On the Relationship between Sum-Product Networks and Bayesian Networks

Han Zhao, Mazen Melibari and Pascal Poupart
Presented by: Han Zhao

May 13, 2015
Outline

Background
- Bayesian Network
- Algebraic Decision Diagram
- Sum-Product Network

Main Result
- Main Theorems
- Sum-Product Network to Bayesian Network
- Bayesian Network to Sum-Product Network

Discussion
Bayesian Network

Definition

A graphical representation of a set of random variables $X_{1:N}$ and their conditional dependencies.

- Node corresponds to random variables (observable or latent) and edges represent conditional dependency between pairs of variables.
Bayesian Network

Definition

A graphical representation of a set of random variables $X_{1:N}$ and their conditional dependencies.

- Node corresponds to random variables (observable or latent) and edges represent conditional dependency between pairs of variables.
- Each node is associated with a local conditional probability distribution (CPD).

<table>
<thead>
<tr>
<th>$H_1$</th>
<th>$P_r(H_1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>0.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$H_2$</th>
<th>$P_r(H_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
</tr>
</tbody>
</table>

| $H_1$ | $H_2$ | $X$ | $P_r(X|H_1, H_2)$ |
|-------|-------|-----|-------------------|
| 0     | 0     | 0   | 0.3               |
| 0     | 0     | 1   | 0.7               |
| 0     | 1     | 0   | 0.3               |
| 0     | 1     | 1   | 0.7               |
| 1     | 0     | 0   | 0.3               |
| 1     | 0     | 1   | 0.7               |
| 1     | 1     | 0   | 0.4               |
| 1     | 1     | 1   | 0.6               |
Bayesian Network

Definition

A graphical representation of a set of random variables $X_{1:N}$ and their conditional dependencies.

- Node corresponds to random variables (observable or latent) and edges represent conditional dependency between pairs of variables.
- Each node is associated with a local conditional probability distribution (CPD).
- A directed acyclic graph (DAG).

<table>
<thead>
<tr>
<th>$H_1$</th>
<th>Pr($H_1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.35</td>
</tr>
<tr>
<td>1</td>
<td>0.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$H_2$</th>
<th>Pr($H_2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
</tr>
</tbody>
</table>

| $H_1$ | $H_2$ | $X$ | Pr($X|H_1, H_2$) |
|-------|-------|-----|------------------|
| 0     | 0     | 0   | 0.3              |
| 0     | 0     | 1   | 0.7              |
| 0     | 1     | 0   | 0.3              |
| 0     | 1     | 1   | 0.7              |
| 1     | 0     | 0   | 0.3              |
| 1     | 0     | 1   | 0.7              |
| 1     | 1     | 0   | 0.4              |
| 1     | 1     | 1   | 0.6              |
Bayesian Network

Definition

Local Markov property: each variable is conditionally independent of its non-descendants given its parents.

\[
\Pr(X_{1:N}) = \prod_{i=1}^{N} \Pr(X_i \mid X_{1:i-1}) = \prod_{i=1}^{N} \Pr(X_i \mid \text{Pa}(X_i))
\]
Bayesian Network

Definition

Local Markov property: each variable is conditionally independent of its non-descendants given its parents.

\[
\Pr(X_{1:N}) = \prod_{i=1}^{N} \Pr(X_i \mid X_{1:i-1}) = \prod_{i=1}^{N} \Pr(X_i \mid Pa(X_i))
\]

Independence induced by the structure of BN
A Bayesian Network over 5 random variables:

\[
\Pr(X_{1:5}) = \Pr(X_5|X_{1:4}) \Pr(X_4|X_{1:3}) \Pr(X_3|X_{1:2}) \Pr(X_2|X_1) \Pr(X_1)
\]

= \frac{8}{90}
Bayesian Network

Example

A Bayesian Network over 5 random variables:

\[
\Pr(X_{1:5}) = \Pr(X_5|X_{1:4}) \Pr(X_4|X_{1:3}) \Pr(X_3|X_{1:2}) \Pr(X_2|X_1) \Pr(X_1) \\
= \Pr(X_5|X_3) \Pr(X_4|X_3) \Pr(X_3|X_1, X_2) \Pr(X_2) \Pr(X_1)
\]
Joint, marginal and conditional probabilistic query in Bayesian Network. Consider the marginal query $\Pr(X_N = \text{True})$

$$\Pr(X_N = \text{True}) = \sum_{X_1} \cdots \sum_{X_{N-1}} \Pr(X_1 : N-1, X_N = \text{True})$$
Joint, marginal and conditional probabilistic query in Bayesian Network. Consider the marginal query $\Pr(X_N = \text{True})$

$$\Pr(X_N = \text{True}) = \sum_{X_1} \cdots \sum_{X_{N-1}} \Pr(X_1: N-1, X_N = \text{True})$$

