LEARNING BAYESIAN NETWORK PARAMETERS FROM SMALL DATA SETS BY USING RANKED NODES METHOD

AZ MİKTARDA VERİDEN SIRALI DÜĞÜMLER YÖNTEMİ İLE BAYES AĞI PARAMETRELERİNİN ÖĞRENİLMESİ

SILA İŞYAR

ASST. PROF. DR. BARBAROS YET

ASST. PROF. DR. CEREN TUNCER ŞAKAR

Submitted to Graduate School of Engineering of Hacettepe University as a Partial Fulfillment to the Requirements for the Award of the Degree of Master of Science in Industrial Engineering.

2020

ÖZET

AZ MİKTARDA VERİDEN SIRALI DÜĞÜMLER YÖNTEMİ İLE BAYES AĞI PARAMETRELERİNİN ÖĞRENİLMESİ

Sıla İŞYAR

Yüksek Lisans, Endüstri Mühendisliği Bölümü

Tez Danışmanı: Dr. Öğr. Üyesi Barbaros YET

Eş Danışman: Dr. Öğr. Üyesi Ceren TUNCER ŞAKAR

Ağustos 2020, 58 sayfa

Bayes Ağları (BA) belirsizlik altında karar destek problemleri için uygun bir modelleme yaklaşımı sunan olasılıksal grafiksel modellerdir. BA'nın hem grafiksel yapısı hem de koşullu olasılık tabloları verilerden öğrenilebilir. Bu çalışma verilerin sınırlı miktarda olduğu durumlarda BA'nın koşullu olasılıklarını öğrenmeye odaklanmaktadır.

Sıralı düğümler yöntemi, değerleri arasında sıralı ilişki bulunan değişkenlerin koşullu olasılık tablolarını tanımlamak için gereken parametre sayısını azaltmak için önerilmiştir. Bu yöntem sıralı BA değişkenlerini Kesilmiş Normal dağılım ile yakınsayarak, koşullu olasılık dağılımlarını tabloların gerektirdiğinden daha az parametre ile tanımlar. Bu avantaja karşın, önceki çalışmalarda sıralı düğümler yöntemi yalnızca BA'nı uzman bilgisi ile tanımlamak için kullanılmış, sıralı düğümleri veriden öğrenmeye yönelik yöntemler incelenmemiştir.

Bu çalışma sıralı düğümleri veriden öğrenmek için bir yaklaşım önermektedir. Bu yaklaşım sıralı düğümlerin sürekli dağılımları arasındaki ilişkiyi öğrenmek için kesilmiş normal regresyon, beta regresyon ve lineer regresyon yöntemlerini kullanmakta ve sonra bunları BA için koşullu olasılık tablolarına dönüştürmektedir. Önerilen yöntemin performansı farklı veri boyutları ve BA yapılarına sahip deneyler ile değerlendirilmiştir.

Anahtar Kelimeler: Karar Destek Sistemleri, Bayes Ağları, Sıralı Düğümler Yöntemi, Veriden Öğrenme, Yapay Zeka, Dereceli Azalma Algoritması

ABSTRACT

LEARNING BAYESIAN NETWORK PARAMETERS FROM SMALL DATA SETS BY USING RANKED NODES METHOD

Sıla İŞYAR

Master of Science, Department of Industrial Engineering

Supervisor: Asst. Prof. Dr. Barbaros YET

Co- Supervisor: Asst. Prof. Dr. Ceren TUNCER ŞAKAR

August 2020, 58 pages

Bayesian Networks (BNs) are graphical probabilistic models that offer a suitable modeling approach for decision support problems under uncertainty. Both the graphical structure and conditional probability tables of BNs can be learned from data. This study focuses on learning conditional probability of BNs in cases where data are in limited amounts.

The ranked nodes method has been proposed to reduce the number of parameters required to define conditional probability tables of variables with ordinal states. This method assigns an underlying Truncated Normal distribution to ordinal BN variables, and it defines the conditional probability distribution with equations with fewer parameters than those required by tables. Despite this advantage, in the previous studies, the method of

ranked nodes was used only to elicit BNs from expert knowledge, and methods for learning ranked nodes from data were not examined.

This study proposes an approach to learn ranked nodes from data. This approach uses Truncated Normal regression, Beta regression and linear regression to learn the relations between the underlying continuous distributions of ranked nodes, and then it transforms these to conditional probability tables for BNs. The performance of the proposed method has been evaluated using experiments with different data sizes and BN structures.

Keywords: Decision Support Systems, Bayesian Networks, Ranked Nodes Method, Learning from Data, Artificial Intelligence, Gradient Descent Algorithm

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my supervisor Asst. Prof. Dr. Barbaros YET and my co-supervisor Asst. Prof. Dr. Ceren TUNCER ŞAKAR for their great support, brilliant ideas, valuable time and encouragement that make this study possible. The experiences that I gained by working with them will be a guide in my personal and professional life.

I wish to thank my mother and my father for always being on my side. I am always grateful to them for their trust and support for me.

This work was supported by an Institutional Links grant, ID 352394702, under the Turkey – Newton - Katip Çelebi Fund partnership. The grant is funded by the UK Department of Business, Energy and Industrial Strategy (BEIS) and The Scientific and Technological Research Council of Turkey (TUBITAK - ID 217M518) and delivered by the British Council. For further information, please visit www.newtonfund.ac.uk.

Sıla İŞYAR

August 2020, Ankara

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

BNs	: Bayesian Networks
CPTs	: Conditional Probability Tables
PDF	: Probability Density Function
CDF	: Cumulative Distribution Function
TNormal Distribution	: Truncated Normal Distribution
MLE	: Maximum Likelihood Estimation

1. INTRODUCTION

It is often inevitable to deal with uncertainty when modeling the domains of real-world applications. All models are a simplified representation of reality, and the uncertainty stems from the impossibility of modeling all the different conditions and exceptions in the models (Wiegerinck et al. 2013). Most of real-world decision support models require modeling uncertain domains and reasoning under that uncertainty.

Bayesian Networks (BNs) is one of the decision support methods using probabilistic reasoning method. It is widely used to develop solutions to large-scale decision support problems in the fields of medical diagnostics, software reliability, law, and military practice. These graphic models can be used for example for medical diagnosis and diagnostics, modeling genetic inheritance of diseases, segmenting and describing images, encrypting messages sent through a noisy channel, and localizing and mapping the robot (Koller & Friedman, 2009).

The BNs are formed of nodes and directional arcs. The nodes of BN represent the variables of it and directional arcs model the causality between these nodes. The graphical representation of BN visualizes the causal relationships between its variables and BN's parameters represent the conditional dependencies of it. In addition, BNs model the quantitative power of the connections between variables, allowing automatic updating of probabilistic beliefs about themselves as new information is inserted into the BN model (Korb & Nicholson, 2011).

Modeling an uncertain domain with BN requires two steps as building the model's structure and learning the model's parameters. The graphical structure of BN is created by representing the causal relationships between dependent nodes. After determining the graphical structure, the conditional probability distributions are defined which represent the strength of these causal relationships. As the number of conditional probabilities of

BN structure increase, the amounts of parameters that must be learned increase, and a lot of data is needed to learn these parameters. In most real-world decision support problems, there is not enough data to learn parameters (Yet et al. 2014).

Ranked nodes method is proposed by Fenton et al. (2007) for the cases where the data is insufficient. Ranked Nodes are nodes which are ordered in a continuous and monotonic manner. Ranked Nodes have an underlying continuous probability distribution which will require fewer parameters. But up to now, that method is used to determine conditional probability distributions from expert knowledge; a method for learning the parameters of ranked nodes from data is not yet available in the reviewed literature.

The purpose of this thesis study is to develop methods for learning BN parameters from limited data. The proposed method uses regression and machine-learning based approaches on the underlying continuous distribution of ranked nodes and transforms these learned nodes to typical discrete BN nodes so that they can be used with standard BN software and inference algorithms. The method is implemented in R programming language. The applicability of the developed parameter learning method in different sizes and structures are tested with experiments.

In the remainder of this study, Chapter 2 gives information about BNs. Chapter 3 summarizes the BN structure and parameter learning methods. Chapter 4 covers information about the ranked nodes method and Chapter 5 covers the ranked nodes learning approach from data that are improved in this study. In Chapter 6 some BN models learned by the improved algorithms are given and Chapter 7 summarizes the results.

2. BAYESIAN NETWORKS

2.1. Bayes Theorem

Bayes theorem shows how the conditional probability $P(A \mid B)$ is related to its inverse conditional probability $P(B \mid A)$ as shown below:

$$P(A \mid B) = \frac{P(B \mid A) \times P(B)}{P(A)} \tag{1}$$

Bayes theorem enables us to update our prior belief based on observed evidence when *A* is interpreted as our belief about an uncertain event and *B* as the observed data. Computation of Bayes theorem for large and interdependent uncertain events is challenging. In this case, BNs offer a suitable modeling approach for representing and computing joint probability distributions of such events.

2.2. Bayesian Networks

BNs are probabilistic graphical models in which the causal relationships between variables are expressed by a graphical structure and the conditional probability distributions underlying it. Efficient algorithms are available to make complex Bayesian inference calculations on the BN structure (S. L. Lauritzen & Spiegelhalter, 1988). A BN structure with two nodes *A* and *B* is given in Figure 2.1. There is an edge that comes from node *A* to node *B;* it means there is a direct dependence between *A* and *B*. In other words, the probability distribution of *B* is affected by the probability distribution of *A*. In that position node *A* is called as the parent node and node *B* is called as the child node.

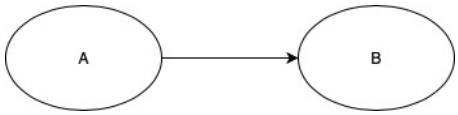


Figure 2.1 Simple BN structure

Conditional Probability Tables (CPTs) of BNs include the parameters of the variables' probability distributions. If a variable has no parent nodes its CPT contains only the probabilities of its states, but if it has parent nodes it contains the probabilities of combinations of its states with its parent variables' states.

Consider the BN in Figure 2.1 with two variables *A* and *B*. Each one has binary values: *A* has a_1 and a_2 and *B* has b_1 and b_2. Their joint probability distribution *P(A, B)* is given below:

$$P(A = a_1, B = b_1) = 0.70$$

$$P(A = a_1, B = b_2) = 0.10$$

$$P(A = a_2, B = b_1) = 0.15$$

$$P(A = a_2, B = b_2) = 0.05$$

That probability distribution is denoted with a factorization as below. Here, distribution of *A* is a prior distribution, and distribution of *B* is a conditional distribution.

$$P(A, B) = P(A) \times P(B \mid A)$$

$\mathbf{a_1}$	$\mathbf{a_2}$
0.8	0.2

	a_1	a_2
b_1	0.875	0.750
b_2	0.125	0.250

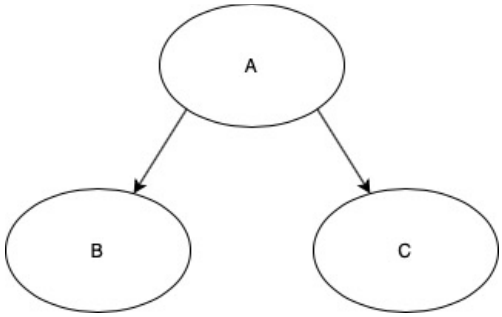


Figure 2.2 BN structure with two child nodes

If the BN consists of more variables, its joint probability distribution becomes more complex (Figure 2.2). For example, if a variable *C* with three values c_1, c_2, and c_3 is added as a child of *A* and *B,* the joint distribution will have 12 parameters.

$$P(A = a_1, B = b_1, C = c_1) = 0.10 \qquad P(A = a_2, B = b_1, C = c_1) = 0.03$$

$$P(A = a_1, B = b_1, C = c_2) = 0.02 \qquad P(A = a_2, B = b_1, C = c_2) = 0.07$$

$$P(A = a_1, B = b_1, C = c_3) = 0.04 \qquad P(A = a_2, B = b_1, C = c_3) = 0.05$$

$$P(A = a_1, B = b_2, C = c_1) = 0.02 \qquad P(A = a_2, B = b_2, C = c_1) = 0.01$$

$$P(A = a_1, B = b_2, C = c_2) = 0.03 \qquad P(A = a_2, B = b_2, C = c_2) = 0.03$$

$$P(A = a_1, B = b_2, C = c_3) = 0.05 \qquad P(A = a_2, B = b_2, C = c_3) = 0.01$$

In that case, the equation given below expresses the joint probability distribution.

$$P(A, B, C) = P(A) \times P(B \mid A) \times P(C \mid A)$$

a_1	a_2
0.8	0.2

	a_1	a_2
b_1	0.875	0.750
b_2	0.125	0.250

	a_1	a_2
c_1	0.150	0.2
c_2	0.288	0.5
c_3	0.562	0.3

As the distribution gets more complex the number of independent parameters contained in that distribution becomes larger. In the first model, 3 independent values are needed to represent the joint distribution and BN. But as the joint distribution gets larger the number of independent parameters increases to 12 for that distribution and 7 for BN.

BNs are graphical tools that represent joint distributions more compactly. Knowing causal interactions and dependencies or independencies in BNs allows the representation of joint distribution more compactly. For example, in a BN with 4 variables, if there is no chance of knowing the independence of conditionality between the variables, the structure will have arcs between all pair of nodes. Figure 2.3 shows the distribution of *A, B, C,* and *D.* In that situation, BN do not encode any conditional independence assertion on the joint probability distribution.

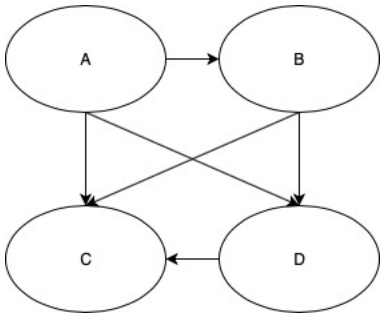


Figure 2.3 BN structure without independences

The joint probability function of variables given in Figure 2.3 can be expressed as below.

$$P(A, B, C, D) = P(A) \times P(B \mid A) \times P(\, C \mid A, B, D) \times P(D \mid A, B)$$

On the other hand, if, for example, it is known that *B* and *D* are independent of *A,* and *C* is independent of *D*, this reduces the arcs between these variables and allows the joint probability function of *A, B, C* and *D* to be expressed more compactly as in Figure 2.4.

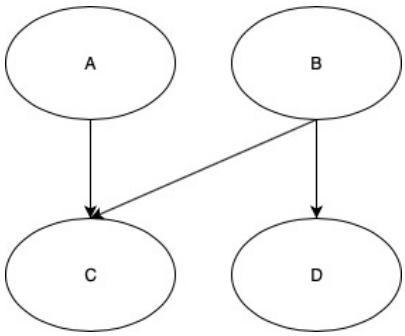


Figure 2.4 BN structure with independences

The joint probability function of variables given in Figure 2.4 can be expressed as below.

$$P(A, B, C, D) = P(A) \times P(B) \times P(C \mid A, B) \times P(D \mid B)$$

BNs can denote the joint probability distributions by factorization. Another way of representing the joint probability is showing the independencies of the model (Koller & Friedman, 2009).

Calculating the marginal probability of some nodes in BN that has complex structure is a very challenging task. To do that a variable elimination algorithm can be used. This algorithm eliminates a variable from joint probability distribution by multiplying its marginal probability with the joint probability and adds up the result and goes on to eliminate all variables. Variable elimination algorithm computes the marginal probability of any node at once, and if another marginal probability is needed another process of calculation will be needed for that variable. In situations like that junction tree algorithm will be more effective because it computes whole marginal probabilities in BN at once (S. L. Lauritzen & Spiegelhalter, 1988).

Entering an evidence to any variable means instantiating (or knowing the possibility of) the state of that variable to some outcome. Knowing the outcome of any variable is called hard evidence. In other words, if it is exactly known that the state of node A is a_1, i.e. $P(A=a_1) = 1$, it is defined as 'a hard evidence (observation) entered to A (or A is instantiated to a_1)'.

But if the state of A is known not certainly but its likelihood function is known, it is called uncertain evidence. For example, if the state of node A is equal to a_1 with %80 probability and equal to a_2 with %20 probability (0.8, 0.2) there is an uncertain evidence about A.

In BNs if we call parent nodes as causes and child nodes as effects;

- If an evidence is entered to cause node, that evidence will update the CPT of the effect node and it is called causal inference.
- If an evidence is entered to effect node, that evidence will update the CPT of the cause node and it is called diagnostic inference.

A set of three nodes in a BN can connect to each other in 3 different ways; i.e. serial, diverging and converging connection.

Serial Connection

Accordance with the definition of causal inference, if an is evidence entered to *A* in serial connection it will update *B* and then *C* (Figure 2.5). Similarly, if an evidence is entered to *C* it will update respectively *B* and then *A* by diagnostic connection. But if an evidence is entered to *B*, then any evidence entered to *A* cannot transform to *C* because *B* blocks that transformation.

The information flow from *A* to *B* or from *B* to *C* is called forward (or causal) inference. Inversely the information flow from *C* to *B* or from *B* to *A* is called backward (or diagnostic) inference.

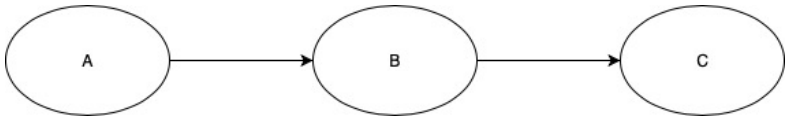


Figure 2.5 Serial connection

A and *C* are conditionally independent given *B*.

Diverging Connection

Diverging connection has a structure like Figure 2.6. An evidence entered to *B* is transmitted relevantly to *A* and then to *C* in diverging connection. But after entering an observation to *A*, if another evidence is entered to *B,* it cannot update *C*.

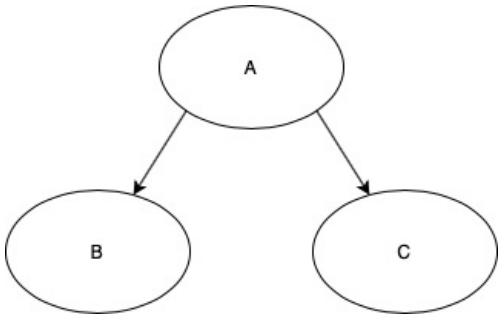


Figure 2.6 Diverging connection

A and *C* are conditionally independent given *B*.

