

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/267511164>

Learning Parameters in Canonical Models Using Weighted Least Squares

Conference Paper · September 2014

DOI: 10.1007/978-3-319-11433-0_24

CITATIONS

0

READS

33

2 authors:



[Krzysztof Nowak](#)

European Space Agency

9 PUBLICATIONS 7 CITATIONS

[SEE PROFILE](#)



[Marek Jozef Druzdel](#)

University of Pittsburgh

185 PUBLICATIONS 3,442 CITATIONS

[SEE PROFILE](#)

Learning Parameters in Canonical Models using Weighted Least Squares

Krzysztof Nowak^{1,3} & Marek J. Drużdżel^{1,2}

¹ Białystok University of Technology, Białystok, Poland

² School of Information Sciences, Pittsburgh, USA

³ European Space Agency, Noordwijk, The Netherlands.

`knowak.ai@gmail.com, marek@sis.pitt.edu`

Abstract. We propose a novel approach to learning parameters of canonical models from small data sets using a concept employed in regression analysis: weighted least squares method. We assess the performance of our method experimentally and show that it typically outperforms simple methods used in the literature in terms of accuracy of the learned conditional probability distributions.

Keywords: Bayesian networks, canonical models, noisy-MAX gates, parameter learning, weighted least squares

1 Introduction

Methodologies for extracting information from data have been one of the key factors for the success of modern artificial intelligence. Bayesian networks — one of the prime examples among probabilistic modeling techniques — are widely acclaimed and used by the scientific communities as well as the industry. Nowadays, data for many problem domains are freely accessible and in many cases growing at an exponential rate. Nonetheless, in some fields the amount of data is small, usually due to the cost of acquisition or high complexity of the problem. The latter increases the number of parameters required for the accurate modeling of the problem, which in turn calls for learning samples of large size. A class of interactions within Bayesian networks, the so called ICI (*Independence of Causal Influence*) models, find their applications in problems where obtaining an adequately sampled dataset is infeasible. In this paper we focus on learning parameters for the ICI models by framing the problem in terms of linear algebra and then calculating the values of parameters by means of the weighted least squares. We follow this up by an empirical test for learning accuracy of the proposed method and highlight the cases in which it outperforms the two common approaches to this problem: Expectation-maximization approach and the method proposed by Oniško and Drużdżel [1].

2 ICI Models

ICI models [2–5] are based on the assumption of independence of causal influences. An effect variable, along with its causes, may fit an ICI model if the mech-

anisms through which the causes impact the effect do not interact among each other. This simple restriction greatly simplifies elicitation of parameters from data and experts. Because some conditional probabilities can now be expressed as a function of a far smaller set of parameters, the number of independent parameters required to define the CPT (Conditional Probability Table) of the child node is reduced from exponential to linear in the number of parents. Figure 1 shows an example of such model.

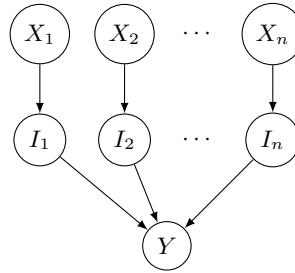


Fig. 1: Structure of an ICI model: additional auxiliary nodes $I_1 - I_n$ are called *inhibitors*. The independence of mechanisms is represented by a lack of edges among the inhibitor nodes I_1 to I_n .

2.1 Noisy-OR/MAX

Deterministic models usually rely on a function that takes a set of input signals which determine the state of the child node Y . The deterministic OR function makes it impossible for the child to be activated if none of the parent nodes is present. Noisy-OR [2, 3] and Noisy-MAX [6] models are cases of deterministic OR and deterministic MAX functions applied to the ICI framework [5]. Every variable has a special state indicating that the phenomenon that it represents is “absent,” we call such state the “distinguished state.” The term “activated state” will then refer to any of the non-distinguished states. Uncertainty is introduced to the model by adding a separate layer of inhibitor nodes (I_1 through I_n in Figure 1), which are activated based on their corresponding states of parents (nodes X_1 through X_n) with a certain probability. Once that is done, the states of inhibitor nodes are used as input signals for the deterministic function. As it turns out, all we need to provide to the model are the probabilities of every parent variable activating the child independently, that is when every variable other than the one in question is in its distinguished state. Thus, a Noisy-OR

parameter p_i describes the probability of i -th parent activating the child:⁴

$$p_i = P(+y | \neg x_1, \neg x_2, \dots, +x_i, \dots, \neg x_n) . \quad (1)$$

Obtaining conditional probabilities for the combination of states of nodes X_i other than the ones already described by the Noisy-MAX parameters can be derived using the following equation:

$$P(+y | \mathbf{x}) = 1 - \prod_{i \in I(\mathbf{x})} (1 - p_i) , \quad (2)$$

where $I(\mathbf{x})$ is a set of indices of parents in combination \mathbf{x} which are in their activated states. In our case, OR is the deterministic function — the child is active when any of the inhibitors is active. Deterministic MAX function is a generalization of the deterministic OR as proposed independently by Díez [6] and Srinivas [7]. For that reason, we can treat Noisy-OR as a binary case of the Noisy-MAX model. In further sections, we will occasionally use Noisy-OR/MAX interchangeably, knowing that general ideas apply equally well to binary and non-binary variables.

