Gaussian Sum-Product Networks Learning in the Presence of Interval Censored Data

Pierre Clavier, Olivier Bouaziz, Gregory Nuel

Gaussian Sum-Product Networks Learning in the Presence of Interval Censored Data

Pierre Clavier PIERRE.CLAVIER.C@GMAIL.COM
Université de Paris, CNRS, MAP5 UMR 8145

Olivier Bouaziz OLIVIER.BOUAZIZ@PARISDESCARTES.FR
Université de Paris, CNRS, MAP5 UMR 8145

Grégory Nuel GREGORY.NUEL@MATH.CNRS.FR
LPSM, CNRS 8001, Probability and Statistics, Sorbonne Université

Abstract

Sum-Product Networks (SPNs) can be seen as deep mixture models that have demonstrated efficient and tractable inference properties. In this context, graph and parameters learning have been deeply studied but the standard approaches do not apply to interval censored data. In this paper, we derive an approach for learning SPN parameters based on maximum likelihood using Expectation-Maximization (EM) in the context of interval censored data. Assuming the graph structure known, our algorithm makes possible to learn Gaussian leaves parameters of SPNs with right, left or interval censored data. We show that our EM algorithm for incomplete data outperforms other strategies such as the midpoint for censored intervals or dropping incomplete values.

Keywords: Sum-Product Network; EM algorithm; Interval censoring; Maximum likelihood

1. Introduction

Sum-Product Networks (SPNs), introduced by Poon and Domingos (2011) are a recent type of deep probabilistic models that are able to represent high dimensional distributions and are attractive due to their inference properties. Contrary to other probabilistic models such as the Bayesian networks (BNs) or the Markov Networks (MNs), SPNs can perform exact and tractable inference in linear time of the Network. Moreover Zhao et al. (2015) have shown that it is possible to convert SPNs to BNs or MNs and vice-versa. One can think of SPNs as deep neural networks where the non linearity of activation functions are replaced by product nodes. However SPNs own a specific semantic in the sense they can encode a joint probability distribution over a set of random variables and can be seen as a deep and hierarchical mixture model (Martens and Medabalimi, 2014).

Many algorithms have been proposed to estimate parameters of SPNs such as discriminative training of SPNs (Gens and Domingos, 2012), Expectation-Maximization (EM) algorithm (Peharz, 2015; Desana and Schnörr, 2016), Bayesian Moment Matching (Rashwan et al., 2016) or Concave-Convex procedure (Zhao et al., 2016). Moreover, Trapp et al. (2019b) have proposed a Bayesian approach to learn both graph and parameters at the same time and which deal with missing data. However none of those approaches can deal with interval censored data. The aim of this paper is to develop a new method designed to learn the parameters from a fixed SPN graph in the context of interval censoring. Our approach is based on maximum likelihood and hierarchical mixture representation (see Desana and Schnörr, 2016).

The paper is constructed as follows. In Section 2 we will review SPN definition and properties. Then, Section 3 explains the hierarchical mixture model representation of SPNs. In Section 4, the focus is on the Gaussian truncated leaves of SPNs designed to take into account incomplete data. Then we derive an EM algorithm for SPNs in the context of incomplete data in Section 5. Section 6 demonstrates effectiveness of this algorithm to learn parameters of SPNs in the context of interval censored data on three real datasets.

2. Background

Considering $\mathcal{X}$ a set of random variables which can be discrete or continuous, Sum-Product Networks denoted $\mathcal{S}$ have been introduced by Poon and Domingos (2011) and can be defined as rooted Directed Acyclic Graphs (DAGs), $\mathcal{G}(V, E)$ with distributions over random variables in $\mathcal{X}$ as leaves and sum and product nodes as internal nodes. We will write $\varphi_q(X_q)$ the probability densities measure of a leave q on the random variables $X_q \subset \mathcal{X}$. For any node q, its children are denoted $i \in \mathrm{ch}(q)$ and the density of a node S_q can be defined recursively as:

$$S_q(X_q) = \begin{cases} \varphi_q(X_q) & \text{if } q \text{ is a leaf} \\ \prod_{i \in \mathrm{ch}(q)} S_i(X_q) & \text{if } q \text{ is product node} \\ \sum_{i \in \mathrm{ch}(q)} w_i^q S_i(X_q) & \text{if } q \text{ is a sum node} \end{cases}$$

weights w_i^q of the SPN, are strictly positive and we will said that a SPN is *normalised* if all weights of a sum node sum up to 1 i.e : $\sum_{i \in \mathrm{ch}(q)} w_i^q = 1$. In the following, we will only consider *normalised* SPNs. We write $\mathrm{sc}(q)$ the scope of a node q which is the set of all random variables in the leaves of the sub-SPN rooted at node q. Under the two following conditions (Poon and Domingos, 2011) on product and sum nodes, SPNs are able to encode a joint probability density over $\mathcal{X}$ and is said to be *valid*.

Definition 1 *(Decomposability) A product node q of $\mathcal{S}$ is said to be decomposable iff :*

$$\forall i, i' \in \mathrm{ch}(q), i \neq i' \Rightarrow \mathrm{sc}(i) \cap \mathrm{sc}(i') = \emptyset$$

If all product node of $\mathcal{S}$ are decomposable, then the SPN $\mathcal{S}$ is said to be decomposable.

