Online Incremental Structure Learning of Sum–Product Networks

Sang-Woo Lee, Min-Oh Heo, and Byoung-Tak Zhang

School of Computer Science and Engineering,
Seoul National University,
1 Gwanak-ro, Gwanak-gu, Seoul 151-744, Korea
{slee,moheo,btzhang}@bi.snu.ac.kr

Abstract. Sum–product networks (SPNs) are deep architectures that can learn and infer at low computational costs. The structure of SPNs is especially important for their performance; however, structure learning for SPNs has until now been introduced only for batch-type dataset. In this study, we propose a new online incremental structure learning method for SPNs. We note that SPNs can be represented by mixtures of basis distributions. Online learning of SPNs can be formulated as an online clustering problem, in which a local assigning instance corresponds to modifying the tree-structure of the SPN incrementally. In the method, the number of hidden units and even layers are evolved dynamically on incoming data. The experimental results show that the proposed method outperforms the online version of the previous method. In addition, it achieves the performance of batch structure learning.

Keywords: sum–product networks, structure learning, online learning, incremental learning, deep architecture, probabilistic graphical model.

1 Introduction

As learning the structure of graphical models is one of the most important issues in machine learning fields, many researchers have contributed to its study. Noteworthy examples of these research studies are Bayesian networks [1], Markov networks [2], deep networks [3], and sum–product networks (SPNs) [4].

Studies on online learning, however, are limited because it is not easy to change the form of probability tables without information about forgotten training data. Nevertheless, online learning is an essential problem in machine learning, and there are some learning environments in which it should be applied, such as large-scale data learning or lifelong learning. In a successful study, a single-layer denoising autoencoder was learned using online incremental structure learning [5]. It had been verified that this model learns the changing probability distribution of data. They also argued the possibility of extension to multi-layer models.

In order to solve the online learning problem using enough representation power, however, we focus on the SPN, a hierarchical architecture that includes sum nodes, product nodes, and univariate nodes [6]. In recent research on SPNs [4], Gens and Domingos used hierarchical biclustering to learn the structure of SPNs well.

The framework that they used is, however, not suitable for online learning because entire data should be used in the structure learning steps. Toward making an incremental model, we first note that structure learning of SPNs is highly dependent on clustering instances. Using this perspective, we convert the online structure learning problem to an online clustering problem. We propose a simple mini-batch clustering algorithm which can modify the number of clusters dynamically on incoming data, and apply it to incremental structure learning. The experiments show that it outperforms the online version of the previous method [4], and achieves the performance of batch structure learning.

The remainder of the paper is organized as follows: In section 2, we introduce a brief definition of SPNs, and a previous study on structure learning. In section 3, we suggest online incremental structure learning methods. In section 4, we show the experimental results of applying online incremental learning methods, and conclude this paper in section 5.

2 Sum–Product Networks

2.1 Representation of SPNs

SPNs are one of probabilistic graphical models (PGMs) which have specialized structure for fast inference. They are constrained to have tree structures (rooted directed acyclic graphs), and their leaves represent univariate distribution such as multinomial distribution for discrete variables, Gaussian, Poisson and other continuous distributions. Internal nodes in the tree represent products or sums of their children with the corresponding weights as the Figure 1. Recursive definition of SPNs introduced in [4] is as follows.

Definition 1. An SPN is defined as follows:

1. A tractable univariate distribution is an SPN.
2. A product of SPNs with disjoint scopes is an SPN.
3. A weighted sum of SPNs with the same scope is an SPN, provided all weights are positive.
4. Nothing else is an SPN.

The scope of an SPN is defined as the set of variables that appear in it. The sub-SPN S_i is a sub-tree at a node *i* as a root and corresponds to a probability distribution over its scope.

