BACHARELADO EM MATEMÁTICA

Thiago Peres Casagrande

Um Estudo sobre Algoritmos para Aprendizado Estrutural de Circuitos Probabilísticos

São Paulo

2º Semestre de 2022

Thiago Peres Casagrande

Um Estudo sobre Algoritmos para Aprendizado Estrutural de Circuitos Probabilísticos

Monografia apresentada à disciplina
MAT–0148 — Introdução ao Trabalho Científico,
Departamento de Matemática,
Instituto de Matemática e Estatística,
Universidade de São Paulo.

Área de Concentração: CIÊNCIAS EXATAS E DA TERRA

Orientador: Denis Deratani Mauá

São Paulo

2º Semestre de 2022

Ficha catalográfica elaborada com dados inseridos pelo(a) autor(a)
Biblioteca Carlos Benjamin de Lyra
Instituto de Matemática e Estatística
Universidade de São Paulo

Casagrande, Thiago Peres
Um Estudo sobre Algoritmos para Aprendizado Estrutural de Circuitos Probabilíısticos / Thiago Peres Casagrande; orientador, Denis Deratani Mauá. - São Paulo, 2022.
34 p.: il.

Trabalho de Conclusão de Curso (Graduação) - Ciência da Computação / Instituto de Matemática e Estatística / Universidade de São Paulo.
Bibliografia

1. Circuitos Probabilísticos. 2. Aprendizado de Máquina. 3. Modelos Probabilísticos. 4. Modelos Gerativos. I. Deratani Mauá, Denis.

Bibliotecárias do Serviço de Informação e Biblioteca
Carlos Benjamin de Lyra do IME-USP, responsáveis pela
estrutura de catalogação da publicação de acordo com a AACR2:
Maria Lúcia Ribeiro CRB-8/2766; Stela do Nascimento Madruga CRB 8/7534.

Folha de Avaliação

Aluno: Thiago Peres Casagrande

Título: Um Estudo sobre Algoritmos para Aprendizado Estrutural de Circuitos Probabilísticos

Data: 2º Semestre de 2022

Banca Examinadora

Denis Deratani Mauá (Orientador)

Renato Geh

Mateus Espadoto

AGRADECIMENTOS

Ao meu orientador, Denis, pelas diversas reuniões e sugestões, sem as quais este trabalho não seria possível. Mas acima disso, agradeço pelo apoio, paciência, e por instigar minha curiosidade e incentivar meu empenho no decorrer deste estudo.

Aos meus pais, pelo apoio infindável que recebi no decorrer da minha trajetória dentro da graduação, seja ela qual fosse. Pelo esforço admirável para comigo e por tentarem, frente a qualquer dificuldade, criar um terreno fértil para que eu traçasse o caminho que tracei.

Ao meu irmão, Heitor, pela fé que teve em acompanhar meus passos, mesmo nos momentos mais dúbeis. Pelas conversas em momentos difíceis e pelo incentivo em buscar propósito nos meus estudos.

Aos meus amigos Jeremias, Amaral, Marco, Gustavo(s), Levi e Pedro por toda a ajuda nesse trajeto. Mas, principalmente, por tornarem, em cada conversa, em cada café e em cada bandejão, o caminho tortuoso da graduação mais agradável e leve.

RESUMO

CASAGRANDE, T. **Um Estudo sobre Algoritmos para Aprendizado Estrutural de Circuitos Probabilísticos**. 2022. 34 p. Monografia (Bacharelado em Matemática) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2º Semestre de 2022.

Este texto é uma monografia de Iniciação Científica para o estudo de circuitos probabilísticos, que são modelos probabilísticos gerativos sobre os quais é possível realizar diferentes tipos de inferência de forma exata e computacionalmente eficiente. Um circuito probabilístico é especificado através de sua estrutura, um grafo direcionado acíclico que modela as relações de (in)dependência condicionais e contextuais entre as variáveis aleatórias do modelo, e de seus parâmetros, que são pesos associados às arestas do grafo e parâmetros de distribuições univariadas associadas às folhas. Nesse projeto, o aluno fez uma revisão da literatura dos principais algoritmos de aprendizado de estrutura de circuitos probabilísticos a partir de conjunto de dados, comparando suas complexidades em relação à dimensionalidade do problema e à quantidade de dados. Com base nesse estudo, o aluno implementou uma dessas técnicas bem como suas devidas otimizações, utilizando-se da biblioteca *open source* do grupo de pesquisa em linguagem Julia, analisando-a em relação à eficiência de inferência e comparando seus resultados com os presentes na literatura.

Palavras-chave: Circuitos Probabilísticos. Aprendizado de Máquina. Modelos Probabilísticos. Modelos Gerativos.

ABSTRACT

CASAGRANDE, T. **A study on Structural Learning Algorithms for Probabilistic Circuits**. 2022. 34 p. Monografia (Bacharelado em Matemática) – Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2º Semestre de 2022.

This text presents an undergraduate thesis for the study on probabilistic circuits, which are generative probabilistic models that enables the computation of to compute different types of inference in a exact and computationally efficient way. A probabilistic circuit is specified through its structure, a directed acyclic graph that models the conditional relations of (in)dependence between the random variables in the model, and its parameters, which are both the weights associated to the edges of the graph and the parameters of the univariate distributions associated to the leafs. In this project the student did a review of the literature of the main structural learning algorithms of probabilistic circuits based on a dataset, comparing their complexity in regard to the dimensionality of the problem and the amount of data available. Based on this study, the student implemented one of these techniques and its optimizations, using the open source library of the research group using the Julia programming language, analysing it on its inference efficiency and comparing its results with the ones found on the literature.

Keywords: Probabilistic Circuits. Machine Learning. Probabilistic Models. Generative Models.

Lista de Figuras

Lista de Abreviaturas e Siglas

DAG	Grafo Acíclico Direcionado (*Directed Aciclic Graph*)
PDF	Função Densidade de Probabilidade (*Probability density function*)
VA	Variável Aleatória (*Random Variable*)
MAP	*Maximum a posteriori*
EVI	Evidência (*Evidence*)
MAR	Marginal
FFM	Modelo Completamente Fatorizado (*Fully Factorized Model*)
PC	Circuito Probabilístico (*Probabilistic Circuit*)

LISTA DE SÍMBOLOS

X, Y, Z	Variáveis Aleatórias
$\mathbf{X}, \mathbf{Y}, \mathbf{Z}$	Conjuntos de Variáveis Aleatórias
$val(X)$	Suporte da Variável Aleatória X
$\boldsymbol{\theta}$	Vetor de Parâmetros
$\mathcal{G}$	Grafo
n, m, p	Nós de um grafo
$ch(n)$	Conjunto dos nós filhos do nó n
$sc(n)$	Escopo do nó n

Sumário

Capítulo 1

Introdução

Desde a compreensão de letras e signos à associação de imagens entre si, o reconhecimento de padrões permeia grande parte dos problemas que, como seres humanos, solucionamos diariamente. A tentativa de automatização, ou mesmo expansão, desta nossa capacidade por meio de técnicas computacionais é do que se trata o Aprendizado de Máquina [2]. Modelos nesta área, desenvolvidos com o intuito de representar a realidade, tem, por vezes, uma capacidade de reconhecimento muito mais acurada e veloz que uma pessoa comumente poderia ter. Não a toa, recorremos cada vez mais a técnicas de Aprendizado de Máquina em nosso cotidiano: seja em situações mais mundanas, como reconhecimento de algorismos escritos à mão, ou mesmo em situações mais complexas, como é o caso de carros autônomos, necessitamos de algoritmos para tarefas importantíssimas que cumpram o papel preditivo que nos é, na maioria dos casos, deficiente.

Com isto, faz-se fundamental o desenvolvimento de modelos que representem fielmente a realidade – ou ao menos de forma suficientemente boa. Neste texto será estudado um modelo que visa à modelagem probabilística de uma população (isto é, de qualquer fenômeno que se deseje analisar) conhecido por Circuito Probabilístico. Antes, no entanto, faz-se necessária a introdução de outro modelo probabilístico que será parte da composição dos Circuitos Probabilísticos: o Modelo de Misturas. Junto a ele, será introduzido o conceito de tratabilidade e de classes de inferência, bem como uma deficiência deste último modelo; mostraremos, pois, como nosso modelo principal soluciona tal deficiência, e, posteriormente, como obter um Circuito Probabilístico por meio de um conjunto de dados.

A ordem dos temas abordados nesta monografia se dá da seguinte forma: no Capítulo 2, são introduzidos formalmente os Modelos de Mistura, exemplificando-os. Ainda, é definida a tratabilidade para modelos probabilísticos para classes de inferência, para em seguida tratar das vantagens e desvantagens deste modelo. No capítulo 3, é definido um Circuito Probabilístico,

e como sua estrutura permite representar outros modelos probabilísticos; são ditas, também, as vantagens deste novo modelo sobre os de misturas. São ainda definidas propriedades importantes deste modelo em relação à tratabilidade. No capítulo 3, é apresentado o problema de construção de um Circuito Probabilístico, especificamente de sua estrutura; é apresentada uma técnica para a construção destes a partir de um conjunto de dados, conhecida por LearnSPN, bem como suas características, positivas e negativas, e devidas otimizações. Por fim, são apresentadas ainda neste capítulo resultados de testes de inferência para a técnica implementada de maneira comparativa com os resultados apresentados na literatura.

Capítulo 2

Modelos de Mistura

O Aprendizado de Máquina (do inglês, *Machine Learning* ou, apenas, **ML**) tem como objetivo o estudo e desenvolvimento de algoritmos e técnicas para a realização de tarefas complexas a partir de observações. Para tal, uma abordagem extremamente comum e estatisticamente adequada é a probabilística: assume-se que fenômeno analisado é descrito por uma ou por um conjunto de variáveis aleatórias; desta forma, a tarefa restringe-se a inferir, por meio de uma amostra representativa desta população, esta distribuição de probabilidades. Com esta distribuição em mãos, efetua-se a tarefa complexa de interesse por meio da computação de alguma probabilidade, seja ela de realizar predição ou inferência. Dizemos que o modelo obtido por esta abordagem é um **modelo probabilístico**, que, sucintamente, nada mais é do que uma representação particular de distribuições de probabilidade.

