# Parameter Learning for Hybrid Bayesian Networks With Gaussian Mixture and Dirac Mixture Conditional Densities

**Peter Krauthausen** and **Uwe D. Hanebeck**

*Abstract*— **In this paper, the first algorithm for learning hybrid Bayesian Networks with Gaussian mixture and Dirac mixture conditional densities from data given their structure is presented. The mixture densities to be learned allow for nonlinear dependencies between the variables and exact closed-form inference. For learning the network's parameters, an incremental gradient ascent algorithm is derived. Analytic expressions for the partial derivatives and their combination with messages are presented. This hybrid approach subsumes the existing approach for purely discrete-valued networks and is applicable to partially observable networks, too. Its practicability is demonstrated by a reference example.**

## I. Introduction

Ever since their introduction, Bayesian Networks (BN) [1] are widely used for machine learning, robotics, control, and information fusion problems. As probabilistic graphical models, they allow for an intuitive modeling of causal relations along with a probabilistic description of the quality of the dependencies. Formally, a BN is an augmented directed acyclic graph $G = (E,V)$ consisting of nodes $v_i \in V$ and edges $e_{ij} \in E$. Nodes $v_i$ correspond to random variables $\mathbf{x}_i$ and edges $e_{ij} = (v_i, v_j)$ correspond to conditional densities $f(x_j|x_i)$. Even though BN are most often used for modeling dependencies between discrete-valued variables, they may be used to relate continuous variables [2], [3] or mixed-valued sets of variables [4], [5]. The use of hybrid BN is advantageous for many applications: they avoid discretization of continuous variables and allow an intuitive modeling of the problem at hand. This is especially the case, when nonlinear dependencies can be represented. Yet, the modeling freedom offered comes along with problems instantiating the BN. Typically, a BN is created on the basis of expert knowledge and known data cases. As the models and the available data amounts grow, the need for automated learning algorithms of BN parameters based on a rough initial estimate becomes increasingly relevant, in particular this includes dynamic systems.

This paper presents the first algorithm for learning the parameters of BN with Gaussian and Dirac mixture conditional densities [5] with given structure from data. This is a very powerful class of models, as it allows for modeling nonlinear dependencies between the variables, including discrete children of continuous parent nodes. The Dirac mixtures are employed to map the discrete random variables' symbols to the continuous domain, so that all calculations may be performed entirely in the continuous domain. Gaussian mixtures are used to model dependencies between continuous variables and a combination of Gaussian and Dirac mixtures is used for representing hybrid densities. The devised algorithm is capable of learning the parameters of this type of densities, in the case of fully and only partially observable node values, i.e., hidden nodes. The rest of this paper is structured as follows. Initially, related work will be presented and the specific type of hybrid BN will be introduced. After giving the mathematical problem formulation, the actual algorithm will be derived and its applicability demonstrated by a reference experiment [5].

P. Krauthausen und U. D. Hanebeck are with the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics, Karlsruhe Institute of Technology, Germany, {Peter.Krauthausen@kit.edu, Uwe.Hanebeck@ieee.org}.

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center SFB 588 on "*Humanoid robots – Learning and Cooperating Multimodal Robots*".

## II. Related Work

Parameter learning for BN has been an active research topic for long [6]. The existing approaches can be categorized according to full or partial observability of the nodes, the type of conditional density, Bayesian or frequentistic learning, and the specific algorithm applied. In the case of full observable data, conditional densities need to be learned from value assignments of the nodes and their parents. The learning problem decomposes over the network. In contrast, for partial observability, the values of some nodes are not available. In this case, learning the conditional densities is not independent of the rest of the network, as the values are replaced by corresponding posterior densities, obtained from standard BN inference. In each learning step, the network needs to be evaluated to obtain the posterior densities based on the new parameters of the entire network. The unobserved parts of the BN can be understood as an imposition of an inner structure facilitating the larger learning task. The parameters of the unobserved parts are learned, too.

Most research has been limited to discrete-valued BN only. The introduction of BN with *conditional Gaussian* densities [2], [7] led to the development of learning algorithms for hybrid BN [8]. Learning hybrid BN with mixture conditional densities has not received much attention so far. The most prominent approaches are *Hierarchical Mixture of Experts* (HME) [9] and *Mixture Density Networks* (MDN) [10]. Both are nonlinear function approximators augmented by probabilistic output. HME are trees of expert functions composing a nonlinear function, where each function is given a probabilistic interpretation. In contrast, MDN consist of nonlinear processing of input values by an artificial neural network whose outputs are the parameters of a conditional density mapping from the input to the output. This approach

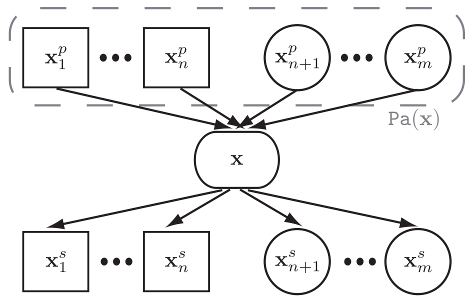


Fig. 1. Hybrid BN with variable $\mathbf{x}$, parent variables $\mathbf{x}_i^p$ and child variables $\mathbf{x}_i^s$. The square (circular) nodes correspond to discrete-(continuous-)valued variables and the rounded rectangle represents a variable that can be discrete or continuous.

is also capable of handling mixed-valued sets of variables. Yet, both approaches are only meant to be used in forward inference.

