mathematics

MDPI

Article

Advanced Approach for Distributions Parameters Learning in Bayesian Networks with Gaussian Mixture Models and Discriminative Models

Irina Deeva *, Anna Bubnova and Anna V. Kalyuzhnaya

NSS Lab, ITMO University, Saint Petersburg 197101, Russia
* Correspondence: ideeva@itmo.ru; Tel.:+7-928-292-0889

Abstract: Bayesian networks are a powerful tool for modelling multivariate random variables. However, when applied in practice, for example, for industrial projects, problems arise because the existing learning and inference algorithms are not adapted to real data. This article discusses two learning and inference problems on mixed data in Bayesian networks—learning and inference at nodes of a Bayesian network that have non-Gaussian distributions and learning and inference for networks that require edges from continuous nodes to discrete ones. First, an approach based on the use of mixtures of Gaussian distributions is proposed to solve a problem when the joint normality assumption is not confirmed. Second, classification models are proposed to solve a problem with edges from continuous nodes to discrete nodes. Experiments have been run on both synthetic datasets and real-world data and have shown gains in modelling quality.

Keywords: Bayesian networks; parameters learning; Gaussian mixture model; classification models

MSC: 60E05; 62H22; 62C10

Citation: Deeva, I.; Bubnova, A.; Kalyuzhnaya, A.V. Advanced Approach for Distributions Parameters Learning in Bayesian Networks with Gaussian Mixture Models and Discriminative Models. *Mathematics* **2023**, *11*, 343. https://doi.org/10.3390/math11020343

Academic Editors: Vitaly Schetinin, Livija Jakaite and Dayou Li

Received: 6 December 2022
Revised: 30 December 2022
Accepted: 3 January 2023
Published: 9 January 2023

1. Introduction

Bayesian networks (BNs) are a powerful tool for modelling and presenting multidimensional data [1–3]. The core of Bayesian networks is a directed acyclic graph, at the vertices of which there are one-dimensional random variables, and the edges represent the presence of a statistical relationship between the values at the vertices [4,5]. Such a probabilistic-oriented model allows for statistical inference, filling in gaps, and detecting anomalous values in different areas [6]. However, the existing set of learning and inference algorithms is often unable to consider all the features of real data. For example, discrete Bayesian networks have been widely developed as well as algorithms for learning the structure and parameters of such networks. Such networks usually use conditional probability tables (CPT) to model distributions at nodes that have parents [7]. More recent approaches to learning discrete Bayesian networks use various parametric distributions, such as the Dirichlet distribution [8,9] or the Bernoulli distribution [10]. However, such networks cannot be used in cases where it is necessary to work with either continuous data or mixed data (continuous and discrete). For continuous data, there are Gaussian Bayesian networks (GBNs) [11,12]. GBN is a Bayesian network whose nodes are continuous and conditional distributions are represented as linear Gaussians. However, this kind of network has its limitations. First of all, it is impossible to use discrete nodes. Therefore, there are Hybrid Bayesian networks (HBNs) [13]. Such networks are capable of combining both discrete nodes and continuous ones using conditional Gaussian distributions [14]. However, HBNs have their limitations. First, this involves working with non-Gaussian distributions; often, discretisation is used if the joint normality assumption is not confirmed, but it leads to information loss. Additionally, sometimes, to solve the problem of approximating such distributions, non-parametric methods of representing distributions are used,

for example, using variational approximations [15] and copulas [16,17]. However, the main disadvantage of these approaches is the need to select the type of distribution for each application on real data, and the gates do not offer any solutions for automatic selection of distributions. In addition, such networks usually assume edges from discrete nodes to continuous ones, while sometimes the logic of dependencies suggests the opposite. Deep learning approaches can be used to solve such problems [18,19], but their main drawback is, firstly, the requirement for a sufficient amount of training data, and secondly, the training time can be quite large.

The purpose of this study is to solve two main problems for Hybrid Bayesian networks—modelling complex non-Gaussian distributions in Bayesian networks and modelling distributions in networks with edges from continuous nodes to discrete ones. We answer the questions in which cases the proposed methods are applicable and describe the use of the proposed methods on real datasets to demonstrate an increase in the quality of modelling. The main contribution of this study is that, firstly, a complete adaptation of mixture models of Gaussian distributions is performed for the problems of parametric learning of a Bayesian network. In addition, we propose to automatically estimate the number of mixture components, which increases the flexibility of the resulting distributions and the quality of the approximation. Secondly, we propose an adaptation of classification models for modelling conditional distributions, when a discrete child node has continuous parents, we propose a comparison of various classification models, on the basis of which conclusions are made about the limits of applicability of the compared models.

Of course, Gaussian mixtures are a well-known method for approximating non-Gaussian distributions. However, their application in the context of BNs has been limited to objects with continuous characteristics and has not extended to the situation with mixed data [20,21]. It should also be noted that Bayesian networks use not only mixtures of Gaussian, but also truncated exponential distributions [22] and polynomials [23]. However, these variants are rather aimed at solving the problem of continuous parents and discrete children and are outside the domain of Conditional Gaussian BNs. Some of our proposed methods for solving this problem have been implemented before, such as the application of logit regression [24], or KNN [25], but the set of models available in our library is richer. It can easily be extended by any serialisable classification method. It should also be noted that this paper considers the effect of component crowding for continuous parents on the usability of an edge from them to a discrete child, which ties our two problems together.

2. Problem Statement

Speaking about the widespread use of hybrid Bayesian networks in practice, we are faced with some limitations in their use on real data since existing approaches usually describe classical cases in data. A general hybrid Bayesian network is a directed acyclic graph (DAG) G representing the joint probability distribution of a given set of variables $\mathbf{V}$, including discrete variables $\mathbf{X}$ and continuous ones $\mathbf{Y}$:

$$HBN = (\mathbf{X}, \mathbf{Y}, \mathbf{L}, \mathbf{M}) \tag{1}$$

where $\mathbf{L}$ is a set of directed links between variables and $\mathbf{M}$ is a set of conditional probability distributions. Moreover, often a set of conditional distributions $\mathbf{M}$ simulates based on a Gaussian distribution, and a set of links $\mathbf{L}$ excludes the possibility of connecting links from continuous nodes to discrete ones. Thus, the developed approaches should be able to perform learning and inference in hybrid Bayesian networks with the following sets of conditional distributions and links:

$$\mathbf{M} = \{CPT, Gaussian\} \cup \{non - Gaussian\} \tag{2}$$

$$\mathbf{L} = \{\mathbf{X} \to \mathbf{X}, \mathbf{X} \to \mathbf{Y}, \mathbf{Y} \to \mathbf{Y}\} \cup \{\mathbf{Y} \to \mathbf{X}\} \tag{3}$$

Thus, the task is to combine the classical approaches to learning the parameters of distributions in the nodes of the Bayesian network and approaches based on mixtures of Gaussian distributions and classification models in order to extend the applicability of Bayesian networks to real data. In a general sense, the problem can be formalized as follows. For a hybrid network G (Figure 1) and a training data set D, which is tabular data, it is necessary to train the distribution parameters, which look like this:

$$P(\theta|G,D) = \{P_1(y_1|w_{y_1},\mu_{y_1},\Sigma_{y_1}), P_2(x_1|f(y_1)), P_3(x_2|t(x_1)), \\ P_4(y_2|w_{y_2|x_1,y_1},\mu_{y_2|x_1,y_1},\Sigma_{y_2|x_1,y_1})\}, \quad (4)$$

where w, μ, Σ—vector of weights, vector of means and matrix of covariance for Gaussian mixture model; f—classification model; t—conditional probability table model.

