Neural Networks 164 (2023) 645–666

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Review

A survey of sum–product networks structural learning

Riting Xia [a,b], Yan Zhang [d], Xueyan Liu [a,c,*], Bo Yang [a,c,*]

[a] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[b] College of Artificial Intelligence, Jilin University, Changchun, Jilin, 130012, China
[c] College of Computer Science and Technology, Jilin University, Changchun, Jilin, 130012, China
[d] College of Communication Engineering, Jilin University, Changchun, Jilin, 130012, China

ARTICLE INFO

Article history:
Received 5 December 2022
Received in revised form 17 March 2023
Accepted 7 May 2023
Available online 12 May 2023

Keywords:
Sum–product networks
SPN structure learning
Deep neural models
Probabilistic graphical models
Tractable
Expressive efficiency

ABSTRACT

Sum–product networks (SPNs) in deep probabilistic models have made great progress in computer vision, robotics, neuro-symbolic artificial intelligence, natural language processing, probabilistic programming languages, and other fields. Compared with probabilistic graphical models and deep probabilistic models, SPNs can balance the tractability and expressive efficiency. In addition, SPNs remain more interpretable than deep neural models. The expressiveness and complexity of SPNs depend on their own structure. Thus, how to design an effective SPN structure learning algorithm that can balance expressiveness and complexity has become a hot research topic in recent years. In this paper, we review SPN structure learning comprehensively, including the motivation of SPN structure learning, a systematic review of related theories, the proper categorization of different SPN structure learning algorithms, several evaluation approaches and some helpful online resources. Moreover, we discuss some open issues and research directions for SPN structure learning. To our knowledge, this is the first survey to focus specifically on SPN structure learning, and we hope to provide useful references for researchers in related fields.

© 2023 Elsevier Ltd. All rights reserved.

Contents

1. Introduction ... 646
2. Preliminaries ... 648
   2.1. Mathematical preliminaries ... 648
      2.1.1. Basic definitions about graphs ... 648
      2.1.2. Network polynomial ... 648
      2.1.3. Arithmetic circuits ... 649
   2.2. Sum–product networks ... 649
      2.2.1. Basic definitions of SPNs ... 649
      2.2.2. Inference of SPNs ... 650
3. SPN structure learning based on different method types ... 651
   3.1. Handcrafted structure learning ... 652
   3.2. Data-based structure learning ... 652
      3.2.1. Heuristic algorithms ... 652
      3.2.2. Non-heuristic algorithms ... 655
   3.3. Random structure learning ... 655
      3.3.1. The random structure learning method ... 655
      3.3.2. Some other methods related to random structure learning ... 656
   3.4. Ensemble structure learning ... 656
   3.5. Summary and discussion ... 657
4. SPN structure learning based on data types ... 658
   4.1. SPN structure learning built on general data ... 658
   4.2. SPN structure learning built on online data ... 658
   4.3. SPN structure learning built on sequence data ... 658

* Corresponding authors at: College of Computer Science and Technology, Jilin University, Changchun, Jilin, 130012, China.
E-mail addresses: xiart19@mails.jlu.edu.cn (R. Xia), yanz19@mails.jlu.edu.cn (Y. Zhang), xueyanliu@jlu.edu.cn (X. Liu), ybo@jlu.edu.cn (B. Yang).

https://doi.org/10.1016/j.neunet.2023.05.010
0893-6080/© 2023 Elsevier Ltd. All rights reserved.

4.4. SPN structure learning built on relational data 659
4.5. Summary and discussion 659
5. SPN structure learning in practice 659
5.1. Datasets 659
5.1.1. General datasets 659
5.1.2. Online datasets 661
5.1.3. Relational datasets 661
5.1.4. Sequence datasets 661
5.2. Evaluation tasks 661
5.2.1. Log-likelihood 661
5.2.2. Node classification 661
5.2.3. Link prediction 662
5.2.4. Learning and inference time 662
5.3. Open source software 662
6. Discussion 662
7. Conclusion 663
CRediT authorship contribution statement 663
Declaration of competing interest 663
Data availability 663
Acknowledgments 663
References 663

1. Introduction

Probabilistic models (PMs) (Rao, Olshausen, & Lewicki, 2002; Xiao et al., 2018) aim to model the data-generating distribution and answer statistical queries through probability inference. Traditional PMs, such as probabilistic graphical models (PGMs) (Koller & Friedman, 2009), use graph structures to represent the dependence between variables, which have strong logicality and interpretability. However, PGMs do not strike a good balance between inference and expressivity (for many distributions, especially relatively complex ones, PGMs require approximate inference). Recently, a novel deep probabilistic model, named Sum–product Network (SPN) was proposed (Poon & Domingos, 2011). Compared with PGMs, Sum–product Networks (SPNs) are expressive efficient[1] (Martens & Medabalimi, 2014) and good at tractable[2] inference, becoming an emerging research hotspot.

SPNs have many advantages compared with neural networks (NNs), PMs and other deep probabilistic models, such as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), Variational Autoencoders (VAEs) (Kingma & Welling, 2014), and Normalizing Flows (NFs) (Kingma & Dhariwal, 2018). In order to help readers better understand SPNs, we compare NNs, PGMs, and SPNs from four aspects: structure (nodes and edges), interpretability, expressive ability and inference, as shown in Table 1. In general, SPNs mainly have the following three advantages: (1) SPNs are deep probabilistic models with complete probabilistic semantics. Different from most NNs (NNs are "black boxes" whose neurons have no intuitive interpretation.), such as deep neural networks (DNNs), each node in SPNs calculates the local (unnormalized) probability for the variables in its scope, which makes each node in SPNs have clear semantics and certain interpretability. (2) SPNs are expressive efficient, which means that SPNs can characterize various types of probability distribution functions both compactly and effectively. Many compact distributions cannot be represented in the compact form of PGMs but SPNs can (Poon & Domingos, 2011). (3) The inference of SPNs is tractable over both shallow models and high treewidth models.

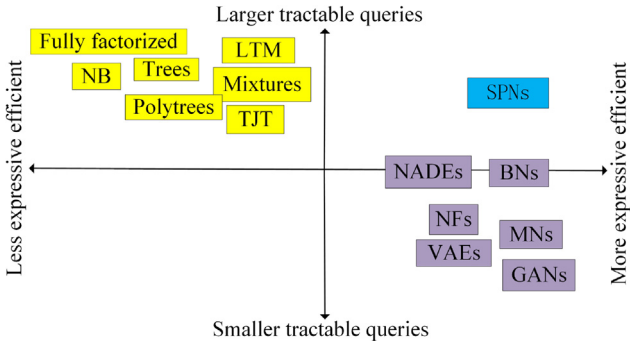


Fig. 1. The expressive efficiency and tractability of different models. *Source:* Image extracted from Vergari et al. (2020).

Although some PMs are also tractable in terms of inference, they are shallow models whose expressive efficiency is limited. In Fig. 1, we show the comparison (which was summarized by Vergari, Choi, Peharz, and Van den Broeck (2020)) of the expressive efficiency and tractability of SPNs with other PMs and deep probabilistic models, including completely disconnected graph (Fully factorized), a convex combination of k (simpler) probabilistic models (Mixtures), latent tree models (LTM) (Choi, Tan, Anandkumar, & Willsky, 2011), Thin Junction Trees (TJT) (Bach & Jordan, 2001), learning with mixtures of trees (Trees) (Meila & Jordan, 2000), Polytrees (Dasgupta, 1999), Naive Bayes (NB), Markov networks (MNs), Bayesian networks (BNs), GANs, VAEs, NFs and Neural Autoregressive Distribution Estimator (NADEs) (Larochelle & Murray, 2011). It can be concluded from Fig. 1 that expressive models (the bottom-right models in Fig. 1) like GANs are not tractable and tractable models (the top-left models in Fig. 1), like Fully factorized, are not expressive. However, SPNs are both expressive and tractable. Due to the above advantages, SPNs have been used in various tasks such as graph representation learning (Xia, Zhang, Liu, & Yang, 2023), causal inference (Zečević, Dhami, Karanam, Natarajan, & Kersting, 2021), computer vision (van de Wolfshaar & Pronobis, 2019), neuro-symbolic artificial intelligence (Skryagin et al., 2020), natural language processing (Cheng, Kok, Pham, Chieu, & Chai, 2014; Peharz, Kapeller, Mowlaee and Pernkopf, 2014b) and robotics (Zheng, Pronobis, & Rao, 2018).

Different from the structure of PGMs, the structure of SPNs are computational graphs similar to NNs (Vergari et al., 2020)

[1] Expressive efficient: Expressive efficient refers to the ability to compactly represent many types of functions, and to ensure that the probability distribution functions which are represented by SPNs are effective.

[2] Tractable: The probability value of the query can be calculated accurately and quickly.

Table 1
Comparison of PGMs, NNs and SPNs.

Models	PGMs	NNs	SPNs
Nodes	Random variables	Unit of computations (neurons)	Unit of computations (local probability)
Edges	Dependencies	Order of execution	Order of execution
Interpretability	Good interpretability	Poor interpretability	Certain interpretability
Expressive ability	Less expressiveness	Expressiveness	Expressive efficiency
Inference	Conditioning (Darwiche, 2001; Sang, Beame, & Kautz, 2005) Message propagation (Choi & Darwiche, 2010; Sontag, Globerson, & Jaakkola, 2011) Elimination (Dechter, 1998; Zhang & Poole, 1994) (intractable for expressive models)	Feedforward propagation Backward propagation	Feedforward propagation Backward propagation (tractable)

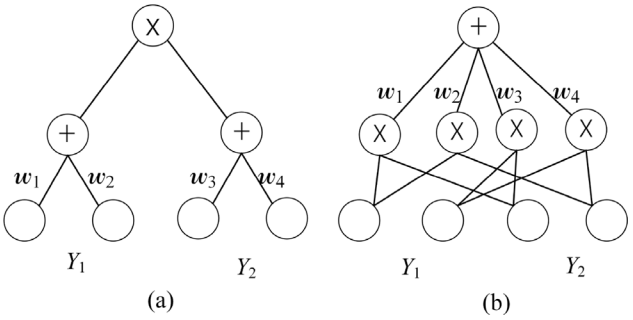


Fig. 2. Expressive efficiency and complexity with different structures of an SPN, where Y represents variable, w represents weight parameter, $+$ represents sum node, $\times$ represents product node, and bottom nodes represent leaf nodes.

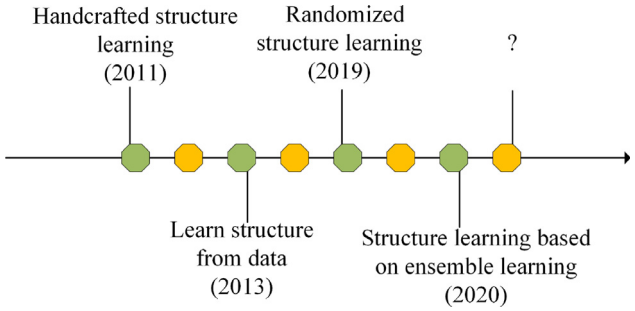


Fig. 3. The development of SPN structure learning methods.

(SPNs are defined in Section 2.2.1, and an example of the SPN structure is shown in Fig. 5). The expressiveness[3] and complexity of SPNs depend on their structures as shown in Fig. 2. Assuming that there are two variables Y_1 and Y_2, and we can construct different structures of SPNs to show their different distributions and relationships between them. In Fig. 2(a), Y_1 and Y_2 follow independent distributions. This SPN has fewer nodes and edges but can only capture independent distributions over Y_1 and Y_2, i.e., its expressiveness is weak but the computational complexity is low. In Fig. 2(b), Y_1 and Y_2 can follow arbitrary distributions, which means the SPN with this structure can represent all distributions over Y_1 and Y_2. However, the structure size[4] of SPN will increase exponentially with the increase of variables in Fig. 2(b). That is to say, the SPN in Fig. 2(b) is strong in terms of expressiveness but the computational complexity is high. Therefore, it is of great significance to design a reasonable structure for an SPN to improve expressiveness and reduce complexity.

SPN structure learning[5] is a vital research topic for SPNs, since the expressiveness and complexity of SPNs depend on their structures. Most SPN structure learning algorithms that perform well in many fields are heuristic, such as LearnSPN (Gens & Pedro, 2013). In recent years, many new methods different from heuristic algorithms have been proposed, such as the well-principled Bayesian method for SPN structure learning (Trapp et al., 2017), the simple and scalable method named RAT-SPNs for leveraging SPNs in deep learning (Peharz, Vergari et al., 2020) and ensemble SPN that explicitly reformulates sum nodes as residual functions to enable the combination of arbitrary groups of SPNs into a larger model (Ventola, Stelzner, Molina, & Kersting, 2020). Different types of SPN structure learning methods are represented in the timeline as shown in Fig. 3. In general, there is relatively little research beyond heuristic methods, but these methods are of great significance to the research and wide application of SPNs.

To the best of our knowledge, there exists only one survey on SPNs (Sánchez-Cauce, París, & Díez, 2021), which introduces the definition of SPNs, the comparison of related models, inference and learning algorithms, applications and software libraries of SPNs, etc. Compared with the existing work, our paper is mainly reflected in three aspects: (1) Our paper focuses on a different scope. Literature (Sánchez-Cauce et al., 2021) has made a comprehensive review of SPNs, but the review of SPN structure learning is not particularly detailed. Due to the influence of SPN structure and the limitation of space, we mainly focus on the SPN structure learning. (2) Our paper reviewed the SPN structure learning algorithms more comprehensively. Literature (Sánchez-Cauce et al., 2021) only summarizes one type of SPN structure learning algorithms, i.e., data-based SPN structure learning algorithms, while our paper systematically investigates four types of SPN structure learning algorithms, including constructing structure manually, learning structure from data, constructing structure randomly, and learning structure by ensemble learning methods. To the best knowledge, our work is the first survey systematically reviewing the SPN structure learning algorithms. (3) Our paper focuses on more recently published papers. Literature (Sánchez-Cauce et al., 2021) summarizes the studies of SPNs in 2019 and before, while

[3] Expressiveness: Expressiveness refers to the ability of a probability distribution function to represent many types of distributions, especially complex types of distributions, in this paper.

[4] Structure size: Here, the structure size represents the number of edges of SPN, and the size of SPN increases with the increase of the number of edges.

[5] SPNs learning can be divided into structure and parameter learning, SPNs are similar to PGMs in that they cover a wide range of mature techniques for parameter learning (Butz, Oliveira, dos Santos, & Teixeira, 2019; Gens & Domingos, 2012; Peharz, Gens, Pernkopf, & Domingos, 2016; Rashwan, Poupart, & Zhitang, 2018; Rashwan, Zhao, & Poupart, 2016; Trapp, Madl, Peharz, Pernkopf, & Trappl, 2017; Trapp, Peharz, Ge, Pernkopf, & Ghahramani, 2019; Vergari et al., 2019; Zhao, Adel, Gordon and Amos, 2016; Zhao, Poupart and Gordon, 2016). However, the situation of SPN structure learning is remarkably different. Our emphasis is on SPN structure learning.

Table 2
Key notations.

Symbols	Definitions and descriptions
G	Graph
E	Set of edges in G
V	Set of nodes or vertices in G
X, Y, Z	Random variables
x, y, z	Values of random variables X, Y *and* Z
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	Sets of random variables
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	Partial evidence for random variables X, Y, Z
$\mathbf{val}(X)$	The set of values of an random variables X
$\mathbf{val}(\mathbf{X})$	$\mathbf{val}(\mathbf{X}) = \times_{n=1}^{N}\mathbf{val}(X_n)$
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Elements from $\mathbf{val}(\mathbf{X})$, $\mathbf{val}(\mathbf{Y})$, $\mathbf{val}(\mathbf{Z})$
$\lambda_{X=x}$	Indicator variable for state x of a finite-state X
$\boldsymbol{\lambda}$	Vector collecting all indicator variables
Φ	Unnormalized distribution
f_Φ	NP of unnormalized distribution Φ
$\mathcal{S}$	Sum–product network
S	Sum node
P	Product node
N	Generic node in an SPN
L	Leaf node in an SPN
w	Parameters of sum-weights in an SPN
θ	Parameters of leaf node in an SPN
$\mathbf{x}[\mathbf{Y}], \mathbf{x}[Y]$	Projection of $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ onto $\boldsymbol{Y} \subseteq \mathbf{X}$ and $Y \in \mathbf{X}$

this paper comprehensively covers the latest development and cutting-edge achievements in the field of SPN structure learning, such as well-principled Bayesian method for SPN structure learning (Trapp et al., 2017), many improved versions of RAT-SPNs (Peharz, Lang et al., 2020) and ensemble SPNs (Ventola et al., 2020). In particular, in order to promote the wide application of SPNs, this paper focuses on the research progress of SPN structure learning based on different data types and introduces relevant datasets in detail. Considering the significance of SPN structure learning and hoping to provide a valuable reference for future researchers, we deeply analyze and discuss the SPN structure learning methods and forecast future research trends.

The rest of this survey is organized as follows. We introduce related theories of SPNs and analyze its principle in depth in Section 2. In Sections 3 and 4, we summarize and analyze the SPN structure learning methods based on different method types and data types they deal with, respectively. Based on the types of structure learning methods, we divide SPN structure learning algorithms into four types: constructing structure manually, learning structure from data, constructing structure randomly, and learning structure by ensemble learning methods in Section 3. According to the data types, the SPN structure learning algorithms can be classified into 4 categories: SPN structure learning built on general data, online data, relational data and sequence data, respectively, as shown in Section 4. In Section 5, we present some datasets, evaluation approaches and online resources. Some open issues, challenges and directions for future work are discussed in Section 6. We conclude this survey in Section 7.

