Tractable Mode-Finding in Sum-Product Networks with Gaussian Leaves

Tiago Madeira[1], **Denis Deratani Mauá**[1]

[1]Institute of Mathematics and Statistics – University of São Paulo (USP)

{madeira,ddm}@ime.usp.br

Abstract. *In this work, we leverage the relation between Sum-Product Networks (SPNs) and Gaussian mixtures to propose an algorithm that adapts the Expectation-Maximization method to efficiently find the modes of SPNs with Gaussian leaves. We discuss how the algorithm can be used to perform Maximum-A-Posteriori inference in SPNs learned from continuous data with theoretical advantages over the existing methods in the literature, and how it can be used to shrink the size of learned models. As an additional example of the use of the algorithm, we perform an SPN-based hierarchical clustering of digit images. Thus, our proposed algorithm can be used for model analysis, model compression, and exploratory data analysis.*

1. Introduction

Sum-Product Networks (SPNs) are a relatively recent class of expressive statistical models, that exploit the use of arithmetic circuits [Darwiche 2003, Rooshenas and Lowd 2014] to efficiently represent complicated probability distributions. Due to their graphical structure, which encodes context-specific independence among random variables (RVs), SPNs can be considered probabilistic graphical models (PGMs) [Koller and Friedman 2009]. However, SPNs differ from other PGMs from an important computational perspective: unlike Bayesian Networks and Markov Networks, exact marginal and conditional probability inference (i.e, the computation of conditional probabilities over a set of RVs) in SPNs takes linear time with respect to the size of the network. The ability to capture a rich set of independences and produce reliable and fast inference has rendered SPNs a competitive approach for many challenging Machine Learning tasks [Poon and Domingos 2011, Llerena and Mauá 2017, Peharz et al. 2014, Cheng et al. 2014, Amer and Todorovic 2016].

SPNs are also akin to neural networks in the sense that an SPN is defined by a directed acyclic computation graph where each (inner) node computes a function of its input [Hsu et al. 2017]. SPNs learned from data can be considered deep models, due to their often large number of parameters and layers. However, SPNs distinguish themselves from other types of neural networks since their structure naturally deliver a principled probabilistic where each sub-network represents a joint distribution and standard probabilistic operations such as marginalization and conditioning are derived directly (and efficiently) by message-passing through the structure. SPNs can also be learned online [Lee et al. 2013, Jaini et al. 2016] and in distributed fashion [Rashwan et al. 2016].

In this work, we consider the special class of SPNs with Gaussian distributions at their leaves (input distribution), which we name Gaussian SPNs (GSPNs). GSPNs are compact representations of Gaussian Mixture Models (GMMs) with a large number of components, meaning that they are a convex combination of exponentially many Gaussian densities (with respect to the size of the respective SPN).

GMMs themselves are an expressive class of models for density estimation. In fact, according to [Carreira-Perpiñán 2000], the family of Gaussian mixtures is a universal approximator for continuous densities. Moreover, they inherit some advantages of the Gaussian distribution such as being analytically tractable for many types of computations. To our knowledge, the relation between GSPNs and GMMs has been so far unexplored in the literature.

Finding modes has several applications. We show how it can be used to perform Maximum-A-Posteriori (MAP) inference in SPNs learned from continuous data with theoretical advantages over the existing methods in the literature and present experiments of nonparametric hierarchical clustering via mode identification, which is an approach that can arguably be used for model analysis and to generate good and diverse representatives from continuous datasets. This method can also be used for model compression, as we recursively learn simpler and smaller models.

In this work, we leverage this relation and propose an algorithm adapting an EM-style method, namely Modal EM [Li et al. 2007], to find local maxima (modes) of GSPNs. We discuss the correctness and runtime complexity of our algorithm, and illustrate with an application of mode-finding in GSPNs for data exploration of the MNIST digit image dataset.

2. Sum-Product Networks

We start by giving a brief background on SPNs and discussing their relation with GMMs.

Given a function f over a set of random variables $\mathcal{X}$, we call the set $\mathcal{X}$ the *scope* of f and denote it as $\mathrm{scope}(f) := \mathcal{X}$. Then, a *Sum-Product Network (SPN)* is [Gens and Domingos 2013]:

- a univariate probability distribution; or
- a weighted sum of SPNs with the same scope and nonnegative weights; or
- a product of SPNs with disjoint scopes.

We assume without loss of generality that SPNs are normalized, i.e., weighted sums in the definition above add up to 1. That implies that SPNs specify normalized probability distributions over their scope [Zhao et al. 2016]. [Peharz 2015] proved that any SPN can be normalized in linear time with respect to its size.