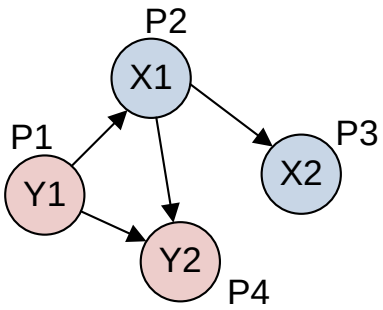


Figure 1. An example of a hybrid Bayesian network, blue nodes are discrete data, pink nodes are continuous data.

3. Gaussian Mixture Model in Parameter Learning

3.1. Gaussian Mixture Model

The Gaussian mixture model is a parametric probability density function presented as a weighted sum of the densities of the Gaussian components in Equation (5).

$$p(x) = \sum_{i=1}^{k} w_i g(x|\mu_i,\Sigma_i), \quad (5)$$

where k—number of components, $g(x|\mu_i,\Sigma_i)$—Gaussian mixture component, μ_i and Σ_i—vector of means and matrix of covariance.

It is necessary to determine how we will find the number of mixture components in the training process to use mixtures of Gaussian distributions. Automatic selection of the number of components, firstly, allows you to choose the type of mixture that best describes the approximated distribution, and secondly, allows you to control the size of the resulting model without making it too complicated, unlike, for example, deep learning methods, where such control is cannot always be achieved and sometimes it leads to overfitting. There are many different methods to determine the number of components, for example, the use of information criteria (BIC, AIC) [26]. However, when choosing a method for finding the number of components, it is necessary to pay attention not only to the accuracy of the method but also to the time of its calculation, since this method will be used in parameters learning of BNs and we do not want to increase the time of this learning significantly. Since estimates based on BIC and AIC work the fastest, it was decided to use them in the form of some integral measure. Thus, the problem of selecting the number of components can be formalized as follows:

$$k_{opt} = \arg \max_{1 \leqslant k \leqslant k_{max}} E(\mathbf{Y},k), \quad (6)$$

where k_{max}—this is the upper limit on the number of components (in this study, it is assumed to be 10), and E is the evaluation function, which is equal to:

$$E(\mathbf{Y}, k) = \lfloor u_1 AIC(\mathbf{Y}, k) + u_2 BIC(\mathbf{Y}, k) \rfloor, \tag{7}$$

where u_1 and u_2 are the weights of the estimates, in this study, they are taken equal to 0.5.

Then, when the number of components is determined, it is necessary to start training the parameters of the mixture model. To do this, we can use the EM algorithm, for which there are variations [27–29].

3.2. Gaussian Mixture Regression

To further explain how mixtures of Gaussian distributions can be used in parameters learning of Bayesian networks, we divide all continuous nodes into three groups: nodes without parents, nodes with continuous parents, nodes with discrete parents, nodes with discrete and continuous parents. If a continuous node has no parents, then it learns the usual mixture of Gaussian distributions. If a node has only discrete parents, it learns its mixture of Gaussian distributions for each combination of values of discrete parents. If a continuous node has no parents, then it learns the usual mixture of Gaussian distributions. If a node has only discrete parents, it learns its mixture of Gaussian distributions for each combination of values of discrete parents. However, if a continuous node has a continuous parent node, then it becomes necessary to learn regression. When using mixtures, the best option may be to use a Gaussian regression mixture [30]. Initial marginal parameters of the i-th component, for example, in BN with two continuous nodes $y_1 \rightarrow y_2$:

$$\mu_i = \begin{bmatrix} \mu_{y_1,i} \\ \mu_{y_2,i} \end{bmatrix}, \Sigma_i = \begin{bmatrix} \sigma^2_{y_1,i} & \sigma_{y_1 y_2,i} \\ \sigma_{y_1 y_2,i} & \sigma^2_{y_2,i} \end{bmatrix} \tag{8}$$

However, when we sample from node y_2, with a specific value in node y_1, we need to calculate the conditional mathematical expectation and variance for each component i:

$$y'_{2,i} = \mu_{y_2,i} + \left(\frac{\sigma_{y_1,y_2,i}}{\sigma^2_{y_1,i}}\right)(y_1 - \mu_{y_1,i}) \tag{9}$$

$$\sigma^2_{y'_{2,i}} = \frac{\sigma^2_{y_2|y_1,i}}{n_i}\left(1 + \frac{(y_1 - \mu_{y_1,i})^2}{\sigma^2_{y_1,i}}\right) \tag{10}$$

where n_i is the number of elements in the component. Conditional mathematical expectation and variance for the i-th component are summed up according to the weights:

$$y'_2 = \sum_{i=1}^{N} w_i y'_{2,i} \tag{11}$$

$$\sigma^2_{y'_2} = \sum_{i=1}^{N} w_i \sigma^2_{y'_{2,i}} \tag{12}$$

Figure 2 shows an example of regression for a two-dimensional mixture with three components.

3.3. Algorithm for Parameter Learning and Inference with GMM

Now, let us finalize the algorithm for parameters learning of the Bayesian network using GMM. Two approaches were chosen to determine the number of components—based on AIC and BIC since these approaches turned out to be the fastest in time. However, since the AIC often overestimates the number of components and the BIC underestimates, in the end, we use both estimates and take the average value rounded to the nearest integer. Algorithm 1 shows the final algorithm for parameters learning in continuous

nodes. Algorithm 2 shows the final algorithm for inference in continuous nodes based on the parents' values. The learning algorithm takes as input a data set in the form of tabular data. First, the structure is built using the Hill-Climbing algorithm [31]. Then the resulting structure and dataset go into the parametric learning procedure, in which all continuous distributions are approximated using mixtures of Gaussian distributions (Section 3.1). Further, in order to produce probabilistic inference from the already trained network in continuous nodes that have parents, regression based on mixtures of Gaussian distributions is used (Section 3.2). This approach is somewhat slower than the classical Gaussian or discretization based approaches, since more time is spent estimating mixture distributions ($O(NDK)$ where N—number of points in data, D—number of dimensions of GMM, K—number of components of GMM), but the learning algorithm can be paralleled, since distributions are estimated at each node independently.

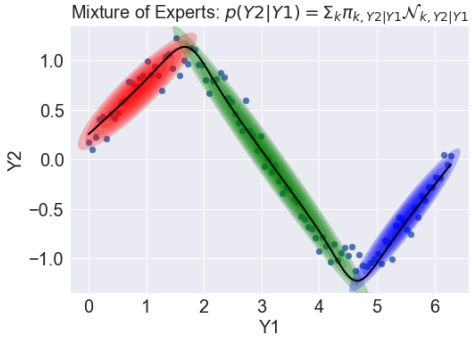


Figure 2. An example of GMR for a two-dimensional mixture with three components.

