Why and When Can Deep - but Not Shallow Networks Avoid the Curse of Dimensionality

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, Qianli Liao

Presented by: William Fedus, Christos Tsirigotis, Breandan Considine

What’s the Idea of the Paper?

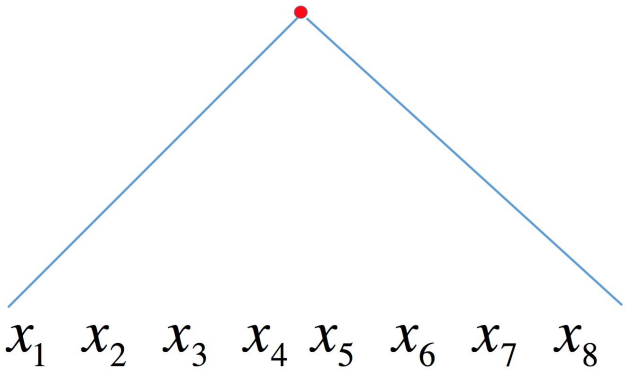
TL;DR

Deep nets, but not *shallow* nets, can efficiently approximate functions of functions

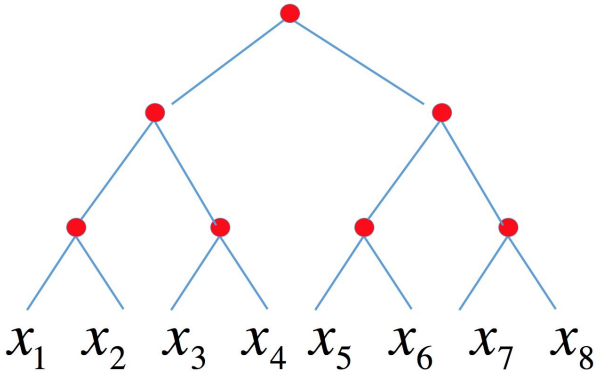
Functions of Functions

Functions of functions, or *compositional functions,* are a frequently occuring special class of functions we often care about in ML (e.g. natural language, object recognition, hierarchical RL, etc.)

$$f(x_1, \cdots, x_8)$$

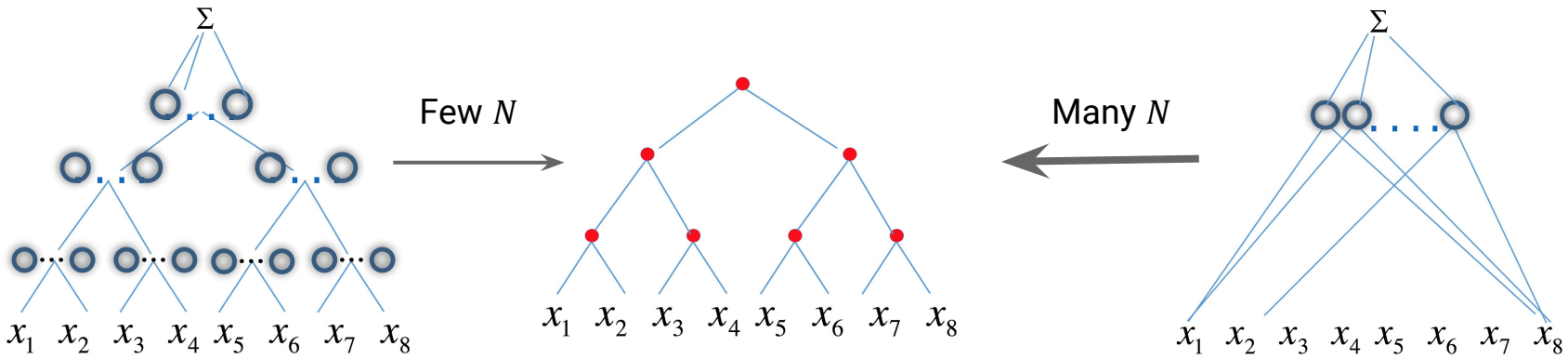


$$f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$$



Deep Nets Efficiently Model this Hierarchy

Deep nets are able to much more *efficiently* approximate compositional functions using fewer units N than shallow networks can



General Outline

General Outline

1. Theory: When Deep is “Better” than Shallow
2. Theorem 2 Proof
3. Example + Further Commentary

Theory: When Deep is “Better” than Shallow

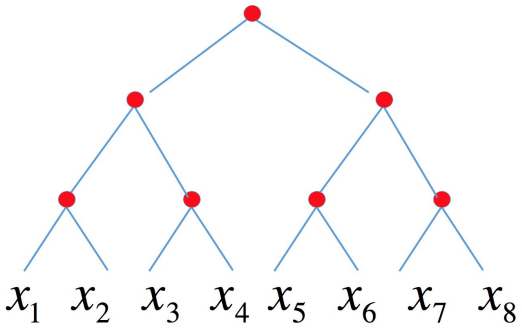
When Deep is “Better” than Shallow?

