How to Generate the Network you Want with the PC Learning Algorithm

Emad Alsuwat
Computer Science and Engineering
University of South Carolina
Alsuwat@email.sc.edu

Marco Valtorta
Computer Science and Engineering
University of South Carolina
Mgv@cec.sc.edu

Csilla Farkas
Computer Science and Engineering
University of South Carolina
Farkas@cec.sc.edu

Abstract

Data integrity is a key component of effective Bayesian network structure learning algorithms, namely PC algorithm, design and use. Given the role that integrity of data plays in these outcomes, this research demonstrates the importance of data integrity as a key component in machine learning tools in order to emphasize the need for carefully considering data integrity during tool development and utilization. To meet this purpose, we study how an adversary could generate a desired network with the PC algorithm. Given a Bayesian network B_1 and a database DB_1 generated by B_1 and a second Bayesian network, B_2, which is equal to B_1, except for a minor change like a missing link, a reversed link, or an additional link, we explore and analyze what is the minimal number of changes such as additions, deletions, substitutions to DB_1 that lead to a database DB_2 that, when given as input to PC algorithm, results in B_2.
Keywords: Adversarial Machine Learning, Bayesian Networks, Data Poisoning Attacks, The PC Algorithm.

1 Introduction and Motivation

There has been a massive increase in the use of machine learning for diverse computer applications. 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 [2]. It has been shown that an adversary can corrupt machine learning models by manipulating the input dataset [3, 24].

Therefore, it is essential to not only consider the presence of adversarial opponents but also design effective and resilient learning algorithms.

Adversarial machine learning is the research field that studies the design of efficient machine learning algorithms in adversarial environments [7]. In the presence of adversarial opponents (*the offense*), the problem of securing machine learning systems becomes harder since adversaries will attempt to corrupt machine learning models by crafting their inputs (known as *adversarial samples*) in an intelligent way. *Adversarial samples* are input datasets to machine learning algorithms that are crafted by adversaries to deliberately corrupt machine learning models [20]. Among different adversarial attacks, *data poisoning attacks*, which aim to corrupt the machine learning model by contaminating the data in the training phase, are considered one of the most important emerging security threats against machine learning systems [15].

Previous research has been conducted on data poisoning attacks against machine learning algorithms such as Support Vector Machines (SVMs) [3,5,9,14,17,25,26], Neural Networks (NNs) [27], and other machine learning algorithms [15]. Surprisingly, to date, no study has been performed on evaluating the vulnerabilities of Bayesian network learning algorithms against adversarial attacks.

In investigating data poisoning attacks on Bayesian network algorithms, we study attacks that aim to invalidate the Bayesian model either by modifying the input sample to delete the weakest edge or by adding cases to the original input sample to add the most believable yet incorrect edge.

In this paper, our main focus is to evaluate the robustness of Bayesian network learning algorithms on multiple adversarial samples against the proposed attacks. Our experiments show that the PC algorithm is vulnerable to data poisoning attacks. The learned Bayesian network model is vulnerable if an adversary can only inject a small number of adversarial samples into the original dataset.

We present our findings pertaining to the robustness of the PC algorithm against data poisoning attacks. The main contributions are the following: 1) We define a novel measure of the strengths of links between variables in Bayesian networks and present our detailed analysis. 2) We demonstrate how to use the defined link strengths measure to add edges to and delete edges from a Bayesian network model. 3) We present two types of data poisoning attacks against the PC structure learning algorithm and implement the attacks. 4) We have implemented our approach and presented the results.

The rest of this paper is structured as follows. In section 2, we present the background information. Our link strengths measure in discrete Bayesian networks is presented in section 3. In section 4, we present two types of data poisoning attacks against the PC algorithm. In section 5, we present our empirical results. In section 6, we provide conclusions and directions for future work.

2 Background Information

2.1 Structure Learning in Bayesian Networks

There are three main approaches to learning the structure of Bayesian networks: *constraint-based*, *score-based*, or *hybrid* algorithms. We will focus on constraint-based algorithms, namely 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 [22]. According to [23], 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* [12, 19]), "which is a variant of the original PC algorithm due to [23]" [8].

2.2 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, $P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$, 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 ($P(B) > 0$) [16].

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:

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)} \tag{1}$$

In equation 1, $P(\theta \mid D)$ 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 [11]. 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:

$$\begin{gathered} P(\theta \mid D) \propto P(D \mid \theta) \times P(\theta) \\ Posterior \propto Likelihood \times Prior \end{gathered} \tag{2}$$

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. 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 [21]. 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.

$$\begin{gathered} P(\theta \mid D) \propto Beta(\alpha, \beta) \times Binomial(n, \theta) \\ P(\theta \mid D) \propto Beta(y + \alpha, n - y + \beta) \end{gathered} \tag{3}$$

Equation 3 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 [21]. Thus, $Beta(y+\alpha, n-y+\beta)$ is the posterior distribution.

