Learning hybrid Bayesian networks using mixtures of truncated exponentials ⋆

Vanessa Romero

Cajamar Savings Bank
Plaza Barcelona, 5
04006 Almería, Spain

Rafael Rumí and Antonio Salmerón *

Dpt. Statistics and Applied Mathematics
University of Almería
La Cañada de San Urbano s/n
04120 Almería, Spain

Abstract

In this paper we introduce an algorithm for learning hybrid Bayesian networks from data. The result of the algorithm is a network where the conditional distribution for each variable is a mixture of truncated exponentials (MTE), so that no restrictions on the network topology are imposed. The structure of the network is obtained by searching over the space of candidate networks using optimisation methods. The conditional densities are estimated by means of Gaussian kernel densities that afterwards are approximated by MTEs, so that the resulting network is appropriate for using standard algorithms for probabilistic reasoning. The behaviour of the proposed algorithm is tested using a set of real-world and artificially generated databases.

Key words: Bayesian networks, Mixtures of Truncated Exponentials, continuous variables, structural learning, parameter learning, kernel methods, simulated annealing.

⋆ This work has been supported by the Spanish Ministry of Education and Science, projects TIC2001-2973-C05-02 and TIN2004-06204-C03-01, and by FEDER funds
* Corresponding author
Email addresses: aromerofernandez@cajamar.es (Vanessa Romero), rrumi@ual.es (Rafael Rumí), Antonio.Salmeron@ual.es (Antonio Salmerón).

1 Introduction

Mixtures of truncated exponentials, abbreviated as MTE, were introduced as a model for dealing with discrete and continuous variables simultaneously in Bayesian networks without imposing any restriction on the network topology and avoiding the rough approximations of methods based on the discretisation of the continuous variables [1]. The ability of MTEs for fitting several common probability models has been widely studied in the last two years [2,3].

The problem of learning Bayesian networks with MTEs can be structured into three tasks: Learning the structure of the network, estimating the marginal distributions for the root nodes (univariate MTEs) and obtaining the conditional distributions for non-root nodes (conditional MTEs). There are methods for learning univariate [4,5] and conditional MTEs [6].

In this paper we propose a method for inducing the structure of an MTE network from data. The method is based on a search procedure over the space of candidate network, and the search is guided by optimisation techniques. Furthermore, the parameter learning technique developed in [4,5] is improved by means of smoothing the empirical density of the data using kernel densities, that afterwards are approximated by MTEs.

The paper is organised as follows. The necessary concepts relative to the MTE distribution are reviewed in section 2. Section 3 is devoted to introduce the proposed schemes for structural learning. The performance of the new algorithms is experimentally tested as reported in section 4 and the paper ends with conclusions in section 5.

2 The MTE model

Throughout this paper, random variables will be denoted by capital letters, and their values by lowercase letters. Boldfaced characters will be used for random vectors. The state space of the vector $\mathbf{X}$ is denoted by $\Omega_{\mathbf{X}}$. The MTE model is defined by its corresponding potential and density as follows [1]:

Definition 1 (MTE potential) *Let* $\mathbf{X}$ *be a mixed* n*-dimensional random vector. Let* $\mathbf{Y} = (Y_1, \ldots, Y_d)$ *and* $\mathbf{Z} = (Z_1, \ldots, Z_c)$ *be the discrete and continuous parts of* $\mathbf{X}$*, respectively, with* $c + d = n$*. We say that a function* $f : \Omega_{\mathbf{X}} \mapsto \mathbb{R}_0^+$ *is a* Mixture of Truncated Exponentials potential (MTE potential) *if one of the next conditions holds:*

i. $\mathbf{Y} = \emptyset$ *and* f *can be written as*

$$f(\mathbf{x}) = f(\mathbf{z}) = a_0 + \sum_{i=1}^{m} a_i \exp\left\{\sum_{j=1}^{c} b_i^{(j)} z_j\right\} \tag{1}$$

for all $\mathbf{z} \in \Omega_{\mathbf{Z}}$*, where* a_i*,* $i = 0, \ldots, m$ *and* $b_i^{(j)}$*,* $i = 1, \ldots, m$*,* $j = 1, \ldots, c$ *are real numbers.*

ii. $\mathbf{Y} = \emptyset$ *and there is a partition* $D_1, \ldots, D_k$ *of* $\Omega_{\mathbf{Z}}$ *into hypercubes such that* f *is defined as*

$$f(\mathbf{x}) = f(\mathbf{z}) = f_i(\mathbf{z}) \quad \text{if} \quad \mathbf{z} \in D_i \ ,$$

where each f_i*,* $i = 1, \ldots, k$ *can be written in the form of equation (1).*

iii. $\mathbf{Y} \neq \emptyset$ *and for each fixed value* $\mathbf{y} \in \Omega_{\mathbf{Y}}$*,* $f_{\mathbf{y}}(\mathbf{z}) = f(\mathbf{y}, \mathbf{z})$ *can be defined as in* ii.

