CLC__________ Number__________
UDC__________ Available for reference ☐ Yes ☐ No

Southern University of Science and Technology

A Dissertation for Bachelor's Degree

Thesis Title: Sum-Product Network and Its Application to Image Completion

Student Name: Yilin ZHENG

Student ID: 11510506

Department: Computer Science and Engineering

Program: Computer Science and Technology

Thesis Advisor: Prof. Ke Tang & Prof. Shan He

Date: May 14th, 2019

COMMITMENT OF HONESTY

1. I solemnly promise that the paper presented comes from my independent research work under my supervisor's supervision. All statistics and images are real and reliable.
2. Except for the annotated reference, the paper contents no other published work or achievement by person or group. All people making important contributions to the study of the paper have been indicated clearly in the paper.
3. I promise that I did not plagiarize other people's research achievement or forge related data in the process of designing topic and research content.
4. If there is violation of any intellectual property right, I will take legal responsibility myself.

Signature: ___________________

Date:___________________

Preface

This thesis is my final report about my undergraduate project. This thesis is co-supervised by Prof. Ke Tang [1] and Prof. Shan He [2].

Before introducing the main content, I would like to summarize the process of this thesis. I met Prof. He during his visit to my university(SUSTech) via doctor candidate Chengbin Hou in 2018. Before that, I had some research attempts on graphs or networks and joined the discussion with Chengbin Hou and Guoji Fu. They are interested in graph representation and embedding technologies, which later stimulate my interests on that during the participation. So, I was interested in Prof. He and his research since his direction contains graphs or networks. Late in the year, when asked to decide the topic, I contacted Prof. He for a feasible topic related to graphs or networks after making an agreement with Prof. Tang. That was the reason that I chose this topic. I am grateful to Prof. Tang for his respect for my choice and to Prof. He for his great idea. During this work, Prof. He asked doctor candidate Weifeng Li to help me. We had discussed several times to make sure that I can complete this project in time. The original goal was to do a new application to gene dataset, but later, given the time and my actual ability, the target was changed to reproduce the results of the image completion presented in the original paper. Fortunately, I completed the work in time with high completeness. During the period, Prof. Tang encouraged me when I encountered problems. Also, Dr. Guiying Li and Dr. Xiaofen Lu gave me some help when the program encountered bugs.

This topic is about a new kind of probabilistic graphical model. In this thesis, I first gave a comprehensive literature review both in probabilistic theory and graph theory. They are the fundamental of this architecture. Then in the experiment part, I implemented the architecture in C++ according to the Java code provided by the proposer and reproduced the experiment on TaiYi cluster. At last, I did a comparison between my results and proposer's results and analysis the difference.

This process gave me an unforgettable memory as an ending of my university life. After graduation, I will work in the industry, so this thesis might be the last report I wrote, which is also a good memory hard to forget.

Yilin ZHENG

May, 2019 at SUSTech

[1] Southern University of Science and Technology(SUSTech)

[2] University of Birmingham

Contents

List of Tables

List of Figures

[摘　要]

和积网络（SPN）是 2011 年被提出的一种全新的概率图模型。传统的概率图模型饱受一些缺点的制约，例如巨大的计算量和难以驾驭的推理过程。相比于传统的概率图模型，和积网络采用截然不同的语法和语义来降低计算量，并使推理过程容易驾驭，收到了越来越多研究者的关注。此论文记录了我在和积网络方面的一些研究工作。我首先实现了 Poon 提出的一种和积网络架构，然后在集群上复现了图片补全实验，并将最后的结果与作者原来的结果进行比较和分析以验证我实现的模型。虽然我的结果比原始结果略差，但是补全的结果表明我的实现是正确的。在最后，我还讨论了本工作并且提出了一些可行的研究方向。

[关键词]:　和积网络, 图片补全, 机器学习, 概率图模型, 推理

[ABSTRACT]

Sum-product network(SPN) is a new kind of probabilistic graphical model(PGM) first proposed in 2011. The traditional PGMs suffers from some weaknesses such as larger computation burden and intractability of the inference. Compared to traditional PGMs, SPN uses very different syntax and semantics to lower the computations and make the inference tractable, which appeals to more and more researchers. This thesis is about my research work on SPN. I implemented the Poon's architecture of SPN at first, then reproduced the image completion experiment on the cluster, and compared my results with original results to validate my implementation. Although my results are a little worse than the original results, my completed images showed that my implementation is correct. At last, I discussed this work and proposed some feasible future work.

[Keywords]: Sum-product network, image completion, machine learning, probabilistic graphical model, inference

Notations

Ω	Sample space
ω	Elementary event
A	Events
P	Probability distribution
p	Probability mass function(PMF)
$\mathcal{A}$	Sigma-algebra
$\emptyset$	Empty set
$\mathbf{val}(X)$	Image of random variable X, set of its values
$\mathcal{G}$	Graph, directed or undirected graph, structure of PGMs
$\mathbf{V}$	Set of nodes
$\mathbf{E}$	Set of edges
V	Vertex or node
E	Edge
$\mathbf{pa}(N)$	Paths to node N
$\mathbf{cir}(N)$	Circle passing node N
l	Length of a path
$\mathcal{B}$	Bayesian network
$\mathbf{par}(N)$	Parents of node N
$\mathbf{chi}(N)$	Children of node N
$\mathbf{nei}(N)$	Neighbours of node N
$\mathbf{ance}(N)$	Ancestors of node N
$\mathbf{decs}(N)$	Descendants of node N
nance	Non-ancestors of node N
ndesc	Non-descendants of node N
x, y, z	Values of random variable
Θ	Parameters in PGMs
$\mathcal{L}$	Likelihood
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	Random variables
λ	Indicator variables
$\boldsymbol{\lambda}$	Vector of indicator variables
$\overline{X}$	Negation of random variable X
$\Phi(\mathbf{X})$	Parameters of random variables **X**
$\mathbf{S}$	Sum node in SPN
$\mathbf{P}$	Product node in SPN
$\mathbf{sc}(N)$	Scope of node N

N	Node in SPN, either sum node or product node
F	Father node
C	Child node
$\mathcal{D}$	Dataset

Chapter 1 Introduction

Probabilistic graphical models(PGMs) are widely used tools to model the probability distribution of data. For many years, researchers focus the topics on traditional PGMs but suffer from some disadvantages such as intractability and large computation burden. In 2011, Poon and Domingos in[1] first proposed SPN and conducted the experiment on image dataset to validate this new kind of probabilistic graphical model. Sum-product network(SPN) is a new probabilistic graphical model for data learning and inference. Since it resolves some problems encountered in classical probabilistic graphical models, many topics about SPN have been conducted during these years. My topic is also about SPN. In this chapter, I will present the motivation of this project in Section 1.1. Then I will state the target of my research in Section 1.2. The last part, Section 1.3, will be an introduction to the rest of this thesis.

1.1 Motiavtion

The motivation for this topic is to learn about the SPN and to do some fundamental work for further research on SPN and its application. Why SPN can appeal to so many researchers, the details will be presented after a review of the traditional PGMs in Chapter 3. The weakness the traditional PGMs suffers from is the main motivation for researchers to study and explore the SPN.

1.2 Target

The original target is to conduct a new application of SPN to cancer dataset. However, to adjust to the requirement of the undergraduate thesis, the target was then changed to reproduce the application of image completion via Poon's architecture in[1]. The final results contain the source code of the experiment program, the reproduced results of image completion, the analysis of the results, and a thesis paper.

1.3 About This Thesis

The following chapters contain five parts. In Chapter 2, I will review some theories about probability and graph since they are the theoretical basis of PGMs and later the thesis, some terminologies will be mentioned. Chapter 3 is a literature review of the traditional PGMs and their weaknesses. Chapter 4 will introduce SPN in more details. Then, Chapter 5 will present the experiment part from both the program and the results. The last chapter, Chapter 6, is my discussion and some feasible future work about this topic.

Chapter 2 Background

Probabilistic graphical models are constructed on the combination of both probability and graph. So, the theories related to such models play as a theoretical foundation behind them. This chapter will review some important definitions of probability theory in Section 2.1 and graph theory in Section 2.2 for a better understanding of traditional PGMs and SPN. Also, some terminology will be mentioned again in later content.

2.1 Probability Theory

In this section, we will describe the basic formalism of *probability space*, *Sigma-algebra*, *conditional probability*, *Bayes' rule*, and *random variables*(RVs). RVs are the most important and interesting object in probabilistic modelling like machine learning, so, some concepts dealing with RVs will also be presented like *expectations*, *marginal* and *conditional distributions*.

2.1.1 Sigma-algebra and Probability Space

To describe the probability in a mathematical way, let's first introduce some symbols representing the components of probability theory. To describe the randomness, we need a space containing all the possible events, which is called *sample space*, denoted as Ω, and for any *elementary event* in the space, denoted as ω, we pick it randomly every time, which are regarded as random trial. We can find that for any trials such as $\omega = \omega_1$, $\omega = \omega_2$ and $\omega_1 \neq \omega_2$, the elementary events are *mutually exclusive*. To assign the probability to the randomness, we can say, probability "0" means "impossible" while probability "1" means "certain". However, considering the probability of one random trial makes no sense, so scientists further define a set of elementary events $A(\omega \in A)$, called *event*, and assign the probability to the event. So far, all the possibilities of all the elementary event $\omega \in A$ sum up to be the probability of event A, and for any sub-events of A, the sum of all the sub-events are equal to the probability of the event A. Besides, the probability of our sample space Ω is 1.

Using a function P to measure the probability, it can be represented as:

$$P(A) = \sum_{i \in \mathbb{N}} \omega_i > 0 \quad \text{for} \quad \forall \omega_i \in A$$

$$P(A) = \sum_{i \in \mathbb{N}} A_i > 0 \quad \text{for} \quad \forall A_i \subset A, \cup_{i \in \mathbb{N}} A_i = A \text{ and } \cap_{i \in \mathbb{N}} A_i = \emptyset$$

$$P(\Omega) = 1$$

This definition is accessible for the discrete case but will be tough for the continuous situation, so theorists then introduce σ-algebra[2].

Definition 2.1.1 (Sigma-algebra)**.** *For Ω be a set, a σ-algebra over Ω is a collection $\mathcal{A}$ containing Ω, its complements and countable unions. The σ-algebra has the following properties:*

1. $\Omega \in \mathcal{A}$
2. $A \in \mathcal{A} \implies A^c \in \mathcal{A}$
3. $A_n \in \mathcal{A}, \forall n \in \mathbb{N} \implies \cup_{n\in\mathbb{N}} A_n \in A$

Example Let Ω be the event of a die, which is $\Omega = \{1, 2, 3, 4, 5, 6\}$. One of a σ-algebra over Ω can be $\mathcal{A} = \{\emptyset, \{1, 3, 5\}, \{2, 4, 6\}, \Omega\}$.

Based on σ-algebra, theorists further defined the measure space[3].

Definition 2.1.2 (Measure Space)**.** *Given Ω is a set, $\mathcal{A}$ is a σ-algebra over Ω, a tuple $(\Omega, \mathcal{A})$ is a measurable space. A measure on the measurable space $(\Omega, \mathcal{A})$ is a function $f\colon \mathcal{A} \mapsto [0, \infty]$ with the properties:*

1. $f(\emptyset) = 0$
2. $f(\cup_{i\in\mathbb{N}} A_i) = \sum_{i\in\mathbb{N}} f(A_i), \forall A_i \in A, i \neq j \Leftrightarrow A_i \cap A_j = \emptyset$.

The triple $(\Omega, \mathcal{A}, f)$ is called a measure space.

Based on the measure space, if a measure space has the additional property $f(\Omega) = 1$, the measure space changes to be a probability space[2].

Definition 2.1.3 (Probability Space)**.** *Given a measure space $(\Omega, \mathcal{A})$ and a measure P on the measure space with property $P(\Omega) = 1$, the triple $(\Omega, \mathcal{A}, P)$ is called a probability space.*

2.1.2 Conditional Probability, Bayes' Rule and Independence

Upon probability space, theorists can further describe the probability.

Definition 2.1.4 (Conditional Probability[4])**.** *Given a probability space $(\Omega, \mathcal{A}, P)$, let A, B be events where $A, B \in \Omega$, the conditional probability is described as:*

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(AB)}{P(B)}, \textit{where } P(B) > 0.$$

If we exchange the position of A and B and combine both, we can finally get the Bayes' rule.

Definition 2.1.5 (Bayes' Rule)**.**

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

Definition 2.1.6 (Independence of Events)**.** *In a probability space $(\Omega, \mathcal{A}, P)$, if events $A \in \mathcal{A}$ and $B \in \mathcal{A}$ are independent, then for their probability, there exists the relation:*

$$P(AB) = P(A)P(B).$$

2.1.3 Random Variable, Probability Mass Function, Probability Density Function and Expectation

Definition 2.1.7 (Measure Function[4]). *Let* $(\Omega_1, \mathcal{A}_1)$, $(\Omega_2, \mathcal{A}_2)$ *be two measurable spaces, a function* f *is called* $\mathcal{A}_1 - \mathcal{A}_2$*-measurable if* $f\colon \Omega_1 \mapsto \Omega_2$.

Definition 2.1.8 (Random Variable[4]). X *is called a random variable defined on the probability space* $(\Omega, \mathcal{A}, P)$ *if* X *is a* $\mathcal{A} - \mathcal{B}(\mathbb{R})$*-measurable function* $f : \Omega \mapsto \mathbb{R}$. $\mathbf{val}(X)$ *denotes the set of all the values of* X*, which is defined as the image of* Ω *under* X.

Definition 2.1.9 (Distribution of Random Variables[3]). *Given a random variable* X *defined on the probability space* $(\Omega, \mathcal{A}, P)$*, the probability measure* $P_X = P \circ X^{-1}$ *is the distribution of the random variable* X.

Definition 2.1.10 (Probability Mass Function(PMF)). *Given a discrete RV* X *defined on the probability space* $(\Omega, \mathcal{A}, P)$*, the probability mass function(PMF) of* X *is defined as:*

$$p_X(x) = P_X(\{x\}) = P_X(X = x), x \in \mathbf{val}(X).$$

Definition 2.1.11 (Cumulative Distribution Function(CDF)[4]). *Given a RV* X *defined on the probability space* $(\Omega, \mathcal{A}, P)$*, the cumulative distribution function(CDF) of* X *is defined as:*

$$F_X = P_X\left((-\infty, x]\right) = P_X(X \leq x).$$

Any CDF of a RV X is non-decreased, right continuous, and has two limits:

$$\lim_{x \to 0} F_X(x) = 0 \text{ and } \lim_{x \to \infty} F_X(x) = 1.$$

Definition 2.1.12 (Probability Density Function(PDF)). *Given a continuous RV* X *defined on the probability space* $(\Omega, \mathcal{A}, P)$*, the probability density function(PDF)* X *is defined as:*

$$F_X(x) = \int_{-\infty}^{x} p_X(x')dx'.$$

Definition 2.1.13 (Joint Probability Distribution[4]). *Given a probability space* $(\Omega, \mathcal{A}, P)$*,* X_n *be* n *RVs defined on the probability space,* $X_n : \Omega \mapsto \mathbb{R}^N$*, with* $\mathbf{X}(\omega) = (X(\omega_1), \ldots, X(\omega_n))^T$*, the joint probability distribution is defined as:*

$$P_{\mathbf{X}} = P \circ \mathbf{X}^{-1}.$$

Expectation of an RV is a very common concepts used to assess the distribution of the RV.