Algorithm 1 Structure and Parameters learning for continuous nodes in BNs with GMM

procedure STRUCTURE LEARNING(D)
  Input: $D = \{ \mathbf{x}_1, ..., \mathbf{x}_n, ..., \mathbf{y}_1, ..., \mathbf{y}_n \}$
  Output: $\{ V, E \}$
  $\{ V, E \}$ = HillClimbing (D, *start_dag*)
  return $\{ V, E \}$
procedure PARAMETERS LEARNING(D, $\{ V, E \}$)
  Input: $D = \{ \mathbf{x}_1, ..., \mathbf{x}_n, ..., \mathbf{y}_1, ..., \mathbf{y}_n \}$, BN structure $\{ V, E \}$
  Output: dictionary with distributions parameters for each node in BN
  params = empty dictionary
  for *node* in *BN_structure* **do**
    if *node* is continuous **and** parents(*node*) are empty **then**
      ncomponents = (BIC(*node*, *D*) + AIC(*node*, *D*)) / 2
      m = GMM(*ncomponents*).fit(*node*, *D*)
      params[*node*] = $\{ m.means, m.covariances, m.weights \}$
    if *node* is continuous **and** parents(*node*) are continuous **then**
      nodes = [*node*, *parents*(*node*)]
      ncomponents = (BIC(*nodes*, *D*) + AIC(*nodes*, *D*)) / 2
      ml = GMM(*ncomponents*).fit(*nodes*, *D*)
      params[*node*] = $\{ m.means, m.covariances, m.weights \}$
  return *params*

Algorithm 2 Inference for continuous nodes in BNs with GMM

procedure INFERENCE(*parents_values*)
  Input: *parents_values* = $[x_1, ...x_n, ...y_1, ..., y_n]$
  Output: sampled value in a node
  sampled_value = 0
  conditional_distribution = take *parameters* from *BN_parameters* by discrete parents values $[x_1, ...x_n]$
  mean = *conditional_distribution*[*mean*]
  covariances = *conditional_distribution*[*covariances*]
  w = *conditional_distribution*[*weights*]
  model = GMM(*n_components* = length(*w*), *means* = *mean*, *covariances* = *covariances*, *weights* = *w*)
  sampled_value = model.conditional_sampling(continuous parents values $[y_1, ..., y_n]$)
  return *sampled_value*

4. Classification Models in Parameters Learning

In general, Bayesian network structure search methods allow continuous variables to be parents of discrete variables. The main problem arises at the parameters learning stage since such pairs of parents and children require a model estimating the discrete distribution concerning some continuous data. Furthermore, most applications also require the ability to predict discrete values from parental data. Classification models handle this well. Algorithm 3 shows the algorithm for parameters learning in discrete nodes using a classifier. Algorithm 4 shows the algorithm for inference in discrete nodes based on parents values.

Next, we describe the classifiers implemented in the BAMT library [32] and used for experiments. The main focus will be on how these models can obtain distributions since this is not the most common way to use them.

4.1. Logistic Regression

This multiclass classification method uses a softmax function to estimate the probability of a conditional discrete distribution.

This is a linear model, where for each kth of K classes, the vector W_k and the free factor $W_{k,0}$ are estimated, and the probability of this class for the parent vector X is given by the formula:

$$p_k(X) = \frac{\exp\left(XW_k + W_{0,k}\right)}{\Sigma_{l=1}^{K} \exp\left(XW_l + W_{0,l}\right)} \tag{13}$$

As any linear model, it cannot always reliably approximate the real situation. All classifier implementations from this and below are taken from the sklearn package in Python.

4.2. Linear Discriminant Analysis (LDA)

This model describes class k through Gaussian distributions with mean μ_k and covariance matrix Σ_k. It is assumed that the covariance matrices coincide and are equal to some Σ. The linear model from these parameters and the marginal distribution approximates $\log P(y = k|X)$ from which the conditional discrete distribution is easily obtained.

Similar to the model above, it has a certain level of linearity, so, for example, $\log P(y = k|X)$ is given by the following formula:

$$\log P(y = k|X) = W_k^t X + W_{0,k} + \text{Const} \tag{14}$$

where vector W_k and free factor $W_{k,0}$ are calculated using mean μ_k and covariance matrix Σ_k.

Algorithm 3 Parameter learning for discrete nodes in BNs with a classifier

procedure PARAMETERS LEARNING(D, $\{ V, E \}$, classifier)
  Input: $D = \{ \mathbf{x}_1, ..., \mathbf{x}_n, ..., \mathbf{y}_1, ..., \mathbf{y}_n \}$, BN structure $\{ V, E \}$
  Output: dictionary with distributions parameters for each node in BN
  params = empty dictionary
  for *node* in *BN_structure* **do**
    if *node* is discrete **and** parents(*node*) are empty **then**
      $params[node]$ = discrete_distribution_from($node$, D)
    if *node* is discrete **and** parents(*node*) are discrete **then**
      $nodes = [node, parents(node)]$
      node_params = $\emptyset$
      combinations = all combinations of $parents(node)$ values
      for $\{ v_1, ..., v_k \}$ in *combinations* **do**
        $subsample = \{ \mathbf{x}_i : x_{ij_1} = v_1, ..., x_{ij_k} = v_k \}$
        $node\_params \cup \{ \{ v_1, ..., v_k \} :$ conditional_probability_table$(subsample) \}$
      $params[node] = node\_params$
    if *node* is discrete **and** parents(*node*) are continuous **then**
      $nodes = [node, parents(node)]$
      m = classifier.fit($nodes$, D)
      $params[node] = \{$ serialization$(m) \}$
    if *node* is discrete **and** parents(*node*) are continuous **and** discrete **then**
      cont_parents = parents_continuous(*node*)
      disc_parents = parents_discrete(*node*)
      $nodes = [node, cont\_parents]$
      node_params = $\emptyset$
      combinations = all combinations of *disc_parents* values
      for $\{ v_1, ..., v_k \}$ in *combinations* **do**
        $subsample = \{ \mathbf{x}_i : x_{ij_1} = v_1, ..., x_{ij_k} = v_k \}$
        m = classifier.fit($nodes$, $subsample$)
        $node\_params \cup \{ \{ v_1, ..., v_k \} :$ serialization$(m) \}$
      $params[node] = node\_params$
  return *params*

Algorithm 4 Inference for discrete nodes in BNs with classifier

procedure INFERENCE($parents\_values$)
  Input: $parents\_values = [x_1, ...x_n, ...y_1, ..., y_n]$
  Output: sampled value in a node
  $serialized\_model$ = take *parameters* from *BN_parameters* by discrete parents values $[x_1, ...x_n]$
  $model$ = deserialization($serialized\_model$)
  $conditional\_distribution$ = model.predict_proba (continuous parents values $[y_1, ..., y_n]$)
  $sampled\_value$ = choice($conditional\_distribution$, $model.classes$)
  return $sampled\_value$

4.3. K-Nearest Neighbors (kNN)

In the classical approach, the basic idea is that an object is assigned the class that is the most common or closest on average among the k neighbours of a given element, which are taken from the training dataset and for which the classes are known. Discrete distributions are also obtained from the frequency or ratio of the mean distance to all averages for each class among the k nearest neighbours. This model is quite flexible due to the possibility

of selecting the metric and the number of neighbours and non-linearity. However, it may cause problems with overtraining.