- Both shallow and deep networks are *universal**
- However, for hierarchical compositions of *local* functions, deep nets achieve the same accuracy with *exponentially fewer* parameters!
  - In other words, deep nets avoid the “*curse of dimensionality*”. *Termed by Bellman in 1961.*

universal = approximate arbitrarily well any continuous function of n variables on compact domain*

Theorem 2 Preliminaries (1/2)

In the theorem, we consider a hierarchical binary tree function f with n variables

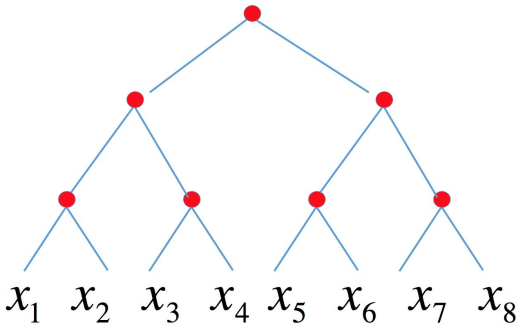


Parameters

- N : number of units in neural net
- ϵ : required accuracy
- n : number of variables for the function f
- $m \geq 1$: integer smoothness parameter

Theorem 2 Preliminaries (2/2)

In the theorem, we consider a hierarchical binary tree function f with n variables



Additional Assumption

W_m^n is the *set of functions* of n variables with continuous partial derivatives of orders up to $m < \infty$ such that

$$||f|| + \sum_{1 \leq |k|_1 \leq m} ||D^k f|| \leq 1, \tag{1}$$

where D^k denotes the partial derivative indicated by the multi integer $k \geq 1$, and $|k|_1$ is the sum of the components of k.

Deep Nets Require *Exponentially* Fewer Variables

For this function, the number of units N needed to achieve accuracy ϵ is given by

***Shallow* networks (Theorem 1):**

$$N = \mathcal{O}(\epsilon^{-n/m}) \text{ and is the best possible}$$

***Deep* networks (Theorem 2):**

$$N = \mathcal{O}((n-1)\epsilon^{-2/m})$$

Theorem 2 Proof

Theorem 2 Proof Outline

- Constituent functions $\in W_m^2$, can be approximated by shallow nets

$$N = \mathcal{O}(\epsilon^{-n/m}) \quad \Rightarrow \quad \varepsilon = cN^{-m/2}$$

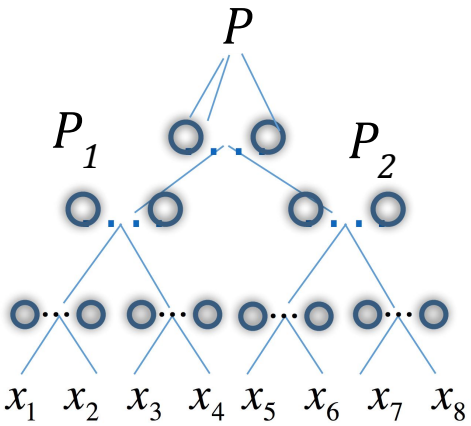
From Theorem 1

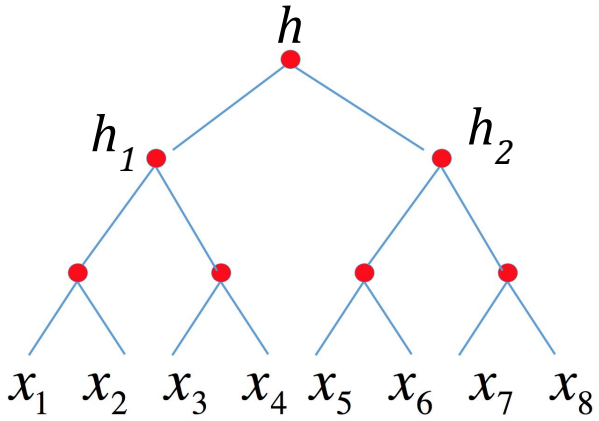
- W_m^n is a compact set in Sobolev space $\Rightarrow W_m^n$ is Lipschitz $\Rightarrow W_m^{n,2}$ is Lipschitz