Na tarefa de encontrar uma função de probabilidade adequada para uma determinada população, no entanto, podemos nos deparar com casos suficientemente complexos, que não podem ser modelados por distribuições de probabilidade univariadas (ou multivariadas) já conhecidas; é adequado buscar, em muitos casos, uma distribuição que capture particularidades das variáveis da população que uma distribuição de probabilidade simples não o faria. Ainda assim, existe o interesse de não abandonar as distribuições conhecidas, como é o caso da distribuição normal, haja vista que, em termos de inferência, seus estimadores são conhecidos e facilmente calculados (como é o caso do estimador de máxima verossimilhança, por exemplo). Para a conciliação destas duas necessidades, introduzimos o modelo gerativo probabilístico conhecido por **Modelo de Misturas** (do inglês, *Mixture Models*) por meio de um primeiro exemplo:

Exemplo 2.0.1. Modelo de Misturas

Suponha que queiramos inferir a Função Densidade de Probabilidade (do inglês, Probability Density Function, ou apenas **PDF**) de um conjunto de variáveis aleatórias (ou apenas **VAs**) $\mathbf{X}$ que assume valores em $\mathbb{R}^2$. Suponha ainda que, para essa estimação, tenha-se a seguinte amostra,

representada na figura 2.1.:

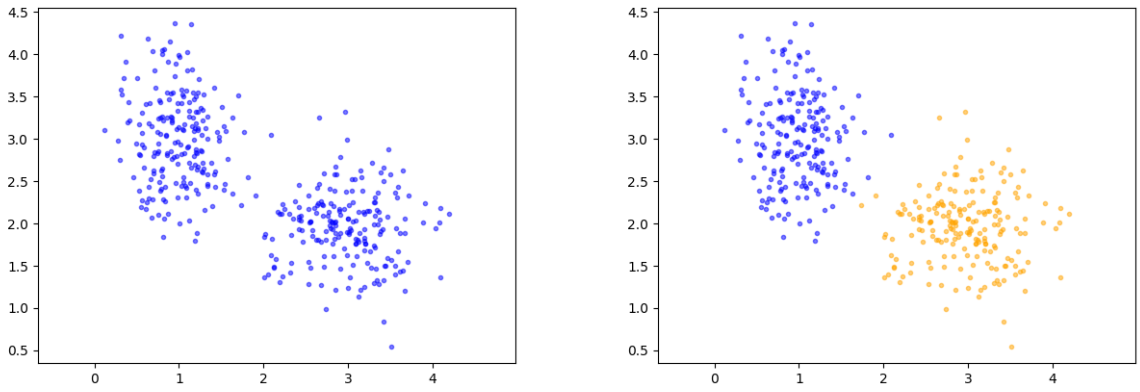

Figura 2.1: Na figura à esquerda temos a amostra de **X**; à direita, a mesma amostra, evidenciando sua estratificação.

É notável que a amostra se divide em dois subgrupos, como vemos na segunda imagem. Com isto, cada subamostra possui uma moda, o que torna razoável utilizar individualmente para cada subgrupo uma distribuição contínua; assumimos, para isso, que cada subamostra é representativa para cada subpopulação. Utilizaremos para ambas uma distribuição normal bivariada para então estimar seus parâmetros. Com isto, o problema se reduz a uma tarefa simples de inferência estatística. Abaixo, na figura 2.2, temos as respectivas distribuições de probabilidade, após a estimação de parâmetros por meio dos elementos de cada subamostra:

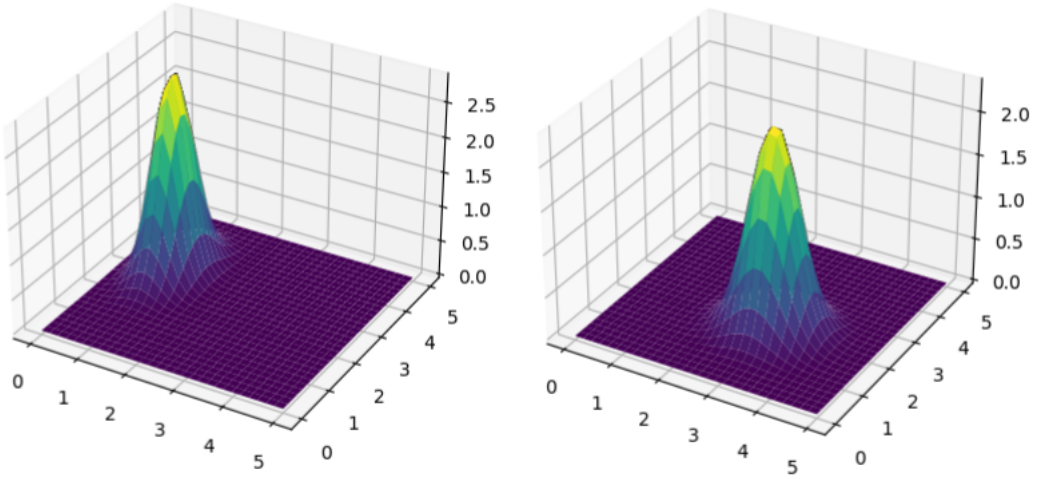

Figura 2.2: Distribuições de probabilidade de cada subpopulação.

Feito isto, basta "mesclar"as funções acima para obter a PDF que descreve a nossa população. Isto é feito por meio de uma combinação convexa: se $f_{\mathbf{X_1}}, f_{\mathbf{X_2}}$ são as PDFs de cada subpopulação então a PDF da população será dada por $f_{\mathbf{X}}(\mathbf{x}) = \omega_1 f_{\mathbf{X_1}} + \omega_2 f_{\mathbf{X_2}}$, em que $\omega_1, \omega_2 \in \mathbb{R}$ cumprem a seguinte restrição:

- $\omega_1, \omega_2 \geq 0$
- $\omega_1 + \omega_2 = 1$

Isto implica que $f_{\mathbf{X}}$ seja, também, uma PDF, representada na figura 2.3:

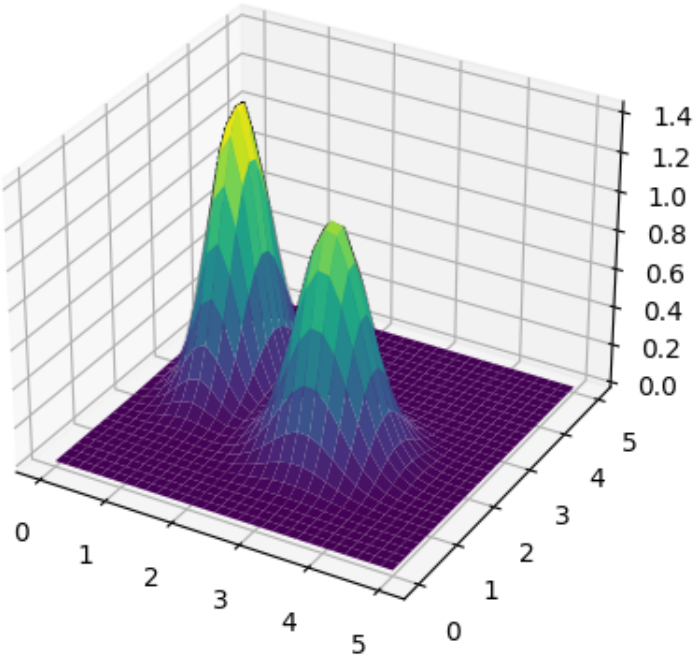

Figura 2.3: Distribuição de probabilidade da população.

Este exemplo sintetiza de forma sucinta o modelo de misturas que é definido a seguir:

Definição 2.0.2. Modelo de Mistura

Seja $\boldsymbol{X_1, ..., X_n}$ VAs, em que $\boldsymbol{X_j}$ assume valores em $\mathbb{R}^m$ $\forall j \in \{1, ..., n\}$ e um conjunto $\{\omega_1, ..., \omega_n\} \subset \mathbb{R}$ tal que $\omega_i \geq 0$ e $\sum_{i=1}^{n} \omega_i = 1$.

Então, um modelo de misturas M é uma PDF descrita por

$$f_M(\mathbf{x}) = \sum_{i=1}^{n} \omega_i f_{\mathbf{X_i}}(\mathbf{x}) \tag{2.1}$$

Sucintamente, dizemos que um modelo de misturas (finito) é uma **combinação convexa de funções densidade de probabilidade**. Ainda, denotaremos por GMM (do inglês, *Gaussian Mixture Model*) o modelo cujas $f_{\mathbf{X_i}}$ forem PDFs de VAs normais.

2.1 Modelos Tratáveis

Naturalmente, modelos de mistura destacam-se especialmente por sua **capacidade expressiva**, como foi evidenciado no exemplo 1.0.1. Sua intrínseca capacidade de representar subpopulações dentro de sua população torna o modelo particularmente interessante e sofisticado para a determinação de sua distribuição de probabilidade. Na verdade, sabe-se que GMMs podem aproximar uniformemente qualquer distribuição contínua [1]:

para qualquer distribuição contínua $f_{\mathbf{X}}$ com $val(\mathbf{X}) \in \mathbb{R}^m$ existe uma sequência de PDFs normais multivariadas $\{f_{\mathbf{X_n}}\}_{n\in\mathbb{N}}$ e uma sequência $\{\omega_n\}_{n\in\mathbb{N}}$ para as quais vale que $\sum_{n=1}^{M} f_{\mathbf{X_n}} \xrightarrow[unif]{} f_{\mathbf{X}}$.

Não só isso, mas o fato deste modelo ser determinado por uma combinação de distribuições já conhecidas facilita a resolução de determinadas tarefas. Veremos, por exemplo, que sobre um modelo de misturas é possível computar probabilidades marginais de maneira exata e computacionalmente "fácil". Entretanto, com isto, surge a necessidade de definir com precisão como se configura este modelo em termos de computação de tarefas. Para tal, deve-se falar de classes de inferência e tratabilidade de modelos.

Dizemos, informalmente, que uma **classe de inferências** (do inglês, *class of (probabilistic) queries*) refere-se a um conjunto de operações que possuem a mesma estrutura que podem ser realizados sobre um modelo probabilístico [1]. De todas elas, a classe de inferência que será de particular interesse nos próximos capítulos é a **classe de inferência MAP** (ou, *class of maximum à posteriori queries*), e que, por isso, merece ser cuidadosamente definida:

Definição 2.1.1. Classe de Inferência MAP

Seja M um modelo probabilístico sobre as VAs $\mathbf{X}$ e $P_M(\mathbf{x})$ a função densidade de probabilidade conjunta que este modelo descreve. Dizemos que as inferências da classe de inferência MAP são aquelas que calculam:

$$\underset{\mathbf{y}\in val(\mathbf{Y})}{argmax} P_M(\mathbf{y}|\mathbf{e}) = \underset{\mathbf{y}\in val(\mathbf{Y})}{argmax} P_M(\mathbf{y},\mathbf{e}) \tag{2.2}$$

em que $\mathbf{e} \in val(\mathbf{E})$, $\mathbf{y} \in val(\mathbf{Y})$ são estados parciais para uma partição arbitrária de $\mathbf{X}$, isto é, $\mathbf{Y} \cap \mathbf{E} = \emptyset$ e $\mathbf{Y} \cup \mathbf{E} = \mathbf{X}$.

Denotaremos a classe de inferência MAP por $\mathcal{Q}_{MAP}$.