Naive enumeration is exponential in the number of variables
Bayesian Network

Inference

Joint, marginal and conditional probabilistic query in Bayesian Network. Consider the marginal query $\Pr(X_N = \text{True})$

$$
\Pr(X_N = \text{True}) = \sum_{X_1} \cdots \sum_{X_{N-1}} \Pr(X_1:N-1, X_N = \text{True})
$$

Naive enumeration is exponential in the number of variables

Factorization helps to reduce the inference complexity by taking advantage of the distributive law of $\times$ over $+$
Bayesian Network

Inference

Joint, marginal and conditional probabilistic query in Bayesian Network. Consider the marginal query $\Pr(X_N = \text{True})$

$$\Pr(X_N = \text{True}) = \sum_{X_1} \cdots \sum_{X_{N-1}} \Pr(X_1:N-1, X_N = \text{True})$$

Naive enumeration is exponential in the number of variables

Factorization helps to reduce the inference complexity by taking advantage of the distributive law of $\times$ over $+$

$$\Pr(X_5 = \text{True}) = \sum_{X_{1:4}} \Pr(X_{1:4}, X_5 = \text{True})$$
Bayesian Network

Inference

Joint, marginal and conditional probabilistic query in Bayesian Network. Consider the marginal query $\Pr(X_N = \text{True})$

$$\Pr(X_N = \text{True}) = \sum_{X_1} \cdots \sum_{X_{N-1}} \Pr(X_1:N-1, X_N = \text{True})$$

Naive enumeration is exponential in the number of variables
Factorization helps to reduce the inference complexity by taking advantage of the distributive law of $\times$ over $+$

$$\Pr(X_5 = \text{True}) = \sum_{X_{1:4}} \Pr(X_{1:4}, X_5 = \text{True})$$

$$= \sum_{X_{1:4}} \Pr(X_5 = \text{True}|X_3) \Pr(X_4|X_3) \Pr(X_3|X_1, X_2) \Pr(X_2) \Pr(X_1)$$
Bayesian Network

Inference

Joint, marginal and conditional probabilistic query in Bayesian Network. Consider the marginal query \( \Pr(X_N = \text{True}) \)

\[
\Pr(X_N = \text{True}) = \sum_{X_1} \cdots \sum_{X_{N-1}} \Pr(X_1:N-1, X_N = \text{True})
\]

Naive enumeration is exponential in the number of variables
Factorization helps to reduce the inference complexity by taking advantage of the distributive law of \( \times \) over \( + \)

\[
\Pr(X_5 = \text{True}) = \sum_{X_1:4} \Pr(X_1:4, X_5 = \text{True})
\]

\[
= \sum_{X_1:4} \Pr(X_5 = \text{True}|X_3) \Pr(X_4|X_3) \Pr(X_3|X_1, X_2) \Pr(X_2) \Pr(X_1)
\]

\[
= \sum_{X_3} \Pr(X_5 = \text{True}|X_3) \sum_{X_2} \Pr(X_2) \sum_{X_1} \Pr(X_1) \Pr(X_3|X_1, X_2)
\]

\[
\sum_{X_4} \Pr(X_4|X_3)
\]
Bayesian Network

Inference

Exact inference algorithms for Bayesian Networks:

- **Variable Elimination/Sum-Product algorithm**
- Belief Propagation/Message Passing algorithm

General question:

\[
\sum_{X_H \subseteq X} \prod_{n=1}^{N} \Pr(X_n | Pa(X_n))
\]
Bayesian Network

Inference

Exact inference algorithms for Bayesian Networks:

- **Variable Elimination/Sum-Product algorithm**
- **Belief Propagation/Message Passing algorithm**

General question:

\[
\sum_{X_H \subseteq X} \prod_{n=1}^{N} \Pr(X_n \mid Pa(X_n))
\]