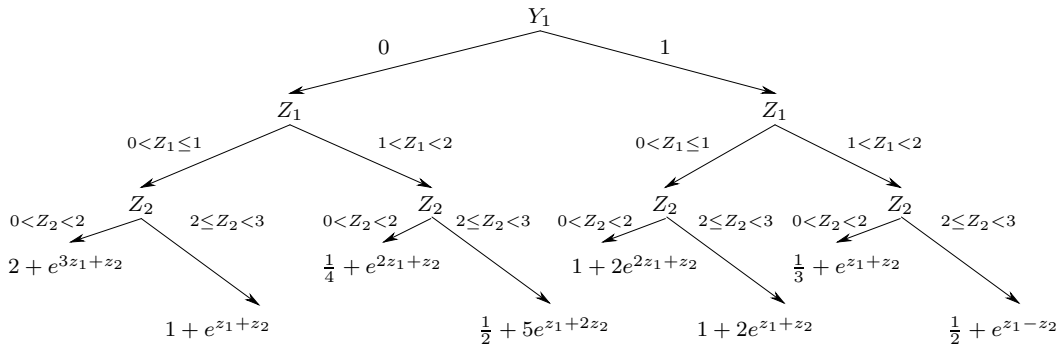


Fig. 1. An example of mixed probability tree.

Definition 2 (MTE density) *An MTE potential* f *is an* MTE density *if*

$$\sum_{\mathbf{y} \in \Omega_{\mathbf{Y}}} \int_{\Omega_{\mathbf{Z}}} f(\mathbf{y}, \mathbf{z}) d\mathbf{z} = 1 \ , \tag{2}$$

where $\mathbf{Y}$ *and* $\mathbf{Z}$ *are the discrete and continuous coordinates of* $\mathbf{X}$ *respectively.*

In a Bayesian network, two types of probability density functions can be found:

(1) For each variable X which is a root of the network, a density $f(x)$ is given.
(2) For each variable X with parents $\mathbf{Y}$, a conditional density $f(x|\mathbf{y})$ is given.

A *conditional MTE density* $f(x|\mathbf{y})$ is an MTE potential $f(x, \mathbf{y})$ such that after fixing $\mathbf{y}$ to each of its possible values, the resulting function is a density for X.

In [1] a data structure was proposed to represent MTE potentials, called *mixed probability trees* or mixed trees for short. Mixed trees can represent MTE potentials defined by parts. Each entire branch in the tree determines one sub-region of the space where the potential is defined, and the function stored in the leaf of a branch is the definition of the potential in the corresponding sub-region. An example of an MTE potential represented as a mixed tree can be seen in figure 1.

The operations required for probability propagation in Bayesian networks (restriction, marginalisation and combination) can be carried out by means of algorithms very similar to those described for discrete probability trees in [7,8].

3 Structural learning with MTEs

Given a mixed random vector $\mathbf{X} = \{X_1, \ldots, X_n\}$, and a sample of $\mathbf{X}$,

$$D = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\} \ ,$$

our aim is to design a method for obtaining a Bayesian network with variables $\mathbf{X}$, that agrees with the data D.

Basically, the problem of learning Bayesian networks from data can be approached as repeating the next three steps until an optimal network is obtained:

(1) Selection of a candidate structure G.
(2) Estimation of the conditional distributions, $\hat{\theta}$, for G.
(3) Determination of the quality of $(G, \hat{\theta})$.

Our proposal consists of exploring the space of possible networks for variables $\mathbf{X}$ using an optimisation approach. The starting point will be a network without arcs. With respect to the movement operators, we have considered arc insertion, deletion and reversal. After each movement, the conditional distributions corresponding to the families involved in the change are estimated.

The problem of estimating the parameters of truncated distributions has been previously studied [9,10], as in the case of the truncated Gamma [11,12], but the number of parameters is usually equal to one, and the maximum likelihood estimator (that not always exists) or the UMVUE (Uniformly of Minimum Variance Unbiased Estimator) is obtained by means of numerical methods [13,14]. In the case of the MTE models, no similar techniques have been applied so far, due to the high number of parameters involved in the MTE densities.

Another usual way to compute maximum likelihood estimates in mixture models is the *EM algorithm* [15,16]. The difficulty in applying this method to learning MTE models lies in the fact that we may have negative coefficients for some of the densities we are combining and also in the computation of the conditional expectations in each iteration of the algorithm.