Converging Connection

Converging connection's structure is given in Figure 2.7. Entering an evidence to *A* will affect *B* but it cannot affect *C*. After entering an evidence to *B*, entering another evidence to *A* will update *C*. It is called **explaining away** behavior of BNs. Observing the child evidence of BN provides some predictions about its parents in BNs that have converging connections.

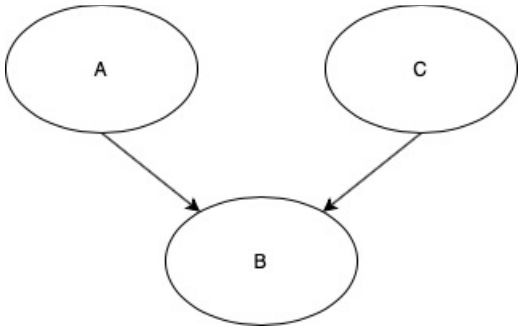


Figure 2.7 Converging connection

A and *C* are conditionally independent given *B*.

D-separation: D-separation occurs in 2 conditions.

When there are two different variables *A* and *B* in a BN that are not connected to each other but, they connected to a middle variable *V* between them:

- if the connection is serial or diverging and the middle variable V is observed these two different variables A and B will be d-separated.
- if the connection is converging and the middle variable V and its descendants are not observed these two different variables A and B will be d-separated.

If these two conditions are not provided A and B will be d-connected.

A Markov Blanket of a node contains a set of variables which makes that node independent of other nodes when they are observed. Markov blanket of any node contains in its:

- Parents,
- Children and
- Other parents of its children.

3. CONSTRUCTION OF BAYESIAN NETWORKS

The dependencies in BNs are represented by CPTs in an uncertain domain and shown by edges when the variables are represented by nodes. The following steps are used to model and evaluate an uncertain domain using a BN:

1. Define the variables and create them as nodes.
2. Determine the type of node as discrete or continuous.
3. Determine the states of each node.
4. Identify the relational variables in the problem or identify the unrelated variables which are independent.
5. Connect the variables which are dependent to each other by adding an arc from cause to reason variable.
6. Create the CPTs of each node.
7. Validate and test the model.

Figure 3.1 represents a BN model about predicting the risk of flood levels. Firstly, the probable causes of flood may be defined as rainfall, river water level, and quality of flood barrier. Then the node types of flood, rainfall, river water level, and quality of flood barrier can be determined as discrete, discrete, continuous, and continuous respectively. Then the relationship between variables is determined. Since flood causes are rainfall, river water level, and quality of flood barriers, these nodes are defined as parents of flood node and three arcs are added from these nodes to flood node. Then CPT of flood is created which is conditioned to its parent nodes (N. Fenton & Neil, 2012).

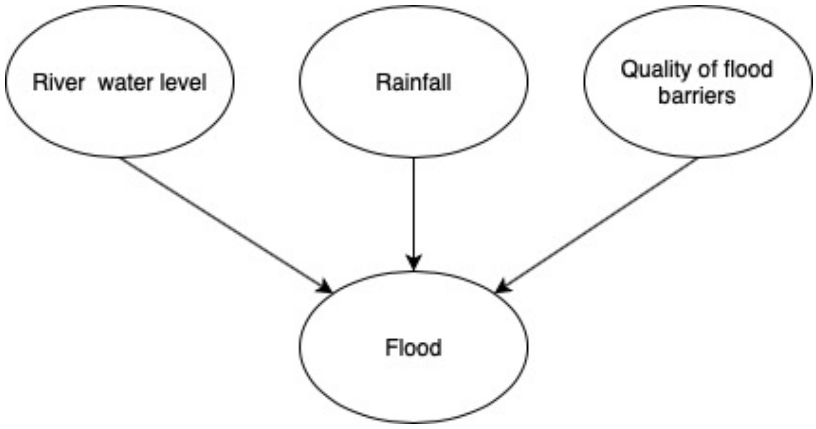


Figure 3.1 A flood risk example of BN

The BNs are used to reasoning between variables so, they should represent all dependencies and independencies between variables in the most correct way. And also, they should not contain redundant or missing links between variables. Because of these reasons building a structure of a BN is a challenging task. The first task of building BNs is determining the dependencies and building the BN structure according to reasoning mechanism. In the remainder of this section, previous studies about defining structure and parameters are shown respectively in Sections 3.1 and 3.2.

3.1. Learning Structure of Bayesian Networks

3.1.1. Learning BN Structure by Knowledge Engineering Methods

Building the graphical structure of BNs is a challenging task especially for BNs which have large structures (Neil et al. 2000). In the study just cited, they mentioned about two different challenges of building large scale BNs which are building the correct graphical structure that represents all reasoning mechanisms and eliciting parameters of those BNs from domain experts. Their interest was on building the right structure for BNs and as there is no reasoning methodology in BNs they came with an approach of “building blocks” which called idioms.

Building BN structure is handled in three steps in that paper. Firstly, they improved a process flow to manage the process of building BN and risks encountered. This process flow begins with identifying the problem and continues with matching the problem to

statements, integrating statements with objects, creating CPTs, and validating BNs. Then they combined the idioms approach with object-oriented BN approach (Koller & Pfeffer, 1997) which is based on representing BNs by dividing it into manageable and reusable fragments in order to simplify the building process of large structured BNs.

Neil et al. (2000) defined 5 types of idioms as given below:

1. Definitional/Synthesis Idiom models the definitional relations with input nodes that define the synthetic node.
2. Cause-Consequence Idiom models the cause nodes as input nodes and consequences of that cause nodes as output nodes.
3. Measurement Idiom models the reasons affecting the calculation precision and the estimated value of a measurement.
4. Induction Idiom models the statistical reasoning between nodes; for example, a parameter node can be estimated by observation nodes.
5. Reconciliation Idiom reconciles the results created by different methods as a single entity.

3.1.2. Learning BN Structure from Data

3.1.2.1. Constraint-based algorithms

BN structures can be learned with constraint-based algorithms according to their conditional dependencies. Firstly, these algorithms build a skeleton of BN according to the conditional independencies in the BN. Afterward, edge directions are defined. Here according to conditional dependencies, the converging connections are determined, then the remaining edges are joined so that they do not create a new converging relation, and finally, the remaining edges are joined so that they do not create a cycle (Spirtes et al. 2000).

3.1.2.2. Score-based algorithms

Score-based learning requires a score function for comparing BN structures and a search algorithm to construct a BN structure that has a better score. Score functions typically reward higher likelihood function values and penalize the complexity of the model to avoid overfitting. Firstly, a starting BN structure is determined. The score function for that BN structure is calculated and an edge operation (deletion, insertion or inversion of an edge) is made and the score function is calculated again. Edge operation continues until a stopping rule defined for the algorithm is satisfied.

3.1.3. Learning BN Structure by Hybrid Learning Methods

There are some studies about learning BN structures by hybrid learning methods; for example, studies that use both constraint-based algorithms and score-based algorithms. A hybrid learning algorithm can build a skeleton of BN by a constraint-based algorithm and then it can determine the directions of edges by a score based algorithm (Tsamardinos et al. 2006).

3.2. Learning Parameters of Bayesian Networks

3.2.1. Exponential Growth

The CPT in BN includes the probabilities of each state if a node has no parent. But if it has any parent it contains the probabilities of combination of its states with its parent nodes' states. Those parameters can be elicited from an expert or data or both.

In BNs, increasing the number of parent nodes of any node, gets the size of its CPT larger. For example, a node that has 5 states and one parent node which has 5 states has 20 parameters needed to be elicited to build its CPT. When one more parent node which again has 5 states is added to that node, CPT will have 100 parameters and so on. Similarly, increasing the number of states of that node or of its parents increases its CPT. A node that has 6 states and 2 parents with 6 states will have 180 parameters.

For a node that has 5 states, the number of parameters needed to be learned according to its number of parents which have the same number of states are shown in Table 3.1 given below.

Table 3.1 Exponential growth of CPTs according to number of parents

Parents	Number of Parameters
0	4
1	4x5 = 20
2	4x5x5 = 100
3	4x5x5x5 = 500
4	4x5x5x5x5 = 2500
5	4x5x5x5x5 = 12500

If a node with binary states has n parent nodes that have binary states, its CPT will have 2^n parameters. This is called the exponential growth of CPT.

3.2.2. Learning BN Parameters by Knowledge Engineering Methods

The step of learning parameters of CPTs is difficult because of the exponential growth of CPTs. In large BN structures, eliciting the parameters of CPT from experts is time-consuming and infeasible as the CPTs get large. Pearl (1988) suggested Canonical methods to reduce the parameters need for elicitation. One of these methods is Noisy-OR gate that represents n cause nodes' effects on one effect node. Noisy-OR gate requires the following conditions to be implemented:

- The probability of a cause node occurring in the absence of other cause nodes is known.
- The probability of an effect node occurring is known when no cause node has occurred.
- Occurring of a cause node is independent of occurring probability of other cause nodes.

If there are n cause nodes as x_i $(i=1, 2, \ldots, n)$ with occurring probabilities of p_i and a binary-valued effect node y, the probability of node Y occurring given a subset of X will be as in equation (2).

$$P(Y \mid X) = 1 - \prod_{i:xi \in X} (1 - pi) \tag{2}$$

The number of parameters needed for producing CPT reduces from 2^n parameters to n parameters in Noisy-OR approach.

The Noisy-OR method can be used to decrease the parameters elicited from an expert. There are some studies about learning Noisy-OR parameters from data. Oniśko et. al. (2001) used Noisy-OR gate to learn the parameters of CPT from data. The Noisy-OR method was tested with the HEPAR II model with a small clinical data set collected from patients for the diagnosis of liver disorders. The single-disorder model structured by using expert opinion and medical literature was developed by the assumption that the disorders were independent of each other. The multiple-disorder model was developed by applying some changes to the single-disorder model. While there was no data for half of the CPT parameters in the single-disorder model, the percentage of the parameters without data in the multi-disorder model decreased to 0.1%. The greatest diagnostic accuracy is achieved as 48% for multiple-disorder Noisy-OR CPT learned from the data, 46% for multiple-disorder Noisy-OR CPT learned from the specialist, and 45% for normal CPT. Finally, it was found that Noisy-OR multiple-disorder model parameters had 6.7% better diagnostic accuracy than normal multiple-disorder model parameters and 14.3% better diagnostic accuracy than single-disorder model Noisy-OR parameters.

Díez (1993) and Srinivas (1993) improved the Noisy-MAX method which is a general version of the Noisy-OR method for n-level nodes. Noisy-MAX methods allow the building of multi-leveled cause and effect nodes distinctly from the Noisy-OR gate.

Zagorecki & Druzdzel (2013) tested the Noisy-OR and Noisy-MAX methods used to solve the exponential increase of parameters of CPT by comparing them with existing CPT. They tested Noisy-OR, Noisy-MAX, and existing CPTs in 3 cases (ALARM, HAILFINDER, HEPAR 2). The fact that causal effects are independent in BN makes it possible to use the Noisy-OR and Noisy-MAX methods. In this article, the Noisy-OR method was not mentioned in detail, but the parameters and building of CPTs of the Noisy-MAX method was explained. The distance between the CPTs was measured by Euclidean distance and Kullback-Leibler (KL) distance in order to compare the CPTs produced by the Noisy-MAX method and the original CPTs. It has been proved that the Euclidean distance between the CPTs produced by Noisy-MAX and the original CPTs is only a minimum value; using this feature, it is concluded that the most appropriate Noisy-MAX distribution can be found for a given CPT. In their article, the Noisy-MAX algorithm was tested on 3 cases to test the compatibility of the CPTs. The average distance and maximum distance values of the CPT nodes produced with Noisy-MAX were measured separately with the original CPT nodes for cases with and without weighting of the parent nodes. In two of the three cases, the maximum distance was less than 0.1 for 50% of the nodes in between. This shows that the Noisy-MAX method provides a good fit for a significant percentage distribution. In order to see that the results were not coincidental, random CPTs were produced by the Noisy-OR method. It was observed that the CPTs found by Noisy-MAX method did not show good agreement with these CPTs. In addition, it was stated that goodness of fit is not related to the number of parameters of the CPT and that the Noisy-MAX method is as good as the small CPTs in the big CPT.

The ranked nodes method was proposed by Fenton et al. (2007) in order to reduce the number of parameters required to build CPTs. The ranked nodes method is another approximation to simplify CPTs of ordinal nodes by using an underlying bounded continuous scale. Ranked nodes can use a variety of functions for modelling the relationship between a node and its parents. Ranked nodes have been primarily used for parameter elicitation from experts. Detailed description of the ranked nodes method will be provided in Chapter 4.

3.2.3. Learning BN Parameters from Data

The parameters of BN can be learned from data. Let *M(G, Q)* be a BN model that has a graphical structure *G*, parameters *Q*, cluster of parameters *U* and data set *D*. The likelihood of *M* according to every data in the dataset *D* is defined by probability *P(d/M)*.

$$L(M \,/\, D) = \prod_{d \in D} P(d \mid M) \tag{3}$$

The log of likelihood which called loglikelihood of *M* is calculated.

$$LL(M \,/\, D) = \prod_{d \in D} \log\left(P(d \mid M)\right) \tag{4}$$

The maximum likelihood approach learns the parameters *Q* by maximizing the likelihood or loglikelihood function for a known structure *G* and dataset *D*. The likelihood method calculates the model that gives maximum likelihood according to data set.

$$Q = argmax_Q L\left(M_Q \,/\, D\right) = argmax_Q LL\left(M_Q \,/\, D\right) \tag{5}$$

To implement likelihood method, the parameters should be independent of each other and the uncertainty between different parameters should be independent. In the situation of missing data an expectation maximization algorithm (Steffen L. Lauritzen, 1995) can be used.

4. RANKED NODES METHOD

4.1. Overview

Fenton et al. (2007) proposed ranked nodes to simplify the parameter space of BN nodes by approximating their CPT by an underlying continuous variable. This enables probability distribution of the ranked nodes to be defined with fewer parameters.

Ranked nodes have ordinal states that are corresponding to a continuous numerical scale that is bounded between 0 and 1. The CPT of the ordinal states is generated by discretizing the continuous scale with equal intervals for each ordinal state. Users interact with the ordinal states when they enter evidence or calculate the BN, the underlying continuous scale is invisible to the user and only used for generating the CPTs when the BN is calculated.

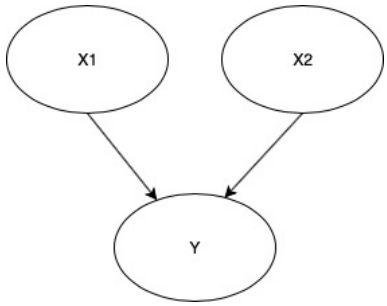


Figure 4.1 An example of BN structure which has ranked nodes

An example of BN which has ranked nodes is given in Figure 4.1. The nodes X_1, X_2 and Y respectively have 5, 3 and 5 states that can be measured on subjective ordinal scales. For example, the states of node X_1 are "Very Low" "Low" "Medium", "High" and "Very High". The underlying continuous scale of X_1 is divided into 5 equal intervals where the state "Very Low" is corresponding to the interval [0-0.2), the state "Low" is corresponding to [0.2-0.4) and so on. To calculate the CPT of child node Y the "central tendency" approach is used. The underlying continuous scale of Y is defined by a function of X_1 and X_2, and its CPT is calculated based on this function.

Fenton et al. (2007) used Truncated Normal (TNormal) distribution bounded between 0 and 1 to define the underlying continuous scale of ranked nodes. The Normal distribution that is truncated between [0, 1] is called TNormal distribution. It is defined by ignoring the probability mass function out of [0, 1] and normalizing the remaining Normal distribution and denoted as TNormal $(\mu, \sigma^2, 0, 1)$.

Since the ranked nodes have an underlying continuous scale that is TNormal distribution, less parameter require to model the CPTs. Parent nodes are defined by two parameters mean and variance and the child nodes are measured by weighted average function of its parent nodes.

The cause node is denoted as a central tendency of effect nodes so an analogy with regression is made such that effect of each parent node X_i $(i=1,2,\ldots,n)$ to the child node Y is written as the linear regression function. Here the contribution of each cause node is denoted by weights w_i and an ε is added as an error factor. Here the error factor is distributed normally with mean 0 and variance σ^2. The variance is calculated as the inverse of the sum of weights.

$$y_i = \sum_{i=1}^{n} w_i X_i + e \tag{6}$$

$$\sigma^2 = \frac{1}{\sum_{i=1}^{n} w_i} \tag{7}$$

Finally, the child node Y is normalized. The expected value of Y will be $\sum_{i=1}^{m} w_i X_i$ and the variance of Y will be $\frac{1}{\sum_{i=1}^{n} w_i}$.

$$P(Y|\underline{X}) = TNormal\left[\frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i}, \frac{1}{\sum_{i=1}^{n} w_i}, 0,1\right] \tag{8}$$

$$P(\underline{X}, Y) = P(Y \mid \underline{X}) \prod_{i=1}^{n} P(X_i) \tag{9}$$

Fenton et al. (2007) proposed four weighted functions to define the probability distribution of ranked nodes with parents. These are weighted mean *(WMEAN),* weighted maximum *(WMAX),* weighted minimum *(WMIN)* and a mixture of minimum and maximum *(MIXMINMAX)* functions as shown in (10), (11), (12), and (13).

$$WMEAN = \frac{\sum_{i=1}^{n} w_i X_i}{\sum_{i=1}^{n} w_i} \tag{10}$$

$$\underset{\forall i=1\ldots n}{\text{WMIN}} = \min \left[\frac{w_i X_i + \sum_{i \neq j}^{n} X_j}{w_i + (n-1)} \right] \; where \; w_i \geq 0 \tag{11}$$

$$\underset{\forall i=1\ldots n}{\text{WMAX}} = \max \left[\frac{w_i X_i + \sum_{i \neq j}^{n} X_j}{w_i + (n-1)} \right] \; where \; w_i \; \geq 0 \tag{12}$$

$$MIXMINMAX = \frac{w_{min} MIN(\underline{X}) + \; w_{max} MAX(\underline{X})}{w_{min} + \; w_{max}} \tag{13}$$

Here w_i and X_i are the weight and underlying numerical value of the parent i, respectively, and $\underline{X}$ is the set of all parents. Fenton et al. (2007) proposed ranked nodes primarily to improve parameter elicitation from domain experts. These functions and their interpretation are found to be suitable for expert elicitation as they can model a variety of shapes and their weights can be elicited in a relatively easier way than eliciting a full CPT.