2.2 Leaky Noisy-OR/MAX

Deterministic OR function activates the child node only when at least one of the parents is in its non-distinguished state. This is not always the case in the real world — absence of any signal from the parental causes may still activate the child variable. Sometimes a problem does not allow for explicit modeling of each possible cause, either because it is not well understood, or because it would require a large number of additional variables that would have minimal impact on the child. In that case the unmodelled causes can be aggregated into a single node called *leak* [2] (see Figure 2). Since leak is modeling mechanisms that we

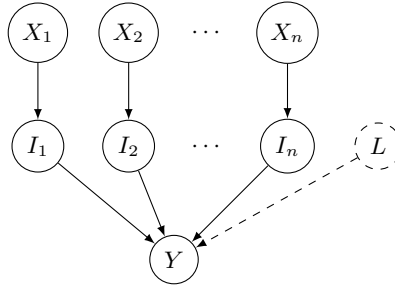


Fig. 2: Structure of a Leaky ICI model: L is the implicit leak node.

⁴ We will use lowercase to denote the states of variables. Distinguished states are preceded with a negation sign, while activated states with a plus sign, e.g., $+x_1$ and $\neg x_1$ are activated and distinguished states of variable X_1 respectively.

do not control or actively observe, but that are present, we add the probability of leak (p_L) to the product on the right hand side of Equation 3:

$$P(+y|\mathbf{x}) = 1 - (1 - p_L) \prod_{i \in I(\mathbf{x})} (1 - p_i) . \quad (3)$$

2.3 Eliciting Parameters of ICI Models from Data

The problem of learning Noisy/Leaky-MAX parameters from data was addressed previously by Onisko and Druzdzel [1], Zagorecki et al. [8], and others (e.g., [4, 9]). The main problem with defining and learning the CPT is the exponential growth of the table in the number of parents. The number of independent parameters in a CPT (I_{CPT}) can be calculated using Equation 4:

$$I_{\text{CPT}} = (|Y| - 1) \prod_{i=1}^n |X_i| , \quad (4)$$

where $|Y|$ and $|X_i|$ denote the number of states of the child node and the i -th parent respectively. The corresponding number of independent parameters for the Leaky-MAX gate (I_{MAX}) is:

$$I_{\text{MAX}} = (|Y| - 1) \sum_{i=1}^n (|X_i| - 1) + |Y| - 1 . \quad (5)$$

We can see that the ICI models reduce the number of parameters required logarithmically. In real-life problems, it is not uncommon to find models consisting of thousands of variables, some having many parent nodes. In such cases, the size of the model's CPTs can become huge. Not only is the computational complexity of inference within such network high, but constructing such model in the first place can be daunting. What we would have to ask the expert are specific questions about conditional probabilities for every possible scenario that can occur within the problem domain. Assuming n binary parents, this involves eliciting 2^n numerical values from experts (this task becomes already cumbersome for even small values of n). This effort can be eased by learning the parameters from data, which in its basic form revolves around estimating the required conditional probabilities according to the distributions observed in the data. This, however, does not resolve the problem completely. Given a dataset of m records we obtain an average of $\frac{m}{2^n}$ records per conditional probability distribution within a CPT. When the probability distribution over various combinations of parent states is skewed, some parameters will have no corresponding records within the data at all. Most likely we will not be able to supply a sufficiently large dataset to maintain a reasonable records per parameter ratio, which is inherently associated with a large error. Thus, reducing the number of required parameters which define the model not only simplifies the storage and inference, but also the learning from both data and experts.

Oniško and Druždzel [1] propose a simple method of learning Noisy-MAX parameters from data by limiting learning to only those records that describe the parameters directly, namely the cases where only one of the parent nodes is in its activated state. Another approach by Zagórecki and Druždzel [10] aims at learning the full CPT first and then fitting the Noisy/Leaky-MAX parameters which are the closest (in Euclidian distance) to the original CPT distribution. In this research, we propose yet another approach which somewhat resembles the previous two: Learning Leaky-MAX parameters directly from the data (yet using the information from the full dataset), while also minimizing the sum of distances towards each of the probabilistic scenarios in data using the weighted least squares method.

3 Least Squares Approximation

Least Squares (also known as “Ordinary” or “Linear” Least Squares) is a parameter estimation method, commonly used in regression analysis [11]. As we will show in this section, this approach resonates well with the constraints of our problem and can be employed for the purpose of learning parameters of canonical models from data. In this research we focus on the Leaky-OR/MAX gates, yet the same approach can be employed to the Leaky-AND/MIN models and other canonical gates, as long as it is possible to frame the problem in terms of a system of linear equations.

3.1 Expressing Probabilistic Information as a System of Linear Equations

Following Equation 3 we can express any conditional probability within the CPT using the Leaky-MAX parameters. Thus, given an arbitrary observation of parent states \mathbf{x} and child state \mathbf{y} we have the following:

$$1 - \hat{P}(+y|\mathbf{x}) = (1 - \hat{p}_L) \prod_{i \in I(\mathbf{x})} (1 - \hat{p}_i) , \quad (6)$$

where $\hat{P}(+y|\mathbf{x})$, \hat{p}_L and \hat{p}_i are the estimations of $P(+y|\mathbf{x})$, p_L and p_i from data. Taking the logarithm of both sides gives us:

$$\log(1 - \hat{P}(+y|\mathbf{x})) = \log(1 - \hat{p}_L) + \sum_{i \in I(\mathbf{x})} \log(1 - \hat{p}_i) . \quad (7)$$

By introducing substitutions $\hat{q} = \log(1 - \hat{P}(+y|\mathbf{x}))$, $\hat{q}_L = \log(1 - \hat{p}_L)$ and $\hat{q}_i = \log(1 - \hat{p}_i)$ we obtain a linear equation:

$$\hat{q} = \hat{q}_L + \sum_{i \in I(\mathbf{x})} \hat{q}_i , \quad (8)$$

with \hat{q} as a constant, and $\hat{q}_L, \hat{q}_1, \dots, \hat{q}_n$ as the unknowns. Repeating the steps above for each combination \mathbf{x} observed within the data gives us a system of linear

equations. Solving for \hat{q}_i and \hat{q}_L gives us the corresponding values of $\log(1 - \hat{p}_i)$ and $\log(1 - \hat{p}_L)$, from which we can obtain the original Leaky-MAX parameters \hat{p}_i and \hat{p}_L :

$$\hat{p}_k = 1 - \exp(\hat{q}_k) \quad \text{for } k \in \{1, \dots, n, L\}. \quad (9)$$