4.4. Decision Tree

The decision tree recursively partitions the feature space in steps relative to some value to group objects from the same class. The conditional discrete distribution for this method is the proportions of class representation in the corresponding decision tree leaf. This model has a risk of overtraining, and a rather large number of hyperparameters, which in the experiments are chosen by default according to the sklearn package implementation.

4.5. Random Forest

A method that trains multiple decision trees on different subsamples of data and uses averaging of their results. The conditional discrete distribution is the average of the distributions of these trees. The problems of this model are similar to the model above, although it has more flexibility.

5. Experiments

5.1. Synthetic Data

5.1.1. Parameters Learning on Different Kinds of GMM

Some experiments were carried out on synthetic data to identify the features and limitations of mixtures for parameters learning. The primary purpose of these experiments is to answer for which data the use of mixtures increases modelling quality.

We should select a metric to compare various mixtures with each other. The most obvious choice is the degree of cohesion of the mixture's components, which indicates how close the components are to each other. It is possible to calculate such a metric, for example, using the Mahalanobis distance or using a linear discriminant [33]. However, these metrics are highly dependent on the generated values since they are calculated on points. Therefore, we have chosen a metric that estimates the rate of overlap of the components (OLR) [34]. In essence, this level is calculated as the ratio of the probability at the lowest point (saddle) to the probability at the peak point. Thus, the maximum distance between the components will correspond to OLR equal to 0, and the minimum distance between the components—OLR equals 1. The Jensen–Shannon divergence was chosen to assess the quality of modelling [35]. It was decided to use the Jensen–Shannon divergence, since it has the property of symmetry and its values are normalized from 0 to 1, which is quite convenient when comparing the results.

For the experiment, two simple structures were generated—two-dimensional and three-dimensional random variables (Figure 3).

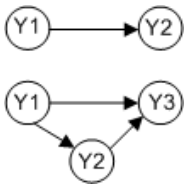


Figure 3. Structures for a synthetic experiment. All nodes are continuous.

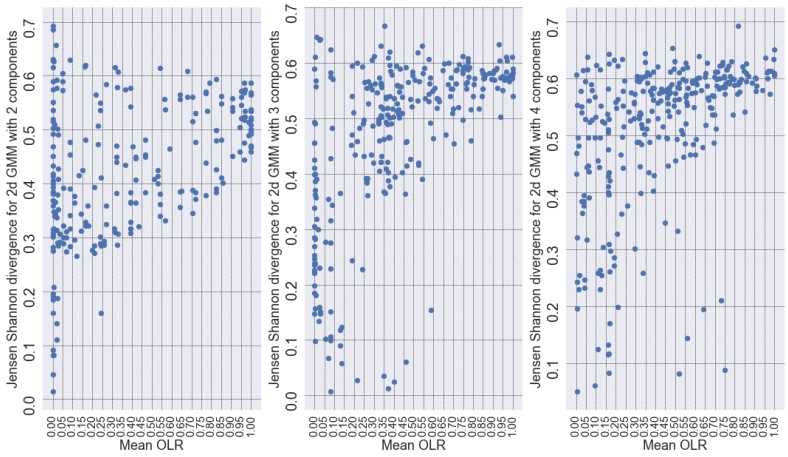
Overall, 300 different mixtures were generated; the means ranged from 0 to 20, the variance ranged from 1 to 20. The experiment steps were the following:

1. For each combination of parameters, a data sample was generated from the mixture;
2. The network parameters were trained on the sample;
3. Then a sample was generated from the Bayesian network, and the mixture parameters were trained on it;
4. The mixtures were compared, and the divergence between them was calculated.

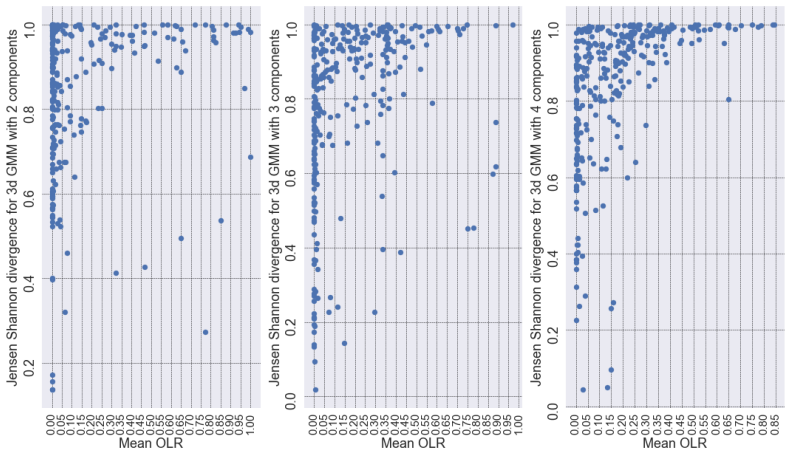
Next, to compare the simulation based on mixtures and based on a single Gaussian distribution, the following experiment was performed:

1. For each combination of parameters, a sample was generated from the mixture;
2. The sample was divided into train and test in a ratio of 90 to 10%;
3. The network parameters were trained on the train in two ways—based on one Gaussian distribution and based on GMM;
4. For each observation in the test, the value in each node was removed sequentially;
5. The deleted value in a node was restored as an average value after sampling in this node;
6. The root mean square error (RMSE) of restoration was calculated for two approaches—based on one Gaussian distribution and based on mixtures.

Figure 4 shows the results of the first experiment. It can be seen that the divergence grows along with the decrease in the distance between the components. Figure 5 shows by how what percent the average RMSE decreases when we use mixtures for training and recovery; the percentage is calculated relative to RMSE obtained during training and recovery using the Gaussian distribution. It can be seen that, on average, the use of mixtures reduces the error by 40 percent, and this percentage grows with an increase in the distance between the components and an increase in the number of components.

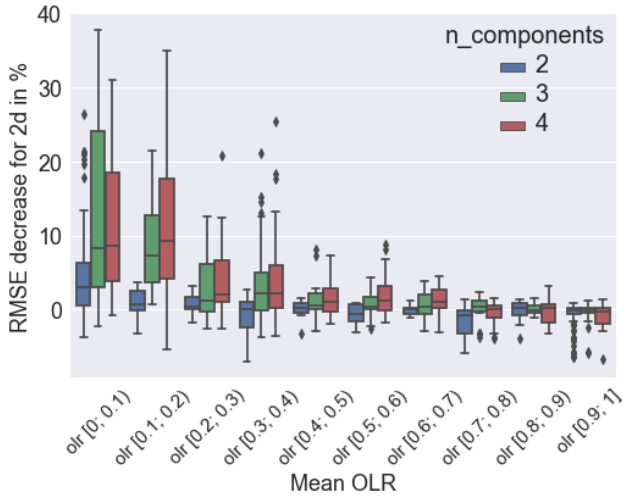


(**a**)