m ≥ 1
using ∞-norm

$W_m^n \supseteq W_m^{n,2}$

Theorem 2 Proof Outline





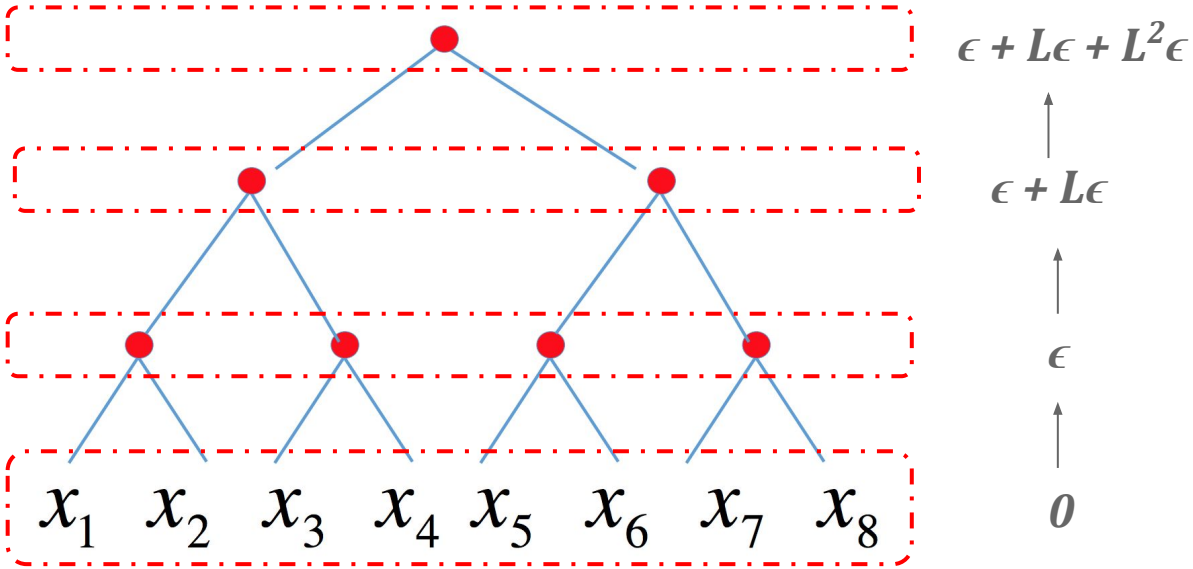
$$\|P(P_1,P_2)-h(h_1,h_2)\| \le \|P(P_1,P_2)-h(P_1,P_2)\|+\|h(P_1,P_2)-h(h_1,h_2)\|$$

Use "base" shallow error: $\|P-h\|_\infty < \epsilon$

Use Lipschitz property

Theorem 2 Proof Outline

$$\|P_{cur.node}(P_1, P_2) - h_{cur.node}(h_1, h_2)\| \leq \epsilon_{shallow} + Le_{prev.node}$$



Theorem 2 Proof Outline

Expanding recursively for the binary tree of n inputs ⇨ depth $log_2 n$

$$\epsilon_{total} = (1 + L + L^2 + \ldots + L^{log_2 n})\epsilon$$

$$\epsilon_{total} = \frac{1 - L^{1+log_2 n}}{1 - L}(cN^{-m/2})$$

Remember for shallow:

$$\varepsilon = cN^{-m/2}$$

Theorem 2 Proof Outline

Expanding recursively for the binary tree of n inputs ⇨ depth $\log_2 n$

$$\epsilon_{total} = \frac{1-L^{1+log_2 n}}{1-L}\left(cN^{-m/2}\right)$$

Deep number of neurons/units:

$$N_{total} = (n-1)N$$

$$\epsilon_{total} = c\frac{1-L^{1+log_2 n}}{1-L}\left(\frac{N_{total}}{n-1}\right)^{-m/2}$$

Theorem 2 Proof Outline

Expanding recursively for the binary tree of $\boldsymbol{n}$ inputs

$$\epsilon_{total} = c\frac{1-L^{1+log_2 n}}{1-L}\left(\frac{N_{total}}{n-1}\right)^{-m/2}$$

$$N_{total} = (n-1)\epsilon_{total}^{-2/m}\left(\frac{1-L}{c(1-L^{1+log_2 n})}\right)^{-2/m} = (n-1)\epsilon_{total}^{-2/m}\Theta(n^{2\times(log_2 L)/m})$$

Conveniently our assumptions imply that:
L ≤ 1 (compact Sobolev)