An SPN is usually represented as a rooted directed acyclic graph (DAG) where leaves represent univariate distributions, internal nodes are associated with a $+$ (sum) or a $\times$ (product) operation, and edges pointing from sum nodes have nonnegative weights that sum up to 1.

As previously mentioned, an SPN represents a tractable probability distribution over its scope. Given an SPN $\mathcal{S}$, we denote its probability density function as $\mathcal{S}(\cdot)$. Given a random vector $\mathbf{X} = (X_1, \cdots, X_n)$, a valuation $\mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$ is called evidence. Given an evidence $\mathbf{x}$ we compute its density in the SPN $\mathcal{S}$ in linear time by traversing the SPN from the bottom up. For a node u in $\mathcal{S}$, let $S_u(\mathbf{x})$ denote the value of the node u in the SPN and $\mathrm{ch}(u)$ denote the children of u in the DAG. Then,

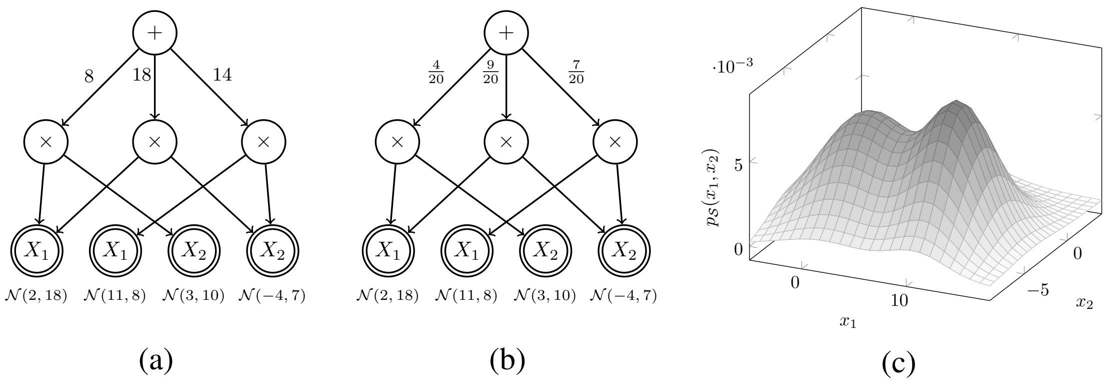

+
8
18
14
×
X_1
X_2
$\mathcal{N}(2,18)$ $\mathcal{N}(11,8)$ $\mathcal{N}(3,10)$ $\mathcal{N}(-4,7)$
(a)
$\frac{4}{20}$
$\frac{9}{20}$
$\frac{7}{20}$
(b)
$\cdot 10^{-3}$
$p_{\mathcal{S}}(x_1, x_2)$
5
0
0
10
x_1
0
−5
x_2
(c)

Figure 1. (a) An unnormalized GSPN $\mathcal{S}$. (b) A normal SPN $\mathcal{S}'$ representing the same distribution of $\mathcal{S}$. (c) Plot of the PDF of $\mathcal{S}$.

- if u is a leaf, $S_u(\mathbf{x})$ is the density of $\mathbf{x}$ in the univariate distribution;
- if u is a weighted sum of SPNs, $S_u(\mathbf{x}) = \sum_{v \in ch(u)} w(u,v) S_v(\mathbf{x})$;
- if u is a product of SPNs, $S_u(\mathbf{x}) = \prod_{v \in \mathrm{ch}(u)} S_v(\mathbf{x})$.

The density of $\mathbf{x}$ in the SPN $\mathcal{S}$ is the value of the root node, $\mathcal{S}(\mathbf{x}) := S_{\mathrm{root}(\mathcal{S})}(\mathbf{x})$. Computing a density takes linear time in the size of the network.

A *Gaussian Sum-Product Network (GSPN)* is a SPN in which all leaves are univariate Gaussian distributions. Figure 1 shows a representation of an unnormalized GSPN in (a), a normal GSPN representing the same distribution of it in (b), and a plot of the probability density function (PDF) of the distribution in (c).

2.1. Induced trees and Gaussian Mixture Models

By definition, SPNs represent mixture distributions. [Zhao et al. 2016] showed that any SPN is equivalent to a mixture of trees where each tree corresponds to a product of univariate distributions.

