# Shallow versus Deep Sum-Product Networks

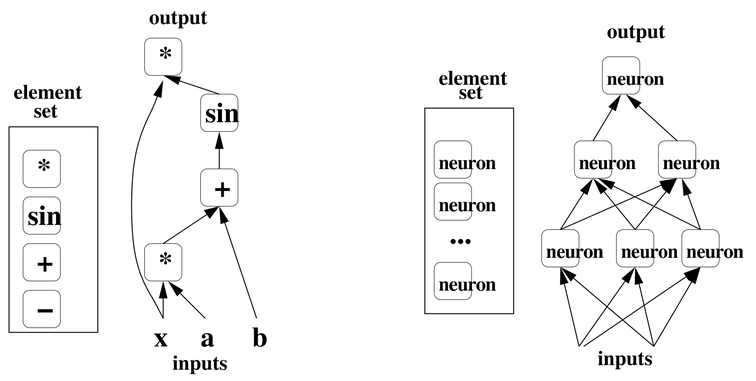
Olivier Delalleau
Yoshua Bengio
*Université de Montréal*

## Motivations

- Deep learning algorithms are based on multiple levels of representation, corresponding to a deep circuit.
- It has been suggested that deep architectures are more powerful in the sense of being able to more efficiently represent highly-varying functions [2, 1].
- Very few theoretical results up to now to confirm this idea, mostly in the case of networks of linear threshold units with positivity constraints on the weights [3].
- [1] suggests that polynomials represented by deep sum-product networks would be more efficient, but no proof.
- This work aims at showing families of circuits for which a deep architecture can be exponentially more efficient than a shallow one, in the context of polynomials (over real values).

## Depth

- Computation can be organized in a directed acyclic graph where each node computes a function of the outputs of its predecessors in the graph.
- Depth is the length of the longest path from input to output.
- A family of circuits is defined by allowing a different set of computational elements (what computations are allowed at each node), as well as a different connectivity scheme.

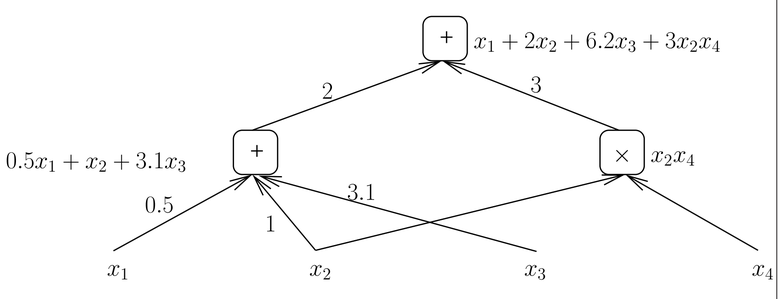



## Deep Architectures

Motivations for deep architectures:

- Learn high-level representations.
- Take advantage of unsupervised learning and unlabeled examples.
- Biological inspiration: the brain is a deep architecture.
- Cognitive inspiration: humans learn concepts hierarchically and compose old concepts to define new ones.
- Computational and statistical efficiency, if highly-varying functions can be more efficiently represented with deep architectures.

## Sum-Product Networks

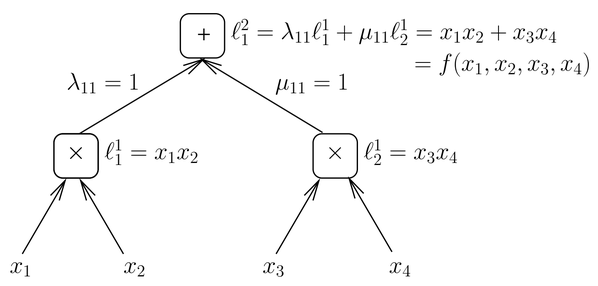



- Neural networks made of units that either compute a product of their inputs, or a sum of their inputs (weighted by positive weights).
- [4] proposed such deep sum-product networks to compute tractable graphical models.

## Methodology

- **Goal:** Find families of functions computed by sum-product networks that can be represented exponentially more compactly by a deep network compared to a shallow one (= with only one hidden layer).
- **Approach:** Design deep sum-product networks that compute complicated polynomials, then find a lower bound on the number of hidden units required by a shallow network to compute the same polynomial.

## Network Family $\mathcal{F}$




- $2i$ layers and $n = 4^i$ input variables
- alternate additive and multiplicative binary units
- unit $\ell_j^k$ takes as inputs $\ell_{2j-1}^{k-1}$ and $\ell_{2j}^{k-1}$

**Lemma 1** *Any element $\ell_j^k$ can be written as a (positively) weighted sum of products of input variables, such that each input variable $x_t$ is used in exactly one unit of $\ell^k$. Moreover, the number $m_k$ of products found in the sum computed by $\ell_j^k$ does not depend on $j$ and obeys the following recurrence rule for $k \geq 0$:*

- *if $k+1$ is odd, then $m_{k+1} = m_k^2$,*
- *otherwise $m_{k+1} = 2m_k$.*

**Proposition 1** *The number of products in the sum computed in the output unit $l_1^{2i}$ is $m_{2i} = 2^{\sqrt{n}-1}$.*

