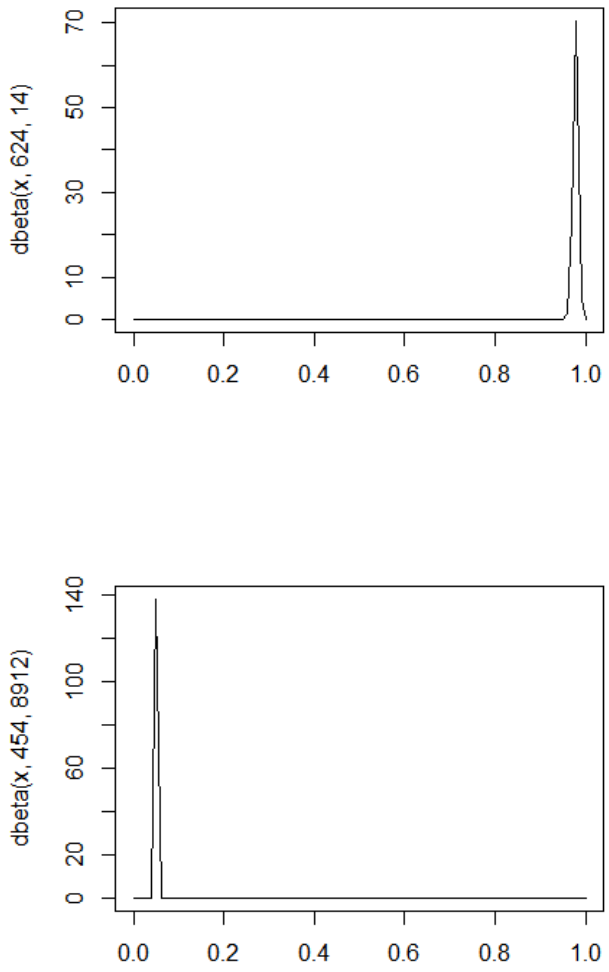


Figure A.6: Beta Distribution for P($X \mid E$)

$$
\begin{aligned}
P(X = no \mid E = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(9364, 8911) \\
&= Beta(8911 + 1, 9364 - 8911 + 1) \\
&= Beta(8912, 454)
\end{aligned}
$$

Table A.7: The contingency table of the observed counts of P($D \mid E$)

	E		
D	no	yes	missing
no	5522	117	0
yes	3842	519	0
missing	0	0	0

A.1.7 CALCULATING P($D \mid E$)

$$\begin{aligned} P(D = yes \mid E = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(636, 519) \\ &= Beta(519 + 1, 636 - 519 + 1) \\ &= Beta(520, 118) \end{aligned}$$

$$\begin{aligned} P(D = yes \mid E = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(9364, 3842) \\ &= Beta(3842 + 1, 9364 - 3842 + 1) \\ &= Beta(3843, 5523) \end{aligned}$$

$$\begin{aligned} P(D = no \mid E = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\ &= Beta(y + \alpha, n - y + \beta) \\ &= Beta(1, 1) \times Binomial(636, 117) \\ &= Beta(117 + 1, 636 - 117 + 1) \\ &= Beta(118, 520) \end{aligned}$$