Definition 2 *(Completeness) We say that a sum node q of $\mathcal{S}$ is complete iff :*

$$\forall i, i' \in ch(q) \quad \mathrm{sc}(i) = \mathrm{sc}(i')$$

If all sum nodes of $\mathcal{S}$ are complete, then the SPN $\mathcal{S}$ is said to be complete.

Forward and backward passes in SPNs : We can evaluate the probability of an evidence x which is a realisation of X. Using *forward pass* or *bottom-up evaluation* for every node q, we will denote $S_q(x)$ the evaluation of x at node q. Moreover, we introduce the backward pass of $\mathcal{S}$ at node q, the formal partial derivative $\frac{\partial S}{\partial S_q}$ which can be computed in linear size of the network like forward pass (Gens and Domingos, 2012). Forward and backward passes will be of capital importance in the derivation of the EM algorithm in SPNs.

Parameters of the SPN $\mathcal{S}$: Let $\mathrm{Sum}(\mathcal{S})$ be the set of all sum nodes of $\mathcal{S}$. In this paper, we attempt to estimate the parameters of $\mathcal{S}$ which are all the sum node weights included in $W = \{w_i^q \,|q \in \mathrm{Sum}(\mathcal{S}), i \in \mathrm{ch}(q)\}$ and all the leaves parameters denoted as $\theta = \{\theta_\ell \,|\ell \in \mathcal{L}(\mathcal{S}\}$ with $\mathcal{L}(\mathcal{S})$ the set of all leaves of the SPN. Finally, the evaluation of the density at node q given all weights and leaves parameters will be written as $S_q(x|W,\theta)$ for a node q and $S(x|W,\theta)$ for the root node of $\mathcal{S}$.

3. SPNs as Deep Mixture Models

SPNs can be seen as mixture models (Dennis and Ventura, 2015; Desana and Schnörr, 2016). The notion of membership to a class in classic mixture models is replaced by the notion of membership to an *induced tree*. These *induced trees* will be denoted as $c \in \mathcal{C}$ with $\mathcal{C}$ the set of all induced trees. They can be constructed as follows according to Desana and Schnörr (2016):

Definition 3 *An induced tree $c \in \mathcal{C}$ is defining starting from the root of $\mathcal{S}$ as follows:*

- *Include the root of $\mathcal{S}$ in c*
- *If q is a product node, include in c all children $i \in \mathrm{ch}(q)$. Continue with all children.*
- *If q is a sum node, include only one child $i \in \mathrm{ch}(q)$. Continue with the only chosen child.*
- *Then if q is a leaf, include it in c.*

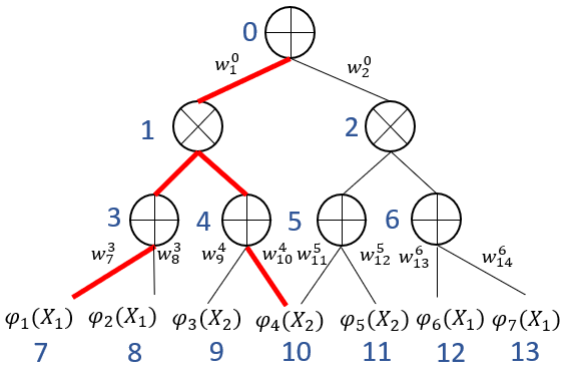

Figure 1: An induced tree in red.

An induced tree is represented in Fig. 1. Considering $i \in \mathrm{ch}(q)$ and $q \in \mathrm{Sum}(c)$, we denote $E(c)$ the set of all edges of c starting from a sum node q to one of its child i. Then the probability of belonging to an induced tree c is:

$$p(c|W) = \prod_{(q,i)\in E(c)} w_i^q$$

Considering the set of all leaves included in c, $\mathcal{L}(c)$ the likelihood of x at the root given an induced tree c is the product of the leaves evaluations included in $\mathcal{L}(c)$.

$$p(x|c,\theta) = \prod_{\ell\in\mathcal{L}(c)} \varphi_\ell(x|\theta_\ell)$$

In Fig. 1, with $X = (X_1, X_2)$ we get for the induced c in red : $p(c|W) = w_1^0 w_7^3 w_{10}^4$ and $p(x|\theta, c) = \varphi_1(X_1)\varphi_4(X_2)$

The fundamental property of induced trees (Dennis and Ventura, 2015) is that summing out all the induced trees, we obtain the likelihood at the root of the SPN $S(x|\theta)$.

Theorem 1 *Dennis and Ventura (2015)*

$$S(x|\theta, W) = \sum_{c\in\mathcal{C}} p(x, c|\theta) = \sum_{c\in\mathcal{C}} p(c)p(x|c, \theta) = \sum_{c\in\mathcal{C}} \prod_{(q,i)\in E(c)} w_i^q \prod_{\ell\in\mathcal{L}(c)} \varphi_l(x|\theta_\ell) \tag{1}$$

This property allows us to use induced trees as latent classes to estimate parameters of the SPN using the EM algorithm.

