

UNIVERSAL APPROXIMATION THEOREMS – PRECISION CHALLENGES

Jan Pospíšil¹

partially joint work with Veronika Báčová¹ and Kamil Ekštein²

¹Department of Mathematics

²Department of Computer Science and Engineering

Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic

AI Tea Talks

April 14, 2023



DEPARTMENT OF
MATHEMATICS



Universal Approximation Theorems

- Motivation

- Light introduction - "UATs for Dummies"

- Arbitrary-width case

- Arbitrary-depth case

- Bounded depth and bounded width case

Challenge 1

- Approximation of a function of two variables

Challenge 2

- Black-Scholes-Merton implied volatility

Challenge 3

- Heston model calibration to option prices

Further reading


Question: Can we approximate **any** function using a neural network?

Example

Let us consider a very simple continuous non-linear function of one variable that is bounded on a compact domain, for example

$$f(x) = x^2, \quad x \in [-1, 1].$$

If we approximate the function by a single- or multi-layer feed-forward neural network (FNN), **what precision can be guaranteed theoretically and what precision can be achieved practically?** Is there a *gap*?

 ADCOCK, B. AND DEXTER, N. (2021), *The gap between theory and practice in function approximation with deep neural networks*. SIAM J. Math. Data Sci. 3(2), 624–655, ISSN 2577-0187, DOI [10.1137/20M131309X](https://doi.org/10.1137/20M131309X), Zbl [1483.65028](https://zbmath.org/journal/1483.65028), MR4253805

Notes on precision:

- ▶ **Machine epsilon** (sometimes also called **machine precision**) ε is *an upper bound on the relative approximation error due to rounding in floating point arithmetic*.
- ▶ In numerical analysis, machine epsilon *is dependent* on the type of rounding used and is also called **unit roundoff**, which has the symbol bold Roman **u**.
- ▶ Alternative definition (much more widespread, less formal): machine epsilon is *the difference between 1 and the next larger floating point number*, i.e. it *is independent* of rounding method and may be equivalent to **u** or $2\mathbf{u}$, i.e. ε equals the value of the *unit in the last place* relative to 1, i.e. $b^{-(p-1)}$, where b is the *base* and p is the *precision* and the unit roundoff is $\mathbf{u} = \varepsilon/2$ in *round-to-nearest* mode, and $\mathbf{u} = \varepsilon$ in *round-by-chop* (towards zero).
- ▶ In the IEEE 754 standard for floating-point arithmetic, the machine epsilon ε is:
 - ▶ $2^{-23} \approx 1.19 \times 10^{-7}$ in single precision (binary32),
 - ▶ $2^{-53} \approx 2.22 \times 10^{-16}$ in double precision (binary64).
- ▶ Definition $\varepsilon/2$ is used in LAPACK, Scilab, etc.
whereas ε is used in ISO C, C/C++, Python, Mathematica, MATLAB, etc.

Notes on speed: many commonly-available GPUs are optimized to perform single precision computations much more quickly, for example:

- ▶ NVIDIA Tesla P100 GPUs operate at 4.7 TFLOPS in double vs. 9.3 TFLOPS in single precision, implying a 1:2 ratio for single vs. double precision computation time.
- ▶ On the other hand, common off-the-shelf consumer GPUs such as the NVIDIA GeForce GTX 1080 Ti operate at 0.355 TFLOPS in double precision vs. 11.5 TFLOPS in single precision, implying a 1:32 ratio for single vs. double precision computation time.

Many papers use rather vague statements (not speaking about calling theorem a definition) that usually appear in different levels of detail (Hohman, Conlen, Heer, and Chau, 2020):

1. *Neural networks can approximate any function that exists. However, we do not have a guaranteed way to obtain such a neural network for every function.*

Many papers use rather vague statements (not speaking about calling theorem a definition) that usually appear in different levels of detail (Hohman, Conlen, Heer, and Chau, 2020):

2. For any function, there is guaranteed to be a neural network with a finite number (but perhaps a large number) of neurons so that for every possible input, x , the network outputs a close approximation of the function's value $f(x)$. However, we do not have any guarantees on how to train a neural network to learn the correct mapping parameters. There exists

Many papers use rather vague statements (not speaking about calling theorem a definition) that usually appear in different levels of detail (Hohman, Conlen, Heer, and Chau, 2020):

3. From mathematical theory of artificial neural networks, the universal approximation theorem states that a FNN with a single hidden layer containing a finite number (but perhaps a large number) of neurons can approximate continuous functions on compact subsets of \mathbf{R}^n , as long as the activation function is bounded and continuous. While this says that a simple neural network can represent a wide variety of interesting functions under appropriate parameters, it does not describe how to algorithmically learn such parameters.

Many papers use rather vague statements (not speaking about calling theorem a definition) that usually appear in different levels of detail (Hohman, Conlen, Heer, and Chau, 2020):

3. From mathematical theory of artificial neural networks, the universal approximation theorem states that a FNN with a single hidden layer containing a finite number (but perhaps a large number) of neurons can approximate continuous functions on compact subsets of \mathbf{R}^n , as long as the activation function is bounded and continuous. While this says that a simple neural network can represent a wide variety of interesting functions under appropriate parameters, it does not describe how to algorithmically learn such parameters.

There exist also "visual proofs", e.g. one by Michael Nielsen:
<http://neuralnetworksanddeeplearning.com/chap4.html>

Universal Approximation Theorems

Motivation

Light introduction - "UATs for Dummies"

Arbitrary-width case

Arbitrary-depth case

Bounded depth and bounded width case

Challenge 1

Approximation of a function of two variables

Challenge 2

Black-Scholes-Merton implied volatility

Challenge 3

Heston model calibration to option prices

Further reading

Theorem (Cybenko (1989))

Let $I_n = [0, 1]^n$ denote the n -dimensional unit cube and let $C(I_n)$ be the space of continuous functions on I_n . Let σ be any continuous sigmoidal function, i.e. $\sigma(t) \rightarrow \begin{cases} 1 & \text{as } t \rightarrow +\infty, \\ 0 & \text{as } t \rightarrow -\infty. \end{cases}$