[1] Vale ressaltar que sua tradução ao que é chamado na literatura é, em certa medida, infeliz: as tarefas que podem ser exigidas de um modelo probabilístico não são todas de caráter inferencial; pode-se por exemplo calcular a probabilidade de um estado completo $P(\mathbf{X} = \mathbf{x})$, e esta, apesar disso, será chamada de uma **classe de inferência EVI** (ou *class of complete evidence queries*)

Note que o lado direito da equação (2.2) segue diretamente da definição de probabilidade condicional: $P_M(\mathbf{y}|\mathbf{e}) = \frac{P_M(\mathbf{y},\mathbf{e})}{P_M(\mathbf{e})}$. Como a maximização não é afetada pela constante $P_M(\mathbf{e})$, basta que seja maximizado o termo à direita da equação (2.2).

Definição 2.1.2. Modelo Tratável

Uma classe de inferências $\mathcal{Q}$ é dita tratável sobre uma família de modelos $\mathcal{M}$ se qualquer inferência Q $\in \mathcal{Q}$ sobre qualquer modelo M $\in \mathcal{M}$ pode ser realizada de maneira exata e em tempo polinomial, isto é, $\mathcal{O}(poly|M|)$. De maneira equivalente, dizemos que M é um modelo tratável para $\mathcal{Q}$, ou, quando não especificado $\mathcal{Q}$, simplesmente que M é um modelo tratável.

Um comentário que se faz pertinente é que tratabilidade não se trata de uma propriedade absoluta; pelo contrário, é possível que um modelo probabilístico seja tratável para uma classe de inferências mas intratável para diversas outras. Isto, como será verificado em capítulos seguintes, depende de alguns fatores, tais como a estrutura do modelo. No momento, a fim de ilustrar o conceito de tratabilidade, é interessante verificá-lo para o modelo de misturas para a classe de inferências marginais, definida abaixo:

Definição 2.1.3. Classe de Inferência Marginal

Seja M um modelo probabilístico sobre as VAs $\mathbf{X}$ e $P_M(\mathbf{x})$ a PDF conjunta que este modelo descreve. Dizemos que as inferências da classe de inferência marginal é aquela que calcula a seguinte probabilidade:

$$P_M(Z_1 \in \mathcal{L}_1, ..., Z_k \in \mathcal{L}_k, \mathbf{E} = \mathbf{e}) = \int_{\mathcal{L}_1 \times ... \times \mathcal{L}_k} P_M(\mathbf{Z}, \mathbf{e}) d\mathbf{Z} \tag{2.3}$$

$$= \int_{\mathcal{L}_1} ... \int_{\mathcal{L}_k} P_M(Z_1, ..., Z_k, \mathbf{e}) dZ_1 ... dZ_k \tag{2.4}$$

em que $\mathbf{e} \in val(\mathbf{E})$ é um estado parcial para algum subconjunto $\mathbf{E}$ de $\mathbf{X}$, e $\mathbf{Z} = \mathbf{X} - \mathbf{E}$ é o conjunto de k VAs que é integrado sobre a coleção de intervalos $\{\mathcal{L}_i\}_{i=1}^k$, em que cada intervalo pertence ao suporte de sua respectiva VA em $\mathbf{Z}$, isto é, $\mathcal{L}_i \in val(Z_i); \forall i \in \{1, ..., k\}$.

Denotaremos a classe de inferência marginal por $\mathcal{Q}_{MAR}$.

Proposição 2.1.4. ***Tratabilidade de Modelos de Mistura para Classe de Inferência Marginal***

Seja M um modelo de misturas e f_M sua respectiva PDF contínua, onde

$$f_M(\mathbf{x}) = \sum_{i=1}^{n} \omega_i f_{\mathbf{X_i}}(\mathbf{x}) \tag{2.5}$$

Se as funções $f_{\mathbf{X_i}}$ são tratáveis para a $\mathcal{Q}_{MAR}$, então f_M é tratável para $\mathcal{Q}_{MAR}$.

Demonstração: Lembremos que, por definição, a computação de uma inferência de $\mathcal{Q}_{MAR}$ envolve o cálculo da equação (2.4), replicado abaixo:

$$\int_{\mathcal{L}_1} ... \int_{\mathcal{L}_k} \sum_{i=1}^{n} \omega_i f_{\mathbf{X_i}}(Z_1, ..., Z_k, \mathbf{e}) dZ_1 ... dZ_k = \sum_{i=1}^{n} \omega_i \int_{\mathcal{L}_1} ... \int_{\mathcal{L}_k} f_{\mathbf{X_i}}(Z_1, ..., Z_k, \mathbf{e}) dZ_1 ... dZ_k \tag{2.6}$$

Como $f_{\mathbf{X_i}}$ é tratável, a integral de $f_{\mathbf{X_i}}$ em um intervalo $\mathcal{L}_i$ pode ser calculada em tempo polinomial para cada $i \in \{1, ..., n\}$. Consequentemente, as n integrais da equação acima são calculadas em tempo polinomial, que implica que sua soma é também calculada neste tempo. Portanto, a integral de (2.6) é calculada em tempo $\mathcal{O}(poly|M|)$. □

2.2 Desvantagens deste Modelo

Vimos na seção anterior que utilizar um modelo de misturas é extremamente vantajoso, em especial em relação à expressividade. Entretanto, isto traz consigo uma desvantagem: sua capacidade representativa está diretamente associada ao tamanho do modelo (isto é, a quantidade de distribuições de probabilidade presentes na soma). Consequentemente, se o conjunto de dados é suficientemente sofisticado o modelo de misturas correspondentes também o será, e o aprendizado de parâmetros deste modelo, por seu volume, será computacionalmente custoso. Em outras palavras, apesar de sua capacidade expressiva, este não é um modelo sucinto, como nota-se pelo exemplo abaixo:

Exemplo 2.2.1. Tomemos como exemplo a amostra de uma VA $\mathbf{X}$ que assume valores no $\mathbb{R}^2$, representada abaixo na figura 2.4:

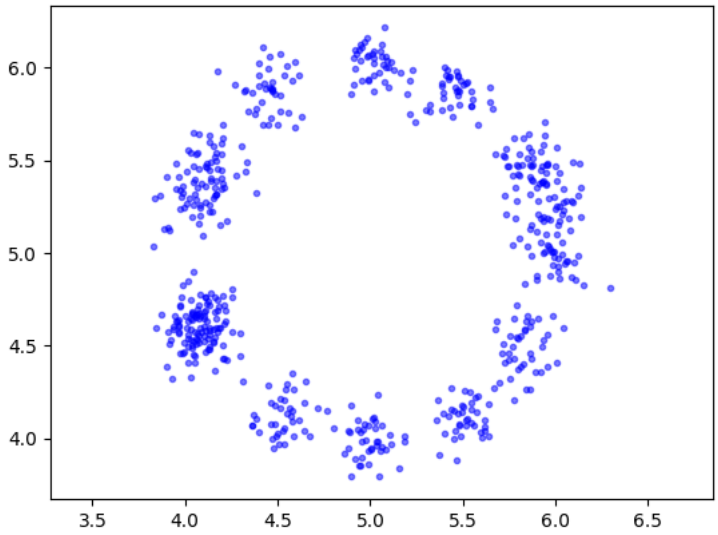

Figura 2.4: Amostra com diversas subamostras.

Nota-se, pela figura acima, que a amostra pode ser separada em diversas subamostras. Logo, se tentarmos encontrar uma PDF por meio de um modelo de misturas para esta população, teremos que primeiramente encontrar os parâmetros de diversas PDFs normais, uma para cada subpopulação, para em seguida encontrar os pesos da combinação convexa mais adequados. Ao final, obteremos a distribuição representada na figura 2.5:

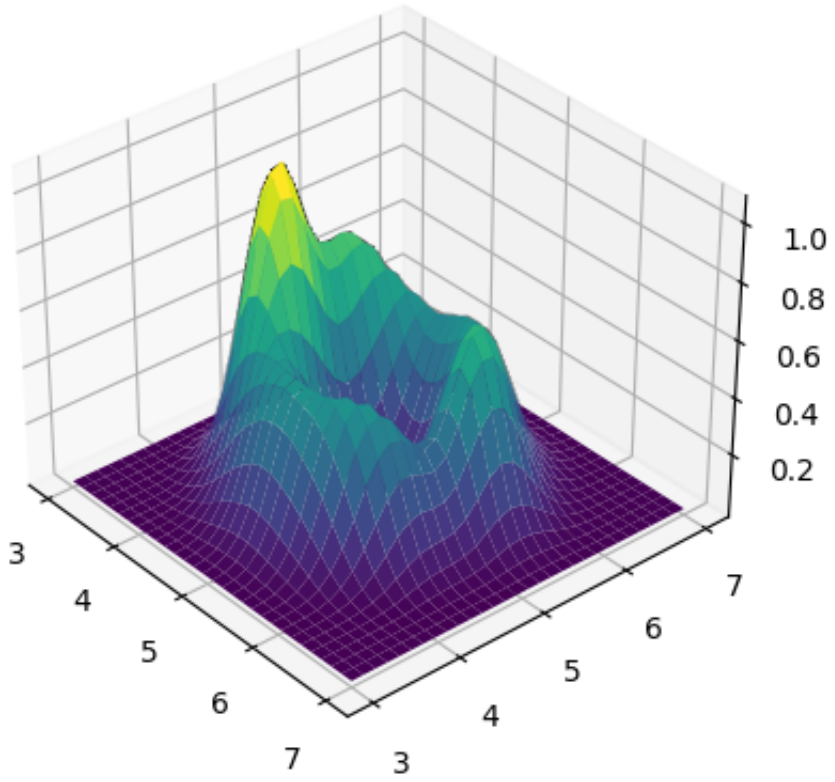

Figura 2.5: Distribuição de Probabilidade da população.

Este exemplo torna evidente que, apesar de expressiva por representar todas as modas na população acima, o modelo de misturas pode trazer consigo uma quantidade grande de parâmetros, o que não é conveniente em termos de aprendizado de parâmetros.

Esta inconveniência presente no modelo de misturas faz surgir a necessidade de outro modelo probabilístico, um que exija menos parâmetros para computar uma PDF sem perder sua capacidade expressiva. Espera-se ainda que este novo modelo consiga conservar, em alguma medida, a tratabilidade, para que a inferência exata e em tempo polinomial ainda seja possível. O modelo que corrige estas desvantagens de forma eficiente é conhecido por Circuito Probabilístico, e é descrito no capítulo 3.

Capítulo 3

Circuitos Probabilísticos

Vimos no capítulo anterior um modelo com funções e características extremamente convenientes para um modelo probabilístico. Porém, suas limitações fazem nascer a necessidade de um novo modelo. Os **Circuitos Probabilísticos** (do inglês, *Probabilistic Circuits*, ou simplesmente **PCs**) são uma classe de modelos que equilibram expressividade com custo inferencial. Trazem consigo a estrutura de grafos que **Modelos de Probabilidade em Grafos**, como Redes Bayesianas, possuem conciliado, ainda, à tratabilidade do modelo de misturas. Não só isso, mas este novo modelo surge como uma maneira de unificar diferentes modelos gerativos: veremos como sua estrutura permite representar (ou mesmo redefinir, de forma equivalente), por exemplo, modelos de mistura (hierárquicos) e modelos completamente fatorizados. Para tal, veremos diferentes formulações deste novo modelo:

3.1 Formulação de PCs

Definição 3.1.1. Definição Descritiva de PC

Um Circuito Probabilístisco $\mathcal{C}$ sobre as VAs $\mathbf{X}$ é uma dupla $(\mathcal{G}, \boldsymbol{\theta})$.