Currently, ranked nodes are only implemented in the AgenaRisk BN software, the next section describes this implementation.

4.2. AgenaRisk Implementation of Ranked Nodes

The current AgenaRisk implementation of ranked nodes focuses on eliciting the parameters of ranked nodes (mean and variance for parent nodes and mean, variance and weights for child nodes) from domain experts. In other words, no approach is provided to learn ranked nodes from data. The CPTs of variables without parents are defined based on the TNormal distributions of those nodes. Figure 4.2 shows the TNormal distribution of ranked nodes given in Figure 4.1. The mean of node X_1 is 0.8 and the variance of X_1 is 0.05. The mean of node X_2 is 0.3 and the variance of X_2 is 0.1. AgenaRisk calculates the CDF of that TNormal distributions and discretizes these CDFs by using the associated intervals of node X_1 and X_2. The resulting CPTs are given in Figure 4.3.

For child nodes, the combination of samples is used to create CPTs. Suppose that node Y has m parent nodes $X_1, X_2, \ldots, Xm$. AgenaRisk generates s number of samples for each interval corresponding to states of parent nodes. Then it uses those samples to calculate the expected value of node Y. For example, for the part of the CPT regarding $P(Y \mid X_1 = x_1, \ldots, X_m = x_m)$ the algorithm generates s number of samples from the intervals that correspond to states $x_1, x_2, \ldots, x_m$. Then the expected value of node Y is created by implementing the weighted function to that s^m combinations of samples. It then calculates the CDF of the TNormal distribution for all s^m expected values, discretizes these CDFs by using the associated intervals of Y, and averages these discretized probabilities (details of this approach is shown by Fenton et al., (2007)). In Figure 4.3 the CPT of Y is calculated by implementing a weighted function $\mu_y = 2X_1 + X_2$ and $\sigma_y^2 = 0.01$.

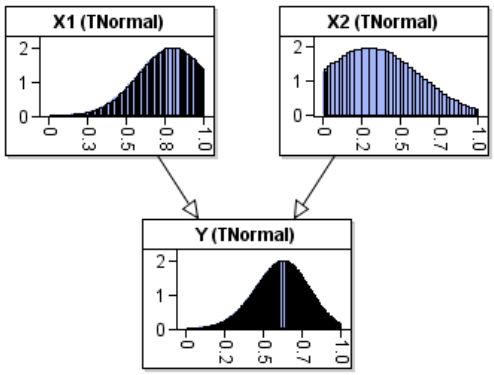


Figure 4.2 TNormal distributions of ranked nodes

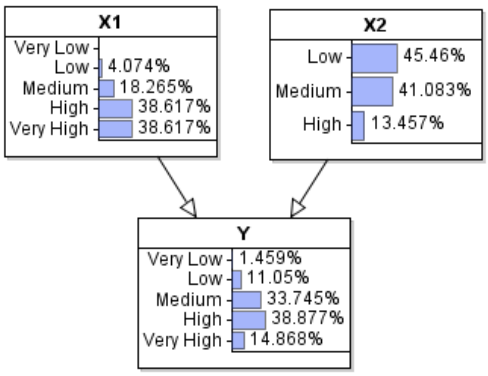


Figure 4.3 CPTs of Ranked Nodes

Rather than using s^m combinations of equidistant samples, Nunes et al. (2017) generated n random samples for the regression model for each state combination of the parents and generates a CPT based on these samples. Nunes et al. (2017) showed that their results and CPT generation times are close to the AgenaRisk implementation.

Virtanen (2018) mentioned the ranked nodes method in their paper and then they mentioned the lack of studies made about generating CPTs with RNM. The lack of knowledge about practical usage of ranked nodes method led them to make some experiments about the application of ranked nodes method. They studied the effect of sample size s on CPTs of ranked nodes and found that as the sample size increases, the elements of CPT generated with ranked nodes method converge toward probabilities obtained from the regression model. But unfortunately, the computational load increases also as the sample size increases. They studied some other factors i.e. sample size parameter's value, the feasible range of the weights, etc. They showed that the use of partitioned weight expressions and the elementary RNM-compatibility of nodes are the factors that affect the accuracy of fit the most.

Laitila & Virtanen (2016) mentioned the steps of ranked nodes method required to build the CPTs of BN with interval scales. But some challenges may be encountered when applying it to nodes with interval scales. They mentioned the difficulty of representing expert judgments on ranked nodes that have interval scales. They proposed an effective

approach which represents expert judgments in those situations. The first step of the three steps of that approach is the discretization of interval scales iteratively up to most preferable discretization. Then weight expressions and weights of ranked nodes are determined, and finally, the CPTs are refined.

Fenton et al. (2019) pointed to a problem encountered because of the conditional Inter-Causal Independence (CII) property of leaky Noisy-OR. If the cause variables are defined by $x_1, \ldots, x_n$ and effect variable is defined by y, the CII property of Noisy-OR causes the failure of explaining away behavior of BN when the effect variable of Noisy-OR is false $(y=0)$. In this situation, the cause variables will be independent of each other and the explaining away behavior of Noisy-OR will fail. Fenton et al. (2019) tried to solve that deficiency of explaining away behavior in Noisy-OR by eliminating the CII property of it. They provided the conditional dependence between cause nodes by taking into consideration the three forms of causal inference which are synergy, interference, and inhibition. They built an extension of Noisy-OR by adding an explaining away parameter to the Noisy-OR function. After proofing the extended version of Noisy-Or they pointed out the importance of choosing the optimal explaining away parameter to avoid the undesired properties that can occur.

Noguchi et al. (2019) made another study about the explaining away problem of Noisy-OR when the effect node is observed as false. In that study, they stated the similarity between Noisy-OR and ranked nodes as both of those approaches contain several effect variables that lead to a cause variable. One difference is the Noisy-OR has binary states when the ranked nodes can have multiple and ordered states. They propose the use of ranked nodes instead of Noisy-OR because of the anti-correlation property of ranked nodes when the causes are conditioned on effect. That property means that the states of two effect nodes of a cause node are not likely to be the same. So, the Noisy-OR problems can be modeled by ranked nodes by transforming its states to ordinal states and using the weighted sum function. At the same time, this enables the child to have multiple states whereas Noisy-OR variables can have two states.

Despite these capabilities, the use of ranked nodes for data-driven parameter learning has not been investigated in the reviewed studies. In the following section, we present a method to learn ranked nodes CPTs from data.

5. LEARNING RANKED NODES PARAMETERS FROM DATA

Ranked nodes have an underlying TNormal distribution and the underlying continuous distribution of ranked nodes with parents are defined by a TNormal regression model where parents are independent variables and the child is the dependent variable as defined in the previous chapter. In this study, our process aims to learn the CPTs of ranked nodes from data by several algorithms.

The algorithm is able to work with complete and ordinal, and known BN structures (Figure 5.1). We implemented the algorithm in the R environment. The implementation takes the BN structure and ordinal data as input and returns CPTs of ranked nodes as outputs.

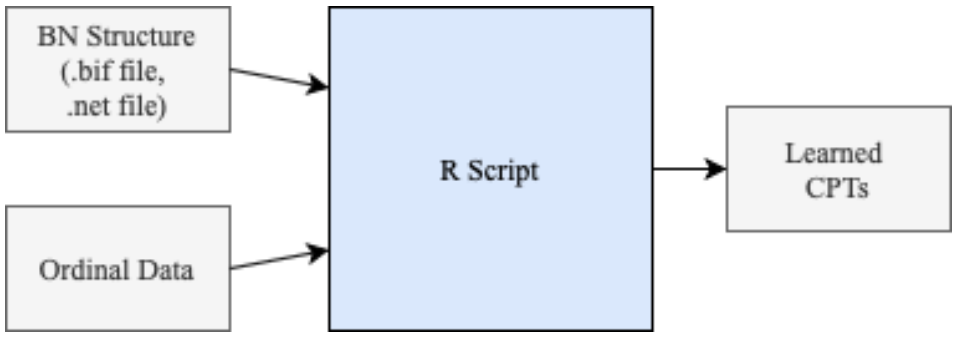


Figure 5.1 The inputs and outputs of the proposed Regression algorithms

In the remainder of this section, the transformation of the data step of the algorithm is given in Section 5.1. Then the approach to learn the nodes with no parent nodes is given in Section 5.2, and the methods to learn the parameters of nodes with parents are given in Section 5.3.

5.1. Transforming the Categorical Data to Numerical Data

Our algorithm uses regression to estimate the parameters of the ranked nodes. Therefore, firstly the categorical data of ordinal variables are transformed into numerical data by using the median point of the associated interval. For example, Table 5.1 shows an ordinal

variable with five states. Ranked nodes map each state to intervals with equal width in its underlying continuous scale. We use the median point of the associated intervals when we transform the categorical data to numerical data for our approach.

Table 5.1 The states and corresponding intervals of Ranked Nodes

State	Associated Intervals	Median Value
Very Low	0.0 – 0.2	0.1
Low	0.2 – 0.4	0.3
Medium	0.4 – 0.6	0.5
High	0.6 – 0.8	0.7
Very High	0.8 – 1.0	0.9

After transforming the data, the parameters of TNormal distribution is estimated for the nodes without parents and a TNormal regression model is set for the nodes with parents. Then, a sampling-based approach is run to generate CPTs based on that regression model.

The categorical data and BN structure are inserted to the R code firstly, and the R code transforms the categorical data to numerical data by calculating the median of the intervals. A BN that has ranked nodes X_1, X_2, X_3 and Y is given in Table 5.2. Nodes X_1, X_2 and X_3 are parents of node Y and they have the data shown below. Their corresponding numerical data is given in Table 5.3.

Table 5.2 An example of ranked nodes categorical data

X1	X2	X3	Y
Low	Very Low	Medium	Low
High	High	High	High
Low	Very Low	Medium	Medium
Low	Medium	Medium	Medium
Very Low	Very Low	Medium	Very Low

Table 5.3 Corresponding numeric data of categorical data in Table 5.2

X1	X2	X3	Y
0.3	0.1	0.5	0.3
0.7	0.7	0.7	0.7
0.3	0.1	0.5	0.5
0.3	0.5	0.5	0.5
0.1	0.1	0.5	0.1

5.2. Nodes with No Parent Nodes

In a BN, marginal distribution of the associated random variable needs to be defined for nodes that have no parents. CPTs of those nodes are created by calculating the mean and variance of the underlying distribution based on data. After transforming categorical data to numeric data, its mean and standard deviation are calculated. Since the underlying distribution used for ranked nodes is the TNormal distribution, the mean and variance of TNormal distribution are calculated in the following section.

5.2.1. Deriving TNormal Distribution from Normal Distribution

The TNormal(μ, σ, a, b) distribution is defined by specifying parameters of general normal distribution μ and σ, then a truncation range [a, b]. Its probability density function (PDF) is represented by $\psi(\mu, \sigma, a, b; x)$ and its cumulative distribution function (CDF) is represented by $\psi(\mu, \sigma, a, b; x)$. The TNormal PDF is defined by modifying the Normal PDF $\phi(\mu, \sigma; x)$ for the truncation interval where $\Phi(\mu, \sigma; x)$ is the CDF of the Normal distribution. The values outside the truncation range are set to zero, and the values inside are uniformly scaled.

$$\psi(\mu, \sigma, a, b; x) = \begin{cases} 0 & if\ x < a \\ \dfrac{\phi(\mu, \sigma^2; x)}{\Phi(\mu, \sigma^2; b) - \Phi(\mu, \sigma^2; a)} & a \le x \le b \\ 0 & if\ x > b \end{cases} \tag{14}$$

The mean $\overline{\mu}$ and variance $\overline{\sigma}^2$ of the TNormal distribution can be determined as in equations (15), (16) and (17).

$$\alpha = \frac{a-\mu}{\sigma}$$
$$\beta = \frac{b-\mu}{\sigma} \tag{15}$$

$$\bar{\mu} = \mu - \sigma \frac{\phi(0,1,\beta) - \phi(0,1,\alpha)}{\Phi(0,1,\beta) - \Phi(0,1,\alpha)} \tag{16}$$

$$\bar{\sigma}^2 = \sigma^2 \left(1 - \frac{\beta\phi(0,1,\beta) - \alpha\phi(0,1,\alpha)}{\Phi(0,1,\beta) - \Phi(0,1,\alpha)} - \left(\frac{\phi(0,1,\beta) - \phi(0,1,\alpha)}{\Phi(0,1,\beta) - \Phi(0,1,\alpha)}\right)^2\right) \tag{17}$$

The CPT of ranked nodes that have no parent nodes is built by calculating the CDF of TNormal distribution of each state interval. Suppose that a random variable X_i which has no parent nodes has n ($n=0,1,2, \ldots, n$) number of states. Let x_i, which denotes the state of variable X_i, has $[l_i, u_i]$ state interval (l_i corresponds to lower bound and u_i correspond to upper bound of state x_i). The probability of state i of node X_i is determined by the CDF of TNormal distribution that lies between the intervals of $[l_i, u_i]$.

$$P(X_i = x_i) = \psi(\bar{\mu}_x, \bar{\sigma}_x, a, b; xi) = \frac{\Phi(\bar{\mu}, \bar{\sigma}; ui) - \Phi(\bar{\mu}, \bar{\sigma}; li)}{\Phi(\bar{\mu}, \bar{\sigma}; 1) - \Phi(\bar{\mu}, \bar{\sigma}; 0)} \tag{18}$$

Then the CPT is built by placing these probabilities to each state interval as in Table 5.4.

Table 5.4 CPT of nodes with no parent nodes

X1	**X2**	**…**	**Xn**
$P(X_i = x_1)$	$P(X_i = x_2)$	…	$P(X_i = x_n)$

5.3. Nodes with Parent Nodes

The conditional probability distribution of nodes with parents needs to be defined in a BN. In this section we presented TNormal, Beta and Linear regression methods for learning the conditional probability distribution of ranked nodes from data.

5.3.1. TNormal Regression Method

The algorithm improved to apply TNormal Regression method to ranked nodes is summarized below.

Transform Ordinal Data to Numerical Data

for *each node in the BN*

if *the node has no parents*

Estimate the parameters of the TNormal distribution.

Discretize the TNormal distribution with equal intervals for each state of the node.

else

Apply TNormal regression using gradient descent.

for *each state combination of the parent nodes*

Generate n uniform samples for the TNormal regression for the parent combination.

Count the samples for each child interval to define the CPT of the parent combination.

end for

end if

end for

TNormal regression model defines the dependent node Y with a TNormal distribution where μ is defined by a linear regression function equation (19) where X_i and w_i are respectively the independent variable i and its coefficient, w_0 is the constant term and m is the number of independent variables.

$$Y = w_0 + \sum_{i=1}^{m} w_i X_i \tag{19}$$

Suppose that, A random variable Y has *n* parent nodes $X_1, X_2, \ldots, X_n$. The states of random variable X_i are denoted by x_i and the states of Y are denoted by y_i for child node. So, the state intervals are denoted by $[l_i, u_i]$. The probabilities of state i of node Y given $X_1, X_2, \ldots, X_n$ is calculated by sampling approach using TNormal regression parameters.

Firstly, categorical data are transformed into numerical data. Then a regression is applied to numerical data and the weights of that regression are determined by gradient descent algorithm. Finally, *n* number of samples for each state interval of each parent node are generated, and for different combinations of that state intervals the child node is calculated by putting these weights and samples into the regression equation by the optimal weights calculated. The number of samples falling to each state interval of the child node is counted. The result is divided by the total number of samples generated. The result is written to CPT as the probability of that state interval of the child node. That approach is mentioned in detail in next sections.

5.3.1.1. Determining TNormal Regression Coefficients by Gradient Descent

A linear regression method is used to build CPTs of child nodes. In that method, the linear regression coefficients of numerical data are calculated by the help of the gradient descent algorithm.