**Lemma 2** *The products computed in the output unit $l_1^{2i}$ can be split in two groups, one with products containing only variables $x_1, \ldots, x_{\frac{n}{2}}$ and one containing only variables $x_{\frac{n}{2}+1}, \ldots, x_n$.*

**Lemma 3** *The products computed in the output unit $l_1^{2i}$ involve more than one input variable.*

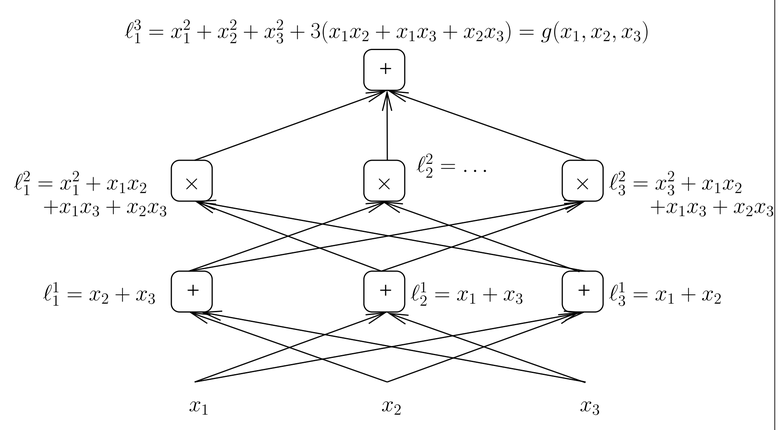
**Lemma 4** *Any shallow sum-product network computing $f \in \mathcal{F}$ must have a "sum" unit as output.*

**Lemma 5** *Any shallow sum-product network computing $f \in \mathcal{F}$ must have only multiplicative units in its hidden layer.*

**Lemma 6** *Any shallow negative-weight sum-product network computing $f \in \mathcal{F}$ must have at least $2^{\sqrt{n}-1}$ hidden units, if its output unit is a sum and its hidden units are products.*

**Corollary 1** *Any shallow sum-product network computing $f \in \mathcal{F}$ must have at least $2^{\sqrt{n}-1}$ hidden units.*

## Network Family $\mathcal{G}$



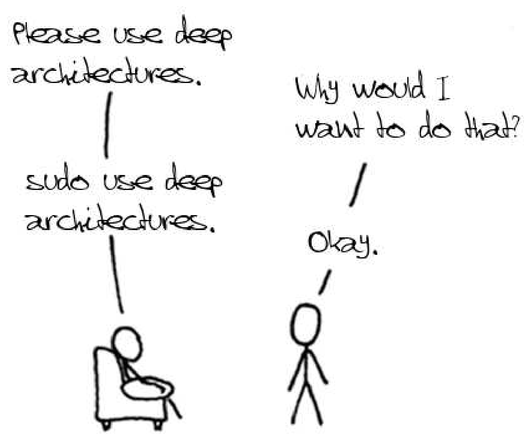

- $2i+1$ layers and $n$ variables ($n$ independent of $i$)
- alternate multiplicative and additive units
- unit $\ell_j^k$ takes as inputs $\{\ell_m^{k-1} | m \neq j\}$

**Proposition 2** *For any $0 \leq j \leq i$, and any product of variables $P = \Pi_{t=1}^n x_t^{\alpha_t}$ such that $\alpha_t \in \mathbb{N}$ and $\sum_t \alpha_t = (n-1)^j$, there exists a unit in $\ell^{2j}$ whose computed value, when expanded as a weighted sum of products, contains $P$ among these products.*

**Corollary 2** *The output $g_{in}$ of a sum-product network in $\mathcal{G}_{in}$, when expanded as a sum of products, contains all products of variables of the form $\Pi_{t=1}^n x_t^{\alpha_t}$ such that $\alpha_t \in \mathbb{N}$ and $\sum_t \alpha_t = (n-1)^i$.*

**Proposition 3** *A shallow negative-weight sum-product network computing $g_{in} \in \mathcal{G}_{in}$ must have at least $(n-1)^i$ hidden units.*

## A Deep Argument




*Inspired from* `http://xkcd.com`

## Conclusions

- New examples of probably superior efficiency of deep architectures.
- **Some deep sum-product networks with $n$ inputs and depth $\log_4 n$ can represent with $O(n)$ units what would require $O(2^{\sqrt{n}})$ units for a depth-2 network.**
- **Some deep sum-product networks with $n$ inputs and depth $k$ can represent with $O(nk)$ units what would require $O((n-1)^k)$ units for a depth-2 network.**

## References

[1] Bengio, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning 2*, 1 (2009), 1–127. Also published as a book. Now Publishers, 2009.

[2] Bengio, Y., and LeCun, Y. Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*, L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Eds. MIT Press, 2007.

[3] Håstad, J., and Goldmann, M. On the power of small-depth threshold circuits. *Computational Complexity 1* (1991), 113–129.

[4] Poon, H., and Domingos, P. Sum-product networks: A new deep architecture. In *Proceedings of the Twenty-seventh Conference in Uncertainty in Artificial Intelligence (UAI)* (Barcelona, Spain, 2011).