Fundamental Belief: Universal Approximation Theorems

Ju Sun
Computer Science & Engineering
University of Minnesota, Twin Cities

September 21, 2020


**Logistics**

- HW 0 posted *(due: midnight Sep 30)*

**HOMEWORK SET 0**

CSCI 5980/8980 Think Deep Learning (Fall 2020)

**Due** 11:59 pm, Sep 30 2020

**Instruction** Please typeset your homework in **LaTeX** and submit it as a single PDF file in Canvas. No late submission will be accepted. For each problem, your should acknowledge your collaborators if any. For problems containing multiple subproblems, there are often close logic connections between the subproblems. So always remember to build on previous ones, rather than work from scratch.

**Notation** We will use small letters (e.g., $u$) for scalars, small boldface letters (e.g., $a$) for vectors, and capital boldface letters (e.g., $A$) for matrices. $\mathbb{R}$ is the set of real numbers. $\mathbb{R}^n$ is the space of $n$-dimensional real vectors, and similarly $\mathbb{R}^{m \times n}$ is the space of $m \times n$ real matrices. The dotted equal sign $\overset{=}{\text{means defining}}$.

**Problem 1 (Chain rules, gradient and Hessian)** Recall from calculus that for a multivariate function $f(\mathbf{x})$ mapping from $\mathbb{R}^n$ to $\mathbb{R}$, i.e., $f : \mathbb{R}^n \mapsto \mathbb{R}$, the $i$-th partial derivative of $f$ is defined as $\frac{\partial f}{\partial x_i}$, i.e., the univariate derivative with respect to the $i$-th variable while holding the other variables constant. This generalizes naturally to the matrix case, where we consider $f(\mathbf{X})$ with $\mathbf{X} \in \mathbb{R}^{m \times n}$.

- CSCI8980 lecture scribing (2 scribes per session, 2 reviewers per session)

- Review of Scipy, Numpy, MSI resource, Colab + Project ideas (Sep 28 or Oct 05, TBD)
Typesetting and matrix calculus

– \LaTeX source of homework also posted in Canvas

Mind \LaTeX! Mind your math!

* Ten Signs a Claimed Mathematical Breakthrough is Wrong

Inspired by Sean Carroll’s closely-related Alternative-Science Respectability Checklist, without further ado I now offer the Ten Signs a Claimed Mathematical Breakthrough is Wrong.

1. The authors don’t use TeX. This simple test (suggested by Dave Bacon) already catches at least 60% of wrong mathematical breakthroughs. David Deutsch and Lov Grover are among the only known false positives.

* Paper Gestalt (50%/18%, 2009) \implies Deep Paper Gestalt (50%/0.4%, 2018)

– Matrix Cookbook? Yes and No

http://www2.imm.dtu.dk/pubdb/pubs/3274-full.html
Recap

Why should we trust NNs?

Visual proof of UAT

UAT in rigorous form

From shallow to deep NNs

Suggested reading
Recap I

biological neuron vs. artificial neuron

biological NN vs. artificial NN

Artificial NN: (over)-simplification on neuron & connection levels
Recap II

Zoo of NN models in ML

- Linear regression
- Perception and Logistic regression
- Softmax regression
- Multilayer perceptron (feedforward NNs)

Also:
- Support vector machines (SVM)
- PCA (autoencoder)
- Matrix factorization
Brief history of NNs:

- 1943: first NNs invented (McCulloch and Pitts)
- 1958 –1969: perceptron (Rosenblatt)
- 1969: *Perceptrons* (Minsky and Papert)—end of perceptron
- 1980’s–1990’s: Neocognitron, CNN, back-prop, SGD—we use today
- 1990’s–2010’s: SVMs, Adaboosting, decision trees and random forests
- 2010’s–now: DNNs and deep learning
Outline

Recap

Why should we trust NNs?

Visual proof of UAT

UAT in rigorous form

From shallow to deep NNs

Suggested reading
Supervised learning

General view:
- Gather training data
  \((x_1, y_1), \ldots, (x_n, y_n)\)
- Choose a family of functions, e.g., \(\mathcal{H}\), so that there is \(f \in \mathcal{H}\) to ensure \(y_i \approx f(x_i)\) for all \(i\)
- Set up a loss function \(\ell\)
- Find an \(f \in \mathcal{H}\) to minimize the average loss
  \[
  \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell (y_i, f(x_i))
  \]

NN view:
- Gather training data
  \((x_1, y_1), \ldots, (x_n, y_n)\)
- Choose a NN with \(k\) neurons, so that there is a group of weights, e.g., \((w_1, \ldots, w_k, b_1, \ldots, b_k)\), to ensure \(y_i \approx \{\text{NN} (w_1, \ldots, w_k, b_1, \ldots, b_k)\} (x_i)\) for all \(i\)
- Set up a loss function \(\ell\)
- Find weights \((w_1, \ldots, w_k, b_1, \ldots, b_k)\) to minimize the average loss
  \[
  \min_{w's, b's} \frac{1}{n} \sum_{i=1}^{n} \ell [y_i, \{\text{NN} (w_1, \ldots, w_k, b_1, \ldots, b_k)\} (x_i)]
  \]