Gradient descent algorithm is an iterative optimization algorithm. It can be used as a supervised learning algorithm by minimizing the error of a predictive model. If x is a feature or input variable and y is a target or output variable in the training set the hypothesis function h can be created as a linear regression function maps from x's to y's. So, the hypothesis or linear regression function tries to estimate the real values of the output variable y.

$$h_w(x) = w_0 + w_1 x \tag{20}$$

The purpose is to find the best regression coefficients *w*'s that predict the output values (*y*'s). To do that, a minimization approach is used such as minimizing the squared difference between the value predicted by the regression function and the real value. If the size of the training set is *m*, the sum of mean squared errors of the training examples *i*=1 to *m* is tried to be minimized.

$$minimize_{w_0 w_1} \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2 \tag{21}$$

The cost function is written as the square difference between the value predicted by the regression function and the actual value in the minimization approach. The gradient descent algorithm aims to minimize the cost function that depends on the regression coefficient to provide the perfect regression function that fits the data. Note that gradient descent is used for finding the optimum of different types of cost functions, not only the linear regression function.

$$J(w_0, w_1) = \sum_{i=1}^{m} (h_w(x^{(i)}) - y^{(i)})^2 \tag{22}$$

$$J(w_0, w_1) = \sum_{i=1}^{m} ((w_0 + w_1(x^{(i)})) - y^{(i)})^2 \tag{23}$$

$$minimize_{w_0 w_1} J(w_0, w_1) \tag{24}$$

The algorithm is started with initial values to try to find the minimum of that cost function by changing the unknown parameters iteratively with a learning rate. The algorithm decreases the value of the cost function by moving it to the decreasing direction. So, the gradient of the cost function is calculated, and the parameters are updated by learning rate

times that gradient. It is important to update all parameters simultaneously at each iteration.

$$w_0 =: w_0 - \alpha \frac{\partial}{\partial w_0} J(w_0, w_1)$$
$$w_1 =: w_1 - \alpha \frac{\partial}{\partial w_1} J(w_0, w_1) \tag{25}$$

The algorithm repeatedly updates all parameters until it finds the local or global optimum or reaches a stopping rule such as the value of cost function being less than a small number as in (26).

$$Repeat\ until\ convergence\ \{w_j =: w_j - \alpha \frac{\partial}{\partial w_j} J(w_0, w_1, \dots, w_n)\ for\ j = 0,1, \dots, n\} \tag{26}$$

When Normal regression function estimates the dependent or child variable by equation $Y = w_0 + \sum_{i=1}^{m} w_i X_i$, TNormal regression function estimates the dependent variable as in Equation (27). According to the estimation of the dependent or child variable *Y*, the cost function is given in (28).

$$h_w(x) = w_0 + \sum_{i=1}^{m} w_i X_i - \sigma \frac{\phi(0,1,\beta) - \phi(0,1,\alpha)}{\Phi(0,1,\beta) - \Phi(0,1,\alpha)} \tag{27}$$

$$J(w_0, \dots, w_m, \sigma) = Y - \left(w_0 + \sum_{i=1}^{m} w_i X_i - \sigma \frac{\phi(0,1,\beta) - \phi(0,1,\alpha)}{\Phi(0,1,\beta) - \Phi(0,1,\alpha)} \right) \tag{28}$$

5.3.1.2. Applying Gradient Descent to Matrix Notation of Cost Function

The regression function can estimate the dependent variable based on more than one independent variable. That case of multiple variables is called multivariate regression. The multiplication rule of matrix simplifies the notation of regression. By taking advantage of matrix multiplication rule, the hypothesis can be written as equation (29) and the continuing formulas are represented by matrix notation. In addition to hypothesis function, the cost function is represented by matrix and the gradient descent is applied to that cost function by taking derivative of that cost function according to parameters and variance. The data have m number of training examples and n features. The parent variables are an $m \times n$ dimensional matrix $\boldsymbol{X}$ and child variable $\boldsymbol{w}$ is a vector which is n dimensional. The hypothesis is given in formula (29). From now on, the multiplication of the parent matrix and child vector will be used in the cost function.

$$h_w(x) = [w_0 \quad w_1 \quad \dots \quad w_n] \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix} = \boldsymbol{w}^T \boldsymbol{X} \tag{29}$$

$$h_w(x) = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_n^{(2)} \\ \dots & \dots & \dots & \dots \\ x_1^{(m)} & x_2^{(m)} & \dots & x_n^{(m)} \end{bmatrix} \times \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_n \end{bmatrix} = \boldsymbol{X}\boldsymbol{w} \tag{30}$$

$$J(w, \sigma) = y - \left(\mu - \sigma \left(\frac{\varphi(\mu, \sigma^2, b) - \varphi(\mu, \sigma^2, a)}{\phi(\mu, \sigma^2, b) - \phi(\mu, \sigma^2, a)} \right) \right) \tag{31}$$

$$J(w, \sigma) = \mathrm{y} - \left(\boldsymbol{X}\boldsymbol{w} - \sigma \left(\frac{\varphi\left(\frac{1 - \boldsymbol{X}\boldsymbol{w}}{\sigma}\right) - \varphi\left(\frac{0 - \boldsymbol{X}\boldsymbol{w}}{\sigma}\right)}{\phi\left(\frac{1 - \boldsymbol{X}\boldsymbol{w}}{\sigma}\right) - \phi\left(\frac{0 - \boldsymbol{X}\boldsymbol{w}}{\sigma}\right)} \right) \right) \tag{32}$$

A gradient descent algorithm is applied to that cost function to find the best parameters that fit the regression formula. The partial derivative of the cost function according to parameter vector $\boldsymbol{w}$ and the standard deviation σ is calculated. The parameter vector $\boldsymbol{w}$

and the standard deviation σ are updated up to find the parameters that made the derivative of cost function smaller than a defined threshold value (stopping rule). The partial derivative of cost function according to $\boldsymbol{w}$ is given in equation (35) and according to standard deviation is given in (36).

$$Repeat\ until\ convergence\ \{w =: w - \ \alpha \frac{\partial}{\partial w} J(w, \sigma)\} \tag{33}$$

$$Repeat\ until\ convergence\ \{\sigma =: \sigma - \ \alpha \frac{\partial}{\partial \sigma} J(w, \sigma)\} \tag{34}$$

$$A = \frac{1 - \boldsymbol{Xw}}{\sigma}$$

$$B = \frac{0 - \boldsymbol{Xw}}{\sigma}$$

$$\frac{d}{dw} J = \left(\frac{-X(\varphi(\mathrm{A})\mathrm{A}) - (\varphi(\mathrm{B})\mathrm{B}) - \frac{\big(\varphi(\mathrm{A}) - \varphi(\mathrm{B})\big)\big(\varphi(\mathrm{A}) - \varphi(\mathrm{B})\big)}{\phi(\mathrm{A}) - \phi(\mathrm{B})}}{\big(\phi(\mathrm{A}) - \phi(\mathrm{B})\big)} + 1 \right) \left(y - \left(\sigma \frac{\varphi(\mathrm{A}) - \varphi(\mathrm{B})}{\phi(\mathrm{A}) - \phi(\mathrm{B})} + \boldsymbol{Xw} \right) \right) \tag{35}$$

$$\frac{d}{d\sigma} J = \left(\frac{\Big(\big(\varphi(\mathrm{A})\ (1 - \boldsymbol{Xw})\big) - \big(\varphi(\mathrm{B})\ (0 - \boldsymbol{Xw})\big)\Big)\big(\varphi(\mathrm{A}) - \varphi(\mathrm{B})\big)}{\big(\phi(\mathrm{A}) - \phi(\mathrm{B})\big)} + \left(\frac{(\varphi(\mathrm{A})\ (1 - \boldsymbol{Xw})^2) - (\varphi(\mathrm{B})\ (\boldsymbol{Xw})^2)}{\sigma} \right)}{\sigma} + \varphi(\mathrm{A}) - \varphi(B) \right) \left(y - \left(\sigma \left(\frac{\big(\varphi(\mathrm{A}) - \varphi(\mathrm{B})\big)}{\big(\phi(\mathrm{A}) - \phi(\mathrm{B})\big)} + (-\boldsymbol{Xw}) \right) \right) \right) \tag{36}$$

Then the optimal parameters of TNormal regression founded by the gradient descent algorithm are used to generate CPTs of ranked nodes.

5.3.2. Beta Regression Method

Fenton et al. (2007) described the dependent variable in ranked nodes method with TNormal distribution rather than Normal distribution because the dependent variable in ranked nodes is distributed in the [0,1] interval. Beta regression is also a suitable underlying distribution for ranked nodes as they are bounded between the [0,1] interval.

The Beta probability density function (PDF) of dependent variable y is represented by $\pi(y; p, q)$ and it is given in (37) where $p > 0$ and $q > 0$ are parameters that provide different shapes to distribution and $\Gamma(.)$ is the gamma function.

$$\pi(y; p, q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} y^{p-1}(1-y)^{q-1}, \qquad 0 < y < 1 \tag{37}$$

The mean μ and variance σ^2 of the Beta distribution are given in equations (38) and (39).

$$\mu = \frac{p}{(p+q)} \tag{38}$$

$$\sigma^2 = \frac{pq}{(p+q)^2(p+q+1)} \tag{39}$$

It is possible to define the Beta distribution with the mean (μ) and precision parameter (ϕ). If the mean of distribution is presented by $\mu = p/(p+q)$ and precision parameter is presented by $\phi = p + q$, then the parameters can be replaced by $p = \mu\phi$ and $q = (1 - \mu)\phi$ and PDF of Beta distribution can be as (40) where $0 < \mu < 1$ and $\phi > 0$.

$$\pi(y;\mu,\phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \qquad 0 < y < 1 \tag{40}$$

If it is assumed that variables $y_1, \ldots, y_n$ are independent variables which have Beta distribution and each y_t $(t=1,\ldots,n)$ is dependent on k variables $x_{t1}, \ldots, x_{tk}$, the mean of y_t is μ_t and defined by a link function $g(.)$ where $\beta = (\beta_1, \ldots, \beta_k)$ is a vector of regression parameters.

$$g(\mu_t) = \sum_{i=1}^{k} x_{ti}\beta_i = n_t \tag{41}$$

Some of link functions can be used for defining the mean of the dependent variable. If the logit specification function $g(\mu) = \log\{\mu/(1-\mu)\}$ is used the mean will be as in (41) where $x_t^T = (x_{t1}, \ldots, x_{tk})$ for $t= 1,\ldots,n$.

$$\mu_t = \frac{e^{x_t{}^T\beta}}{1 + e^{x_t{}^T\beta}} \tag{42}$$

To provide the positiveness of precision parameter another logit function $g(\phi) = \log\{-w_i\delta\}$ can be used.

These formulas will be useful when the Beta regression is used to learn the parameters of CPTs. After drawing a Beta regression between the parents and child node data, the Beta regression parameters $\beta = (\beta_1, \ldots, \beta_k)$ and precision parameters are found by using the maximum likelihood approach implemented in the betaReg package of R. Then by the logit link function the mean and precision of Beta regression are calculated. Then using these mean and precision parameters the p and q shape parameters can be calculated by $p = \mu\phi$ and $q = (1 - \mu)\phi$ and these shape parameters are used for generating Beta distributed samples for our dependent variable y. The steps of Beta regression algorithm are given below.

Transform Ordinal Data to Numerical Data

for *each node in the BN*

if *the node has no parents*

Estimate the parameters of the TNormal distribution.

Discretize the TNormal distribution with equal intervals for each state of the node.

else

Apply Beta regression.

for *each state combination of the parent nodes*

Generate n uniform samples for each state combination of each parent node.

Insert the n uniform samples to Beta regression equation.

Apply inverse link function to Beta regression outputs and find the mean and precision parameters of Beta distribution.

Calculate the shape parameters by using mean and precision parameters.

Generate n Beta distributed samples by shape parameters.

Count the samples for each child interval to define the CPT of the variable.

end for

end if

end for

5.3.3. Linear Regression Method

Ranked Nodes are defined in truncated interval, so the linear regression method is not appropriate for them. But, in order to make a comparison between the TNormal and Beta regression methods, the linear regression method is added to R code.

The main difference between linear regression algorithm and TNormal regression is that, in TNormal regression the Normal distribution of ranked nodes are truncated. But basically, in linear regression method, a linear regression is applied to data and the coefficients are inserted to linear regression equation (19) without any truncation. The outputs are counted for each state of child node and the CPT is built.

5.4. Sampling and Building CPTs

Once the regression models for nodes with parents are defined by using the approaches described in Section 5.3, we built discrete CPTs based on these for the BN. We followed a similar approach with Nunes et al. (2017) to generate the CPTs of the nodes with parents. Once we define the TNormal regression models for these nodes, we generated *n* samples for each state combination of the parents $X_1 = x_1$, $X_2 = x_2$, …, $X_m = x_m$, and use these samples on the TNormal regression model. We counted the number of samples associated with each interval of *Y* to compute the $P(Y \mid X_1 = x_1, \ldots, X_m = x_m)$.In order to generate the CPT of a node, we generated *n* samples for each combination of state interval of parent nodes from the regression model associated with that node. Afterward, we founded the ratio of samples associated with each state interval of the child node to calculate the conditional probability of that state.

TNormal regression method is applied to the BN in Figure 4.1 which has 2 parent nodes with 3 states each. The TNormal regression parameters of the BN is obtained. For each combination of state interval of parent nodes, *n* samples are generated through gradient descent algorithm. For variable *Y*, the weighted function is found as $Y = 0.13 + 0.44X_1 - 0.14X_2$. The *n* random samples are inserted to that equation and for each combination of parent nodes, n outputs are reached.

For example, when X_1=*Low*, 100 samples are generated between 0 and 0.33. Likewise, for X_2=*Low*, 100 samples are generated between 0 and 0.33. These samples are inserted to the regression formula and *n* outputs are generated for variable *Y*. Because variable *Y* has 3 states, the outputs that are between 0 and 0.33 are counted for *Y*=*Low*, the outputs

that are between 0.33 and 0.66 are counted for state *Y=Medium* and the remaining outputs are counted for state *Y=High* as given in Table 5.5. These numbers of outputs are divided to *n* (the total number of samples) and the resulting probabilities are equal to CPT of variable *Y* in Table 5.6.

Table 5.5 The number of outputs falls between each state interval of child node *Y*

	X1	**0 – .33**	**0 - .33**	**0 – .33**	**.33 – .67**	**.33 – .67**	**.33 – .67**	**.67 - 1**	**.67 - 1**	**.67 - 1**
	X2	**0 – .33**	**.33 - .67**	**.67 - 1**	**0 - .67**	**.33 - .67**	**.67 - 1**	**0 – .33**	**.33 –.67**	**.67 - 1**
Y	**0 - .33**	97	95	96	48	67	80	2	16	31
	.33 - .66	3	5	4	52	33	20	93	82	69
	.66 – 1	0	0	0	0	0	0	5	2	0

Table 5.6 The resulting CPT of child node *Y*

	X1	**LOW**	**LOW**	**LOW**	**MEDIUM**	**MEDIUM**	**MEDIUM**	**HIGH**	**HIGH**	**HIGH**
	X2	**LOW**	**MEDIUM**	**HIGH**	**LOW**	**MEDIUM**	**HIGH**	**LOW**	**MEDIUM**	**HIGH**
Y	**LOW**	0.97	0.95	0.96	0.48	0.67	0.8	0.02	0.16	0.31
	MEDIUM	0.03	0.05	0.04	0.52	0.33	0.2	0.93	0.82	0.69
	HIGH	0	0	0	0	0	0	0.05	0.02	0

6. EXPERIMENTS

The proposed ranked nodes algorithms described in the previous chapter are tested with different BN models. The R implementation has 3 alternative algorithms to learn the ranked nodes. The algorithms started to work by inserting a BN structure and BN data into the model. Then they converted categorical data to numeric data and ask what the learning algorithm will be used for. The BN models are learned by these different algorithms as flowchart given in Figure 6.1.

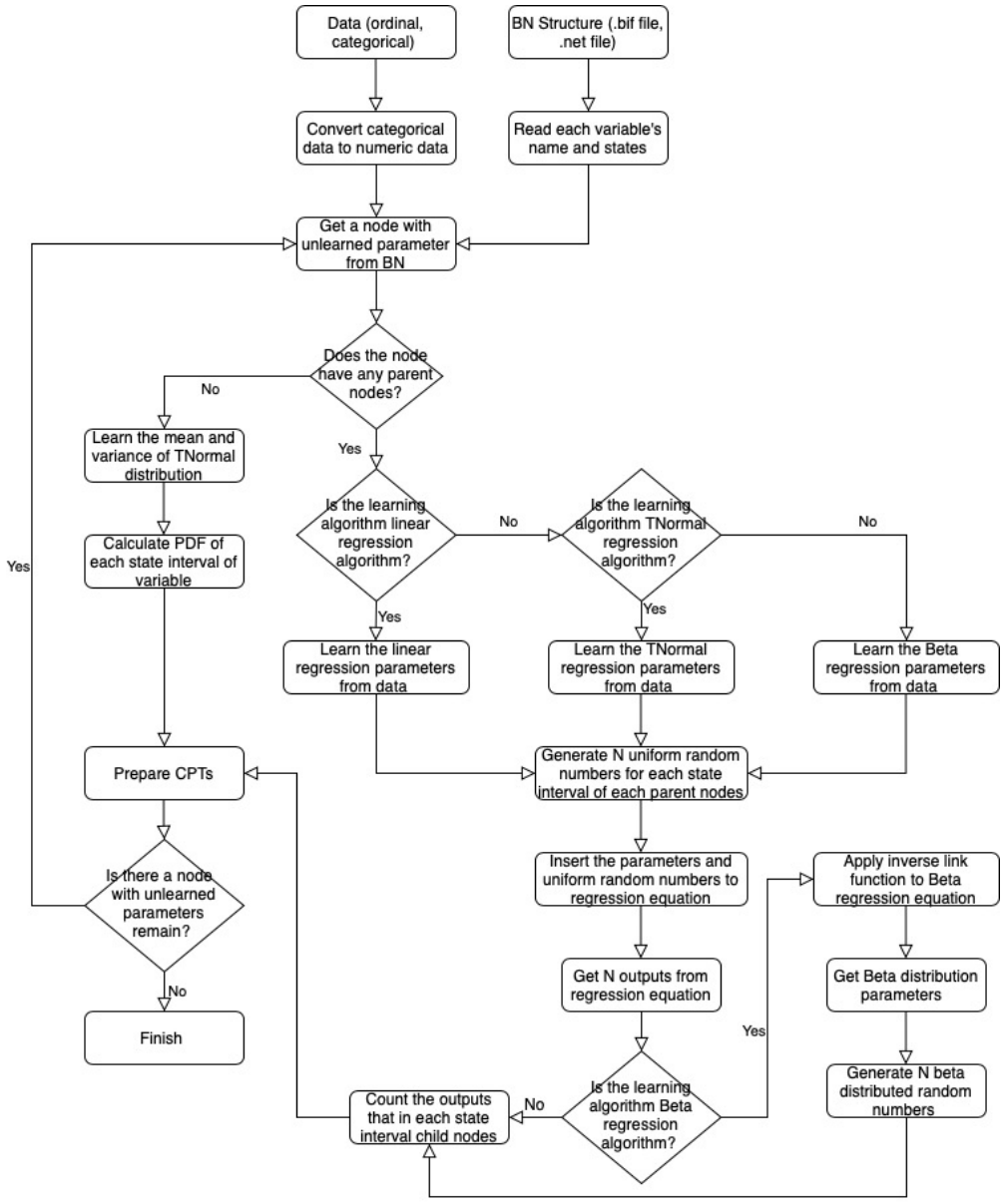


Figure 6.1 The algorithm of R code

For different algorithms of the developed R code, the models are learned, and the results are given in the next section.

