Deep learning

Sum Product Networks

Hamid Beigy

Sharif university of technology

December 23, 2019

Table of contents

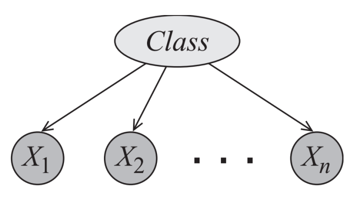
Introduction

1. We assume $x_1, x_2, \ldots, x_m$ are IID random variables are sampled from an unknown distribution $\mathcal{D}$, where each x_i is k-dimensional vector.
2. We require to specify a high-dimensional distribution $p(x_1, \ldots, x_k)$ on the data and possibly some latent variables.
3. The specific form of p will depend on some parameters w.
4. The basic operations will be to
   - **Structure learning:** Specifying the parametric/non-parametric form of $p(x_1, \ldots, x_k)$.
   - **Parameter learning:** Adjusting $p(x_1, \ldots, x_k)$ to the data.
   - **Inference:** Computing marginals and modes of $p(x_1, \ldots, x_k)$.
5. Working with fully flexible joint distributions is intractable!

Structure learning

1. How the form of density function is specified?
2. We specify the form of density function in such a way that parameter learning and inference become easier.
3. For example, we can consider the following conditional form.

$$p(x_1, \dots, x_k) = p(x_1|x_2)p(x_1|x_3)\dots p(x_1|x_k)$$

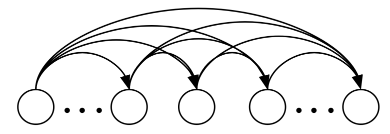


Structure learning

1. Or consider the following forms

$$p(x_1, \ldots, x_k) = p(x_k|x_{k-1})p(x_{k-1}|x_{k-2}) \ldots p(x_2|x_1)$$

$$p(x_1, \ldots, x_k) = \prod_{i=1}^{k} p(x_i|x_1, x_2, \ldots, x_{i-1})$$

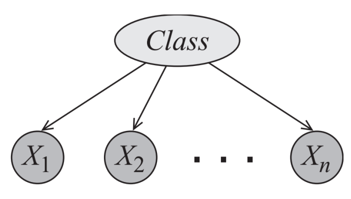


2. We must work with structured or compact distributions.
   For example, distributions in which the random variables interact directly with only very few others in simple ways (why?).
3. One solution is to use probabilistic graphical models.

Some Inference queries

1. **Simple queries:** computing posterior marginal $p(x_1|E = e)$
2. **Conjunctive queries:** Computing $p(x_1, x_2|E = e)$
3. How do you answer the following query?
   $p(x_1) = \sum_{x_2} \sum_{x_3} p(x_1, x_2, x_3)$
4. How do you answer the query $p(x_1)$ when density function has the following form?

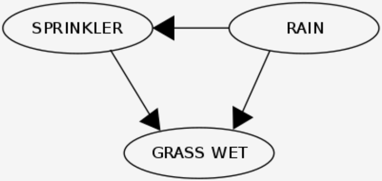
$$p(x_1, \ldots, x_k) = p(x_1|x_2)p(x_1|x_3) \ldots p(x_1|x_k)$$



Bayesian Networks

1. A simple Bayesian network

RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99



RAIN T	RAIN F
0.2	0.8

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$$p(G, S, R) = p(G|S, R)p(S|R)p(R)$$

Bayesian Networks

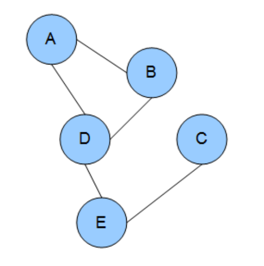
1. How calculate $p(x_1, \ldots, x_k)$ using Bayesian networks?
2. If a Bayesian network can be factorized, then we can write

$$p(x_1, \ldots, x_k) = \prod_{v \in V} p(x_v | pa(v))$$

   where $pa(v)$ is the set of parents of v.
3. Cooper proved that exact inference in Bayesian networks is NP-hard.

Markov networks

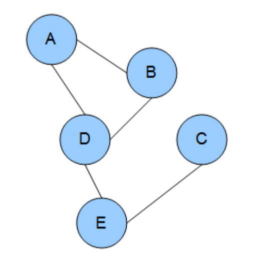
1. A Markov network is a set of random variables having a Markov property described by an undirected graph.



2. Each edge represents dependency.
   - A depends on B and D.
   - B depends on A and D.
   - D depends on A, B, and E.
   - E depends on D and C.
   - C depends on E.

Markov networks

1. Consider the following network.



2. Well assume p is a general undirected model of the following form

$$p(x_1, \ldots, x_n; w) = \frac{\bar{p}(x_1, \ldots, x_n; w)}{Z(w)} = \frac{1}{Z(w)} \prod_k \phi_k(x_{\{}k\}; w),$$

where the ϕ_k are the factors and $Z(w)$ is the normalization constant and $x_{\{}k\}$ is a subset of variables .