Definition 2.1.14 (Expectation). *Given a random variable* X *of distribution* p_X*, we define the expectation of the* X *in discrete and continuous respectively:*

$$\mathbb{E}(X) = \begin{cases} \sum_{x \in \mathbf{val}(X)} x \cdot p_X(x), & x \text{ is discrete} \\ \int_{x \in \mathbf{val}(X)} x \cdot p_X(x)dx, & x \text{ is continuous} \end{cases}$$

2.2 Graph Theory

Probabilistic graphical models are essentially graphs to model the joint distribution of random variables. This section will review some concepts of graph theory.

2.2.1 Graph, Directed/Undirected Graph

In graph theory, the symbol $\mathcal{G}$ usually denotes a graph, either a directed or an undirected graph, with **V**, **E** denoting the vertices and edges in the graph respectively.

Definition 2.2.1 (Graph[5]). *A graph $\mathcal{G}$ is defined as a pair $(\mathbf{V}, \mathbf{E})$ consisting of a vertices set* **V** *and an edge set* **E***, together with an incidence function $\phi_{\mathcal{G}}$ which associates with each edges and an unordered pair of vertices.*

The nodes in a graph can contain any useful information, such as the probability of a variable, the order between all other nodes or the reachability of a path. For edges in a graph, we focus the direction of the edge, for an example, edge (V_i, V_j), if the order of both nodes matters, we said the edge is directed, otherwise we recognize both edge (V_i, V_j), (V_j, V_i) are contained in **E**.

Definition 2.2.2 (Directed Graph[4]). *A graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a directed graph if all the edges $(V_i, V_j) \in \mathbf{V}(i \neq j)$ are directed.*

In a directed graph $\mathcal{G}$, $(V_i, V_j)(i \neq j)$ and (V_j, V_i) are regarded as two different edges in **V**.

Definition 2.2.3 (Undirected Graph[4]). *A graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is an undirected graph if all the edges $(V_i, V_j) \in \mathbf{V}(i \neq j)$ are undirected.*

2.2.2 Neighbours, Parents and Children

Definition 2.2.4 (Neighbours[4]). *Neighbours are defined in undirected graphs. Given an undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, in a edge (V_i, V_j), V_i is the neighbour of V_j and vice versa.* $\mathbf{nei}(V)$ *denotes the neighbour of vertex V.*

Definition 2.2.5 (Parents and Children[4]). *Parents and children are defined in directed graphs to distinguish the direction of the edge. Given a directed graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, in a edge (V_i, V_j), also denoted as $(V_i \to V_j)$, V_i is a parent of V_j, and V_j is a child of V_i.* $\mathbf{par}(V)$ *denotes the parents of vertex V, and* $\mathbf{chi}(V)$ *denotes the children of vertex V.*

2.2.3 Path, Trial and Cycles

Definition 2.2.6 (Path and Trial[6][4]). *A path* $\mathbf{pa}_n$ *is a non-empty graph $P = (V, E)$ of the form*

$$V = \{v_0, v_1, \ldots, v_n\} \quad E = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{n-2}, v_{n-1})\},$$

where $v_i(i = 0, 1, 2, \ldots, n)$ are all distinct and linked by the edge $(v_j, v_{j+1})(j = 0, 1, \ldots, n-1)$. While in a directed graph, a trail between V_1 and V_n is the tuple $\{V_1, V_2, \ldots, V_n\}$ with each edge $(V_i \to V_{i+1})$ or $(V_{i+1} \to V_i) \in$ **E** *$(i = 1, \ldots, n-1)$.*

Definition 2.2.7 (Cycles[6]). *If* $\mathbf{pa}_n$ *is a path, and the length of the path is larger than 3, then the graph* $\mathbf{cir}(P) = \mathbf{pa}_n + (v_{n-1}, v_0)$ *is a cycle.*

2.2.4 Directed Acyclic Graphs, Ancestor and Descendant

Definition 2.2.8 (Directed Acyclic Graphs[4]). *Given a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, it is defined as a directed acyclic graph(DAG) if there are no cycles.*

Definition 2.2.9 (Ancestor and Descendant). *In a DAG, given a path from $V_i \in \mathbf{V}$ to $V_j \in \mathbf{V}$, V_i is the ancestor of V_j, and V_j is one of the descendants of V_i.* $\mathbf{ance}(V)$ *and* $\mathbf{decs}(V)$ *represent the ancestors and descendants of node V respectively, while* **nance** *and* **ndesc** *represent non-ancestor and non-descendant.*

2.2.5 Rooted DAG, Directed/Undirected Tree

Definition 2.2.10 (Rooted DAG[4]). *A DAG is a rooted DAG if there exists a unique node V_0 with all other nodes $V_i (i \neq 0)$, all the paths from V_0 to V_i have the length $l \geq 1$ but all the paths from V_i to V_0 have the length of $l = 0$.*

Definition 2.2.11 (Undirected Tree). *An undirected graph is an undirected graph tree.*

Definition 2.2.12 (Directed Tree[4]). *A rooted DAG is a directed tree if for all nodes $V_i (i \in (0, |\mathbf{V}| - 1))$ that the paths from V_i to itself are of length $l \leq 1$.*

2.2.6 Subgraph, Supergraph and Induced Graph

Definition 2.2.13 (Subgraph and Supergraph[4]). *A graph $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$ is a subgraph of graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ if $\mathcal{G}'$ is formed by the vertex set $\mathbf{V}' \subseteq \mathbf{V}$ and the edge set $\mathbf{E}' \subseteq \{(V_i, V_j) | i \neq j, V_i, V_j \in \mathbf{V}\}$. The graph $\mathcal{G}$ is called supergraph of $\mathcal{G}'$.*

Definition 2.2.14 (Induced Graph[4]). *An induced graph $\mathcal{G}' = (\mathbf{V}', \mathbf{E}')$ is graph formed by the set $\mathbf{V}' \subseteq \mathbf{V}$ and the edges set $\mathbf{E}' \subseteq \mathbf{E}$.*

The difference between the subgraph and the induced graph is whether the edges are contained in the original graph. A subgraph may contain edges that not exist in the original graph, while an induced graph can only contain the subset of edges in the original graph.

2.2.7 Cliques, Complete Graph

Definition 2.2.15 (Clique[7]). *Given an undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$, a clique is a subset of vertices $\mathbf{C} \subseteq \mathbf{V}$ such that all pairs of distinct vertices are adjacent. In mathematical formalism, a clique is:*

$$\forall V_i, V_j \in \mathbf{C} \subseteq \mathbf{V} \ (i \neq j), (V_i, V_j) \in \mathbf{E}.$$

Further, a maximal clique is a clique which becomes no longer a clique if adding a vertice $V \in \mathbf{V} \setminus \mathbf{C}$.

Definition 2.2.16 (Complete Graph). *An undirected graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is a complete graph if $\mathbf{V}$ is the maximal clique of graph $\mathcal{G}$.*

Chapter 3 Literature Review

Probabilistic graph models(PGMs) are a type of probabilistic tool to model the probability distribution, enabling people to assess and analyze the random quantities. By using probabilistic models, people can do reasoning and make decisions with uncertainty. This chapter will review the traditional PGMs and summarize the learning and inference methods in Section 3.1. Then in Section 3.2, the weakness of traditional PGMs will be pointed out. At last, Section 3.3 gives out the definition of evidence and network polynomials to describe a network.

3.1 Traditional Probabilistic Graphical Model

Probabilistic graphical models(PGMs) are a combination of probability theory and graph theory. In this section, I will briefly review the popular traditional PGMs: Bayesian networks, and Markov random fields, followed by the learning and inference methods on these models. PGMs are trying to use a graph to describe the probability distributions. However, compared to sum-product network, they are very different from syntax and semantics. Besides, traditional PGMs suffer from some problems such as larger computations and intractability.

3.1.1 Bayesian Network

Definition 3.1.1 (Bayesian Network[4] [8])**.** *A Bayesian network(BN) $\mathcal{B}$ over RVs* $\mathbf{X}$ *is defined as a tuple $(\mathcal{G}, \{p_{\mathbf{X}}\})$ where $\mathcal{G}$ is a DAG over RVs* $\mathbf{X}$ *and $p_{\mathbf{X}}$ is the conditional probability distribution of RVs* $\mathbf{X}$*. The graph $\mathcal{G}$ is a tuple $(\mathbf{X}, \mathbf{E})$ where X is the random variables in the domain and the edges correspond to direct influence of one vertex on another. A Bayesian network defines the following joint probability distribution:*

$$p_{\mathcal{B}}(X) = \prod_{X \in \mathbf{X}} p(X|\mathbf{par}(X))$$

Example of A Bayesian Network Here we use an example to illustrate the Bayesian network. In Figure 3.1, there are five random variable A, B, C, E, D, with each edge from an RV to another RV indicating their conditional independence. According to the definition of BN, we can easily yield the joint probability distribution:

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

The local conditional independencies of BN[8]:

Definition 3.1.2 (Local Conditional Independence)**.** *Given a Bayesian network $\mathcal{B} = (\mathcal{G}, p_X)$, the joint conditional distribution p_X satisfied the property:*

$$X \perp\!\!\!\perp (\mathbf{ndesc}(X) \setminus \mathbf{par}(X)) | \mathbf{par}(X).$$

The RV X is conditional independent of its non-descendants if given its parents.

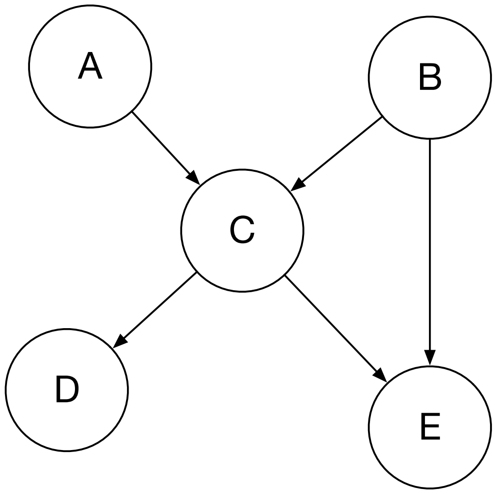


Figure 3.1: Example of a Bayesian Network

3.1.2 Markov Random Field

Definition 3.1.3 (Markov Random Field[4])**.** *A Markov random field(MRF) $\mathcal{M}$ over RVs* $\mathbf{X}$ *is defined as a tuple $(\mathcal{G}, \{\Psi_{\mathbf{C}_l}\}_{l=1}^{L})$. In the tuple, the $\mathcal{G}$ is an undirected graph with the maximal clique $\mathbf{C}_1, \ldots, \mathbf{C}_L \subseteq \mathbf{X}$ and the potential Ψ_{C_i} are nonnegative functions over the maximal clique: $\Psi_{\mathbf{C}_l} : \mathbf{val}(\mathbf{C}_l) \mapsto [0, \infty)$. The mathematical formalism of MRF is given as:*

$$p_{\mathcal{M}}(\mathbf{X}) = \frac{1}{\mathcal{Z}_{\mathcal{M}}} \Phi_{\mathcal{M}}(\mathbf{X}) = \frac{1}{\mathcal{Z}_{\mathcal{M}}} \prod_{l=1}^{L} \Psi_{\mathbf{C}_l}(\mathbf{X}_{\mathbf{C}_l})$$

where $\mathcal{Z}_{\mathcal{M}}$ is a normalization factor:

$$\mathcal{Z}_{\mathcal{M}} = \int_{\mathbf{val}(X_1)} \cdots \int_{\mathbf{val}(X_N)} \Phi_{\mathcal{M}}(X_1, \ldots, X_N) dX_1 \ldots dX_N$$

for continuous variables or

$$\mathcal{Z}_{\mathcal{M}} = \sum_{X_1}^{X_N} \Phi_{\mathcal{M}}(X_1, \ldots, X_N)$$

for discrete variables.

The normalization factor $\mathcal{Z}_{\mathcal{M}}$ is called the partition function and the network is called $\Phi_{\mathcal{M}}$ the unnormalized Markov network(MN).

Example of MRF We also give a simple example of MRF. Figure 3.2 is a MRF with five variables A, B, C, D, E, assume that RVs are discrete. Apparently, there two cliques circle in red or blue dotted line respectively:

$$\begin{aligned} \mathbf{C}_1 &= \{A, B, C\} \text{ and} \\ \mathbf{C}_2 &= \{B, D, E\}. \end{aligned} \tag{3.1}$$

According to the definition, the probability distribution is:

$$p_{\mathcal{M}}(A, B, C, D, E) = \frac{1}{\mathcal{Z}} \Psi_{\mathbf{C}_1}(A, B, C) \Psi_{\mathbf{C}_2}(B, D, E)$$

and the partition function:

$$\mathcal{Z}_{\mathcal{M}} = \sum_{\mathbf{val}(X_1)}^{\mathbf{val}(X_5)} \Phi_{\mathcal{M}}(A, B, C, D, E).$$

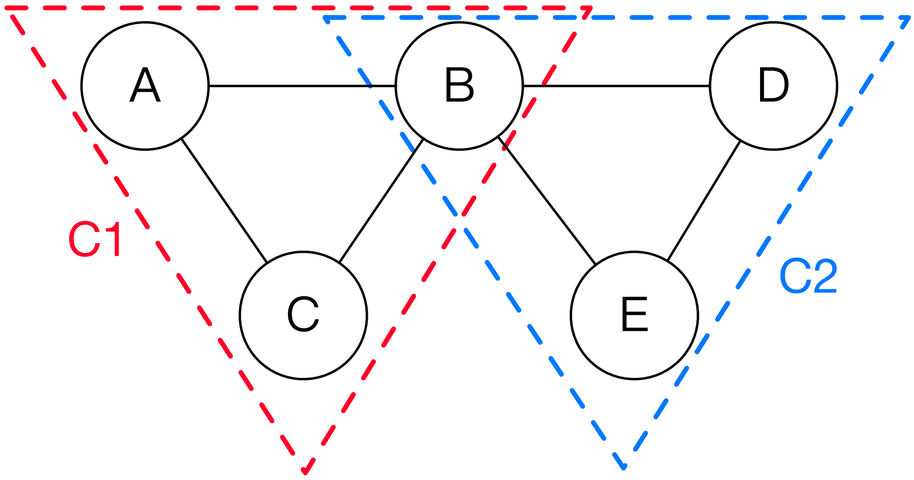


Figure 3.2: Example of a Markov Random Field

Definition 3.1.4 (Separation in MRF[4]). *Given a MRF $\mathcal{M} = (\mathcal{G}, \mathbf{X})$, RVs $X_i, X_j \in \mathbf{X}$ $(i \neq j)$ are separated by $\mathbf{Z} \subset \mathbf{X}$ $(X_i, X_j \notin \mathbf{Z})$ if all paths between X_i and X_j are passing the RVs $X_k \in \mathbf{Z}$. Similarly, any two subsets $\mathbf{X}_1, \mathbf{X}_2 \subset \mathbf{X}$ are separated by $\mathbf{Z}$ if following conditions are met:*

1. $\mathbf{X}_1 \cap \mathbf{Z} = \emptyset$ *and* $\mathbf{X}_2 \cap \mathbf{Z} = \emptyset$,
2. $\mathbf{X}_1 \cap \mathbf{X}_2 = \emptyset$,
3. $\forall X \in \mathbf{X}_1, X' \in \mathbf{X}_2$ *are separated by* $X_z \in \mathbf{Z}$.