6.1. The Results of Experiments

The TNormal regression algorithm, Beta regression algorithm and linear regression algorithm are tested in R with several BN models that are available in BN repository (bnlearn.com). The bnlearn package is the most widely used R package for learning the BN structure and parameters. It comprises constraint-based, data-based and hybrid structure learning algorithms. Also, it comprises maximum likelihood estimation (MLE) and Bayesian parameter learning functions which are used for comparing the proposed regression algorithms developed in this study. Note that there is no data-based ranked learning algorithm available for ranked nodes, and because of that the current MLE algorithm used for discrete data is used in this study for comparison.

Bn.fit is one of the functions of bnlearn package that learn the BN parameters if the structure and the data of BN are inserted to R. It uses MLE method to fit the parameters of BN. We compared the 3 proposed algorithms with MLE method implemented in the bnlearn package. The distance between each parameter of CPTs learned by MLE and true CPTs are calculated. The same calculation for CPTs learned by the proposed algorithms (TNormal regression algorithm, Beta regression algorithm and linear regression algorithm) and true CPTs are made. We used Hellinger distance to assess the similarity of the learned and true probability distributions. Other scores and distance measures such as Bhattacharyya distance or Kullback-Leibler divergence can also be used. Hellinger distance is closely related with the Bhattacharyya distance and it offers a symmetric distance measure unlike Kullback-Leibler divergence. The Hellinger distance decrease as the similarity between the distributions increase, hence lower scores are favorable. The Hellinger distance to assess the similarity of learned and true models are used for this calculation. The Hellinger distance of two discrete probability distributions $P = \{p_1, \ldots p_k\}$ and $Q = \{q_i, \ldots, q_k\}$ can be calculated as follows:

$$H(P,Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}\left(\sqrt{p_i} - \sqrt{q_i}\right)^2} \tag{43}$$

The comparison of the performance of the TNormal Regression, Beta regression and linear regression learning with MLE was made by using the BN models from the BN repository. These models are selected because 1) most nodes in these BNs have ordinal states, 2) their sizes and number of parents differ, 3) they are widely used for evaluating performance. Table 6.1 includes the size and complexity of these BNs. The experiments learned the BNs by using datasets with 50, 100, 200, 300, 500, 1000, 5000, 10000, 20000 and 50000 sample sizes for each of these BNs.

Table 6.1 Properties of BN Models

BN	**Nodes**	**Arcs**	**Parameters**	**Max. Number of Parents**
Child	20	25	230	2
Insurance	27	52	984	3
Mildew	35	46	540150	3
Barley	48	84	114005	4

The steps of the experiments are summarized below:

1. The 50, 100, 200, 300, 500, 1000, 5000, 10000, 20000 and 50000 sized samples are taken randomly from data.
2. The data and structure of these models are inserted into the R code. The models are learned with MLE method and proposed regression algorithms. Then by calculating Hellinger distances between the true CPTs and learned CPTs by MLE, TNormal regression, Beta regression and Linear regression, the comparisons are made.
3. For all sample sizes selected randomly, the BN CPTs are generated 10 times and the mean distance of these 10 models' CPTs are calculated. Finally, the mean of these distances is shown as the final distance between true CPTs and learned CPTs. The standard deviation and 95% credible intervals of these distances are shown in the Appendix.

6.1.1. Child Model

The Child Model given in Figure 6.2 is the simplest BN model used. It has 20 nodes; 230 parameters and its maximum number of parents is 2.

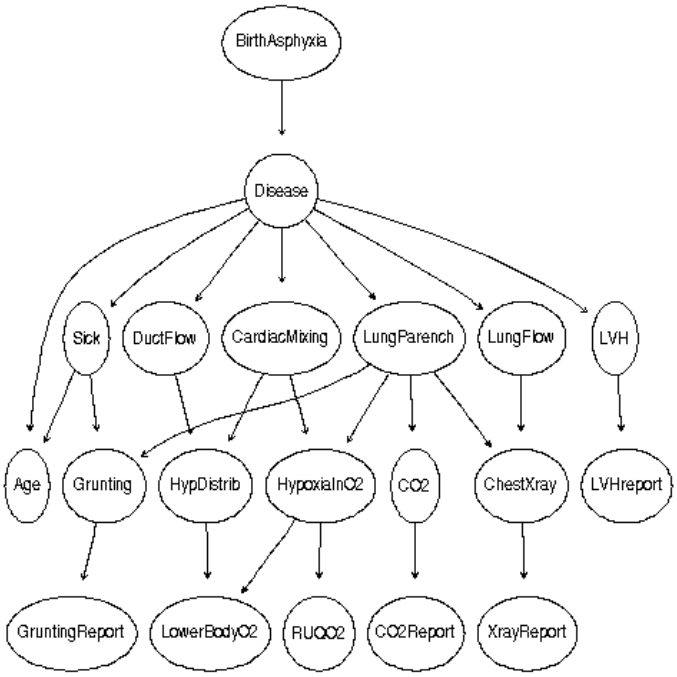


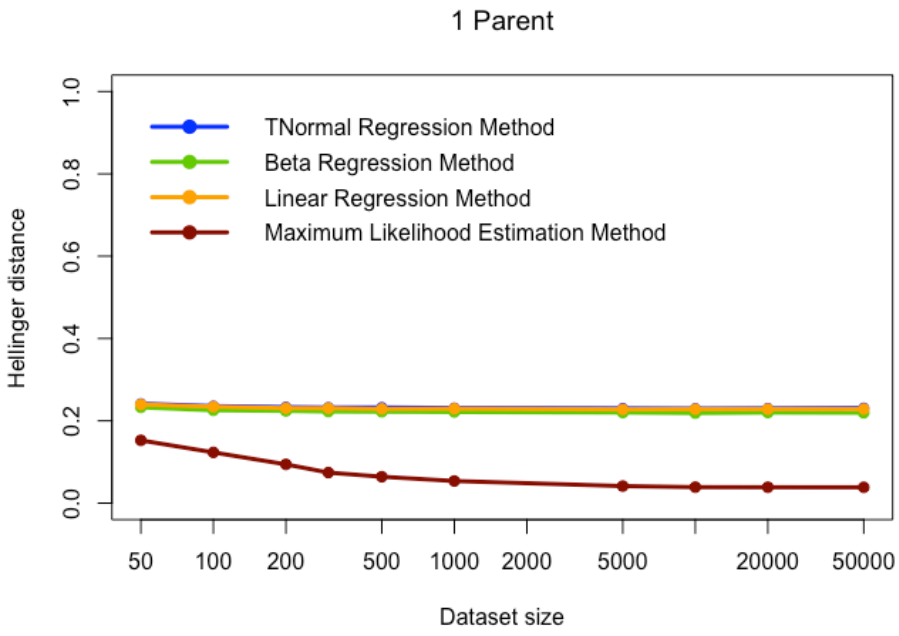
Figure 6.2 The structure of Child BN Model

The Child BN model was learned with TNormal regression algorithm, Beta regression algorithm and Linear regression algorithm. Then the Child BN model was learned by MLE algorithm. The graphs in Figure 6.3 are separated according to the variables' number of parents. For variables with one parent, the graphs of distances between learned CPTs and true CPTs are drawn as a) and for two parents, they are drawn as b). The distances between true CPTs and TNormal regression's CPTs learned by 50, 100, 200, 500, 1000, 5000, 10000 and 50000 size of data are shown by blue line, the distance between true CPTs and Beta regression's CPTs are shown by the green line, the distance between true CPTs and linear regression's CPTs are shown by the orange line and, the distances between true CPTs and CPTs learned by MLE are shown by the red line.

The performances of TNormal regression, Beta regression and linear regression algorithms was close to each other. The performances of these algorithms did not change

by increasing the data size for both variables that have one parent and two parents. The performance of MLE algorithm increased as the data size increases. And the overall performance of MLE was better than proposed algorithm for the Child model.

a)



b)

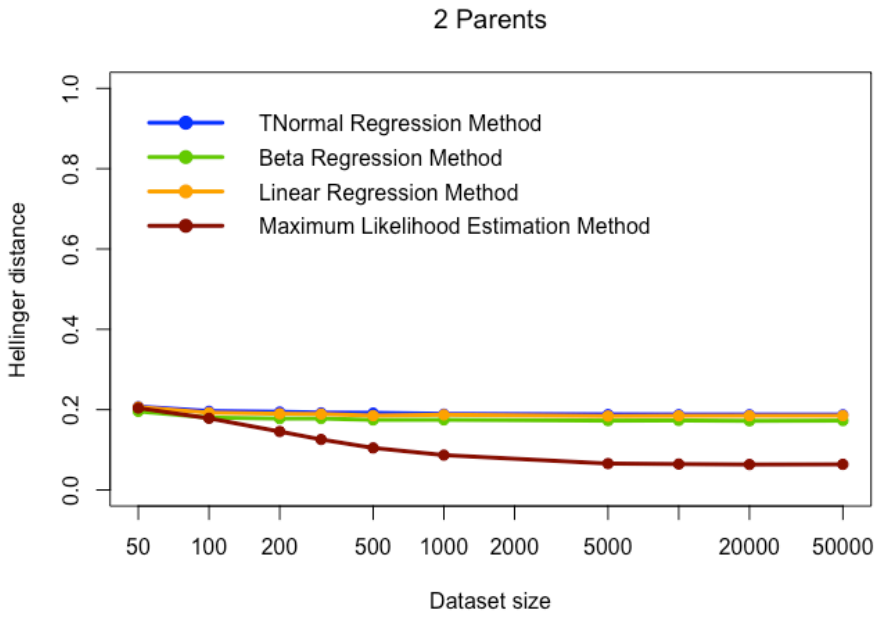


Figure 6.3 The Hellinger distances between learned CPTs and true CPTs for Child Model with a)1 parent, b)2 parents

6.1.2. Insurance Model

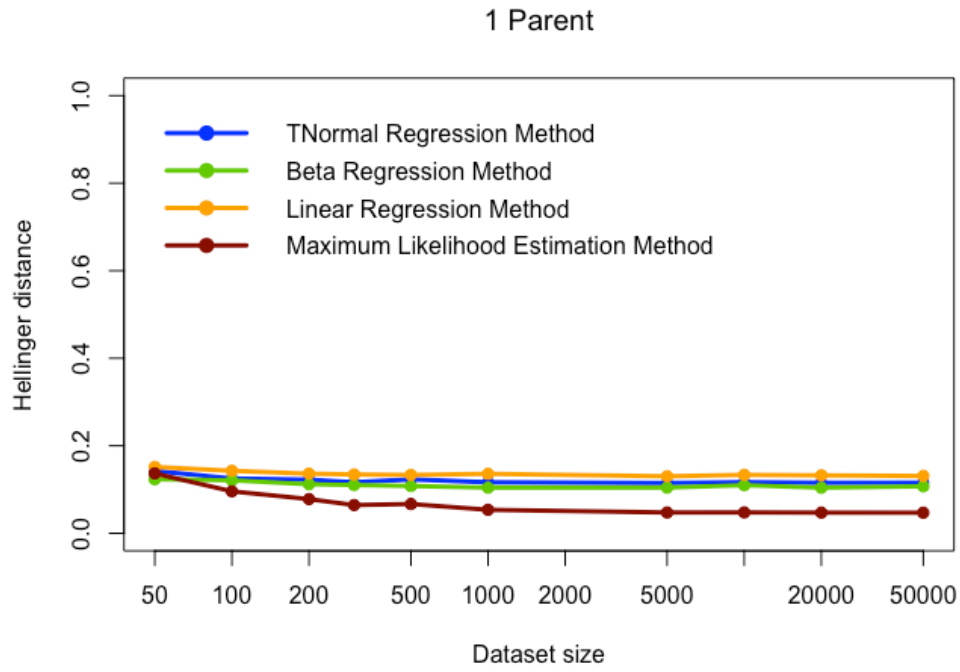
The Insurance BN model provided from BN repository is a discrete model. Its structure is given in Figure 6.4. Insurance model is generated for risk estimation of car insurance. It has 27 nodes and 1400 parameters. The maximum number of parents in Insurance model is 3. It is learned by algorithms proposed in this study and then the Hellinger distances between learned CPTs and true CPTs are calculated.

Figure 6.4 The structure of Insurance BN model

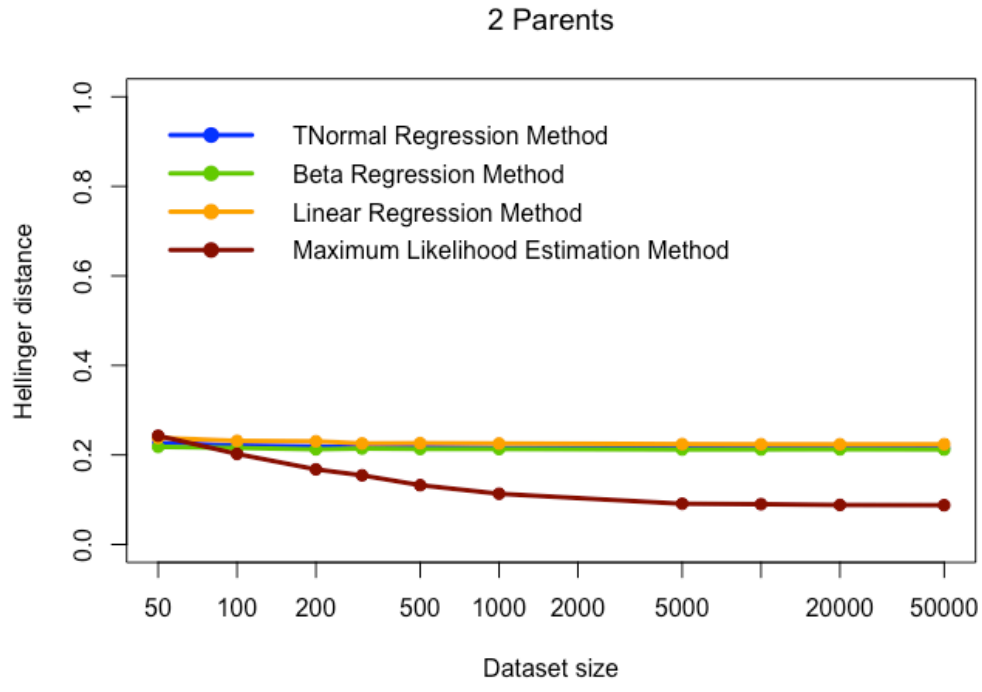
The performances of the algorithms for Insurance BN model are given in the graphs in Figure 6.5. Here the performance of TNormal regression and Beta regression algorithms was better than MLE for all number of parents only when the CPTs learned with a data size of 50. As the data size increases to 100, MLE algorithm performed better than all the proposed algorithms. The performance of proposed algorithms was close to each other, but Beta regression algorithm was the best performance among them. The proposed approaches performed good but as the model is not too complex and the data is enough

for MLE algorithm, the proposed algorithm could not reach the performance of MLE algorithm.

a)



b)



c)

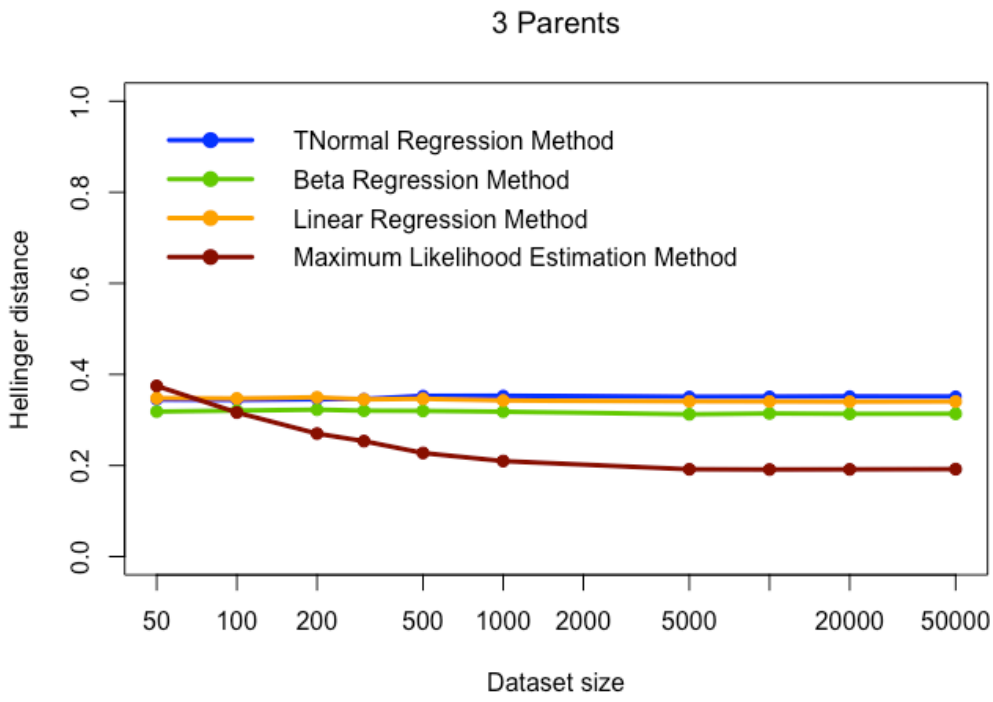


Figure 6.5 The Hellinger distances between learned CPTs and true CPTs for Insurance Model with a)1 parent, b)2 parents, c)3 parents

6.1.3. Mildew Model

Mildew BN in Figure 6.6 is a medium sized BN that has 35 nodes and 540150 parameters. The maximum number of parents is 3 in Mildew BN.