$\mathcal{G}$ é um grafo computacional representado por um grafo direcionado acíclico (do inglês, *Directed Acycic Graph*, ou, apenas, **DAG**) enraízado [1]. Deste grafo, cada unidade computacional, ou nó, pode ser de três tipos:

- nó folha, que representa uma distribuição de probabilidade tratável;
- nó soma;
- nó produto.

[1] ao contrário de um DAG enraízado, no entanto, a direção das arestas apontarão sempre ao vértice mais próximo à raiz. Ainda, muitas vezes a direção das arestas será oportunamente omitida, já que sabemos, pela definição, como ela será.

As arestas que ligam os nós soma aos seus filhos possuem pesos (valores não-negativos), estes especificados em $\boldsymbol{\theta}$. Nos referiremos, também, aos nós produto e soma como nós internos.

$\boldsymbol{\theta}$ é um vetor de parâmetros deste circuito, definido por $\boldsymbol{\theta}_F \cup \boldsymbol{\theta}_S$. $\boldsymbol{\theta}_F$ é o conjunto dos parâmetros das distribuições de probabilidade representadas pelos nós folhas, enquanto $\boldsymbol{\theta}_S$ são os pesos das arestas dos nós somas presentes no circuito.

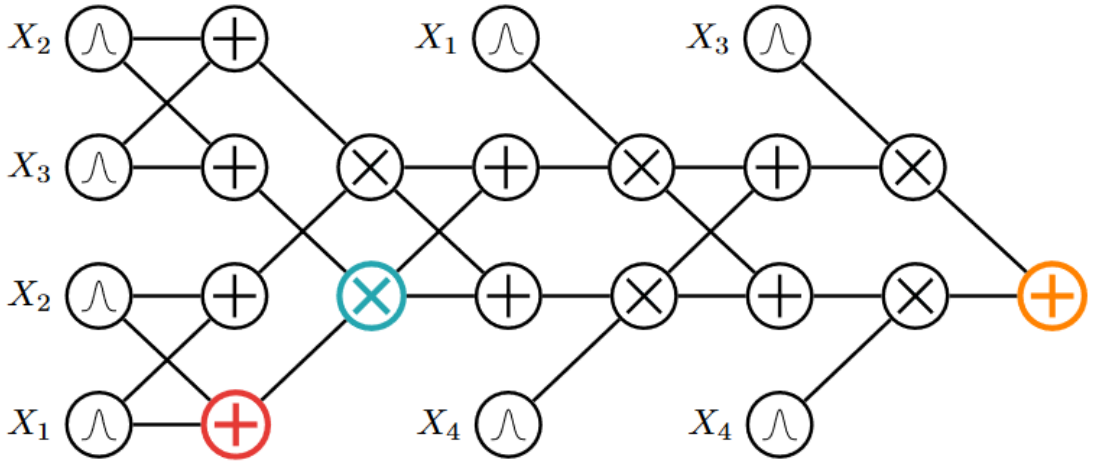

Figura 3.1: Um Circuito Probabilístico cujos nós folhas são distribuições de probabilidade gaussianas. A direção das arestas está omitida. Fonte: [4].

Usualmente, os pesos das arestas que entram em um nó folha são positivos e tem como soma 1, tal como foi definido em um modelo de misturas. Com isto, é possível definir um PC recursivamente a seguir:

Definição 3.1.2. Definição Recursiva de PC

Um Circuito Probabilístico $\mathcal{C}$ sobre as variáveis aleatórias $\mathbf{X}$ é necessariamente um dos itens abaixo:

1. Uma distribuição de probabilidade tratável sobre $\mathbf{X}$;
2. Uma combinação convexa de circuitos probabilísticos $\mathcal{C}_1, ..., \mathcal{C}_n$, cada qual descrito sobre $\mathbf{U}_j \subseteq \mathbf{X}$, isto é:
$$\sum_{j=1}^{n} \omega_j \mathcal{C}_j(\mathbf{U}_j) \tag{3.1}$$
3. Um produto de circuitos probabilísticos $\mathcal{C}_1, ..., \mathcal{C}_n$, cada qual descrito sobre $\mathbf{U}_j \subseteq \mathbf{X}$, dois a dois disjuntos, isto é:
$$\prod_{j=1}^{n} \mathcal{C}_j(\mathbf{U}_j) \text{ em que } \mathbf{U}_j \cap \mathbf{U}_k = \emptyset \tag{3.2}$$

Com a definição recursiva acima entende-se duas coisas: a primeira é que torna-se claro qual será a saída dos nós internos (e consequentemente o que estes representam); a segunda é que faz-se evidente que um PC $\mathcal{C}$ representa uma distribuição conjunta de probabilidade de $\mathbf{X}$. De fato, pode-se demonstrar isto indutivamente:

Proposição 3.1.3. *Seja $\mathcal{C}$ um PC sobre as VAs $\mathbf{X}$. Se para cada nó soma n_s os pesos de sua aresta são não negativos e somam 1, então $\mathcal{C}$ representa uma PDF, isto é, ele satisfaz:*

- $\mathcal{C}(\mathbf{x}) \geq 0 \; \forall \mathbf{x} \in val(\mathbf{X})$
- $\int_{val(\mathbf{X})} \mathcal{C}(\mathbf{x}) dx = 1$

Demonstração: A demonstração é feita de forma indutiva, onde o nosso caso base são os nós folha: nós folhas, por definição, representam uma distribuição de probabilidade.
Agora, assumimos, por hipótese de indução, que as entradas dos nós internos são PDFs. Então:

- para n_s nó soma, sua saída será a combinação convexa de suas entradas. Como um modelo de misturas computa uma PDF, então n_s é uma PDF.
- para n_p nó produto, sua saída será o produto de suas entradas. Sejam $\mathcal{C}_1, ..., \mathcal{C}_m$ entradas de n_p, cada qual descrita sobre $\mathbf{U}_j$ para $j \in \{1, ..., m\}$. Sendo $\mathbf{X} = \bigcup_{j=1}^{m} \mathbf{U_j}$ então:

$$\mathcal{C}_j(\mathbf{u}_j) \geq 0 \; \forall j \in \{1, ..., m\} \implies \prod_{j=1}^{m} \mathcal{C}_j(\mathbf{u}_j) \geq 0$$

e

$$\int_{val(\mathbf{X})} n_p(\mathbf{x}) d\mathbf{x} = \int_{val(\mathbf{U_1})} ... \int_{val(\mathbf{U_m})} \prod_{j=1}^{m} \mathcal{C}_j(\mathbf{u}_j) d\mathbf{u}_1 ... d\mathbf{u}_m$$

$$\prod_{j=1}^{m} \int_{val(\mathbf{U_1})} \mathcal{C}_1(\mathbf{u}_1) d\mathbf{u}_1 ... \int_{val(\mathbf{U_m})} \mathcal{C}_m(\mathbf{u}_m) d\mathbf{u}_m = \prod_{j=1}^{m} 1 = 1.$$

o mostra que a saída de n_p é uma PDF.

□

3.2 Modelos vistos como PCs

Como dito anteriormente, PCs tratam de fornecer uma generalização para outros modelos probabilísticos. Para isso, definimos primeiramente um modelo relacionado à independência de VAs em uma população:

Definição 3.2.1. Modelo Completamente Fatorizado

Suponha que $\mathbf{X_1}, ..., \mathbf{X_n}$ são VAs de uma população duas a duas independentes. Isto implica que podemos escrever a PDF conjunta da seguinte maneira:

$$f_M(\mathbf{x}) = f_{\mathbf{X_1},...,\mathbf{X_n}}(\mathbf{x_1}, ..., \mathbf{x_n}) = \prod_{i=1}^{n} f_{\mathbf{X_i}}(\mathbf{x_i}) \text{ em que } \mathbf{x_i} \in val(\mathbf{X_i}) \tag{3.3}$$

O modelo acima descrito por $f_M(\mathbf{x})$ recebe o nome de **Modelo Completamente Fatorizado** (*Fully Factorized Model*, ou apenas FFM).

Com isto, note que é natural enxergar nós produto como a representação de um modelo completamente fatorizado de suas entradas, já que a saída deste nó será justamente o produtório delas, como é ilustrado na figura 3.2.

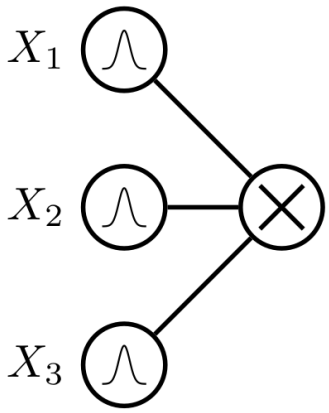

Figura 3.2: Um Circuito Probabilístico representando um Modelo de fatorização. Fonte: [4].

De maneira análoga, pode-se entender nós soma cujos filhos são nós folhas como modelos de mistura, como vemos na figura 3.3. Na verdade, é possível representar uma generalização deste modelo, conhecido por **modelo de misturas hierárquico**. A PDF deste modelo pode ser definida recursivamente:

Definição 3.2.2. Modelo de Misturas Hierárquico

Um Modelo profundo de Misturas é uma PDF $f_\mathbf{M}$ descrita necessariamente por um dos itens abaixo:

1. uma combinação convexa de $f_{\mathbf{X_1}}, ..., f_{\mathbf{X_n}}$ em que $f_{\mathbf{X_i}}$ é uma distribuição tratável $\forall i \in \{1, ..., n\}$, isto é,
$$f_M(\mathbf{x}) = \sum_{i=1}^{n} \omega_i f_{\mathbf{X_i}}(\mathbf{x}); \tag{3.4}$$
2. uma combinação convexa de $f_{\mathbf{M_1}}, ..., f_{\mathbf{M_n}}$ em que $f_{\mathbf{X_i}}$ é um modelo profundo de misturas

$\forall i \in \{1, ..., n\}$, isto é,

$$f_M(\mathbf{x}) = \sum_{i=1}^{n} \omega_i f_{\mathbf{M_i}}(\mathbf{x}) \tag{3.5}$$

Este caso é, também, redefinível pela estrutura de PCs, onde este modelo seria representado por um PC cujos nós seriam exclusivamente nós dos tipos folha e soma, como mostra a figura 3.3.

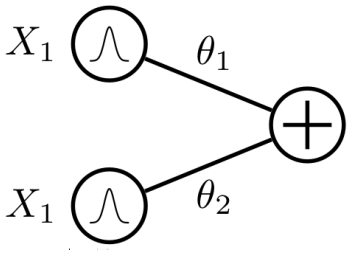

Figura 3.3: Um Circuito Probabilístico representando um Modelo de Misturas. Fonte: [4].