Example 1. Let the binary nodes X_1 , X_2 and Y form a Leaky-OR model, giving us 8 possible observations in the sample (Figure 3a). Let us assume a sample of 100 records reflecting the structure (Table 3b). Expressing the observations from

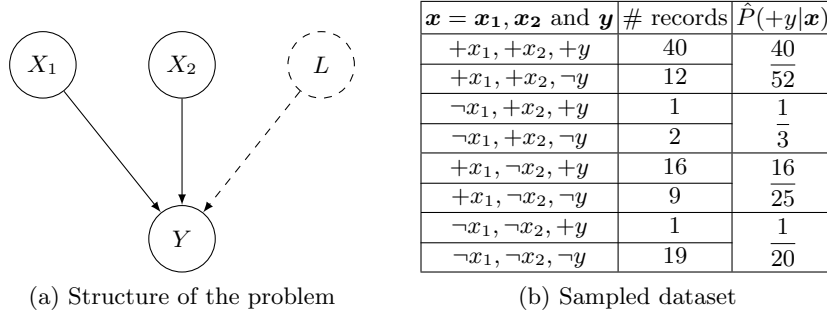


Fig. 3: Leaky-OR structure (left) along with a sample of 100 records (right).

the sample using the Equation 6, gives us the following system of equations:

$$\begin{bmatrix} (1 - \hat{p}_1) \cdot (1 - \hat{p}_2) \cdot (1 - \hat{p}_L) \\ (1 - \hat{p}_1) \cdot (1 - \hat{p}_2) \cdot (1 - \hat{p}_L) \\ (1 - \hat{p}_1) \cdot (1 - \hat{p}_L) \\ (1 - \hat{p}_L) \end{bmatrix} = \begin{bmatrix} 1 - \hat{P}(+y | +x_1, +x_2) \\ 1 - \hat{P}(+y | \neg x_1, +x_2) \\ 1 - \hat{P}(+y | +x_1, \neg x_2) \\ 1 - \hat{P}(+y | \neg x_1, \neg x_2) \end{bmatrix}. \quad (10)$$

We take logarithm of both sides of each of the equations:

$$\begin{bmatrix} \log(1 - \hat{p}_1) + \log(1 - \hat{p}_2) + \log(1 - \hat{p}_L) \\ \log(1 - \hat{p}_2) + \log(1 - \hat{p}_L) \\ \log(1 - \hat{p}_1) + \log(1 - \hat{p}_L) \\ \log(1 - \hat{p}_L) \end{bmatrix} = \begin{bmatrix} \log(1 - \frac{40}{52}) \\ \log(1 - \frac{1}{3}) \\ \log(1 - \frac{16}{25}) \\ \log(1 - \frac{1}{20}) \end{bmatrix}, \quad (11)$$

and apply the substitutions $\hat{q}_L = \log(1 - \hat{p}_L)$ and $\hat{q}_i = \log(1 - \hat{p}_i)$, which gives us the following system of linear equations:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \hat{q}_L \end{bmatrix} = \begin{bmatrix} \log(\frac{12}{52}) \\ \log(\frac{2}{3}) \\ \log(\frac{9}{25}) \\ \log(\frac{19}{20}) \end{bmatrix} = \begin{bmatrix} -1.466 \dots \\ -0.405 \dots \\ -1.021 \dots \\ -0.051 \dots \end{bmatrix}. \quad (12)$$

■

3.2 Weighted Least Squares Method

In the previous section, we have shown a transition from discrete data, obeying the assumptions of the Leaky-MAX model, to a system of linear equations. The input in our case is an overdetermined system of linear equations, which can be thought of as a set of hyperplanes. Least squares method allows for finding a solution that minimizes the sum of vertical distances to each of the planes.

Given an overdetermined system:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \quad (13)$$

where \mathbf{A} is an $m \times n$ matrix of real-valued coefficients, and $\mathbf{x} \in R^n$ and $\mathbf{b} \in R^m$, we can obtain an approximate solution \mathbf{x}' solving the following:

$$\mathbf{A}^T \mathbf{A} \mathbf{x}' = \mathbf{A}^T \mathbf{b}. \quad (14)$$

Thus obtained solution \mathbf{x}' minimizes the sum of the squares of the errors for each of the m equations. Since in our case the equations are obtained from data, least squares method has no means of distinguishing between these equations that were represented by a larger fraction of the dataset (potential lower error) and those equations that were represented by only a handful of records (higher error). For that reason we employ the weighted variant of the method [11], thus allowing to promote the more significant equations.