Due to the difficulties described above, the seminal paper on estimating MTEs from data [4], followed an approach based on regression techniques for the case of univariate densities. Besides the estimation of the parameters, the construction of an MTE density involves the determination of the number of terms and the splits into which its domain is partitioned. Heuristics to approach these issues are proposed in [4]. The estimation procedure can be summarised in the next algorithm.

Algorithm MTE-fitting

INPUT:

- A sample $x_1, \ldots, x_n$.

OUTPUT:

- Estimates of the parameters of the fitted model, $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ and $\hat{k}$.

(1) Using sample $x_1, \ldots, x_n$, obtain two vectors $(x_1^*, \ldots, x_n^*)$ and $(y_1, \ldots, y_n)$, where the first one contains values of the variable, and the second one contains their corresponding empirical density values.
(2) Divide the range of the variable into subintervals in terms of concavity/convexity and increase/decrease of the curve determined by the points in vectors $(x_1^*, \ldots, x_n^*)$ and $(y_1, \ldots, y_n)$.
(3) For each subinterval do:
   - Fit $y = f(x) = k + a \exp\{bx\} + c \exp\{dx\}$, using an iterative least squares procedure.
(4) Normalise the whole function to integrate up to one.
(5) Let $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ and $\hat{k}$ be the coefficients of the normalised function.

(6) RETURN($\hat{a}, \hat{b}, \hat{c}, \hat{d}, \hat{k}$).

Although the core of this algorithm is Step 3, the results strongly depend on Step 1, in two ways:

(1) *The accuracy of the estimation of the empirical density using the given sample.* If the estimation is poor, the result can be a density far away from the original one.
(2) *The size of the vectors obtained.* Even if the empirical density is properly captured, if the exponential regression in Step 3 is computed from a scarce set of points, the accuracy of the approximation can be poor.

The method described in [4] used the empirical histogram as an approximation of the actual density of the sample points. In this work, with the aim of avoiding the two problems mentioned above, we have decided to improve the estimation procedure using kernel approximations to the empirical density.

First of all, a Gaussian kernel density [17] is fitted to the data corresponding to each leaf. A Gaussian kernel density for a variable X and a sample $\{x_1, \ldots, x_m\}$ is defined as

$$f(x) = \frac{1}{mh} \sum_{i=1}^{m} K\left(\frac{x - x_i}{h}\right) \ , \tag{3}$$

where

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{u^2}{2}\right\} \quad \forall u \in \mathbb{R} \ . \tag{4}$$

The value of h in equation (3) is selected in order to minimise the mean squared error of the kernel density with respect to the data. The optimal value is [17]:

$$h_{optimal} = 1.059 \sigma m^{-1/5} \ , \tag{5}$$

where σ is estimated as the standard deviation of sample $\{x_1, \ldots, x_m\}$.

The Gaussian kernel density provides a smooth approximation to the empirical density of the data, which is specially useful in situations in which the amount of data is scarce, since peaks in the density owing to the lack of data are not as rough as in the case of using histograms.

Out of the fitted Gaussian kernel density, $f(x)$, an artificial sample consisting of pairs $(x_1, f(x_1)), \ldots, (x_h, f(x_h))$, is drawn by taking equidistant points, and an MTE density is obtained from it using the regression-based algorithm described in [4,5].

Besides the Gaussian kernels, there are several other types of kernel functions, like the *Biweight* and *Epanechnikov* kernels [17].

The performance of Gaussian kernels versus histograms or other types of kernels for fitting MTE densities is tested in figures 2 and 3, which respectively show the MTE density obtained through histogram and kernel approximations out of a sample of 200 points randomly sampled from a standard normal distribution. More arguments supporting the use of Gaussian kernels are analysed in [18].