All taking advantage of the distributivity of $\times$ over $+$ (can be extended to any semirings)
Bayesian Network

Inference

Exact inference algorithms for Bayesian Networks:

- **Variable Elimination/Sum-Product algorithm**
- **Belief Propagation/Message Passing algorithm**

General question:

\[
\sum_{X_H \subseteq X} \prod_{n=1}^{N} \Pr(X_n | Pa(X_n))
\]

All taking advantage of the distributivity of \( \times \) over \( + \) (can be extended to any semirings)

1. \( \pi \leftarrow \) an ordering of the hidden variables to be eliminated
2. \( \Phi \leftarrow \{T_H \mid H \text{ is a hidden variable}\} \)
3. **for** each hidden variable \( H \) in \( \pi \) **do**
4. \( P \leftarrow \{T_X \mid T_X \text{ includes } H\} \)
5. \( \Phi \leftarrow \Phi \setminus P \cup \{\sum_H \prod_{T \in P} T\} \)
6. **end for**
Algebraic Decision Diagram

Motivation

How to represent the conditional probability distribution (CPD) associated with each variable in Bayesian Network?
Algebraic Decision Diagram

Motivation

How to represent the conditional probability distribution (CPD) associated with each variable in Bayesian Network?

Tabular representation

A real function over 4 boolean variables

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$f(\cdot)$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$f(\cdot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Algebraic Decision Diagram

Motivation

How to represent the conditional probability distribution (CPD) associated with each variable in Bayesian Network?

Tabular representation

A real function over 4 boolean variables

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$f(\cdot)$</th>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$f(\cdot)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Observation: Once $X_1 = 0$, the value of the function is independent of the value taken by $X_2$. 
Algebraic Decision Diagram

Motivation

Let $X$, $Y$ and $Z$ be three random variables.
Algebraic Decision Diagram

Motivation

Let $X$, $Y$ and $Z$ be three random variables.

Independence

$$X \perp Y \Rightarrow \forall x, y \quad \Pr(x, y) = \Pr(x) \Pr(y)$$

Conditional Independence

$$X \perp Y \mid Z \Rightarrow \forall x, y, z \quad \Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z)$$

Context Specific conditional Independence (CSI)

$$X \perp Y \mid Z = z \Rightarrow \exists z \forall x, y \quad \Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z)$$

Both independence and conditional independence can be encoded in the structure of Bayesian Network, but CSI cannot.
Algebraic Decision Diagram

Motivation

Let $X$, $Y$ and $Z$ be three random variables.

Independence

$$X \perp \perp Y \Rightarrow \forall x, y \quad \Pr(x, y) = \Pr(x) \Pr(y)$$

Conditional Independence

$$X \perp \perp Y \mid Z \Rightarrow \forall x, y, z \quad \Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z)$$
Algebraic Decision Diagram

Motivation

Let $X$, $Y$ and $Z$ be three random variables.

Independence

$X \perp \perp Y \Rightarrow \forall x, y \quad \Pr(x, y) = \Pr(x) \Pr(y)$

Conditional Independence

$X \perp \perp Y \mid Z \Rightarrow \forall x, y, z \quad \Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z)$

Context Specific conditional Independence (CSI)

$X \perp \perp Y \mid Z = z \Rightarrow \exists z \forall x, y \quad \Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z)$
Algebraic Decision Diagram

Motivation

Let $X$, $Y$ and $Z$ be three random variables.

Independence

$$X \perp \perp Y \Rightarrow \forall x, y \quad \Pr(x, y) = \Pr(x) \Pr(y)$$

Conditional Independence

$$X \perp \perp Y \mid Z \Rightarrow \forall x, y, z \quad \Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z)$$

Context Specific conditional Independence (CSI)

$$X \perp \perp Y \mid Z = z \Rightarrow \exists z \forall x, y \quad \Pr(x, y \mid z) = \Pr(x \mid z) \Pr(y \mid z)$$

Both independence and conditional independence can be encoded in the structure of Bayesian Network, but CSI cannot.
**Algebraic Decision Diagram**

**Motivation**

**Decision Tree representation**

Use decision tree to capture the context specific dependencies

![Decision Tree Diagram]

\( \mathbf{X}_2 \) does not appear in the left branch of \( \mathbf{X}_1 \) and \( \mathbf{X}_4 \) does not appear in the branch when \( \mathbf{X}_3 \) takes value 1.
Algebraic Decision Diagram