Given the system of equations (13) and a diagonal matrix \mathbf{W} with weights on its main diagonal,⁵ we solve for the weight-induced approximate solution \mathbf{x}'' using the following:

$$\mathbf{A}^T \mathbf{W} \mathbf{A} \mathbf{x}'' = \mathbf{A}^T \mathbf{W} \mathbf{b}. \quad (15)$$

Example 2. Let us continue the previous example and assume an overdetermined system in Equation 12. By assuming

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} -1.466 \dots \\ -0.405 \dots \\ -1.021 \dots \\ -0.051 \dots \end{bmatrix}, \mathbf{A}^T \mathbf{A} = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 4 \end{bmatrix}, \mathbf{A}^T \mathbf{b} = \begin{bmatrix} -1.871 \dots \\ -2.487 \dots \\ -2.944 \dots \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \hat{q}_1 \\ \hat{q}_2 \\ \hat{q}_L \end{bmatrix}, \quad (16)$$

we can solve for an ordinary least squares solution \mathbf{x}'

$$\mathbf{A}^T \mathbf{A} \mathbf{x}' = \mathbf{A}^T \mathbf{b} \Rightarrow \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 4 \end{bmatrix} \mathbf{x}' = \begin{bmatrix} -1.821 \dots \\ -2.555 \dots \\ -2.894 \dots \end{bmatrix} \Rightarrow \mathbf{x}' = \begin{bmatrix} -0.399 \dots \\ -1.015 \dots \\ -0.028 \dots \end{bmatrix}. \quad (17)$$

Since \mathbf{x}' is a vector of approximate parameters q'_1 , q'_2 and q'_L , we apply Equation 9 to obtain the corresponding parameters p'_1 , p'_2 and p'_L :

$$\begin{bmatrix} p'_1 \\ p'_2 \\ p'_L \end{bmatrix} = \begin{bmatrix} 1 - \exp(q'_1) \\ 1 - \exp(q'_2) \\ 1 - \exp(q'_L) \end{bmatrix} = \begin{bmatrix} 0.329 \dots \\ 0.638 \dots \\ 0.028 \dots \end{bmatrix}. \quad (18)$$

⁵ $w_{ii} \in \mathbf{W}$ describes the weight of the i -th equation. Normalization of the weights is not necessary.

Alternatively, we can apply weights:⁶

$$\mathbf{W} = \begin{bmatrix} 52 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 25 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix}, \mathbf{A}^T \mathbf{W} \mathbf{A} = \begin{bmatrix} 55 & 52 & 55 \\ 52 & 77 & 77 \\ 55 & 77 & 100 \end{bmatrix}, \mathbf{A}^T \mathbf{W} \mathbf{b} = \begin{bmatrix} -77.465 \dots \\ -101.790 \dots \\ -104.033 \dots \end{bmatrix}, \quad (19)$$

and solve for a weighted least squares solution \mathbf{x}'' :

$$\mathbf{A}^T \mathbf{W} \mathbf{A} \mathbf{x}'' = \mathbf{A}^T \mathbf{W} \mathbf{b} \Rightarrow \begin{bmatrix} 55 & 52 & 55 \\ 52 & 77 & 77 \\ 55 & 77 & 100 \end{bmatrix} \mathbf{x}'' = \begin{bmatrix} -77.465 \dots \\ -101.790 \dots \\ -104.033 \dots \end{bmatrix} \Rightarrow \mathbf{x}'' = \begin{bmatrix} -0.432 \dots \\ -0.988 \dots \\ -0.041 \dots \end{bmatrix}. \quad (20)$$

As previously highlighted, in order to obtain the final parameters p_1'', p_2'' and p_L'' , we employ the Equation 9:

$$\begin{bmatrix} p_1'' \\ p_2'' \\ p_L'' \end{bmatrix} = \begin{bmatrix} 1 - \exp(q_1'') \\ 1 - \exp(q_2'') \\ 1 - \exp(q_L'') \end{bmatrix} = \begin{bmatrix} 0.351 \dots \\ 0.628 \dots \\ 0.040 \dots \end{bmatrix}. \quad (21)$$

■

4 Empirical Performance

This section describes experiments performed to test our approach in practice. We perform two types of experiments: learning parameters using datasets generated from an ideal Leaky-MAX definition and a definition “distorted” by a varying noise parameter κ . The latter models the situation in which we are fitting a Leaky-MAX gate to a distribution that is not exactly a Leaky-MAX.

4.1 Data Generation

Data for the experiment were prepared and generated using the GeNIe and SMILE[Ⓢ] software packages.⁷ We focus on learning networks composed of only one child and the number of binary parents varying between 2 and 6 plus leak. After testing the accuracy of our method on non-binary parents, we have found its performance to be correlated with the number of possible equations for given test case. For that reason we model the “hardness” of the problem simply by a varying number of binary parents. In order to provide statistical results, we generate a thousand randomized networks for each number of parents. The prior probabilities of parents, as well as the Leaky-MAX definition of the child are randomized uniformly each time. For each of the 5000 networks, we generate 20 random datasets consisting of 100, 200, ..., 1900 and 2000 records, which gives us 100,000 datasets total.

⁶ For simplicity we assume $w_{ii} \in W$ to be the number of records describing i -th equation (see Table 3b).

⁷ Available at <http://genie.sis.pitt.edu/>.

Noise-Induced Leaky-MAX: In order to test the accuracy of both methods in cases when the original distribution differs from the ideal Leaky-MAX, we introduce the “distortion” to the definition. We expand the Leaky-MAX parameters to a full CPT and introduce the noise to each of the probability distributions \mathbf{p}_k (columns in CPT), by sampling a variate x from the Dirichlet distribution, as defined by the following probability density function:

$$\text{Dir}(x_1, \dots, x_{n-1}; \alpha_1, \dots, \alpha_n) = \frac{1}{B(\alpha)} \prod_{i=1}^n x_i^{\alpha_i-1}, \quad (22)$$

where $\alpha = \kappa \cdot \mathbf{p}_k$ and $B(\alpha)$ is the multinomial Beta function.

Figure 4 shows a visualization of the distortion of the distribution $q = [0.2, 0.3, 0.5]$ for decreasing values of κ . For the noise-induced data, we generate similar groups of 100,000 learning datasets for $\kappa = 100, 25$ and 10.

