

# Sum-Product Network and Its Application to Image Completion

A thesis defense

Yilin Zheng

Supervised by: Ke Tang & Shan He

Southern University of Science and Technology(SUSTech)

May 17, 2019

# Contents

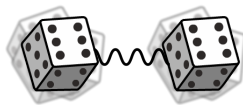
- 1 Introduction
- 2 Sum-Product Network
- 3 Application: Image Completion
- 4 Conclusion and Future Work

# Independence of Probability

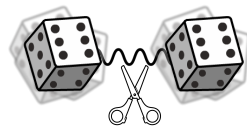
Two dice:



Case #1: Complete Independence



Case #2: Complete Dependence



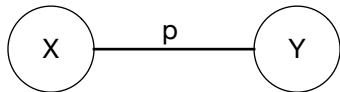
Case #3: Conditional Independence

- Most cases are **conditional independence**.
- The key is the **dependence** of events.

# Probabilistic Graphical Model

Probabilistic graphical model(PGM):

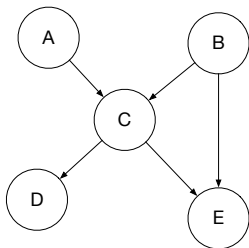
- Node(vertex): random variables
- Edge: condition(association)
- Weight: probability
- Network polynomial: computation



# Traditional PGM: Bayesian Network

Bayesian Network(BN):

- Graph type: directed acyclic graph(DAG)
- Representation: conditional dependence
- Network polynomial:  $p_{\mathcal{B}}(X) = \prod_{X \in \mathbf{X}} p(X|\mathbf{parent}(X))$



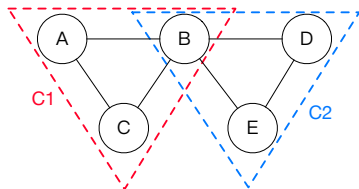
$$p_{\mathcal{B}}(A, B, C, D, E) = p(A)p(B)p(C|A, B)p(D|C)p(E|B, C)$$

Figure: An Example of BN

# Traditional PGM: Markov Random Field

Markov Random Field(MRF):

- Graph type: undirected graph
- Representation: Markov properties
- Network polynomial:  $p_{\mathcal{M}}(\mathbf{X}) = \frac{1}{Z_{\mathcal{M}}} \prod_{l=1}^L \Psi_{C_l}(\mathbf{X}_{C_l})$



$$p_{\mathcal{M}}(X_1, X_2, X_3, X_4, X_5) = \frac{1}{Z} \Psi_{C_1}(X_1, X_2, X_4) \Psi_{C_2}(X_2, X_3, X_5)$$

Figure: An Example of MRF

# Learning

Learning:

1 Object:

- Structure learning
- Parameters learning

2 Approach:

- Generative learning
- Discriminative learning

# Inference

Inference:

- Marginalization
- Conditions
- Most probable explanation
- Maximum a-posterior



# Weaknesses

Weaknesses:

- 1 Complexity scales unproportionally
- 2 Approximate learning
- 3 Intractability
- 4 Separation of learning and inference

# Motivation

Why SPN:

- 1 Complexity scales up linearly
- 2 Exact learning
- 3 Tractability
- 4 Combination of learning and inference

# Target

Targets:

- 1 Implement an SPN
- 2 Reproduce the application to image completion

# Arithmetic Circuit(AC)

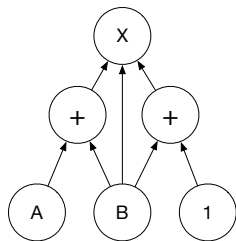


Figure: An Example of AC

Network polynomial:

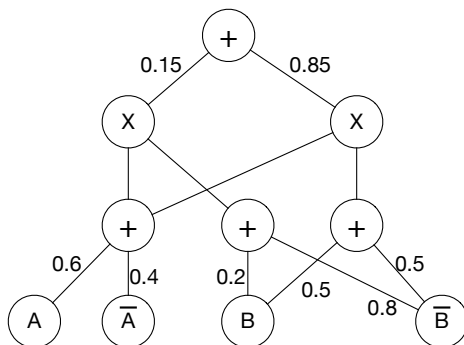
$$f(A, B, 1) = (A + B)B(B + 1)$$

# What is SPN

Sum-product network (SPN):

- Graph type: DAG
- Leaves: random variables
- Internal node: sum node, product node
- Root: sum node
- Network polynomial:  $f_S(\mathbf{X}) = \sum_{X \in \mathbf{X}} \Phi(X) \prod_{X \in \mathbf{X}} (X)$