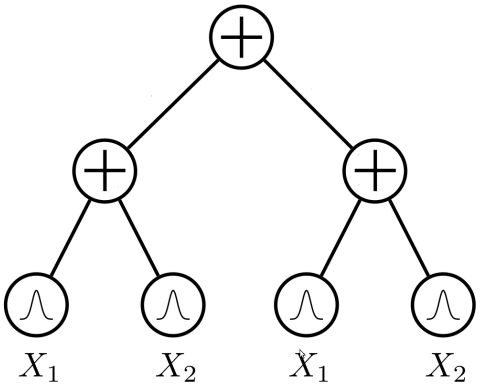

Figura 3.4: Um Circuito Probabilístico representando um Modelo de Misturas Hierárquico, com os pesos das arestas omitidos. Fonte: adaptação de [4].

3.3 Inferência sobre PCs

A motivação principal para a construção de modelos probabilísticos é a possibilidade de utilizar a distribuição de probabilidades que este fornece para realizar tarefas distintas. O fato de um modelo probabilístico fornecer de maneira explícita a PDF de nossa população permite que utilizemos o mesmo modelo para solucionar mais de um problema, ou, melhor dizendo, realizar mais de um tipo de inferência. É destas inferências, tais como as condições para realizá-las, que trataremos neste momento. A princípio, cabe um exemplo simples de inferência EVI (ou inferência de instância completa) para o circuito exemplificado na figura 3.1. Primeiramente, definimos esta classe de inferência:

Definição 3.3.1. Inferência EVI

Seja M um modelo probabilístico sobre as VAs $\mathbf{X}$ e $f_M(\mathbf{x})$ a função densidade de probabilidade conjunta que este modelo descreve (contínua ou discreta). Dizemos que as inferências da classe de inferência EVI são aquelas que calculam $f_M(\mathbf{y})$ para qualquer instância completa $\mathbf{y} \in val(\mathbf{X})$.

Denotaremos a classe de inferência EVI por $\mathcal{Q}_{EVI}$.

Exemplo 3.3.2. Inferência EVI

Suponha que para a figura 3.1, temos as seguintes distribuições ordenadas de cima para baixo: $\mathcal{N}(-1,2)$ e $\mathcal{N}(-2,0.1)$ para a VA X_1; $\mathcal{N}(0.6,0.1)$ e $\mathcal{N}(0,1)$ para X_2; $\mathcal{N}(1.5,0.2)$ e $\mathcal{N}(1.0,0.5)$ para X_3; $\mathcal{N}(0,1)$ e $\mathcal{N}(0,0.1)$ para X_4.

Então, avaliamos qual será a imagem da PDF para a instância completa $\mathbf{x} = (-1.85, 0.5, -1.3, 0.2)$. Isto é feito por uma realização dos nós folhas à raiz, efetuando as operações dos respectivos nós. Ao fim, o valor de nossa inferência EVI será dado pela saída do nó raiz (neste caso, o nó soma indicado em laranja na imagem 3.1), como mostrado na figura 3.5.

Ao fim, obtemos como valor $f_M(\mathbf{x}) = 0.75$. Vale ressaltar, este valor **não** representa uma probabilidade para VAs contínuas, como é o caso aqui (representaria se todas as VAs fossem discretas). Este número é apenas o valor que a função do modelo assume na instância específica $\mathbf{x}$. Este valor, apesar de não ser uma probabilidade, pode ser útil caso se tenha outras informações à respeito de sua PDF; se soubermos, por exemplo, qual a instância $\mathbf{x}_{MAP}$ associada à moda e seu respectivo valor, podemos avaliar qualquer instância em comparação a $\mathbf{x}_{MAP}$, e se esta se encontra próxima da moda da distribuição.

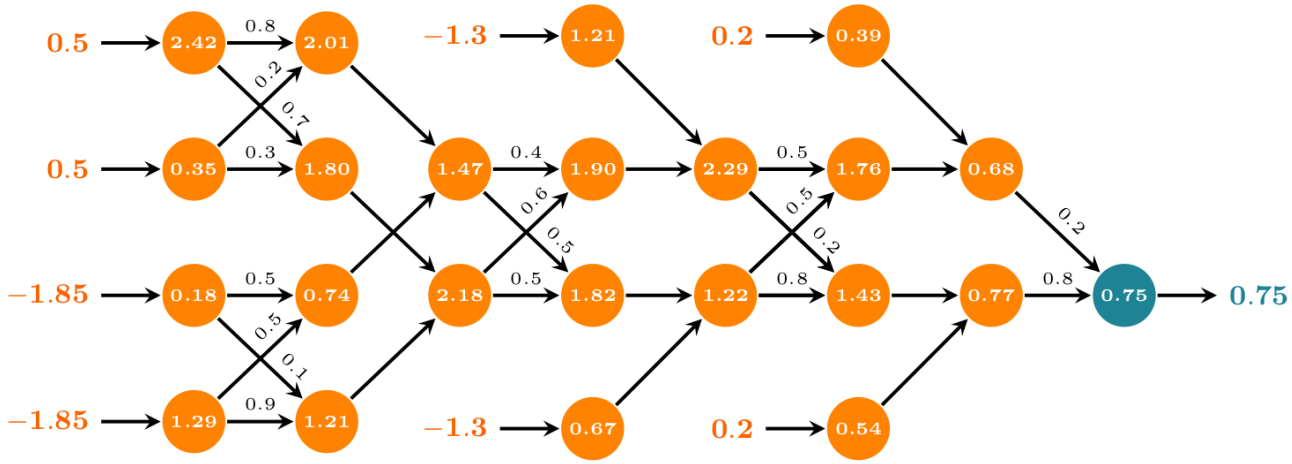

Figura 3.5: Inferência EVI realizada sobre um PC. Fonte: [4].

Para a realização de diferentes inferências sobre este modelo probabilístico, fazem-se ne-

cessárias algumas propriedades atribuidas à estrutura do PC. Para descrevê-las, cabe, primeiramente, definir a **função escopo**:

Definição 3.3.3. Função Escopo

Seja $\mathcal{C}$ um circuito probabilístico sobre $\mathbf{X}$, e $\mathcal{V}$ o conjunto de nós de $\mathcal{G}$, estrutura de $\mathcal{C}$. Definimos $sc : \mathcal{V} \rightarrow \mathcal{P}(\mathbf{X})$ a função que satisfaz as condições:

1. se $n \in \mathcal{V}$ é um nó folha que descreve uma distribuição de $\mathbf{U} \subseteq \mathbf{X}$, então $sc(n) = \mathbf{U}$;
2. se $n \in \mathcal{V}$ é um nó produto ou soma, $sc(n) = \bigcup_{m \in ch(n)} sc(m)$, sendo $ch(n)$ o conjunto de nós filhos de n.

Agora, enumeramos propriedades estruturais de um PC, ressaltando em seguida a importância de cada uma para a realização de um tipo específico de inferência: a da classe MAP. São elas **decomponibilidade e determinismo** [4].

Definição 3.3.4. Decomponibilidade

Um nó produto n de um PC $\mathcal{C}$ é dito decomponível se os escopos dos filhos de n são dois-a-dois disjuntos, isto é, $sc(m) \cap sc(p) = \emptyset \ \forall m, p \in ch(n)$. Um Circuito Probabílistico $\mathcal{C}$ é dito decomponível se todos os seus nós produto são decomponíveis.

Definição 3.3.5. Determinismo

Um nó soma n de um PC $\mathcal{C}$ é dito determinístico se no máximo um de seus filhos tem como saída um valor não nulo para qualquer entrada totalmente instanciada (ou seja, qualquer entrada que é um estado completo). Um Circuito Probabilístico $\mathcal{C}$ é dito determinístico se todos os seus nós soma são determinísticos.

As duas características definidas são propriedades suficientes em um PC para a computação de inferência MAP de maneira exata em tempo linear [3, 12], como mostrou Chan e Darwiche [9]. Verificaremos em seguida, de forma mais detalhada, porque a omissão destas restrições implicam em uma falha do cálculo da inferência MAP.

3.4 Inferência MAP

A inferência MAP é particularmente interessante do ponto de vista estatístico, pois ela dará a instância de maior probabilidade da distribuição em questão (ou a maior moda da distribuição). Para tarefas preditivas, esta instância será extremamente útil, por representar a melhor predição possível. Por exemplo, se quisermos efetuar a tarefa de reconstrução de imagens utilizando

PCs, precisamos encontrar o estado mais provável dos *pixels*. Isto é feito justamente utilizando a inferência MAP.

Para descrever como é realizada esta inferência e demonstrar que as condições postas são suficientes, introduzimos primeiramente as seguintes definições associadas à estrutura de um PC:

Definição 3.4.1. Função Maximizadora de Distribuição

Seja $\mathcal{C}_L$ uma função de entrada de um PC associada a um nó folha n. Então, sua função maximizadora de distribuição, denotada por $\mathcal{C}_L^{max}$ calcula

$$\max_{\mathbf{y} \in val(\mathbf{Y})} C_L(\mathbf{y}) \text{ onde } \{\mathbf{Y}\} = sc(n). \tag{3.6}$$

Definição 3.4.2. Circuito Maximizador

Seja $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ sobre as VAs $\mathbf{X}$. Então, seu Circuito Maximizador será $\mathcal{C}_{max} = (\mathcal{G}_{max}, \boldsymbol{\theta}_{max})$ obtido pela substituição de cada nó soma $n \in \mathcal{G}$ por um nó de maximização, isto é, aquele que computa a função $\max_{m \in ch(n)} \theta_m v(m)$ em que θ_m é o peso da aresta entre m e n e $v(m)$ é a saída do nó m; e $\boldsymbol{\theta}_{max}$ é obtido substituindo cada nó folha por sua função maximizadora de distribuição.

Com estas definições, podemos mostrar que, para um PC decomponível e determinístico, o circuito maximizador obtido calculará o **estado MAP** deste circuito [4]. A maneira como ele fará isso se assemelha à inferência EVI; no entanto, utiliza-se o circuito maximizador para isto. Neste caso, é realizada uma avaliação do circuito maximizador, onde serão obtidos, primeiramente, os valores de máximos das distribuições dos nós folha, e subsequentemente os valores de máximos dos próximos nós. Ao final, será obtido o valor que maximiza o nó raiz. Se desejamos verificar a moda correspondente ao estado $\mathbf{x}_{MAP}$, basta realizar uma inferência EVI para a instância encontrada, já que esta também é tratável. O processo para encontrar a instância MAP está também descrito de forma algorítmica no Apêndice A.