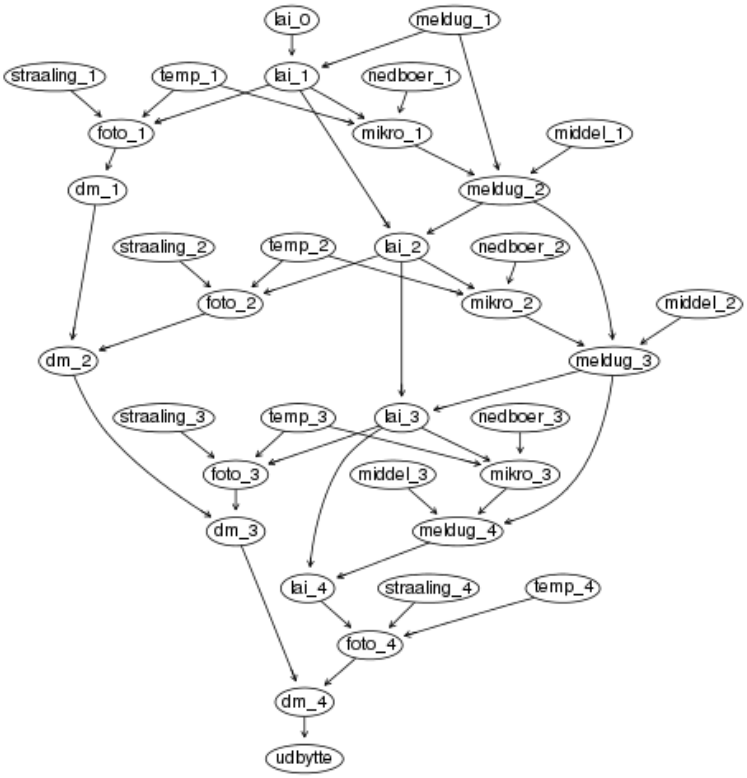
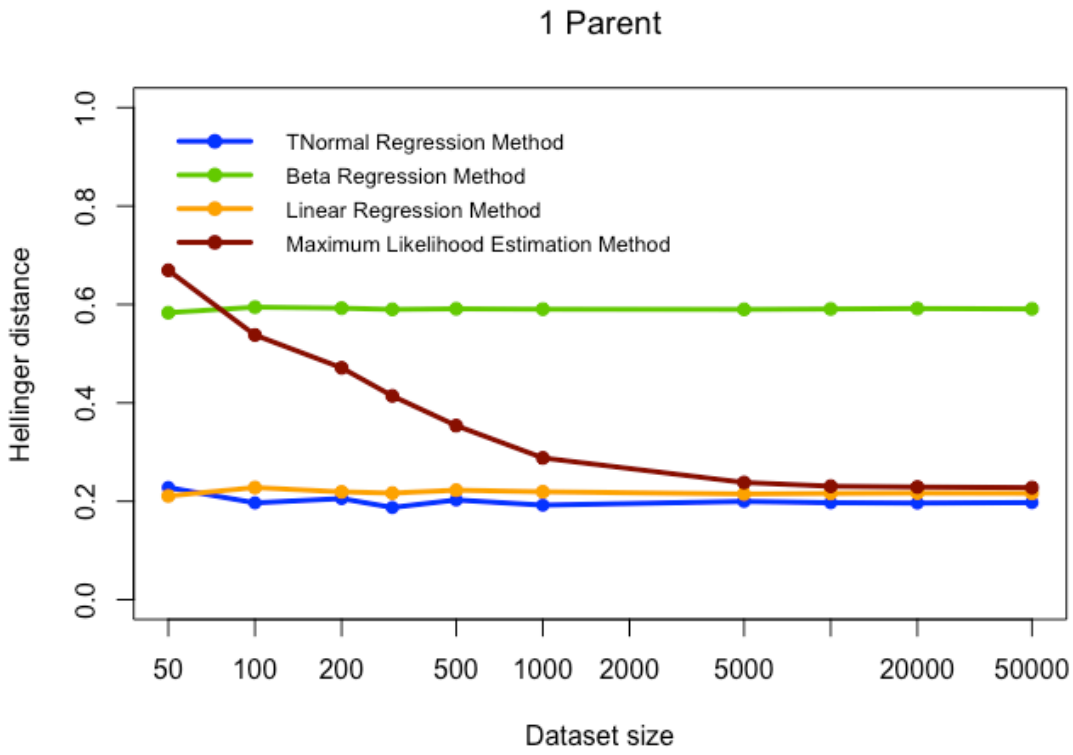


Figure 6.6 The structure of Mildew BN model

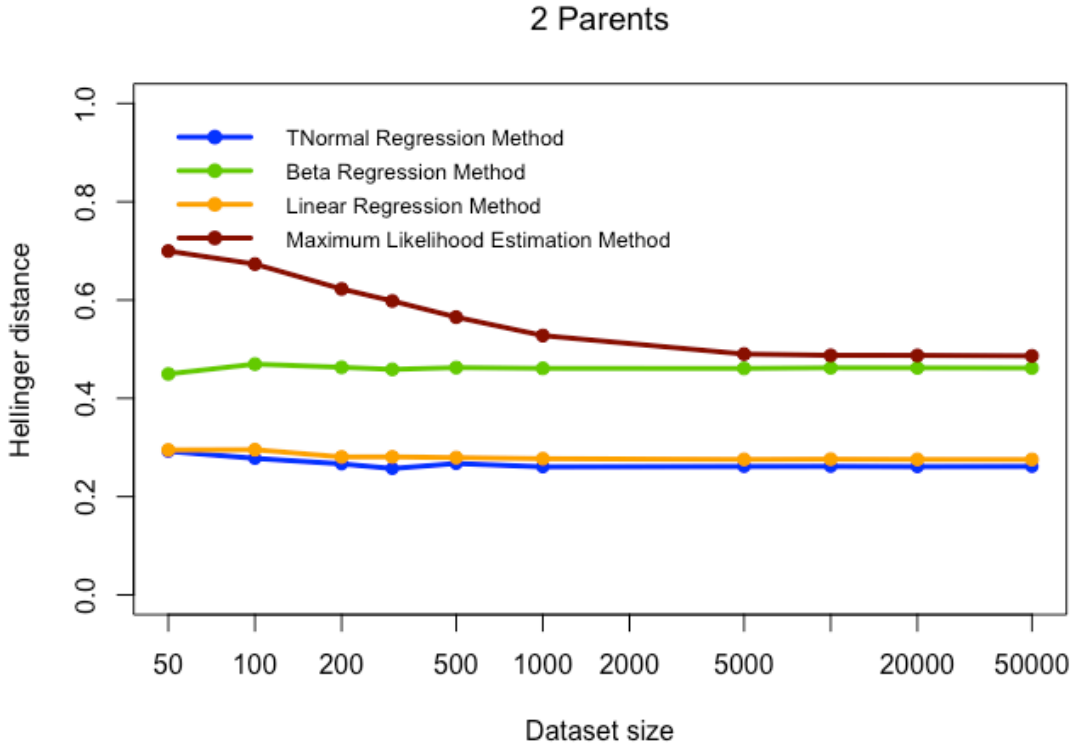
In the Mildew BN, the TNormal regression algorithm and linear regression algorithm provided better performances than MLE algorithm for all sample sizes for nodes with 1 parent and 2 parents (Figure 6.7). For nodes that have 3 parents, they performed better than MLE algorithm where the sample size is smaller than 1000. The MLE algorithm caught the performance of the TNormal Regression algorithm with sample sizes of 1000 for nodes with 3 parents.

Beta regression algorithm performed better than MLE algorithm only when the sample size is 50 for nodes with 1 parent. For 2 parents, it performed better than MLE algorithm but worse than TNormal and linear regression. When the sample size is smaller than 1000 it performed better than MLE for 3 parents.

a)


1 Parent
TNormal Regression Method
Beta Regression Method
Linear Regression Method
Maximum Likelihood Estimation Method
Hellinger distance
0.0
0.2
0.4
0.6
0.8
1.0
50
100
200
500
1000
2000
5000
20000
50000
Dataset size

b)


2 Parents
TNormal Regression Method
Beta Regression Method
Linear Regression Method
Maximum Likelihood Estimation Method
Hellinger distance
0.0
0.2
0.4
0.6
0.8
1.0
50
100
200
500
1000
2000
5000
20000
50000
Dataset size

c)

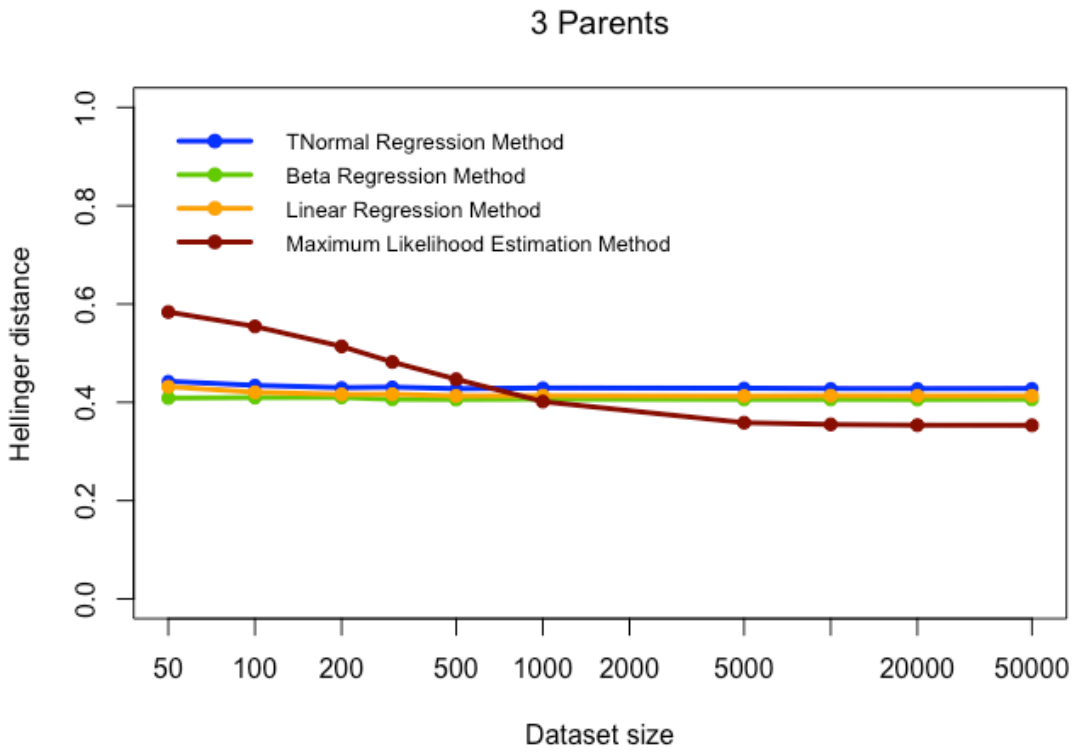


Figure 6.7 The Hellinger distances between learned CPTs and true CPTs for Mildew Model with a)1 parent, b)2 parents, c)3 parents

6.1.4. Barley Model

Barley model given in Figure 6.8 is the largest BN model used in this study. It has 48 nodes and 114005 parameters. Maximum number of parents in this model is 4.

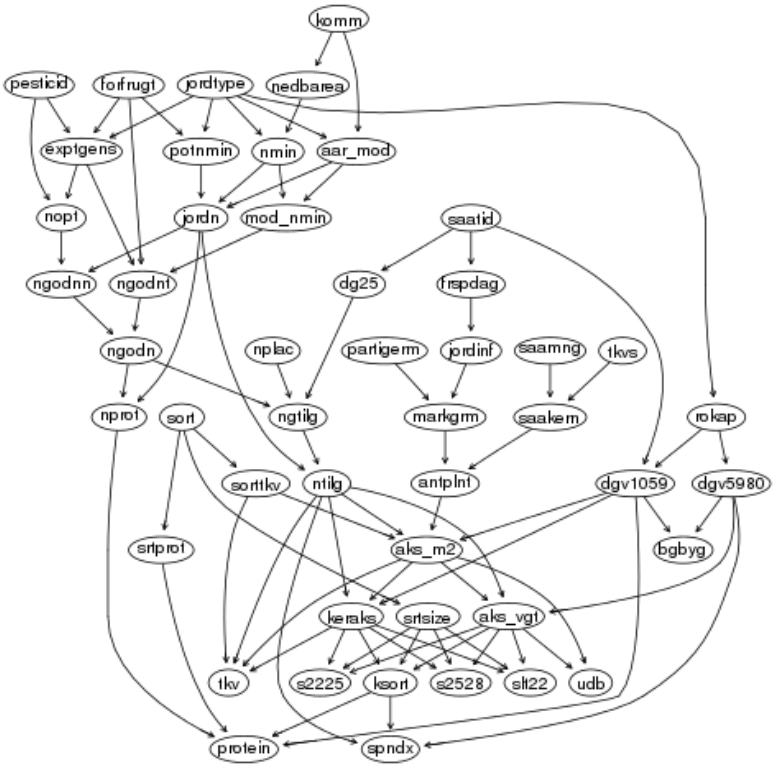
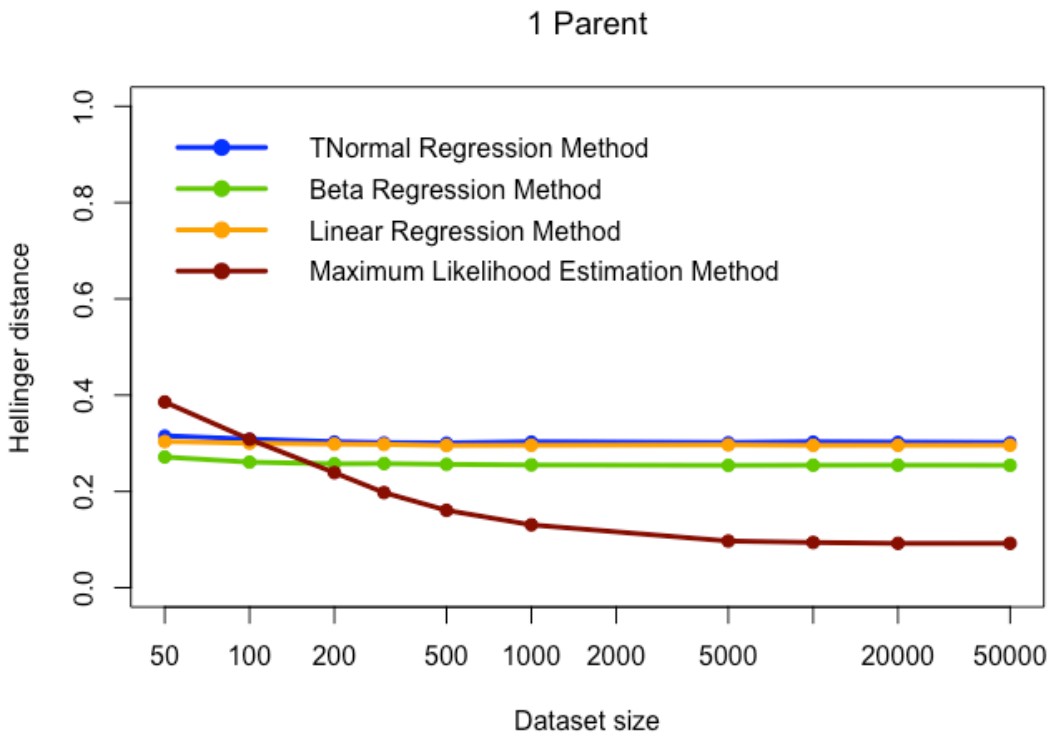


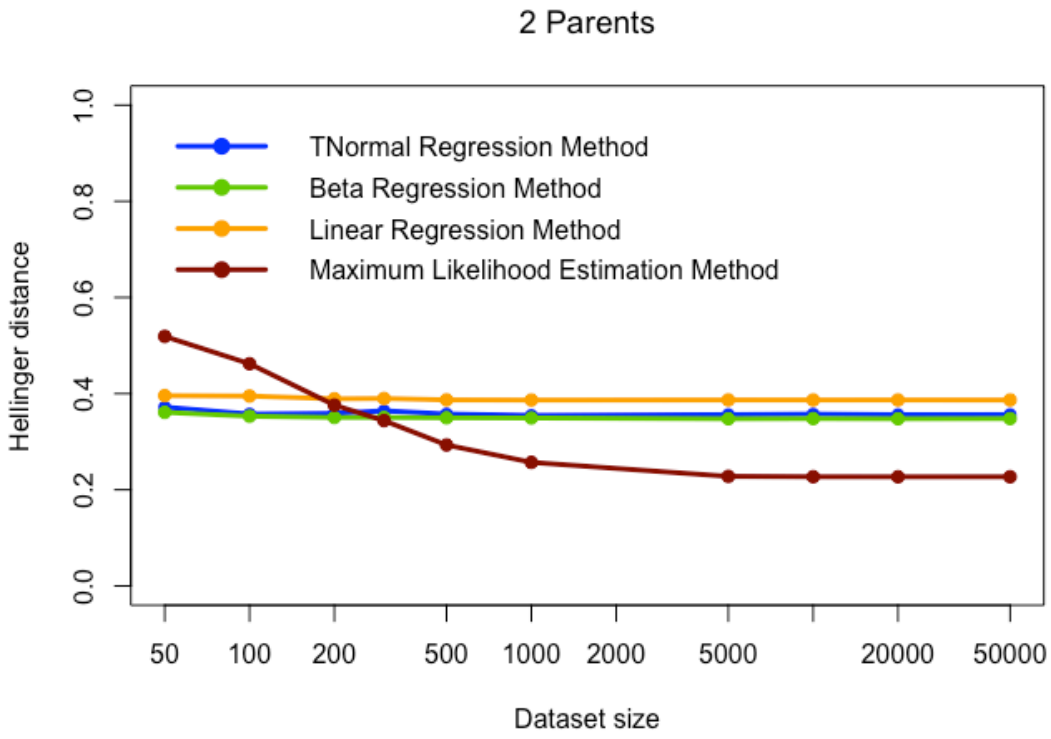
Figure 6.8 The structure of Barley BN model

In the Barley BN, TNormal regression and Beta regression algorithms provided better performance for all numbers of parents in a small sample size (Figure 6.9). MLE was able to catch the performance of the TNormal regression and Beta regression algorithms with 100 and 300 sizes for nodes with 1 and 2 parents respectively. The proposed algorithms had better performance in all sample sizes for nodes with 3 and 4 parents. The performance of MLE become better when the sample size increases, but for 3 parent and 4 parents MLE could not catch the performance of TNormal and Beta regression.