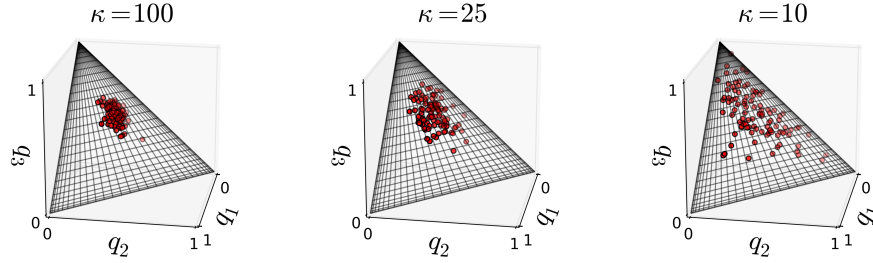


Fig. 4: Distortion of original distribution $q = [0.2, 0.3, 0.5]$ with a varying parameter κ visualized by sampling a 100 new variates from the Dirichlet distribution.

4.2 Tested Learning Algorithms.

We compare three approaches to learning the Leaky-MAX parameters: (1) Simple learning method by Onisko and Druzdzel [1], (2) Leaky-MAX fitting to CPT by Zagorecki and Druzdzel [10], and (3) Weighted Least Squares method proposed in this paper. Small datasets may miss some of the information necessary for the estimation of the parameters: if m records describe a given combination of parent states, and the effect variable is activated in k of those records, we can safely assume that the approximate conditional probability that the variable will be activated is then equal to $\frac{k}{m}$. However, when none or all of the m records contain the child in an activated state, we end up with probabilities 0 and 1 respectively, which are best avoided when defining a probabilistic model [12]. Our method of handling such extreme cases (for both methods) is as follows:

- If none of the observed m records contains the child in an activated state, assume probability $\frac{1}{m+1}$.

- If all of the observed m records contain the child in an activated state, assume probability $\frac{m}{m+1}$.
- If a given scenario is nonexistent in the data, i.e., the joint probability of a given combination of parents states was very small, assume the uniform distribution for the child variable (Simple learning only).

In the remainder of this section, we will describe the three methods in detail:

Simple Learning

1. Perform a sweep through the dataset in search of records with only one parent being in its activated state and compute the conditional probabilities for the child node.
2. Since leak (denoted as L) is always taken into account, we are also interested in the conditional probability of the child being present, when all of the parents are in their corresponding distinguished states.
3. So far, the obtained values are the so-called *compound parameters* p_c [2], as they reflect the fact that implicit causes (leak node) might have acted upon the effect variable. In order to obtain each of the corresponding *net parameters* p_n [6], we compute the following:

$$p_n = 1 - \frac{1 - p_c}{1 - p_L} . \quad (23)$$

In case of a small dataset, it might occur that $p_c < p_L$, leading to p_n being a negative value, in which case we assume a uniform distribution.

Weighted Least Squares Method

1. Express the data as system of linear equations (Equations 6 through 8).
2. To reduce the potential error in the estimations, remove the equations which were represented by fewer than 10 records. While more relaxed and flexible rules can be considered here, they impact the general performance of the algorithm minimally, and only for the smallest datasets.
3. Assume the following weight for each of the equations:

$$w = (m \cdot 0.5^\delta)^2 , \quad (24)$$

where m is the total number of records describing a given equation, and δ is the number of parents that were in their activated state in a given probabilistic relationship. We want to promote the equations which were represented by a larger share of the sample, but at the same time we want to penalize complex equations with many activated parents. Higher prevalence of the latter results in more complex linear combination for the final parameter, increasing the error propagation. We have experimented with the degree of penalty that could be applied to each equation, and found that weight

decreasing exponentially with the number of activated parents led to highest accuracy ⁸. Proposed constants achieved the best results in preliminary experiments.

4. Apply the Weighted Least Squares method as described in the previous section. If any errors occur during learning (i.e., computed probability is negative or larger than 1), compute given parameter using the Simple learning method described above.

Leaky-MAX Fitting to CPT

1. Initialize the CPT and the prior probabilities with uniform distributions.
2. Perform the Expectation-maximization learning of the full CPT.
3. Perform fitting of the Leaky-MAX parameters to thus obtained CPT using the algorithm of Zagórecki and Druzdzel [10].⁹

4.3 Performance Assessment

This section describes our methodology for the performance evaluation of the tested algorithms (see Figure 5):

1. The first step differs for each of the experiments:
 - (a) Leaky-MAX experiment: Parameters of the network (parents' prior probabilities and CPT) are randomized uniformly.
 - (b) Noise-induced experiment: Parameters of the network are randomized uniformly and then subjected to a Dirichlet noising with parameter κ .
2. Thus obtained reference network is used for sampling of the datasets.
3. Tested algorithms learn the Leaky-MAX parameters from the data and expand them to full CPT.
4. Learned CPTs are compared against the original CPT using the Hellinger distance [13].

We use two metrics based on the Hellinger distance to measure the average and maximal errors within the learned CPT:

$$H_{\text{MAX}}(\mathbb{P}, \mathbb{Q}) = \max\{H(\mathbb{P}, \mathbb{Q}, j) \mid j \in J\}, \quad (25)$$

$$H_{\text{AVG}}(\mathbb{P}, \mathbb{Q}) = \text{avg}\{H(\mathbb{P}, \mathbb{Q}, j) \mid j \in J\}, \quad (26)$$

where $H(\mathbb{P}, \mathbb{Q}, j)$ is the Hellinger distance between the j -th columns (out of J total) of the CPTs \mathbb{P} and \mathbb{Q} :

$$H(\mathbb{P}, \mathbb{Q}, j) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^I \left(\sqrt{\mathbb{P}_j^i} - \sqrt{\mathbb{Q}_j^i} \right)^2}, \quad (27)$$

and I is the number of states of the child node.

⁸ Linear and polynomial penalty functions were also considered.