Then finite sums of the form

$$G(x) = \sum_{j=1}^N \alpha_j \sigma(w_j^T x + \theta_j)$$

are dense in $C(I_n)$ with respect to the supremum norm. In other words, given any $f \in C(I_n)$ and $\varepsilon > 0$, there is a sum, $G(x)$, of the above form, for which

$$|G(x) - f(x)| < \varepsilon \quad \text{for all } x \in I_n.$$

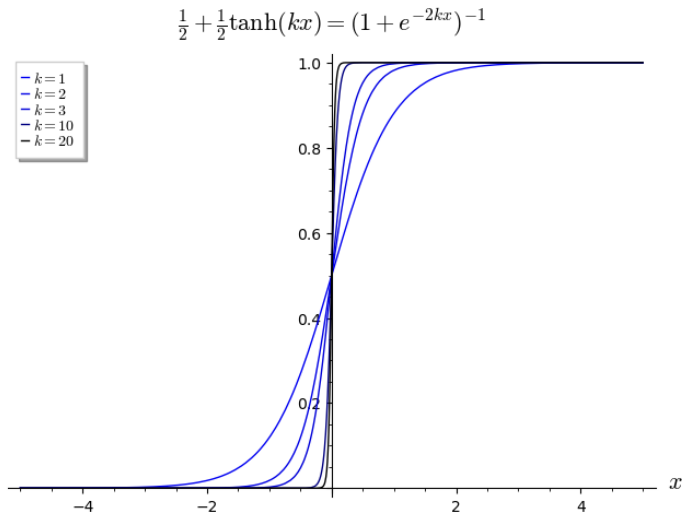
Proof: Idea in 1D ($n = 1$), different than in the paper.

Let $H(x) := \lim_{w \rightarrow +\infty} \sigma(w \cdot x) = \begin{cases} 1, & x > 0, \\ 0 & x < 0, \end{cases}$ (Heaviside step function).



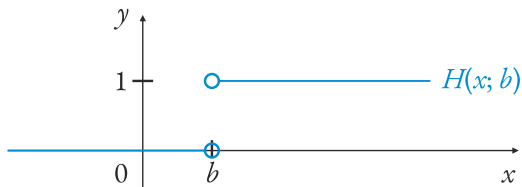
In *operational calculus*, useful answers seldom depend on which value is used for $H(0)$.



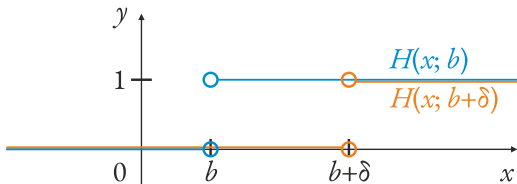
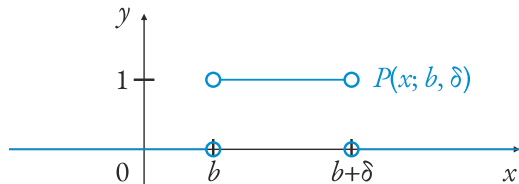


https://commons.wikimedia.org/wiki/File:Step_function_approximation.png

$$H(x; b) := \lim_{w \rightarrow +\infty} \sigma(w \cdot (x - b)) = \begin{cases} 1, & x > b, \\ 0 & x < b, \end{cases} \quad (\text{shifted Heaviside step function}).$$



$$P(x; b, \delta) := H(x; b) - H(x; b + \delta) \quad (\text{piece}).$$



Since f is continuous, it is also right-continuous, i.e. $\lim_{x \downarrow b} f(x) = f(b)$ for all $b \in I_n$, i.e.

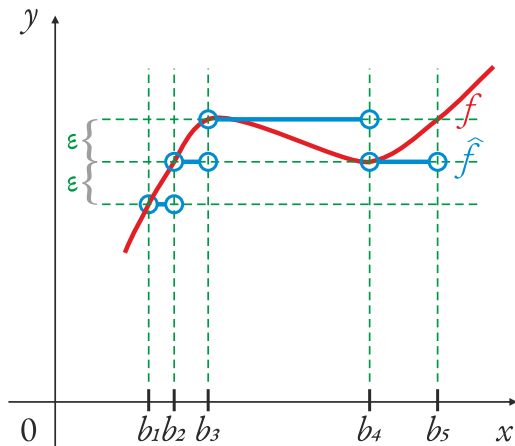
$$\forall \varepsilon > 0 \exists \delta > 0 \forall x \in I_n : x \in (b, b + \delta) \Rightarrow |f(x) - f(b)| < \varepsilon$$

$$|f(x) - f(b)P(x; b, \delta)| < \varepsilon.$$

We can repeatedly construct a piece-wise constant approximation of the function

$$\hat{f}(x) := \sum_{j=1}^N \alpha_j P(x, b_j, \delta_j),$$

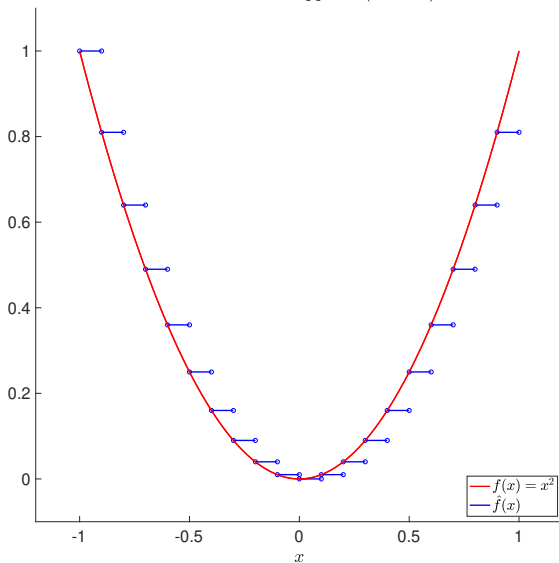
where $\alpha_j = f(b_j)$ and $b_{j+1} := b_j + \delta_j$, $j = 1, \dots, N$.