So far, no frequentistic or Bayesian learning algorithm for hybrid BN has been proposed for mixtures of the type of [5], [11] presented in more detail in Sec. III. Even though less general than the above approaches, this conditional density representation allows for closed-form analytic inference, like message passing. The proposed frequentistic learning algorithm of a point parameter estimate may serve as a preparatory work to Bayesian learning of BN with mixture densities, which is still an open field of research. In contrast to frequentistic learning, a Bayesian approach aims at determining a distribution over the parameter sets [12]. Regarding frequentist learning algorithms, approaches can be distinguished as using the *Expectation Maximization* algorithm [13] and gradient ascent [14], [15].

In this paper, a gradient ascent approach for learning the mixture densities is proposed. In Sec. V, the approach will be shown to subsume the known gradient approach for discrete-valued BN under certain assumptions.

## III. Hybrid Bayesian Networks

In this section, the type of hybrid BN to be learned is explicated and some definitions are given, e.g., the normal density $\mathcal{N}(x-\mu,\sigma) := \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\}$ and the Dirac delta function $\delta(x-\mu)$. The *parent* function $\mathtt{Pa}(x)$ returns the set of variables corresponding to nodes that are parents of the node representing $\mathbf{x}$. Using this, the shorthand $f(x|x_1^p,\ldots,x_m^p) = f(x|\mathtt{Pa}(x))$ is introduced, cf. Fig. 1. Realizations of random variables will be denoted by $\mathtt{i} \equiv (x=i)$. The posterior distribution of a variable $\mathbf{x}$ in the hybrid BN of [5], [11] is then given by

$$f(x) = \begin{cases} \sum_{i=1}^{|\mathbf{x}|} \alpha_i\,\delta(x-i) & \text{discrete } \mathbf{x}\,, \\ \sum_{i=1}^{M} \gamma_i\,\mathcal{N}(x-\mu_i,\sigma_i) & \text{continuous } \mathbf{x}\,. \end{cases} \quad (1)$$

In (1), the discrete events of the variables are mapped to integers $i$ in the continuous domain. This mapping is arbitrary but fixed. Assuming w.l.o.g. a node partition, as in Fig. 1, the conditional densities of the BN [5] are

$$\begin{aligned} f(x|\mathtt{Pa}(x)) &= \textstyle\sum_{j=1}^{M} \gamma_j\, f_j(\mathbf{x}) \prod_{k=1}^{m} f_{j,k}(x_k^p) \quad (2) \\ &= \textstyle\sum_{j=1}^{M} \gamma_j\, f_j(\mathbf{x}) \underbrace{\prod_{k=1}^{n} f_{j,k}(x_k^p)}_{\textbf{discrete}} \underbrace{\prod_{k=n+1}^{m} f_{j,k}(x_k^p)}_{\textbf{continuous}}\,. \end{aligned}$$

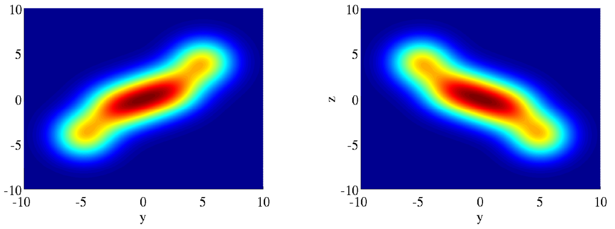


Fig. 2. Unnormalized hybrid cond. density: $f(\mathbf{z}|\mathbf{y},\mathbf{u}=1)$ and $f(\mathbf{z}|\mathbf{y},\mathbf{u}=2)$.

The structure of (2) is favorable for learning the parameters due to its decomposition in axis-aligned densities for $\mathbf{x}$ and each variable in $\mathtt{Pa}(\mathbf{x})$. Specializing (2) according to the type of densities is performed by substituting $f(.)$ with a normal density or a weighted sum of Dirac delta functions. To clarify the definitions, we explicate a modified example from [5].

In Ex. 1, $m$ is the number of normal densities necessary for describing the nonlinearity $f(\mathbf{z}|\mathbf{y},\mathbf{u}=i)$. The two columns of Tab. 1 correspond to the parameters for the two distinct densities, relative to the value of $\mathbf{u}$. Each value $\beta_{jk}$ corresponds to selecting the $j$-th component assigned from the value $\mathbf{u}=k$ of the discrete parent node. In case of more discrete-valued parent variables, the combination yields a value for the respective value combination of the parent nodes, i.e., $\mathtt{Pa}(x)=i$.

*Example 1 (Hybrid Conditional Density): Given the random variables* $\mathbf{y},\mathbf{z} \in \mathbb{R}$ *and* $\mathbf{u} \in \{1,2\}$*, an example hybrid conditional density for a nonlinearity similar to [5] is*

$$\begin{aligned} f(\mathbf{z}|\mathbf{y},\mathbf{u}) &= \textstyle\sum_{j=1}^{M} \gamma_j\, f_j(\mathbf{z})\, f_{j,u}(\mathbf{u})\, f_{j,y}(\mathbf{y}) \\ &= \textstyle\sum_{j=1}^{M} \gamma_j\, \mathcal{N}(z-\mu_j,\sigma_j) \\ &\quad \cdot \left[\textstyle\sum_{l=1}^{|\mathbf{u}|} \beta_{jl}\,\delta(u-l)\right] \mathcal{N}(y-\mu_{jy},\sigma_{jy}) \end{aligned}$$