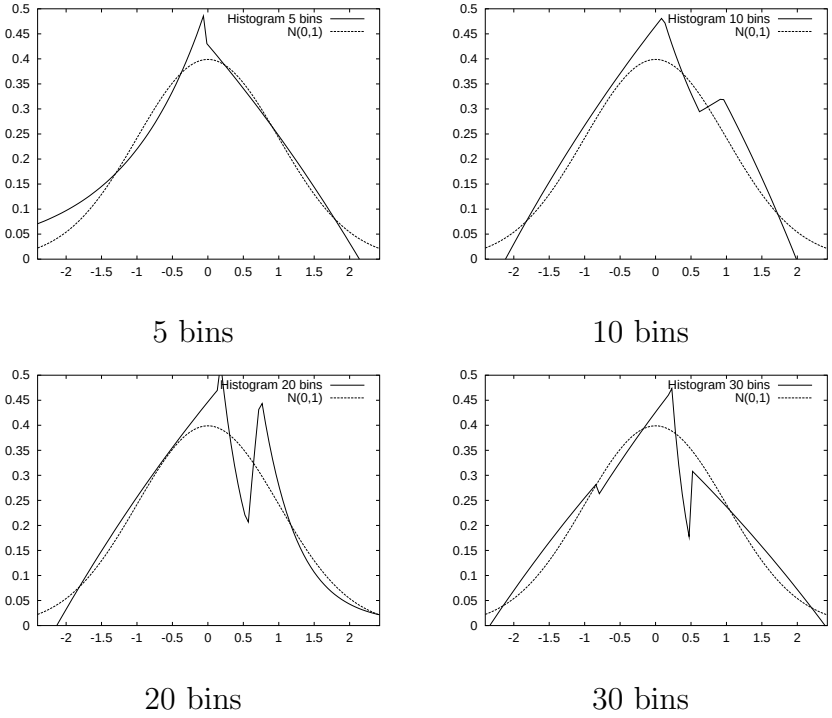
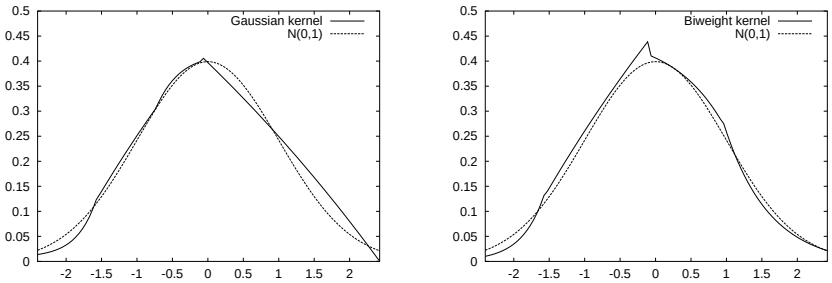
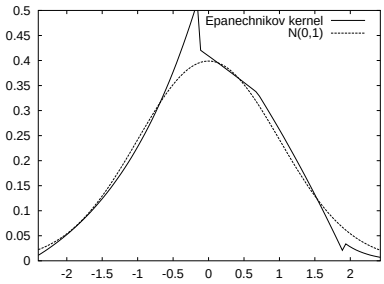


Fig. 2. Results of fitting an MTE from 5, 10, 20 and 30 bin histograms.

The method described so far for constructing estimators for the parameters of the univariate MTE density is not valid for the conditional case, since more restrictions should be imposed over the parameters in order to force the MTE potential to integrate up to 1 for each combination of values of the conditioning variables, i.e. to force the MTE potential to actually be a conditional density. This problem was approached in [6] by partitioning the domain of the conditioning variables and then fitting a univariate density in each one of the splits using the method described above. More precisely, the algorithm learns a mixed tree in which the leaves contain MTE densities that depend only on the child variable, and that represents the density of the child variable given by the values contained in the region determined by the corresponding branch of the mixed tree. The tree is learnt in such a way that the leaves discriminate as much as possible, following a schema similar to the construction of decision trees [19].



Gaussian kernel Biweight kernel



Epanechnikov kernel

Fig. 3. Results of fitting an MTE from Gaussian, Biweight and Epanechnikov kernels.

3.2 Measuring the quality of a candidate network

In order to measure the quality of a Bayesian network, we propose to use a metric based on the asymptotic approximation to the classical Bayesian metric proposed in [20] for networks with continuous variables. The idea is to construct a score that takes into account the likelihood of the data given the candidate network but penalising those ones with complex structure.

We define the following metric:

$$Q(G|D,\hat{\theta}) = \log L(D;G,\hat{\theta}) - \frac{\log m}{2}\mathrm{Dim}(G), \tag{6}$$

where $L(D;G,\hat{\theta})$ is the likelihood of the data given the current network and $\mathrm{Dim}(G)$ is the number of parameters needed to specify the network G that must be learnt from the data.

The amount of parameters that are learnt from data for the conditional density of any variable given its parents in the network, is the result of adding:

(1) The number of parameters in the leaves of the corresponding mixed tree that will be used to represent the density (see Section 2).
(2) The number of points that determine the partition of the domain of each

continuous variable.

Along this paper, for the sake of simplicity, we will assume that all the MTE potentials that will appear in the learnt network will have a constant number of parameters, say k, which means that the potentials will have the form

$$f(x) = a_0 + a_1 e^{b_1 x} + \cdots + a_t e^{b_t x} \ ,$$

with $k = 2t + 1$. A potential defined in this way will be called a *k-parameter MTE potential*. For instance, a 5-parameter MTE potential has an independent term and two exponentials.

