Gaussian Sum-Product Networks Learning in the Presence of Interval Censored Data
Pierre Clavier, Olivier Bouaziz, Gregory Nuel

To cite this version:
Pierre Clavier, Olivier Bouaziz, Gregory Nuel. Gaussian Sum-Product Networks Learning in the Presence of Interval Censored Data. Probabilistic and Graphical Models, Sep 2020, Aalborg, Denmark. hal-02859894

HAL Id: hal-02859894
https://hal.science/hal-02859894
Submitted on 23 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Gaussian Sum-Product Networks Learning in the Presence of Interval Censored Data

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

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

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

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 Schnör, 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 Schnör, 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 $X$ a set of random variables which can be discrete or continuous, Sum-Product Networks denoted $S$ have been introduced by Poon and Domingos (2011) and can be defined as rooted Directed Acyclic Graphs (DAGs), $G(V,E)$ with distributions over random variables in $X$ 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 $X_q \subset X$. For any node $q$, its children are denoted $i \in \text{ch}(q)$ and the density of a node $S_q$ can be defined recursively as:

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

weights $w^q_i$ 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 : $\sum_{i \in \text{ch}(q)} w^q_i = 1$. In the following, we will only consider normalised SPNs. We write $\text{sc}(q)$ 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 $X$ and is said to be valid.

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

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

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

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

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

If all sum nodes of $S$ are complete, then the SPN $S$ 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 $S$ 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 $S$:

Let $\text{Sum}(S)$ be the set of all sum nodes of $S$. In this paper, we attempt to estimate the parameters of $S$ which are all the sum node weights included in $W = \{w_q^i \mid q \in \text{Sum}(S), i \in \text{ch}(q)\}$ and all the leaves parameters denoted as $\theta = \{\theta_\ell \mid \ell \in \mathcal{L}(S)\}$ with $\mathcal{L}(S)$ 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 $S$.

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 $S$ as follows:

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

An induced tree is represented in Fig. 1. Considering $i \in \text{ch}(q)$ and $q \in \text{Sum}(c)$, 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:

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

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: \( p(c|W) = w_1^0 w_2^3 w_4^4 \) 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)

\[
S(x|\theta, W) = \sum_{c \in \mathcal{C}} p(x, c|\theta) = \sum_{c \in \mathcal{C}} p(c)p(x|c, \theta) = \sum_{c \in \mathcal{C}} \prod_{(q,i) \in E(c)} w^q_i \prod_{\ell \in L(c)} \varphi_\ell(x|\theta_\ell) \tag{1}
\]

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 \( \ell \) in \([1; N]\):

- If \( -\infty < a_{j,\ell} < b_{j,\ell} < \infty \), then \( X_{j,\ell} \) is censored by interval that is, \( X_{j,\ell} \in [a_{j,\ell}, b_{j,\ell}] \) with probability one.
- If \( -\infty = a_{j,\ell} < b_{j,\ell} < \infty \), then \( X_{j,\ell} \) is left-censored, that is \( X_{j,\ell} \in ]-\infty, b_{j,\ell}] \) with probability one.
- If \( -\infty < a_{j,\ell} < b_{j,\ell} = \infty \), then \( X_{j,\ell} \) is right-censored, that is \( X_{j,\ell} \in [a_{j,\ell}, \infty[ \) 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 \( x_{j,\ell} \leq b_{j,\ell} \):

\[
P(X_{j,\ell} \leq x_{j,\ell} | a_{j,\ell} \leq X_{j,\ell} \leq b_{j,\ell}) = \frac{P(a_{j,\ell} \leq X_{j,\ell} \leq x_{j,\ell})}{P(a_{j,\ell} \leq X_{j,\ell} \leq b_{j,\ell})} = \frac{\Phi_\ell(x_{j,\ell}) - \Phi_\ell(a_{j,\ell})}{\Phi_\ell(b_{j,\ell}) - \Phi_\ell(a_{j,\ell})}
\]

With \( \Phi_\ell \) 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 \( \varphi_\ell \) the Gaussian density of leave \( \ell \), then

\[
\varphi_\ell(x) = \frac{1}{\sigma_\ell \sqrt{2\pi}} \exp \left( -\frac{1}{2 \sigma^2_\ell} (x - \mu_\ell)^2 \right). \text{The density of the truncated variable } X \text{ is of the form :}
\]

\[
\varphi_\ell(x_j,\ell | a_{j,\ell} \leq X_{j,\ell} \leq b_{j,\ell}) = \frac{\varphi_\ell(x_{j,\ell})}{\Phi_\ell(b_{j,\ell}) - \Phi_\ell(a_{j,\ell})} \mathbf{1}\{x_{j,\ell} \in [a_{j,\ell}, b_{j,\ell}]\}
\]

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
removing notations of old parameters which are constants for \( W \) parameter which is a constant for the new parameters of interest. After some computation and with \( \eta \) and \( W \) decomposed into two terms: one depending on \( \eta \) and \( \theta \).