*as depicted in Fig. 2, having the following parameters*

| $\mathbf{j \le m}$, $(\mathbf{u}=1)$ | $\mathbf{j > m}$, $(\mathbf{u}=2)$ |
|---|---|
| $\mu_j = \nu j + 0.75\cdot(\nu j)^3$ | $\mu_j = -\nu(j-m)$ $+0.75\cdot(-\nu(j-m))^3$ |
| $\mu_{jy} = \upsilon j - 5$ | $\mu_{jy} = \upsilon(j-m) - 5$ |
| $\beta_{j1} = 1$ | $\beta_{j1} = 0$ |
| $\beta_{j2} = 0$ | $\beta_{j2} = 1$ |

*and* $\sigma_j = \sigma_{jy} = 1.75$, $m=5$, $\upsilon = 10/m$, $M = m\cdot|\mathbf{u}|$, $\nu = 3/m$, *and* $\sum_{j=1}^{M}\gamma_j = D$*. Here, D is the size of the interval in which y values are obtained.*

In this section, the hybrid BN representation was introduced. We refer the interested reader to [5], [11] for details on inference with this model. Ex. 1 demonstrates the *roles* of the parameters to be learned.

## IV. Problem Formulation

The idea of learning a BN can be phrased as an optimization problem: given a data set $\mathcal{D} = \{d_1,\ldots,d_{|\mathcal{D}|}\}$, calculate the parameters $\theta$ of a given hybrid BN that are most likely to produce $\mathscr{D}$. Here, the data, the structure of the hybrid BN, and an initial set of parameters $\theta_0$ are assumed to be given. This means that for a hybrid Bayesian network the parameters of the Gaussian mixtures and Dirac mixtures in the conditional densities are to be estimated on the basis of data points. For setting the initial parameters of the Gaussian mixtures, i.e., the appropriate number of components and the weights, means and variances, advice can be found in [16]. Note the component positions in the Dirac mixtures correspond to symbols of discrete variables and are fixed points in the continuous domain. The component weights are the probabilities for the corresponding discrete events.

Regarding the data, each data point is a *full* or *partial* value assignment to the variables in the hybrid BN. In order to solve the optimization problem, the commonly used log-likelihood is applied as a performance criterion [6], [14], [15]. It is given by

$$L_{\mathscr{D}}(\theta) \equiv \tfrac{1}{|\mathscr{D}|} \log f(\mathscr{D}|\theta) = \tfrac{1}{|\mathscr{D}|} \sum_{d \in \mathscr{D}} \log f(d|\theta) . \qquad (3)$$

## V. Learning Algorithm

In order to find the maximum of $L_{\mathscr{D}}(\theta)$, we propose a gradient ascent method. In this section, we introduce the overall learning algorithm. A detailed derivation can be found in Sec. VI. The following constraints need to be kept so that the results are valid conditional densities

$$f(\mathtt{a}|\mathtt{b}) \geq 0 , \qquad \int f(x|\mathtt{b})\, \mathrm{d}x = 1 . \qquad (4)$$

These constraints postulate conditional probabilities to be positive and the integral of the conditional density $f(x|\mathtt{b})$ to equal 1 for any given $\mathtt{Pa}(x) = \mathtt{b}$. For conditional densities with continuous $\mathbf{x}$, the samples are assumed to be restricted to $\mathtt{b} \in [b_{min}, b_{max}]$. The latter constraint is then approximated by $\sum_{j=1}^{M} \gamma_j = (b_{max} - b_{min}) =: D$. Note that the approximate constraint will require a normalization for a fixed value of $\mathtt{Pa}(x)$ *ex post* in general. To meet the constraints for the weights, the penalty $P(\theta_k) := \exp(-(\gamma_j/c)) + \exp(-(\sum_{j=1}^{M} \gamma_j - D)/c))$ and for the discrete probabilities $C(\theta_k)$ [15]

$$C(\theta_k) := \begin{cases} \sum_{o=1}^{|\mathbf{x}|} \frac{\partial}{\partial \alpha_{k,oi}} L_{\mathscr{D}}(\theta_k) & \theta_{k,i} = \alpha_{k,ij} , \\ 0 & \text{else} , \end{cases} \qquad (5)$$

were added to the target function. Note that for all parameters $\lambda \in \theta$, except for $\alpha_{ij}$ and $\gamma_j$, the constraint terms are zero, yielding the following update equation

$$\theta_{k+1} = \theta_k + \eta \left( \tfrac{\partial}{\partial \theta} L_{\mathscr{D}}(\theta_k) - C(\theta_k) \right) - \tfrac{\partial}{\partial \theta} P(\theta_k) . \qquad (6)$$

The update equation (6) comprises two intuitive factors: the first may be understood as an update on the current parameter estimate, while the other asserts the constraints. Alg. 1 summarizes this section. In Sec. VI, the analytic expressions for $\frac{\partial}{\partial \theta} L_{\mathscr{D}}(\theta_k)$ used in Alg. 1 will be derived.

