APPROXIMATION COMPLEXITY OF MAP INFERENCE IN SUM-PRODUCT NETWORKS

Diarmaid Conaty
Queen’s University Belfast, UK

Denis D. Mauá
Universidade de São Paulo, Brazil

Cassio P. de Campos
Queen’s University Belfast, UK

UAI 2017

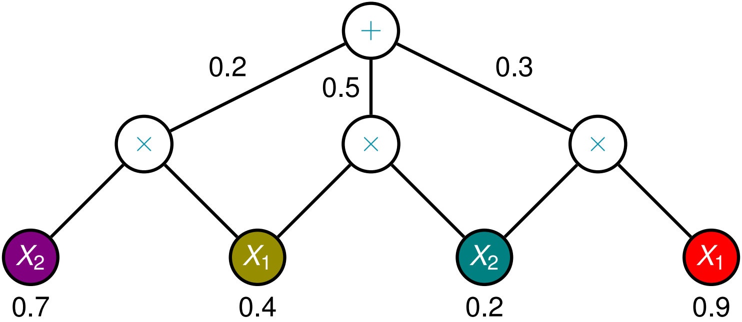
Sum-Product Networks

Efficient representation of the arithmetic expression describing probability function

Sum-Product Network

- Rooted DAG with internal nodes $+$ and $\times$, and univariate distributions on leaves
  - Product nodes have disjoint scope
  - Sum nodes have identical scope
- Represents rich mixture probability distribution (with exponentially many components)
- Allows efficient marginal inference; usually learned from data

Sum-Product Network



$$S(X_1, X_2) = 0.2P_1(X_1)P_2(X_2) + 0.5P_1(X_1)P_2(X_2) + 0.3P_1(X_1)P_2(X_2)$$

Maximum a Posteriori Inference

MAP

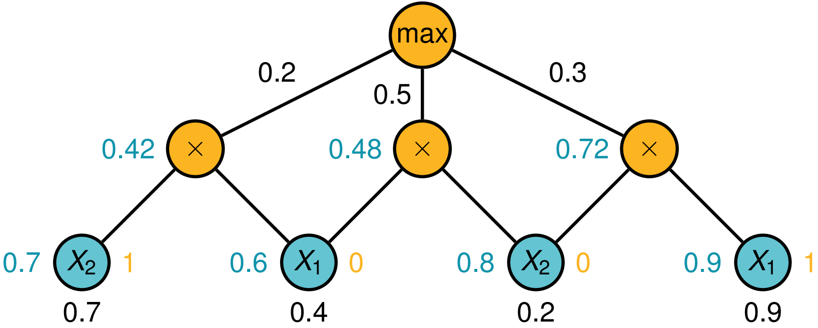
Given a sum-product net S specified with rational weights and an assignment e, find x^* such that $S(x^*) = \max_{x \sim e} S(x)$

Complexity of Maximum a Posteriori Inference

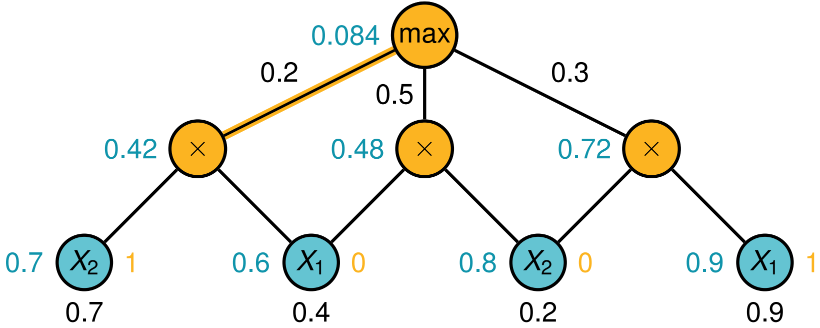
Previously known:

- Peharz (2015) proved NP-hardness from MAXSAT
- Peharz et al. (2016) adapted NP-hardness of MAP in Naive-Bayes BN (de Campos, 2011)
- Is tractable when supports of children of sum node are disjoint (Peharz et al. 2016)
- In practice: Max-Product Algorithm (Poon & Domingos 2011)