Definition 3.1.5 (Conditional Independence of MRF[4][8]). *Conditional independence of MRF can be summarized into three properties:*

1. ***Local Markov Property.*** *A random variable X is conditionally independent from all other nodes except its neighbours when given its neighbour. The mathematical formalism is:*
$$X \perp\!\!\!\perp (\mathbf{X} \setminus (X \cup \mathbf{nei}(X))) | \mathbf{nei}(X).$$
2. ***Pairwise Markov Property.*** *For any two nodes X_i and X_j, $i \neq j$, are non-adjacent in an MRF, they are conditionally independent given all other nodes excluding themselves. The formalism is:*
$$X_i \perp\!\!\!\perp X_j | (\mathbf{X} \setminus X_i, X_j), i \neq j \text{ and } (X_i, X_j) \notin \mathbf{E}.$$
3. ***Global Markov Property.*** *Given any two sets $\mathbf{X}_1, \mathbf{X}_2 \subset \mathbf{X}$, if they are separated by $\mathbf{Z} \subset \mathbf{X}$, and all these sets are mutually disjoint, then $\mathbf{X}_2, \mathbf{X}_2$ are conditionally independent. This property can be formalized as:*
$$\mathbf{X}_1 \perp\!\!\!\perp \mathbf{X}_2 | \mathbf{Z},\ \mathbf{X}_1 \cap \mathbf{X}_2 = \emptyset,\ \mathbf{X}_1 \cap \mathbf{Z} = \emptyset,\ \text{and } \mathbf{X}_2 \cap \mathbf{Z} = \emptyset.$$

3.1.3 Learning

This section will roughly discuss the methods of constructing a model for a probability distribution. PGMs are used to model the joint distribution of data via combing both probability theory and graph theory. For a very simple probability distribution, a feasible way might construct a BN by hand since BNs are more accessible for its syntax and semantics. However, in the machine learning field, we are going to learn PGMs automatically, which means the learning process contains both learning the structure and learning the parameters by some algorithms. Generally, there are two common approaches in machine learning: discriminative learning and generative learning.

3.1.3.1 Generative Learning

Generative learning approach gives out the model by learning the process of generating the data. It tries to find the way how the data can be generated, which leads to model the joint probability distribution directly. In principle, the marginal and conditional probability can be derived from the joint probability distribution.

3.1.3.2 Discriminative Learning

Discriminative learning approach learns the joint distribution indirectly. In common situations, the data or samples used to learning PGMs are drawn from the real distribution which might not be modelled exactly. So, the discriminative learning approach focus on the final goal of modeling the joint distribution, which is to solve some practical problems such as classification.

3.1.3.3 Structure Learning and Parameter Learning

For learning PGMs, there are two key parts learned: structure and parameters. **Structure learning** is to learn the structure of RVs, which is to find the relations or associations between RVs. One approach to learn the structure is minimum description length(MDL) principle proposed in[9], which is to find the shortest and most compact representation of data $\mathcal{D}$.

Parameter learning, however, is to learn the weight or the values of RVs in the model. Maximum a-posterior(MAP) approach is a method used to learning parameters. MAP gives out the model according to the likelihood $\mathcal{L}$ of the model when given data $\mathcal{D}$.

If using $\mathcal{G}$ to represent the structure, Θ to denote the parameters, combining the learning approaches mentioned above, there will be various learning methods available, either learning the structure or the parameter via generative or discriminative, even hybrid approach.

3.1.4 Inference

After PGMs are constructed, the following process to perform reasoning on the model is called inference. This section will summarize the common inference scenarios[4].

3.1.4.1 Marginalization

Given certain query RVs $\mathbf{X}^q \subset \mathbf{X}$, calculate the marginal distribution $p(X^q)$, which is to marginalize $\mathbf{X} \setminus \mathbf{X}^q = \mathbf{X}^m = \{X_1^m, X_2^m, \ldots, X_K^m\}$. The mathematical formalism is:

$$p(\mathbf{X}^q) = \int_{\mathbf{val}(X_1^m)} \cdots \int_{\mathbf{val}(X_N^m)} p(\mathbf{X}^q, X_1^m, X_2^m, \ldots, X_K^m) \mathbf{d}X_1 \ldots \mathbf{d}X_K$$

for continuous RVs, and

$$p(\mathbf{X}^q) = \sum_{X_1^m}^{X_K^m} p(\mathbf{X}^q, X_1^m, X_2^m, \ldots, X_K^m)$$

for discrete RVs.

3.1.4.2 Conditions

Given RVs $\mathbf{X} = \mathbf{X}^{\mathbf{q}} \cup \mathbf{X}^o \cup \mathbf{X}^m$, the inference goal is to compute the posterior distribution of the query $\mathbf{X}^q$ conditioned on the observation $\mathbf{x}^o \in \mathbf{X}^o$. And the formalism is:

$$p(\mathbf{X}^q | \mathbf{x}^o) = \frac{p(\mathbf{X}^q, \mathbf{x}^o)}{p(\mathbf{x}^o)},$$

where $p(\mathbf{X}^q, \mathbf{x}^o)$ and $p(\mathbf{x}^o)$ are determined by the marginalization $\mathbf{X}^m$ and $\mathbf{X}^m \cup \mathbf{X}^q$.

3.1.4.3 Most Probable Explanation

MPE is a process to find the most probable explanation of query RVs $\mathbf{X}^q$ given the observed RVs $\mathbf{X}^o$, where $\mathbf{X} = \mathbf{X}^q \cup \mathbf{X}^o$. To find the MPE, we should compute:

$$\mathbf{x}^{q*} = \text{arg max}_{\mathbf{X}^q \in \mathbf{val}(\mathbf{X}^q)} p(\mathbf{x}^q | \mathbf{x}^o) = \text{arg max}_{\mathbf{X}^q \in \mathbf{val}(\mathbf{X}^q)} p(\mathbf{x}^q, \mathbf{x}^o).$$

3.1.4.4 Maximum A-Posterior

MAP is a process similar to MPE but ignores $\mathbf{X}^m$. The RVs $\mathbf{X}$ are still split into query RVs $\mathbf{X}^q$, observed RVs $\mathbf{X}^o$ and marginalized RVs $\mathbf{X}^m$. Here we will compute:

$$\begin{aligned}
\mathbf{x}^{q*} &= \text{arg max}_{\mathbf{X}^q \in \mathbf{val}(\mathbf{X}^q)} p(\mathbf{x}^q | \mathbf{x}^o) = \text{arg max}_{\mathbf{X}^q \in \mathbf{val}(\mathbf{X}^q)} p(\mathbf{x}^q, \mathbf{x}^o) \\
&= \text{arg max}_{\mathbf{X}^q \in \mathbf{val}(\mathbf{X}^q)} \int_{\mathbf{val}(X_1^m)} \cdots \int_{\mathbf{val}(X_K^m)} p(\mathbf{x}^q, x_1^m, x_2^m, \ldots, x_K^m) \mathbf{d}x_1 \ldots \mathbf{d}x_K \text{ (continuous)} \\
&= \sum_{\mathbf{x}_1^m}^{\mathbf{x}_K^m} p(\mathbf{x}^q, x_1^m, x_2^m, \ldots, x_K^m) \text{ (discrete)}.
\end{aligned} \tag{3.2}$$

These inferences are all NP-hard according to[8]. The goal of targeting the tractable models finally leads to the design of the algorithms adapted to the complexity of PGMs. One feasible solution is to relax the aim of exact inference which uses the factorization properties, and swift to model an approximate inference.

3.2 Weaknesses of Traditional PGMs

Traditional PGMs suffer from some weaknesses. These models separate the inference from learning, however, the inference is normally a sub-process of learning. This separation often causes the exact learning improper. But, using approximate learning to construct the PGMs has other disadvantages[4]:

1. Too many trials due to the unknown what kind of approximate learning is correct.
2. Even we find the correct approximate learning and the right algorithm, the results will still be unpredictable since the learning itself is not exact.
3. The approximate learning is a trade-off of time and accuracy. This approach turns the original difficulty to the difficulty of finding a good trade-off, which is still hard.

To avoid the problems caused by approximate learning, we should reconsider the exact learning and find the tractable models. However, in most time, a tractable traditional PGM is either sparsely connected or too simplistic which cannot give out a good representation of data.

3.3 Evidence and Network Polynomial

Definition 3.3.1 (Evidence[8])**.** *An evidence is defined as an instantiation* $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ *to the subset of RVs* $\mathbf{X}$.

If each $\mathbf{x} \in \mathbf{val}(\mathbf{X})$ is assigned a value, it is called complete evidence. Given a subset of $\mathbf{Y} \in \mathbf{X}$ and complete evidence $\mathbf{y}$, the probability of the partial evidence can be evaluated by marginalizing $\mathbf{Z} = \mathbf{X} \setminus \mathbf{Y}$. Typically, to evaluate the probability of evidence, is to compute $p(\mathbf{x})$.

In[10], network polynomials is introduced to describe the distribution over finite-state RVs and then generalized in[1] for unnormalized distributions.

Definition 3.3.2 (Network Polynomial)**.** *A network polynomial of a network* $\mathcal{N}$ *over RVs* $\mathbf{X}$ *is defined as a polynomial function mapping the network* $\mathcal{N}$ *to RVs* $\mathbf{X}$.

Given a finite-state RVs $\mathbf{X}$, we can define the indicator variable(IV) $\lambda_{\mathbf{X}=x} \in \mathbb{R}$ to represent the state of RVs. IVs can be assigned values for corresponding RVs' state. The vector form of IV is donated by $\boldsymbol{\lambda}$.

For Bayesian network, the network polynomials in Darwiche[10] are:

$$f_{\mathcal{B}}(\mathbf{x}) = \sum_{\mathbf{x}\in\mathbf{val}(\mathbf{X})} \prod_{X\in\mathbf{X}} \lambda_{\mathbf{X}=\mathbf{x}[X]} \theta^{\mathbf{X}}_{\mathbf{x}[X],[\mathbf{par}(X)]}$$

where $p_{\mathcal{B}}(\mathbf{x}) = \prod_{X\in\mathbf{X}} \theta^{\mathbf{X}}_{\mathbf{x}[X],[\mathbf{par}(X)]}$ is the probability distribution of BN.

For unnormalized distribution, the network polynomials in[1] are defined as:

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

where $\Phi(\mathbf{x})$ is the probability distribution with the conditions that $\forall\mathbf{x} \in \mathbf{val}(\mathbf{X}), \Phi(\mathbf{x}) \geq 0$ and $\exists\mathbf{x} \in \mathbf{val}(\mathbf{X}), \Phi(\mathbf{x}) > 0$.

Chapter 4 Sum-Product Network

In Section 3.2 of Chapter 3, the weakness of traditional PGMs were pointed out. A feasible solution is to design the models with less conditional independencies among RVs[4]. The mixture of distributions is proposed to support this model. In the mixture of distributions, RVs are augmented by marginalized latent RVs. Sum-Product Network(SPN) is such type of PGM. In this chapter, Section 4.1 introduces the arithmetic circuit, which is alike to SPM on the structure. Section 4.2 introduces the SPN. The key properties of SPN will be introduced in Section 4.3. Section 4.4 and Section 4.5 summarize the learning methods and the inference algorithm on SPN, respectively.

4.1 Arithmetic Circuit

Definition 4.1.1 (Arithmetic Circuit)**.** *An arithmetic circuit(AC) is a rooted acyclic directed graph with its numeric inputs and the internal arithmetic operation nodes. The arithmetic operations includes addition*$(+)$*, subtraction*$(-)$*, multiplication*$(\times)$*, and division*$(\div)$*. The output of the AC is computed in the root.*

Figure 4.1 is an example of AC with two variables A, B, and a constant 1. The network polynomial of this AC is:

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

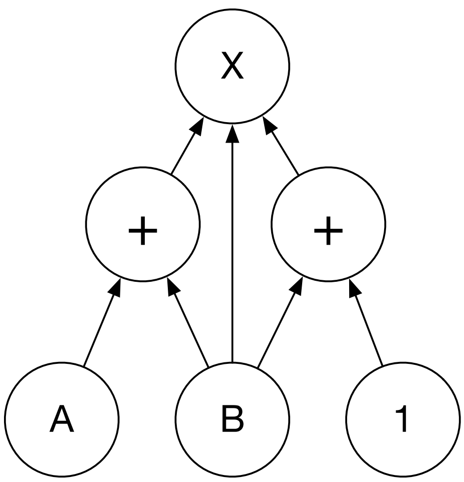


Figure 4.1: Example of an Arithmetic Circuit

4.2 Sum-product Network

In[1], Poon and Domingos proposed Sum-Product Network(SPN) as a new type of PGMs to overcome the weaknesses of the intractability of traditional PGMs. SPNs is an undirected acyclic graph over finite-state RVs with numeric inputs and the only two operations sum and product in the internal nodes.

