ON THE UNIVERSAL APPROXIMATION THEOREMS

SIDDHARTH SHASTRY

Abstract. This paper focuses on the mathematical underpinnings of neural networks, with significant attention given to the universal approximation theorem (UAT). We begin one of the most popular activation functions in neural network theory. First, the UAT for sigmoidal functions is described through the lens of George Cybenko's work. We then discuss the limitations of Cybenko's work with particular regard to the shortcomings of sigmoidal functions and the vanishing gradient problem. Next, we show that the UAT can be extended to the rectified linear unit function (ReLU). The ReLU function is preferred in modern day machine learning due to its ability to better mitigate the vanishing gradient problem. Finally, having focussed on single layer perceptron networks thus far, we introduce the multi-layer perceptron (MLP) network - a more robust neural network architecture - and extend the UAT to ReLU activated MLP networks.

Contents

1. Introduction

The neural network is a versatile mathematical model that can be used to solve a variety of tasks, ranging from discrete classification to more continuous regression. The neural network is built as a series of inputs that are fed through a linear transformation; the output of this transformation is then passed through an "activation function", which introduces an element of non-linearity that allows these networks to learn to more complex patterns. One of the most popular of these activation functions is the sigmoid class of functions.

Definition 1.1. A sigmoidal function is defined as follows:

$$\sigma(x) = \begin{cases} 1 & x \to \infty \\ 0 & x \to -\infty \end{cases}$$

George Cybenko shows that such sigmoidal functions are discriminatory.

Date: September 18, 2024.

Definition 1.2. For all $n \in \mathbb{N}$ we define A^r as the set of all affine transformations from $\mathbb{R}^n \to \mathbb{R}$

$$A(x) = w \cdot x + b$$

Definition 1.3. Let I_n be an n dimensional cube, $[0,1]^n$. $M(I_n)$ is the space of finite and signed Borel measures on I_n. A function f is known as discriminatory if it satisfies the condition below for any measure $\mu \in M(I_n)$:

$$\int_{I_n} f(A(x))d\mu = 0$$

implies that for all $y \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$, μ is the zero measure.

Cybenko also shows that finite sums of the form:

$$\sum_{i=1}^{l} \alpha_i \sigma(w \cdot x + b_i) = \sum_{i=1}^{l} \alpha_i \sigma(A_i(x))$$

where σ is a sigmoidal function (or any discriminatory function) such as the logistic function, $\sigma(x) = \frac{e^x}{1+e^x}$, are dense in the space of continuous functions over I_n, denoted as $C(I_n)$; $x \in \mathbb{R}^n$. In the context of a neural network, we can think of the vector $w \in \mathbb{R}^n$ as the weights between the input layer and the single hidden layer. $q \in R$ represents the number of nodes in the hidden layer. α represents the weights between the hidden layer and the output layer.

Theorem 1.4. *For any $\epsilon > 0$ and function $f \in C(I_n)$, there exists a finite sum $G(x)$ of the form above that satisfies $|G(x) - f(x)| < \epsilon$ for all $x \in I_n$.*

This finding is significant because it implies that a single layer, sigmoid-activated neural network can approximate any function in $C(I_n)$. However, the nature of sigmoidal functions – whose behavior it is to squish inputs with more extreme values into a far more constrained range – does not play well with the practical necessities of programmed neural networks. Sigmoidal activations often lead to the vanishing gradient problem, in which repeated multiplication of gradients smaller than one causes layer outputs to eventually "vanish". A more practical alternative to sigmoidal activations is the Rectified Linear Unit or ReLU class of functions. A goal of this paper is to expand the so-called Universal Approximation Theorem to more broader use-cases, starting with a neural network with ReLU activation.

To see the UAT for sigmoig-activated networks in action, we observe a selection of three function including a small polynomial ($f(x) = 3x^2 + 12x + 4$), a larger polynomial ($f(x) = 9x^4 + 123x^2 - 4x + 902$), and the exponential function ($f(x) = e^x$). The convergence of the network to the three functions is shown in figures 1, 2 and 3 respectively.

2. Expanding The UAT To ReLU

To understand why sigmoidal activations do not compliment more complex modifications of the neural network model very well, we must first understand the vanishing gradient problem, which in turn is dependent on the backpropagation algorithm in the gradient descent process.