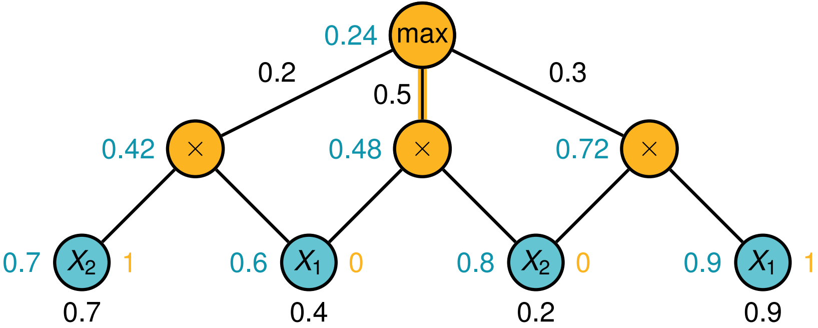
Max-Product Algorithm



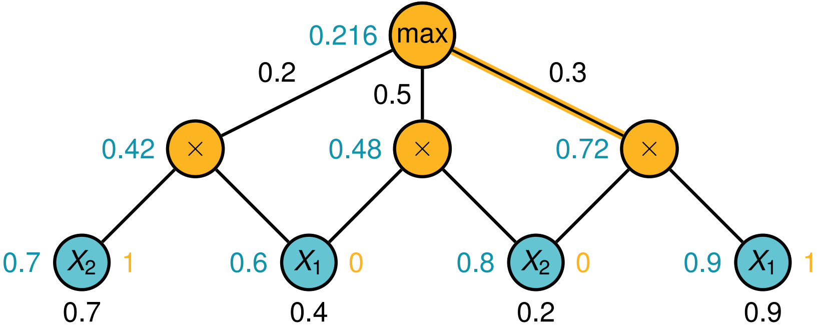
Max-Product Algorithm



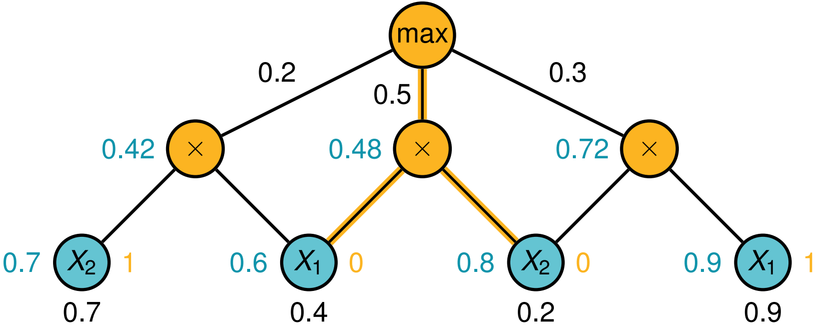
Max-Product Algorithm



Max-Product Algorithm

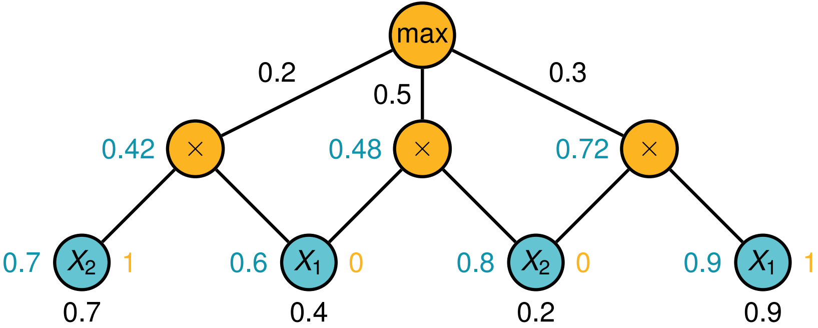


Max-Product Algorithm



$\text{maxprod}(S) = (0, 0)$ $\quad$ $S(X_1 = 0, X_2 = 0) = 0.3$

Max-Product Algorithm



Linear time

Complexity of Approximate MAP

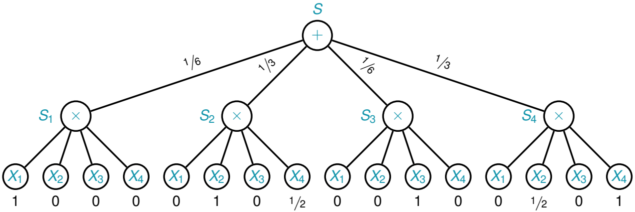
This work:

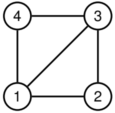
- New proof of NP-hardness
- Establish approximation complexity of MAP
- Worst-case bounds for Max-Product quality
- Improved approximate algorithm: ArgMax-Product
- Empirical results comparing both algorithms

Theorem

MAP in sum-product networks is NP-complete even if there is no evidence, and the underlying graph is a tree of height 2

Proof: Reduction from maximum independent set:





Corollary

Unless P = NP, there is no $(m-1)^{\varepsilon}$*-approximation algorithm for MAP in sum-product networks for any* $0 \leqslant \varepsilon < 1$*, where* m *is the number of internal nodes of the networks, even if there is no evidence and the underlying graph is a tree of height 2*

COROLLARY