4. Gaussian Incomplete Data

One of the characteristics of real data is the existence of incomplete observations. Indeed, data are often only partially collected which could lead censorship and truncation. For all j in $[\![1; n]\!]$, $X_j = (X_{j,1}, \ldots, X_{j,N})$ is not observed but is known to lie into the interval $[a_{j,1}, b_{j,1}] \times \ldots \times [a_{j,n}, b_{j,N}]$ written with a slide abuse of notation $[a_j, b_j]$ with $a_j, b_j \in \mathbb{R}^N$. Four types of censoring can occur. For ℓ in $[\![1; N]\!]$:

- If $-\infty < a_{j,\ell} < b_{j,\ell} < \infty$, then $X_{j,l}$ is censored by interval that is, $X_{j,l} \in [a_{j,l}, b_{j,l}]$ with probability one.
- If $-\infty = a_{j,\ell} < b_{j,\ell} < \infty$, then $X_{j,l}$ is left-censored, that is $X_{j,l} \in]-\infty, b_{j,l}]$ with probability one.
- If $-\infty < a_{j,\ell} < b_{j,\ell} = \infty$, then $X_{j,l}$ is right-censored, that is $X_{j,l} \in [a_{j,l}, \infty[$ with probability one.
- If $a_{j,\ell} = -\infty$ and $b_{j,\ell} = +\infty$ the data provide no indication on $X_{j,\ell}$.

In the following we assume X_j to be a Gaussian vector and we assume non informative interval censoring in the same way as in Zhang et al. (2005). One can easily derive the Gaussian truncated density in one dimension. Considering one random variable $X_{j,\ell}$, then for $x_{j,\ell} \leqslant b_{j,\ell}$:

$$P(X_{j,\ell} \leqslant x_{j,\ell} | a_{j,\ell} \leqslant X_{j,\ell} \leqslant b_{j,\ell}) = \frac{P(a_{j,\ell} \leqslant X_{j,\ell} \leqslant x_{j,\ell})}{P(a_{j,\ell} \leqslant X_{j,\ell} \leqslant b_{j,\ell})} = \frac{\Phi_\ell(x_{j,\ell}) - \Phi_\ell(a_{j,\ell})}{\Phi_\ell(b_{j,\ell}) - \Phi_\ell(a_{j,\ell})}$$

With Φ_ℓ the cumulative distribution function of the random variable X_j. For $x_{j,\ell} > b_{j,\ell}$, this probability is one. In the Gaussian case, let φ_ℓ the Gaussian density of leave ℓ, then $\varphi_\ell(x) = \frac{1}{\sigma_\ell\sqrt{2\pi}} \exp\big(-\frac{1}{2\sigma_\ell^2}(x-\mu_\ell)^2\big)$.The density of the truncated variable X is of the form :

$$\varphi_\ell(x_{j,\ell} | a_{j,\ell} \leqslant X_{j,\ell} \leqslant b_{j,\ell}) = \frac{\varphi_\ell(x_{j,\ell})}{\Phi_\ell(b_{j,\ell}) - \Phi_\ell(a_{j,\ell})} \mathbb{1}\{x_{j,\ell} \in [a_{j,\ell}, b_{j,\ell}]\}$$

Integrating this density, one can compute the different moments of univariate truncated Gaussian density which will be useful for the EM updates in the next section. The expectation $m_{j,\ell}$ and

variance $V_{j,\ell}$ for observation j and leaf ℓ in the case of finite interval censoring are :

$$m_{j,\ell} = E[X_{j,\ell}|a_{j,\ell} \leqslant X_{j,\ell} \leqslant b_{j,\ell}] = \mu_\ell + \frac{\varphi_\ell(a_{j,\ell}) - \varphi_\ell(b_{j,\ell})}{\Phi_\ell(b_{j,\ell}) - \Phi_\ell(a_{j,\ell})}$$

$$V_{j,\ell} = V[X_{j,\ell}|a_{j,\ell} \leqslant X_{j,\ell} \leqslant b_{j,\ell}] = \sigma^2 + \frac{(a_{j,\ell} - \mu_\ell)\varphi_\ell(a_{j,\ell}) - (b_{j,\ell} - \mu_\ell)\varphi_\ell(b_{j,\ell})}{\Phi_\ell(b_{j,\ell}) - \Phi_\ell(a_{j,\ell})} - \Big(\frac{\varphi_\ell(a_{j,\ell}) - \varphi_\ell(b_{j,\ell})}{\Phi_\ell(b_{j,\ell}) - \Phi_\ell(a_{j,\ell})}\Big)^2$$

In the case of incomplete data, we still have forward and backward pass which are well defined and we will write these quantities $S_q([a_j, b_j])$ and $\frac{\partial S}{\partial S_q}([a_j, b_j])$. Moreover, for multivariate truncated Gaussian probability measure, there are no simple form of moments and numerical approximation are often used to compute mean and variance (BG and Wilhelm, 2009). The aim of the following section is to learn parameters of truncated Gaussian leaves in SPNs.