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

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

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

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

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

\[
p(x_j|c, a_j, b_j, W_{\text{old}}, \theta_{\text{old}}) = p(x_j|c, a_j, b_j, W_{\text{old}}, \theta_{\text{old}})p(c|a_j, b_j, W_{\text{old}}, \theta_{\text{old}})
= \eta_j^c \prod_{\ell \in L(c)} \varphi_{\ell}(x_j|a_j, b_j, \theta_{\text{old}}),
\]

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 \( \theta \) (See Annex B), \( Q \) can be decomposed into two terms: one depending on \( W \) called \( Q_W(W) \) and another term depending only on \( \theta \) denoted \( Q_\theta(\theta) \). This decomposition allows to separate the maximization with respect to \( W \) and \( \theta \).

\[
Q(W, \theta|W_{\text{old}}, \theta_{\text{old}}) = Q_W(W) + Q_\theta(\theta) \tag{3}
\]
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):

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

The quantity $\beta^q_i$ 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: $\forall q \in \text{Sum}(\mathcal{S}), \sum_{i \in \text{ch}(q)} w^q_i = 1$ we obtain:

$$w^q_i = \frac{\beta^q_i}{\sum_{i' \in \text{ch}(q)} \beta^q_{i'}}$$

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 $\theta$ of the form:

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

and

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

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

$$\mu_{\ell} = \frac{\sum_{j=1}^n \alpha_{j,\ell} m_{j,\ell}}{\sum_{j=1}^n \alpha_{j,\ell}} \quad \text{and} \quad \sigma^2_{\ell} = \frac{\sum_{j=1}^n \alpha_{j,\ell} (V_{j,\ell} + (m_{j,\ell} - \mu_{\ell})^2)}{\sum_{j=1}^n \alpha_{j,\ell}}$$

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 $G$, incomplete data $[a_j, b_j]$, and all forward $S_q([a_j, b_j])$ and backward passes $\frac{\partial S_q}{\partial S_q}([a_j, b_j])$.

1. for $q \in \text{Sum}(S)$ and $i \in \text{ch}(q)$ do
2. $\beta^q_i = w^q_i \sum_{n=1}^n \frac{1}{S([a_j, b_j])} \frac{\partial S}{\partial S_q}([a_j, b_j]) S_i([a_j, b_j])$
3. $w^q_i = \frac{\beta^q_i}{\sum_{i \in \text{ch}(q)} \beta^q_i}$
4. for each leaf $\ell$ do
5. $\alpha_{j,\ell} \leftarrow \frac{1}{S(x)} \frac{\partial S}{\partial S_q}([a_j, b_j]) S_\ell([a_j, b_j])$
6. $\mu_\ell = \sum_{j=1}^n \alpha_{j,\ell} m_{j,\ell}$
7. $\sigma^2_\ell = \frac{\sum_{j=1}^n \alpha_{j,\ell} (V_{j,\ell} + (m_{j,\ell} - \mu_\ell)^2)}{\sum_{j=1}^n \alpha_{j,\ell}}$

---

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 $L \sim U[0, 5]$ and the position of the observation in the interval $t \sim U[0, 1]$. 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 Schnör, 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 $[a_j, -\infty)$ 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 $W^*, \theta^*$ this estimator and we compute:

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

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].

<table>
<thead>
<tr>
<th>Banknote dataset (4 variables)</th>
<th>Our method</th>
<th>EM midpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>MEAN$^2$</td>
</tr>
<tr>
<td>n=300</td>
<td>32.2 ± 11.6</td>
<td>18.3 ± 2.78</td>
</tr>
<tr>
<td>n=500</td>
<td>28.9 ± 15.3</td>
<td>13.5 ± 4.0</td>
</tr>
<tr>
<td>n=1000</td>
<td>12.9 ± 9.9</td>
<td>4.4 ± 1.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Wine dataset (8 variables)</th>
<th>Our method</th>
<th>EM midpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>MEAN$^2$</td>
</tr>
<tr>
<td>n=300</td>
<td>1125.7 ± 356.4</td>
<td>288.1 ± 18.7</td>
</tr>
<tr>
<td>n=1000</td>
<td>429.2 ± 178.9</td>
<td>53.7 ± 19.0</td>
</tr>
<tr>
<td>n=1500</td>
<td>195.0 ± 32.2</td>
<td>27.0 ± 14.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Abalone dataset (8 variables)</th>
<th>Our method</th>
<th>EM midpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MSE</td>
<td>MEAN$^2$</td>
</tr>
<tr>
<td>n=1000</td>
<td>51.1 ± 55.6</td>
<td>25.3 ± 6.8</td>
</tr>
<tr>
<td>n=2000</td>
<td>23.4 ± 5.2</td>
<td>18.3 ± 1.2</td>
</tr>
<tr>
<td>n=3000</td>
<td>19.2 ± 4.8</td>
<td>13.6 ± 3.3</td>
</tr>
</tbody>
</table>

Table 1: Result for 100% 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 midpoint 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 \( \ell \) 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 \( \ell \) belongs to an induced tree \( c \), 0 otherwise then :

\[
\sum_{c \in C} \eta^c_j \delta^c_{(i,q)} = w^q_{i,\text{old}} \frac{1}{S([a_j, b_j])} \frac{\partial S}{\partial S_q}([a_j, b_j])
\]

\[
\sum_{c \in C} \eta^c_j \delta^c_\ell = \frac{1}{S([a_j, b_j])} T_\ell([a_j, b_j]) = \alpha_{j,\ell}
\]

Appendix B.