Definition 4.2.1 (Sum-Product Network)**.** *A Sum-product network $\mathcal{S}$ over finite-state RVs X is a tuple $(\mathcal{G}, \Phi(\mathbf{X}))$ where $\mathcal{G}$ is a rooted DAG and $\Phi(\mathbf{X})$ is a set of non-negative parameters. In SPN,*

1. *leaves are numeric input represented by indicator variables,*
2. *internal nodes are sum nodes or product nodes which appear alternately,*
3. *root is always a sum node.*

The network polynomials of SPN are similar to that in AC. The form is:

$$f_{\mathcal{S}} = \sum_{X \in \mathbf{X}} \Phi(X) \prod_{X \in \mathbf{X}} (X).$$

Example Figure 4.2 is a simple example of SPN with three RVs A, B and their negations $\overline{A}$, and $\overline{B}$. With the parameters in the edges, the network polynomials is:

$$\begin{aligned} f_{\mathcal{S}}(A, B, \overline{A}, \overline{B}) =& 0.15(0.6A + 0.4\overline{A}) + 0.85(0.6A + 0.4\overline{A})(0.5B + 0.5\overline{B}) \\ =& 0.273AB + 0.327A\overline{B} + 0.182\overline{A}B + 0.218\overline{AB} \end{aligned} \tag{4.1}$$

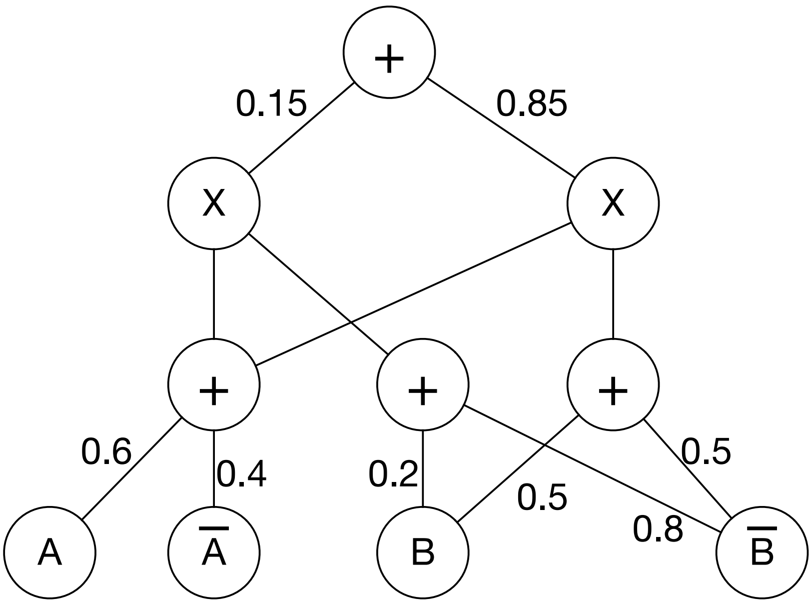


Figure 4.2: Example of a Sum-Product Network

4.3 Properties of SPN

Definition 4.3.1 (Scope of Node)**.** *Given a SPN $\mathcal{S} = (\mathcal{G}, \Phi(\mathbf{X}))$ over RVs $\mathbf{X}$, for a node $N \in \mathbf{N}$, the scope of node N is denoted as:*

$$\mathbf{sc}(N) \begin{cases} \{X\} & \textit{if } N \textit{ is a leaf,} \\ \cup_{\mathbf{C} \in \mathbf{chi}(N)} \mathbf{sc}(\mathbf{C}) & \textit{if } N \textit{ is an internal node.} \end{cases} \tag{4.2}$$

Theorem 4.3.1 (Completeness[4][1]). *A sum node* **S** *is complete if and only if all its children have the same scope. A sum-product network $\mathcal{S}$ is complete if and only if all its sum nodes are complete.*

Theorem 4.3.2 (Consistency[4][1]). *A product node* **P** *is consistent if and only if there is no negation of a variable in* **P** *appears in one child of another product node* $\mathbf{P}'$. *A sum-product network $\mathcal{S}$ is consistent if and only if all its product node are consistent.*

Theorem 4.3.3 (Validity[4][1]). *A sum-product network $\mathcal{S}$ is defined to be validate if and only it is both complete and consistent.*

Besides, a valid SPN will always compute the correct probability of evidence.

Theorem 4.3.4 (Decomposability[4][1]). *A product node* **P** *is decomposable if and only if there is no variables appear in more than one scope of its children. A sum-product network $\mathcal{S}$ is decomposable if and only if all its product nodes are decomposable.*

Completeness requires an SPN has the same scope for the children of the same sum node. Consistency requires that each variable and its negation should be in the same scope of a product node. Decomposability is to constrain the scopes of the children of the same product node no overlap.

Example Figure 4.3 is a valid and decomposable SPN.

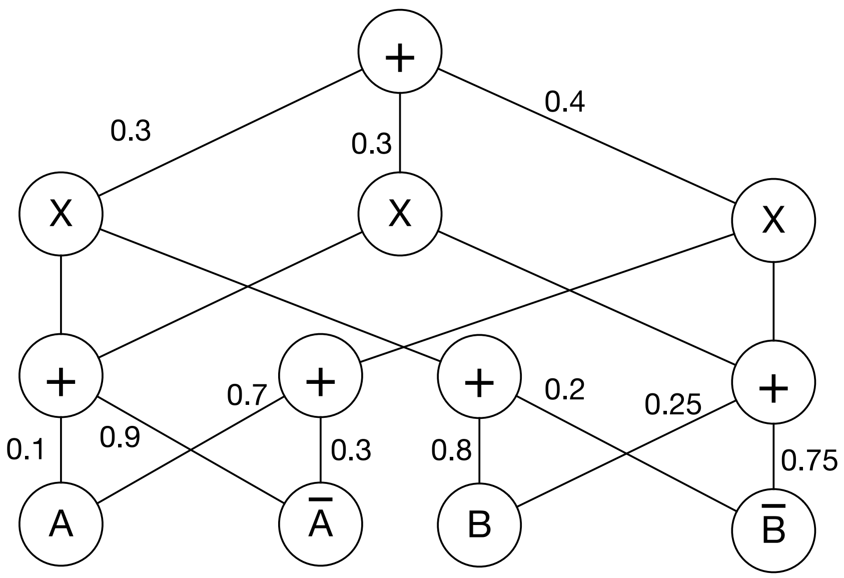


Figure 4.3: Example of a Valid and Decomposable SPN

4.4 Learning on SPN

SPN has its syntax and semantics largely differ from the traditional PGMs and makes itself feasible to incorporate the learning with inference, called inference-ware learning. In the case of inference-ware learning, the upper bound of learning is also the upper bound of the inference. This indicates that it is possible to get a model with an acceptable inference cost by a reasonable learning cost. Recall that in Section 4.3 only a valid SPN can output the correct probability, and a valid SPN is further constrained by the structure. However, without appropriate parameters, the model cannot obtain satisfying results. In this section, some learning methods on SPN for structure learning and parameter learning will be reviewed.

4.4.1 Structure Learning

Structure learning is to construct the relations or associations among numeric input, internal sum, and product nodes for further conduction of parameter learning. The structure of SPNs is various for the different domain-specific problems, this part will introduce the Poon-Domingos architecture for rectangle regions[1], Dennis-Venture architecture for non-rectangle regions[11], and a Top-down scheme[12].

4.4.1.1 Poon-Domingos Architecture

The Poon-Domingos architecture proposed in[1] constructs the SPN by arranging data in rectangles since the image data are matrices. First, a single sum node is assigned to the image as the root of the whole SPN. Then, split the overall image into all possible regions of two-subrectangles along two dimensions. For each region, keep the same number of sum nodes. Recursively repeat the second step to split more subrectangles in a smaller size. After that, all pairs of sum nodes from two sub-regions are connected as the children of a product node. Later, connect the product nodes as the children of the sum nodes and finally build an SPN. Apparently, the size of the SPN is related to the size of the image data. And therefore, the authors used a coarser way for aggregations to lower the computation. The final structure of the SPN is a dense structure.

Figure 4.4 shows the architecture of Poon-Domingos architecture.

4.4.1.2 Dennis-Venture Architecture

The Dennis-Venture architecture[11] is to extend the Poon-Domingos architecture for non-rectangle data. The regions in this architecture are found via k-mean clustering, which shifts the original shape-driven method to the data-driven approach. By clustering methods, the data are no longer limited to the rectangle shape.

4.4.2 Parameter Learning

Given an SPN with a fixed structure, the next step is to learn the parameters or weights. This section will introduce two common approaches for parameter learning.

4.4.2.1 Gradient Methods

The first approach is the gradient method[4]. The derivatives of the SPN can be obtained from the network polynomials.

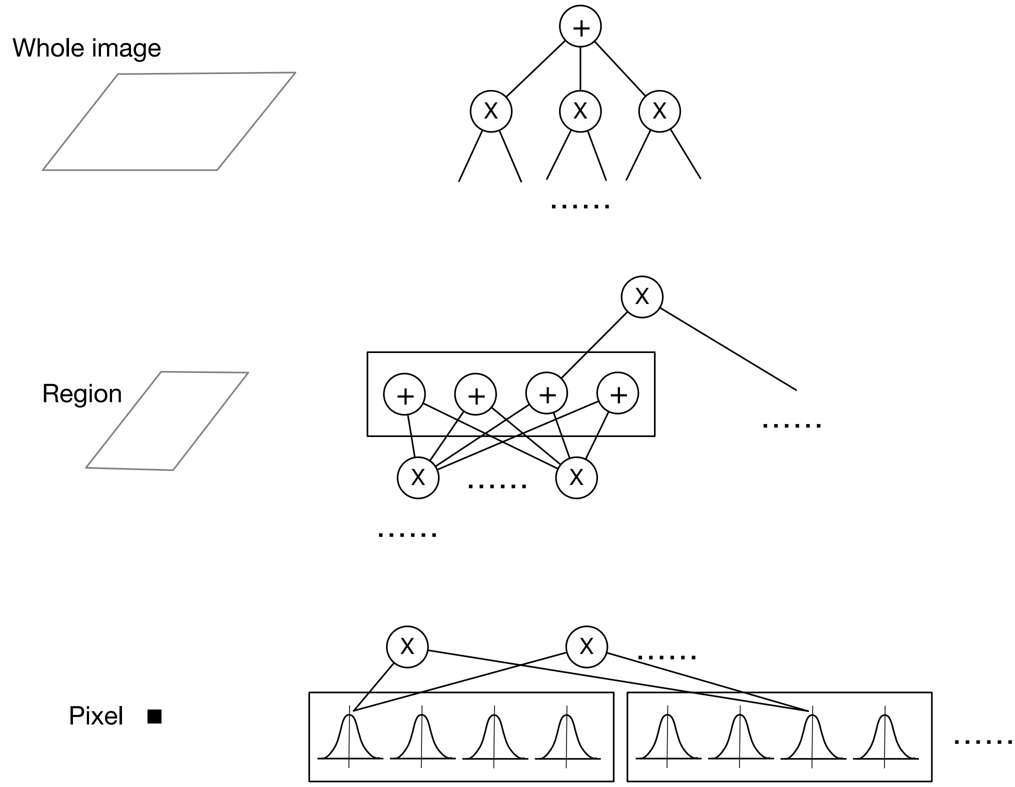


Figure 4.4: Poon-Domingos Architecture

Given an SPN $\mathcal{S} = (\mathcal{G}, \Phi(\mathbf{X}))$ over RVs $\mathbf{X}$ and a dataset $\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\}$ of L i.i.d. samples. The goal is to maximize the log-likelihood $\log \mathcal{L}$:

$$\max \quad \log \mathcal{L} = \sum_{l=1}^{L} \mathcal{S}(\mathbf{x}^{(l)}) \tag{4.3}$$

$$\text{s.t.} \quad \sum_{\mathbf{C} \in \mathbf{chi}(\mathbf{S})} \phi_{\mathbf{S},\mathbf{C}} = 1, \forall \mathbf{S} \tag{4.4}$$

$$\phi_{\mathbf{S},\mathbf{C}} \geq 0, \forall \mathbf{S}, \mathbf{C} \in \mathbf{chi}(\mathbf{S}). \tag{4.5}$$

And for any node $\mathbf{N}$, the derivative of the log-likelihood of the l^{th} is:

$$\frac{\partial \log \mathcal{S}}{\partial \mathbf{N}}(\mathbf{x}^{(l)}) = \frac{1}{\mathcal{S}(\mathbf{x}^{(l)})} \frac{\partial \mathcal{S}}{\partial \mathbf{N}}(\mathbf{x}^{(l)}) \tag{4.6}$$

where the derivative $\frac{\partial \mathcal{S}}{\partial \mathbf{N}}$ can be computed via backpropagation.

The derivative w.r.t the sum weights is:

$$\frac{\partial \log \mathcal{S}}{\partial \phi_{\mathbf{S},\mathbf{C}}}(\mathbf{x}^{(l)}) = \frac{\partial \log \mathcal{S}}{\partial \mathbf{S}}(\mathbf{x}^{(l)})\mathbf{C}(\mathbf{x}^{(l)}) = \frac{1}{\mathcal{S}(\mathbf{x}^{(l)})} \frac{\partial \mathcal{S}}{\partial \mathbf{S}}(\mathbf{x}^{(l)})\mathbf{C}(\mathbf{x}^{(l)}) \tag{4.7}$$

and the derivative of the log-likelihood of Equation 4.3 is:

$$\frac{\partial \log \mathcal{L}}{\partial \phi_{\mathbf{S},\mathbf{C}}} = \sum_{l=1}^{L} \frac{\partial \log \mathcal{S}}{\partial \phi_{\mathbf{S},\mathbf{C}}}(\mathbf{x}^{(l)}). \tag{4.8}$$

At last, using a appropriate step size η, we can update the parameters by:

$$\phi \leftarrow \phi + \eta \nabla \log \mathcal{L}. \tag{4.9}$$

4.4.2.2 EM Algorithm

If interpreting the SPN as a latent RVs model, researchers can use the EM algorithm for parameter learning. In this thesis, we do not review the latent RVs interpretation of SPN so we will not go into the detail of the foundations of the EM process on SPN. In[1], Poon, and Domingos first proposed the EM algorithm on SPN but later in[4], Peharz found it was wrong and refined it[4].

Algorithme 1 EM for SPN Weights

Input: An SPN $\mathcal{S}$ with fixed structure
Output: An SPN $\mathcal{S}$ with fixed structure and proper parameters.

```
Initialize Φ
while not converged do
    ∀S ∈ 𝒮, ∀C ∈ chi(S) : n_{S,C} ← 0
    for l = 1, ..., L do
        Input x^(l) to 𝒮
        Evaluate 𝒮 (upward-pass)
        Backprop 𝒮 (backward-pass)
        ∀S ∈ 𝒮, ∀C ∈ chi(S) : n_{S,C} ← n_{S,C} + (1/𝒮)(∂𝒮/∂S) C φ_{S,C}
    end for
    ∀S ∈ 𝒮, ∀C ∈ chi(S) : n_{S,C} ← n_{S,C} / Σ_{C'∈ch(S)} n_{S,C'}
end while
return 𝒮
```

4.5 Inference on SPN

If setting the indicator variables(IVs), people can compute the marginalization via network polynomials, then further interpret the probability distribution through the derivatives of the network polynomials. This process is called the differential approach of inference.

Given an SPN $\mathcal{S}$ with a node $\mathbf{N}$, there two cases.

1. $\mathbf{N}$ is the root, then the derivative is:

$$\frac{\partial \mathcal{S}}{\partial \mathbf{N}}(\boldsymbol{\lambda}) = 1$$

.