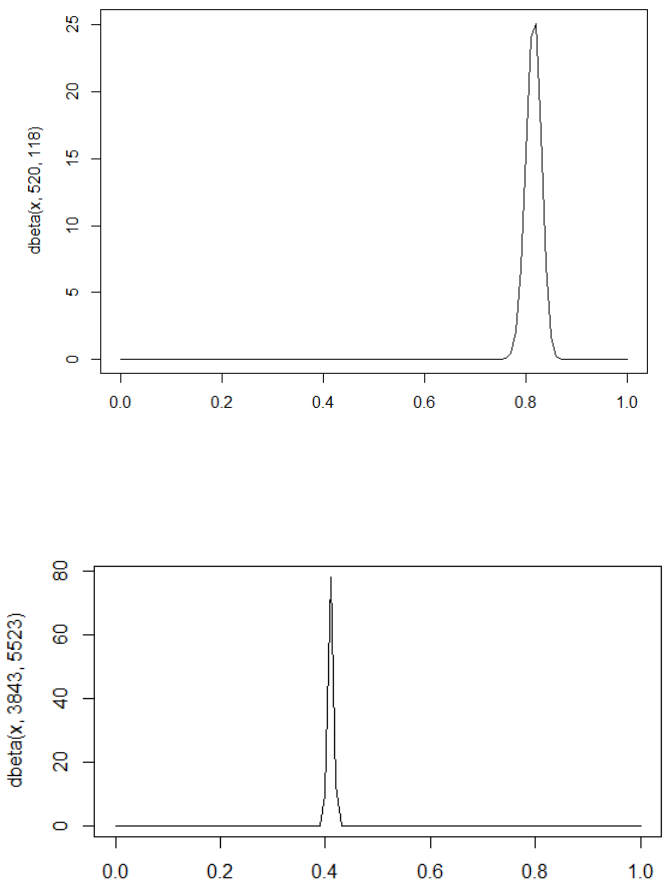


Figure A.7: Beta Distribution for P($D \mid E$)

$$
\begin{aligned}
P(D = no \mid E = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(9364, 5522) \\
&= Beta(5522 + 1, 9364 - 5522 + 1) \\
&= Beta(5523, 3843)
\end{aligned}
$$

Table A.8: The contingency table of the observed counts of P($D \mid B$)

	B		
D	no	yes	missing
no	4745	894	0
yes	724	3637	0
missing	0	0	0

A.1.8 CALCULATING P($D \mid B$)

$$
\begin{aligned}
P(D = yes \mid B = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(4531, 3637) \\
&= Beta(3637 + 1, 4531 - 3637 + 1) \\
&= Beta(3638, 895)
\end{aligned}
$$

$$
\begin{aligned}
P(D = yes \mid B = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(5469, 724) \\
&= Beta(724 + 1, 5469 - 724 + 1) \\
&= Beta(725, 4746)
\end{aligned}
$$

$$
\begin{aligned}
P(D = no \mid B = yes) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(4531, 894) \\
&= Beta(894 + 1, 4531 - 894 + 1) \\
&= Beta(895, 3638)
\end{aligned}
$$

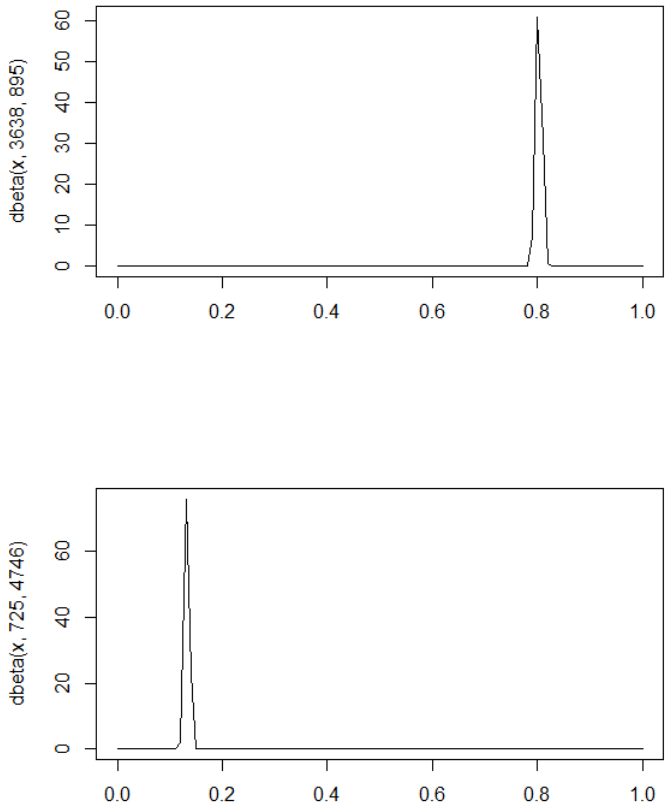


Figure A.8: Beta Distribution for P($D \mid B$)

$$
\begin{aligned}
P(D = no \mid B = no) &= Beta(\alpha, \beta) \times Binomial(n, \theta) \\
&= Beta(y + \alpha, n - y + \beta) \\
&= Beta(1, 1) \times Binomial(5469, 4745) \\
&= Beta(4745 + 1, 5469 - 4745 + 1) \\
&= Beta(4746, 725)
\end{aligned}
$$

APPENDIX B

COMPUTATIONS OF LINK STRENGTH MEASURE (L_S)

B.1 USING L_S ON THE CHEST CLINIC NETWORK

Given the Posterior distributions table for each link of the Chest Clinic Network as follows (Calculated in Appendix A):