Given an SPN $\mathcal{S}$ over $X_1, \cdots, X_n$, let $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ be a subgraph of $\mathcal{S}$. $\mathcal{T}$ is called an *induced tree* from $\mathcal{S}$ if it can be constructed recursively, starting from the root node and then including all children of product nodes and exactly one child of sum nodes (with the corresponding edges). As proved by [Zhao et al. 2016], an induced tree $\mathcal{T}$ is an SPN, therefore $\mathcal{T}(\mathbf{X})$ represents a probability distribution. The density function of such distribution is given by:

$$\mathcal{T}(\mathbf{x}) = \prod_{(u,v) \in E_\mathcal{T}} w(u,v) \prod_{j=1}^{n} T_j(x_j), \tag{1}$$

where $w(u,v)$ is the weight of the edge $(u,v) \in E_\mathcal{T}$ if u is a sum node or 1 if u is a product node; $T_j(X_j)$ is the probability distribution of a leaf of $\mathcal{T}$ (note that $\mathcal{T}$ contains n leaves, one for each variable).

Let $\tau_\mathcal{S}$ denote the number of unique induced trees from $\mathcal{S}$, namely, its *network cardinality*, and $\mathcal{T}^i$ denote the i-th unique induced tree of $\mathcal{S}$. Then [Zhao et al. 2016],

$$\mathcal{S}(\mathbf{x}) = \sum_{i=1}^{\tau_\mathcal{S}} \mathcal{T}^i(\mathbf{x}). \tag{2}$$

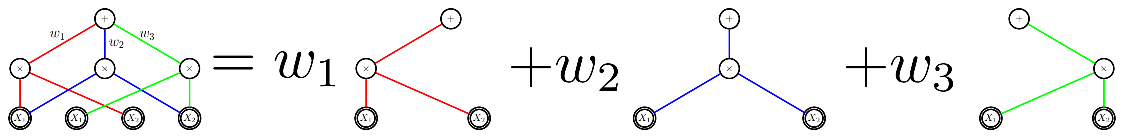


Figure 2. An SPN as a mixture of induced trees. Source: [Zhao et al. 2016].

This result is illustrated in Figure 2. The network cardinality of $\mathcal{S}$ depends on its structure and is exponential in the height of the SPN. Given an SPN $\mathcal{S}$, from equations 1 and 2 we have:

$$\mathcal{S}(\mathbf{x}) = \sum_{i=1}^{\tau_{\mathcal{S}}} w_i T^i(\mathbf{x}), \tag{3}$$

where $w_i := \prod_{(u,v)\in E_{\mathcal{T}_i}} w(u,v)$ and $T^i(\mathbf{x}) := \prod_{j=1}^{n} T^i_j(x_j)$ for all $i = 1, \cdots, \tau_{\mathcal{S}}$ (we are just splitting $\mathcal{T}_i$).

Let Z be the latent variable that corresponds to the mixture, i.e., $\mathcal{S}(\mathbf{x} \mid z) = T^z(\mathbf{x})$, and let $x_k, \cdots, x_l$ be values of RVs in $\mathbf{X}$. Then, for all $z \sim Z$, we have that

$$\mathcal{S}(x_k, \cdots, x_l \mid z) = T^z(x_k, \cdots, x_l) = T^z_k(x_k) \cdots T^z_l(x_l) = \mathcal{S}(x_k \mid z) \cdots \mathcal{S}(x_l \mid z), \tag{4}$$

which implies that the (observable) variables are independent given the mixture. If $\mathcal{S}$ is a GSPN, then $T^z(\mathbf{X})$ is a PDF formed by the product of the PDFs of independent Gaussian RVs. Thus, $T^z(\mathbf{X})$ is a multivariate Gaussian distribution with a diagonal covariance matrix. Therefore, a GSPN represents a GMM, where in each component the variables are uncorrelated. However, GSPNs have an exponential network cardinality (with respect to the height of the network); therefore, they represent GMMs with a huge number of components, which makes them much more expressive than the usual learned GMMs while still tractable.

3. Finding the Modes of Gaussian Sum-Product Networks

The problem of finding global or local maxima of a density function has been studied since long before the introduction of SPNs. The most common approach to solve it is performing hill-climbing algorithms [Russell and Norvig 2010, Carreira-Perpiñán 2000].

The density function we would like to maximize is an exponential mixture of multivariate Gaussian distributions, as shown in Section 2. [Carreira-Perpiñán and Williams 2003b, Carreira-Perpiñán and Williams 2003a] analyzed the number of modes of Gaussian mixtures and demonstrated that the number of modes can exceed the number of components of mixtures. That means that GSPNs can have an exponential number of modes, so finding the global optimum by local search would require a large number of restarts. [Améndola et al. 2019] argued that it is not known whether the number of modes is finite for general Gaussian mixtures. In the case of a finite number of modes, they proved an upper bound for the number of modes of a mixture of k Gaussians with n variables that is exponential in k and n: $2^{n+\binom{k}{2}}(5+3n)^k$. Thus, GSPNs can have over exponentially many modes on their size.