Motivation

Algebraic Decision Diagram
Decision Tree cannot reuse isomorphic sub-graphs
Algebraic Decision Diagram

Motivation

Decision Tree cannot reuse isomorphic sub-graphs
Algebraic Decision Diagram

Motivation

Decision Tree cannot reuse isomorphic sub-graphs
Algebraic Decision Diagram

Motivation

Decision Tree cannot reuse isomorphic sub-graphs

Using directed acyclic graphs instead of trees!
Algebraic Decision Diagram

Discussion

- Algebraic Decision Diagram is a data structure to compactly encode any discrete function with finite support.
- Context Specific Independence (CSI) can be encoded using Algebraic Decision Diagram (better than tabular representation).
- Efficiently avoid the replication problem by reusing isomorphic subgraph (better than decision tree representation).
- We use Algebraic Decision Diagram to encode local CPDs in Bayesian Network.
A Sum-Product Network is a

- Directed acyclic graph of indicator variables, sum nodes and product nodes.
- Each edge emanated from a sum node is associated with a non-negative weight.
- Value of a product node is the product of its children.
- Value of a sum node is the weighted sum of its children.
Sum-Product Network

Example

\[ f(I_{x_1}, I_{\bar{x}_1}, I_{x_2}, I_{\bar{x}_2}) = 10(I_{x_1} + 4I_{\bar{x}_1})(6I_{x_2} + 14I_{\bar{x}_2}) + 6(6I_{x_1} + 4I_{\bar{x}_1})(2I_{x_2} + 8I_{\bar{x}_2}) + 9(9I_{x_1} + I_{\bar{x}_1})(2I_{x_2} + 8I_{\bar{x}_2}) = 594I_{x_1}I_{x_2} + 1776I_{x_1}I_{\bar{x}_2} + 306I_{\bar{x}_1}I_{x_2} + 824I_{\bar{x}_1}I_{\bar{x}_2} \]
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

**Joint Inference**

$$\Pr(X_1 = 1, X_2 = 0) = \frac{1776 + 10}{3500} = \frac{35}{90}$$
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Joint Inference

\[ \Pr(X_1 = 1, X_2 = 0) \]

Setting \( I_{x_1} = 1, I_{\bar{x}_1} = 0, I_{x_2} = 0, I_{\bar{x}_2} = 1 \).
Sum-Product Network

Inference

Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Joint Inference

\[ \Pr(X_1 = 1, X_2 = 0) \]

Setting \( \mathbb{I}_{x_1} = 1, \mathbb{I}_{\bar{x}_1} = 0, \mathbb{I}_{x_2} = 0, \mathbb{I}_{\bar{x}_2} = 1. \)
Sum-Product Network

Inference

Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Joint Inference

\[ \Pr(X_1 = 1, X_2 = 0) \]

Setting \( I_{x_1} = 1, I_{\bar{x}_1} = 0, I_{x_2} = 0, I_{\bar{x}_2} = 1 \).
Sum-Product Network

Inference

Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Joint Inference

Pr(\(X_1 = 1, X_2 = 0\))? Setting \(\mathbb{I}_{x_1} = 1, \mathbb{I}_{\overline{x}_1} = 0, \mathbb{I}_{x_2} = 0, \mathbb{I}_{\overline{x}_2} = 1\).
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

**Joint Inference**

\[ \Pr(X_1 = 1, X_2 = 0) \]

Setting \( I_{x_1} = 1, I_{\bar{x}_1} = 0, I_{x_2} = 0, I_{\bar{x}_2} = 1 \).

\[ \Pr(X_1 = 1, X_2 = 0) = \frac{1776}{594 + 1776 + 306 + 824} = \frac{1776}{3500} \]
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Marginal Inference

Pr(X₁ = 1) ?
Sum-Product Network

Inference

Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Marginal Inference

\[ \Pr(X_1 = 1) \]

Setting \( \mathbb{I}_{x_1} = 1, \mathbb{I}_{\neg x_1} = 0, \mathbb{I}_{x_2} = 1, \mathbb{I}_{\neg x_2} = 1 \).
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Marginal Inference

$$\Pr(X_1 = 1)$$ ? Setting $I_{x_1} = 1$, $I_{\bar{x}_1} = 0$, $I_{x_2} = 1$, $I_{\bar{x}_2} = 1$. 