Examples and Commentary

Building Intuition with an example

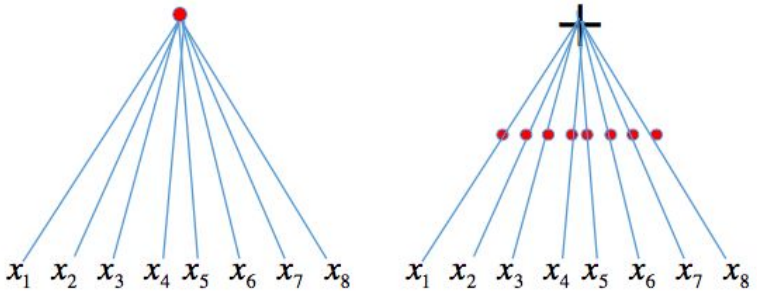
Consider function Q, a polynomial with coordinatewise degree of 2^{11}

$$
\begin{aligned}
Q(x,y) \;=\; & (Ax^2y^2 + Bx^2y \\
& + Cxy^2 + Dx^2 + 2Exy \\
& + Fy^2 + 2Gx + 2Hy + I)^{2^{10}}
\end{aligned}
$$

A *deep* network may approximate the polynomial with 39 units.

Building Intuition with an example

$$Q(x,y) = (Ax^2y^2 + Bx^2y + Cy^2x + Dx^2 + 2Exy + Fy^2 + 2Gx + 2Hy + I)^{2^{10}}$$
$$= (Ax^2y^2 + Bx^2y + Cy^2x + Dx^2 + 2Exy + Fy^2 + 2Gx + 2Hy + I)^{1024}$$
$$= A^{1024}x^{2048}y^{2048} + \ldots + I^{1024}$$

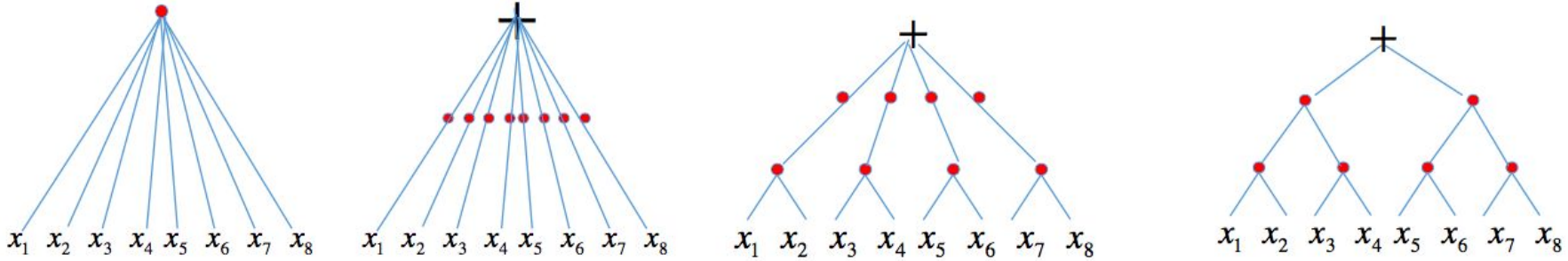


Building Intuition with an example

$$Q(x,y) = (Ax^2y^2 + Bx^2y + Cy^2x + Dx^2 + 2Exy + Fy^2 + 2Gx + 2Hy + I)^{2^{10}}$$
$$= (Ax^2y^2 + Bx^2y + Cy^2x + Dx^2 + 2Exy + Fy^2 + 2Gx + 2Hy + I)^{1024}$$
$$= A^{1024}x^{2048}y^{2048} + \ldots + I^{1024}$$
$$or\ldots$$
$$Z(x,y) = Ax^2y^2 + Bx^2y + Cy^2x + Dx^2 + 2Exy + Fy^2 + 2Gx + 2Hy + I$$
$$Q(x,y) = ((((((((((Z^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2$$

Building Intuition with an example

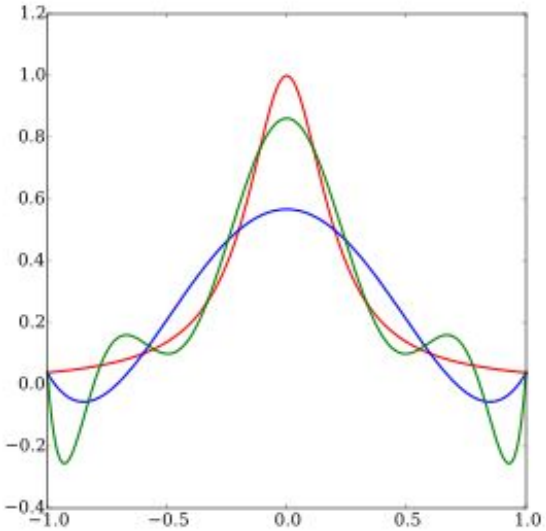
$$Q(x,y) = (Ax^2y^2 + Bx^2y + Cxy^2 + Dx^2 + 2Exy + Fy^2 + 2Gx + 2Hy + I)^{2^{10}}$$