3.1. Modal EM

Investigating clustering of mixtures, [Li et al. 2007] introduced a hill-climbing method named *Modal EM* that solves a local maximum of a mixture density by ascending iterations starting from any initial point. It has some advantages over other existing methods such as being proved, ascending and very quick. Given a mixture density of τ components $p(\mathbf{x}) = \sum_k^\tau w_k p^k(\mathbf{x})$ and an initial value $\mathbf{x}^{(0)}$, Modal EM finds a local maximum of the mixture by alternating the following two steps (starting with $r = 0$):

$$\textbf{Expectation}\text{: Let } q_k = \frac{w_k p^k(\mathbf{x}^{(r)})}{p(\mathbf{x}^{(r)})}, \text{ for } k = 1, \cdots, \tau. \tag{5}$$

$$\textbf{Maximization}\text{: Compute } \mathbf{x}^{(r+1)} = \arg\max_{\mathbf{x}} \sum_k^\tau q_k \log p^k(\mathbf{x}). \tag{6}$$

In this work, we adapt that method to create an algorithm for efficiently computing a local maximum of the mixture represented by a GSPN. Each iteration of the algorithm takes $\Theta(n|\mathcal{S}|)$ where n is the number of random variables and $|\mathcal{S}|$ is the number of nodes in the GSPN. The network is traversed from the bottom up and each node propagates $2n$ values. The algorithm pseudo-code is shown in Algorithm 1.

Algorithm 1 Modal EM for GSPNs

```
Input: $\mathbf{x}^r$
Output: $\mathbf{x}^{(r+1)}$
for all node $v$ in reverse topological order do
    if $v$ is a leaf then
        ▷ Let $y$ be the RV in the scope of $v$; let $\mu$ and $\sigma$ be the parameters of the Gaussian of $v$; let $v(z)$ be the value of the Gaussian of $v$ at $z$.
        $N_y^v \leftarrow \frac{v(x_y^r)\mu}{\sigma^2}$, $D_y^v \leftarrow \frac{v(x_y^r)}{\sigma^2}$
        for all RV $z \neq y$ do
            $N_z^v \leftarrow D_z^v \leftarrow v(x_y^r)$
        end for
    else if $v$ is a product node then
        for all RV $z$ do
            $N_z^v \leftarrow \prod_{c \in \mathrm{ch}(v)} N_z^c$
            $D_z^v \leftarrow \prod_{c \in \mathrm{ch}(v)} D_z^c$
        end for
    else if $v$ is a sum node then
        for all RV $z$ do
            $N_z^v \leftarrow \sum_{c \in \mathrm{ch}(v)} w(v,c) N_z^c$
            $D_z^v \leftarrow \sum_{c \in \mathrm{ch}(v)} w(v,c) D_z^c$
        end for
    end if
end for
return $\frac{N^{root}}{D^{root}}$
```

We show next that this algorithm performs Modal EM in a GSPN by construction. If $S(\mathbf{x})$ is the density of a GSPN, then $T^k(\mathbf{x})$ ($k = 1, \cdots, \tau$) corresponds to the multiplication of the densities in the leaves of its k-th induced tree, as seen in Subsection 2.1. Let $T_i^k(x_i)$ the density of the leaf with scope i, $X_i \sim \mathcal{N}(\mu_{k_i}, \sigma_{k_i}^2)$, in the k-th induced tree. Then:

$$T^k(\mathbf{x}) = \prod_i^n T_i^k(x_i)\,, \tag{7}$$

where n is the number of RVs in the SPN. Therefore,

$$\begin{aligned} x^{(r+1)} &= \arg\max_{\mathbf{x}} \sum_k q_k \left(\log \prod_i T_i^k(x_i) \right) && (8)\\ &= \arg\max_{\mathbf{x}} \sum_k q_k \left(\sum_i \log T_i^k(x_i) \right) && (9)\\ &= \arg\max_{\mathbf{x}} \sum_k \sum_i \left(q_k \log T_i^k(x_i) \right) && (10)\\ &= \times_i \arg\max_{x_i} \sum_k \left(q_k \log T_i^k(x_i) \right). && (11) \end{aligned}$$

The last equation above states that each coordinate of the $x^{(r+1)}$ vector is obtained separately, by maximizing only over the corresponding dimension $x_i^{(r+1)}$. Hence, for a fixed i, we only need to find x_i that maximizes $g(x_i) := \sum_k q_k \log T_i^k(x_i)$.