Why should we trust NNs?
Function approximation

More accurate description of supervised learning

- Underlying true function: $f_0$
- Training data: $y_i \approx f_0(x_i)$
- Choose a family of functions $\mathcal{H}$, so that $\exists f \in \mathcal{H}$ and $f$ and $f_0$ are close

- Approximation capacity: $\mathcal{H}$ matters (e.g., linear? quadratic? sinusoids? etc)

- Optimization & Generalization: how to find the best $f \in \mathcal{H}$ matters

We focus on approximation capacity now.
A word on notation

- $k$-layer NNs: with $k$ layers of weights (along the deepest path)
- $k$-hidden-layer NNs: with $k$ hidden layers of nodes (i.e., $(k + 1)$-layer NNs)
Think of single-output (i.e., $\mathbb{R}^n \mapsto \mathbb{R}$) problems first

A single neuron

- $\sigma$ identity or linear: linear functions
- $\sigma$ sign function $\text{sign} (\mathbf{w}^\top \mathbf{x} + b)$ (perceptron): 0/1 function with hyperplane threshold
- $\sigma = \frac{1}{1+e^{-z}}$: $\{ \mathbf{x} \mapsto \frac{1}{1+e^{-(\mathbf{w}^\top \mathbf{x}+b)}} \}$
- $\sigma = \max(0, z)$ (ReLU): $\{ \mathbf{x} \mapsto \max(0, \mathbf{w}^\top \mathbf{x} + b) \}$
Think of single-output (i.e., $\mathbb{R}^n \mapsto \mathbb{R}$) problems first

Add depth!

But make all hidden-nodes activations identity or linear

$$\sigma (w_L^T (W_{L-1} (\ldots (W_1 x + b_1) + \ldots) b_{L-1}) + b_L)$$

No better than a single neuron!

Why?
Think of single-output (i.e., $\mathbb{R}^n \mapsto \mathbb{R}$) problems first

Add both depth & nonlinearity!

Surprising news: 
universal approximation theorem

The 2-layer network can approximate arbitrary continuous functions arbitrarily well, provided that the hidden layer is sufficiently wide.

— so we don’t worry about limitation in the capacity
Outline

Recap

Why should we trust NNs?

Visual proof of UAT

UAT in rigorous form

From shallow to deep NNs

Suggested reading
Visual “proof”
(http://neuralnetworksanddeeplearning.com/chap4.html)

Think of \( \mathbb{R} \rightarrow \mathbb{R} \) functions first, \( \sigma = \frac{1}{1+e^{-z}} \)

- Step 1: Build “step” functions
- Step 2: Build “bump” functions
- Step 3: Sum up bumps to approximate
Step 1: build step functions

\[ y = \frac{1}{1 + e^{-(wx+b)}} = \frac{1}{1 + e^{-w(x-b/w)}} \]

- Larger \( w \), sharper transition
- Transition around \(-b/w\), written as \( s\)
Step 2: build bump functions

$$0.6 \times \text{step}(0.3) - 0.6 \times \text{step}(0.6)$$

Write \( h \) as the bump height
Step 3: sum up bumps to approximate

two bumps

five bumps

ultimate idea ... familiar?

Message: all $\mathbb{R} \to \mathbb{R}$ functions can be “well” approximated using 2-layer NN’s
What about high-dimensional?

Similar story

– Step 1: Build “step” functions
– Step 2: Build “bump” functions
– Step 3: Build “tower” functions
– Step 4: Sum up bumps to approximate

Steps 1 & 2: build step and bump functions

- **Step in** $x$ by setting large weight for $x$
- **Bump in** $x$ by diff of two steps in $x$
Step 3: build tower functions

sum up $x$, $y$ bumps to obtain a stair tower

threshold to obtain a sharp tower
Step 4: sum up towers for approximation

**Message**: all $\mathbb{R}^2 \rightarrow \mathbb{R}$ functions can be “well” approximated using 3-layer NN’s

**Question**: Possible using 2-layer NNs only?
General cases?

– **What about** $\mathbb{R}^n \mapsto \mathbb{R}$ **functions?**
  
  The “step $\mapsto$ (bump) $\mapsto$ tower $\mapsto$ tower array” construction carries over

– **What about** $\mathbb{R}^n \mapsto \mathbb{R}^m$ **functions?**
  
  Approximate each $\mathbb{R}^n \mapsto \mathbb{R}$ separately and then glue them together

**Message:** All $\mathbb{R}^n \mapsto \mathbb{R}^m$ functions can be “well” approximated using 2-layer NN’s
Recap

Why should we trust NNs?