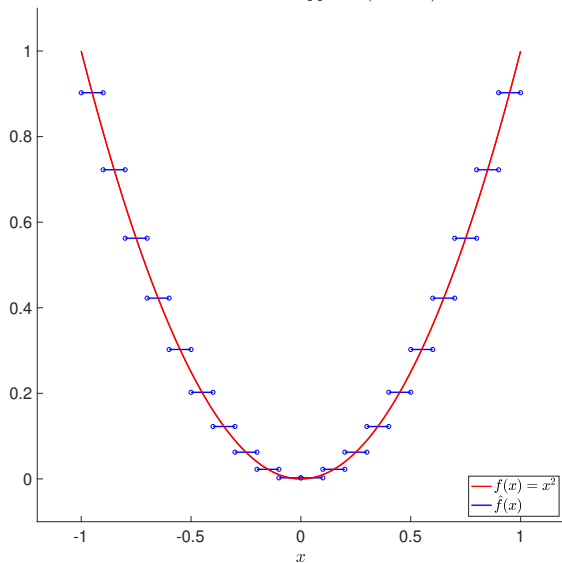
Since P is *constructed* by sigmoidal functions, this actually *proves* the theorem. □

Example: $f(x) = x^2$, $x \in [-1, 1]$ (and equidistant partitioning)

Piecewise const. approx. ($N = 20$)

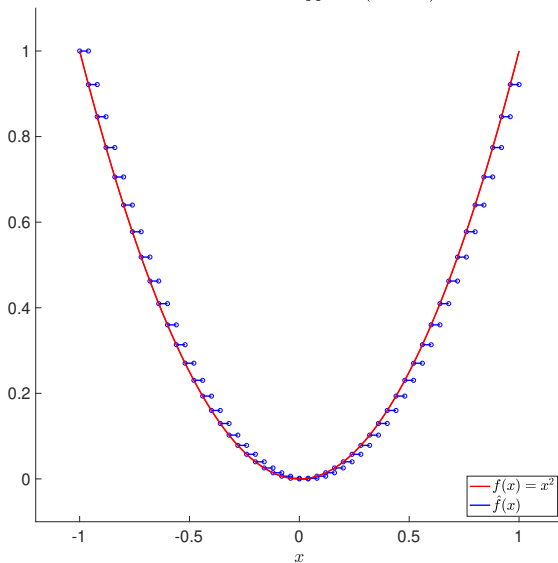


Piecewise const. approx. ($N = 20$)

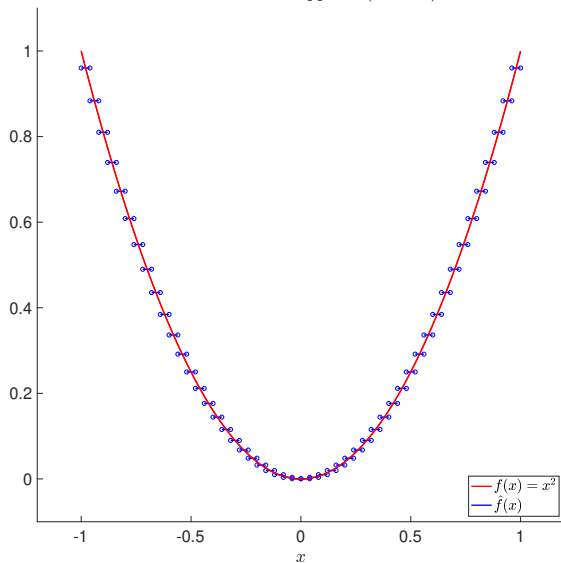


Example: $f(x) = x^2$, $x \in [-1, 1]$ (and equidistant partitioning)

Piecewise const. approx. ($N = 50$)

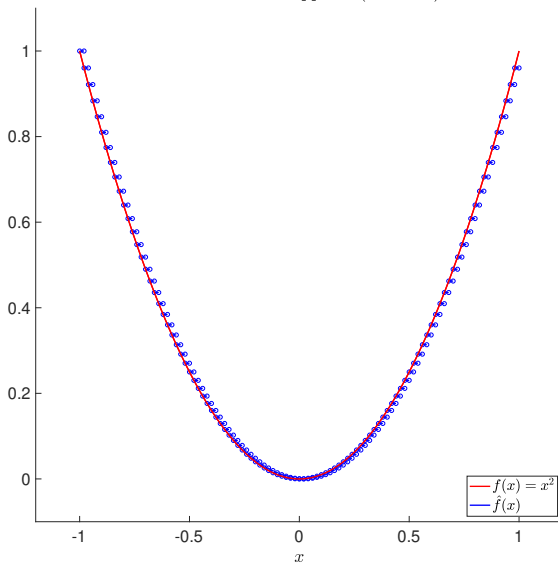


Piecewise const. approx. ($N = 50$)

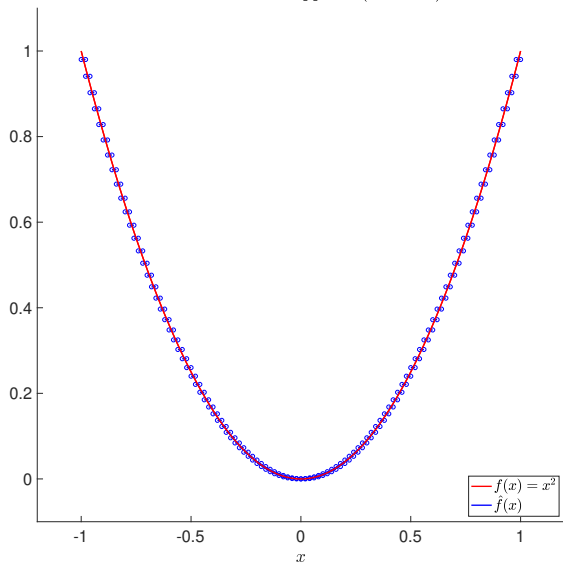


Example: $f(x) = x^2$, $x \in [-1, 1]$ (and equidistant partitioning)

Piecewise const. approx. ($N = 100$)

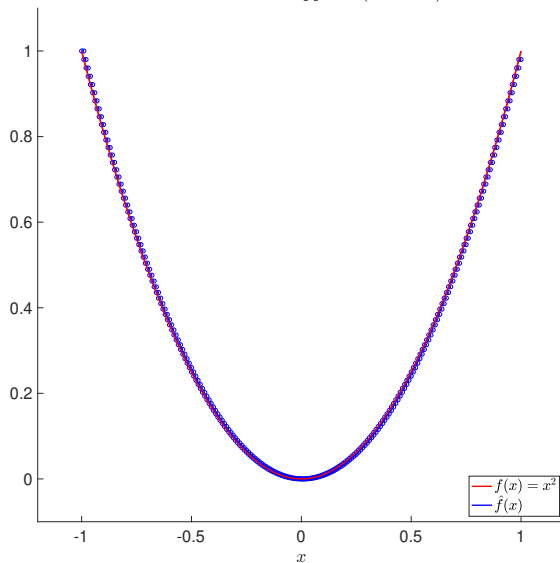


Piecewise const. approx. ($N = 100$)