2. $\mathbf{N}$ is an internal node, then the derivative should be

$$\frac{\partial \mathcal{S}}{\partial \mathbf{N}}(\boldsymbol{\lambda}) = \sum_{\mathbf{F} \in \mathbf{pa}(\mathbf{N})} \frac{\partial \mathcal{S}}{\partial \mathbf{F}} \frac{\partial \mathbf{F}}{\partial \mathbf{N}}(\boldsymbol{\lambda})$$

If **F** is a sum node, the derivative is:

$$\mathbf{F} = \sum_{\mathbf{C} \in \mathbf{pa}(\mathbf{F})} \phi_{\mathbf{F},\mathbf{C}} \mathbf{C}(\boldsymbol{\lambda}),$$

which is

$$\frac{\partial \mathbf{F}}{\partial \mathbf{N}}(\boldsymbol{\lambda}) = \phi_{\mathbf{F},\mathbf{N}}.$$

If **F** is a product node, the derivative is:

$$\mathbf{F} = \prod_{\mathbf{C} \in \mathbf{pa}(\mathbf{F})} \mathbf{C}(\boldsymbol{\lambda}),$$

which is

$$\frac{\partial \mathbf{F}}{\partial \mathbf{N}}(\boldsymbol{\lambda}) = \prod_{\mathbf{C} \in \mathbf{pa}(\mathbf{F}) \setminus \{\mathbf{N}\}} \mathbf{C}(\boldsymbol{\lambda}).$$

Usually, given numeric input, the probability is evaluated in the bottom-up direction, sorting values for each node along the process.

Chapter 5 SPN for Image Completion

In Chapter 4, I reviewed the details of SPN from its definition to the learning and inference methods. SPN has been widely used in various applications: image completions[1], activity recognition[13], language modelling[14], speech modelling[15], facial attributes analysis[16], robot control[17], and other reasoning and inference scenarios. This chapter will move to one of the application this project focus to reproduce: image completions. In Section 5.1, I will briefly introduce the dataset involved in this experiment. Section 5.2 will introduce the program from the code structure and the roles of the modules with the detailed document. Section 5.3 describes the details of this experiment. The results of the experiment will be presented in 5.4. Last Section 5.5 will analyze the results and compared to original results.

5.1 Dataset

Two datasets are used in this experiment: Caltech and Olivetti.

- Caltech: Caltech [1] is a dataset containing the pictures of 101 objects. For each category, there are 40 to 800 images. Most of the categories have more than 50 images. The size of the image is about 300×200 pixels. In this experiment, the dataset is rescaled to 100×64 pixels.

- Olivetti: Olivetti [2] is a dataset containing face images taken between April 1992 and April 1994 at AT&T Laboratories Cambridge with each image in size 64×64.

5.2 Program

5.2.1 Source Code

The experiment program is referenced from the Java program provided by authors on their website [3]. In our experiment program, there are still three modules: common, evaluation and spn. The code is on Github [4].

- **common**: Contain some helper functions to provide time management, messaging between progress(MPI), parameter settings for training on clusters, and some utilities.

- **evaluation**: Process the dataset, apply the network to the dataset to output models, and evaluate the results generated from the models.

[1] Caltech: http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[2] Olivetti: https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

[3] http://spn.cs.washington.edu/spn/

[4] SPN-and-Image-Completion: https://github.com/Spacebody/SPN-and-Image-Completion

- **spn**: Contain the SPN architecture, including node definition, computation functions, learning, and inference.

For each modules, we provides details document for every source code file. (*.cpp* files are implementations while *.hpp* files store function declarations and class definition.)

- **common**

  1. *my_mpi*: Use OpenMPI to support the messaging in a parallel program. It means that this program will use parallel architecture to accelerate computing.
  2. *parameter*: Control parameters for EM algorithm, SPN, and evaluation.
  3. *timer*: Manage the time to help calculate the time spent on computation.
  4. *utils*: Some helper functions to access time, print log, and do some numeric process.

- **evaluation**

  1. *dataset*: Read and process data from the dataset.
  2. *eval*: Conduct evaluation over the dataset.
  3. *image_completion*: Conduct image completion, which is the application.
  4. *run*: Control the program, which is the main function.

- **spn**

  1. *decomposition*: Decompose the regions.
  2. *generative_learning*: Conduct the learning process to generative the model.
  3. *instance*: Record the mean and variance of an instance, which is calculated from the dataset.
  4. *node*: Define the node, to provide the base class of the sum node and the product node.
  5. *prod_node*: Define the product node, derived from *node*.
  6. *sum_node*: Define the sum node, derived from *node*.
  7. *region*: Compute the mean and variance of regions in the picture(for this image completion application), as well as the MAP.
  8. *SPN*: Define the Sum-product network, including a root, functions of learning and applications. These functions are implemented via calling other modules.

5.2.2 Callgraph

The following picture shows the call graph of this program. The program will start from *run*, which reads the arguments from the command line containing the domain we choose, then the program calls the corresponding processes for a specific dataset, the processes including reading data from data folder(*dataset*), setting instance one by one, learning to construct SPN structure(*generative_learning* and *SPN*), performing inference on the SPN(*SPN*), completing images(*image_completion*), and storing models and complete results to folders(*SPN* and *dataset*).

The callgraph is presented in Figure 5.1

5.3 Experiment

This section will introduce the environment and the process of this experiment.

5.3.1 Software

The software used in this program is OpenMPI, which is an open source message passing library. In this program, slaves learn and update the structure and parameters, then send to the master for aggregation. The master will use buffer to collect data and pass received from slaves.

5.3.2 Hardware

The hardware I used is the TaiYi supercomputer platform. Two datasets are trained with various cores. In the authors' case, dataset *Caltech* was learned on 102 cores, 2 groups, and 50 cores per group. While in TaiYi, I used 80 cores, 2 groups, 39 cores for slaves and 1 core for the master per group. Dataset *Olivetti* was learned with 51 cores in the original paper, while in my case, 40 cores were used.

5.3.3 Process

The process of the experiment can be summarized into three steps.

1. Compile *run* via *Makefile* on compilation node,
2. Run program on platform,
3. Move data to local,
4. Compile *eval* and evaluate the output.

Four experiments were conducted.

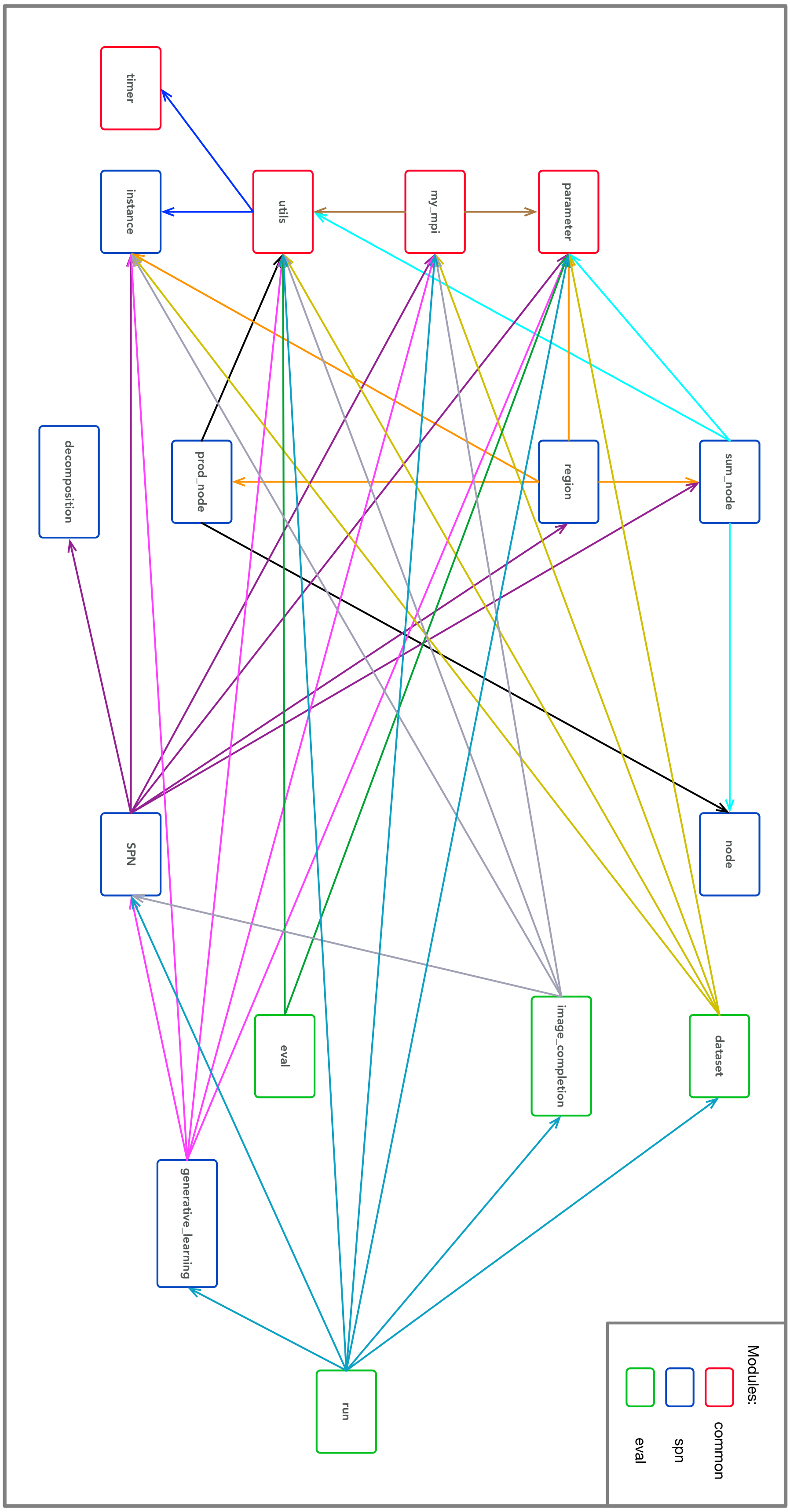


Figure 5.1: Callgraph of Program

1. Experiment #1: Caltech: 80 cores with input size 64×64, Olivetti: 40 cores with input size 64×64.

2. Experiment #2: Caltech: 120 cores with input size 64×64, Olivetti: 80 cores with input size 64×64.

3. Experiment #3: Caltech: 80 cores with input size 100×64.

4. Experiment #4: Caltech: 120 cores with input size 100×64.

Since the number of cores allocated on TaiYi can only be a multiple of 40, so the number of cores should be considered a factor to influence the results. Besides, I noticed that for *Caltech*, Poon used 64×64 as the input size while the image was scaled to 100×64, so I considered that the input size of Caltech data might be another factor. So, two factors were considered to be possible to influence the results: *core number used to training*, *input size of data*. Experiment #1 and #2 were to exam the first factor, and Experiment #2 and Experiment #4 plusing Experiment #1 and Experiment #3 were to check the second factor.

5.4 Results

The section will show a part of the evaluation. The complete results can be referred to Section A.1, A.2, A.3, and A.4 in Appendix A.

5.4.1 Experiment #1

Caltech-101: 80 cores, input size: 64×64;

Olivetti: 40 cores, input size: 64×64.

Table 5.1: MSE on LEFT and BOTTOM

Category	MSE on LEFT	MSE on BOTTOM
Caltech(ALL)	5464	5406
Face	3733	3855
helicopter	3881	4553
dolphin	4570	4888
Olivetti	1084	1156

5.4.2 Experiment #2

Caltech-101: 120 cores, input size: 64×64;

Olivetti: 80 cores, input size: 64×64.

Table 5.2: MSE on LEFT and BOTTOM

Category	MSE on LEFT	MSE on BOTTOM
Caltech(ALL)	5371	5344
Face	3789	3971
helicopter	3552	4150
dolphin	4604	4812
Olivetti	1084	1156

5.4.3 Experiment #3

Caltech-101: 80 cores, input size: 100×64.

Table 5.3: MSE on LEFT and BOTTOM

Category	MSE on LEFT	MSE on BOTTOM
Caltech(ALL)	5595	2789
Faces	1268	1045
helicopter	2338	1179
dolphin	4788	2211

5.4.4 Experiment #4

Caltech-101: 120 cores, input size: 100×64.

Table 5.4: MSE on LEFT and BOTTOM

Category	MSE on LEFT	MSE on BOTTOM
Caltech(ALL)	5587	2775
Face	1268	1045
helicopter	2316	1185
dolphin	4700	2174

5.5 Analysis

This section will analysis the results and compapre my results with original results presented in[1].

5.5.1 Factors

In Section 5.3, two factors, the number of cores and the input size, were considered to be influential.

Figure 5.2 shows the comparison of the results from Experiment #1 and Experiment #2. The x-axis is the results from Experiment #1, while the y-axis is the results from Experiment #2. The top two pictures are about MSE on LEFT, while the bottom two pictures are about MSE on BOTTOM. The red lines in the right picture is a diagonal line which shows the performance of both results. The MSE of a category goes larger as the point($+$) get closer to the axis. From the comparison, it is obvious that the number of cores involved in the process is not a significant factor.

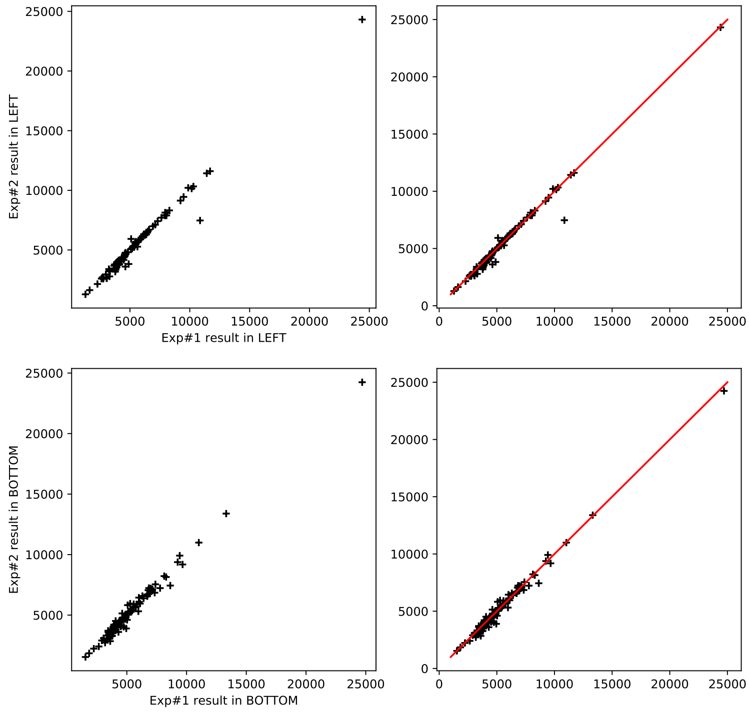


Figure 5.2: Scatter of Caltech MSE in Experiment #1 and #2

Similarly, compare the results of Experiment #2 and #4(Figure 5.3), Experiment #1 and #3(Figure 5.4), the increment of input size can reduce the converged MSE.

5.5.2 Performance

Most of the results in Experiment #1 in Section 5.4.1 is worse than the Poon's results, however, on some categories, I achieved the lower MSE. The comparison of some categories are shown in Table 5.5 and Table 5.6.