# Example



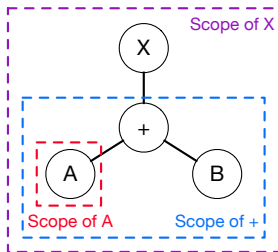
$$\begin{aligned}
 f_S(X_1, X_2, X_3, \bar{X}_1, \bar{X}_2, \bar{X}_3) \\
 &= 0.95X_1(0.1X_2 + 0.9\bar{X}_2) \\
 &\quad (0.5X_3 + 0.5\bar{X}_3) \\
 &+ 0.05(0.2X_2 + 0.8\bar{X}_2) \\
 &\quad (0.3X_3 + 0.7\bar{X}_3)\bar{X}_1
 \end{aligned}$$

Figure: An Example of SPN

# Scope

## Definition (Scope of a Node(sc))

$$\text{sc}(N) \begin{cases} \{X\} & \text{if } N \text{ is a leaf} \\ \cup_{C \in \text{chi}(N)} \text{sc}(C) & \text{if } N \text{ is an internal node} \end{cases}$$



# Properties

## Properties:

- **Completeness:** a sum node's children have the same scope
- **Consistency:** a variable and its negation are in the same scope of a product node
- **Validity:** completeness and consistency
- **Decomposability:** no variable appears in more than one child of a product node

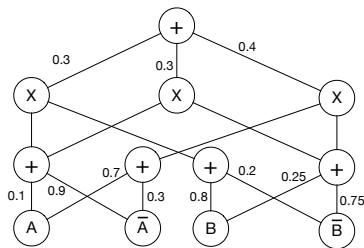


Figure: An Example of Valid SPN



# Learning Methods

## Learning:

- Structure learning:
  - Poon-Domingos Architecture: Rectangle regions
  - Dennis-Venture Architecture: Any shape regions
- Parameter learning:
  - Gradient method: maximize log-likelihood
  - EM algorithm: introduce latent variables, inference in E-step, update weights in M-step

# Inference Methods

Differential approach:

- 1 Compute the marginal distribution via network polynomials
- 2 Interpret the probability distribution through derivatives

# Poon-Domingos Architecture

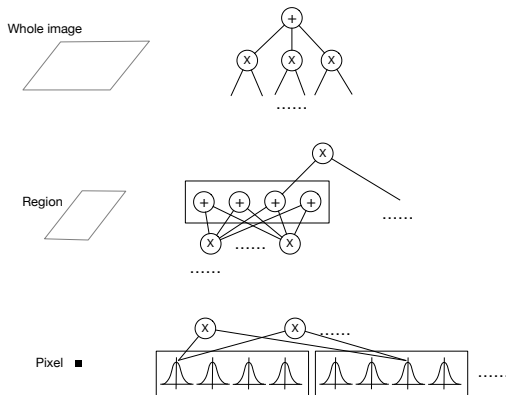


Figure: Poon-Domingos Architecture

# Program

Code:

- **common**: helper functions for time management, messaging between progress(MPI), parameter settings, and some utilities.
- **evaluation**: dataset processing, model generation, evaluation.
- **spn**: SPN architecture, including node definition, computation functions, the learning, and inference.

# Callgraph

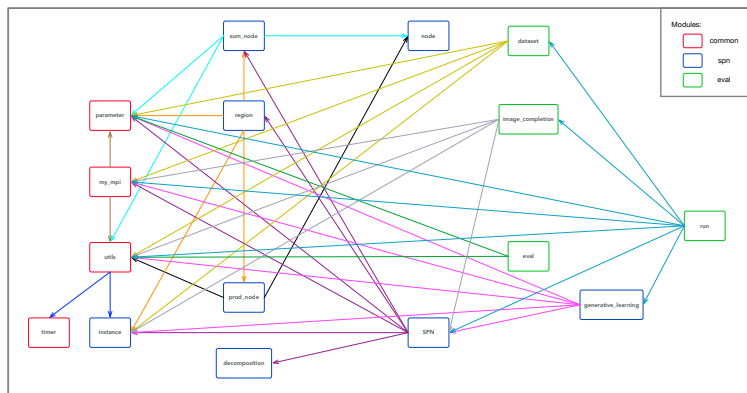


Figure: Callgraph of Program

# Enviroment

## Enviroment:

- Platform: TaiYi
- Library: OpenMPI C++
- Dataset:
  - **Caltech:**
    - 101 categories, from 40 to 800 images per category
    - about  $300 \times 200$  pixels, rescaled to  $100 \times 64$  pixels
  - **Olivetti:**
    - face images taken between Apr. 1992 and Apr. 1994 at AT&T Laboratories Cambridge
    - size  $64 \times 64$  pixels

# Experiments

## Experiments:

- 1 **Caltech**: 80 cores, **Olivetti**: 40 cores, size  $64 \times 64$
- 2 **Caltech**: 120 cores, **Olivetti**: 80 cores, size  $64 \times 64$
- 3 **Caltech**: 80 cores, size  $100 \times 64$
- 4 **Caltech**: 120 cores, size  $100 \times 64$