Por fim, resta explicar como as restrições estruturais descritas são importantes para a utilização do circuito maximizador. Primeiramente, em relação ao determinismo:

Exemplo 3.4.3. Tomemos um circuito descrito como o da figura 3.3, onde $\theta_1 = 0.2, \theta_2 = 0.8$ e as distribuições de X_1 são, de cima para baixo, $\mathcal{N}(4,1)$ e $\mathcal{N}(5,1)$. O nó soma presente claramente não é determinístico, já que os dois nós folhas descrevem a VA X_1; logo, para qualquer entrada totalmente instânciada, $x \in val(X_1)$, as saídas de ambos os nós folhas serão não nulas. Com isto, efetuamos a abordagem do circuito maximizador: tomamos os valores das modas de cada nó folha (são elas, $x = 4$ e $x = 5$). Em seguida, calculamos a saída do nó soma, dada por $0.2 \times 0.399 + 0.8 \times 0.399 = 0.399$, como vemos na figura 3.6.

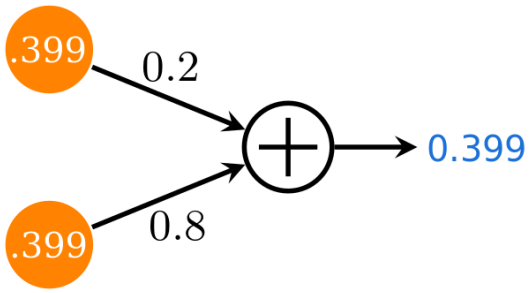

Figura 3.6: Tentativa de Inferência MAP em um PC não determinístico. Fonte: adaptação de [4].

Porém, note que se $f(x)$ e $g(x)$ são as PDFs descritas acima, então a saída do nó soma será $h(x) = 0.2 \times f(x) + 0.8 \times g(x)$. Com isto, podemos calcular o máximo desta distribuição, $\max_{x \in val(X_1)} h(x) = 0.371$, que é diferente do valor encontrado pela abordagem do circuito maximizador. Esta diferença é ilustrada pela PDFs representadas na figura 3.7.

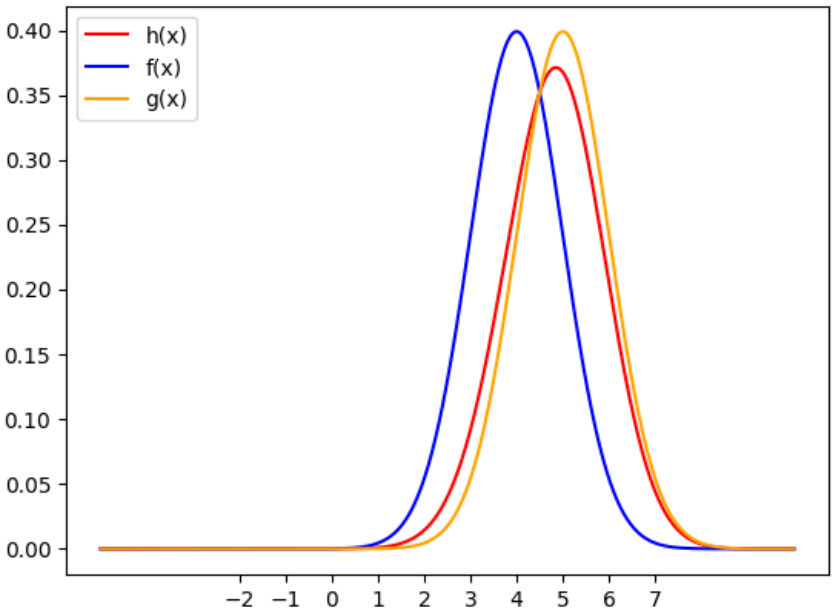

Figura 3.7: PDFs de $f(x), g(x)$ e $h(x)$.

Isto evidencia que não podemos executar a abordagem de um circuito maximizador neste caso, o que mostra a importância da restrição de nós determinísticos: É necessário que para toda instância tenhamos apenas uma saída não nula para evitar este caso.

Quanto à decomponibilidade, esta se faz necessária por uma razão um pouco mais evidente. Esta impõe uma restrição sobre os escopos de um nó produto, onde as funções obtidas por meio de seus filhos vão ser compostas por VAs distintas, o que implica que a maximização do produto

será apenas o produto da maximização. Em outras palavras, se $\mathcal{C}_1, ..., \mathcal{C}_m$ são subcircuitos filhos de um nó produto n decomponível, a maximização deste é dada simplesmente por:

$$\max_{\mathbf{x} \in val(\mathbf{X})} n(\mathbf{x}) = \max_{\mathbf{x_j} \in val(\mathbf{X_j})} \prod_{j=1}^{m} C(\mathbf{x_j}) = \prod_{j=1}^{m} \max_{\mathbf{x_j} \in val(\mathbf{X_j})} C(\mathbf{x}_j) \tag{3.7}$$

Isto conclui a importância de tais características, e por consequência o porquê de serem conservadas ao construir um PC, tópico que será abordado no capítulo seguinte.

Capítulo 4

Aprendizado de Estruturas

O **Aprendizado Estrutural** de PCs é o processo de obtenção de um circuito a partir do conjunto de dados previamente obtido. Supõe-se a construção do PC para a posterior realização de inferências, por isso se faz necessário a garantia de qualidade do circuito, ou seja, espera-se que ele represente uma distribuição de probabilidades adequada às variáveis do conjunto de dados. Ainda, espera-se que, se possível, o circuito obtido possa conter as propriedades descritas no capítulo anterior, visando à garantia de tratabilidade destas inferências.

A tarefa de encontrar uma estrutura de um PC adequada a um conjunto de dados, entretanto, não é algo trivial: não é claro quais operações devem ser realizadas sobre o conjunto de dados – e em qual ordem devem ser efetuadas – para a obtenção desta estrutura. A depender das escolhas das operações sobre seu conjunto, serão obtidos circuitos (e consequentemente PDFs) diferentes. Mais do que isso: se o conjunto de dados for inadequadamente explorado, a estrutura obtida possivelmente representará uma PDF inapropriada para a população em questão. Com isto, torna-se evidente a importância do desenvolvimento de técnicas para a construção de estrutura de um PC: o conjunto de dados deve ser explorado da melhor forma possível para que se obtenha, por conseguinte, a PDF mais adequada aos dados.

Na literatura, diversas técnicas foram exploradas para o aprendizado estrutural de PCs [7, 8, 13, 10]. A técnica mais extensamente estudada e aqui implementada foi o algoritmo, ou esquema de algoritmo, proposto por Gens e Domingos [8] e aprimorado por Vergari [15], denominado **LearnSPN**. Este obtém um circuito probabilístico a partir de um particionamento recursivo do conjunto de dados, comumente representado por uma matriz, onde as colunas representam as diferentes variáveis e as linhas representam as diferentes instâncias (ou amostras). O LearnSPN se utiliza de técnicas diferentes para a segmentação de colunas e de linhas, criando nós de diferentes tipos, mais adequados para cada divisão. Na seção abaixo, está descrito em detalhes o processo para a criação de nós utilizado pelo algoritmo:

4.1 LearnSPN

Criação de um Nó Folha

Se a matriz de entrada possui apenas uma coluna, um nó folha que representa uma distribuição univariada é criado, cujas probabilidades são determinadas pela frequência de cada instância, realizando uma suavização de Laplace com a utilização de uma constante pré-determinada.

Criação de um Nó Produto

Se a matriz de entrada possui mais de uma coluna, o algoritmo tenta criar um nó produto. Isto é feito com o auxílio de uma estrutura de grafos: cada coluna é representada por um vértice, e realiza-se um teste de independência para cada dupla de variáveis; se o teste aponta a uma dependência, seus vértices correspondentes na estrutura de grafos são conectados por uma aresta. Para a representação da estrutura de grafos e contagem de componentes conexas, foi implementada a estrutura de dados conhecida como Union-Find (explicada no Apêndice) utilizando as devidas heurísticas visando à otimização (compressão de caminhos e união pelo rank) [14]. Já para a verificação de (in)dependência das colunas, foi implementado o teste de hipótese G (ou *G-test*), adequado apenas a variáveis categóricas, e realizado duas-a-duas variáveis.

Ao fim deste processo, obtém-se um grafo $\mathcal{G}$. Em seguida, verifica-se, ainda com o Union-Find, a quantidade de componentes conexas deste grafo: se $\mathcal{G}$ possui mais de uma componente conexa, digamos $G_1, ..., G_m$, onde as variáveis de cada componente G_j são $V_{1,j}, ..., V_{n,j}$ um nó produto é criado, e seus filhos serão os sub-circuitos $S_1, ..., S_m$. Cada sub-circuito S_j é obtido utilizando a matriz de colunas $V_{1,j}, ..., V_{n,j}$. Se $\mathcal{G}$ possui somente uma componente conexa, entende-se, pois, que todas as variáveis são dois a dois dependentes e realiza-se a criação de um nó soma, descrita a seguir.

Cabe notar que, devido ao processo de segmentação de variáveis por verificação de independência realizado pelo LearnSPN, todo nó produto criado por ele é necessariamente decomponível; logo, todo circuito gerado por este algoritmo é decomponível.

Criação de um Nó Soma

Se a matriz de entrada possui mais de uma linha e a criação de um nó produto não é possível, cria-se obrigatoriamente um nó soma, por meio de uma clusterização. O algoritmo de clusterização implementado nesta etapa foi o K-means. Se as instâncias são similares a ponto do algoritmo K-means não conseguir diferí-las, então, este realiza uma clusterização aleatória sobre as instâncias. Então, necessariamente, ao fim deste processo ter-se-á K clusters (ou classes), cada qual contendo

uma quantidade de linhas da matriz. Suponha que $\mathcal{C}_j$ para $j \in \{1, ...K\}$ representam conjuntos de linhas do conjunto de dados, cada um representando um cluster. Com isto, o nó soma é criado: seus filhos são sub-circuitos $S_1, ..., S_m$ onde S_j é o sub-circuito criado utilizando a matriz cujas linhas são as do conjunto $\mathcal{C}_j$.