**Algorithm 1** Learning Hybrid Bayesian Networks

```
Input: Hybrid BN, $\theta_{k=0}$ initial parameters, $\mathscr{D}$ data set
 1: repeat
 2:   for all families $\{\mathbf{x}, \mathtt{Pa}(\mathbf{x})\}$ do
 3:     for all $d \in \mathscr{D}$ do
 4:       Set BN values according to $d$
 5:       // Inference mechanism of choice
 6:       Calculate posterior probabilities
 7:       Calculate relative factors and constraints for $\lambda$
 8:     end for
 9:     $\theta_{k+1} = \theta_k + \eta \frac{\partial}{\partial \theta} (L_{\mathscr{D}}(\theta_k) - C(\theta_k)) - \frac{\partial}{\partial \theta} P(\theta_k)$
10:   end for
11: until $\Delta(\theta_{k+1}, \theta_k) \approx 0$
```

## VI. Analytic Expressions for the Derivatives

Due to the additivity of (3), the gradient $\frac{\partial}{\partial \lambda} L_{\mathscr{D}}(\theta)$ may be decomposed with respect to the parameters of the conditional densities per data point $d$. Setting $f(d) := f(d|\theta)$, omitting the normalization, and applying the chain rule gives

$$\tfrac{\partial}{\partial \lambda} L_{\mathscr{D}}(\theta) = \sum_{d \in \mathscr{D}} \tfrac{\partial}{\partial \lambda} \log f(d) = \sum_{d \in \mathscr{D}} \underbrace{\tfrac{1}{f(d)} \cdot \tfrac{\partial}{\partial \lambda} f(d)}_{\frac{\partial}{\partial \lambda} l_d(\theta)} . \qquad (7)$$

As has been shown in Ex. 1, the value combinations for the discrete-valued variables determine the continuous part of the conditional density. Given a fixed assignment for the discrete part, the gradient has to be determined for the parameters $\lambda$ governing the respective continuous density. For the rest of this section $\frac{\partial}{\partial \lambda} l_d(\theta)$ will be considered only and w.l.o.g. $\lambda$ implies fixed value combinations $\mathbf{x} = \mathtt{a}$ and $\mathtt{Pa}(x) = \mathtt{b} = \{\mathtt{b}_1, \ldots, \mathtt{b}_n\}$. *Parameter tying* is ignored and all integrations range over $(-\infty, +\infty)$. Like in [14], further simplification of $\frac{\partial}{\partial \lambda} l_d(\theta)$ gives

$$\begin{aligned} \tfrac{\partial}{\partial \lambda} l_d(\theta) &= \tfrac{\partial}{\partial \lambda} \int \cdots \int f(d, x, \mathtt{Pa}(x))\, \mathrm{d}x\, \mathrm{d}\mathtt{Pa}(x) \\ &= \int \cdots \int f(x, \mathtt{Pa}(x)|d) \tfrac{1}{f(x|\mathtt{Pa}(x))} \\ &\qquad \cdot \left( \tfrac{\partial}{\partial \lambda} f(x|\mathtt{Pa}(x)) \right) \mathrm{d}x\, \mathrm{d}\mathtt{Pa}(x) . \end{aligned} \qquad (8)$$

The first two factors in (8) correspond to the probability for a value assignment of $\{\mathbf{x}, \mathtt{Pa}(\mathbf{x})\}$ given the data divided by the probability of the conditional for this assignment. Note that the numerator equals the posterior probability obtainable from *junction tree* inference. If the posterior probability for the $\{\mathbf{x}, \mathtt{Pa}(\mathbf{x})\}$ is not given, it may be calculated from the *messages* passed to them and the conditional probability density. With a normalization $\psi$ and $d := \{\underline{d}, \overline{d}\}$, one obtains

$$\tfrac{f(x, \mathtt{Pa}(x)|d)}{f(x|\mathtt{Pa}(x))} = \psi f(\underline{d}|x) f(\mathtt{Pa}(x)|\overline{d}) . \qquad (9)$$

Here, $\underline{d}$ and $\overline{d}$ denote the data in the BN from below and above $\mathbf{x}$, cf. [5]. In the following, the partial derivatives for all parameters $\lambda$ will be derived. Combined with (9), these give rise to an analytic expression for (8).

## A. Partial Derivatives

The main idea behind calculating $\frac{\partial}{\partial\lambda} f(x|\mathtt{Pa}(x))$ is to calculate the derivatives for fixed discrete value combinations $\mathtt{b}$. For fixed $\mathtt{b}$, this yields

$$s_j(\mathtt{Pa}(\mathbf{x})) := \prod_{k=1}^{n} f_{j,k}(\mathtt{b}_k) = \prod_{k=1}^{n} \underbrace{\sum_{l=1}^{|\mathbf{x}_k^p|} \beta_{jkl}\,\delta(x_k^p - l)}_{=\beta_{jkb_k}\,\delta(x_k^p - \mathtt{b}_k)}, \tag{10}$$

which is only non-zero if $\mathtt{b}$ is attained. In this case, we define $\beta_{jkb_k} = 1$. The term $\frac{\partial}{\partial\lambda} l_d(\theta)$ simplifies for a fixed parameter $\lambda$, i.e., a fixed component $j$, and a fixed value $\mathtt{b}$ to

$$\frac{\partial}{\partial\lambda} f(x|\mathtt{Pa}(x)) = \\ \underbrace{\frac{\partial}{\partial\lambda}\, \gamma_j\, f_j(x)\, s_j(\mathtt{Pa}(x)) \left( \prod_{k=n+1}^{m} \mathcal{N}(x_k^p, \mu_{kj}, \sigma_{kj}) \right)}_{g_j(x|\mathtt{Pa}(x))}, \tag{11}$$

and the set of parameters for $\mathbf{x}$ and the continuous $\mathtt{Pa}(x)$ is

$$\lambda \in \{\, \gamma_j,\ \underbrace{\phi_{f_j}}_{\sim x},\ \underbrace{\mu_{n+1,j},\ \sigma_{n+1,j}, \ldots,\ \mu_{m,j},\ \sigma_{m,j}}_{\sim \text{cont.}\,\mathtt{Pa}(x)} \}\,. \tag{12}$$