The logarithm of the probability density function $f(x)$ of a Gaussian univariate distribution with mean μ and variance σ^2 is:

$$l(x) = \log f(x) = -\log(\sigma) - \frac{1}{2}\log(2\pi) - \frac{(x-\mu)^2}{2\sigma^2}. \tag{12}$$

The first and second derivatives of that function are, respectively:

$$l'(x) = \frac{\mu - x}{\sigma^2}, \text{ and } l''(x) = -\frac{1}{\sigma^2}. \tag{13}$$

That implies that $g(x_i)$ is a sum of quadratic functions with negative second derivative, therefore it has exactly one maximum. Its derivative is:

$$g'(x_i) = \sum_k^\tau \left(q_k \frac{\mu_{k_i} - x_i}{\sigma_{k_i}^2} \right). \tag{14}$$

Thus, to compute $x_i^{(r+1)} = \arg\max_{x_i} g(x_i)$ we can calculate the point where it is zero. By solving $g'(x_i) = 0$ we get:

$$x_i^{(r+1)} = \frac{\sum_k^\tau \frac{q_k \mu_{k_i}}{\sigma_{k_i}^2}}{\sum_k^\tau \frac{q_k}{\sigma_{k_i}^2}} = \frac{\sum_k^\tau \frac{w_k T^k(\mathbf{x}^{(r)}) \mu_{k_i}}{S(\mathbf{x}^{(r)}) \sigma_{k_i}^2}}{\sum_k^\tau \frac{w_k T^k(\mathbf{x}^{(r)})}{S(\mathbf{x}^{(r)}) \sigma_{k_i}^2}} = \frac{\sum_k^\tau \frac{\mu_{k_i}}{\sigma_{k_i}^2} w_k T^k(\mathbf{x}^{(r)})}{\sum_k^\tau \frac{1}{\sigma_{k_i}^2} w_k T^k(\mathbf{x}^{(r)})}. \tag{15}$$

That is the value, for each $i = 1, \cdots, n$, that we want to compute efficiently using the GSPN. Note that the numerator and the denominator of the fraction in Equation 15 are very similar to the evaluation of the GSPN in $\mathbf{x}^{(r)}$, $\mathcal{S}(\mathbf{x}^{(r)}) = \sum_k^\tau w_k T^k(\mathbf{x}^{(r)})$, but there is a constant (based in the parameters of the leaf of the RV i in the k-th induced tree) multiplying w_k in both cases; for the numerator, $\frac{\mu_{k_i}}{\sigma^2_{k_i}}$, and for the denominator, $\frac{1}{\sigma^2_{k_i}}$.

That is why our algorithm performs a bottom-up evaluation of the GSPN propagating $2n$ values for each node: for each RV, one to compute the numerator (N) and the other for computing the denominator (D) of the Equation 15. In the end, we just divide the vectors (N by D) to get $\mathbf{x}^{(r+1)}$.

Our implementation of Modal EM for GSPNs is released as open source in the RPCircuits.jl Julia package for learning and inference with SPNs.[1]

4. Applications

In this section we discuss some potential applications of computing the modes of GSPNs in Maximum-A-Posteriori (MAP) Inference and hierarchical clustering. As our intention here is not to establish state-of-the-art solutions for these tasks, we leave empirical comparisons with competing methods as future work.

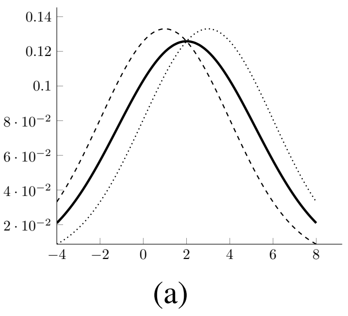
4.1. Maximum-A-Posteriori inference

SPNs are often build to solve structured prediction problems, where a structure solution is found by Maximum-A-Posteriori (MAP) inference in the model, that is, by searching for the most probable values for a set of RVs according to a probability distribution or for the global maximum configuration of the probability density function[2]. MAP inference is useful for many tasks, especially in applications that require reconstructing data such as image completion. Since the problem is $\mathcal{NP}$-Hard in SPNs [Peharz 2015, Peharz et al. 2016, Conaty et al. 2017], different greedy approximation algorithms have been proposed: the first was Max-Product [Poon and Domingos 2011] and then some variations of it such as Beam Search [Park 2002], ArgMax-Product [Conaty et al. 2017], and K-Best Tree [Mei et al. 2018]. [Mei et al. 2018] also proposed an exact algorithm that performs an exhaustive search in the space of solutions with a combination of pruning, heuristic and optimization techniques. More recently, [Mauá et al. 2020] proposed two reformulation approaches to MAP inference in SPNs: one casting the problem as a similar inference in Bayesian networks and other casting it as a mixed-integer linear programming for which there exists efficient solvers.

The above mentioned works focus on discrete SPNs (although some are easily extensible to continuous domains) and do not report experiments with continuous data; most importantly, they usually search only the limited space of modes of the distributions at the leaves and thus deliver solutions of unknown quality, which are unable to ensure

[1]Available at `https://github.com/RenatoGeh/RPCircuits.jl`.