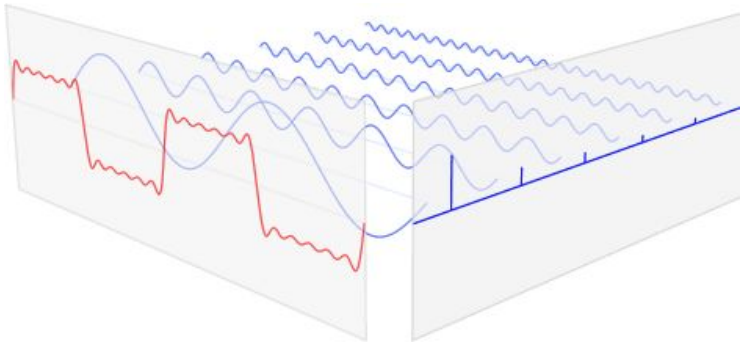


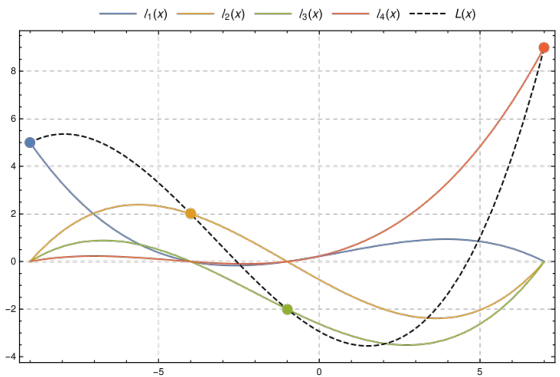
Building Intuition with an example

We can construct an approximation to *any polynomial* using a fixed number of weights and biases. This number grows exponentially faster when the network is shallow than when the network is deep.

Function composition is common in mathematics







Why are Compositional Functions so common?

- Physics
  - The physical world contains many patterns of self-similar structures
  - Sequence of increasing scales that are local at each scale
  - Iterated local functions can be Turing universal
- Neuroscience
  - The way we interpret the world is naturally hierarchical
  - The questions we pose are naturally hierarchical
  - Our neurobiology is a one evolutionary success story

Conclusions

TL;DR

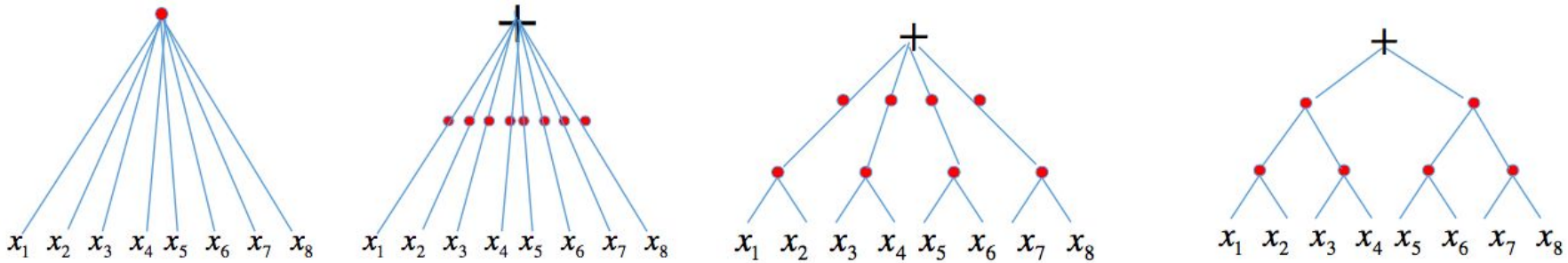
Deep nets efficiently approximate compositional functions through a hierarchy of local computations

MILA

Appendix

Function composition is common in mathematics

- Composition is a common trick when approximating functions
- Deep neural networks are a series nested function compositions
- Poggio et al. are primarily interested in multiplicative composition



The power of deeper networks for expressing natural functions

David Rolnick*, Max Tegmark†
Dept. of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139 and
Dept. of Mathematics, Massachusetts Institute of Technology, Cambridge, MA 02139

(Dated: May 17, 2017)

It is well-known that neural networks are universal approximators, but that deeper networks tend to be much more efficient than shallow ones. We shed light on this by proving that the total number of neurons m required to approximate natural classes of multivariate polynomials of n variables grows only linearly with n for deep neural networks, but grows exponentially when merely a single hidden layer is allowed. We also provide evidence that when the number of hidden layers is increased from 1 to k, the neuron requirement grows exponentially not with n but with $n^{1/k}$, suggesting that the minimum number of layers required for computational tractability grows only logarithmically with n.