We apply our link strengths measure (L_S) (Equation 4.1, which is presented in Chapter 4)

$$L\_S(Variable_1 \rightarrow Variable_2) = \min_{y \in Y}(pdf(\frac{y+\alpha}{\alpha+n+\beta}))$$

to find the strengths of each link as follows:

B.1.1 FINING THE STRENGTHS OF THE EDGE $S \rightarrow L$

$$\begin{aligned} L\_S(S \rightarrow L) = \min(&pdf(Beta(481, 4510)), pdf(Beta(47, 4966)), \\ &pdf(Beta(4510, 481)), pdf(Beta(4966, 47))) \\ &= 50.30727 \end{aligned}$$

Table B.1: Posterior distributions for the Chest Clinic Network.

Link	**Posterior Distributions (Beta Distributions)**
P(T | A)	Beta(10,99) Beta(106,9789) Beta(99,10) Beta(9789,106)
P(L | S)	Beta(481,4510) Beta(47,4966) Beta(4510,481) Beta(4966,47)
P(B | S)	Beta(3019,1972) Beta(1514,3899) Beta(1972,3019) Beta(3899,1514)
P(E | T)	Beta(115,1) Beta(523,9365) Beta(1,115) Beta(9365,523)
P(E | L)	Beta(527,1) Beta(111,9365) Beta(1,527) Beta(9365,111)
P(D | B)	Beta(3638,895) Beta(725,4746) Beta(895,3638) Beta(4746,725)
P(D | E)	Beta(520,118) Beta(3843,5523) Beta(118,520) Beta(5523,3843)
P(X | E)	Beta(624,14) Beta(454,8912) Beta(14,624) Beta(8912,454)

B.1.2 Fining the Strengths of the Edge $S \to B$

$$\begin{aligned} L\_S(S \to B) = \min(&pdf(Beta(3019, 1972)), pdf(Beta(1514, 3899)), \\ &pdf(Beta(1972, 3019)), pdf(Beta(3899, 1514))) \\ &= 56.88552 \end{aligned}$$

B.1.3 Fining the Strengths of the Edge $B \to D$

$$\begin{aligned} L\_S(B \to D) = \min(&pdf(Beta(3638, 895)), pdf(Beta(725, 4746)), \\ &pdf(Beta(895, 3638)), pdf(Beta(4746, 725))) \\ &= 49.30178 \end{aligned}$$

B.1.4 Fining the Strengths of the Edge $L \to E$

$$\begin{aligned} L\_S(L \to E) = \min(&pdf(Beta(527, 1)), pdf(Beta(111, 9365)), \\ &pdf(Beta(1, 527)), pdf(Beta(9365, 111))) \\ &= 129.2983 \end{aligned}$$

B.1.5 Fining the Strengths of the Edge $T \to E$

$$\begin{aligned} L\_S(T \to E) = \min(&pdf(Beta(115, 1)), pdf(Beta(523, 9365)), \\ &pdf(Beta(1, 115)), pdf(Beta(9365, 523))) \\ &= 103.7509 \end{aligned}$$

B.1.6 Fining the Strengths of the Edge $A \to T$

$$\begin{aligned} L\_S(A \to T) = \min(&pdf(Beta(10, 99)), pdf(Beta(106, 9789)), \\ &pdf(Beta(99, 10)), pdf(Beta(9789, 106))) \\ &= 14.75256 \end{aligned}$$

B.1.7 Fining the Strengths of the Edge $E \to X$

$$\begin{aligned} L\_S(E \to X) = \min(&pdf(Beta(624, 14)), pdf(Beta(454, 8912)), \\ &pdf(Beta(14, 624)), pdf(Beta(8912, 454))) \end{aligned}$$

$$= 70.69412$$

B.1.8 Fining the Strengths of the Edge $E \rightarrow D$

$$L\_S(E \rightarrow D) = \min(pdf(Beta(520, 118)), pdf(Beta(3843, 5523)),$$
$$pdf(Beta(118, 520)), pdf(Beta(5523, 3843)))$$
$$= 25.69412$$

APPENDIX C

COMPUTATION OF MUTUAL INFORMATION LINK STRENGTH

C.1 EXPERIMENTAL RESULTS

We will use the mutual information link strength measure (MI) to compute the strengths of links of the original Chest Clinic Network. For each link $X \rightarrow Y$ in the Chest Clinic Network, given the probability for variable X and the conditional probability for variable Y (Chest Clinic Network conditional probabilities are shown in [34]), we will calculate the joint probability for variables X and Y ($P(X, Y)$) in order to be able to compute the strength of every link $X \rightarrow Y$ using the mutual information equation (Equation 4.2).