My results achieved lower LEFT MSE on *bonsai*, *mandolin*, *wrench*, and lower BOTTOM MSE on *garfield*, *wrench*. Most categories are in an acceptable difference. But one of the points probably worth mentioning is that on *yin_yang*, my results have a very large gap between to Poon's results.

Figure 5.5 shows the scatter of both results in all categories. The x-axis is my results, the y-axis is Poon's results. The top two pictures are about LEFT MSE, while the bottom two pictures are about BOTTOM MSE. The red lines in the right half two picture is a diagonal

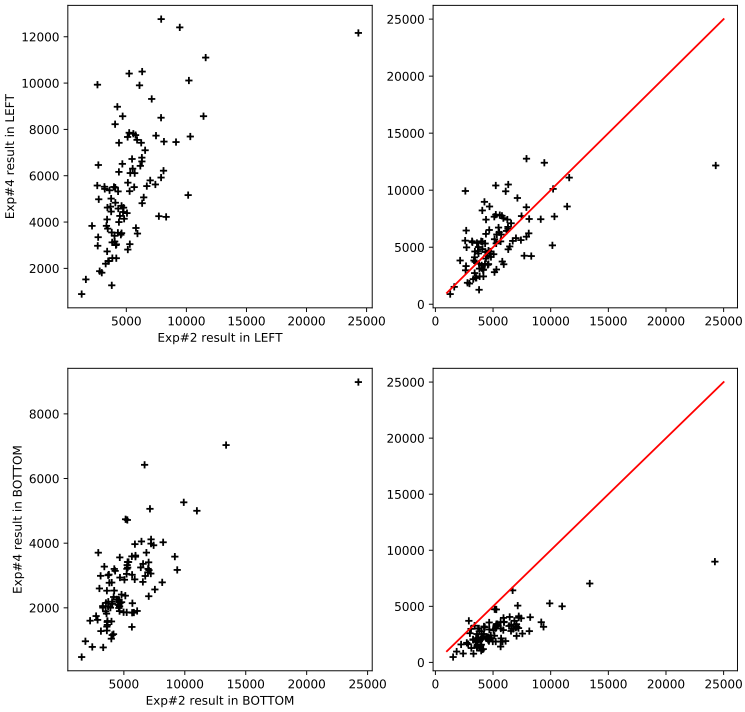


Figure 5.3: Scatter of Caltech MSE in Experiment #2 and #4

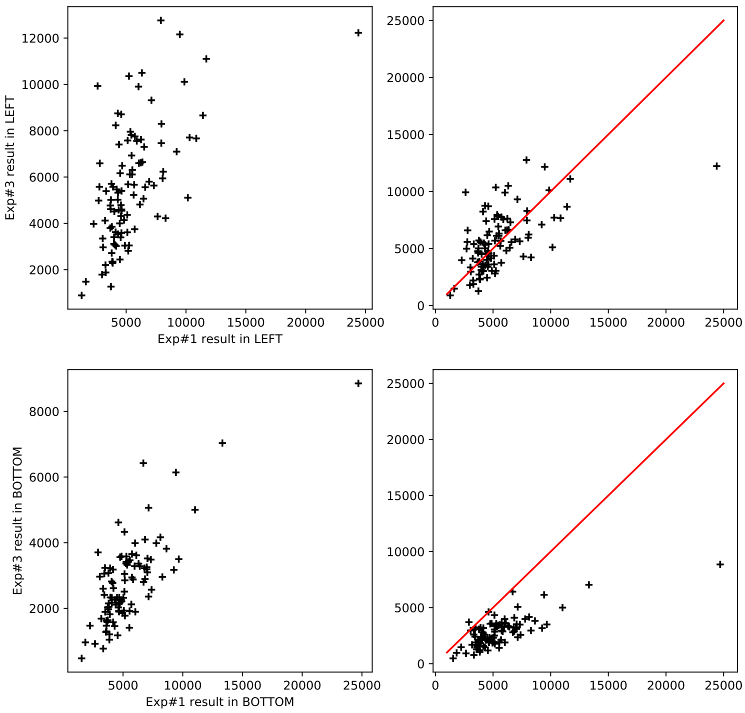


Figure 5.4: Scatter of Caltech MSE in Experiment #1 and #3

Table 5.5: Comparsion on LEFT MSE

Category	MSE on LEFT(Exp #1)	MSE on LEFT(Poon)
Caltech(ALL)	5464	3475
Faces	3733	3416
Leopards	1634	1541
Motorbikes	3871	2581
Bonsai	**4499**	4545
Mandolin	**4061**	4239
Wrench	**3673**	3861
Yin_yang	24402	5222

Table 5.6: Comparsion on BOTTOM MSE

Category	MSE on BOTTOM(Exp #1)	MSE on BOTTOM(Poon)
Caltech(ALL)	5406	3538
Faces	3855	3555
Leopards	1844	1498
Motorbikes	4848	3626
Garfield	**3490**	3583
Wrench	**3673**	3975
yin_yang	24701	5394

line which shows the performance of both results. The MSE of a category is larger as the point(+) get closer to the axis.

Similarly, my results in Experiment #2 has the same performance as Experiment #1 since changing the number of cores cannot improvement the results(Figure 5.6). The details can be referred to Section B.1 in Appendix B.

However, if increase the input size, more categories can achieve a lower MSE. The Figure 5.7 is the scatter of the results from Experiment #3 and Poon's, and Figure 5.8 is the scatter of the results from Experiment #4 and Poon's. Both figures show improved performance, especially on BOTTOM MSE. The details can be referred to Section B.2.

Figure 5.9 shows the comparison of original images, Poon's image, and my images. In the figure, the first column is the comparison of original images and Poon's images, the second column is the comparison of original images and Experiment #2's images, and the third column is the comparison of original images and Experiment #4's images. From the intuitive comparison, my performance of the results of both Experiment #2 and #4 is little worse than Poon's.

The main reason for the difference between both results is the input size of image data. The larger size gives more information about the data and makes a more precise output. Another reason might be the characters of this architecture. The random number is used in the program to randomly break the ties between nodes so as to help the MSE converge. The maximal iteration for training, in Poon's code, is 30, while I tested and found the maximal iteration cannot help the results converge, so I finally set to 1000 and the results can con-

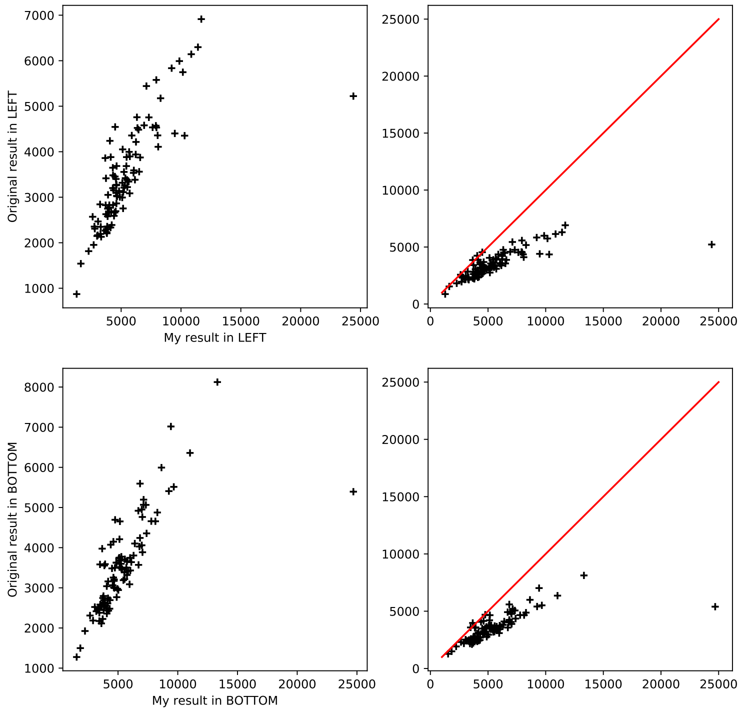


Figure 5.5: Scatter of Caltech MSE in Experiment #1 and Poon's

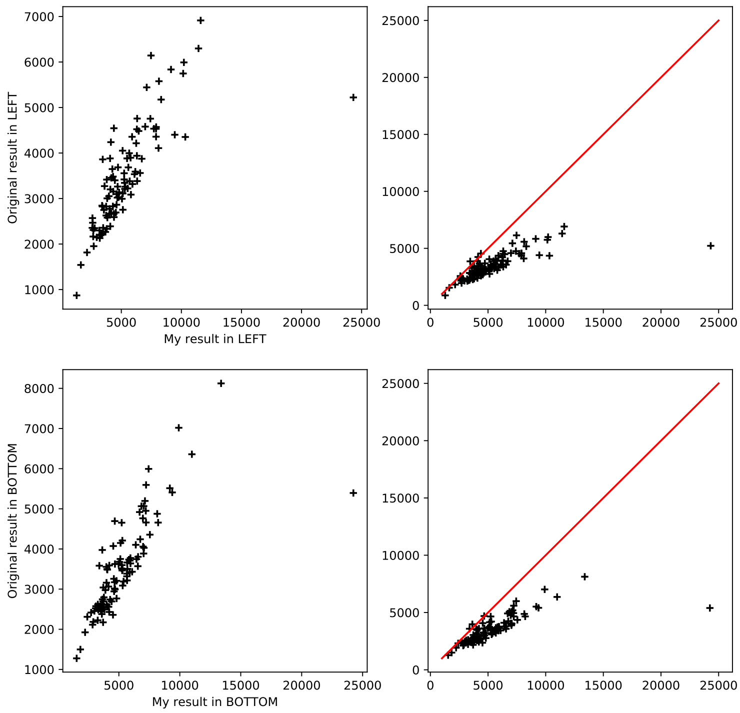


Figure 5.6: Scatter of Caltech MSE in Experiment #2 and Poon's

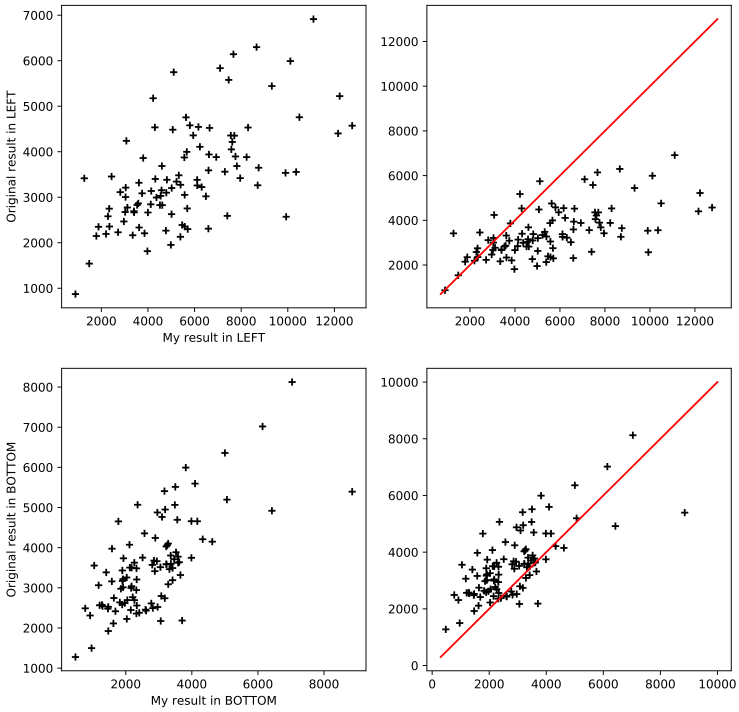


Figure 5.7: Scatter of Caltech MSE in Experiment #3 and Poon's

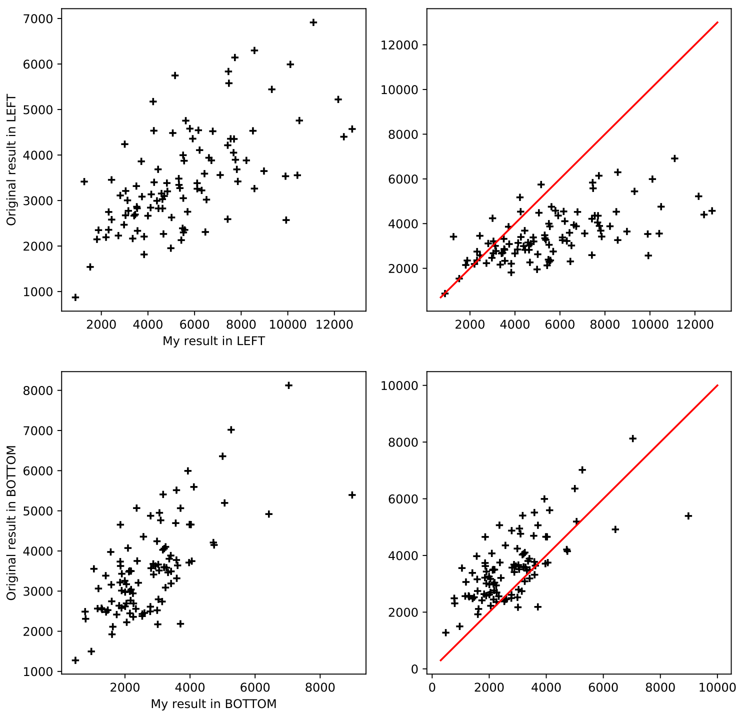
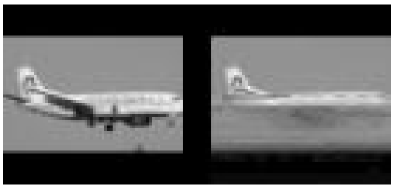
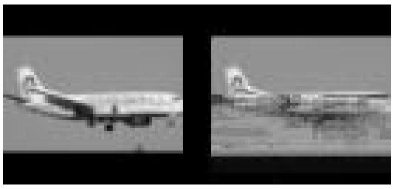


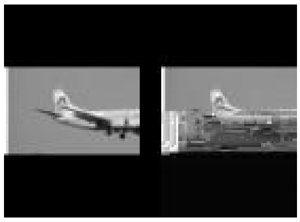
Figure 5.8: Scatter of Caltech MSE in Experiment #4 and Poon's



(a) Airplanes-left(Poon)



(b) Airplanes-left(Exp. #2)

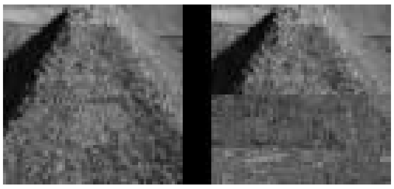


(c) Airplanes-left(Exp. #4)

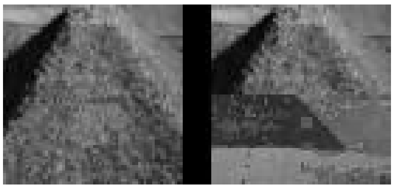
(d) Face_easy-left(Poon)

(e) Face_easy-left(Exp. #2)