$$\Pr(X_1 = 1) = \frac{2370}{594 + 1776 + 306 + 824} = \frac{2370}{3500} = \frac{43}{90}$$
Sum-Product Network
Inference

Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Marginal Inference
\[ \Pr(X_1 = 1) \]
Setting \( I_{x_1} = 1, I_{\bar{x}_1} = 0, I_{x_2} = 1, I_{\bar{x}_2} = 1 \).
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Marginal Inference

\[ \Pr(X_1 = 1) = \frac{2370 	imes 120 	imes 60 	imes 90 + 6 + 9 + 20 + 10}{1001} \]
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Marginal Inference

**Pr**(\(X_1 = 1\)) = \(\frac{2370}{594 + 1776 + 306 + 824} = \frac{2370}{3500}\)
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Conditional Inference

Pr(X₂ = 0|X₁ = 1)?
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Conditional Inference

\[ Pr(X_2 = 0|X_1 = 1) \]
\[ Pr(X_2 = 0|X_1 = 1) = \frac{Pr(X_1=1,X_2=0)}{Pr(X_1=1)} \]
Joint/Marginal/Conditional queries can be answered in linear time in Sum-Product Network.

Conditional Inference
\[
\Pr(X_2 = 0 | X_1 = 1) = \frac{\Pr(X_1 = 1, X_2 = 0)}{\Pr(X_1 = 1)}
\]

Two passes through the Sum-Product Network, one to compute \(\Pr(X_1 = 1, X_2 = 0)\), the other to compute \(\Pr(X_1 = 1)\).
Sum-Product Network
Deep Learning Perspective

Deep structure

- Sum node ⇔ Weighted linear activation function
- Product node ⇔ Component-wise nonlinear activation function
Sum-Product Network

Definition

Definition (scope)

The *scope* of a node in an SPN is defined as the set of variables that have indicators among the node’s descendants: For any node $v$ in an SPN, if $v$ is a terminal node, say, an indicator variable over $X$, then $\text{scope}(v) = \{X\}$, else $\text{scope}(v) = \bigcup_{\tilde{v} \in \text{Ch}(v)} \text{scope}(\tilde{v})$.

Definition (Complete)

An SPN is *complete* iff each sum node has children with the same scope.

Definition (Consistent)

An SPN is consistent iff no variable appears negated in one child of a product node and non-negated in another.
Sum-Product Network

Definition

Definition (Decomposable)

An SPN is decomposable iff for every product node $v$, $\text{scope}(v_i) \cap \text{scope}(v_j) = \emptyset$ where $v_i, v_j \in Ch(v), i \neq j$.

Definition (Valid)

An SPN is said to be valid iff it defines a (unnormalized) probability distribution.

Theorem (Poon and Domingos)

*If an SPN $S$ is complete and consistent, then it is valid.*
Sum-Product Network

Definition

Definition (Decomposable)
An SPN is decomposable iff for every product node $v$, $\text{scope}(v_i) \cap \text{scope}(v_j) = \emptyset$ where $v_i, v_j \in \text{Ch}(v)$, $i \neq j$.

Definition (Valid)
An SPN is said to be *valid* iff it defines a (unnormalized) probability distribution.

Theorem (Poon and Domingos)
*If an SPN $S$ is complete and consistent, then it is valid.*

Valid SPN induces a (unnormalized) probability distribution by the network polynomial defined by the root of the SPN.
Main Theorems

SPN-BN

Let $|S|$ be the size of the SPN, i.e., the number of nodes plus the number of edges in the graph. For a BN $B$, the size of $B$, $|B|$, is defined by the size of the graph plus the size of all the CPDs in $B$. 

Theorem (SPN-BN)
There exists an algorithm that converts any complete and decomposable SPN $S$ over Boolean variables $\mathbf{X}$ into a BN $B$ with CPDs represented by ADDs in time $O(N|S|)$. Furthermore, $S$ and $B$ represent the same distribution and $|B| = O(N|S|)$.

Corollary (SPN-BN)
There exists an algorithm that converts any complete and consistent SPN $S$ over Boolean variables $\mathbf{X}$ into a BN $B$ with CPDs represented by ADDs in time $O(N^2|S|)$. Furthermore, $S$ and $B$ represent the same distribution and $|B| = O(N|S|^2)$. 
Main Theorems

SPN-BN

Let $|S|$ be the size of the SPN, i.e., the number of nodes plus the number of edges in the graph. For a BN $B$, the size of $B$, $|B|$, is defined by the size of the graph plus the size of all the CPDs in $B$.