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

\[
Q(\theta, W) = \sum_{j=1}^{n} \sum_{c \in C} \eta_j^c \left( \sum_{(q,i) \in E(c)} \log w^q_{i} \right) + \int_{[a_j, b_j]} \varphi_{\ell}(x_j|a_j, b_j) \sum_{l \in \mathcal{L}(c)} \log \varphi_{l}(x_j|\theta_l) dx_j
\]

\[
= \sum_{j=1}^{n} \sum_{c \in C} \eta_j^c \left( \sum_{(q,i) \in E(S)} \log w^q_{i} \right) + \sum_{l \in \mathcal{L}(c)} \int_{[a_j, b_j]} \varphi_{\ell}(x_j|a_j, b_j) \log \varphi_{l}(x_j|\theta_l, a_j, b_j) dx_j
\]

\[
= \sum_{j=1}^{n} \left( \sum_{(q,i) \in E(S)} \log w^q_{i} \left( \sum_{c \in C} \delta^c_{(q,i)} \eta_j^c \right) \right)
\]

\[
+ \left( \sum_{l \in \mathcal{L}(S)} \int_{[a_j, b_j]} \varphi_{\ell}(x_j,a_j,\ell, b_j,\ell) \log \varphi_{l}(x_j|a_j,\ell, b_j,\ell) dx_j \right) \left( \sum_{c \in C} \delta^c_j \eta_j^c \right)
\]

\[
= Q_W(W) + Q_\theta(\theta)
\]
B.1 Weight updates

Considering \( Q(W) = \sum_{j=1}^{n} \left( \sum_{(q,i) \in E(S)} \log w_i^q \left( \sum_{c \in C} \delta_{(q,i)}^c \eta_j^c \right) \right) \) we use theorem 2 to express the sum over \( c \) with respect to forward and backward passes and we obtain:

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

B.2 Leaves updates

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

\[
Q(\theta) = \sum_{j=1}^{n} \left( \sum_{\ell \in \ell(S)} \int_{a_j,\ell}^{b_j,\ell} \varphi_{\ell}(x_j,\ell|a_j,\ell, b_j,\ell) \log \varphi_{\ell}(x_j,\ell|\theta, a_j,\ell, b_j,\ell) dx_j,\ell \left( \sum_{c \in C} \delta_i^c \eta_j^c \right) \right)
= \sum_{j=1}^{n} \left( \sum_{\ell \in \ell(S)} \alpha_{j,\ell} I_{j,\ell} \right) \quad \text{with} \quad I_{j,\ell} = \int_{a_j,\ell}^{b_j,\ell} \varphi_{\ell}(x_j,\ell|a_j,\ell, b_j,\ell) \log \varphi_{\ell}(x_j,\ell|\theta, a_j,\ell, b_j,\ell) dx_j,\ell
\]

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

\[
\frac{\partial Q(\theta|\theta_{\text{old}})}{\partial \mu_{\ell}} = \sum_{j=1}^{n} \sum_{\ell \in \ell(S)} \alpha_{j,\ell} \frac{\partial I_{j,\ell}}{\partial \mu_{\ell}} = \sum_{j=1}^{n} \alpha_{j,\ell} \frac{(m_{j,\ell} - \mu_{\ell})}{\sigma_{\ell}^2} = 0
\]
\[
\frac{\partial Q(\theta|\theta_{\text{old}})}{\partial \sigma_{\ell}} = \sum_{j=1}^{n} \sum_{\ell \in \ell(S)} \alpha_{j,\ell} \frac{\partial I_{j,\ell}}{\partial \sigma_{\ell}} = \sum_{j=1}^{n} \alpha_{j,\ell} \left( -\frac{1}{\sigma_{\ell}^2} + \frac{V_{j,\ell} + (m_{j,\ell} - \mu_{\ell})^2}{\sigma_{\ell}^4} \right) = 0
\]

This gives the new updates:

\[
\mu_{\ell} = \frac{\sum_{j=1}^{n} \alpha_{j,\ell} m_{j,\ell}}{\sum_{j=1}^{n} \alpha_{j,\ell}} \quad \sigma_{\ell}^2 = \frac{\sum_{j=1}^{n} \alpha_{j,\ell} (V_{j,\ell} + (m_{j,\ell} - \mu_{\ell})^2)}{\sum_{j=1}^{n} \alpha_{j,\ell}}
\]

References