2.3 Link Strengths in Bayesian Networks

The concept of link strength in Bayesian networks was introduced first by Boerlage in 1992 [4]. 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 $A - B$ [4]. Methods for link strengths measurements are not studied sufficiently. Imme Ebert-Uphoff in her 2009 paper [6] presented a tutorial on how to measure connection strengths and link strengths in discrete Bayesian networks. Ebert-Uphoff concluded that there is limited literature on link strengths, and there is more need to apply and use link strengths measures in structure learning and other purposes [6]. 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.

In this paper, we propose a link strengths measure (denoted by L_S) for discrete Bayesian networks. We then use L_S to determine the weakest edge and the most believable edge in a given causal model. We further study the robustness of the PC algorithm against data poisoning attacks that aim to remove the weakest edge and insert the most believable yet incorrect edge.

3 Measuring Link Strengths from Data in Discrete Bayesian Networks

In this paper, we introduce a novel link strengths measure between two random variables in a discrete Bayesian network model. 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 [4, 6]. 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. Our novel approach is as follows:

Definition 1. *Link Strengths Measure L_S is defined as*

$$L\_S(Variable_1 \to Variable_2) = \min_{y \in Y}(pdf(\frac{y+\alpha}{\alpha+n+\beta})) \tag{4}$$

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

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 $Variable_1 \to Variable_2$ in B_1, build a contingency table [13] for the two discrete variables $Variable_1$ and $Variable_2$ with i and j states, respectively (as shown in table 1). 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 $Variable_1 \to Variable_2$ 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. Note that $\frac{y+\alpha}{\alpha+n+\beta}$ in equation 4 is obtained by simply substituting α with $y+\alpha$ and β with $n-y+\beta$ in $\frac{\alpha}{\alpha+\beta}$.

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. In other words, 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.

	Variable$_2$			
Variable$_1$	**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)

Table 1: A contingency table for two discrete variables $Variable_1$ and $Variable_2$ with i and j states, respectively. The contingency table is structured as follows: $[n_{ij}]$ is the cell's observed counts obtained from dataset DB_1, (e_{ij}) is the cell's expected counts, calculated as follows: (*Observed Row Total* $\times$ *Observed Column Total*) $\div$ (*Observed Grand Total (denoted as n)*), and $< ts_{ij} >$ is the cell's chi-square test statistic, calculated as follows: $(n_{ij} - e_{ij})^2 \div e_{ij}$.

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 for new edges to be added. We claim that the highest L_S value, the most believable the new edge is. A practical usage of the proposed link strengths measure is that it can be used to evaluate the robustness of the PC algorithm against data poisoning attacks.

4 Data Poisoning Attacks against the PC Algorithm

In the process of using data to learn the structure of a Bayesian network model, the PC algorithm assesses conditional independence statements linking variables. The χ^2 statistical test is conducted on the given dataset to outline the statistical independence statement set for the learned causal model [18]. As a result of delineating how the PC algorithm works, adversarial attackers may exploit this knowing by contaminating the input dataset via weak edges removal or insertion of believable, yet incorrect links.

In this paper, we use our link strengths measure to investigate the robustness of the PC algorithm against two types of data poisoning attacks as follows: 1) Data poisoning attacks based on removing the weakest edge and 2) Data poisoning attacks based on inserting the most believable yet incorrect edge.

Due to space limitation, we only present selected algorithms in this work. A complete set of algorithms and further details can be accessed in our technical report [1].

4.1 Data Poisoning Attacks based on Removing the Weakest Edge

As discussed, it is feasible to use link strengths measure to identify and rank causal model edges from weakest to strongest, which means that adversarial opponents may seize the opportunity to poison the learning dataset, the objective being to effectively remove weak edges.

We have developed *Algorithm 4* to check the resilience of the PC algorithm against attacks that target weak edges. Our algorithm calculates the strength of each link in a Bayesian model and then rank the edges from the weakest to the strongest edge. It then checks the robustness of the PC algorithm against the feasibility of deleting the weakest edge. Our empirical results are presented in section 5.

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
1  Procedure Removing a Weak Edge($DB_1$)
2      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 [12])
3      Use $L\_S$ to rank the edges of $B_1$ from the weakest to the strongest
4      Let $A - C$ be the weakest edge to be deleted from $B_1$
5      Test the feasibility of deleting the edge $A - C$ from $B_1$ using Algorithm 3
6      if Algorithm 3 returns $DB_2$ then
7          Return $DB_2$
8      else
9          Return msg "Algorithm 3 failed to delete the link $A - C$ within a feasible
             number of cases"
10     end
11 end
```

4.2 Data Poisoning Attacks based on Adding the Most Believable yet Incorrect Edge

We demonstrate that the use of link strengths measure can be successfully applied to a causal model to accurately identify and rank the edges from most to least in terms of believability. Therefore, adversaries can skillfully use data poisoning attacks to generate input dataset to the Bayesian network model so that integrating the incorrect, yet plausible edges is viable.