Gradient descent is a popular algorithm that, by minimizing a loss function, can find more optimal values for a neural network's weights and biases (by optimal we

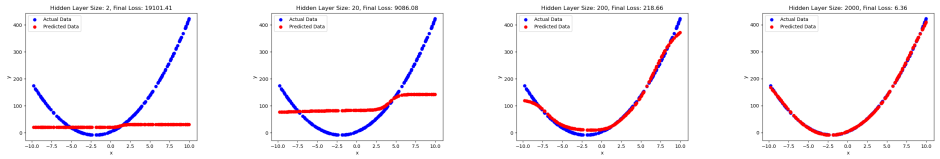

FIGURE 1. Single layer sigmoid activated neural network approximating a simple polynomial. Hidden layer sizes from left to right: 2, 20, 200, 2000. Red → Predicted; Blue → True.

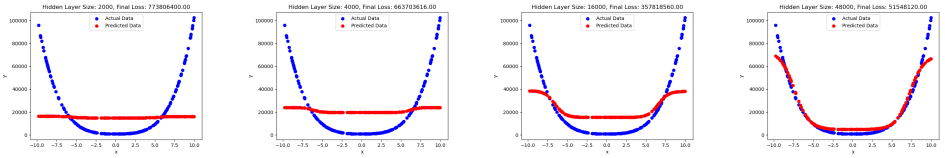

FIGURE 2. Single layer sigmoid activated neural network approximating a large polynomial. Hidden layer sizes from left to right: 2000, 4000, 16000, 48000. Red → Predicted; Blue → True.

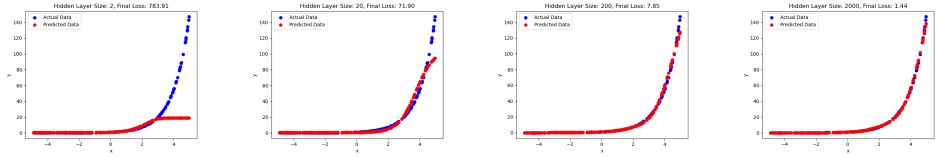

FIGURE 3. Single layer sigmoid activated neural network approximating an exponential. Hidden layer sizes from left to right: 2, 20, 200, 2000. Red → Predicted; Blue → True.

refer to the weights and biases that minimize the loss). Consider a function $E(\hat{y}, y)$ whose input is the output of a neural network G. E then computes a kind of loss metric: mean squared error is a popular choice for regression-related tasks and cross-entropy for classification tasks. We can then compute the partial derivatives of the loss function with respect to the individual weights and biases of the network. The gradients point us in the direction of the steepest increase in loss; the negation of the gradient points us towards the direction of steepest decrease in loss. We define gradient descent as an algorithm more rigorously below. The process of obtaining these partial derivatives is the backpropagation algorithm, which is heavily reliant on reverse-mode auto-differentiation.

Definition 2.1. Mean squared error (MSE) is defined as

$$E(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

where n is the output dimension, $\hat{y}$ is the output/predicted vector, and y is the true value.

Definition 2.2. Cross-entropy loss is defined as

$$E(\hat{y}, y) = \{e_1, e_2, ..., e_N\}^T$$

where

$$e_i = -\sum_{n=1}^{N} w_n log \frac{exp(\hat{y}_n)}{\sum_{c=i}^{N} exp(\hat{y}_c)} y_n$$

and N is the number of classes.

Definition 2.3. Gradient descent is an iterative algorithm that updates the values of weights and biases by a proportion of the partial of the loss function with respect to the corresponding weight/bias.

$$\Delta(w_{i,j})_{k+1} = (w_{i,j})_k - \tau \frac{\partial E}{\partial (w_{i,j})_k}$$

where i indexes the layer, j indexes the weight within that layer, and τ is the learning rate.

Reverse-mode auto-differentiation, and in turn, backpropagation makes extensive use of the chain rule of differentiation by computing partials with respect to the very last layer to begin with, and then using those results to compute partials with respect to the layer before it. Given this format of repeated multiplication, it becomes apparent why a function such as a sigmoidal function, which tends to squish values between a small range, can lead to the so called vanishing gradients.

One method (among many others) to mitigate the effects of the vanishing gradient problem is to use an activation function that can introduce the non-linearity necessary in a neural network without having to squish the output into a minuscule range. The ReLU class of functions achieves just this.

Definition 2.4. The ReLU function Φ is defined such that $\Phi(x) = max(0, x)$.