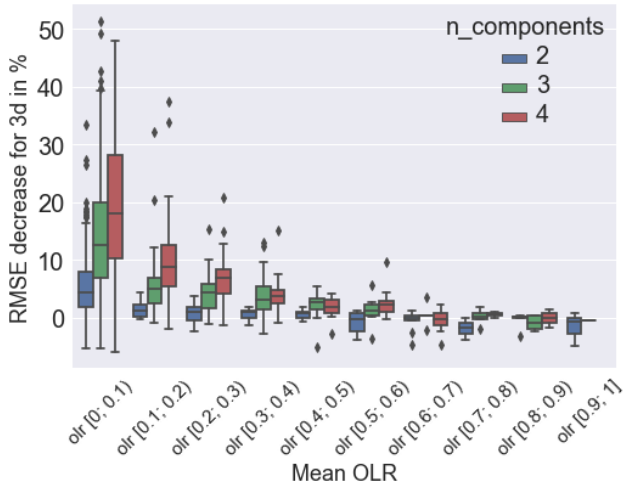


(**b**)

Figure 4. Results of experiments to identify the relationship between the quality of modelling using GMM and the characteristics of the dataset for two-dimensional (**a**) and three-dimensional (**b**) cases with a different number of components.



(**a**)



(**b**)

Figure 5. Reducing the average RMSE of all nodes as a percentage when using GMM for two-dimensional (**a**) and three-dimensional (**b**) cases. A negative value indicates that GMM is outperformed by one Gaussian distribution.

5.1.2. Parameters Learning with Classifiers from a GMM Perspective

The following experiment on synthetic data is conducted to determine the effect of the degree of cohesion on parameters learning with the classifier. The primary purpose of this experiment is to answer on which data the use of classifiers improves the modelling quality of discrete data and how much it degrades it for continuous data.

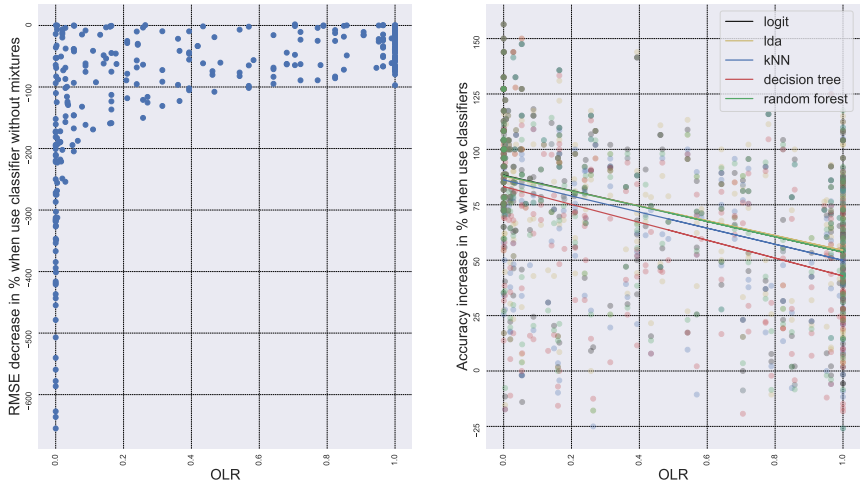
The structures generated for the experiment are two-dimensional random variables of a continuous value and the corresponding component number. Thus, they are related, but it is unclear how the edge between them should be directed.

There were generated 300 different mixtures; the means ranged from 0 to 15, the variance ranged from 1 to 10, each component was taken in equal proportions. Then, the following experiment steps were completed:

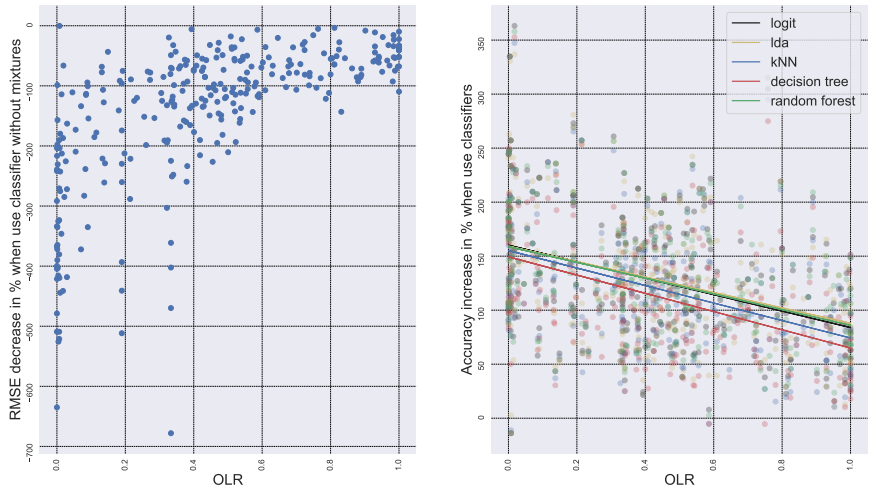
1. For each combination of parameters, a data sample was generated from the mixture;
2. OLR was calculated on the true parameters of the mixtures;
3. The sample was divided into train and test in a ratio of 90 to 10%;

4. The network parameters of two simple networks from an edge oriented one way or the other were trained on training sample by the classical method without the use of mixtures;
5. For each observation in the test, the value in each node was removed sequentially;
6. The deleted value in a continuous node was restored as an average value after sampling in this node, and the discrete value as the mode;
7. RMSE and restoration accuracy were calculated for two approaches—based on an edge from the discrete node to the continuous node with a conditional Gaussian model and an inverted direction with a classifier from the list above.

Figure 6 shows how much the RMSE degrades when we do not use mixtures for training and additional information about the component from the associated discrete variable. Moreover, it shows how the degree of cohesion affects accuracy. The coloured lines on the accuracy graph represent curves obtained by linear regression for each classifier. It should be noted that this has the same effect on methods assuming some Gaussianity of the data and on different ones. In the long run, the mixtures and OLR estimation on them can be used to decide on the network's topology.



(**a**)



(**b**)

Figure 6. A decrease in the RMSE of a continuous node and an increase in the accuracy in determining the components when the edge is reversed for the two-component (**a**) and three-component (**b**) cases. A negative RMSE value indicates how much the estimate is degraded.

It should be noted that such a strong influence is caused by the fact that we cannot use data from children for estimation. However, an algorithm using such data in hybrid networks will inevitably require edges directed from continuous to discrete.

5.2. Real Data

5.2.1. Using GMM for Parameters Learning on Real Data

Two datasets from the industry were selected to conduct experiments on real data. The main requirement for data is the presence of both discrete variables and continuous ones since we would like to test the ability of GMM to work on mixed data. The first dataset is from the oil and gas sector. It represents data on 442 reservoirs, their geological parameters (nine parameters), including four discrete variables (for example, period) and five continuous variables (for example, porosity). The second dataset represents the data of 3000 bank customers and includes four discrete variables (for example, gender) and four continuous variables (for example, the size of a transaction per month). Since these are real data, there are no reference networks for it. At the same time, it is clear that structural learning and the resulting structure can affect the quality of modelling, so we will compare the quality of modelling for different structures obtained by different structure learning methods on mixed data [31]. In these experiments, a comparison is drawn with classical approaches to parametric learning of Bayesian networks, since more modern approaches, unfortunately, do not have an open and easy-to-use software implementation.