C.1.1 THE EDGE $A \rightarrow T$

In the Chest Clinic Network, let the probability for variable A be $P(A = yes) = 0.01$, and the conditional probability for variable T be as follows:

We will compute the the joint probability of variables A and T and the marginal probability of each random variable. The joint probability can be computed as follows:

Table C.1: The conditional probability for variable T

	A	
T	yes	no
yes	0.05	0.01
no	0.95	0.99

Table C.2: The joint probability for variables T and A

	A		
T	yes	no	Marginal Probability
yes	0.0005	0.0099	0.0104
no	0.0095	0.9801	0.9896
Marginal Probability	0.01	0.99	1

$$
\begin{aligned}
P(T = yes, A = yes) &= P(T = yes \mid A = yes)P(A = yes) \\
&= 0.01 * 0.05 \\
&= 0.0005 \\
P(T = yes, A = no) &= P(T = yes \mid A = no)P(A = no) \\
&= 0.99 * 0.01 \\
&= 0.0099
\end{aligned}
$$

$$
\begin{aligned}
P(T = no, A = yes) &= P(T = no \mid A = yes)P(A = yes) \\
&= 0.01 * 0.95 \\
&= 0.0095 \\
P(T = no, A = no) &= P(T = no \mid A = no)P(A = no) \\
&= 0.99 * 0.99 \\
&= 0.9801
\end{aligned}
$$

At this point, we are able to use the mutual information formula to compute the strength of the link $A \rightarrow T$ as follows:

$$
\begin{aligned}
MI(T, A) &= \sum_{t \in T, a \in A} P(t, a) log_2(\frac{P(t, a)}{P(t)P(a)}) \\
&= 0.0005 * log_2(\frac{0.0005}{0.01 * 0.0104}) + 0.0099 * log_2(\frac{0.0099}{0.99 * 0.0104})
\end{aligned}
$$

Table C.3: The conditional probability for variable B

	S	
B	yes	no
yes	0.6	0.3
no	0.4	0.7

$$
\begin{aligned}
&+ 0.0095 * log_2(\frac{0.0095}{0.01 * 0.9896}) + 0.9801 * log_2(\frac{0.9801}{0.99 * 0.9896}) \\
&= 0.0005 * log_2(4.8077) + 0.0099 * log_2(0.9615) \\
&+ 0.0095 * log_2(0.96) + 0.9801 * log_2(1.0004) \\
&= 0.0006
\end{aligned}
$$

C.1.2 THE EDGE $S \rightarrow B$

In the Chest Clinic Network, let the probability for variable S be $P(S = yes) = 0.5$, and the conditional probability for variable B be as follows:

We will compute the the joint probability of variables S and B and the marginal probability of each random variable. The joint probability can be computed as follows:

$$
\begin{aligned}
P(B = yes, S = yes) &= P(B = yes \mid S = yes)P(S = yes) \\
&= 0.5 * 0.6 \\
&= 0.3
\end{aligned}
$$

$$
\begin{aligned}
P(B = yes, S = no) &= P(B = yes \mid S = no)P(S = no) \\
&= 0.5 * 0.3 \\
&= 0.15
\end{aligned}
$$

$$
P(B = no, S = yes) = P(B = no \mid S = yes)P(S = yes)
$$

$$= 0.5 * 0.4$$

$$= 0.2$$

$$P(B = no, S = no) = P(B = no \mid S = no)P(S = no)$$

$$= 0.5 * 0.7$$

$$= 0.35$$

Table C.4: The joint probability for variable B and S

	S		
B	yes	no	Marginal Probability
yes	0.3	0.15	0.45
no	0.2	0.35	0.55
Marginal Probability	0.5	0.5	1

At this point, we are able to use the mutual information formula to compute the strength of the link $S \rightarrow B$ as follows:

$$\begin{aligned} MI(B,S) &= \sum_{b \in B, s \in S} P(b,s) log_2(\frac{P(b,s)}{P(b)P(s)}) \\ &= 0.3 * log_2(\frac{0.3}{0.5 * 0.45}) + 0.2 * log_2(\frac{0.2}{0.5 * 0.55}) \\ &+ 0.15 * log_2(\frac{0.15}{0.5 * 0.45}) + 0.35 * log_2(\frac{0.35}{0.5 * 0.55}) \\ &= 0.3 * log_2(1.3333) + 0.2 * log_2(0.7273) \\ &+ 0.15 * log_2(0.6667) + 0.35 * log_2(1.2727) \\ &= 0.06665 \end{aligned}$$