3. How do yo compute $Z(w)$?

Limitations of Graphical Models

1. Graphical models are limited in some aspects
   - Many compact distributions cannot be represented as a GM.
   - The cost of exact inference in GM is exponential in the worst case (using approximate techniques).
   - Because learning requires inference, learning GM will be difficult .
   - Some distributions require GM with many layers of hidden variables to be compactly encoded.
2. An alternative are sum product networks[1]
   - New deep model with many layers of hidden variables.
   - Exact inference is tractable (linear in the size of the model).

[1] Poon, Hoifung and Pedro Domingos (2011). Sum-Product Networks: a New Deep Architecture. UAI 2011

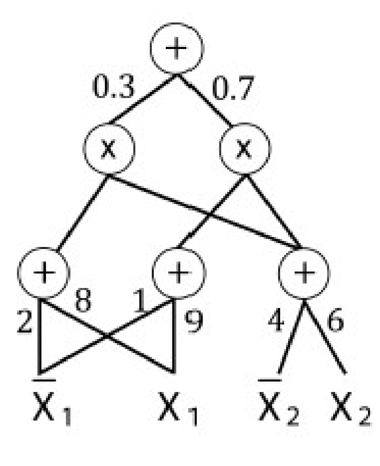
Table of contents

Sum Product Networks

1. A SPN is rooted DAG whose leaves are $x_1, \ldots, x_n$ and $\bar{x}_1, \ldots, \bar{x}_n$ with internal sum and product nodes, where each edge (i, j) emanating from sum node i has a weight $w_{ij} \geq 0$.
2. The value of a product node is the product of the value of its children.
3. The value of a sum node i is $\sum_{j \in Ch(i)} w_{ij} v_j$, where $Ch(j)$ are the children of node i and v_j is the value of node j
4. The value of a SPN is the value of the root after a bottom up evaluation.
5. Layers of sum and product nodes usually alternate.

Sum Product Networks (example)

1. An example of SPN

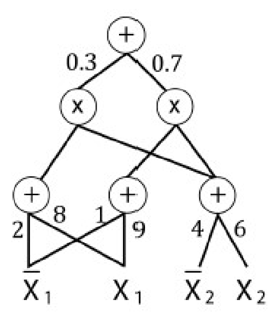


2. What is the output of the above network?

Probabilistic Inference

SPN represents a joint distribution over a set of random variables.

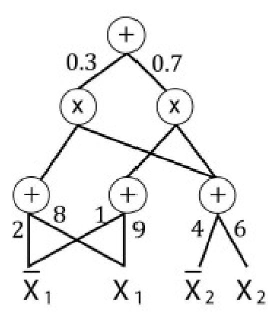
As an example consider
$p(x_1 = true, x_2 = false)$



Marginal Inference

SPN represents a joint distribution over a set of random variables.

As an example consider
$p(x_2 = false)$



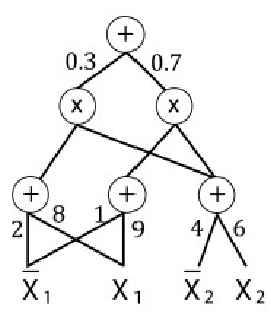
Conditional Inference

SPN represents a joint distribution over a set of random variables.

- As an example consider

$$p(x_1 = true | x_2 = false) = \frac{p(x_1 = true, x_2 = false)}{p(x_2 = false)}$$

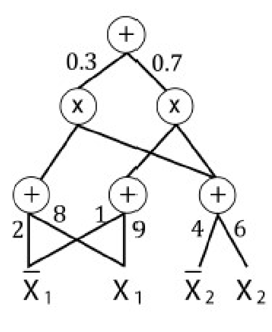
- Hence any inference query can be answered in two bottom-up passes of the network (Linear time complexity).



SPN Semantics

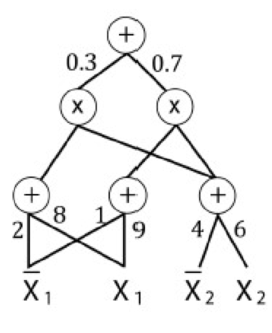
A valid SPN encodes a hierarchical mixture distribution.

- Sum nodes: hidden variables (mixture)
- Product nodes: factorization (independence)



Valid SPN

- The scope of a node is the set of variables that appear in the sub-SPN rooted at the node
- An SPN is decomposable when each product node has children with disjoint scopes.
- An SPN is complete when each sum node has children with identical scopes
- A decomposable and complete SPN is a valid SPN.

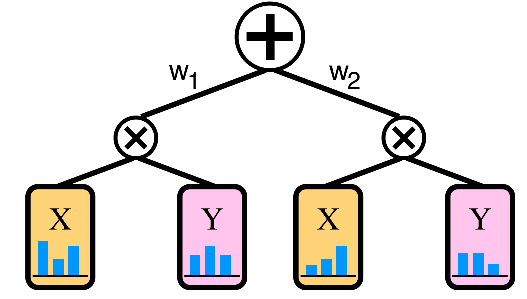


Building and using an SPN

1. We must specify the structure of SPN (structure Estimation or structure learning).
2. We must find the parameters of SPN (parameter learning).
3. We must answer queries (inference).