```
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

1  Procedure Adding the Most Believable yet Incorrect
     Edge($DB_1$)
2      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 [12])
3      Choose a set of edge $Q$ that could be added to $B_1$
4      Use $L\_S$ to rank the set of edges $Q$ from the most to the least believable edge
5      Let $A - C$ be the most believable edge to be added to $B_1$
6      if $A - C$ lies in a a serial or diverging triple $A - B - C$ then
7          Use Algorithm 1 to check the feasibility of adding the link $A - C$
8          if Algorithm 1 returns $DB_2$ then
9              Return $DB_2$
10         else
11             Return msg "Algorithm 1 failed to introduce the link $A - C$"
12         end
13     else if $A - C$ lies in a converging triple $A \rightarrow B \leftarrow C$ then
14         Use Algorithm 2 to check the feasibility of adding the link $A - C$
15         if Algorithm 2 returns $DB_2$ then
16             Return $DB_2$
17         else
18             Return msg "Algorithm 2 failed to introduce the link $A - C$"
19         end
20     else
21         Use Algorithm 5 to check the feasibility of adding the link $A - C$
22         if Algorithm 5 returns $DB_2$ then
23             Return $DB_2$
24         else
25             Return msg "Algorithm 5 failed to introduce the link $A - C$"
26         end
27     end
28 end
```

We have developed *Algorithm 6* to check the robustness of the PC algorithm against this attack. The algorithm starts by learning the structure of the Bayesian network model and then uses the defined link strengths measure to rank a given set of edges that could be added to the learned model from the most to the least believable edge. Our algorithm then checks the robustness of the PC algorithm against the feasibility of adding the most believable edge. Our empirical results are presented in section 5.

5 Empirical Results

In this section, we present the results of using our link strengths measure on the Chest Clinic Network [10] and then demonstrate some of its practical usages. We implemented the Chest Clinic Network (shown in Figure 1) using *Hugin^{TM} Research 8.1*. Then we simulated dataset of $10,000$ cases for our experiments by using *Hugin^{TM} case generator* [12, 19]. We call this dataset as DB_1. Using the PC algorithm on dataset DB_1 with 0.05 significance setting [12], the resulting structure is given in Figure 2. While the two networks belong to different Markov equivalence classes, we will use the network of Figure 2 as the starting point of our experiments.

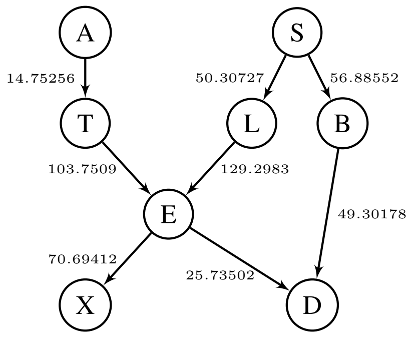

Figure 1: Chest Clinic Network and the result of link strengths (L_S)

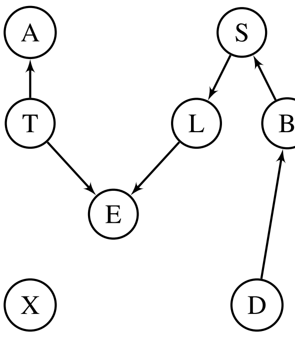

Figure 2: B_1, the result of feeding DB_1 to the PC algorithm with significance level at 0.05

We computed the link strengths using our approach (shown in Figure 1). We evaluated the effectiveness of data poisoning attacks against the PC algorithm (presented in section 4) to poison the Chest Clinic Network dataset DB_1 as follows: First, we validate the effectiveness of data poisoning attacks based on removing the weakest edge described in *Algorithm 4* to contaminate DB_1. Second, we check the resilience of the PC algorithm against the feasibility of data poisoning attacks based on adding most believable yet incorrect edge described in *Algorithm 6* to poison DB_1.

We present our results of deleting the weakest edge from B_1 in Figure 3. We observe that *Algorithm 4* succeeded to determine the weakest edge, $A - T$, and delete it by modifying only 3 cases in our dataset DB_1. Our results of adding the most believable edge to B_1 are presented in Figure 4. We observe that *Algorithm 6* succeeded to fool the PC algorithm and introduce the most believable edge, $B - L$, from the set of edges Q (in our experiment, we let $Q = \{A - S, T - S, D - S, L - B, L - T\}$) by inserting only 13 corrupt cases to our dataset DB_1.

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 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. We also observe that when introducing a new 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 cases than a cell with a lower test statistics value. Overall, we showed that the PC algorithm is vulnerable to data poisoning attacks based removing the weakest edge and adding the most believable yet incorrect edge.