Visual proof of UAT

UAT in rigorous form

From shallow to deep NNs

Suggested reading
Theorem (UAT, [Cybenko, 1989, Hornik, 1991])

Let \( \sigma : \mathbb{R} \to \mathbb{R} \) be a nonconstant, bounded, and continuous function. Let \( I_m \) denote the \( m \)-dimensional unit hypercube \([0, 1]^m\). The space of real-valued continuous functions on \( I_m \) is denoted by \( C(I_m) \). Then, given any \( \varepsilon > 0 \) and any function \( f \in C(I_m) \), there exist an integer \( N \), real constants \( v_i, b_i \in \mathbb{R} \) and real vectors \( w_i \in \mathbb{R}^m \) for \( i = 1, \ldots, N \), such that we may define:

\[
F(x) = \sum_{i=1}^{N} v_i \sigma \left( w_i^T x + b_i \right) = v^T \sigma (W^T x + b)
\]

as an approximate realization of the function \( f \); that is,\[
|F(x) - f(x)| < \varepsilon
\]
for all \( x \in I_m \).
Rigorous proof?

The proof is very technical ... functional analysis

1. **Riesz Representation**: Every linear functional on $C^0([0,1]^k)$ is given by

$$ f \mapsto \int_{[0,1]^k} f(x) d\mu(x), \quad \mu \in \mathcal{M} $$

where $\mathcal{M} = \{\text{finite signed regular Borel measures on } [0,1]^k\}.$

2. **Lemma**: Suppose for each $\mu \in \mathcal{M}$, we have

$$ \int_{[0,1]^k} \phi(w \cdot x + b) d\mu(x) = 0 \quad \forall w, b \quad \Rightarrow \quad \mu = 0. \quad (0.1) $$

Then $\text{Nets}_1(\phi)$ is dense in $C^0([0,1]^k)$.

3. **Lemma**: $\phi$ continuous, sigmoidal $\Rightarrow$ satisfies (0.1).
Thoughts on UAT

- $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be a nonconstant, bounded, and continuous:
  what about ReLU (leaky ReLU) or sign function (as in perceptron)? We have many UAT theorem(s)

- $I_m$ denote the m-dimensional unit hypercube $[0, 1]^m$: this can replaced by any compact subset of $\mathbb{R}^m$

- there exist an integer $N$: but how large $N$ needs to be? (later)

- The space of real-valued continuous functions on $I_m$:
  two examples to ponder on
  - binary classification
  - learn to solve square root
Learn to take square-root

Suppose we lived in a time square-root is not defined ...

- Training data: \( \{x_i, x_i^2\}_i \), where \( x_i \in \mathbb{R} \)
- Forward: if \( x \mapsto y, -x \mapsto y \) also
- To invert, what to output?
  What if just throw in the training data?
Thoughts

- Approximate continuous functions with vector outputs, i.e., $I_m \rightarrow \mathbb{R}^n$? think of the component functions

- Map to $[0, 1]$, $\{-1, +1\}$, $[0, \infty)$? choose appropriate activation $\sigma$ at the output

$$F(x) = \sigma \left( \sum_{i=1}^{N} v_i \sigma (w_i^T x + b_i) \right)$$

... universality holds in modified form

- Get deeper? three-layer NN? change to matrix-vector notation for convenience

$$F(x) = w^T \sigma(W_2 \sigma(W_1 x + b_1) + b_2) \quad \text{as} \quad \sum_{k} w_k g_k(x)$$

use $w_k$’s to linearly combine the same function

- For geeks: approximate both $f$ and $f'$? check out [Hornik et al., 1990]
What about ReLU?

ReLU

difference of ReLU’s

what happens when the slopes of the ReLU’s are changed?

How general $\sigma$ can be? ... enough when $\sigma$ not a polynomial

[Leshno et al., 1993]
Outline

Recap

Why should we trust NNs?

Visual proof of UAT

UAT in rigorous form

From shallow to deep NNs

Suggested reading
What’s bad about shallow NNs?

From UAT, “... there exist an interger N, ...”, but how large?

What happens in 1D?

Assume the target $f$ is 1-Lipschitz, i.e., $|f(x) - f(y)| \leq |x - y|, \forall x, y \in \mathbb{R}$

For $\varepsilon$ accuracy, need $\frac{1}{\varepsilon}$ bumps
What’s bad about shallow NNs?

From UAT, “... there exist an integer $N$, ...”, but how large?

What happens in 2D? Visual proof in 2D first

$\sigma(w^T x + b)$, $\sigma$ sigmoid approach 2D step function when making $w$ large

Credit: CMU 11-785
Visual proof for 2D functions

Keep increasing the number of step functions that are distributed evenly...
What’s bad about shallow NNs?