2. Preliminaries

We will introduce the basic mathematical theories related to SPNs in this section, such as graphs theory, network polynomial (NP), and arithmetic circuit (AC). Then, we summarize basic definitions and inferences of SPNs as the theoretical foundations for SPN structure learning. Before carrying out this related work, we first depict key notations in this paper in Table 2.

2.1. Mathematical preliminaries

2.1.1. Basic definitions about graphs

We will briefly review the concepts of graph theory relevant to this survey in this section.

Definition 1 (*Graph*[6])**.** A graph $\mathrm{G} = (\mathbf{V}, \mathbf{E})$ is consisted of node sets $\mathbf{V}$ (any objects) and edge sets $\mathbf{E} = \{(n_i, n_j) | n_i, n_j \in \mathbf{V}\}$.

Different from undirected graphs, edges are oriented for directed graphs.

Definition 2 (*Cycle*)**.** A graph $\mathrm{G} = (\mathbf{V}, \mathbf{E})$ is cyclic if it starts at some vertex $M (M \in \mathbf{V})$ and returns to vertex M by several edges.

Definition 3 (*Directed Acyclic Graphs*)**.** A directed graph $\mathrm{G} = (\mathbf{V}, \mathbf{E})$ is a directed acyclic graph (DAG) if it cannot start from a vertex $M (M \in \mathbf{V})$ and return to vertex M by several edges.

Definition 4 (*Rooted Directed Acyclic Graphs*)**.** A DAG is a rooted directed acyclic graph if there is only one node of DAG having no parents.

Definition 5 (*Induced Graph*)**.** Let $\mathrm{G} = (\mathbf{V}, \mathbf{E})$ be a graph and $\mathbf{V}' \subseteq \mathbf{V}$. The $\mathrm{G}' = (\mathbf{V}', \mathbf{E}')$ is induced graph if it contains $\mathbf{E}' = \{(V_i, V_j) \in \mathbf{E} \mid (V_i, V_j) \in \mathbf{V}'\}$.

2.1.2. Network polynomial

NP is a description of distributions over finite-state random variables (Peharz et al., 2016), which can be regarded as another expression of the probability table. Poon and Domingos (2011) generalized the NP to arbitrary unnormalized distributions.

SPNs based on the notion of NP are inspired by the basic work of efficient inference in BNs.[7] Many important conclusions of NP can be applied to SPNs. For example, the marginalization of NP can be simplified by setting indicator variables (IVs) corresponding to the random variables (RVs) states. In addition, the derivatives of NP provide certain probabilistic explanations. In the following, we review the concept of NP related to SPNs.

Definition 6 (*Network Polynomial of Unnormalized Distributions Peharz, 2015*)**.** Let Φ be an unnormalized probability distribution over discrete RVs $\mathbf{X}$, i.e. $\forall \mathbf{x} \in \mathbf{val}(\mathbf{X}) : \Phi(\mathbf{x}) \geq 0$, $\exists \mathbf{x} \in \mathbf{val}(\mathbf{X}) : \Phi(\mathbf{x}) > 0$. The NP f_Φ of Φ is defined as

$$f_\Phi(\boldsymbol{\lambda}) = \sum_{\mathbf{x} \in \mathbf{val}(\mathbf{X})} \Phi(\mathbf{x}) \prod_{X \in \mathbf{X}} \lambda_{X=\mathbf{x}[X]}. \tag{1}$$

where $\mathbf{X}$ is a set of finite-state RVs, $\mathbf{x}$ is the value of $\mathbf{X}$, x is the values of random variables X, $\mathbf{x}[\mathrm{X}]$ is the projection of $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ onto X, $\lambda_{X=\mathrm{x}} \in \mathbb{R}$ is an IV, vector $\boldsymbol{\lambda}$ containing all IVs. $\lambda_{X=\mathrm{x}}$ is restricted to values $\{0, 1\}$ and function of $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ are defined as:

$$\lambda_{X=x}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x}[X] = x \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The formula $\prod_{X \in \mathbf{X}} \lambda_{X=\mathbf{x}[X]}$ represents the product of all indicators with value 1 in state x. When $\lambda \equiv 1$, $f_\Phi(\lambda)$ returns the normalization constant of Φ.

[6] In this paper, "graph" and "network" express the same meaning, and can be used interchangeably.

[7] The NP has an exponential size in the number of variables, but it can be calculated precisely using arithmetic circuits of polynomial size (tractable arithmetic circuits), which leads to the concept of SPNs.

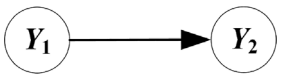


Fig. 4. An example of a BN.

Example 1. The BN in Fig. 4 can be expressed as an NP:

$$f(Y_1, \bar{Y_1}, Y_2, \bar{Y_2}) = p(Y_2 \mid Y_1)p(Y_1)Y_1Y_2 + p(Y_2 \mid \bar{Y_1})p(\bar{Y_1})\bar{Y_1}Y_2 + p(\bar{Y_2} \mid Y_1)p(Y_1)Y_1\bar{Y_2} + p(\bar{Y_2} \mid \bar{Y_1})p(\bar{Y_1})\bar{Y_1}\bar{Y_2}. \tag{3}$$

where Y is RVs, and Y is restricted to values $\{0, 1\}$.

The differential approach to the inference of NP has certain probabilistic explanations. The derivatives of the SPN function with respect to the indicator variables are considered. Eq. (1) is an exponential sum over terms

$$t_{\mathbf{x}}(\boldsymbol{\lambda}) = \Phi(\mathbf{x}) \prod_{X \in \mathbf{X}} \lambda_{X=\mathbf{x}[X]}. \tag{4}$$

The derivative of $t_{\mathbf{x}}(\boldsymbol{\lambda})$ is

$$\frac{\partial t_{\mathbf{x}}(\boldsymbol{\lambda})}{\partial \lambda_{X=x}} = \begin{cases} \Phi(\mathbf{x}) \prod_{X' \in \mathbf{X} \setminus \{X\}} \lambda_{X'=\mathbf{x}[X']} & \text{if } \mathbf{x}[X] = x \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The derivative of the NP is the sum over the derivatives of the form Eq. (5), which equals the evaluation of f_Φ at a modified IV vector $\boldsymbol{\lambda}'$:

$$\frac{\partial f_\Phi}{\partial \lambda_{X=x}}(\boldsymbol{\lambda}) = f_\Phi(\boldsymbol{\lambda}'), \text{ where} \tag{6}$$

$$\lambda'_{Y=y} = \begin{cases} \lambda_{Y=y} & \text{if } Y \neq X \\ 1 & Y = X,\ y = x \\ 0 & Y = X,\ y \neq x. \end{cases} \tag{7}$$

The derivative of the NP is

$$\frac{\partial f_\Phi}{\partial \lambda_{X=x}}(\boldsymbol{\lambda}(\boldsymbol{\chi})) = \sum_{x' \in \boldsymbol{\chi}[\mathbf{X} \setminus \{X\}]} \Phi(x, \mathbf{x}'). \tag{8}$$

where $\boldsymbol{\chi}$ represents the partial evidence for random variables set $\mathbf{X}$.

In particular, for any $\mathbf{x} \in \mathbf{val}(\mathbf{X})$

$$\frac{\partial f_\Phi}{\partial \lambda_{X=x}}(\boldsymbol{\lambda}(\mathbf{x})) = \Phi(x, \mathbf{x}[\mathbf{X} \setminus \{X\}]). \tag{9}$$

According to Eqs. (4) to (9), the form of conditional probabilities is

$$\Phi(x, \boldsymbol{\chi}[\mathbf{X} \setminus \{X\}]) \propto \frac{\partial f_\Phi}{\partial \lambda_{X=x}}(\boldsymbol{\lambda}(\boldsymbol{\chi})). \tag{10}$$

The conditional distribution in Eq. (10) is useful for approximating MAP solvers (Park, 2002; Park & Darwiche, 2004). Higher order derivatives have similar semantics with respect to IVs, taking second derivatives as an example:

$$\frac{\partial^2 f_\Phi}{\partial \lambda_{X=x} \partial \lambda_{Y=y}}(\boldsymbol{\lambda}(\boldsymbol{\chi})) = \begin{cases} \sum_{\mathbf{x}' \in \boldsymbol{\chi}[\mathbf{X} \setminus \{X,Y\}]} \Phi(x, y, \mathbf{x}') & \text{if } X \neq Y \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

2.1.3. Arithmetic circuits

The inference representation of SPNs is closely related to Arithmetic Circuits (ACs). Darwiche (2003) pointed out that the NP of a BN over a discrete random vector can be effectively represented by AC.

Definition 7 (*Arithmetic Circuit*)**.** For a rooted directed acyclic graph G, where the numeric inputs are leaf nodes, the arithmetic operations $(+, -, \times, \div)$ are internal nodes, and the root value is the output of AC. The number of edges in an AC indicates the size of AC, and the size of AC increases as the number of edges increases.

Different from ACs, SPNs are limited to weighted sum and product operations. In addition, SPNs define probabilistic interpretation in a more direct manner than ACs. The representations of ACs and SPNs are equivalent in discrete domains (Rooshenas & Lowd, 2014), i.e., using SPNs with a smaller or equal number of edges and nodes can represent a decomposable and smooth AC equivalently.

2.2. Sum–product networks

In this section, we summarize the definitions and inference methods of SPNs from different perspectives to help understand the existing SPN structure learning methods.

2.2.1. Basic definitions of SPNs

SPNs can be defined in a number of ways. Poon and Domingos (2011) defined SPNs as finite-state SPNs (Finite-state SPNs in Definition 8), which is used to represent the probability distribution over finite-state RVs. Most of the literature after (Gens & Pedro, 2013) defined SPNs as a combination of probability distributions (Generalized SPNs in Definition 13), which generalized SPN to continuous RVs and discrete RVs with infinitely many states. In addition, Trapp et al. (2017) defined SPNs as a combination of computational graph and scope function (SPNs consisting of computational graph and scope-function (CG & SC for short) in Definition 14). In the following, we will introduce these three types of SPNs.

Definition 8 (*Finite-state SPNs*)**.** An SPN $\mathcal{S} = (\mathrm{G}, \boldsymbol{w})$ over finite-state RVs $\mathbf{X}$ is a rooted DAG $\mathrm{G} = (\mathbf{V}, \mathbf{E})$ with sum-weights parameters $\boldsymbol{w}$. The nodes $V \in \mathbf{V}$ are functions mapping $\boldsymbol{\lambda}$ to $\mathbb{R}$. Different types of nodes N in SPN are defined as follows:

(1) Any leaf node value is indicators: $\lambda_{X=\mathrm{x}}$.
(2) The value of any sum node is calculated by the sum of its own children with sum-weights $\boldsymbol{w}$:

$$S(\boldsymbol{\lambda}) = \sum_{C \in ch(S)} w_{S,C} C(\boldsymbol{\lambda}), \tag{12}$$

where $w_{S,C} \in \boldsymbol{w}$ represents the weight parameter of the edge from node S to node C, $ch(S)$ represents the child nodes of node S.
(3) The value of any product node is calculated by the product of its own children:

$$p(\boldsymbol{\lambda}) = \prod_{C \in ch(P)} C(\boldsymbol{\lambda}). \tag{13}$$

Definition 9 (*Scope of SPNs*)**.** Let $\mathcal{S}$ be an SPN over RVs $\mathbf{X}$. The scope of node M, denoted as $\mathrm{sc}(M)$, is defined as:

$$sc(M) = \begin{cases} \{X\} & \text{if } M \text{ is leaf node} \\ \cup_{C \in ch(M)} sc(C) & \text{otherwise.} \end{cases} \tag{14}$$

Taking an SPN in Fig. 5 as an example, the scope of n_{19}, n_{14}, n_8, n_2, n_1 are as follows:
$\mathrm{sc}(n_{19}) = \{X_6\}$, $\mathrm{sc}(n_{14}) = \{X_5, X_6\}$, $\mathrm{sc}(n_8) = \{X_3, X_4, X_5, X_6\}$, $\mathrm{sc}(n_2) = \{X_1, X_2, X_3, X_4, X_5, X_6\}$, the scope of $\mathcal{S}$ is the scope of its root node, i.e., $\mathrm{sc}(\mathcal{S}) = \mathrm{sc}(n_1) = \{X_1, X_2, X_3, X_4, X_5, X_6\}$.

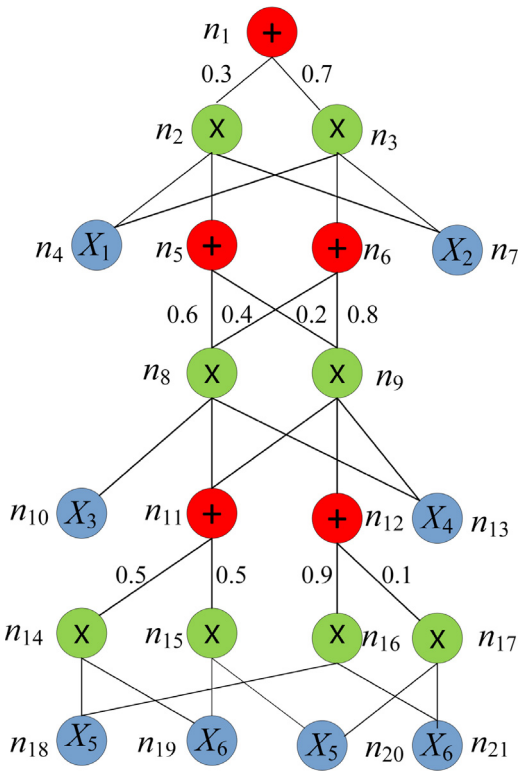


Fig. 5. An example of the SPN.

SPNs cannot always compute the probability of evidence correctly. To ensure the validity of SPNs, two conditions, namely consistency and completeness, have been proposed in Poon and Domingos (2011).

Definition 10 (*Consistency*)**.** An SPN $\mathcal{S} = (\mathrm{G}, \boldsymbol{w})$ is consistent if different children of any product node do not appear non-negated and negated of the same variable.

Definition 11 (*Completeness*)**.** An SPN $\mathcal{S} = (\mathrm{G}, \boldsymbol{w})$ is complete iff $sc(C_1) = sc(C_2)$, $\forall$*sum node* $S \in \mathcal{S}, \forall C_1, C_2 \in ch(S)$.

An alternative condition of consistency is decomposability, which is a stronger and more intuitive property.

Definition 12 (*Decomposability*)**.** An SPN $\mathcal{S} = (\mathrm{G}, \boldsymbol{w})$ is decomposable iff $sc(C_1) \cap sc(C_2) = \varnothing, \forall$*product node* $P \in \mathcal{S}, \forall C_1, C_2 \in ch(P), C_1 \neq C_2$.

It is worth noting that Definitions 9 to 12 can also be applied to Generalized SPNs and CG & SC SPNs.

The derivative of SPNs function with respect to the IVs (Peharz et al., 2016) is

$$\frac{\partial \mathcal{S}(\mathbf{e})}{\partial \lambda_{X=x}} = \mathcal{S}(X = x, \mathbf{e} \backslash X), \tag{15}$$

where $\mathbf{e}$ is the evidence of SPN $\mathcal{S}$, and $\mathbf{e}$ is changed such that X is set to x. It is important that all X and x can be evaluated using back-propagation pass after evaluating the evidence $\mathbf{e}$. Similarly, the second derivative with respect to the IVs is

$$\frac{\partial^2 \mathcal{S}_{\mathbf{e}}}{\partial \lambda_{X=x} \partial \lambda_{Y=y}} = \begin{cases} \mathcal{S}(X = x, Y = y, \mathbf{e} \backslash \{X, Y\}) & \text{if } X \neq Y \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

In fact, many SPN structure learning methods default the hypothesis that the input distributions are arbitrary. We will introduce a simple and understandable definition: Generalized SPNs (Peharz, 2015).

Definition 13 (*Generalized SPNs*)**.** An SPN $\mathcal{S} = (\mathrm{G}, \boldsymbol{w})$ over RVs (continuous RVs and discrete RVs with infinitely many states) $\mathbf{X}$ is a rooted DAG with arbitrary distributions leaves L and whose internal nodes are sum nodes S and product nodes P. The different types of nodes N in SPN are defined as follows:

(1) The value of a leaf node is calculated by evaluating its probability distribution respectively on a subset of $\mathbf{X}_L \subset \mathbf{X}$, denoted as $\mathrm{L}(x) = p(x_L|\theta_L)$, where $\boldsymbol{X}_L$ is a subset of $\mathbf{X}$ and θ_L is the parameter of the distribution at leaf L.
(2) The value of a product node is calculated by the product of its all children, i.e., $p(x) = \prod_{N \in ch(P)} N(x)$.
(3) The value of a sum node is calculated by the weighted sum of its all children, i.e., $S(x) = \sum_{N \in ch(S)} w_{S,N} N(x)$, where the non-negative weight $w_{S,N} \in \boldsymbol{w}$ is associated with the edge $(S, N) \in \mathbf{E}$ and S is normalized if $\sum_{N \in ch(S)} w_{S,N} = 1$.
(4) The value of the root node is calculated by evaluating its probability distribution $\mathcal{S}$, denoted as $p_{\mathcal{S}} = \frac{\mathcal{S}(x)}{Z}$, where Z is the normalization constant of SPNs.