Unless P = NP, there is no $(m-1)^{\varepsilon}$*-approximation algorithm for MAP in sum-product networks for any* $0 \leqslant \varepsilon < 1$*, where* m *is the number of internal nodes of the networks, even if there is no evidence and the underlying graph is a tree of height 2*

THEOREM

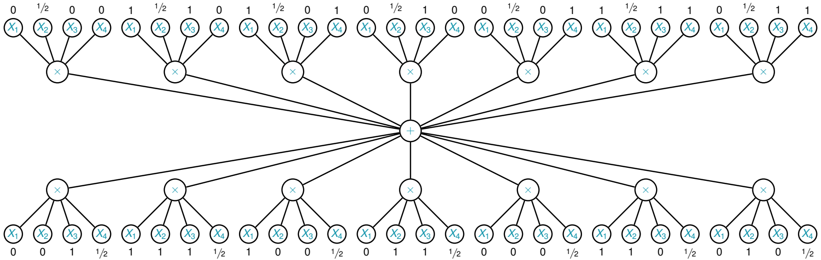
Max-Product returns a $(m-1)$*-approximation for sum-product networks whose underlying graph has height at most 2, where* m *is the number of internal nodes.*

Theorem

Unless P=NP, there is no $2^{s^{\varepsilon}}$-approximation algorithm for MAP in sum-product-networks for any $0 \leqslant \varepsilon < 1$, where s is the size of the input, even if there is no evidence and the underlying graph is a tree of height 3

Proof: Reduction from SAT using a polynomial number of "independent" copies of SAT-solving network.

E.g. $(\neg X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_3 \vee X_4)$



Theorem

Let S^+ denote the sum nodes in sum-product network S, and d_i be the number of children of sum node $S_i \in S^+$. Then Max-Product finds an $(\prod_{S_i \in S^+} d_i)$-approximation

Corollary

Max-Product returns a $2^{\varepsilon \cdot s}$-approximation for some $0 < \varepsilon < 1$, where s is the size of the network

So Far...

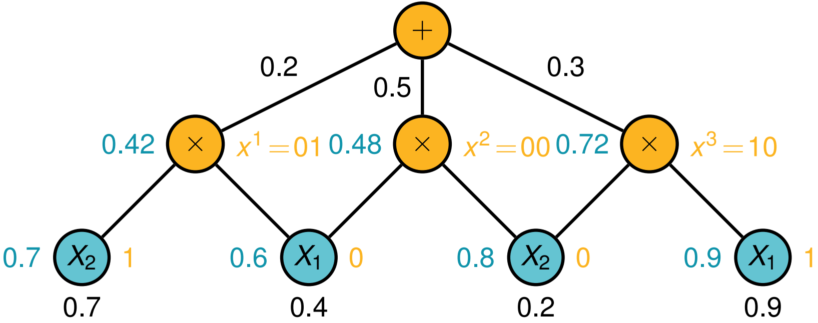
- Max-Product is optimal (in the worst-case)
- Can we find an algorithm with same worst-case performance but better average-case performance?

