ESTUDO SOBRE SUM-PRODUCT NETWORKS E APRENDIZAGEM PROFUNDA

PTC2669 - INTRODUÇÃO A INTELIGÊNCIA COMPUTACIONAL
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA - USP

RELATÓRIO 2

RENATO LUI GEH
NUSP: 8536030

Resumo. Modelos probabilísticos baseados em grafo representam uma distribuição de probabilidade de forma compacta. Apesar de bastante expressivos, os modelos clássicos que conhecemos ou tem inferência intratável e aprendizado difícil, como Redes Bayesianas e Redes de Markov, ou são muito restritas com relação a sua representatividade. Sum-Product Networks (SPNs) é uma classe de modelos que representa uma distribuição de probabilidade tratável e tem inferência exata e tratável. Além disso, SPNs se mostram bastante interessantes devido a sua arquitetura profunda. Quanto mais camadas forem adicionadas não somente aumentamos a representatividade do modelo como também mantemos a tractabilidade da inferência.

1. Motivação

Modelos probabilísticos clássicos baseados em grafos, tais como Redes Bayesianas e Redes de Markov, conseguem representar uma grande variedade de distribuições de probabilidade. No entanto, tais modelos tem sua performance reduzida quando a representação requer uma largura de árvore muito grande. Além disso, devido a sua intractabilidade quanto a inferência exata, muitas vezes devemos utilizar algoritmos aproximados para computar inferência. Como aprendizado usa inferência como subrotina, a verossimilhança durante o aprendizado pode acabar sofrendo devido às técnicas de aproximação, o que potencialmente não ocorreria se tivéssemos a possibilidade de computar inferência exata em tempo subexponencial.

Recentemente houve um grande foco em arquiteturas profundas, onde o modelo apresenta várias camadas ocultas onde cada variável nesta camada age como uma variável latente, o que pode amenizar o problema da largura de árvore e ainda assim manter uma alta representabilidade [Ben09] No entanto, modelar tais arquiteturas é uma tarefa difícil. Além do mais, tanto inferência quanto aprendizado tornam-se ainda mais difíceis.

Redes Soma-Produto (RSPs), conhecidas como Sum-Product Networks (SPNs) na literatura, são modelos probabilísticos baseados em grafos onde o problema da largura de árvore é solucionado por meio de variáveis latentes e camadas ocultas. SPNs são interessantes pois apesar de serem uma arquitetura profunda, ainda assim

mantém inferência exata e tratável. Além disso, são modelos mais gerais que os anteriormente propostos para distribuições tratáveis.

2. Sum-Product Networks

Uma SPN é um DAG em que todo subgrafo enraizado em um vértice arbitrário é uma SPN. Iremos definir SPNs abaixo.

Definição 2.1. *Uma SPN tem uma definição recursiva. Definimos que uma SPN S_i pode ser apenas:*

(i) Uma distribuição monovariável $p(\mathbf{X})$ ou;
(ii) Um nó soma tal que $S_i = \sum_{j \in Ch(i)} w_{ij} v_j$ onde para cada filho $j, k \in Ch(i)$, $Sc(S_j) = Sc(S_k)$ ou;
(iii) Um nó produto tal que $S_i = \prod_{j \in Ch(i)} v_j$ onde para cada filho $j, k \in Ch(i)$, $Sc(S_j) \cap Sc(S_k) = \emptyset$.

Intuitivamente, SPNs podem ser vistas como redes em que capturamos semelhanças entre instâncias e independência entre variáveis. Nós somas representam semelhança entre os dados, já que cada filho de um nó soma é apenas uma mistura de distribuições com mesmo escopo. Já nós produtos podem ser vistos como uma relação de independência entre as variáveis de cada escopo de cada filho do nó. Note que cada filho de um nó produto é independente de outro filho do mesmo nó pai, já que seus escopos são disjuntos. Nós folhas são definidos como distribuições monovariáveis, no entanto é possível estender a definição para distribuições multivariáveis.

Inferência é natural à estrutura de uma SPN. Computar a probabilidade de evidência de uma SPN resume-se a computar o valor de cada nó em formato *top-down*, com a configuração das folhas da SPN de reguladas para que sejam consistentes com a evidência. Computar a probabilidade marginal é semelhante, porém determinamos que folhas cujos escopos foram eliminados tenham valores 1. Para computarmos o MAP de uma SPN ao invés de termos as folhas com valores 1 quando não existe evidência determinamos como a valoração mais provável da distribuição como valor da folha, ou seja uma valoração tal que sua probabilidade seja maximal na distribuição.

3. Aprendizado

Como nós somas são relações de semelhança entre as instâncias dos dados, podemos tomar cada filho de um nó soma como um *cluster* de semelhança. Podemos aplicar *k-means clustering* para separarmos k clusters. Cada um desses k clusters será um filho de um nó soma e terão mesmo escopo.

Nós produtos representam independência entre variáveis. Para separarmos as variáveis em conjuntos onde toda variável em um conjunto é independente de uma variável em outro conjunto podemos criar um grafo de independência:

Definição 3.1. *Um grafo de independência é um grafo $G = (X, E)$ com variáveis X como vértices e arestas $E = \{e_{ij} : X_i — X_j \iff X \not\perp Y\}$.*

Para acharmos as independências par-a-par podemos utilizar o teste de independência por Chi-Quadrado ou por Entropia.

Proposição 3.1. *Seja um grafo de independência $G = (X, E)$, os k-subgrafos $H_0, \ldots, H_k \subseteq G$ desconexos tem escopos independentes par-a-par.*

Desta forma, podemos identificar os subgrafos desconexos e atribuir cada um desses conjuntos disjuntos a um filho do nó produto.

Como nós folhas são apenas distribuições monovariáveis, podemos aprender nós folhas por meio da técnica tradicional de MLE, que resume-se a adquirir as contagens e normalizarmos para obtermos as frequências de cada possíveis valorações da variável.

Algoritmo 1 LearnSPN [GD13]

Input Conjunto $\mathbf{X}$ de variáveis, conjunto $\mathbf{I}$ de instâncias
Output Uma SPN resultante do aprendizado estrutural

if $|\mathbf{X}| = 1$ **then**
    Retorna uma distribuição monovariável de $\mathbf{X}$
end if
Tente dividir as variáveis $\mathbf{X}$ em q partições $\mathbf{X}_1, \ldots, \mathbf{X}_q$ onde $\mathbf{X}_i$ é (aproximadamente) independente de todo $\mathbf{X}_j$ para $i \neq j$.
if dá para dividir **then**
    return $\prod_{i=1}^{q}$ LearnSPN($\mathbf{X}_i$, $\mathbf{I}$)
else
    Divida as instâncias $\mathbf{I}$ em partições $\mathbf{I}_1, \ldots, \mathbf{I}_k$ tal que $\mathbf{I}_i$ seja uma coleção de instâncias mais similares possíveis entre si.
    return $\sum_{i=1}^{k} \frac{|\mathbf{I}_i|}{|\mathbf{I}|}$ LearnSPN($\mathbf{X}$, $\mathbf{I}_i$)
end if

Referências

[Ben09] Yoshua Bengio. "Learning Deep Architectures for AI". Em: *Foundations and Trends in Machine Learning* (2009).

[GD13] Robert Gens e Pedro Domingos. "Learning the Structure of Sum-Product Networks". Em: *International Conference on Machine Learning* 30 (2013).