C.1.3 THE EDGE $S \rightarrow L$

In the Chest Clinic Network, let the probability for variable S be $P(S = yes) = 0.5$, and the conditional probability for variable L be as follows:

Table C.5: The conditional probability for variable L

	S	
L	yes	no
yes	0.1	0.01
no	0.9	0.99

We will compute the the joint probability of variables S and L and the marginal probability of each random variable. The joint probability can be computed as follows:

$$
\begin{aligned}
P(L = yes, S = yes) &= P(L = yes \mid S = yes)P(S = yes) \\
&= 0.5 * 0.1 \\
&= 0.05 \\
P(L = yes, S = no) &= P(L = yes \mid S = no)P(S = no) \\
&= 0.5 * 0.01 \\
&= 0.005 \\
P(L = no, S = yes) &= P(L = no \mid S = yes)P(S = yes) \\
&= 0.5 * 0.9 \\
&= 0.45 \\
P(L = no, S = no) &= P(L = no \mid S = no)P(S = no) \\
&= 0.5 * 0.99 \\
&= 0.495
\end{aligned}
$$

Table C.6: The joint probability for variables L and S

	S		
L	yes	no	Marginal Probability
yes	0.05	0.005	0.055
no	0.45	0.495	0.945
Marginal Probability	0.5	0.5	1

At this point, we are able to use the mutual information formula to compute the strength of the link $S \rightarrow L$ as follows:

$$
\begin{aligned}
MI(L,S) &= \sum_{l \in L, s \in S} P(l,s) log_2(\frac{P(l,s)}{P(l)P(s)}) \\
&= 0.05 * log_2(\frac{0.05}{0.055 * 0.5}) + 0.005 * log_2(\frac{0.005}{0.5 * 0.055}) \\
&+ 0.45 * log_2(\frac{0.45}{0.945 * 0.5}) + 0.495 * log_2(\frac{0.495}{0.945 * 0.5}) \\
&= 0.05 * log_2(1.8182) + 0.005 * log_2(0.1818) \\
&+ 0.45 * log_2(0.9524) + 0.495 * log_2(1.0476) \\
&= 0.0303485
\end{aligned}
$$

C.1.4 THE EDGE $T \rightarrow E$

In the Chest Clinic Network, let the probability for variable L be $P(L = yes) = 0.055$, variable T be $P(T = yes) = 0.0104$ and the conditional probability for variable E be as follows:

Table C.7: The conditional probability for variable E

	T = Yes		T = No	
	L = Yes	L = No	L = Yes	L = No
E = Yes	1	1	1	0
E = No	0	0	0	1

We will compute the the joint probability of variables E, L and T and the marginal probability of each random variable. The joint probability can be computed as follows:

$$P(E,T,L) = P(E \mid T,L)P(T,L)$$

$$P(E,T,L) = P(E \mid T,L)P(T)P(L)$$

$$P(E = yes, T = yes, L = yes) = P(E = yes \mid T = yes, L = yes)P(T = yes)$$

$$P(L = yes)$$

$$= 1 * 0.0104 * 0.055$$

$$= 0.000572$$

$$P(E = yes, T = yes, L = no) = P(E = yes \mid T = yes, L = no)P(T = yes)$$

$$P(L = no)$$

$$= 1 * 0.0104 * 0.945$$

$$= 0.009828$$

$$P(E = yes, T = no, L = yes) = P(E = yes \mid T = no, L = yes)P(T = no)$$

$$P(L = yes)$$

$$= 1 * 0.9896 * 0.055$$

$$= 0.05428$$

$$P(E = yes, T = no, L = no) = P(E = yes \mid T = no, L = no)P(T = no)$$

$$P(L = no)$$

$$= 0 * 0.9896 * 0.945$$

$$= 0$$

$$P(E = no, T = yes, L = yes) = P(E = no \mid T = yes, L = yes)P(T = yes)$$

$$P(L = yes)$$

$$= 0 * 0.0104 * 0.055$$

$$= 0$$

$$P(E = no, T = yes, L = no) = P(E = no \mid T = yes, L = no)P(T = yes)$$

$$P(L = no)$$

$$= 0 * 0.0104 * 0.945$$

$$= 0$$

$$P(E = no, T = no, L = yes) = P(E = no \mid T = no, L = yes)P(T = no)$$

$$P(L = yes)$$

$$= 0 * 0.9896 * 0.055$$

$$= 0$$

$$P(E = no, T = no, L = no) = P(E = no \mid T = no, L = no)P(T = no)$$

$$P(L = no)$$

$$= 1 * 0.9896 * 0.945$$

$$= 0.935172$$

Table C.8: The joint probability for variables E, T and L

	T = Yes		T = No	
	L = Yes	L = No	L = Yes	L = No
E = Yes	0.000572	0.009828	0.054428	0
E = No	0	0	0	0.935172

In order to obtain the joint probability table for variables E and T, we marginalize the joint probability table for variables E, T, and L (Table C.8) as follows:

Table C.9: The joint probability for variables E and T

	T		
E	yes	no	Marginal Probability
yes	0.0104	0.054428	0.064828
no	0	0.935172	0.935172
Marginal Probability	0.0104	0.9896	1

At this point, we are able to use the mutual information formula to compute the strength of the link $T \rightarrow E$ as follows:

$$
\begin{aligned}
MI(E,T) &= \sum_{e\in E, t\in T} P(e,t) log_2(\frac{P(e,t)}{P(e)P(t)}) \\
&= 0.0104 * log_2(\frac{0.0104}{0.064828 * 0.0104}) + 0.054428 * log_2(\frac{0.054428}{0.064828 * 0.9896}) \\
&+ 0 * log_2(\frac{0}{0.935172 * 0.0104}) + 0.935172 * log_2(\frac{0.935172}{0.935172 * 0.9896}) \\
&= 0.0104 * log_2(15.4254) + 0.054428 * log_2(0.8484) \\
&+ 0 + 0.935172 * log_2(1.0105) \\
&= 0.0296
\end{aligned}
$$

C.1.5 THE EDGE $L \rightarrow E$

The joint probability table for variables E and L can be obtained through marginalization of the joint probability table for variables E, T, and L (Table C.8) as follows:

Table C.10: The joint probability for variables E and L

	L		
E	yes	no	Marginal Probability
yes	0.055	0.009828	0.064828
no	0	0.935172	0.935172
Marginal Probability	0.055	0.945	1

At this point, we are able to use the mutual information formula to compute the strength of the link $L \rightarrow E$ as follows:

$$
\begin{aligned}
MI(E,L) &= \sum_{e\in E, l\in L} P(e,l) log_2(\frac{P(e,l)}{P(e)P(l)}) \\
&= 0.055 * log_2(\frac{0.055}{0.064828 * 0.055}) + 0.009828 * log_2(\frac{0.009828}{0.064828 * 0.945}) \\
&+ 0 * log_2(\frac{0}{0.935172 * 0.055}) + 0.935172 * log_2(\frac{0.935172}{0.935172 * 0.945}) \\
&= 0.055 * log_2(15.4254) + 0.009828 * log_2(0.1604) \\
&+ 0 + 0.935172 * log_2(1.0582) \\
&= 0.2675
\end{aligned}
$$

C.1.6 THE EDGE $B \rightarrow D$

In the Chest Clinic Network, let the probability for variable B be $P(B = yes) = 0.45$, variable E be $P(E = yes) = 0.0648$ and the conditional probability for variable D be as follows:

Table C.11: The conditional probability for variable D

	E = Yes		E = No	
	B = Yes	B = No	B = Yes	B = No
D = Yes	0.9	0.7	0.8	0.1
D = No	0.1	0.3	0.2	0.9

We will compute the the joint probability of variables D, E and B and the marginal probability of each random variable. The joint probability can be computed as follows:

$$P(D, E, B) = P(D \mid E, B)P(E, B)$$

$$P(D, E, B) = P(D \mid E, B)P(E)P(B)$$

$$P(D = yes, T = yes, L = yes) = P(D = yes \mid T = yes, L = yes)P(T = yes)$$

$$P(L = yes)$$

$$= 0.9 * 0.064828 * 0.45$$

$$= 0.0263$$

$$P(D = yes, E = yes, B = no) = P(D = yes \mid E = yes, B = no)P(E = yes)$$

$$P(B = no)$$

$$= 0.7 * 0.064828 * 0.55$$

$$= 0.0249$$

$$P(D = yes, E = no, B = yes) = P(D = yes \mid E = no, B = yes)P(E = no)$$

$$P(B = yes)$$

$$= 0.8 * 0.935172 * 0.45$$

$$= 0.3367$$

$$P(D = yes, E = no, B = no) = P(D = yes \mid E = no, B = no)P(E = no)$$
$$P(B = no)$$
$$= 0.1 * 0.935172 * 0.55$$
$$= 0.0514$$
$$P(D = no, E = yes, B = yes) = P(D = no \mid E = yes, B = yes)P(E = yes)$$
$$P(B = yes)$$
$$= 0.1 * 0.064828 * 0.45$$
$$= 0.0029$$
$$P(D = no, E = yes, B = no) = P(D = no \mid E = yes, B = no)P(E = yes)$$
$$P(B = no)$$
$$= 0.3 * 0.064828 * 0.55$$
$$= 0.0107$$
$$P(E = no, T = no, L = yes) = P(E = no \mid T = no, L = yes)P(T = no)$$
$$P(L = yes)$$
$$= 0.2 * 0.935172 * 0.45$$
$$= 0.0842$$
$$P(E = no, T = no, L = no) = P(E = no \mid T = no, L = no)P(T = no)$$
$$P(L = no)$$
$$= 0.9 * 0.935172 * 0.55$$
$$= 0.4629$$

Table C.12: The joint probability for variables D, E and B

	E = Yes		E = No	
	B = Yes	B = No	B = Yes	B = No
D = Yes	0.0263	0.0249	0.3367	0.0514
D = No	0.0029	0.0107	0.0842	0.4629

In order to obtain the joint probability table for variables D and B, we marginalize the joint probability table for variables D, E, and B (Table C.12) as follows:

Table C.13: The joint probability for variables D and B

	B		
D	yes	no	Marginal Probability
yes	0.363	0.0763	0.4393
no	0.0871	0.4736	0.5607
Marginal Probability	0.4501	0.5499	1

At this point, we are able to use the mutual information formula to compute the strength of the link $B \rightarrow D$ as follows:

$$
\begin{aligned}
MI(D,B) &= \sum_{d \in D, b \in B} P(d,b) log_2(\frac{P(d,b)}{P(d)P(b)}) \\
&= 0.363 * log_2(\frac{0.363}{0.4393 * 0.4501}) + 0.0763 * log_2(\frac{0.0763}{0.4393 * 0.5499}) \\
&+ 0.0871 * log_2(\frac{0.0871}{0.5607 * 0.4501}) + 0.4736 * log_2(\frac{0.4736}{0.5607 * 0.5499}) \\
&= 0.363 * log_2(1.8358) + 0.0763 * log_2(0.3158) \\
&+ 0.0871 * log_2(0.3451) + 0.4736 * log_2(1.536) \\
&= 0.3508
\end{aligned}
$$

C.1.7 THE EDGE $E \rightarrow D$

The joint probability table for variables D and E can be obtained through marginalization of the joint probability table for variables D, E, and B (Table C.12) as follows:

Table C.14: The joint probability for variables D and E

	E		
D	yes	no	Marginal Probability
yes	0.0512	0.3881	0.4393
no	0.0136	0.5471	0.5607
Marginal Probability	0.0648	0.9352	1

At this point, we are able to use the mutual information formula to compute the strength of the link $E \rightarrow D$ as follows:

$$
\begin{aligned}
MI(D,E) &= \sum_{d \in D, e \in E} P(d,e) log_2(\frac{P(d,e)}{P(d)P(e)}) \\
&= 0.0512 * log_2(\frac{0.0512}{0.0.0648 * 0.4393}) + 0.3881 * log_2(\frac{0.3881}{0.9352 * 0.4393}) \\
&+ 0.0136 * log_2(\frac{0.0136}{0.0648 * 0.5607}) + 0.5471 * log_2(\frac{0.5471}{0.9352 * 0.5607}) \\
&= 0.0512 * log_2(1.7986) + 0.3881 * log_2(0.9447) \\
&+ 0.0136 * log_2(0.3743) + 0.5471 * log_2(1.0434) \\
&= 0.02575
\end{aligned}
$$

C.1.8 THE EDGE $E \rightarrow X$

In the Chest Clinic Network, let the probability for variable E be $P(E = yes) = 0.0.0648$, and the conditional probability for variable X be as follows:

Table C.15: The conditional probability for variable X

	E	
X	yes	no
yes	0.98	0.05
no	0.02	0.95

We will compute the the joint probability of variables E and X and the marginal probability of each random variable. The joint probability can be computed as follows:

$$
\begin{aligned}
P(X = yes, E = yes) &= P(X = yes \mid E = yes)P(E = yes) \\
&= 0.98 * 0.0.064828 \\
&= 0.0635 \\
P(X = yes, E = no) &= P(X = yes \mid E = no)P(E = no) \\
&= 0.05 * 0.935172
\end{aligned}
$$

$$= 0.0468$$

$$P(X = no, E = yes) = P(X = no \mid E = yes)P(E = yes)$$

$$= 0.02 * 0.064828$$

$$= 0.0013$$

$$P(X = no, E = no) = P(X = no \mid E = no)P(E = no)$$

$$= 0.95 * 0.935172$$

$$= 0.8884$$

Table C.16: The joint probability for variables X and E

	E		
X	yes	no	Marginal Probability
yes	0.0635	0.0468	0.1103
no	0.0013	0.8884	0.8897
Marginal Probability	0.0648	0.9352	1

At this point, we are able to use the mutual information formula to compute the strength of the link $E \rightarrow X$ as follows:

$$MI(X, E) = \sum_{x \in X, e \in E} P(x, e) log_2(\frac{P(x, e)}{P(x)P(e)})$$

$$= 0.0635 * log_2(\frac{0.0635}{0.1103 * 0.0.0648}) + 0.0468 * log_2(\frac{0.0468}{0.1103 * 0.9352})$$

$$+ 0.0013 * log_2(\frac{0.0013}{0.8897 * 0.0.0648}) + 0.8884 * log_2(\frac{0.8884}{0.8897 * 0.9352})$$

$$= 0.0635 * log_2(3.1513) + 0.0468 * log_2(-1.1402)$$

$$+ 0.0013 * log_2(-5.4739) + 0.8884 * log_2(0.0945)$$

$$= 0.2236$$

APPENDIX D

CORRUPTED CASES USED TO ADD LINKS TO CHEST CLINIC NETWORK

D.1 CORRUPTED CASES USED IN OUR EXPERIMENTS

In this appendix, we present the cases that were added to the dataset DB_1 with the goal of introducing malicious links to Chest Clinic network. The added cases are as follows:

D.1.1 CASES TO INTRODUCE THE LINK $D - S$

Table D.1: 74 cases to be added to DB_1 to introduce the link $D - S$

X	B	D	A	S	L	T	E
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No

Table D.1 continued from previous page

X	B	D	A	S	L	T	E
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No

Table D.1 continued from previous page

X	B	D	A	S	L	T	E
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No

Table D.1 continued from previous page

X	B	D	A	S	L	T	E
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No
No	No	Yes	No	Yes	No	No	No

D.1.2 CASES TO INTRODUCE THE LINK $B - L$

Table D.2: 13 cases to be added to DB_1 to introduce the link $B - L$

X	B	D	A	S	L	T	E
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No

Table D.2 continued from previous page

X	B	D	A	S	L	T	E
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No

D.1.3 CASES TO INTRODUCE THE LINK $A - E$

Table D.3: 3 cases to be added to DB_1 to introduce the link $A - E$

X	B	D	A	S	L	T	E
No	No	No	Yes	No	No	No	Yes
No	No	No	Yes	No	No	No	Yes
No	No	No	Yes	No	No	No	Yes

D.1.4 CASES TO INTRODUCE THE LINK $T - L$ TO BREAK THE UNSHIELDED COLLIDER E

Table D.4: 8 cases to be added to DB_1 to break the unshielded collider E

X	B	D	A	S	L	T	E
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No

Table D.4 continued from previous page

X	B	D	A	S	L	T	E
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No

D.1.5 CASES TO INTRODUCE THE LINK $T - L$ TO CHANGE THE DIRECTIONS OF THE TRIPLE $T - E - L$

Table D.5: 17 cases to be added to DB_1 to change the directions of the triple $T - E - L$

X	B	D	A	S	L	T	E
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No

Table D.5 continued from previous page

X	B	D	A	S	L	T	E
No	No	No	No	No	Yes	Yes	No
No	No	No	No	No	Yes	Yes	No

D.1.6 CASES TO INTRODUCE THE LINK $B - L$

Table D.6: 13 cases to be added to DB_1 to introduce the link $B - L$

X	B	D	A	S	L	T	E
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No
No	Yes	No	No	No	Yes	No	No