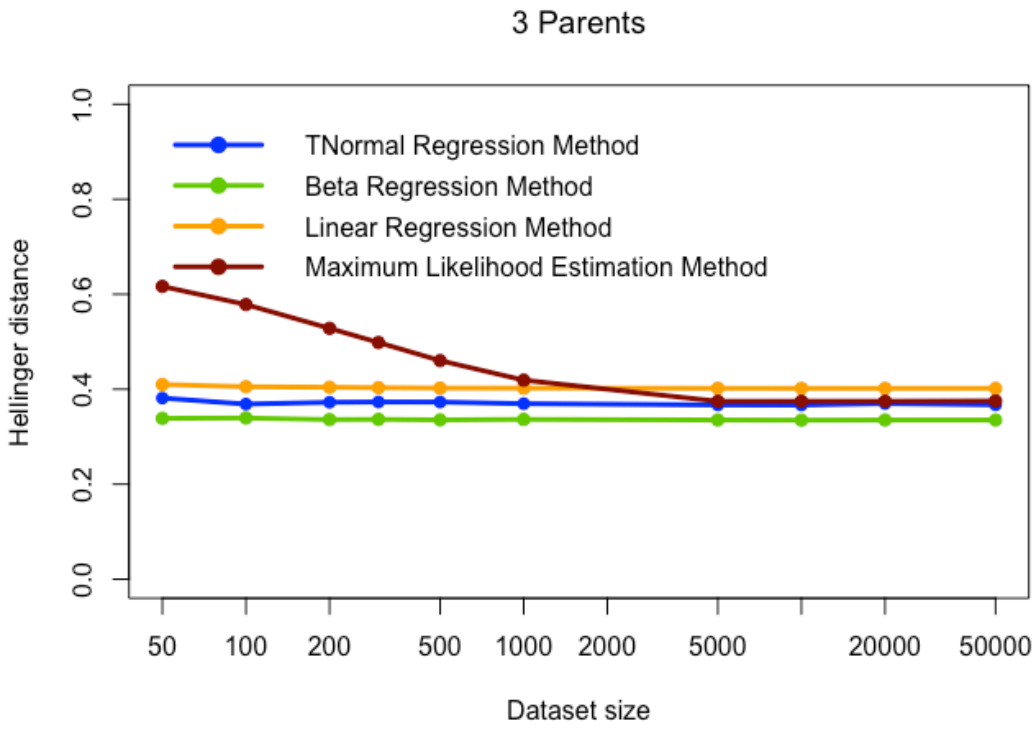
a)


1 Parent
TNormal Regression Method
Beta Regression Method
Linear Regression Method
Maximum Likelihood Estimation Method
Hellinger distance
0.0
0.2
0.4
0.6
0.8
1.0
50
100
200
500
1000
2000
5000
20000
50000
Dataset size

b)


2 Parents
TNormal Regression Method
Beta Regression Method
Linear Regression Method
Maximum Likelihood Estimation Method
Hellinger distance
0.0
0.2
0.4
0.6
0.8
1.0
50
100
200
500
1000
2000
5000
20000
50000
Dataset size

c)



d)

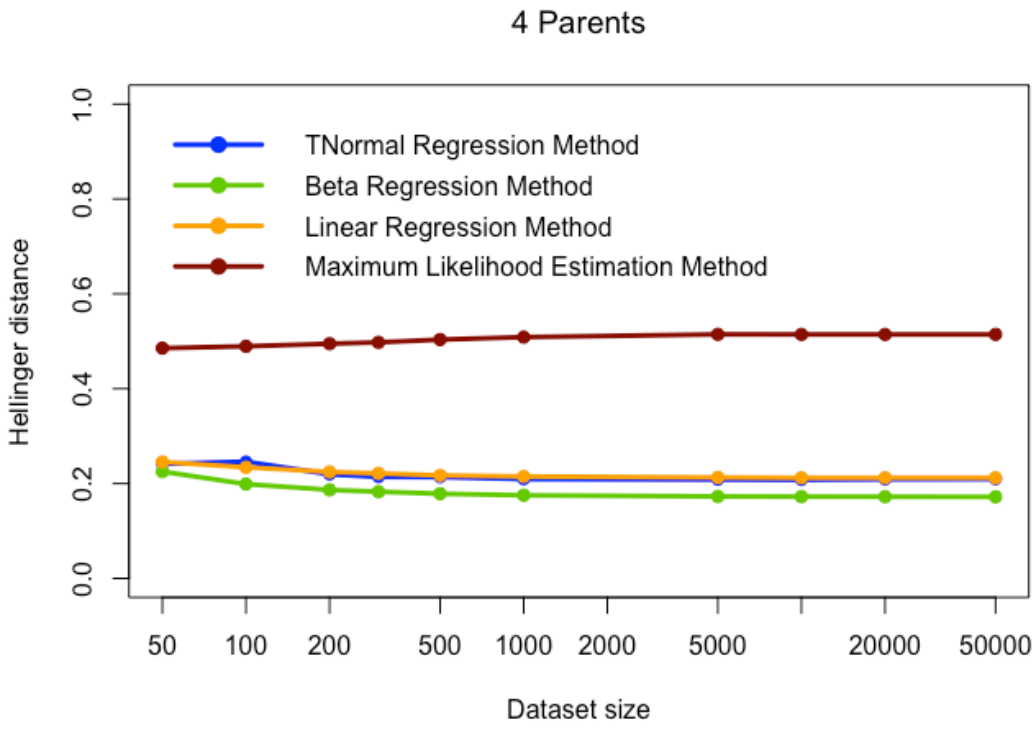


Figure 6.9 The Hellinger distances between learned CPTs and true CPTs for Barley Model with a)1 parent, b)2 parents, c)3 parents and d)4 parents

Three proposed parameter learning algorithms for ranked nodes are tested with several BNs and are compared with the existing MLE learning algorithm. The results showed that the MLE learning algorithm needs more data as the parent number of variables is increased. For complex BN models like Barley model, even the data size of 50000 was not enough for MLE to learn the parameters of CPTs of variables which have 4 parents. On the other hand, the proposed algorithms presented good performance for BNs with complex structures. They learned the CPTs of variables with several parents even with the small data size as 50. Also, the performance of proposed algorithms did not change with data size. The proposed algorithms modeled the linear relationship between variables, and because of this property they could perform good when the data size is small.

7. CONCLUSION

This thesis proposed 3 methods to learn ranked nodes from data. The TNormal, Beta and linear regression algorithms estimated the parameters of ranked nodes without parents and fit a regression model to ranked nodes with parents. Then they generated CPTs from these distributions and regression models using a sampling-based approach. The performances of the methods are evaluated using four BN models with different sizes and complexities.

The proposed TNormal, Beta and linear regression algorithms gave similar performance for most of the models. Also, they gave better results when the models get complicated. The small-sized and less complicated BNs can be learned by MLE method better than regression algorithms so there is no need for an algorithm that learns from small data sizes. But as the model gets complex and the dependencies in a BN increase, it gets difficult to learn the BN with small data size because there is not enough data that represent all of the possible dependencies in BNs. In those situations, algorithms that learn from small data size such as regression algorithms can be a solution. In complex BN models tested, when the data size is small the performance of regression algorithms was better than MLE for all number of parents.

The performances of proposed regression algorithms were better than MLE algorithm when the number of parents of variables increases. By the same reason given previously with small data size, there was not enough data to learn the parameters by MLE for different state combinations of parent nodes. Mildew and Barley BN models learned better with TNormal than MLE method when the number of parents gets larger.

Increasing the data size increased the performance of MLE algorithm but the performance of TNormal regression algorithm did not change much when the data size increased. In all BN models tested, the MLE method tended to perform better when the data size is increased; on the other hand, the performance of regression methods did not change.

The proposed methods performed better than MLE in these experiments when the size of the CPT was large or when the sample size was small. In future studies, developing a hybrid approach that uses the proposed method interchangeably with the conventional learning technique depending on the size of the available data can be studied. For example, the small-sized CPTs of BN model can be learned by MLE method and large sized CPTs can be learned by TNormal regression model. This study can be expanded for other weighed functions defined for ranked nodes i.e. weighted minimum and weighted maximum functions. Also, there may be models which has uncompleted or missing data. The expanding algorithm that handle missing data problem can be improved also.

8. REFERENCES

Díez, F. J. (1993). Parameter adjustment in Bayes networks. The generalized noisy OR–gate. In *Uncertainty in Artificial Intelligence*. https://doi.org/10.1016/b978-1-4832-1451-1.50016-0

Fenton, N. E., Neil, M., & Caballero, J. G. (2007). Using ranked nodes to model qualitative judgments in bayesian networks. *IEEE Transactions on Knowledge and Data Engineering*, *19*(10), 1420–1432. https://doi.org/10.1109/TKDE.2007.1073

Fenton, N. E., Noguchi, T., & Neil, M. (2019). An Extension to the Noisy-OR Function to Resolve the "Explaining Away" Deficiency for Practical Bayesian Network Problems. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2019.2891680

Fenton, N., & Neil, M. (2012). Risk assessment and decision analysis with bayesian networks. In *Risk Assessment and Decision Analysis with Bayesian Networks*. https://doi.org/10.1201/b21982

Koller, D., & Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series). In *Foundations*. https://doi.org/10.1016/j.ccl.2010.07.006

Koller, D., & Pfeffer, A. (1997). Object-oriented Bayesian networks. *Proceedings of the Thirteenth Conference on …*. https://doi.org/10.1016/j.fsigen.2007.12.003

Korb, K., & Nicholson, A. (2011). *Bayesian Artifical Intelligence*.

Laitila, P., & Virtanen, K. (2016). Improving Construction of Conditional Probability Tables for Ranked Nodes in Bayesian Networks. *IEEE Transactions on Knowledge*

and Data Engineering, *28*(7), 1691–1705. https://doi.org/10.1109/TKDE.2016.2535229

Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society: Series B (Methodological)*. https://doi.org/10.1111/j.2517-6161.1988.tb01721.x

Lauritzen, Steffen L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*. https://doi.org/10.1016/0167-9473(93)E0056-A

Neil, M., Fenton, N., & Nielsen, L. (2000). Building large-scale Bayesian networks. *Knowledge Engineering Review*. https://doi.org/10.1017/S0269888900003039

Noguchi, T., Fenton, N., & Neil, M. (2019). Addressing the Practical Limitations of Noisy-OR Using Conditional Inter-Causal Anti-Correlation with Ranked Nodes. *IEEE Transactions on Knowledge and Data Engineering*. https://doi.org/10.1109/TKDE.2018.2873314

Nunes, J., Willamy, R., Perkusich, M., Saraiva, R., Gorgonio, K., Almeida, H., & Perkusich, A. (2017). An Algorithm to Define the Node Probability Functions of Bayesian Networks based on Ranked Nodes. *International Journal of Engineering Trends and Technology*, *52*(3), 151–156. https://doi.org/10.14445/22315381/ijett-v52p223

Oniśko, A., Druzdzel, M. J., & Wasyluk, H. (2001). Learning Bayesian network parameters from small data sets: Application of Noisy-OR gates. *International Journal of Approximate Reasoning*. https://doi.org/10.1016/S0888-613X(01)00039-1

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: : networks of plausible inference (Morgan kaufmann series in representation and reasoning). In *Morgan*

Kaufmann Publishers, San Mateo, Calif. https://doi.org/10.2307/2026705

Spirtes, P., Glymour, C., & Scheines, R. (2000). Causation, prediction, and search, 2nd edition. In *Journal of Marketing Research*.

Srinivas, S. (1993). A Generalization of the Noisy-Or Model. In *Uncertainty in Artificial Intelligence*. https://doi.org/10.1016/b978-1-4832-1451-1.50030-5

Tsamardinos, I., Brown, L. E., & Aliferis, C. F. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*. https://doi.org/10.1007/s10994-006-6889-7

Virtanen, K. (2018). *On Theoretical Principle and Practical Applicability of Ranked Nodes Method for Constructing Conditional Probability Tables of Bayesian Networks*. 1–13.

Wiegerinck, W., Burgers, W., & Kappen, B. (2013). Bayesian Networks, Introduction and Practical Applications. *Intelligent Systems Reference Library*. https://doi.org/10.1007/978-3-642-36657-4_12

Yet, B., Perkins, Z., Fenton, N., Tai, N., & Marsh, W. (2014). Not just data: A method for improving prediction with knowledge. *Journal of Biomedical Informatics*. https://doi.org/10.1016/j.jbi.2013.10.012

Zagorecki, A., & Druzdzel, M. J. (2013). Knowledge engineering for bayesian networks: How common are noisy-MAX distributions in practice'. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, *43*(1), 186–195. https://doi.org/10.1109/TSMCA.2012.2189880

APPENDIX

Appendix 1 – Summary Tables of Child Model

CHILD MODEL (1 PARENT)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.239	0.006	[0.231, 0.251]	0.233	0.008	[0.220, 0.246]	0.240	0.010	[0.230, 0.261]	0.154	0.011	[0.140, 0.171]
100	0.242	0.008	[0.229, 0.256]	0.224	0.005	[0.216, 0.230]	0.233	0.005	[0.226, 0.242]	0.120	0.013	[0.099, 0.140]
200	0.234	0.006	[0.228, 0.243]	0.225	0.005	[0.217, 0.229]	0.234	0.004	[0.228, 0.239]	0.085	0.006	[0.077, 0.094]
300	0.234	0.004	[0.230, 0.240]	0.221	0.004	[0.216, 0.229]	0.230	0.003	[0.225, 0.234]	0.072	0.007	[0.061, 0.080]
500	0.233	0.002	[0.229, 0.235]	0.221	0.003	[0.216, 0.223]	0.228	0.003	[0.224, 0.232]	0.052	0.005	[0.044, 0.061]
1000	0.233	0.001	[0.230, 0.234]	0.220	0.001	[0.218, 0.222]	0.229	0.002	[0.224, 0.231]	0.038	0.003	[0.034, 0.043]
5000	0.229	0.001	[0.228, 0.231]	0.220	0.001	[0.216, 0.222]	0.229	0.001	[0.227, 0.230]	0.017	0.001	[0.014, 0.019]
10000	0.231	0.002	[0.228, 0.233]	0.220	0.002	[0.217, 0.222]	0.229	0.002	[0.225, 0.231]	0.012	0.001	[0.011, 0.013]
20000	0.231	0.001	[0.230, 0.233]	0.220	0.001	[0.218, 0.222]	0.228	0.001	[0.226, 0.231]	0.009	0.001	[0.008, 0.010]
50000	0.231	0.002	[0.228, 0.233]	0.219	0.001	[0.217, 0.223]	0.228	0.001	[0.225, 0.230]	0.007	0.000	[0.007, 0.008]

CHILD MODEL (2 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.204	0.014	[0.187, 0.229]	0.193	0.013	[0.176, 0.209]	0.207	0.011	[0.191, 0.224]	0.229	0.017	[0.197, 0.247]
100	0.202	0.011	[0.184, 0.216]	0.184	0.008	[0.174, 0.197]	0.195	0.006	[0.186, 0.205]	0.189	0.011	[0.176, 0.209]
200	0.194	0.005	[0.189, 0.202]	0.178	0.003	[0.173, 0.182]	0.187	0.004	[0.182, 0.193]	0.142	0.013	[0.128, 0.163]
300	0.187	0.002	[0.185, 0.190]	0.175	0.002	[0.171, 0.178]	0.188	0.005	[0.182, 0.193]	0.120	0.009	[0.107, 0.132]
500	0.191	0.003	[0.187, 0.197]	0.176	0.04	[0.173, 0.183]	0.187	0.003	[0.183, 0.192]	0.096	0.05	[0.088, 0.102]
1000	0.187	0.003	[0.184, 0.191]	0.174	0.003	[0.169, 0.178]	0.185	0.002	[0.181, 0.189]	0.069	0.004	[0.064, 0.074]
5000	0.188	0.001	[0.185, 0.189]	0.172	0.001	[0.170, 0.173]	0.185	0.001	[0.183, 0.187]	0.029	0.002	[0.026, 0.034]
10000	0.187	0.002	[0.184, 0.191]	0.173	0.001	[0.171, 0.174]	0.185	0.002	[0.182, 0.188]	0.021	0.001	[0.018, 0.022]
20000	0.187	0.001	[0.185, 0.189]	0.172	0.001	[0.171, 0.173]	0.183	0.002	[0.180, 0.187]	0.015	0.001	[0.013, 0.018]
50000	0.187	0.001	[0.186, 0.189]	0.171	0.002	[0.169, 0.174]	0.183	0.001	[0.181, 0.186]	0.012	0.000	[0.011, 0.014]

Appendix 2- Summary Tables of Insurance Model

INSURANCE MODEL (1 PARENT)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.138	0.020	[0.117, 0.176]	0.112	0.016	[0.096, 0.140]	0.137	0.012	[0.120, 0.159]	0.128	0.034	[0.091, 0.188]
100	0.123	0.021	[0.104, 0.166]	0.108	0.019	[0.087, 0.146]	0.131	0.027	[0.110, 0.185]	0.109	0.023	[0.076, 0.143]
200	0.123	0.014	[0.104, 0.145]	0.101	0.010	[0.088, 0.118]	0.125	0.026	[0.104, 0.176]	0.084	0.018	[0.066, 0.118]
300	0.116	0.012	[0.103, 0.138]	0.097	0.004	[0.093, 0.104]	0.118	0.006	[0.110, 0.128]	0.069	0.009	[0.052, 0.084]
500	0.119	0.011	[0.105, 0.135]	0.096	0.003	[0.092, 0.101]	0.119	0.014	[0.103, 0.148]	0.062	0.010	[0.052, 0.080]
1000	0.113	0.006	[0.105, 0.124]	0.099	0.005	[0.089, 0.103]	0.117	0.008	[0.106, 0.132]	0.054	0.007	[0.042, 0.062]
5000	0.114	0.004	[0.109, 0.117]	0.096	0.002	[0.094, 0.099]	0.117	0.004	[0.112, 0.125]	0.049	0.004	[0.042, 0.054]
10000	0.114	0.005	[0.107, 0.121]	0.096	0.004	[0.092, 0.103]	0.116	0.003	[0.111, 0.120]	0.048	0.001	[0.045, 0.050]
20000	0.113	0.003	[0.108, 0.118]	0.096	0.002	[0.091, 0.100]	0.115	0.003	[0.112, 0.119]	0.048	0.002	[0.046, 0.051]
50000	0.116	0.004	[0.111, 0.123]	0.098	0.003	[0.092, 0.100]	0.115	0.002	[0.112, 0.119]	0.048	0.001	[0.047, 0.049]