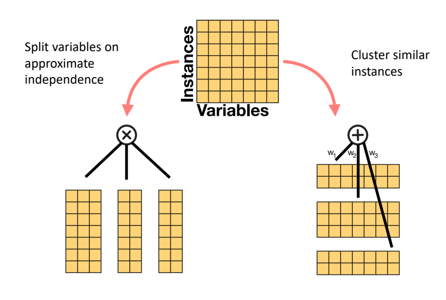
Structure learning

1. What is SPN for univariate distribution?
2. $\rightarrow$ A univariate distribution is an SPN
3. What is SPN for product of disjoint random variables?
4. $\rightarrow$ A product of SPNs over disjoint variables is an SPN.
5. What is SPN for a mixture model?
6. $\rightarrow$ A weighted sum of SPNs over the same variables is an SPN.



Structure learning

1. In a structure learning, one alternates between
   - Data Clustering: sum nodes
   - Variable partitioning: product nodes



2. Some others use SVD decomposition[2]

[2] Adel, Tameem, David Balduzzi, and Ali Ghodsi. Learning the Structure of Sum-Product Networks via an SVD-based Algorithm. UAI 2015.

SPN learning

1. Initialize the SPN using a dense valid SPN.
2. Learn the SPN weights using gradient descent or EM.
3. Add some penalty to the weights so that they tend to be zero.
4. Prune edges with zero weights at convergence.

Algorithm 1 LearnSPN

Input: Set D of instances over variables X.
Output: An SPN with learned structure and parameters.

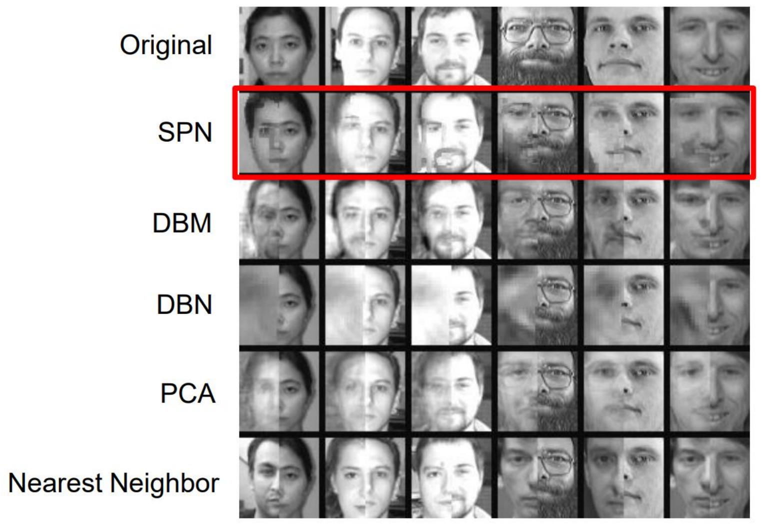
```
S ← GenerateDenseSPN(X)
InitializeWeights(S)
repeat
    for all d ∈ D do
        UpdateWeights(S, Inference(S, d))
    end for
until convergence
S ← PruneZeroWeights(S)
return S
```

Table of contents

Image completion

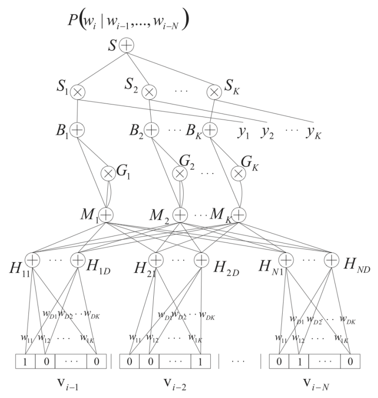
1. Main evaluation: Caltech-101
   - 101 categories, e.g., faces, cars, elephants
   - Each category: 30 – 800 images
2. Also, Olivetti [Samaria & Harter, 1994] (400 faces)
3. Each category: Last third for test
4. Test images: Unseen objects

Image completion



Language modeling

1. Fixed structure SPN encoding the conditional probability $p(w_i|w_{i-1}\ldots,w_{i-n})$ as an nth order language model[3].



[3] Cheng et al., Language Modeling with Sum-Product Networks, InterSpeech, 2014

Language modeling

1. Perplexity scores (PPL) of different language models

Model	Individual PPL	+KN5
TrainingSetFrequency	528.4	
KN5 [3]	141.2	
Log-bilinear model [4]	144.5	115.2
Feedforward neural network [5]	140.2	116.7
Syntactical neural network [8]	131.3	110.0
RNN [6]	124.7	105.7
LDA-augmented RNN [9]	113.7	98.3
SPN-3	**104.2**	**82.0**
SPN-4	**107.6**	**82.4**
SPN-4’	**100.0**	**80.6**

Other applications

1. Image completion
2. Image classification
3. Activity recognition
4. Click-through logs
5. Nucleic acid sequences
6. Collaborative filtering

Advantages of SPNs

1. Unlike graphical models, SPNs are tractable over high treewidth models.
2. SPNs are deep architectures with full probabilistic semantics
3. SPNs can incorporate features into an expressive model without requiring approximate inference.