So Far...

- Max-Product is optimal (in the worst-case)
- Can we find an algorithm with same worst-case performance but better average-case performance?
- Yes, if we admit quadratic runtime

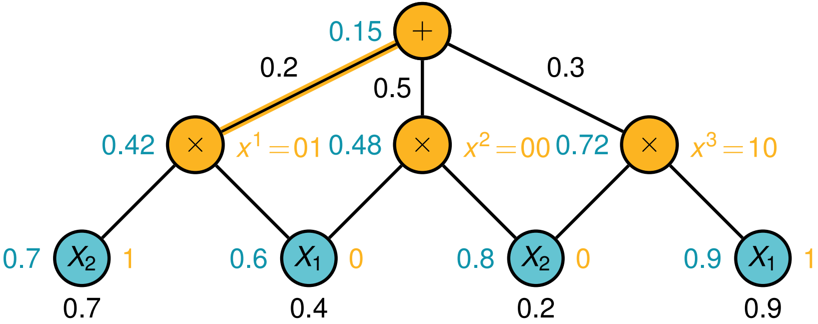
Argmax-Product Algorithm: Single upward pass propagating (partial) configurations and selecting best configuration at sum nodes

ARGMAX-PRODUCT ALGORITHM



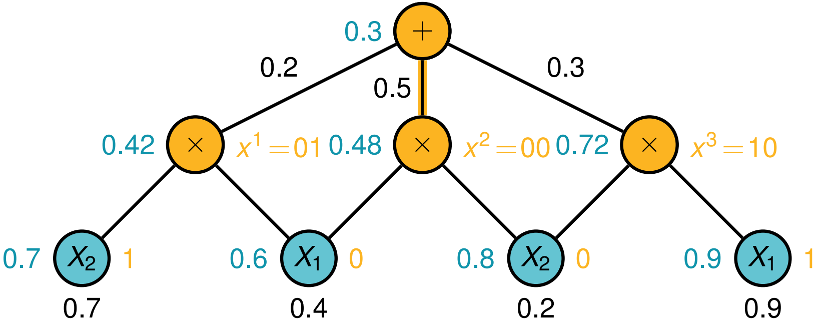
$$\text{amap}(S) = \underset{x \in \{x^1, \dots, x^t\}}{\arg\max} \sum_{j=1}^{t} w_j S_j(x)$$

Argmax-Product Algorithm



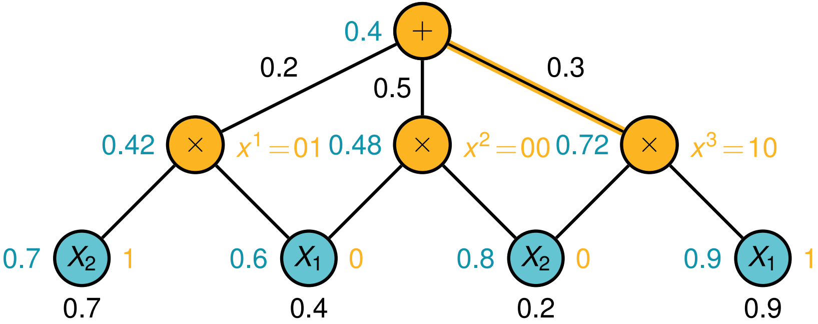
$$\operatorname{amap}(S) = \underset{x \in \{x^1, \dots, x^t\}}{\arg\max} \sum_{j=1}^{t} w_j S_j(x)$$