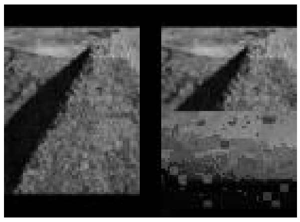
(f) Face_easy-left(Exp. #4)



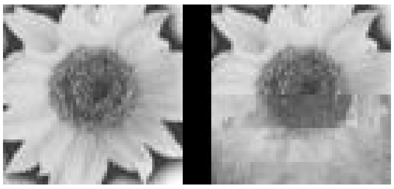
(g) Pyramid-bottom(Poon)



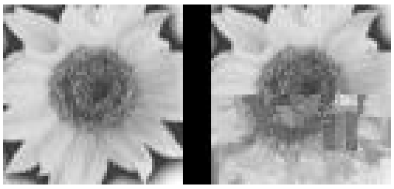
(h) Pyramid-bottom(Exp. #2)



(i) Pyramid-bottom(Exp. #4)

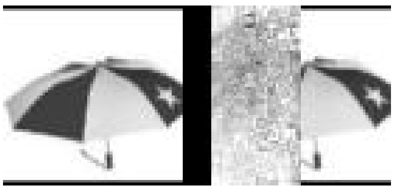


(j) Sunflower-bottom(Poon)

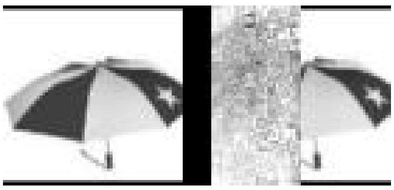


(k) Sunflower-bottom(Exp. #2)

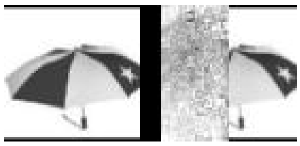
(l) Sunflower-bottom(Exp. #4)



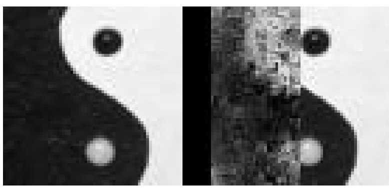
(m) Umbrella-left(Poon)



(n) Umbrella-left(Exp. #2)

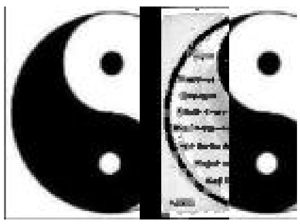


(o) Umbrella-left(Exp. #4)



(p) Yin_yang-left(Poon)

(q) Yin_yang-left(Exp. #2)



(r) Yin_yang-left(Exp. #4)

Figure 5.9: Comparsion of Original Images, Poon's Images and My Images

verge. My results didn't achieve the same MSE in many categories. The reason I think is the randomness existing in this architecture. However, I thought the difference is acceptable if dividing the total MSE to each item for most categories.

5.5.3 Time

Table 5.7 shows the time cost of Poon's experiments and my experiments.

Table 5.7: Time Cost of Poon's Experiments and My Experiments

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

Compare to Poon's experiments, my experiments cost for more time due to multiple reasons. One reason might be the diversity in the programming language and the MPI library they used. The message will be copied from the buffer in my implementation, which may need some time. The second reason might be the number of cores they allocated. Although the number of cores has no effect on the performance but will affect the time used for synchronization. The third reason is the difference in input size of image data. As the input size goes larger, the network becomes more complex and the learning and inference take more time.

Chapter 6 Discussion and Future Work

This chapter will give some discussion on this thesis topic and feasible future work.

6.1 Discussion

This thesis topic is to implement Poon's architecture for SPN. Before introducing my work, I reviewed the related theory and traditional PGMs. Later I gave a detailed introduction of SPN including the inference and learning methods. I re-implemented the Poon's architecture and conducted the experiments on the same dataset, compared the results and analyzed the reason. Although the reproduced results are not as good as the original results, through comparison of the evaluation and the completed images, I validated that the implementation is valid and correct. This work is the headstone for further research about SPN.

I learned a new PGM with the related technology, more about the probability theory and graph theory, the process to reproduce research results, and how to write a comprehensive paper to present my work. From this process, I also realized that reproducing research results is not an easy thing since to grap something unfamiliar is hard at first and need to keep more patient along the way. The most important thing is that I harvest the experience of conducting a research topic which will give me inspiration and guidance in the future.

6.2 Future Work

There are some future work can be done to extend this research.

The first one is to modify Poon's architecture to improve performance. Poon's architecture is suitable to handle the rectangle data, like images. The method to cluster on the data is from two dimensions of the rectangle data, which is a little intuitive. There can also be some other methods to cluster on the data from a more reasonable way, such as k-means cluster, hierarchical cluster, and so on. Second, there are some other applications of SPN can be explored, such as semantic analysis, text analysis, and medical diagnostics. Essentially, almost all learning and inference problems used to resolved via traditional PGMs can also be resolved by SPN. The last direction is to develop new algorithms for learning and inference. A more efficient algorithm can enable the model to output better results.

Conclusions

This work implemented Poon's architecture for SPN and reproduced the image completion experiment to validate the implementation. The results were compared and analyzed to attest the conclusion. Besides, this thesis also gave a comprehensive review of related theory and techniques. In the final, I also summarized the thesis and gave some possible future work to extend this work.

References

[1] POON H, DOMINGOS P. Sum-product networks: A new deep architecture[C]//2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops). [S.l.]: IEEE, 2011: 689-690.

[2] TAO T. An introduction to measure theory: volume 126[M]. [S.l.]: American Mathematical Society Providence, 2011.

[3] KLENKE A. Probability theory: a comprehensive course[M]. [S.l.]: Springer Science & Business Media, 2013.

[4] PEHARZ R. Foundations of sum-product networks for probabilistic modeling[D]. [S.l.]: PhD thesis, Medical University of Graz, 2015.

[5] BONDY J A, MURTY U S R. Graph theory: volume 290[M]. [S.l.]: Springer, 2008.

[6] DIESTEL R. Graduate texts in mathematics[M]. [S.l.]: Springer-Verlag New York, Incorporated, 2000.

[7] LUCE R D, PERRY A D. A method of matrix analysis of group structure[J]. Psychometrika, 1949, 14(2):95-116.

[8] KOLLER D, FRIEDMAN N, BACH F. Probabilistic graphical models: principles and techniques [M]. [S.l.]: MIT press, 2009.

[9] RISSANEN J. Modeling by shortest data description[J]. Automatica, 1978, 14(5):465-471.

[10] DARWICHE A. A differential approach to inference in bayesian networks[J]. Journal of the ACM (JACM), 2003, 50(3):280-305.

[11] DENNIS A, VENTURA D. Learning the architecture of sum-product networks using clustering on variables[C]//Advances in Neural Information Processing Systems. [S.l.: s.n.], 2012: 2033-2041.

[12] GENS R, PEDRO D. Learning the structure of sum-product networks[C]//International conference on machine learning. [S.l.: s.n.], 2013: 873-880.

[13] AMER M R, TODOROVIC S. Sum product networks for activity recognition[J]. IEEE transactions on pattern analysis and machine intelligence, 2016, 38(4):800-813.

[14] CHENG W C, KOK S, PHAM H V, et al. Language modeling with sum-product networks[C]// Fifteenth Annual Conference of the International Speech Communication Association. [S.l.: s.n.], 2014.

[15] PEHARZ R, KAPELLER G, MOWLAEE P, et al. Modeling speech with sum-product networks: Application to bandwidth extension[C]//2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). [S.l.]: IEEE, 2014: 3699-3703.

[16] LUO P, WANG X, TANG X. A deep sum-product architecture for robust facial attributes analysis [C]//Proceedings of the IEEE International Conference on Computer Vision. [S.l.: s.n.], 2013: 2864-2871.

[17] PRONOBIS A, RICCIO F, RAO R P. Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments[C]//ICAPS 2017 Workshop on Planning and Robotics. [S.l.: s.n.], 2017.

[18] PEHARZ R, GEIGER B C, PERNKOPF F. Greedy part-wise learning of sum-product networks[C]// Joint European Conference on Machine Learning and Knowledge Discovery in Databases. [S.l.]: Springer, 2013: 612-627.

[19] ROOSHENAS A, LOWD D. Learning sum-product networks with direct and indirect variable interactions[C]//International Conference on Machine Learning. [S.l.: s.n.], 2014: 710-718.

Appendix A Experiment Results

A.1 Experiment #1

Table A.1: Complete Results of Experiment #1

Category	MSE on LEFT	MSE on Bottom
Caltech(ALL)	5464	5406
Faces	3733	3855
Faces_easy	2284	2234
Leopards	1634	1844
Motorbikes	3871	4848
accordion	7629	8291
airplanes	1284	1537
anchor	5517	3940
ant	4569	4664
barrel	5237	5248
bass	5668	5112
beaver	5245	5694
binocular	6319	7141
bonsai	4499	4755
brain	2799	3057
brontosaurus	6240	6988
buddha	3240	3727
butterfly	4621	4390
camera	7902	6700
cannon	4325	3617
car_side	3833	2654
ceiling_fan	4546	4688
cellphone	9221	9667
chair	7950	8128
chandelier	2705	3507
cougar_body	6153	6718
cougar_face	4666	5749
crab	4605	4645
crayfish	3677	3435
crocodile	2988	3341

Table A.1 continued from previous page

Category	MSE on LEFT	MSE on Bottom
crocodile_head	3275	3608
cup	5434	5278
dalmatian	10156	9255
dollar_bill	4164	3863
dolphin	4570	4888
dragonfly	4588	3780
electric_guitar	8295	7139
elephant	3732	4321
emu	5162	4133
euphonium	6506	5748
ewer	5099	5070
ferry	5714	6029
flamingo	4378	3796
flamingo_head	4321	5125
garfield	4131	3490
gerenuk	4214	3433
gramophone	6062	6008
grand_piano	10858	8636
hawksbill	3297	3568
headphone	11697	13317
hedgehog	4628	4045
helicopter	3881	4553
ibis	3795	4299
inline_skate	7937	6842
joshua_tree	5500	5349
kangaroo	3042	3608
ketch	3775	4739
lamp	6590	6404
laptop	5127	4611
llama	3707	3783
lobster	6921	7035
lotus	4465	3898
mandolin	4061	5160
mayfly	3975	5548
menorah	4618	4057

Table A.1 continued from previous page

Category	MSE on LEFT	MSE on Bottom
metronome	9484	5575
minaret	4482	4170
nautilus	6451	7390
octopus	5355	4878
okapi	4404	4161
pagoda	10302	6794
panda	11417	9427
pigeon	4825	6302
pizza	5311	5449
platypus	8096	6839
pyramid	4120	4063
revolver	5628	3814
rhino	3988	5966
rooster	5730	6110
saxophone	6217	5840
schooner	4011	4680
scissors	3330	4198
scorpion	4896	4952
sea_horse	5092	5025
snoopy	7112	7783
soccer_ball	9866	11026
stapler	7317	6986
starfish	3811	3718
stegosaurus	8050	7047
stop_sign	4304	3919
strawberry	2620	2908
sunflower	2766	3330
tick	5457	5151
trilobite	3071	3179
umbrella	5889	5122
watch	4303	4495
water_lilly	5174	5529
wheelchair	6035	5271
wild_cat	6362	7331
windsor_chair	3904	4614

Table A.1 continued from previous page

Category	MSE on LEFT	MSE on Bottom
wrench	3673	3673
yin_yang	24402	24701
Olivetti	1084	1156

A.2 Experiment #2

Table A.2: Complete Results of Experiment #2

Category	MSE on LEFT	MSE on Bottom
Caltech(ALL)	5371	5344
Faces	3789	3971
Faces_easy	2146	2229
Leopards	1634	1844
Motorbikes	3838	4977
accordion	7709	8146
airplanes	1284	1537
anchor	5529	4162
ant	4464	4650
barrel	5237	5248
bass	5657	5119
beaver	5292	5698
binocular	6319	7141
bonsai	4383	4665
brain	2673	3091
brontosaurus	6240	6988
buddha	3399	3655
butterfly	4725	4540
camera	7902	6700
cannon	4298	3604
car_side	3490	2408
ceiling_fan	4546	4688
cellphone	9137	9182
chair	7891	8223
chandelier	2705	3507
cougar_body	6318	6559

Table A.2 continued from previous page

Category	MSE on LEFT	MSE on Bottom
cougar_face	4670	5656
crab	4605	4645
crayfish	3681	3606
crocodile	2957	3305
crocodile_head	3282	3645
cup	5585	5950
dalmatian	10148	9379
dollar_bill	4170	4045
dolphin	4604	4812
dragonfly	4695	3772
electric_guitar	8315	7040
elephant	3737	4405
emu	5130	4003
euphonium	6566	5676
ewer	5928	5823
ferry	5802	6089
flamingo	4330	3530
flamingo_head	4314	5224
garfield	4067	3396
gerenuk	4082	3707
gramophone	6166	6444
grand_piano	7471	7441
hawksbill	2784	3117
headphone	11606	13387
hedgehog	4773	3806
helicopter	3552	4150
ibis	3404	3604
inline_skate	8133	7232
joshua_tree	5697	5272
kangaroo	2658	2845
ketch	3181	4032
lamp	6706	6387
laptop	5109	5140
llama	3428	3275
lobster	6988	6971

Table A.2 continued from previous page

Category	MSE on LEFT	MSE on Bottom
lotus	4385	3577
mandolin	4134	5232
mayfly	4034	5676
menorah	3605	3721
metronome	9449	5907
minaret	4166	3984
nautilus	6448	7545
octopus	5240	4679
okapi	4391	4283
pagoda	10326	7060
panda	11426	9914
pigeon	4819	6575
pizza	5328	5383
platypus	8097	6753
pyramid	3788	4518
revolver	5276	3705
rhino	4014	5313
rooster	5773	5953
saxophone	6296	5890
schooner	3739	4807
scissors	3208	4206
scorpion	3824	3895
sea_horse	5056	4607
snoopy	7114	7229
soccer_ball	10209	10991
stapler	7418	7209
starfish	3346	3265
stegosaurus	7893	7032
stop_sign	4254	4213
strawberry	2599	2901
sunflower	2583	2993
tick	5478	5047
trilobite	2621	2722
umbrella	5895	5285
watch	4088	4054

Table A.2 continued from previous page

Category	MSE on LEFT	MSE on Bottom
water_lilly	5125	5658
wheelchair	6098	5301
wild_cat	6294	6849
windsor_chair	3969	4588
wrench	3454	3639
yin_yang	24315	24242
Olivetti	1084	1156

A.3 Experiment #3

Table A.3: Complete Results of Experiment #3

Category	MSE on LEFT	MSE on Bottom
Caltech(ALL)	5595	2789
Faces	1268	1045
Faces_easy	3976	1472
Leopards	1481	965
Motorbikes	2285	2259
accordion	4298	2957
airplanes	890	478
anchor	6306	2335
ant	4316	1899
barrel	10357	3367
bass	5666	2510
beaver	3046	2123
binocular	10497	5063
bonsai	6165	3560
brain	6593	2965
brontosaurus	7618	3255
buddha	4120	1640
butterfly	4598	2113
camera	12762	6424
cannon	3537	1297
car_side	2346	921
ceiling_fan	3396	2163