If we denote by $|X|$ the number of possible values of X if it is discrete, or the number of splits into which its domain is divided, if X is continuous, the number of parameters necessary to specify the density for the family of X (i.e., variable X and its parents, $pa(X)$, denoted as $fa(X) = \{X\} \cup pa(X)$) can be expressed as

$$\mathrm{Par}(fa(X)) = k \prod_{Y \in fa(X)} |Y| \tag{7}$$

if X is continuous, and

$$\mathrm{Par}(fa(X)) = \prod_{Y \in fa(X)} |Y| \tag{8}$$

if X is discrete, since in this case each leaf of the mixed tree contains exactly one real value.

Regarding the partition of the domain, we will consider a constant number of splits for every continuous variable. Let s denote the number of splits. It means that, whenever a variable is to be split, $s - 1$ splitting points must be determined.

The number of splitting points that must be inferred from the data for each family, cannot be determined beforehand, since the order in which the variables are arranged in the mixed tree may increase or decrease it. For instance, consider a family composed by two variables, one of which is discrete and the other one being continuous. If the continuous variable is located in the root of the mixed tree, the number of splitting points to be determined is equal to $s - 1$, whilst if the discrete variable is in the root, the number of splitting points increases up to $d \times (s - 1)$, where d is the number of possible values of the discrete variable.

Due to the reason mentioned above, we will use an upper bound of the number of splitting points instead. Assuming that there are N_c and N_d continuous and

discrete variables respectively in the family of a variable X, $fa(X)$, and that d is the number of possible values of the discrete variables in $fa(X)$, an upper bound for the number of splitting points that must be learnt when constructing the mixed tree for $fa(X)$ is given by

$$\text{Spl}(fa(X)) = (s-1)^{N_c} d^{N_d} \ . \tag{9}$$

Thus, according to equations (7), (8) and (9) , the dimension of a family $fa(X)$ can be computed as

$$\text{Dim}(fa(X)) = \text{Par}(fa(X)) + \text{Spl}(fa(X_i)) \ , \tag{10}$$

and the dimension of a network G as

$$\text{Dim}(G) = \sum_{i=1}^{n} \text{Dim}(fa(X_i)) \ . \tag{11}$$

Thus, the metric in equation (6) can be expressed as

$$\begin{aligned} Q(G|D,\hat{\theta}) &= \log L(D;G,\hat{\theta}) - \frac{\log m}{2}\text{Dim}(G) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} \log p_j(x_j^{(i)}|pa(x_j^{(i)})) - \frac{\log m}{2}\sum_{i=1}^{n}\text{Dim}(X_i) \ . \end{aligned}$$

This metric can be decomposed as

$$Q(G|D,\hat{\theta}) = \sum_{j=1}^{n} Q(X_j|D,\hat{\theta}) \ , \tag{12}$$

where

$$Q(X_j|D,\hat{\theta}) = \sum_{i=1}^{m} \log p_j(x_j^{(i)}|pa(x_j^{(i)})) - \frac{\log m}{2}\text{Dim}(fa(X_j)) \ , \tag{13}$$

which means that after carrying out a modification over a network, only the part of the metric corresponding to the two variables affected by the operation has to be re-computed.

Once the metric is defined, it can be used to guide the process of learning the network from data. The problem can be viewed as the optimisation of the metric defined in equation (6) within the space of candidate networks. In the next subsection we will describe the optimisation procedure adopted in this paper.

In a preliminary version of this work [21] an optimal network was selected using a hill climbing method that started from a network without arcs and performed a greedy search trying to optimise the metric in equation (6). We will refer to this method as Algorithm HC, standing for Hill Climbing. The only difference of the version of Algorithm HC employed in this work with respect to the one used in [21] is that the parameters are estimated using kernel methods as described in section 3.1. The detailed algorithm is as follows:

Algorithm HC(G_0,D)

INPUT:

- G_0: The initial network (without arcs).
- D: The database.

OUTPUT: The learnt network.

(1) $G := G_0$.
(2) Estimate the conditional distributions for G as described in section 3.1.
(3) Let $\hat{\theta}$ be the estimated conditional distributions.
(4) Compute the quality of G as $q := Q(G|D, \hat{\theta})$.
(5) FOR each G' belonging to the neighbourhood of G,
  (a) Let $\hat{\theta}'$ be the conditional distributions for G'.
  (b) Compute the quality of the modified network:

$$q' := Q(G'|D, \hat{\theta}') \ .$$

  (c) IF $q < q'$
    - $G := G'$.
    - $q := q'$.
(6) RETURN(G).