Theorem 2.5. *ReLU functions are discriminatory*

Proof. Consider two functions: $f_1(x) = \max(0, x+1)$ (modified ReLU) and $f_2(x) = \max(0, x)$ (ReLU). Now consider a bounded composition of the two,

$$g(x) = f_1(x) - f_2(x) = \begin{cases} 0 & x \leq -1 \\ x+1 & -1 < x < 0 \\ 1 & x \geq 0 \end{cases}$$

.

Observe that $g(x) \to 1$ as $x \to \infty$ and $g(x) \to 0$ as $x \to -\infty$; therefore, we say that g is sigmoidal. By definition, g is discriminatory which means that the zero measure is the only such measure μ satisfying

$$\int_{I_n} g(A(x)) d\mu = 0$$

$$\implies \int_{I_n} f_1(A(x)) d\mu - \int_{I_n} f_2(A(x)) d\mu = 0$$

$$\implies \int_{I_n} f_1(A(x)) d\mu = 0, \int_{I_n} f_2(A(x)) d\mu = 0 \iff \mu = 0$$

Thus f_1 & f_2 are discriminatory and the ReLU function is discriminatory. □

Corollary 2.6. *Sums of the form*

$$\sum_{i=1}^{N} \alpha_i \Phi(A_i(x))$$

are dense within the space of continuous functions over $C(I_n)$, where Φ is the ReLU function.

Proof. This follows from Theorem 2.2 and from Cybenko's results; ReLU is a discriminatory function and all such sums with a discriminatory activation are dense within the space of continuous functions over $C(I_n)$. □

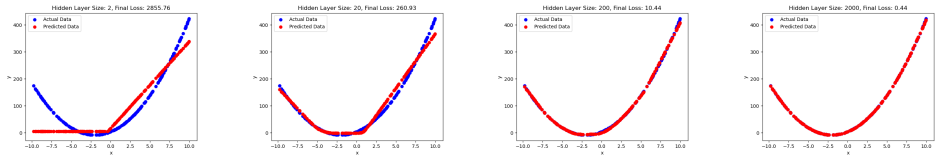

FIGURE 4. Single layer ReLU activated neural network approximating a simple polynomial. Hidden layer sizes from left to right: 2, 20, 200, 2000. Red → Predicted; Blue → True.

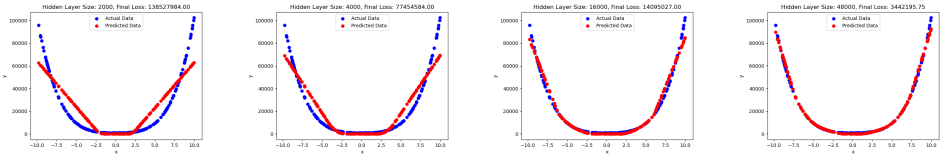

FIGURE 5. Single layer ReLU activated neural network approximating a large polynomial. Hidden layer sizes from left to right: 2000, 4000, 16000, 48000. Red → Predicted; Blue → True.

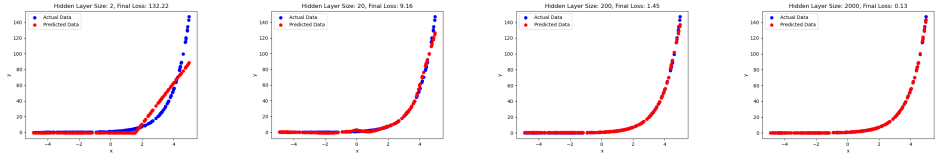

FIGURE 6. Single layer ReLU activated neural network approximating an exponential. Hidden layer sizes from left to right: 2, 20, 200, 2000. Red → Predicted; Blue → True.

3. Beyond the Single Hidden-Layer Network

Once again, we observe empirical evidence for the UAT for ReLU activated networks in figures 4 ($f(x) = 3x^2 + 12x + 4$), 5 ($f(x) = 9x^4 + 123x^2 - 4x + 902$), and 6 ($f(x) = e^x$). The exponential function provides evidence that the UAT does apply to non-polynomial functions as well. The larger polynomial shows that an increase in the rate of growth of a function tends to require an increase in the hidden layer's size.