I. INTRODUCTION

Deep learning has lately been shown to be a very powerful tool for a wide range of problems, from image segmentation to machine translation. Despite its success, many of the techniques developed by practitioners of artificial neural networks (ANNs) are heuristics without theoretical guarantees. Perhaps most notably, the power of feed-network of types other than the standard feedforward model. The problem has also been posed for sum-product networks [11] and restricted Boltzmann machines [12]. Cohen, Sharir, and Shashua [13] showed, using tools from tensor decomposition, that shallow arithmetic circuits can express only a measure-zero set of the functions expressible by deep circuits. A weak generalization of this result to convolutional neural networks was shown in [14].

TL;DR

- Both shallow and deep neural networks can approximate any polynomial, but deep networks can approximate much more efficiently given a fixed number of units.
- The approximating deep network does not need to exactly match the architecture of the compositional function as long as the graph or tree associated with the function is contained in the graph associated with the network.

[Old] Outline

Deep networks avoid the *curse of dimensionality* for *compositional functions*

1. Review function approximation
    - Shallow nets
    - Deep nets
    - Different activations
2. Curse of dimensionality
3. Compositional functions
    - What is it?
    - Why is compositionality so common?
    - Hierarchically local compositional functions
4. An illustrative example
5. Conclusion
    - Revisit highlight of points

Building Intuition with an example

H. N. Mhaskar, “Neural networks for optimal approximation of smooth and analytic functions,” Neural Computation, vol. 8, no. 1, pp. 164–177, 1996.

LEMMA 3.2. *Let ϕ satisfy the conditions of Theorem 2.1, $m \geq 1$ be an integer and $\mathbf{k} \geq 0$ be any multi-integer in $\mathbf{Z}^s$ with $\max_{1 \leq j \leq s} |k_j| \leq m$. Then for every $\epsilon > 0$, there exists $G_{\mathbf{k},m,\epsilon} \in \Pi_{\phi;(6m+1)^s,s}$ such that*

$$||T_{\mathbf{k}} - G_{\mathbf{k},m,\epsilon}||_\infty \leq \epsilon. \tag{3.16}$$

The weights and thresholds of each $G_{\mathbf{k},m,\epsilon}$ may be chosen from a fixed set with cardinality not exceeding $(6m+1)^s$.

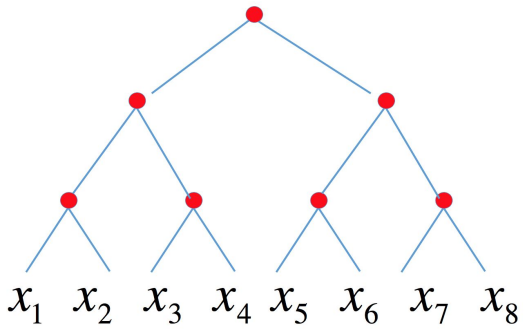
Generalizations

- TODO(christos): Discuss the results of Theorem 3 (general degree of fan-in d_v to node v), offered without proof.
- TODO(christos): Discuss Theorem 4 (ReLU), offered without proof.

Theorem 2 Preliminaries (1/2)

In the theorem, we consider a hierarchical binary tree function f with n variables

$$f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)))$$



Let $I^n = [-1, 1]^n$, $\mathbb{X} = C(I^n)$ be the space of all continuous functions on I^n, with $\|f\| = \max_{x \in I^n} |f(x)|$. Let $\mathcal{S}_{N,n}$ denote the class of all shallow networks with N units of the form

$$x \mapsto \sum_{k=1}^{N} a_k \sigma(\langle w_k, x \rangle + b_k),$$

Theorem 2 Preliminaries (2/2)

Parameters

- N : number of units in neural net
- ϵ : required accuracy
- n : number of variables for the function f
- $m \geq 1$: integer smoothness parameter
- W_m^n is the *set of functions* of n variables with continuous partial derivatives of orders up to $m < \infty$ such that

$$||f|| + \sum_{1 \leq |k| \leq m} ||D^k f|| \leq 1, \tag{1}$$

where D^k denotes the partial derivative indicated by the multi integer $k \geq 1$, and $|k|_1$ is the sum of the components of k.