[2]Strictly, MAP problem considers three disjoint sets $\mathbf{X}^q$, $\mathbf{X}^0$ and $\mathbf{X}^m$ such that $\mathbf{X} = \mathbf{X}^q \cup \mathbf{X}^0 \cup \mathbf{X}^m$ and consists in finding the most probable configuration for $\mathbf{X}^q$ given an evidence x^0 on variables $\mathbf{X}^0$ and ignoring (marginalizing) the RVs in $\mathbf{X}^m$. However, [Mei et al. 2018] proved that, for SPNs, every such problem can be reduced to a special case of MAP inference without evidence and RVs to marginalize in linear time in the size of the network. That is why we can say it consists in simply finding the global maximum of a probability density function.



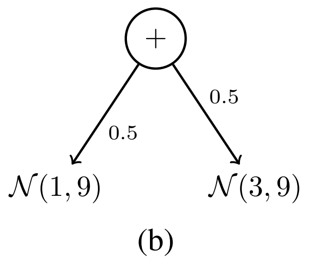


Figure 3. (a) Plot of the PDFs of three distributions: $X \sim \mathcal{N}(1,9)$ **(dashed line);** $X \sim \mathcal{N}(3,9)$ **(dotted line);** $X \sim 0.5\mathcal{N}(1,9) + 0.5\mathcal{N}3,9$ **(solid line). (b) GSPN representing the univariate GMM** $X \sim 0.5\mathcal{N}(1,9) + 0.5\mathcal{N}(3,9)$**. The existing algorithms are unable to find its optimal MAP,** $X = 2$**.**

even local optimality (i.e., they do not necessarily find a mode of the joint density). To see why searching over the modes of the leaves might fail at finding a local maximal, consider the simple univariate GSPN in Figure 3, shown as a circuit and the respective density. As can be seen from the left plot, the maximum of the density occurs at $X = 2$ ($p(2) \approx 0.125$). However, greedy algorithms such as MaxProduct [Poon and Domingos 2011] and K-Best Tree [Mei et al. 2018] find either solutions $X = 1$ or $X = 3$ ($p(1) = p(3) \approx 0.119$).

We can use the Modal EM algorithm to improve MAP solutions as follows. First, Modal EM can be run starting with the solution found by any earlier mentioned algorithm. That way we can guarantee that we are finding a stationary point; in the previous univariate GSPN example, that would result in us finding the optimal MAP. Second, we can run Modal EM from several input data points (e.g., a subsample of the training data set) to find different modes of the density function; then we can compare their likelihoods and pick the maximum. That is similar to the one of the methods proposed in [Llerena and Mauá 2017] for discrete SPNs. Note that such an approach improves on any given MAP Inference algorithm for GSPNs with only a small overhead.

4.2. Hierarchical clustering

[Li et al. 2007] proposed using Modal EM in Gaussian Mixture Models as a way to perform semi-parametric clustering. Their approach is to consider that a cluster is formed by the instances that ascend to the same mode of the density function. Modes are arguably good representatives of clusters, since each of them correspond to the local maximum of all points of a cluster. Since there are many modes in Gaussian mixtures, this method is extended for hierarchical clustering by recursively learning models from the modes found in the previous iterations. In their work, they use kernel density estimators with increasing bandwidths.

As an illustrative example of extending such an idea to GSPNs, which arguably capture a more flexible and expressive class of statistical models, we show empirical results of iteratively learning new GSPNs from the modes found by Modal EM in the previously model. That way we iteratively learn simpler and smaller models using the modes of the previously learned models, which arguably provide a representative summary of the data and model. This is therefore connected to two applications: hierarchical semi-parametric clustering and model compression.

Table 1. SPNs learned from MNIST-0 training set at iteration 1 and modes found in the first iteration at iteration 2. For each iteration, the tables show the number of instances used to learn the model, the network size (given by the number of nodes in the SPN), the number of clusters as found by running Modal EM starting from every point in the training set, and the log of the average likelihood for the test set in the learned SPN.

Iteration	# Instances	Network size	# Clusters	Log Avg. Likelihood
1	5,923	74,556	201	7,707
2	201	7,851	10	3,438

We use the classical MNIST database of handwritten digits. The dataset contains 60,000 28x28 grayscale images of the 10 digits (0–9) in the training set and 10,000 images in the test set. Each of the 784 pixels contain an integer value ranging from 0 to 255. We considered only the images of the digit 0 from the MNIST database. In such a subset, which we will call MNIST-0, there are 5,923 images in the training set and 980 images in the test set.