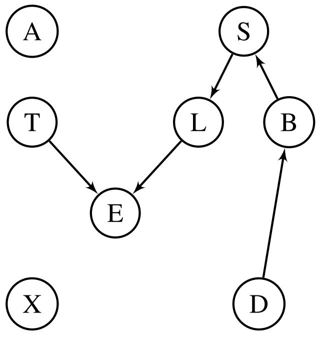

Figure 3: The result of removing the weakest link in B_1, $A \to T$

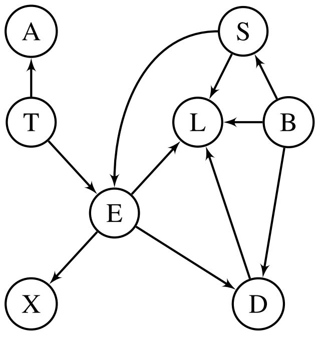

Figure 4: The result of adding the most believable link to B_1, $B \to L$.

6 Conclusion and Future Work

In this paper, we demonstrated the vulnerability of the PC structure learning algorithm. We have developed a theoretical framework to classify data poisoning attacks against the PC algorithm. We also performed experimental studies using the widely used Chest Clinic dataset. Our findings indicate that the PC algorithm is highly sensitive to data poisoning attacks. We also demonstrated that attackers could corrupt the learning outcome in a way that the PC algorithm 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. We believe that using this measure will guide defensive measurements. Our ongoing work includes the development of methods that will reduce the risk of unauthorized compromise against the PC algorithm via data poisoning.

References

[1] E. Alsuwat, M. Valtorta, and C. Farkas. Bayesian structure learning attacks. Technical report, University of South Carolina, SC, USA, 2018.

[2] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, Nov 2010.

[3] B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1467–1474. Omnipress, 2012.

[4] B. Boerlage. *Link strength in bayesian networks*. PhD thesis, University of British Columbia, 1992.

[5] C. Burkard and B. Lagesse. Analysis of causative attacks against svms learning from data streams. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, pages 31–36. ACM, 2017.

[6] I. Ebert-Uphoff. Tutorial on how to measure link strengths in discrete bayesian networks. Technical report, Georgia Institute of Technology, 2009.

[7] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58. ACM, 2011.

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

[9] P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894, 2017.

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

[11] S. M. Lynch. *Introduction to applied Bayesian statistics and estimation for social scientists*. Springer Science & Business Media, 2007.

[12] A. L. Madsen, F. Jensen, U. B. Kjaerulff, and M. Lang. The hugin tool for probabilistic graphical models. *International Journal on Artificial Intelligence Tools*, 14(03):507–543, 2005.

[13] M. L. McHugh. The chi-square test of independence. *Biochemia medica: Biochemia medica*, 23(2):143–149, 2013.

[14] S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, pages 2871–2877, 2015.

[15] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli. Towards poisoning of deep learning algorithms with back-gradient optimization. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 27–38. ACM, 2017.

[16] R. E. Neapolitan et al. *Learning bayesian networks*, volume 38. Pearson Prentice Hall Upper Saddle River, NJ, 2004.

[17] A. Newell, R. Potharaju, L. Xiang, and C. Nita-Rotaru. On the practicality of integrity attacks on document-level sentiment analysis. In *Proceedings of the 2014 Workshop on Artificial Intelligent and Security Workshop*, pages 83–93. ACM, 2014.

[18] T. D. Nielsen and F. V. Jensen. *Bayesian networks and decision graphs*. Springer Science & Business Media, 2009.

[19] K. G. Olesen, S. L. Lauritzen, and F. V. Jensen. ahugin: A system creating adaptive causal probabilistic networks. In *Uncertainty in Artificial Intelligence, 1992*, pages 223–229. Elsevier, 1992.

[20] N. Papernot, P. McDaniel, and I. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[21] H. Raiffa and R. Schlaifer. *Applied statistical decision theory*. Div. of Research, Graduate School of Business Administration, Harvard Univ., 1961.

[22] M. Scutari. Learning bayesian networks with the bnlearn r package. *Journal of Statistical Software*, 35(3):1–22, 2010.

[23] P. Spirtes, C. N. Glymour, and R. Scheines. *Causation, prediction, and search*. MIT press, 2000.

[24] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[25] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160:53–62, 2015.

[26] H. Xiao, H. Xiao, and C. Eckert. Adversarial label flips attack on support vector machines. In *ECAI*, pages 870–875, 2012.

[27] C. Yang, Q. Wu, H. Li, and Y. Chen. Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340*, 2017.