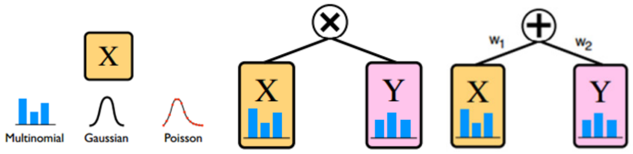


Fig. 1. Types of nodes in SPN: Univariate node (left); Product node (middle); Sum node (right)

Let $x_1, \ldots, x_n$ are variables in an SPN S and $x = (x_1, \ldots, x_n)$ be a state in their possible world. $S_i[x]$ be the probability of the i-th node of the state x, and $S_0[x]$ be the probability represented by the root node (the *0*-th node). Considering the weights, the recursive relationship among $S_i[x]$ of arbitrary nodes is as follows:

$$S_i[x^{(i)}] = \begin{cases} \sum_{j \in ch(i)} w_{ij} S_j[x^{(j)}] & , \textit{ if } S_i \textit{ is sum node} \\ \prod_{j \in ch(i)} S_j[x^{(j)}] & , \textit{ if } S_i \textit{ is product node} \\ c_i P_i(x^{(i)}) & , \textit{ if } S_i \textit{ is univariate node} \end{cases} \quad (1)$$

where $x^{(i)}$ represents the input variables that appear in a sub-SPN S_i, *ch*(i) is the index set of child nodes of node i, and P_i is a generic univariate distribution of the variable, which $x^{(i)}$ contains only one variable. The joint probability distribution of an SPN *S* is $P(x) = S_0[x]$ if the weights at each sum node sum to one.

Many previous works related to SPNs, including sigma-pi neural networks [7], arithmetic circuits [8], and other compact representations exist. However, SPNs are a more general probabilistic model that enjoys enough representation power. SPNs show a remarkable performance in image classification tasks [9] and video learning [10].

2.2 Structure Learning of SPNs

In the seminal work of SPNs [6], the structure of SPNs was built in the application-oriented manner. After that, structure learning for SPNs has been introduced in [4, 11]. Dennis & Ventura [11] collected variables using regional relationship to find better structure to solve the image completion task. Gens & Domingos [4] firstly proposed batch-type structure learning method by splitting the variables into mutually independent subsets toward compact representation of joint distribution minimizing the representation power loss.

As a brief introduction, structure learning in [4] is a recursive procedure given dataset *T* and set of variables *V*. At first, it checks whether the variables can be split into mutually independent subsets. If possible, the split recursions are done for each subset, and return the products of the resulting SPNs (building internal product nodes). Otherwise, the instances *T* are clustered into similar subsets, and it returns the weighted sum of the resulting SPNs (building internal sum nodes). Weights for child nodes are determined to be proportional to the number of the assigned instances. At the end of the recursive process, all leaves consist of a univariable node.

At the clustering process, they use a naïve Bayes mixture model to pick most likely component in the mixture, where all variables are independent conditioned on the cluster. They use a hard expectation-maximization (hard EM) algorithm, where each instance is wholly assigned to only its most probable cluster in expectation.

3 Online Incremental Structure Learning Methods

Online structure learning of PGMs is a challenging task because there is no guarantee that new incoming instances follow the learnt model. If new instances are not explained with the structure, it should be modified. Fortunately, in the case of SPNs, online structure learning is deeply related to hard clustering, so we use this property.

We first suggest simple mini-batch incremental clustering problem in Algorithm 1. In Algorithm 1, new instances are assigned to one of the existing clusters. New clusters are added with new instances if they are needed. To find an appropriate number of clusters *k*, we increase the number of clusters one by one until likelihood does not increase any more than threshold.

```
Algorithm 1. IncrementalClustering($T$,$V$,$M$)
input: set of mini-batch instances $T$, set of variables $V$,
a cluster model $M$
output: sets of instances assigned to the existing cluster
{$T_i$}, sets of instances assigned to the new cluster {$T_j$}
likelihood = -inf
while true
  while model $M$ is converge
    assign $T$ with variables $V$ and model $M$
    update model $M$ with $T$
  end
  calculate likelihood
  if increase of likelihood is less than threshold
    break
  end
  add a new cluster to the model $M$
end
```