Analogous to Eqs. (15) and (16), the differential method can be extended to the SPN with different types of input distributions (Peharz, Tschiatschek, Pernkopf, & Domingos, 2015).

To help understand the SPN structure learning algorithm proposed in Trapp et al. (2017), we will introduce a more general definition in Definition 14, i.e., an SPN can be defined in terms of a computational graph and a scope function. Note that the computational graph G is a simple AC whose internal nodes are sum and product. Different from other definitions of SPNs, the computational graph does not need to satisfy the structural conditions, such as completeness and decomposability, and these structural conditions are expressed by scope function in Definition 14.

Definition 14 (*CG & SC SPNs*)**.** An SPN is defined as a 4-tuple $\mathcal{S} = (\mathrm{G}, \psi, \boldsymbol{w}, \theta)$, where the parameters G, ψ, $\boldsymbol{w}$ and θ represent computational graph, scope-function, weights of sum nodes and parameters of leaf nodes, respectively. The details of Definition 14 are as follows:

Computational graph. The computational graph G is a rooted DAG, containing sum, product and leaf nodes: S, P and L.

Scope function. The scope function $\psi : N \mapsto 2^N$, assigning each node in G to a sub-set of $\mathbf{X}$ (2^X represents the power set of $\mathbf{X}$). The specific characteristics are as follows:

(1) The scope function $\psi(N) = \mathbf{X}$ iff N is the root node.
(2) The scope function $\psi(N) = \cup_{N' \in ch(N)} \psi(N')$, if N is an internal node or a root node.
(3) (completeness). $\forall S \in \mathbf{S}, N, N' \in ch(S) : \psi(N) = \psi(N')$.
(4) (decomposability). $\forall P \in \mathbf{P}, N, N' \in ch(P) : \psi(N) \cap \psi(N') = \emptyset$.

The application of the scope function ψ can be interpreted as masking nodes in the computational graph and selecting a sub-graph that satisfies the structural conditions of SPNs. Therefore, various different efficient structures of SPNs can be expressed by applying different scope functions on the same computational graph.

2.2.2. Inference of SPNs

Similar to the PGMs, the inference of SPNs denotes calculating the probability of complete evidence or marginal probability, or inferring some unobserved variables based on the evidence of observed variables. In this section, we introduce three inference methods and provide three simple examples to help clarify the process of each method.

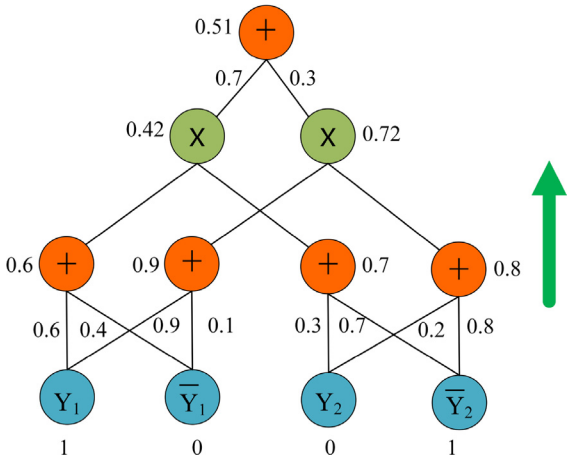


Fig. 6. Complete evidence inference of an SPN.

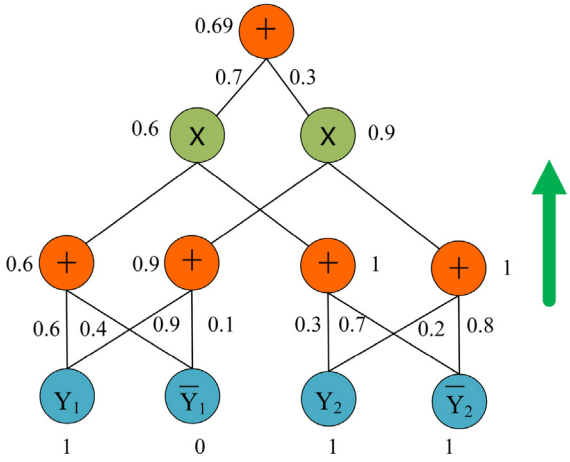


Fig. 7. Marginal inference of an SPN.

Complete Evidence Inference Complete evidence inference for an SPN is to calculate evidence probability $P(Y = y)$, i.e, evaluate $\mathcal{S}$ in a bottom-up (feedforward) fashion. A simple example of complete evidence inference is shown as Example 2.

Example 2. Fig. 6 shows the complete evidence inference for an SPN. $\mathcal{S}(y)$ is obtained by bottom-up pass way from the leaves to the root node with indicators $y_1 = 1, \bar{y_1} = 0, y_2 = 0, \bar{y_2} = 1$. Z is calculated with indicators $y_1 = 1, \bar{y_1} = 1, y_2 = 1, \bar{y_2} = 1$ in a bottom-up pass. The complete evidence probability of $\mathbf{Y} = (y_1 = 1, y_2 = 0)$, denoted as $P(y) = \mathcal{S}(y)/Z$, can be calculated using $P(y) = \mathcal{S}(y)/Z = \mathcal{S}(y_1 = 1, \bar{y_1} = 0, y_2 = 0, \bar{y_2} = 1)/Z(y_1 = 1, \bar{y_1} = 1, y_2 = 1, \bar{y_2} = 1) = (0.7(0.6\times1+0.4\times0)(0.3\times0+0.7\times1)+0.3(0.9\times1+0.1\times0)(0.2\times0+0.8\times1))/((0.7(0.6\times1+0.4\times1)(0.3\times1+0.7\times1)+0.3(0.9\times1+0.1\times1)(0.2\times1+0.8\times1))) = 0.51$, as shown in Fig. 6.

Marginal Inference Marginal inference for an SPN is to calculate marginal probability query like $p(\mathbf{X} = \mathbf{x})$, $\mathbf{X} \subset \mathbf{Y}$, i.e., evaluate $\mathcal{S}$ in a bottom-up (feedforward) fashion. Example 3 presents a simple case of marginal evidence inference.

Example 3. Fig. 7 shows the marginal evidence inference for an SPN. The marginal evidence probability of $\mathbf{X} = \{\mathbf{y} : y_1 = 1\}$, denoted as $P(\mathbf{X}) = \mathcal{S}(\mathbf{X})/Z$, is obtained by bottom-up pass way from the leaves to the root node with $y_1 = 1, \bar{y_1} = 0, y_2 = 1, \bar{y_2} = 1$. The marginal probability value can be calculated using $P(y) = \mathcal{S}(y)/Z = \mathcal{S}(y_1 = 1, \bar{y_1} = 0, y_2 = 1, \bar{y_2} = 1)/Z(y_1 = 1, \bar{y_1} = 1, y_2 = 1, \bar{y_2} = 1) = (0.7(0.6\times1+0.4\times0)(0.3\times1+0.7\times1)+0.3(0.9\times1+0.1\times0)(0.2\times1+0.8\times1))/((0.7(0.6\times1+0.4\times1)(0.3\times1+0.7\times1)+0.3(0.9\times1+0.1\times1)(0.2\times1+0.8\times1))) = 0.69$, as shown in Fig. 7.

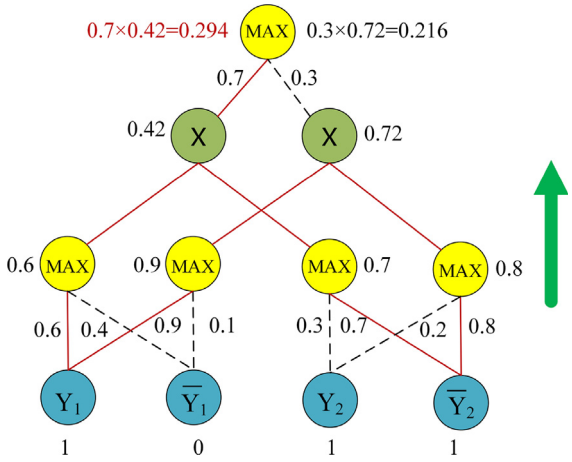


Fig. 8. MPE inference of an SPN in bottom-up pass.

The Most Probable Explanation Inference The most probable explanation (MPE) inference for an SPN is to calculate

$$\mathbf{q}^* = \arg\max_{x\sim Z} p(\mathbf{X}, \mathbf{x}). \tag{17}$$

where $\mathbf{X}, \mathbf{Z} \subset \mathbf{Y}$, $\mathbf{X} \cap \mathbf{Z} = \emptyset$, $\mathbf{X} \cup \mathbf{Z} = \mathbf{Y}$, and the value of observed variables can be inferred using MPE inference. The approximation of MPE (Peharz et al., 2016) can be answered in linear time. The value of MPE is obtained in a bottom-up and top-down fashion.

Example 4. Figs. 8 and 9 show the MPE inference for an SPN. The problem of "what is the most likely value of Y_2 if $Y_1 = 1$?" can be solved by MPE. The specific calculation process is:

(1) Replace sum nodes by nodes named MAX firstly, where the MAX represents the maximum value of the product of its children and corresponding edge weights.
(2) Evaluate $\mathcal{S}$ with indicators $y_1 = 1, \bar{y_1} = 0, y_2 = 1, \bar{y_2} = 1$ in a bottom-up pass in Fig. 8.
(3) In top-down passing, select the children of the MAX nodes with the highest value and select all the children of product nodes until the values of the reached leaf nodes are the values to be found.

We get the most likely value for Y_2 is 0 in Fig. 9.

3. SPN structure learning based on different method types

In this section, we survey a family of "SPN structural learning" algorithms according to the types of structure learning methods as shown in Fig. 3. These methods establish the fundamental approaches and principles for the structural learning methodologies towards SPNs.

The SPN structure learning methods aim to find the optimal (or near-optimal) structures of SPNs, which can express complex probability distributions in reasonable structures, as mentioned in Section 1 and Fig. 2. Based on the research progress of SPN structure learning, we summarize the existing SPN structure learning algorithms into four types. For ease of description and differentiation, we name the four types of SPN structure learning methods as Handcrafted structure learning, Data-based structure learning, Random structure learning and Ensemble structure Learning, respectively.

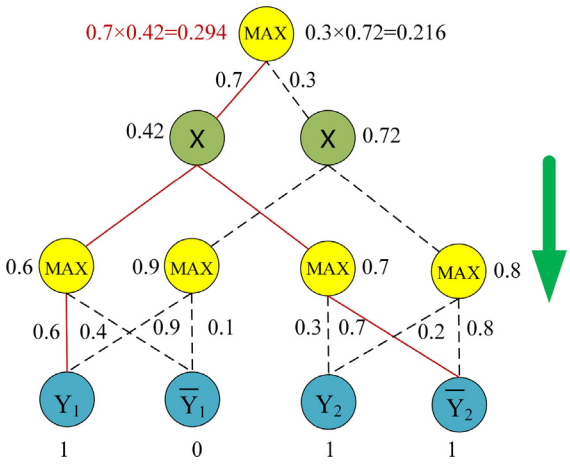


Fig. 9. MPE inference of an SPN in top-down pass.

3.1. Handcrafted structure learning

The handcrafted structure learning methods (Gens & Domingos, 2012; Poon & Domingos, 2011) are fixed and manually designed for image data, which is not easy to extend to other fields. Poon and Domingos (2011) proposed a handcrafted structure learning method named PD architecture, which manually sets the SPNs structure with a fixed number of layers. As shown in Fig. 10, the PD architecture starts from sum nodes corresponding to whole regions and divides regions recursively using axis-aligned cuts. The 'cut' corresponds to product nodes, and the segmented regions correspond to sum nodes or leaf nodes. While the PD architecture has been successful in building an initial "fixed" structure suitable for image data, it is challenging to specify the SPN manually. Moreover, the structure must satisfy the validity conditions of the SPN to produce excellent properties.

3.2. Data-based structure learning

In this section, we survey a family of SPN structure learning algorithms that learn the structure from data. The data-based structure learning methods are flexible and can be applied to fields other than images. We classify the data-based structure learning methods into heuristic methods and non-heuristic methods.

3.2.1. Heuristic algorithms

There is much relevant literature on heuristic methods (greedy methods) up to now, which are mainly divided into top-down (iteratively cluster data matrix) and bottom-up (start by the marginal RVs) algorithms.

(1) Top-down algorithms

BuildSPN (Dennis & Ventura, 2012) is the earliest algorithm for data-based structure learning methods. The idea of BuildSPN is to find subsets of variables that are highly correlated and introduce latent variables to explain dependencies. The performance of BuildSPN improved significantly compared to the PD architecture as mentioned in Section 3.1. However, BuildSPN is subject to several limitations, including (a) being tailored for image data, (b) potential separation of highly correlated variables during the clustering process, and (c) exponential increases in both time and space complexity with the number of variables. To mitigate these issues, several top-down algorithms have been proposed, such as LearnSPN (Gens & Pedro, 2013) and its variants.

Gens and Domingos proposed a greedy and top-down algorithm named LearnSPN (Gens & Pedro, 2013) that inspired many algorithms and variations. The structure of an SPN is produced by recursively clustering instances (produce sum nodes) and partitioning variables (produce product nodes), as shown in Fig. 11. LearnSPN can be considered as a basic framework of SPN structure learning because it does not specify methods for partitioning variables or clustering similar instances.

LearnSPN (Gens & Pedro, 2013) contains 2 steps for structure learning: segmentation by G-test and clustering using Hard EM, respectively. The main processes are as follows:

(1) Splitting columns into pairs by a greedy G Test :

$$G(X_i, X_j) = 2 \sum_{x_i \sim X_i} \sum_{x_j \sim X_j} c(x_i, x_j) \cdot \log \frac{c(x_i, x_j) \cdot \mid T \mid}{c(x_i)(x_j)}. \tag{18}$$

(2) Clustering instances into $|C|$ sets with online Hard-EM:

$$P(\mathbf{X}) = \sum_{C_i \in \mathbf{C}} P(C_i) \prod_{X_j \in \mathbf{X}} P(X_j \mid C_i). \tag{19}$$

We show an example of LearnSPN using a visual representation of SPN structure learning (Vergari et al., 2020), as described below.

(1) The clustering of instances generates sum nodes, as shown in Fig. 12(a).
(2) Find the independent groups of RVs in each cluster (produce the product node), as shown in Fig. 12(b).
(3) If failure to find independent groups of RVs, clustering instances to produce sum nodes, as shown in Fig. 12(c).
(4) If there exist single variables, define the single variables as leaf nodes, as shown in Fig. 12(d).
(5) Continue until there are no more leaves to expand.
(6) Parameters of weights are initialized by clustering ratios, as shown in Fig. 12(d).

LearnSPN was critiqued for the following reasons. (1) The complex network structures are realized by increasing the number of children of sum nodes explosively, resulting in many redundant nodes. (2) The structures generated by LearnSPN are mostly shallow structures, while deeper structures have stronger expressive power.

Many variants of LearnSPN have been proposed, and one influential modification is the indirect–direct SPN named ID-SPN (Rooshenas & Lowd, 2014). ID-SPN combines both direct (direct dependencies between variables) and indirect (dependencies between latent variables) methods. ID-SPN performs a top-down search similar to LearnSPN, i.e., clustering instances and splitting variables to create sum and product nodes. However, ID-SPN prematurely ends the process of top-down search before obtaining univariate distributions and ID-SPN learns an AC through the tractable Markov networks (Lowd & Rooshenas, 2013) to represent a tractable multivariate distribution. Fig. 13 shows an example of a structure learned by the ID-SPN, where the structure before obtaining univariate distributions is represented as the sum and product nodes, while the underlying structure is abbreviated with the nodes labeled "AC". The AC components encode graphical models over the observed variables represented as AC, and may involve many nodes and edges. It is noteworthy that while ID-SPN remains competitive when compared to recent approaches, it is accompanied by a substantial increase in both running time and model complexity.

There are also many other variants of LearnSPN as listed in Table 3, and each method is briefly described below. We divide the strategies of the variants of LearnSPN into three types respectively in Table 3, including clustering instances (Butz et al., 2018; Krakovna & Looks, 2016; Liu & Luo, 2019; Vergari, Mauro, & Esposito, 2015), segmentation variables (Mauro, Esposito, Ventola,

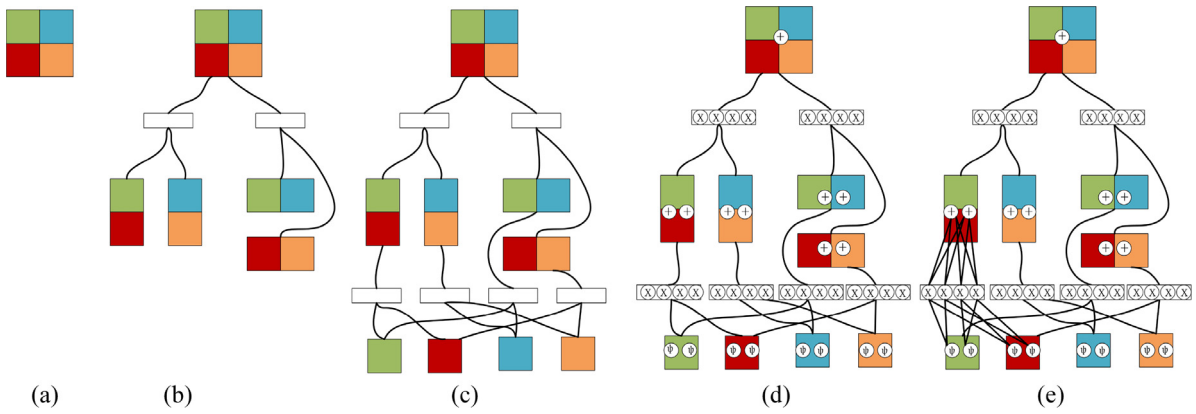


Fig. 10. Visual representation of SPN structure learning. (a) represents the whole region. (b) and (c) represent dividing regions with axis-aligned cuts recursively. For (d), 'cut' corresponds to the product node and the segmented regions correspond to sum nodes or leaf nodes. (e) shows the connection mode that satisfies validity of SPNs.