5. EM for Incomplete Data

In this section, we will derive a version of the Expectation-Maximisation algorithm for SPNs and apply it to incomplete data. The EM algorithm (Dempster et al., 1977) is an iterative algorithm especially effective for maximizing the likelihood in the context of hidden variables. In our example, there are two types of hidden variables: *the membership to an inducted tree* c is not observed and some observations X_{jl} are partially or not observed. At each iteration of EM, given the current update of the parameter $W_{\text{old}}, \theta_{\text{old}}$, we aim at maximizing a surrogate function Q with respect to W and θ of the form :

$$Q(W, \theta|W_{\text{old}}, \theta_{\text{old}}) = \sum_{j=1}^{n} \sum_{c \in \mathcal{C}} \int_{[a_j, b_j]} p(x_j, c|a_j, b_j, W_{\text{old}}, \theta_{\text{old}}) \log p(x_j, c, a_j, b_j|W, \theta) dx_j, \quad (2)$$

where the integral is N-multidimensional. The joint probability of latent class and observed variables can be expressed according to Theorem 1.

$$p(x_j, c, a_j, b_j|\theta, W) = p(x_j|c, a_j, b_j, \theta)p(a_j, b_j|W, \theta, c)p(c|W) \propto \prod_{(q,i) \in E(c)} w_i^q \prod_{\ell \in \mathcal{L}(c)} \varphi_\ell(x_j|a_j, b_j, \theta)$$

The posterior of latent classes given parameters of the previous iteration can be decomposed into :

$$\begin{aligned} p(x_j, c|a_j, b_j, W_{\text{old}}, \theta_{\text{old}}) &= p(x_j|c, a_j, b_j, W_{\text{old}}, \theta_{\text{old}})p(c|a_j, b_j, W_{\text{old}}, \theta_{\text{old}}) \\ &= \eta_j^c \prod_{\ell \in \mathcal{L}(c)} \varphi_l(x_j|a_j, b_j, \theta_{\text{old}}), \end{aligned}$$

with $\eta_j^c = p(c|a_j, b_j, W_{\text{old}}, \theta_{\text{old}})$ the probability of membership to an induced tree c given the old parameter which is a constant for the new parameters of interest. After some computation and removing notations of old parameters which are constants for W and θ (See Annex B), Q can be decomposed into two terms : one depending on W called $Q_W(W)$ and another term depending only on θ denoted $Q_\theta(\theta)$. This decomposition allows to separate the maximization with respect to W and θ.

$$Q(W, \theta|W_{\text{old}}, \theta_{\text{old}}) = Q_W(W) + Q_\theta(\theta) \quad (3)$$

5.1 Weights Updates

In this part, the parameter of interest is W and we are looking for $W^* = \arg\max_W Q_W(W)$ with (See B.1):

$$Q_W(W) = \sum_{q \in \text{Sum}(\mathcal{S})} \sum_{i \in \text{ch}(q)} \beta_i^q \log w_i^q \quad \text{and} \quad \beta_i^q = \sum_{j=1}^n w_{i,\text{old}}^q \frac{1}{S([a_j, b_j])} \frac{\partial S}{\partial S_q}([a_j, b_j]) S_i([a_j, b_j])$$

The quantity β_i^q is a constant with respect to W and only depends on the weights of the previous iteration, forward, backward passes which are computed in linear time of the network. Carrying out the optimization under constraints : $\forall q \in \text{Sum}(\mathcal{S}), \sum_{i \in \text{ch}(q)} w_i^q = 1$ we obtain :

$$w_i^q = \frac{\beta_i^q}{\sum_{i' \in \text{ch}(q)} \beta_{i'}^q} \tag{4}$$

This is exactly the same form of updates for weights updates as Desana and Schnörr (2016); Zhao et al. (2016) for complete data (see Desana and Schnörr, 2016, for more details). The only difference with complete data is that we do not evaluate forward and backward passes in a single point but for the interval $[a_j, b_j]$. The main difference between updates of complete and incomplete data lies in the updates of leaves.

5.2 Leaves Updates

Simplifying the quasi likelihood (see B.2), we obtain a function which only depends on θ of the form :

$$Q_\theta(\theta) = \sum_{j=1}^n \left(\sum_{\ell \in \mathcal{L}(\mathcal{S})} \alpha_{j,\ell} I_{j,\ell}(\theta) \right) \quad \text{with} \quad \alpha_{j,\ell} = \frac{1}{S([a_j, b_j])} \frac{\partial S}{\partial S_\ell}([a_j, b_j]) S_\ell([a_j, b_j])$$

$$\text{and} \quad I_{j,\ell}(\theta) = \int_{a_{j,\ell}}^{b_{j,\ell}} \varphi_\ell(x_{j,\ell} | a_{j,\ell}, b_{j,\ell}) \log \varphi_\ell(x_{j,\ell} | \theta, a_{j,\ell}, b_{j,\ell}) dx_{j,\ell}$$

Finally after some computation (see B.2), we derive the following updates of mean and variance of Gaussian leaves:

$$\mu_\ell = \frac{\sum_{j=1}^n \alpha_{j,l} m_{j,\ell}}{\sum_{j=1}^n \alpha_{j,\ell}} \qquad \sigma_\ell^2 = \frac{\sum_{j=1}^n \alpha_{j,\ell} \big(V_{j,\ell} + (m_{j,\ell} - \mu_\ell)^2\big)}{\sum_{j=1}^n \alpha_{j,\ell}} \tag{5}$$

They are a function of forward, backward passes, mean and variance of truncated density $m_{j,\ell}$ and $V_{j,\ell}$. For Gaussian leaves, we obtain a closed form of updates under the context of missing data. Simple expressions for other parametric leaves could be derived such as truncated exponential distribution but there exists no closed form for truncated binary variables for example. In this case, approximate numerical optimization would be necessary for this type of variables in the M step of the EM algorithm. In the following experimentation, we will focus only on Gaussian leaves but extension to other truncated parametric leaves could also be possible. Algorithm 1 summarizes all steps of the EM algorithm for SPNs in the context of incomplete data.