INSURANCE MODEL (2 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.231	0.008	[0.222, 0.247]	0.222	0.005	[0.217, 0.230]	0.237	0.005	[0.228, 0.244]	0.246	0.006	[0.240, 0.254]
100	0.221	0.007	[0.214, 0.234]	0.216	0.003	[0.210, 0.221]	0.229	0.004	[0.224 0.234]	0.207	0.007	[0.196, 0.216]
200	0.219	0.004	[0.214, 0.225]	0.215	0.004	[0.210, 0.221]	0.228	0.005	[0.224, 0.237]	0.171	0.006	[0.160, 0.177]
300	0.219	0.002	[0.215, 0.221]	0.211	0.003	[0.208, 0.216]	0.225	0.002	[0.221, 0.227]	0.152	0.009	[0.146, 0.159]
500	0.217	0.002	[0.214, 0.220]	0.323	0.004	[0.316, 0.328]	0.225	0.001	[0.222, 0.227]	0.132	0.006	[0.121, 0.141]
1000	0.217	0.001	[0.215, 0.219]	0.210	0.001	[0.209, 0.213]	0.224	0.001	[0.220, 0.225]	0.116	0.007	[0.108, 0.130]
5000	0.216	0.001	[0.214, 0.217]	0.210	0.001	[0.209, 0.211]	0.223	0.001	[0.222, 0.224]	0.092	0.002	[0.089, 0.093]
10000	0.215	0.001	[0.214, 0.216]	0.210	0.001	[0.209, 0.212]	0.223	0.001	[0.222, 0.224]	0.090	0.001	[0.089, 0.091]
20000	0.216	0.001	[0.215, 0.218]	0.210	0.001	[0.209, 0.211]	0.223	0.001	[0.222, 0.224]	0089	0.001	[0.088, 0.090]
50000	0.216	0.001	[0.215, 0.217]	0.210	0.001	[0.209, 0.212]	0.223	0.001	[0.221, 0.224]	0.088	0.000	[0.087, 0.088]

INSURANCE MODEL (3 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.349	0.020	[0.328, 0.386]	0.316	0.010	[0.304, 0.331]	0.351	0.012	[0.332, 0.367]	0.370	0.013	[0.350, 0.386]
100	0.340	0.020	[0.319, 0.375]	0.316	0.015	[0.299, 0.337]	0.339	0.016	[0.319, 0.361]	0.313	0.014	[0.297, 0.336]
200	0.352	0.010	[0.334, 0.366]	0.320	0.005	[0.313, 0.326]	0.339	0.009	[0.325, 0.348]	0.278	0.014	[0.252, 0.293]
300	0.348	0.008	[0.335, 0.361]	0.325	0.006	[0.314, 0.331]	0.342	0.007	[0.333, 0.354]	0.251	0.008	[0.241, 0.262]
500	0.353	0.005	[0.346, 0.361]	0.322	0.004	[0.316, 0.329]	0.344	0.005	[0.335, 0.350]	0.231	0.010	[0.217, 0.243]
1000	0.350	0.002	[0.345, 0.352]	0.317	0.004	[0.209, 0.322]	0.344	0.005	[0.334, 0.351]	0.209	0.005	[0.201, 0.216]
5000	0.352	0.003	[0.346, 0.356]	0.320	0.002	[0.317, 0.322]	0.344	0.002	[0.342, 0.346]	0.193	0.002	[0.190, 0.197]
10000	0.351	0.002	[0.349, 0.354]	0.320	0.001	[0.319, 0.322]	0.344	0.001	[0.342, 0.345]	0.192	0.001	[0.191, 0.193]
20000	0.351	0.002	[0.347, 0.353]	0.320	0.001	[0.319, 0.321]	0.344	0.001	[0.342, 0.345]	0.192	0.001	[0.190, 0.192]
50000	0.351	0.001	[0.350, 0.353]	0.320	0.001	[0.319, 0.321]	0.344	0.001	[0.343, 0.346]	0.192	0.000	[0.192, 0.193]

Appendix 3 - Summary Tables of Mildew Model

MILDEW MODEL (1PARENT)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.227	0.044	[0.168, 0.298]	0.578	0.029	[0.525, 0.610]	0.243	0.027	[0.196, 0.286]	0.667	0.012	[0.652, 0.687]
100	0.208	0.032	[0.159, 0.258]	0.588	0.011	[0.565, 0.598]	0.222	0.021	[0.192, 0.253]	0.548	0.020	[0.509, 0.570]
200	0.218	0.022	[0.184, 0.241]	0.589	0.009	[0.578, 0.602]	0.235	0.012	[0.217, 0.251]	0.445	0.029	[0.402, 0.482]
300	0.207	0.017	[0.188, 0.238]	0.591	0.006	[0.581, 0.595]	0.231	0.012	[0.215, 0.248]	0.399	0.019	[0.376, 0.428]
500	0.208	0.017	[0.179, 0.225]	0.588	0.004	[0.578, 0.592]	0.223	0.009	[0.210, 0.235]	0.338	0.016	[0.319, 0.366]
1000	0.208	0.010	[0.193, 0.221]	0.594	0.003	[0.588, 0.597]	0.223	0.008	[0.211, 0.236]	0.280	0.014	[0.261, 0.303]
5000	0.208	0.006	[0.199, 0.217]	0.591	0.002	[0.588, 0.595]	0.223	0.005	[0.215, 0.230]	0.235	0.001	[0.233, 0.237]
10000	0.208	0.003	[0.202, 0.213]	0.591	0.002	[0.589, 0.594]	0.224	0.002	[0.220, 0.229]	0.231	0.002	[0.227, 0.234]
20000	0.208	0.002	[0.204, 0.210]	0.591	0.001	[0.589, 0.593]	0.225	0.002	[0.221, 0.228]	0.229	0.001	[0.227, 0.231]
50000	0.208	0.002	[0.205, 0.210]	0.591	0.001	[0.590, 0.593]	0.225	0.002	[0.222, 0.228]	0.227	0.000	[0.226, 0.229]

MILDEW MODEL (2 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.280	0.015	[0.260, 0.301]	0.436	0.040	[0.374, 0.483]	0.311	0.016	[0.290, 0.338]	0.711	0.005	[0.705, 0.720]
100	0.279	0.015	[0.261, 0.305]	0.451	0.034	[0.392, 0.491]	0.296	0.012	[0.280, 0.317]	0.672	0.005	[0.666, 0.682]
200	0.272	0.013	[0.251, 0.291]	0.456	0.016	[0.427, 0.470]	0.285	0.009	[0.272, 0.298]	0.624	0.006	[0.613, 0.631]
300	0.270	0.008	[0.258, 0.283]	0.459	0.010	[0.445, 0.473]	0.283	0.009	[0.274, 0.296]	0.600	0.006	[0.593, 0.607]
500	0.263	0.008	[0.250, 0.273]	0.458	0.008	[0.444, 0.468]	0.284	0.006	[0.277, 0.295]	0.562	0.006	[0.556, 0.572]
1000	0.261	0.006	[0.255, 0.270]	0.464	0.005	[0.458, 0.473]	0.280	0.004	[0.276, 0.286]	0.524	0.005	[0.520, 0.533]
5000	0.265	0.004	[0.260, 0.270]	0.461	0.004	[0.456, 0.467]	0.278	0.003	[0.274, 0.283]	0.490	0.001	[0.488, 491]
10000	0.265	0.002	[0.263, 0.268]	0.463	0.002	[0.458, 0.465]	0.279	0.002	[0.276, 0.282]	0.488	0.001	[0.487, 0.489]
20000	0.264	0.001	[0.262, 0.266]	0.463	0.002	[0.460, 0.465]	0.278	0.001	[0.277, 0.279]	0.487	0.001	[0.487, 0.487]
50000	0.264	0.000	[0.263, 0.265]	0.462	0.001	[0.461, 0.464]	0.278	0.001	[0.277, 0.278]	0.486	0.000	[0.485, 0.486]

MILDEW MODEL (3 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.440	0.008	[0.428, 0.454]	0.409	0.013	[0.386, 0.428]	0.428	0.007	[0.418, 0.439]	0.579	0.004	[0.575, 0,585]
100	0.435	0.006	[0.428, 0.445]	0.406	0.010	[0.387, 0.415]	0.420	0.004	[0.414, 0.426]	0.554	0.002	[0.550, 0.558]
200	0.430	0.003	[0.424, 0.435]	0.407	0.008	[0.397, 0.421]	0.417	0.004	[0.409, 0.421]	0.512	0.004	[0.506, 0.517]
300	0.430	0.002	[0.427, 0.433]	0.406	0.06	[0.398, 0.414]	0.414	0.002	[0.411, 0.418]	0.483	0.005	[0.475, 0.488]
500	0.430	0.003	[0.425, 0.434]	0.404	0.003	[0.399, 0.408]	0.416	0.002	[0.413, 0.420]	0.445	0.005	[0.438, 0.452
1000	0.427	0.001	[0.425, 0.430]	0.406	0.003	[0.401, 0.409]	0.415	0.002	[0.412, 0.419]	0.401	0.003	[0.396, 0.406]
5000	0.429	0.001	[0.426, 0.431]	0.405	0.002	[0.402, 0.407]	0.415	0.001	[0.413, 0.415]	0.358	0.001	[0.356, 0.360]
10000	0.429	0.001	[0.428, 0.431]	0.405	0.001	[0.404, 0.406]	0.413	0.001	[0.412, 0.414]	0.355	0.000	[0.354, 0.355]
20000	0.429	0.001	[0.427, 0.429]	0.406	0.001	[0.405, 0.407]	0.414	0.001	[0.413, 0.415]	0.354	0.000	[0.354, 0.355]
50000	0.429	0.000	[0.428, 0.429]	0.406	0.001	[0.405, 0.407]	0.413	0.000	[0.413, 0.414]	0.353	0.000	[0.353, 0.354]

Appendix 4 - Summary Tables of Barley Model

BARLEY MODEL (1 PARENT)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.314	0.006	[0.304, 0.323]	0.267	0.008	[0.251, 0.274]	0.302	0.010	[0.287, 0.315]	0.380	0.001	[0.364, 0.393]
100	0.306	0.005	[0.299, 0.315]	0.263	0.005	[0.258, 0.271]	0.299	0.004	[0.294, 0.307]	0.305	0.010	[0.289, 0.323]
200	0.305	0.004	[0.298, 0.311]	0.259	0.005	[0.253, 0.266]	0.296	0.003	[0.292, 0.301]	0.238	0.009	[0.227, 0.254]
300	0.303	0.002	[0.300, 0.306]	0.258	0.002	[0.255, 0.262]	0.298	0.003	[0.294, 0.301]	0.202	0.004	[0.197, 0.208]
500	0.305	0.002	[0.303, 0.308]	0.257	0.003	[0.252, 0.261]	0.296	0.002	[0.292, 0.299]	0.165	0.006	[0.153, 0.172]
1000	0.303	0.002	[0.300, 0.305]	0.258	0.002	[0.253, 0.260]	0.297	0.002	[0.293, 0.299]	0.131	0.004	[0.124, 0.138]
5000	0.303	0.01	[0.301, 0.304]	0.255	0.002	[0.253, 0.258]	0.295	0.002	[0.293, 0.298]	0.096	0.002	[0.095, 0.100]
10000	0.303	0.002	[0.299, 0.306]	0.255	0.001	[0.253, 0.257]	0.295	0.001	[0.293, 0.297]	0.094	0.001	[0.093, 0.095]
20000	0.302	0.002	[0.301, 0.305]	0.255	0.002	[0.252, 0.258]	0.296	0.001	[0.294, 0.298]	0.092	0.001	[0.092, 0.093]
50000	0.255	0.002	[0.252, 0.256]	0.255	0.002	[0.252, 0.257]	0.296	0.001	[0.294, 0.298]	0.091	0.000	[0.091, 0.092]

BARLEY MODEL (2 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.362	0.007	[0.354, 0.374]	0.360	0.007	[0.352, 0.371]	0.396	0.007	[0.387, 0.407]	0.516	0.008	[0.505, 0.529]
100	0.358	0.008	[0.351, 0.357]	0.355	0.005	[0.348, 0.364]	0.394	0.005	[0.387, 0.401]	0.451	0.009	[0.436, 0.467]
200	0.357	0.003	[0.352, 0.363]	0.349	0.003	[0.346, 0.354]	0.389	0.003	[0.385, 0.394]	0.378	0.007	[0.369, 0.392]
300	0.356	0.004	[0.351, 0.360]	0.351	0.002	[0.348, 0.354]	0.389	0.002	[0.385, 0.392]	0.336	0.005	[0.325, 0.341]
500	0.355	0.003	[0.369, 0.378]	0.349	0.004	[0.342, 0.352]	0.388	0.002	[0.385, 0.393]	0.294	0.004	[0.289, 0.302]
1000	0.355	0.002	[0.350, 0.357]	0.348	0.003	[0.341, 0.351]	0.387	0.002	[0.384, 0.390]	0.258	0.000	[0.255, 0.261]
5000	0.353	0.003	[0.346, 0.357]	0.346	0.004	[0.339, 0.350]	0.387	0.001	[0.386, 0.388]	0.228	0.000	[0.227, 0.229]
10000	0.352	0.005	[0.346, 0.382]	0.344	0.005	[0.337, 0.349]	0.387	0.001	[0.387, 0.388]	0.227	0.000	[0.226, 0.228]
20000	0.345	0.005	[0.342, 0.355]	0.343	0.004	[0.339, 0.350]	0.387	0.000	[0.386, 0.387]	0.227	0.000	[0.226, 0.227]
50000	0.339	0.004	[0.337, 0.342]	0.339	0.002	[0.337, 0.342]	0.387	0.001	[0.387, 0.388]	0.227	0.000	[0.226, 0.228]

BARLEY MODEL (3 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.382	0.011	[0.360, 0.392]	0.348	0.009	[0.337, 0.365]	0.412	0.007	[0.402, 0.420]	0.615	002	[0.612, 0.619]
100	0.373	0.009	[0.360, 0.384]	0.339	0.005	[0.332, 0.346]	0.409	0.004	[0.403, 0.416]	0.578	0.003	[0.573, 0.583]
200	0.372	0.010	[0.359, 0.385]	0.340	0.004	[0.336, 0.347]	0.404	0.003	[0.400, 0.408]	0.530	0.005	[0.521, 0.536]
300	0.372	0.002	[0.369, 0.375]	0.341	0.004	[0.335, 0.345]	0.403	0.002	[0.399, 0.407]	0.500	0.004	[0.492, 0.503]
500	0.373	0.003	[0.369, 0.378]	0.340	0.004	[0.334, 0.347]	0.403	0.002	[0.400, 0.405]	0.460	0.004	[0.453, 0.467]
1000	0.371	0.004	[0.366, 0.380]	0.338	0.002	[0.335, 0.342]	0.403	0.001	[0.401, 0.405]	0.416	0.004	[0.410, 0.422]
5000	0.372	0.004	[0.367, 0.378]	0.339	0.003	[0.336, 0.346]	0.402	0.000	[0.402, 0.403]	0.375	0.001	[0.374, 0.376]
10000	0.372	0.005	[0.369, 0.382]	0.339	0.004	[0.335, 0.346]	0.402	0.001	[0.401, 0.403]	0.374	0.000	[0.374, 0.375]
20000	0.372	0.005	[0.367, 0.381]	0.344	0.007	[0.337, 0.355]	0.402	0.000	[0.401, 0.402]	0.375	0.000	[0.375, 0.375]
50000	0.339	0.004	[0.335, 0.345]	0.339	0.004	[0.335, 0.345]	0.402	0.000	[0.401, 0.402]	0.375	0.000	[0.375, 0.375]

BARLEY MODEL (4 PARENTS)												
	TNORMAL REG.			BETA REG.			LINEAR REG.			MLE		
	μ	σ	CI	μ	σ	CI	μ	σ	CI	μ	σ	CI
50	0.256	0.018	[0.228, 0.278]	0.211	0.035	[0.175, 0.277]	0.265	0.019	[0.244, 0.420]	0.485	0.000	[0.484, 0.485]
100	0.228	0.014	[0.209, 0.252]	0.194	0.006	[0.182, 0.202]	0.240	0.012	[0.221, 0.260]	0.489	0.000	[0.488, 0.490]
200	0.223	0.008	[0.213, 0.235]	0.188	0.007	[0.179, 0.199]	0.226	0.008	[0.217, 0.238]	0.494	0.000	[0.493, 0.494]
300	0.216	0.004	[0.212, 0.224]	0.186	0.010	[0.171, 0.201]	0.220	0.005	[0.214, 0.228]	0.498	0.001	[0.498, 0.499]
500	0.215	0.005	[0.210, 0.222]	0.178	0.003	[0.174, 0.182]	0.218	0.005	[0.213, 0.227]	0.503	0.000	[0.502, 0.504]
1000	0.211	0.003	[0.207, 0.215]	0.175	0.003	[0.171, 0.180]	0.214	0.002	[0.211, 0.218]	0.509	0.001	[0.508, 0.510]
5000	0.210	0.002	[0.207, 0.212]	0.171	0.002	[0.169, 0.174]	0.212	0.001	[0.212, 0.213]	0.514	0.000	[0.514, 0.515]
10000	0.209	0.001	[0.208, 0.211]	0.171	0.001	[0.170, 0.172]	0.212	0.001	[0.212, 0.213]	0.514	0.000	[0.514, 0.514]
20000	0.209	0.000	[0.208, 0.209]	0.171	0.001	[0.169, 0.172]	0.212	0.000	[0.211, 0.212]	0.514	0.000	[0.514, 0.515]
50000	0.171	0.000	[0.170, 0.171]	0.171	0.000	[0.170, 0.171]	0.212	0.000	[0.211, 0.212]	0.514	0.000	[0.514, 0.515]