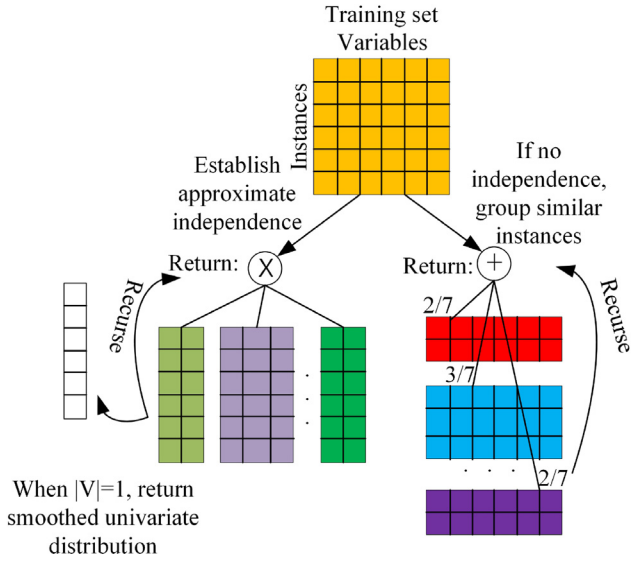


Fig. 11. Structure learning of LearnSPN.
Source: Image extracted from Gens and Pedro (2013).

& Vergari, 2017) and other strategies (Bueff, Speichert, & Belle, 2018; Dennis & Ventura, 2015; Molina et al., 2018; Rahman & Gogate, 2016).

For the first two strategies, in order to learn deeper and more robust networks, Vergari et al. (2015) proposed a variant of LearnSPN with three modifications: (1) Cut the data matrix into two pieces for each split (The modification not only slows down the process of instance clustering and variable segmentation but also helps to learn deeper structures). (2) Learn tree distributions as leaves (The strong naive factorization independence assumptions lead to regularize hardly). (3) Learn bagged sum nodes. In this case, LearnSPN achieves the same performance as ID-SPN.

While LearnSPN and ID-SPN are effective in handling discrete variables and complete data (meaning that the features of the instance are not missing), they do not take into account other variable types or the presence of missing data. Krakovna and Looks (2016) introduced a lightweight version of LearnSPN called MiniSPN, which not only speeds up the running time of model, but also can deal with heterogeneous features and missing data, increasing the practicality of the algorithm. MiniSPN chooses the simplified naive Bayesian hybrid model as the clustering method. Meanwhile, in the variable partitioning step, MiniSPN uses the subset of rows where neither variables are missing for independent testing. Later, Butz et al. (2018) compared different combinations of different variants of LearnSPN and concluded that the best results were obtained by using G-test for the partition of variables and K-means or Gaussian mixture methods for the clustering of instances. Liu and Luo (2019) proposed two algorithms to balance various performances of SPNs in view of the fact that most existing algorithms improve one aspect of performance at the expense of other aspects. The main idea of these algorithms is to replace the method of instance clustering with clustering methods that can adaptively determine the appropriate number of instance clusters. In order to accelerate LearnSPN, Mauro et al. (2017) proposed entropy method and random subspace method for the partition of variables.

For other strategies, Bueff et al. (2018) proposed an algorithm named LearnWMISPN, which combines the weighted model with LearnSPN and determines the order of each polynomial through Bayesian information criteria (Schwarz, 1978). To overcome the difficulty of mixed PGM, Molina et al. (2018) proposed an algorithm inspired by the successes of deep models named MSPNs. MSPNs is based on the Hirschfeld–Gebelein–Rényi Maximum Correlation Coefficient in decomposition (splitting) and conditioning (clustering) operations. The experimental results showed that the MSPNs could learn more complex distributions in hybrid domains. However, MSPNs is highly prone to overfitting, sensitive to the setting of hyperparameters and lack robustness.

In contrast to previous algorithms of tree-structured SPNs, some algorithms are not limited to learning tree-structured SPNs. For example, Dennis and Ventura (2015) introduced an algorithm named SearchSPN, which learns the structure through greedy search methods. SearchSPN incorporates the algorithm of modified LearnSPN to search for candidate structures greedily. In the same vein, Rahman and Gogate (2016) proposed an algorithm to learn graphs from tree SPNs by merging similar sub-SPNs in tree SPNs. To select the candidate sub-SPNs for merging, a mean-field style approximation (defined as Eq. (20)) of the distance between two distributions is used.

$$\mathrm{D}(P_1 \parallel P_2) \approx \frac{1}{|\mathbf{V}|} \sum_{V_i \in \mathbf{V}} D(P_1(V_i) \parallel P_2(V_i)), \tag{20}$$

where D represents distance functions such as mean square error, KL divergence, Euclidean distance, etc. P_1 and P_2 represent distributions over V. This modification of LearnSPN reduces the parameters of the model, but at the expense of learning time for some datasets. It is worth mentioning that SearchSPN outperformed other algorithms for high-dimensional datasets, such as ID-SPN.

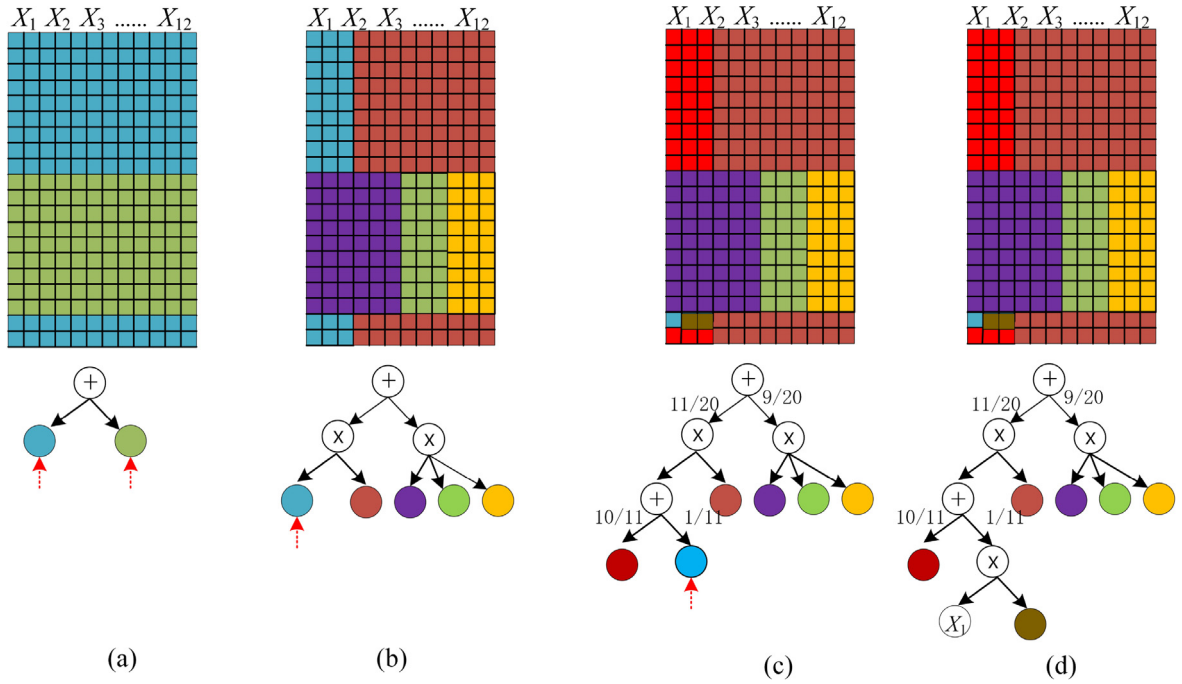


Fig. 12. Visual representation of SPN structure learning.

Table 3
The variants of LearnSPN.

Reference	Main strategies			Characteristics
	Clustering instances (chopping)	Segmentation variables (slicing)	Other strategies	
Vergari et al. (2015)	Bagging	Binary splits + Chow-Liu trees in the leaves	–	Comparable to ID-SPN.
Dennis and Ventura (2015)	–	–	Not limited to tree SPNs: taking a search approach to structure learning.	Likelihood is comparable to ID-SPN. Faster (execution time). Smaller (nodes number).
Molina et al. (2018)	Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient	Hirschfeld-Gebelein-Rényi Maximum Correlation Coefficient	Adopt piecewise polynomial approximations of the univariate leaf densities.	Can capture complex distributions across a broad range of hybrid domains.
Krakovna and Looks (2016)	Simplification of Naive Bayes hybrid model	Perform an independence test with the subset of rows where both variables are existed.	–	Faster. Applicable to heterogeneous features and missing data.
Butz et al. (2018)	K-means Mutual information	G-Test Gaussian mixture	–	Studied the different combinations of algorithms.
Rahman and Gogate (2016)	–	–	Post-processing methods: inducing graph from tree SPNs by merging similar sub-structures.	Smaller computational complexity and better generalization accuracy.
Mauro et al. (2017)	–	Approximate splitting methods.	–	Reduce learning inference times.
Liu and Luo (2019)	Adaptively determine the quantity of instance clusters.	–	–	Balance between different performance of the network.
Bueff et al. (2018)	–	–	Combine LearnSPN with weighted model integration.	Balance between granularity and predictive power.

The variants of LearnSPN improve the performance of LearnSPN to some extent, but they still inherit the shortcomings of LearnSPN. In general, although LearnSPN has many disadvantages and the performance of LearnSPN and its variants cannot compete with the latest algorithms on some tasks, LearnSPN is still widely used because of its modularity and simplicity.

There are other top-down algorithms besides LearnSPN and its variants. Adel, Balduzzi, and Ghodsi (2015) developed an algorithm named SVD-SPN, which constructs the structure of SPNs by finding rank-1 matrices. Experimental results showed that SVD-SPN achieved similar results to LearnSPN and ID-SPN for discrete binary datasets, but SVD-SPN outperformed LearnSPN and ID-SPN for multi-category datasets. In addition, a graph partitioning-based method named Prometheus was proposed in Jaini, Ghose, and Poupart (2018). Prometheus captures the interaction of variables by utilizing correlation matrices directly and clustering

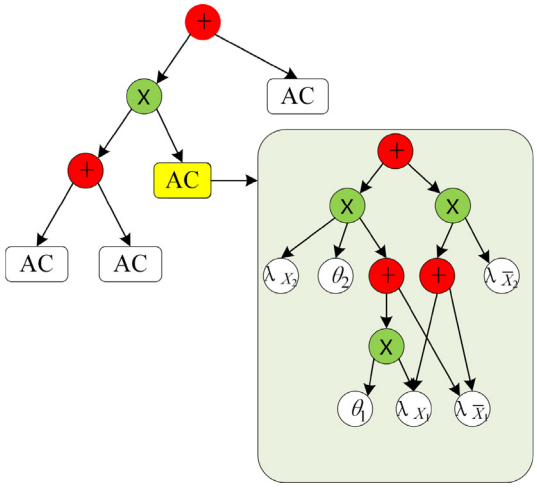


Fig. 13. Example of an ID-SPN model. .
Source: Image extracted from Rooshenas and Lowd (2014).

indirectly. According to the idea of graph partitioning methods, Prometheus not only learns the DAG structures with subtree sharing directly, but also removes the sensitive correlation threshold hyper-parameters. Since Prometheus can also be used for online data learning, we will introduce it in detail in Section 4.2.

(2) Bottom-up algorithms

Peharz, Geiger, and Pernkopf (2013) introduced a method to learn the parameters and structures in a bottom-up way. The main idea of this method is greedily merging the models of small variable ranges into larger variable ranges by means of statistical dependency test guidance. The maximum mutual information principle is used in parameter learning. Later, Hsu, Kalra, and Poupart (2017) proposed an algorithm called oSLRAU. The main idea of oSLRAU is to progressively build an SPN structure in a bottom-up manner by discovering dependencies and modifying nodes of product to represent these dependencies. The relevant researches on bottom-up algorithms are relatively little, but it is an interesting direction.

3.2.2. Non-heuristic algorithms

Both bottom-up and top-down algorithms are heuristic structure learning methods, which are prone to overfitting compared with pure Bayesian models. In this section, we review the work of some important non-heuristic SPN structure learning methods, which mitigate the overfitting problem by using a principled optimization objective function compared with heuristic algorithms.

Peharz, Gens and Domingos (2014) proposed a structure learning method named selSPT, which uses a global scoring function to trade off training likelihood and model complexity in terms of inference cost. selSPT used a similar scoring function as in (Lowd & Domingos, 2008):

$$\begin{aligned} \mathrm{S}(\mathrm{D},\mathrm{G}) &= \mathrm{LL}(\mathrm{D},\mathrm{G}) - \lambda \mathrm{O}(\mathrm{G}), \\ \mathrm{O}(\mathrm{G}) &= c_S \sum_{S \in \mathbf{S}_G} |ch(S)| + c_P \sum_{P \in \mathbf{P}_G} |ch(P)|, \end{aligned} \tag{21}$$

where $\lambda, c_S, c_P \geq 0$ are trade-off parameters, S_G and P_G are sum and product nodes in G, LL(D, G) is the log-likelihood of structure G on dataset D, O(G) is the inference cost in the worst case. It is the first time the structure of SPNs was optimized in a "principled" manner and the results showed that selSPT is comparable to LearnSPN.

Notable attempts for principled SPN structure learning include models using Bayesian non-parametric priors (Lee, Watkins, & Zhang, 2014; Trapp et al., 2016). Two Bayesian non-parametric models based on Dirichlet Processes for SPNs were proposed in Lee et al. (2014), a tree structure and a dag of hierarchical Dirichlet Processes respectively. However, the posterior structure of the induced trees is neglected. Later, Trapp et al. (2016) proposed a principled approach named Infinite Sum–Product Trees (iSPT) by introducing infinite Sum–Product Trees (SPTs). Compared with the infinite Gaussian mixture model, iSPT can achieve better results in density estimation. It is worth noting that the previous work treated scopes as an inherent property of nodes in the SPN structure. Trapp et al. (2019) proposed a well-principled Bayesian algorithm for SPN structure learning called Bayesian SPNs. Bayesian SPNs decouple the SPN structure by searching over graph structure and learning the scope function. The graph structure of Bayesian SPNs can be obtained by learning from data or constructing a random structure. Therefore, we believe that Bayesian SPNs are also a random structure learning algorithm. We will introduce Bayesian SPNs in detail in Section 3.3.1.

3.3. Random structure learning

Learning SPNs from data is complicated by the fact that the structures of SPNs require specific structural constraints to ensure accurate inference. To address this challenge, researchers have proposed a method of using random structures for SPN structure learning. We define random structure learning as the initial structures of SPNs constructed randomly by some methods, rather than manually constructing or learning from data. To the best of our knowledge, there are only three random structure learning methods (Peharz, Vergari et al., 2020; Rashwan et al., 2016; Trapp et al., 2019). In this section, we will review all these three random structure learning methods. In order to highlight the importance of random structure learning methods, we also introduce some methods related to random structure learning in Section 3.3.2.

3.3.1. The random structure learning method

Rashwan et al. (2016) proposed the first algorithm using a random structure named OBMM, which generates corresponding SPNs by a simple structure generator. However, OBMM is neither compiled into the computation graphs nor does it use deep learning techniques.

Most SPN structure learning algorithms have the following problems: (1) it is difficult to tune, (2) the learned SPN structure is limited and difficult to scale to other datasets, and (3) it is not easy to integrate with deep learning frameworks. To solve the above problems, Peharz, Vergari et al. (2020) proposed a simple and scalable method named RAT-SPNs, which utilize the region graph (Dennis & Ventura, 2012) (as shown in Definition 15) as a concise representation of the structure.

Definition 15 (*Region Graph*)**.** A region graph is a rooted DAG composed of regional nodes and partition nodes. Different types of nodes in the region graph are defined as follows:

(1) The root node of any region graph must be a region node.
(2) The children of any region node are partition nodes, and the children of any partition node are region nodes.
(3) A region node can be regarded as a sum node and a partition node can be regarded as a product node.
(4) All nodes have scopes just like nodes in SPNs.