Table A.3 continued from previous page

Category	MSE on LEFT	MSE on Bottom
cellphone	7096	3500
chair	8295	4166
chandelier	4983	1893
cougar_body	4813	2802
cougar_face	6487	3655
crab	3585	1919
crayfish	4771	2410
crocodile	1786	774
crocodile_head	2201	1472
cup	7814	3582
dalmatian	5106	3172
dollar_bill	3022	1209
dolphin	4788	2211
dragonfly	8708	3076
electric_guitar	4223	2359
elephant	5013	2262
emu	5686	2773
euphonium	7300	2942
ewer	3614	1930
ferry	3748	1896
flamingo	4582	2152
flamingo_head	5323	3049
garfield	8234	3234
gerenuk	5468	3058
gramophone	6599	3987
grand_piano	7668	3817
hawksbill	1879	1280
headphone	11101	7035
hedgehog	4546	2819
helicopter	2338	1179
ibis	2718	1460
inline_skate	7464	4097
joshua_tree	6111	3323
kangaroo	3346	1627
ketch	5706	2177

Table A.3 continued from previous page

Category	MSE on LEFT	MSE on Bottom
lamp	5561	3285
laptop	7578	4618
llama	4615	1973
lobster	5799	3098
lotus	3999	1821
mandolin	3074	1777
mayfly	3117	1920
menorah	5399	2176
metronome	12159	3466
minaret	2439	1579
nautilus	5065	2568
octopus	7957	3584
okapi	7405	3182
pagoda	7706	3225
panda	8660	6141
pigeon	4142	3361
pizza	6115	3417
platypus	6230	2894
pyramid	3622	2323
revolver	5226	2047
rhino	3406	3286
rooster	7754	3618
saxophone	6617	2880
schooner	4515	2331
scissors	5392	2615
scorpion	3039	1853
sea_horse	4365	2321
snoopy	9315	3986
soccer_ball	10111	5002
stapler	5632	3192
starfish	3853	2032
stegosaurus	5943	3522
stop_sign	8750	3225
strawberry	9930	3706
sunflower	5577	2599

Table A.3 continued from previous page

Category	MSE on LEFT	MSE on Bottom
tick	6930	2855
trilobite	2966	1687
umbrella	7556	4329
watch	5018	2326
water_lilly	2812	1409
wheelchair	9903	3418
wild_cat	6645	3486
windsor_chair	5575	2030
wrench	3796	1584
yin_yang	12227	8853

A.4 Experiment #4

Table A.4: Complete Results of Experiment #4

Category	MSE on LEFT	MSE on Bottom
Caltech(ALL)	5587	2775
Faces	1268	1045
Faces_easy	3835	1602
Leopards	1523	966
Motorbikes	2444	1864
accordion	4253	2788
airplanes	890	478
anchor	6306	2335
ant	4265	2044
barrel	10412	3207
bass	5508	2376
beaver	3045	2143
binocular	10497	5063
bonsai	6165	3560
brain	6463	2990
brontosaurus	7423	3206
buddha	4114	1593
butterfly	4435	2092
camera	12762	6424

Table A.4 continued from previous page

Category	MSE on LEFT	MSE on Bottom
cannon	3537	1297
car_side	2313	791
ceiling_fan	3453	2150
cellphone	7454	3584
chair	8505	4028
chandelier	4983	1893
cougar_body	4813	2802
cougar_face	6512	3592
crab	3521	1899
crayfish	4666	2530
crocodile	1813	775
crocodile_head	2201	1472
cup	7814	3582
dalmatian	5163	3173
dollar_bill	3038	1159
dolphin	4700	2174
dragonfly	8569	3029
electric_guitar	4223	2359
elephant	5013	2262
emu	5700	2782
euphonium	7100	3023
ewer	3503	1855
ferry	3744	1903
flamingo	4582	2152
flamingo_head	5323	3049
garfield	8224	3275
gerenuk	5488	3008
gramophone	6426	4054
grand_piano	7731	3935
hawksbill	1879	1280
headphone	11101	7035
hedgehog	4621	2775
helicopter	2316	1185
ibis	2732	1419
inline_skate	7474	4118

Table A.4 continued from previous page

Category	MSE on LEFT	MSE on Bottom
joshua_tree	6111	3323
kangaroo	3346	1627
ketch	5524	2186
lamp	5552	3249
laptop	7673	4740
llama	4626	2006
lobster	5799	3098
lotus	3999	1821
mandolin	3005	1860
mayfly	3160	1849
menorah	5372	2189
metronome	12405	3970
minaret	2439	1579
nautilus	5065	2568
octopus	7856	2937
okapi	7421	3143
pagoda	7693	3165
panda	8570	5264
pigeon	4142	3361
pizza	6115	3417
platypus	6217	2985
pyramid	3552	2254
revolver	5332	2044
rhino	3407	3246
rooster	7754	3618
saxophone	6617	2880
schooner	4452	2411
scissors	5431	2534
scorpion	3123	2005
sea_horse	4386	2246
snoopy	9315	3986
soccer_ball	10111	5002
stapler	5623	3060
starfish	3836	2054
stegosaurus	5921	3408

Table A.4 continued from previous page

Category	MSE on LEFT	MSE on Bottom
stop_sign	8978	3203
strawberry	9930	3706
sunflower	5577	2599
tick	6723	2869
trilobite	2975	1746
umbrella	7548	4718
watch	4835	2122
water_lilly	2812	1409
wheelchair	9903	3418
wild_cat	6781	3708
windsor_chair	5516	1995
wrench	3716	1562
yin_yang	12165	8984

Appendix B Experiment Analysis

B.1 Experitment #2 VS Poon's

Table B.1: Comparsion on LEFT MSE

Category	MSE on LEFT(Exp #2)	MSE on LEFT(Poon)
Caltech(ALL)	5371	3475
Faces	3789	3416
Leopards	1634	1541
Motorbike	3838	2581
Bonsai	**4383**	4545
Mandolin	**4134**	4239
Wrench	**3454**	3861
yin_yang	24315	5222

Table B.2: Comparsion on BOTTOM MSE

Category	MSE on BOTTOM(Exp #2)	MSE on BOTTOM(Poon)
Caltech(ALL)	5344	3538
Faces	3971	3555
Leopards	1844	1498
Motorbike	4977	3626
Bonsai	**4665**	4694
Mandolin	**3396**	3583
Wrench	**3639**	3975
yin_yang	24242	5394

B.2 Experiment #3 VS Poon's

Table B.3: Comparsion on LEFT MSE

Category	LEFT MSE(Exp #3)	LEFT MSE(Poon)
Caltech(ALL)	5595	3475
Faces	**1268**	3416
Leopards	**1481**	1541
Motorbikes	**2285**	2581
accordion	**4298**	4535
beaver	**3046**	3211
car_side	**2346**	2357
crocodile	**1786**	2148
dalmatian	**5106**	5748
electric_guitar	**4223**	5175
hawksbill	**1879**	2350
helicopter	**2338**	2749
mandolin	**3074**	4239
minaret	**2439**	3455
water_lilly	**2812**	3112
wrench	**3796**	3861
yin_yang	12227	5222

Table B.4: Comparsion on BOTTOM MSE

Category	MSE on BOTTOM(Exp #3)	MSE on BOTTOM(Poon)
Caltech(ALL)	**2789**	3538
Faces	**1045**	3555
Faces_easy	**1472**	1924
Leopards	**965**	1498
Motorbikes	**2259**	3626
accordion	**2957**	4876
airplanes	**478**	1276
anchor	**2335**	2559
ant	**1899**	3172
barrel	**3367**	3598
bass	**2510**	3751
beaver	**2123**	3504
binocular	**5063**	5196
bonsai	**3560**	4694
brontosaurus	**3255**	4059
buddha	**1640**	2746
butterfly	**2113**	4073

Table B.4 continued from previous page

Category	MSE on BOTTOM(Exp #3)	MSE on BOTTOM(Poon)
cannon	**1297**	2543
car_side	**921**	2310
ceiling_fan	**2163**	2995
cellphone	**3500**	5514
chair	**4166**	4657
chandelier	**1893**	2582
cougar_body	**2802**	3571
crab	**1919**	3013
crocodile	**774**	2489
crocodile_head	**1472**	2527
cup	**3582**	3775
dalmatian	**3172**	5408
dollar_bill	**1209**	2565
dolphin	**2211**	2767
electric_guitar	**2359**	5068
elephant	**2262**	2693
euphonium	**2942**	3663
ewer	**1930**	3735
ferry	**1896**	3432
flamingo	**2152**	2447
flamingo_head	**3049**	3514
garfield	**3234**	3583
grand_piano	**3817**	5994
hawksbill	**1280**	2575
headphone	**7035**	8124
helicopter	**1179**	3064
ibis	**1460**	2480
inline_skate	**4097**	5595
joshua_tree	**3323**	3465
kangaroo	**1627**	2113
ketch	**2177**	3499
lamp	**3285**	4103
llama	**1973**	2620
lobster	**3098**	4764
lotus	**1821**	2637

Table B.4 continued from previous page

Category	MSE on BOTTOM(Exp #3)	MSE on BOTTOM(Poon)
mandolin	**1777**	4654
mayfly	**1920**	3216
menorah	**2176**	3040
metronome	**3466**	3711
minaret	**1579**	3160
nautilus	**2568**	4356
octopus	**3584**	3625
pagoda	**3225**	4028
panda	**6141**	7018
pigeon	**3361**	3805
platypus	**2894**	4242
pyramid	**2323**	2359
revolver	**2047**	2694
rooster	**3618**	3640
saxophone	**2880**	3414
schooner	**2331**	3207
scorpion	**1853**	2976
sea_horse	**2321**	2944
snoopy	**3986**	4659
soccer_ball	**5002**	6359
stapler	**3192**	4950
starfish	**2032**	2224
stegosaurus	**3522**	3886
stop_sign	**3225**	3590
tick	**2855**	3678
trilobite	**1687**	2416
watch	**2326**	3484
water_lilly	**1409**	3385
wheelchair	**3418**	3492
wild_cat	**3486**	5065
windsor_chair	**2030**	3257
wrench	**1584**	3975
yin_yang	8853	5394

B.3 Experiment #4 VS Poon's

Table B.5: Comparsion on LEFT MSE

Category	MSE on LEFT(Exp #4)	MSE on LEFT(Poon)
Caltech(ALL)	5587	3475
Faces	**1268**	3416
Leopards	**1523**	1541
Motorbikes	**2444**	2581
accordion	**4253**	4535
beaver	**3045**	3211
car_side	**2313**	2357
crocodile	**1813**	2148
dalmatian	**5163**	5748
electric_guitar	**4223**	5175
hawksbill	**1879**	2350
helicopter	**2316**	2749
mandolin	**3005**	4239
minaret	**2439**	3455
water_lilly	**2812**	3112
wrench	**3716**	3861
yin_yang	12165	5222

Table B.6: Comparsion on BOTTOM MSE

Category	MSE on BOTTOM(Exp #4)	MSE on BOTTOM(Poon)
Caltech(ALL)	**2775**	3538
Faces	**1045**	3555
Faces_easy	**1602**	1924
Leopards	**966**	1498
Motorbikes	**1864**	3626
accordion	**2788**	4876
airplanes	**478**	1276
anchor	**2335**	2559
ant	**2044**	3172
barrel	**3207**	3598
bass	**2376**	3751
beaver	**2143**	3504
binocular	**5063**	5196
bonsai	**3560**	4694
brontosaurus	**3206**	4059
buddha	**1593**	2746
butterfly	**2092**	4073
cannon	**1297**	2543
car_side	**791**	2310

Table B.6 continued from previous page

Category	MSE on BOTTOM(Exp #4)	MSE on BOTTOM(Poon)
ceiling_fan	**2150**	2995
cellphone	**3584**	5514
chair	**4028**	4657
chandelier	**1893**	2582
cougar_body	**2802**	3571
crab	**1899**	3013
crocodile	**775**	2489
crocodile_head	**1472**	2527
cup	**3582**	3775
dalmatian	**3173**	5408
dollar_bill	**1159**	2565
dolphin	**2174**	2767
electric_guitar	**2359**	5068
elephant	**2262**	2693
euphonium	**3023**	3663
ewer	**1855**	3735
ferry	**1903**	3432
flamingo	**2152**	2447
flamingo_head	**3049**	3514
garfield	**3275**	3583
grand_piano	**3935**	5994
hawksbill	**1280**	2575
headphone	**7035**	8124
helicopter	**1185**	3064
ibis	**1419**	2480
inline_skate	**4118**	5595
joshua_tree	**3323**	3465
kangaroo	**1627**	2113
ketch	**2186**	3499
lamp	**3249**	4103
llama	**2006**	2620
lobster	**3098**	4764
lotus	**1821**	2637
mandolin	**1860**	4654
mayfly	**1849**	3216

Table B.6 continued from previous page

Category	MSE on BOTTOM(Exp #4)	MSE on BOTTOM(Poon)
menorah	**2189**	3040
minaret	**1579**	3160
nautilus	**2568**	4356
octopus	**2937**	3625
pagoda	**3165**	4028
panda	**5264**	7018
pigeon	**3361**	3805
platypus	**2985**	4242
pyramid	**2254**	2359
revolver	**2044**	2694
rooster	**3618**	3640
saxophone	**2880**	3414
schooner	**2411**	3207
scorpion	**2005**	2976
sea_horse	**2246**	2944
snoopy	**3986**	4659
soccer_ball	**5002**	6359
stapler	**3060**	4950
starfish	**2054**	2224
stegosaurus	**3408**	3886
stop_sign	**3203**	3590
tick	**2869**	3678
trilobite	**1746**	2416
watch	**2122**	3484
water_lilly	**1409**	3385
wheelchair	**3418**	3492
wild_cat	**3708**	5065
windsor_chair	**1995**	3257
wrench	**1562**	3975
yin_yang	8984	5394

Acknowledgements

I would like to thank my supervisor Prof. Ke Tang for his support to my thesis topic, and another supervisor Prof. Shan He for his great idea and supports during my work. He asked Mr. Weifeng Li to assist me. Mr. Weifeng Li is in British and we discussed via phone or E-mail to ensure the progress. Prof. Bo Tang is the inspector of my interim report. I'd like to say thanks for his advice on my reports and presentation. Besides, Dr. Xiaofen Lu and Dr. Guiying Li helped me when my program encountered some problems. I also get suggestions about my thesis and reports from Dr. Xiaofen Lu. Moreover, I am grateful to Wenchao Zhang for his thesis template which saved me a lot to time. Finally, I would like to say thanks to all my friends for the wonderful four years.

Yilin ZHENG

May, 2019