The term $f_j(x)$ in (11) inherits the dependence on the type of $\mathbf{x}$ from the conditional density—$\phi_{f_j} := \{\mu_j, \sigma_j\}$ or $\phi_{f_j} := \{\alpha_j\}$. For all $\lambda$, (11) can be given in the following form

$$\frac{\partial}{\partial\lambda} f(x|\mathtt{Pa}(x)) = c_\lambda(x, \mathtt{Pa}(x))\, g_j(x|\mathtt{Pa}(x)) \tag{13}$$

with

$$c_\lambda(x, \mathtt{Pa}(x)) \in \left\{ 1/\gamma_j,\quad \frac{(\mathbf{x}-\mu)}{\sigma^2},\quad \frac{1}{\sigma}\left[ \left( \frac{(\mathbf{x}-\mu)^2}{\sigma^2} \right) - 1 \right] \right\}.$$

The partial derivatives for $\mu$ and $\sigma$ need to be specialized for $\mu_j$, $\sigma_j$ or $\mu_{k,j}$, $\sigma_{k,j}$. For discrete $\mathbf{x}$, a fixed value gives

$$f_j(\mathtt{a}) := \sum_{i=1}^{|\mathbf{x}|} \alpha_{ij}\,\delta(x - \mathtt{a}) = \alpha_{\mathtt{a}j}\,\delta(x - \mathtt{a}) \tag{14}$$

which in turn gives rise to

$$c_\alpha(x, \mathtt{Pa}(x)) := \frac{1}{\alpha_{\mathtt{a}j}}\,. \tag{15}$$

This definition is necessary, as the chain rule cannot be applied in (8) for discrete probabilities.

## B. Combining Partial Derivatives with Messages

After calculating the partial derivatives, these need to be combined with the evidence (9) according to (8). The solution will be obtained in two steps: First, (8) will be simplified by calculating the product of the factors for *discrete (continuous)* $\mathtt{Pa}(x)$ only. Second, the terms involving $\mathbf{x}$ will be simplified. This involves distinguishing between *discrete (continuous)* $\mathbf{x}$. For a generic parent set, one obtains

$$f(\mathtt{Pa}(x)|\overline{d}) \;=\; \left( \prod_{i=1}^{n} f_d(x_i^p|\overline{d}) \right) \cdot \left( \prod_{i=n+1}^{m} f_c(x_i^p|\overline{d}) \right) \tag{16}$$

with the factors, cf. [5], [11],

$$f_d(x_i^p|\overline{d}) = \textstyle\sum_{l=1}^{|\mathbf{x}_i^p|} \overline{\beta}_{il}\,\delta(x_i^p - l)\,, \tag{17}$$

$$f_c(x_i^p|\overline{d}) = \textstyle\sum_{s=1}^{S} \gamma_s\, \mathcal{N}(x_i^p - \overline{\mu}_{si}, \overline{\sigma}_{si})\,. \tag{18}$$

Solving for all *discrete* $\mathtt{Pa}(x)$ corresponds to calculating

$$\int \cdots \int s_j(\mathtt{Pa}(x)) \left( \textstyle\prod_{i=1}^{n} f_d(x_i^p|\overline{d}) \right) \mathrm{d}x_1^p \ldots \mathrm{d}x_n^p \\ = \left( \prod_{i=1}^{n} \beta_{jib_i}\, \overline{\beta}_{il} \right) =: c_{d,ij}\,. \tag{19}$$

Note that $c_d$ is independent of the remaining variables. Solving for the *continuous* $\mathtt{Pa}(x)$ amounts to solving the following integral for each continuous $\mathbf{x}_i^p$

$$\int c_\lambda(x, \mathtt{Pa}(x))\, \mathcal{N}(x_i^p, \mu_{ij}, \sigma_{ij}^2)\, f_c(x_i^p|\overline{d})\, \mathrm{d}x_i^p =: c_{c,sij}. \tag{20}$$

Since $f_c(x_i^p|\overline{d})$ is a normal density, (20) results in a normal density with a constant factor. Solving (20) amounts to distinguishing the different cases for $\lambda$. Using the definitions $\eta_{sij} = \mathcal{N}(\mu_{ij} - \overline{\mu}_{si}, \sqrt{\sigma_{ij}^2 + \overline{\sigma}_{si}^2})$, $\mu_{sij} = \frac{\overline{\mu}_{si}\sigma_{ij}^2 + \mu_{ij}\overline{\sigma}_{si}^2}{\sigma_{ij}^2 + \overline{\sigma}_{si}^2}$ and $\sigma_{sij} := \sqrt{\frac{\sigma_{ij}^2 \overline{\sigma}_{si}^2}{\sigma_{ij}^2 + \overline{\sigma}_{si}^2}}$ , one obtains

$$c_{c,sij} = \eta_{sij} \cdot \begin{cases} \frac{1}{\sigma_{ij}^2}(\mu_{sij} - \mu_{ij}) & \lambda = \mu_{ij}\,, \\ \frac{1}{\sigma_{ij}^3}[(\mu_{sij} - \mu_{ij})^2 + \xi\,\sigma_{sij}^2 - \sigma_{ij}^2] & \lambda = \sigma_{ij}\,, \\ 1 & \text{else}\,. \end{cases}$$