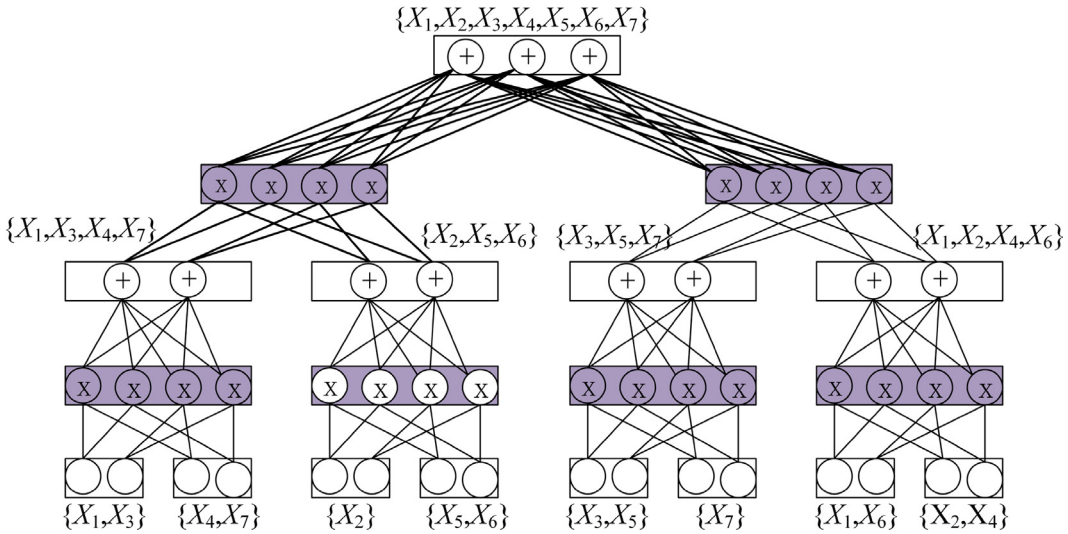


Fig. 14. An example of a RAT-SPN. *Source:* Image extracted from Peharz, Vergari et al. (2020).

RAT-SPNs can easily construct the corresponding structure by using the random region graph: construct a random region graph firstly, then populate it with SPN nodes (an instantiated RAT-SPN is shown in Fig. 14). RAT-SPNs can not only scale to millions of parameters by generating unspecialized random structures but also apply GPU-based optimizations. The experimental results showed that RAT-SPN has comparable classification accuracy as DNNs on MNIST, Fashion-MNIST and other datasets. RAT-SPNs are also capable of detecting outliers and are robust to missing features. In general, SPNs can achieve remarkable improvements in classification, and robust and outlier detection, even without complex structure learning methods.

Bayesian SPNs are well-principled methods for SPN learning, which can learn the structure and parameters of SPN simultaneously. Based on Definition 14, Bayesian SPNs decomposed the SPN structure learning into two steps, (1) constructing a computational graph randomly, (2) learning the scope function by Bayesian inference over the graph. A computational graph (Fig. 15(a)) and its corresponding SPN structure (Fig. 15(b)) are shown in Fig. 15, where the notation ψ denotes the scope function and the SPN structure is obtained by posterior inference on an encoding of ψ. The generative model is as follows:

$$\begin{aligned}
&w_S \mid \alpha \sim Dir(w_S \mid \alpha)\ \forall S,\\
&Z_{S,n} \mid w_S \sim Cat(Z_{S,n} \mid w_S)\ \forall S, \forall n\\
&V_P \mid \beta \sim Dir(V_P \mid \beta)\ \forall P,\\
&y_{P,d} \mid V_P \sim Cat(y_{P,d} \mid V_P)\ \forall P, \forall d\\
&\theta_L \mid \gamma \sim Dir(\theta_L \mid \gamma)\ \forall L,\\
&x_n \mid z_n, y, \theta \sim \prod_{L \in T(z_n)} L(x_{y,n} \mid \theta_L)\ \forall n.
\end{aligned} \tag{22}$$

where the notation w denotes the sum-weights, θ denotes the leaf-parameters, and the notation $\mathbf{x}_{y,n}$ denotes the evaluation of L on the scope induced by $\mathbf{y}$. The generative model is illustrated in Fig. 16.

In general, Bayesian SPNs have the following advantages, including robustness for missing features, natural extension to non-parametric formulations, and prevention of overfitting.

3.3.2. Some other methods related to random structure learning

The existing random structure learning algorithms have achieved good results, but there are still many problems, such as scalable to larger datasets and solve the overfitting problem. Peharz, Lang et al. (2020) pointed out that previous work was not easily scalable to larger datasets due to the sparse connected computational graphs. The Einsum Networks named EiNets was proposed, which proceeds by combining many arithmetic operations in a single monolithic einsum-operation. The experimental results showed that the results of EiNets are comparable with RAT-SPN for density estimation, but outperformed it in speed and memory saving. In addition, EiNets can be used for large datasets such as SVHN and CelebA. Later, a self-attentional mechanism (Yu, Dhami and Kersting, 2021) is proposed to select the most relevant part of the probabilistic circuit to reduce the size of the structure.

Shih, Sadigh, and Ermon (2021) proposed an algorithm named HyperSPNs, which introduces a smaller-scale external neural network to generate the parameters of large SPNs (such as RAT-SPN and EiNets). HyperSPNs provides regularization to SPNs with a large number of parameters by reducing the degrees of freedom through a soft weight-sharing strategy. As shown in Fig. 17, parameters in the SPN model are trained by the blue module.

3.4. Ensemble structure learning

We define structure learning based on ensemble learning as ensemble structure learning. Many disadvantages and challenges of SPN structure learning methods exist in Sections 3.1, 3.2 and 3.3, such as (1) none of the methods used previously can be well generalized to larger and deeper structures to accommodate complex and high-dimensional data, (2) single circuit might not be accurate enough, for example, random structure learning methods may also yield larger and deeper structures, but the assumption of independence may be wrong, which leads to sub-optimal problems, (3) deeper and wider probabilistic models are more difficult to train.

To compensate for these shortcomings, Ventola et al. (2020) proposed an ensemble learning framework named ResSPN, which makes it possible to combine arbitrary groups of SPNs into a larger model. ResSPN first generates several sub-SPNs randomly through LearnSPN and conducts residual links to the internal nodes of these SPNs (i.e., ensemble SPNs by explicitly reformulating sum nodes as residual functions, as shown in Fig. 18). The root nodes of sub-SPNs are connected through each node into a

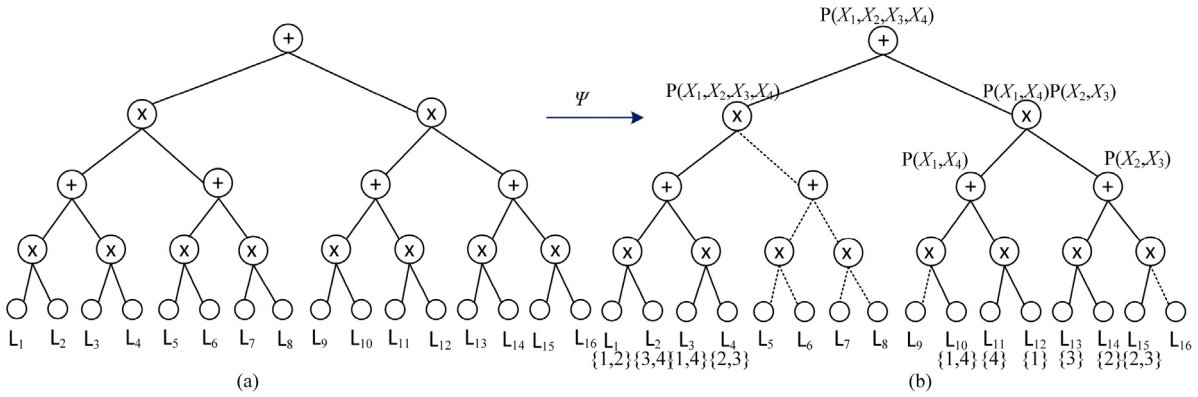


Fig. 15. An example of Bayesian SPNs.
Source: Image extracted from Trapp et al. (2017).

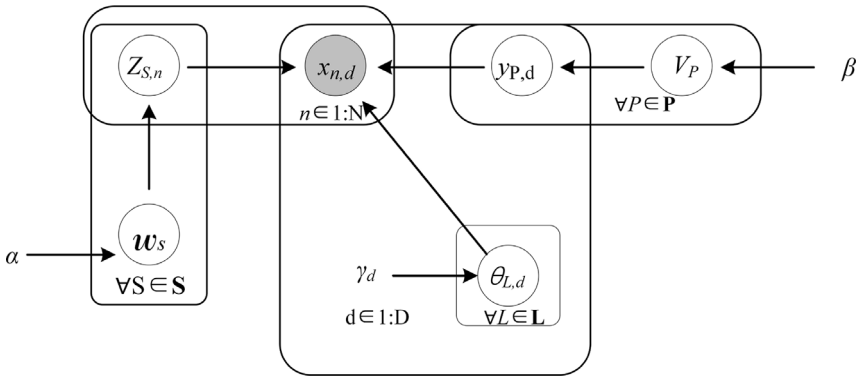


Fig. 16. Plate notation of Bayesian SPNs for Bayesian structure and parameter learning.
Source: Image extracted from Trapp et al. (2017).

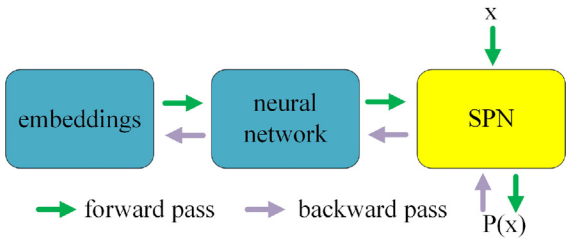


Fig. 17. The architecture of HyperSPN. Green arrows show the materialization of the SPN in the forward pass. Purple arrows represent the flow of gradients during backward pass.
Source: Image extracted from Shih et al. (2021).

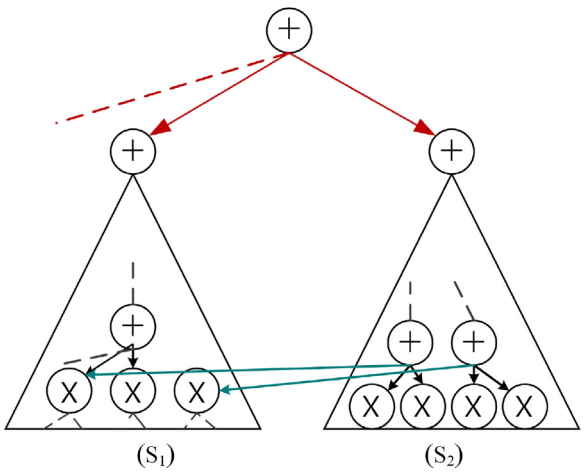


Fig. 18. Ensemble SPNs by explicitly reformulating sum nodes as residual functions. The blue edges denote residual links between the SPN of (S_1) and (S_2), which allow (S_2) to fit a residual. The red edges realize the global mixture.
Source: Image extracted from Ventola et al. (2020).

larger SPN. Finally, the simplified SPN is obtained by the pruning method. The experimental results showed that the ResSPNs are easy to optimize and competitive with existing SPN structure learning methods.

3.5. Summary and discussion

We make a comparative analysis of SPN structure learning methods in Table 4. Generally, most of the SPN structure learning methods are based on data, which demonstrates the importance of data-based structure learning in SPN structure learning. Comparatively, random structure learning and ensemble structure learning are relatively new research topics, and there are few related studies. However, as the development of SPNs has been accelerated by the emergence of RAT-SPN and EiNets, random structure learning shows great potential for future research and applications.

Currently, there are still many problems to be solved in SPN structure learning, which are still in the research stage. Designing an appropriate SPN structure learning method is an open topic that requires further exploration. Additionally, in some domains

Table 4
Comparison of characteristics of SPN structure learning methods.

Methods	Main references	Advantages	Disadvantages
Handcrafted structure learning	Gens and Domingos (2012) and Poon and Domingos (2011)	Expert-designed structures are highly targeted.	1. Fixed architectures are difficult to apply to other fields except the image field. 2. Time-consuming and laborious. 3. Inflexible (the most valuable nodes may not be in fixed structure).
Data-based structure learning	Adel et al. (2015), Dennis and Ventura (2012, 2015), Gens and Pedro (2013), Krakovna and Looks (2016), Lee et al. (2014), Molina et al. (2018), Peharz, Gens et al. (2014), Rahman and Gogate (2016), Rooshenas and Lowd (2014), Trapp et al. (2016) and Vergari et al. (2015)	Can be applied to other fields except the image field.	1. Complex structures, difficult to be used for large datasets. 2. Not easy to integrate with deep learning framework. 3. Weak generalization ability (different datasets need to be relearned). 4. Difficult to learn deep structures.
Random structure learning	Peharz, Vergari et al. (2020), Rashwan et al. (2016) and Trapp et al. (2019)	Simple, scalable method, without specific data.	The independence assumption can be wrong, which leads to sub-optimal problems.
Ensemble structure learning	Ventola et al. (2020)	Making it possible to combine arbitrary groups of SPNs into a larger model.	Inherits the shortcomings of ensemble learning.

and data types, the task of designing an appropriate structure remains unexplored.

4. SPN structure learning based on data types

As previously mentioned, SPNs have been successfully applied across various fields. Considering that the data types across these fields may exhibit considerable heterogeneity, it is of great research value to investigate SPN structure learning methods under various data types (See Section 5.1 for a detailed analysis of the datasets). In this section, we focus on the study of SPN structure learning based on different data types. We summarize and categorize the existing SPN structure learning data into four types based on the data source and characteristics, namely general data, online data, sequence data, and relational data.

4.1. SPN structure learning built on general data

For convenience, We name the data containing only samples (instances) and features (variables) as general data, such as image data (such as MNIST LeCun, Bottou, Bengio, & Haffner, 1998 and Fashion-MNIST Xiao, Rasul, & Vollgraf, 2017). Since the SPN structure learning algorithms reviewed in Section 3 are mainly for general data type, we will not repeat them in this section.

4.2. SPN structure learning built on online data

Most SPN structure learning algorithms assume that the full datasets are available. However, the computers may not have enough memory to store large datasets at once and the data arrive constantly sometimes. It is necessary to learn algorithms that can update structures as new data arrives. Jaini et al. (2016) proposed an online algorithm for SPNs, but this algorithm is aimed at parameter learning in continuous SPNs. The idea of this algorithm is to generate the random network structure first, and then the parameters are learned through an extensible online learning technique.

oSLRAU (Hsu et al., 2017) as mentioned in Section 3.2.1 is also an online structure learning algorithm for Gaussian SPNs, which adopts a bottom-up method to build the SPN structure incrementally. In contrast to Lee, Heo, and Zhang (2013) (The product node cannot be changed once it is constructed.), the oSLRAU derives its correlations by detecting correlations and constantly changing product nodes. Dennis and Ventura (2017) proposed an online method based on SearchSPN named ONLINESEARCHSPN. ONLINESEARCHSPN is suitable for data with changing input distributions and the training dataset is large in the offline setting. Kalra et al. (2018) described an algorithm for feed-forward and recurrent SPNs that can also be used to learn larger datasets.

Prometheus (Jaini et al., 2018) as mentioned in Section 3.2.1 is a graph partitioning-based algorithm, which can efficiently create multiple variable decompositions and be used for SPN structure learning in continuous or discrete domains. Prometheus adopts graph partitions to construct sequences of product nodes with different factor distributions, and induces the mixed distribution on these product nodes as sum nodes. Direct interactions between variables are captured by an affinity matrix using similarity measures. Correspondingly, the similarity is defined as

$$\left|\rho_{z_i z_j}\right| = \left| \frac{\sum_{k=1}^{n}(z_i^{(k)} - \bar{z}_i)(z_j^{(k)} - \bar{z}_j)}{\sqrt{\sum_{k=1}^{n}(z_i^{(k)} - \bar{z}_i)^2}\sqrt{\sum_{k=1}^{n}(z_j^{(k)} - \bar{z}_j)^2}} \right| \tag{23}$$

where $\bar{z}_i = \frac{\sum_{k=1}^{n} z_i^{(k)}}{n}$ *and* $\bar{z}_j = \frac{\sum_{k=1}^{n} z_j^{(k)}}{n}$.

In addition, a lower-dimensional version of Prometheus was created to scale to higher-dimensional datasets, in which not all variables are used in each step.

4.3. SPN structure learning built on sequence data

Sequence data refers to a collection of data points that are ordered or arranged in a specific sequence. Existing research on SPN structure learning methods based on sequence data is limited. Only three SPN structure learning methods (Kalra et al., 2018; Melibari, Poupart, Doshi, & Trimponias, 2016; Yu, Ventola and Kersting, 2021) related to sequence data are found. However, there exist numerous tasks associated with sequence data, including but not limited to modeling of time series data, gene sequence data, and protein sequence data.

A dynamic SPNs (DSPNs) (Melibari et al., 2016) is proposed for sequence data composed of instances with different lengths. DSPNs is an anytime algorithm based on the framework of search-and-score, which can be described as the following steps: (1) DSPNs starts with an arbitrary structure and generates several adjacent structures. (2) The adjacent structures are sorted according to the scoring function (likelihood of the data). (3) Select the

best adjacent structure. The results showed that DSPNs outperforms static algorithms of SPNs on sequence data and achieves a higher likelihood than Hidden Markov Model, dynamic Bayesian networks and recurrent neural networks on several temporal datasets.

Kalra et al. (2018) proposed a structure learning method that extends oSLRAU to dynamic settings. The method matches the chain length to the horizon by expanding SPNs and determines the time to change the new correlation in template by sharing weights and covariance matrixes. The experimental results showed that this method has higher accuracy and scalability than offline search-and-score technology (Melibari et al., 2016). In addition, Yu, Ventola et al. (2021) proposed an algorithm named Whittle sum–product networks (WSPNs), which can obtain the distribution on time series data. WSPN adopts a data-based structure or random structure as a structure learning method.

4.4. SPN structure learning built on relational data

In many real-world applications, there are many dependent instances (such as knowledge graphs and social networks), which are called relational data. If the dependency between instances is taken into account, the effectiveness of the model can be improved. There are only two Refs. Nath and Domingos (2015) and Zheng et al. (2018) on SPN structure learning methods based on relational data, which are respectively applied in the fields of social networks and robotics.