From UAT, “... there exist an interger N, ...”, but how large?

What happens in $2D$?

Assume the target $f$ is 1-Lipschitz, i.e., $|f(x) - f(y)| \leq \|x - y\|_2$, $\forall x, y \in \mathbb{R}^2$

For $\varepsilon$ accuracy, need $O(\varepsilon^{-2})$ bumps. What about the $n$-D case? $O(\varepsilon^{-n})$. 
What’s good about deep NNs?

- Learn Boolean functions \((f : \{+1, -1\}^n \mapsto \{+1, -1\})\): DNNs can have \#nodes linear in \(n\), whereas 2-layer NN needs exponential nodes (more in HW1)

- What general functions set deep and shallow NNs apart?

A family: compositional function [Poggio et al., 2017]
Compositional functions

\[ f(x_1, \cdots, x_8) = h_3(h_{21}(h_{11}(x_1, x_2), h_{12}(x_3, x_4)), \]
\[ h_{22}(h_{13}(x_5, x_6), h_{14}(x_7, x_8)) \]  \hspace{1cm} (4)

$W^n_m$: class of $n$-variable functions with partial derivatives up to $m$-th order, $W^{n,2}_m \subset W^n_m$ is the compositional subclass following binary tree structures

**Theorem 1.** Let $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ be infinitely differentiable, and not a polynomial. For $f \in W^n_m$ the complexity of shallow networks that provide accuracy at least $\epsilon$ is

\[ N = \mathcal{O}(\epsilon^{-n/m}) \text{ and is the best possible.} \]  \hspace{1cm} (5)

**Theorem 2.** For $f \in W^{n,2}_m$ consider a deep network with the same compositional architecture and with an activation function $\sigma : \mathbb{R} \rightarrow \mathbb{R}$ which is infinitely differentiable, and not a polynomial. The complexity of the network to provide approximation with accuracy at least $\epsilon$ is

\[ N = \mathcal{O}((n - 1)\epsilon^{-2/m}). \]  \hspace{1cm} (6)

from [Poggio et al., 2017] ; see Sec 4.2 of [Poggio et al., 2017] for lower bound
Nonsmooth activation

A terse version of UAT

**Proposition 2.** Let $\sigma =: \mathbb{R} \to \mathbb{R}$ be in $C^0$, and not a polynomial. Then shallow networks are dense in $C^0$.

Shallow vs. deep

**Theorem 4.** Let $f$ be a $L$-Lipschitz continuous function of $n$ variables. Then, the complexity of a network which is a linear combination of ReLU providing an approximation with accuracy at least $\epsilon$ is

$$N_s = \mathcal{O} \left( \left( \frac{\epsilon}{L} \right)^{-n} \right),$$

whereas that of a deep compositional architecture is

$$N_d = \mathcal{O} \left( (n - 1) \left( \frac{\epsilon}{L} \right)^{-2} \right).$$

from [Poggio et al., 2017]
Width-bounded DNNs

Narrower than $n + 4$ is fine

**Theorem 1** (Universal Approximation Theorem for Width-Bounded ReLU Networks). For any Lebesgue-integrable function $f: \mathbb{R}^n \to \mathbb{R}$ and any $\epsilon > 0$, there exists a fully-connected ReLU network $\mathcal{A}$ with width $d_m \leq n + 4$, such that the function $F_\mathcal{A}$ represented by this network satisfies

$$\int_{\mathbb{R}^n} |f(x) - F_\mathcal{A}(x)| dx < \epsilon.$$  \hfill (3)

But no narrower than $n - 1$

**Theorem 3.** For any continuous function $f: [-1, 1]^n \to \mathbb{R}$ which is not constant along any direction, there exists a universal $\epsilon^* > 0$ such that for any function $F_A$ represented by a fully-connected ReLU network with width $d_m \leq n - 1$, the $L^1$ distance between $f$ and $F_A$ is at least $\epsilon^*$:

$$\int_{[-1,1]^n} |f(x) - F_A(x)| dx \geq \epsilon^*.$$  \hfill (5)

from [Lu et al., 2017]; see also [Kidger and Lyons, 2019]

Deep vs. shallow still active area of research
Number one principle of DL

**Fundamental theorem of DNNs**

Universal approximation theorems

**Fundamental slogan of DL**

Where there is a function, there is a NN... and go ahead fitting it!
Outline

Recap

Why should we trust NNs?

Visual proof of UAT

UAT in rigorous form

From shallow to deep NNs

Suggested reading
Suggested reading

- Chap 4, Neural Networks and Deep Learning (online book)

- Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. (by Poggio et al)
  https://arxiv.org/abs/1611.00740

- Expressivity of Deep Neural Networks (by Ingo Gühring, Mones Raslan, Gitta Kutyniok)