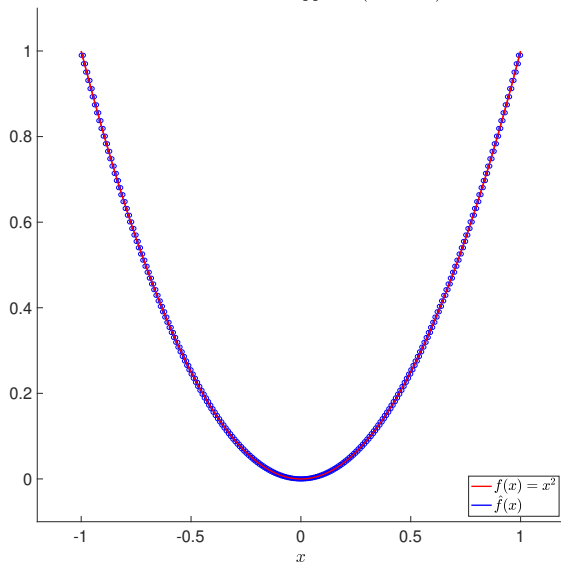


Example: $f(x) = x^2$, $x \in [-1, 1]$ (and equidistant partitioning)

Piecewise const. approx. ($N = 200$)



Piecewise const. approx. ($N = 200$)



Example: $f(x) = x^2$, $x \in [-1, 1]$ (and equidistant partitioning)

Question: Why would we ever need a FNN with more than one hidden layer?

Example: $f(x) = x^2$, $x \in [-1, 1]$ (and equidistant partitioning)

Question: Why would we ever need a FNN with more than one hidden layer?

- ▶ Simple calculation gives us

$$\max |f(x) - \hat{f}(x)| \sim O\left(\frac{1}{N}\right).$$

- ▶ Straightforward implication: in order to achieve a machine precision ε , we would need really a huge number of neurons N :

$$\frac{1}{N} \leq \varepsilon \quad \Rightarrow \quad N \geq \frac{1}{\varepsilon}$$

- ▶ In single precision (32-bit) arithmetic: $N \geq 8,403,361$,
in double precision (64-bit) arithmetic: $N \geq 4.5 \times 10^{15}$.

Some notes:

- ▶ From piece-wise constants we could proceed to **polynomials**. In particular the Stone-Weierstrass theorem states that **any continuous function defined on a compact set can be uniformly approximated by a polynomial function**. This means that for any continuous function $f(x)$ defined on a compact set K , and for any $\varepsilon > 0$, there exists a polynomial $P(x)$ such that

$$|f(x) - P(x)| < \varepsilon \quad \text{for all } x \in K.$$

We can construct a FNN with a single-hidden layer to approximate monomial function of the form x^k , $k \in \mathbf{N}$ and consequently to approximate any polynomial $P(x)$.

- ▶ Note that approximation of a function by the Taylor's polynomial has a very strong assumption (f must be many times differentiable), but this can lead to a significant reduction of N , needed number of neurons in the hidden layer.
- ▶ **Funahashi (1989)** worked on the same problem as **Cybenko (1989)** independently.
- ▶ Two-layered FNN (i.e. the one that does not have any hidden layers) is not capable of approximating general nonlinear continuous functions (**Widrow and Lehr, 1990**).

Universal Approximation Theorems

Motivation

Light introduction - "UATs for Dummies"

Arbitrary-width case

Arbitrary-depth case

Bounded depth and bounded width case

Challenge 1

Approximation of a function of two variables

Challenge 2

Black-Scholes-Merton implied volatility

Challenge 3

Heston model calibration to option prices

Further reading

Some notes:


- ▶ Multilayer FNNs are universal approximators (Hornik, Stinchcombe, and White, 1989): *standard multilayer FNNs with as few as one hidden layer using arbitrary squashing functions are capable of approximating any Borel measurable function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available*
- ▶ A standard multilayer FNN can approximate any continuous function to any degree of accuracy **if and only if** the network's activation functions are not polynomial (Leshno, Lin, Pinkus, and Schocken, 1993).
- ▶ Review of results from 60s-80s by Widrow and Lehr (1990), from 90s by Pinkus (1999); Scarselli and Chung Tsoi (1998).

📄 MHASKAR, H. N. AND POGGIO, T. (2016), *Deep vs. shallow networks: an approximation theory perspective*. Anal. Appl., Singap. 14(6), 829–848, ISSN 0219-5305, DOI [10.1142/S0219530516400042](https://doi.org/10.1142/S0219530516400042), Zbl [1355.68233](https://zbmath.org/?q=ser/1355.68233), MR[3564936](https://www.ams.org/mathscinet-getitem?mr=MR3564936)

Abstract: The paper briefly reviews several recent results on hierarchical architectures for learning from examples, that may formally explain the conditions under which Deep Convolutional Neural Networks perform much better in function approximation problems than shallow, one-hidden layer architectures. The paper announces new results for a non-smooth activation function — the ReLU function — used in present-day neural networks, as well as for the Gaussian networks. We propose a new definition of relative dimension to encapsulate different notions of sparsity of a function class that can possibly be exploited by deep networks but not by shallow ones to drastically reduce the complexity required for approximation and learning.

📄 GRIPENBERG, G. (2003), *Approximation by neural networks with a bounded number of nodes at each level*. J. Approx. Theory 122(2), 260–266, ISSN 0021-9045, DOI [10.1016/S0021-9045\(03\)00078-9](https://doi.org/10.1016/S0021-9045(03)00078-9), Zbl [1019.41018](https://zbmath.org/?q=1019.41018), MR1988304

Abstract: It is shown that the general approximation property of feed-forward multilayer perceptron networks can be achieved in networks where the number of nodes in each layer is bounded, but the number of layers grows to infinity. This is the case provided the activation functions are twice continuously differentiable and not linear.