Nath and Domingos (2015) introduced a new tractable relational probabilistic architecture called Relational Sum–Product Network (RSPN), which models relational data on the basis of tractable first-order probabilistic logic (TML) (Domingos & Webb, 2012). RSPN constructs the SPN structure by jointly modeling a group of interacting instances. This method enables the RSPN structure to learn the probability distributions of interacting instances and the probabilities of relationships between objects. The structure learning algorithm of RSPN is named LearnRSPN, which is a relational extension of LearnSPN. Compared with Markov Logic Networks (Richardson & Domingos, 2006), LearnRSPN is faster in running time and has higher prediction accuracy.

RSPN builds graphs of unfixed sizes by aggregating multiple variables using an aggregate statistic. In contrast, Zheng et al. (2018) introduced an algorithm named GraphSPN, which directly models arbitrary output variables associated with the graph nodes and constructs a structure for each graph instance. In addition, a method based on GraphSPN for learning topological spatial relations in semantic maps made by a mobile robot was proposed. The template SPNs are learned by using a simple structure learning technique (Pronobis & Rao, 2017). GraphSPN initializes the SPN with dense structure firstly, i.e., recursively creating nodes based on multiple random decompositions of variable sets into multiple subsets. The final structure consists of combining the products of subsets in each decomposition and the summation of these products. Experimental results showed that GraphSPN is robust to noise and uncertainty in practical scenarios.

4.5. Summary and discussion

Since the data types may be different in different application scenarios, the study of SPN structure learning based on different data types will promote the wide application of SPNs. In this section, we review the literature (Kalra et al., 2018; Lee et al., 2013; Melibari et al., 2016; Pronobis & Rao, 2017; Zheng et al., 2018) on SPN structure learning under different data types. Comparatively, there are few kinds of research on SPN structure learning methods of other data types except general data, which are relatively new research topics. Many challenges remain to be solved. For example, how to design an SPN structure learning algorithm that takes graph data as input and optimizes the SPN structure by maximizing the likelihood function? In addition, how to design an SPN structure learning method suitable for large and complex graph data? In general, SPN structure learning based on different data types has great potential in both research and applications.

5. SPN structure learning in practice

In this section, we list datasets related to SPN structure learning methods in detail. The main reasons are as follows: (1) In order to verify the effectiveness of SPN structure learning method. (2) To help better explore the practical application value of SPNs and promote the generation of SPN benchmark dataset (currently, SPNs are mainly aimed at general data, but there are many other data types in reality, such as relational data, for which there is no benchmark dataset at present). In addition, we also summarize the evaluation tasks and source codes that are commonly used in SPN structure learning algorithms.

5.1. Datasets

Datasets are an important part for empirical analysis of SPN structure learning methods. The datasets used for evaluating SPN structure learning algorithms can be divided into four categories according to the data types as mentioned in Section 4: general datasets, online datasets, relational datasets and sequence datasets. There are relatively mature benchmark datasets to evaluate SPN structure learning methods for general data. However, there are few structural learning methods for other data types and almost no common benchmark datasets, such as structural learning methods based on relational data and sequence data. It is a challenge to study SPN structure learning methods for different data types, and it is necessary to design appropriate benchmark datasets for different data types.

In order to facilitate the empirical analysis of the surveyed algorithms, we also list public and real-world datasets with different characteristics from various domains in Table 5.

5.1.1. General datasets

The general datasets used in SPN structure learning can be roughly divided into three categories: image datasets, Twenty real-world datasets and UCI Benchmarks. We will cover these general datasets related to SPN structure learning in detail in this section.

Image Datasets We provide a detailed account of the image datasets utilized for SPN structure learning. The datasets serve various purposes including classification, image completion, and outlier detection tasks (Dennis & Ventura, 2012; Molina et al., 2018; Peharz, Lang et al., 2020; Peharz, Vergari et al., 2020). Further information regarding these datasets is presented in a tabular format in Table 6.

- Caltech-101 (Fei-Fei, 2004). This is a well-known dataset containing 101 categories of images, such as scenery, faces, helicopters, and animals.
- Olivetti face (Samaria & Harter, 1994). This dataset contains 400 face images taken between April 1992 and April 1994 at *AT&T* Laboratories Cambridge.
- SVHN (Netzer et al., 2011) and CelebA (Liu et al., 2015). SVHN and CelebA are all RGB images. SVHN contains images of Arabic digits and the images are from the house numbers taken in Google Street View. CelebA contains face images. Larger and more complex datasets are available since EiNets (Peharz, Lang et al., 2020) was proposed in 2020.

Table 5
The datasets used in SPN structure learning experiments ordered by data types.

Data type	Datasets	Domain	Main references
General data	Image datasets (Coates, Ng, & Lee, 2011; Fei-Fei, 2004; Krizhevsky, 2009; LeCun et al., 1998; Liu, Luo, Wang, & Tang, 2015; Netzer et al., 2011; Samaria & Harter, 1994; Xiao et al., 2017).	Image	Dennis and Ventura (2012), Gens and Domingos (2012), Kalra et al. (2018), Molina et al. (2018), Peharz et al. (2013), Peharz, Lang et al. (2020), Peharz, Vergari et al. (2020), Poon and Domingos (2011), Ventola et al. (2020) and Yu, Dhami et al. (2021)
	Twenty real-world datasets (Lowd & Davis, 2010; Van Haaren & Davis, 2012)	Plant habitats, news, Web page, Joke score, and many others.	Adel et al. (2015), Butz et al. (2018), Dennis and Ventura (2015), Gens and Pedro (2013), Krakovna and Looks (2016), Peharz et al. (2013), Peharz, Gens et al. (2014), Peharz, Lang et al. (2020), Peharz, Vergari et al. (2020), Rahman and Gogate (2016), Rooshenas and Lowd (2014), Trapp et al. (2019), Ventola et al. (2020), Vergari et al. (2015) and Yu, Dhami et al. (2021)
	General datasets from UCI machine learning repository (Asuncion & Newman, 2007)	Financial, medical, automotive, physics, Chem., and others.	Bueff et al. (2018), Molina et al. (2018) and Trapp et al. (2019)
	IMDB (Yueming, 2018)	Movie	Peharz, Vergari et al. (2020)
Online data	Online datasets from UCI machine learning repository, Drive (Staal, Abràmoff, Viergever, & van Ginneken, 2004), Voxforge (MacLean, 2018), Power[a], Network (Liang & Klein, 2009), MSD (Michela, Annika, & Spyridon, 2019), GasSen[a], GasSenH[a], Twenty real-world datasets, Airline (Harvard, 2008).	Financial, medical, automotive, physics, Chem., flights, and other sectors.	Dennis and Ventura (2017), Hsu et al. (2017), Jaini et al. (2016) and Lee et al. (2013)
Relational data	Relational data from the UW-CSE database (Khosravi, Schulte, Hu, & Gao, 2012), Semantic maps (Zheng et al., 2018), artificial social networks (Singla & Domingos, 2008).	UW-CSE domain, Friends and Smokers domain, Robotics.	Nath and Domingos (2015) and Zheng et al. (2018)
Sequence data	Sequence datasets from the UCI machine learning repository, eegEye[a], libras[a].	Digits, physical, and other sectors.	Kalra et al. (2018), Melibari et al. (2016) and Yu, Ventola et al. (2021)

[a]Represent the datasets used in SPN structure learning algorithm without labeled data source and we still cannot determine the data source after extensive search.

Table 6
The image datasets used in SPN structure learning. The number of samples (#Ins), the number of classes (#C), the size of image, characteristic and task are also provided for each dataset.

Datasets	#C	#Ins	Size (pixel)	Characteristic	Task
Caltech-101	101	9146	64 × 64	Small	Reconstruction
Olivetti face	None	400	64 × 64	Small	Reconstruction
CIFAR-10	10	60 000	32 × 32	Medium	Classification Reconstruction
STL-10	10	13 000	96 × 96	Small	Reconstruction
MNIST	10	70 000	28 × 28	Medium	Classification Reconstruction Outlier detection
Fashion-MNIST	10	70 000	28 × 28	Medium	Classification
SVHN	None	630 420	32 × 32 RGB	Large	Reconstruction Outlier detection
CelebA	10,177	202,599	128 × 128 RGB	Large	Reconstruction

- CIFAR-10 (Krizhevsky, 2009), MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017) and STL-10 (Coates et al., 2011). These are all 10-class image datasets. Fashion-MNIST is an alternative to MNIST, which aims to categorize fashion items rather than digits. STL-10 has larger pixel images and less labeled data than CIFAR-10. CIFAR-10, MNIST and Fashion-MNIST are the most commonly used evaluation datasets for SPN structure learning algorithms.

Twenty Real-world Datasets Twenty real-world datasets is much larger than those in the empirical evaluation of typical graphical model structure learning. The Twenty real-world datasets are detailed in Lowd and Davis (2010) and Van Haaren and Davis (2012). All the features in Twenty real-world datasets are binary. These datasets were used to evaluate the log-likelihood and pseudo-loglikelihood of SPN structure learning methods.

Table 7
Online datasets for SPN structure learning. (#Vars) indicates the number of observed variables, (#Dats) indicates the number of data points.
Source: Part of the data in the table are extracted from the Supplementary of Kalra et al. (2018).

Dataset	#Vars	#Dats
Sensorless	48	–
Abalone	8	–
Banknote	4	–
Quake	4	–
Kinematics	8	–
CA	22	–
FlowSize	3	–
Network	3	434,873
Power	4	2,049,280
GasSenH	10	928,991
GasSen	16	8,386,765
Voxforge	39	3,603,643
MSD	90	515,344

The first seven rows in the table represent seven real-world small and continuous online datasets, and the last six rows represent six large and continuous online datasets.

UCI Benchmarks The UCI Machine learning repository with many different domains currently maintains 622 datasets and is widely used in many fields. Each dataset consists of different proportions of continuous, categorical, and discrete features. UCI datasets used in SPN structure learning can be found in Bueff et al. (2018), Molina et al. (2018) and Trapp et al. (2017).

5.1.2. Online datasets

The online SPN structure learning method mainly means that the SPN structure can be constantly updated as new data comes in (according to the relevant literature in Section 4.2), so the online dataset has an intersection with other types of datasets. For example, some large datasets of other types can learn the SPN structure through the online SPN structure learning method to alleviate the problem of insufficient memory. The online datasets here mainly refer to the dataset used for the verification of the online SPN structure learning method.

The online datasets used in SPN structure learning are mainly divided into two categories: continuous and discrete datasets. Continuous datasets are divided into large and small datasets (such as Table 7). For discrete datasets, Twenty real-world datasets are mainly used as the benchmark dataset at present.

5.1.3. Relational datasets

There are only two papers for SPN structure learning based on relational data and no relevant benchmark datasets.[8] The UW-CSE database (Khosravi et al., 2012) and artificial social networks in Friends and Smokers domain (Singla & Domingos, 2008) have been used to evaluate the relational SPNs proposed in Nath and Domingos (2015). In addition, Zheng et al. (2018) contributed a dataset[9] of the topological graph to verify the validity of the proposed method.

5.1.4. Sequence datasets

Due to the limited number of research studies on SPN structure learning methods for sequence data, a standardized benchmark dataset for the evaluation of the performance of SPN structure learning on sequence data is yet to be established. Kalra et al. (2018) and Melibari et al. (2016) adopted several sequence datasets from the UCI repository (Asuncion & Newman, 2007) to verify structure learning methods, as described in Table 8. While both algorithms Kalra et al. (2018) and Melibari et al. (2016) have evaluated their methods using the ozLevel and JapanVowels datasets, the number of instances considered in each study differs. Additionally, Kalra et al. (2018) has employed the eegEye and libras datasets to validate their approach.

Table 8
Sequence datasets, (#i, length, #Vars) indicates the number of data instances, length of each sequence and the number of observed variables.
Source: Table extracted from Kalra et al. (2018) and Melibari et al. (2016).

Dataset	#i	length	#Vars
ozLevel	2533/2170	24	2
PenDigits	10 992	16	7
ArabicDigits	8800	40	13
JapanVowels	640/270	16	12
ViconPhysic	200	3026	27
hillValley	600	100	1
eegEye	14 970	14	1
libras	350	90	1

2533/2170 indicates that the number of instances used by the two algorithms (Kalra et al. (2018) and Melibari et al. (2016)) is 2533 and 2170, respectively. Similarly, 640/270 means the same thing.

5.2. Evaluation tasks

In this section, we analyze the main evaluation tasks of SPN structure learning to help later researchers understand and apply SPN structure learning methods.

5.2.1. Log-likelihood

The canonical objective function of SPN structure learning is maximizing the log-likelihood:

$$L_L(\theta) = \frac{1}{N}\sum_{i=1}^{N} \log \mathcal{S}(x_n), \tag{24}$$

where θ denotes the sum weights and parameters of input distributions. $\mathcal{S}(X)$ denotes values of the root node (distribution over $\mathbf{X}$). The maximum log-likelihood and conditional (marginal) log-likelihood values can be used to evaluate the quality of the structure (Dennis & Ventura, 2012; Lee et al., 2014; Molina et al., 2018; Peharz, Gens et al., 2014; Rooshenas & Lowd, 2014; Trapp et al., 2016). Most structure learning algorithms use log-likelihood to evaluate the performance of the structure. The marginal likelihood values over inputs $\mathbf{X}$ can also be used to detect outliers (Peharz, Lang et al., 2020; Peharz, Vergari et al., 2020).

5.2.2. Node classification

Let $\mathbf{X} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be a training set of features x_n and labels y_n. The main objective function for the classification task is cross-entropy. For example, RAT-SPN (Peharz, Vergari et al., 2020) trained classifiers by minimizing the cross-entropy:

$$L_{CE}(\theta) = -\frac{1}{N}\sum_{n=1}^{N} \log \frac{S_{y_n}(x_n)}{\sum_{y'} S_{y'}(x_n)}, \tag{25}$$

where $S_{y_n}(x_n)$ denotes the value of the root node to which category y_n belongs under the condition that the input is x_n. The classification accuracy can be used to evaluate the structure of SPNs. In addition, the robustness of SPN structure learning algorithms in missing data can also be inferred by classification accuracy (Peharz, Lang et al., 2020; Peharz, Vergari et al., 2020). In order to trade off cross-entropy and log-likelihood,

[8] As there is no unified dataset for the verification of SPN structure learning methods based on relational data (and the standards for the dataset in the two available literature sources are not uniform.), we do not list the dataset in detail, but give the website of the dataset or the referenced literature.

[9] http://coldb.org/graphs.

Table 9
A summary of the source code.

Handcrafted structure learning	
Methods	Source code
SPN (Poon & Domingos, 2011)	http://spn.cs.washington.edu/spn/
Data-based structure learning	
Methods	Source code
LearnSPN (Gens & Pedro, 2013)	https://github.com/RenatoGeh/gospn
ID-SPN (Rooshenas & Lowd, 2014)	https://libra.cs.uoregon.edu/
LearnSPN-B/T/B (Vergari et al., 2015)	https://github.com/arranger1044/spyn
oSLRAU (Hsu et al., 2017)	https://github.com/whsu/spn
GraphSPNs (Zheng et al., 2018)	https://github.com/zkytony/graphspn
Random structure learning	
Methods	Source code
Bayesian SPNs (Trapp et al., 2019)	https://github.com/trappmartin/BayesianSumProductNetworks
RAT-SPN (Peharz, Vergari et al., 2020)	https://github.com/cambridge-mlg/RAT-SPN
EiNets (Peharz, Lang et al., 2020)	https://github.com/cambridge-mlg/EinsumNetworks
HyperSPN (Shih et al., 2021)	https://github.com/ermongroup/HyperSPN

Since there is no unified software (such as c, c++, python, Julia and FPGA), software libraries (such as LibSPN, SPFlow, Tachyon, GoSPN, BayesianSumProductNetworks.jl and SumProductNetworks.jl) and frameworks (such as PyTorch and TensorFlow) for SPN structure learning, we do not conduct a unified experimental comparison of different algorithms. See https://arranger1044.github.io/probabilistic-circuits/code.

RAT-SPN combines Eqs. (24) and (25) into a hybrid generative-discriminative (Bouchard & Triggs, 2004) loss:

$$L_{L+CE}(\theta) = \lambda L_{CE}(\theta) - (1-\lambda)\frac{L_L(\theta)}{|\mathbf{X}|}. \tag{26}$$

When $0 < \lambda < 1$, the equation Eq. (26) permits classifiers of the model to apprehend the distributions over $\mathbf{X}$. This is a crucial characteristic of handling uncertainty over inputs, such as missing values of input features.

5.2.3. Link prediction

Graph data (relational data) composed of nodes and edges has become a research hotspot. This data type may not only have the features of high dimensions, but also has an edge matrix, which makes it difficult to apply SPNs to graph-structured data. The link prediction task is to predict the existence of edges between nodes in the graph, which has been widely concerned in recommendation systems, knowledge graphs and other fields (Lü & Zhou, 2011). There is only one relevant research method (RSPNs) (Nath & Domingos, 2015) for SPN structure learning using link prediction task. RSPNs validated the link prediction task using artificial social networks, which were constructed in the Friends-and-Smokers domains (Singla & Domingos, 2008) using a generalization of the Barabási–Albert preferential attachment model (Barabási & Albert, 1999). There is no SPN structure learning algorithm that can be applied to actual social networks, such as Twitter (De Choudhury et al., 2010), YOUTUBE (Tang & Liu, 2009b), and BLOGCATALOG (Tang & Liu, 2009a).