We can extend the above clustering process to learn the structure of SPNs. Algorithm 2 illustrates online incremental learning algorithm for SPNs. The algorithm hierarchically adds new child nodes onto the sum nodes in whole layers. The clustering process is used in algorithm 2 as one part. It basically uses the distributions of child nodes. If there is no model for applying to new cluster, however, it also use naïve Bayes model, as the structure learning on the previous study does. After clustering, existing child nodes are augmented recursively, whereas new child nodes are constructed by previous structure learning methods. Our learning method is hybrid of methods in parameter learning [6] and structure learning [4] as illustrated in Figure 2. This method also use hard EM of SPNs as previous methods does. Previous studies show this hierarchical hard clustering strategy is powerful in practice and verify our new learning method is valid.

If the model explains new data well, the structure isn't changed much. Otherwise, the SPN increases their nodes to express new data. The method makes models learn different tree-structure of SPNs if the incoming order of data flow is shuffled with the same data stream.

```
Algorithm 2. AugmentSPN($T$,$V$,$M$)
input: set of mini-batch instances $T$, set of variable $V$,
and an SPN $M$
output: an augmented cluster model $M'$
if root of $M$ is univariate node
  $M$ = ParameterLearnNode($T$,$V$,$M$)
elseif root of $M$ is product node
  for each child node $M_k$ of root
    $M_k$ = AugmentSPN($T$,$V_k$, $M_k$)
  end
elseif root of $M$ is sum node
  ($\{T_i\}$,$\{T_j\}$) = IncrementalClustering($T$,$V$,$M$)
  for each instance sets of $i$-th existing cluster
    $M_i$ = ParameterLearnNode($T_i$, $V$, $M_i$)
    $M_i$ = AugmentSPN($T_i$, $V$, $M_i$)
  end
  for each instance sets of $j$-th new cluster
    $M_j$ = StructureLearnSPN($T_j$, $V$)
  end
end
```

Note that, on the other hands, the variable subsets split by the product node will not change after the product node is generated once. If they are not mutual independent on new data, the product node cannot fully explain them. It may cause the model heavier with meaningless product nodes. It is a major limitation of our method.

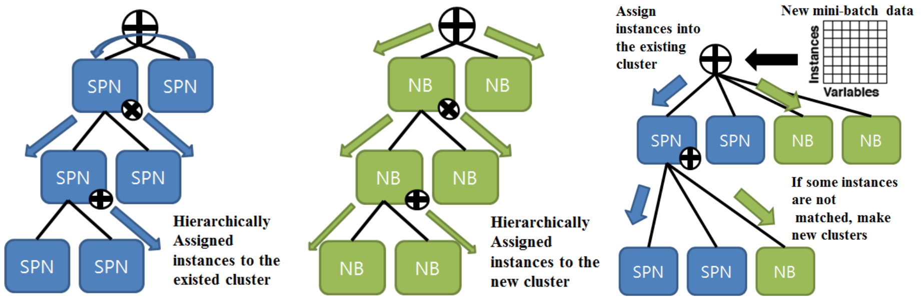

SPN
SPN
SPN
SPN
SPN
SPN
Hierarchically Assigned instances to the existed cluster
NB
NB
NB
NB
NB
NB
Hierarchically Assigned instances to the new cluster
Assign instances into the existing cluster
New mini-batch data
Instances
Variables
SPN
SPN
NB
NB
If some instances are not matched, make new clusters
SPN
SPN
NB

Fig. 2. Parameter learning [6] (left); Structure learning [4] (middle); Online incremental structure learning of SPNs (right). Online structure learning method is hybrid of methods in parameter learning and structure learning. Arrows indicate flows of assigning instances to the node.

4 Experiments

We evaluated online incremental structure learning methods on variants of the "handwritten optic digits" [12] to illustrate our argument. These digit data include an image pixel and a digit class. However, pixels are binarized for our experiments. We make two settings: “homo” and “hetero.” In the hetero setting, we only reorder the dataset according to the class label. In other words, the models first meet all the digit0 images, then the digit1 images, and so on. The homo setting uses only the dataset’s own order, and it yields a stable probability distribution of data. The hetero setting, however, yields a dynamically changing probability distribution of data in the mini-batch task.