The neighbourhood of a given network considered in Step (5) consists of the networks obtained from it by inserting, removing or reversing an arc as long as no directed cycles are created.

Besides the greedy search, we have also used the simulated annealing algorithm to explore the space of candidate networks, in order to reduce the risk of reaching local optima, which is the main drawback of greedy search. We will refer to this method as Algorithm SA, standing for Simulated Annealing.

We have adopted a cooling scheme determined by the following temperature:

$$T(i) = \frac{100}{\log(i+1)} \ , \tag{14}$$

where i is the iteration of the algorithm. The decision of using this temperature was made after performing several preliminary experiments with other temperatures suggested in [22].

The pseudo-code for the learning algorithm based on Simulated Annealing is the following.

Algorithm SA(G_0,nit,D)

INPUT:

- G_0: The initial network (without arcs).
- nit: The number of iterations.
- D: The database.

OUTPUT: The learnt network.

(1) $G := G_0$.
(2) Estimate the conditional distributions for G as described in section 3.1.
(3) Let $\hat{\theta}$ be the estimated conditional distributions.
(4) Compute the quality of G as $q := Q(G|D, \hat{\theta})$.
(5) FOR $i := 1$ TO nit
  (a) Compute the temperature

$$T(i) = \frac{100}{\log(i+1)} \ .$$

  (b) Choose at random one network G' from the neighbourhood of network G.
  (c) Let $\hat{\theta}'$ be the conditional distributions for G'.
  (d) Compute the quality of the modified network:

$$q' := Q(G'|D, \hat{\theta}') \ .$$

  (e) Generate a random number $r \in (0, 1)$.
  (f) Compute

$$p := \min\left\{\exp\left\{\frac{q'-q}{T(i)}\right\}, 1\right\} \ .$$

  (g) IF $(r < p)$ OR $(q' > q)$
    - $G := G'$.
    - $q := q'$.
(6) RETURN(G).

Table 1
Structural information about the artificial networks used in the experiments.

	net5	net10
No. of links	6	13
No. of discrete vars.	1	2
No. of continuous vars.	4	8

As in the case of Algorithm HC, the neighbourhood considered in Step (5)(c) consists of the networks obtained from it by inserting, removing or reversing an arc as long as no directed cycles are created.

4 Experimental evaluation

The experimental analysis reported in this section has been carried out using an implementation of algorithms SA and HC in Java language that we have included in the Elvira system [23] (see `http://leo.ugr.es/elvira`).

We have tested the new algorithms in two different experiments using synthetic and real-world data. We will describe both experiments separately.

4.1 Tests with synthetic data

We have considered two synthetic databases with 500 instances each one, denoted as db5 and db10 obtained respectively by forward sampling from two artificial networks called net5 and net10. Table 1 contains the information regarding the structure of these networks.

The structure of networks net5 and net10 was generated as follows:

- For each variable, the number of parents is selected according to a Poisson distribution with mean 0.8 .
- The parents are selected at random, among those that do not violate the DAG condition.

The parameters (conditional distributions) of the artificial networks are generated according to the following procedure:

- The number of values of each discrete variable is selected uniformly at random from the set $\{2, 3, 4\}$.
- The values in the probability tables of the discrete variables are generated

Table 2
Results of the experiment for database db5.

Algorithm	SA	HC	K2	MTE_K2
LL	2	1	2	2
CL	0	0	2	2
IL	1	1	0	0
NL	1	0	0	0
Q	-38.8013	2.4803	-31.8704	**66.7387**

Table 3
Results of the experiment for database db10.

Algorithm	SA	HC	K2	MTE_K2
LL	7	3	8	8
CL	0	2	0	0
IL	2	0	0	0
NL	5	1	0	0
Q	-561.5937	**-415.0141**	-1004.2223	-721.7311

Table 4
Results of the experiment for database diabetes.

Algorithm	SA	HC	K2	MTE_K2
Q	-6908.481	-6945.2861	-7051.232	**-6496.7746**

from a negative exponential distribution with mean 0.5, and they are normalised afterwards.
- The number of splits of the state space of each continuous variable is set to 3.
- The number of exponential terms for each MTE potential is equal to 2.
- The independent term of each MTE potential is generated from a negative exponential distribution with mean 0.01.
- The coefficients of the exponential terms in the potentials are generated from a negative exponential distribution with mean randomly selected within the interval $(1, 10)$.
- The coefficients of the exponents in the exponential terms are generated from a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 5$.