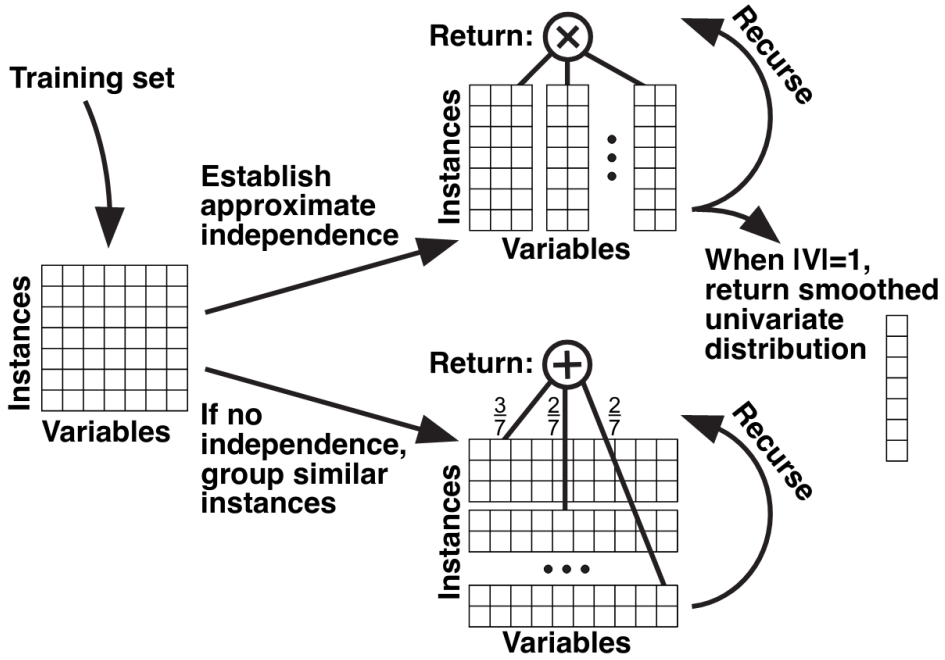

Figura 4.1: Esquema ilustrativo de obtenção de um PC pelo LearnSPN. Fonte: [8]

4.2 Aprimorações do LearnSPN

Verificou-se que o algoritmo constrói um circuito probabilístico por meio de uma estratégia gulosa (ou, do inglês, **greedy**) [15], o que significa que o LearnSPN toma a decisão ótima localmente (ou seja, a cada vez que é chamado). Consequentemente, isto, por vezes, resulta em *overfitting*, isto é, um circuito sem grande capacidade de generalizar os dados. Não só isso, mas esta característica gera um circuito demasiadamente grande (com muitos nós), o que torna inferências mais computacionalmente custosas e, por conseguinte, mais demoradas. Visando à obtenção de circuitos menores e que melhor representem o conjunto de dados, algumas técnicas descritas na literatura [15] foram tomadas, além de um necessário ajuste de hiperparâmetros a depender do conjunto de dados, este último detalhado na última seção deste capítulo.

A primeira delas é uma delimitação de divisões para a criação de um nó soma. Isto significa que o algoritmo K-means fica restrito à criação de apenas dois clusters (ou duas classes). Esta restrição evita que o circuito gerado possua nós soma com muitos filhos, e tem como objetivo limitar a característica gulosa do algoritmo, fazendo com que todas as instâncias do conjunto de

dados sejam melhor exploradas. Consequentemente, esta decisão resulta na criação de circuitos menos "largos"e mais profundos.

A segunda aprimoração realizada foi restringir a criação do primeiro nó a ser um nó soma. Isto impede que ocorra o caso degenerado em que todas as variáveis seriam consideradas independentes, implicando que, para n colunas o circuito seria composto somente por um nó produto e n nós folhas, seus filhos. É interessante que este caso seja evitado por ser um modelo grosseiramente simplificado, já que dificilmente todas as variáveis de um conjunto de dados real seriam duas-a-duas independentes. Apesar disso, é um caso que o LearnSPN consideraria plausível se o conjunto de dados em questão tiver poucas instâncias e que deve, portanto, ser evitado.

4.3 Algoritmo Modal EM

Outra possível aprimoração ao LearnSPN é o algoritmo **Modal EM**. Idealizado para misturas de distribuições de probabilidade gaussianas, ou, ainda, circuitos cujos nós folhas são distribuições contínuas, o algoritmo proposto por Li, Ray, e Lindsay [11] efetua uma busca para encontrar máximos locais da função em questão, ou seja, as modas da distribuição. Esta busca é realizada por meio de uma estratégia de *hill climbing*, inicializada a partir de um ponto aleatório $\mathbf{x}$ instância completa deste circuito e, por meio de sucessivas iterações, este ponto convergirá a uma moda da distribuição.

O algoritmo descrito pode ser de extrema utilidade para a realização de uma tarefa semelhante à de inferência MAP em circuitos cujos nós folhas são distribuições gaussianas. Em vez de realizar uma avaliação das folhas à raiz e encontrar apenas um estado MAP, pode-se utilizar o Modal EM inicializando-o em diversos pontos $\mathbf{x} \in \mathbf{X}$, sendo $\mathbf{X}$ o conjunto de VAs do PC em questão. Desta forma, seriam encontradas diversas modas da distribuição. Apesar de nenhuma delas (exceto uma) estar associada ao estado MAP, elas poderiam ser úteis para tarefas preditivas para as quais uma quantidade maior de instâncias mais prováveis (e suficientemente distintas) fosse necessário.

Outra motivação compatível com o LearnSPN para estudar esta aprimoração seria o fato de que, pela sua técnica de *hill climbing* para cada ponto, o algoritmo Modal EM realiza implicitamente uma clusterização da nossa amostra, em que cada cluster é associado a uma moda. Em especial para o LearnSPN, um algoritmo guloso, seria especialmente interessante redução estrutural do circuito, tentando conceder menos peso às modas menores (de menor importância para o modelo) por meio da clusterização relatada [11]. Com isto, seria possível a criação de um circuito que generalize cada vez mais os dados a partir do seguinte processo:

- Cria-se um PC $\mathcal{C}$ a partir do conjunto de dados, utilizando o LearnSPN;
- Utiliza-se o Modal EM para clusterizar os pontos do conjunto de dados entre os clusters $K_1, ..., K_n$ a partir de n modas da distribuição;
- Considera-se a possibilidade de descarte de um cluster K_p, se seu tamanho for menor que um valor determinado. Neste caso, poucos pontos estariam associados à moda que gerou o cluster K_p o que seria evidência de que não seria uma moda relevante ao modelo. Em outras palavras, pode-se dizer que esta moda "pequena"teria sido gerada apenas por ruído de nossa amostra;
- Para cada K_j não descartado, gera-se um novo subcircuito $\mathcal{C}_j$. Cria-se também um nó soma (raiz), cujos filhos serão $\mathcal{C}_1, ...\mathcal{C}_m$. e então atualiza-se o circuito $\mathcal{C}$ para este novo, descrito pela soma convexa dos subcircuitos gerados.

É relevante ressaltar que o processo descrito acima pode ser realizado quantas vezes forem necessárias. Há uma desvantagem evidente, entretanto, que é a geração de diversos circuitos. De fato, este processo pode ser demasiadamente demorado, a depender do conjunto de dados. No entanto, este traz consigo, aparentemente, a vantagem de mitigar o possível *overfitting* associado ao LearnSPN. A clusterização por modas poderia, em certa medida, "limpar"nosso modelo de ruídos provenientes da amostras, e gerar, por conseguinte, um PC mais robusto e que melhor representaria a população em questão, contribuindo em muito para posteriores inferências.

4.4 Resultados de Testes

O algoritmo LearnSPN foi implementado na linguagem *Julia* se utilizando da biblioteca de código aberto elaborada pelo grupo de pesquisa do orientador [6]. Os testes foram realizados sobre conjuntos de dados reais de dados categóricos (binários, mais especificamente) disponibilizados publicamente e obtidos em [5]. Para cada conjunto de dados, foi, primeiramente, construído um PC e avaliado o tempo de construção deste circuito. A Tabela 1 indica este tempo, junto à dimensão da matriz de treino de cada conjunto de dados.

O segundo teste realizado foi em relação à inferência MAP, indicado pela Tabela 2. Para cada conjunto de dados, outros PCs foram gerados, agora com a variação dos hiperparâmetros, que são p-valor do teste G e constante de suavização de Laplace. A coluna **Tempo** indica o tempo minímo (para toda combinação de hiperparãmetros) para obtenção do valor da coluna Log-likelihood; em **Hiperparâmetros** estão indicados os p-valor e constante de Laplace, respectivamente; a coluna **Número de Nós** indica o número de nós de cada circuito para cada hiper-

parâmetro; a coluna **Log-likelihood** é o valor obtido na maximização do *log-likelihood*, que é o logaritmo da PDF conjunta computada pelo PC para instâncias completas, isto é

$$\log\{\max_{\mathbf{x}\in val(\mathbf{X})} \mathcal{C}(\mathbf{x})\} \text{ onde } \mathcal{C} \text{ é o PC obtido.}$$

Na coluna **PoonDomingos** estão reportados os valores de maximização do *log-likelihood* obtidos por Poon e Domingos, enquanto as linhas em negrito na mesma tabela indicam o melhor ajuste de hiperparâmetros, isto é, o conjunto (p-valor, constante de Laplace) que gerou uma inferência mais próxima dos resultados reportados por Poon e Domingos.

Os testes foram realizados em um computador cujas configurações são: Intel i7-2600 64bits 3.40GHz com 4 *cores*, 8GB e swap de 16GB.

Conjunto de Dados	Dimensão do Conjunto de Dados	Tempo de Treinamento
NLTCS	16×16181	342.873s
plants	69×17412	258.847s
DNA	180×1600	148.452s

Tabela 1: Resultados do Aprendizado Estrutural.

Dataset	Tempo	Hiperparâmetros (p-valor, const. Laplace)	Número de Nós	Log-likelihood	Poon&Domingos
NLTCS	805.232ms	(0.0015, 0.2)	48990	-5.463	-6.110
		(0.0015, 0.4)	50750	-4.996	
		(0.0015, 0.6)	50272	-7.663	
		(0.0015, 0.8)	57107	-5.972	
		(0.0001, 0.2)	54169	-5.533	
		(0.0001, 0.4)	60451	-5.005	
		(0.0001, 0.6)	46025	-6.247	
		(0.0001, 0.8)	**49660**	**-6.010**	
plants	5.572s	(0.0015, 0.2)	214607	-9.314	-12.977
		(0.0015, 0.4)	**248605**	**-10.530**	
		(0.0015, 0.6)	214275	-5.794	
		(0.0015, 0.8)	225673	-6.612	
		(0.0001, 0.2)	207879	-8.256	
		(0.0001, 0.4)	205354	-8.144	
		(0.0001, 0.6)	233813	-9.945	
		(0.0001, 0.8)	217033	-8.078	
DNA	2.688s	(0.0015, 0.2)	170053	-13.335	-82.523
		(0.0015, 0.4)	172876	-22.330	
		(0.0015, 0.6)	172705	-28.841	
		(0.0015, 0.8)	**172157**	**-31.913**	
		(0.0001, 0.2)	170810	-13.090	
		(0.0001, 0.4)	171824	-19.101	
		(0.0001, 0.6)	171842	-23.281	
		(0.0001, 0.8)	171809	-27.996	

Tabela 2: Resultados da inferência MAP.