We evaluated the suggested model in a mini-batch environment. The number of mini-batches was 16. Three models were compared. The first is “classical online learning”, which uses only the first mini-batch for structure learning which following previous research studies. The second is “online ensemble learning” which ensembles 16 SPNs constructed by “classical online learning” method. “Classical online learning” methods, It is possible that more complex model may do better, which is why this second model used in the comparison. Our models are quite larger than the first comparable model, but slightly smaller than the second model. The third model is “batch learning,” which uses the whole dataset for structure learning and is exactly same as the classical methods used in previous studies.

We tested two methods for evaluating performances, one of which measures likelihood. The other goal is to infer the probability of a subset of the variables (the query) given the values of another (the evidence). We used 50% of the variables as the query and 30% of the variables as the evidence in the experiments.

Figure 3 shows the performance results of the various learning methods as a hetero handwritten dataset arrives. The suggested model not only outperforms naïve online models, but also achieves the performances of batch structure learning. The results imply that the suggested learning method represents well the probability distribution of the data. We also catch that “online ensemble learning” method do better than “classical online learning” methods, which means that previous studies may not have fully tuned their own model.

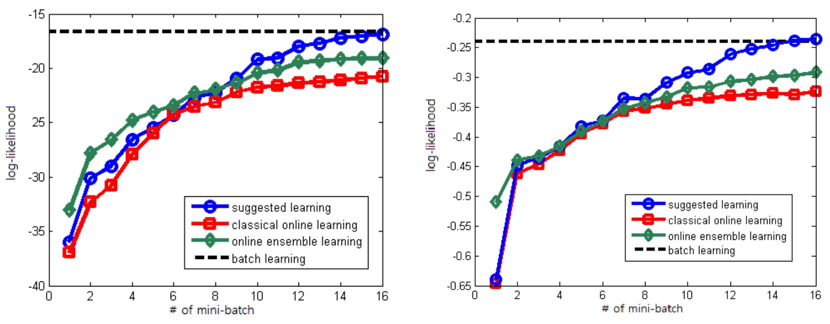


Fig. 3. Log-likelihoods (left). Average conditional log-likelihoods for arbitrary query and evidence (right).

We also investigated the form of the structure of changing SPNs. According to different characteristics of the order in which the data arrive, the structure changes differently. First, the complexities of SPNs are different. Figure 4 shows that, in the hetero setting, the models need more nodes to represent the probability distribution as new data arrive. Second, the structure of SPNs or the numbers of child nodes of a root differ. Figure 5 shows that the different sequence-order of data evolves different forms of structure. Figure 5 (a, c) shows the results of the first mini-batch step, and Figure 5 (b, d) shows the result of the final mini-batch step. Figure 5 (a-b) shows the structure change of the homo setting when distribution of data is balanced. When data with similar distribution arrive, SPNs increase their depth. Figure 5 (c-d) shows the structure change of the hetero setting when the order of data is according to the class. When data with dynamically changing distribution arrive, SPNs increase their width.

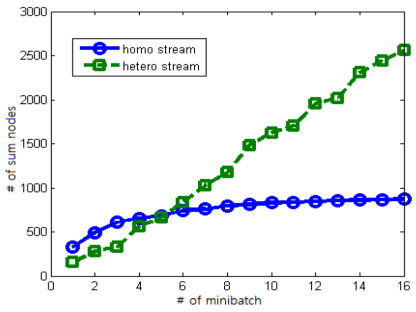
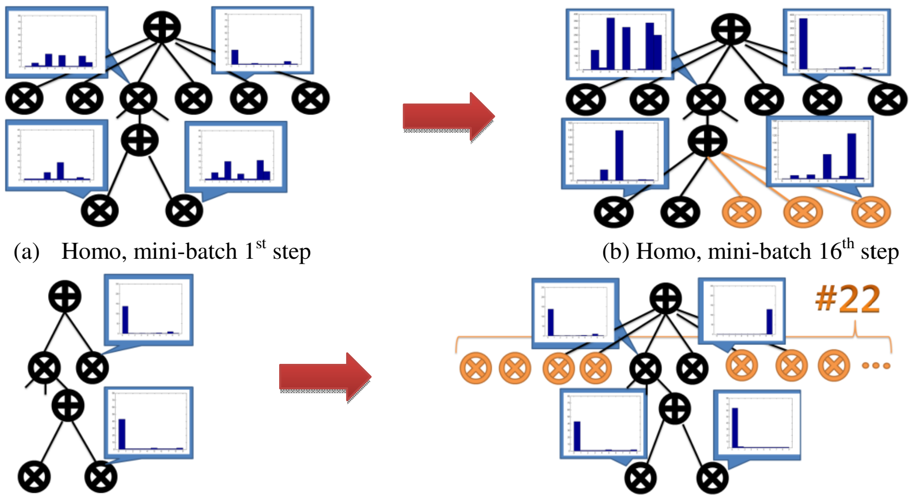