⁹ Implementation of the Leaky-MAX fitting algorithm is available in GeNie/SMILE[©] software packages at <http://genie.sis.pitt.edu/>.

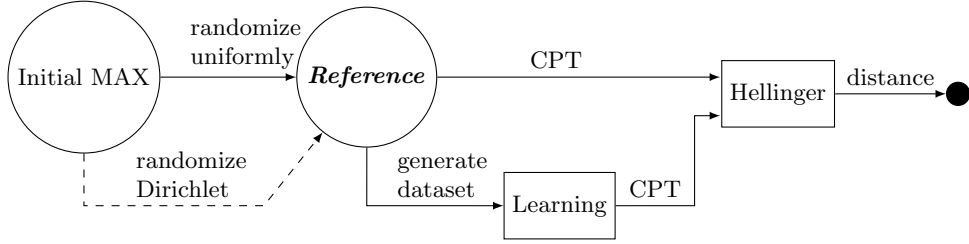


Fig. 5: Experiment flowchart. Dashed line is a noise-induced variant to the uniform randomization of the reference Leaky-MAX parameters.

4.4 Results

Figure 6 presents the results of the first experiment. We can observe almost unanimous tendency for the Least Squares-based learning to achieve better accuracy than the Simple learning method. However, for 2 and 3 parents Leaky-MAX fitting seems to achieve the best performance in both average and maximal error. The situation changes as we increase the number of parents to 5 and 6, in which case the accuracy of the Leaky-MAX fitting deteriorates drastically.

Figure 7 presents the results in more detail for 100, 1000 and 2000 records. Each of the boxplots represents 1000 experiment repetitions, with the center box describing the 50% of cases between the two quartiles (also known as the *interquartile range* – IQR). The whiskers extend further by a distances of $1.5 \cdot \text{IQR}$, with the outliers marked in red. Above each of the boxplots we gather four statistical measures (rounded to 4 decimal points): mean (μ), median (\tilde{x}), standard deviation (σ) and a *win ratio* (ω), last of which was also shown on a separate plot on the right. Win ratio is simply the fraction of the 1000 experiments in which given method achieved the minimal distance among other algorithms (including ties). The best value for each of the statistics among the tested algorithms is signified with the boldface font.

Leaky-MAX fitting achieves the best overall performance for 2 parents by almost all statistical measures across the experiment. Similar scenario (although not as apparent) occurs for 3 parents, with Least Squares method performing akin to Leaky-MAX fitting in mean, median and win ratio. However, when we look at the case of 4 parents, the Least Squares method starts to outperform the remaining methods and loses only to Simple learning in standard deviation for 100 records. The effect continues to persist for 5 and 6 parents — for the latter, Least Squares obtains the best statistical indicators for all three dataset sizes. Analyzing the corresponding plots of the win ratio suggests that the Least Squares method is almost consistently better than the Simple learning approach, while for 4 and 6 parents it clearly outperforms both methods.

We show the results of the second experiment for 3, 4, and 5 parents in Figure 8. Since all methods assume certain unmet properties about the data, it is expected that the learning quality will deteriorate with lower values of κ . For relatively small deviation from the ideal Leaky-MAX ($\kappa = 100$), we observe that

the Least Squares method achieves the accuracy as good as the Simple learning method, and not clearly different from the previous experiment. As we divert away from the perfect scenario ($\kappa = 25$ and 10) a degeneration in accuracy starts to occur, especially for the Least Squares method when assessed by the H_{MAX} metric. The Leaky-MAX fitting method seems to be more resistant to distortion, especially when one considers maximal error – the degeneration is not as severe as it is for the remaining methods.

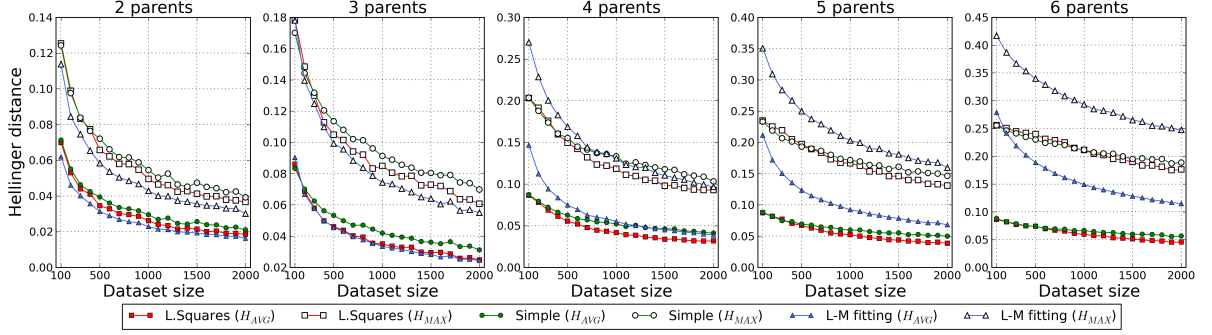


Fig. 6: Simple learning, Least Squares method and Leaky-MAX fitting approach. Average over 1000 repetitions.

4.5 Discussion

The degrading performance of the Leaky-MAX fitting for 4 and more parents is not surprising, as the main drawback of the method is the lack of information on the accuracy of the estimated parameters in the CPT after the Expectation-maximization algorithm. When the number of parents is small (i.e., 2 or 3 nodes, giving 4 and 8 parameters respectively), individual parameters in the CPT are estimated based on a larger share of the sample. However, exponential increase in the number of CPT parameters, without a corresponding exponential increase in the number of records, must lead to higher frequency of poorly estimated or missing entries in a CPT. Missing information on the individual parameters “quality” leads to higher error propagation during the Leaky-MAX fitting phase.