We learned a sequence of GSPNs using the LearnSPN implementation in open-source Python library SPFlow [Molina et al. 2019],[3] as follows. We first learn an initial GSPN from the MNIST-0 dataset, and use our Modal EM algorithm with each datapoint in the training set to obtain a set of representative summaries (modes). Points that "converge" to the same mode are assigned to the same clustering. The set of modes obtained form a new dataset, from which we learn a new GSPN and the process repeats until the number of modes is sufficiently small. In our experiments, two iterations sufficed to obtain a small set of models/clusters.

The results are shown in Table 1. We report the size of the network and the log of the average likelihood of the test set in the learned SPN. That way we can see how good the model is to represent the examples in the test set. Figure 4 shows the hierarchical clustering obtained by the process for the MNIST-0 dataset. For clarity, we omit the initial 5,923 instances of the training dataset, and show only the modes found by Modal EM in the first and second iterations. Even in a network with about 10% of the size of the original network, the modes appear to be good representatives of the diversity of the dataset.

One drawback of such an approach is that high density regions (or equivalent, large cluster) are underrepresented in the new dataset, while low density regions (small cluster) are overrepresented. We could mitigate such issues by optimizing for weighted log-likelihood in the structural and parametric learning algorithms, or more simply by over/undersampling models according to the respective region's density. We leave for future work an empirical comparison of such ideas.

5. Conclusions and future work

Sum-Product Networks (SPNs) are a class of statistical models that generalize Finite Mixture Models while retaining nice computational properties such as linear-time marginal inference and parameter learning. As with Finite Mixture Models, finding the modes of such densities remains a challenging problem.

[3]Available at `https://github.com/SPFlow/SPFlow`.

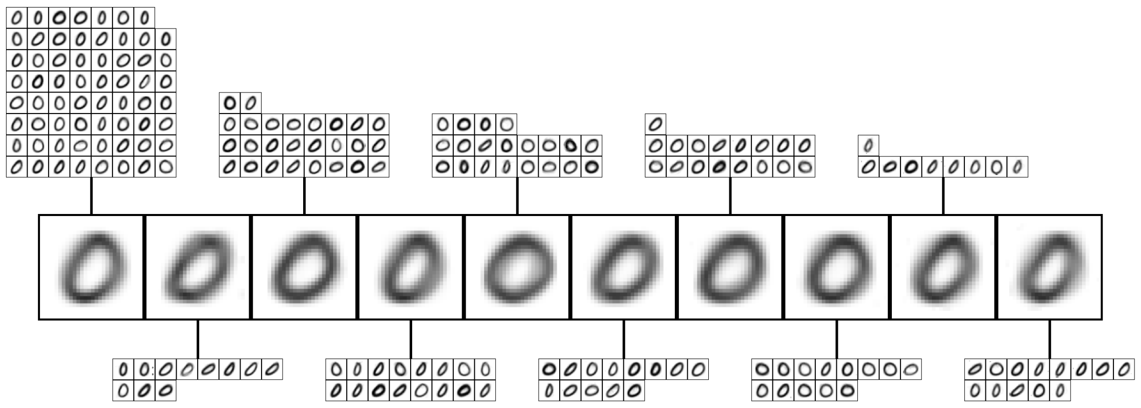


Figure 4. Hierarchical clustering: Modes (representatives of the clusters) found in 2 iterations of Modal EM in GSPNs learned from MNIST-0 dataset. The smaller images correspond to the modes found in the first GSPN, the bigger ones correspond to the modes found in the second GSPN (learned from the modes from the first GSPN). Modes from the first GSPN are connected to their modes in the second GSPN.

Based on previous work on Gaussian Mixture Models, in this work we developed a linear-time algorithm for finding the local maxima of a joint density distribution described as a Sum-Product-Network with Gaussian leaves. To our knowledge, the proposed algorithm, Model EM (named from its relation to Expectation-Maximization approach of latent variable models), is the first work specially desgined to address the problem of finding modes in continuous Sum-Product-Networks.

After developing the algorithm and analysing the correctness and time efficiency of the algorithm, we discuss some practical applications of mode finding in performing MAP inference, hierarchical clustering and data/model compression. For MAP inference, we argued that Modal EM can be used to improve the solution of any existing algorithm, leading to an algorithm which provably finds local optimal (a property most current algorithms lack). For hierarchical clustering, we performed an illustrative experiment with images of the digit zero from the MNIST dataset that described how the approach can be used to categorize, explore or compress data.

We leave as future work a more extensive empirical comparison of Modal EM in such applications. Another avenue to explore in future work is the use of Modal EM for discrete SPNs. In principle, we can treat Bernoulli leaves in discrete Sum-Product-Networks as continuous variables and apply the same formulas to derive an approximate algorithm for local maximizers of the joint distributions. The solutions obtained by the algorithm then can be either rounded or fed into another algorithm to obtain valid solutions.