Algorithm 1: EM for a Gaussian SPN with incomplete data

Input: A fixed SPN with its graph $\mathcal{G}$, incomplete data $[a_j, b_j]$, and all forward $S_q([a_j, b_j])$ and backward passes $\frac{\partial S}{\partial S_q}([a_j, b_j])$.

for $q \in \mathrm{Sum}(\mathcal{S})$ *and* $i \in \mathrm{ch}(q)$ **do**

$\beta_i^q = w_i^q \sum_{j=1}^n \frac{1}{S([a_j,b_j])} \frac{\partial S}{\partial S_q}([a_j, b_j]))S_i([a_j, b_j])$

$w_i^q = \frac{\beta_i^q}{\sum_{i \in \mathrm{ch}(q)} \beta_i^q}$

for *each leaf* ℓ **do**

$\alpha_{j,\ell} \leftarrow \frac{1}{S(x)} \frac{\partial S}{\partial S_q}([a_j, b_j])S_\ell([a_j, b_j])$

$\mu_\ell = \frac{\sum_{j=1}^n \alpha_{j,l} m_{j,\ell}}{\sum_{j=1}^n \alpha_{j,\ell}}$

$\sigma_\ell^2 = \frac{\sum_{j=1}^n \alpha_{j,\ell}\left(V_{j,\ell} + (m_{j,\ell} - \mu_\ell)^2\right)}{\sum_{j=1}^n \alpha_{j,\ell}}$

6. Evaluation and Results

In this section we evaluate the performance of our estimation method on simulated data. The simulations are based on real data which are sampled (with replacement) and modified in order to generate a mixture of exact, interval-censored and missing data. Evaluation of the estimation parameters is then performed by comparing our estimates to the one based on the complete dataset.

Protocol: Starting from the complete dataset, a graph structure is learned using the LearnSPN algorithm (Gens and Pedro, 2013). In what follows, parameters estimation will be performed based on this structure. Then, we generate n samples by resampling with replacement the complete dataset.

For each of these samples, a proportion of interval-censored data is created given an observation $x_{j,\ell}$. We simulate the length of the censoring interval $L \sim \mathcal{U}[0, 5]$ and the position of the observation in the interval $t \sim \mathcal{U}[0, 1]$. Then the lower and upper bounds of the interval are : $a_{j,\ell} = x_{j,\ell} - tL$ and $b_{j,\ell} = x_{j,\ell} + (1 - t)L$. For left censoring (right censoring) we replace the left bound by $-\infty$ (respectively by $+\infty$). A proportion of missing data is also created by simply randomly removing some observations. As a result, two scenarios are considered:

- In the first simulation setting all the observations are interval-censored. In this setting, we will compare our estimator to the mid-point estimator which consists in replacing intervals by their midpoint and obtain exact data. Then the standard EM approach for SPNs is applied (Desana and Schnörr, 2016).

- The second simulation setting consists of a mixture of 5% of missing data, 5% of left censored data, 5% right censored data and 5% interval censored data. Two competitors are studied in this setting. The first one uses the DROP strategy where all incomplete observations are excluded and only exact data are kept. In the second one, interval-censored data are replaced by their midpoint. For left and right censored data, they are replaced by the interval bound: in case of left censoring, the interval $]-\infty, b_j]$ is replaced by b_j and in case of right-censoring, the interval $[a_j, \infty[$ is replaced by a_j. Finally missing data are replaced by the median of the observations in the corresponding leaf. This strategy is called *EM median/mid/detection.*

In those two simulation scenarios, our EM estimator for SPNs with incomplete data is also implemented. Our algorithm is initialized from the parameters learned via LearnSPN. Finally, in order to evaluate the performance of all estimators a criterion is constructed based on the EM estimator evaluated on the complete dataset Desana and Schnörr (2016). We denote by W^*, θ^* this estimator and we compute:

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}\left(\sum_{w\in W}(w^* - \hat{w}_i)^2 + \sum_{\mu\in\theta}(\mu^* - \hat{\mu}_i)^2 + \sum_{\sigma\in\theta}(\sigma^* - \hat{\sigma}_i)^2\right) = \text{MEAN}^2 + \text{VAR}$$

where $m = 500$ is the number of replications, and $\hat{w}, \hat{\mu}, \hat{\sigma}$ are the estimates obtained from one of the aforementioned estimation strategy. The second equality comes from the standard bias/variance decomposition. In our simulations, 500 replications were generated from different sample sizes on each dataset. The three real dataset come from UCI Machine Learning Repository and are the *Banknote* dataset composed of 4 Gaussian variables, the *Abalone* and the *Wine* dataset composed of 8 Gaussian variables. For all methods, MSE, squared mean and variance are reported in Table 2 for the first scenario and in Table 3 for the second scenario. The implementation was based on the SPFlow library (Molina et al., 2019).