 YAROTSKY, D. (2017), *Error bounds for approximations with deep ReLU networks*. Neural Networks 94, 103–114, ISSN 0893-6080, DOI [10.1016/j.neunet.2017.07.002](https://doi.org/10.1016/j.neunet.2017.07.002)

Abstract: We study expressive power of shallow and deep neural networks with piece-wise linear activation functions. We establish new rigorous upper and lower bounds for the network complexity in the setting of approximations in Sobolev spaces. In particular, we prove that deep ReLU networks more efficiently approximate smooth functions than shallow networks. In the case of approximations of 1D Lipschitz functions we describe adaptive depth-6 network architectures more efficient than the standard shallow architecture.

Universal Approximation Theorems

Motivation

Light introduction - "UATs for Dummies"

Arbitrary-width case

Arbitrary-depth case

Bounded depth and bounded width case

Challenge 1

Approximation of a function of two variables

Challenge 2

Black-Scholes-Merton implied volatility

Challenge 3

Heston model calibration to option prices

Further reading

TODO: we will probably need a separate seminar for this case...

Universal Approximation Theorems

Motivation

Light introduction - "UATs for Dummies"

Arbitrary-width case

Arbitrary-depth case

Bounded depth and bounded width case

Challenge 1

Approximation of a function of two variables

Challenge 2

Black-Scholes-Merton implied volatility

Challenge 3

Heston model calibration to option prices

Further reading

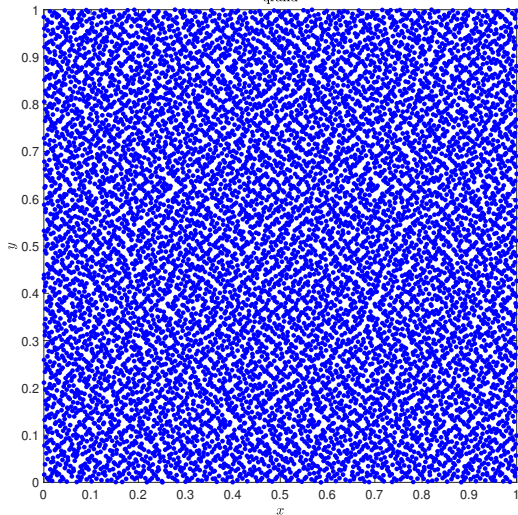
Challenge 1: Approximate a given real function of two variables using ANN of your choice in order to achieve the best possible precision.

- ▶ Analytical formula is NOT provided (although it is probably not difficult to guess it - please do not share this particular finding among other participants),
- ▶ Function $f(x, y)$ is non-linear (it is neither a polynomial), for $(x, y) \in D = [0, 1]^2$ it is bounded and continuous.
- ▶ For training purposes, there are 1.2×10^6 sample points in `challenge1_trainset.csv` (ca 61 MB), it is a CSV file with 1.2×10^6 rows and three columns $x, y, f(x, y)$. Points $[x, y] \in D$ were generated using the 2D Sobol's low discrepancy sequence (see also `challenge1_generator.m`). All numbers are provided in double precision.
- ▶ I leave it on you how many sample points you take for training your ANN.

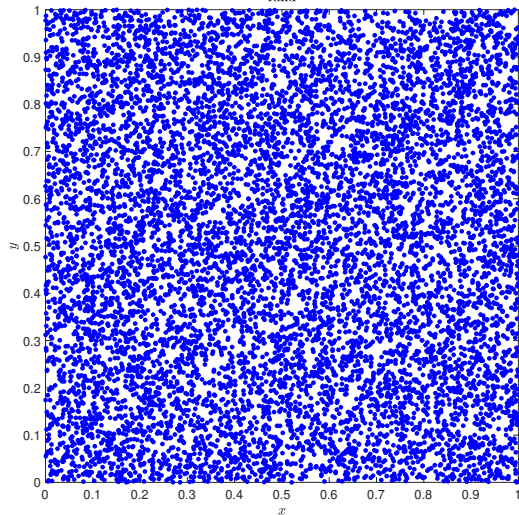
Questions: What precision can you achieve (in the maximal norm)?

Once your ANN is trained, how quickly can you calculate one million function values?

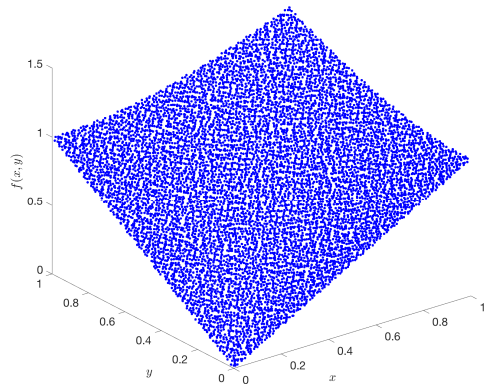
grand



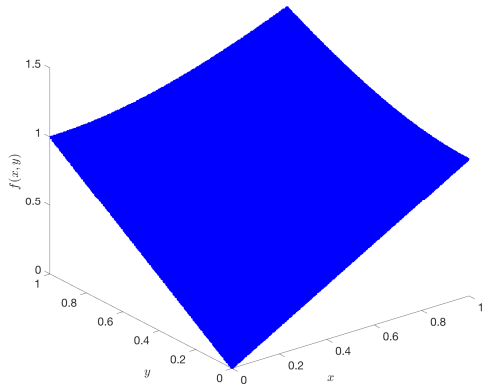
rand



$N = 10000$ sample points



$N = 100000$ sample points



Universal Approximation Theorems

Motivation

Light introduction - "UATs for Dummies"

Arbitrary-width case

Arbitrary-depth case

Bounded depth and bounded width case

Challenge 1

Approximation of a function of two variables

Challenge 2

Black-Scholes-Merton implied volatility

Challenge 3

Heston model calibration to option prices

Further reading

1969 Merton's portfolio problem (consumption vs. investment)

1971 Merton's model for pricing European options (equity = option in firm's asset)

1971 *Theory of rational option pricing*,

1973 ICAPM *International Capital Asset Pricing Model*

First one who uses continuous-time default probabilities to model options on the common stock of a company, i.e. he uses stochastic calculus in finance

Columbia University

California Institute of Technology

Massachusetts Institute of Technology

📄 MERTON, R. C. (1973), *Theory of rational option pricing*. Bell J. Econ. 4(1), 141–183, ISSN 0005-8556, DOI [10.2307/3003143](https://doi.org/10.2307/3003143), Zbl [1257.91043](https://zbmath.org/?q=sernum/1257.91043), MR0496534



1973 The pricing options and corporate liabilities,
Together with Fischer Sheffey Black (1938-1995), the famous
Black-Scholes formula, a fair price for a European call option (i.e. the
right to buy one share of a given stock at a specified price and time).