To summarize, the application of SPN to link prediction remains a challenging problem.

5.2.4. Learning and inference time

Learning and inference time are also essential evaluation tasks that can be used to compare the complexity and timeliness of SPN structure learning algorithms. There are many relevant research methods for SPN structure learning using learning and inference time as tasks, such as LearnSPN and ID-SPN.

5.3. Open source software

We have collated the source code links for all current SPN structure learning algorithms in Table 9.

6. Discussion

SPNs first gained attention due to their tractability of inference (Liu, Mandt, & den Broeck, 2022; Zečević et al., 2021). Since SPNs were proposed in 2011, great progress has been made in inference, representation, learning and application (Amer & Todorovic, 2016; Butz et al., 2019; Peharz, Gens et al., 2014; Peharz, Vergari et al., 2020; Sánchez-Cauce et al., 2021; Skryagin, Stammer, Ochs, Dhami, & Kersting, 2022; Skryagin et al., 2020; Stelzner, Peharz, & Kersting, 2019; Trapp et al., 2019; Vergari, Peharz, Mauro, & Esposito, 2017; Wang & Wang, 2018). However, some challenges in studying SPNs still remain. In theory, including representation, inference, and learning, we believe that enhancing their interpretability and generalization ability are promising and potential research directions. Getting good results when using SPNs to handle various tasks is certainly important, but understanding the underlying reasons is equally crucial. Enhancing their interpretability will help us find the reasons. Furthermore, existing SPNs have focused on dealing with Euclidean space data, such as images, with few focusing on other data types, particularly complex networks (Jiao et al., 2023; Martínez, Berzal, & Talavera, 2017; Nath & Domingos, 2015; Zheng et al., 2018). Therefore, enhancing the generalization ability of SPNs to handle diverse data types is essential. For the application of SPNs, SPNs can serve as a tool for modeling neuro-symbolic systems (Wang & Yang, 2022), because tractable boolean and arithmetic circuits (Hitzler & Sarker, 2021) have already been used as tools for modeling neuro-symbolic systems. As SPN is an important branch of tractable ACs, we believe that utilizing SPNs to model neuro-symbolic systems is an interesting research direction.

As we mentioned in Section 1, SPN structure learning is vital for balancing the expressiveness and complexity of SPNs. Although several kinds of SPN structure learning methods and their applications for real-world tasks have been proposed in the past 10 years, as we discussed in Sections 3 to 5, some issues in this research area still exist. The research directions to address these issues are as follows.

Scalability Extending the SPN structure learning algorithms to large-scale datasets[10] is challenging. While some SPN structure

[10] In this paper, large-scale datasets mean the datasets contain too many samples, or the datasets contain samples with the high-dimensional feature vectors.

learning methods have been proposed to improve their scalability (Peharz, Lang et al., 2020; Peharz, Vergari et al., 2020), much work remains to be done in learning tractable models for larger data points and feature dimensions in tractable time. For example, EiNets (Peharz, Lang et al., 2020) have been demonstrated to scale well to SVHN (Netzer et al., 2011) and CelebA (Liu et al., 2015) datasets containing 630,420 and 202,599 samples, respectively, which were previously unattainable. However, there are many larger datasets, such as ImageNet (Deng et al., 2009) with 1,4197,122 images, in the real world. It is necessary to develop SPN structure learning frameworks that make models based on SPNs truly scalable to real-world production settings to prevent the disconnect between the academic research community and the application of methods.

Generalization ability SPN structure learning methods are mainly applied to general data types, such as image data. However, there are few SPN structure learning methods for other data types, such as relational data (Zheng et al., 2018), dynamic data (Melibari et al., 2016), online data (Kalra et al., 2018), and sequence data (Kalra et al., 2018). With the development of SPNs, it is necessary to learn the SPN structure for different types of data in different domains. In addition, there is no general or unified software, software library or framework for current SPN structure learning, which leads to inconvenient application and limits research progress. In general, improving the generalization ability of structure learning is helpful in promoting the development of SPNs.

Trade-off between expressiveness and complexity The structure of SPNs can affect their expressiveness and complexity. Some SPN structure learning methods improve the expressiveness currently, but they also increase the complexity, such as the methods that learn structures from data (Rooshenas & Lowd, 2014; Vergari et al., 2015). In addition, the random structure learning methods reduce the model complexity, but they also weaken the expressiveness (Peharz, Vergari et al., 2020; Trapp et al., 2017). It is significant to design an appropriate structure that can balance expressiveness and complexity. We believe that automatic machine learning techniques (Zöller & Huber, 2021) and neural architecture search techniques (Elsken, Metzen, & Hutter, 2019) are two promising approaches for balancing the complexity and expressiveness of SPN structures automatically.

Interpretability One of the key factors of whether artificial intelligence can be widely used is interpretability. Interpretability means that the person is able to understand the choices the model makes in its decision-making process. Although SPNs have certain probabilistic properties (Smit, 2020) that make it possible to partially explain why they make certain predictions, the independence of SPNs is less intuitive and difficult to read due to their more complex structure. Can the interpretability of SPNs be improved by enabling the model to make explainable decisions through SPN structure learning methods? For example, how to achieve mutual transformation between BNs and SPNs through a simple SPN structure learning method so that the performance of the two methods can be complementary, especially to enable SPNs to utilize the interpretability of BNs. Considering the significance of interpretability, we believe that improving the interpretability of SPN structure learning is also an important research direction.

Lifelong learning Most SPN structure learning methods mainly solve specific tasks on batch-type datasets,[11] such as image reconstruction and classification on Caltech-101 (Fei-Fei, 2004), MNIST (LeCun et al., 1998) and SVHN (Netzer et al., 2011) . In addition to batch-type datasets, there are few algorithms that can learn SPN structure on other types of datasets, such as the SPN structure learning algorithm based on online data. However, in reality, many datasets are changed over time (such as online data and sequence data) or the dataset is too large to be handled at once time, so lifelong learning is a promising research topic for the SPN structure learning in the feature.

[11] In machine learning, a batch-type dataset refers to a dataset that allows all the data to be loaded into memory at once and then inputted into the model in batches for training.

7. Conclusion

SPN structure learning is still a young and promising research field. Designing an appropriate SPN structure learning method to ensure that SPNs can be applied to practical applications is a frontmost question.

We have collected information spread in many publications and presented it in a coherent framework, trying to offer a comprehensive survey to SPN structure learning. In this survey, we introduced the relevant theories of SPNs and summarized the current SPN structure learning algorithms into four types according to the types of structure learning methods. To enlarge the application scope of SPNs, we summarized the existing SPN structure learning algorithms into four types according to data types. We also show the datasets and evaluation tasks. Finally, we discussed open issues and directions for future work.

To the best of our knowledge, this is the first survey on SPN structure learning, which is one of the most important research fields in SPNs since almost all models of SPNs need to learn its structure. We hope that this survey has provided a comprehensive and self-contained overview of the recent progress, challenges as well as future research directions in SPN structure learning to both academia and industry.

CRediT authorship contribution statement

Riting Xia: Investigation, Data curation, Methodology, Writing, Visualization. **Yan Zhang:** Investigation, Writing – review & editing. **Xueyan Liu:** Supervision, Investigation, Writing – review & editing. **Bo Yang:** Conceptualization, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant [2021ZD0112501, 2021ZD0112502]; the National Natural Science Foundation of China under Grant [U22A2098, 62172185, 62202200, 62206105]. We would like to thank the anonymous reviewers for their valuable comments.

References

Adel, T., Balduzzi, D., & Ghodsi, A. (2015). Learning the structure of sum–product networks via an svd-based algorithm. In *Proceedings of the thirty-first conference on uncertainty in artificial intelligence* (pp. 32–41). AUAI Press, URL: http://auai.org/uai2015/proceedings/papers/83.pdf.

Amer, M. R., & Todorovic, S. (2016). Sum product networks for activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*, 800–813. http://dx.doi.org/10.1109/TPAMI.2015.2465955.

Asuncion, A., & Newman, D. (2007). Uci machine learning repository. http://archive.ics.uci.edu/ml.

Bach, F., & Jordan, M. (2001). Thin junction trees. In *Advances in neural information processing systems* (pp. 569–576). URL: https://proceedings.neurips.cc/paper/2001/hash/325995af77a0e8b06d1204a171010b3a-Abstract.html.

Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, *286*, 509–512, URL: https://www.science.org/doi/full/10.1126/science.286.5439.509.

Bouchard, G., & Triggs, B. (2004). The tradeoff between generative and discriminative classifiers. In *16th IASC international symposium on computational statistics (COMPSTAT'04)* (pp. 721–728). URL: https://hal.inria.fr/inria-00548546.

Bueff, A., Speichert, S., & Belle, V. (2018). Tractable querying and learning in hybrid domains via sum–product networks. CoRR abs/1807.05464. URL: http://arxiv.org/abs/1807.05464, arXiv:1807.05464.

Butz, C. J., Oliveira, J. S., dos Santos, A. E., & Teixeira, A. L. (2019). Deep convolutional sum–product networks. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 3248–3255). http://dx.doi.org/10.1609/aaai.v33i01.33013248.

Butz, C. J., Oliveira, J. S., Santos, A. E., Teixeira, A. L., Poupart, P., & Kalra, A. (2018). An empirical study of methods for spn learning and inference. In *International conference on probabilistic graphical models* (pp. 49–60). PMLR, URL: http://proceedings.mlr.press/v72/butz18a.html.

Cheng, W. C., Kok, S., Pham, H. V., Chieu, H. L., & Chai, K. M. A. (2014). Language modeling with sum–product networks. In *Fifteenth annual conference of the international speech communication association* (pp. 2098–2102). ISCA, URL: http://www.isca-speech.org/archive/interspeech_2014/i14_2098.html.

Choi, A., & Darwiche, A. (2010). Relax, compensate and then recover. In *JSAI international symposium on artificial intelligence* (pp. 167–180). Springer, http://dx.doi.org/10.1007/978-3-642-25655-4_16.

Choi, M. J., Tan, V. Y., Anandkumar, A., & Willsky, A. S. (2011). Learning latent tree graphical models. *Journal of Machine Learning Research*, *12*, 1771–1812, URL: https://dl.acm.org/doi/10.5555/1953048.2021056.

Coates, A., Ng, A., & Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 215–223). JMLR.org, URL: http://proceedings.mlr.press/v15/coates11a/coates11a.pdf.

Darwiche, A. (2001). Recursive conditioning. *Artificial Intelligence*, *126*, 5–41. http://dx.doi.org/10.1016/S0004-3702(00)00069-2.

Darwiche, A. (2003). A differential approach to inference in bayesian networks. *Journal of the ACM*, *50*, 280–305. http://dx.doi.org/10.1145/765568.765570.

Dasgupta, S. (1999). Learning polytrees. In *Proceedings of the fifteenth conference on uncertainty in artificial intelligenc* (pp. 134–141). Morgan Kaufmann, URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=162&proceeding_id=15.

De Choudhury, M., Lin, Y. R., Sundaram, H., Candan, K. S., Xie, L., & Kelliher, A. (2010). How does the data sampling strategy impact the discovery of information diffusion in social media? In *Fourth international AAAI conference on weblogs and social media* (pp. 34–41). The AAAI Press, URL: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM10/paper/view/1521.

Dechter, R. (1998). Bucket elimination: A unifying framework for probabilistic inference. In *Learning in graphical models, Vol. 89* (pp. 75–104). Springer, http://dx.doi.org/10.1007/978-94-011-5014-9_4.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE computer society conference on computer vision and pattern recognition (CVPR 2009), 20-25 2009* (pp. 248–255). Miami, Florida, USA: IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2009.5206848.

Dennis, A., & Ventura, D. (2012). Learning the architecture of sum–product networks using clustering on variables. In *Advances in neural information processing systems* (pp. 3239–3247). URL: https://proceedings.neurips.cc/paper/2012/hash/f33ba15effa5c10e873bf3842afb46a6-Abstract.html.

Dennis, A., & Ventura, D. (2015). Greedy structure search for sum–product networks. In *Twenty-fourth international joint conference on artificial intelligence* (pp. 932–938). AAAI Press, URL: http://ijcai.org/Abstract/15/136.

Dennis, A., & Ventura, D. (2017). Online structure-search for sum–product networks. In *2017 16th IEEE international conference on machine learning and applications* (pp. 155–160). IEEE, http://dx.doi.org/10.1109/ICMLA.2017.0-164.

Domingos, P., & Webb, W. (2012). A tractable first-order probabilistic logic. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 1902–1909). URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5028.

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of the Machine Learning Research*, *20*, 55:1–55:21, URL: http://jmlr.org/papers/v20/18-598.html.

Fei-Fei, L. (2004). Learning generative visual models from few training examples. In *Workshop on generative-model based vision, IEEE proc. CVPR* (p. 178). IEEE Computer Society, http://dx.doi.org/10.1109/CVPR.2004.383.

Gens, R., & Domingos, P. (2012). Discriminative learning of sum–product networks. In *Advances in neural information processing systems* (pp. 3248–3256). URL: https://proceedings.neurips.cc/paper/2012/hash/573f7f25b7b1eb79a4ec6ba896debefd-Abstract.html.

Gens, R., & Pedro, D. (2013). Learning the structure of sum–product networks. In *International conference on machine learning* (pp. 873–880). JMLR.org, URL: http://proceedings.mlr.press/v28/gens13.html.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680). Curran Associates, Inc, URL: https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Harvard, D. (2008). Data expo 2009: Airline on time data. http://dx.doi.org/10.7910/DVN/HG7NV7.

Hitzler, P., & Sarker, M. K. (Eds.), (2021). *Frontiers in artificial intelligence and applications*: *Vol. 342, Neuro-symbolic artificial intelligence: the state of the art*. IOS Press, http://dx.doi.org/10.3233/FAIA342.

Hsu, W., Kalra, A., & Poupart, P. (2017). Online structure learning for sum–product networks with gaussian leaves. In *5th international conference on learning representations, ICLR 2017, Toulon, France, April (2017) 24-26, workshop track proceedings* (p. 1). OpenReview.net.

Jaini, P., Ghose, A., & Poupart, P. (2018). Prometheus: Directly learning acyclic directed graph structures for sum–product networks. In *International conference on probabilistic graphical models* (pp. 181–192). PMLR, URL: http://proceedings.mlr.press/v72/jaini18a.html.

Jaini, P., Rashwan, A., Zhao, H., Liu, Y., Banijamali, E., Chen, Z., et al. (2016). Online algorithms for sum–product networks with continuous variables. In *Conference on probabilistic graphical models* (pp. 228–239). PMLR, URL: http://proceedings.mlr.press/v52/jaini16.html.

Jiao, L., Chen, J., Liu, F., Yang, S., You, C., Liu, X., et al. (2023). Graph representation learning meets computer vision: A survey. *IEEE Transactions on Artificial Intelligence*, *4*, 2–22. http://dx.doi.org/10.1109/TAI.2022.3194869.

Kalra, A., Rashwan, A., Hsu, W. S., Poupart, P., Doshi, P., & Trimponias, G. (2018). Online structure learning for feed-forward and recurrent sum–product networks. In *Advances in neural information processing systems* (pp. 6944–6954). Curran Associates, Inc, URL: https://proceedings.neurips.cc/paper/2018/hash/66121d1f782d29b62a286909165517bc-Abstract.html.

Khosravi, H., Schulte, O., Hu, J., & Gao, T. (2012). Learning compact Markov logic networks with decision trees. *Machine Learning*, *89*, 257–277. http://dx.doi.org/10.1007/s10994-012-5307-6.

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems* (pp. 10236–10245). Curran Associates, Inc, URL: https://proceedings.neurips.cc/paper/2018/hash/d139db6a236200b21cc7f752979132d0-Abstract.html.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In *2nd international conference on learning representations, ICLR 2014, Banff, AB, Canada, April (2014) 14-16, conference track proceedings* (p. 1). URL: http://arxiv.org/abs/1312.6114.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT Press, URL: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11886.

Krakovna, V., & Looks, M. (2016). A minimalistic approach to sum–product network learning for real applications. (p. 1). arXiv abs/1602.04259. URL: http://arxiv.org/abs/1602.04259, arXiv:1602.04259.

Krizhevsky, A. (2009). Cifar-10 and cifar-100 datasets. http://www.cs.toronto.edu/kriz/cifar.html.

Larochelle, H., & Murray, I. (2011). The neural autoregressive distribution estimator. In *JMLR workshop and conference proceedings*, *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 29–37). URL: http://proceedings.mlr.press/v15/larochelle11a/larochelle11a.pdf.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324. http://dx.doi.org/10.1109/5.726791.

Lee, S. W., Heo, M. O., & Zhang, B. T. (2013). Online incremental structure learning of sum–product networks. In *International conference on neural information processing* (pp. 220–227). Springer, http://dx.doi.org/10.1007/978-3-642-42042-9_28.

Lee, S. W., Watkins, C., & Zhang, B. T. (2014). Non-parametric bayesian sum–product networks. In *ICML workshop on learning tractable probabilistic models* (p. 1). Citeseer, URL: http://spn.cs.washington.edu/papers/ltpm2014_paper_6.pdf.

Liang, P., & Klein, D. (2009). Online em for unsupervised models. In *Proceedings of human language technologies: the 2009 annual conference of the north american chapter of the association for computational linguistics* (pp. 611–619). The Association for Computational Linguistics, URL: https://aclanthology.org/N09-1069/.

Liu, Y., & Luo, T. (2019). The optimization of sum–product network structure learning. *Journal of Visual Communication and Image Representation*, *60*, 391–397. http://dx.doi.org/10.1016/j.jvcir.2019.02.012.