Banknote dataset (4 variables)						
	Our method			EM midpoint		
	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR
n=300	$\mathbf{32.2 \pm 11.6}$	18.3 ± 2.78	13.9 ± 3.26	33.1 ± 11.7	19.0 ± 3.08	14.1 ± 3.93
n=500	$\mathbf{28.9 \pm 15.3}$	13.5 ± 4.0	15.4 ± 5.5	33.3 ± 16.1	16.1 ± 1.9	17.2 ± 4.4
n=1000	$\mathbf{12.9 \pm 9.9}$	4.4 ± 1.2	8.5 ± 1.3	23.3 ± 7.8	15.1 ± 0.9	8.2 ± 5.4
Wine dataset (8 variables)						
	Our method			EM midpoint		
	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR
n=500	$\mathbf{1125.7}$ $\mathbf{\pm 356.4}$	288.1 ± 18.7	837.6 ± 527.6	1369.1 ± 310.1	739.1 ± 12.1	630.0 ± 217.7
n=1000	$\mathbf{429.2}$ $\mathbf{\pm 178.9}$	53.7 ± 19.0	375.5 ± 97.9	461.1 ± 176.0	113.2 ± 18.3	347.9 ± 108.7
n=1500	$\mathbf{195.0}$ $\mathbf{\pm 32.2}$	27.0 ± 14.6	168.0 ± 18.0	434.0 ± 255.9	59.7 ± 16.9	374.3 ± 128.3
Abalone dataset (8 variables)						
	Our method			EM midpoint		
	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR
n=1000	$\mathbf{51.1 \pm 55.6}$	25.3 ± 6.8	25.8 ± 23.1	70.3 ± 4.4	61.1 ± 1.6	9.2 ± 7.9
n=2000	$\mathbf{23.4 \pm 5.2}$	18.3 ± 1.2	5.1 ± 0.6	68.6 ± 1.6	65.9 ± 2.3	2.7 ± 1.8
n=3000	$\mathbf{19.2 \pm 4.8}$	13.6 ± 3.3	5.6 ± 1.8	68.5 ± 1.9	63.9 ± 3.8	4.6 ± 2.4

Table 1: Result for 100% interval censoring datasets

Banknote dataset (4 variables)									
	Our method			EM median/mid/detection			DROP		
	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR
n=300	$\mathbf{31.9}$ $\pm\mathbf{14.8}$	12.7 ± 1.9	19.2 ± 6.26	48.2 ± 20.1	32.6 ± 0.9	15.6 ± 4.7	46.4 ± 13.6	30.0 ± 10.1	16.4 ± 8.1
n=500	$\mathbf{30.5}$ $\pm\mathbf{18.1}$	11.1 ± 4.5	19.4 ± 8.0	33.8 ± 29.6	11.0 ± 5.2	22.8 ± 16.8	42.7 ± 32.9	12.7 ± 5.19	30.0 ± 16.4
n=1000	$\mathbf{23.1}$ $\pm\mathbf{10.3}$	9.9 ± 1.4	13.2 ± 3.1	33.4 ± 12.0	17.7 ± 1.18	15.7 ± 2.8	33.7 ± 11.0	14.7 ± 1.42	19.0 ± 2.8
Wine dataset (8 variables)									
	Our method			EM median/mid/detection			DROP		
	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR
n=500	$\mathbf{222.0}$ $\pm\mathbf{172.9}$	52.8 ± 6.72	169.2 ± 114.3	1526.8 ± 173.1	766.7 ± 4.9	760.1 ± 309.4	1698.8 ± 1418.9	266.1 ± 27.1	1432.7 ± 919.4
n=1000	$\mathbf{139.1}$ $\pm\mathbf{109.4}$	31.4 ± 1.57	107.6 ± 52.3	1447.2 ± 22.2	1276.3 ± 9.81	170.9 ± 8.25	1208.7 ± 562.0	68.0 ± 20.3	1140.7 ± 374.7
n=1500	$\mathbf{72.8}$ $\pm\mathbf{26.5}$	12.4 ± 3.6	60.4 ± 30.0	1173.6 ± 40.5	1013.9 ± 1.9	159.7 ± 58.8	947.9 ± 412.4	22.1 ± 13.7	926.8 ± 299.7
Abalone dataset (8 variables)									
	Our method			EM median/mid/detection			DROP		
	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR	MSE	MEAN^2	VAR
n=1000	$\mathbf{33.1}$ $\pm\mathbf{30.4}$	6.3 ± 3.05	26.8 ± 19.9	106.2 ± 2.4	101.1 ± 1.2	5.1 ± 3.4	51.0 ± 45.2	13.4 ± 4.7	37.6 ± 26.6
n=2000	$\mathbf{10.8}$ $\pm\mathbf{10.3}$	3.1 ± 2.9	7.7 ± 5.6	103.7 ± 2.1	102.6 ± 0.8	1.1 ± 0.4	12.3 ± 6.3	6.6 ± 3.1	5.7 ± 4.7
n=3000	$\mathbf{3.2}$ $\pm\mathbf{2.1}$	1.7 ± 1.5	1.5 ± 1.1	93.4 ± 6.7	91.9 ± 3.2	1.5 ± 0.4	14.4 ± 13.8	2.5 ± 2.1	11.9 ± 9.9

Table 2: Results for "mixed interval censoring" datasets

In the first experiment with 100% interval-censored data, our EM estimator with incomplete data has better results than the midpoint strategy in terms of MSE. The variance of the mid point strategy is sometimes lower than the variance of our algorithm but has always a bigger bias.