Curse of Dimensionality

-”Curse of dimensionality” coined by Bellman in 1961

-Many algorithms do not work well in high dimensions

-In high-D, most of the mass of a multivariate Gaussian distribution is not near the mean, but in a “shell” of increasing distance from the mean

-Naive measures of function approximation tend to break down in high dimensions

-”Blessing of non-uniformity”: most real-world applications can be modeled with a low dimensional manifold

Observations

1. Although compositional functions are just a subset of functions of n variables, these look the same to a shallow network

$$W_m^n \supseteq W_m^{n,2}$$

2. The deep network only needs to contain the acyclic graph representing the function as a computation *subgraph,* it doesn't need to match it exactly

Prior Version of Proof

Theorem 2 Proof Outline (2/2)

- Bounded node-level **error** and **Lipschitz** property:

$$\|P(P_1, P_2) - h(h_1, h_2)\| \leq \epsilon_{this} + L\epsilon_{prev}$$

- Expanding recursively for the binary tree of ***n*** inputs:

$$\epsilon_{total} = (1 + L + L^2 + \ldots + L^{log_2 n})\epsilon$$

$$\epsilon_{total} = c\frac{1-L^{\log_2 n}}{1-L}\left(\frac{N_{total}}{n-1}\right)^{-m/2}$$

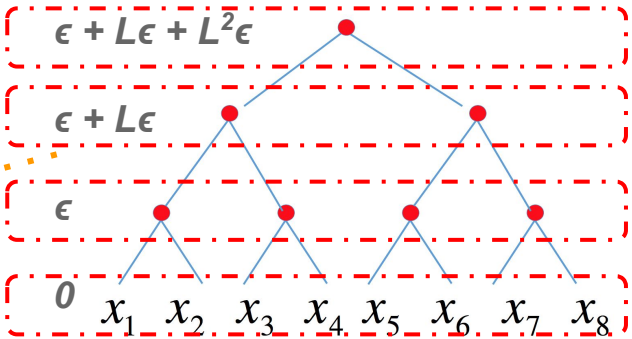
$$N_{total} = (n-1)\epsilon_{total}^{-2/m}\left(\frac{1-L}{c(1-L^{log_2 n})}\right)^{-2/m} = (n-1)\epsilon_{total}^{-2/m}\Theta(n^{2\times(log_2 L)/m})$$

□

Theorem 2 Proof Outline (2/2)

- Bounded node-level **error** and **Lipschitz** property:

$$\|P(P_1, P_2) - h(h_1, h_2)\| \leq \epsilon_{this} + L\epsilon_{prev}$$



- Expanding recursively for the binary tree of ***n*** inputs:

$$\epsilon_{total} = (1 + L + L^2 + \ldots + L^{log_2 n})\epsilon$$

$$\epsilon_{total} = c\frac{1 - L^{\log_2 n}}{1 - L}\left(\frac{N_{total}}{n-1}\right)^{-m/2}$$

$$N_{total} = (n-1)\epsilon_{total}^{-2/m}\left(\frac{1-L}{c(1-L^{log_2 n})}\right)^{-2/m} = (n-1)\epsilon_{total}^{-2/m}\Theta(n^{2\times(log_2 L)/m})$$

□

Theorem 2 Proof Outline (2/2)

- Bounded node-level **error** and **Lipschitz** property:

$$\|P(P_1, P_2) - h(h_1, h_2)\| \leq \epsilon_{this} + L\epsilon_{prev}$$

- Expanding recursively for the binary tree of ***n*** inputs:

$$\epsilon_{total} = (1 + L + L^2 + \ldots + L^{log_2 n})\epsilon$$

$$\epsilon_{total} = c\frac{1-L^{\log_2 n}}{1-L}\left(\frac{N_{total}}{n-1}\right)^{-m/2}$$

$$N_{total} = (n-1)\epsilon_{total}^{-2/m}\left(\frac{1-L}{c(1-L^{log_2 n})}\right)^{-2/m} = (n-1)\epsilon_{total}^{-2/m}\Theta(n^{2\times(log_2 L)/m})$$

□

Remember for shallow:

$$\varepsilon = cN^{-m/2}$$

Deep number of neurons/units:

$$N_{total} = (n-1)N$$

Theorem 2 Proof Outline (2/2)