Here we spot a potential issue: real world use cases often require more complex functions that span multiple dimensions. Simply scaling a single hidden layer is not the most efficient method by which we can learn these complex functions. It is here that more clever models, such as the multi-layer perceptron, convolutional neural network (for computer vision), and the recurrent neural network (to further aid in dealing with the vanishing gradient problem), become more desirable.

At the most fundamental level, although it is still possible to model more complex, and multidimensional, functions with a single hidden layer as is shown in "Multilayer feedforward networks are universal approximators" (Hornik et. all, 1988), it often takes fewer parameters to achieve similar results with a deeper, rather than a shallower and wider, network (Safran & Shamir, 2020). See figure 7 which models $f(x, y, z) = (x^2 + 4z, e^y + x, yx - z^2))$ for a demonstration of the UAT for multi-dimensional/vector valued functions. As such, the deep Multi-Layer Perceptron network (MLP) is a natural step forward from the architecture used in this paper thus far.

Definition 3.1. Let $\sum_l^{r,s}(\sigma)$ represent a MLP with an input dimension of r and an output dimension of s. Let l be the number of hidden layers plus one, to account for the output layer as well. σ is the activation function for this network. We define the iterative activations of nodes in the network as follows

$$(a_k)_i = G_k(A_i(a_{k-1}))$$

for $i = 1, ..., q$ and $k = 1, ..., l$. Here, l is the number of layers so a_k represents the activations of the nodes in a particular layer. More specifically, a_k is a q-element vector with elements $a_k)_i$

Theorem 3.2. *Let F be a class of functions: $\mathbb{R} \to \mathbb{R}$ and G be a class of functions: $\mathbb{R}^n \to \mathbb{R}$. Let F be uniformly dense on compact sets in C^1 and let G be uniformly dense on compact sets in C^n. We then define $\Phi := \{f \circ g : f \in F, g \in G\}$. We say that Φ is uniformly dense on compact sets in C^n.*

Proof. Let h be an arbitrary function in C^n. Let K be a compact subset of R^n and choose some $\epsilon > 0$. There exists a function $g \in G$ such that $\sup_{x \in K} |h(x) - g(x)| < \frac{\epsilon}{2}$, by definition. A continuous mapping f from a compact metric space X into a metric space Y implies that f(X) is compact (Rudin, 1953). Therefore, $\{h(x) : x \in K\}$ is compact. This implies that $\{g(x) : x \in K\}$ is bounded. Let S be the closure of $\{g(x) : x \in K\}$; S is also compact. We now observe that there exists some $f \in F$ such that $\sup_{s \in S} |f(s) - s| < \frac{\epsilon}{2}$.
Therefore

$$\begin{aligned}\sup_{x \in K} |f(g(x)) - h(x)| &= \sup_{x \in K} |f(g(x)) - g(x) + g(x) - h(x)| \\ &\leq \sup_{x \in K} |f(g(x)) - g(x)| + \sup_{x \in K} |g(x) - f(g(x))| < \epsilon\end{aligned}$$