In the second experiment with mixed interval censoring, the EM with incomplete data outperforms *DROP* and the *EM median/mid/detection* strategy in the three datasets for all sample size. When n increases, the DROP method outperforms the *EM median/mid/detection* which has a strong bias that does not decrease when the dataset becomes larger because of left and right censoring which are difficult to tackle. *DROP* method is often outperformed by our algorithm as it has higher variance as we drop several points.

7. Conclusion

We derived in this paper an algorithm to learn the parameters of SPNs in the context of incomplete Gaussian data using the EM algorithm. This approach gives closed solutions for the update parameters by adapting the EM algorithm to incomplete data. Experimental results based on three real datasets showed that our algorithm outperforms classic strategies in the context of incomplete data. This idea could be extended to other parametric leaves but without the assurance of having closed form of updates parameters. As a result, this would require some numerical approximations during the M step of the EM algorithm.

Appendix A.

These two properties are of capital importance and have been demonstrated in Desana and Schnörr (2016). They allow to link the expectation on an induced tree c to forward and backward passes flowing in edge (q, i) or in leaf ℓ and they allow fast computation of this expectation.

Theorem 2 *Desana and Schnörr (2016) Let* $\delta^c_{(i,q)} = 1$ *and* $\delta^c_\ell = 1$ *if respectively the edge* (i, q) *or the leaf* ℓ *belongs to an induced tree* c, 0 *otherwise then :*

$$\sum_{c\in\mathcal{C}} \eta^c_j \delta^c_{(i,q)} = w^q_{i,\text{old}} \frac{1}{S([a_j, b_j])} \frac{\partial S}{\partial S_q}([a_j, b_j]) S_i([a_j, b_j])$$
$$\sum_{c\in\mathcal{C}} \eta^c_j \delta^c_\ell = \frac{1}{S([a_j, b_j])} T_\ell([a_j, b_j]) S_\ell([a_j, b_j]) = \alpha_{j,\ell}$$

Appendix B.

Starting from the quasi likelihood and removing old parameter notations we obtain :

$$\begin{aligned}
Q(\theta, W) &= \sum_{j=1}^{n} \sum_{c\in\mathcal{C}} \eta^c_j \Big(\sum_{(q,i)\in E(c)} \log w^q_i + \int_{[a_j,b_j]} \varphi_l(x_j|a_j, b_j) \sum_{l\in\mathcal{L}(c)} \log \varphi_l(x_j|\theta_l) dx_j \Big) \\
&= \sum_{j=1}^{n} \sum_{c\in\mathcal{C}} \eta^c_j \Big(\sum_{(q,i)\in E(c)} \log w^q_i + \sum_{l\in\mathcal{L}(c)} \int_{a_{j,l}}^{b_{j,l}} \varphi_l(x_j|a_j, b_j) \log \varphi_l(x_j|\theta_l, a_j, b_j) dx_j \Big) \\
&= \sum_{j=1}^{n} \Big(\sum_{(q,i)\in E(\mathcal{S})} \log w^q_i \Big(\sum_{c\in\mathcal{C}} \delta^c_{(q,i)} \eta^c_j \Big) \Big) \\
&+ \Big(\sum_{l\in\mathcal{L}(\mathcal{S})} \int_{a_{j,\ell}}^{b_{j,\ell}} \varphi_\ell(x_{j,\ell}|a_{j,\ell}, b_{j,\ell}) \log \varphi_\ell(x_{j,\ell}|\theta, a_{j,\ell}, b_{j,\ell}) dx_{j,\ell} \Big(\sum_{c\in\mathcal{C}} \delta^c_j \eta^c_j \Big) \Big) \\
&= Q_W(W) + Q_\theta(\theta)
\end{aligned}$$

B.1 Weight updates

Considering $Q_W(W) = \sum_{j=1}^{n} \Big(\sum_{(q,i)\in E(\mathcal{S})} \log w_i^q \Big(\sum_{c\in\mathcal{C}} \delta^c_{(q,i)} \eta^c_j \Big) \Big)$ we use theorem 2 to express the sum over c with respect to forward and backward passes and we obtain :

$$Q_W(W) = \sum_{q\in \mathrm{Sum}(\mathcal{S})} \sum_{i\in ch(q)} \beta_i^q \log w_i^q \quad \text{with} \quad \beta_i^q = \sum_{j=1}^{n} w^q_{i,old} \frac{1}{S([a_j,b_j])} \frac{\partial S}{\partial S_q}([a_j,b_j]) S_i([a_j,b_j])$$