The results reflect whether $\lambda$ is a parameter of the continuous $\mathbf{x}_i^p$ being integrated out or not. We observed the estimator reduces variances to extremely small values in the partially observable case. Based on empirical findings, the factor $\xi := 1$ was set to $\xi := 2$, which corrected this problem. Note, that this factor changes the optimization problem, is given without proof, and applicability in general is questionable. Inserting $c_{c,ij} = \sum_{s=1}^{S} \gamma_s\, c_{c,sij}$ and $c_d$ into (8) yields

$$\frac{\partial}{\partial\lambda} l_d(\theta) = \psi\, c_{d,ij}\, c_{c,ij} \int c_\lambda(x, \mathtt{Pa}(x))\, f(\underline{d}|x)\, f_j(x)\, \mathrm{d}x \tag{21}$$

with the following factors depending on the type of $\mathbf{x}$

$$f_d(\underline{d}|x) = \textstyle\sum_{t=1}^{M} \gamma_t \sum_{r=1}^{|\mathbf{x}|} \underline{\beta}_{tr}\,\delta(x - r)\,, \tag{22}$$

$$f_c(\underline{d}|x) = \textstyle\sum_{t=1}^{T} \gamma_t\, \mathcal{N}(x - \underline{\mu}_t, \underline{\sigma}_t)\,. \tag{23}$$

After some minor calculations, the following results are obtained for the different types of $\mathbf{x}$. For *discrete* $\mathbf{x}$, the integral in (21) simplifies to

$$\frac{\partial}{\partial\lambda} l_d(\theta) = \psi\, c_{d,ij}\, c_{c,ij} \sum_{t=1}^{M} \gamma_t\, \alpha_{\mathtt{a}j}\, \underline{\beta}_{t\mathtt{a}j} \tag{24}$$

and for *continuous* $\mathbf{x}$, one obtains

$$\frac{\partial}{\partial\lambda} l_d(\theta) = \psi\, c_{d,ij}\, c_{c,ij} \sum_{t=1}^{M} \gamma_t\, c_{\lambda,x,tij} \tag{25}$$

with

$$c_{\lambda,\mathbf{x},tij} = \eta_{tj} \cdot \begin{cases} \frac{1}{\sigma_j^2}(\mu_{tj} - \mu_j) & \lambda = \mu_j\,, \\ \frac{1}{\sigma_j^3}[(\mu_{tj} - \mu_j)^2 + \xi\,\sigma_{tj}^2 - \sigma_j^2] & \lambda = \sigma_j\,, \\ 1 & \text{else}\,, \end{cases}$$

and $\eta_{tj} = \mathcal{N}(\underline{\mu}_t - \mu_j, \sqrt{\underline{\sigma}_t^2 + \sigma_j^2})$, $\sigma_{tj} := \sqrt{\frac{\sigma_j^2 \underline{\sigma}_t^2}{\sigma_j^2 + \underline{\sigma}_t^2}}$ as well as $\mu_{tj} := \left( \frac{\underline{\mu}_t \sigma_j^2 + \mu_j \underline{\sigma}_t^2}{\sigma_j^2 + \underline{\sigma}_t^2} \right)$. The final expressions in (24) and (25) are analytic and allow for a simple implementation of Alg. 1, i.e., $\frac{\partial}{\partial \lambda} L_{\mathscr{D}}$ can be replaced by a summation over the above partial derivatives for all data according to (3).

## VII. Subsumption of the Discrete Case

In this section, the proposed approach is shown to subsume the results for BN of purely discrete-valued variables [14]. In this case, (2) simplifies to

$$f(\mathbf{x}|\mathrm{Pa}(\mathbf{x})) = \sum_{j=1}^{M} \gamma_j f_j(\mathbf{x}) \prod_{k=1}^{n} f_{j,k}(x_k^p)$$

$$= \sum_{j=1}^{M} \gamma_j \sum_{i=1}^{|\mathbf{x}|} \alpha_{ij}\, \delta(x-i) \prod_{k=1}^{n} \sum_{l=1}^{|\mathbf{x}_\mathbf{k}^\mathbf{p}|} \beta_{jkl}\, \delta(x_k^p - l)\,. \qquad (26)$$

The derivatives are considered with respect to a value combination $\{\mathtt{a}, \mathtt{b}\}$ only. In (26), $M$ is the total number of value combinations of the parent variables. Setting $\gamma_j = 1$, $\beta_{jkl} = 1$ for $l = b_k$, and $\alpha_{ij} = p_{\mathtt{a}|\mathtt{b}} = \mathrm{P}(\mathtt{a}|\mathtt{b})$, one may simplify (26)

$$f(\mathtt{a}|\mathtt{b}) = p_{\mathtt{a}|\mathtt{b}}\, \delta(x - \mathtt{a}) \prod_{k=1}^{n} \delta(x_k^p - \mathtt{b}_k)\,.$$

Using (15), one obtains

$$\frac{\partial}{\partial p_{\mathtt{a}|\mathtt{b}}} f(\mathtt{a}|\mathtt{b}) = \frac{1}{p_{\mathtt{a}|\mathtt{b}}} f(\mathtt{a}|\mathtt{b}) = \delta(x - \mathtt{a}) \prod_{k=1}^{n} \delta(x_k^p - \mathtt{b}_k)\,.$$

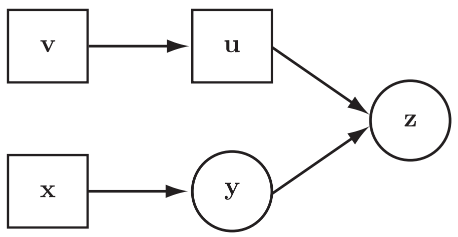
This result inserted into (8) yields

$$\frac{\partial}{\partial p_{\mathtt{a}|\mathtt{b}}} l_d(\theta) =$$
$$\int \cdots \int \frac{f(\mathtt{a},\mathtt{b}|d)}{f(\mathtt{a}|\mathtt{b})} \left( \frac{\partial}{\partial p_{\mathtt{a}|\mathtt{b}}} f(\mathtt{a}|\mathtt{b}) \right) \mathrm{d}\mathbf{x}\, \mathrm{d}\mathrm{Pa}(\mathbf{x}) = \frac{p_{\mathtt{ab}|\mathtt{d}}}{p_{\mathtt{a}|\mathtt{b}}}\,,$$

the default gradient approach for purely discrete BN [14].