If we analyze the cases of 100–300 records and 5 parents, Least Squares and Simple learning methods perform similarly in terms of win ratio. This is due to the fact that the Least Squares method falls back to Simple learning when the information within data is poorly estimated. However, as we increase the number of records to 400 and more, we can notice an increasing difference between the two methods, suggesting a higher frequency of “exclusive wins” for the Least Squares approach (similarly for 6 parents with a threshold of 500 records).

Noise-induced experiment revealed a poor performance of Least Squares approach for the non-Leaky-MAX data. It is expected, as the assumptions about

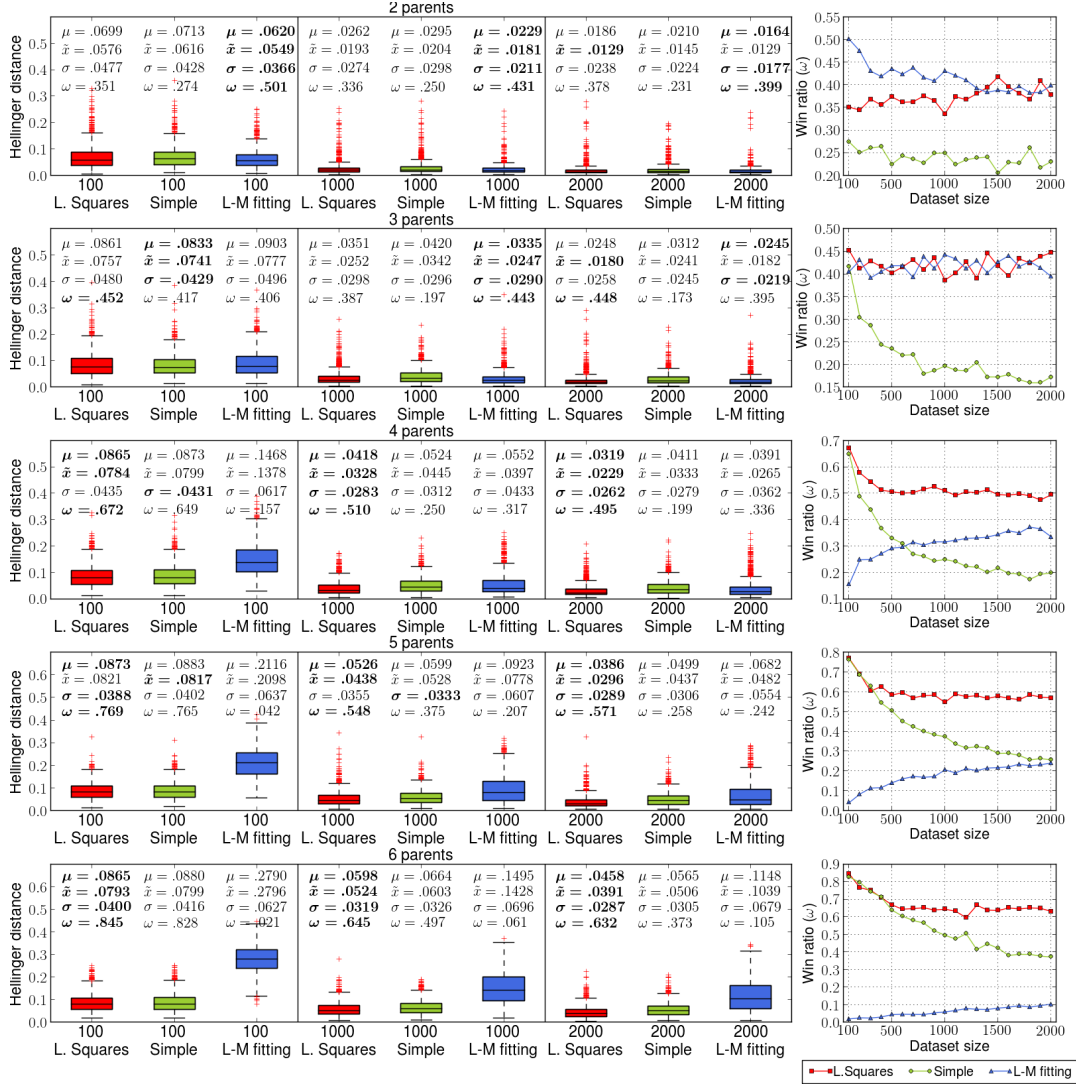


Fig. 7: Statistical performance of Least Squares (*L.Squares*), Simple learning (*Simple*) and Leaky-MAX fitting (*L-M fitting*) methods. Mean (μ), median (\tilde{x}), standard deviation (σ) and win ratio (ω) over 1000 experiment repetitions (H_{AVG} metric).

the Leaky-MAX distributions are not met – most of the presumed relationships between the equations and computations done, are simply incorrect. This leads to a larger degree of error propagation in the final parameters. Simple learning focuses only on the subset of data, without considering any relationships between observations. Intuitively, minor utilization of the false assumptions about the data in the simple method leads to fewer points of failure.