B.2 Leaves updates

Using theorem 2 to replace the sum over induced trees with the constant $\alpha_{j,\ell}$ we obtain :

$$\begin{aligned} Q(\theta) &= \sum_{j=1}^{n} \Big(\sum_{l\in\mathcal{L}(\mathcal{S})} \int_{a_{j,\ell}}^{b_{j,\ell}} \varphi_\ell(x_{j,\ell}|a_{j,\ell},b_{j,\ell}) \log \varphi_\ell(x_{j,\ell}|\theta,a_{j,\ell},b_{j,\ell}) dx_{j,\ell} \Big(\sum_{c\in\mathcal{C}} \delta^c_l \eta^c_j \Big) \Big) \\ &= \sum_{j=1}^{n} \Big(\sum_{l\in\mathcal{L}(\mathcal{S})} \alpha_{j,l} I_{j,l} \Big) \quad \text{with} \quad I_{j,\ell} = \int_{a_{j,\ell}}^{b_{j,\ell}} \varphi_\ell(x_{j,\ell}|a_{j,\ell},b_{j,\ell}) \log \varphi_\ell(x_{j,\ell}|\theta,a_{j,\ell},b_{j,\ell}) dx_{j,\ell} \end{aligned}$$

One can easily compute $I_{j,\ell} = -\log(\sigma_l) - \frac{(V_{j,l}+(m_{j,l}-\mu_l)^2}{2\sigma_l^2} + C$, with $C \in \mathbb{R}$ a constant. Then we compute the derivative of $Q_\theta(\theta)$ with respect to the two parameters of interest μ_ℓ and σ_ℓ.

$$\begin{aligned} \frac{\partial Q(\theta|\theta_{old})}{\partial \mu_\ell} &= \sum_{j=1}^{n} \sum_{\ell\in\mathcal{L}(S)} \alpha_{j,\ell} \frac{\partial I_{j,\ell}}{\partial \mu_\ell} = \sum_{j=1}^{n} \alpha_{j,\ell} \frac{(m_{j,\ell}-\mu_\ell)}{\sigma_\ell^2} = 0 \\ \frac{\partial Q(\theta|\theta_{old})}{\partial \sigma_\ell} &= \sum_{j=1}^{n} \sum_{\ell\in\mathcal{L}(\mathcal{S})} \alpha_{j,l} \frac{\partial I_{j,\ell}}{\partial \sigma_\ell} = \sum_{j=1}^{n} \alpha_{j,\ell} \Big(\frac{-1}{\sigma_\ell} + \frac{V_{j,\ell}+(m_{j,\ell}-\mu_\ell)^2}{\sigma_\ell^3} \Big) = 0 \end{aligned}$$

This gives the new updates :

$$\mu_\ell = \frac{\sum_{j=1}^{n} \alpha_{j,l} m_{j,\ell}}{\sum_{j=1}^{n} \alpha_{j,\ell}} \qquad \sigma_\ell^2 = \frac{\sum_{j=1}^{n} \alpha_{j,\ell} \big(V_{j,\ell} + (m_{j,\ell}-\mu_\ell)^2 \big)}{\sum_{j=1}^{n} \alpha_{j,\ell}}$$

References

M. BG and S. Wilhelm. Moments calculation for the double truncated multivariate normal density. *Available at SSRN 1472153*, 2009.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

A. Dennis and D. Ventura. Greedy structure search for sum-product networks. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

M. Desana and C. Schnörr. Learning arbitrary sum-product network leaves with expectation-maximization. *arXiv preprint arXiv:1604.07243*, 2016.

R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 3239–3247, 2012.

R. Gens and D. Pedro. Learning the structure of sum-product networks. In *International conference on machine learning*, pages 873–880, 2013.

J. Martens and V. Medabalimi. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

A. Molina, A. Vergari, K. Stelzner, R. Peharz, P. Subramani, N. D. Mauro, P. Poupart, and K. Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

R. Peharz. *Foundations of sum-product networks for probabilistic modeling*. PhD thesis, PhD thesis, Medical University of Graz, 2015.

R. Peharz, S. Tschiatschek, F. Pernkopf, and P. Domingos. On theoretical properties of sum-product networks. In *Artificial Intelligence and Statistics*, pages 744–752, 2015.

H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

A. Rashwan, H. Zhao, and P. Poupart. Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *Artificial Intelligence and Statistics*, pages 1469–1477, 2016.

M. Trapp, R. Peharz, H. Ge, F. Pernkopf, and Z. Ghahramani. Bayesian learning of sum-product networks. *CoRR*, abs/1905.10884, 2019a. URL `http://arxiv.org/abs/1905.10884`.

M. Trapp, R. Peharz, H. Ge, F. Pernkopf, and Z. Ghahramani. Bayesian learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 6344–6355, 2019b.

Z. Zhang, L. Sun, X. Zhao, and J. Sun. Regression analysis of interval-censored failure time data with linear transformation models. *Canadian Journal of Statistics*, 33(1):61–70, 2005.

H. Zhao, M. Melibari, and P. Poupart. On the relationship between sum-product networks and bayesian networks. In *International Conference on Machine Learning*, pages 116–124, 2015.

H. Zhao, P. Poupart, and G. J. Gordon. A unified approach for learning the parameters of sum-product networks. In *Advances in neural information processing systems*, pages 433–441, 2016.