Acknowledgments

This work was partly supported by the FAPESP-IBM grant #2019/07665-4 and CNPq PQ grant #304012/2019-0.

References

Améndola, C., Engström, A., and Haase, C. (2019). Maximum Number of Modes of Gaussian Mixtures. *Information and Inference: A Journal of the IMA*.

Amer, M. R. and Todorovic, S. (2016). Sum Product Networks for Activity Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(4):800–813.

Carreira-Perpiñán, M. Á. (2000). Mode-finding for mixtures of gaussian distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1318–1323.

Carreira-Perpiñán, M. Á. and Williams, C. K. (2003a). An isotropic Gaussian mixture can have more modes than components. *Institute for Adaptive and Neural Computation*, 4(2).

Carreira-Perpiñán, M. Á. and Williams, C. K. (2003b). On the number of modes of a Gaussian mixture. In Griffin, L. D. and Lillholm, M., editors, *Scale Space Methods in Computer Vision*, volume 2695, pages 625–640, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cheng, W. C., Kok, S., Pham, H. V., Chieu, H. L., and Chai, K. M. A. (2014). Language modeling with sum-product networks. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 2098–2102.

Conaty, D., Mauá, D. D., and de Campos, C. P. (2017). Approximation Complexity of Maximum A Posteriori Inference in Sum-Product Networks. In Elidan, G. and Kersting, K., editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*, pages 322–331. AUAI Press.

Darwiche, A. (2003). A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305.

Gens, R. and Domingos, P. (2013). Learning the structure of sum-product networks. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, pages 873–880, Atlanta, GA, USA. PMLR.

Hsu, W., Kalra, A., and Poupart, P. (2017). Online Structure Learning for Sum-Product Networks with Gaussian Leaves.

Jaini, P., Rashwan, A., Zhao, H., Liu, Y., Banijamali, E., Chen, Z., and Poupart, P. (2016). Online Algorithms for Sum-Product Networks with Continuous Variables. In Antonucci, A., Corani, G., and Campos, C. P., editors, *Proceedings of the Eighth International Conference on Probabilistic Graphical Models*, pages 228–239, Lugano, Switzerland. PMLR.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*.

Lee, S.-W., Heo, M.-O., and Zhang, B.-T. (2013). Online Incremental Structure Learning of Sum–Product Networks. In *LNCS*, volume 8227, pages 220–227.

Li, J., Ray, S., and Lindsay, B. G. (2007). A Nonparametric Statistical Approach to Clustering via Mode Identification. *Journal of Machine Learning Research*, 8(59):1687–1723.

Llerena, J. V. and Mauá, D. D. (2017). On using sum-product networks for multi-label classification. In *Proceedings of the 6th Brazilian Conference on Intelligent Systems*, BRACIS, pages 25–30.

Mauá, D. D., Ribeiro, H. R., Katague, G. P., and Antonucci, A. (2020). Two Reformulation Approaches to Maximum-A-Posteriori Inference in Sum-Product Networks. In Jaeger, M. and Nielsen, T. D., editors, *Proceedings of the Tenth International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, pages 293–304. PMLR.

Mei, J., Jiang, Y., and Tu, K. (2018). Maximum A Posteriori Inference in Sum-Product Networks. In *AAAI Conference on Artificial Intelligence*.

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

Park, J. D. (2002). MAP Complexity Results and Approximation Methods. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 388–396, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Peharz, R. (2015). *Foundations of Sum-Product Networks for Probabilistic Modeling*. PhD thesis, Graz University of Technology.

Peharz, R., Gens, R., Pernkopf, F., and Domingos, P. (2016). On the Latent Variable Interpretation in Sum-Product Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(10):2030–2044.

Peharz, R., Kapeller, G., Mowlaee, P., and Pernkopf, F. (2014). Modeling speech with sum-product networks: Application to bandwidth extension. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 3699–3703. IEEE.

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

Rashwan, A., Zhao, H., and Poupart, P. (2016). Online and distributed Bayesian moment matching for parameter learning in sum-product networks. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016*, pages 1469–1477.

Rooshenas, A. and Lowd, D. (2014). Learning sum-product networks with direct and indirect variable interactions. In *31st International Conference on Machine Learning, ICML 2014*, pages I–710–I–718, Beijing, China. JMLR.org.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence A Modern Approach*. 3 edition.

Zhao, H., Poupart, P., and Gordon, G. (2016). A unified approach for learning the parameters of Sum-Product Networks. In *Advances in Neural Information Processing Systems*, pages 433–441. Curran Associates, Inc.