Theorem (SPN-BN)

There exists an algorithm that converts any complete and decomposable SPN $S$ over Boolean variables $X_{1:N}$ into a BN $B$ with CPDs represented by ADDs in time $O(N|S|)$. Furthermore, $S$ and $B$ represent the same distribution and $|B| = O(N|S|)$. 
Main Theorems

**SPN-BN**

Let $|S|$ be the size of the SPN, i.e., the number of nodes plus the number of edges in the graph. For a BN $B$, the size of $B$, $|B|$, is defined by the size of the graph plus the size of all the CPDs in $B$.

**Theorem (SPN-BN)**

There exists an algorithm that converts any complete and decomposable SPN $S$ over Boolean variables $X_{1:N}$ into a BN $B$ with CPDs represented by ADDs in time $O(N|S|)$. Furthermore, $S$ and $B$ represent the same distribution and $|B| = O(N|S|)$.

**Corollary (SPN-BN)**

There exists an algorithm that converts any complete and consistent SPN $S$ over Boolean variables $X_{1:N}$ into a BN $B$ with CPDs represented by ADDs in time $O(N|S|^2)$. Furthermore, $S$ and $B$ represent the same distribution and $|B| = O(N|S|^2)$. 
Remark

The BN $B$ generated from $S$ has a simple bipartite DAG structure, where all the source nodes are hidden variables and the terminal nodes are the Boolean variables $X_1$.

Remark

Assuming sum nodes alternate with product nodes in SPN $S$, the depth of $S$ is proportional to the maximum in-degree of the nodes in $B$, which, as a result, is proportional to a lower bound of the tree-width of $B$. 
Main Theorems

SPN-BN

Remark
The BN $\mathcal{B}$ generated from $\mathcal{S}$ has a simple bipartite DAG structure, where all the source nodes are hidden variables and the terminal nodes are the Boolean variables $\mathbf{X}_{1:N}$.
Remark
The BN $\mathcal{B}$ generated from $\mathcal{S}$ has a simple bipartite DAG structure, where all the source nodes are hidden variables and the terminal nodes are the Boolean variables $\mathbf{X}_{1:N}$.

Remark
Assuming sum nodes alternate with product nodes in SPN $\mathcal{S}$, the depth of $\mathcal{S}$ is proportional to the maximum in-degree of the nodes in $\mathcal{B}$, which, as a result, is proportional to a lower bound of the tree-width of $\mathcal{B}$. 
Main Theorems

BN-SPN

Theorem (BN-SPN)

Given the BN $B$ with ADD representation of CPDs generated from a complete and decomposable SPN $S$ over Boolean variables $X_{1:N}$, the original SPN $S$ can be recovered by applying the Variable Elimination algorithm to $B$ in $O(N|S|)$. 

Remark

The combination of the above two theorems shows that distributions for which SPNs allow a compact representation and efficient inference, BNs with ADDs also allow a compact representation and efficient inference (i.e., no exponential blow up).
Main Theorems

BN-SPN

Theorem (BN-SPN)

*Given the BN $\mathcal{B}$ with ADD representation of CPDs generated from a complete and decomposable SPN $\mathcal{S}$ over Boolean variables $\mathbf{X}_{1:N}$, the original SPN $\mathcal{S}$ can be recovered by applying the Variable Elimination algorithm to $\mathcal{B}$ in $O(N|\mathcal{S}|)$.*

Remark

The combination of the above two theorems shows that distributions for which SPNs allow a compact representation and efficient inference, BNs with ADDs also allow a compact representation and efficient inference (i.e., no exponential blow up).
1. Define *Normal SPN*, a sub-class of SPN
Road Map

1. Define *Normal SPN*, a sub-class of SPN
2. Taking advantage of normal SPN, show linear transformation from SPN to BN
Road Map

1. Define *Normal SPN*, a sub-class of SPN
2. Taking advantage of normal SPN, show linear transformation from SPN to BN
3. Taking advantage of Variable Elimination algorithm on ADD, show linear transformation from BN to SPN
Definition (Normal Sum-Product Network)