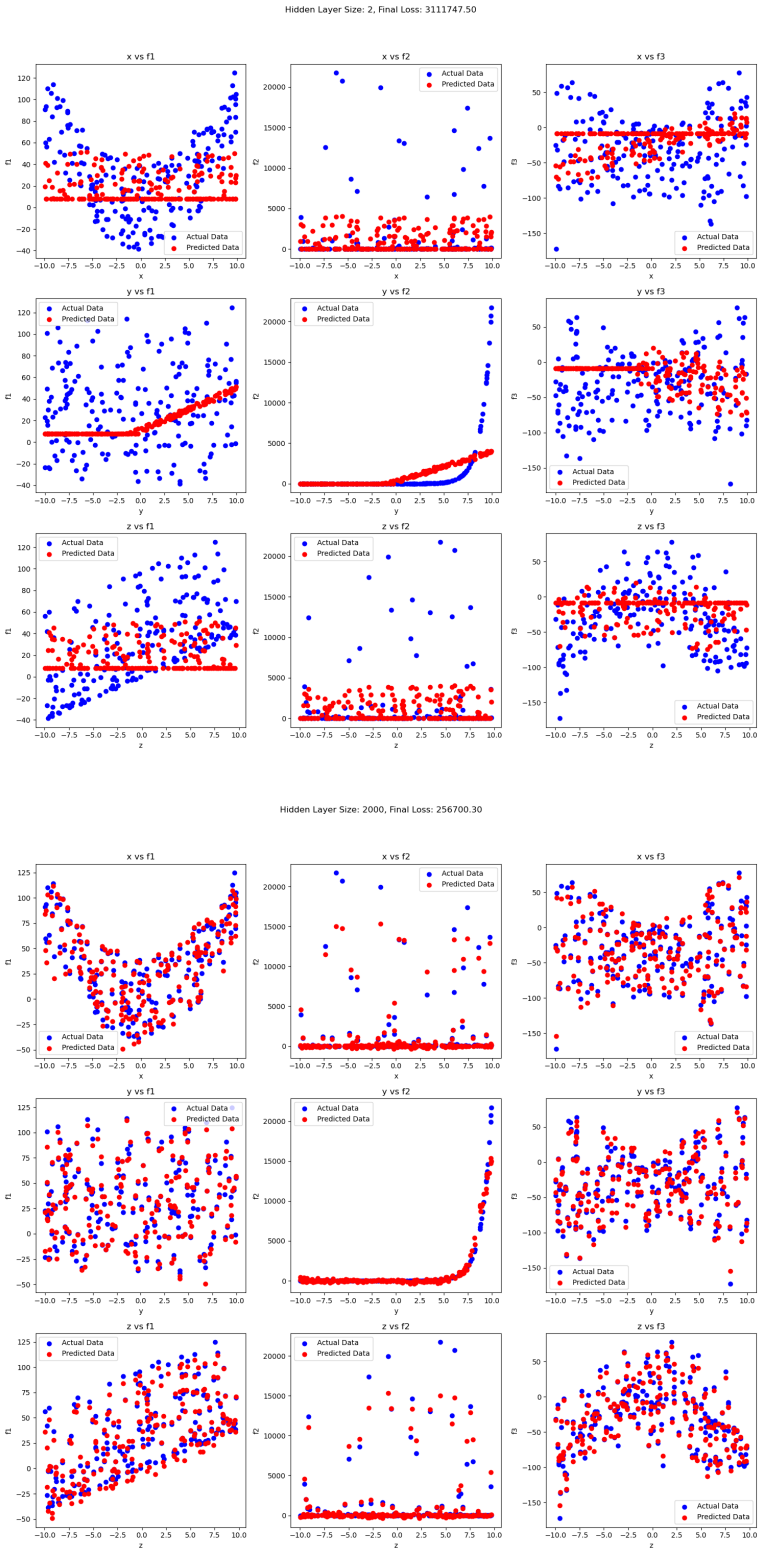

FIGURE 7. Single layer ReLU activated neural network approximating a multi-dimensional function

□

Theorem 3.3. *The UAT as described for single hidden-layer neural networks also applies to deep neural networks.*

Proof. For the sake of simplicity, let us assume the size of the output is 1 ($s = 1$). We can simply apply Theorem 3.2. To generalize this to larger output sizes ($s \geq 2$). We want to show that the class of functions of multi-layer neural networks is uniformly dense on compact sets in C^r.

$$J_k = \{\sum_{i_k} \beta_{i_k} \sigma(A_{i_k}(\sum_{i_{k-1}} \beta_{i_k i_{k-1}} \sigma(...(\sum_{i_1} \beta_{i_k ... i_1} \sigma(A_{i_k ... i_1}(x))...)))\}$$

We have already shown this to be true for $k = 1$. We can now perform an induction on k to show it holds for all. Suppose J_k is uniformly dense on compact sets in C^r. Then $J_{k+1} = \{\Sigma_j \beta_j \sigma(A_j(g_j(x))) : g_j \in J_k\}$. We know that the set $\{\Sigma_j \beta_j \sigma(A_j(\cdot))\}$ is uniformly dense on compact sets in C^1. Therefore, by Theorem 3.2, J_{k+1} must be compact on C^r. □

4. Acknowledgements

I would like to thank my mentor Elena Isasi Theus for her help in suggesting useful resources, validating proof ideas, reviewing my understanding of core subjects, and guiding me through the process of writing a paper. I would also like to thank Professor J. Peter May for organizing such an invaluable learning experience.

References

[1] George Cybenko. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems. 1989.
[2] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer Feedforward Networks are Universal Approximators. Neural Networks, Vol. 2. 1989.
[3] https://github.com/sud295/UniversalApproximationTheorem
[4] https://pytorch.org/docs/stable/index.html
[5] Walter Rudin. Principles of Mathematical Analysis. McGraw Hill. 1953.