Stanford University



BLACK, F. AND SCHOLES, M. (1973), *The pricing of options and corporate liabilities*. J. Polit. Econ. 81(3), 637–654, ISSN 0022-3808, DOI [10.1086/260062](https://doi.org/10.1086/260062), Zbl [1092.91524](https://zbmath.org/?q=ser/1092.91524), MR3363443

1997 Nobel Prize in Economics:

for a new method to determine the value of derivatives



Robert C. Merton



Myron S. Scholes

In BSM model, underlying asset price is modelled as a **geometric Brownian motion (gBm)**:

$$dS(t) = rS(t) dt + \sigma S(t) dW(t),$$

where $r > 0$ is the *constant interest rate*, $\sigma > 0$ is the *volatility* and $W(t)$ is the *Wiener process*. Let $c(t, x)$ be the value of the European call option (with strike price K and maturity T) at time t if $S(t) = x$. The $c(t, x)$ satisfies the **Black-Scholes-Merton PDE**

$$c_t(t, x) + rx c_x(t, x) + \frac{1}{2} \sigma^2 x^2 c_{xx}(t, x) - rc(t, x) = 0, \quad \text{for all } t \in [0, T), x \geq 0$$

with terminal condition $c(T, x) = (x - K)^+$. It is a **backward parabolic PDE**. Boundary conditions must also be provided: $c(t, 0) = 0$ for all $t \in [0, T]$ and

$$\lim_{x \rightarrow +\infty} [c(t, x) - (x - e^{-r(T-t)}K)] = 0 \quad \text{for all } t \in [0, T].$$

This slide is NOT needed for the challenge.

The solution of the Black-Scholes-Merton PDE is

$$c(t, x) = xN(d_+(\tau, x)) - Ke^{-r\tau}N(d_-(\tau, x)), \quad 0 \leq t < T, x > 0,$$

where $\tau = T - t$ is **time to maturity**, N is the CDF for standard normal distribution

$$N(y) = \frac{1}{2}[1 + \operatorname{erf}(y/\sqrt{2})] = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^y e^{-z^2/2} dz = \frac{1}{\sqrt{2\pi}} \int_{-y}^{+\infty} e^{-z^2/2} dz$$

and

$$d_{\pm}(\tau, x) = \frac{1}{\sigma\sqrt{\tau}} \left[\ln \frac{x}{K} + \left(r \pm \frac{1}{2}\sigma^2 \right) \tau \right].$$

This formula IS needed for the challenge, it is implemented as the `myblsprice` (both in MATLAB and Python and in a more general form that considers also the *dividend rate* q).

Let $C^*(K, T)$ denotes the current market price of a call option with strike K and time to maturity T . **Implied volatility** $\sigma_{\text{imp}}(K, T)$ is such a value of the volatility parameter that when substituted into the BSM formula, it gives the market price $C^*(K, T)$:

$$C^*(K, T) = \text{BSM}(S, K, r, T, \sigma_{\text{imp}}(K, T)).$$

Calculation of the implied volatility can be converted into the root-finding problem (this is how `blsprice` works):

$$f(\sigma) = \text{BSM}(S, K, r, T, \sigma) - C^*(K, T).$$


Another possibility is to consider the optimization problem $\min_{\sigma} f^2(\sigma)$ (see `myblsimpv`).

Challenge 2: Use ANN to approximate the inverse Black-Scholes-Merton formula, i.e. train your ANN in order to calculate the implied volatilities as precisely as possible.

- ▶ For training purposes no dataset is prepared, use the `myblsprice` directly.
- ▶ For inspiration see also [Liu, Oosterlee, and Bohte \(2019\)](#).
- ▶ **Basis points** (BPS), or *bips*, are a unit of measure used in finance to describe the percentage change in the value of financial instruments or the rate change in an index or other benchmark. One basis point is equivalent to 0.01% (1/100th of a percent) or 0.0001 in decimal form.

Questions: What precision can you achieve?

Once your ANN is trained, how quickly can you calculate one million implied volatilities?

 LIU, S., OOSTERLEE, C., AND BOHTE, S. (2019), *Pricing options and computing implied volatilities using neural networks*. Risks 7(1), 16, ISSN 2227-9091, DOI [10.3390/risks7010016](https://doi.org/10.3390/risks7010016)

Universal Approximation Theorems

Motivation

Light introduction - "UATs for Dummies"

Arbitrary-width case

Arbitrary-depth case

Bounded depth and bounded width case

Challenge 1

Approximation of a function of two variables

Challenge 2

Black-Scholes-Merton implied volatility

Challenge 3


Heston model calibration to option prices

Further reading

Arguably the most popular mean-reverting **stochastic volatility model** is the Heston (1993) model

$$\begin{aligned}dS(t) &= rS(t) dt + \sqrt{v(t)}S(t) d\widetilde{W}^S(t), \\dv(t) &= -\kappa(v(t) - \theta) dt + \sigma\sqrt{v(t)} d\widetilde{W}^v(t), \\d\widetilde{W}^S(t) d\widetilde{W}^v(t) &= \rho dt,\end{aligned}$$

where θ represents a **long term variance**, κ is a **reversion rate** and σ denotes **volatility of volatility** parameter. Popularity of the model comes from its tractability and from the existence of a semi-closed solution for European option prices.