Poon's experiments: 102 cores, **Olivetti**: 51 cores, size  $64 \times 64$

# Comparison on MSE(1)

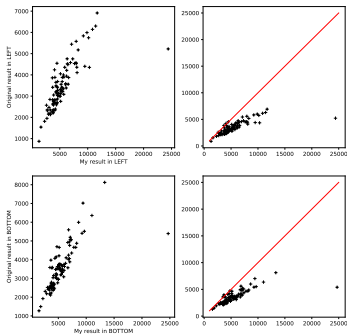


Figure: Exp. #1 vs Poon's(Caltech)

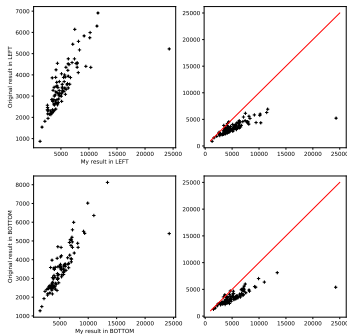


Figure: Exp. #2 vs Poon's(Caltech)



# Comparison on MSE(2)

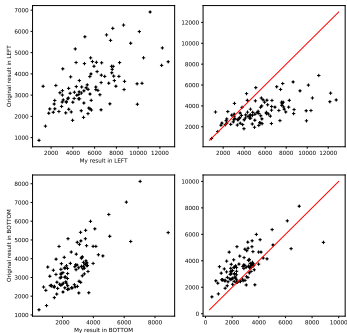


Figure: Exp. #3 vs Poon's(Caltech)

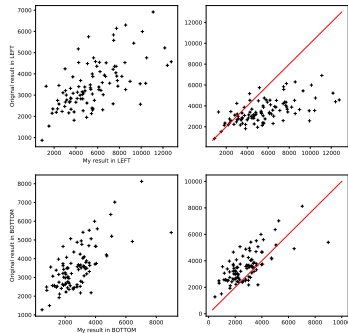


Figure: Exp. #4 vs Poon's(Caltech)

# Comparison on Number of Cores

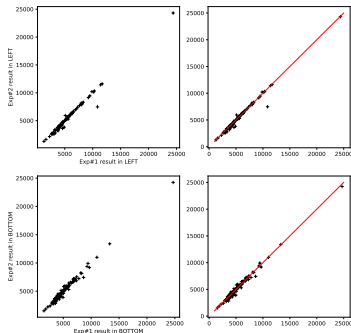


Figure: Exp. #1 vs Exp. #2(Caltech)

# Comparison on Input Size

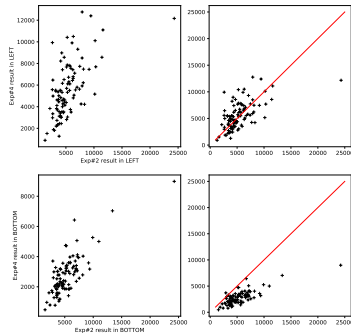
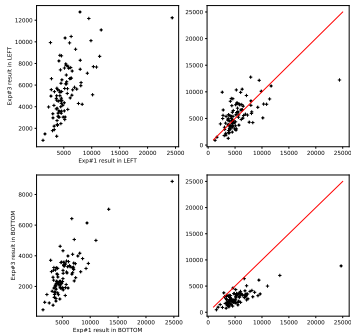


Figure: Exp. #1 vs Exp. #3 (Caltech)      Figure: Exp. #2 vs Exp. #4 (Caltech)

# Comparison on Image(1)



Figure: Airplanes-bottom(Poon's, Exp. #2, Exp. #4)



Figure: Yin\_yang-left(Poon's, Exp. #2, Exp. #4)

# Comparison on Image(2)



Figure: Pyramid-bottom(Poon's, Exp. #2, Exp. #4)



Figure: Sunflower-bottom(Poon's, Exp. #2, Exp. #4)

# Comparison on Time

Time Cost	Poon's	Exp. #1	Exp. #2	Exp. #3	Exp. #4
Caltech-101	$\leq 2$ hours	$\leq 4$ hours	$\leq 7$ hours	$\leq 11$ hours	$\leq 19$ hours
Olivetti	within a few minutes	$\leq 4$ minutes	$\geq 72$ hours	Nan	Nan

**Table:** Time Cost of Poon's Experiments and My Experiments

# Analysis

Conclusion:

- Number of cores makes no influence
- Larger input size leads to lower MSE
- My implementation is valid

Why:

- Randomness in architecture
- Difference between implementation
- Complexity of model

# Conclusion

## Conclusion:

- My implementation is valid and successful
- Reproduction is not easy



# Future Work

Future work:

- 1 Architecture improvement
- 2 More applications
- 3 New algorithms for learning and inference

# Acknowledgement

Advisors:

Prof. Ke Tang

Prof. Shan He

Inspector:

Prof. Bo Tang

Committee Members:

Prof. Qi Wang (Committee chair)

Prof. Jianqiao Yu (Committee member)

Prof. Jialin Liu (Committee member)

# Thanks

Thanks for listening!

# Q & A

## Questions ?