Apêndice A

Apêndice

Algorithm 1 Inferência MAP sobre um PC

Input: um PC $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ sobre $\mathbf{X}$.
Output: $\max_{\mathbf{x} \in val(\mathbf{X})} \mathcal{C}(\mathbf{x})$
  $N \leftarrow OrdemFeedForward(\mathcal{G})$

  for each $n \in N$ **do**
    if n é nó folha **then**
      $U \leftarrow sc(n)$
      $r_n \leftarrow max_{u \in val(U)} v_n(u)$ ▷ $v_n(u)$ é o output do nó n para input u
    else if n é nó produto **then**
      $U_m \leftarrow sc(m)\ \forall m \in ch(n)$
      $r_n \leftarrow \max \prod_{m \in ch(n)} v_m(u_m)$
    else if n é soma **then**
      $U_m \leftarrow sc(m)\ \forall m \in ch(n)$
      $r_n \leftarrow \max_{m \in ch(n)} \theta_m v_m(u_m)$
    end if
  end for
  return r_n

Algorithm 2 Find (Encontra a raiz de nós)

Input: $u \in \mathcal{G}$, parent(u)
▷ parent(u) é o nó do qual u é filho.
▷ Se u não é filho de nenhum outro nó, então dizemos que $parent(u) = u$.
  if parent$(u) = u$ **then**
    return u
  else if parent$(u) \neq u$ **then**
    parent$(u) \leftarrow Find($parent$(u))$ ▷ Execução da compressão de caminhos.
  end if

Algorithm 3 Union (União de nós)

Input: $u, v \in \mathcal{G}, ch(u), ch(v), rank(u), rank(v)$ ▷ $ch(u), ch(v)$ são listas dos filhos de cada nó. ▷ $rank(u), rank(v)$ são os ranks de cada nó.

```
raiz(u) ← Find(u)
raiz(v) ← Find(v)
if parent(u) ≠ parent(v) then                          ▷ União por rank dos nós.
    if rank(u) = rank(v) then
        parent(v) ← raiz(u)
        ch((raiz(u))+ = raiz(v)
        rank(u)+ = 1
    else if rank(u) > rank(v) then
        parent(v) ← raiz(u)
        ch(raiz(u))+ = raiz(v)
    else if rank(u) > rank(v) then
        parent(u) ← raiz(v)
        ch(raiz(v))+ = raiz(u)
    end if
end if
```

Os algoritmos 2 e 3 tratam da estrutura de operações sobre a estrutura de dados Union-Find. Trata-se da criação de um grafo em forma de árvore, em que efetuar-se-ão as operações de encontrar a raiz de um dado nó, e a de unir dois nós. O objetivo desta estrutura é utilizar a estrutura de árvore de um grafo para associar (ou desassociar) elementos de um conjunto, atribuindo nós para cada um destes elementos. Primeiro, cabe explicar o conceito de rank. **Rank** trata-se de um número associado a um nó u que mede quantos nós estão acima dele. No início, se dois nós u, v não possuem arestas ligadas a eles, e efetuamos a operação Union(u, v), então o rank de um deles será acrescido de 1.

Além disso, esta estrutura de dados, junto às suas operações *Union* e *Find* possui duas otimizações, elencadas abaixo:

- A primeira otimização, declarada **compressão de caminhos** está presente em *Find*. Esta trata de, ao encontrar a raiz de algum nó u por meio da função Find(u), tentar diminuir a altura desta árvore, atualizando parent(u) para raiz(u). Isto é feito como exemplifica a figura A.1.

- A segunda otimização, declarada **união por rank** está presente em *Union*. Esta, por sua vez, trata de unir dois nós de maneira a não aumentar desnecessariamente o rank de algum dos nós. Para isso, esta otimização evita que dois nós sejam unidos diretamente por uma aresta; em vez disso, suas raízes são unidas, e haverá um caminho entre ambos, como exemplifica a figura A.2.

Por fim, para contar a quantidade de componentes conexas, basta encontrar a raiz de cada

nó utilizando o Find(). O número de raizes corresponderá, por construção, ao número de componentes conexas do grafo em questão.

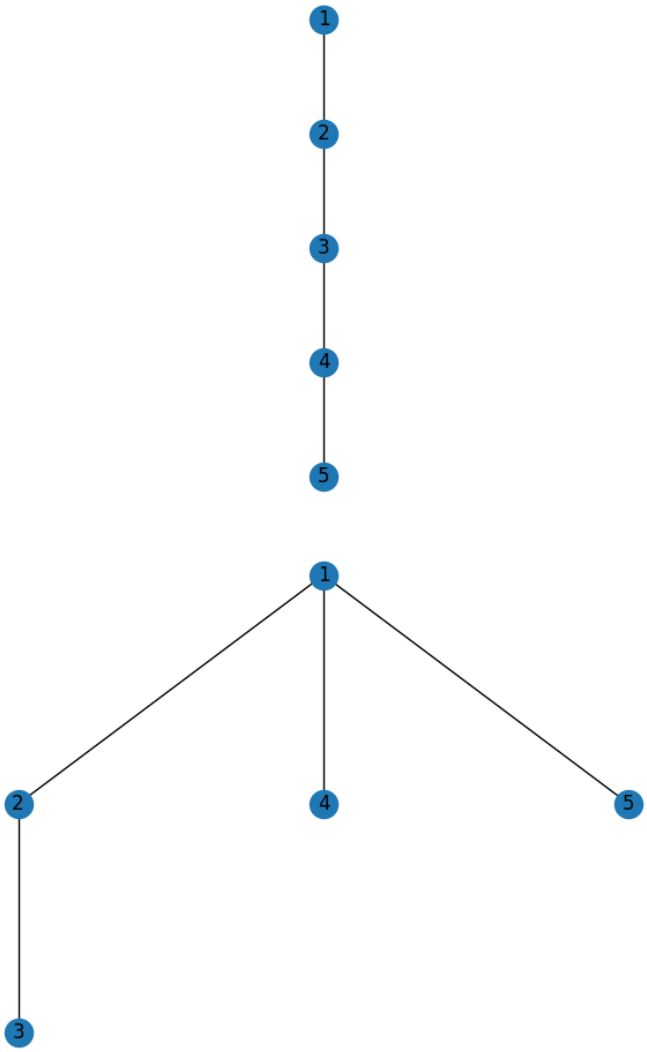

Figura A.1: Acima, um exemplo de árvore. Abaixo, o resultado das operações Find(4) e Find(5) com esta função implementada junto à compressão de caminhos.

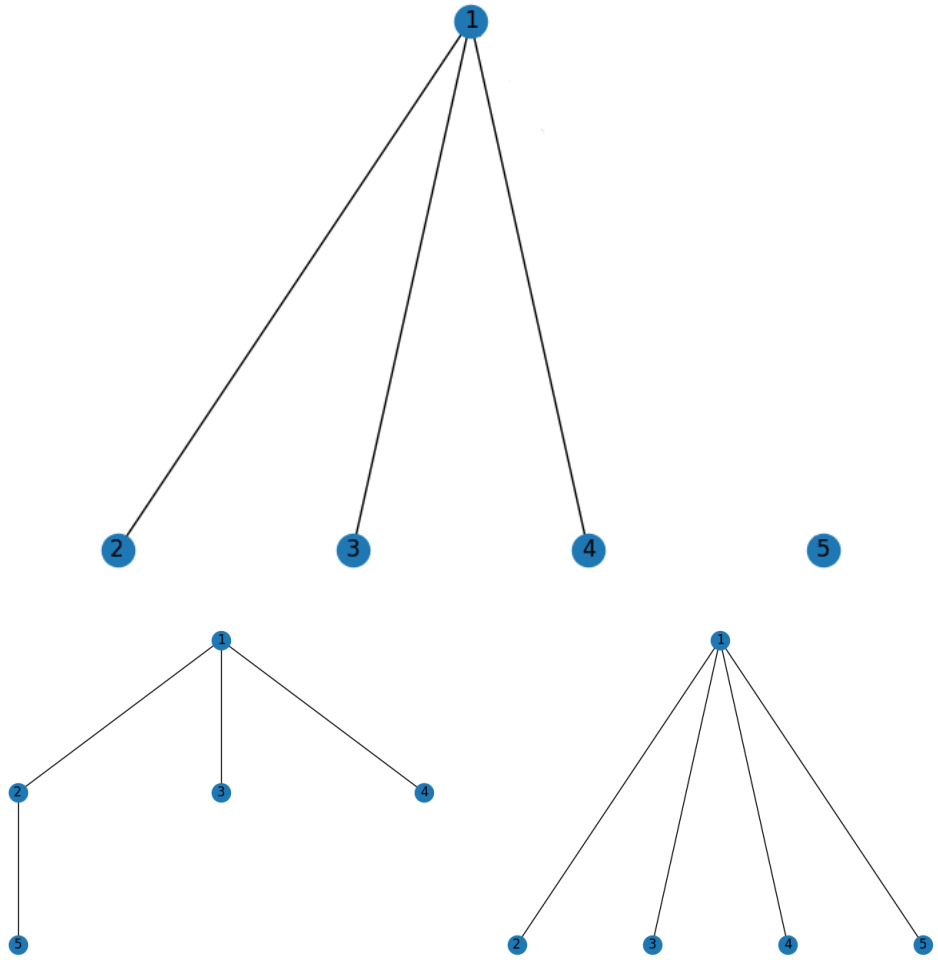

Figura A.2: Acima, um grafo antes de executar a função Union. Abaixo e à esquerda, o resultado das operações Union(2,5) sem a otimização de união por rank. Abaixo e à direita, o resutaldo desta mesma operação com a otimização união por rank.

Referências Bibliográficas

[1] ATHANASSIA G. BACHAROGLOU. Approximation of probability distributions by convex mixtures of gaussian measures. *Proceedings of the American Mathematical Society*, 138(7):2619–2628, 2010.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[3] Hei Chan and Adnan Darwiche. On the robustness of most probable explanations. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, page 63–71, Arlington, Virginia, USA, 2006. AUAI Press.

[4] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models, 2022.

[5] Guy Van den Broeck. `https://github.com/UCLA-StarAI/Density-Estimation-Datasets`.

[6] Renato Geh and Denis Mauá. `https://github.com/RenatoGeh/RPCircuits.jl`.

[7] Renato Lui Geh. Scalable learning of probabilistic circuits. Master's in computer science dissertation, University of São Paulo, April 2022.

[8] Robert Gens and Domingos Pedro. Learning the structure of sum-product networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 873–880, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[9] Jinbo Huang, Mark Chavira, and Adnan Darwiche. Solving map exactly by searching on compiled arithmetic circuits. volume 2, 01 2006.

[10] Priyank Jaini, Amur Ghose, and Pascal Poupart. Prometheus : Directly learning acyclic directed graph structures for sum-product networks. In *Proceeding of the International Conference on Probabilistic Graphical Models (PGM)*, volume 72, pages 181–192, 2018.

[11] Jia Li, Surajit Ray, and Bruce Lindsay. A nonparametric statistical approach to clustering via mode identification. *Journal of Machine Learning Research*, 8:1687–1723, 08 2007.

[12] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(10):2030–2044, October 2017.

[13] Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect interactions. In *Proceedings of the Thirty-First International Conference on Machine Learning (ICML)*, 2014.

[14] Ronald L. Rivest Clifford Stein Thomas H. Cormen, Charles E. Leiserson. *Introduction to algorithms*. MIT Press, 2001.

[15] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening sum-product network structure learning. In Annalisa Appice, Pedro Pereira Rodrigues, Vítor Santos Costa, João Gama, Alípio Jorge, and Carlos Soares, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 343–358, Cham, 2015. Springer International Publishing.