- Bounded node-level **error** and **Lipschitz** property:

$$\|P(P_1, P_2) - h(h_1, h_2)\| \leq \epsilon_{this} + L\epsilon_{prev}$$

- Expanding recursively for the binary tree of ***n*** inputs:

$$\epsilon_{total} = (1 + L + L^2 + \ldots + L^{log_2 n})\epsilon$$

$$\epsilon_{total} = c\frac{1-L^{\log_2 n}}{1-L}\left(\frac{N_{total}}{n-1}\right)^{-m/2}$$

$$N_{total} = (n-1)\epsilon_{total}^{-2/m}\left(\frac{1-L}{c(1-L^{log_2 n})}\right)^{-2/m} = (n-1)\epsilon_{total}^{-2/m}\Theta(n^{2\times(log_2 L)/m})$$

□

Conveniently our assumptions imply that:
L ≤ 1 (compact Sobolev)

If the kernel is **local**, i.e.

$$\lim_{||x-x_i||\to\infty} K(x, x_i) \to c_i$$

then when x gets farther from the training set

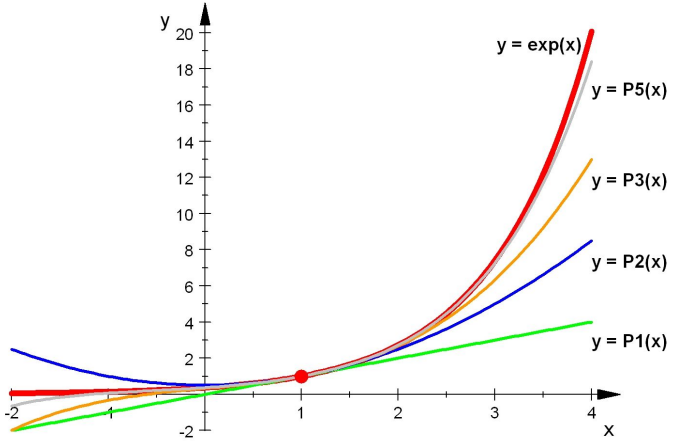
$$f(x) \to b + \sum_i \alpha_i c_i$$

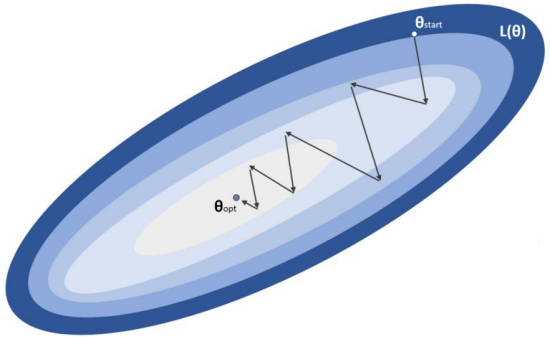
After becoming approx. linear, *the predictor becomes either constant or (approximately) the nearest neighbor predictor (e.g. with the Gaussian kernel)*

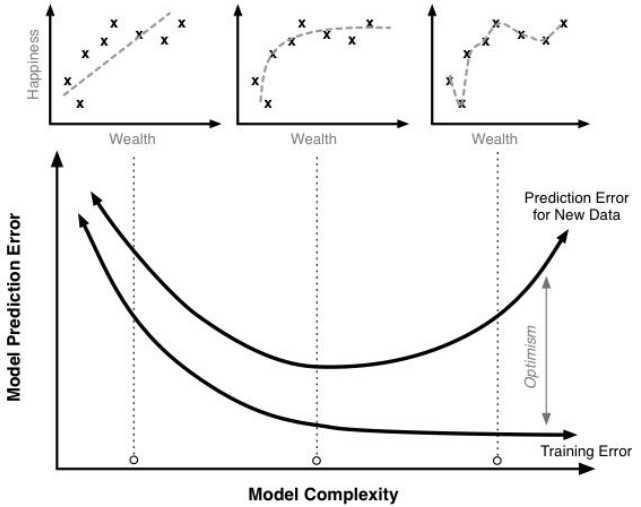
In high dimensions, a random test point tends to be **equally far** from most training examples.

Deep Learning Theory

- Functions approximation: what classes of functions can be approximated?
- Optimization techniques (ie. n-th order, SGD, gradient free methods)
- Generalization







Appendix: Abandoning Weight Sharing

Benefit (mostly) from Hierarchy *not* Weight Sharing

The authors show that deep conv nets that do not share parameters are still able to achieve low validation losses on CIFAR-10. They premise that the biggest advantage comes from hierarchy, not from weight sharing of kernels.

Of course, in practice, memory considerations still make weight sharing a nice idea.