 HESTON, S. L. (1993), *A closed-form solution for options with stochastic volatility with applications to bond and currency options*. Rev. Financ. Stud. 6(2), 327–343, ISSN 0893-9454, DOI [10.1093/rfs/6.2.327](https://doi.org/10.1093/rfs/6.2.327), Zbl [1384.35131](https://zbmath.org/?q=ser/1384.35131), MR[3929676](https://www.ams.org/mathscinet/item?id=MR3929676)

$$C(S, v, \tau) = S - Ke^{-r\tau} \frac{1}{2\pi} \int_{-\infty+i/2}^{\infty+i/2} e^{-ikX} \frac{\hat{H}(k, v, \tau)}{k^2 - ik} dk,$$


where $X = \ln(S/K) + r\tau$ and $\hat{H}(k, v, \tau)$ is the so called **fundamental transform**:

$$\hat{H}(k, v, \tau) = \exp \left(\frac{2\kappa\theta}{\sigma^2} \left[qg - \ln \left(\frac{1 - he^{-\xi q}}{1 - h} \right) \right] + vg \left(\frac{1 - e^{-\xi q}}{1 - he^{-\xi q}} \right) \right), \text{ with}$$

$$g = \frac{b - \xi}{2}, \quad h = \frac{b - \xi}{b + \xi}, \quad q = \frac{\sigma^2 \tau}{2},$$

$$\xi = \sqrt{b^2 + \frac{4(k^2 - ik)}{\sigma^2}},$$

$$b = \frac{2}{\sigma^2} (ik\rho\sigma + \kappa).$$

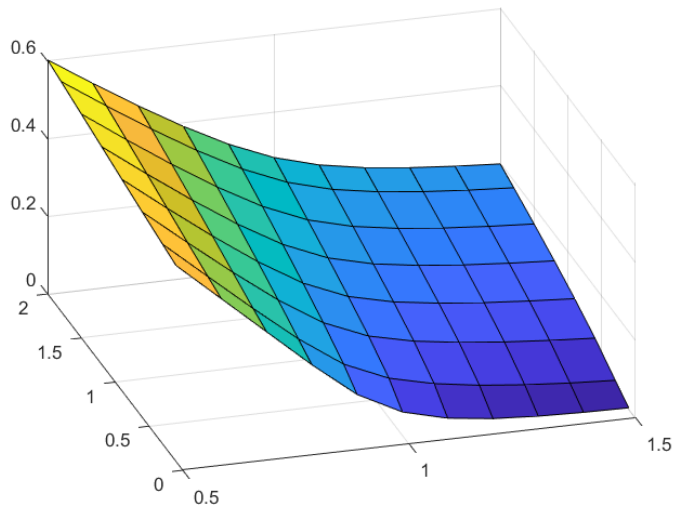
 LEWIS, A. L. (2000), Option Valuation Under Stochastic Volatility: With Mathematica code. Newport Beach, CA: Finance Press, ISBN 9780967637204, Zbl [0937.91060](#), MR1742310

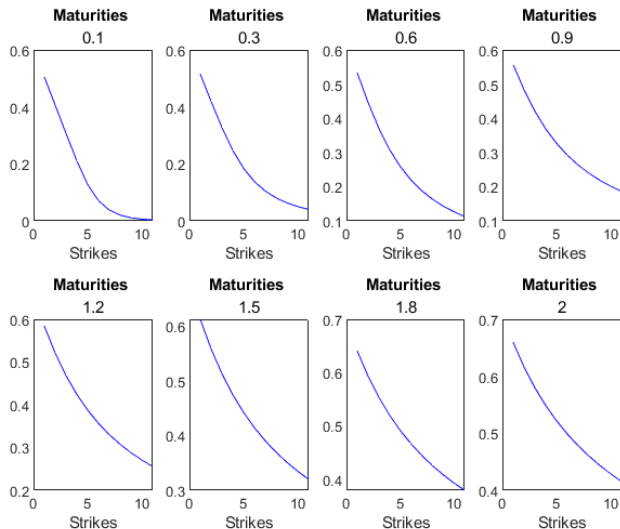
Challenge 3: Train ANN for calibration of Heston model to option prices

- ▶ For training purposes, there are ca 80 thousand sample surfaces in `trainset_prices_4.csv` (ca 136 MB), it is a CSV file with 80 052 rows and 94 columns. In the first six columns there are six model parameters $(r, v_0, \kappa, \theta, \sigma, \rho)$ and the remaining 88 values are the option value prices calculated at the rectangular lattice points – there are 8 different maturities τ with 11 strikes K each.
- ▶ Model parameters were generated using the 6D Sobol's low discrepancy sequence, `trainset` is pre-processed (cleaned, parameters and prices are already scaled, etc.).
- ▶ Model is highly sensitive to changes in the parameter values, especially in the **volatility of volatility** parameter σ , i.e. the desired error upper bound for each model parameter should be 10^{-6} (in order to get 1 bps precision for prices).

Questions: What precision can you achieve (say RMSE in the six parameter values)?

Once your ANN is trained, how quickly can you calculate one million function values?





- ▶ neural networks, approximation theory, and finite precision computation: [Wray and Green \(1995\)](#)
- ▶ universality in convolution networks: [Zhou \(2020\)](#)
- ▶ universal function approximation on graphs: [Brüel-Gabrielsson \(2020\)](#)
- ▶ Choose Your Weapon: Survival Strategies for Depressed AI Academics: [Togelius and Yannakakis \(2023\)](#)

Summary of today's talk:

Universal Approximation Theorems

- Motivation

- Light introduction - "UATs for Dummies"