To validate the results, we will use leave-out validation since there are not a lot of data. Thus, the experiment contains the following steps:

1. Structural learning;
2. Parameter learning in three ways—based on one Gaussian distribution [36], based on mixtures and based on discretization [37];
3. Then, we take a test sample, sequentially delete the values in continuous nodes and restore them using sampling from the network in two ways—based on one Gaussian distribution and based on mixtures;
4. Calculate the quality of restoration as RMSE.

Tables 1 and 2 show the recovery results for all continuous dataset variables, the best result for the parameter among all training methods is highlighted in bold. Thus, it can be seen that, firstly, GMM performs better on all structure learning methods, and secondly, all the best results for the parameters are also obtained when using mixtures. This confirms the stability of the proposed mixture-based parametric learning approach. It can be seen that sometimes the gain in quality is not very large, however, in the given subject areas that are shown in the experiments, such an increase in quality is significant, moreover, this can be explained by the weak distinguishability of the components in the data, while with good distinguishability, the gain is about 20 % (Section 5.1.1).

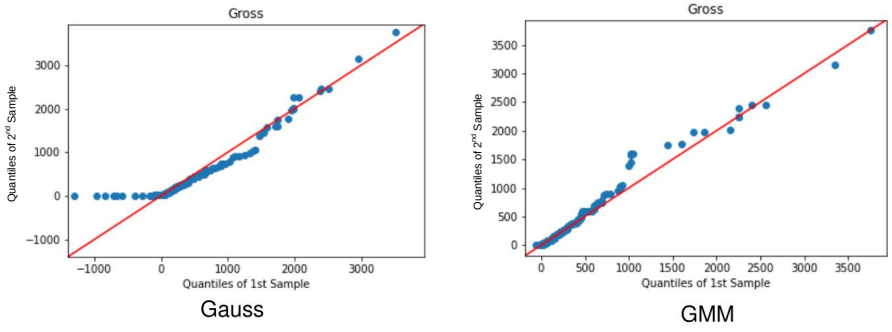
Additionally, for the geological dataset, an experiment was conducted in which three continuous features (Gross, Permeability and Depth) were selected, for each parameter, sampling was performed from the network based on the Gaussian distribution and based on mixtures, then the resulting samples were compared with the original distributions using qq-plots. For this experiment, a network was chosen that was built using the AIC evaluation function. It can be seen (Figure 7) that modelling using mixtures of distributions makes it possible to more accurately reproduce the original distributions than the approach based on the Gaussian distribution does, since it is clear that some quantiles are poorly reproduced.

Table 1. RMSE of restoration for continuous nodes with different methods of structural learning (BIC, AIC, LL, MI) and parameter learning methods for oil and gas parameters.

Node	LL			BIC		
	GMM	Gauss	Discrete	GMM	Gauss	Discrete
Gross	**438.01**	492.7	1137.1	**409.1**	441.7	879.2
Netpay	**85.1**	93.4	278	**88.1**	90.4	172.6
Porosity	**7.1**	7.1	13.9	**5.8**	5.9	8.85
Permeability	**990.5**	1103.2	2356.7	**1058**	1117.2	2450.2
Depth	**1058.8**	1063.1	1191.1	**990.7**	993.1	1110.3
Node	**AIC**			**MI**		
	GMM	**Gauss**	**Discrete**	**GMM**	**Gauss**	**Discrete**
Gross	**436.5**	492.3	853.4	**409.1**	441.5	1137.03
Netpay	92.1	**91.9**	172.4	**87.4**	90.4	277.9
Porosity	**5.8**	5.9	9.08	5.9	**5.8**	13.9
Permeability	**989.4**	1103.7	1786.3	**1038.7**	1114.5	2356.7
Depth	1035.4	**1034.5**	1082.9	1035.1	**1033.7**	1187.5

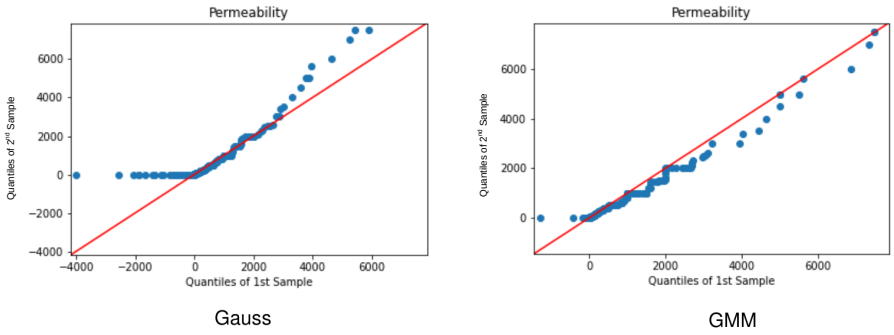
Table 2. RMSE of restoration for continuous nodes with different methods of structural learning (BIC, AIC, LL, MI) and parameters learning methods for bank customers' data.

Node	LL			BIC		
	GMM	Gauss	Discrete	GMM	Gauss	Discrete
Mean_tr	**6194**	6722	24,502	**6219.5**	6787.1	23,871.1
Median_tr	**5555.8**	6148.4	23,821.7	**5559.5**	6182.9	24,876.1
Tr_per_month	**22.6**	22.7	116.8	**21.2**	22.3	114.3
Node	**AIC**			**MI**		
	GMM	**Gauss**	**Discrete**	**GMM**	**Gauss**	**Discrete**
Mean_tr	**6225.1**	6793.1	24,501.2	**6213.5**	6710.6	24,510.3
Median_tr	**5569**	6178.8	24,123.5	**5594**	6143.2	23,728.6
Tr_per_month	22.7	**22.6**	114.8	**22.3**	22.4	117.1

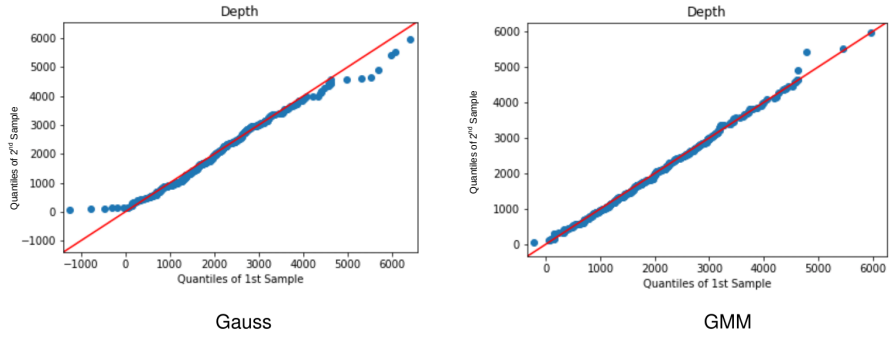


(**a**)

Figure 7. *Cont.*



(**b**)



(**c**)

Figure 7. Comparison of distributions sampled from the network and original distributions for three continuous geological features for two training methods—based on the classical Gaussian distribution and based on the proposed mixture algorithm (GMM).