Fig. 4. Growth of complexity. If incoming data are similar to the distribution of the model, the complexity of the model converges. Otherwise, the complexity of the model increases.



(a) Homo, mini-batch 1st step

(b) Homo, mini-batch 16th step

(c) Hetero, mini-batch 1st step

(d) Hetero, mini-batch 16th step

Fig. 5. Different structures evolve according to different orders of datasets

5 Conclusion

In this paper, an online incremental learning method for SPNs was suggested and their performances were tested. In our method, the number of hidden units and even layers are evolved dynamically on incoming data to follow the changing distribution of data.

This paper is also a source of motivation for some future studies. First, our methods should be verified on larger-scale data as online learning tasks may be needed in large-scale data environments. Second, the concept of incremental learning can also be used in batch learning with non-parametric Bayesian methods. By applying a non-parametric Bayesian technique, we could more plausibly learn the structure of a Bayesian network. We hope this property can be used for lifelong learning to allow the model to catch new concepts and concept drift [13].

Acknowledgements. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2010-0017734, Videome), supported in part by KEIT grant funded by the Korea government (MKE) (10035348, mLife).

References

1. Chickering, D.M., Heckerman, D., Meek, C.: A Bayesian approach to learning Bayesian networks with local structure. In: Conference on Uncertainty in Artificial Intelligence, vol. 13 (1997)
2. Ravikumar, P., Wainwright, M.J., Lafferty, J.D.: High-dimensional ising model selection using L1-regularized logistic regression. The Annals of Statistics 38(3), 1287–1319 (2010)
3. Bengio, Y.: Learning deep architectures for AI. Foundation and Trends in Machine Learning 2(1), 1–127 (2009)
4. Gens, R., Domingos, P.: Learning the structure of sum-product networks. In: International Conference on Machine Learning, vol. 30 (2013)
5. Zhou, G., Sohn, K., Lee, H.: Online incremental feature learning with denoising autoencoder. In: International Conference on Artificial Intelligence and Statistics, vol. 15 (2012)
6. Poon, H., Domingos, P.: Sum-product networks: A new deep architecture. Uncertainty in Artificial Intelligence 27 (2011)
7. Zhang, B.-T., Muhelnbein, H.: Synthesis of sigma-pi networks by the breeder genetic programming. In: Proceedings of First IEEE Conference on Evolutionary Computation (1994)
8. Darwiche, A.: A differential approach to inference in Bayesian networks. In: Conference on Uncertainty in Artificial Intelligence, vol. 16 (2003)
9. Gens, R., Domingos, P.: Discriminative learning of sum-product networks. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
10. Amer, M.R., Todorovic, S.: Sum-product networks for modeling activities with stochastic structure. In: Computer Vision and Pattern Recognition (2012)
11. Dennis, A., Ventura, D.: Learning the architecture of Sum-Product networks using Clustering on variable. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
12. UCL Machine Learning Depository, `http://archive.ics.uci.edu/ml/`
13. Zhang, B.-T., Ha, J.-W., Kang, M.: Sparse population code models of word learning in concept drift. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 34 (2012)