An SPN is said to be normal if

1. It is complete and decomposable.
2. For each sum node in the SPN, the weights of the edges emanating from the sum node are nonnegative and sum to 1.
3. Every terminal node in an SPN is a univariate distribution over a Boolean variable and the size of the scope of a sum node is at least 2 (sum nodes whose scope is of size 1 are reduced into terminal nodes).
Normal Sum-Product Network

Definition

Definition (Normal Sum-Product Network)

An SPN is said to be normal if

1. It is complete and decomposable.
2. For each sum node in the SPN, the weights of the edges emanating from the sum node are nonnegative and sum to 1.
3. Every terminal node in an SPN is a univariate distribution over a Boolean variable and the size of the scope of a sum node is at least 2 (sum nodes whose scope is of size 1 are reduced into terminal nodes).

Theorem (Normal Transformation)

For any complete and consistent SPN \( S \), there exists a normal SPN \( S' \) such that \( \Pr_S(\cdot) = \Pr_{S'}(\cdot) \) and \( |S'| = O(|S|^2) \).
Normal Sum-Product Network

Example

Each terminal node is a univariate distribution.
Normal Sum-Product Network

Example

- Each terminal node is a univariate distribution.
- Each internal sum node corresponds to a hidden variable with multinomial distribution which defines a mixture model.
Normal Sum-Product Network

Example

- Each terminal node is a univariate distribution.
- Each internal sum node corresponds to a hidden variable with multinomial distribution which defines a mixture model.
- Each internal product node encodes a rule of context specific independence over its children.
Given a normal SPN $S$ over $X_{1:N}$, construct:

- A hidden node $H_v$ for each sum node $v$ in $S$.
Given a normal SPN $S$ over $\mathbf{X}_{1:N}$, construct:

- A hidden node $H_v$ for each sum node $v$ in $S$.
- An observable node $X_n$ for each variable $X_n$ in $S$. 

The structure of $B$ is a directed bipartite graph, with a layer of hidden nodes pointing to a layer of observable nodes.
SPN-BN
Structure Construction

Given a normal SPN $S$ over $X_{1:N}$, construct:

- A hidden node $H_v$ for each sum node $v$ in $S$.
- An observable node $X_n$ for each variable $X_n$ in $S$.
- A directed from $H_v$ to $X_n$ iff $X_n$ appears in the sub-SPN rooted at $v$ in $S$. 

The structure of $B$ is a directed bipartite graph, with a layer of hidden nodes pointing to a layer of observable nodes.
SPN-BN
Structure Construction

Given a normal SPN $S$ over $X_{1:N}$, construct:

- A hidden node $H_v$ for each sum node $v$ in $S$.
- An observable node $X_n$ for each variable $X_n$ in $S$.
- A directed from $H_v$ to $X_n$ iff $X_n$ appears in the sub-SPN rooted at $v$ in $S$.

The structure of $B$ is a directed bipartite graph, with a layer of hidden nodes pointing to a layer of observable nodes.
Given a normal SPN $S$ over $\mathbf{X}_{1:N}$, construct:

- A decision stump from the sum node $v$ for each hidden node $H_v$.

![Diagram of SPN-CPD Construction]

$A_H = \begin{bmatrix} \frac{4}{7} & \frac{6}{35} & \frac{9}{35} \\ X_1 & X_1 & X_2 & X_2 \end{bmatrix}$
Given a normal SPN $S$ over $X_{1:N}$, construct:

- A decision stump from the sum node $v$ for each hidden node $H_v$.
- An induced sub-SPN $S_{X_n}$ by node set $\{X_n\}$ from $S$ and then contract all the product nodes in $S_{X_n}$.
Given a normal SPN $S$ over $X_{1:N}$, construct:

- A decision stump from the sum node $v$ for each hidden node $H_v$.
- An induced sub-SPN $S_{X_n}$ by node set $\{X_n\}$ from $S$ and then contract all the product nodes in $S_{X_n}$.
Theorem

For any normal SPN $S$ over $X_{1:N}$, the constructed BN $B$ encodes the same probability distribution, i.e., $Pr_S(x) = Pr_B(x), \forall x$. 
Theorem
For any normal SPN $S$ over $\mathbf{X}_{1:N}$, the constructed BN $B$ encodes the same probability distribution, i.e., $\Pr_S(x) = \Pr_B(x), \forall x$.