## VIII. Limitations - Complexity, Identifiability, Convergence, and Bias

The learning algorithm is at least as complex as the inference algorithms for BN, as it is used as a subfunction. Inference for BN is known to be NP-hard [12] in general. Inference in hybrid BN is at least as complex as in the subsumed class of discrete-valued BN. In fact, it is known [4] that exact inference in polytree hybrid BN with Gaussian densities is NP-hard, even though the inference in polytree discrete BN takes only linear time. This renders learning hybrid BN with mixture densities for general graphs and polytrees NP-hard. Note that expensive computations due to complex Gaussian mixture densities may be alleviated by well known mixture reduction algorithms. From a practical standpoint, whenever BN inference is performed, part of the work of the proposed algorithm has been done already. The parameters may be adapted, as described in Alg. 1, using a batch of observations or single observations at a time. Regarding identifiability, the presented approach *inherits* all problems involved with learning mixture densities, i.e., identifiability only under certain conditions, e.g., up to a permutation [16]. Additionally, BN may be used for modeling *unsupervised learning* problems. A prominent example is *blind source separation* problem (BSS), i.e., an inverse to a nonlinear mixing of independent signals shall be learned. It is known that solving BSS is possible for certain sets of functions only [17]. It remains an open question, whether this approach yields favorable behavior for more complicated unsupervised problems with structure assumptions, e.g., multiple consecutive hidden nodes. Furthermore, convergence needs to be proven and is so far not guaranteed. As this is a nonlinear optimization problem, it can be expected that convergence to a *local* maximum within a restricted vicinity may be guaranteed at most. Due to all these restrictions, the proposed algorithm should be applied to the most reduced problem possible, e.g., in case parts of the net are fully observable it is preferable to decompose these parts. As mentioned in Sect. VI-B, a heuristic factor for correcting too small variances was introduced. For this a proof of correctness and generality needs to be performed. Furthermore, this change in the partial derivatives changes the optimization problem, so that the result of the estimator may only be close to the maximum likelihood estimator.



| $\mathbf{x}$ | $\mu_y$ | $\sigma_y$ |
|---|---|---|
| 1 | −5 | 2 |
| 2 | 5 | 2 |

,

| $\mathbf{v}$ | $P(\mathbf{u}=1\|\mathbf{v})$ | $P(\mathbf{u}=2\|\mathbf{v})$ |
|---|---|---|
| 1 | 0.2 | 0.8 |
| 2 | 0.6 | 0.4 |

.

Fig. 3. Hybrid network example similar to [5] and all $\gamma_i = 1$.

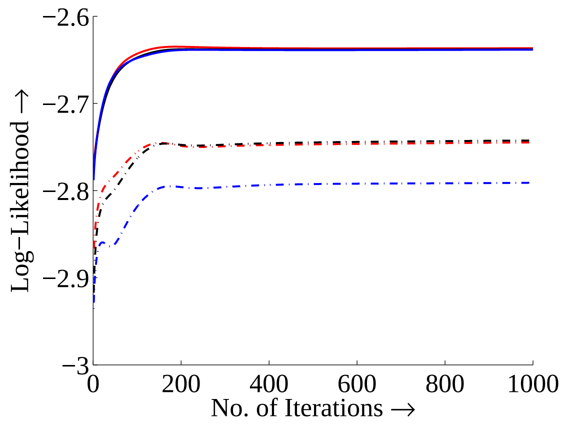


Fig. 4. Log-likelihood of the non-hidden nodes, $\mathbf{x}, \mathbf{v}$, and $\mathbf{z}$, of the BN on training (solid) and test (dashed) sets for no, small, and large levels of noise in the data (red, black, blue) in case of partially observable data.

## IX. Experiments

In this section, the results of applying the learning algorithm to a nonlinear example similar to [5] are presented. The reference BN consists of the parameters and the graph given

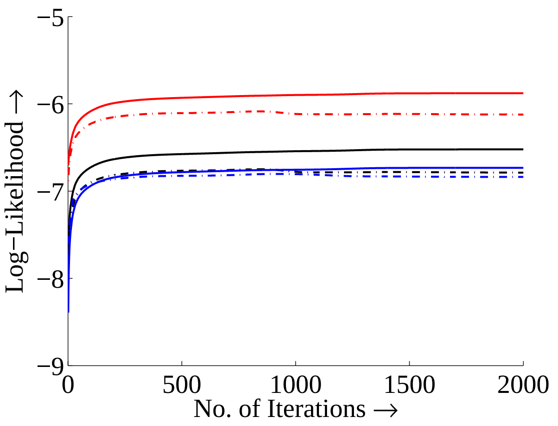


Fig. 5. Log-likelihood of the BN on training (solid) and test (dashed) sets for no, small, and large levels of noise in the data (red, black, blue) in case of fully observable data.