- Arbitrary-width case

- Arbitrary-depth case

- Bounded depth and bounded width case

Challenge 1

- Approximation of a function of two variables

Challenge 2

- Black-Scholes-Merton implied volatility

Challenge 3

- Heston model calibration to option prices

- Further reading

- ADCOCK, B. AND DEXTER, N. (2021), *The gap between theory and practice in function approximation with deep neural networks*. SIAM J. Math. Data Sci. 3(2), 624–655, ISSN 2577-0187, DOI [10.1137/20M131309X](https://doi.org/10.1137/20M131309X), Zbl [1483.65028](https://zbmath.org/journals/SIAM/JMSDS/2021/3/2/624.html), MR[4253805](https://mathscinet.org/mathscinet/?q=MR4253805).
- BLACK, F. AND SCHOLES, M. (1973), *The pricing of options and corporate liabilities*. J. Polit. Econ. 81(3), 637–654, ISSN 0022-3808, DOI [10.1086/260062](https://doi.org/10.1086/260062), Zbl [1092.91524](https://zbmath.org/journals/JPE/1973/81/3/637.html), MR[3363443](https://mathscinet.org/mathscinet/?q=MR3363443).
- BRÜEL-GABRIELSSON, R. (2020), *Universal function approximation on graphs*. In H. LAROCHELLE, M. RANZATO, R. HADSELL, M. BALCAN, AND H. LIN, eds., *Advances in Neural Information Processing Systems*, vol. 33, pp. 19762–19772, Curran Associates, Inc.
- CYBENKO, G. (1989), *Approximation by superpositions of a sigmoidal function*. Math. Control Signals Syst. 2(4), 303–314, ISSN 0932-4194, DOI [10.1007/BF02551274](https://doi.org/10.1007/BF02551274), Zbl [0679.94019](https://zbmath.org/journals/MCSYS/1989/2/4/303.html), MR[1015670](https://mathscinet.org/mathscinet/?q=MR1015670).
- FUNAHASHI, K.-I. (1989), *On the approximate realization of continuous mappings by neural networks*. Neural Networks 2(3), 183–192, ISSN 0893-6080, DOI [10.1016/0893-6080\(89\)90003-8](https://doi.org/10.1016/0893-6080(89)90003-8).
- GRIPENBERG, G. (2003), *Approximation by neural networks with a bounded number of nodes at each level*. J. Approx. Theory 122(2), 260–266, ISSN 0021-9045, DOI [10.1016/S0021-9045\(03\)00078-9](https://doi.org/10.1016/S0021-9045(03)00078-9), Zbl [1019.41018](https://zbmath.org/journals/JAPROX/2003/122/2/260.html), MR[1988304](https://mathscinet.org/mathscinet/?q=MR1988304).
- HESTON, S. L. (1993), *A closed-form solution for options with stochastic volatility with applications to bond and currency options*. Rev. Financ. Stud. 6(2), 327–343, ISSN 0893-9454, DOI [10.1093/rfs/6.2.327](https://doi.org/10.1093/rfs/6.2.327), Zbl [1384.35131](https://zbmath.org/journals/RFSTUD/1993/6/2/327.html), MR[3929676](https://mathscinet.org/mathscinet/?q=MR3929676).
- HOHMAN, F., CONLEN, M., HEER, J., AND CHAU, D. H. P. (2020), *Communicating with interactive articles*. Distill DOI [10.23915/distill.00028](https://doi.org/10.23915/distill.00028).
- HORNIK, K., STINCHCOMBE, M., AND WHITE, H. (1989), *Multilayer feedforward networks are universal approximators*. Neural Networks 2(5), 359–366, ISSN 0893-6080, DOI [10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).

- LESHNO, M., LIN, V. Y., PINKUS, A., AND SCHOCKEN, S. (1993), *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*. Neural Networks 6(6), 861–867, ISSN 0893-6080, DOI [10.1016/S0893-6080\(05\)80131-5](https://doi.org/10.1016/S0893-6080(05)80131-5).
- LEWIS, A. L. (2000), *Option Valuation Under Stochastic Volatility: With Mathematica code*. Newport Beach, CA: Finance Press, ISBN 9780967637204, Zbl [0937.91060](https://zbmath.org/?q=serif/0937.91060), MR1742310.
- LIU, S., OOSTERLEE, C., AND BOHTE, S. (2019), *Pricing options and computing implied volatilities using neural networks*. Risks 7(1), 16, ISSN 2227-9091, DOI [10.3390/risks7010016](https://doi.org/10.3390/risks7010016).
- MERTON, R. C. (1973), *Theory of rational option pricing*. Bell J. Econ. 4(1), 141–183, ISSN 0005-8556, DOI [10.2307/3003143](https://doi.org/10.2307/3003143), Zbl [1257.91043](https://zbmath.org/?q=serif/1257.91043), MR0496534.
- MHASKAR, H. N. AND POGGIO, T. (2016), *Deep vs. shallow networks: an approximation theory perspective*. Anal. Appl., Singap. 14(6), 829–848, ISSN 0219-5305, DOI [10.1142/S0219530516400042](https://doi.org/10.1142/S0219530516400042), Zbl [1355.68233](https://zbmath.org/?q=serif/1355.68233), MR3564936.
- PINKUS, A. (1999), *Approximation theory of the MLP model in neural networks*. In Acta numerica, 1999, vol. 8 of Acta Numer., pp. 143–195, Cambridge: Cambridge Univ. Press, ISBN 0-521-77088-2, DOI [10.1017/S0962492900002919](https://doi.org/10.1017/S0962492900002919), Zbl [0959.68109](https://zbmath.org/?q=serif/0959.68109), MR1819645.
- SCARSELLI, F. AND CHUNG TSOI, A. (1998), *Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results*. Neural Networks 11(1), 15–37, ISSN 0893-6080, DOI [10.1016/S0893-6080\(97\)00097-X](https://doi.org/10.1016/S0893-6080(97)00097-X).
- TOGELIUS, J. AND YANNAKAKIS, G. N. Y. (2023), *Choose your weapon: Survival strategies for depressed AI academics*, available at arXiv: <https://arxiv.org/abs/2304.06035>.
- WIDROW, B. AND LEHR, M. A. (1990), *30 years of adaptive neural networks: perceptron, Madaline, and backpropagation*. Proceedings of the IEEE 78(9), 1415–1442, DOI [10.1109/5.58323](https://doi.org/10.1109/5.58323).

- WRAY, J. AND GREEN, G. G. (1995), *Neural networks, approximation theory, and finite precision computation*. Neural Networks 8(1), 31–37, DOI [10.1016/0893-6080\(94\)00056-R](https://doi.org/10.1016/0893-6080(94)00056-R).
- YAROTSKY, D. (2017), *Error bounds for approximations with deep ReLU networks*. Neural Networks 94, 103–114, ISSN 0893-6080, DOI [10.1016/j.neunet.2017.07.002](https://doi.org/10.1016/j.neunet.2017.07.002).
- ZHOU, D.-X. (2020), *Universality of deep convolutional neural networks*. Appl. Comput. Harmon. Anal. 48(2), 787–794, ISSN 1063-5203, DOI [10.1016/j.acha.2019.06.004](https://doi.org/10.1016/j.acha.2019.06.004), Zbl [1434.68531](https://zbmath.org/?q=ser/1434.68531), MR4047545.

Thank you for your attention!

Jan Pospíšil, honik@kma.zcu.cz