Each database was randomly divided into two parts: one for training and one for test, containing 70% and 30% of the instances respectively. Then, using the two training databases, we run the following algorithms:

- Algorithms SA with 100 iterations and HC, as described in section 3.3.

- Algorithm K2, as described in [24]. Since this algorithm is designed for qualitative variables, the database was previously discretised using the k-means algorithm with 3 categories per variable.
- Algorithm MTE_K2. The same as K2, but once the structure is obtained, the conditional distributions are fitted using mixtures of truncated exponentials.

For each learnt network, we recorded the number of links (LL), number of coincident links (CL), number of inverted links (IL) and number of new links (NL), i.e. those not coincident nor inverted. We also computed the quality (Q) of each network using the test databases and computed according to equation (6). The results are shown in tables 2 and 3. The number of links of the original networks are shown in table 1.

In the experiments, we have used 5-parameter MTE potentials in the learnt networks. It means that in each leaf of the mixed tree corresponding to a conditional distribution, the fitted MTE potential has the form:

$$f(x) = a_0 + a_1 e^{b_1 x} + a_2 e^{b_2 x} \quad .$$

Furthermore, the number of splits into which the state space of the variables is split is set to 3.

4.2 Experiments with real-world data

A similar experiment was run using a real-world database, called diabetes and taken from the UCI machine learning repository [25]. The database contains 768 instances of 9 variables, 8 of them continuous and another one discrete. The database was also randomly divided into two databases for train and test, with 70% and 30% of the instances respectively.

In this case, the original network is unknown, so that we have not measured any structural parameter in the learnt network, but only the quality. The results for this database can be seen in table 4.

4.3 Results discussion

According to the results displayed in tables 2, 3 and 4, the following conclusions can be drawn:

- The use of MTE potentials instead of discrete distribution is in general a good choice. This is specially clear looking at the results of algorithms K2

and MTE_K2: the last one clearly outperforms the other, which is particularly significant given that both of them provide the same network structure, and the only difference is the use of MTE potentials.
- There are no big differences between HC and SA in the reported experiments. However, it must be taken into account that in the experiments, the number of iterations in algorithm SA was set to 100 and in each one of them, only one network from the neighbourhood of the current network is explored. This was aimed at keeping a computational cost similar to HC, in order to make a fair comparison.
- Algorithm MTE_K2 in general outperforms the other ones in two out of the three experiments. One reason for this good performance is that K2 requires the nodes to be ordered. In these experiments, we provided the algorithm with the right order taken from the original artificial networks, what makes more likely to get the correct links. However, SA and HC do not use any node order. Thus, if the order of the variables is known, it is a good idea to use MTE_K2 instead of HC or SA.

5 Conclusions

We have introduced two algorithms for learning the structure of Bayesian networks with discrete and continuous variables simultaneously, in which the MTE model is used. So far, algorithms for estimating marginal and conditional MTE densities existed [4–6], but the obtainment of the network structure remained unsolved. Furthermore, the estimation algorithms proposed in [4,6] are improved by means of using Gaussian kernels in order to get smoother representations of the empirical density. Additionally, we have shown that the use of an algorithm for discrete variables to capture the structure, and then fitting MTE potentials can be valid in some situations, for instance if an order of the nodes is known.

Perhaps the main feature of the methods proposed here is that they can be used to construct a Bayesian network from any kind of mixed data, without worrying about the structural restriction imposed by the Conditional Gaussian model [26], which requires discrete nodes not to have continuous parents, and without discretising the continuous variables.

Due to the lack of previous material on learning hybrid networks without structure restrictions, it is difficult to design an experimental setting to test the performance of the algorithms proposed here. Nevertheless, the experiments described in section 4 seem to support their correctness.

However, still much effort must be invested in order to reach an entirely satisfactory solution for the structural MTE learning problem. For instance, the

metric used in this paper is known to have good properties when the conditional distributions in the network belong to the curved exponential family [27]. The MTE distribution does not belong to this family, and thus the asymptotic properties of the metric should be studied.

Another aspect that influences the performance of the structural learning algorithm is the estimation of the parameters. The technique we have used here improves the existing estimation methods for MTEs. Since the structural and parametric learning are problems that can be studied independently, new improvements in the estimation of the parameters will automatically benefit the structural learning algorithm that we propose.

With respect to the search scheme, we are planning to used methods that try to avoid reaching local optima by more sophisticated means than the simulated annealing approach. Our next goal is to implement a version based on the *stochastic variable neighbourhood search* algorithm [28].

References

[1] S. Moral, R. Rumí, A. Salmerón, Mixtures of truncated exponentials in hybrid Bayesian networks, in: Lecture Notes in Artificial Intelligence, Vol. 2143, 2001, pp. 135–143.