in Ex. 1 as well as Fig. 3. The example is representative as it entails a purely discrete CPT and hybrid conditional densities in the form a single Gaussian density and a Gaussian mixture density. The effective number of parameters to be learned is 36. For the experiments, a set of complete data cases was randomly generated. Partial observability was achieved by hiding the values of the *non-leaf* and *non-root* nodes. The initial parameter estimates $\theta_0$ were obtained by adding white noise to the true model. In the first experiment, a total of 500 data cases was generated (red). The training set consisted of 450 and the test set of 50 examples. Additionally, this data was subjected to two levels of noise. For a small level of noise in the data (black), 10% of the data was corrupted by noise at random. The binary discrete values for $\mathbf{x}$ and $\mathbf{v}$ were toggled and additive white Gaussian noise was added to the continuous measurements of $\mathbf{z}$ with $\sigma = 0.25$. For the large noise level (blue), twice the number of data points were changed and twice the standard deviation was used. For the partially observable learning case, the log-likelihood of the *non-leaf* and *non-root* nodes for the three data sets with respect to the number of iterations are given in Fig. 4. Clearly, all log-likelihood scores improve, but note that around iteration 175 for the non-noisy data set the learning algorithm produces a slightly higher likelihood for the training set than at the end of learning. We think this abnormality is due to the heuristic change in the gradient. Yet, removing the heuristic factor will result in low variance hybrid conditional densities that give numerical issues. Note that overfitting occurs, which can be seen between 175 to 200 in Fig. 4. For the fully observable case, the log-likelihood of the data is presented for the same data sets. Note that the heuristic variance correction cancels out for this case by default and therefore does not impact this case. All likelihoods improve monotonically. Again, overfitting can be observed in Fig. 5 after iteration 500.

## X. Conclusions

In this paper, the first algorithm for learning hybrid Bayesian Networks with Gaussian mixture and Dirac mixture conditional densities [5], [11] from data given their structure is presented. A gradient ascent learning algorithm for maximizing the log-likelihood was derived. In conjunction with an adaption rate, the analytic gradient expressions yield an incremental learning algorithm. The proposed algorithm can be used for learning the parameters of fully and only partially observable BN and is shown to subsume existing algorithms for purely discrete BN. The practicability of this approach is demonstrated on the basis of a default example. It remains future work to prove convergence, to prove the correctness of the given variance correction factor and to investigate more efficient ways of asserting the constraints of the learning algorithm necessary to determine valid conditional densities. As overfitting occurs, it appears promising to improve the generalization capability of the algorithm, especially with regard to learning with small number of samples only.

## References

[1] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan-Kaufmann Publishers, INC., 1988.

[2] S. Lauritzen, "Propagation of Probabilities, Means and Variances in Mixed Graphical Association Models," *Journal of the American Statistical Association*, vol. 87, pp. 1098–1108, 1992.

[3] E. Driver and D. Morrell, "Implementation of Continuous Bayesian Networks Using Sums of Weighted Gaussians," in *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, 1995.

[4] U. Lerner, *Hybrid Bayesian Networks for Reasoning About Complex Systems*, Ph.D. thesis, Stanford University, 2002.

[5] O. C. Schrempf and U. D. Hanebeck, "A New Approach for Hybrid Bayesian Networks Using Full Densities," in *Proceedings of the 6th International Workshop on Computer Science and Information Technologies (CSIT 2004)*, Budapest, Hungary, Oct. 2004, vol. 1, pp. 32–37.

[6] D. Heckerman, "A Tutorial on Learning with Bayesian Networks," in *Learning in Graphical Models*. MIT Press, Cambridge, MA, 1999.

[7] S. Lauritzen and F. Jensen, "Stable Local Computation with Conditional Gaussian Distributions," *Statistics and Computing*, vol. 11, no. 2, pp. 191–203, 1999.

[8] S. Lauritzen, "The EM algorithm for Graphical Association Models with Missing Data," *Compututational Statistics Data Analysis*, vol. 19, no. 2, pp. 191–201, 1995.

[9] M.I. Jordan and R.A. Jacobs, "Hierarchical Mixtures of Experts and the EM algorithm," *Neural Computation*, vol. 6, no. 2, pp. 181–214, 1994.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer Science+Business Media, LLC, 2006.

[11] O. Schrempf and U. D. Hanebeck, "Evaluation of Hybrid Bayesian Networks using Analytical Density Representations," in *Proceedings of the 16th IFAC World Congress (IFAC 2005)*, Czech Republic, 2005.

[12] K. Murphy, *Dynamic Bayesian Network : Representation, Inference and Learning*, Ph.D. thesis, UC Berkeley, 2002.

[13] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society*, vol. 39, no. 1, 1977.

[14] J. Binder, D. Koller, S. Russell, and P. Smyth, "Adaptive Probabilistic Networks with Hidden Variables," *Machine Learning*, vol. 29, pp. 213–244, 1997.

[15] E. Bauer, D. Koller, and Y. Singer, "Update Rules for Parameter Estimation in Bayesian Networks," in *Proceedings of the Thirteenth Annual Conference on Uncertainty in AI (UAI)*, Providence, Rhode Island, 1997, pp. 3–13.

[16] G. McLachlan and D. Peel, *Finite Mixture Models*, Wiley-Interscience, 2000.

[17] A. Taleb, "A Generic Framework for Blind Source Separation in Structured Nonlinear Models," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1819–1830, 2002.