Argmax-Product Algorithm



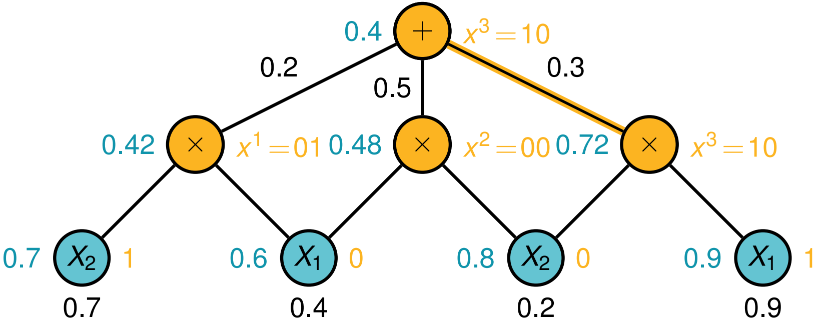
$$\text{amap}(S) = \underset{x \in \{x^1, \ldots, x^t\}}{\arg\max} \sum_{j=1}^{t} w_j S_j(x)$$

Argmax-Product Algorithm



$$\mathrm{amap}(S) = \underset{x \in \{x^1, \ldots, x^t\}}{\arg\max} \sum_{j=1}^{t} w_j S_j(x)$$

ARGMAX-PRODUCT ALGORITHM



$$\mathrm{amap}(S) = \underset{x \in \{x^1, \ldots, x^t\}}{\arg\max} \sum_{j=1}^{t} w_j S_j(x)$$

Theorem

- *Argmax-Product always finds a configuration that is at least as good as the configuration found by MaxProduct:*

$$S(\mathrm{amap}(S, e)) \geqslant S(\mathrm{maxprod}(S, e))$$

- *There is a network such that Argmax-Product returns an exponentially better configuration than Max-Product:*

$$S(\mathrm{amap}(S, e)) > 2^m S(\mathrm{maxprod}(S, e))\,,$$

where m is the number of sum nodes in S

Experiments: Argmax-Product Vs. Max-Product

Vertices	% Edges	Nodes	Ratio	StDev
10	10	111	1.89	0.95
10	20	111	2.16	1.09
10	40	111	2.12	1.01
10	60	111	2.04	0.89
20	10	421	1.94	0.88
20	20	421	2.89	1.72
20	40	421	3.02	1.24
20	60	421	2.60	1.04
40	10	1641	2.64	1.42
40	20	1641	3.37	1.45
40	40	1641	2.33	0.73
40	60	1641	2.27	0.76
80	10	6481	3.96	1.81
80	20	6481	2.10	0.49
80	40	6481	1.07	0.26
80	60	6481	1.04	0.20

Table: SPNs encoding randomly generated instances of the maximum independent set problem

Experiments: Argmax-Product Vs. Max-Product

Dataset	No. of variables	No. of samples	Height	No Evidence	50% Evidence	
				Ratio	Ratio	StDev
audiology	70	204	12	1.0000	1.0029	0.0133
breast-cancer	10	258	24	1.1572	1.1977	0.1923
car	7	1556	3	1.1028	1.0514	0.0514
cylinder-bands	33	487	62	1.1154	1.1185	0.0220
flags	29	175	26	1.3568	1.3654	0.0363
ionosphere	34	316	42	1.1176	1.1109	0.0273
nursery	9	11665	3	1.6225	1.2060	0.2926
primary-tumor	18	306	114	1.0882	1.0828	0.0210
sonar	61	188	26	1.2380	1.2314	0.0261
vowel	14	892	3	1.0751	1.0666	0.0229

Table: SPNs learned using LearnSPN from UCI datasets

Conclusion

Height	Lower bound	Upper bound
1	1	1
2	$(m-1)^{\varepsilon}$	$m-1$
$\geqslant 3$	$2^{s^{\varepsilon}}$	2^{s}

- MAP is hard even to approximate in SPNs
- Max-Product performance is optimal in worst-case
- Simple modification to algorithm can significantly improve quality (with a decrease in runtime performance)
- Goal: quality of Argmax-Product with runtime of Max-Product