[2] B. Cobb, P. Shenoy, Inference in hybrid Bayesian networks with mixtures of truncated exponentials, in: Proceedings of 6th Workshop on Uncertainty Processing, Hejnice, Czech Republic, 2003, pp. 47–63.

[3] B. Cobb, P. Shenoy, R. Rumí, Approximating probability density functions with mixtures of truncated exponentials, in: Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU-2004), In press, Perugia, Italy, 2004.

[4] S. Moral, R. Rumí, A. Salmerón, Estimating mixtures of truncated exponentials from data, in: J. Gámez, A. Salmerón (Eds.), Proceedings of the First European Workshop on Probabilistic Graphical Models, 2002, pp. 156–167.

[5] R. Rumí, A. Salmerón, S. Moral, Estimating mixtures of truncated exponentials in hybrid Bayesian network, Test (2005) In press.

[6] S. Moral, R. Rumí, A. Salmerón, Approximating conditional MTE distributions by means of mixed trees, in: Lecture Notes in Artificial Intelligence, Vol. 2711, 2003, pp. 173–183.

[7] D. Kozlov, D. Koller, Nonuniform dynamic discretization in hybrid networks, in: D. Geiger, P. Shenoy (Eds.), Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, Morgan & Kaufmann, 1997, pp. 302–313.

[8] A. Salmerón, A. Cano, S. Moral, Importance sampling in Bayesian networks using probability trees, Computational Statistics and Data Analysis 34 (2000) 387–413.

[9] W. Smith, A note on truncation and sufficient statistics, Annals of Mathematical Statistics 28 (1957) 247–252.

[10] J. Tukey, Sufficiency, truncation and selection, Annals of Mathematical Statistics 20 (1949) 309–311.

[11] L. Hegde, R. Dahiya, Estimation of the parameters of a truncated Gamma distribution, Communications in Statistics. Theory and Methods 18 (1989) 561–577.

[12] G. Nath, Unbiased estimates of reliability for the truncated Gamma distribution, Scandinavian Actuarial Journal (3) (1975) 181–186.

[13] M. El-Taha, W. Evans, A new estimation procedure for the right-truncated exponential distribution, in: Proceedings of the 23th Pittsburgh Conference on Modelling and Simulation, 1992, pp. 427–434.

[14] Y. Sathe, S. Varde, Minimum variance unbiased estimates of reliability for the truncated exponential distribution, Technometrics 11 (1969) 609–612.

[15] A. Dempster, N. Laird, D. Rubin, Maximun likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society B 39 (1977) 1 – 38.

[16] R. Redner, H. Walker, Mixture densities, maximum likelihood and the EM algorithm, SIAM Review 26 (1984) 195–239.

[17] J. Simonoff, Smoothing methods in Statistics, Springer, 1996.

[18] R. Rumí, Kernel methods in Bayesian networks, in: Proceedings of the 1st International Conference of Mediterranean Mathematicians, Almería, Spain, 2005.

[19] J. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.

[20] E. Castillo, J. Gutiérrez, A. Hadi, Expert systems and probabilistic network models, Springer-Verlag, New York, 1997.

[21] V. Romero, R. Rumí, A. Salmerón, Structural learning of Bayesian networks with mixtures of truncated exponentials, in: Proceedings of the 2nd European Workshop on Probabilistic Graphical Models (PGM'04), Leiden, The Netherlands, 2004, pp. 177–184.

[22] C. Robert, G. Casella, Monte Carlo statistical methods, Springer, 1999.

[23] Elvira Consortium, Elvira: An environment for creating and using probabilistic graphical models, in: J. Gmez, A. Salmern (Eds.), Proceedings of the First European Workshop on Probabilistic Graphical Models, 2002, pp. 222–230.

[24] G. Cooper, E. Herskovits, A Bayesian method for the induction of probabilistic networks from data, Machine Learning 9 (1992) 309–347.

[25] C. Blake, C. Merz, UCI repository of machine learning databases, www.ics.uci.edu/∼mlearn/MLRepository.html, university of California, Irvine, Dept. of Information and Computer Sciences (1998).

[26] S. Lauritzen, Propagation of probabilities, means and variances in mixed graphical association models, Journal of the American Statistical Association 87 (1992) 1098–1108.

[27] D. Haughton, On the choice of a model to fit data from an exponential family, Annals of Statistics 16 (1988) 342–355.

[28] L. de Campos, J. Puerta, Stochastic local and distributed search algorithms for learning belief networks, in: Proceedings of the III International Symposium on Adaptive Systems (ISAS): Evolutionary Computation and Probabilistic Graphical Models, 2001, pp. 109–115.