Theorem
There exists an algorithm, for any normal SPN $S$ over $\mathbf{X}_{1:N}$, constructs an equivalent BN in time $O(N|S|)$. 
Theorem
For any normal SPN $S$ over $X_{1:N}$, the constructed BN $B$ encodes the same probability distribution, i.e., $Pr_S(x) = Pr_B(x), \forall x$.

Theorem
There exists an algorithm, for any normal SPN $S$ over $X_{1:N}$, constructs an equivalent BN in time $O(N|S|)$.

Theorem
$|B| = O(N|S|)$, where BN $B$ is constructed from the normal SPN $S$ over $X_{1:N}$.
Extend Algebraic Decision Diagram to *Symbolic* Algebraic Decision Diagram where $+,-,\times,/$ are allowed to be internal nodes.
Extend Algebraic Decision Diagram to *Symbolic* Algebraic Decision Diagram where $+,-,\times,/\,$ are allowed to be internal nodes.

**Example**

Given symbolic ADDs $A_{X_1}$ over $X_1$ and $A_{X_2}$ over $X_2$. A symbolic ADD $A_{X_1,X_2}$ over $X_1, X_2$ encodes a function over $X_1$ and $X_2$ such that $A_{X_1,X_2}(x_1,x_2) \triangleq (A_{X_1} \otimes A_{X_2})(x_1,x_2) = A_{X_1}(x_1) \times A_{X_2}(x_2)$. 


Extend Algebraic Decision Diagram to *Symbolic* Algebraic Decision Diagram where $+, −, ×, /$ are allowed to be internal nodes.

**Example**

Given symbolic ADDs $A_{X_1}$ over $X_1$ and $A_{X_2}$ over $X_2$. A symbolic ADD $A_{X_1, X_2}$ over $X_1$, $X_2$ encodes a function over $X_1$ and $X_2$ such that $A_{X_1, X_2}(x_1, x_2) \triangleq (A_{X_1} \otimes A_{X_2})(x_1, x_2) = A_{X_1}(x_1) \times A_{X_2}(x_2)$.

Define two operations in symbolic ADD:

- *Multiplication* between pairs of symbolic ADDs
- *Summing Out* one internal variable in symbolic ADD
Theorem (SPN-BN)

There exists a variable ordering such that applying Variable Elimination with the ordering to BN with ADDs builds the original SPN $S$ in $O(N|S|)$. 

BN-SPN
Algorithm
Discussion

- SPNs and BN with ADDs share the same representational power.

- SPNs with any depth $\iff$ directed bipartite BN.

- SPNs are history recording or caching of the inference process on BN.

- The depth of SPN is linearly proportional to a lower bound of the tree-width of the BN.

- SPNs can be viewed as hierarchical mixture models with reusability.

- CSI are key to allow linear exact inference on BN with high tree-width.
Discussion

- SPNs and BN with ADDs share the same representational power.
- SPNs with any depth $\Leftrightarrow$ directed bipartite BN.
Discussion

- SPNs and BN with ADDs share the same representational power.
- SPNs with any depth $\Leftrightarrow$ directed bipartite BN.
- SPNs are *history recording* or *caching* of the inference process on BN.
Discussion

- SPNs and BN with ADDs share the same representational power.
- SPNs with any depth ⇔ directed bipartite BN.
- SPNs are *history recording* or *caching* of the inference process on BN.
- The depth of SPN is linearly proportional to a lower bound of the tree-width of the BN.
Discussion

- SPNs and BN with ADDs share the same representational power.
- SPNs with any depth $\Leftrightarrow$ directed bipartite BN.
- SPNs are *history recording* or *caching* of the inference process on BN.
- The depth of SPN is linearly proportional to a lower bound of the tree-width of the BN.
- SPNs can be viewed as hierarchical mixture models with reusability.
Discussion

- SPNs and BN with ADDs share the same representational power.
- SPNs with any depth ⇔ directed bipartite BN.
- SPNs are *history recording* or *caching* of the inference process on BN.
- The depth of SPN is linearly proportional to a lower bound of the tree-width of the BN.
- SPNs can be viewed as hierarchical mixture models with reusability.
- CSI are key to allow linear exact inference on BN with high tree-width.
Thanks

Question and Answering

short version: ICML 2015
full version: arXiv:1501.01239