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision* (pp. 3730–3738). IEEE Computer Society, http://dx.doi.org/10.1109/ICCV.2015.425.

Liu, A., Mandt, S., & den Broeck, G. V. (2022). Lossless compression with probabilistic circuits. In *The tenth international conference on learning representations, ICLR 2022, virtual event, April (2022) 25-29* (p. 1). OpenReview.net, URL: https://openreview.net/forum?id=X_hByk2-5je.

Lowd, D., & Davis, J. (2010). Learning markov network structure with decision trees. In *2010 IEEE international conference on data mining* (pp. 334–343). IEEE. IEEE Computer Society, http://dx.doi.org/10.1109/ICDM.2010.128.

Lowd, D., & Domingos, P. (2008). Learning arithmetic circuits. In P. McAllester (Ed.), *Proceedings of the 24th conference in uncertainty in artificial intelligence, Helsinki, Finland, July 9-12* (pp. 383–392). AUAI Press, URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=1961&proceeding_id=24.

Lowd, D., & Rooshenas, A. (2013). Learning markov networks with arithmetic circuits. In *Artificial intelligence and statistics* (pp. 406–414). PMLR, URL: http://proceedings.mlr.press/v31/lowd13a.html.

Lü, L., & Zhou, T. (2011). Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, *390*, 1150–1170. http://dx.doi.org/10.1016/j.physa.2010.11.027.

MacLean, K. (2018). Voxforge. https://www.voxforge.org/home.

Martens, J., & Medabalimi, V. (2014). On the expressive efficiency of sum product networks. (p. 1). arXiv abs/1411.7717. URL: http://arxiv.org/abs/1411.7717, arXiv:1411.7717.

Martínez, V., Berzal, F., & Talavera, J. C. C. (2017). A survey of link prediction in complex networks. *ACM Computational Surveys*, *49*, 69:1–69:33. http://dx.doi.org/10.1145/3012704.

Mauro, N. D., Esposito, F., Ventola, F. G., & Vergari, A. (2017). Alternative variable splitting methods to learn sum–product networks. In *Conference of the Italian association for artificial intelligence* (pp. 334–346). Springer, http://dx.doi.org/10.1007/978-3-319-70169-1_25.

Meila, M., & Jordan, M. I. (2000). Learning with mixtures of trees. *Journal of Machine Learning Research*, *1*, 1–48, URL: http://jmlr.org/papers/v1/meila00a.html.

Melibari, M., Poupart, P., Doshi, P., & Trimponias, G. (2016). Dynamic sum product networks for tractable inference on sequence data. In *In conference on probabilistic graphical models* (pp. 345–355). PMLR, URL: http://proceedings.mlr.press/v52/melibari16.html.

Michela, A., Annika, R., & Spyridon, B. (2019). Medical segmentation decathlon. http://medicaldecathlon.com/.

Molina, A., Vergari, A., Di Mauro, N., Natarajan, S., Esposito, F., & Kersting, K. (2018). Mixed sum–product networks: A deep architecture for hybrid domains. In *Proceedings of the thirty-second AAAI conference on artificial intelligence, (AAAI-18)* (pp. 3828–3835). AAAI Press, URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16865.

Nath, A., & Domingos, P. M. (2015). Learning relational sum–product networks. In *Twenty-ninth AAAI conference on artificial intelligence* (pp. 2878–2886). AAAI Press, URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/10027.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *In NIPS workshop on deep learning and unsupervised feature learning* (p. 1). URL: http://www.iapr-tc11.org/dataset/SVHN/nips2011_housenumbers.pdf.

Park, J. D. (2002). Map complexity results and approximation methods. In *Proceedings of the eighteenth conference on uncertainty in artificial intelligence* (pp. 388–396). URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=883&proceeding_id=18.

Park, J. D., & Darwiche, A. (2004). Complexity results and approximation strategies for map explanations. *Journal of Artificial Intelligence Research*, *21*, 101–133. http://dx.doi.org/10.1613/jair.1236.

Peharz, R. (2015). *Foundations of sum–product networks for probabilistic modeling* (Ph.D. thesis), Medical University of Graz.

Peharz, R., Geiger, B. C., & Pernkopf, F. (2013). Greedy part-wise learning of sum–product networks. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 612–627). Springer, http://dx.doi.org/10.1007/978-3-642-40991-2_39.

Peharz, R., Gens, R., & Domingos, P. (2014). Learning selective sum–product networks. In *Proceedings of the 31st international conference on machine learning* (p. 1). URL: https://pure.tugraz.at/ws/portalfiles/portal/2673084/selSPN.pdf.

Peharz, R., Gens, R., Pernkopf, F., & Domingos, P. (2016). On the latent variable interpretation in sum–product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*, 2030–2044. http://dx.doi.org/10.1109/TPAMI.2016.2618381.

Peharz, R., Kapeller, G., Mowlaee, P., & Pernkopf, F. (2014b). Modeling speech with sum–product networks: Application to bandwidth extension. In *2014 IEEE international conference on acoustics, speech and signal processing* (pp. 3699–3703). IEEE, http://dx.doi.org/10.1109/ICASSP.2014.6854292.

Peharz, R., Lang, S., Vergari, A., Stelzner, K., Molina, A., Trapp, M., et al. (2020). Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International conference on machine learning* (pp. 7563–7574). PMLR, URL: http://proceedings.mlr.press/v119/peharz20a.html.

Peharz, R., Tschiatschek, S., Pernkopf, F., & Domingos, P. (2015). On theoretical properties of sum–product networks. In *Artificial intelligence and statistics* (pp. 744–752). PMLR, URL: http://proceedings.mlr.press/v38/peharz15.html.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., et al. (2020). Random sum–product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in artificial intelligence* (pp. 334–344). PMLR, URL: http://proceedings.mlr.press/v115/peharz20a.html.

Poon, H., & Domingos, P. (2011). Sum–product networks: A new deep architecture. In *2011 IEEE international conference on computer vision workshops (ICCV Workshops)* (pp. 689–690). IEEE, http://dx.doi.org/10.1109/ICCVW.2011.6130310.

Pronobis, A., & Rao, R. P. (2017). Learning deep generative spatial models for mobile robots. In *2017 IEEE/RSJ international conference on intelligent robots and systems* (pp. 755–762). IEEE, http://dx.doi.org/10.1109/IROS.2017.8202235.

Rahman, T., & Gogate, V. (2016). Merging strategies for sum–product networks: From trees to graphs. In *Proceedings of the thirty-second conference on uncertainty in artificial intelligence* (pp. 617–626). AUAI Press, URL: http://auai.org/uai2016/proceedings/papers/71.pdf.

Rao, R., Olshausen, B. A., & Lewicki, M. S. (2002). Probabilistic models of the brain: Perception and neural function. *Journal of Mathematical Psychology*.

Rashwan, A., Poupart, P., & Zhitang, C. (2018). Discriminative training of sum–product networks by extended baum-welch. In *International conference on probabilistic graphical models* (pp. 356–367). PMLR, URL: http://proceedings.mlr.press/v72/rashwan18a.html.

Rashwan, A., Zhao, H., & Poupart, P. (2016). Online and distributed bayesian moment matching for parameter learning in sum–product networks. In *Artificial intelligence and statistics* (pp. 1469–1477). PMLR, URL: http://proceedings.mlr.press/v51/rashwan16.html.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Rooshenas, A., & Lowd, D. (2014). Learning sum–product networks with direct and indirect variable interactions. In *International conference on machine learning* (pp. 710–718). PMLR, URL: http://proceedings.mlr.press/v32/rooshenas14.html.

Samaria, F. S., & Harter, A. C. (1994). Parameterisation of a stochastic model for human face identification. In *Proceedings of 1994 IEEE workshop on applications of computer vision* (pp. 138–142). IEEE, http://dx.doi.org/10.1109/ACV.1994.341300.

Sánchez-Cauce, R., París, I., & Díez, F. J. (2021). Sum–product networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*, 3821–3839. http://dx.doi.org/10.1109/TPAMI.2021.3061898.

Sang, T., Beame, P., & Kautz, H. A. (2005). Performing bayesian inference by weighted model counting. In *The twentieth national conference on artificial intelligence and the seventeenth innovative applications of artificial intelligence conference* (pp. 475–481). URL: http://www.aaai.org/Library/AAAI/2005/aaai05-075.php.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*, 461–464, URL: https://www.jstor.org/stable/2958889.

Shih, A., Sadigh, D., & Ermon, S. (2021). Hyperspns: Compact and expressive probabilistic circuits. In *Advances in neural information processing systems(NIPS), Vol. 34* (pp. 8571–8582). Curran Associates, Inc, URL: https://proceedings.neurips.cc/paper/2021/hash/481fbfa59da2581098e841b7afc122f1-Abstract.html.

Singla, P., & Domingos, P. M. (2008). Lifted first-order belief propagation. In *Proceedings of the twenty-third AAAI conference on artificial intelligence* (pp. 1094–1099). AAAI Press, URL: http://www.aaai.org/Library/AAAI/2008/aaai08-173.php.

Skryagin, A., Stammer, W., Ochs, D., Dhami, D. S., & Kersting, K. (2022). Neural-probabilistic answer set programming. In G. Kern-Isberner, G. Lakemeyer, & T. Meyer (Eds.), *Proceedings of the 19th international conference on principles of knowledge representation and reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022* (p. 1). URL: https://proceedings.kr.org/2022/48/.

Skryagin, A., Stelzner, K., Molina, A., Ventola, F. G., Yu, Z., & Kersting, K. (2020). Sum–product logic: Integrating probabilistic circuits into deepproblog. In *Working notes of the ICML 2020 workshop on bridge between perception and reasoning: graph neural networks and beyond* (p. 1). URL: https://www.aiml.informatik.tu-darmstadt.de/papers/skryagin2020bbpr_spl.pdf.

Smit, A. P. (2020). *Merging deep neural networks and probabilistic models using sum product networks* (Ph.D. thesis), Stellenbosch: Stellenbosch University.

Sontag, D., Globerson, A., & Jaakkola, T. (2011). *Introduction to dual composition for inference* (pp. 1–37). MIT Press, (Chapter 1).

Staal, J., Abràmoff, M., Viergever, M. A., & van Ginneken, B. (2004). Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, *23*, 501–509. http://dx.doi.org/10.1109/TMI.2004.825627.

Stelzner, K., Peharz, R., & Kersting, K. (2019). Faster attend-infer-repeat with tractable probabilistic models. In K. Chaudhuri, & R. Salakhutdinov (Eds.), *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 2019* (pp. 5966–5975). Long Beach, California, USA: PMLR, URL: http://proceedings.mlr.press/v97/stelzner19a.html.

Tang, L., & Liu, H. (2009a). Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 817–826). ACM, http://dx.doi.org/10.1145/1557019.1557109.

Tang, L., & Liu, H. (2009b). Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on information and knowledge management* (pp. 1107–1116). ACM, http://dx.doi.org/10.1145/1645953.1646094.

Trapp, M., Madl, T., Peharz, R., Pernkopf, F., & Trappl, R. (2017). Safe semi-supervised learning of sum–product networks. In *Proceedings of the thirty-third conference on uncertainty in artificial intelligence, UAI* (p. 1). AUAI Press, URL: http://auai.org/uai2017/proceedings/papers/108.pdf.

Trapp, M., Peharz, R., Ge, H., Pernkopf, F., & Ghahramani, Z. (2019). Bayesian learning of sum–product networks. In *Advances in neural information processing systems* (pp. 6344–6355). URL: https://proceedings.neurips.cc/paper/2019/hash/5421e013565f7f1afa0cfe8ad87a99ab-Abstract.html.

Trapp, M., Peharz, R., Skowron, M., Madl, T., Pernkopf, F., & Trappl, R. (2016). Structure inference in sum–product networks using infinite sum–product trees. In *NIPS workshop on practical bayesian nonparametrics* (p. 1). Curran Associates, Inc, URL: https://www2.spsc.tugraz.at/www-archive/downloads/BNPNIPS_2016_paper_9.pdf.

van de Wolfshaar, J., & Pronobis, A. (2019). Deep generalized convolutional sum–product networks for probabilistic image representations. (p. 1). arXiv abs/1902.06155. URL: http://arxiv.org/abs/1902.06155, arXiv:1902.06155.

Van Haaren, J., & Davis, J. (2012). Markov network structure learning: A randomized feature generation approach. In *Twenty-sixth AAAI conference on artificial intelligence* (p. 1). AAAI Press, URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5107.

Ventola, F., Stelzner, K., Molina, A., & Kersting, K. (2020). Residual sum–product networks. In *International conference on probabilistic graphical models* (pp. 545–556). PMLR, URL: http://proceedings.mlr.press/v138/ventola20a.html.

Vergari, A., Choi, Y., Peharz, R., & Van den Broeck, G. (2020). Probabilistic circuits: Representations, inference, learning and applications. In *Tutorial at the the 34th AAAI conference on artificial intelligence* (p. 1). URL: http://starai.cs.ucla.edu/slides/AAAI20.pdf.

Vergari, A., Mauro, N. D., & Esposito, F. (2015). Simplifying, regularizing and strengthening sum–product network structure learning. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 343–358). Springer, http://dx.doi.org/10.1007/978-3-319-23525-7_21.

Vergari, A., Molina, A., Peharz, R., Ghahramani, Z., Kersting, K., & Valera, I. (2019). Automatic bayesian density analysis. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 5207–5215). http://dx.doi.org/10.1609/aaai.v33i01.33015207.

Vergari, A., Peharz, R., Mauro, N. D., & Esposito, F. (2017). Encoding and decoding representations with sum- and max-product networks. In *5th international conference on learning representations, ICLR 2017, Toulon, France, April (2017) 24-26, workshop track proceedings* (p. 1). OpenReview.net, URL: https://openreview.net/forum?id=rkndY2VYx.

Wang, J., & Wang, G. (2018). Hierarchical spatial sum–product networks for action recognition in still images. *IEEE Transactions on Circuits Systems and Video Technology*, *28*, 90–100. http://dx.doi.org/10.1109/TCSVT.2016.2586853.

Wang, W., & Yang, Y. (2022). Towards data-and knowledge-driven artificial intelligence: A survey on neuro-symbolic computing. CoRR abs/2210.15889 URL: https://doi.org/10.48550/arXiv.2210.15889, arXiv:2210.15889.

Xia, R., Zhang, Y., Liu, X., & Yang, B. (2023). A survey of sum–product networks structural learning. In *WWW '23: The ACM web conference 2023, Austin, TX, USA, May (2023) 1–5* (p. 1). ACM, http://dx.doi.org/10.1145/3543507.3583517.

Xiao, P., Du, B., Yun, S., Li, X., Zhang, Y., & Wu, J. (2018). Probabilistic graph embedding for unsupervised domain adaptation. In *24th international conference on pattern recognition, ICPR 2018, Beijing, China, August (2018) 20-24* (pp. 1283–1288). IEEE Computer Society, http://dx.doi.org/10.1109/ICPR.2018.8546005.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. (p. 1). arXiv abs/1708.07747. URL: http://arxiv.org/abs/1708.07747, arXiv:1708.07747.

Yu, Z., Dhami, D. S., & Kersting, K. (2021). Sum–product-attention networks: Leveraging self-attention in probabilistic circuits. (p. 1). arXiv abs/2109.06587. URL: https://arxiv.org/abs/2109.06587, arXiv:2109.06587.

Yu, Z., Ventola, F. G., & Kersting, K. (2021). Whittle networks: A deep likelihood model for time series. In M. Meila, & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 2021, virtual event* (pp. 12177–12186). PMLR, URL: http://proceedings.mlr.press/v139/yu21c.html.

Yueming (2018). Imdb 5000 movie dataset. https://www.kaggle.com/carolzhangdc/imdb-5000-movie-dataset.

Zečević, M., Dhami, D., Karanam, A., Natarajan, S., & Kersting, K. (2021). Interventional sum–product networks: Causal inference with tractable probabilistic models. In *Advances in neural information processing systems(NIPS), Vol. 34* (pp. 15019–15031). Curran Associates, Inc, URL: https://proceedings.neurips.cc/paper/2021/hash/7eb7eabbe9bd03c2fc99881d04da9cbd-Abstract.html.

Zhang, N. L., & Poole, D. (1994). A simple approach to bayesian network computations. In *Proc. of the tenth canadian conference on artificial intelligence* (p. 1). URL: https://hdl.handle.net/1783.1/757.

Zhao, H., Adel, T., Gordon, G., & Amos, B. (2016). Collapsed variational inference for sum–product networks. In *International conference on machine learning* (pp. 1310–1318). PMLR, URL: http://proceedings.mlr.press/v48/zhaoa16.html.

Zhao, H., Poupart, P., & Gordon, G. J. (2016). A unified approach for learning the parameters of sum–product networks. In *Advances in neural information processing systems* (pp. 433–441). Curran Associates, Inc., URL: https://proceedings.neurips.cc/paper/2016/hash/6c9882bbac1c7093bd25041881277658-Abstract.html.

Zheng, K., Pronobis, A., & Rao, R. (2018). Learning graph-structured sum–product networks for probabilistic semantic maps. In *Proceedings of the AAAI conference on artificial intelligence* (pp. 4547–4555). URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16923.

Zöller, M., & Huber, M. F. (2021). Benchmark and survey of automated machine learning frameworks. *Journal of the Artificial Intelligence Research*, *70*, 409–472. http://dx.doi.org/10.1613/jair.1.11854.