5.2.2. Using Classifiers on Real Data

In addition to the two industry datasets mentioned above, two classification benchmarks were chosen to perform experiments on real data: the well-known Iris and Glass, which contains information about the types of glass and its various elements. Unfortunately, there is only one discrete variable in these datasets, so it is evident that classical hybrid networks with the prohibition of edges from continuous parents to discrete children are not very suitable here.

The first set represents parameters of 150 irises, including one discrete variable with three classes and four continuous variables (e.g., sepal length). The second dataset represents 214 glass samples and includes one discrete variable with seven classes (e.g., glass from utensils) and eight continuous variables (e.g., aluminium content).

To confirm the results, we calculate the recovery accuracy for each individual discrete parameter on the full set of samples from a dataset. Therefore, the experiment consists of the following steps:

1. Structural learning with and without prohibition for four different score-functions (LL, BIC, AIC and MI);
2. Parameter learning, and if there is no prohibition then for fully discretised data, where continuous data are replaced by five bins of discretisation, and for all available classifiers;
3. For each discrete parameter and each sample in a dataset, we remove value in the node and recover it by selecting the most frequent value from a conditional distribution estimated and described using a table or classifier;
4. Calculation of the quality of recovering as accuracy.

Figure 8 shows the reconstruction results for a classical network with a ban on edges from continuous nodes to discrete nodes (Blacklist), for a network without a ban, but with

data discretization (Full discrete) and for networks with classifiers. It can be seen that, firstly, discretization which allows to remove bans on connections cannot always improve quality. This justifies the search for other ways to solve this problem. The situation with discretization is more complicated because it is difficult to automatically select its parameters, which is why five equal bins were considered in the experiments for uniformity. Secondly, the existing solution using logistic regression also does not always improve the situation. This justifies an expansion of the list of classifiers under consideration. Additionally, from this list a more appropriate model for the distribution can be selected, which definitely improve the quality.

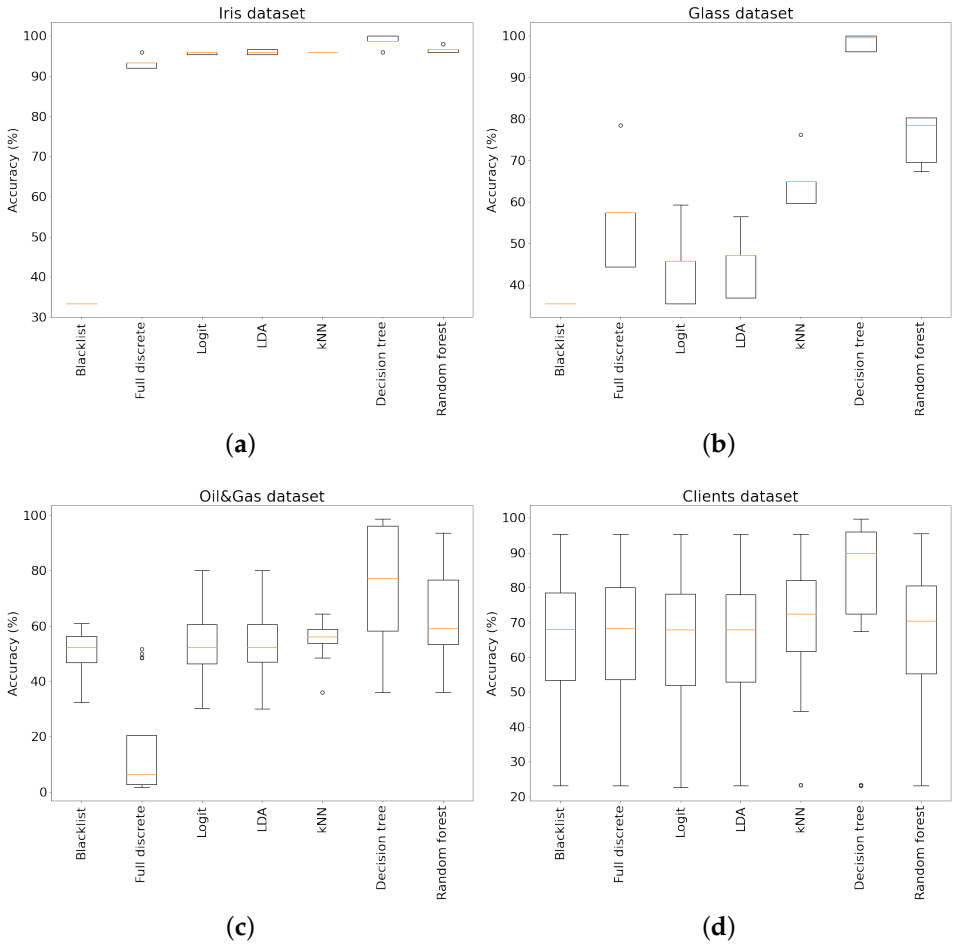


Figure 8. Experimental results comparing the quality of recovery in terms of accuracy of discrete variables for a classical network with prohibition of continuous parents for discrete children (Blacklist), for networks without prohibition but with discretization of continuous (Full discrete) and for different classifiers as ways of parametric description of dependencies. For Iris (**a**), Glass (**b**), Reservoir (**c**) and Clients (**d**) datasets. The closer the accuracy to 100 percent, the better the quality.

It should be clarified that for these datasets, the objective was purely accurate classification and prediction and the accuracy of the estimation of continuous variables is not considered.

6. Conclusions

The paper proposed solutions to two problems—parameter learning of Bayesian networks with non-Gaussian distributions and parameter learning for networks where "continuous parent–discrete child" relationships are essential. The first solution was based on the use of mixtures of Gaussian distributions. Experiments have shown the limitations in using this training method; in particular, this training gives a substantial increase in quality

(about 40%) on datasets with low cohesion of components. Additionally, experiments on real data showed that mixtures increase the quality of a model by an average of 15%, regardless of the structured learning method. The second solution was based on the use of classification models. Experiments have shown that on datasets where dependency logic requires connections with continuous parents and discrete children, using the proposed solution increases the quality of recovery by at least 30%. The proposed approaches to learning can work somewhat slower, but the slowdown is no more than one and a half times. In the future, it is planned to study the possibility of executing the proposed algorithms on the GPU in order to speed them up.

In the future, it is planned to research combining two solutions in one network and identifying for which data the joint use of both solutions will increase quality both on continuous nodes and on discrete nodes. It would also be interesting to investigate how modelling quality is improved when using such parameters learning on various elementary graph structures.

Experiments and data can be found in the repository in [32].

Author Contributions: Methodology, I.D., A.B. and A.V.K.; Software, I.D. and A.B.; Investigation, I.D.; Writing—original draft, I.D. and A.B.; Supervision, A.V.K.; Project administration, A.V.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Ministry of Science and Higher Education, grant number 075-15-2020-808.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Experiments and data can be found in the repository in [32].