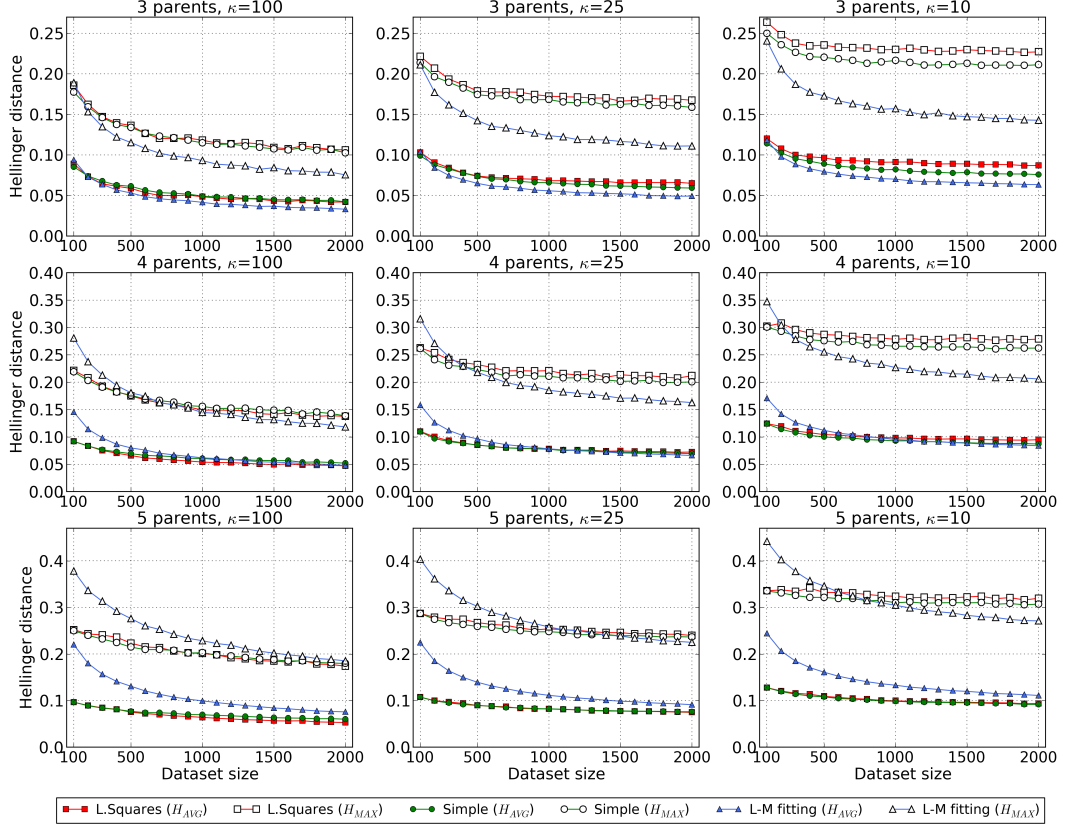


Fig. 8: Average accuracy of the Least Squares, Simple learning and Leaky-MAX fitting methods for various degrees of noise introduced to the learning data.

5 Conclusions

Learning Leaky-MAX parameters from data can be improved in terms of accuracy by employing a method able to extract more information from the sample. The method proposed in this research achieves better results in terms of learning precision than the simple approach and the Leaky-MAX fitting approach, assuming that the data fits the Leaky-MAX definition. By employing the assumptions of canonical models, we show that it is possible to render a connection between the probabilistic relationships and linear algebra. This opens up room for future research, as we believe that the approach thus presented resonates equally well with other canonical models. Besides a possible wider application of our solution, the core method itself can be improved as the problem of quasi linear regression presented in this research was given much attention in the field of regression analysis.

Acknowledgments

We acknowledge the support the National Institute of Health under grant number U01HL101066-01. Implementation of this work is based on SMILE[©], a Bayesian inference engine developed at the Decision Systems Laboratory and available at <http://genie.sis.pitt.edu/>.

References

1. Oniśko, A., Druzdzal, M.J., Wasyluk, H.: Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates. In: Working notes on the European Conference on Artificial Intelligence (ECAI) Workshop Bayesian and Causal Networks: From Inference to Data Mining. (August 22 2000)
2. Henrion, M.: Some practical issues in constructing belief networks. In: Uncertainty in Artificial Intelligence 3 Annual Conference on Uncertainty in Artificial Intelligence (UAI-87), Amsterdam, NL, Elsevier Science (1987) 161–173
3. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)
4. Heckerman, D.: A tutorial on learning with Bayesian networks. In Jordan, M.I., ed.: Learning in Graphical Models. The MIT Press, Cambridge, Massachusetts (1998) Available on web at <ftp://ftp.research.microsoft.com/pub/tr/TR-95-06.ps>.
5. Díez, F.J., Druzdzal, M.J.: Canonical probabilistic models for knowledge engineering. Technical report, UNED, Madrid, Spain (2006)
6. Díez, F.: Parameter adjustment in Bayes networks. the generalized noisy OR-gate. In: Proceedings of the Ninth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-93), San Francisco, CA, Morgan Kaufmann (1993) 99–105
7. Srinivas, S.: A generalization of the noisy-OR model. In: Proceedings of the Ninth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-93), San Francisco, CA, Morgan Kaufmann (1993) 208–215
8. Zagorecki, A., Voortman, M., Druzdzal, M.J.: Decomposing local probability distributions in Bayesian networks for improved inference and parameter learning. In Sutcliffe, G., Goebel, R., eds.: Recent Advances in Artificial Intelligence: Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference (FLAIRS-2006), Menlo Park, CA, AAAI Press (2006) 860–865
9. Friedman, N., Goldszmidt, M.: Learning Bayesian networks with local structure. In Jordan, M.I., ed.: Learning and Inference in Graphical Models. The MIT Press, Cambridge, MA (1999) 421–459
10. Zagorecki, A., Druzdzal, M.J.: Knowledge engineering for Bayesian networks: How common are noisy-MAX distributions in practice? IEEE Transactions on Systems, Man, and Cybernetics: Systems **43**(1) (January 2013) 186–195
11. Wasserman, L.: All of nonparametric statistics. Springer (2006)
12. Oniśko, A., Druzdzal, M.J.: Impact of precision of Bayesian network parameters on accuracy of medical diagnostic systems. Artificial Intelligence in Medicine **57**(3) (March 2013) 197–206
13. Gibbs, A.L., Su, F.E.: On choosing and bounding probability metrics. International statistical review **70**(3) (2002) 419–435