Acknowledgments: This research is financially supported by the Ministry of Science and Higher Education, Agreement #075-15-2020-808.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kalinina, A.; Spada, M.; Burgherr, P. Application of a Bayesian hierarchical modeling for risk assessment of accidents at hydropower dams. *Saf. Sci.* **2018**, *110*, 164–177. [CrossRef]
2. Gehl, P.; D'ayala, D. Development of Bayesian Networks for the multi-hazard fragility assessment of bridge systems. *Struct. Saf.* **2016**, *60*, 37–46. [CrossRef]
3. Afenyo, M.; Khan, F.; Veitch, B.; Yang, M. Arctic shipping accident scenario analysis using Bayesian Network approach. *Ocean. Eng.* **2017**, *133*, 224–230. [CrossRef]
4. Koller, D.; Friedman, N. *Probabilistic Graphical Models: Principles and Techniques*; MIT Press: Cambridge, MA, USA, 2009 .
5. Neapolitan, R.E. *Learning Bayesian Networks*; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2004; Volume 38.
6. Deeva, I.; Bubnova, A.; Andriushchenko, P.; Voskresenskiy, A.; Bukhanov, N.; Nikitin, N.O.; Kalyuzhnaya, A.V. Oil and Gas Reservoirs Parameters Analysis Using Mixed Learning of Bayesian Networks. In Proceedings of the International Conference on Computational Science, Krakow, Poland, 16–18 June 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 394–407.
7. Cooper, G.F.; Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Mach. Learn.* **1992**, *9*, 309–347. [CrossRef]
8. Scutari, M. An empirical-Bayes score for discrete Bayesian networks. In Proceedings of the Conference on Probabilistic Graphical Models, Lugano, Switzerland, 6–9 September 2016, PMLR: Lugano, Switzerland: 2016; pp. 438–448.
9. Saputro, D.R.S.; Widyaningsih, P.; Handayani, F.; Kurdhi, N.A. Prior and posterior dirichlet distributions on bayesian networks (BNs). *AIP Conf. Proc.* **2017**, *1827*, 20036.
10. Dolera, E.; Favaro, S. Rates of convergence in de Finetti's representation theorem, and Hausdorff moment problem. *Bernoulli* **2020**, *26*, 1294–1322. [CrossRef]
11. Lauritzen, S.L. *Graphical Models*; Clarendon Press: Wotton-Under-Edge, UK, 1996; Volume 17.
12. Yin, J.; Li, H. A sparse conditional Gaussian graphical model for analysis of genetical genomics data. *Ann. Appl. Stat.* **2011**, *5*, 2630. [CrossRef]
13. Lerner, U.N. *Hybrid Bayesian Networks for Reasoning about Complex Systems*; Stanford University: Stanford, CA, USA, 2003.
14. Langseth, H.; Nielsen, T.D.; Rumí, R.; Salmerón, A. Inference in hybrid Bayesian networks. *Reliab. Eng. Syst. Saf.* **2009**, *94*, 1499–1509. [CrossRef]

15. Bishop, C.; Spiegelhalter, D.; Winn, J. VIBES: A variational inference engine for Bayesian networks. *Adv. Neural Inf. Process. Syst.* **2002**, *15*, 777–784.
16. Hanea, A.; Napoles, O.M.; Ababei, D. Non-parametric Bayesian networks: Improving theory and reviewing applications. *Reliab. Eng. Syst. Saf.* **2015**, *144*, 265–284. [CrossRef]
17. Ghosh, A.; Ahmed, S.; Khan, F.; Rusli, R. Process safety assessment considering multivariate non-linear dependence among process variables. *Process. Saf. Environ. Prot.* **2020**, *135*, 70–80. [CrossRef]
18. Astudillo, R.; Frazier, P. Bayesian optimization of function networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 14463–14475.
19. Monti, S.; Cooper, G.F. Learning hybrid Bayesian networks from data. In *Learning in Graphical Models*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 521–540.
20. Liu, H. Bayesian Networks and Gaussian Mixture Models in Multi-Dimensional Data Analysis with Application to Religion-Conflict Data. Ph.D. Thesis, Arizona State University, Tempe, AZ, USA, 2012.
21. Roos, J.; Bonnevay, S.; Gavin, G. Dynamic Bayesian networks with Gaussian mixture models for short-term passenger flow forecasting. In Proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), Nanjing, China, 24–26 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–8.
22. Cobb, B.R.; Shenoy, P.P. Inference in hybrid Bayesian networks with mixtures of truncated exponentials. *Int. J. Approx. Reason.* **2006**, *41*, 257–286. [CrossRef]
23. Shenoy, P.P.; West, J.C. Inference in hybrid Bayesian networks using mixtures of polynomials. *Int. J. Approx. Reason.* **2011**, *52*, 641–657. [CrossRef]
24. Rijmen, F. Bayesian networks with a logistic regression model for the conditional probabilities. *Int. J. Approx. Reason.* **2008**, *48*, 659–666. [CrossRef]
25. Sierra, B.; Lazkano, E.; Martínez-Otzeta, J.M.; Astigarraga, A. Combining Bayesian Networks, k Nearest Neighbours Algorithm and Attribute Selection for Gene Expression Data Analysis. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Cairns, Australia, 4–6 December 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 86–97.
26. McLachlan, G.J.; Rathnayake, S. On the number of components in a Gaussian mixture model. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2014**, *4*, 341–355. [CrossRef]
27. Verbeek, J.J.; Vlassis, N.; Kröse, B. Efficient greedy learning of Gaussian mixture models. *Neural Comput.* **2003**, *15*, 469–485. [CrossRef]
28. Nasios, N.; Bors, A.G. Variational learning for Gaussian mixture models. *IEEE Trans. Syst. Man Cybern. B* **2006**, *36*, 849–862. [CrossRef]
29. Pernkopf, F.; Bouchaffra, D. Genetic-based EM algorithm for learning Gaussian mixture models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2005**, *27*, 1344–1348. [CrossRef]
30. Cohn, D.A.; Ghahramani, Z.; Jordan, M.I. Active learning with statistical models. *J. Artif. Intell. Res.* **1996**, *4*, 129–145. [CrossRef]
31. Bubnova, A.V.; Deeva, I.; Kalyuzhnaya, A.V. MIxBN: Library for learning Bayesian networks from mixed data. *arXiv* **2021**, arXiv:2106.13194
32. BAMT. Repository Experiments and Data. Available online: https://github.com/ITMO-NSS-team/BAMT.git (accessed on 1 February 2021).
33. Tipping, M.E. Deriving Cluster Analytic Distance Functions from Gaussian Mixture Models. In Proceedings of the 1999 Ninth International Conference on Artificial Neural Networks ICANN 99, Edinburgh, UK, 7–10 September 1999; Volume 2, pp. 815–820.
34. Sun, H.; Wang, S. Measuring the component overlapping in the Gaussian mixture model. *Data Min. Knowl. Discov.* **2011**, *23*, 479–502. [CrossRef]
35. Menéndez, M.; Pardo, J.; Pardo, L.; Pardo, M. The jensen-shannon divergence. *J. Frankl. Inst.* **1997**, *334*, 307–318. [CrossRef]
36. Zhang, K.; Peters, J.; Janzing, D.; Schölkopf, B. Kernel-based conditional independence test and application in causal discovery. *arXiv* **2012**, arXiv:1202.3775
37. Agresti, A.; Min, Y. Effects and non-effects of paired identical observations in comparing proportions with binary matched-pairs data. *Stat. Med.* **2004**, *23*, 65–75. [CrossRef]